The development of an electrochemical sensing device for controlled drugs.

WADDELL, S.A.

2019

THE DEVELOPMENT OF AN ELECTROCHEMICAL SENSING DEVICE FOR CONTROLLED DRUGS

STUART ANDREW WADDELL

PhD **2019**

THE DEVELOPMENT OF AN ELECTROCHEMICAL SENSING DEVICE FOR CONTROLLED DRUGS

Stuart Andrew Waddell

A thesis submitted in partial fulfilment of the requirements of the Robert Gordon University for the degree of Doctor of Philosophy

November 2019

Declaration

I hereby declare that the work or any portion of it, referred to in this thesis has not been submitted in support of an application for another degree or qualification of this, or any other university or institute of learning. This is an original piece of work undertaken by myself. All results and work other than my own are clearly cited and acknowledged.

Stuart Andrew Waddell

Abstract

Forensic chemists can be faced with a wide array of substances to test when attending clandestine drug manufacture crime scenes. Whilst many techniques exist at their disposal such as chemical colour test reagents, and immunoassays, these methods are at best semi-quantitative and often subject to false positives.

Electrochemical methods of detection offer a potential solution to this problem, as the equipment is portable, cheap, and robust. The analysis is quantitative and, if the electrode/electrolyte combination is designed properly, it can be extremely sensitive and selective.

The scientific literature contains many examples of voltammetric analyses of controlled drugs. A square wave voltammetric analysis of the novel psychoactive substance benzyl-piperazine is reported here – representing the first time this analysis has been established. A limit of detection of 6 μM was achieved, and resolution against the similar ecstasy-type drug 3-4-methylenedioxymethylamphetamine (MDMA) was demonstrated.

Two innovative USB powered prototype potentiostats have been developed. As proof of concept, an ATMega328P microcontroller was used in conjunction with 12-bit digital-to-analog and analog-to-digital converters (MAX532 and MCP3304 respectively). Using ferricyanide for redox at a glassy carbon electrode, reversible cyclic voltammetric analyses and square wave linear calibration (2.7 to 13.7 μM, R^2=0.998) were achieved by the first prototype. The second prototype extended the compliance range (from ±2.5 V to ±12 V) and improved the signal to noise ratio. The second prototype also

achieved a linear calibration using square wave voltammetry of MDMA (41 to 82 μM, R^2=0.995) at a carbon paste electrode.

Acknowledgements and Dedication

The project started under the supervision of Professor Pat Pollard and Dr Catherine Inverarity. I was grateful to Prof Pollard for the freedom she granted me in choosing the topic for my PhD. As I was a member of staff within the School of Pharmacy and Life Science, I thought it best to align the research to the topics I was teaching *viz.* electrochemistry and forensic science. Prof Pollard and Dr Inverarity both provided vital guidance at the start of the project. Subsequently, Professor Radhakrishna Prabhu has taken on the role of my principle supervisor and has provided me with sage advice throughout the whole process. Leaning on his wisdom has been essential. Also, Dr Carlos Fernandez was added as an advisor; his wealth of knowledge in the field of electrochemical research has also been invaluable. Further to this, his encouragement has also been a much needed "shot in the arm" when things looked difficult to complete.

To my supervisors, who have magnanimously put up with my ostrich like behaviour at times.

To my wife, who has been an absolute rock throughout, and most of all to יהוה

Thank-you.

Contents

List of Tables

List of Figures

Abbreviations

A	Electrode area (cm^2)
ADC	Analog to digital converter
AdSDPV	Adsorptive stripping differential pulsed voltammetry
AdSSWV	Adsorptive stripping square wave voltammetry
AdSV	Adsorptive stripping voltammetry
API	Atmospheric pressure ionisation
ASSWV	Anodic Stripping square wave voltammetry
A_V	Amplification in voltage
BR	Britton-Robinson buffer
BZP	Benzylpiperazine
C	Concentration (normally M or mol cm^{-3})
CE	Counter electrode
CNT	Carbon nanotubes
CPE	Carbon paste electrode
CSV	Comma separated value filetype
CV	Cyclic voltammetry
D	Diffusion coefficient (cm^2 s^{-1})
DAC	Digital to analog converter
DC	Direct current
DIGIPOT	Digital potentiometer
DMA	N,N-dimethylaniline
DPV	Differential pulsed voltammetry
E	Measured potential (V)
EMCDDA	European Monitoring Centre for Drugs and Drug Addiction
EMIT	Enzyme-multiple immunoassay technique
E^o	Potential under standard conditions (V)
$E_{P/2}$	Half wave peak potential (V)
F	Faradays constant (96485 C mol^{-1})
FPGA	Field programmable gate array
FPIA	Fluorescence polarisation immunoassay
GCC	GNU compiler collection
GCE	Glassy carbon electrode
GC-MS	Gas chromatography mass spectrometry
GUI	Graphical user interface
HDME	Hanging mercury drop electrode
HRP	Horseradish peroxidase
I	Current (A)
I^2C	Inter-integrated circuit protocol
IC	Integrated circuit
ICSP	In circuit serial programmer
IDE	Integrated development environment
INSTR	Instrumental amplifier

I_p	Peak Current (A)
ITO	Indium doped tin oxide
JFET	Junction gate field effect transistor
k^0	Heterogeneous charge transfer rate constant
kET	Rate of electron transfer kinetics
ksps	Kilo samples per second
LCD	Liquid crystal display
LOD	Limit of detection
LOQ	Limit of quantification
LSB	Least significant bit
LSV	Linear sweep voltammetry
MCPE	Modified carbon paste electrode
MDMA	3,4-methylenedioxymethylamphetamine
MSB	Most significant bit
MWCNT	Multi-walled carbon nanotubes
n	Number of electrons
PB	Phosphate buffer
PCB	Printed circuit board
PDE	Platinum disk electrode
pKa	Dissociation constant
PNG	Portable network graphic
PWM	Pulsed width modulation
R	Gas constant (8.314 J K^{-1} mol^{-1})
R	Resistance (Ω)
R^2	Square of Pearson correlation coefficient
RE	Reference electrode
RP	Reverse phase
SAM	Self-assembled monolayer
SAR	Serial approximation register
SCE	Saturated calomel electrode
SDS	Sodium dodecyl sulfate
SEM	Scanning electron microscope
SG	Savitsky-Golay smoothing
SP	Summing point
SPE	Screen printed electrode
SPI	Serial peripheral interface protocol
SSR	Sum of the square of the residuals
SWCNT	Single walled carbon nanotubes
SWV	Square wave voltammetry
T	Temperature (K)
TIA	Transimpedance amplifier
TLC	Thin layer chromatography
TMB	Tetramethylbenzidine
TMBox	Oxidised tetramethylbenzidine
UART	Universal asynchronous receiver/transmitter
USB	Universal serial bus

UV	Ultra-violet
WE	Working electrode
α_{OX}	Activity of species in oxidised form
α_{RED}	Activity of species in reduced form
β	Transfer coefficient
γ_{OX}	Activity coefficient of oxidised species
γ_{RED}	Activity coefficient of reduced species
δ	Current differential (μA)
ΔE_P	Difference in peak potential (V)
Λ	Ratio between the rate of charge transfer and the rate at which the experiment is conducted
υ	scan rate (V s^{-1})

1. Introduction

1.1. Dynamic Electrochemistry

Ever since the Nobel Prize winning technique of polarography was first introduced by Heyrovský in 1922 (1), there has been a continual expansion in the use of dynamic electroanalysis. The majority of the research over the decades has focussed on the materials that are used for the electrodes/electrolytes and understanding the kinetic behaviour of species in solution. The most commonly used form of this technique is known as voltammetry whereby the sensing electrode is swept across a range of potentials and the resultant current is measured (2).

Year on year has seen advances in terms of selectivity and sensitivity such that voltammetry regularly achieves nano- and pico- molar detection limits for a wide range of analytes such as metals (3,4), drugs and similar sized organic compounds (5-7) and biological macromolecules (8,9). One of the main advantages of this technique however remains its ability to deduce information on reaction mechanism making it one of the most relied upon techniques whenever this information is required (10,11).

1.1.1. The Voltammetric Cell

The simplest form of electrochemical cell consists of two electrodes: a working electrode which is sensitive to analyte activity, and a reference electrode which holds a set potential. This set-up is used in potentiometric analyses whereby analyte concentration is determined by measuring potential difference between the two electrodes under equilibrium conditions.

These two parameters are related by the Nernst Equation (12) shown in Equation 1-1:

Equation 1-1: $$E = E° - \frac{RT}{nF} ln\left(\frac{\alpha_{RED}}{\alpha_{OX}}\right)$$

Where E is the measured potential difference (V), E° is potential that would be measured under standard conditions (V), R is the gas constant (8.314 J K^{-1} mol^{-1}), T is absolute thermodynamic temperature (K), n is the number of electrons involved in the redox reaction, F is Faraday's constant (96485 C mol^{-1}), α_{RED} and α_{OX} are the activities of the reduced and oxidised forms. In contrast to potentiometric analyses which take place under equilibrium conditions, voltammetry is a *dynamic* electrochemical method (*i.e.* there is a net passage of current in a particular direction). As a consequence, the voltammetric cell is composed of three electrodes *viz* the working electrode, the counter electrode, and the reference electrode. It is the flow of current under applied potential which is the signal of interest and closely monitored, however if a current flows through a reference electrode, it will inevitably change potential thereby rendering it useless. Therefore, in order to enable current to flow through the entire circuit a counter electrode is used which performs half of the overall redox reaction. The other half is performed at the working electrode which is where the analyte reacts (13). A generalised schematic of a voltammetric cell is shown in Figure 1-1.

For dynamic analysis, most modern instruments can use potentiostatic (maintain applied potential and measure current) or galvanostatic (*vice versa*) control. However, the term "voltammetry" implies potentiostatic operation that also most commonly exploits the diffusion of the analyte towards the working

electrode (WE) in a quiescent solution. Further details on this will be provided in section 1.3 Electrolyte Considerations.

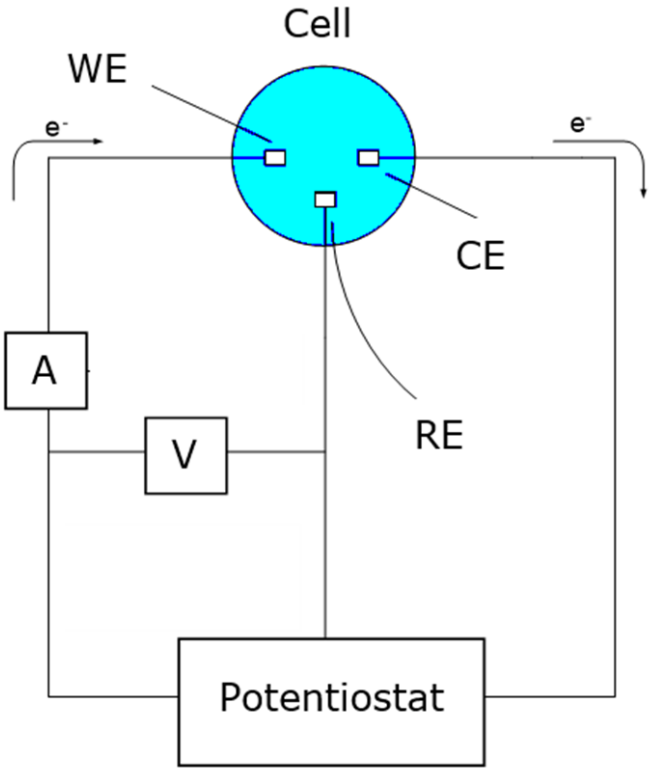


Figure 1-1: Schematic of voltammetric cell. Abbreviations: working electrode (WE), counter electrode (CE), reference electrode (RE), ammeter (A), voltmeter (V). Note that the meters are intrinsic to the potentiostat but are shown here to indicate the series and parallel nature for current and voltage measurement respectively.

1.1.2. The Voltage Profile and Voltammogram

In terms of practical methodology there are two important aspects to consider. Firstly, the voltage profile which details exactly how the potential is changed on the WE over time, and secondly the voltammogram which is the resultant plot of current *vs* potential. This simplest form of voltage profile involves a linear sweep where the start and stop potentials along with the sweep rate need to be defined. In reality, due to instrumental considerations, most measurements are made in a staircase fashion as a result of the digital to analog converter's (DAC) capability. However, as the DAC on modern potentiostats are typically 16-bit, this translates to a theoretical resolution of approximately 0.15

mV, (assuming a 10 V compliance range). This is more than small enough to very closely represent a perfectly linear sweep, although strictly speaking it is not. It should also be noted that often the electrochemical cell itself is often the greatest source of noise in the measurement, commonly resulting from a sluggish response from the reference electrode (RE). Figure 1-2 shows the voltage profile and voltammogram for digitally simulated data using the procedure given by Wang *et al.* (14).

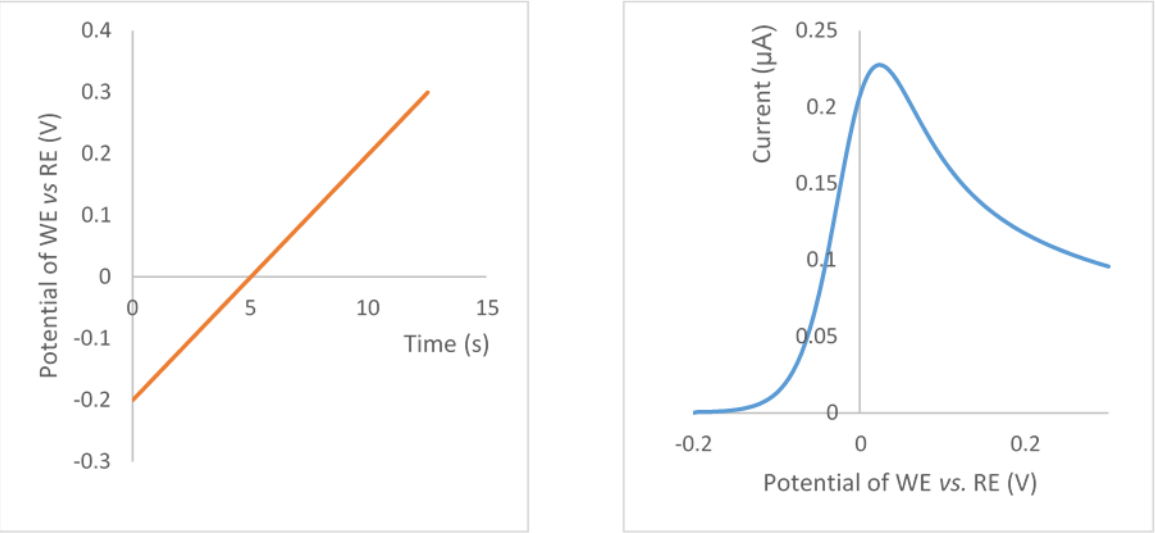


Figure 1-2: Left: Theoretical voltage profile starting at -0.2 V, terminating at 0.3 V with a sweep rate of 40 mV s^{-1}. Right: Digitally simulated voltammogram with formal potential 0.013 V, 1 electron oxidation, 6 x10^{-8} mol cm^{-3} analyte concentration, 1 x10^{-5} cm^2s^{-1} analyte diffusion coefficient, 2.54 x10^{-2} cm^2 electrode area, 0.5 transfer coefficient with a 1 cm s^{-1} heterogeneous rate constant. A digital simulation was chosen to show the ideal peak shape which is ultimately dependent on analyte concentration at the electrode surface.

The first important parameter to draw from the voltammogram is the peak current which is directly proportional to current in diffusion-controlled systems. Secondly the peak potential and half wave potential provide important qualitative information. Often researchers will vary the scan rate in order to see how the peak potential varies in order to determine aspects of the mechanism such as how many electrons are involved in the process. Linear sweep voltammetry (LSV) thus can be used for both quantitative analytical chemistry (15) and kinetic studies (16). Further mechanistic information is possible using

the technique of cyclic voltammetry (CV) whereby an initial LSV is performed then potential on the WE is swept back in the opposite direction at the same rate. The reversibility of the analyte has the largest influence on the shape of the voltammogram. By way of example Figure 1-3 illustrates the data generated from CV analysis on a real sample.

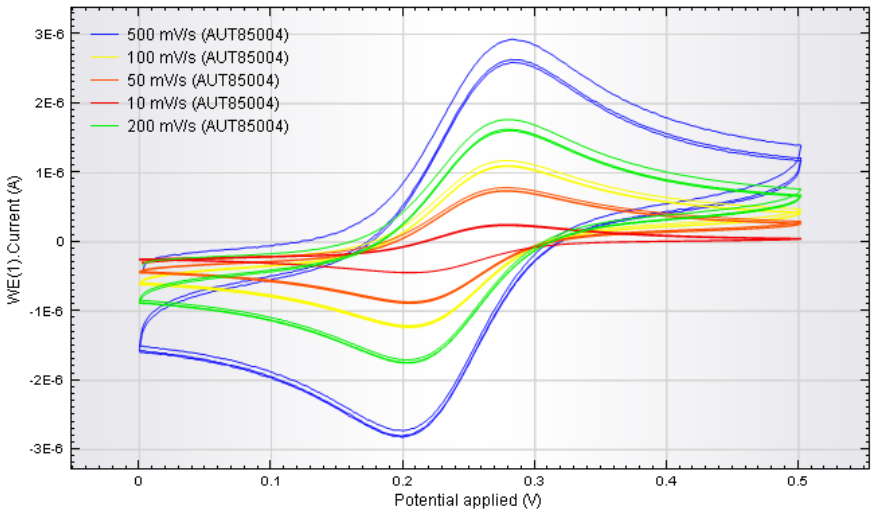


Figure 1-3: Overlay of CV data from a solution of 200 μM $K_3Fe(CN)_6$ in 0.1 M KCl at a 3 mm glassy carbon WE (run on a Metrohm, Autolab, PGStat128N potentiostat). Scan rates shown in legend.

In Figure 1-3 both the anodic and cathodic waves are visible and it is clear that increasing the scan rate increases the peak height. For an analyte with reversible kinetics the Randles-Sevčik governs how peak current varies with the relevant parameters as shown in Equation 1-2.

Equation 1-2: $$I_P = 2.69 \times 10^5 n^{\frac{3}{2}} A D^{\frac{1}{2}} C v^{\frac{1}{2}}$$

Where A represents electrode area (cm^2), D is the diffusion coefficient ($cm^2\ s^{-1}$), C is concentration ($mol\ cm^{-3}$), υ is the scan rate ($V\ s^{-1}$), other variables as previously stated. Therefore, a common test to see whether a process operates under diffusion control (or if there is adsorption occurring at the electrode instead) is to check that the current is indeed with linear with respect to the square root of the scan rate.

The most efficient voltage sweep profile, in terms of analyte sensitivity, is employed during square wave voltammetry (SWV). Here the voltage is still swept from a start to final potential, however, superimposed on the staircase is square wave pulse resulting in the WE attaining an alternating reducing and oxidising characteristic. An example voltage profile for SWV is shown in Figure 1-4.

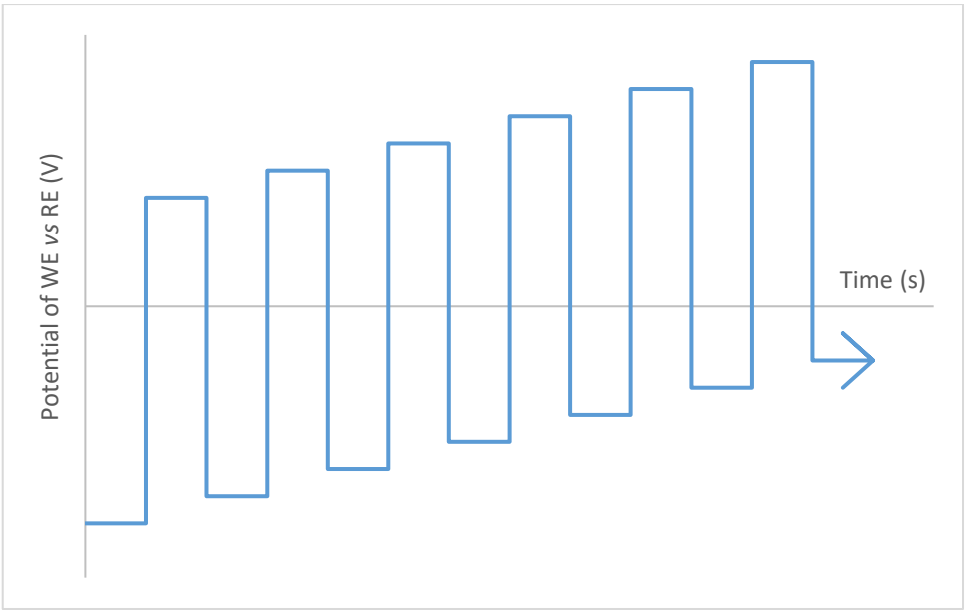


Figure 1-4: Basic representation of first six cycles of SWV voltage profile.

SWV is a differential method meaning the current is measured right at the end of each pulse and for each step the cathodic current is subtracted from the anodic one.

There are several advantages to running the scan this way:

1) Measuring the current at the end of the pulse serves to minimise the effect of the non-Faradic (or charging) current. It is solely the Faradic current that contributes to the quantitative analysis and the non-Faradic interferes with the measurement. However, the non-Faradic current decays over time at a

faster rate than Faradic current so adding a delay of approximately 20 ms takes advantage of this.

2) Due to the alternating nature of the wave, each step converts the analyte into the correct form (*i.e.* reactant) prior to the subsequent steps in the immediate environment of the WE. Having a high concentration of the analyte within the diffusion zone increases the signal.
3) The cathodic and anodic currents are of opposite sign. By convention in Europe cathodic waves tend to be represented as negative currents. This sign difference measured at each pulse effectively doubles the signal when they are subtracted.
4) Differentiation of the signal results in the sigmoidal wave being converted to a wave more akin to a Gaussian peak. This is advantages when detecting small signals.

To attain the best limits of detection (LOD), the pulse height, step height and frequency of the square wave profile can be optimised.

1.2. Electrode Considerations

Of primary importance in the consideration of electrode material is conductivity. Traditionally pure noble metals and mercury have been used for this reason and due to their non-reactivity. These electrodes can be used for both metallic and organic analytes in solution and have been used in this way since the middle of the twentieth century (17,18). The non-reactivity of the electrode is also essential in order to reduce/oxidise the analyte as reversibly as possible. Mercury itself has several distinctive advantages (19):

1. As it is a liquid a new surface can be continually produced for each analysis. This negates the often-cumbersome problem of electrode poisoning.

2. Using mercury affords the analyst access to a wide range of potential owing to the high hydrogen overpotential on its surface. Therefore, a wider range of analytes can be examined.
3. As mercury easily form amalgams with most metals it makes it a prime candidate for use in stripping analysis whereby the analyte metals are reduced *into* the electrode in the pre-concentration step.

It is widely known that mercury is highly toxic even at low doses to the nervous, renal, immune, reproductive and cardiovascular systems in humans (20). Therefore, there has been a move away from its use, however due to its superior nature as an electrode, it is still employed mainly as a thin film electrode in modern research. Initially these thin film electrodes were prepared by simple abrasion of a platinum electrode in pure mercury (21). Recently though this method of electrode preparation has been replaced by the reduction of mercury (II) ions in solution onto a suitable substrate electrode (22).

1.2.1. Carbon as an Electrode Material

The semiconductor nature of graphite (crystalline) and glassy (amorphous) carbon does not limit its use as an electrode in voltammetric experiments. Indeed, both find wide application in contemporary electroanalysis although more commonly as modified electrodes rather than as bare carbon. This is where the carbon acts as a substrate upon which a more electro-selective material is placed. By way of example Lu and Sun successfully formed a Nafion® coating onto a 3mm glassy carbon electrode in order to detect the pesticide paraquat shown in Figure 1-5 (23).

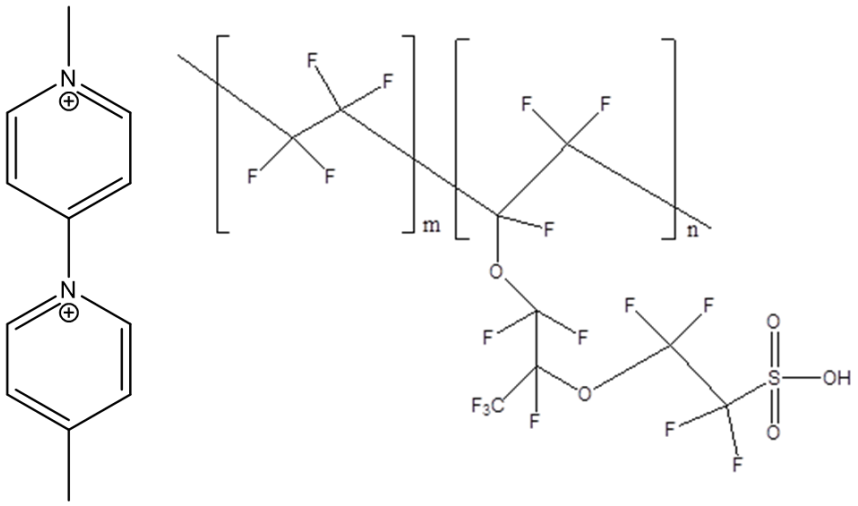


Figure 1-5: Chemical structures of paraquat (left) and Nafion® (right)

Due to the ammonium groups being present in paraquat's aromatic rings, it is highly electro active. Their method involved a pre-concentration step which was performed at open circuit potential. Nafion® itself is essentially polytetrafluoroethylene except for having additional branched chains which terminate in a sulphonic acid group. This means the polymer can carry a charge and hence is regularly employed in this situation. The preparation in this case involved spin coating 4 µL of the polymer dissolved in alcohol directly onto the glassy carbon surface. After pre-concentration cathodic pulsed differential voltammetry was applied from -0.3 to -1 V in the presence of background electrolyte (0.1M $NaClO_4$) and permanganate. This outcome of the research produced a linear range of 1 to 100 ppb paraquat with an LOD of 0.5 ppb.

1.2.2. Carbon Paste Electrodes

The original form of carbon paste electrode (CPE) was a mixture of carbon powder with an organic liquid, normally a mineral oil such as nujol. The liquid phase of the electrode was chosen specifically to be non-electroactive. Originally the CPE was invented to be an alternative to the dropping mercury electrode, although as the CPE did not have a high degree of mobility it was not

of satisfactory use in that respect (24). It did however find application in research through the twentieth century gaining in widespread use as time went on. The evolution of the CPE began in the late 1950's early 1960's with the unmodified type of electrode being used for the analysis of various organic compounds. The earliest form of the electrode was a mixture of carbon powder and bromoform (1:7) contained in a Teflon tube with a platinum wire inserted for connection to the potentiostat (25). The 1970's saw the great leap forward in terms of modification of the electrodes, that is to say other substances were incorporated into the electrodes in order to improve sensitivity and selectivity. Mesarić and Dahmen incorporated a silver halide-silver sulphide mixture into a carbon powder-nujol and carbon powder-paraffin electrode and were able to obtain responses for Cl^-, Br^- and I^- at 5 $x10^{-5}$, 1 $x10^{-5}$ and 5 $x10^{-7}$ M respectively (26). A later, more advanced example from Yao and Musha used a nicotinamide adenine dinucleotide modified electrode in order to detect ethanol and lactic acid by LSV (27). The linear working range they obtained was 0.05 to 2 $x10^{-9}$ M. Modified electrodes of this type are normally given the acronym MCPE.

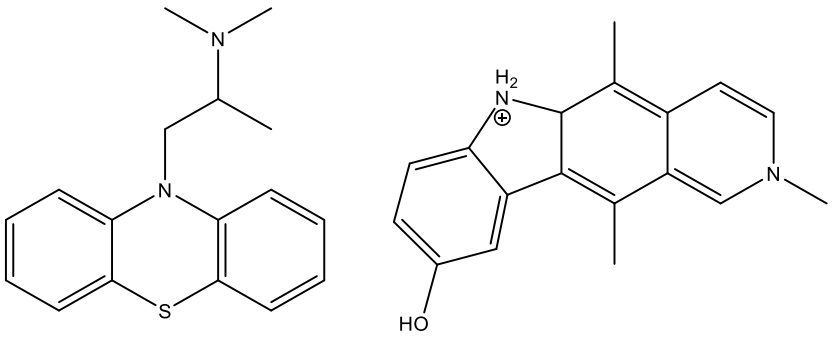


Figure 1-6: Chemical structures of promethazine (left) and celiptium (right)

Historical reviews have called the period of 1980-1990 the "Era of Chemically Modified Carbon Pastes" due the large volume of research being published on this topic (24). Two illustrative examples are given here to typify

the type of work which was being done during the period. Firstly, where promethazine (structure shown in Figure 1-6) was chosen as a model compound in order to investigate the viability of using fatty acid modified electrodes to detect drugs (28). This was shown to be successful and the LOD for this drug was as low as 100 pM although the exact composition of the paste did affect the result.

Later analysis of an anti-cancer drug celiptium (structure shown in Figure 1-6) used lipid MCPE for the analysis by CV, differential pulsed voltammetry (DPV) and adsorptive stripping voltammetry (AdSV). As with most research of this kind, factors such as pH, ionic strength and interfering ions were investigated but an excellent detection limit of 200 pM was obtained (29).

Enzymatic MCPE came to the fore of research during the 1990's which now involves the analysis of a range of analytes from very simple molecules such as H_2O_2 through drug-type molecules (*e.g*. ascorbic acid, caffeine, dopamine and many more similar biologically important compounds) to DNA (30).

Two examples from the period are given again to illustrate the command that the technology had at the time in order to perform this type of analysis. Previously discussed techniques have all involved voltammetry which is under diffusion control, however it is also possible to perform amperometric analysis which is under convection control. Either by the use of a rotating electrode or flowing stream of electrolyte, the mass transport towards the electrode can be controlled reproducibly. One such analysis on a rotating MCPE was performed in order to detect dopamine (31). The modification was provided by polyphenol oxidase sourced from potato tissue. Whilst the analysis was successful with an LOD of 2.5 mM being attained, several molecules (dopac, ascorbic acid and

pyrocatechol) were also found to interfere. Nevertheless, as is often the case, the real strength of this work lay in the ability of CV to be able to interrogate the mechanism of the analyte at the particular electrode's surface.

It is also possible to use a MCPE electrode for the detection of glucose as was demonstrated by Yabuki and Fumino with a view to the application of analysis of soft drinks (32). The electrode was modified with polyethylene glycol modified glucose oxidase, horseradish peroxidase and 1,1-dimethylferrocene. This analysis obtained an LOD of 20 μM and had a linear working range of up to 500μM. An additional advantage of this electrode was the selectivity as the interferants investigated did not have a significant impact on the application at their anticipated levels. The dimethyl ferrocene was added as a mediator; this is a compound that is used to increase the rate of electron transfer (k_{ET}). This always has a beneficial effect on the signal produced. Often the k_{ET} can be too slow for voltammetric analysis to be viable so a fast transfer "electron bridge" is used. In order to be an effective relay a mediator should (33):

1. Be highly electrochemically reversible (ferrocene and related compounds find wide application as mediators for this very reason).
2. Have a redox potential with the working range of the analysis.
3. React rapidly with both the enzyme and the electrode.
4. Have activity which is independent of pH and oxygen concentration.
5. Be stable in both reduced and oxidised form.
6. Be non-toxic where clinical applications are anticipated.

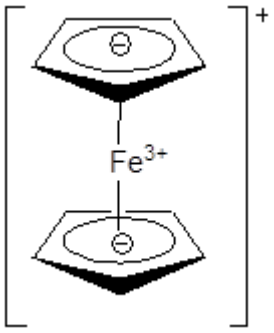


Figure 1-7: Ferrocene in its oxidised state

Ferrocene is a complex of two cyclopentadiene rings which donate electrons to an iron centre which itself is either in oxidation state +2 or +3. As each ring can be considered to have an overall negative charge this means ferrocene can alternate between oxidation states 0 and +1 (oxidised form shown in Figure 1-7). Real advances have been made by substitution of the cyclopentadiene rings in order to change the redox potential of the mediator making it suitable for a range of different enzymes. Ferrocene continues to find application for voltammetric analysis in modern research (34,35).

1.2.3. The Screen-Printed Electrode

The 1990's also saw the advent of the screen-printed electrode (SPE) for a variety of analytes. As shown in Figure 1-8 all three voltammetric electrodes are in close proximity and have connectors that can be plugged into a custom made potentiostat.

The distance between the working and counter electrode is important in terms of the compliance voltage of the potentiostat. This distance should be minimised in order for the potentiostat to be able to apply as small a potential as possible to the counter electrode.

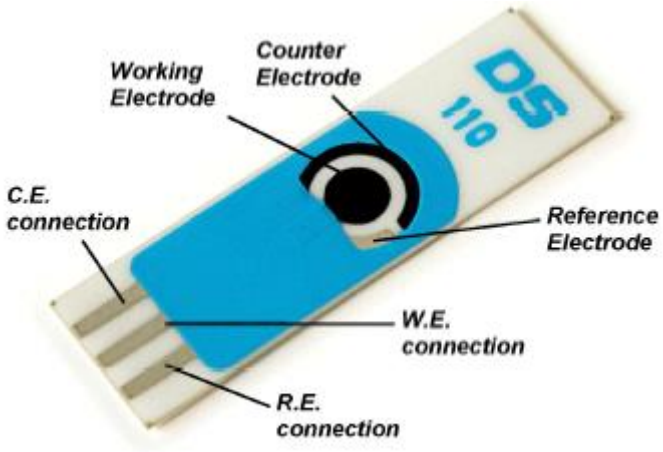


Figure 1-8: Typical layout of SPE (36).

Whereas the distance between the working and reference electrode contributes to what is commonly called the "IR drop" of the cell (the current "I" multiplied by the resistance "R" producing its own voltage). Current flows during a voltammetric analysis largely due the redox reactions which take place and there will be a certain resistance of the solution between electrodes. As the reference is the sole way the analyst can know the potential of the working electrode, the IR drop can contribute to its value skewing the actual working electrode potential from that indicated by the potentiostat (37).

Modification of the carbon SPE was shown as early as 1995 when cobalt phtholcyanine was incorporated into the carbon ink for the monitoring of hydrazines (38). This research also used the alternative of coating the carbon on the surface with mixed valency ruthenium cyanide. Electrodeposition of the ruthenium cyanide was used and as it commonly done cyclic voltammetry was used to determine the efficiency of the film. As both Ru^{2+} and Ru^{3+} cyanide complexes were used there were two distinct peaks anodic peaks which gradually increased in peak current upon subsequent cycles indicating film growth. The sensors produced hydrazine and 1,2-dimethylhydrazine detection limits of approximately 5 µM (estimated from the calibration curves) up to a

linear range of 0.15 mM. This sensor was an illustrative example of how this technology is often designed to be cheap and disposable.

An example from 2009 demonstrated the use of an unmodified carbon SPE in the detection of catechol (39). Although bare graphite was used as the working electrode, it was printed onto a layer of gold in order to negate the effect of the potential drop over the surface due to the semi-conductor properties of graphite. A detection limit of 290 nM was obtained but interestingly through the use of SWV the interferant effect of ascorbic was overcome as there was a separation in the anodic waves of approximately 0.2 V. SPE technology continues to evolve with 149 articles being published on the Elsevier database Science Direct using the term "screen printed electrode" in the title or abstract in 2019 (as of March 13th).

1.2.4. Carbon Nanotubes in Electrochemical Sensing

Over the past fifteen years the use of carbon nanotubes (CNT) in electrochemical sensing has received a great focus of attention. Since the 1996 Nobel Prize in Chemistry for the discovery of fullerenes there has been many discoveries as to the different ways carbon catenates on the nano scale (Nobelprize.org). The most prominent forms that find use are known as single walled and multi walled carbon nanotubes, SWCNT and MWCNT respectively, illustrated in Figure 1-9.

There are three main techniques used for the manufacture of CNT:

1. The arc discharge method – this is generally the simplest method whereby two graphite rods held at high potential are slowly brought towards each other. At some point an electrical arc between the rods will form to equate

the electrical potential during which the carbon is atomised and re-condenses forming the microtubules. If this method is used the sample will require further solvent extraction to remove the CNT from the soot that is also formed (41).

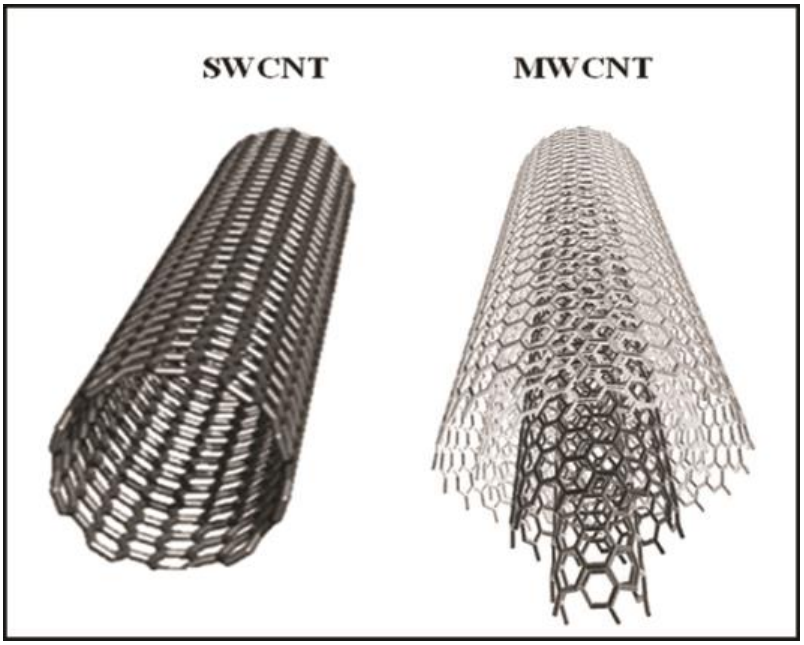


Figure 1-9: Computer generated representation of single- and multi- wall carbon nanotubes (40).

2. The laser ablation method – here an intense pulse from a laser is aimed at a carbon target held within a furnace at approximately 1200 °C. The carbon also contains small amounts of nickel or cobalt which act as a catalyst. These metals were chosen specifically for their electronegativity which needs to be high enough in order to avoid the formation of fullerenes. An inert gas is passed through the furnace and as the ablated graphite condenses it forms a high yield (>70%) of SWCNT (42).
3. The chemical vapour deposition method – this involves small organic molecules (commonly CH_4, C_2H_2 and EtOH) entering a reaction chamber alongside nanoparticles of a catalyst (commonly nickel, cobalt or iron). The chamber is heated to 700 – 900 °C and the CNT particles are grown from the

catalyst surface. For this reason, the actual size of the catalyst particle has a major influence on the diameter of the nanotube (43). This is as opposed to the laser ablation method whereby a single atom of the catalyst is thought to initiate growth.

The specific advantages found in the use of CNT in electrochemical sensing can be summarised as follows (44,45):

1. High electronic conductivity.
2. High mechanical strength.
3. It is possible to chemically modify CNTs and a wide variety of moieties can be added to the tubes thereby improving sensitivity and selectivity. The additions can be categorised by four techniques (46):
    a. Defects in the sidewalls and rims can be functionalised. This is performed by chemical treatment whereby the desired functional groups are often simple amides and carboxylic acids.
    b. Non-covalent interactions can be employed such as the $\pi - \pi$ interactions between pyrene and SWCNT. This has the advantage of not causing any damage to the CNT structure.
    c. Covalent functionalisation of the sidewalls is also possible. Most chemical treatments can be achieved such as nucleophilic, electrophilic, radical and cyclo additions (47).
    d. Endohedral inclusion is whereby the cavity of a CNT is filled with another species. Originally only elements such as gold and silver were used but recently other species such as Cr_2O_3 have been used (48).
4. Synergetic effects have been noticed when CNT are combined with conducting polymers such as Nafion resulting in increased signals.

5. Sensitivity of electrodes using CNTs is thought to be improved due to their high specific area.
6. As with all nanotechnology miniaturisation is a possibility, but the chemical stability of CNTs even renders ultrasensitive single nanotube sensors possible.
7. CNTs exhibit a good degree of biocompatibility which is of course essential if redox reactions involving enzymes are to be used in biosensors.
8. There is the possibility of constructing microarrays from CNTs which are ultrasensitive. It should however also be stated that other conducting polymers such as polypyrrole, polythiophene and polyaniline can be used to construct microarrays of this sort (49).

Two examples are given here to illustrate the usefulness of CNT in terms of electrode modification. Firstly, morphine was successfully determined using a CPE that had been modified with 4-hydroxy-2-(triphenylphosphonio)phenolate (see Figure 1-10) and MWCNT (50). This is the first time that morphine has been successfully analysed at a CPE in the presence of paracetamol which is a commonly found adulterant in heroin samples. DPV was used to provide a linear range of 1 – 950 μM (although composed of two linear sections) and a detection limit of 66 nM was achieved. In this case the MWCNT were involved the mechanism of the electrocatalytic oxidation of the morphine.

Secondly nanomolar sensitivity was also obtained for the anti-hypertensive drug amiloride at a Nafion®-CNT modified glassy carbon electrode. This work contained a clear comparison between the glassy carbon electrode (GCE) alone, Nafion®-GCE and Nafion®-CNT-GCE. It was apparent from electrochemical

impedance studies that the CNT greatly reduced resistance to charge transfer (51).

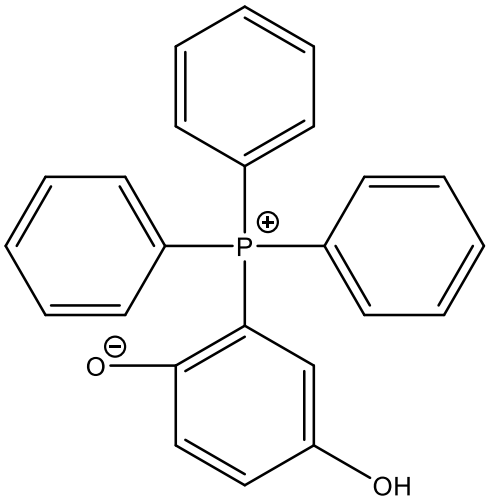


Figure 1-10: Structure of 4-hydroxy-2-(triphenylphosphonio) phenolate (HTP)

1.2.5. Other Electrode Base Materials

The are many other materials which can act as a base for electrodes prior to modification. A notable electrode material which is regularly used in electroanalysis is gold. As gold has the ability to form self-assembled monolayers with many compounds great functionality can be derived (52-54). Another electrode which is growing in popularity is the bismuth film electrode. This is akin to the mercury film electrode and is commonly used for stripping analysis as Sopha *et al* showed with their analysis of sildenafil citrate (55). There are indeed a wide range of electrodes available to the electrochemist (*e.g*. indium doped tin oxide electrode's use in spectroelectrochemistry (56) or the low capacitive currents and wide electrochemical window of the boron doped diamond electrode (57)) each with their own application. Nevertheless, carbon remains one of the most useful for the analysis of organic compounds.

1.3. Electrolyte Considerations

Conductivity is also of high importance when considering which electrolyte to use for voltammetric analysis. This is not normally a problem in the analysis of aqueous samples as a suitable salt is normally present alongside the analyte in order to form the background electrolyte. This non electroactive background electrolyte is also used to mask the charge of the electrodes thereby eliminating the migration of the analyte, an essential feature of diffusion-controlled voltammetry. In aqueous solutions, buffers can be used as the background electrode as they are normally stable at the working potential range.

However organic modifiers such as acetonitrile may be necessary for some analytes to attain solubility. A further development over the last ten years has been the use of room temperature ionic liquids as a background electrolyte in voltammetric experiments (10,58). There are many possible ionic liquids available but the general advantages of using them are:

1. A wide electrochemical potential window, a more extreme potential can be applied to the working electrode without solvent electrolysis.
2. No need to add other salts to the solution as ionic liquids have a high conductivity.
3. Thermal and chemical stability and with negligible volatility.

1.4. Project Rationale

The previous sections in the chapter have demonstrated the efficacy of voltammetry to a wide range of applications. It is to the analysis of substances controlled by Misuse of Drugs Act 1971 in the UK that this thesis seeks to address.

From personal experience of working for a forensic provider and attending crime scenes where the clandestine manufacture of drugs was taking place, the rationale for this work is drawn from first-hand knowledge of the limitations an analyst can feel in that situation.

There are smaller versions of analytical instrumentation available such as a portable gas chromatograph (59). However, for the majority of forensic agencies, these devices are either prohibitively expensive, impractical or unreliable. Therefore, the scene going scientist often has to rely on presumptive tests which are semi-quantitative at best and can be subject to false positives. It is asserted here that voltammetric analysis would make an ideal solution for field testing of bulk drugs. This is due to obvious advantages such as low cost of instrumentation, portability, and ease of use. By careful design of the electrode/electrolyte system in combination with an optimised voltage profile it should be possible to achieve highly sensitive and selective analysis.

The vast majority of controlled drugs are alkaloids (although γ-hydroxybutyric acid and Δ^9 tetrahydrocannabinol in Cannabis are two notable exceptions). The structures of amphetamine (a phenylethylamine) and cocaine (a tropane) are shown in Figure 1-8 by way of example. Therefore, the next chapter will discuss the voltammetric oxidation of amines as this is the most likely target moiety due to the lone pairs of electrons which can be stripped at the electrode interface.

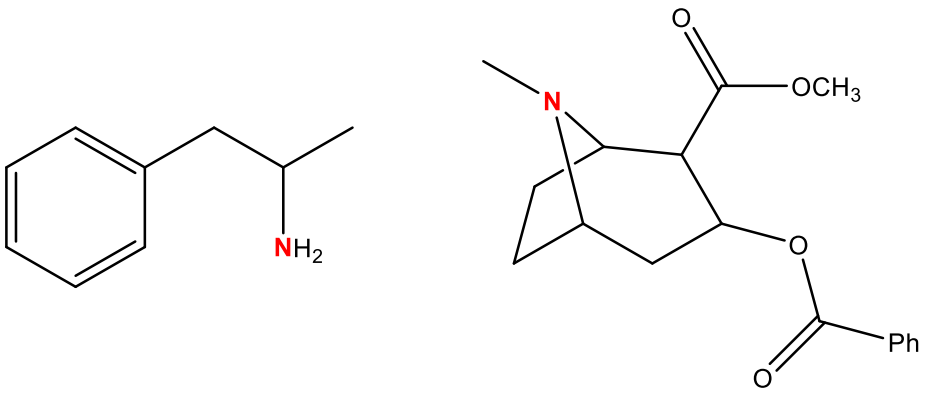


Figure 1-11: Chemical structures of amphetamine (left) and cocaine (right) highlighting their primary and tertiary amine groups respectively in red.

As it can be shown that voltammetric analysis is applicable to the analysis of controlled drugs (see chapters 2 and 3), the overall rationale for this project is to take the technique out of the laboratory environment – where bulky expensive equipment is used – and put it to use in the field. This project will inevitably show what is necessary when designing an analytical system from the ground up. Future commercialisation of the device is a driving factor of the project. Stated alternatively, to have a direct impact on forensic practice in the UK is a goal. To achieve this end, the system that is designed must also be easy to use.

1.5. Project Aims and Objectives

The overall aim of this project is to develop a working prototype of a potentiostat which can be used for on-site analysis. This introduces design constraints in terms of power delivery, size, functionality, and cost. In order to achieve this several objectives must be met:

☑ The general applicability of the technique to forensic analysis must be demonstrated. This will be achieved by way of a synthesis of the scientific literature which exists on the topic.

- ☑ The technique should be expanded to prove that novel analysis of substances not found in the literature is tenable. This will be achieved by the development of an electrode/electrolyte interface for the analysis of the novel psychoactive substance benzylpiperazine (not in the scientific literature at the time of writing).
- ☑ A prototype electronic circuit must be developed that fulfils the requirements of a potentiostat (*i.e.* user-controlled rate of sweeping of a DC potential whilst measuring the current). In addition, appropriate firmware must be developed to control the timing of measurements.
- ☑ A graphical user interface must be developed to relay the method information to the potentiostat and read the signals produced during scans. This should enable the user to log and manipulate data.
- ☑ The overall performance of the instrument must be assessed. This can be addressed by monitoring the signal integrity as differing averaging techniques are employed to improve the signal to noise ratio.

A combination of hardware and software techniques will need to be employed to minimise noise. Whilst this thesis will attempt to separate these issues into separate chapters, they must be developed in tandem by necessity. Nevertheless, the 'acid test' will be the successful analysis of a controlled drug. 3, 4-Methylenedioxymethylamphetamine (MDMA) has been chosen as a suitable material to determine the validity of the instrument.

2. The Electrochemical Oxidation of Amines

2.1. Introduction to Mechanistic Elucidation

The ability to probe and theoretically elucidate redox mechanisms from species as diverse as inorganic ions (60) through simple organic molecules (61) to enzymes (62) is a real strength of voltammetry. However, many mechanisms are complex involving several stages of charge transfer or coupled homogeneous chemical reactions (*e.g.* with solvent) and therefore voltammetry alone may be insufficient to definitively state the processes involved. In such cases the voltammetric data can be used in combination with other techniques such as spectroscopy either independently or simultaneously as in spectroelectrochemistry (63). The latter is commonly achieved by using an optically transparent electrode such as a thin layer of tin oxide doped with indium (ITO) on quartz glass (64). Nevertheless, there are certain mechanistic parameters which electrochemists rely solely on voltammetry to determine. These include: the number of electrons and other species involved, the diffusion coefficient of the redox species, homo/heterogeneous rate constants and charge transfer coefficients. Matsuda and Ayabe introduced a novel parameter (Λ) in 1955 (65) which forms a ratio between the rate of charge transfer and the rate at which the experiment is conducted (given in Equation 2-1). When $\Lambda > 15$ the process can be considered fully reversible, values of Λ between 15 and 10^{-3} produce quasi-reversible voltammetric waves and when $\Lambda < 10^{-3}$ the process is irreversible (66).

Equation 2-1: $$\Lambda = \frac{k^0}{\left(\sqrt{\frac{FDv}{RT}}\right)}$$

Where k^0 is the heterogeneous charge transfer rate constant and the other variables take their normal meaning. The parameter most easily altered by the electrochemist is the sweep rate (υ) and thus the peak shape can be altered using cyclic voltammetry (CV) in order to investigate mass transport and charge transfer rates. Furthermore, by varying the pH of the background electrolyte, the number of protons involved in the redox process can be determined by analysis of the shifting peak potential.

Much of the groundwork to determine the mechanism of amine oxidation was laid in the 1960's and a synopsis is provided here.

2.2. The Oxidation of Dimethylaniline

Some early work by Mizoguchi and Adams (67) investigated the oxidation of N,N-dimethylaniline (DMA) at a platinum WE. Initially they proposed two mechanisms at pH 2-3 as shown in Figures 2-1 and 2-2, both of which begin by the removal of the anilinium proton prior to charge transfer. The major difference between the two mechanisms relates to the number of electrons involved in the first oxidation step. In mechanism one, two DMA molecules are subject to single electron oxidation and then dimerise to form tetramethylbenzidine (TMB) which can itself go on to be further oxidised. In mechanism two, a doubly oxidised molecule reacts with an unoxidised DMA to form TMB (which could of course be oxidised itself as in mechanism one). Therefore, each mechanism leads to the same final product.

The applicability of dynamic electroanalytical techniques to mechanism elucidation was achieved by Mizoguchi and Adams' use of LSV, Tafel analysis and CV.

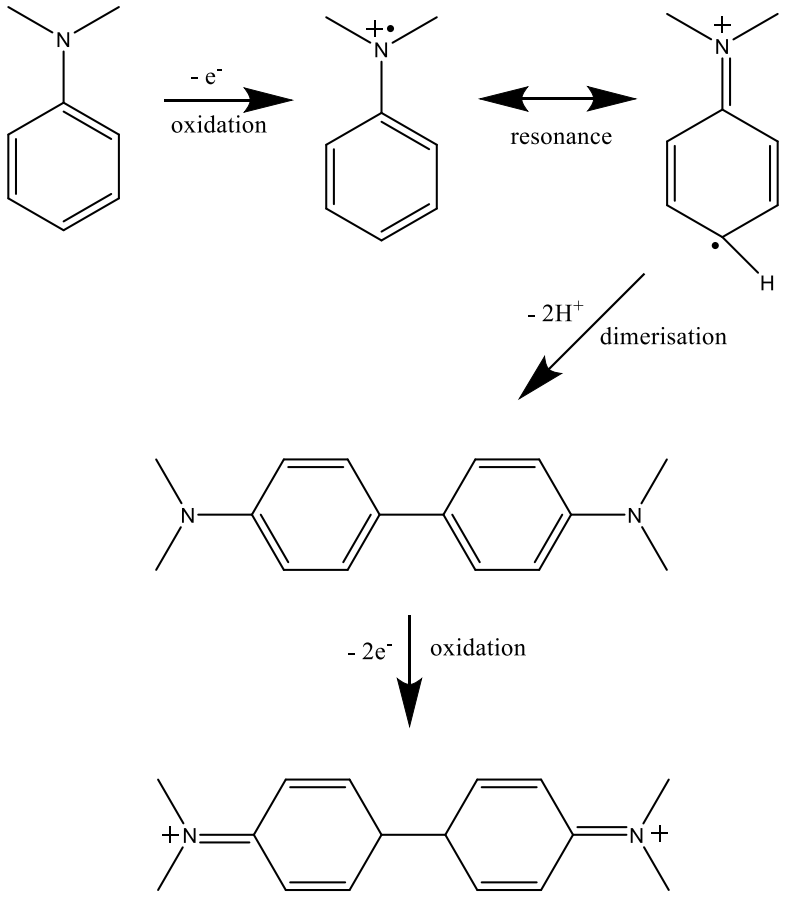


Figure 2- 1: DMA oxidation mechanism one with TMB being subject to further oxidation

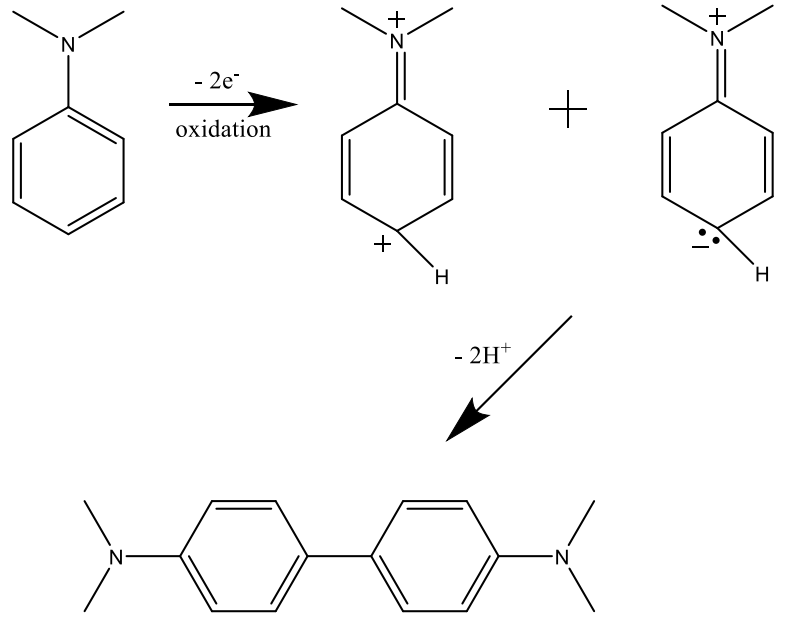


Figure 2- 2: DMA oxidation mechanism two terminating with TMB

2.2.1. Linear Sweep voltammetry Results

Using Britton-Robinson buffers (BR) from pH 2-9 in 1 M Na_2SO_4, the expected amine oxidation variation of half peak potential ($E_{P/2}$) with pH was found with a break at pH 4.9 corresponding to the pK_a of DMA. No change in peak current was noted from pH 1 to pH 4. However, from pH 4 to pH 8 there was a linear increase with pH leading to pH 8 having a 30% higher peak current. The oxidation potential of TMB was a few hundred millivolts less anodic than DMA. This meant that as soon as TMB was formed it would be oxidised. Further to this solubility reasons meant TMB may precipitate above pH 4. These factors taken together mean TMB could not be expected to be seen as a final product in their subsequent spectrophotometric detection of the products.

2.2.2. Tafel Analysis Results

A constant potential was applied to the WE of +0.63 V *vs.* a saturated calomel electrode (SCE) for a range of DMA concentrations. Then using Vetter analysis (68) – where the slope of the graph with log of analyte concentration as abscissa and log of resultant current as ordinate gives the order of the reaction (shown in Figure 2-3) – they were able to deduce the number of electrons transferred was two and that the order of the reaction was 1 with respect to DMA for the rate determining step. This strongly supports mechanism two rather than one.

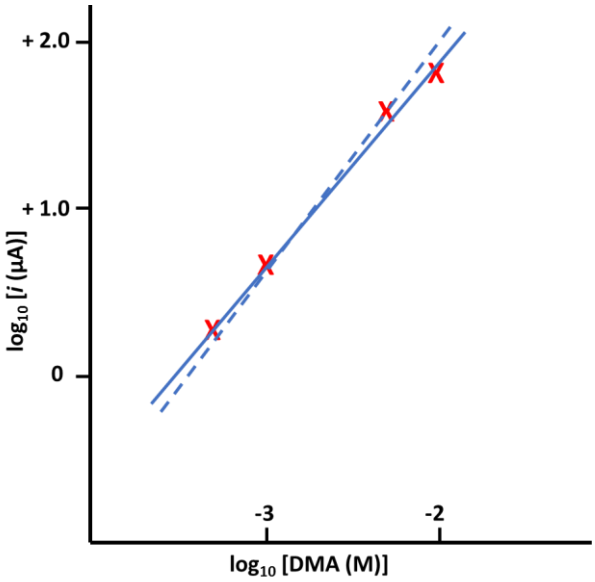


Figure 2- 3: Tafel plot adapted from (67). The two lines represent duplicate experimental results. Slope of dotted line 1.3 and solid line 1.1.

2.2.3. Cyclic Voltammetry Results

An adapted version of the cyclic voltammogram Mizoguchi and Adams' produced is shown in Figure 2-3.

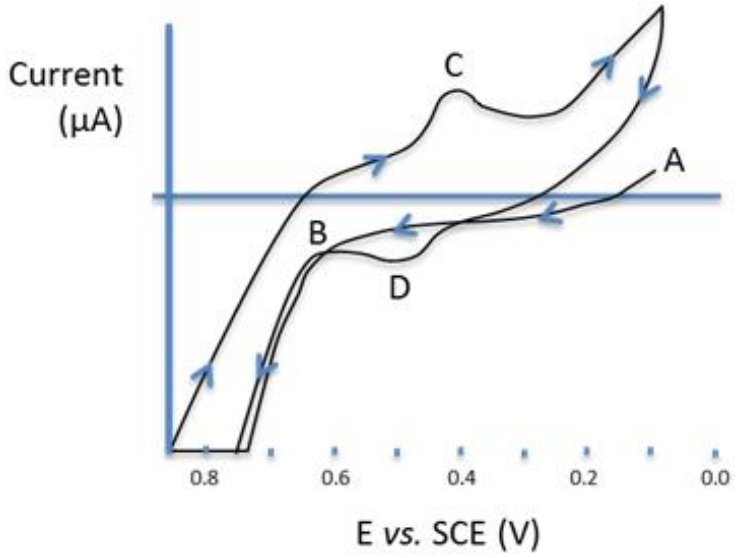


Figure 2- 4: CV results adapted from (67), labels discussed in main text.

From the starting point A in the CV in Figure 2-4 there is only residual current as this is starting material DMA that is present (Figure 2-5). At point B, the oxidation of DMA starts to occur at approximately 0.6 V *vs.* SCE. This

continues until the reverse scan starts at 0.90 V (shown in Figure 2- 6). A cathodic peak, shown as point C, was then observed at approximately 0.45 V. The reaction scheme for this is shown in Figure 2-7. After the CV reaches the switching potential a new peak is observed (point D). This is due to the oxidation of the TMB which was not present in the original scan. This redox couple (C/D) is almost completely reversible and when TMB alone is subject to CV, it displays these same waves (shown in Figure 2-8). Therefore, this is good evidence for the postulated follow-up reaction that was given in Figure 2-1.

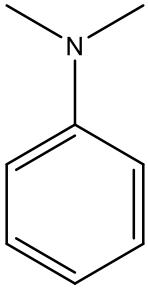


Figure 2- 5: Starting material, DMA, present at electrode at point A in figure 2- 4

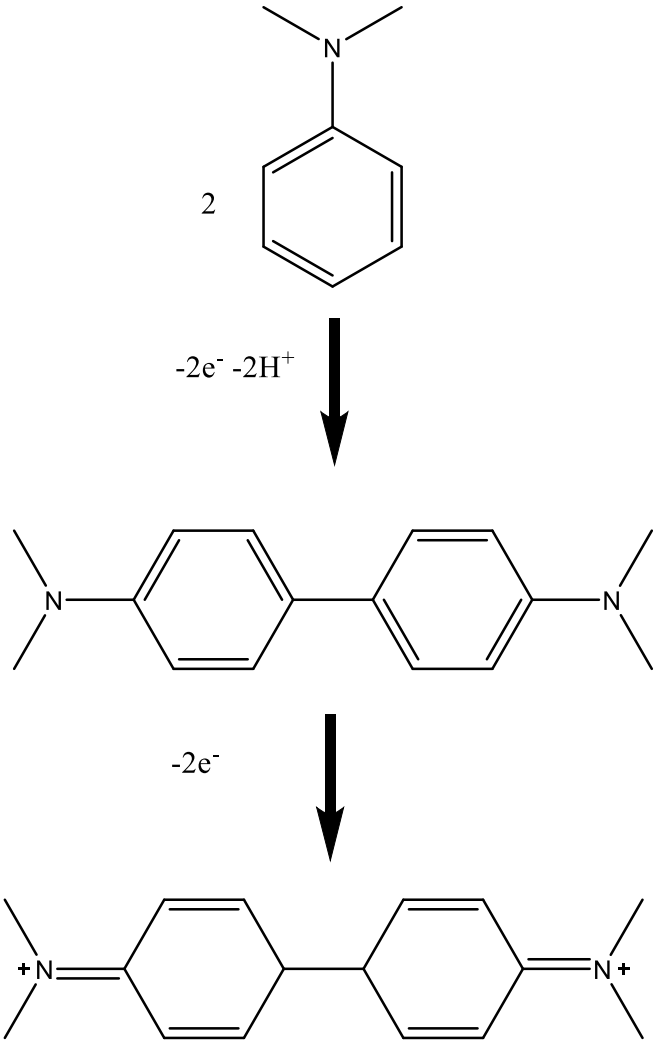


Figure 2- 6: Oxidation of DMA is the dominant reaction at electrode surface at point B in Figure 2- 4.

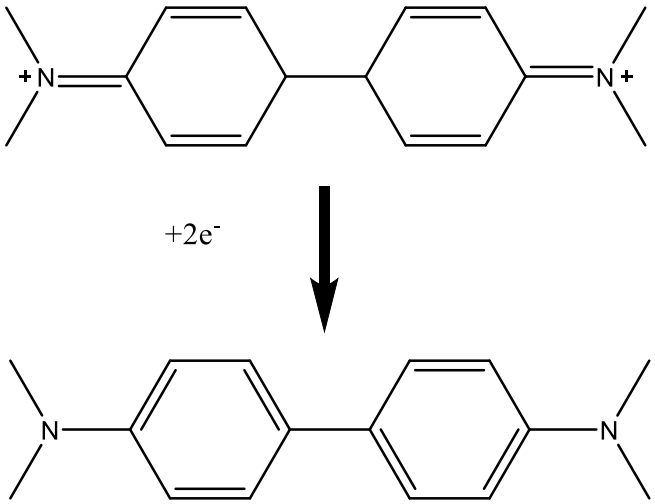


Figure 2- 7: Reduction of TMBox back to TMB dominant reaction at point C in Figure 2- 4.

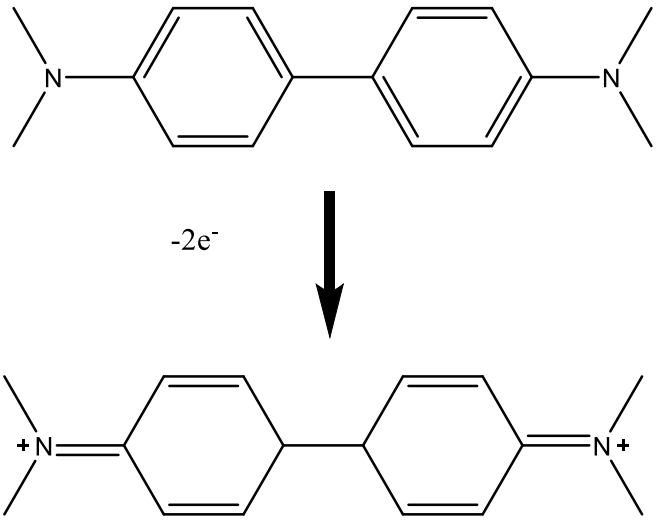


Figure 2- 8: Re-oxidation of TMB is the dominant reaction at the electrode at point D in Figure 2- 4.

2.2.4. Carbon Paste Electrode Cyclic Voltammetry

Further work by Galus and Adams (69) confirmed the mechanism described above for DMA oxidation using CV but at a carbon paste electrode (CPE). However, they also presented evidence of a further homogeneous reaction between DMB and a TMB oxidised radical. They also conducted kinetic studies at the CPE by varying the scan rate (υ). This work involves firstly

determining the product of βn_a under varying υ using Equation 2-2, where β is the anodic transfer coefficient and n_a the apparent number of electrons involved the transfer.

Equation 2-2: $$\left(E_P - E_{P/2}\right) = 1.85\frac{RT}{\beta n_a}$$

Once this is obtained the standard rate constant for the oxidation (k_b^0) can be calculated using a form of the Matsuda and Ayabe equation that has been transformed for irreversible reaction *i.e.* Equation 1:3, where D_{RED} is the diffusion coefficient of the reduced species and γ_{RED} is the activity coefficient of the reduced species.

Equation 2-3: $$E_{P(Irr)} = \frac{RT}{\beta n_a}\left[0.78 - ln\frac{k_b^0\gamma_{RED}}{\sqrt{D_{RED}}} + \frac{1}{2}ln\frac{\beta n_a F\upsilon}{RT}\right]$$

The value of βn_a they determined for the DMA oxidation was 1.06 and from the slope of log k_b^0 *vs.* 1/T the heat of activation was estimated as 175 kJ mol^{-1}.

2.3. Oxidation of Phenylenediamines

Chronopotentiometry was used to investigate several aromatic amines by Mark and Anson (70) at a platinum electrode. The structures of the compounds investigated are shown in Figure 2-9. Importantly they were able to show that below the pK_a of the amines, there was a rapid removal of the ammonium proton prior to oxidation. The rate of deprotonation was estimated to be 10^4 s^{-1} for benzene-1,4-diamine and N^1,N^1-dimethylbenzene-1,4-diamine. This is consistent with the DMA oxidation mechanism Mizoguchi and Adams (71).

Further voltammetric analysis by Lee and Adams (72) at CPE's confirmed some clear trends when oxidising amines.

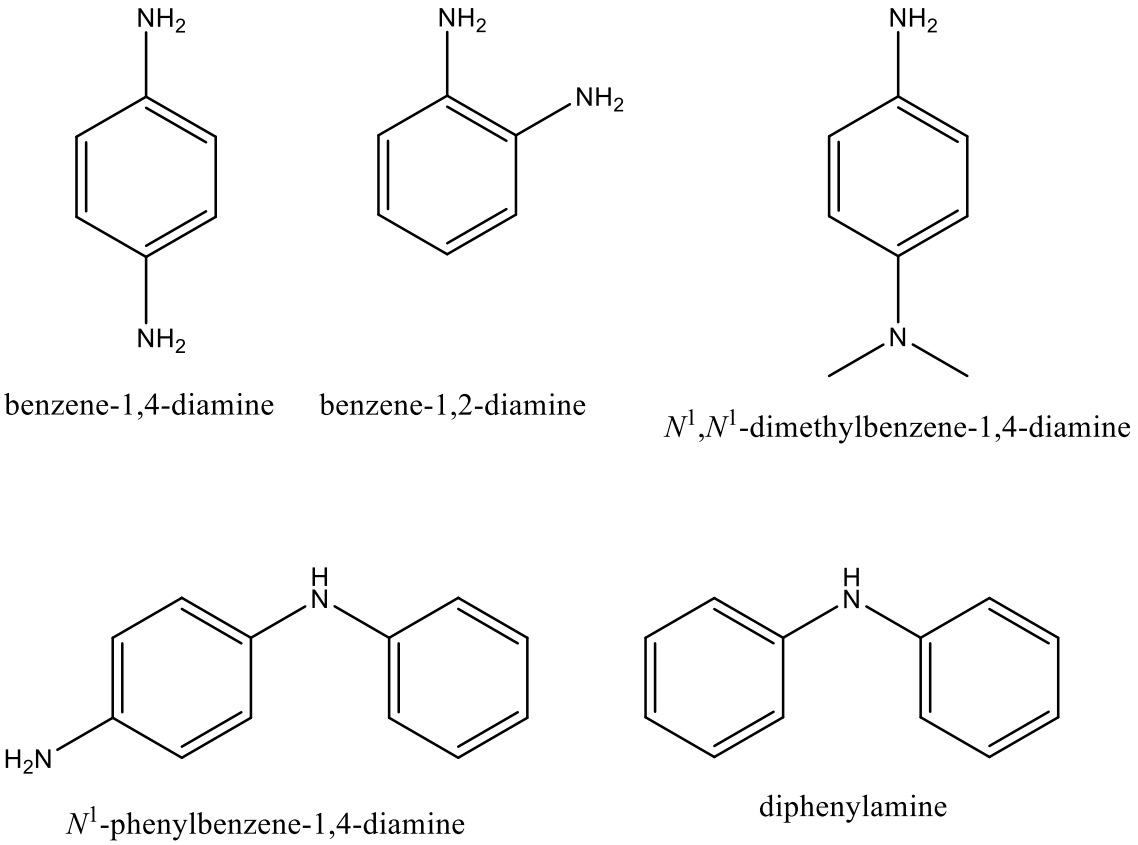


Figure 2- 9: Phenylenediamines investigated by chronopotentiometry

Importantly they were able to show that below the pK_a of the amines, there was a rapid removal of the ammonium proton prior to oxidation. The rate of deprotonation was estimated to be 10^4 s^{-1} for benzene-1,4-diamine and N^1,N^1-dimethylbenzene-1,4-diamine. This is consistent with the DMA oxidation mechanism Mizoguchi and Adams (67).

Further voltammetric analysis by Lee and Adams (73) at CPE's confirmed some clear trends when oxidising amines. Their results shown in Figure 2-10 demonstrate that as the pH increases, the $E_{P/2}$ decreases for all of the phenylenediamines. Otherwise stated, they become easier to oxidise at higher pH which accords physically with deprotonation prior to charge transfer. Another point of note is that it is considerably easier to oxidise the para and

ortho forms *cf.* the meta. The is easily explained by the lack of stable resonance forms in meta radical cation.

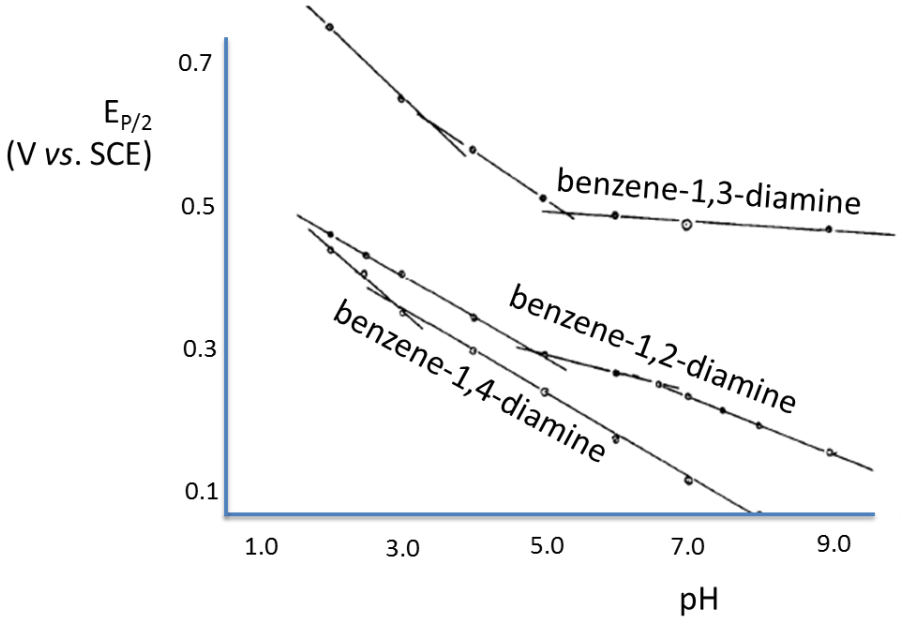


Figure 2- 10: Phenylenediamine oxidation at CPE with varying pH adapted from (73)

Lastly each of the trends have definite breaks where the slope changes. This roughly corresponds to the pK_a of each compound, for which literature values (74) are given in Table 2-1.

Table 2- 1 Phenylethylamine aqueous dissociation constants at 20 °C

	pK_a 1	**pK_a 2**
ortho	4.57	0.80
meta	5.11	2.50
para	6.31	2.97

4

2.4. Oxidation of Aliphatic Amines

In 1964 Mann demonstrated that the oxidation of aliphatic amines can be considered to be completely irreversible (18). This was shown through the oxidation of 19 different compounds at a platinum WE using 0.1 M $NaClO_4$ in acetonitrile as the electrolyte. An interesting, linear pattern emerged when the Hammett-Taft polar substituent constants of secondary and tertiary amines

were plotted against the peak potential. However, primary aliphatic amines did not follow the same pattern.

The pattern of irreversibility for aliphatic amine oxidation has also been confirmed at a glassy carbon WE (75). However, in this case no oxidation was observed at all for primary amines. Secondary amines showed a single anodic peak whilst tertiary amines showed two. The reason why there are two peaks for compounds such as trimethylamine is shown in the mechanism that was proposed in Figure 2-11. This scheme shows the final products are a secondary amine and aldehyde and that it is two electron process.

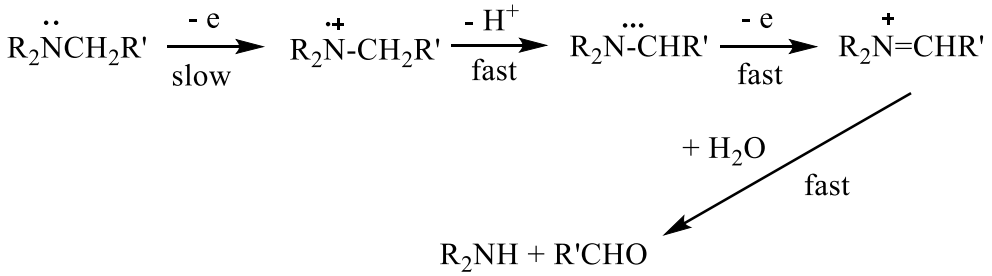


Figure 2- 11: Mechanism of oxidation of tertiary aliphatic amine (75) terminating with secondary amine and aldehyde

In the early 1970s, oxidative dealkylation of secondary and tertiary aliphatic amines was the focus of some research. It was suggested that anodic dealkylation can occur successively through the entire range of aliphatic amines *i.e.* tertiary ➔ secondary ➔ primary ➔ ammonia and finally to elemental nitrogen (76). The glassy carbon electrode was also shown to be suitable for the anodic dealkylation in aqueous alkaline media (77).

2.4.1. Oxidation of Tropanes

The oxidation mechanism of tropane was found to follow many complex stages (78) depending on the solvent environment. N-dealkylation was also observed, however there was no cleavage of the other two C–N bonds (see

Figure 2-12). Dimerization was also observed across the amine group both directly (*i.e.* N–N) and with intervening methylene groups.

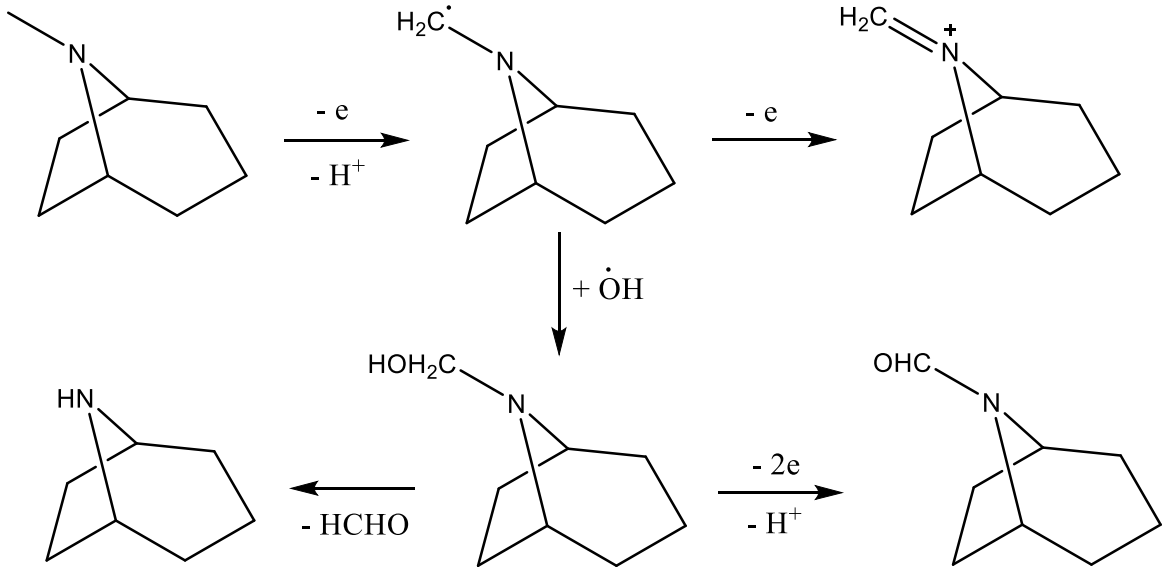


Figure 2- 12: Possible oxidation of tropane routes at a platinum WE

2.5. Oxidation of Aromatic Amines

Figure 2-13 outlines the oxidation mechanism of triphenylamine (TPA) as discovered by Seo *et al*. (79). The work was conducted by CV at a platinum WE using 0.1 M tetraethylammonium perchlorate in acetonitrile as the electrolyte. They demonstrated a reversible ECEE mechanism (as abbreviated by the Reinmuth notation (11)). The homogeneous chemical step involved a dimerization to form tetraphenylbenzidine (TPB). Electron paramagnetic resonance studies were also used to produce spectra of the cation radicals.

Nelson and Adams later built on this work by determining the values of the rate constants for the dimerization of a range of substituted triphenylamines (80). Here the chronoamperometric results of benzidine formation at a Pt WE was also confirmed by visible absorption spectroscopy.

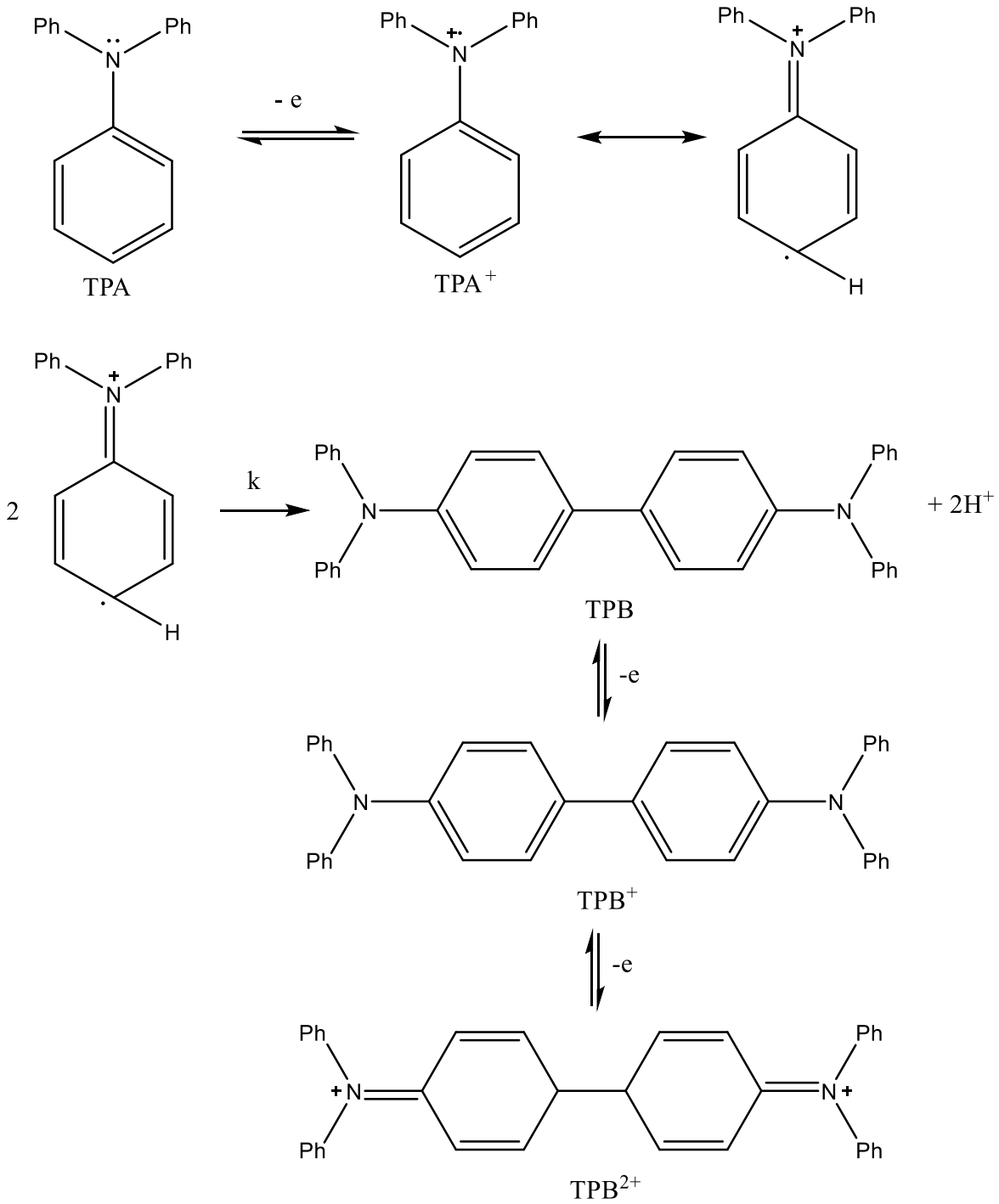


Figure 2- 13: Oxidation mechanism of TPA to TPB at Pt WE in 0.1 M $C_8H_{20}ClNO_4$ in CH_3CN

Unequivocal determination of the production of oxidation of diphenyl amines came a few years later (81). Through the interpretation of CV data supported by spectroscopy, benzoquinone and para-substituted aniline were shown to form upon the oxidation of N-phenyl substituted p- aminophenols in

acidic aqueous media. Figure 2-14 shows how this is due to reaction of the oxidation product with solvent.

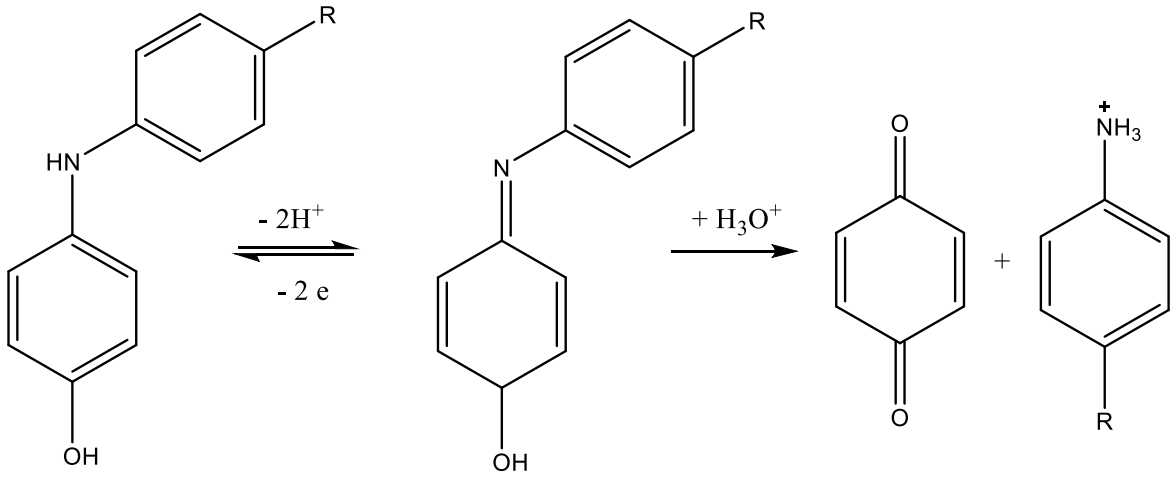


Figure 2- 14: Oxidation of N-phenyl substituted p-aminophenols in aqueous acid at a Pt WE

Subsequently Portis *et al.* were able to show how substitution of hydroxyl or methoxyl groups on the phenyl rings affected the anodic peak potential (82). They also drew analogies between enzymatic action, such as the conversion of dopamine to homovanillic acid (see Figure 2-15) and that of simple oxidation at a Pt anode. In both cases the C-N bond is broken followed by an increase in oxidation. However, in the physiological case the end product is the carboxylic acid, whereas in the electrochemical cell it would terminate as an aldehyde.

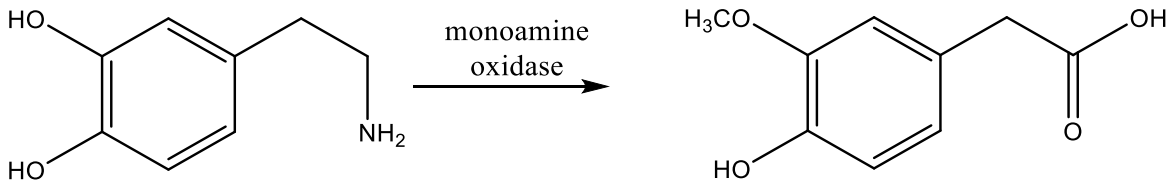


Figure 2- 15: Enzymatic oxidation of dopamine

A theoretical investigation of the oxidation of aniline using the quantum-chemical method Austen Model 1, was found to have good consistency with voltammetric data (83). This computational chemistry method showed how the benzidine formation, that had been empirically discovered decades earlier, was possible.

Oxidation of aromatic amines is also possible at a carbon electrode as demonstrated by Andrade *et al.* (230). Here they were able to discriminate between two analogues of a novel psychoactive substance 25B NBOMe and 25I NBOMe (shown in Figure 2-16). They were able to achieve a limit of detection 0.01 mg mL^{-1} in a pH 7 phosphate buffer at a screen-printed glassy carbon electrode.

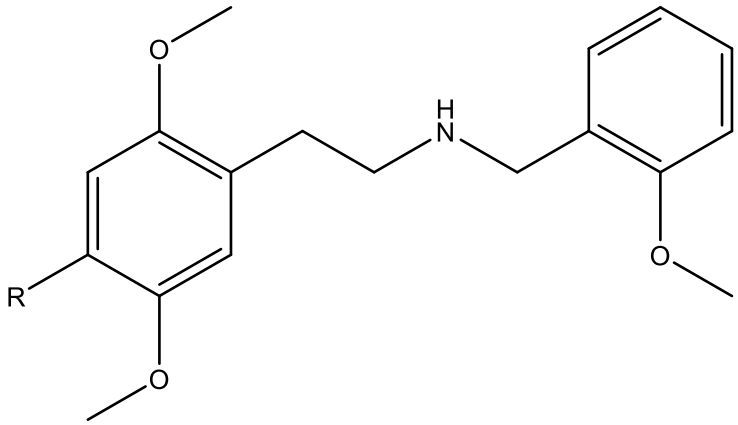


Figure 2- 16: Structure of phenethylamine derivatives R=Br for 25B NBOMe and R=I for 25I NBOMe.

2.6. Early Work in Forensic Electrochemistry

The oxidation mechanism for ecstasy type compounds was proposed by Garrido *et al.* (128). Whilst no specific mechanisms have been proposed in the literature, certain steps are strongly suspected. This begins with the removal of an electron from the aromatic ring, then very quick dimerization. It has also been shown empirically that the mechanism involves twice as many electrons as H^+ ions. Nevertheless, the electrochemical behaviour of amphetamine (84) and the ecstasy type drug methylenedioxyamphetamine (MDA) (85) was studied as far back as the 1990's.

The oxidative mechanism of MDA and three similar analogues was investigated using three different working electrodes: platinum, glassy carbon

and carbon paste. For all three electrodes an anodic peak was observed for all four compounds, although differential pulse voltammetry (DPV) was necessary to produce analytically viable peaks. A clear relationship between pH and current response was observed where there was sharp drop off above the pKa. However, highly linear correlations were observed for the increase in current with respect to concentration over the range 0.01 to 0.1 M (85).

A dropping mercury electrode was used to study the oxidation of amphetamine and several ecstasy type derivatives by CV and DPV. The difference between *in vivo* mechanisms and that in an electrochemical cell was noted. The effect of pH on the reduction potential of the Schiff bases formed upon oxidation (*cf.* Figures 2-2, 2-7, and 2-8) was monitored. Although not reported in their paper, the slope of the graph of pH vs peak potential on the CV (approximately 68 mV pH^{-1}) would indicate an equal number of protons and electrons are involved in the mechanism. Gas chromatography mass spectrometry was also used to verify the electrochemical results from the oxidation of derivatised amphetamines. The study concluded that the direct measurement of amphetamines in aqueous samples, *e.g.* in urine, was feasible (84).

2.7. Summary

Dynamic electrochemical research over the past fifty years has elucidated the mechanism of the oxidation of amines to a high degree of confidence. This has been demonstrated at a range of electrode surfaces and in a wide range of electrolyte media. It has been shown to be more than sensitive enough for the analysis of "bulk drug" alkaloids such as would be seized by the police or found in a clandestine drug laboratory.

Therefore, the next chapter will discuss the work conducted to assess the viability of the voltammetric analysis the novel psychoactive substance benzylpiperazine. This analysis had not been previously reported in the scientific literature so serves as a good test case to assess the applicability of the technique to a wider range of controlled substances.

3. The Voltammetric Analysis of Benzylpiperazine

3.1. Introduction

The abuse of ecstasy tablets came to prominence across Europe during the late 1980's at which time the major active ingredient was 3,4-methylenedioxymethylamphetamine (MDMA) (86). However, over the years clandestine laboratories have sought to circumvent the law by producing tablets containing compounds which were not under control. This has led to an enormous range of compounds being seized globally. In fact, the most recent World Drugs Report states 479 separate compounds were involved in seizures in 2016 (87). Benzylpiperazine (BZP) is one such compound, (structure shown in Figure 3-1).

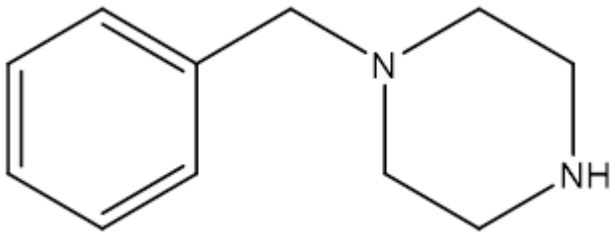


Figure 3- 1: Structure of benzylpiperazine

The importance of BZP in Europe was first noted in the early part of the 21st century as it was being sold as a "legal high" over the internet (88). There was also some confusion at the time with piperazines being sold as "herbal highs" although they are entirely synthetic. This may have been due to structural similarities with the pepper derived compound piperidine (89). Then in 2009 the European Monitoring Centre for Drugs and Drug Addiction (EMCDDA) advised that the member states should control BZP stating: "due to its stimulant properties, risk to health, the lack of medical benefits and following the precautionary principle, there is a need to control BZP, but the control

measures should be appropriate to the relatively low risks of the substance" (71). In the UK, BZP and structurally related analogues were brought under control of the Misuse of Drugs Act 1971, being listed as Class C in 2009 (90).

There already exists a wide array of chromatographic analytical methodologies in the scientific literature regarding the analysis of BZP. A brief synthesis of the analysis of BZP in a range of matrices is shown in Table 3-1. Typical limits of detection (LOD) tend to be in the nanomolar range, and this is indeed necessary for the analysis of clinical samples and wastewater, however it should also be noted that the methods are normally accompanied by a prior extraction step in order to attain LOD values in this range. It is also noteworthy that expensive, lab-based instrumentation is required for this type of analysis which makes it unsuitable for field testing of bulk drugs.

Consequently, a number of presumptive tests have been developed that can detect the presence of BZP. One such novel presumptive spot test was developed by Philp *et al.* (91) using sodium 1,2-naphthoquinone-4-sulphonate to produce a dark red colour for BZP. The proposed reaction scheme for this test is shown in Figure 3-2. They subjected the test to rigorous validation against many active ingredients and excipients commonly found in ecstasy-type seizures and were also able to determine an LOD of 40 µg, which is more than sensitive enough for its purpose. Piperidine has long been known to be a precursor for phencyclidine (92) and as such has been monitored closely. In a United Nations Scientific and Technical Note, the formation of a blue-coloured Simon-Awe complex is described, as shown in Figure 3-3 (93).

Table 3- 1 Chromatographic analytical methodologies for the analysis of BZP see Abbreviations section for explanations

Matrix	*Extraction and Derivatisation*	*Analytical Conditions*	*Limit of Detection*	*Reference*
Urine	Enzymatic hydrolysis with 100 mM acetate buffer containing sulfatase/ β-glucuronidase, then solid phase extraction with Oasis HLB	Liquid Chromatography: SCX column (150 mm x 2 mm); 40 mM pH4 Acetate Buffer:MeCN 25:75 v:v at 0.15 mL min^{-1}; Shimadzu LCMS 2010A mass analyser Gas Chromatography: DB-5MS column (30 m x 0.25 mm x 0.25 µm); He 1.0 mL min^{-1}; Shimadzu GCMS QP-2010 mass analyser	30 nM 300 nM	(94)
Tablets and capsules	10min ultrasonication in 20 mM HCl:MeOH 1:1 v:v	Liquid Chromatography: L-column ODS or SymmetryShield R_{18} (150 mm x 4.6 mm x 5 µm); 10mM SDS in MeCN:H_2O:H_3PO_4 300:700:1 v:v:v at 1mL min^{-1}; Diode array detector at 199-360nm Gas Chromatography: DB-5MS column (30m x 0.25mm x 0.25µm); He 1.1 mL min^{-1}; Agilent N3520 mass analyser	Not given Not given	(95)
Plasma and urine	Centrifugation then deproteinized with 35% $ZnSO_4$ plus enzymatic hydrolysis of urine	Liquid Chromatography: Zorbax C18 column (150 mm x 4.6 mm x 5 µm); gradient using 0.01M pH4.5 NH_4CHOO and MeCN at 1mL min^{-1} ; Agilent MSD model D single stage quadrupole mass analyser	30 nM	(96)
Plasma	Dilution with pH6 Phosphate buffer then solid phase extraction using Chromabond Drug	Liquid Chromatography: Synergi Polar RP column (150 mm x 2 mm x 4 µm); gradient using 0.1% CHOOH in 1mM NH_4CHOO and 0.1% CHOOH in MeOH at 0.25 mL min^{-1}; Sciex API 365 tandem mass analyser	30 nM	(97)
Simulated Wastewater	Filtration the SPE using XRDAH506	Liquid Chromatography: Luna pentafluorophenyl column (50 mm x 4.6 mm x 3 µm); gradient using MeOH and 0.1% CHOOH at 0.5 mL min^{-1}; AB Sciex Q-Trap mass analyser	6 pM	(98)
Simulated tablets	Dilution only for liquid chromatography and heptafluorobutyric acid derivatisation and silylation using SilPrep for gas chromatography	Liquid Chromatography: Hypersil C18 column (125 mm x 3 mm x 3 µm); gradient using pH 3.2 PB and MeCN at 0.4 mL min^{-1}; DAD at 210 to 400 nm Gas Chromatography: heptafluorobutyric acid derivatisation and silylation using SilPrep®; HP-5MS column (30m x 0.25mm x 0.25µm); He 1.0 mL min^{-1}; Agilent 5971A mass analyser	50 µM 3 µM	(99)

Urine	Liquid/liquid extraction using KOH	Gas Chromatography: pentafluoropropionic anhydride derivatisation; J&W column (20 m x 0.18 mm x 0.18 µm); He 1.0 mL min^{-1}; Agilent 5975 mass analyser	150 nM	(100)
Urine	Centrifugation then solid phase extraction using SOLA SCX	Liquid Chromatography: Accucore C18 (100 mm × 2.1 mm x 2.6 µm); gradient elution using 0.1% HCOOH in water and 0.1% HCOOH in MeCN at 0.4 mL min^{-1}; Thermo Scientific NCS-3500RS UltiMate 3000 Binary Rapid system coupled to a Thermo Scientific Q Exactive Mass spectrometer	6 nM	(101)
Hair	Overnight sonication in 0.1% HCOOH	Liquid Chromatography: Kinetex C18 column (100 × 2.1 mm x 2.6 µm); gradient elution using 0.1% HCOOH in 5mM NH_4HCOO and 0.1% HCOOH in 1:1 v:v MeOH: MeCN at 0.35 mL min^{-1} ; Agilent 6460 triple quadrupole mass spectrometer	5 pg/mg of hair	(102)
Tablets	Dissolution in water then freeze dried overnight	Liquid Chromatography: Synergi Hydro-RP Phenomenex column (250 mm × 10 mm); gradient elution using 0.05% CF_3COOH in water and MeCN at 3 mL min^{-1}; UV detection at 208 nm Gas Chromatography: DB1-ms column (15 m x 0.25 mm x 0.25 pm *sic*); He 2 mL min^{-1}; Agilent 5975C MSD Series with a Triple-Axis Detector	Not given Not given	(103)
Tablets	Ultrasonication in 2-methyl-propan-2-ol then centrifugation	Gas Chromatography: Supelco Equity 5 column (30 m x 0.25 mm x 0.25 µm); He 1 mL min^{-1}; Perkin Elmer Clarus Turbomass Gold 500MS detector.	2 nM	(104)
Wastewater	Solid phase extraction using Oasis MCX.	Gas Chromatography: Pentafluoropropionic anhydride derivatisation; Supelco Equity TM-5 column (30 m x 0.25 mm x 0.25 µm); He 1 mL min^{-1}; PerkinElmer Clarus 500 Gas Chromatograph-Mass Spectrometer detector	0.14 pg on column	(105)

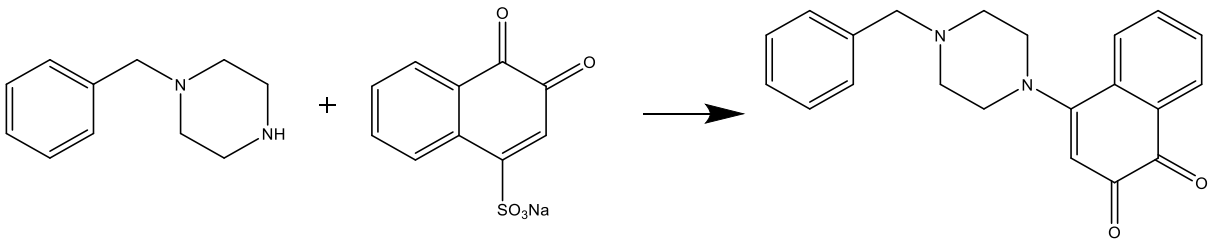


Figure 3- 2: Reaction scheme for the BZP presumptive colour test which uses 1,2-naphthoquinone-4-sulfonic acid sodium salt (91).

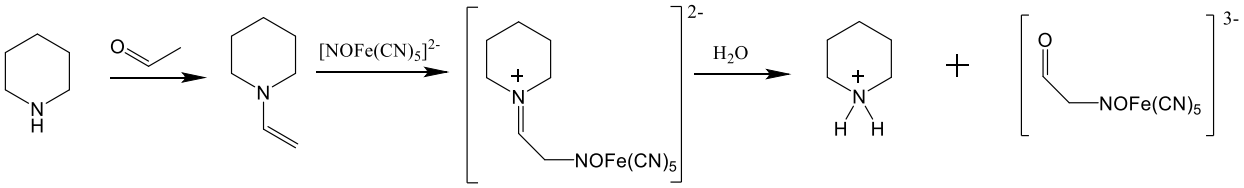


Figure 3- 3: Reaction scheme for the United Nations piperidine colour test (93).

Almost all piperazines contain the same active moiety as piperidine and as such this general test for secondary amines could be applied. It is in fact the amine functionality that enables colorimetric detection for all the common presumptive tests (88).

Microcrystalline identification does not offer as low an LOD as the colorimetric spot tests, but the growing habits that are observed between reagents and drugs can be very specific (106). Elie *et al.* (107) designed a microcrystalline assay for BZP using mercury chloride as the reagent. Upon gentle mechanical assistance for nucleation, BZP was found to form distinctive rectangular plates.

BZP is known to produce psychomotor effects similar to amphetamine (108). Notwithstanding BZP acting on the same receptors as amphetamine, it may not bind to the same antibodies as those used in amphetamine immunoassays. The response of several piperazines was checked against an enzyme-multiple immunoassay technique (EMIT) and a fluorescence polarisation immunoassay (FPIA) initially designed for amphetamine and methylamphetamine (109). The FPIA did not detect BZP in a 100,000 ng mL^{-1} spiked urine sample. However, the amphetamine EMIT did respond to BZP with

cross reactivities of 0.4% and 1.3% at 300 and 12,000 ng mL^{-1} amphetamine equivalents respectively. There are currently no commercially available immunoassays specifically for piperazines (88), however some recent literature has been published to this effect (110). Also, in general with presumptive testing false positives are possible (111) and they are at best semi-quantitative (112).

There can be no doubt as to the general reliability of electrochemical measurements in light of the facts that: globally amperometric quantitation of glucose is relied upon by millions of diabetes patients (113), there is widespread use of fuel-cell breath-alcohol testers by police forces (114) amongst many other medical applications (115). The major advantages of electrochemical analysis over other techniques can be summarised as:

- Miniaturisation enables portability and the analysis of samples of very small volume (116).
- Through proper electrode design a high level of sensitivity, selectivity and stability can be achieved (117).
- The analysis is generally faster and simpler than other techniques (118).
- It is far cheaper than most other techniques with comparable limits of quantification (LOQ) (119).

These benefits could of course be useful in field testing in a forensic context and indeed the relatively new field of "forensic electrochemistry" has found a range of applications in recent years (120) including the voltammetric analysis of explosives and gunshot residue using an innovative "lab-on-a-finger" technique (121). However, the bulk of the published forensic electrochemistry research lies in the area of drug analysis.

Table 3-2 highlights the breadth of scientific literature that already exists specifically pertaining to dynamic electrochemical analysis of drugs of abuse. Drugs from both natural and entirely synthetic sources commonly have LOD values in the low ng mL^{-1} range. However, due to the burgeoning supply of novel psychoactive substances, there are many compounds which have yet to be investigated in this way. It was the aim of this research to assess the viability of voltammetric analysis of BZP as the technique lends itself so well to potential field use. This research comprises the first time the voltammetric analysis of BZP has been reported.

3.2. Experimental

3.2.1. Materials

All voltammetric analysis was carried out using a PGStat128N potentiostat in combination with the NOVA software v1.11 (both Metrohm Autolab). An Ag|AgCl reference electrode and a Pt sheet counter electrode were used for all analysis (both Metrohm Autolab). A 2 mm diameter graphite electrode (Metrohm Autolab), a 2 mm diameter gold and a 2mm diameter platinum electrode (both BASi) were cleaned with an aqueous slurry using 15 μm alumina (Microabrasives Corp.) prior to use as the solid working electrodes for the initial investigation. Three carbon powders of different particle sizes (all Aldrich) and Nujol (Plough UK) were used to form the paste electrodes. The respective ratios of carbon and nujol were triturated for 20 min in a pestle and mortar prior to being housed in a Teflon electrode (BASi) with a 3 mm exposed area. A Zeiss EVO LS scanning electron microscope (SEM) was used to capture images of the carbon powders.

Table 3- 2: Voltammetric analysis of Drugs of Abuse see Abbreviations section for explanations

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
Morphine and noscapine (plasma)	Centrifuged and vortexed with 0.7 M $ZnSO_4$, bicarbonate and THF	HMDE (AdSDPV)	BR pH 10; deposition at -100 mV vs. Ag | AgCl for 150 s, then stripped from 100 to 400 mV at 40 mVs^{-1} with amplitude 100 mV	morphine 3 noscapine 7	(122)
Morphine (urine)	Dilution only	GCE (CV)	50 mM PB at pH 7.4; stirred accumulation at OCP for 90s, then CV from 0 to 700 mV vs. Ag | AgCl at 100 mV s^{-1}	57	(123)
MDMA (standards only)	–	GCE (CV)	50 mM PB at pH 7.4; scanned 1000 to − 500 mV vs. SCE	–	(124)
Cocaine (powder)	Dilution only	PDE modified with cobalt hexacyanoferrate (LSV)	0.1 M $NaClO_4$ in MeCN; 0 to 1200 mV at 100 mV s^{-1} vs. Ag | AgCl	43	(125)
Zopiclone (urine and Imovane® tablets)	Tablets were crushed and centrifuged in H_2O	GCE (AdSSWV)	BR pH 7.08; deposition at 600 mV vs. Ag | AgCl for 120 s, then stripped from 200 to 1400 mV with amplitude 50 mV, frequency 50 Hz and step 10 mV.	66	(126)
Amphetamine, MEA, MDA, MDMA (serum and tablets)	Tablets sonicated for 5 min in H_2O and filtered. MeOH added to serum then	GCE (SWV)	0.2 M PB pH 7; scan 0 to 1500 mV vs. Ag | AgCl with frequency 100 Hz and pulse amplitude 50 mV	MDMA 464	(128)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
	vortexed and centrifuged				
Morphine (standards only)	–	ITO modified with poly(3,4-ethyleenedioxythiophene) and further activated with IrO2 (LSV)	0.5 M sulphate buffer pH 2; scan 0 to 1300 mV vs. Ag | AgCl at 50 m V s^{-1}	–	(129)
Cocaine (standards only)	–	ITO modified by SAM of ferrocene and Au nanoparticles and activated with cocaine aptamer fragment SH-C2 (DPV)	0.1 M PB pH 6.5; scan 700 to 0 mV vs. Ag | AgCl with modulation time 50 ms, interval time 0.5 s, modulation amplitude 25 mV and potential step 5 mV.	30	(130)
Morphine (urine)	Centrifuged and diluted with PB pH 8.0	MCPE using graphite: MWCNT 4:1 w:w in paraffin oil with 1-butyl-3-methylimidazolium hexafluoro phosphate binder (DPV)	0.1 M PB pH 8.0; scan 100 to 600 mV vs. Ag | AgCl with pulse height 100 mV and pulse width 5 mV.	40	(131)
Morphine (serum)	Dilution only	GCE modified with Au nanotube array on an anodic aluminium oxide template (DPV)	0.1 M phosphate and citric acid buffer pH 6.1; other parameters not specified	12	(132)
Cocaine (serum)	Dilution with 10 mM PB pH 7.4 and 1 M NaCl	Au electrode modified by tetrahedron SAM of thiolated aptamers (Chronoamperometry)	Cocaine fuses the aptamers aca-1 and biotinylated aca-2. This is followed by the binding of avidin-HRP conjugates to aca-2's biotin tag. As HRP can now act as a catalyst for H_2O_2 reduction in the presence of	10	(133)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
			3,3′,5,5′-tetramethylbenzidine an amperometric signal is produced		
Codeine (tablets)	Crushed and dissolved in 0.2 M AB pH 4.5	GCE (SWV)	0.2 M AB pH 4.5; scan 0 to 1800 mV vs. Ag | AgCl with frequency 75 Hz, amplitude 50 mV and increment 5 mV	761	(134)
Morphine (urine)	Dilution only	MCPE using electrodeposition of ferrocene and Au nanoparticles (DPV)	0.04 M BR pH 7.4; scan 100 to 600 mV vs. Ag | AgCl with scan rate 10 mV s^{-1} pulse width 25 ms, pulse period 200 ms and pulse amplitude 10 mV	1	(135)
Morphine (urine)	Centrifuged and diluted with PB pH 7.0	MCPE using graphite:MWCNT: n-hexyl-3-methylimidazolium hexafluorophosphate 64.5:15.5:20 w:w:w in paraffin oil (DPV)	0.1 M PB pH 7.0; scan 300 to 600 mV vs. Ag | AgCl with pulse height 80 mV and pulse width 7 mV	6	(136)
THC (hemp and hashish)	10 min US in MeOH then filtered. Filtrate then subject to preparative TLC	GCE (LSV)	0.1 M tetrabutylammonium tetrafluoroborate in *N*,*N*-dimethylformamide:H_2O 9:1 v:v; 30s deposition at − 1200 mV vs. Ag | AgCl then scan − 1000 to 500 mV at 100 mV s^{-1}	0.34	(137)
Morphine (urine and	Urine was filtered prior to dilution in	MCPE using graphite:MWCNT:4-hydroxy-2-(triphenylphosphino)phenolate	0.15 M PB pH 7.0; other parameters not specified	19	(50)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
pharmaceutical formulations)	0.15 PB pH 7.0	200:2:1 w:w:w in paraffin oil (DPV)			
Morphine, noscapine and diamorphine (standards only)	–	GCE modified with graphene nanosheets (DPV)	0.1 M PB pH 8.0; scan from 100 to 1100 mV vs. Ag | AgCl with voltage step 8 mV, scan rate 20 mV s^{-1} and pulse amplitude 50 mV	morphine 114 noscapine 83 diamorphine 185	(138)
Morphine (standards only)	–	Au electrode by SAM of 2-aminoethanethiol (CV)	0.2 M PB pH 6.0; scan 0 to 800 mV vs. Ag | AgCl at a range of scan rates	–	(139)
Morphine (urine)	Dilution only	MCPE using graphite:Al_2O_3 nanoparticles 9:1 w:w in paraffin oil (SWV)	0.1 M PB pH 7.0; other parameters not specified	9	(140)
GHB (standards only)	–	Pt (CV)	0.1 M $HClO_4$ and 0.1 M H_3PO_4; scan − 200 to 1200 mV vs. Ag | AgCl at 10 mV s^{-1}	–	(141)
Codeine (urine)	Centrifuged and diluted with PB pH 4.0	GCE modified with graphene and Nafion (SWV)	0.1 M PBS pH 4.0; stirred accumulation at OCP for 180 s then scan 700 to 1300 mV using amplitude 50 mV, step height 6 mV and frequency 12 Hz	5	(142)
Cocaine (standards only)	–	Pt electrode modified with [UO_2(*N*,*N*′′-Ethylenebis(3-methoxysalicylideneaminato))(H_2O)]·H_2O (CV)	1 M KCl and 0.001 M HCl; scan − 300 to 400 mV vs. Ag | AgCl at 100 mV s^{-1}	21	(143)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
Buprenorphine (urine)	Dilution only	CPE (DPV)	0.24 mM SDS in 0.2 M PB pH 3.0; stirred accumulation at OCP for 120 s then scan 300 to 1000 mV with pulse height 10 mV and scan rate 90 mV s^{-1}	7	(144)
Morphine (urine and pharmaceutical samples)	Centrifuged and filtered prior to dilution	MCPE using graphite: NiO modified MWCNT: 1-methyl-3-butylimidazolium chloride 70:35:25 w:w:w in paraffin oil (SWV)	0.1 M PB pH 7.0; other parameters not specified	3	(145)
Morphine (serum and urine)	Dilution only	GCE modified with electrochemically reduced MWCNT graphene oxide (LSV)	0.1 M PB pH 4.5; stirred accumulation at OCP for 120 s then scan 200 to 900 mV at 100 mV s^{-1}	14	(146)
Morphine (urine)	Supported liquid membrane extraction using 2-nitrophenyl octyl ether containing 10% tris-(2-ethylhexyl) phosphate and 10% di-(2-ethylhexyl) phosphate	Carbon SPE (DPV)	0.1 M NaOH; scan 0 to 600 mV vs. Ag pseudoreference with step 15 mV and pulse amplitude 50 mV	2	(147)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
Methadone (urine and saliva)	MeCN was added to urine then all samples were vortexed and centrifuged	MCPE using graphite:MWCNT 65:10 w:w then electrodeposition of Au nanoparticles (AdSSWV)	0.04 M BR pH 9.0; deposition at 700 mV vs. Ag | AgCl for 100 s then scan 500 to 1250 mV with pulse amplitude 100 mV and frequency 25 Hz	5	(148)
Cocaine (powder)	Dilution only	Carbon SPE modified with MWCNT (SWV)	0.1 M PBS pH 10; scan 0 to 1500 mV vs Ag | AgCl with step potential 12 mV, frequency 25 Hz and amplitude 25 mV	–	(149)
MDMA (Standards only)	–	GCE modified with Nafion and cucurbit[6]uril (CV)	KCl solution; scan 800 to 1300 mV at 100 mV s^{-1}	3	(150)
Methcathinone, mephedrone and 4-MEC (standards only)	–	Carbon SPE (CV)	PB pH 2; scan 0 to 1700 mV vs. SCE at 100 mV s^{-1}	methcathinone 44500 mephedrone 39800 4-MEC 84200	(151)
Morphine (serum)	MeCN was added to serum then samples were vortexed and centrifuged	GCE modified with Nafion and MWCNT (AdSSWV)	0.1 M H_2SO_4; deposition at -500 mV vs. Ag | AgCl for 360 s then scan 700 to 1400 mV with potential step 8 mV, frequency 75 Hz and amplitude 25 mV	9	(152)
Morphine and codeine	Urine was centrifuged and filtered.	Pencil graphite electrode modified with PDDA and	0.1 M PB pH 7.0; stirred accumulation at OCP for 300 s then scan from 0 to 900 mV with	morphine 41 codeine 43	(153)

Analyte/s and (matrix)	Extraction	WE and (technique)	Analytical conditions	LOD (ng mL^{-1})	Ref
(serum and urine)	Serum had 20% v:v $HClO_4$ added, vortexed then centrifuged	aptamers (AdSDPV)	pulse amplitude 50 mV, modulation time 0.05 s and step potential 8 mV		
Cocaine (serum)	Ultrafiltration by centrifuge	GCE with Ag NP and aptamer (DPV)	1 mM Riboflavin and 1 mM NaCl; -0.2 to -0.6V; modulation range of 25 mV and time of 50 ms	0.045	(154)
NBOMe (blotter paper)	Ultrasonication in 1/1 MeOH/H_20	GCE (SWV)	0.1 M $LiClO_4$ in MeOH; 30 s preconcentration at 1.8 V; step 5 mV, amplitude 20 mV, frequency 25 Hz	0.55	(155)
Ethylone (tablets)	Ultrasonication in MeOH	Boron Doped Diamond (SWV)	0.5 M H_2SO_4; step 23 mV, amplitude 42 mV, frequency 81 Hz	2.7	(156)
Xylazine (standards)	N/a	GCE (DPV)	0.1 M BR buffer; step 10 mV, amplitude 70 mV	26	(157)

The chemicals used in the preparation of the background electrolytes were all laboratory reagent grade (Fischer Scientific) and used without any further purification. Background electrolytes were thoroughly degassed by nitrogen bubbling. The potassium ferricyanide was a laboratory reagent of 99% purity. A liquid BZP standard of purity ≥ 90% (Bione) was used for the analysis with the solid electrodes and a BZP hydrochloride salt, ≥98% (Cayman Chemicals) was used in the analysis using the paste electrodes. A 3,4-methylenedioxymethylamphetamine (MDMA) hydrochloride salt standard of purity ≥ 98% (Sigma) was used to assess the selectivity of the method. All drug concentrations are reported as base rather than as the salt. Deionised water with a resistivity of 18.2 MΩ cm was used to prepare the solutions. The software Minitab v16 (Minitab Inc.) was used to produce and analyse the factorial design experiment.

3.2.2. Initial Investigation

A 10 mL cell volume of approximately 50 μM BZP in 0.1 M KCl was prepared and cyclic voltammograms (CVs) were run using the solid graphite, gold and platinum electrodes and each was checked against an appropriate blank. Five scans in total were taken for each cell using a scan rate (υ) of 250 mV s^{-1} and a step height of 2.4 mV.

3.2.3. Carbon Paste Electrode Development

SEM images of the three carbon powders were captured using the secondary electron detector under high vacuum with a working distance of 10 mm and an anode potential of 25 kV. The three different forms of carbon powder were mixed with nujol at two different weight to weight ratios as shown in

Table 3-3. A 10 mM $K_3Fe(CN)_6$ solution was prepared in 0.1 M KCl as a spiking solution for standard addition analysis by square wave voltammetry (SWV). The concentration range 99 to 909 µM of $K_3Fe(CN)_6$ in 0.1 M KCl was tested for each paste electrode using the voltage program deposition at -0.5 V for 30 s then stripping up to 1.0 V with a step of 5 mV, amplitude of 20 mV and a frequency of 25 Hz. Three CVs were also obtained for the 99 µM of $K_3Fe(CN)_6$ cell using three scan rates: 10, 100 and 500 mV s^{-1}. In each CV the working electrode potential started at 0.0 V and was swept to 1.0 V then cycled between -0.7 V and 1.0 V a total of 3 times with a step of 2.4 mV.

Table 3- 3: Ratios of carbon to nujol used in assessment of paste electrodes

Paste Number	Allotrope of Carbon	Particle size (µm)	Carbon (%W/W)	Nujol (%W/W)
1	Graphite	< 45	60	40
2	Graphite	< 45	80	20
3	Graphite	< 150	60	40
4	Graphite	< 150	80	20
5	Glassy carbon	2 – 12	60	40
6	Glassy carbon	2 – 12	80	20

3.2.4. Mechanism Investigation

Seven separate Britton-Robinson (BR) buffers at 40 mM were prepared to cover the pH range 4 to 10. These were used as the background electrolyte for 49 µM BZP cells. Each pH cell was investigated by linear sweep voltammetry (LSV) using the Paste 6 working electrode from -0.8 V to 1.0 V with a scan rate of 500 mV s^{-1} and step of 2.4 mV. A further BR buffer at pH 9.5 was also prepared as the background electrolyte for a 67 µm BZP solution in order to assess the LSV over the same potential range but with varying scan rates using the Paste 6 electrode.

3.2.5. Method Development

A 100 µM BZP was prepared in pH 9.5 BR buffer and analysed by SWV using the Paste 6 electrode. Increasing levels of deposition time were tested in order to assess when electrode saturation occurred. Otherwise the working electrode was swept from – 0.8 V to 1.0 V with a step of 5 mV, and amplitude of 20 mV and a frequency of 25 Hz. Once the best deposition time was determined the SWV was further optimised with the same cell using a factorial design as shown in Table 3-4.

Table 3- 4: Factorial design for optimisation of square wave voltammetry of BZP using Paste 6

Run Order	Step (mV)	Amplitude (mV)	Frequency (Hz)
1	10	5	5
2	1	5	50
3	10	5	50
4	1	50	5
5	10	50	50
6	1	5	5
7	1	50	50
8	10	50	5

3.2.6. Method Validation

The optimised SWV method was used to analyse BZP solutions over the concentration range 10 to 60 µM BZP in pH 9.5 BR buffer using the Paste 6 electrode. 500 mL of Pepsi Cola® was degassed by sonication for 20 min then 20 min nitrogen bubbling. 10.8 mg of BZP hydrochloride was dissolved in 100 mL of the cola and analysed in triplicate using the optimised SWV method. A 3 µM MDMA solution in pH 9.5 BR buffer was analysed by the optimised SWV

method in order to assess the selectivity of the method against the most commonly encountered ecstasy-type compound.

3.3. Results and Discussion

3.3.1. Electrode Material Determination

The three CVs for the initial scan of BZP are shown in Figure 3-4. Platinum did not show any response to BZP that was not also present in the blank CV so was thus discarded from further investigation. The gold electrode showed a reversible peak which was also present in the blank considered to be formation and removal of an oxide layer.

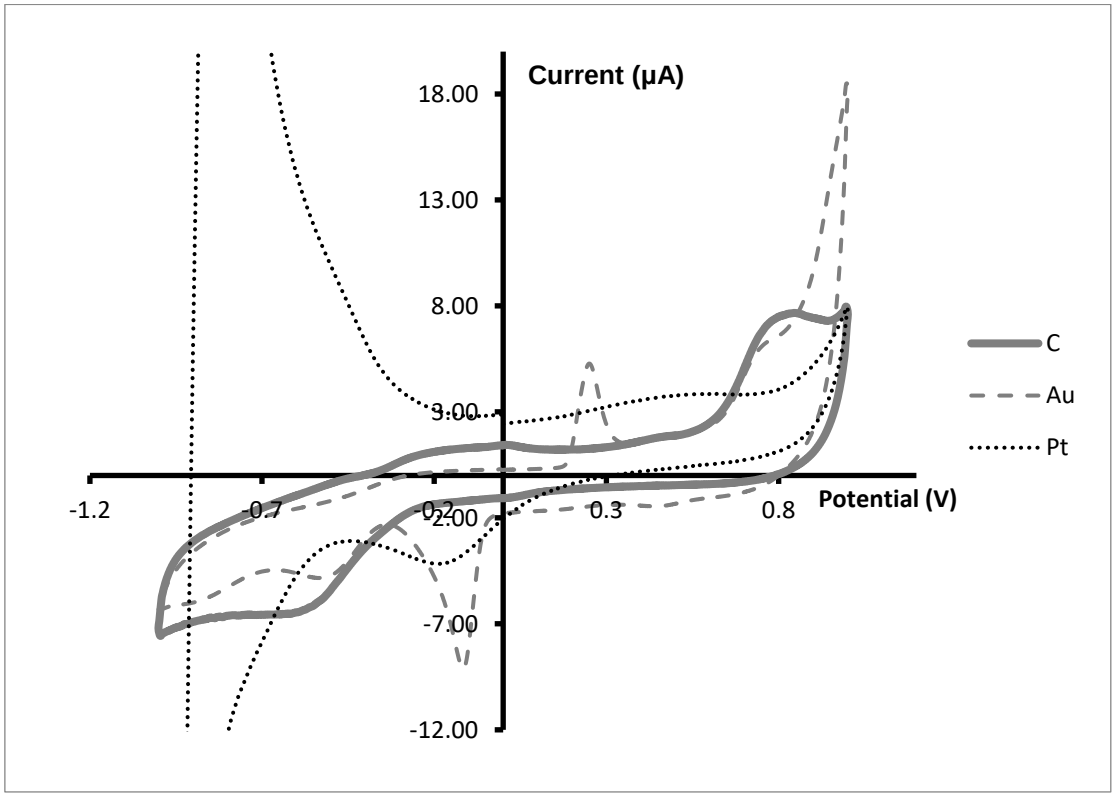


Figure 3- 4: CV of 50 µM BZP using graphite (C), gold (Au) and platinum (Pt) electrodes. The second scan of five is shown in each case and platinum is shown as off scale for clarity of the important oxidative peaks. Scan rate 100 mV s^{-1}.

However, at approximately 0.8 V a non-reversible oxidative peak was present which corresponded to BZP. This peak was also present in the graphite electrode CV. Due to the response to graphite for BZP being highest of those tested and the general wide applicability of carbon electrodes, it was decided to pursue carbon as the electrode material.

The SEM images of the types of carbon that were investigated are shown in Figure 5. The two graphite powders had highly irregular morphology but generally were within the particle size listed by the manufacturer (as estimated by the SEM's sizing capability). However, the glassy carbon particles were much more regular being almost entirely spherical and again falling within the manufacturer specified size. The six different carbon paste electrodes were then evaluated using $K_3Fe(CN)_6$ as a model compound as it is known to have excellent reversibility characteristics.

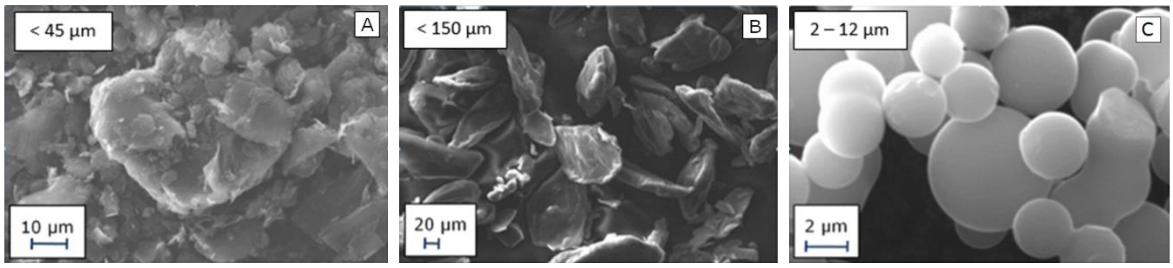


Figure 3- 5: SEM micrographs of carbon powder. The particle size is shown in the top left of each image and the scale is shown in the bottom left. Image A and B are graphite and image C is glassy carbon. The granular appearance of the graphite it notable in comparison with spherical shape of the glassy carbon beads.

It was immediately discovered that Paste 5 failed to produce any signal which is assumed to be due to lack of conductivity. The CVs for the five remaining paste electrodes are shown in Figure 3-6 alongside the Randles-Sevčik – plots of peak current (I_P) versus the square root of the scan rate – for each paste. The most striking feature from the CVs is that Paste 6 shows by far the most reversible characteristics; the peak to peak separations for Pastes 1 to 4 are all greater than 100 mV even at the slowest scan rate whereas ideally for a reversible single electron transfer such as in the ferricyanide ion 60 mV would be expected. Indeed Paste 6 is the only one to have the cathodic wave in the positive potential region, although it also has the highest background capacitance shown by the vertical distance between the forward and reverse waves. The cathodic Randles-Sevčik plot for Paste 4 was not determined because the peak potential E_{PC} was not distinctly formed against the background wave. The CV data for the five functional pastes are summarised in Table 3-5.

As the peak to peak separation (ΔE_P) for Paste 6 is clearly the lowest and the ratio of the slopes for Paste 6's cathodic and anodic Randles- Sevčik plots is close to 1 it has the best reversibility characteristics. The sensitivity of each of the pastes was also investigated by SWV and the voltammograms alongside the regression curves are shown in Figure 3-7 (where the δ symbol indicates the current differential at each voltage step).

Table 3- 5: Summary of assessment data for $K_3Fe(CN)_6$ in 0.1 M KCl for the five functional paste electrodes. The CV data in the columns two and three was measured at 99 μM.

Paste	CV ΔE_p at 100 mV s^{-1} (V)	CV Ratio of slopes from Randles-Sevčik plots	SWV Sensitivity (A M^{-1})	SWV RSD of Peak Potential (%)
1	0.90	1.1	3.69 x10^{-4}	1.43
2	0.45	0.4	5.69 x10^{-3}	1.18
3	0.55	0.6	2.75 x10^{-3}	0.73
4	0.80	No reduction peak observed	2.58 x10^{-3}	0.88
6	0.08	0.9	4.79 x10^{-2}	0.00

Table 3-5 also summarises the two important features from the SWV data namely Paste 6 has a sensitivity which is at least an order of magnitude bigger than that of the other pastes and the precision of the peak potential in Paste 6 is impeccable – it was the same value for each concentration.

Note that the shift in peak potential seen in Figure 3-6 and 3-7 was due to the differing electrocatalytic effects of the electrode material. In general, the more positive the potential has to be to enable oxidation, the slower the rate of electron transfer. Paste 6 clearly has the lowest peak potential (as well as the sharpest peak), which indicates it has the best electron transfer kinetics.

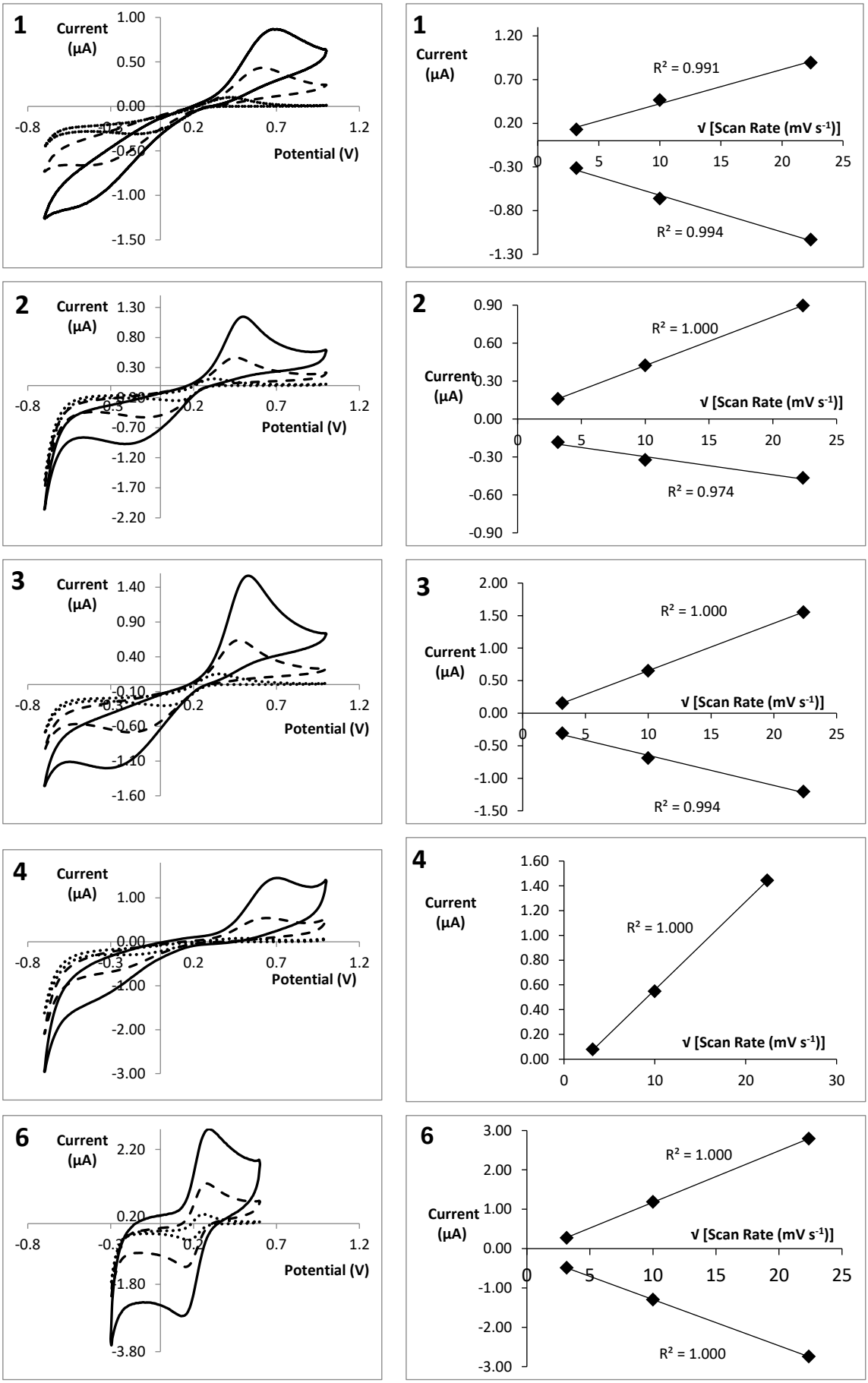


Figure 3- 6: : CVs and Randles-Sevčik plots for the five functional electrodes using 99 µM $K_3Fe(CN)_6$ in 0.1 M KCl. The number is the top left relates to the paste number as given in Table 3-3. The scan rates for the dotted, dashed and solid lines are 10, 100, and 1000 mV/s.

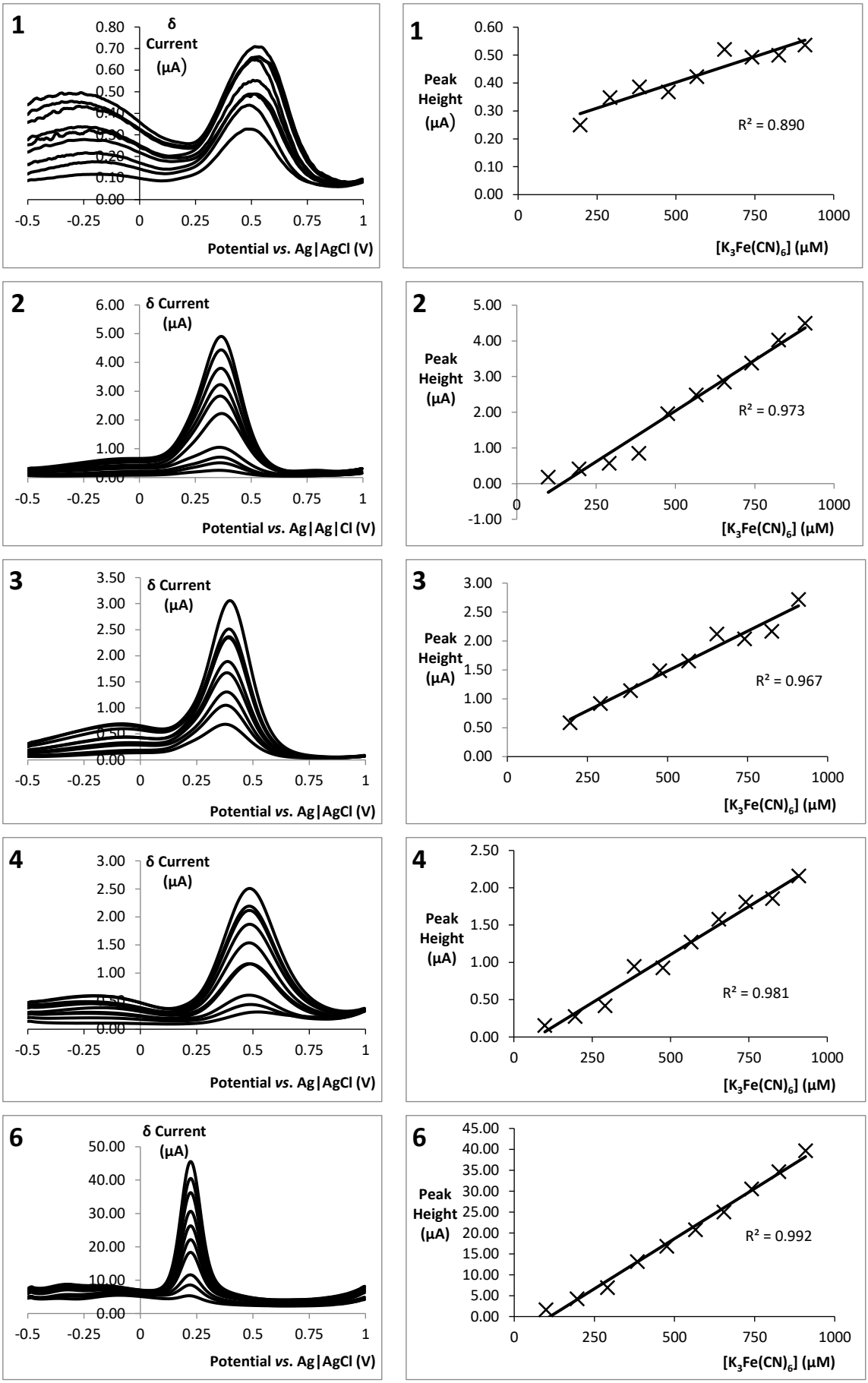


Figure 3- 7: SWVs on the left and the corresponding regression plots for varying concentrations of $K_3Fe(CN)_6$ in 0.1 M KCl. The composition for the paste number in the top left of each is given in Table 3-3.

As Paste 6 clearly had the fastest heterogeneous electron transfer, as shown by its reversibility and sensitivity, it was chosen for the subsequent analysis when the focus of the research returned to BZP.

3.3.2. Mechanism of BZP Oxidation

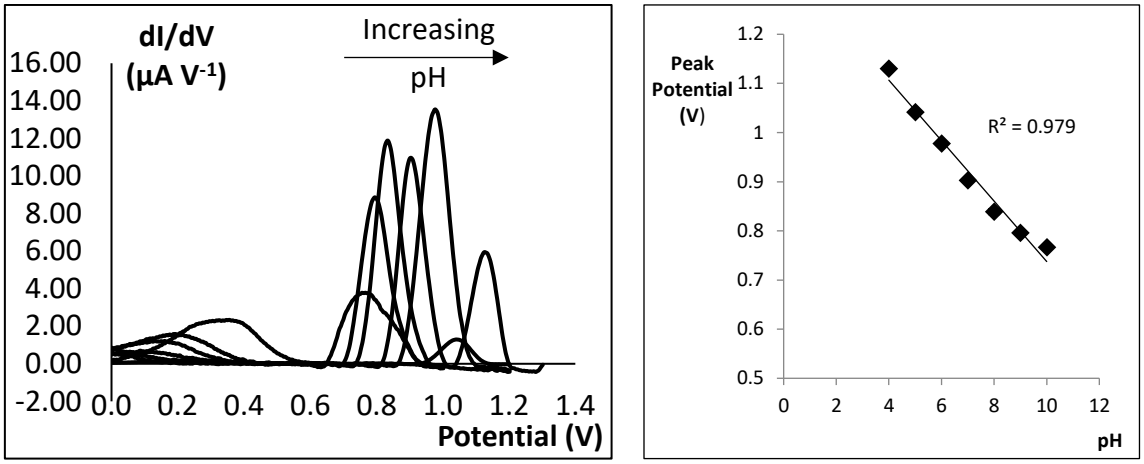


Figure 3- 8: First differential of LSV analysis of 49 µM BZP at varying pH (left), and the regression of line for how the average Peak Potential (EP) changes with pH (n=3) (right).

The data for the LSV analysis at varying pH is shown in Figure 3-8, however it should be noted that the first differential of the current with respect to the potential has been plotted for clarity. It was noted that peak potential (E_P) decreased with increasing pH according to the relationship in Equation 3-1 where ($R^2 \approx 0.98$):

Equation 3-1: $E_P\ (V) = -0.062 \times pH + 1.353\ (V)$

The closeness of the slope to 0.059 V pH^{-1} indicates that an equal number of electrons and protons are involved in the charge transfer mechanism by analogy to the Nernst equation (Equation 1-1). Other than decaying at either extreme of the pH scale there did not appear to be a clear relationship between peak current (I_P) and pH. Therefore, the pH value of 9.5 was chosen for further analysis which is close to the pKa value of BZP (literature value 9.59 (158)).

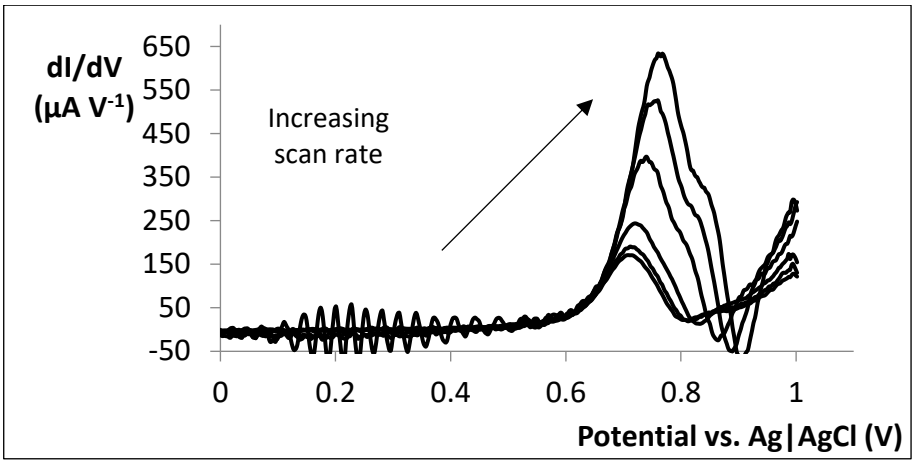


Figure 3- 9: First differential of LSV analysis of 120 µM BZP at varying scan rate in pH 9.5 BR buffer (n=3). Averages used for peak potential values, single scans shown of 150, 175, 250, 500, 750, and 1000 mV s^{-1} scan rate.

This pH serves two advantages: a relatively low value of E_P is used meaning there is less chance of also oxidising interfering compounds and a relatively high I_P is maintained. An E_P / pH relationship such as this also indicates that it is the amine group which is being oxidised rather than the aromatic ring. The mechanism was further investigated by altering the scan rates used in LSV and the results of which are shown in Figure 3-9, where again the first differential of current with respect to potential has been plotted. This data indicated a diffusion limited process as there was a linear relationship between the square root of the scan rate (υ) and the peak current (Equation 3-2) where $R^2 = 0.99$.

Equation 3-2: $I_P\ (A) = 8.802\times10^{-4}\times[\upsilon\ (V\ s^{-1})^{0.5}] - 1.968\times10^{-4}\ (A)$

The (E_P) was also observed to shift to higher values with increasing scan rate and a linear relationship was observed between the ln υ and E_P (Equation 3-3).

Equation 3-3: $E_P\ (V) = 0.0317\times\ln[\upsilon\ (V\ s^{-1})] + 0.7665\ (V)$.

In general, the shift in peak potential for a completely irreversible process with changing scan rate is given by Equation 3-4 (151).

Equation 3-4: $$E = E_f^\circ - \frac{RT}{\alpha nF} \times ln\left(\frac{RTk^\circ}{\alpha nF}\right) + \frac{RT}{\alpha nF} \times ln(\upsilon)$$

Where E_f° is the formal potential, R, T and F have their usual assignments, α is the transfer coefficient, n is the number of electrons involved in the charge transfer and k^o is the heterogenous rate constant. Therefore, by analogy with the last term in Equation 3-4 and the slope of the graph of E_P versus ln(υ), the value of αn can be estimated as 0.8. This data taken in its entirety would seem to support an overall mechanism involving the loss of two electrons and two protons.

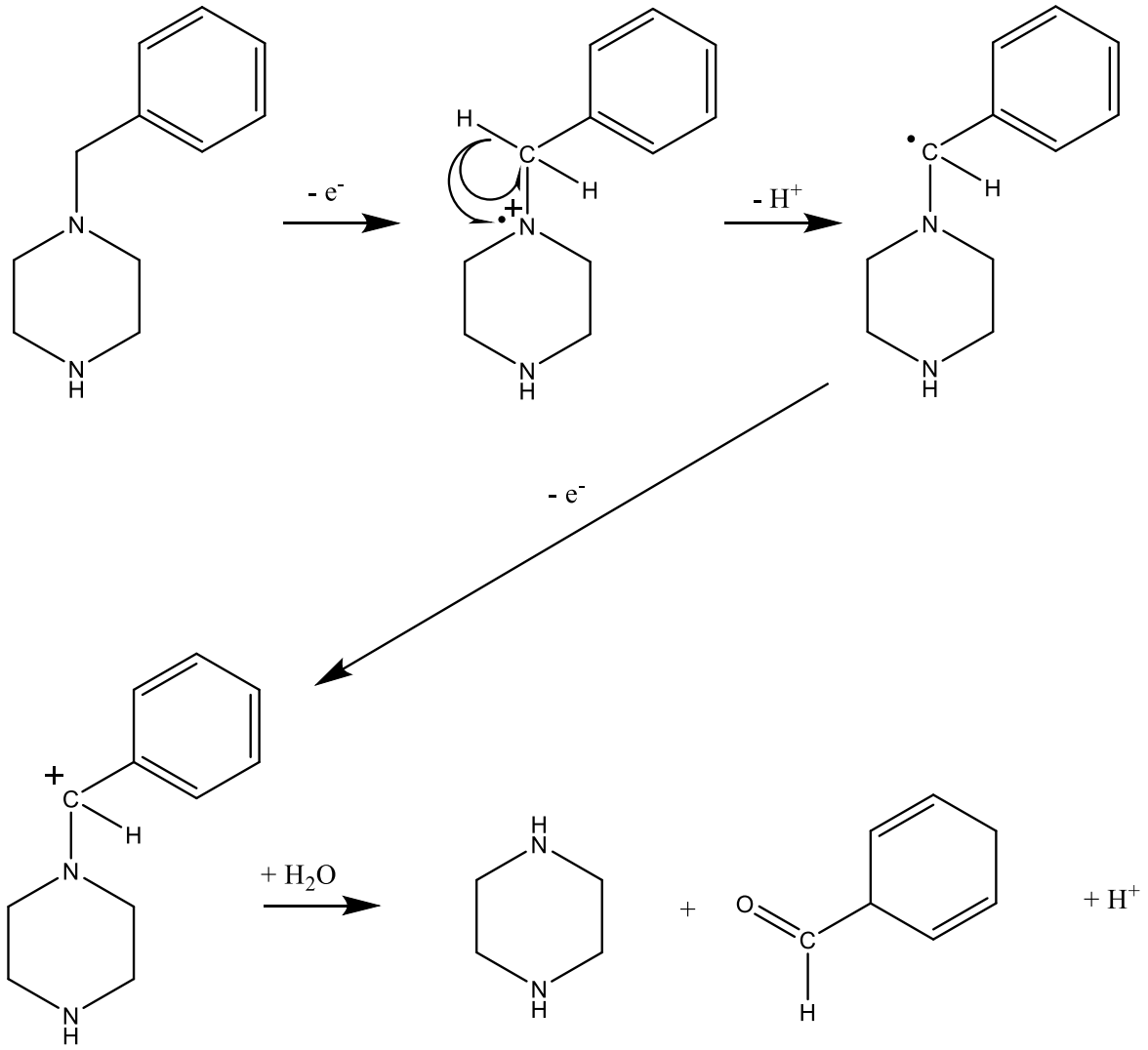


Figure 3- 10: Proposed two electron and two proton mechanism of BZP oxidation at the carbon paste electrode terminating with piperazine and benzaldehyde.

Therefore, the proposed mechanism, as shown in Figure 3-10, is consistent with the oxidation of tripropylamine as described by Portis *et al.* (82). This also explains the complete irreversibility as the tertiary amine is lost at the end of the mechanism due to a homogeneous reaction with the solvent.

3.3.3. Optimisation of BZP Oxidation at the Paste Electrode

The attention was then turned to optimising the SWV method for the analysis of BZP. Increasing lengths of time were used for deposition to determine the point at which the electrode would become saturated. The resulting data of peak current versus deposition is shown in Figure 3-11. It was obvious that there was no advantage to increasing the deposition time beyond 135 seconds, so this was used for subsequent analysis.

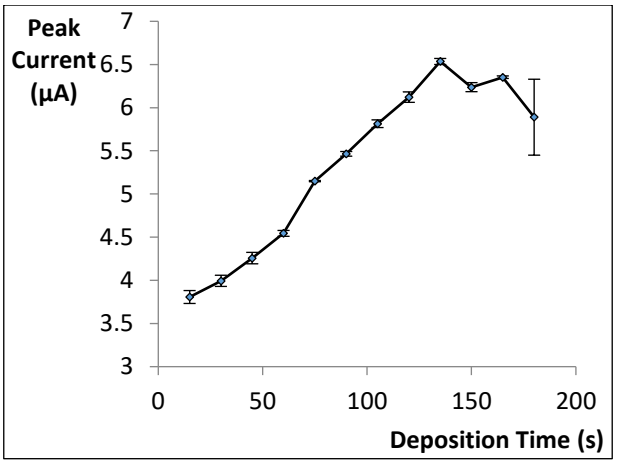


Figure 3- 11: Investigation of SWV peak current with varying deposition time using the Paste 6 electrode for a 120 µM BZP in pH 9.5 BR buffer. The error bars are 1 standard deviation (n = 3).

A full factorial design was then used to investigate the effects of the SWV parameters: step height, amplitude, and frequency. The results were interpreted both in terms of sensitivity (where the absolute value of the peak current was used) and in terms of precision (where the relative standard deviation of the peak current (n = 3) for each setting was used). It was found

that no one factor, or combination of factors had a statistically significant effect (at 95% confidence) over the others in terms of sensitivity. However, the step height, the frequency and the combination of step height and frequency were found to have a statistically significant effect in terms of precision. Both sensitivity and precision were used in order to find the optimised value for each parameter which was step height 10 mV, amplitude 50 mV and frequency 13 Hz.

3.3.4. Validation of the Optimised Method

The validation of the method began with the analysis of range of concentrations of BZP in order to assess linearity and the limit of detection and quantification (LOD and LOQ). The regression data is shown in Figure 3-12. The method was shown to be linear between 12 and 68 μM (R^2 = 0.99) beyond this range it was noted there was slight deviation, however it was also noted that there was linearity if tested over a higher range e.g. 100 to 200 μM (data not shown).

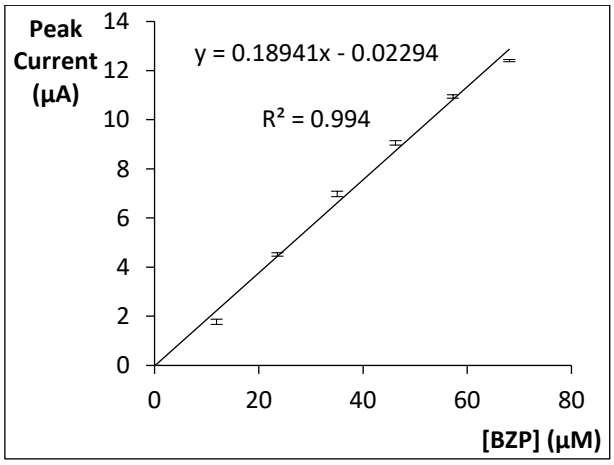


Figure 3- 12: Regression calibration for BZP using the Paste 6 electrode in pH 9.5 BR buffer and the optimised SWV method. The error bars are one standard deviation (n = 3).

The LOD and LOQ were determined using the sum of the square of the residuals method (*i.e*. 3 and 10 times the standard deviation of the blank) and were found to be 6 and 20 µM respectively. As an example, application a 61 µM BZP in Pepsi Cola® solution was tested against a cola blank. The blank did not have any peaks in the region of the BZP oxidation and the concentration was determined by reference to the linear regression used in the LOD determination. A comparison of the concentration by weight and the concentration by calculation showed the values to agree to within 0.08% which was excellent precision (n = 3). Lastly the solution of MDMA was analysed using the optimised SWV and a comparison of the voltammograms with BZP is shown in Figure 3-13.

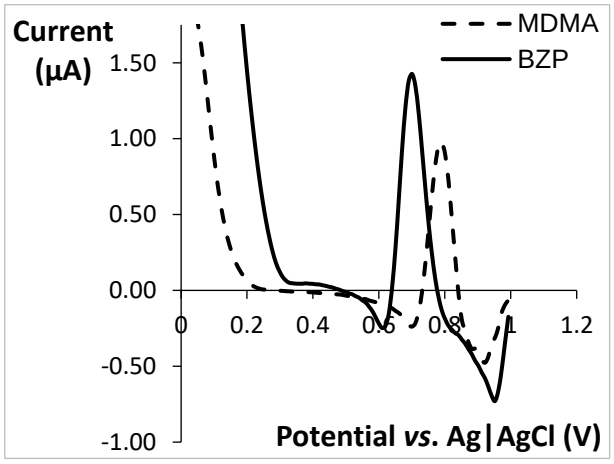


Figure 3- 13: Optimised SWV comparison using Paste 6 electrode in pH 9.5 BR buffer for 3 µM MDMA and 12 µM BZP for comparison.

It was noted that although the peaks were not completely resolved there was a separation of 90 mV and it would be possible to distinguish between them in combination with the analysis of standards. The resolution was found to be 0.45 which was calculated by dividing the difference in peak potential by the average peak width.

3.4. Conclusion

3.4.1. BZP Analysis

An analytical method has been developed which offers the promise of portability, cheapness, speed, precision and accuracy for the analysis of BZP. Although there are many analytical techniques which have superior LOD parameters, this becomes irrelevant in the analysis of bulk drugs which is the goal of this research. An LOQ of 20 μM (which equates to 3.5 mg L^{-1}) is more than sufficient considering the average dose of a tablet is between 50 and 200 mg (159). However, if the analysis of body fluids for the presence of BZP using this technique was to be considered it would be important to assess the resolution between the hydroxylated metabolites and the drug itself. Further to what is described in this chapter a 10% proportion of single and multiwall carbon nanotubes (Aldrich) were added to paste 6 to assess the performance. However, there was no significant difference in these results.

3.4.2. Project Direction

This literature survey in Table 3-2 has clearly illustrated the applicability of dynamic electrochemical measurement to the analysis of drugs of abuse. The subsequent work on BZP has further demonstrated that even if there is no previous electrochemical research available on a novel psychoactive substance, there is a strong likelihood that this analysis will be viable. This is due to the alkaloid nature of the vast majority of these substances; the scientific community's understanding of the anodic oxidation of which has been developed over the past 50 years (chapter 2).

The remainder of the thesis returns to the topic of field testing. A proof-of-concept device which is portable, of low cost, and reliable would need to be

designed to ensure the full scope of the project is met *i.e.* "The development of an electrochemical sensor for controlled drugs".

Whilst there are handheld potentiostats currently available on the market such as the PalmSens (160), and there is ongoing research elsewhere developing low cost potentiostats (161), there is no such device that is targeted towards the application of forensic electrochemistry. It is to this end that the following chapter outlines the development of the novel hardware for the sensor.

4. Potentiostat Hardware

This chapter will describe the development of two potentiostat prototypes (labelled A and B). The main application of these prototype B, the analysis of controlled drugs, will be investigated subsequently. However, it should be borne in mind that they have an architecture which could be applied to various and diverse analyses. Therefore, some generalities regarding the functioning of a potentiostat is discussed prior to the details on the specific circuitry and firmware. The graphical user interface (GUI), the calibration, and the optimisation of the device will be described in the chapter 5.

4.1. Overview of Potentiostatic Instrumentation

Dynamic electrochemical techniques, such as voltammetry, require a potentiostat that is able to apply a set potential difference across a set of electrodes and measure the resulting current. In almost all cases, the potential of the working electrode (WE) will be swept in a particular direction so will be varied over time. In its most simple form, a potentiostat can be thought of to be comprised of three operational amplifiers, a digital to analog converter (DAC) and an analog to digital converter (ADC) as shown in Figure 4-1.

Reading Figure 4-1 from left to right it is shown that the reference electrode (RE), WE and DAC can be considered as inputs and the counter electrode (CE) and ADC are outputs from the circuit. There is no direct electrical connection between WE and the other electrodes shown however the cell's electrolyte achieves this.

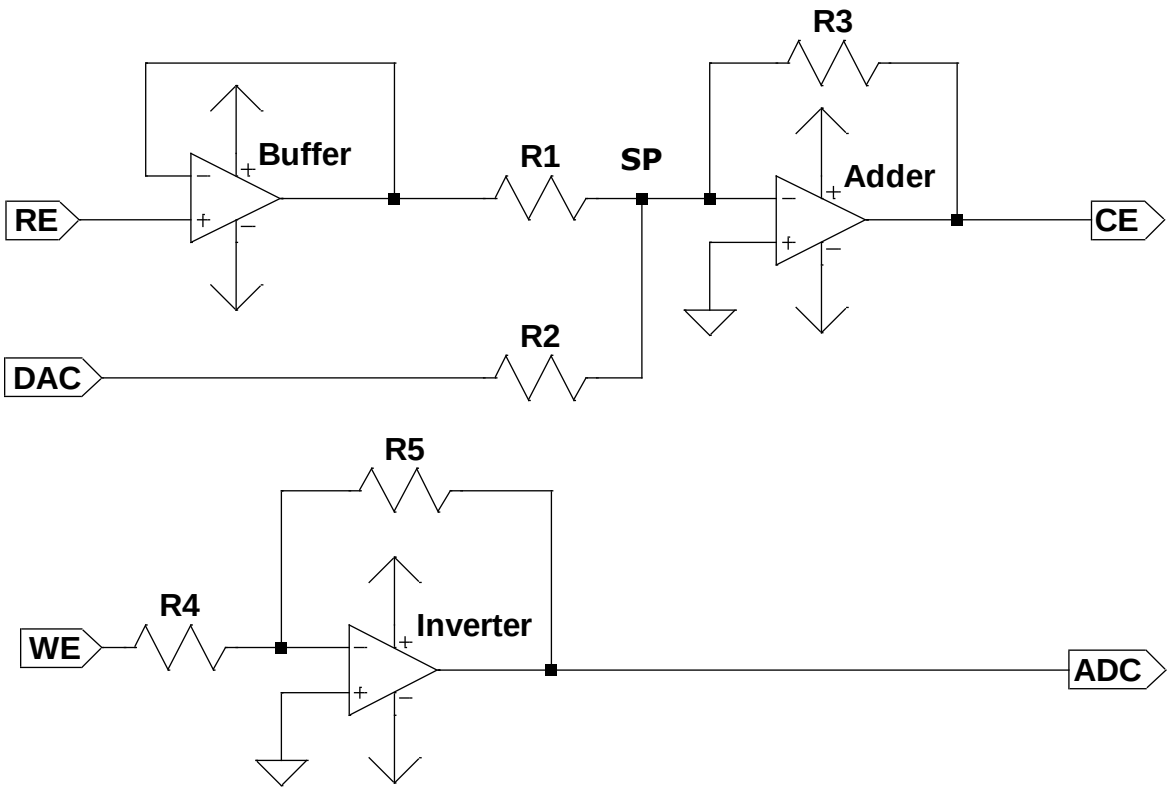


Figure 4- 1: Generalised overview of the components involved in a potentiostatic circuit showing the arrangement of op amps, resistors, and summing point (SP).

Also missing from the diagram is some form of control for the DAC and ADC which could range in complexity from a fairly simple 8-bit AVR microcontroller through to a more complex ARM microprocessor or even a specifically designed chip such as a field programmable gate array (FPGA). The complexity of the design needs to consider the "rate determining step" of the overall circuit *e.g.* a high-end Intel microprocessor would be of no advantage if the settling time of DAC to reach a stable output runs to tens of µs. The slew rates and stability of the operational amplifiers also need to be considered. It is proposed here to create a low cost 12-bit system that would produce comparable results to commercially available potentiostats. However, prior to providing specific details on the circuits produced in this study, further explanation will be provided on the components of the generalised schematic.

4.1.1. The Reference Electrode

The reference electrode is necessary as it informs the system of the current cell potential. It is essential that it is not involved in any faradic processes, otherwise its potential would drift significantly. Nevertheless, it must still be able to influence the potential that is applied to the CE. This in turn necessitates knowing its potential whilst drawing negligible current from it. An operational amplifier (op amp) in the buffer (also known as voltage follower) configuration meets these criteria perfectly. Here the RE is connected directly to the non-inverting input of the op amp whilst the output is fed back to the inverting input (162). This configuration means an ideal output would follow the RE as long as it does not go above or below the positive or negative voltage rails that are supplied (represented by the arrows in Figure 4-1). An actual op amp would not be able to achieve this ideal, however there is an enormous range available today which are classified as "rail-to-rail" (the component distributer Digi-key Electronics lists approximately 20,000 (163)). There is no industrially accepted definition for the term "rail-to-rail" but no op amp will swing completely to both voltage rails. The extent of the output swing is dependent on the sink and source current but if the output is near the rail potential, there will be distortion in its signal (164). Therefore, depending on the device, it may be advisable to leave a headroom of approximately 200 mV at either end. Nevertheless, the suitability of the buffer op amp to obtain the RE potential is well documented (165).

4.1.2. The Counter Electrode

The output from the RE buffer and DAC both pass through resistors which are combined at a summing point (labelled as SP in Figure 4-1) which will be held at virtual ground due to its connection to the inverting input of the op amp. An adder op amp (also known as summing amplifier) is used whereby the output is also fed back through a resistor to the inverting input and the non-inverting input is held at ground. In the situation where the values of the resistors R1, R2, and R3 are equivalent, the output will be equal to sum of the potentials of the DAC and the RE (166). This output is connected directly to the CE in order to supply the potential to cell.

The rail considerations mentioned in Section 4.1.1 are still pertinent however the slew rate of the adder is now much more significant as the output needs to quickly reflect that of the changing input from the DAC. In the scope of most cyclic voltammetric experiments scan rates are in the range of 1 to 1000 mV s^{-1}. Considering the slew rate of typical op amps are given in V μs^{-1}, this would not appear to pose any problem (more detail is provided in Section 4.2.4). However, the instability of op amps which are adversely affected by high capacitive loads can lead to oscillation in the signal. Careful design of the circuit may render the parasitic capacitance inconsequential, but there will always be the inherent capacitance at the electrode/electrolyte boundary. This must be borne in mind as it can have a negative effect on the signal quality.

4.1.3. The Working Electrode

The current measured in voltammetric applications is often in the range of μA or lower. Therefore, in order to accurately determine currents value with reasonable resolution it is necessary to amplify its value. An inverting op amp provides a convenient way to increase the signal whereby the amplification in voltage (A_V) is given by Equation 4-1 in reference to Figure 4-1 (167).

Equation 4-1: $A_V = \frac{R5}{R4}$

Typical resistor values are in the kΩ range in order to maintain a suitable current flow in the feedback loop. Further to this if both positive and negative currents are to be measured - as is the case in voltammetry – the negative rail of the inverter must be below the potential of its non-inverting input (shown as ground in Figure 4.1). One simple solution to this would be to use a buffered voltage divider circuit such that the new "analog ground" is a midway point between the positive voltage and true ground.

4.1.4. Electrochemical Cell Design

The geometry of how the electrode are placed in the electrochemical cell has an influence on the signal. Firstly, the area of the CE should be larger than the WE. This is to ensure that it is the analyte's reaction at the WE that is current limiting. Second the distance between the electrodes is also important. The larger the distance between the WE and CE, the higher the voltage that the potentiostat will need to supply in order to the CE to ensure the WE has the correct potential with respect to the RE. It is possible that this could exceed the compliance of the device. Furthermore, the distance between the RE and the WE should also be minimised in order to have an accurate potential reading.

4.2. Initial Circuit Considerations

4.2.1. The Digital to Analog Converter (DAC)

4.2.1.1. Using an R2R Network

Digital to analog conversion can be performed quite simply using an R2R network as shown in Figure 4-2.

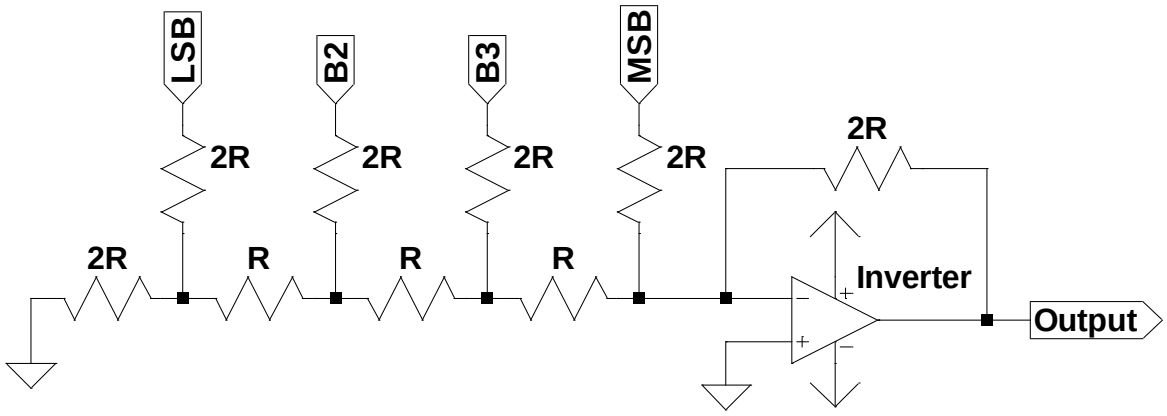


Figure 4- 2: Simple 4-bit R2R network with op amp buffer to be used as DAC.

The least significant bit (LSB), most significant bit (MSB), B2 and B3 can be connected to digital outputs which when active, all hold the same potential. They can be sequentially switched on in order to achieve a range of 16 values between ground and active potential as it is a 4-bit system (168). The resistor values are arbitrary, provided one set (labelled 2R) are double the values of the others (leaving aside factors such as Johnson Noise (169)). Current considerations may also limit the range of actual resistances used. Regardless of the chosen value, the resistance must be known precisely in order to have even steps across the range. Expansion with more bits (resistors and digital outputs) is of course technically possible to achieve reasonable resolution across a given range. However, ensuring the accurate and stability across temperature ranges is a disadvantage of this technique so it was not implemented.

4.2.1.2. Using a Digital Potentiometer

Another simple option to provide analog values across the range is to use a potentiometer or variable resistor as part of voltage divider circuit as shown in Figure 4-3. Many digitally controlled potentiometers (digipots) are available that can communicate with a microcontroller over inter-integrated circuit (I^2C) or serial peripheral interface (SPI) buses. This technique was originally tested for use in the potentiostat model A using an MCP4261, which is a 10 kΩ, 8-bit digipot available from Microchip Technology, Inc. (170). All components used in this project were purchased from Digi-key Electronics.

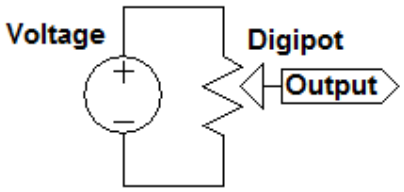


Figure 4- 3: Digipot voltage divider

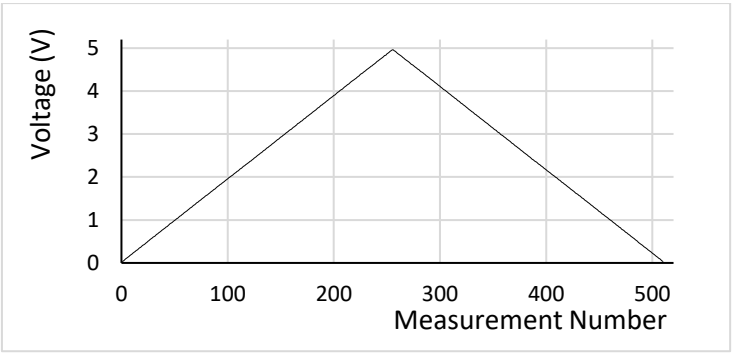


Figure 4- 4: MCP4261 output trail measured by on board ADC of Arduino Uno.

This technique did indeed produce a smooth transfer function across a 5 V range as shown in Figure 4-4. The output was monitored using the 10-bit ADC of an ATMega168 on an Arduino Uno R3 development board (more details are provided in section 4.2.3). The circuit schematic and firmware code that were used are detailed in Appendix 1, (schematics were created using Eagle v7.6.0 (CadSoft). In short 5 V and ground were attached to either end of the digipot and the wiper was used as the output pin. Figure 4-3 shows a triangular waveform, such as that necessary for cyclic voltammetry (CV), is easily achieved once 10-bit conversion was performed using Equation 4-2.

Equation 4-2: $$Voltage = \frac{5 \times ADC\ Reading}{1024}$$

The increasing portion of the waveform was used to test for linearity, resulting in an R^2 = 1.0000 as could be expected. A value for the sum of the square of the residuals (SSR) was found to be less than 0.003 (n = 3).

Furthermore, the period of one wave was found to be approximately 2540 ms resulting in a scan rate of approximately 3.9 V s^{-1} which is more than satisfactory. However, the limit of the 8-bit system meant the resolution was insufficient as the voltage step was approximately 19 mV. In order to properly ascertain electrochemical values, the potential resolution should have an upper value of 10 mV. A higher resolution digipot was not sought for two further reasons. Firstly, the potential is not always swept in a linear fashion but is often pulsed at different values *e.g.* in square wave voltammetry (SWV). It was anticipated that the digipot would not handle that circumstance as well. Secondly, it was noted that there appeared to be a short lifetime for the component with continual sweeping during initial experiments which were conducted with rudimentary potentiostats. Therefore, an alternative strategy was sought.

4.2.1.3. Using a Monolithic Integrated Circuit

Many monolithic integrated circuits (IC) are available to perform DAC functions. One such chip, the MAX5250 from Maxim Integrated (171) was selected due to its supply voltage range of 5 V which would comply with a USB supply and the logic level of the ATMega microcontroller. Simple SPI communication with the device was also explained well in the part's datasheet. The MAX5250 has four channels and although only one is necessary for a simple potentiostat, it was deemed beneficial for future expansion *e.g.* control of more than one WE or configuring a channel to operate as a galvanostat.

The schematic and firmware used to assess the viability of the MAX5250 as detailed in Appendix 2. Similarly, to the digipot test, triangular waves were easily obtained and using the ascending slope to assess linearity gave R^2 = 1.0000 and SSR of less than 0.18 (n = 3). In the initial test, the scan rate was approximately 0.96 V s^{-1}. However, increasing the baud rate from 9600 to 2000000 produced a maximum scan rate of over 16 V s^{-1}, illustrating the serial data transfer over universal serial bus (USB) to be the limiting factor. No signal degradation was observed at the faster scan rate. Furthermore, the resolution of the sweep was improved to a step value of approximately 4.8 mV due to the 10-bit nature of the MAX5250 IC.

In summary, this IC was deemed acceptable to use as the DAC for the initial prototype (model A).

4.2.2. The Analog to Digital Converter

The assessments of the DAC in the section 4.2.1 used an on-board ADC pin of the ATMega 168 microcontroller to measure the output. However, there are two major drawbacks to this approach. Firstly, the time taken for a single measurement is approximately 100 μs due to the abstractions that are part of the Arduino platform (more details on the microcontroller are provided in section 4.2.3). Although this seems a short time frame, it was anticipated that multiple measurements would be taken in the final circuit for signal averaging. This would in turn lead to limitations on the scan rates available. The sampling rate problem could be overcome by using a different programming environment such at the chip manufacturer's own Atmel Studio 7 (172). Nevertheless, the second disadvantage is the inherent 10-bit resolution which is inadequate for precise measurements. Therefore, another option was sought for signal measurement.

The MCP3304 is an eight channel ADC IC manufactured by Microchip Technology (173). The channels can be configured as 4 differential input providing 13-bit resolution *i.e.* 2^{13} = 8192 distinct measurement values over the range 0 to 5 V. Furthermore, a fast sampling rate of 100 kilo samples per second (ksps) is possible which communicating over SPI to the microcontroller. The test circuit in section 4.2.1.3 was expanded to incorporate the MCP3304 to assess its viability. The schematic and firmware for this circuit is shown in Appendix 3.

In the test circuit, a 2.5 V voltage reference was achieved using two 140 kΩ resistors as voltage dividers from the 5 V supply provided by the USB. This was further buffered using voltage follower op amp. Figure 4-5 shows the output of the MCP3304 ADC using a 12-bit correction shown in Equation 4-3.

Equation 4-3: $(CH1 - CH0)\ \ Differential\ Voltage = \frac{5 \times ADC\ Reading}{4096}$

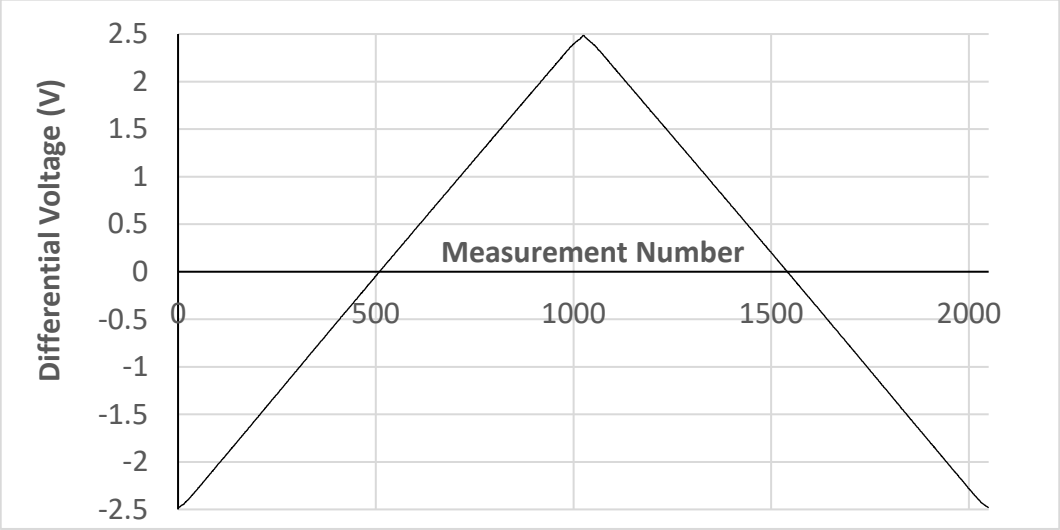


Figure 4- 5: MCP3304 test 5 V sweep output from MAX5250 (see Appendix 3 for circuit)

Note this is plus and minus 12-bit producing the 13-bit stated previously. The output shown represents the ADC's channel 1 (CH1 =2.5 V reference) subtracted by channel 0 (CH0 = DAC sweeping from 0 to 5 V and back again).

The advantage of using the 2.5 V reference extends beyond resolution and is in fact vital. An electrochemical potentiostat must be able to measure both positive and negative currents. Setting the WE at the "analog ground" potential of 2.5 V splits the compliance voltage in half permitting electron flow in both directions for both reduction and oxidation measurements.

For confirmation, linearity was assessed using the ascending slope of the of the graph. This produced an $R^2 = 1.0000$ and SSR not more than 0.33 (n = 3). Using single data point measurement points the maximum scan rate was over 21 V s^{-1}. Next the signal was averaged by polling 100 measurements at each DAC increment which yielded a scan rate of approximately 1.75 V s^{-1} which is still significantly faster than necessary. Whilst there was no difference in response (*e.g.* SSR) for averaging the result in this simple circuit, it was projected that the analog values derived from the electrochemical cell would contain significantly greater noise due to amplification. The MCP3304 has an accuracy of ± 2 LSB at the fastest sampling rate which equates to 2.44 mV of the 5 V range. It has a successive approximation register (SAR) architecture which in general have excellent power efficiency (174). To briefly illustrate how it operates, the two differential inputs to the SAR are shown on the left of Figure 4-6. A capacitor DAC (CDAC) is used to test each bit in sequence beginning with the MSB against the measured value. If the CDAC value is higher, that particular bit is cleared, if not it is retained. The process is repeated until the LSB is reached and the shift register produces the binary output.

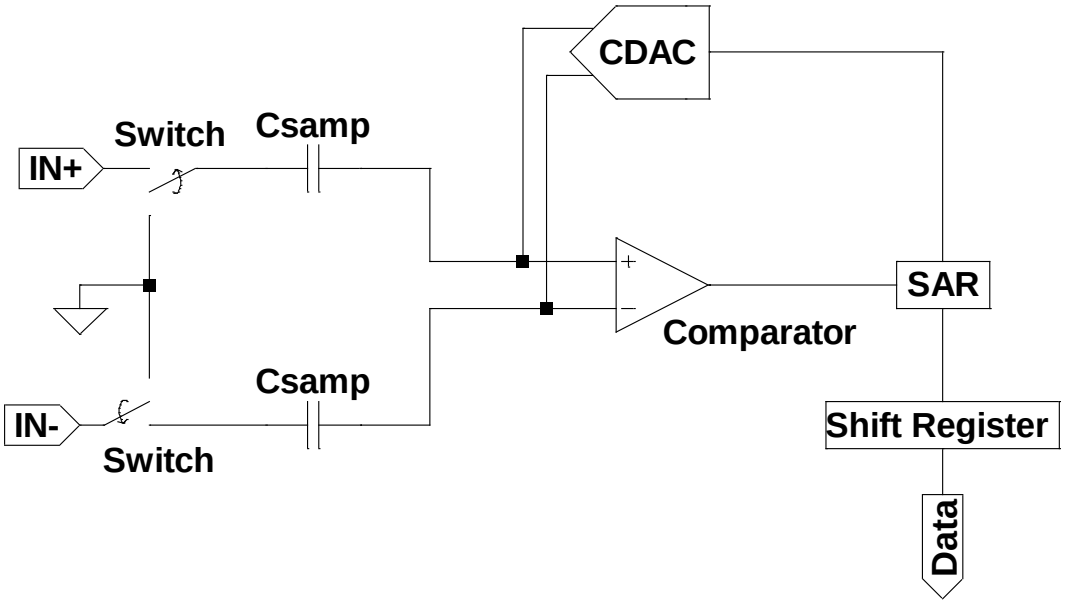


Figure 4- 6: Simplified Schematic of Successive Approximation Register which is often used as ADC architecture

The two differential inputs to the SAR are shown on the left of Figure 4-6. A capacitor DAC (CDAC) is used to test each bit in sequence beginning with the MSB against the measured value. If the CDAC value is higher, that particular bit is cleared, if not it is retained. The process is repeated until the LSB is reached and the shift register produces the binary output.

The MCP3304 finds research uses in a wide range of applications such as devices to measure radiation (175), Hall effect sensors (176) and sensing artificial muscle actuators in ionic polymer-metal composites (177). This component was chosen as the ADC for the system as it would provide a balance between speed of response, resolution and low cost (£2.67 at time of writing, March 2019).

4.2.3. The Microcontroller

4.2.3.1. Using an 8-bit AVR with the Arduino Platform

Arduino is a development platform that enables rapid prototyping for systems with a range of microcontrollers. A dedicated integrated development environment (IDE) and user-friendly library files enable facile development albeit at the expense of fine control of timed functions and final executable code size. However, excellent documentation and a vibrant community of developers has helped Arduino to become the world's leading open-source hardware and software ecosystem (178). The syntax used in the platform is based on the language C (179) and the avr-gcc complier (180) is used in conjunction with the AVRDUDE (181) software to upload to the microcontroller.

The Arduino Uno, shown in Figure 4-7, is a development board that uses an ATMega328P (182) as the main microcontroller chip which has: 14 x digital input/output pins (I/0) at a 5 V logic level, 6 x 10-bit analog input pins, 32 kB of Flash Memory and, I^2C and SPI buses built in. To support the ATMega328P there is a 16 MHz oscillation crystal and in circuit serial programming header pins. Alternatively, the microcontroller can be programmed using the Arduino IDE over USB 2.0 (type B connector) and there is an ATMega16U2 (183) for USB to universal asynchronous receiver/transmitter (UART) conversion. The power port on the board supports 7 to 12 V supplies and is regulated using two op amps (LMV3581DGKR (Texas Instruments) (184)) and a linear regulator (NCP1117ST50T3G (On Semiconductor) (185)).

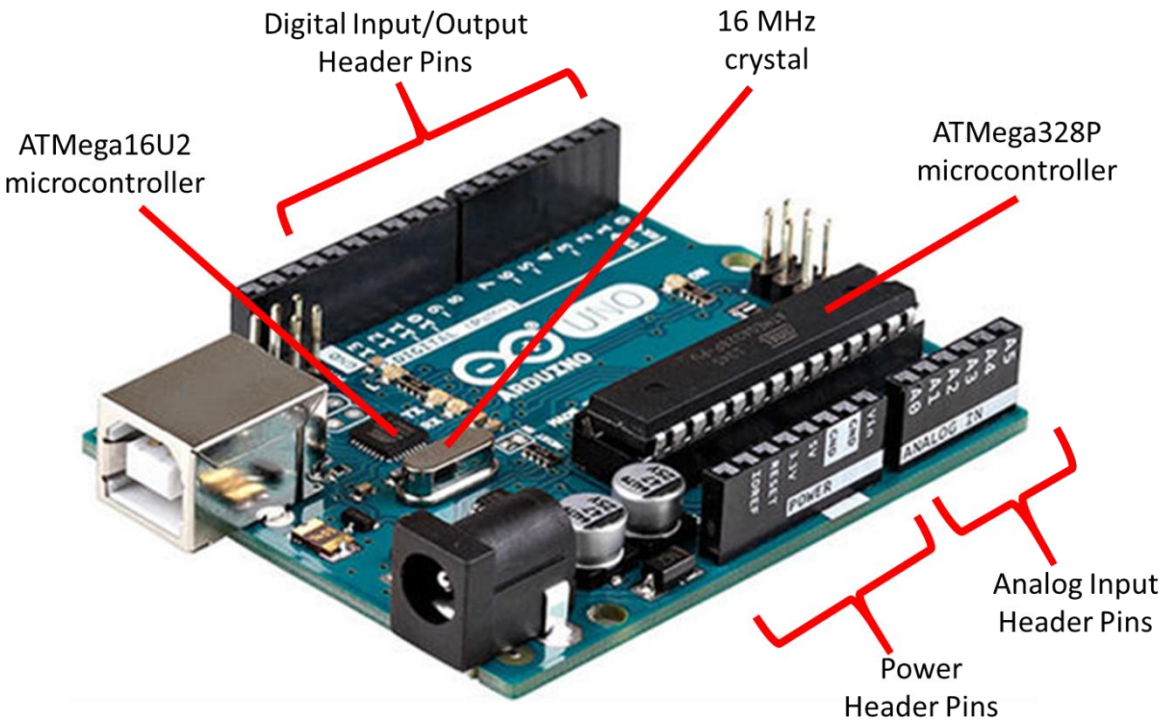


Figure 4- 7: Arduino Uno Rev3 adapted from (186). Parts not labelled in diagram include a barrel jack for power with associated decoupling capacitors. A USB type-B connector and 6 header pins which are for in circuit serial programming (ICSP).

Early rudimentary tests, such as those detailed in the previous sections, proved that an 8-bit AVR microcontroller easily satisfied the hardware requirements for the potentiostat. An advantage of AVR microcontrollers is their high performance yet low power consumption (187). However, it was deemed appropriate that any intensive calculations *e.g.* peak smoothing, should be conducted on the host computer where memory and processor power are in far greater abundance. For early proof-of-concept prototypes the Arduino Uno was chosen to its ease of use and low cost (approximately £17 at time of writing). Once the main components of the circuit have been ascertained, transfer to a 32-bit ARM cortex M0+ system should be easily achievable (given technologies such as Atmel's START (188) which enables graphical configuration of peripherals such as communication buses).

The overview block diagram shown in Figure 4-8 illustrates the USB in the host PC will be used to supply power as well as being used as the data bus;

thus, enabling the device to be portable if a laptop was used. Given portability is one of the main design requirements, any alternative power supply was discarded at this stage.

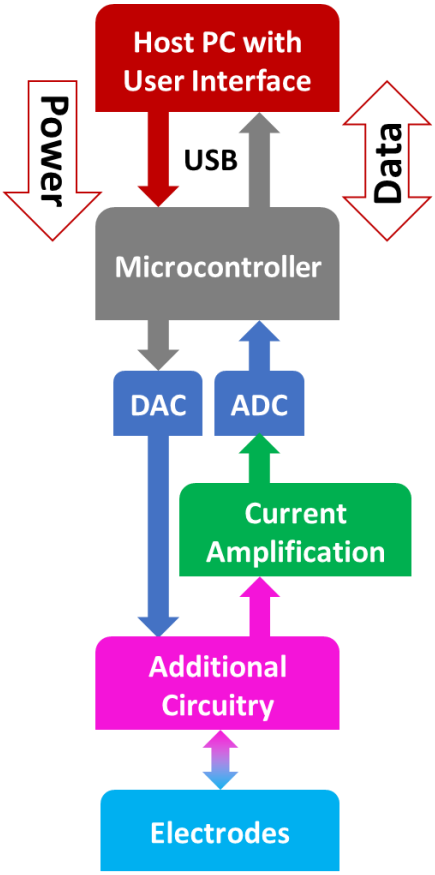


Figure 4- 8: Block diagram of potentiostat strategy. The "Current Amplification" box can be represented by the lower half of Figure 4- 1 and the "Additional Circuitry" can be represented by the upper half of Figure 4- 1.

4.3. Alternative Technologies

There are several main manufacturers of high performance potentiostats most notably Metrohm Autolab (189), Gamry Instruments (190), eDAQ (191) and Ivium (192). Whilst these instruments have excellent sensitivity and extremely low noise, their high cost – in the thousands of pounds – can be prohibitive.

Nevertheless, there are low cost potentiostats which could be viewed as direct competitors to the work presented here. Some initial remarks on the two major systems follow.

4.3.1. Cheapstat

Largely aimed at educational applications, the Cheapstat (193) shown in Figure 4-9 was being sold by IO Rodeo for $135 at the time of writing (194). This system also uses an Atmel AVR microcontroller, the ATxmega32A4U (195). This option allowed the use of the chip's on board 12-bit DAC and 12-bit ADC. As it is an open source development the schematics, Gerber files for manufacture, firmware and software are all available from the IO Rodeo website. The team's original publication detailed successful applications such as CV analysis of ascorbic acid in orange juice, linear sweep voltammetric (LSV) analysis of paracetamol and anodic stripping square wave (ASSWV) analysis of arsenic.

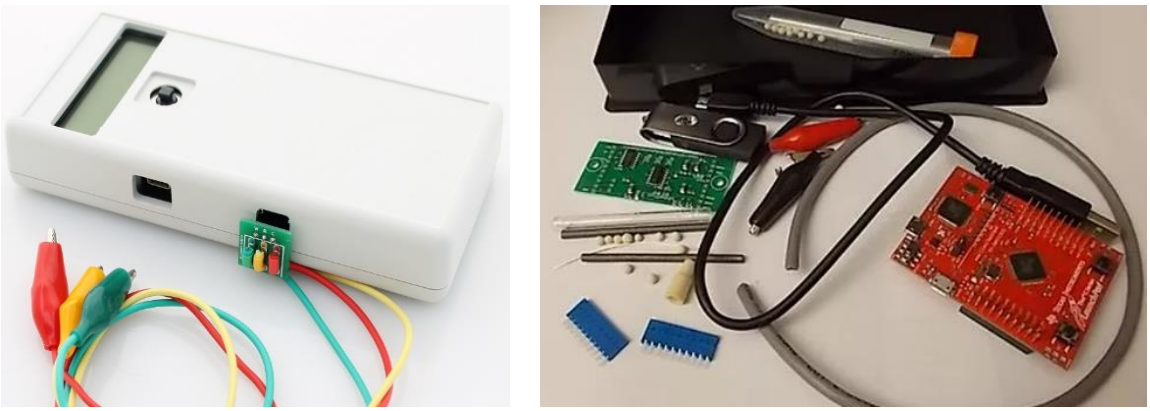


Figure 4- 9: Cheapstat hardware (left) and Wheestat hardware (right). Note in each case assembly is required by the user.

Whilst a GUI is provided to analyse the results, to set the method parameters a somewhat cumbersome joystick is used with details shown on an LCD screen. Given there is already a USB connection, setting the method parameters using the GUI could be argued to be quicker, easier and more

versatile. Also, the research conducted here did not perform any comparison in response to market established potentiostats. Notwithstanding these points, the Cheapstat does represent an excellent example of the trend towards low cost, self-assembly analytical equipment.

4.3.2. Wheestat

The Wheestat (196) shown in Figure 4-9 uses a different platform, Energia by Texas Instruments (197). This platform is highly similar to Arduino as they are both derived from the same original source code known as Wiring (198). However, the library files included Energia are designed to connect to Texas Instruments' Launchpad (199) development boards. The Tiva C Series Launchpad (200) is the red printed circuit board (PCB) in Figure 4-9. This development board uses a 32-bit ARM Cortex-M4 processor which has considerably more speed (80 MHz) than the microcontrollers previously discussed. The green PCB in Figure 4-9 is the Wheestat circuit itself and it is also supplied with electrodes and cables, although some assembly is required. One major advantage of this system is the use of a GUI to program the method parameters for the electrochemical method. This enables saving of files in comma separated values (CSV) format. The online literature for the Wheestat likewise demonstrates some applications such as ferricyanide CV analysis, but also fails to provide any comparison between the output of this instrument and any other. However, as it is also open source all documentation for both the software and hardware is available. The GUI was developed using Processing (201) and is fairly user friendly.

4.3.3. Other Systems in the Scientific Literature

Whilst the two systems described above are available for retail there are several alternative systems published in the scientific literature. Table 4-1 outlines the overall strategies used by these three techniques.

Table 4- 1: Summary of low cost potentiostatic circuits available in scientific literature. Note that although system ② has a superior ADC to the device described in this thesis (24-bit vs. *13-bit), its DAC is entirely insufficient for a general purpose potentiostat.*

System	Microcontroller	DAC	ADC	Ref
①	Arduino Uno	Arduino Uno PWM passing through RC filter (8-bit)	Arduino Uno (10-bit)	(202)
②	Beaglebone	LMP91000 (>4-bit) 14 steps	LMP91000 (24-bit)	(203)
③	Arduino Duemilanove	Manually controlled potentiometers	Arduino Duemilanove (10-bit)	(204)

All three systems shown in Table 4-1 only demonstrated CV as an analytical technique, however system ① did include firmware such that an end user could adapt it to perform other analysis *e.g.* chronoamperometry. System ① illustrates the most basic possible set-up and is targeted for education purposes. Here the DAC is produced by a pulsed width modulation (PWM) signal produced by the Arduino Uno's microcontroller (the ATmega328P). This then passes through a resistor-capacitor (RC) filter to provide smoothing. One obvious drawback of this approach is the low 8-bit resolution than can be achieved. However, system ① did demonstrate the determination of ferricyanide's diffusion coefficient in 0.1 M KCl by CV which accorded well with the literature value (205).

Systems ② and ③ are aimed at the point-of-care diagnostic market and are presented as proof-of-concept research.

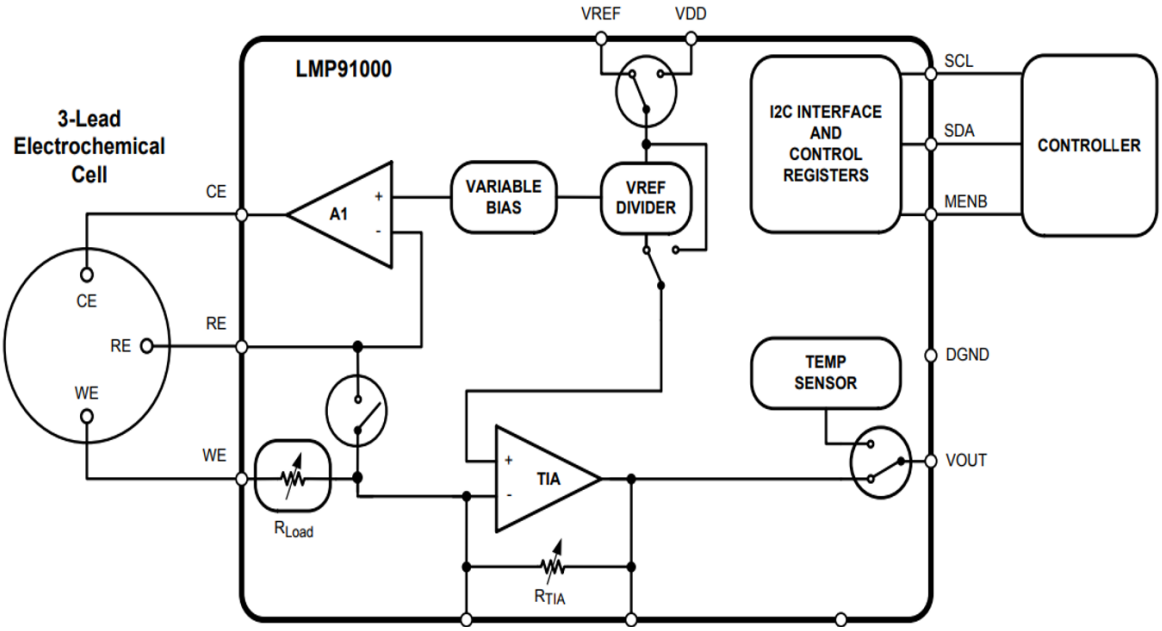


Figure 4- 10: Simplified schematic showing the functional parts of the LMP91000 (206). Note that the I^2C controlled "Variable Bias" which in turn controls the CE only has 14 different settings across the compliance range compared with the 4096 settings the DAC described in this thesis has.

System ② is by far the most sophisticated of those presented by using a Beaglebone (207) development board which has an AM335 1GHz ARM Cortex-A8 microprocessor (208). Coupled to this is the LMP91000 (206) analog front end which functions as a monolithic potentiostat in a tiny 4 x 4 x 0.8 mm package. The functional diagram of the LMP91000, (see Figure 4-10), shows how the adder (A1 in the figure) and transimpedance amplifiers (TIA) are incorporated and controlled by an I^2C bus. Using a TIA is a very common way to measure current and is analogous to the inverter shown in section 4.1. Interestingly the LMP91000 also has an integrated temperature sensor.

The research article for system ② successfully demonstrated the use of immobilised monoclonal anti-cortisol antibody on self-assembled monolayers (SAM) on gold electrodes for cortisol immunosensing (203). One convincing aspect of this research article is the direct comparison of the output of their

novel miniaturised potentiostat against an Autolab Potentiostat/Galvanostat, although they do not state which model.

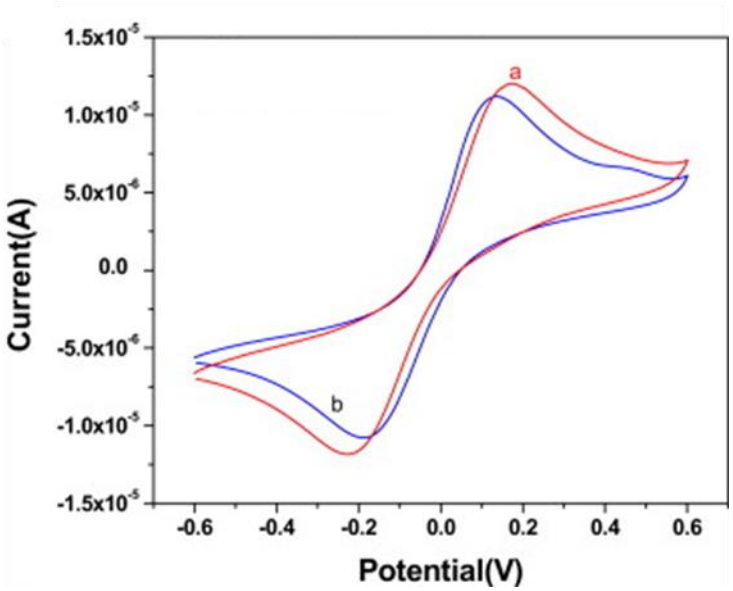


Figure 4- 11: Comparison of an Autolab potentiostat (line a) against system ② (line b) for 50 mV s^{-1} scans of 5 mM $K_3Fe(CN)_6$ in pH 7 buffer on a gold interdigitated micro-electrode (Au-IDE). Adapted from (203).

This is shown in Figure 4-11 and whilst the novel potentiostat voltammogram does show irregularities in the waveform at the more anodic potentials (a shoulder peak) and a greater difference in peak separation, in general there is good agreement.

System ③ uses an Arduino Duemilanove (209) development board which was the precursor to the Uno board prior to its release in 2010. The Duemilanove uses a similar microcontroller to the Uno but with half the on-board flash memory (using an ATmega168 (182)) although this would have negligible impact on careful system design. As seen in Table 4-1 the DAC in this system uses manually controlled potentiometers which seems awkward and lacking in fine control. Two separate potentiometers are used to set the start and stop potentials of the CV and a third is used to control the rate. It is suspected that this particular design would result in very poor precision due to the peak current being directly proportional to the square root of the scan rate in diffusion-

controlled systems. Any slight knock on the potentiometer's dials could skew results significantly and reproducibility from day to day would suffer. Disappointingly, no precision data is presented in the paper. However, direct comparison against a CHI 1040A potentiostat (CH Instruments, Inc.) is provided and agreement between the voltammograms is good although not perfect. For example, the peak current measured by the novel potentiostat is lower than the CHI 1040A. As the researchers are aware, the resolution of their ADC system (10-bit) is somewhat less than ideal, they attribute the difference in peak heights to this. However, it is far more likely to be due to measurement time as any dynamic electrochemical measurement of current is a sum of Faradic and non-Faradic currents. Furthermore, as non-Faradic currents decay at a much quicker rate, the exact point of time after the new potential has been applied that the current is measured can have a dramatic effect on its value. Nevertheless, the excellent application of lysozyme detection using DNA aptamers bound to gold working electrodes was demonstrated using their potentiostat.

4.4. Measurement of Current

Having decided upon using MAX5250 (Maxim Integrated, Inc) for the DAC and the MCP3304 (Microchip Technologies, Inc.) for the ADC, the mode of current measurement and amplification is now considered. This aspect is arguably the most important facet of any potentiostatic circuit as noise introduced at this point will have ramifications seen in the voltammograms. Possibly the simplest way to measure current is to measure the voltage drop across a fixed resistor (current shunt) and knowing its value, apply Ohms law

to calculate the current. This method relies on an accurate and stable comparator amplifier as shown in Figure 4-12.

Figure 4- 12: Current shunt and comparator (left) and current follower op amp configuration also known as TIA (right). Both systems for current measurement.

The currents expected to be measured in an electrochemical can range from nA to mA so a fixed shunt resistor would provide problems with resolution at the lower end of the scale. This could be overcome by using a digipot, however the design is difficult to implement accurately so it was not considered further.

Another very simple technique is to use a current follower as described in Bard and Faulkner's instrumentation section (165), shown in Figure 4-12. Here the current flowing into the op amp (OA1) is met at the summing point by the current flowing through the feedback resistor (R1) to form the virtual ground. By application of Kirchhoff's and Ohm's laws the voltage output is directly proportional to the current by a factor of the value of the feedback resistor shown in Equation 4-4.

Equation 4-4: $E_{ADC} \approx -I_{WE} \times R_{R1}$

The voltage is negative as the signal goes into the inverting input and an approximately equals sign is used because the voltage is in fact also multiplied by the factor shown in Equation 4-5.

Equation 4-5: $\left(1 + \frac{1}{A_V}\right)$

Where A_V is the open loop gain of the op amp which typically takes a value of hundreds of thousands or greater thus the factor very closely approximates 1. Whilst this configuration is suitable for current measurement, it could be improved by the use of an instrumental amplifier configuration (210) as shown in Figure 4-13.

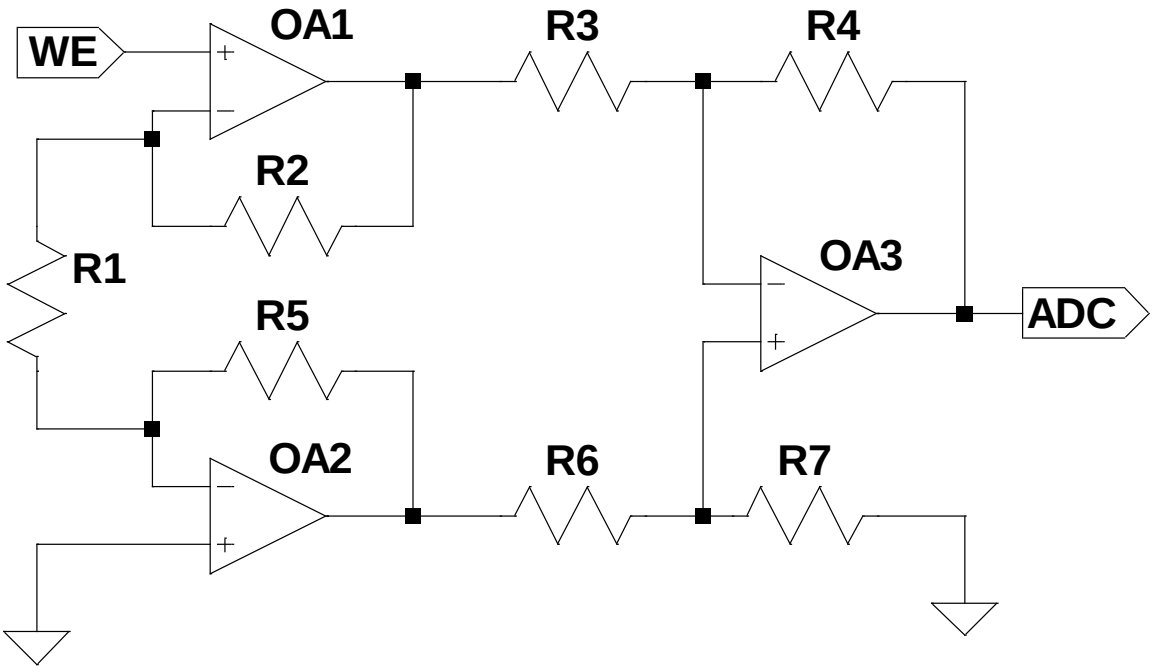


Figure 4- 13: Instrumental amplifier op amp configuration

This has three distinct op amps which working in conjunction in differential mode. The most common implementation uses equal values of R2 to R7 whilst R1 is variable to scale for differing current inputs. Taking the example value of 25 kΩ for resistors R2 to R7 the voltage gain of the instrumental amplifier is shown in Equation 4-6.

Equation 4-6: $$Gain = 1 + \frac{50000}{R1}$$

Instead of using the WE as the direct input to the instrumental amplifier, a current following op amp could be used to bridge between the electrode and the high gain amplifier. This general design was selected as the gain resistor (R1) could be replaced with a digipot providing scaling for current ranging. Many

instrumental amplifiers are available as a monolithic IC and the example chosen for the circuit for Model A is the INA114 (211).

This chip has the option for ground offset reference which is necessary for the present circuit design as the 5 V supply from the USB will be split into ± 2.5 V so anodic and cathodic current can be measured *i.e*. the offset will be to + 2.5 V. The INA114 also exhibits several advantages such as low noise (0.4 μV_{p-p}), high common-mode rejection (115 dB), and low input bias current (2 nA). These features and others have made it reliable for research into minimally invasive surgical tools (212), real time emotion detection electroencephalogram (EEG) systems (213), and eye movement detection interfaces (214) to name a few.

Considering the inverter op amp prior to the instrumental amplifier, a few specifications are of high importance when considering which op amp to choose; namely the bandwidth, input impedance, and drift. The bandwidth is defined as the frequency range over which the amplifier's gain does not fall below half of its maximum (-3 dB). The main ramification of this is the settling time it takes the amplifier to stabilise the output after a change in the input. By way of example a 100 kHz bandwidth op amp should operate well on a time scale of tens of μs. This period is satisfactory for an electrochemical instrument, but attention should be paid to actual slew rate. An ideal potentiostat has the characteristic of infinite input impedance and zero output impedance. Whilst this situation is not possible to achieve with modern op amps, the input impedance is commonly in excess of 1 TΩ again providing satisfactory conditions. Drift refers to how the gain of the op amp will change with changing temperature. This is not as important in electrochemical instruments which are normally conducted at room temperature.

The AD820 (215) was chosen as a general-purpose op amp in the circuit as it has junction gate field-effect transistor (JFET) inputs which has the advantage of very high input impedance. A summary of the AD820 parameters are shown in Table 4-2.

Table 4- 2: AD820 op amp specifications. Note that the slew rate of 3 V μs^{-1} is more than 1000 times the maximum expected sweep and pulse rate. However, the settling time to 0.01% of 1.8 µs (measured under ideal conditions) could have an influence on the signal.

Parameter	Value
Output swing	Rail to rail
Slew rate	3 V μs^{-1}
Settling time to 0.01%	1.8 µs
Unity-gain bandwidth	1.8 MHz
Input offset voltage	800 µV
Voltage drift	2 µV $^{o}C^{-1}$
Input bias current	25 pA
Noise	13 nV $Hz^{-½}$

4.5. Prototype Model A

The general design of prototype A takes the form of an Arduino "shield" *i.e.* a printed circuit board that plugs directly into an Arduino development board. The schematic is shown in Figure 4-14 and the board layout is given in Figures 4-15 and Figure 4-16. The schematics and Gerber files for manufacture were produced using the software Circuit Maker (Altium (216)). The bare PCB was manufactured by PCB Express (part of the Newbury Electronics group) and populated using through hole components in the laboratory.

4.5.1. General Working Principles

The prototype was designed to function as a simple form of Mealy finite state machine (217). That is there are four main functions (states) it will exist in during operation namely: main, CV, LSV and SWV.

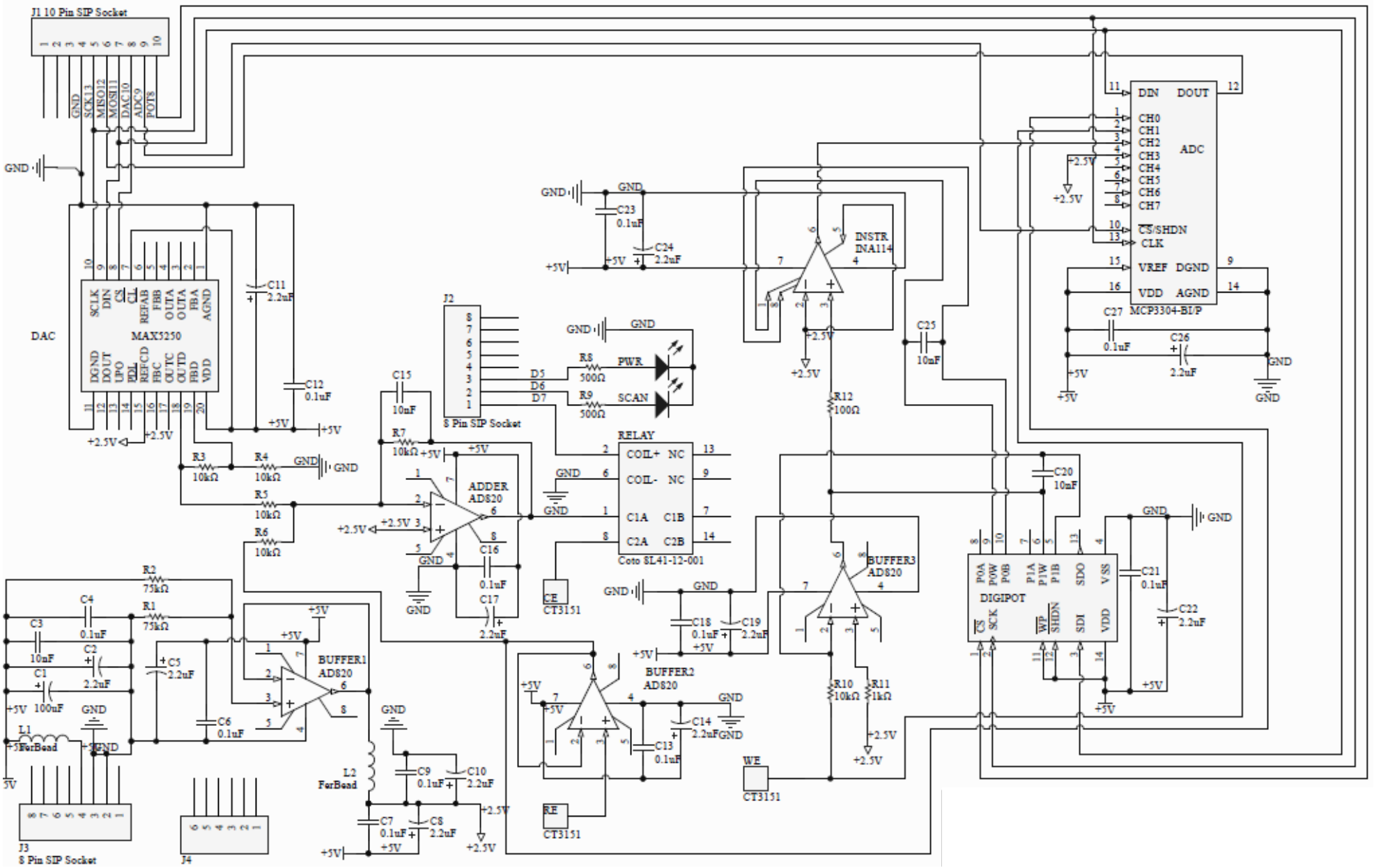


Figure 4- 14: Schematic for prototype A

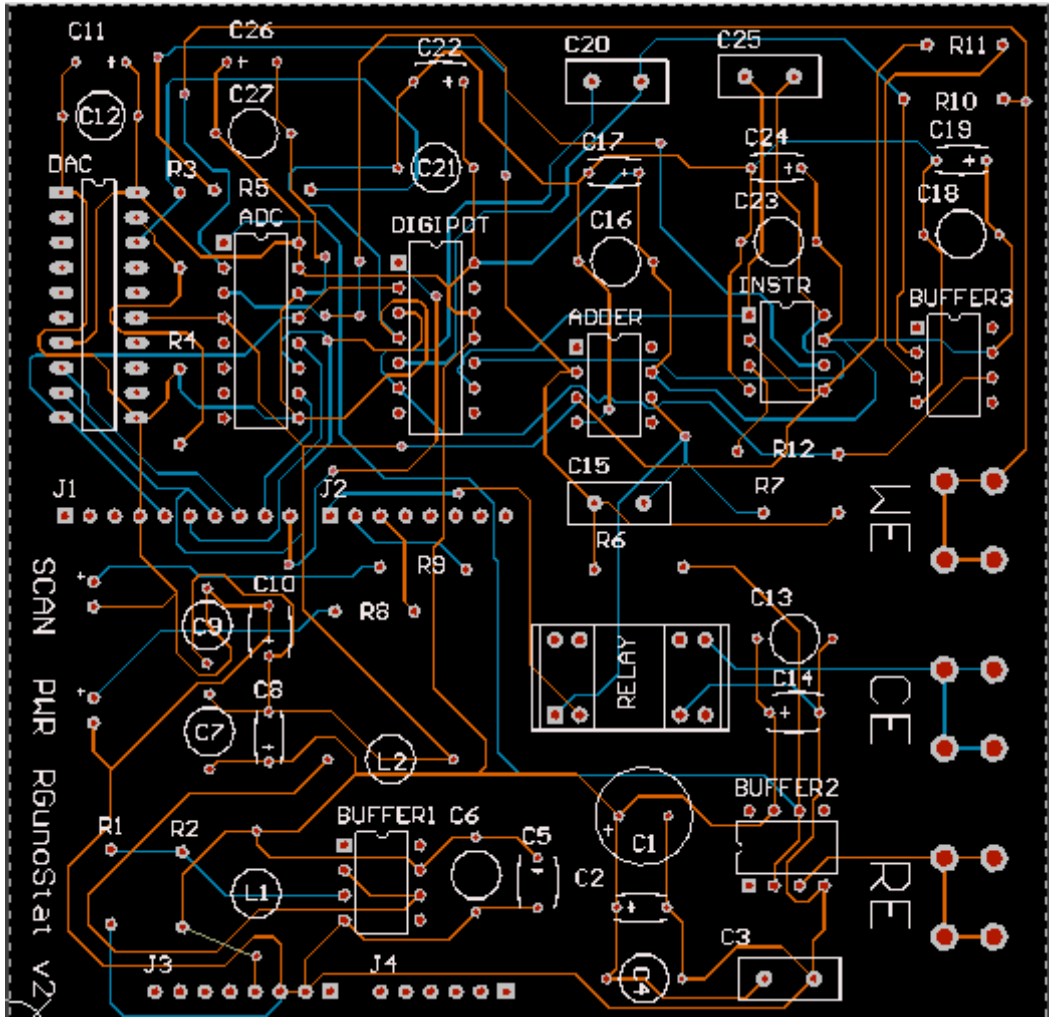


Figure 4- 15: Board layout for prototype A. Top layer traces shown in brown and bottom layer in blue.

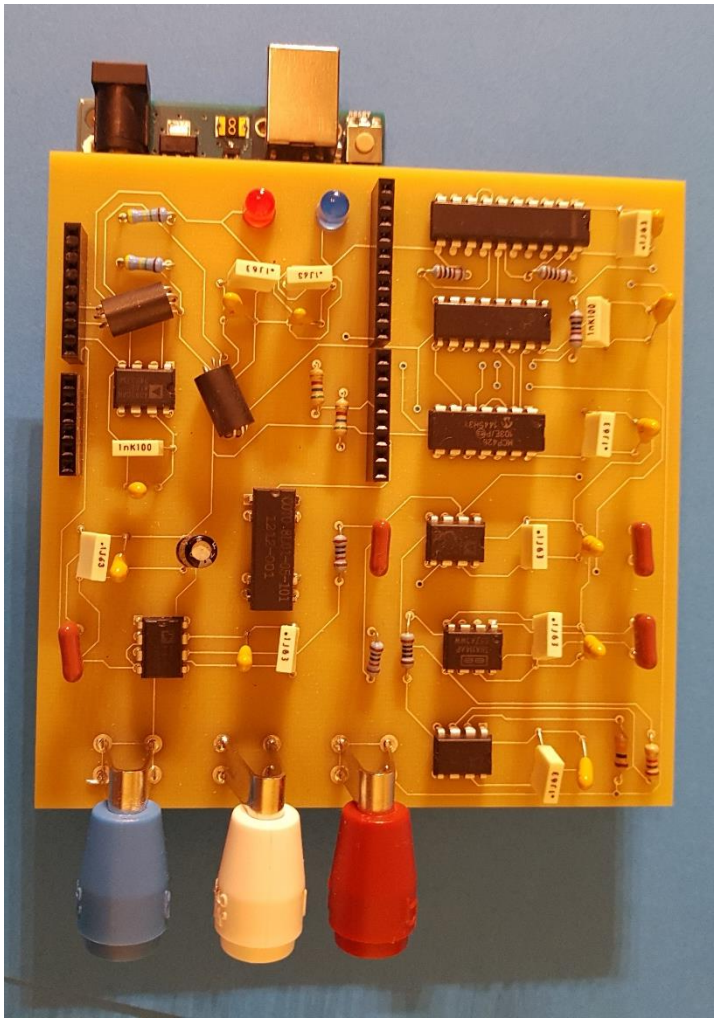


Figure 4- 16: Arduino Uno board partially visible under potentiostat at top of image. Image is rotated 90⁰ clockwise with respect to Figure 4.15

The "main" function serves only to hold the microcontroller in a waiting state to obtain the electrochemical method parameters (*e.g.* stop/start voltages) from the host PC prior to opening the correct method. During potential sweeps the starting voltage is obtained by sweeping the DAC until it is achieved. The DAC is then swept at the given scan rate until the switching potential is met and then the DAC sweep is reversed until back at the starting potential. Therefore, to state it explicitly, individual potential data points between the extremes are not expressly set by the potentiostat – the current is measured at those potentials which happen to be caused by incrementing the DAC. For pulsed methods, the following pseudo-code outlines the principle of operation for an anodic scan:

1. Find starting potential
2. Add amplitude + step potentials
3. Wait for given instrumental delay
4. Measure current
5. Subtract amplitude potential
6. Wait for given instrumental delay
7. Measure current
8. Repeat from step 2 until stop potential is reached

This procedure makes it very easy to produce the square waves for pulsed analysis. The delays in steps 3 and 6 are necessary to allow the non-Faradic processes to equilibrate as far as possible and are measured in ms. If these wait periods are equal the instrument performs SWV, however if step 6 is set to be longer than step 6 the waveform produced is that of differential pulsed voltammetry (DPV).

4.5.2. Design Notes on Protype A

Two LED's were added for basic debugging, one to indicate that the microcontroller had power and the other to notify when it was in scanning mode (CV, LSV or SWV). Capacitors C1 to C4 were added to decouple the 5 V supply from the USB. The particular types of capacitors used are shown in Table 4-3.

Table 4- 3: Capacitor types used in prototype A. The schematic reference is given in Figure 4- 14.

Schematic Reference	Capacitance (μF)	Type	Voltage rating (V)
C1	100	Aluminium	50
C2, C5, C8, C10, C11, C14, C17, C19, C22, C24, C26	2.2	Tantalum	16
C4, C6, C7, C9, C12, C13, C16, C18, C21, C23, C27	0.1	Ceramic	63
C3, C15, C20, C25	0.01	Ceramic	400

All ICs including the op amps were bypassed using 2.2 and 0.1 μF capacitors in parallel and 0.01 μF was used in parallel across the feedback loops of the op amps to increase stability. The 2.5 V reference was produced from the USB supply using a voltage divider of two 75 k Ω resistors (R1 and R2) which was buffered by an op amp (BUFFER1). The 5V supply and 2.5 V reference was also passed through ferrite beads to filter noise (L1 and L2). The 2.5 V reference is used as a signal ground for the analog part of the circuit. The ADC channels used are shown in Table 4-4.

Table 4- 4: MCP3304 (ADC) differential channel inputs. The circuit connection is shown in Figure 4- 14.

ADC	Circuit Connection	Purpose
CH0	Reference electrode (RE) buffered by op amp (BUFFER2)	CH1-CH0 for WE potential
CH1	Working electrode (WE) direct connection	
CH2	Instrumental amplifier output (INSTR)	CH2-CH3 for cell current
CH3	2.5 V reference	

The gain resistor for the INA114 instrumental amplifier (INSTR) was occupied by a 10 kΩ MCP4261 (DIGIPOT) in order to vary the gain according to Equation 4-6. The other channel of the digipot was connected to the feedback loop of the transimpedance amplifier (BUFFER3). However, this was maintained at 10 kΩ in order to produce a gain of 1 in conjunction with R10. A relay (Coto,

8L41, (218)) was added prior to the counter electrode to enable an on/off switch for the overall cell.

Two 10 kΩ resistors (R3 and R4) are necessary to provide full swing from the DAC and are linked to its internal op amp. Also, two 10 kΩ resistors are connected to the summing point of ADDER for the DAC and reference electrode (R5 and R6 respectively).

4.5.3. Firmware

The complete source code that runs the microcontroller on the Arduino Uno development board is provided here in code block textboxes. The pre-processor section contains two header files to facilitate the serial peripheral interface and the ADC as shown in Code Block 4-1. The argument of the MCP3304 adc1 object, "9", refers to the slave select pin for the ADC chip.

```
#include <SPI.h>
#include <MCP3304.h>
MCP3304 adc1 (9);
```

Code Block 4- 1: Pre-processor header files for prototype A firmware

Most variables were stored globally and are shown in Code Blocks 4-2 and 4-3. Throughout the code blocks comments are provided which are preceded by "//" to explain variables or procedures.

```
// "measureNumber" is the number of times the current and voltage will be
// sampled before an average is taken
const int measureNumber = 1000;

// pulsed method parameters where the "Time" variables relate to the
// deposition step in SWV given in ms and the "potential" variables are 10-bit
// translations from 0 to 5 V
int stirrerTime, equilTime, depPotential, pulseHeight;

// variables relating to the digipot, "wiper" variables denote the address i.e.
// which channel to use and "level" is an 8-bit scale from approximately 39 Ω
// to 10 kΩ
const byte INA114wiper = 0;
const byte AD820wiper = 16;
byte INA114level, AD820level;
```

Code Block 4- 2: Global variables for prototype A firmware (part 1 of 2)

```
// variables relating to the 13-bit ADC values of the measurements
int WEanodic, WEcathodic, WEvoltageMeasure, WEvoltage;
int WEcurrentMeasure, WEcurrent;
long WEvoltageTally, WEcurrentTally;
// "DACpoint" represents a single increment on the 10-bit DAC
const float DACpoint = 4.8828;
// variables measured in ms relating to scan rate
unsigned long timer, wholeTimer;
float postWaitFloat;
unsigned long postWait;
// constants for the microcontroller pin values for each of the components
const int ADCselect = 9;
const int DACselect = 10;
const int POTselect = 8;
const int POWERledpin = 5;
const int SCANledpin = 6;
const int RELAYpin = 7;
// to store the actual voltage parameters set by the user
float beginning, ending, deposition;
// "numScans": Number of scans, "preWait": after E applied wait before
// measuring I "preWait", "start" and "finish": 10-bit representation of
// start and end of E sweep
int numScans, preWait, start, finish, rate, stepSize;
// variables to store the 10-bit values to send to DAC over SPI.  "fulldac" is
// the actual value which ranges from 61440 (binary 1111000000000000)
// representing 0 V to 65532 (binary1111111111111100) representing 5V.
// However, it must be sent using 8-bit words so "dac1" and "dac2" are used
// to split it into a high and low byte
unsigned int fulldac;
int dac1, dac2;
// the host PC will send the method parameters to the microcontroller as a
// single string where the values are separated by a comma.  The character
// arrays and strings here are used to parse the host PC input to separate out
// the terms so they can be converted to their numerical values
// "fb1S" and "fb2S" refer to the digipot values which are being used as a
// feedback and gain resistor for the WE current amplification
const char s[2] = ",";
char str[80];
char *token[13];
int i = 0;
int values[13];
String strings[13];
String methodS, startS, endS, rateS, cyclesS, depPotS, depTimeS, equiTimeS,
pulseS, stepS, fb1S, fb2S;
```

Code Block 4- 3: Global variables for prototype A (part 2 of 2)

The setup() function only runs once when the microcontroller is first powered up in order to set the registers of the input/output pins and initialise

any peripherals (*e.g.* SPI). A BAUD rate of 250000 bps was used, a 10 ms pre-measurement wait is set, and the SPI protocol is set to most significant bit first. The setup function is given in Code Block 4-4.

```
void setup()
{
        // Set direction of digital pins
        pinMode (ADCselect, OUTPUT);
        pinMode (DACselect, OUTPUT);
        pinMode (POTselect, OUTPUT);
        pinMode (POWERledpin, OUTPUT);
        pinMode (SCANledpin, OUTPUT);
        pinMode (RELAYpin, OUTPUT);
        // Initiate SPI
        Serial.begin (250000);
        SPI.setBitOrder (MSBFIRST);
        SPI.begin();
        preWait = 10;
}
```

Code Block 4- 4: Initialisation setup() function for prototype A firmware

The Arduino IDE uses a loop() function where an infinite loop is included by the compiler, although not visible in the code. Code Blocks 4-5, 4-6 and 4-7 provide the loop() function along with accompanying comments.

```
void loop()
{
        // Turn cell off by disconnecting the relay, turn power LED on but
        // scanning LED off, set slave select pins for DAC, ADC and digipot to
        // high i.e. so the chips are not engaged
        digitalWrite (RELAYpin, LOW);
        digitalWrite (POWERledpin, HIGH);
        digitalWrite (SCANledpin, LOW);
        digitalWrite (DACselect, HIGH);
        digitalWrite (POTselect, HIGH);
        digitalWrite (ADCselect, HIGH);
        // set the DAC to 2.5 V (midway point)
        fulldac = 63488;
        dac1 = highByte (fulldac);
        dac2 = lowByte (fulldac);
        applyDAC (dac1, dac2);
        i = 0;
```

Code Block 4- 5: Main loop() function for prototype A firmware (part 1 of 3)

```
// wait for the host PC to send method parameters string
while (Serial.available () > 0)
{
        // once string is received store and then parse into separate parts
        // shown as "token" and store the values in an array
        str [i] = char (Serial.read ());
        i++;
        delay (100);
}
if (i > 0)
// Method data received from PC so break into separate string values
// store as string values prior to conversion to numerical values
{
        i = 0;
        token[i] = strtok (str, s);
        while (token[i] != NULL)
        {
                i++;
                token[i] = strtok (NULL, s);
         }
        for (i = 0; i < 13; i++)
        {
                strings[i] = ((char*)token[i]);
        }
        for (i = 0; i < 13; i++)
        {
                values[i] = strings[i].toInt();
        }
}
// convert the array strings with parameter names
methodS = strings [0];
startS = strings [1];
endS = strings [2];
rateS = strings [3];
cyclesS = strings [4];
depPotS = strings [5];
depTimeS = strings [6];
equiTimeS = strings [7];
pulseS = strings [8];
stepS = strings [9];
fb1S = strings [10];
fb2S = strings [11];
```

Code Block 4- 6: Main loop() function for prototype A firmware (part 2 of 3)

```
        // convert the method parameters to numerical values
        beginning = startS.toFloat();
        ending = endS.toFloat();
        rate = rateS.toInt();
        numScans = cyclesS.toInt();
        deposition = depPotS.toFloat();
        stirrerTime = depTimeS.toInt ();
        equilTime = equiTimeS.toInt();
        pulseHeight = pulseS.toInt();
        stepSize = stepS.toInt();
        AD820level = fb1S.toInt ();
        INA114level = fb2S.toInt ();
        // set the method specified levels of gain for current measurement
        applyDIGIPOT (AD820level, INA114level);
        // run appropriate method function (all variables are global)
        if (methodS == "CV")
        {
                CV ();
        }
        else if (methodS == "SWV")
        {
                SWV ();
        }
        else if (methodS == "LSV")
        {
                LSV ();
        }
}
```

Code Block 4- 7: Main loop() function for prototype A firmware (part 3 of 3)

The function to set the gain for the WE's transimpedance amplifier (AD820) and instrumental amplifier (INA114), namely applyDIGIPOT() is shown in Code Block 4-8. It should be noted that in practice the AD820 level was held at 255 *i.e.* 10 kΩ to maintain a gain of 1 at this stage. However, the functionality was maintained in case excessive currents were measured and it was necessary to reduce the signal to avoid saturation. A further benefit of using a digipot to affect the gain is that it can be adjusted while the system is running in order to provide maximum resolution whilst avoiding saturation of the ADC *i.e.* extremes of 0 or 5 V.

```
void applyDIGIPOT (int AD820level, int INA114level)
{
        // SPI protocol requires the slave select pin to be low prior to
        // communication.  Two bytes are sent to the MCP4261 in each transfer
        // (wiper and level), the first to provide the address of the channel
        // and the second will set the resistance with 0 representing ~ 39 Ω and
        // 255 representing 10 kΩ.  The levels are received as arguments from
        // the calling function.
        digitalWrite (POTselect, LOW);
        SPI.transfer (AD820wiper);
        SPI.transfer (AD820level);
        digitalWrite (POTselect, HIGH);
        digitalWrite (POTselect, LOW);
        SPI.transfer (INA114wiper);
        SPI.transfer (INA114level);
        digitalWrite (POTselect, HIGH);
}
```

Code Block 4- 8: Gain setting applyDIGIPOT() function for prototype A firmware

Next the CV() function is detailed in Code Blocks 4.9, 4.10 and 4.11. The function has some initialising calculations followed by two sections which deal separately with cathodic and anodic sweeps. CV() calls the other functions outRange(), getInitial(), and applyCE() are given in Code Blocks 4.12, 4.13 and 4.14 respectively.

```
void CV ()
{
        // set index of string array of method parameters to null to enable it to
        // be filled by new method parameters for subsequent scans
        strings[0] = "";
        // to facilitate faster scan rates, the step size increments that are
        // applied to the DAC must be increased.  "rate" is an integer value in
        // mV s-1.  The total number of data points is sacrificed to achieve the
        // appropriate rate.
        stepSize = (rate / 100) + 1;
        stepSize = stepSize * 8;
        // to ensure accuracy in scan rate, the DAC will not be incremented
        // until a delay time has passed which calculated here in ms first as a
        // floating point, then converted to an integer value
        postWaitFloat = stepSize * DACpoint * 1000.0 / (2.4 * rate);
        postWait = postWaitFloat;
        ending = (ending * 4096.0 / 5.0);
        // convert switching potentials to integer values
        start = beginning;
        finish = ending;
```

Code Block 4- 9: 8 Initial calculations in cyclic voltammetry function CV() for prototype A firmware (part 1 of 3)

```
// this section runs if initial sweep of CV is cathodic
if (start < finish)
{
        // find the correct DAC value to match the user defined starting E
        getInitial (start);
        // a delay is added to allow non-Faradic processes to equilibrate
        // further
        delay (250);
        // turn on scanning indicator LED
        digitalWrite (SCANledpin, HIGH);
        // "wholeTimer" marks the beginning of the analysis time in ms
        // used for scan rate determination
        wholeTimer = millis ();
        // "numScans" is the user defined number of cycles in the CV
        // given as an outer "for" loop and "while" loops are used to
        // perform the DAC sweeps down then up
        for (int runNum = 0; runNum < numScans; runNum++)
        {
                // keep sweeping DAC down until the user defined
                // switching potential is reached
                while (finish > WEvoltage)
                {
                        // "timer" marks the beginning of a single data point
                        // measurement in ms
                        timer = millis ();
                        // increment the DAC and check that this does not
                        // take the value to send outside the range which
                        // it can apply i.e. the compliance voltage, then
                        // physically apply the value to the counter electrode
                        // note "stepSize" is being added making the DAC
                        // output more positive, however the ADDER
                        // amplifier inverts the potential prior to the counter
                        // electrode thus providing cathodic sweep.
                        fulldac = fulldac + stepSize;
                        if (fulldac >= 65532) outRange ();
                        applyCE (fulldac);
                }
                // once switching potential has been reached sweep back
                // down to starting potential
                while (start < WEvoltage)
                {
                        timer = millis ();
                        fulldac = fulldac - stepSize;
                        if (fulldac <= 61440) outRange ();
                        applyCE (fulldac);
                }
        }
```

Code Block 4- 10: Code for cyclic voltammetry if first sweep is cathodic in function CV() for prototype A firmware (part 2 of 3)

```
                // disconnect cell and turn of scan indicator LED
                digitalWrite (RELAYpin, LOW);
                digitalWrite (SCANledpin, LOW);
        }
        // this section runs if initial sweep of CV is anodic
        else if (start > finish)
        {
                getInitial (start);
                delay (250);
                digitalWrite (SCANledpin, HIGH);
                wholeTimer = millis ();
                for (int runNum = 0; runNum < numScans; runNum++)
                {
                        while (finish < WEvoltage)
                        {
                                timer = millis ();
                                fulldac = fulldac - stepSize;
                                if (fulldac <= 61440) outRange ();
                                applyCE (fulldac);
                        }
                        while ((start) >= WEvoltage)
                        {
                                timer = millis ();
                                fulldac = fulldac + stepSize;
                                if (fulldac >= 65532) outRange ();
                                applyCE (fulldac);
                        }
                }
                digitalWrite (RELAYpin, LOW);
                digitalWrite (SCANledpin, LOW);
        }
        // notify host PC that scan has finished in order to save data as comma
        // separated value file
        Serial.print (9999);
        Serial.print (',');
        Serial.print (9999);
        Serial.println (',');
}
```

Code Block 4- 11: Code for cyclic voltammetry if first sweep is anodic in function CV() for prototype A firmware (part 3 of 3)

The outRange() function is called in situations where the potentiostat has reached its compliance voltage limit. It switches the cell off, notifies the host PC this has occurred by sending "7777" and returns to the main loop() function to await the new method parameters. outRange() is detailed in Code Block 4-12.

```
void outRange ()
{
        digitalWrite (RELAYpin, LOW);
        digitalWrite (SCANledpin, LOW);
        Serial.print (7777);
        Serial.print (',');
        Serial.print (7777);
        Serial.println (',');
        delay (5000);
        loop ();
}
```

Code Block 4- 12: In cases where the compliance voltage is exceeded the outRange() function will disconnect the cell for prototype A firmware

The getInitial() function is used to set the DAC such that the potential applied to the cell produces the user specified starting voltage between the working and reference electrodes and is shown in Code Block 4-13. A local variable "commence" was used as the integer starting value, however as "fulldac" is global there is no need to return any values. The function getWEvoltage() is also called here and is defined in Code Block 4-15.

The applyCE() function, in Code Block 4-14, sets the DAC to the value it receives as "fulldac" (by calling the separate function applyDAC() shown in Code Block 4-16), then measures the WE potential (versus RE) and cell current again using a separate function call, measure(), shown in Code Block 4-18.

The differential voltage between channels 1 and 0 on the ADC measures the WE *vs.* the RE potential and is accomplished by the getWEvoltage() function, shown in Code Block 4-15; an integer value is returned to the original calling method function.

The serial peripheral interface (SPI) communication with the DAC is handled by the applyDAC() function shown in Code Block 4.16. The purpose here is to influence the CE potential and in so doing affect the electric field of the whole cell.

```
void getInitial (int commence)
{
	// switch cell on
	digitalWrite (RELAYpin, HIGH);
	// use ADC to measure the potential difference between WE and RE
	WEvoltage = getWEvoltage ();
	// sweep DAC until ADC measures the user specified starting voltage
	// either up or down depending on difference between present value
	// and desired value
	while (commence > WEvoltage)
	{
		fulldac = fulldac + 4;
		if (fulldac >= 65532) outRange ();
		dac1 = highByte (fulldac);
		dac2 = lowByte (fulldac);
		applyDAC (dac1, dac2);
		delay (20);
		WEvoltage = getWEvoltage ();
	}
	while (commence < WEvoltage) {
		fulldac = fulldac - 4;
		if (fulldac <= 61440) outRange ();
		dac1 = highByte (fulldac);
		dac2 = lowByte (fulldac);
		applyDAC (dac1, dac2);
		delay (20);
		WEvoltage = getWEvoltage ();
	}
}
```

Code Block 4- 13: Align potentiostat WE potential with user defined starting value using the getInitial() function for prototype A firmware

```
void applyCE (int fulldac)
{
	// potential applied to CE, DAC value calculated elsewhere (scan loop)
	dac1 = highByte (fulldac);
	dac2 = lowByte (fulldac);
	applyDAC (dac1, dac2);
	WEvoltage = getWEvoltage ();
	measure (WEvoltage);
}
```

Code Block 4- 14: After each DAC increment as part of the method functions (CV() etc.), its new value is applied using the applyCE() function, then voltage is measured for prototype A firmware

```
int getWEvoltage ()
{
        // a summing tally is used so a more accurate voltage can be obtained
        // by taking the average of many measurements (specified by
        // "measureNumber").  Reset tally to zero prior to each measurement.
        // The MCP3304 adc1 object enables the "readDiff()" function.
        WEvoltageTally = 0;
        for (int i = 0; i < measureNumber; i++)
        {
                WEvoltageMeasure = adc1.readDiff (1);
                WEvoltageTally += WEvoltageMeasure;
        }
        WEvoltage = WEvoltageTally / measureNumber;
        return (WEvoltage);
}
```

Code Block 4- 15: WE vs RE potential is measured using the getWEvoltage() function for prototype A firmware

```
void applyDAC (int dac1, int dac2)
{
        digitalWrite (DACselect, LOW);
        SPI.transfer (dac1);
        SPI.transfer (dac2);
        digitalWrite (DACselect, HIGH);
}
```

Code Block 4- 16: The block forces the CE to the new potential using the applyDAC() function for prototype A firmware

The measure() function shown in Code Block 4-18 is used to gather the data together and send it to the host PC. This function calls two others, firstly testRange() (Code Block 4-17) which is used to ensure the ADC inputs are greater than 0.78 V and less than 4.22 V. If any ADC input exceeds these values, the instrumental amplifier gain is reduced by increasing the resistance of the digipot.

The allowable range (or compliance voltage) does not extend to 0 and 5 V due to "clipping" of the operational amplifiers whereby they are not able to amplify signals that approach the rails (which in this circuit are themselves 0 and 5 V). The second function called by measure() is getWEcurrent() and this is shown in Code Block 4-19.

```
void testRange ()
{
        // ensure ADC inputs are > 0.78 V and < 4.22 V by changing feedback
        // resistance to instrumental amplifier
        if (WEcurrent > 1410 || WEcurrent < -1480)
        {
                INA114level += 3;
                if (INA114level >= 255) INA114level = 255;
                applyDIGIPOT (255, INA114level);
                // inform host PC of change in gain for current calculation
                Serial.print (4444);
                Serial.print (',');
                Serial.print (4444);
                Serial.println (',');
        }
}
```

Code Block 4- 17: Ensure the amplifiers do not saturate the ADC using the testRange() function for prototype A firmware

```
void measure (int WEvoltage)
{
        // allow non-Faradic processes to equilibrate before measurement then
        // wait until "postWait" time has passed to maintain correct scan rate
        delay (preWait);
        WEcurrent = getWEcurrent ();
        Serial.print (WEvoltage);
        Serial.print (',');
        Serial.print (WEcurrent);
        Serial.println (',');
        testRange ();
        while (millis () < (timer + postWait)) {  }
}
```

Code Block 4- 18: Send 12-bit versions of voltage and current to host PC using the measure() function for prototype A firmware

The getWEcurrent() function uses a similar strategy to the voltage measurement *i.e.* averaging across many values to reduce noise, as in general terms the signal to noise ratio is directly proportional to the square root of the number of samples. The "signal ground" value of 2.5 V is automatically subtracted from the current measurement due to the differential mode of the ADC. A functional flowchart, to provide an overview of the operation is given in Figure 4-17.

```
int getWEcurrent ()
{
        WEcurrentTally = 0;
        for (int i = 0; i < measureNumber; i++)
        {
                WEcurrentMeasure = adc1.readDiff (2);
                WEcurrentTally += WEcurrentMeasure;
        }
        return (WEcurrentTally/measureNumber);
}
```

Code Block 4- 19: The cell current is measured using the getWEcurrent() function and returned to the measure() function for prototype A firmware

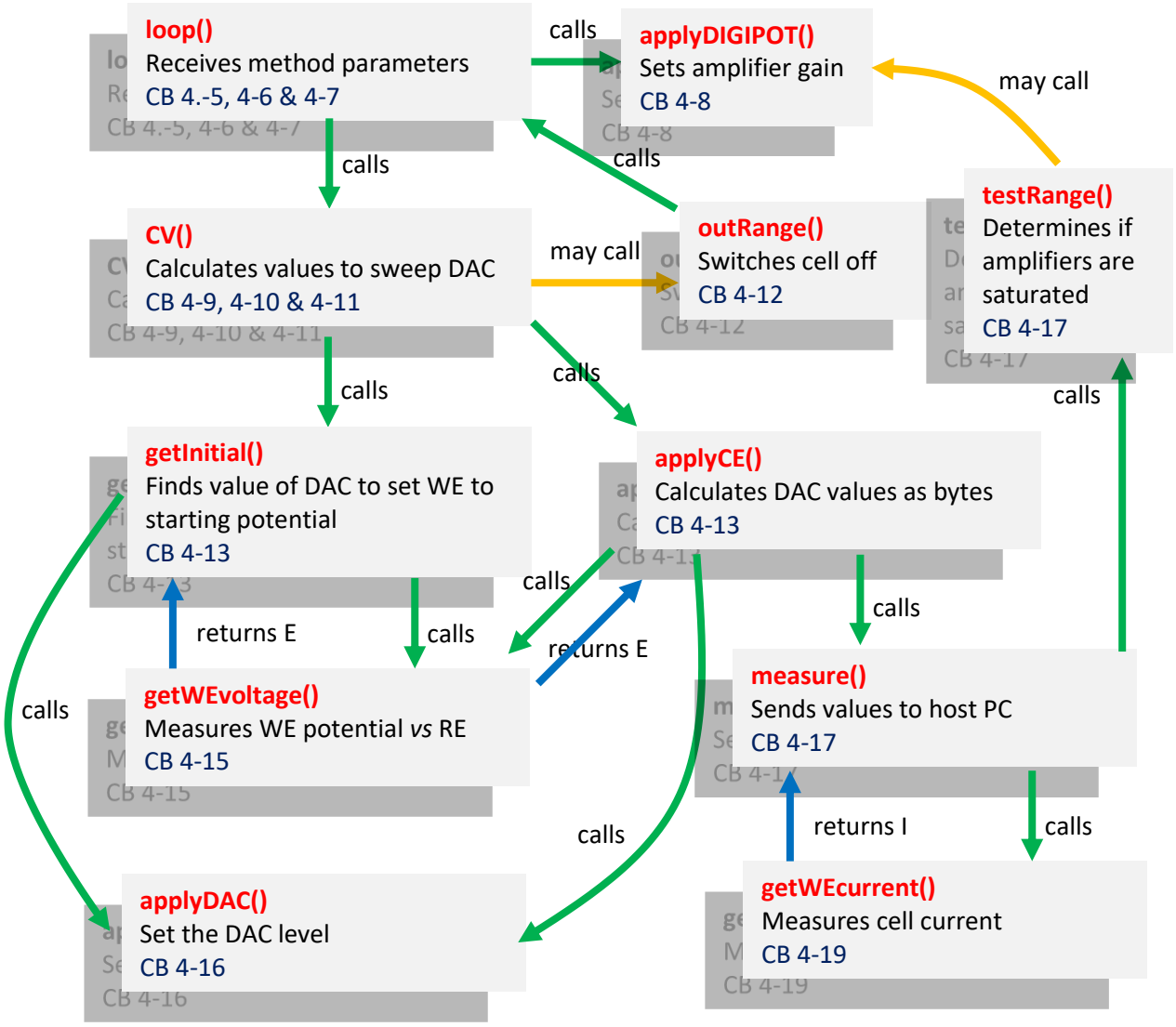


Figure 4- 17: Overview of firmware functions for cyclic voltammetry for prototype A. The return values of voltage (E) and current (I) are indicated with blue arrows. The code block numbers are given underneath a brief description of the purpose of the function.

The remaining code blocks for the other two techniques (LSV and SWV) function in a similar way to who CV operates so less intervening text is provided. SWV is in code blocks 4-20 to 4-22 and LSV in 4-88 to 4-88.

```
void SWV ()
{
        strings[0] = "";
        // convert deposition time values from s to ms
        stirrerTime *= 1000;
        equilTime *= 1000;
        // lowest two bits are not used in the DAC all values are multiplied by 4
        pulseHeight *= 4;
        stepSize *= 4;
        beginning = (beginning * 4096.0 / 5.0);
        ending = (ending * 4096.0 / 5.0);
        deposition = (deposition * 4096.0 / 5.0);
        start = beginning;
        finish = ending;
        depPotential = deposition;
        applyDIGIPOT (AD820level, INA114level);
        getInitial (depPotential);
        // Communicate his host PC at correct deposition E
        // Indicate to user to switch stirrer on
        Serial.print (2222);
        Serial.print (',');
        Serial.print (2222);
        Serial.println (',');
        delay (stirrerTime);
        // Indicate to user to switch stirrer off
        Serial.print (3333);
        Serial.print (',');
        Serial.print (3333);
        Serial.println (',');
        delay (equilTime);
        Serial.print (5555);
        Serial.print (',');
        Serial.print (5555);
        Serial.println (',');
        getInitial (start);
        digitalWrite (SCANledpin, HIGH);
```

Code Block 4- 20: Square Wave Voltammetric function (1 of 3) deposition stage

```
        // use SWVdown() if cathodic sweep
        if (start > finish) SWVdown (pulseHeight, stepSize);
        else
        {
                // use this code if anodic sweep
                while (WEvoltage < finish)
                {
                        fulldac = fulldac + (pulseHeight + stepSize);
                        if (fulldac >= 65532) outRange ();
                        dac1 = highByte (fulldac);
                        dac2 = lowByte (fulldac);
                        applyDAC (dac1, dac2);
                        WEvoltage = getWEvoltage ();
                        Serial.print (WEvoltage);
                        Serial.print (',');
                        delay (preWait);
                        WEanodic = getWEcurrent ();
                        fulldac = fulldac - pulseHeight;
                        dac1 = highByte (fulldac);
                        dac2 = lowByte (fulldac);
                        applyDAC (dac1, dac2);
                        delay (preWait);
                        WEcathodic = getWEcurrent ();
                        WEcurrent = WEanodic - WEcathodic;
                        Serial.print (WEcurrent);
                        Serial.println (',');
                }
        }
        // switch off power to electrode and inform PC scan is finished
        digitalWrite (RELAYpin, LOW);
        Serial.print (9999);
        Serial.print (',');
        Serial.print (9999);
        Serial.println (',');
}
```

Code Block 4- 21: Square Wave Voltammetric function (2 of 3) anodic pulsed sweep

A separate function call of SWVdown() was used for the cathodic pulsed sweep as shown in code block 4-22. This was introduced at the time for the simplicity, however, as is the case through most of this first Arduino iteration, more compact code could have been created. Nevertheless, it does serve to illustrate plainly the separate function that the device is undertaking.

```
void SWVdown (int pulseHeight, int stepSize)
{
        while (WEvoltage > finish)
        {
                fulldac = fulldac - (pulseHeight + stepSize);
                if (fulldac <= 61440) outRange ();
                dac1 = highByte (fulldac);
                dac2 = lowByte (fulldac);
                applyDAC (dac1, dac2);
                WEvoltage = getWEvoltage ();
                Serial.print (WEvoltage);
                Serial.print (',');
                delay (preWait);
                WEanodic = getWEcurrent ();
                fulldac = fulldac + pulseHeight;
                dac1 = highByte (fulldac);
                dac2 = lowByte (fulldac);
                applyDAC (dac1, dac2);
                delay (preWait);
                WEcathodic = getWEcurrent ();
                WEcurrent = WEanodic - WEcathodic;
                Serial.print (WEcurrent);
        }
        digitalWrite (RELAYpin, LOW);
        Serial.print (9999);
        Serial.print (',');
        Serial.print (9999);
        Serial.println (',');
}
```

Code Block 4- 22: Square Wave Voltammetric function (3 of 3) cathodic pulsed sweep

At this stage, the method parameters were sent to and received from the potentiostat using the Arduino Serial Monitor. The data was exported to Microsoft Excel and stored as csv files. Initial experiments verified the potentiostat was working; conducted on a dummy cell shown in Figure 4-18.

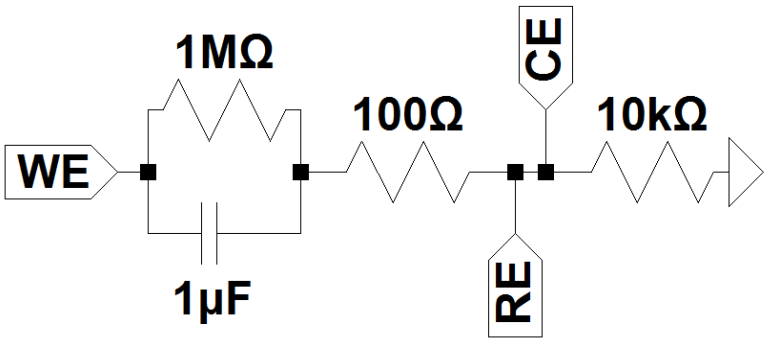


Figure 4- 18: Dummy cell for current measurement verification

```
void LSV ()
{
	strings[0] = "";
	applyDIGIPOT (AD820level, INA114level);
	stepSize = (rate / 100) + 1;
	postWaitFloat = stepSize * DACpoint * 1000.0 / (2 * rate);
	postWait = postWaitFloat;
	stepSize = stepSize * 4;
	beginning = (beginning * 4096.0 / 5.0);
	ending = (ending * 4096.0 / 5.0);
	start = beginning;
	finish = ending;
	if (start < finish)
	{
		getInitial (start);
		delay (250);
		digitalWrite (SCANledpin, HIGH);
		while (finish > WEvoltage)
		{
			timer = millis ();
			fulldac = fulldac + stepSize;
			if (fulldac >= 65532) outRange ();
			applyCE (fulldac);
		}
	}
	if (start > finish)
	{
		getInitial (start);
		delay (250);
		digitalWrite (SCANledpin, HIGH);
		while (finish < WEvoltage)
		{
			timer = millis ();
			fulldac = fulldac - stepSize;
			if (fulldac <= 61440) outRange ();
			applyCE (fulldac);
		}
	}
	digitalWrite (SCANledpin, LOW);
	digitalWrite (RELAYpin, LOW);
	Serial.print (9999);
	Serial.print (',');
	Serial.print (9999);
	Serial.println (',');
}
```

Code Block 4- 23: Linear sweep function for prototype Model A

4.5.4. Preliminary Results

The importance of the capacitors for decoupling and stabilisation of the op amp feedback loops is illustrated in Figure 4-19. The LSV data here was run on the dummy cell using a bread board setup of prototype so components could be removed or added. The figure on the left, without the decoupling capacitors, illustrates the heteroscedastic noise to a greater extent than if they are included. The summary data is given in Table 4-5.

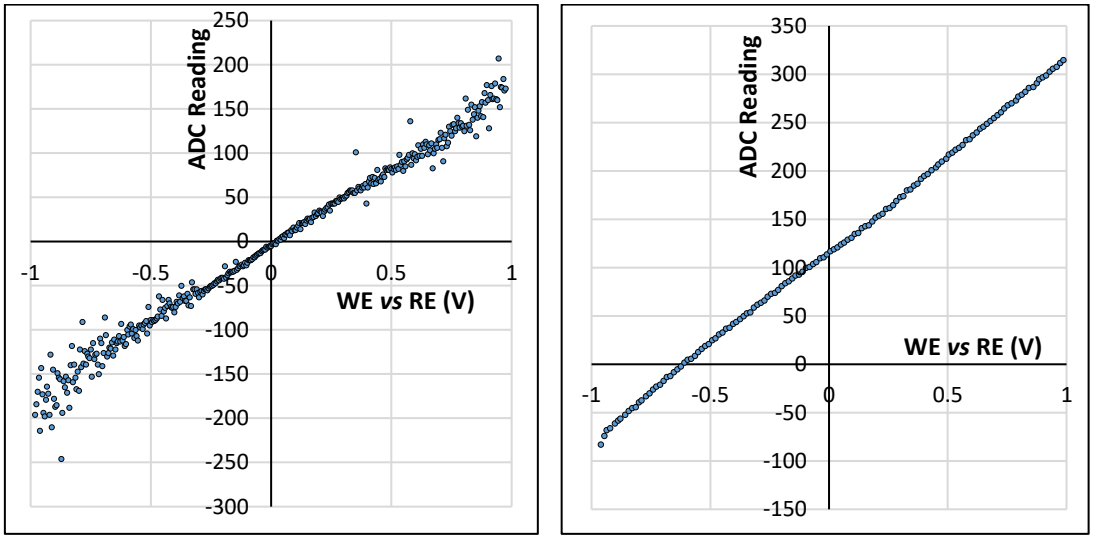


Figure 4- 19: LSV of dummy cell (Figure 4- 18) at a scan rate of 100 mV s^{-1}. Without decoupling and feedback capacitors (left) with capacitors (right).

It should be noted that the data on the y-axis has arbitrary units as it is the raw ADC reading (*i.e.* the device has yet to be calibrated).

Table 4- 5: Summary data from noise test on prototype Model A

Setup	R^2	SSR
Without Capacitors	0.9866	50476
With Capacitors	0.9993	1228

Whilst there are significant improvements in both the square of the Pearson correlation coefficient and the sum of the square of the residuals, there is a commensurate skewing of the signal to more positive readings for the

current when the capacitors are included. This is due entirely to the 10 nF ceramic capacitors that were in parallel with the feedback loops of the op amps. It was deemed that the benefits of the increased stability of the analog front end outweighed the skewing of the results. This was because at a later stage the signals could be translated back to their correct values through calibration. The advantages of having the variable gain system using the 10 kΩ digipot (MCP4261) as the drop resistors across the instrumental amplifier (INA114) was seen immediately. Setting this to as low a value as possible - without causing ADC saturation – improved resolution of the signal. Through trial and error of the settings satisfactory results were eventually obtained (for this stage) when analysis was conducted on ferricyanide solutions using a graphite WE, a platinum sheet CE and an Ag|AgCl reference electrode (all Metrohm, Autolab). The results are shown in Figures 4-20 and 4-21.

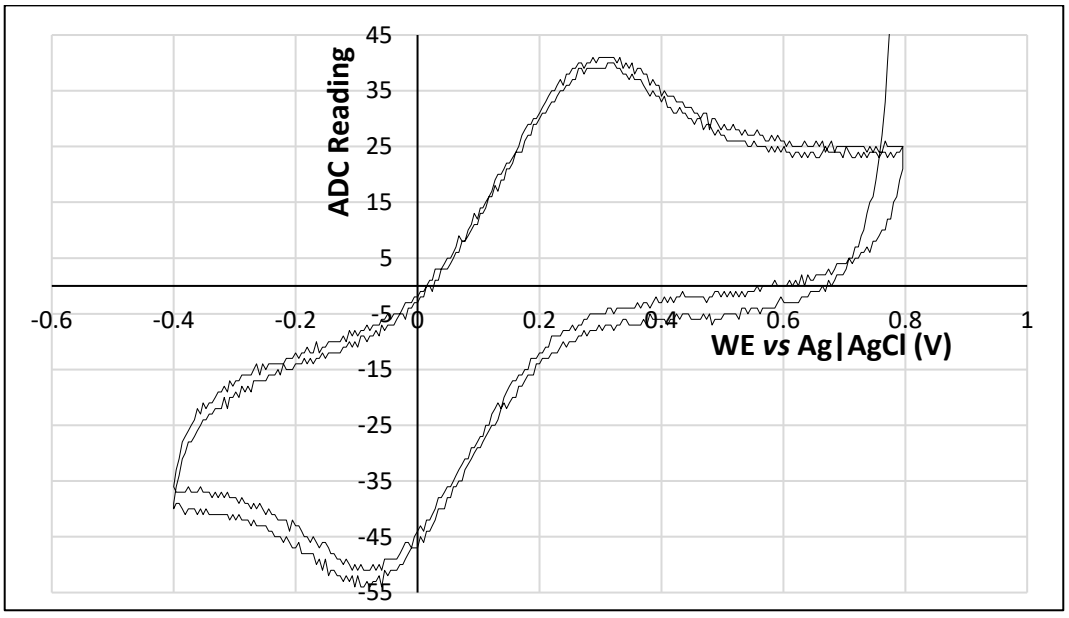


Figure 4- 20: CV data of 10 mM $K_3Fe(CN)_6$ using carbon WE; 2 cycles run from 0.8 V to -0.4 V and back at 100 mV s^{-1}.

There is a fair degree of noise visible in the CV shown in Figure 4-20, however, more significantly, the peak to peak separation was larger than expected. Nevertheless, a reasonable symmetrical scan shape was visible.

Noise was also visible in the SWV calibration that was run on a range of ferricyanide concentrations.

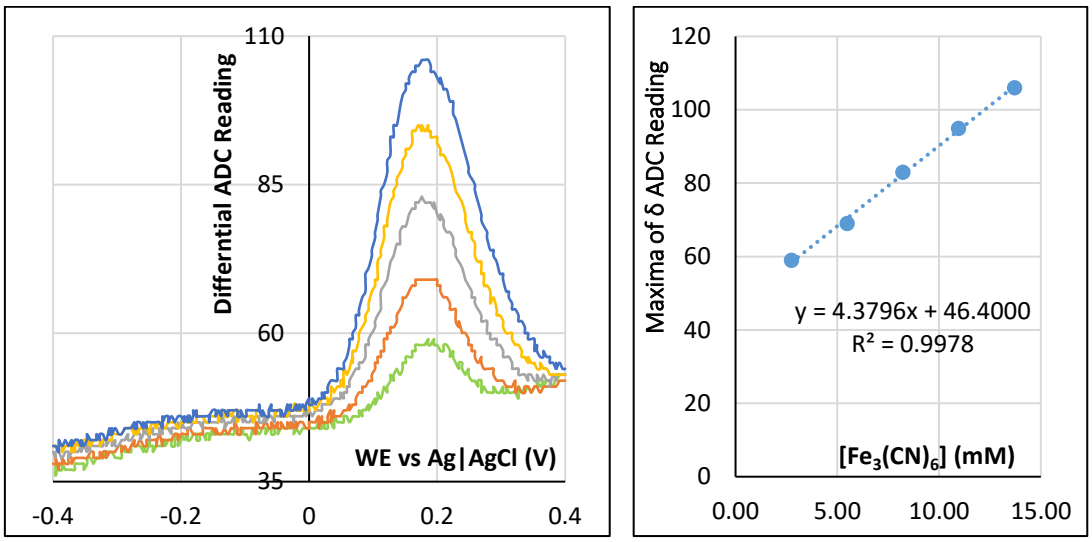


Figure 4- 21: SWV of a range of ferricyanide concentrations (left) regression curve of maximum datapoints (right) using carbon WE in prototype A

The SWV method run in Figure 4-21 involved a 60 s deposition step at –0.4 V (30 s stirred, 30 s equilibration). The anodic square wave sweep followed with pulse height 80 mV, step height 10 mV at 5 Hz to 0.4 V.

4.5.5. Prototype A Drawbacks

The Arduino platform is an enabling technology make development of embedded systems fast and simple. However, it is not suitable for commercialisation due to licence agreements and there are technical drawbacks as well such as in fine control of microcontroller timing and the memory space that is required for its bootloader. Therefore, it was deemed necessary to port the system to a stand-alone AVR setup.

There were also issues such as noise that plagued the system, the data shown in Figures 4-20 and 4-21 is the best that could be achieved after a long process of tweaking how the system deals with the signal (gain, averaging, timing of measurements, *etc.*).

Furthermore, the was a problem with compliance as the system would often struggle to reach voltages as low as – 0.8 V in a real electrochemical cell. Lastly, the digipot solution to programmable gain caused sporadic problems such the chips were regularly needing to be replaced. It would also be better to have the system automatically correct the gain on-the-fly rather than relying on the user setting it to start with. For these reasons, amongst others, the next prototype was developed.

4.6. Prototype Model B

4.6.1. Design Notes on Prototype B

4.6.1.1. Power Considerations

In order to address the lack of power affect the compliance voltage a MAX743 switch mode regular was added (Maxim Integrated, (219)). This enabled the 5 V supplied by the USB cable to be transformed to ± 12 V. This would have the benefit of enabling a wider range of background electrolyte to be analysed such as those involving organic solvents. Additional LC filtering recommended by the data sheet was included for each voltage rail (25 μH with 2.2 μF). Using this system had two knock-on effects: firstly, the DAC had be changed as the MAX5250 did not support this voltage and secondly whilst the efficiency of the MAX743 is quoted as being typically 82%, brown out occurred due to the USB 2.0 system only being able to supply ~ 2.5 W.

To tackle the lack of power, a second USB channel was added; one was dedicated to the digital signal, the other was used solely to supply the analog system with power. Separating analog and digital grounds can also have benefits in terms of improved signal/noise.

The DAC chosen was a MAX532 (Maxim Integrated, (220)). This has two 12-bit channels and can operate across the range of voltages necessary. The datasheet states it has a 2.5 µs settling time to ± 1 LSB. Similar tests were conducted to those in section 4.2.1.3 and satisfactory results were obtained. An 8L41 double pole double throw relay (Coto Technology, (218)) was also added to the power rails to enable to user to choose between ± 5 V and ± 12 V. This was done to improve resolution was high compliance was not necessary. An LTC1250 chopper op amp (Analog Devices, (221)) was used to buffer to + 2.5 V rail instead. This was done to reduce noise.

4.6.1.2. Digital Communication

The same type of microcontroller (ATMega328P) was used as it had proved viable during the work for prototype A. However, UART to USB was needed so the dedicated module UM232R (FTDI, (222)). The MCP3304 ADC was retained.

4.6.1.3. Signal

To replace the faulty digipot system a digital multiplex chip was used instead. The MAX4617 (Maxim Integrated, (223)) is an 8:1 MUX meaning that there are 8 separate gain levels that can be applied to the AD620 (Analog Devices, (224)) instrumental amplifier. The AD620 is a like for like swap with the INA114 that was used in prototype A. The change was made due to chip availability and the option of running simulations in the LTspice software (Analog Devices). The resistor values where chosen to provide an even spread of the gain levels and are shown in Table 4-6.

Table 4- 6: Programmable gain control. Resistors connected to 8:1 MUX for prototype B. Gain calculation performed using Equation 4-6.

MUX Channel	Resistor (Ω)	Approximate Gain
0	7	7058
1	47	1052
2	100	495
3	470	106
4	1000	50
5	4700	12
6	10000	6
7	33000	2

Parasitic capacitance is problem that can have a negative effect on the stability of op amps. This in combination with the fact that electrochemical cells have a significant double layer capacitance at the electrode/electrolyte interface means a strategy is required to overcome this. For prototype Model B the AD826 (Analog Devices, (225)) op amp was chosen as the circuit's CE summing amplifier. This was done because the AD826 was designed to drive a theoretically infinite capacitance. As it is a 2-channel amplifier, it was also used to buffer the RE signal. According to the datasheet, the AD826 has a 350 V μs^{-1} slew rate with a 70 ns settling time to 0.01%.

4.6.2. Board Layout

As the circuit was more complex than prototype A and through hole components were still required to make hand soldering easy, a decision was taken to produce three separate boards and stack them to save space. The upper layer housed the analog front end and indicator LEDs whilst the middle layer housed the ADC and the DAC (which now required an external op amp). The lower layer contained the microcontroller, the UART to USB converter and the power regulating components.

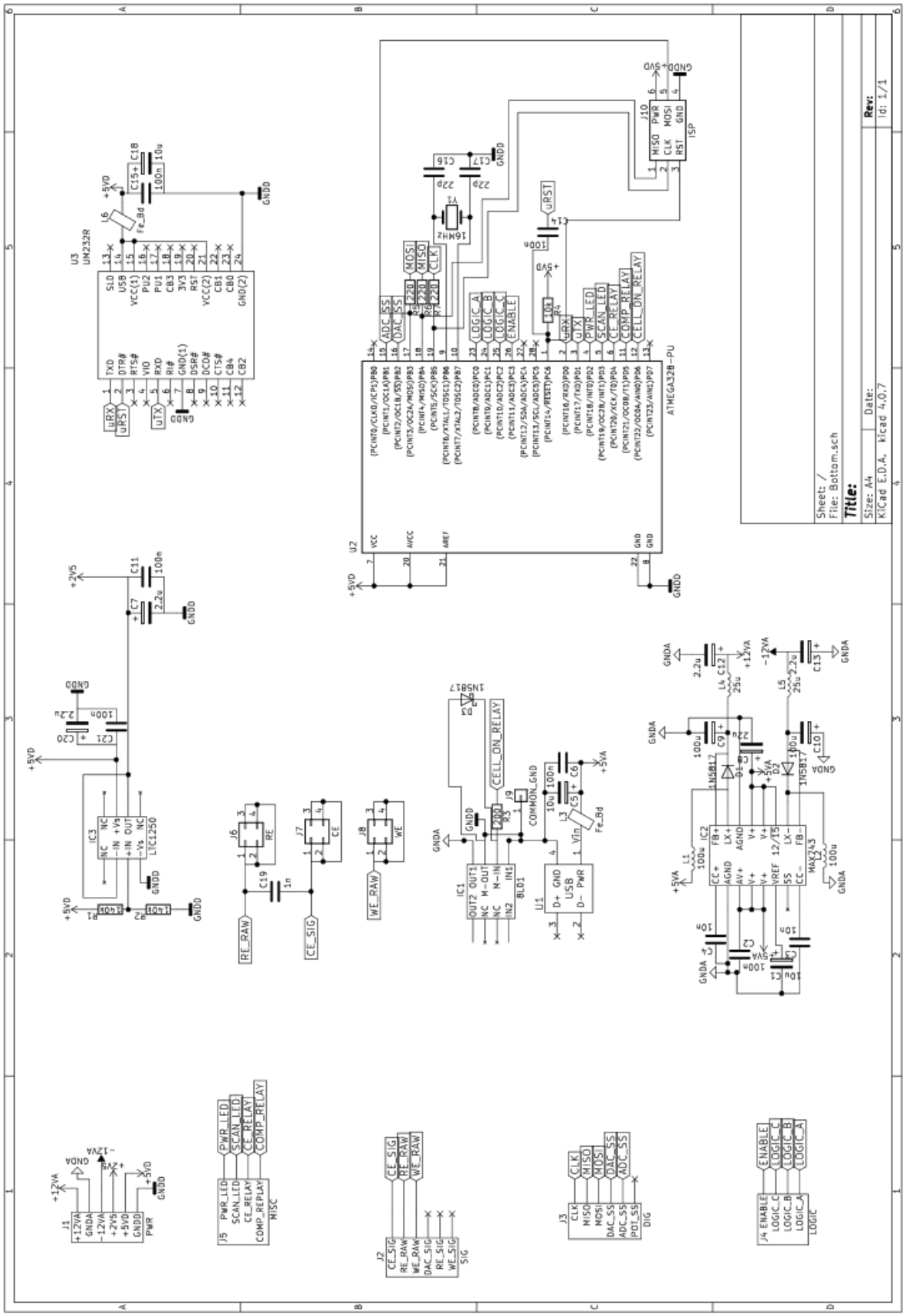


Figure 4- 22: Schematic for lower layer of prototype Model B

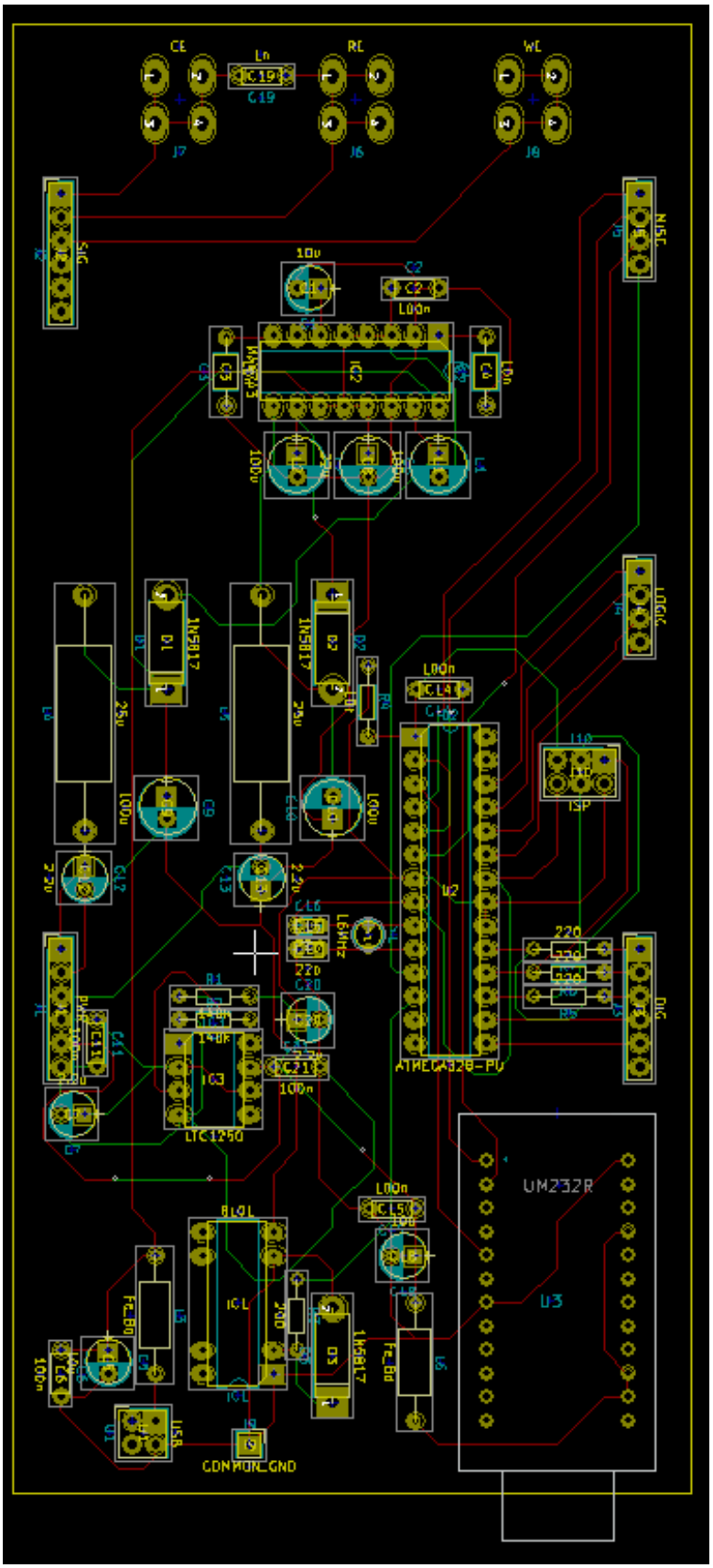


Figure 4- 23: Board layout for lower layer of prototype Model B. Top copper traces in red, underside traces in green.

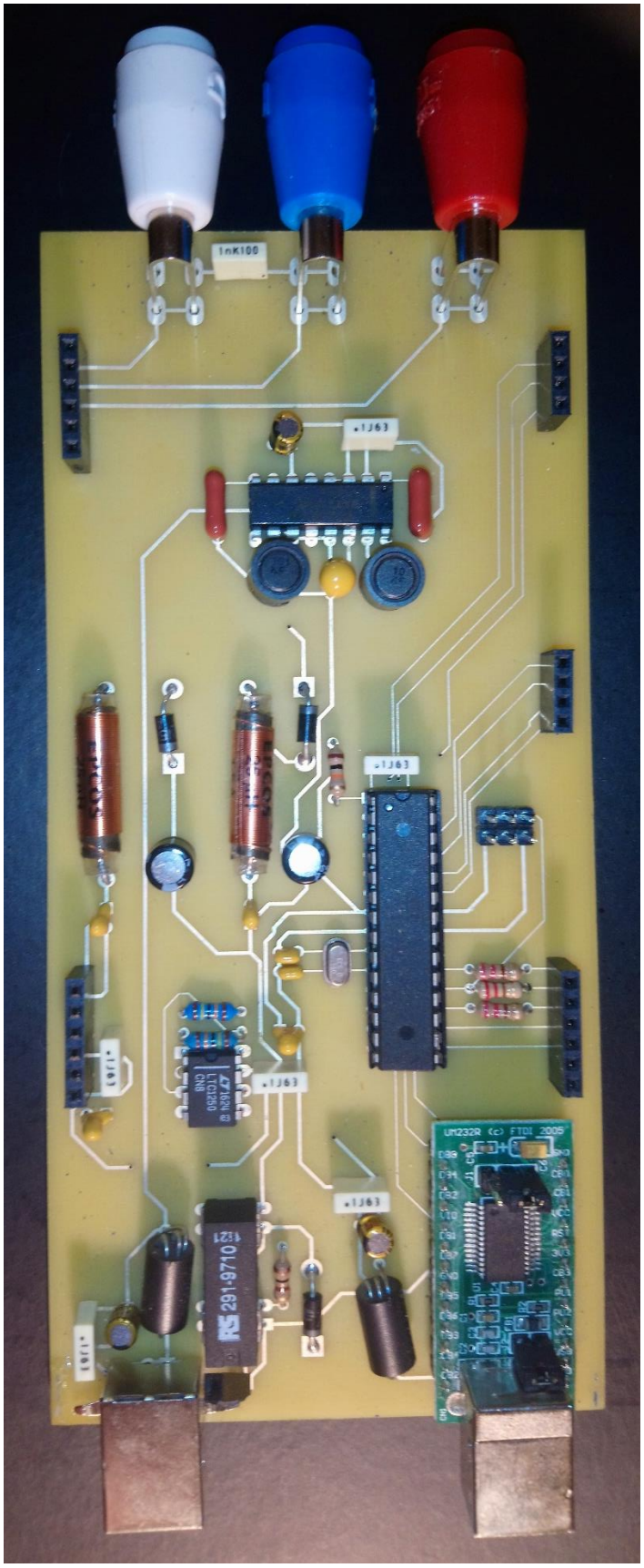


Figure 4- 24: Populated board for lower layer of prototype Model B

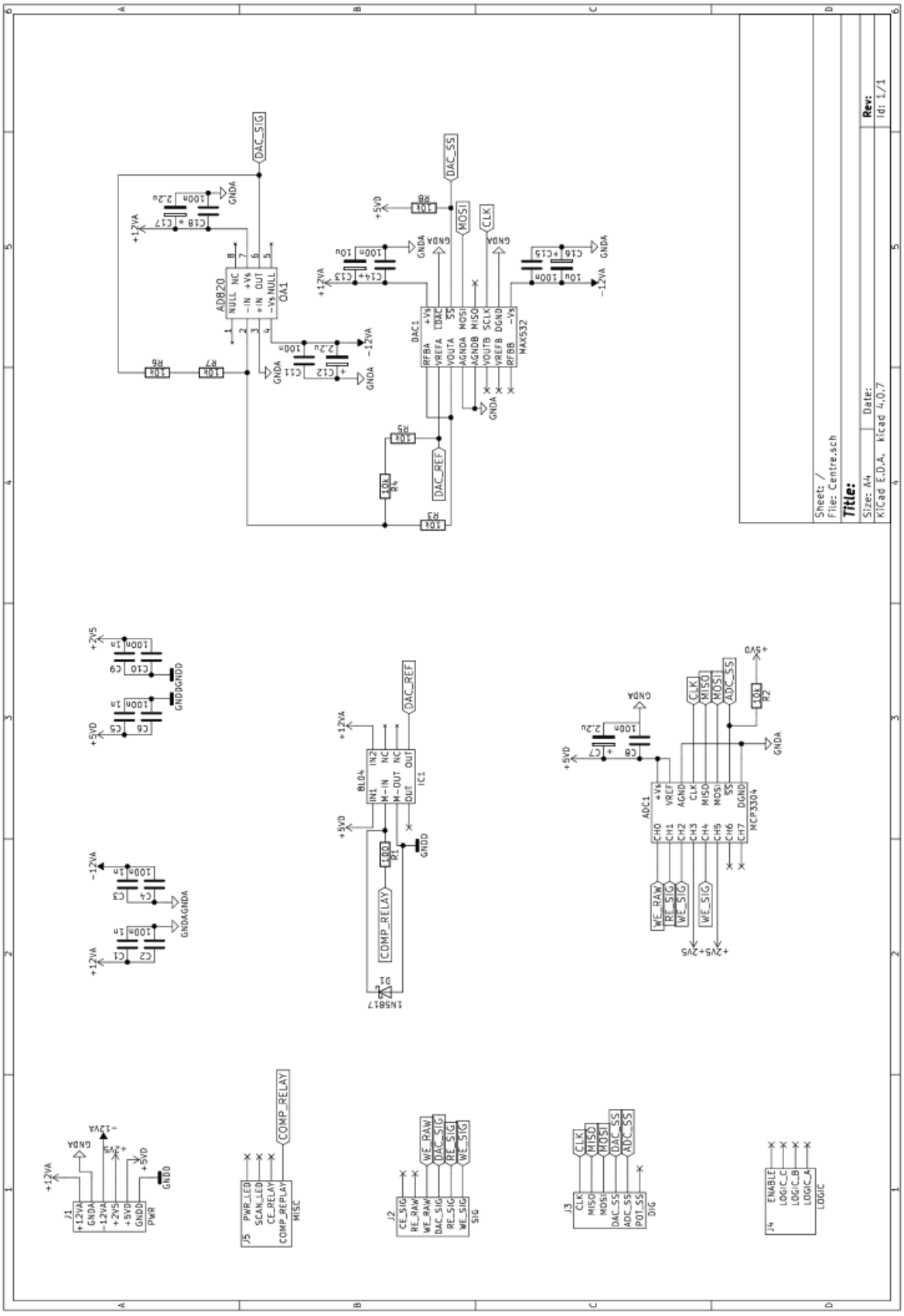


Figure 4- 25: Schematic for middle layer of prototype Model B

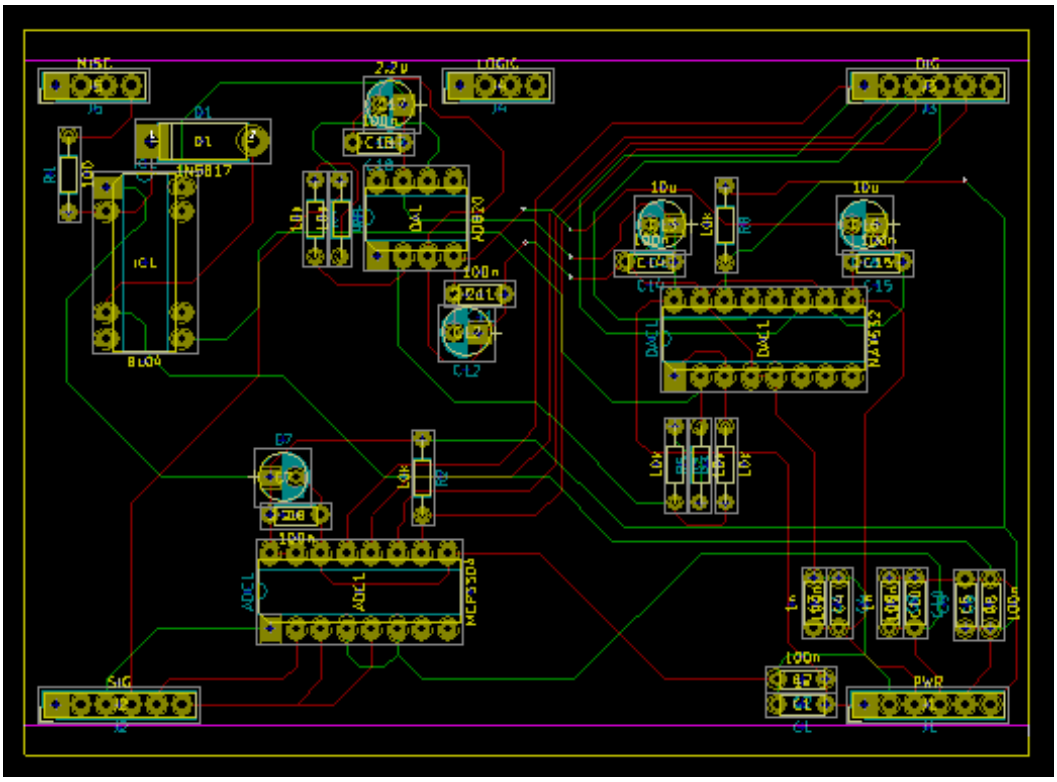


Figure 4- 26: Board layout for middle layer of prototype Model B. Top copper traces in red, underside traces in green.

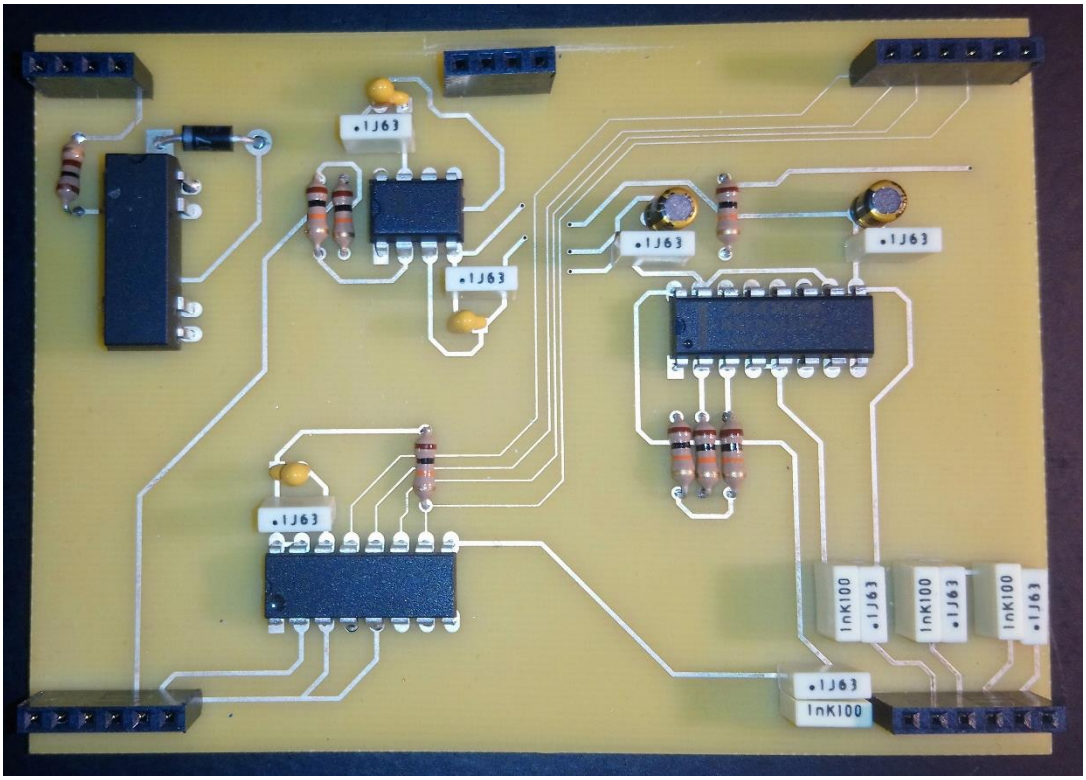


Figure 4- 27: Populated board for middle layer of prototype Model B

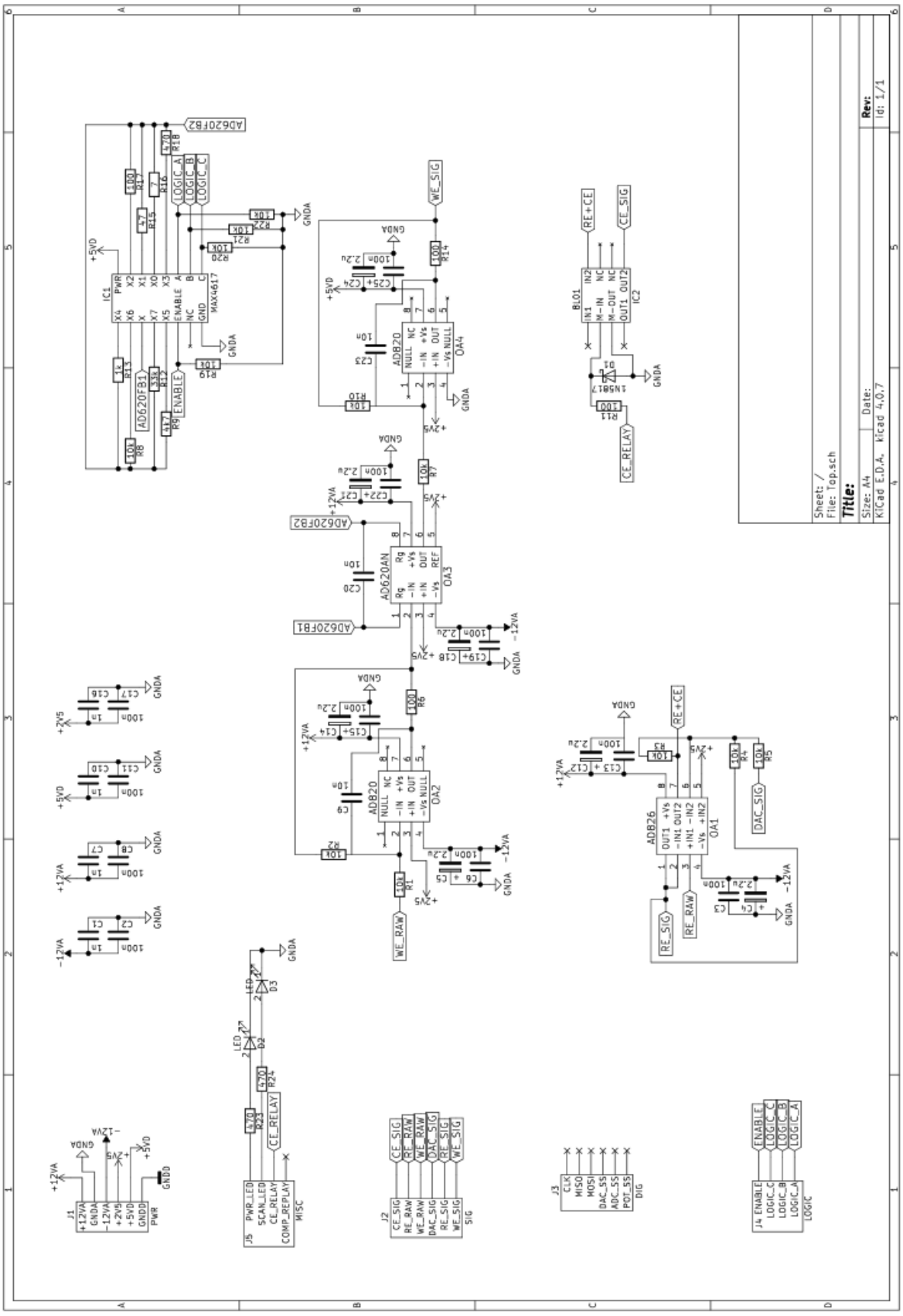


Figure 4- 28: Schematic for top layer of prototype Model B

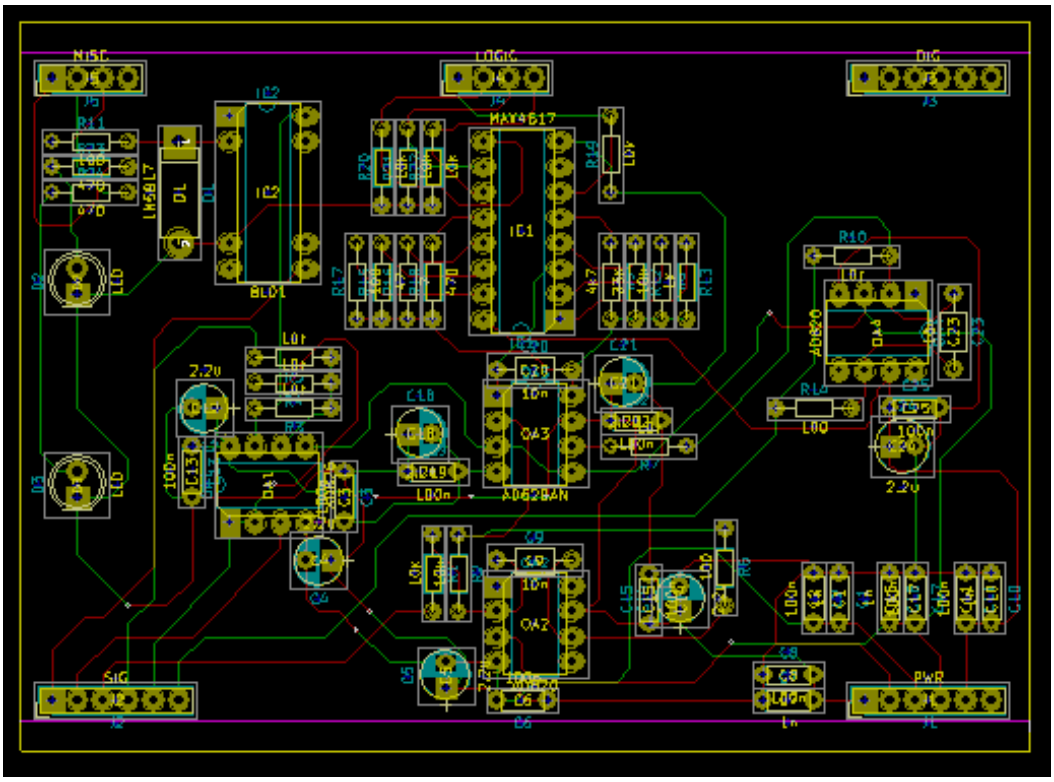


Figure 4- 29:: Board layout for top layer of prototype Model B. Top copper traces in red, underside traces in green.

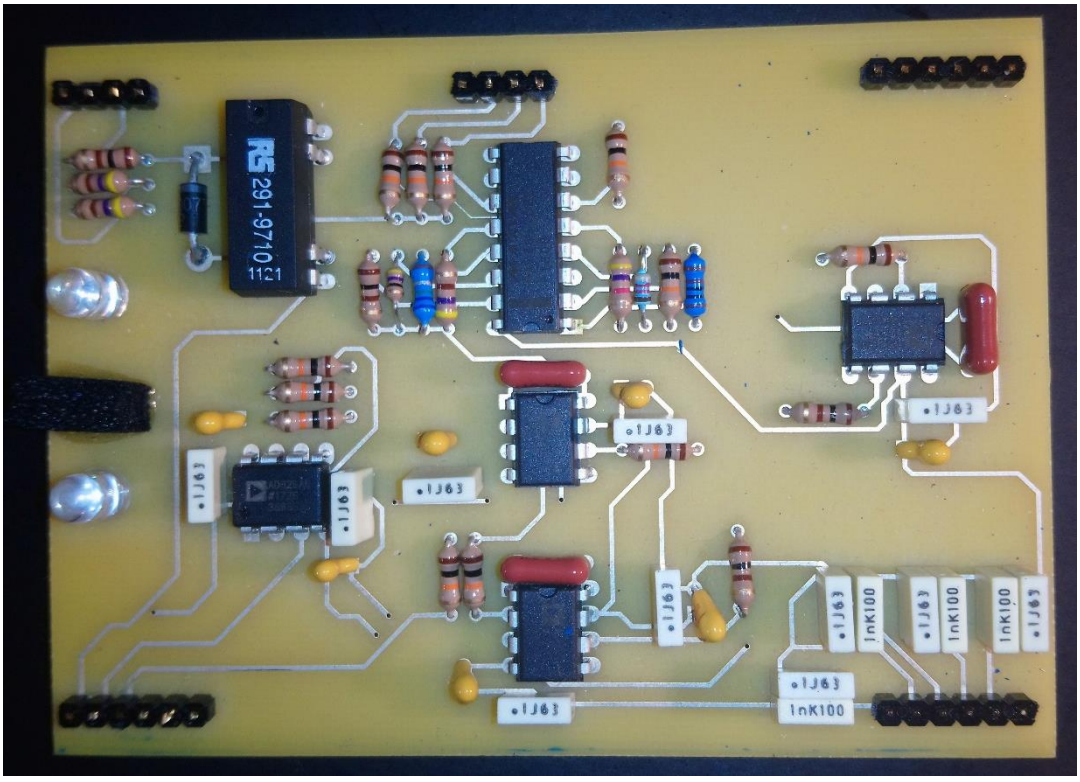


Figure 4- 30: Populated board for top layer of prototype Model B

The schematics and board layouts were created using KiCad (KiCad Developers Team) and are shown alongside photographs of the populated boards in Figures 4-22 to 4-30.

4.6.3. Firmware

The firmware was developed and compiled using Atmel Studio 7 (Microchip Atmel, (172)). An AVR-ISP-MK2 (Olimex, (226)) was used to upload the hex-code through an in-circuit serial header on the lower layer board.

The overall architecture of how the firmware worked was largely the same as for prototype A. There were ten C source files created to separate out the different functions and the code for these are given in Appendices 4 to 13. Additional debugging functions were added to improve the signal quality alongside automatic current ranging. The debugging functions were labelled as "ALPHA_TESTS" and more will be explained about them in the next chapter.

4.7. Project Direction

Linear and cyclic scans were run on prototype B using the dummy cell (Figure 4-18). The results for these scans were only marginally better than what was achieved with prototype A. The next stages in making the product viable were identified and can be described in three main objectives:

- Product a graphical user interface (GUI) – end users should be easily able to enter their method parameters and review their data. Exporting results to MS Excel is tedious and time consuming. Therefore, the GUI was considered the top priority.

- Optimise the way the signal is processed – there are several parameters that can influence the quality of signal and they need to be investigated in tandem.
- Calibrate the instrument – the raw ADC signal from the current of the working electrode should be translated into amperes. Once this achieved direct comparison with a commercial instrument will be possible.

These objectives occupy the work conducted in the next chapter.

5. Graphical User Interface Development, Instrument Optimisation and Calibration

5.1. Graphical User Interface Design

The Processing computer language (201) was originally released in 2001 but is now in its third main version. It has implementations for Python, Raspberry Pi and Android which has resulted in a vibrant user community. It enables easy to produce, high quality graphic applications that are intuitively designed using its many library functions.

As previously mentioned in section 4.3.2 the Wheestat made by Smoky Mountain Scientific (196), is a proof-of-concept potentiostat uses Processing to provide its graphical user interface (GUI). A screen capture of the software is shown in Figure 5- 1.

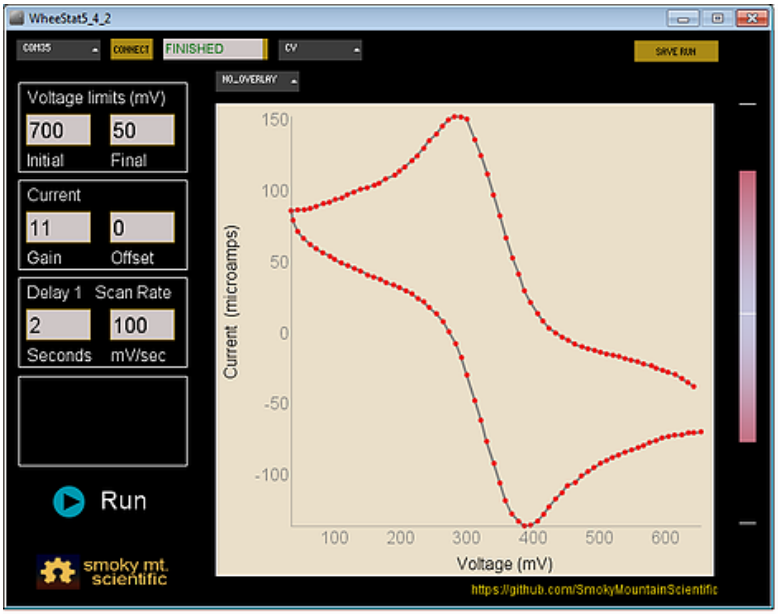


Figure 5- 1: Wheestat GUI screen capture of CV data. Note that the positive cathodic and negative anodic current convention used in the United States of America is employed here.

The ability of Processing to easily handle serial data from the USB in conjunction with the many GUI library files meant it was an obvious choice to use to develop the interface for prototype B.

The code for the GUI that was created is given in Appendix 14 and a screen capture of some data that was run on the system is given in Figure 5-2. A new object class was created (named "Scan") for each scan to store the data; the data is automatically saved as a csv file when the scan finishes. Besides the inbuilt Processing library files two further libraries are called at the start: interfascia and controlP5. These were used to instantiate the button, slider and textbox objects.

The left-hand side of the screen is for instrument control, where the user will first select a method from CV, LSV or SWV (note chronoamperometry is yet to be developed for this system). The relevant textboxes then appear to allow the method parameters to be entered. The "Current Range" slider controls the MAX4617 8:1 MUX *i.e.* the feedback and hence the gain of the AD620 instrumental amplifier. There is a "Hold Gain" checkbox available, whereby the user can disable the automatic current ranging that occurs when scans are running. This may be of use when there are small peaks in a certain potential range and the saturation of the ADC at other potentials is of no consequence. A toggle button to switch between the ± 5 V and ± 12 V compliance ranges is also included on this side of the screen. Lastly the "Choose Port" button ensures the correct device is selected for the USB communication.

The right-hand side of the screen holds the user controls relating to data analysis. An optional "Filename:" textbox is provided to specify what the csv files should be saved as. However, if this is blank the system will default to using the date and time.

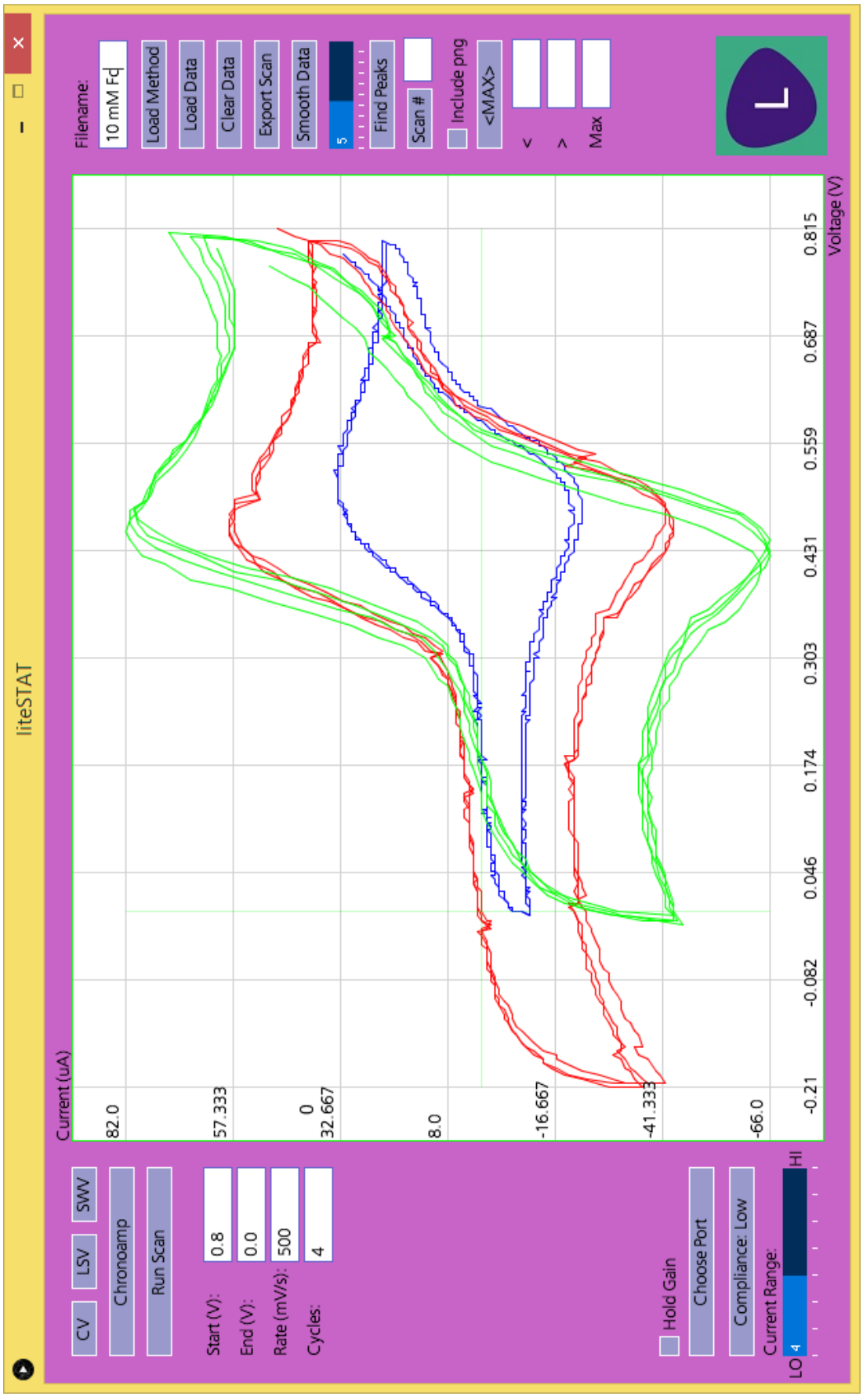


Figure 5- 2: Screen capture of liteSTAT GUI showing CV scans of 10, 100 and 500 mV s^{-1} of 10 mM ferricyanide at a carbon WE. Data aquired using prototype B.

There are buttons to load, export and clear the data here as well as loading a saved method. The method details are saved alongside the data in the same csv file each time it is run.

There is a "Smooth Peaks" toggle button that is linked to the slider just below This will employ Savitsky-Golay (SG) smoothing (227) to the data and the user can specify the number of coefficients using the slider. An example of the efficacy of the smoothing is illustrated in Figure 5-3 where the noisy SWV data on the left has not been SG filtered. The data was generated using the prototype B system using a glassy carbon WE on a 10 mM ferricyanide solution. The centre data has been filtered with 5 coefficients and the one on the right has been filtered with 10.

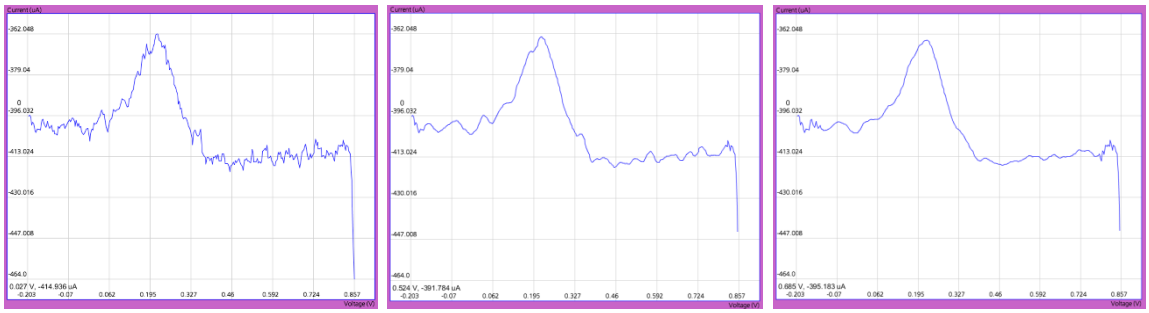


Figure 5- 3: Savitsky-Golay filtering example on SWV data of ferricyanide sample. Raw data (left), 5 coefficient SG smoothing (centre), and 10 coefficient SG smoothing (right).

The data analysis section also enables peak integration where the user manually draws the baseline. Alternatively, the user can find maximum and minimum current values between specified potential ranges. Lastly there is a checkbox to save the screen as a portable network graphic (png) file if it needs to be appended to a report.

The name "liteSTAT" was given to the GUI which seemed to function well, although several amendments are still required before it can be made available for general use.

5.2. Signal Optimisation

The inspection of the signal produce by prototype B required several more software scripts to be written. These were all produced using Processing; the code for them is not shown here but available on request.

5.2.1. Generation of Signal Performance Data

The signal produced by the prototype Model B potentiostat depends on several factors as outlined below:

- The analog front end – the way current and voltage are amplified uses different hardware. Therefore, it is to be expected that there would be considerably less noise in the voltage signal compared to current signal which is subject to high gain amplification.
- The wait time – upon the application of the voltage to cell by the potentiostat there is a period of several ms waiting time before the current is measured. This is especially necessary in electrochemical cells to allow the Non-Faradic currents to dissipate. However, it can also have a bearing the way the prototype B's op amps are able to settle to a stable value regardless of what is happening in the cell.
- The measurement number – taking more samples and averaging them was already known to have an enormously beneficial effect on the signal quality. However, there is an obvious trade off: taking more samples of the signal requires more time. It was not known what the optimum number of samples to take was.

- The DAC step – one workaround to speeding up the scan speed would be to use greater increments of the DAC. A single MAX532 increment can be considered to be roughly equivalent to 2.4 mV (as it is a 12-bit system and assuming the prototype B is set to the low compliance). However, larger increments would also mean fewer data points across the voltage sweep and may impact on the settling time of the op amps.

Further to this it was not understood how the interplay been DAC step, measurement number and wait time would affect the signal. Therefore, it was decided to write some control software that would run a range of settings so the data could be interrogated. Every combination of the variables that are shown in Table 5-1 was run in triplicate using the dummy cell (Figure 4-18) in a linear sweep from -1 to +1 V measuring the current output.

Table 5- 1: Measurement of current signal, data generation parameters

Parameter	**Range Run**
DAC step	Integer values 1 to 20
Wait time	Integer values 0 to 30 (ms)
Measurement Number	8, 16, 32, 64, 128, 256, 512

This resulted in a total of 13440 individual scans being run, each generating their own csv file. These files contained every sample of the signal that the prototype B measured which was an enormous volume of data (1.35 GB) rather than just the average. The prototype B was controlled by some new software (called "Sample_Runner") designed specifically to run these ranges of samples one after the other. However, each of the triplicate runs still took over 24 hrs to complete. The measurement numbers are listed in the factors of 2^x to enable bit shifting to done when calculating the average, as division is a notoriously long process in an 8-bit AVR environment.

In addition to current measurement scans, the parameters shown in Table 5-2 were also run to measure the voltage using the same setup.

Table 5- 2: Measurement of voltage signal, data generation parameters

Parameter	Range Run
DAC step	Integer values 1 to 20
Wait time	Integer values 0 to 10 (ms)
Measurement Number	8, 16, 32, 64

The voltage data was only run once, producing a total of 880 scans.

5.2.2. Analysis of Signal Performance Data

In order to view the data another Processing application was developed (called "Alpha_Validation"). This included the facility of averaging the signal using different techniques. For example, if as many as 512 samples are taken for a current signal at a given voltage it would be possible to:

- take an average of all 512 samples as the signal
- taken an average of the last 256, or the last 128 samples, etc.
- calculate a moving average for the signal using Equation 5-1 then use the very last moving average value as the signal
- calculate an average of the moving average values for the signal
- calculate an average from the last 256 moving average values, etc

Equation 5-1: $$MAV_{Sample} = \frac{\alpha \times MAV_{PREVIOUS} + Sample}{\alpha + 1}$$

Where MAV_{Sample} is the moving average value of the current sample and the factor α can be tuned to affect the sensitivity. The larger the value of α the greater the smoothing effect will be, so the system will be less susceptible to noise. However, increasing α values also cause the system to be insensitive to true changes in the signal. In an 8-bit AVR environment, values of 3, 7, 15 are preferred to enable bit shifting rather than division.

The Alpha_Validation application was also programmed to highlight the values that the different systems of averaging would take. The analysis of two example data points is shown in Figure 5-4. The checkboxes along the right-hand side of the screen enabled the different averaging mechanisms to be visualised on top of the data.

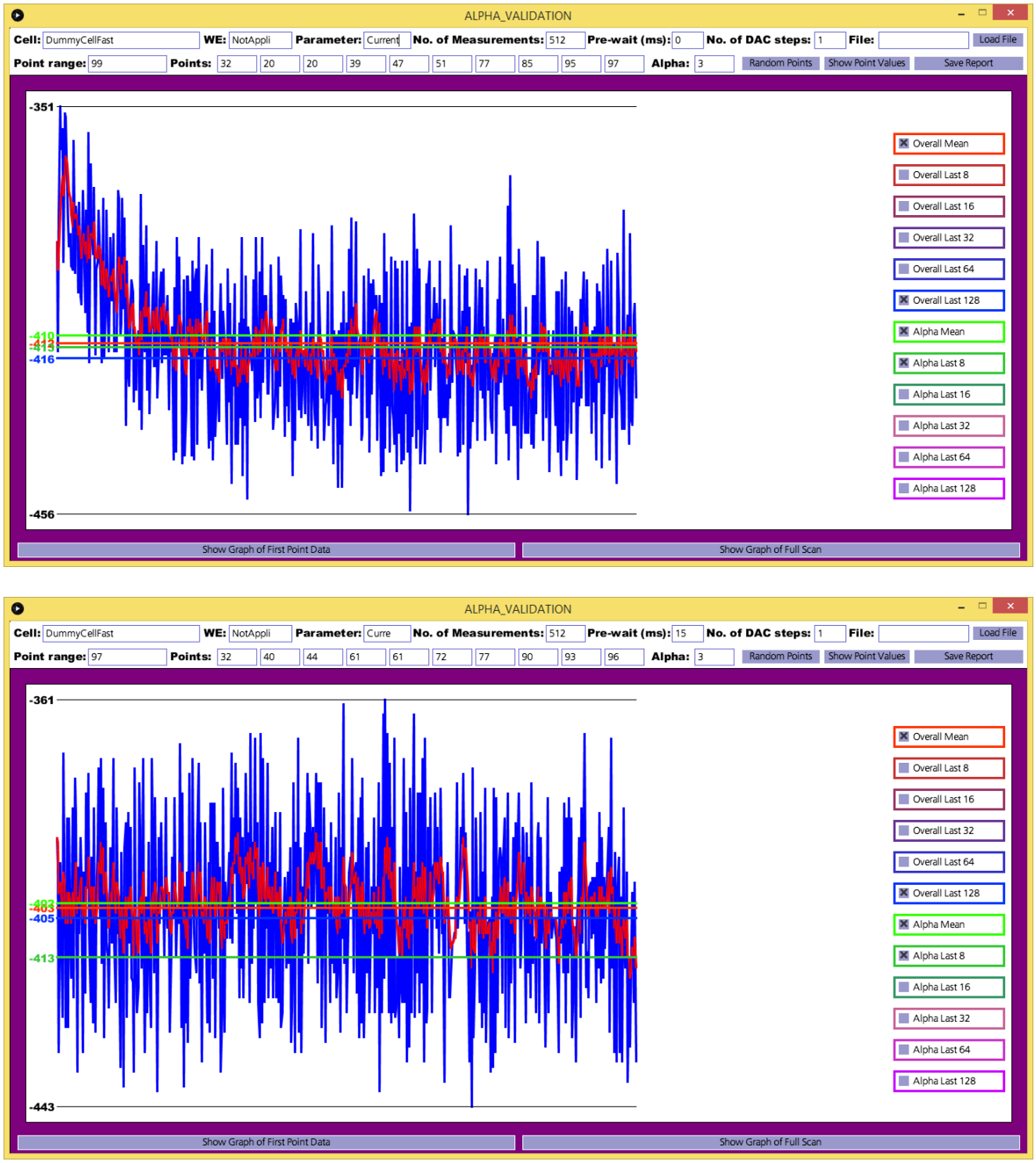


Figure 5- 4: Alpha_Validation data viewer of two single data points from different current scan parameters. 512 Samples, DAC increment=1, wait time=0 ms (top) 512 Measurments, DAC increment=1, wait time=15 ms (bottom). Raw data points in blue, moving average in red.

The two measurements shown in Figure 5-4 were for scans that had the parameters shown in Table 5-3.

Table 5- 3: Measurement parameters for samples in Figure 5-4

Figure 5-4 Position	Number of Samples per Data Point	DAC increment	Wait time (ms)
Top	512	1	0
Bottom	512	1	15

Using α=3, Table 5-4 shows the results that were calculated for the two current measurement data points (each composed of 512 samples). The values in the table are the raw ADC readings.

Table 5- 4: Summary data of Alpha_Validation analysis of scans shown in Figure 5-4

Averaging System used to find Value for Data Point	Figure 5-4 (Top) Value	Figure 5-4 (Bottom) Value
Maximum raw data point	-351	-361
Minimum raw data point	-456	-443
Mean of all raw data Points	-412	-403
Mean of last 8 raw data points	-415	-418
Mean of last 16 raw data points	-415	-413
Mean of last 32 raw data points	-415	-409
Mean of last 64 raw data points	-416	-406
Mean of last 128 raw data points	-416	-405
Mean of all alpha corrected points	-410	-402
Mean of last 8 alpha corrected points	-413	-413
Mean of last 16 alpha corrected points	-413	-410
Mean of last 32 alpha corrected points	-413	-406
Mean of last 64 alpha corrected points	-414	-404
Mean of last 128 alpha corrected points	-415	-404

There are several points to be drawn out from the analysis of these two samples. Firstly, the faster sample (Figure 5-4 top) shows a decay at the start of the measurement. The majority of this effect is due to the charge and discharge times of the 1 μF capacitor in the dummy cell. This effect would be magnified in an actual electrochemical cell due to the diffusion of ions in solution. The wait time of 15 ms (Figure 5-4 bottom) is enough to completely negate this effect.

Secondly, the data from both samples is extremely noisy. The range shown for the faster scan is approximately 20% greater than the slower scan. However, even the slower scan has the range of 82 ADC units which is around 2% of the MCP3304's total available range (at the optimised gain of the instrumental amplifier). This is unacceptably noisy and will be discussed in section 5.2.2.1.

Thirdly, and somewhat counter intuitively, the faster scan seemed to produce the most consistent values for the different averaging techniques (Table 5-4). For the scan with 0 ms wait time there was a total range of 6 ADC units between all the averaging techniques, whereas for the 15 ms they varied by 16 ADC units.

More functionality was added to the Alpha_Validation software to show the full scan from -1 to +1 V for each of the averaging techniques. Whilst Figure 5-4 shows the data that is generated for a single current measurement, Figure 5-5 illustrates the entire voltammogram. Checkboxes were again created so that the scan for each averaging technique could be viewed in isolation or together. Linear regression of the scan line for each averaging technique was also calculated, in the normal fashion shown in Equations 5-2 and 5-3.

Equation 5-2: $$Slope = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

Equation 5-3: $$Intercept = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

In the case of the dummy cell, the data should be linear; theoretically starting at the data point (-1V, -1μA) moving through (0V, 0μA) and finishing at (1V, 1μA). Having the slope/intercept enabled the calculation of the sum of the square of the residuals (SSR) for each averaging technique.

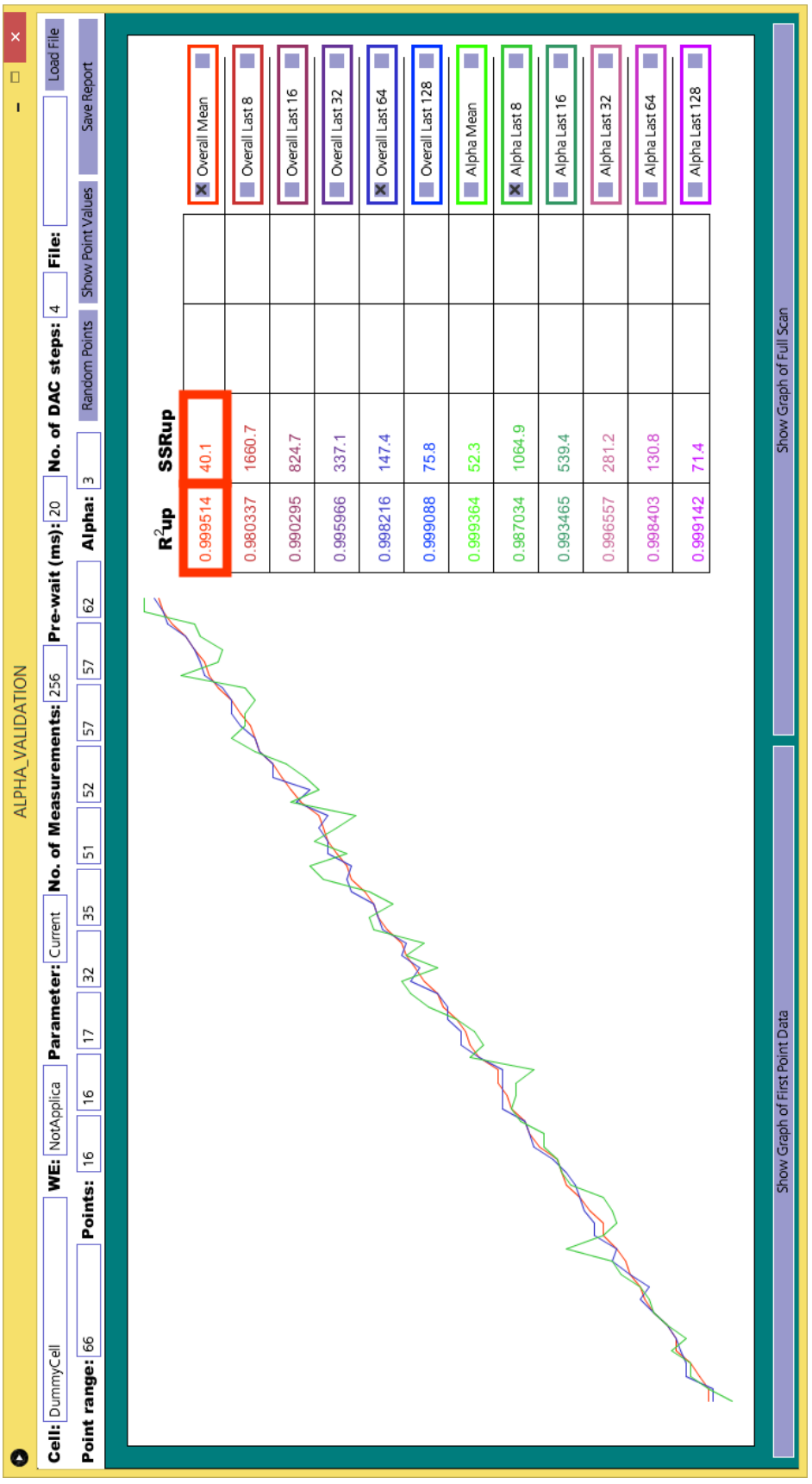


Figure 5- 5: Alpha_Validation of voltammogram produced for settings of current measurement DAC step=4, wait time=20 ms, 256 samples per data point. Shown is the overall average of each sample (red), the average of the last 64 points (blue) and the average of the last 8 moving average points when alpha=3 (green).

The square of the Pearson correlation coefficient for each averaging technique was also calculated using Equation 5-4.

Equation 5-4: $$R^2 = \left(\frac{n(\sum xy)-(\sum x)(\sum y)}{\sqrt{(n(\sum x^2)-(\sum x)^2)(n(\sum y^2)-(\sum y)^2)}}\right)^2$$

The software then calculated the maximum R^2 value and the minimum SSR and used a thick box to highlight which averaging technique was the best. For Figure 5-5, the best averaging technique was the average of all 256 samples. It is also apparent how noisy the average of the last 8 moving average points scan is (shown in green).

Whilst it was deemed necessary to be able to view the data points to get an understanding of how the different averaging affected the signal, the scale of the data meant an alternative strategy was necessary to review it. Thus far data from only 3 out of the total 14320 scans has been presented. Therefore, new software (called "Meta_Analysis") was created that would load a batch of 112 scans at a time and ascertain the optimum SSR and R^2 values for each. The data was sliced such it presented an analysis page for all the sample measurement number and wait times for a single prescribed DAC point increment size. As 20 different DAC increments were used this meant the analysis had to be performed 20 times for each replicate. Table 5-5 shows the averaging methods that produced the highest R^2 values for the DAC increment of 1. Table 5-6 shows the averaging methods that produced the lowest SSR values for the same set of scans. The key to the data labels that appear in these tables is given in Table 5-7.

Table 5- 5: Assessment of best method to find average data point based on R^2 for DAC=1 current measurement, abbreviations explained in Table 5 -7.

	Averaging algorithm with highest R^2 value						
	Number of sample measurements per data point						
Wait (ms)	**8**	**16**	**32**	**64**	**128**	**256**	**512**
0	UA	UA	UA	UA	UL64	UL128	UL256
1	A31A	A3A	UA	UA	UA	UL128	UL256
2	UA	UA	UA	UA	UA	UL128	UL256
3	UA	UA	UA	UA	UA	UL128	A3L256
4	UA	UA	UA	UA	UA	UL128	UL256
5	UA	UA	UA	UA	UA	UA	UL256
6	UA	UA	UA	UA	UA	UA	A3L256
7	UA	UA	UA	UA	UA	UA	UA
8	UA	UA	UA	UA	UA	UA	UA
9	UA	UA	UA	UA	UA	UA	UL256
10	UA	UA	UA	UA	UA	UA	A3A
11	UA	UA	UA	UA	UA	UA	UA
12	UA	UA	UA	UA	UA	UA	A3A
13	UA	UA	UA	UA	UA	A3A	UA
14	UA	UA	UA	UA	UA	UA	UA
15	UA	UA	UA	UA	UL64	A3A	UA

Table 5- 6:Assessment of best method to find average data point based on SSR for DAC=1 current measurement, abbreviations explained in Table 5 -7.

	Averaging algorithm with lowest SSR value						
	Number of sample measurements per data point						
Wait (ms)	**8**	**16**	**32**	**64**	**128**	**256**	**512**
0	UA	UA	UA	UA	UL64	UL128	UL256
1	A31A	A3A	A3A	A3A	UA	UL128	UL256
2	UA	UA	UA	UA	UA	UL128	UL256
3	UA	UA	UA	UA	UA	UL128	UL256
4	UA	UA	UA	UA	UA	UL128	UL256
5	UA	UA	UA	UA	UA	UA	UL256
6	UA	UA	UA	UA	UA	UA	UL256
7	UA	UA	UA	UA	UA	UA	UA
8	UA	UA	UA	UA	UA	UA	UA
9	UA	UA	UA	UA	UA	UA	UL256
10	UA	UA	UA	UA	UA	UA	A3A
11	UA	UA	UA	UA	UA	UA	UA
12	UA	UA	UA	UA	UA	UA	A3A
13	UA	UA	UA	UA	UA	A3A	UA
14	UA	UA	UA	UA	UA	UA	UA
15	UA	UA	UA	UA	UL64	A3A	UA

Table 5- 7: Explanation of labels used in Tables 5-5 and 5-6

Label	Explanation
UA	Unfiltered average of all the sample points
UL64	Unfiltered average of last 64 points
UL128	Unfiltered average of last 128 points
UL256	Unfiltered average of last 256 points
A31A	Average of all moving average filtered (α =31) sample points
A3A	Average of all moving average filtered (α =3) sample points
A3L256	Average last 256 moving average filtered (α =3) sample points

Most of the scans performed best when the total average of all the ADC data points was taken. The most important point to take from Tables 5-5 and 5-6 is that there is no substitute for the number of sample measurements. The more that can be included, the better the average will be and using a moving average does not outweigh this affect. For clarity only the data for the scans where the DAC increment was 1 is shown, however the other 19 values did show a similar trend. The analysis was conducted for all scans testing α values of 3, 7, 15 and 31. Generally speaking, α=3 had the best performance for the moving average filtered data. However, all the other values did appear sporadically as the best method in the tables. Tables 5-5 and 5-6 show a trend towards the top right-hand side (*i.e.* faster scans with more measurements). Here the simple averaging of all points does not dominate, but the last half of the data performs better. That can be explained by comparing the top and bottom parts of Figure 5-4 where the data seems to settle after the first third of the measurements in the fast scan (top).

The results from the voltage measurements were so close that it would not really have too great an impact on the voltammogram which type of signal averaging was used as long as the measurement number was > 16. In total, well over 4000 lines of processing code were written in order to investigate the signal.

5.2.2.1. Noise Minimisation Through Hardware Changes

As the results from the signal investigation showed that there was significant noise, it was decided to try some additional filtering prior to the ADC. The prototype B circuit was reconstructed on a breadboard and various RC filters were applied as shown in Figure 5-6.

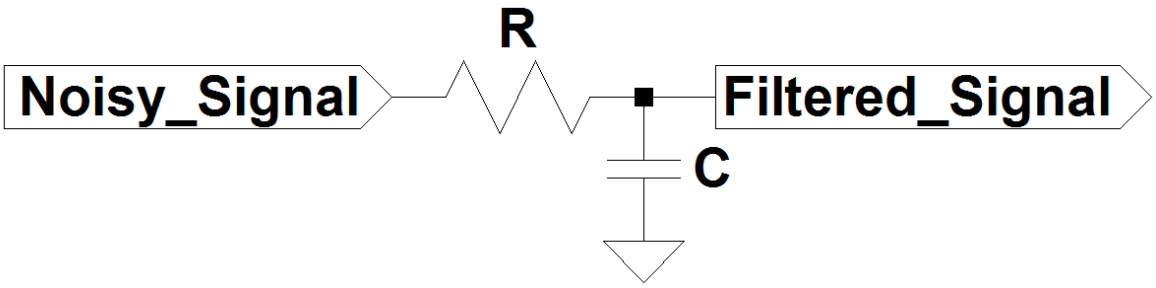


Figure 5- 6: Generalised schematic of RC low pass filter

The cut-off frequency for a filter of this type is given in Equation 5-5 in Hertz if the resistor value is in Ohms and the capacitor is in Farads.

Equation 5-5: $$f_c = \frac{1}{2\pi RC}$$

Attempts were made to use very low frequency band pass filters placed between the last op amp and the ADC in the circuit. However, despite trying a variety of settings the noise could not be improved when running dummy cell scans. As an alternative, the MAX7401 (228) chip was tried. This is an 8th order, lowpass, Bessel filter in a monolithic chip. When CV and SWV scans were run on ferricyanide solutions the performance deteriorated with the inclusion of the filter. This is thought to be due to not driving the ADC efficiently. At this stage the hardware analog filters were removed and not pursued any further.

5.3. Calibration of the Prototype Model B

As previously mentioned, the dummy cell shown in Figure 4-18 would be ideal to calibrate the system, as Ohms law allows the calculation of what the current should be for any given application of potential to the CE. The Sample_Runner software was modified slightly to allow runs at the specified scan rates, but with varying feedback gains. This data was used to regress the curves and workout what the correction factors should be for the current at each setting of the drop resistor across the instrumental amplifier.

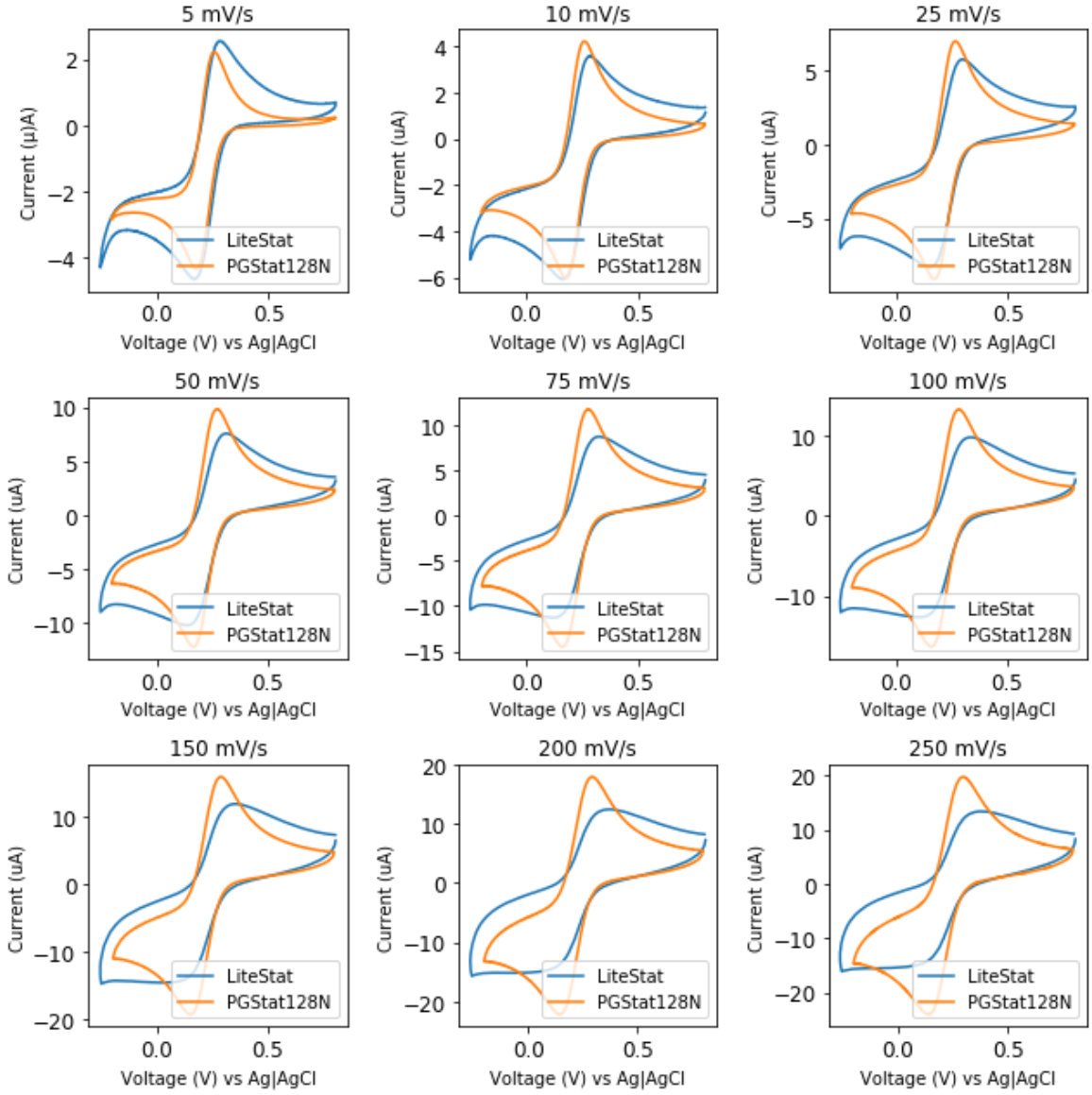


Figure 5- 7: Comparison of prototype B (liteStat) vs Autolab (PGStat128N) instruments for slow CV scans of 100 µM $K_4Fe(CN)_6$ in 0.1 M KCl at a graphite WE.

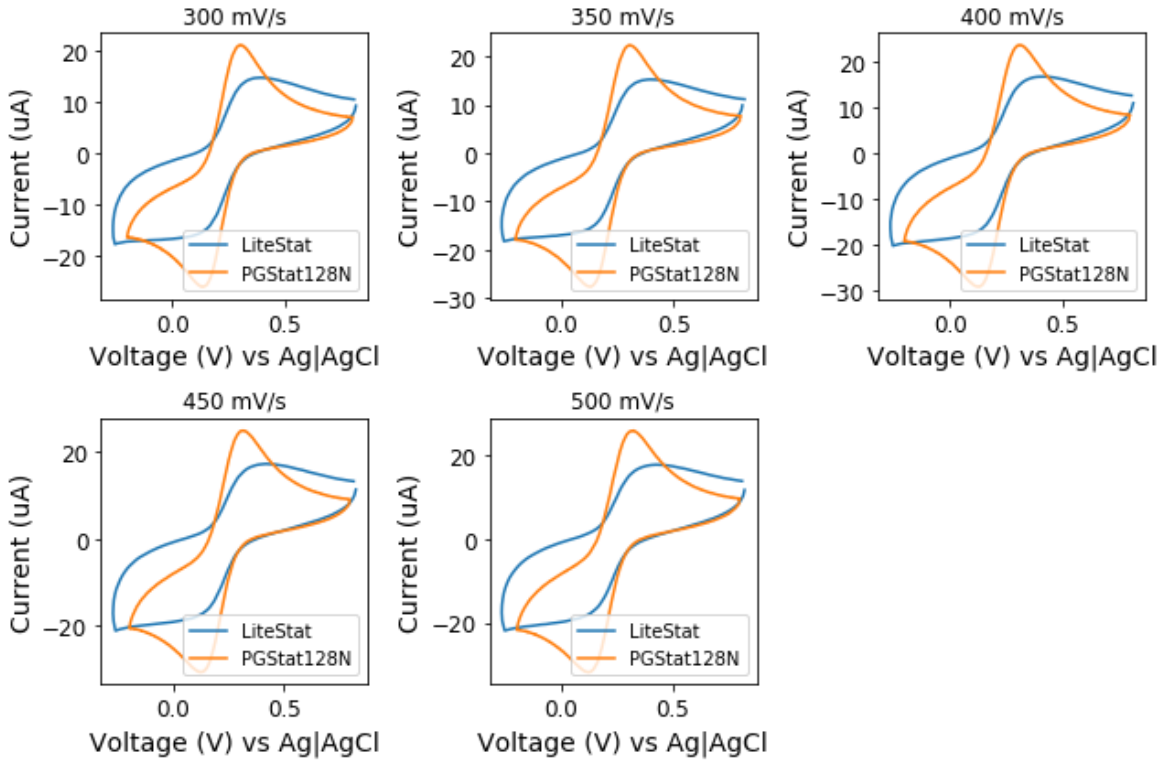


Figure 5- 8: Comparison of prototype B (liteStat) vs Autolab (PGStat128N) instruments for fast CV scans of 100 µM $K_4Fe(CN)_6$ in 0.1 M KCl at a graphite WE.

Once this was obtained a comparison was made of the prototypes B's performance against a commercially available instrument was made.

The PGStat128N from Metrohm, Autolab was connected to a cell that contained approximately 100 µM ferricyanide in 0.1 M KCl. A 3mm glassy carbon disc electrode (BASi) was used as the WE, a platinum sheet was the CE and an Ag|AgCl reference (both Metrohm, Autolab). Various CV scan rates were run and the PGStat128N was then disconnected and the prototype B was connected. The CV process was repeated, and the results of the comparison are shown in Figure 5-7 and 5-8.

The results of these scans showed two main things:

- the calibration method had been successful as the current values from all sets of scans were on the same scale, and closely matched at the slow scan speed.

- Using the averaging method had caused an error which was affecting peak shape. The current that is being measured is too heavily influenced by the non-Faradic electrochemical processes, so the fine structure of wave is lost. The faster the scan rate, the more pronounced this effect was, which corresponds well with having a more compressed diffusion layer at the electrode surface.

In an effort to produce smooth signals from the prototype B, there has been a consequent error introduced into how the measurement is made. To combat this a new strategy is required which are detailed in the further work section.

5.4. Application of the Prototype B to the Analysis of Controlled Drug Samples

Notwithstanding the error in qualitative measurement stated above of linear scans, the prototype B could still be used for its intended quantitative purpose. To validate this a new batch of Paste 6 was created as shown in section 3.2.3 to use as WE material. As described in that chapter a 40 mM pH 9.5 BR buffer was also prepared to act as the electrolyte in a 50 mL cell. The platinum sheet CE and Ag|AgCl RE were also used as before. A 1 mg ml^{-1} solution of 3,4-methylenedioxymethylamphetamine (MDMA) in methanol (certified reference material (Supelco)) was used to add increasing concentrations to the cell such that concentrations over the range 40 to 80 μM was achieved. A SWV method was developed with a deposition step of 30 s at -0.45 V with stirring, then 30 s at the same potential without stirring to allow the sample to become quiescent. This was followed by the anodic sweep from -0.45 V to +1.0 V with a step height of 4 mV, a pulse height of 100 mV and a frequency of 1 Hz.

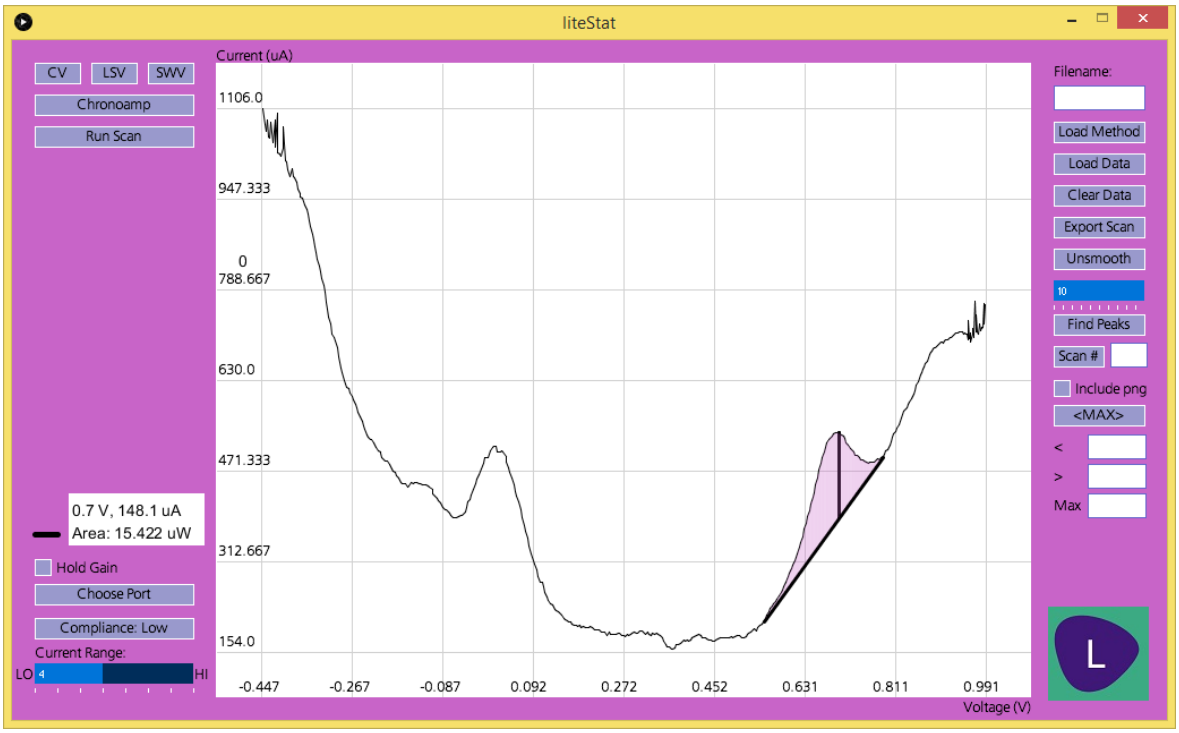


Figure 5- 9: SWV of 41.40 µM MDMA. Peak shown at approx. 0.7 V was used for quantitation.

Figure 5-9 shows the 41.40 µM MDMA sample with the shaded area of the peak integration indicating the oxidation peak of the analyte. Each standard was run in triplicate and an average was taken for the calibration graph in Figure 5-10. Further to this a 10 coefficient Savitsky-Golay smoothing filter was applied to all the data.

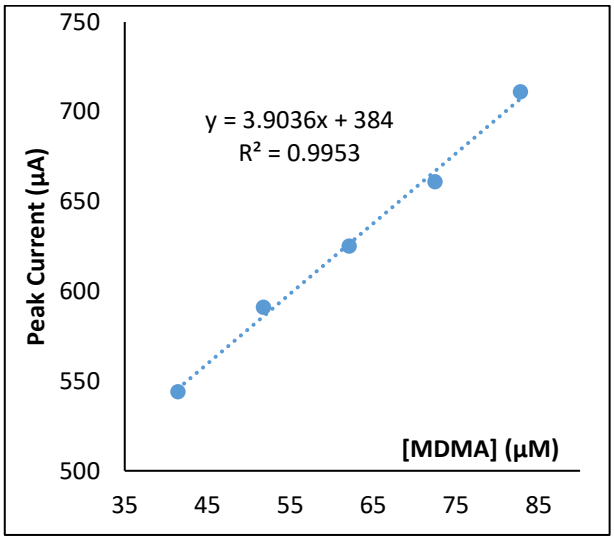


Figure 5- 10: MDMA 40 mM pH 9.5 BR buffer SWV calibration using Paste 6 electrode with Prototype Model B

The R^2 value of 0.9953 clearly demonstrated linearity over the range of concentrations applied. This was a fairly narrow range, however the behaviour of the analyte at the electrode surface can often skew analyte calibrations over larger ranges. As it was really the prototype B that was being tested, it was decided to use a large enough range that could demonstrate if the equipment was working, without inference from the chemical effects, at the electrode surface.

6. Conclusion and Further Work

6.1. Overview

A proof-of-concept analytical device has been developed and been shown to have applicability to forensic electrochemistry. This is a rather niche application, although it is growing over time as Table 3-2 proves. Nevertheless, the device could be applied to a range of other analyses *e.g.* corrosion analysis or battery testing. Technological solutions for potentiostats are already available. However, if a device is designed with a specific application in mind, there will be an emphasis on which parameters are important which will be different in different fields. The device (and its application) demonstrated here has been designed from the ground up and although further work is necessary before it is a viable product to aid the provision of forensic science, major strides have been taken towards this end.

6.2. Objective Mapping

Through the summary of the scientific literature in chapters 2 and 3, the general applicability of the technique to forensic analysis has been demonstrated. This summary represents the state of the scientific literature in 2019. Although still a young field, it is continuously growing (Figure 6-1).

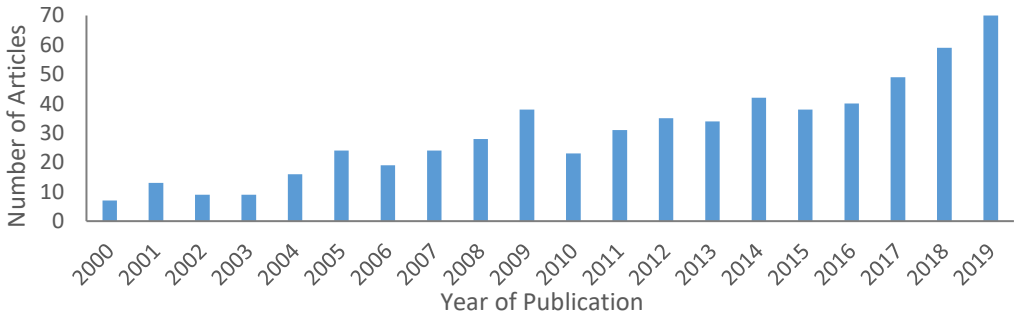


Figure 6- 1: Number of articles published on Science Direct database which contain the key terms "forensic" and "electrochemistry" over time.

Subsequent analysis of benzylpiperazine by voltammetry for the first time, demonstrated the wider applicability of the technique. It is pragmatic to compartmentalise the different aspects of an electroanalytical system. Firstly, there is the chemical aspect involving the electrode material and background electrolyte. Whilst this was not the main focus of the project, it is an essential component of the analysis and so cannot be ignored. Through the development of the carbon paste electrode and buffer system, a limit of quantification of 20 μM and resolution against the most commonly abused drug of its type was achieved.

The second important aspect relates to the hardware necessary to achieve the analysis *i.e.* the potentiostat itself. Often analysts are happy to rely on whatever instruments are commercially available, but it was a core characteristic of this project that entire system be developed. This was done with a view to future commercialisation. The design choices of the two prototype potentiostats are described in chapter 4. Tests on dummy cells and ferricyanide solutions did indeed prove that they were performing correctly. The second prototype improved the quality of the signal, by RC filtering and expanded the compliance range from ± 2.5 V to ± 12 V. The source code files for the firmware written in the language C, is given in appendices 4 to 13.

When viewing an analytical system as a whole, the user control interface cannot be ignored. This is the final aspect of the compartmentalised system. As the language Processing 3 has easy to implement libraries for displaying graphical information as well as serial communication, it was chosen to build the graphical user interface. Chapter 5 has screen captures of the software that was developed. Additional filtering features, such as Savitsky-Golay, were added here rather than on the potentiostat microcontroller to take advantage of

the far greater processing power and memory of the controlling computer. The source code file for the GUI is given in appendix 14. Data is stored as comma separated value files which users can save and load and find peak areas, *etc*. Chapter 5 also includes the enormous volume of test data that was performed on the system in a summarised form. Here the essential measurement parameters were robustly tested resulting in a dataset of 14320 scans. Obviously, automation was necessary for both the running of the scans and their subsequent interpretation. Chapter 5 also explains the packages composed in Processing 3 which were used to ascertain the best measurement parameters by finding the maximum R^2 and minimum sum of the square of residuals for linear scans run on the dummy cell. The overall conclusion from this work was that a minimum wait time of 10 ms was advisable (preferably greater) and that no algorithm was a substitute for number of measurements.

Finally, a successful SWV analysis of MDMA was reported using the new system, whereby there was an R^2 value of 0.995 over the concentration range 40 to 80 μM.

6.3. Future Work

The main drawback of the project was the difference in signal that is measured during CV analysis between prototype B and the commercially available instrument. It should be noted however that the PGStat128N retails for approximately £10,000 whilst the cost of manufacture of the prototype one was below £200.

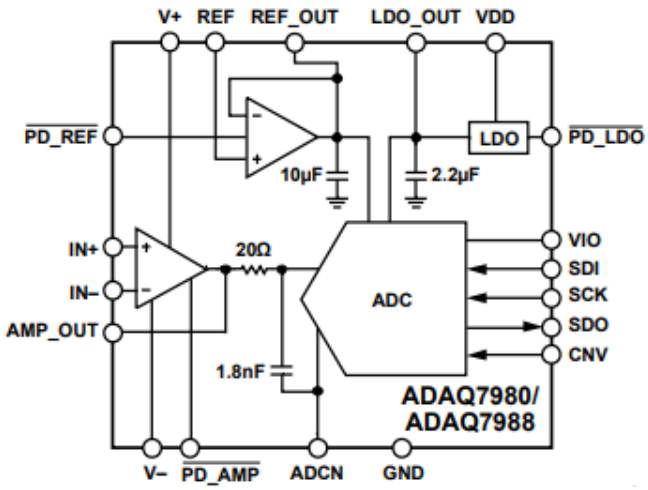


Figure 6- 2: Functional diagram of ADAQ7980 (229)

One step that could be taken for future work would be to use a different ADC; one with higher resolution. The micromodule ADAQ7980 (Analog Devices, (229)) is a 16-bit data acquisition system that boasts a sampling rate of 1 MHz. It is available in a 5x4 mm sized package and retails for under £20 at the time of writing. As Figure 6-1 shows, the functional diagram of the module includes its own op amps to both drive the ADC and buffer the reference. This follows a trend in the industry, where greater functionality is now being offered as "modules".

Using the higher resolution and stabilisation of the signal would mean less samples would need to be taken per measurement and the subsequent voltammograms would more closely resemble the "true" value that is provided by instruments that are currently commercially available.

Whilst the objectives of this project have been achieved, the analysis of a single compound is insufficient to characterise the performance of the system. Future work would inevitably involve the analysis of a wider range of controlled drugs.

7. References

(1) Zuman P. Electrolysis with a Dropping Mercury Electrode: J. Heyrovský's Contribution to Electrochemistry. *Critical Reviews in Analytical Chemistry.* 2001; 31(4):281-289 http://dx.doi.org/10.1080/20014091076767.

(2) Fatouros N, Krulic D. *Conditions for a nearly perfect match between pulse voltammetry and linear scan voltammetry.* 2018ID: 271406.

(3) Javanbakht M, Divsar F, Badiei A, Fatollahi F, Khaniani Y, Ganjali MR, et al. Determination of picomolar silver concentrations by differential pulse anodic stripping voltammetry at a carbon paste electrode modified with phenylthiourea-functionalized high ordered nanoporous silica gel. *Electrochimica Acta.* 2009; 54(23):5381-5386.

(4) Lu Y, Liang X, Niyungeko C, Zhou J, Xu J, Tian G. *A review of the identification and detection of heavy metal ions in the environment by voltammetry.* 2018ID: 271360.

(5) Guo X, Song Z, Tian X, Song J. Single-Sweep Voltammetric Determination of Tamoxifen at Carbon Paste Electrode. *Analytical Letters.* 2008; 41(7):1225-1235.

(6) Pemberton RM, Mottram TT, Hart JP. Development of a screen-printed carbon electrochemical immunosensor for picomolar concentrations of estradiol in human serum extracts. *Journal of Biochemical and Biophysical Methods.* 2005; 63(3):201-212.

(7) Sipa K, Brycht M, Skrzypek S. *The effect of the supporting electrolyte on the voltammetric determination of the veterinary drug nitroxinil.* 2018ID: 271406.

(8) Wang J. Nanoparticle-based electrochemical DNA detection. *Analytica Chimica Acta.* 2003; 500(1–2):247-257.

(9) Chai H, Xu J, Xu J, Ding S, Tang Y, Miao P. *Star trigon structure-aided DNA walker for amplified electrochemical detection of DNA.* 2019ID: 272020.

(10) Rogers EI, Lawrence NS, Compton RG. The electrochemical oxidation of ruthenocene in various room temperature ionic liquids. *Journal of Electroanalytical Chemistry.* 2011; 657(1–2):144-149.

(11) Testa AC, Reinmuth WH. Stepwise Reactions in Chronopotentiometry. *Analytical Chemistry.* 1961; 33(10):1320-1324.

(12) Brett CMA, Garcia MBQ, Lima JFC. On the suppression of zinc-copper interactions in square wave anodic stripping voltammetry in flowing solution by addition of gallium ions. *Analytica Chimica Acta.* 1997; 339(1–2):167-172.

(13) Harris DC. *Quantitiative Chemical Analysis.* 7th ed. : W.H. Freeman & Company; 2008.

(14) Wang S, Wang J, Gao Y. Development and Use of an Open-Source, User-Friendly Package To Simulate Voltammetry Experiments. *Journal of Chemical Education.* 2017; 94(10):1567--1570.

(15) Yu W, Yang J, Wu X, Gu Y, Xiao J, Yu J, et al. *Study on dewaterability limit and energy consumption in sewage sludge electro-dewatering by in-situ linear sweep voltammetry analysis.* 2017ID: 271942.

(16) Moya AA, Sistat P. *Reaching the limiting current regime by linear sweep voltammetry in ion-exchange membrane systems.* 2018ID: 271357.

(17) DeMars RD, Shain I. Anodic Stripping Voltammetry Using the Hanging Mercury Drop Electrode. *Analytical Chemistry.* 1957; 29(12):1825-1827.

(18) Mann CK. Cyclic Stationary Electrode Voltammetry of Some Aliphatic Amines. *Analytical Chemistry.* 1964; 36(13):2424-2426.

(19) Ozkan SA. Principles and Techniques of Electroanalytical Stripping Methods for Pharmaceutically Active Compounds in Dosage Forms and Biological Samples. *Current Pharmaceutical Analysis.* 2009; 5(2):127-143.

(20) Zahir F, Rizwi SJ, Haq SK, Khan RH. Low dose mercury toxicity and human health. *Environmental toxicology and pharmacology.* 2005; 20(2):351-360.

(21) Moros SA. Mercury-Film Electrode for Precision Voltammetry. *Analytical Chemistry.* 1962; 34(12):1584-1587.

(22) Maleki N, Absalan G, Safavi A, Farjami E. Ultra trace adsorptive stripping voltammetric determination of atrazine in soil and water using mercury film electrode. *Analytica Chimica Acta.* 2007; 581(1):37-41.

(23) Lu T, Sun I. Electrocatalytic determination of paraquat using a nafion film coated glassy carbon electrode. *Talanta.* 2000; 53(2):443-451.

(24) Švancara I, Vytřas K, Barek J, Zima J. Carbon Paste Electrodes in Modern Electroanalysis. *Critical Reviews in Analytical Chemistry.* 2001; 31(4):311-345.

(25) Adams RN. Carbon Paste Electrodes. *Analytical Chemistry.* 1958; 30(9):1576-1576.

(26) Mesarić S, Dahmen EAMF. Ion-selective carbon-paste electrodes for halides and silver(I) ions. *Analytica Chimica Acta.* 1973; 64(3):431-438.

(27) Yao T, Musha S. Electrochemical enzymatic determinations of ethanol and l-lactic acid with a carbon paste electrode modified chemically with nicotinamide adenine dinucleotide. *Analytica Chimica Acta.* 1979; 110(2):203-209.

(28) Khodari M, Kauffmann J, Patriarche GJ, Ghandour MA. Applications in drug analysis of carbon paste electrodes modified by fatty acids. *Journal of pharmaceutical and biomedical analysis.* 1989; 7(12):1491-1497.

(29) Arcos J, Kauffmann J-, Patriarche GJ, Sanchez-Batanero P. Voltammetric determination of Celiptium with carbon paste and lipid-modified carbon paste electrodes. *Analytica Chimica Acta.* 1990; 236(0):299-305.

(30) Zima J, Švancara I, Barek J, Vytřas K. Recent Advances in Electroanalysis of Organic Compounds at Carbon Paste Electrodes. *Critical Reviews in Analytical Chemistry.* 2009; 39(3):204-227.

(31) Forzani ES, Rivas GA, Solis VM. Amperometric determination of dopamine on an enzymatically modified carbon paste electrode. *Journal of Electroanalytical Chemistry.* 1995; 382(1–2):33-40.

(32) Yabuki S, Mizutani F. Modifications to a carbon paste glucose-sensing enzyme electrode and a reduction in the electrochemical interference from L-ascorbate. *Biosensors and Bioelectronics.* 1995; 10(3–4):353-358.

(33) Eggins B. *Biosensors: An Introduction.* 1st ed. Chichester, England: Wiley Teubner; 1996.

(34) Bugarinović JP, Pešić MS, Minić A, Katanić J, Ilić-Komatina D, Pejović A, et al. *Ferrocene-containing tetrahydropyrazolopyrazolones: Antioxidant and antimicrobial activity.* 2018ID: 271941.

(35) King AJ, Zatsikha YV, Blessener T, Dalbec F, Goff PC, Kayser M, et al. *Ultrafast electron-transfer in a fully conjugated coumarin-ferrocene donor-acceptor dyads.* 2019ID: 271394.

(36) Metrohmusa.com. *SCREEN-PRINTED ELECTRODES (SPEs).* [homepage on the Internet]. cited 2012 10th August]. Available from: http://www.metrohmusa.com/Products/Echem/DropSens-electrodes/Screen-Printed-Electrodes/index.html.

(37) Monk PMS. *Fundamentals of Electroanalytical Chemistry.* Chichester: Wiley; 2001.

(38) Wang J, Pamidi PVA. Disposable screen-printed electrodes for monitoring hydrazines. *Talanta.* 1995; 42(3):463-467.

(39) Mersal GAM. Electrochemical Sensor for Voltammetric Determination of Catechol Based on Screen Printed Graphite Electrode. *International Journal of Electrochemical Science.* 2009; 4:1167 - 1177.

(40) Choudhary V, Gupta A. Polymer/Carbon Nanotube Nanocomposites. ; 2011.

(41) Iijima S. Helical microtubules of graphitic carbon. *Nature.* 1991; 354(6348):56-58.

(42) Popov VN. Carbon nanotubes: properties and application. *Materials Science and Engineering: R: Reports.* 2004; 43(3):61-102.

(43) Durrer L, Helbling T, Zenger C, Jungen A, Stampfer C, Hierold C. SWNT growth by CVD on Ferritin-based iron catalyst nanoparticles towards CNT sensors. *Sensors and Actuators B: Chemical.* 2008; 132(2):485-490.

(44) Agüí L, Yáñez-Sedeño P, Pingarrón JM. Role of carbon nanotubes in electroanalytical chemistry: A review. *Analytica Chimica Acta.* 2008; 622(1–2):11-47.

(45) Hu C, Hu S. Carbon Nanotube-Based Electrochemical Sensors: Principles and Applications in Biomedical Systems. *Journal of Sensors.* 2009; 2009(Article ID 187615).

(46) Delgado JL, Herranz MA, Martin N. The Nano-forms of Carbon. *Journal of Materials Chemistry.* 2008; 18:1417 - 1426.

(47) Roy N, Sengupta R, Bhowmick AK. Modifications of carbon for polymer composites and nanocomposites. *Progress in Polymer Science.* 2012; 37(6):781-819.

(48) Bajpai A, Gorantla S, Löffler M, Hampel S, Rümmeli MH, Thomas J, et al. The filling of carbon nanotubes with magnetoelectric Cr2O3. *Carbon.* 2012; 50(4):1706-1709.

(49) Jiang H, Zhang A, Sun Y, Ru X, Ge D, Shi W. Poly(1-(2-carboxyethyl)pyrrole)/polypyrrole composite nanowires for glucose biosensor. *Electrochimica Acta.* 2012; 70(0):278-285.

(50) Reza Shishehbore M, Zare HR, Nematollahi D. Electrocatalytic determination of morphine at the surface of a carbon paste electrode spiked with a hydroquinone derivative and carbon nanotubes. *Journal of Electroanalytical Chemistry.* 2012; 665(0):45-51.

(51) Desai PB, Srivastava AK. Determination of amiloride at Nafion–CNT-nano-composite film sensor employing adsorptive stripping differential pulse voltammetry. *Sensors and Actuators B: Chemical.* 2012; 169(0):341-348.

(52) Sun Y, Wang S, Zhang X, Huang Y. Simultaneous determination of epinephrine and ascorbic acid at the electrochemical sensor of triazole SAM modified gold electrode. *Sensors and Actuators B: Chemical.* 2006; 113(1):156-161.

(53) Motaghedifard M, Ghoreishi SM, Behpour M, Moghadam Z, Salavati-Niasari M. Electrochemical study of new self-assembled monolayer of 2-hydroxy-N'1-[(E)-1-(3-methyl-2-thienyl) methylidene] benzohydrazide on gold electrode as an epinephrine sensor element. *Journal of Electroanalytical Chemistry.* 2012; 682(0):14-22.

(54) Xu X, Makaraviciute A, Pettersson J, Zhang S, Nyholm L, Zhang Z. *Revisiting the factors influencing gold electrodes prepared using cyclic voltammetry.* 2019ID: 271353.

(55) Sopha H, Hocevar SB, Pihlar B, Ogorevc B. Bismuth film electrode for stripping voltammetric measurement of sildenafil citrate. *Electrochimica Acta.* 2012; 60(0):274-277.

(56) Rybakiewicz R, Skorka L, Louarn G, Ganczarczyk R, Zagorska M, Pron A. *N-substituted dithienopyrroles as electrochemically active monomers: Synthesis, electropolymerization and spectroelectrochemistry of the polymerization products.* 2019ID: 271355.

(57) Macpherson JV. A practical guide to using boron doped diamond in electrochemical research. *Physical Chemistry Chemical Physics.* 2015; 17:2935--2949.

(58) Afzali M, Mostafavi A, Nekooie R, Jahromi Z. A *novel voltammetric sensor based on palladium nanoparticles/carbon nanofibers/ionic liquid modified carbon paste electrode for sensitive determination of anti-cancer drug pemetrexed.* 2019ID: 271359.

(59) Nagashima H, Kondo T, Nagoya T, Ikeda T, Kurimata N, Unoke S, et al. *Identification of chemical warfare agents from vapor samples using a field-portable capillary gas chromatography/membrane-interfaced electron ionization quadrupole mass spectrometry instrument with Tri-Bed concentrator.* 2015ID: 271409.

(60) Qafsaoui W, Kendig MW, Perrot H, Takenouti H. Coupling of electrochemical techniques to study copper corrosion inhibition in 0.5 mol L−1 NaCl by 1-pyrrolidine dithiocarbamate. *Electrochimica Acta.* 2013; 87(0):348-360.

(61) Rutkowska IA, Koster MD, Blanchard GJ, Kulesza PJ. Enhancement of ethanol oxidation at Pt and PtRu nanoparticles dispersed over hybrid zirconia-rhodium supports. *Journal of Power Sources.* 2014; 272(0):681-688.

(62) Kalimuthu P, Heath MD, Santini JM, Kappler U, Bernhardt PV. Electrochemically driven catalysis of Rhizobium sp. NT-26 arsenite oxidase with its native electron acceptor cytochrome c552. *Biochimica et Biophysica Acta (BBA) - Bioenergetics.* 2014; 1837(1):112-120.

(63) Shah AA. UV-Vis Spectroelectrochemistry. [ID: 192]. Reproduced In: Kreysa G, Ota K, Savinell RF, editors. *Encyclopedia of Applied Electrochemistry.* New York: Springer; 2014. p. 2099-2102.

(64) Olech K, Sołoducho J, Laba K, Data P, Lapkowski M, Roszak S. The Synthesis and Characterization of -3,4-Ethylenedioxythiophene Derivatives with Electroactive Features. *Electrochimica Acta.* 2014; 141(0):349-356.

(65) Matsuda H, Ayabe Y. Zur Theorie der Randles-Sevčikschen Kathodenstrahl-Polarographie. *Zeitschrift für Elektrochemie.* 1955; 59:494-503.

(66) Compton RG, Banks CE. Cyclic Voltammetry at Macroelectrodes. [ID: 195]. Reproduced *Understanding Voltammetry.* 2nd ed. London: Imperial College Press; 2011. p. 119-120.

(67) Mizoguchi T, Adams RN. Anodic oxidation studies of N,N-dimethylaniline. I. Voltammetric and spectroscopic investigations at platinum electrodes. *Journal of the American Chemical Society.* 1962; 84(11):2058-2061.

(68) Vetter KJ. *Transactions of the Symposium on Electrode Processes.* New York: John Wiley and Sons; 1961.

(69) Galus Z, Adams RN. Anodic oxidation studies of N,N-dimethylaniline. II. Stationary and rotated disk studies at inert electrodes. *Journal of the American Chemical Society.* 1962; 84(11):2061-2065.

(70) Mark HB, Anson FC. Electro-oxidation of Phenylenediamines and Related Compounds at Platinum Electrodes. *Analytical Chemistry.* 1963; 35(6):722-724.

(71) EMCDDA. [ID: 92]. *Report on the risk assessment of BZP in the framework of the Council decision on new psychoactive substances.* European Monitioring Centre for Drugs and Drug Addiction; 2009.

(72) European Monitoring Centre for Drugs and Drug Addiction. [ID: 87]. *Council decision: 'Appropriate controls' for BZP - new drug BZP to be placed under control across the EU. News Release.* Lisbon: European Monitoring Centre for Drugs and Drug Addiction; 2008.

(73) Lee HY, Adams RN. Anodic Voltammetry and EPR Studies of Isomeric Phenylenediamines. *Analytical Chemistry.* 1962; 34(12):1587-1590.

(74) Lide DR. *The Handbook of Chemistry and Physics.* 82nd ed. London: CRC Press; 2001.

(75) Masui M, Sayo H, Tsuda Y. Anodic oxidation of amines. Part I. Cyclic voltammetry of aliphatic amines at a stationary glassy-carbon electrode. *Journal of the Chemical Society B: Physical Organic.* 1968; :973-976.

(76) Portis LC, Bhat VV, Mann CK. Electrochemical Dealkylation of Aliphatic Tertiary and Secondary Amines. *Journal of Organic Chemistry.* 1970; 35(7):2175--2178.

(77) Masaichiro M, Sayo H. Anodic oxidation of Amines. Part II. Electrochemical Dealkylation of Aliphatic Tertiary Amines. *Journal of the Chemical Society B: Physical Organic.* 1971; :1593--1596.

(78) Laube BL, Asirvatham MR, Mann CK. Electrochemical Oxidation of Tropanes. *Journal of Organic Chemistry.* 1977; 42(4):670--674.

(79) Seo ET, Nelson RF, Fritsch JM, Marcoux LS, Leedy DW, Adams RN. Anodic Oxidation Pathways of Aromatic Amines. Electrochemical and Electron Paramagnetic Resonance Studies. *Journal of the American Chemical Society.* 1966; 88(15):3498-3503.

(80) Nelson RF, Adams RN. Anodic oxidation pathways of substituted triphenylamines. II. Quantitative studies of benzidine formation. *Journal of the American Chemical Society.* 1968; 90(15):3925-3930.

(81) Leedy DW, Adams RN. Anodic Oxidation Pathways of Aromatic Amines. IV. Diphenylamine Systems in Aqueous Acid Solution. *Journal of the American Chemical Society.* 1970; 92(6):1646--1650.

(82) Portis LC, Klug J. T., Mann CK. Electrochemical Oxidation of Some Phenethylamines. *Journal of Organic Chemistry.* 1974; 39(24):3488-3494.

(83) Larumbe D, Moreno M, Gallardo I, Betrán J, Andrieux CP. Theoretical study of the Oxidation Mechansim of Aromatic Amines. *Journal of the Chemical Society, Perkin Transactions 2.* 1991; :1437--1443.

(84) Keller T, Mutz M, Aderjan R, Latscha R.P. Study of the electrochemical behaviour of amphetamine and its derivatives in aqueous solution. *Journal of Analytical Chemistry.* 1999; 363:270 - 276.

(85) Squella JA, Cassels BK, Arata M, Bavestrello MP, Nuñez-Vergara LJ. Electrochemical oxidation of methylenedioxyamphetamines. *Talanta.* 1993; 40(9):1379-1384.

(86) Brunt TM, Niesink RJM, van den Brink W. Impact of a transient instability of the ecstasy market on health concerns and drug use patterns in The Netherlands. *International Journal of Drug Policy.* 2012; 23(2):134-140.

(87) United Nations Office on Drugs and Crime. *World Drug Report.* Vienna: UNODC; 2018.

(88) Arbo MD, Bastos ML, Carmo HF. Piperazine compounds as drugs of abuse. *Drug and alcohol dependence.* 2012; 122(3):174-185.

(89) Sá Monteiro M, de Lourdes Bastos M, de Pinho PG, Carvalho M. Update on 1-benzylpiperazine (BZP) part pills. *Archives of Toxicology.* 2013; 87:929--947.

(90) *Misuse of Drugs Act 1971 (Amendment) Order 2009 2009.*

(91) Philp M, Shimmon R, Stojanovska N, Tahtouh M, Fu S. Development and validation of a presumptive colour spot test method for the detection of piperazine analogues in seized illict material. *Analytical Methods.* 2013; 5(20):5402--5410.

(92) Shulgin AT, MacLean DE. Illicit Synthesis of Phencyclidine (PCP) and Several of Its Analogs. *Clinical toxicology.* 1976; 9(4):553--560.

(93) Kovar K and Laudszun M. [ID: 118]. *Chemsitry and reaction mechanisms of rapid tests for drugs of abuse and precursors chemicals.* Pharmazeutisches Institut der Universität Tübingen, Auf der Morgenstelle 8, D-7400 Tübingen, Federal Republic of Germany: United Nations; 1989.

(94) Tsutsumi H, Katagi M, Miki A, Shima N, Kamata T, Nishikawa M, et al. Development of simultaneous gas chromatography–mass spectrometric and liquid chromatography–electrospray ionization mass spectrometric determination method for the new designer drugs, N-benzylpiperazine (BZP), 1-(3-trifluoromethylphenyl)piperazine (TFMPP) and their main metabolites in urine. *Journal of Chromatography B.* 2005; 819(2):315-322.

(95) Takahashi M, Nagashima M, Suzuki J, Seto T, Yasuda I, Yoshida T. Creation and application of psychoactive designer drugs data library using liquid chromatography with photodiode array spectrophotometry detector and gas chromatography–mass spectrometry. *Talanta.* 2009; 77(4):1245-1272.

(96) Antia U, Tingle MD, Russell BR. Validation of an LC-MS Method for the Detection and Quantification of BZP and TFMPP and their Hydroxylated Metabolites in Human Plasma and its Application to the Pharmacokinetic Study of TFMPP in Humans*. *Journal of forensic sciences.* 2010; 55(5):1311-1318.

(97) Wohlfarth A, Weinmann W, Dresen S. LC-MS/MS screening method for desinger amphetamines, tryptamines, and piperazines. *Analytical and Bioanalytical Chemistry.* 2010; 396(7):2403-2414.

(98) Chen C, Kostakis C, Irvine RJ, White JM. Increases in use of novel synthetic stimulant are not directly linked to decreased use of 3,4-methylenedioxy-N-methylamphetamine (MDMA). *Forensic science international.* 2013; 231(1–3):278-283.

(99) Boumrah Y, Rosset M, Lecompte Y, Bouanani S, Khimeche K, Dahmani A. Development of a targeted GC/MS screening method and validation of an HPLC/DAD quantification method for piperazines–amphetamines mixtures in seized material. *Egyptian Journal of Forensic Sciences.* 2014; 4(3):90-99.

(100) Dickson AJ, Vorce SP, Holler JM, Lyons TP. Detection of 1-Benzylpiperazine, 1-(3-Trifluoromethylphenyl)-piperazine, and 1-(3-Chlorophenyl)-piperazine in 3,4-Methylenedioxymethamphetamine-Positive Urine Samples. *Journal of analytical toxicology.* 2010; 34:464--469.

(101) Concheiro M, Castaneto M, Kronstrand R, Huestis MA. Simultaneous determination of 40 novel psychoactive stimulants in urine by liquid chromatography–high resolution mass spectrometry and library matching. *Journal of Chromatography A.* 2015; 1397:32-42.

(102) Strano-Rossi S, Odoardi S, Fisichella M, Anzillotti L, Gottardo R, Tagliaro F. Screening for new psychoactive substances in hair by ultrahigh performance liquid chromatography–electrospray ionization tandem mass spectrometry. *Journal of Chromatography A.* 2014; 1372:145-156.

(103) Beckett NM, Cresswell SL, Grice DI, Carter JF. Isotopic profiling of seized benzylpiperazine and trifluoromethylphenylpiperazine tablets using δ13C and δ15N stable isotopes. *Science & Justice.* 2015; 55(1):51-56.

(104) Kuleya C, Hall S, Gautam L, Cole MD. *An optimised gas chromatographic-mass spectrometric method for the chemical characterisation of benzylpiperazine and 1-arylpiperazine based drugs.* The Royal Society of Chemistry; 2014ID: 212.

(105) Mwenesongole EM, Gautam L, Hall SW, Waterhouse JW, Cole MD. *Simultaneous detection of controlled substances in waste water.* The Royal Society of Chemistry; 2013ID: 213.

(106) Elie LE, Baron MG, Croxton RS, Elie MP. Reversing microcrystalline tests—An analytical approach to recycling of microcrystals from drugs of abuse. *Forensic science international.* 2011; 207(1–3):e55-e58.

(107) Elie L, Baron M, Croxton R, Elie M. Microcrystalline identification of selected designer drugs. *Forensic science international.* 2012; 214(1–3):182-188.

(108) Nelson ME, Bryant SM, Aks SE. Emerging drugs of abuse. *Disease-a-Month.* 2014; 60(3):110-132.

(109) de Boer D, Bosman IJ, Hidvégi E, Manzoni C, Benkö AA, dos Reys LJAL, et al. Piperazine-like compounds: a new group of designer drugs-of-abuse on the European market. *Forensic science international.* 2001; 121(1–2):47-56.

(110) Castaneto MS, Barnes AJ, Concheiro M, Klette KL, Martin TA, Huestis MA. Biochip array technology immunoassay performance and quantitative confirmation of designer piperazines for urine workplace drug testing. *Analytical and Bioanalytical Chemistry.* 2015; 407(16):4639-4648.

(111) Fitzsimons MG, Ishizawa Y, Baker KH. Drug testing physicians for substances of abuse: case report of a false-positive result. *Journal of clinical anesthesia.* 2013; 25(8):669-671.

(112) Choodum A, Nic Daeid N. Rapid and semi-quantitative presumptive tests for opiate drugs. *Talanta.* 2011; 86:284-292.

(113) Wang J. Electrochemical glucose biosensors. *Chemical reviews.* 2008; 108(2):814-825.

(114) Kriikku P, Wilhelm L, Jenckel S, Rintatalo J, Hurme J, Kramer J, et al. Comparison of breath-alcohol screening test results with venous blood alcohol concentration in suspected drunken drivers. *Forensic science international.* 2014; 239(0):57-61.

(115) Skládal P. 5 - Electrochemical detection for biological identification. [ID: 180]. Reproduced In: Schaudies RP, editor. *Biological Identification.* : Woodhead Publishing; 2014. p. 131-152.

(116) Monticelli D, Laglera LM, Caprara S. Miniaturization in voltammetry: Ultratrace element analysis and speciation with twenty-fold sample size reduction. *Talanta.* 2014; 128(0):273-277.

(117) Shervedani RK, Rezvaninia Z, Sabzyan H, Zali Boeini H. Characterization of gold-thiol-8-hydroxyquinoline self-assembled monolayers for selective recognition of aluminum ion using voltammetry and electrochemical impedance spectroscopy. *Analytica Chimica Acta.* 2014; 825(0):34-41.

(118) Kaur B, Srivastava R. Simultaneous electrochemical determination of nanomolar concentrations of aminophenol isomers using nanocrystalline zirconosilicate modified carbon paste electrode. *Electrochimica Acta.* 2014; 141(0):61-71.

(119) Liu Z, Huang X. Voltammetric determination of inorganic arsenic. *TrAC Trends in Analytical Chemistry.* 2014; 60(0):25-35.

(120) Smith JP, Randviir EP, Banks CE. An introduction to Forensic Electrochemistry. [ID: 226]. Reproduced In: Katz E, Halámek J, editors. *Forensic Science: A Multidisciplinary Approach.* Weinheim, Germany: Wiley-VCH; 2016. p. 89-102.

(121) Bandodkar AJ, O'Mahony AM, Ramirez J, Samek IA, Anderson SM, Windmiller JR, et al. Solid-state Forensic Finger sensor for integrated sampling and detection of gunshot residue and explosives: towards 'Lab-on-a-finger'. *Analyst.* (18):5288-5295.

(122) Niazi A, Ghasemi J, Zendehdel M. Simultaneous voltammetric determination of morphine and noscapine by adsorptive differential pulse stripping method and least-squares support vector machines. *Talanta.* 2007; 74(2):247-254.

(123) Li F, Song J, Gao D, Zhang Q, Han D, Niu L. Simple and rapid voltammetric determination of morphine at electrochemically pretreated glassy carbon electrodes. *Talanta.* 2009; 79(3):845-850.

(124) Macedo C, Branco P, Ferreira L, Lobo A, Capela J, Fernandes E, et al. Synthesis and Cyclic Voltammetry Studies of 3,4-Methylenedioxymethylamphetamine (MDMA) Human Metabolites. *Journal of Health Sciences.* 2007; 53(1):31 - 42.

(125) Oiye ÉN, Figueiredo NBd, Andrade JFd, Tristão HM, Oliveira MFd. Voltammetric determination of cocaine in confiscated samples using a cobalt hexacyanoferrate film-modified electrode. *Forensic science international.* 2009; 192(1–3):94-97.

(126) Yılmaz S. Adsorptive stripping voltammetric determination of zopiclone in tablet dosage forms and human urine. *Colloids and Surfaces B: Biointerfaces.* 2009; 71(1):79-83.

(127) de Carvalho LM, Correia D, Garcia SC, de Bairros AV, Nascimento PCd, Bohrer D. A new method for the simultaneous determination of 1,4-benzodiazepines and amfepramone as adulterants in phytotherapeutic formulations by voltammetry. *Forensic science international.* 2010; 202(1–3):75-81.

(128) Garrido EMPJ, Garrido JMPJ, Milhazes N, Borges F, Oliveira-Brett AM. Electrochemical oxidation of amphetamine-like drugs and application to electroanalysis of ecstasy in human serum. *Bioelectrochemistry.* 2010; 79(1):77-83.

(129) Tang K, Liu L, Hong Y, Hu J, Zhang J, Cao C. Fabrication of indium tin oxides (ITO)-supported poly(3,4-ethylenedioxythiophene) electrodes coated with active IrO2 layer for morphine electrooxidation. *Journal of Applied Electrochemistry.* 2010; 40(9):1699-1704.

(130) Du Y, Chen C, Yin J, Li B, Zhou M, Dong S, et al. Solid-State Probe Based Electrochemical Aptasensor for Cocaine: A Potentially Convenient, Sensitive, Repeatable, and Integrated Sensing Platform for Drugs. *Analytical Chemistry.* 2010; 82(4):1556-1563.

(131) Ensafi A, Rezaei B, Krimi-Maleh H. An ionic liquid-type multiwall carbon nanotubes paste electrode for electrochemical investigation and determination of morphine. *Ionics.* 2011; 17(7):659-668.

(132) Yang G, Chen Y, Li L, Yang Y. Direct electrochemical determination of morphine on a novel gold nanotube arrays electrode. *Clinica Chimica Acta.* 2011; 412(17–18):1544-1549.

(133) Wen Y, Pei H, Wan Y, Su Y, Huang Q, Song S, et al. DNA Nanostructure-Decorated Surfaces for Enhanced Aptamer-Target Binding and Electrochemical Cocaine Sensors. *Analytical Chemistry.* 2011; 83(19):7418-7423.

(134) Câmpean A, Tertiş M, Săndulescu R. Voltammetric determination of some alkaloids and other compounds in pharmaceuticals and urine using an electrochemically activated glassy carbon electrode. *Central European Journal of Chemistry.* 2011; 9(4):688-700.

(135) Atta NF, Galal A, Wassel AA, Ibrahim AH. Sensitive Electrochemical Determination of Morphine Using Gold-Nanoparticles-Ferrocene Modified Carbon Paste Electrode. *International Journal of Electrochemical Science.* 2012; 7:10501--10518.

(136) Ensafi AA, Izadi M, Rezaei B, Karimi-Maleh H. N-hexyl-3-methylimidazolium hexafluoro phosphate/multiwall carbon nanotubes paste electrode as a biosensor for voltammetric detection of morphine. *Journal of Molecular Liquids.* 2012; 174(0):42-47.

(137) Balbino MA, de Menezes MMT, Eleotério IC, Saczk AA, Okumura LL, Tristão HM, et al. Voltammetric determination of Δ9-THC in glassy carbon electrode: An important contribution to forensic electroanalysis. *Forensic science international.* 2012; 221(1–3):29-32.

(138) Navaee A, Salimi A, Teymourian H. Graphene nanosheets modified glassy carbon electrode for simultaneous detection of heroine, morphine and noscapine. *Biosensors and Bioelectronics.* 2012; 31(1):205-211.

(139) Hong Y, Hu J, Zhang J, Cao C. Enhanced Electrocatalytic Activity for Morphine Oxidation at 2-Aminoethanethiol Self-Assembled Monolayer (SAM)-Modified Gold Electrode. *Electrocatalysis.* 2013; 4(4):302-305.

(140) Arabali V, Sadeghi R. Surface properties of nano-Al2O3 film and its application in the preparation of morphine electrochemical sensor. *Ionics.* 2013; 19(12):1775-1782.

(141) Jiménez-Pérez R, Sevilla JM, Pineda T, Blázquez M, González-Rodríguez J. Electrochemical behaviour of gamma hydroxybutyric acid at a platinum electrode in acidic medium. *Electrochimica Acta.* 2013; 111(0):601-607.

(142) Li Y, Li K, Song G, Liu J, Zhang K, Ye B. Electrochemical behavior of codeine and its sensitive determination on graphene-based modified electrode. *Sensors and Actuators B: Chemical.* 2013; 182(0):401-407.

(143) de Oliveira LS, Balbino MA, de Menezes MMT, Dockal ER, de Oliveira MF. Voltammetric analysis of cocaine using platinum and glassy carbon electrodes chemically modified with Uranyl Schiff base films. *Microchemical Journal.* Accepted for publication 26 April 2013.

(144) Behpour M, Valipour A, Keshavarz M. Determination of buprenorphine by differential pulse voltammetry on carbon paste electrode using SDS as an enhancement factor. *Materials Science and Engineering: C.* 2014; 42(0):500-505.

(145) Sanati AL, Karimi-Maleh H, Badiei A, Biparva P, Ensafi AA. A voltammetric sensor based on NiO/CNTs ionic liquid carbon paste electrode for determination of morphine in the presence of diclofenac. *Materials Science and Engineering: C.* 2014; 35(0):379-385.

(146) Li Y, Zou L, Li Y, Li K, Ye B. A new voltammetric sensor for morphine detection based on electrochemically reduced MWNTs-doped graphene oxide composite film. *Sensors and Actuators B: Chemical.* 2014; 201(0):511-519.

(147) Ahmar H, Tabani H, Hossein Koruni M, Davarani SSH, Fakhari AR. A new platform for sensing urinary morphine based on carrier assisted electromembrane extraction followed by adsorptive stripping voltammetric detection on screen-printed electrode. *Biosensors and Bioelectronics.* 2014; 54(0):189-194.

(148) Afkhami A, Soltani-Felehgari F, Madrakian T. A sensitive electrochemical sensor for rapid determination of methadone in biological fluids using carbon paste electrode modified with gold nanofilm. *Talanta.* 2014; 128(0):203-210.

(149) Asturias-Arribas L, Alonso-Lomillo MA, Domínguez-Renedo O, Arcos-Martínez MJ. Sensitive and selective cocaine electrochemical detection using disposable sensors. *Analytica Chimica Acta.* 2014; 834(0):30-36.

(150) Tadini MC, Balbino MA, Eleoterio IC, de Oliveira LS, Dias LG, Jean-François Demets G, et al. Developing electrodes chemically modified with cucurbit[6]uril to detect 3,4-methylenedioxymethamphetamine (MDMA) by voltammetry. *Electrochimica Acta.* 2014; 121(0):188-193.

(151) Smith JP, Metters JP, Irving C, Sutcliffe OB, Banks CE. Forensic electrochemistry: the electroanalytical sensing of synthetic cathinone-derivatives and their accompanying adulterants in legal high products. *Analyst.* 2014; 139(2):389-400.

(152) Nigović B, Sadiković M, Sertić M. Multi-walled carbon nanotubes/Nafion composite film modified electrode as a sensor for simultaneous determination of ondansetron and morphine. *Talanta.* 2014; 122(0):187-194.

(153) Ensafi AA, Heydari-Bafrooei E, Rezaei B. Different interaction of codeine and morphine with DNA: A concept for simultaneous determination. *Biosensors and Bioelectronics.* 2013; 41(0):627-633.

(154) Roushani M, Shahdost-fard F. *A novel ultrasensitive aptasensor based on silver nanoparticles measured via enhanced voltammetric response of electrochemical reduction of riboflavin as redox probe for cocaine detection.* 2015ID: 271353.

(155) Oiye ÉN, Midori Toia Katayama J, Fernanda Muzetti Ribeiro M, de Oliveira MF. *Electrochemical analysis of 25H-NBOMe by Square Wave Voltammetry.* 2017ID: 314565.

(156) Scheel GL, de Oliveira FM, de Oliveira LLG, Medeiros RA, Nascentes CC, Tarley CRT. *Feasibility study of ethylone determination in seized samples using boron-doped diamond electrode associated with solid phase extraction.* 2018ID: 271353.

(157) Mendes LF, Souza e Silva ÂR, Bacil RP, Serrano SHP, Angnes L, Paixão TRLC, et al. *Forensic electrochemistry: Electrochemical study and quantification of xylazine in pharmaceutical and urine samples.* 2019ID: 271355.

(158) Bishop SC, McCord BR, Gratz SR, Loeligner JR, Witkowski MR. Simultaneous separation of different types of amphetamine and piperazine designer drugs by capillary electrophoresis with a chiral selector. *Journal of forensic sciences.* 2005; 50(2):325-35.

(159) Cohen BMZ, Butler R. BZP-party pills: A review of research on benzylpiperazine as a recreational drug. *International Journal of Drug Policy.* 2011; 22(2):95-101.

(160) PalmSens. *Portable potentiostat - PalmSens.* [homepage on the Internet]. The Netherlands: PalmSens; 15 March 2019 cited 2019 March 2019]. Available from: https://www.palmsens.com/potentiostat/portable-potentiostat/.

(161) Dobbelaere T, Vereecken PM, Detavernier C. *A USB-controlled potentiostat/galvanostat for thin-film battery characterization.* 2017ID: 314097.

(162) Morris J. C. *Analogue Electronics.* 2nd Edition ed. Great Britain: Arnold; 1999.

(163) Digi-key Electronics. *Linear - Amplifiers - Instrumentation, OP Amps, Buffer Amps | DigiKey.* [homepage on the Internet]. Thief River Falls, Minnesota, United States: Digi-key Electronics; 2016 [updated 09/08/2017; cited 2017 08/09]. Available from: https://www.digikey.co.uk/products/en/integrated-circuits-ics/linear-amplifiers-instrumentation-op-amps-buffer-amps/687?k=operational+amplifier&k=&pkeyword=operational+amplifier&pv41=421&pv41=135&pv41=112&FV=ffe002af&mnonly=0&ColumnSort=0&page=1&quantity=0&ptm=0&fid=0&pageSize=25.

(164) Microchip Technology Incorporated. *What Does "Rail-toRail" Operation Really Mean?* [homepage on the Internet]. Chandler, Arizona, United States: Microchip Technology Incorporated; 2004 [updated 2004; cited 2017 08/09]. Available from: http://www.microchip.com/stellent/groups/sitecomm_sg/documents/training_tutorials/en528065.pdf.

(165) Bard AJ, Faulkner LR. *Electrochemical Methods: Fundamentals and Application.* 2nd ed. New York: Wiley; 2001.

(166) Williams GE. *Analog Electronics: Devices, Circuits and Techniques.* St Paul, Minnesota, United States: West Publishing Company; 1996.

(167) Watson J. *Mastering Electronics.* 4th Edition ed. Hampshire, UK: MacMillan; 1996.

(168) Plant M. *Understand Electronics.* London, UK: Hodder Headline; 2010.

(169) Fitzpatrick D. *Chapter 14 - Noise Analysis.* Newnes; 2018ID: 318460.

(170) Microchip Technology Incorporated. *MCP4261 - Mixed Signal - Digital Potentiometers.* [homepage on the Internet]. Chandler, Arizona, United States: Microchip Technology Incorporated; 2017 cited 2017 08/09]. Available from: http://www.microchip.com/wwwproducts/en/MCP4261.

(171) Maxim Integrated. *MAX5250 Low-Power, Quad, 10-Bit Voltage-Output DAC with Serial Interface.* 19-1171; Rev 1; 10/02 ed. [monograph on the Internet]. 120 San Gabriel Drive, Sunnyvale, CA: Maxim Integrated; 2002 [cited 10/08/2017] Available from: https://datasheets.maximintegrated.com/en/ds/MAX5250.pdf.

(172) Atmel Corporation. *Atmel® Studio 7 — Easier to Use and More Powerful than Ever - Overview.* [homepage on the Internet]. San Jose, California, United States: Atmel Corporation; 2016 cited 2017 08/13]. Available from: http://www.atmel.com/microsite/atmel-studio/.

(173) Microchip Technology Incorporated. *MCP3302/04 13-Bit Differential Input, Low Power A/D Converter with SPI Serial Interface.* DS21697F ed. [monograph on the Internet]. Chandler, Arizona, United States: Microchip Technology Incorporated; 2011 [cited 13/08/2017] Available from: http://ww1.microchip.com/downloads/en/DeviceDoc/21697F.pdf.

(174) Fan H. *Effective method to improve linearity of high-resolution SAR ADC.* 2017ID: 244; ID: 271437.

(175) Vasović ND, Ristić GS. A new microcontroller-based RADFET dosimeter reader. *Radiation Measurements.* 2012; 47(4):272-276.

(176) S. T. Wu, Z. L. Wang. Equilateral Measurement of Rotational Positions With Magnetic Encoders. *IEEE Transactions on Instrumentation and Measurement.* 2016; 65(10):2360-2368.

(177) T. Kyriakos and L. Theodore. Comparison of improved methods for tracking movements of IPMC actuators. Comparison of improved methods for tracking movements of IPMC actuators. *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing SystemsID: 246; ID: 1.* ; 2011. p. 467-472.

(178) Arduino. *Arduino - AboutUs.* [homepage on the Internet]. Italy: Arduino; 2017 cited 2017 08/09]. Available from: https://www.arduino.cc/en/Main/AboutUs.

(179) Kernighan BW, Ritchie DM. *The C Programming Language.* 1st ed. ed. Englewood Cliffs, New Jersey, United States: Prentice Hall; 1978.

(180) Stallman R. *avr-gcc - GCC Wiki.* [homepage on the Internet]. Boston, Massachusetts, United States: Free Software Foundation; 2017 [updated 03/06/2017; cited 2017 08/24]. Available from: https://gcc.gnu.org/wiki/avr-gcc.

(181) Dean BS. *AVRDUDE - AVR Downloader/UploaDEr.* [homepage on the Internet]. Boston, Massachusetts, United States: Free Software Foundation; 2010 [updated 08/01/2010; cited 2017 08/24]. Available from: http://www.nongnu.org/avrdude/.

(182) Atmel Corporation. *ATmega48A/PA/88A/PA/168A/PA/328/P. ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KBYTES IN-SYSTEM PROGRAMMABLE FLASH.* Atmel-8271JS-AVR- ATmega-Datasheet_11/2015 ed. [monograph on the Internet]. San Jose, California, United States: Atmel Corporation; 2015 [cited 16/08/2017] Available from: http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_summary.pdf.

(183) Atmel Corporation. *ATmega8U2, ATmega16U2, ATmega32U2. 8-bit AVR Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller.* 7799E–AVR–09/2012 ed. [monograph on the Internet]. San Jose, California, United States: Atmel Corporation; 2010 [cited 16/08/2017] Available from: http://www.atmel.com/Images/doc7799.pdf.

(184) Texas Instruments. *LMV3xx Low-Voltage Rail-to-Rail Output Operational Amplifiers.* SLOS263W –AUGUST 1999–REVISED OCTOBER 2014 ed. [monograph on the Internet]. Dallas, Texas, United States: Texas Instruments; 2017 [cited 16/08/2017] Available from: http://www.ti.com/lit/ds/symlink/lmv324.pdf.

(185) ON Semiconductor. *NCP1117, NCV1117. A Low-Dropout Positive Fixed and Adjustable Voltage Regulators.* Rev. 28 ed. [monograph on the Internet]. Phoenix, Arizona, United States: ON Semiconductor; 2017 [cited 16/08/2017] Available from: http://www.onsemi.com/pub/Collateral/NCP1117-D.PDF.

(186) Arduino. *Arduino Uno Rev3 SMD - Arduino Genuino.* [homepage on the Internet]. Italy: Arduino; 2017 cited 2017 09/08]. Available from: https://store.arduino.cc/arduino-uno-rev3.

(187) M. Liang and X. Wang. The design of intelligent robot based on embedded system. The design of intelligent robot based on embedded system. *The 2011 International Conference on Advanced Mechatronic SystemsID: 257; ID: 1.* ; 2011. p. 23-28.

(188) Microchip Technology Incorporated. *START.* [homepage on the Internet]. Chandler, Arizona, United States: Microchip Technology Inc.; 2017 cited 2017 08/17]. Available from: http://start.atmel.com/.

(189) Metrohm Autolab. *Metrohm Autolab.* [homepage on the Internet]. Herisau, Switzerland: Metrohm Autolab; 2017 [updated 05/05/2017; cited 2017 08/17]. Available from: http://www.metrohm-autolab.com/.

(190) Gamry Instruments. *Electrochemical Instruments-Galvanostat/Potentiostat Manufacturer.* [homepage on the Internet]. Warminster, PA, United States: Gamry Instruments; 2017 cited 2017 08/17]. Available from: https://www.gamry.com/.

(191) eDAQ. *Home : eDAQ - Instruments for Electrochemistry Chromatography Electrophoresis Conductivity pH Biosensors.* [homepage on the Internet]. Denistone East, NSW, Australia: eDAQ; 2017 cited 2017 08/17]. Available from: https://www.edaq.com/.

(192) Ivium Technologies B.V. *Innovative Electrochemical Instrumentation from Ivium Technologies | Ivium Technologies B.V.* [homepage on the Internet]. Eindhoven, Netherlands: Ivium Technologies B.V.; 2016 [updated November 2016; cited 2017 08/17]. Available from: http://www.ivium.nl/.

(193) Rowe AA, Bonham AJ, White RJ, Zimmer MP, Yadger RJ, Hobza TM, et al. CheapStat: An Open-Source, "Do-It-Yourself" Potentiostat for Analytical and Educational Applications . *PLoS ONE.* 2011; 6(9):e23783.

(194) IO Rodeo. *CheapStat: open source potentiostat | IO Rodeo.* [homepage on the Internet]. Pasadena, California, United States: IO Rodeo; 2017 cited 2017 08/17]. Available from: https://iorodeo.com/collections/cheapstat-open-source-potentiostat/products/cheapstat-open-source-potentiostat.

(195) Microchip Technology Incorporated. *8/16-bit Atmel XMEGA Microcontroller. ATxmega128A4U / ATxmega64A4U / ATxmega32A4U / ATxmega16A4U.* Atmel-8387H-AVR-ATxmega16A4U-32A4U-64A4U-128A4U-Datasheet_09/2014 ed. [monograph on the Internet]. San Jose, California, United States: Microchip Technology Incorporated; 2014 [cited 17/08/2017] Available from: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8387-8-and16-bit-AVR-Microcontroller-XMEGA-A4U_Datasheet.pdf.

(196) Summers J. *WheeStat Potentiostat Smoky Mountain Scientific.* [homepage on the Internet]. Cullowhee, North Carolina, United States: Smoky Mt Scientific; 2017 [updated August, 2017; cited 2017 08/17]. Available from: http://www.smokymtsci.com/project

(197) Wessels R. *Energia.* [homepage on the Internet]. Dallas, Texas, United States: Energia; 2016 [updated 08/11/2016; cited 2017 08/17]. Available from: http://energia.nu/.

(198) Barragán H. *Wiring.* [homepage on the Internet]. Bogotá, Colombia: Wiring; 2017 [updated 05/06/2017; cited 2017 08/17]. Available from: http://wiring.org.co/.

(199) Texas Instruments. *TI LaunchPad development kits | Overview | TI.com.* [homepage on the Internet]. Dallas, Texas, United States: Texas Instruments; 2017 cited 2017 08/17]. Available from: http://www.ti.com/lsds/ti/tools-software/launchpads/overview/overview.page.

(200) Texas Instruments. *Tiva C Series TM4C123G LaunchPad (EK-TM4C123GXL).* [homepage on the Internet]. Dallas, Texas, United States: Texas Instruments; 2017 cited 2017 08/17]. Available from: https://www.ti.com/ww/en/launchpad/launchpads-connected-ek-tm4c123gxl.html.

(201) Fry B, Reas C. *Processing.org.* [homepage on the Internet]. Cambridge, Massachusetts, United States: Processing; 2017 cited 2017 08/17]. Available from: https://processing.org/.

(202) Meloni GN. Building a Microcontroller Based Potentiostat: A Inexpensive and Versatile Platform for Teaching Electrochemistry and Instrumentation. *Journal of Chemical Education.* 2016; 93(7):1320-1322.

(203) Cruz AFD, Norena N, Kaushik A, Bhansali S. A low-cost miniaturized potentiostat for point-of-care diagnosis. *Biosensors and Bioelectronics*. 2014; 62(0):249-254.

(204) Loncaric C, Tang Y, Ho C, Parameswaran MA, Yu H. A USB-based electrochemical biosensor prototype for point-of-care diagnosis. *Sensors and Actuators B: Chemical.* 2012; 161(1):908-913.

(205) Stackelberg MV, Pilgram M, Toome V. Bestimmung von Diffusionskoeffizienten einiger Ionen in wäßriger Lösung in Gegenwart von Fremdelektrolyten. I (Determination of diffusion coefficients of some ions in aqueous solution in the presence of foreign electrolytes. I). *Berichte der Bunsengesellschaft für physikalische Chemie (Reports of the Bunsen Society for Physical Chemistry).* 1953; 57(5):342-350.

(206) Texas Instruments. *LMP91000 Sensor AFE System: Configurable AFE Potentiostat for Low-Power Chemical Sensing Applications.* [homepage on the Internet]. Dallas, Texas, United States:

Texas Instruments; 2014 [updated December; cited 2017 08/18]. Available from: http://www.ti.com/lit/ds/symlink/lmp91000.pdf.

(207) Beagleboard. *Beagleboard.org - black.* [homepage on the Internet]. Dallas, Texas, United States: Beagleboard; 2017 [updated Wed Mar 22 2017; cited 2017 08/18]. Available from: https://beagleboard.org/black.

(208) Texas Instruments. *AM3358 Sitara Processor: ARM Cortex-A8, 3D Graphics, PRU-ICSS | TI.com.* [homepage on the Internet]. Dallas, Texas, United States: Texas Instruments; 2017 [updated January; cited 2017 08/18]. Available from: https://www.ti.com/product/am3358.

(209) Arduino. *Arduino - Arduino Board Duemilanove.* [homepage on the Internet]. Ivrea, Italy: Arduino; 2009 cited 2017 08/18]. Available from: https://www.arduino.cc/en/Main/ArduinoBoardDuemilanove.

(210) Southampton Electrochemistry Group. *Instrumental Methods in Electrochemistry.* Chichester, UK: Ellis Horwood; 1985.

(211) Burr Brown. *INA114. Precision INSTRUMENTATION AMPLIFIER.* PDS-1142D ed. [monograph on the Internet]. Dallas, Texas, United States: Texas Instruments; 1998 [cited 18/08/2017] Available from: http://www.ti.com/lit/ds/symlink/ina114.pdf.

(212) H. A. Wurdemann, S. Sareh, A. Shafti, Y. Noh, A. Faragasso, D. S. Chathuranga, et al. Embedded electro-conductive yarn for shape sensing of soft robotic manipulators. Embedded electro-conductive yarn for shape sensing of soft robotic manipulators. *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* 2015. p. 8026-8029.

(213) Wei Y, Wu Y, Tudor J. A real-time wearable emotion detection headband based on EEG measurement. *Sensors and Actuators A: Physical*. 2017; 263(0):614-621.

(214) Borghetti D, Bruni A, Fabbrini M, Murri L, Sartucci F. A low-cost interface for control of computer functions by means of eye movements. *Computers in Biology and Medicine* 2007; 37(12):1765-1770.

(215) Analog Devices. *AD820. Single-Supply, Rail-to-Rail, Low Power, FET Input Op Amp.* Rev. H ed. [monograph on the Internet]. Norwood, Massachusetts, United States: Analog Devices; 2011 [cited 18/08/2017] Available from: http://www.analog.com/media/en/technical-documentation/data-sheets/AD820.pdf.

(216) Altium. *Free PCB Design Software | CircuitMaker.* [homepage on the Internet]. San Diego, California, United States: Altium; 2017 cited 2017 08/24]. Available from: https://circuitmaker.com/.

(217) Wilson P. Chapter 22 - Finite State Machines in VHDL and Verilog. In: Wilson P, editor. *Design Recipes for FPGAs (Second Edition).* Oxford: Newnes; 2016. p. 305-309.

(218) Coto Technology. *8L SERIES/SPARTAN DIP REED RELAYS.* 06112015 [monograph on the Internet]. North Kingstown, Rhode Island, United States: Coto Technology; 2015 [cited 24/08/2017] Available from: http://cotorelay.com/wp-content/uploads/2014/09/8L_series_reed_relay_datasheet1.pdf.

(219) Maxim Integrated. *MAX743, Dual-Output, Switch-Mode Regulator (+5V to ± 15V or ±12V).* 19-2873; Rev 0; 7/90 ed. [monograph on the Internet]. SunnyVale, California, USA: Maxim Integrated; 1990 [cited 16/3/19] Available from: https://datasheets.maximintegrated.com/en/ds/MAX743.pdf.

(220) Maxim Integrated. *MAX532, Dual, Serial-Input, Voltage-Output, 12-Bit MDAC.* 19-0046; Rev. 1; 3/94 ed. [monograph on the Internet]. Sunnyvale, California, USA: Maxim Integrated; 1994 [cited 16/3/19] Available from: https://datasheets.maximintegrated.com/en/ds/MAX532.pdf.

(221) Linear Technology. *LTC1250, Very Low Noise Zero-Drift Bridge Amplifier .* LT/GP 0205 1K REV B ed. [monograph on the Internet]. Milpitas, California, USA: Analog Devices; 1994 [cited

16/3/19] Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/1250fb.pdf.

(222) Future Technology Devices International. *UM232R USB - Serial UART Development Module Datasheet.* Version 1.3 ed. [monograph on the Internet]. Glasgow, UK: Future Technology Devices International; 2018 [cited 16/3/19] Available from: https://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UM232R.pdf.

(223) Maxim Integrated. *MAX4617/MAX4618/MAX4619, High-Speed, Low-Voltage, CMOS Analog Multiplexers/Switches.* 19-1502; Rev 3; 12/12 ed. [monograph on the Internet]. San Jose, California, USA: Maxim Integrated; 2012 [cited 16/3/19] Available from: https://datasheets.maximintegrated.com/en/ds/MAX4617-MAX4619.pdf.

(224) Analog Devices. *AD620, Low Cost Low Power Instrumentation Amplifier.* Rev. H ed. [monograph on the Internet]. Norwood, Massachusetts, USA: Analog Devices; 2011 [cited 16/3/19] Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/AD620.pdf.

(225) Analog Devices. *AD826, High-Speed, Low-Power Dual Operational Amplifier.* Rev. C ed. [monograph on the Internet]. Norwood, Massachusetts, USA: Analog Devices; 2010 [cited 16/3/19] Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/AD826.pdf.

(226) Olimex Ltd. *AVR-ISP-MK2 programmer, USER'S MANUAL.* Revision M ed. [monograph on the Internet]. Plovdiv, Bulgaria: Olimex Ltd; 2016 [cited 16/3/19] Available from: https://www.olimex.com/Products/AVR/Programmers/AVR-ISP-MK2/resources/AVR-ISP-MK2.pdf.

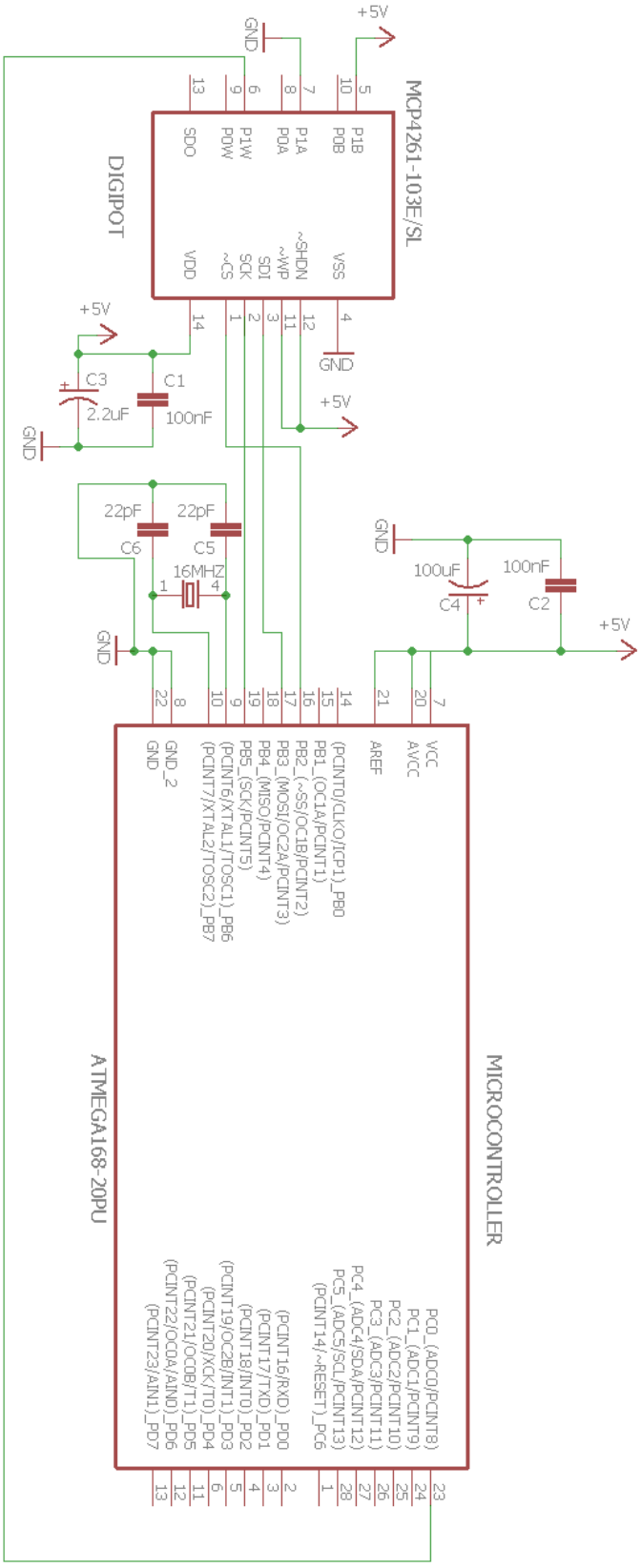
(227) Savitzky A, Golay MJE. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry.* 1964; 36(8):1627-1639.

(228) Maxim Integrated. *MAX7401 8th-Order, Lowpass, Bessel, Switched-Capacitor Filters.* 19-4788; Rev 1; 6/99 ed. [monograph on the Internet]. Sunnyvale, California, USA: Maxim Integrated; 1999 [cited 17/3/19] Available from: https://datasheets.maximintegrated.com/en/ds/MAX7401-MAX7405.pdf.

(229) Analog Devices. *ADAQ7980/ADAQ7988 16-Bit, 1 MSPS, µModule Data Acquisition System.* Rev. A ed. [monograph on the Internet]. Norwood, Massachusetts, USA: Analog Devices; 2017 [cited 17/3/19] Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/adaq7980-7988.pdf.

(230) Andrade AFB, Mamo SK, Gonzalez-Rodriguez J. Rapid Screening Method for New Psychoactive Substances of Forensic Interest: Electrochemistry and Analytical Determination of Phenethylamines Derivatives (NBOMe) via Cyclic and Differential Pulse Voltammetry. Analytical Chemistry 2017; 89(3):1445-1452 https://doi.org/10.1021/acs.analchem.6b02426

Appendix 1: Schematic and Firmware for Testing the MCP4261 for use as a DAC

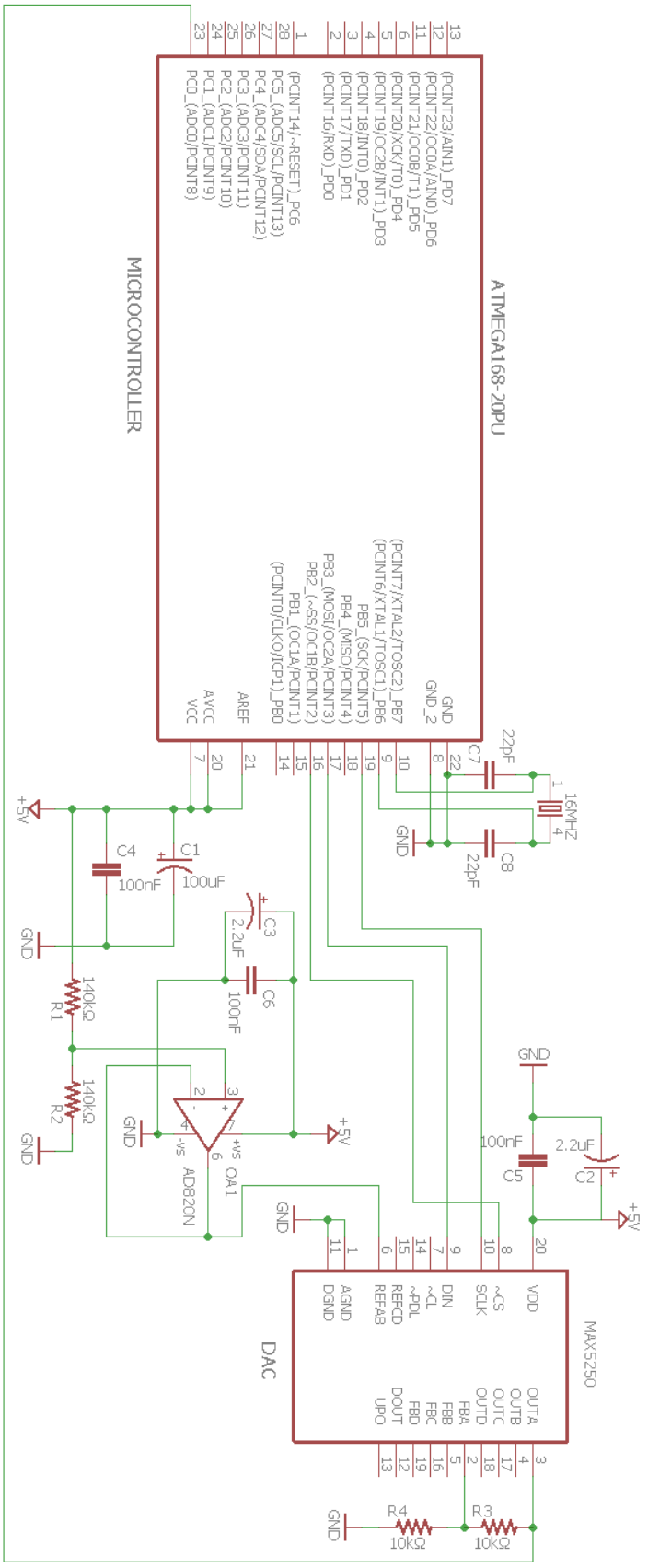


```
// Import Arduino SPI bus library
#include <SPI.h>
// Arduino ADC pin
const int readPin = A0;
// Address mask 0b00010000 for wiper 1 of MCP4261 to update new level
const int address = 16;
// SPI slave select pin for MCP4261
const int digipotSelect = 10;

byte level;
long timer;

void setup ()
{
	// Set port directions
	pinMode (readPin, INPUT);
	pinMode (digipotSelect, OUTPUT);
	// Disable the digipot
	digitalWrite (digipotSelect, HIGH);
	// Set BAUD rate for serial and SPI settings
	Serial.begin (9600);
	SPI.setBitOrder (MSBFIRST);
	SPI.begin ();
}
void loop ()
{
	timer = millis ();
	// set 8-bit digipot to start at GND then sweep up to 5 V at the wiper
	for (level = 255; level > 0; level--)
	{
		digitalWrite (digipotSelect, LOW);
		SPI.transfer (address);
		SPI.transfer (level);
		digitalWrite (digipotSelect, HIGH);
		// Use Arduino's serial monitor to plot values
		Serial.println (analogRead (readPin));
	}
	// sweep back down from 5 V to GND
	for ( ; level < 256; level++)
	{
		digitalWrite (digipotSelect, LOW);
		SPI.transfer (address);
		SPI.transfer (level);
		digitalWrite (digipotSelect, HIGH);
		Serial.println (analogRead (readPin));
	}
	// Print the time for full scan in ms
	Serial.print (millis() - timer);
}
```

Appendix 2: Schematic and Firmware for Testing the MAX5250 for use as a DAC



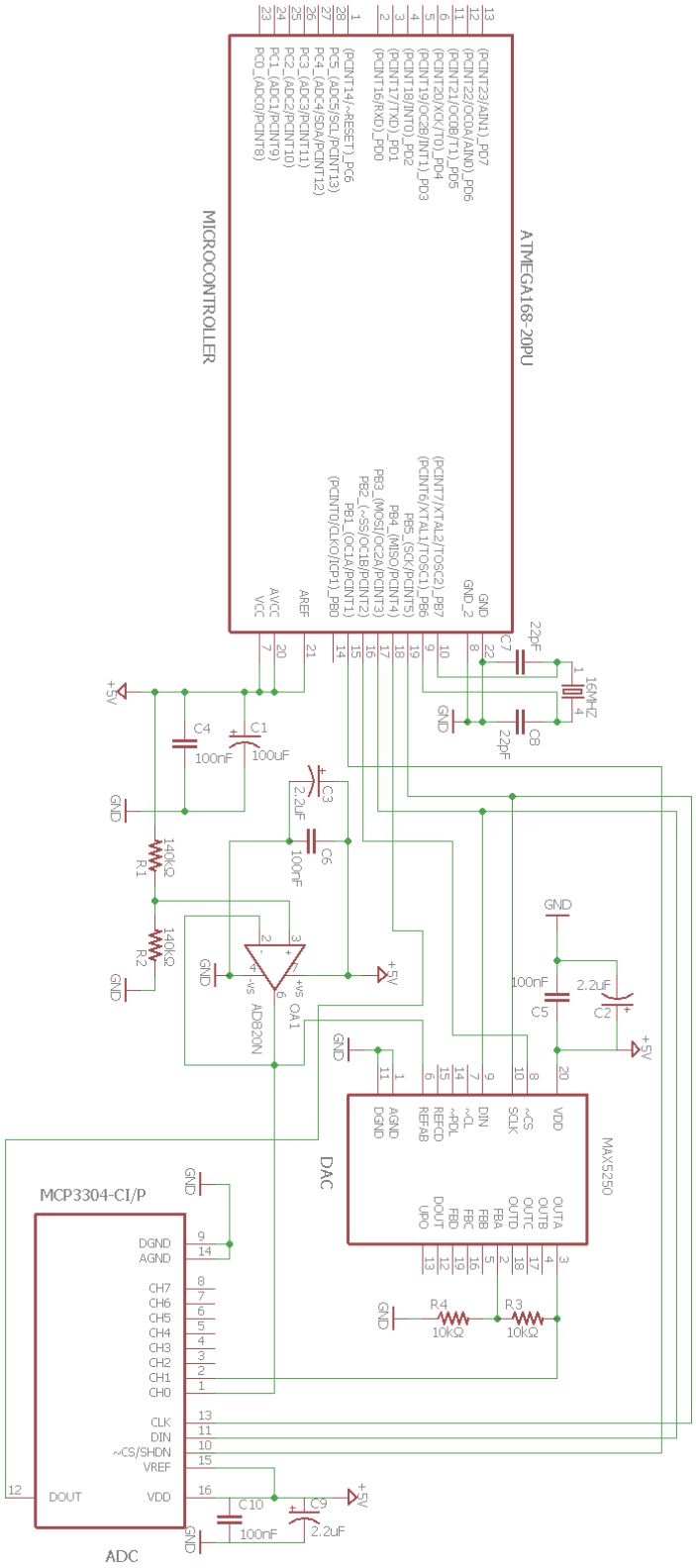
```
#include <SPI.h>

const int readPin = A0;
// MAX5250 has 4 channels, these values correspond to min and max setting
// for channel A
const int DACmin = 12288;
const int DACmax = 16380;
unsigned int DACvalue;
const int DACselect = 10;
byte dac1, dac2;
long timer;

void setup()
{
	pinMode (readPin, INPUT);
	pinMode (DACselect, OUTPUT);
	digitalWrite (DACselect, HIGH);
	Serial.begin (9600);
	SPI.setBitOrder (MSBFIRST);
	SPI.begin ();
}

void loop()
{
	timer = millis ();
	// sweep DAC up in increments of 4 necessary due to SPI commands
	for (DACvalue = DACmin; DACvalue <= DACmax; DACvalue += 4)
	{
		dac1 = highByte (DACvalue);
		dac2 = lowByte (DACvalue);
		digitalWrite (DACselect, LOW);
		SPI.transfer (dac1);
		SPI.transfer (dac2);
		digitalWrite (DACselect, HIGH);
		Serial.println (analogRead (readPin));
	}
	for ( ; DACvalue >= DACmin; DACvalue -= 4)
	{
		dac1 = highByte (DACvalue);
		dac2 = lowByte (DACvalue);
		digitalWrite (DACselect, LOW);
		SPI.transfer (dac1);
		SPI.transfer (dac2);
		digitalWrite (DACselect, HIGH);
		Serial.println (analogRead (readPin));
	}
	Serial.print (millis() - timer);
}
```

Appendix 3: Schematic and Firmware for Testing the MCP3304 for use as an ADC



```
#include <SPI.h>

const int DACmin = 12288;
const int DACmax = 16380;
const int DACselect = 10;
const int ADCselect = 9;
const byte ADCconfig1 = 0b00001000;
const byte ADCconfig2 = 0b10000000;
int measurements;
unsigned int DACvalue;
byte dac1, dac2, adc1, adc2;
int ADCvalue;
long timer, ADCtally;

void setup()
{
	pinMode (DACselect, OUTPUT);
	pinMode (ADCselect, OUTPUT);
	digitalWrite (DACselect, HIGH);
	digitalWrite (ADCselect, HIGH);
	Serial.begin (2000000);
	SPI.setBitOrder (MSBFIRST);
	SPI.begin ();
}

void loop()
{
	timer = millis ();
	measurements = 1;
	for (DACvalue = DACmin; DACvalue <= DACmax; DACvalue += 4)
	{
		applyDAC (DACvalue);
		Serial.println (readADC (measurements));
	}
	for ( ; DACvalue >= DACmin; DACvalue -= 4)
	{
		applyDAC (DACvalue);
		Serial.println (readADC (measurements));
	}
	Serial.print (millis() - timer);
	timer = millis ();
	measurements = 100;
	for (DACvalue = DACmin; DACvalue <= DACmax; DACvalue += 4)
	{
		applyDAC (DACvalue);
		Serial.println (readADC (measurements));
	}
	for ( ; DACvalue >= DACmin; DACvalue -= 4)
	{
		applyDAC (DACvalue);
		Serial.println (readADC (measurements));
	}
```

```
  Serial.print (millis() - timer);
}

void applyDAC (unsigned int DACvalue)
{
        dac1 = highByte (DACvalue);
        dac2 = lowByte (DACvalue);
        digitalWrite (DACselect, LOW);
        SPI.transfer (dac1);
        SPI.transfer (dac2);
        digitalWrite (DACselect, HIGH);
}

int readADC (int measurements)
{
        ADCtally = 0;
        for (int i = 0; i < measurements; i++)
        {
                digitalWrite (ADCselect, LOW);
                SPI.transfer (ADCconfig1);
                adc1 = SPI.transfer (ADCconfig2);
                adc2 = SPI.transfer (0);
                digitalWrite (ADCselect, HIGH);

                adc1 &= 0b00011111;
                ADCvalue = adc1 << 8;
                ADCvalue += adc2;
                ADCvalue = ADCvalue << 3;
                ADCvalue = ADCvalue / 8;
                ADCtally += ADCvalue;
        }
        return (ADCtally / measurements);
}
```

Appendix 4: “main.h” Header File for Firmware for Prototype Model B

```
// “main.h” Header File for Firmware for Prototype Model B page 1 of 3

#ifndef MAIN_H_
#define MAIN_H_

#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <avr/interrupt.h>
#include <util/atomic.h>

#define PWR_LED_ON                PORTD |= (1<<PORTD2)
#define PWR_LED_OFF               PORTD &= ~(1<<PORTD2)
#define SCAN_LED_ON               PORTD |= (1<<PORTD3)
#define SCAN_LED_OFF        PORTD &= ~(1<<PORTD3)
#define CE_ON                     PORTD |= (1<<PORTD4)
#define CE_OFF                           PORTD &= ~(1<<PORTD4)
#define SET_COMPLIANCE_HI   PORTD |= (1<<PORTD5)
#define SET_COMPLIANCE_LO   PORTD &= ~(1<<PORTD5)
#define CELL_ON                          PORTD |= (1<<PORTD6)
#define CELL_OFF                  PORTD &= ~(1<<PORTD6)


#define POT_SELECT                PORTB &= ~(1<<PORTB0)
#define POT_DESELECT        PORTB |= (1<<PORTB0)
#define ADC_SELECT                PORTB &= ~(1<<PORTB1)
#define ADC_DESELECT        PORTB |= (1<<PORTB1)
#define DAC_SELECT                PORTB &= ~(1<<PORTB2)
#define DAC_DESELECT        PORTB |= (1<<PORTB2)

#define SPI_WAIT                  while(!(SPSR & (1<<SPIF)))

#define POT_SS                           PORTB0 // Slave selects
#define ADC_SS                           PORTB1
#define DAC_SS                           PORTB2
#define MOSI                      PORTB3
#define MISO                      PORTB4
#define SPI_CLK                          PORTB5
#define PWR_LED                          PORTD2
#define SCAN_LED                  PORTD3
#define CE_RELAY                  PORTD4
#define COMPLIANCE_RELAY    PORTD5
#define CELL_RELAY                PORTD6

#define FOSC 16000000UL // Clock Speed
#define BAUD 1000000UL
#define MYUBRR FOSC/16/BAUD-1
#define CTC_MATCH_OVERFLOW ((F_CPU / 1000) / 8)

#define DAC_MIN                          61440
#define DAC_MID                          63500
#define DAC_MAX                          65500
#define DAC_START                 63500
```

```
// “main.h” Header File for Firmware for Prototype Model B page 2 of 3

#define POT_START                 10
// “main.h” Header File for Firmware for Prototype Model B page

#define SHORT_WAIT                10
#define LONG_WAIT                 100
#define PRE_WAIT                  10

/*
       ADC CHANNELS: CH0    WE_RAW
                     CH1    REF_SIG
                     CH2    WE_SIG
                     CH3    +2V5
                     CH4    NC
                     CH5    NC
                     CH6    NC
                     CH7    NC

*/
#define ADC_VOLTAGE_CONFIG1       0b00001000
#define ADC_VOLTAGE_CONFIG2       0b00000000
#define ADC_VOLTAGE_CONFIG3       0b00000000
#define ADC_CURRENT_CONFIG1       0b00001001
#define ADC_CURRENT_CONFIG2       0b00000000
#define ADC_CURRENT_CONFIG3       0b00000000
#define ADC_CURRENT_CONFIG4       0b00001010
#define ADC_CURRENT_CONFIG5       0b00000000
#define ADC_CURRENT_CONFIG6       0b00000000
#define POT_CONFIG                0b00000000

int INPUT_RECEIVED, ADC_READING, VOLTAGE, CURRENT, START_I, STOP_I, CYCLES_I,
RATE_I, POT_I, COMPLIANCE;
int VOLTAGE_MEASUREMENT_NUMBER, CURRENT_MEASUREMENT_NUMBER, METHOD, CYCLES, WAIT,
STEP, FIRST_PASS;
int CURRENT_CATHODIC, CURRENT_ANODIC, OLD_VOLTAGE, OLD_CURRENT;
int DEPOSITION_V, DEPOSITION_T, EQUILIBRIUM_T, PULSE, INITIAL_V;
int SCAN_NUMBER;
int FEEDBACK, HOLD_GAIN;
unsigned int DAC_LEVEL, MEASUREMENTS;
long int VOLTAGE_ADDER, CURRENT_ADDER;
volatile unsigned long ACTUAL_MILLIS;
unsigned long MILLIS_START, RUN_START, MEASURE_START;
char VOLTAGE_BUFFER[20], CURRENT_BUFFER[20], INPUT_COUNTER, INPUT_STRING[128],
INPUT_CHAR;
unsigned char DAC_LO, DAC_HI, ADC_HI, ADC_LO, POT_LEVEL, junk1, junk2;
int az, ARRAY_COUNTER;
/*
Arrays no longer used
int OLD_VOLTAGE_ARRAY[16];
int OLD_CURRENT_ARRAY[512];
*/
int SPARE_VOLTAGE_MEASUREMENT_NUMBER, SPARE_CURRENT_MEASUREMENT_NUMBER,
VOLTAGE_SHIFTER, CURRENT_SHIFTER;
int ALPHA, ALPHA_SHIFT, POINT_NUMBER;
int NEW_VOLTAGE, NEW_CURRENT;
float ALPHA_CALC_FLOAT;

int main (void);
void SPI_INITIALISE (void);
void BOARD_INITIALISE (void);
void SET_DAC (void);
```

```
// “main.h” Header File for Firmware for Prototype Model B page 3 of 3

int READ_VOLTAGE (void);
unsigned char SPI_CHAR (unsigned char CONFIG_SETTING);
void UART_INITIALISE (unsigned int ubrr);
void UART_TX_STRING (char* StringPtr);
void UART_TX_CHAR (unsigned char data);
unsigned char UART_RX_CHAR (void);
unsigned char * UART_RX_STRING (void);
void SET_POT (void);
int READ_CURRENT (void);
void CV ();
unsigned int GET_INITIAL (int INITIAL_V);
int MEASURE_VOLTAGE ();
int MEASURE_CURRENT ();
void TEST_DAC_RANGE (void);
void TEST_REF_RANGE (void);
void CV_DOWN_FIRST ();
void CV_UP_FIRST ();
void TRANSMIT ();
void LSV ();
void LSV_DOWN ();
void LSV_UP ();
void PRINT_TESTS ();
int usart_putchar_printf(char var, FILE *stream);
void LSV_TEST ();
void LSV_DOWN_TEST ();
void LSV_UP_TEST ();
void speedTest (void);
void TRANSMIT_UNFILTERED (void);
void TEST_WE_RANGE (void);
void SWV (void);
int READ_PULSE_CURRENT (void);
void SWV_DOWN (void);
void SWV_UP (void);
void SET_FEEDBACK (int FEEDBACK);
void VOLTAGE_ALPHA_TEST (void);
void VOLTAGE_ALPHA_UP (void);
void VOLTAGE_ALPHA_DOWN (void);
void CURRENT_ALPHA_TEST (void);
void CURRENT_ALPHA_UP (void);
void CURRENT_ALPHA_DOWN (void);

// Necessary for printf
static FILE mystdout = FDEV_SETUP_STREAM(usart_putchar_printf, NULL,
_FDEV_SETUP_WRITE);

#endif /* MAIN_H_ */
```

Appendix 5: “main.c” Start File for Firmware for Prototype Model B

```
// “main.c” Start File for Firmware for Prototype Model B page 1 of 2

#include <avr/io.h>
#include "main.h"

ISR (TIMER1_COMPA_vect)
{
	ACTUAL_MILLIS++;
}

int main (void)
{
	SPI_INITIALISE ();
	UART_INITIALISE (MYUBRR);
	BOARD_INITIALISE ();
	INPUT_RECEIVED = 0;
	FIRST_PASS = 1;
	TCCR1B |= (1<<CS11)|(1<<WGM12);
	OCR1AH = (CTC_MATCH_OVERFLOW >> 8);
	OCR1AL = CTC_MATCH_OVERFLOW;
	TIMSK1 |= (1<<OCIE1A);
	sei();
	stdout = &mystdout;
    while (1)
    {
		UART_RX_STRING ();
		if (INPUT_RECEIVED)
		{

			ACTUAL_MILLIS = 0;
			if (COMPLIANCE == 2)
			{
				SET_COMPLIANCE_HI;
			}
			else
			{
				SET_COMPLIANCE_LO;
			}
			switch (METHOD)
			{
				case 1:
					CV ();
					break;
				case 2:
					LSV ();
					break;
				case 3:
					SWV ();
					break;
				case 4:
					// CHRONOAMPEROMETRY (not implemented yet)
					break;
				case 5:
					VOLTAGE_ALPHA_TEST();
					break;
				case 6:
					CURRENT_ALPHA_TEST();
// “main.c” Start File for Firmware for Prototype Model B page 2 of 2
```

```
                                        break;
                                case 9:
                                        // PORT CHECK (not implemented)
                                        break;
                                default:
                                        INPUT_RECEIVED = 0;
                                        UART_TX_STRING("333333");
                                        break;
                        }
                }
        }
}

void BOARD_INITIALISE (void)
{
        DDRD |= (1<<PWR_LED)|(1<<SCAN_LED)|
        (1<<CE_RELAY)|(1<<COMPLIANCE_RELAY)|(1<<CELL_RELAY);
        DDRC |= (1<<PORTC0)|(1<<PORTC1)|(1<<PORTC2)|(1<<PORTC3);
        PORTC &= ~(1<<PORTC3);
        FEEDBACK = 4;
        SET_FEEDBACK(FEEDBACK);
        PWR_LED_ON;
        SCAN_LED_OFF;
        SET_COMPLIANCE_LO;
        DAC_LEVEL = 64500;
        SET_DAC();
        CE_OFF;
        CELL_OFF;
}
```

Appendix 6: “UART.c” Communication File for Firmware for Prototype Model B

```
// “UART.c” Comm File for Firmware for Prototype Model B page 1 of 5

#include "main.h"

void UART_INITIALISE(unsigned int ubrr)
{
	/*Set baud rate */
	UBRR0H = (unsigned char)(ubrr>>8);
	UBRR0L = (unsigned char)ubrr;
	/*Enable receiver and transmitter */
	UCSR0B = (1<<RXEN0)|(1<<TXEN0);
	/* Set frame format: 8data, 2stop bit */
	UCSR0C = (1<<USBS0)|(3<<UCSZ00);
}

void UART_TX_STRING (char* StringPtr)
{
	while(*StringPtr != 0x00)
	{
		UART_TX_CHAR (*StringPtr);
		StringPtr++;
	}
}

void UART_TX_CHAR (unsigned char data)
{
	UDR0 = data;
	while(!(UCSR0A&(1<<UDRE0)));
}

unsigned char UART_RX_CHAR (void)
{
	while(!(UCSR0A & (1<<RXC0)));
	return(UDR0);
}
unsigned char * UART_RX_STRING (void)
{
	INPUT_COUNTER = 0;
	unsigned char dummy;
	while (UCSR0A & (1<<RXC0))
	{
		dummy = UDR0;
	}
	//receive the characters until exclamation mark (ASCII for "!" = 33)
	while ((INPUT_CHAR = UART_RX_CHAR ()) != 33)
	{
		INPUT_STRING[INPUT_COUNTER++] = INPUT_CHAR;
	}
	INPUT_RECEIVED = 1;
	//insert NULL to terminate the string
	INPUT_STRING[INPUT_COUNTER] = '\0';
	//length = strlen(string);
	METHOD = 0;
	COMPLIANCE = 0;
	POT_I = 0;
	START_I = 0;
	STOP_I = 0;
// “UART.c” Comm File for Firmware for Prototype Model B page 2 of 5
```

```
        CYCLES_I = 0;
        RATE_I = 0;
        WAIT = 0;
        STEP = 0;
        DEPOSITION_V = 0;
        DEPOSITION_T = 0;
        EQUILIBRIUM_T = 0;
        PULSE = 0;
        HOLD_GAIN = 0;
        ALPHA = 0;
        ALPHA_SHIFT = 0;
        VOLTAGE_SHIFTER = 0;
        CURRENT_SHIFTER = 0;
        VOLTAGE_MEASUREMENT_NUMBER = 0;
        CURRENT_MEASUREMENT_NUMBER = 0;
        METHOD = INPUT_STRING[0] - '0';
        COMPLIANCE = INPUT_STRING[1] - '0';
        for (INPUT_COUNTER=2; INPUT_COUNTER<6; INPUT_COUNTER++)
        {
               POT_I = POT_I * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=6; INPUT_COUNTER<10; INPUT_COUNTER++)
        {
               START_I = START_I * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=10; INPUT_COUNTER<14; INPUT_COUNTER++)
        {
               STOP_I = STOP_I * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=14; INPUT_COUNTER<17; INPUT_COUNTER++)
        {
               CYCLES_I = CYCLES_I * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=17; INPUT_COUNTER<21; INPUT_COUNTER++)
        {
               WAIT = WAIT * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=21; INPUT_COUNTER<25; INPUT_COUNTER++)
        {
               VOLTAGE_MEASUREMENT_NUMBER = VOLTAGE_MEASUREMENT_NUMBER * 10 +
               (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=25; INPUT_COUNTER<29; INPUT_COUNTER++)
        {
               CURRENT_MEASUREMENT_NUMBER = CURRENT_MEASUREMENT_NUMBER * 10 +
               (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=29; INPUT_COUNTER<33; INPUT_COUNTER++)
        {
               STEP = STEP * 10 + (INPUT_STRING[INPUT_COUNTER] - '0');
        }
        for (INPUT_COUNTER=33; INPUT_COUNTER<37; INPUT_COUNTER++)
        {
               DEPOSITION_V = DEPOSITION_V * 10 + (INPUT_STRING[INPUT_COUNTER] -
        '0');
        }
        for (INPUT_COUNTER=37; INPUT_COUNTER<41; INPUT_COUNTER++)
        {
               DEPOSITION_T = DEPOSITION_T * 10 + (INPUT_STRING[INPUT_COUNTER] -
'0');
        }
```

```
// “UART.c” Comm File for Firmware for Prototype Model B page 3 of 5

	for (INPUT_COUNTER=41; INPUT_COUNTER<45; INPUT_COUNTER++)
	{
		EQUILIBRIUM_T = EQUILIBRIUM_T * 10 + (INPUT_STRING[INPUT_COUNTER] -
		'0');
	}
	for(INPUT_COUNTER=45; INPUT_COUNTER<49; INPUT_COUNTER++){
		PULSE = PULSE * 10 + ( INPUT_STRING[INPUT_COUNTER] - '0' );
	}
	for(INPUT_COUNTER=49; INPUT_COUNTER<50; INPUT_COUNTER++){
		HOLD_GAIN = HOLD_GAIN * 10 + ( INPUT_STRING[INPUT_COUNTER] - '0' );
	}
	for(INPUT_COUNTER=50; INPUT_COUNTER<53; INPUT_COUNTER++){
		ALPHA = ALPHA * 10 + ( INPUT_STRING[INPUT_COUNTER] - '0' );
	}
	for(INPUT_COUNTER=53; INPUT_COUNTER<54; INPUT_COUNTER++){
		ALPHA_SHIFT = ALPHA_SHIFT * 10 + ( INPUT_STRING[INPUT_COUNTER] - '0'
);
	}
	POT_I -= 1000;
	FEEDBACK = POT_I;
	SET_FEEDBACK(FEEDBACK);
	START_I -= 4096;
	STOP_I -= 4096;
	CYCLES_I -= 100;
	WAIT -= 1000;
	PULSE -= 1000;
	VOLTAGE_MEASUREMENT_NUMBER -= 1000;
	CURRENT_MEASUREMENT_NUMBER -= 1000;
	/*
	SPARE_CURRENT_MEASUREMENT_NUMBER = CURRENT_MEASUREMENT_NUMBER - 256;
	SPARE_VOLTAGE_MEASUREMENT_NUMBER = VOLTAGE_MEASUREMENT_NUMBER - 64;
	*/
	STEP -= 1000;
	DEPOSITION_V -= 4096;
	DEPOSITION_T -= 1000;
	EQUILIBRIUM_T -= 1000;
	DEPOSITION_T *= 1000;
	EQUILIBRIUM_T *= 1000;
	ALPHA -= 100;
	// This code doesn't work brings a zero for ALPHA_SHIFT so using switch/case
	// below instead
	//ALPHA_CALC_FLOAT = log((ALPHA + 1)) / log (2); This code doesn't work
brings a zero for ALPHA_SHIFT
	//ALPHA_SHIFT = (int) ALPHA_CALC;  so using switch/case below instead
//	switch (ALPHA)
//	{
//	case 3:
//		ALPHA_SHIFT = 2;
//		break;
//	case 7:
//		ALPHA_SHIFT = 3;
//		break;
//	case 15:
//		ALPHA_SHIFT = 4;
//		break;
//	case 31:
//		ALPHA_SHIFT = 5;
//		break;
//	case 63:
```

```
//            ALPHA_SHIFT = 6;
//            break;
// “UART.c” Comm File for Firmware for Prototype Model B page 4 of 5

//     case 127:
//            ALPHA_SHIFT = 7;
//            break;
//     default:
//            ALPHA_SHIFT = 3;
//            break;
//     }
       switch (VOLTAGE_MEASUREMENT_NUMBER)
       {
       case 16:
              VOLTAGE_SHIFTER = 3;
              break;
       case 32:
              VOLTAGE_SHIFTER = 4;
              break;
       case 64:
              VOLTAGE_SHIFTER = 5;
              break;
       case 128:
              VOLTAGE_SHIFTER = 6;
              break;
       default:
              VOLTAGE_SHIFTER = 3;
              break;
       }
       switch (CURRENT_MEASUREMENT_NUMBER)
       {
       case 16:
              CURRENT_SHIFTER = 3;
              break;
       case 32:
              CURRENT_SHIFTER = 4;
              break;
       case 64:
              CURRENT_SHIFTER = 5;
              break;
       case 128:
              CURRENT_SHIFTER = 6;
              break;
       case 256:
              CURRENT_SHIFTER = 7;
              break;
       case 512:
              CURRENT_SHIFTER = 8;
              break;
       case 1024:
              CURRENT_SHIFTER = 9;
              break;
       default:
              CURRENT_SHIFTER = 3;
              break;
       }
       /* Old method for printing out debugging numbers – still works
       printf("3216,%d,%d,\n", ALPHA, ALPHA_SHIFT);
       printf("3216,%d,%d,\n", VOLTAGE_MEASUREMENT_NUMBER,
       CURRENT_MEASUREMENT_NUMBER);
       printf("3216,%d,%d,\n", VOLTAGE_SHIFTER, CURRENT_SHIFTER);
        */
```

```
// “UART.c” Comm File for Firmware for Prototype Model B page 5 of 5

	if (START_I > STOP_I)
	{
		START_I += 50;
	}
	else
	{
		START_I -= 50;
	}
	while (UCSR0A & (1<<RXC0))
	{
		dummy = UDR0;
	}
	//return the received an unsigned char*
	unsigned char *returner = NULL;
	return (returner);
}
void TRANSMIT (void)
{
	printf ("%d,%d,\n", VOLTAGE, CURRENT);
	/*
	itoa (VOLTAGE, VOLTAGE_BUFFER, 10);
	UART_TX_STRING (VOLTAGE_BUFFER);
	UART_TX_STRING(",");
	itoa (CURRENT, CURRENT_BUFFER, 10);
	UART_TX_STRING (CURRENT_BUFFER);
	UART_TX_STRING(",\n");
	*/
}

void TRANSMIT_UNFILTERED (void)
{
	itoa (VOLTAGE, VOLTAGE_BUFFER, 10);
	UART_TX_STRING (VOLTAGE_BUFFER);
	UART_TX_STRING(",");
	itoa (CURRENT, CURRENT_BUFFER, 10);
	UART_TX_STRING (CURRENT_BUFFER);
	UART_TX_STRING(",\n");
}

//this function is called by printf as a stream handler
int usart_putchar_printf(char var, FILE *stream)
{
	// translate \n to \r for br@y++ terminal
	if (var == '\n') UART_TX_CHAR('\r');
	UART_TX_CHAR(var);
	return 0;
}
```

Appendix 7: “SPI.c” Control File for Firmware for Prototype Model B

```
// “SPI.c” Control File for Firmware for Prototype Model B page 1 of 2

#include "main.h"

void SPI_INITIALISE (void)
{
	// Set MOSI, SCK, SS's as Output
	DDRB |= (1<<POT_SS)|(1<<ADC_SS)|(1<<DAC_SS)|(1<<MOSI)|(1<<SPI_CLK);
	DDRB &= ~(1<<MISO);
	// Set SS's high (deselected)
	POT_DESELECT;
	ADC_DESELECT;
	DAC_DESELECT;
	// Enable SPI, Set as Master
	//Prescaler: Fosc/16, Enable Interrupts
	SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

unsigned char SPI_CHAR (unsigned char CONFIG_SETTING)
{
	SPDR = CONFIG_SETTING;
	SPI_WAIT;
	return(SPDR);
}

void SET_DAC (void)
{
	DAC_HI = (DAC_LEVEL >> 8);
	DAC_LO = ((DAC_LEVEL<< 8) >> 8);
	DAC_SELECT;
	SPI_CHAR (DAC_HI);
	SPI_CHAR (DAC_LO);
	DAC_DESELECT;
}

int READ_VOLTAGE (void)
{
	ADC_SELECT;
	SPI_CHAR (ADC_VOLTAGE_CONFIG1);
	ADC_HI = SPI_CHAR (ADC_VOLTAGE_CONFIG2);
	ADC_LO = SPI_CHAR (ADC_VOLTAGE_CONFIG3);
	ADC_DESELECT;
	ADC_READING = ADC_HI << 8;
	ADC_READING += ADC_LO;
	ADC_READING = ADC_READING << 3;
	ADC_READING = ADC_READING / 8;
	return (ADC_READING);
}
```

```
// “SPI.c” Control File for Firmware for Prototype Model B page 2 of 2

int READ_CURRENT (void)
{
        ADC_SELECT;
        SPI_CHAR (ADC_CURRENT_CONFIG1);
        ADC_HI = SPI_CHAR (ADC_CURRENT_CONFIG2);
        ADC_LO = SPI_CHAR (ADC_CURRENT_CONFIG3);
        ADC_DESELECT;
        ADC_READING = ADC_HI << 8;
        ADC_READING += ADC_LO;
        ADC_READING = ADC_READING << 3;
        ADC_READING = ADC_READING / 8;;
        return (ADC_READING);
}

int READ_PULSE_CURRENT (void)
{
        ADC_SELECT;
        SPI_CHAR (ADC_CURRENT_CONFIG4);
        ADC_HI = SPI_CHAR (ADC_CURRENT_CONFIG5);
        ADC_LO = SPI_CHAR (ADC_CURRENT_CONFIG6);
        ADC_DESELECT;
        ADC_READING = ADC_HI << 8;
        ADC_READING += ADC_LO;
        ADC_READING = ADC_READING << 3;
        ADC_READING = ADC_READING / 8;
        return (ADC_READING);
}

void SET_POT (void)
{
        PWR_LED_OFF;
        POT_SELECT;
        SPI_CHAR(0);
        SPI_CHAR(POT_LEVEL);
        POT_DESELECT;
}
```

Appendix 8: “TESTS.c” QA File for Firmware for Prototype Model B

```
// “TESTS.c” QA File for Firmware for Prototype Model B page 1 of 3

#include "main.h"

void SET_FEEDBACK (int FEEDBACK)
{
	PORTC = FEEDBACK;
}

unsigned int GET_INITIAL (int INITIAL_V)
{
	DAC_LEVEL = DAC_START;
	SET_DAC ();
	VOLTAGE = MEASURE_VOLTAGE ();
	if (VOLTAGE > INITIAL_V)
	{
		while (VOLTAGE > INITIAL_V)
		{
			DAC_LEVEL -=50;
			TEST_DAC_RANGE();
			SET_DAC();
			_delay_ms(40);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
	else
	{
		while (VOLTAGE < INITIAL_V)
		{
			DAC_LEVEL +=50;
			TEST_DAC_RANGE();
			SET_DAC();
			_delay_ms(40);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
	_delay_ms(LONG_WAIT);
	TEST_REF_RANGE();
	VOLTAGE = MEASURE_VOLTAGE();
	if (VOLTAGE > INITIAL_V)
	{
		while (VOLTAGE > INITIAL_V)
		{
			DAC_LEVEL -=10;
			TEST_DAC_RANGE();
			SET_DAC();
			TEST_REF_RANGE();
			_delay_ms(40);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
```

```
// “TESTS.c” QA File for Firmware for Prototype Model B page 2 of 3
	else
	{
		while (VOLTAGE < INITIAL_V)
		{
			DAC_LEVEL +=10;
			TEST_DAC_RANGE();
			SET_DAC();
			TEST_REF_RANGE();
			_delay_ms(40);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
	_delay_ms(LONG_WAIT);
	TEST_REF_RANGE();
	VOLTAGE = MEASURE_VOLTAGE();
	if (VOLTAGE > INITIAL_V)
	{
		while (VOLTAGE > INITIAL_V)
		{
			DAC_LEVEL--;
			SET_DAC();
			TEST_DAC_RANGE();
			_delay_ms(20);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
	else
	{
		while (VOLTAGE < INITIAL_V)
		{
			DAC_LEVEL++;
			SET_DAC();
			TEST_DAC_RANGE();
			_delay_ms(20);
			VOLTAGE = MEASURE_VOLTAGE();
		}
	}
	TEST_REF_RANGE();
	return (DAC_LEVEL);
}

void TEST_WE_RANGE (void)
{
	if (HOLD_GAIN > 1)
	{

	}
	else
	{
		if (CURRENT > 1800 || CURRENT < -1800)
		{
			FEEDBACK++;
			if (FEEDBACK > 7)
			{
				FEEDBACK = 7;
			}
			SET_FEEDBACK (FEEDBACK);
			printf ("8765,8765,%d,\n", FEEDBACK);
		}
	}
}
```

```
// “TESTS.c” QA File for Firmware for Prototype Model B page 3 of 3

void TEST_DAC_RANGE (void)
{
	if (DAC_LEVEL >= DAC_MAX || DAC_LEVEL <= DAC_MIN)
	{
		DAC_LEVEL = DAC_START;
		SET_DAC();
		UART_TX_STRING("7778,7778,1,\n");
		PWR_LED_ON;
		SCAN_LED_OFF;
		CE_OFF;
		CELL_OFF;
		main();
	}
}
void TEST_REF_RANGE (void)
{
	ADC_SELECT;
	SPI_CHAR (0b00001100);
	ADC_HI = SPI_CHAR (0b10000000);
	ADC_LO = SPI_CHAR (0b00000000);
	ADC_DESELECT;
	ADC_READING = ADC_HI << 8;
	ADC_READING += ADC_LO;
	ADC_READING = ADC_READING << 3;
	ADC_READING = ADC_READING / 8;
	if (ADC_READING > 4090 || ADC_READING < 10)
	{
		DAC_LEVEL = DAC_START;
		SET_DAC();
		UART_TX_STRING("7779,7779,1,\n");
		PWR_LED_ON;
		SCAN_LED_OFF;
		CE_OFF;
		CELL_OFF;
		main();
	}
}
```

Appendix 9: “MEASUREMENTS.c” Signal File for Firmware for Prototype Model B

```
// “MEASUREMENTS.c” Signal File for Firmware for Prototype Model B page 1 of 3

#include "main.h"
#include <math.h>

/*////////////////////////////////////////////////////////////////////////////
///////
//  JUST AVERAGING START
////////////////////////////////////////////////////////////////////////////
//////*/


int MEASURE_VOLTAGE (void)
{
	VOLTAGE_ADDER = 0;
	for (MEASUREMENTS = 0; MEASUREMENTS < VOLTAGE_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		VOLTAGE_ADDER += READ_VOLTAGE();
	}
	VOLTAGE = VOLTAGE_ADDER / VOLTAGE_MEASUREMENT_NUMBER;
	return (VOLTAGE);
}

int MEASURE_CURRENT (void)
{
	CURRENT_ADDER = 0;
	for (MEASUREMENTS = 0; MEASUREMENTS < CURRENT_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		CURRENT_ADDER += READ_CURRENT();
	}
	CURRENT = CURRENT_ADDER / CURRENT_MEASUREMENT_NUMBER;
	return (CURRENT);
}

/*////////////////////////////////////////////////////////////////////////////
////////
//  JUST AVERAGING END
////////////////////////////////////////////////////////////////////////////
////*/

//////////////////////////////////////////////////////////////////
// READ ONLY BEGIN
//////////////////////////////////////////////////////////////////
/*
int MEASURE_VOLTAGE (void)
{
	return (READ_VOLTAGE());
}

int MEASURE_CURRENT (void)
{
	return (READ_CURRENT());
}
```

```
// “MEASUREMENTS.c” Signal File for Firmware for Prototype Model B page 2 of 3

*/
//////////////////////////////////////////////////////////////////
// READ ONLY END
//////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////
// BASIC DSP WORKING VERSION START NO ARRAY ALPHA = 3
//////////////////////////////////////////////////////////////////

/*
int MEASURE_VOLTAGE (void)
{
	OLD_VOLTAGE = READ_VOLTAGE();
	ARRAY_COUNTER = 0;
	for (MEASUREMENTS = 0; MEASUREMENTS < VOLTAGE_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		VOLTAGE_ADDER = READ_VOLTAGE() + (3 * OLD_VOLTAGE);
		OLD_VOLTAGE = (VOLTAGE_ADDER >> 2);
	}
	return (OLD_VOLTAGE);
}

int MEASURE_CURRENT (void)
{
	OLD_CURRENT = READ_CURRENT();
	for (MEASUREMENTS = 0; MEASUREMENTS < CURRENT_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		CURRENT_ADDER = READ_CURRENT() + (3 * OLD_CURRENT);
		OLD_CURRENT = (CURRENT_ADDER >> 2);
	}
	return (OLD_CURRENT);
}

*/

/*//////////////////////////////////////////////////////////////////////////////
//////
////////////////////     BASIC DSP WORKING VERSION NO ARRAY END
///////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
////*/
```

```
// “MEASUREMENTS.c” Signal File for Firmware for Prototype Model B page 3 of 3

//////////////////////////////////////////////////////////////////
// BASIC DSP WORKING VERSION START WITHOUT ARRAY ALPHA = 3 WITH AVERAGING
//////////////////////////////////////////////////////////////////
/*

int MEASURE_VOLTAGE (void)
{
	VOLTAGE_ADDER = 0;
	OLD_VOLTAGE = READ_VOLTAGE();
	for (MEASUREMENTS = 0; MEASUREMENTS < VOLTAGE_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		NEW_VOLTAGE = (READ_VOLTAGE() + (3 * OLD_VOLTAGE));
		OLD_VOLTAGE = (NEW_VOLTAGE >> 2);
		VOLTAGE_ADDER += OLD_VOLTAGE;
	}
	VOLTAGE = VOLTAGE_ADDER / VOLTAGE_MEASUREMENT_NUMBER;
	return (VOLTAGE);
}

int MEASURE_CURRENT (void)
{
	CURRENT_ADDER = 0;
	OLD_CURRENT = READ_CURRENT();
	for (MEASUREMENTS = 0; MEASUREMENTS < CURRENT_MEASUREMENT_NUMBER;
MEASUREMENTS++)
	{
		NEW_CURRENT = (READ_CURRENT() + (3 * OLD_CURRENT));
		OLD_CURRENT = (NEW_CURRENT >> 2);
		CURRENT_ADDER += OLD_CURRENT;
	}
	CURRENT = CURRENT_ADDER / CURRENT_MEASUREMENT_NUMBER;
	return (CURRENT);
}
*/

/*////////////////////////////////////////////////////////////////////////////////
//////
////////////////////     BASIC DSP WORKING VERSION END WITHOUT ARRAY  WITH AVERAGNG
//////
////////////////////////////////////////////////////////////////////////////////
////*/
```

Appendix 10: “CV.c” CV File for Firmware for Prototype Model B

```
// “CV.c” CV File for Firmware for Prototype Model B page 1 of 3

#include "main.h"

void CV (void)
{
	SCAN_NUMBER = 1;
	SCAN_LED_ON;
	CE_ON;
	CELL_ON;
	_delay_ms(500);
	DAC_LEVEL = GET_INITIAL (START_I);
	SET_DAC();
	TEST_DAC_RANGE();
	TEST_REF_RANGE();
	PWR_LED_OFF;

	VOLTAGE = MEASURE_VOLTAGE();
	{
		if (VOLTAGE >= STOP_I)
		{
			CV_DOWN_FIRST ();
		}
		else
		{
			CV_UP_FIRST ();
		}
	}
	INPUT_RECEIVED = 0;
	FIRST_PASS = 1;
	SCAN_LED_OFF;
	PWR_LED_ON;
	DAC_LEVEL = DAC_START;
	SET_DAC();
	CE_OFF;
	CELL_OFF;
	printf("7775,%lu,%d,\n9999,9999,%d,\n", (ACTUAL_MILLIS - RUN_START),
SCAN_NUMBER, SCAN_NUMBER);
}

void CV_DOWN_FIRST (void)
{
	for (CYCLES = 0; CYCLES < CYCLES_I; CYCLES++)
	{
		printf ("9876,9876,%d,\n", SCAN_NUMBER);
		TEST_REF_RANGE();
		while (VOLTAGE > STOP_I)
		{
			MEASURE_START = ACTUAL_MILLIS;
			DAC_LEVEL -= STEP;
			TEST_DAC_RANGE();
			SET_DAC();
			while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
			{
			}
			VOLTAGE = MEASURE_VOLTAGE();
			CURRENT = MEASURE_CURRENT();
// “CV.c” CV File for Firmware for Prototype Model B page 2 of 3
```

```
                        if (FIRST_PASS)
                        {
                                if (VOLTAGE < (START_I - 50))
                                {
                                        printf ("%d,%d,%d,\n", VOLTAGE, CURRENT,
SCAN_NUMBER);
                                        TEST_WE_RANGE();
                                }
                                else
                                {
                                        RUN_START = ACTUAL_MILLIS;
                                        TEST_WE_RANGE();
                                }
                        }
                        else
                        {
                                printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
                                TEST_WE_RANGE();
                        }
                }
                TEST_REF_RANGE();
                if (FIRST_PASS)
                {
                        START_I -= 50;
                        FIRST_PASS = 0;
                }
                printf ("6789,6789,%d,\n", SCAN_NUMBER);
                while (VOLTAGE < START_I)
                {
                        MEASURE_START = ACTUAL_MILLIS;
                        DAC_LEVEL += STEP;
                        TEST_DAC_RANGE();
                        SET_DAC();
                        while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                        {
                        }
                        VOLTAGE = MEASURE_VOLTAGE();
                        CURRENT = MEASURE_CURRENT();
                        printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
                        TEST_WE_RANGE();
                }
                SCAN_NUMBER++;
                TEST_REF_RANGE();
        }
}

void CV_UP_FIRST (void)
{
        for (CYCLES = 0; CYCLES < CYCLES_I; CYCLES++)
        {
                printf ("6789,6789,%d,\n", SCAN_NUMBER);
                while (VOLTAGE < STOP_I)
                {
                        MEASURE_START = ACTUAL_MILLIS;
                        DAC_LEVEL += STEP;
                        TEST_DAC_RANGE();
                        SET_DAC();
                        while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                        { }
```

```
// “CV.c” CV File for Firmware for Prototype Model B page 3 of 3

                        VOLTAGE = MEASURE_VOLTAGE();
                        CURRENT = MEASURE_CURRENT();
                        if (FIRST_PASS)
                        {
                                if (VOLTAGE > (START_I + 50))
                                {
                                        printf ("%d,%d,%d,\n", VOLTAGE, CURRENT,
SCAN_NUMBER);
                                        TEST_WE_RANGE();
                                }
                                else
                                {
                                        RUN_START = ACTUAL_MILLIS;
                                        TEST_WE_RANGE();
                                }
                        }
                        else
                        {
                                printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
                                TEST_WE_RANGE();
                        }
                }
                TEST_REF_RANGE();
                if (FIRST_PASS)
                {
                        START_I += 50;
                        FIRST_PASS = 0;
                }
                printf ("9876,9876,%d,\n", SCAN_NUMBER);
                while (VOLTAGE > START_I)
                {
                        MEASURE_START = ACTUAL_MILLIS;
                        DAC_LEVEL -= STEP;
                        TEST_DAC_RANGE();
                        SET_DAC();
                        while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                        {
                        }
                        VOLTAGE = MEASURE_VOLTAGE();
                        CURRENT = MEASURE_CURRENT();
                        printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
                        TEST_WE_RANGE();
                }
                SCAN_NUMBER++;
                TEST_REF_RANGE();
        }
}
```

Appendix 11: “LSV.c” LSV File for Firmware for Prototype Model B

```
// “LSV.c” LSV File for Firmware for Prototype Model B page 1 of 2

#include "main.h"

void LSV (void)
{
	SCAN_NUMBER = 1;
	SCAN_LED_ON;
	CE_ON;
	CELL_ON;
	_delay_ms(500);
	DAC_LEVEL = GET_INITIAL (START_I);
	TEST_DAC_RANGE();
	SET_DAC();
	TEST_REF_RANGE();
	PWR_LED_OFF;
	VOLTAGE = MEASURE_VOLTAGE();
	{
		if (VOLTAGE >= STOP_I)
		{
			LSV_DOWN ();
		}
		else
		{
			LSV_UP ();
		}
	}
	INPUT_RECEIVED = 0;
	FIRST_PASS = 1;
	SCAN_LED_OFF;
	PWR_LED_ON;
	DAC_LEVEL = DAC_START;
	SET_DAC();
	CE_OFF;
	CELL_OFF;
	printf("7775,%lu,%d,\n9999,9999,%d,\n", (ACTUAL_MILLIS - RUN_START),
SCAN_NUMBER, SCAN_NUMBER);
}

void LSV_DOWN (void)
{
	printf ("9876,9876,%d,\n", SCAN_NUMBER);
	while (VOLTAGE > STOP_I)
	{
		MEASURE_START = ACTUAL_MILLIS;
		DAC_LEVEL -= STEP;
		TEST_DAC_RANGE();
		SET_DAC();
		while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
		{
		}
		VOLTAGE = MEASURE_VOLTAGE();
		CURRENT = MEASURE_CURRENT();
		if (VOLTAGE < (START_I - 50))
		{
			printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
			TEST_WE_RANGE();
```

```
// “LSV.c” LSV File for Firmware for Prototype Model B page 2 of 2

                }
                else
                {
                        RUN_START = ACTUAL_MILLIS;
                        TEST_WE_RANGE();
                }
        }
}

void LSV_UP (void)
{
        printf ("6789,6789,%d,\n", SCAN_NUMBER);
        while (VOLTAGE < STOP_I)
        {
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL += STEP;
                SET_DAC();
                TEST_DAC_RANGE();
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                VOLTAGE = MEASURE_VOLTAGE();
                CURRENT = MEASURE_CURRENT();
                if (VOLTAGE > (START_I + 50))
                {
                        printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
                        TEST_WE_RANGE();
                }
                else
                {
                        RUN_START = ACTUAL_MILLIS;
                        TEST_WE_RANGE();
                }
        }

}
```

Appendix 12: “SWV.c” SWV File for Firmware for Prototype Model B

```
// “SWV.c” SWV File for Firmware for Prototype Model B page 1 of 2

#include "main.h"

void SWV (void)
{
	SCAN_NUMBER = 1;
	SCAN_LED_ON;
	CE_ON;
	CELL_ON;
	_delay_ms(500);
	DAC_LEVEL = GET_INITIAL (DEPOSITION_V);
	SET_DAC();
	MEASURE_START = ACTUAL_MILLIS;
	printf("2222,2222,2222,\n");
	while (ACTUAL_MILLIS < MEASURE_START + DEPOSITION_T)
	{
	}
	PWR_LED_OFF;
	MEASURE_START = ACTUAL_MILLIS;
	printf("3333,3333,3333,\n");
	while (ACTUAL_MILLIS < MEASURE_START + EQUILIBRIUM_T)
	{
	}
	VOLTAGE = MEASURE_VOLTAGE();
	DAC_LEVEL = GET_INITIAL (START_I);
	SET_DAC();
	VOLTAGE = MEASURE_VOLTAGE();
	printf ("5555,5555,5555,\n");
	if (VOLTAGE >= STOP_I)
	{
		SWV_DOWN ();
	}
	else
	{
		SWV_UP ();
	}
	INPUT_RECEIVED = 0;
	FIRST_PASS = 1;
	SCAN_LED_OFF;
	PWR_LED_ON;
	DAC_LEVEL = DAC_START;
	SET_DAC();
	CE_OFF;
	CELL_OFF;
	printf("7775,%lu,%d,\n9999,9999,%d,\n", (ACTUAL_MILLIS - RUN_START),
SCAN_NUMBER, SCAN_NUMBER);

}

void SWV_DOWN (void)
{
	printf ("9876,9876,%d,\n", SCAN_NUMBER);
	while (VOLTAGE > STOP_I)
	{
		MEASURE_START = ACTUAL_MILLIS;
		DAC_LEVEL = DAC_LEVEL - (STEP + PULSE);
// “SWV.c” SWV File for Firmware for Prototype Model B page 2 of 2
```

```
                TEST_DAC_RANGE();
                SET_DAC();
                TEST_REF_RANGE();
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                CURRENT_CATHODIC = MEASURE_CURRENT();
                TEST_WE_RANGE();
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL += PULSE;
                TEST_DAC_RANGE();
                SET_DAC();
                TEST_REF_RANGE();
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                VOLTAGE = MEASURE_VOLTAGE();
                CURRENT_ANODIC = MEASURE_CURRENT();
                TEST_WE_RANGE();
                CURRENT = CURRENT_CATHODIC - CURRENT_ANODIC;
                printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
        }

}

void SWV_UP (void)
{
        printf ("6789,6789,%d,\n", SCAN_NUMBER);
        while (VOLTAGE < STOP_I)
        {
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL = DAC_LEVEL + STEP + PULSE;
                TEST_DAC_RANGE();
                SET_DAC();
                TEST_REF_RANGE();
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                CURRENT_ANODIC = MEASURE_CURRENT();
                TEST_WE_RANGE();
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL -= PULSE;
                TEST_DAC_RANGE();
                SET_DAC();
                TEST_REF_RANGE();
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                VOLTAGE = MEASURE_VOLTAGE();
                CURRENT_CATHODIC = MEASURE_CURRENT();
                TEST_WE_RANGE();
                CURRENT = CURRENT_ANODIC - CURRENT_CATHODIC;
                printf ("%d,%d,%d,\n", VOLTAGE, CURRENT, SCAN_NUMBER);
        }
}
```

Appendix 13: “ALPHA_TEST.c” debugging File for Firmware for Prototype Model B

```
// “ALPHA_TEST.c” debugging File for Firmware for Prototype Model B page 1 of 4

#include "main.h"

void VOLTAGE_ALPHA_TEST (void)
{
	SCAN_NUMBER = 1;
	SCAN_LED_ON;
	CE_ON;
	CELL_ON;
	_delay_ms(500);
	DAC_LEVEL = GET_INITIAL (START_I);
	SET_DAC();
	PWR_LED_OFF;
	VOLTAGE = MEASURE_VOLTAGE();
	{
		if (VOLTAGE >= STOP_I)
		{
			VOLTAGE_ALPHA_DOWN ();
		}
		else
		{
			VOLTAGE_ALPHA_UP ();
		}
	}
	INPUT_RECEIVED = 0;
	FIRST_PASS = 1;
	SCAN_LED_OFF;
	PWR_LED_ON;
	DAC_LEVEL = DAC_START;
	SET_DAC();
	CE_OFF;
	CELL_OFF;
	//  For Voltage alpha test only
////////////////////////////////////////////////////////
	// printf("CELL:\tDummy Cell\tWE:\tNot app\tParameter:\tVoltage\tMeasurements
Number:\t%d\tPre-Wait\t%d\tDAC step:\t%d\tPoint Range:\t%d\tTime:\t%lu\n",
VOLTAGE_MEASUREMENT_NUMBER, WAIT, STEP, POINT_NUMBER, (ACTUAL_MILLIS - RUN_START));
	printf ("7775,7775,%d,\n", POINT_NUMBER);
	_delay_ms(100);
	printf ("9999,9999,%lu,\n", (ACTUAL_MILLIS - RUN_START));
	////////////////////////////////////////////////////////////////////////////
///////////////

}

void VOLTAGE_ALPHA_DOWN (void)
{
	RUN_START = ACTUAL_MILLIS;
	POINT_NUMBER = 0;
	while (VOLTAGE > STOP_I)
	{
		MEASURE_START = ACTUAL_MILLIS;
		DAC_LEVEL -= STEP;
```

```
                SET_DAC();
                TEST_DAC_RANGE();
                TEST_REF_RANGE();
// “ALPHA_TEST.c” debugging File for Firmware for Prototype Model B page 2 of 4

                for (int WAITER = 0; WAITER < WAIT; WAITER++)
                {
                }
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                for (int i = 0; i < VOLTAGE_MEASUREMENT_NUMBER; i++)
                {
                        VOLTAGE = READ_VOLTAGE();
                        printf("%d,%d,%d,\n", POINT_NUMBER, i, VOLTAGE);
                }
                POINT_NUMBER++;
        }
}

void VOLTAGE_ALPHA_UP (void)
{
        RUN_START = ACTUAL_MILLIS;
        POINT_NUMBER = 0;
        while (VOLTAGE < STOP_I)
        {
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL += STEP;
                SET_DAC();
                TEST_DAC_RANGE();
                TEST_REF_RANGE();
                for (int WAITER = 0; WAITER < WAIT; WAITER++)
                {
                }
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                for (int i = 0; i < VOLTAGE_MEASUREMENT_NUMBER; i++)
                {
                        VOLTAGE = READ_VOLTAGE();
                        printf("%d,%d,%d,\n", POINT_NUMBER, i, VOLTAGE);
                }
                POINT_NUMBER++;
        }
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////
 // Current section

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////

 void CURRENT_ALPHA_TEST (void)
 {
        SCAN_NUMBER = 1;
        SCAN_LED_ON;
        CE_ON;
        CELL_ON;
        _delay_ms(500);
        DAC_LEVEL = GET_INITIAL (START_I);
        SET_DAC();
```

```
        PWR_LED_OFF;
        VOLTAGE = MEASURE_VOLTAGE();
        {
// “ALPHA_TEST.c” debugging File for Firmware for Prototype Model B page 3 of 4

                if (VOLTAGE >= STOP_I)
                {
                        CURRENT_ALPHA_DOWN ();
                }
                else
                {
                        CURRENT_ALPHA_UP ();
                }
        }
        INPUT_RECEIVED = 0;
        FIRST_PASS = 1;
        SCAN_LED_OFF;
        PWR_LED_ON;
        DAC_LEVEL = DAC_START;
        SET_DAC();
        CE_OFF;
        CELL_OFF;
        printf ("7775,7775,%d,\n", POINT_NUMBER);
        _delay_ms(100);
        printf ("9999,9999,%lu,\n", (ACTUAL_MILLIS - RUN_START));
}

void CURRENT_ALPHA_DOWN (void)
{
        RUN_START = ACTUAL_MILLIS;
        POINT_NUMBER = 0;
        while (VOLTAGE > STOP_I)
        {
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL -= STEP;
                SET_DAC();
                TEST_DAC_RANGE();
                TEST_REF_RANGE();
                for (int WAITER = 0; WAITER < WAIT; WAITER++)
                {
                }
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                VOLTAGE = READ_VOLTAGE();
                for (int i = 0; i < VOLTAGE_MEASUREMENT_NUMBER; i++)
                {
                        CURRENT = READ_CURRENT();
                        printf("%d,%d,%d,\n", POINT_NUMBER, i, CURRENT);
                }
                POINT_NUMBER++;
        }
}

void CURRENT_ALPHA_UP (void)
{
        RUN_START = ACTUAL_MILLIS;
        POINT_NUMBER = 0;
        while (VOLTAGE < STOP_I)
        {
                MEASURE_START = ACTUAL_MILLIS;
                DAC_LEVEL += STEP;
```

```
                SET_DAC();
                TEST_DAC_RANGE();
                TEST_REF_RANGE();
// “ALPHA_TEST.c” debugging File for Firmware for Prototype Model B page 4 of 4

                for (int WAITER = 0; WAITER < WAIT; WAITER++)
                {
                }
                while (ACTUAL_MILLIS < (MEASURE_START + WAIT))
                {
                }
                VOLTAGE = READ_VOLTAGE();
                for (int i = 0; i < VOLTAGE_MEASUREMENT_NUMBER; i++)
                {
                        CURRENT = READ_CURRENT();
                        printf("%d,%d,%d,\n", POINT_NUMBER, i, CURRENT);
                }
                POINT_NUMBER++;
        }
}
```

Appendix 14: liteSTAT GUI Processing Source Code

```
// liteSTAT GUI code for Prototype Model B page 1 of 35

import interfascia.*;
import processing.pdf.*;
import processing.serial.*;
import controlP5.*;

ControlP5 cp5;
GUIController c;

Table calibrationTable;
PImage splashLogo;
PImage linkLogo;
PFont font;
Serial port;
int scanCounter, colourCounter, chosenScan;
float DEPOSITION_V_F, FREQUENCY_F, STEPPER_F, PULSE_F;
int DEPOSITION_T_I, EQUILIBRIUM_T_I, PULSE_I, WAITER_I, STEPPER_I, ALPHA_I;
String DEPOSITION_V_S, DEPOSITION_T_S, EQUILIBRIUM_T_S, PULSE_S, ALPHA_S, SHIFT_S;
float fb2Resistance, fb2Factor, currentGain;
float slope, intercept, interceptUp, interceptDown, slopeUpAverage, slopeDownAverage, slopeUp,
slopeDown;
boolean slopeUpBool = true;
boolean firstSweep = true;
boolean singleScanOnly = false;
int sweepCounter = 0;
// Calibration fields
IFTextField interceptUpT, interceptDownT, slopeT;
IFLabel interceptUpL, interceptDownL, slopeL;
// Calibration fields
IFTextField startT, endT, rateT, cyclesT, depPotT, depTimeT, equiTimeT, pulseT, stepT, saveT,
scanNumberT, waitT, peakStartT, peakEndT, peakMaxT;
IFButton cvB, swvB, lsvB, chronoB, runB, clearB, sgB, exportScanB, loadDataB, loadMethodB, peaksB,
portB, voltB, testB, scanNumberB, maxB;
IFLabel startL, endL, rateL, cyclesL, depPotL, depTimeL, equiTimeL, pulseL, stepL, voidL, debugL, saveL,
highL, lowL, cRangeL, waitL, peakStartL, peakEndL, peakMaxL;
IFLabel gridAL, gridBL, gridCL, gridDL, gridEL, gridFL, gridGL, gridHL, gridIL, gridJL, gridKL, gridLL, gridML,
gridNL, gridOL, gridPL;
IFLabel currentL, voltageL;
IFCheckBox screenGrabCB, holdCB;
int currentSubtractor = 0;
boolean skipPoint = false;
int baseCounter;
float voltF, currentF, voltFS, currentFS, current2FS;
float startF, stopF;
float start, end, depPot, tempsg;
float inX, inY;
int cyclesI, rateI, depTimeI, equiTimeI, pulseI, stepI, frameCounter, pointCounter;
int WAITER, V_NUM, I_NUM, STEPPER, WAITER_ADDER;
String WAITER_S, V_NUM_S, I_NUM_S, STEPPER_S, HOLD_S;
color rguColour = color (200, 100, 200);
boolean filterPoint = false;
boolean portChoice = false;
boolean cv = false;
boolean lsv = false;
boolean swv = false;
boolean chrono = false;
boolean scanning = false;
boolean recordingPDF = false;
boolean showData = false;
boolean smooth = false;
boolean displayPoint = false;
boolean capture = false;
boolean showPeaks = false;
boolean stirrerOn = false;
boolean stirrerOff = false;
```

```
// liteSTAT GUI code for Prototype Model B page 2 of 35

boolean peakDetection = false;
boolean splash = true;
boolean outRange = false;
boolean compliance = false;
boolean notConnected = true;
boolean dontSmooth = false;
boolean noDataYet = true;
String toSendS, methodS, startS, endS, rateS, cyclesS, fb1S, fb2S, depPotS, depTimeS, equiTimeS, pulseS,
stepS, inputS, potS, stopS;
String complianceS = "1";
Scan[] scans = new Scan [0];
float xmin, xmax, ymin, ymax, xmintest, ymintest, xmaxtest, ymaxtest, difftest, diffmax;
float xRange, yRange;
int textX, textY, potI;
float gridAF, gridBF, gridCF, gridDF, gridEF, gridFF, gridGF, gridHF, gridIF, gridJF, gridKF, gridLF, gridMF,
gridNF, gridOF, gridPF, zeroX, zeroY;
String gridAS, gridBS, gridCS, gridDS, gridES, gridFS, gridGS, gridHS, gridIS, gridJS, gridKS, gridLS,
gridMS, gridNS, gridOS, gridPS;
String loadPath, filename;
String activePortS = "";
float pointerX, pointerY, graphX, graphY;
float dx, dy, diff;
float currentLo, currentHi, area;
float areaRectXpos1, areaRectYpos1, areaRectXpos2, areaRectYpos2;
float DAC_POINT;
int slopeNumber = 0;
int smoothLevel, smoothPoints, rowCounter, rowCenter, rowFactor, totalScans;
int lowestActiveScan = 0;
float[] rowVal = new float[20];
boolean holdGain = false;

void setup()
{
  size(1020, 600);
  fb1S = "255";
  scanCounter = -1;
  font = createFont ("Arial", 14);
  textFont (font, 14);
  fill (0);
  stroke (0);
  pointCounter = 0;
  if (Serial.list().length > 0)
  {
    port = new Serial (this, Serial.list()[0], 1000000);
    port.bufferUntil ('\n');
    notConnected = false;
  }
  linkLogo = loadImage ("linkLogo.jpg");
  splashLogo = loadImage ("splashLogo.jpg");
  calibrationTable = loadTable ("calibrationTable.csv", "header");
}

void draw()
{
  if (frameCount < 255)
  {
    splashScreen ();
  }
  else if (frameCount == 255)
  {
    addInitials ();
  }
  else
  {
    background (rguColour);
    fill (255);
    image (linkLogo, 915, 500);
    rect (180, 20, 720, 560);
    for (int i = 220; i <= 900; i += 80)
```

```
// liteSTAT GUI code for Prototype Model B page 3 of 35

  {
   stroke (210);
   line (i, 21, i, 579);
  }
  if (Serial.list().length == 0)
  {

   notConnected = true;
   //removeAllTandL ();
  }
  else
  {
  }
  for (int i = 60; i <= 600; i += 80)
  {
   stroke (210);
   line (181, i, 899, i);
   stroke (0);
   fill (0);
  }
  if (scans.length >= 1)
  {
   removeZero ();
  }
  for (int i = 0; i < scans.length; i++)
  {
   for (int j = 0; j < scans[i].mainTable.getRowCount (); j++)
   {
    TableRow row = scans[i].mainTable.getRow(j);
    // Occasional error on next line "This table has no column named 'voltage'
    xmintest = row.getFloat ("voltage");
    xmaxtest = row.getFloat ("voltage");
    ymintest = row.getFloat ("current");
    ymaxtest = row.getFloat ("current");
    if (xmintest < xmin)
    {
     xmin = xmintest;
    }
    if (xmaxtest > xmax)
    {
     xmax = xmaxtest;
    }
    if (ymintest < ymin)
    {
     ymin = ymintest;
    }
    if (ymaxtest > ymax)
    {
     ymax = ymaxtest;
    }
    if (xmax == xmin)
    {
     xmin = 10000000.0;
    }
    if (ymin == ymax)
    {
     ymin = 1000000000.0;
    }
   }
  }
  for (int i = 0; i < scans.length; i++)
  {
   for (int j = 0; j < scans[i].mainTable.getRowCount (); j++)
   {
    TableRow row = scans[i].mainTable.getRow (j);
    row.setFloat ("xPlot", map (row.getFloat ("voltage"), xmin, xmax, 220.0, 860.0));
    row.setFloat ("yPlot", map (row.getFloat ("current"), ymin, ymax, 540.0, 60.0));
   }
  }
```

```
// liteSTAT GUI code for Prototype Model B page 4 of 35

  if (smooth && showData)
  {
   smoothData ();
   for (int i = 0; i < scans.length; i++)
   {
    stroke (scans[i].scanColour);
    strokeWeight (1);
    for (int j = 1; j < (scans[i].mainTable.getRowCount() - 1); j++)
    {
     TableRow row = scans[i].mainTable.getRow (j);
     TableRow rowl = scans[i].mainTable.getRow (j-1);
     line (rowl.getFloat("xPlot"), rowl.getFloat("ySg"), row.getFloat("xPlot"), row.getFloat("ySg"));
    }
   }
  }
  else
  {
   for (int i = 0; i < scans.length; i++)
   {
    stroke (scans[i].scanColour);
    strokeWeight (1);
    for (int j = 1; j < scans[i].mainTable.getRowCount () - 1; j++)
    {
     TableRow row = scans[i].mainTable.getRow (j);
     TableRow rowl = scans[i].mainTable.getRow (j-1);
     line (rowl.getFloat ("xPlot"), rowl.getFloat ("yPlot"), row.getFloat ("xPlot"), row.getFloat ("yPlot"));
    }
   }
  }
  if (!showData)
  {
   fill (255);
   rect (180, 20, 720, 560);
  }
  strokeWeight (1);
  xRange = (xmax - xmin) / 8;
  yRange = (ymax - ymin) / 6;
  gridAF = xmin;
  gridBF = xmin + xRange;
  gridCF = xmin + (2 * xRange);
  gridDF = xmin + (3 * xRange);
  gridEF = xmin + (4 * xRange);
  gridFF = xmin + (5 * xRange);
  gridGF = xmin + (6 * xRange);
  gridHF = xmin + (7 * xRange);
  gridIF = xmin + (8 * xRange);
  gridJF = ymin;
  gridKF = ymin + yRange;
  gridLF = ymin + (2 * yRange);
  gridMF = ymin + (3 * yRange);
  gridNF = ymin + (4 * yRange);
  gridOF = ymin + (5 * yRange);
  gridPF = ymin + (6 * yRange);
  gridAF = round(gridAF * 1000.0f)/1000.0f;
  gridBF = round(gridBF * 1000.0f)/1000.0f;
  gridCF = round(gridCF * 1000.0f)/1000.0f;
  gridDF = round(gridDF * 1000.0f)/1000.0f;
  gridEF = round(gridEF * 1000.0f)/1000.0f;
  gridFF = round(gridFF * 1000.0f)/1000.0f;
  gridGF = round(gridGF * 1000.0f)/1000.0f;
  gridHF = round(gridHF * 1000.0f)/1000.0f;
  gridIF = round(gridIF * 1000.0f)/1000.0f;
  gridJF = round(gridJF * 1000.0f)/1000.0f;
  gridKF = round(gridKF * 1000.0f)/1000.0f;
  gridLF = round(gridLF * 1000.0f)/1000.0f;
  gridMF = round(gridMF * 1000.0f)/1000.0f;
  gridNF = round(gridNF * 1000.0f)/1000.0f;
  gridOF = round(gridOF * 1000.0f)/1000.0f;
  gridPF = round(gridPF * 1000.0f)/1000.0f;
```

```
// liteSTAT GUI code for Prototype Model B page 5 of 35

   gridAS = str (gridAF);
   gridBS = str (gridBF);
   gridCS = str (gridCF);
   gridDS = str (gridDF);
   gridES = str (gridEF);
   gridFS = str (gridFF);
   gridGS = str (gridGF);
   gridHS = str (gridHF);
   gridIS = str (gridIF);
   gridJS = str (gridJF);
   gridKS = str (gridKF);
   gridLS = str (gridLF);
   gridMS = str (gridMF);
   gridNS = str (gridNF);
   gridOS = str (gridOF);
   gridPS = str (gridPF);
   if (scanning)
   {
    text (potI, 200, 200);
    zeroX = map (0, xmin, xmax, 220.0, 860);
    zeroY = map (0, ymin, ymax, 540.0, 60.0);
    if (zeroX > 200 && zeroX < 860 && zeroY > 60 && zeroY < 540)
    {
     strokeWeight (0.2);
     line (zeroX, 540, zeroX, 60);
     line (220, zeroY, 860, zeroY);
     strokeWeight (1);
    }
    gridAL.setLabel (gridAS);
    gridBL.setLabel (gridBS);
    gridCL.setLabel (gridCS);
    gridDL.setLabel (gridDS);
    gridEL.setLabel (gridES);
    gridFL.setLabel (gridFS);
    gridGL.setLabel (gridGS);
    gridHL.setLabel (gridHS);
    gridIL.setLabel (gridIS);
    gridJL.setLabel (gridJS);
    gridKL.setLabel (gridKS);
    gridLL.setLabel (gridLS);
    gridML.setLabel (gridMS);
    gridNL.setLabel (gridNS);
    gridOL.setLabel (gridOS);
    gridPL.setLabel (gridPS);
   }
   if (!scanning)
   {
    xmin = 1000000000.0;
    xmax = -1000000000.0;
    ymin = 1000000000.0;
    ymax = -1000000000.0;
    xRange = 5.0;
    yRange = 100.0;
    gridAL.setLabel ("");
    gridBL.setLabel ("");
    gridCL.setLabel ("");
    gridDL.setLabel ("");
    gridEL.setLabel ("");
    gridFL.setLabel ("");
    gridGL.setLabel ("");
    gridHL.setLabel ("");
    gridIL.setLabel ("");
    gridJL.setLabel ("");
    gridKL.setLabel ("");
    gridLL.setLabel ("");
    gridML.setLabel ("");
    gridNL.setLabel ("");
    gridOL.setLabel ("");
    gridPL.setLabel ("");
```

```
// liteSTAT GUI code for Prototype Model B page 6 of 35

    }
    displayPoint = rollOver (mouseX, mouseY);
    if (displayPoint && !peakDetection)
    {
      cursor (CROSS);
      pointerX = map (mouseX, 220, 860, xmin, xmax);
      pointerX = round (pointerX * 1000.0f)/1000.0f;
      pointerY = map (mouseY, 60, 540, ymax, ymin);
      pointerY = round (pointerY * 1000.0f)/1000.0f;
      text (pointerX + " V, " + pointerY + " uA", 185, 560);
    }
    else if (!displayPoint && !peakDetection)
    {
      cursor (ARROW);
    }
    else if (peakDetection)
    {
      positivePeak ();
    }
  }
  if (stirrerOn) {
    noStroke ();
    fill (0, 255, 0);
    rect (480, 250, 20, 20);
    text ("Switch Stirrer On", 505, 264);
  }
  if (stirrerOff) {
    noStroke ();
    fill (255, 0, 0);
    rect (480, 250, 20, 20);
    text ("Switch Stirrer Off", 505, 264);
  }
  if (outRange) {
    noStroke ();
    fill (255, 0, 0);
    rect (480, 250, 20, 20);
    text ("Voltage Out of Range", 505, 264);
  }
  if (portChoice) {
    if (!notConnected) {
      if (Serial.list().length > 0) port.stop();
    }
    fill (127);
    rect (180, 20, 720, 560);
    cursor (ARROW);
    String list[];
    list = (Serial.list());
    int a = list.length;
    textFont (font, 14);
    fill (255);
    stroke (255);
    text ("Click on a port number to select it: ", 250, 98);
    int b = 100;
    for (int i = 0; i < a; i++) {
      text (list[i], 500, b);
      strokeWeight (3);
      noFill ();
      rect (490, b - 18, 65, 24);
      if (mousePressed) {
        if (mouseX >= 490 && mouseX <= 550 && mouseY >= (b - 18) && mouseY <= (b + 24)) {
          port = new Serial (this, Serial.list()[i], 1000000);
          port.bufferUntil ('\n');
          notConnected = false;
          activePortS = list[i];
          portChoice = false;
        }
      }
      b += 50;
    }
```

```
// liteSTAT GUI code for Prototype Model B page 7 of 35

  }
}

void serialEvent (Serial port) {
  scanning = true;
  inputS = port.readString ();
  if (inputS != null) {
    //println ("Got piece of data");
    int[] vals = int(split(inputS, ","));
    //println ("Something got through");
    if (vals[0] == 9999 || vals[1] == 9999) {
      finishedScan ();
      filterPoint = false;
      stirrerOn = false;
      stirrerOff = false;
    } else if (vals[0] == 7775)
    {
      println (vals[1]);
    } else if (vals[0] == 2222 && vals[1] == 2222) {
      stirrerOn = true;
      stirrerOff = false;
      scanning = false;
    } else if (vals[0] == 3333 && vals[1] == 3333) {
      stirrerOn = false;
      stirrerOff = true;
      scanning = false;
    } else if (vals[0] == 5555 && vals[1] == 5555) {
      stirrerOn = false;
      stirrerOff = false;
      scanning = true;
    } else if (vals[0] == 9876 || vals[1] == 9876)
    {
      slopeUpBool = false;
      slopeNumber = 0;
      slope = slopeDown;
      intercept = interceptDown;
      sweepCounter++;
    }
    else if (vals[0] == 6789 || vals[1] == 6789)
    {
      slopeUpBool = true;
      slopeNumber = 0;
      slope = slopeUp;
      intercept = interceptUp;
      sweepCounter++;
    }
    else if (vals[0] == 8765 || vals[1] == 8765)
    {
      potI = vals[2] + 1;
      cp5.getController(" ").setValue(potI);
      potI = vals[2] + 1000;
      if (cv || lsv)
      {
        setCurrentFactors (); //<>//
      // Calibration Fields
      }
    }
    else if (vals[0] == 7654)
    {

      print ("Start_I: ");
      print (vals[1]);
      print("\tDac Level: ");
      print (vals[2]);
      print ("\tVoltage: ");
      println (vals[3]);

    }
    else if (vals[0] == 4444 || vals[1] == 4444)
```

```
// liteSTAT GUI code for Prototype Model B page 8 of 35

  {

  }
  else if (vals[0] == 7777 || vals[1] == 7777) {
    outRange = true;
  }
  else if (vals[0] == 7778 || vals[1] == 7778) {
    println ("");
    println ("It was the DAC");
  }
  else if (vals[0] == 7779 || vals[1] == 7779)
  {
    println ("");
    println ("It was the REF");
  }
  else
  {
    voltF = (vals[0] * 5.0 / 4096.0);
    currentF = float (vals[1]);
    if (cv || lsv)
    {
      currentF = (currentF - intercept) / slope;
    }

    try
    {

      voltFS = voltF;
    }
    catch (java.lang.Exception e)
    {
      skipPoint = true;
    }
    try
    {
      currentFS = currentF;

    }
    catch (java.lang.Exception e)
    {
      skipPoint = true;
    }
    if (skipPoint)
    {
    } else
    {
      if (currentFS == 0 || voltFS == 0)
      {
      }
      else
      {
        TableRow row = scans[scanCounter].mainTable.addRow ();
        row.setFloat ("xVal", float (vals[0]));
        row.setFloat ("yVal", float (vals[1]));
        row.setFloat ("voltage", voltF);
        row.setFloat ("current", currentFS);
        row.setInt ("Scan Number", vals[2]);
      }
    }
    skipPoint = false;
    slopeNumber++;
  }
  pointCounter++;
  print (vals[0]);
  print ("\t");
  print (vals[1]);
  print ("\t");
  println (vals[2]);
 }}
```

```
// liteSTAT GUI code for Prototype Model B page 9 of 35

void makeMins () {
   xmin = 1000000000.0;
   xmax = -1000000000.0;
   ymin = 1000000000.0;
   ymax = -1000000000.0;
   xRange = 5.0;
   yRange = 100.0;
}

void setCurrentFactors ()
{
  if (!swv)
  {
    println ("Line1");
    String tableRowUpSlopeS, tableRowUpInterceptS, tableRowDownSlopeS, tableRowDownInterceptS,
complianceRowS;
    println ("Line2");
    potI -= 1000;
    tableRowUpSlopeS = "Fb" + str(potI) + "Up";
    tableRowUpInterceptS = "Fb" + str(potI) + "Up";
    tableRowDownSlopeS = "Fb" + str(potI) + "Down";
    tableRowDownInterceptS = "Fb" + str(potI) + "Down";
    potI += 1000;
    println ("Line3");
    if (compliance)
    {
      complianceRowS = "Hi";
    }
    else
    {
      complianceRowS = "Lo";
    }
    println ("Line4");
    tableRowUpSlopeS += complianceRowS + "Slope";
    tableRowUpInterceptS += complianceRowS + "Intercept";
    tableRowDownSlopeS += complianceRowS + "Slope";
    tableRowDownInterceptS += complianceRowS + "Intercept";
    println ("Look up column:" + tableRowUpSlopeS);
    println ("Line5");
    TableRow rowUpSlope = calibrationTable.matchRow(tableRowUpSlopeS, "Scan Type:");
    println ("Line5.1");
    TableRow rowUpIntercept = calibrationTable.matchRow(tableRowUpInterceptS, "Scan Type:");
    TableRow rowDownSlope = calibrationTable.matchRow(tableRowDownSlopeS, "Scan Type:");
    TableRow rowDownIntercept = calibrationTable.matchRow(tableRowDownInterceptS, "Scan Type:");
    println ("Line6");
    slopeUp = rowUpSlope.getFloat(str(rateI));
    interceptUp = rowUpIntercept.getFloat(str(rateI));
    slopeDown = rowDownSlope.getFloat(str(rateI));
    interceptDown = rowDownIntercept.getFloat(str(rateI));
    if (slopeUpBool)
    {
      slope = slopeUp;
      intercept = interceptUp;
    }
    else
    {
      slope = slopeDown;
      intercept = interceptDown;
    }
    println ("Line7");
  }
}

void setRate ()
{
  rateS = rateT.getValue ();
  rateI = int (rateS);
  if (rateI < 10)
  {
```

```
// liteSTAT GUI code for Prototype Model B page 10 of 35

  rateI = 5;
 }
 else if (rateI >= 10 && rateI < 25)
 {
  rateI = 10;
 }
 else if (rateI >= 25 && rateI < 50)
 {
  rateI = 25;
 }
 else if (rateI >= 50 && rateI < 75)
 {
  rateI = 50;
 }
 else if (rateI >= 75 && rateI < 100)
 {
  rateI = 75;
 }
 else if (rateI >= 100 && rateI < 150)
 {
  rateI = 100;
 }
 else if (rateI >= 150 && rateI < 200)
 {
  rateI = 150;
 }
 else if (rateI >= 200 && rateI < 250)
 {
  rateI = 200;
 }
 else if (rateI >= 250 && rateI < 300)
 {
  rateI = 250;
 }
 else if (rateI >= 300 && rateI < 350)
 {
  rateI = 300;
 }
 else if (rateI >= 350 && rateI < 400)
 {
  rateI = 350;
 }
 else if (rateI >= 400 && rateI < 450)
 {
  rateI = 400;
 }
 else if (rateI >= 450 && rateI < 500)
 {
  rateI = 450;
 }
 else
 {
  rateI = 500;
 }
 rateS = str (rateI);
 rateT.setValue(rateS);
 if (compliance)
 {
  // HIGH COMPLIANCE
  DAC_POINT = 10000.0 / 4096.0;
  complianceS = "2";
  switch (rateI)
  {
   case 5:
   {
    STEPPER = 1;
    WAITER_S = "1000";
    I_NUM_S = "9000";
    V_NUM_S = "9000";
```

```
// liteSTAT GUI code for Prototype Model B page 11 of 35

     break;
    }
    case 10:
    {
     STEPPER = 1;
     WAITER_S = "1000";
     I_NUM_S = "6700";
     V_NUM_S = "4000";
     break;
    }
    case 25:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "5878";
     V_NUM_S = "2998";
     break;
    }
    case 50:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "3455";
     V_NUM_S = "2000";
     break;
    }
    case 75:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "2305";
     V_NUM_S = "2000";
     break;
    }
    case 100:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "1724";
     V_NUM_S = "2000";
     break;
    }
    case 150:
    {
     STEPPER = 3;
     WAITER_S = "1000";
     I_NUM_S = "1611";
     V_NUM_S = "2112";
     break;
    }
    case 200:
    {
     STEPPER = 4;
     WAITER_S = "1000";
     I_NUM_S = "1608";
     V_NUM_S = "2108";
     break;
    }
    case 250:
    {
     STEPPER = 5;
     WAITER_S = "1000";
     I_NUM_S = "1320";
     V_NUM_S = "2400";
     break;
    }
    case 300:
    {
     STEPPER = 6;
```

```
// liteSTAT GUI code for Prototype Model B page 12 of 35

     WAITER_S = "1000";
     I_NUM_S = "1312";
     V_NUM_S = "2400";
     break;
    }
    case 350:
    {
     STEPPER = 7;
     WAITER_S = "1000";
     I_NUM_S = "1301";
     V_NUM_S = "2400";
     break;
    }
    case 400:
    {
     STEPPER = 8;
     WAITER_S = "1000";
     I_NUM_S = "1200";
     V_NUM_S = "2500";
     break;
    }
    case 450:
    {
     STEPPER = 9;
     WAITER_S = "1000";
     I_NUM_S = "1105";
     V_NUM_S = "2600";
     break;
    }
    case 500:
    {
     STEPPER = 10;
     WAITER_S = "1000";
     I_NUM_S = "1090";
     V_NUM_S = "2600";
     break;
    }
    default:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "1724";
     V_NUM_S = "2000";
     break;
    }

  }
 }
 else
 {
  ALPHA_S = "107";
  SHIFT_S = "3";
  DAC_POINT = 5000.0 / 4096.0;
  complianceS = "1";
  switch (rateI)
  {
   case 5:
   {
    STEPPER = 1;
    WAITER_S = "1000";
    I_NUM_S = "5350";
    V_NUM_S = "4000";
    break;
   }
   case 10:
   {
    STEPPER = 1;
    WAITER_S = "1000";
    I_NUM_S = "3200";
```

```
// liteSTAT GUI code for Prototype Model B page 13 of 35

     V_NUM_S = "2450";
     break;
    }
    case 25:
    {
     STEPPER = 2;
     WAITER_S = "1000";
     I_NUM_S = "2710";
     V_NUM_S = "2200";
     break;
    }
    case 50:
    {
     STEPPER = 3;
     WAITER_S = "1000";
     I_NUM_S = "2400";
     V_NUM_S = "1980";
     break;
    }
    case 75:
    {
     STEPPER = 3;
     WAITER_S = "1000";
     I_NUM_S = "1550";
     V_NUM_S = "1900";
     break;
    }
    case 100:
    {
     STEPPER = 4;
     WAITER_S = "1000";
     I_NUM_S = "1550";
     V_NUM_S = "1900";
     break;
    }
    case 150:
    {
     STEPPER = 6;
     WAITER_S = "1000";
     I_NUM_S = "1550";
     V_NUM_S = "1900";
     break;
    }
    case 200:
    {
     STEPPER = 8;
     WAITER_S = "1000";
     I_NUM_S = "1250";
     V_NUM_S = "2200";
     break;
    }
    case 250:
    {
     STEPPER = 9;
     WAITER_S = "1000";
     I_NUM_S = "1100";
     V_NUM_S = "2200";
     break;
    }
    case 300:
    {
     STEPPER = 12;
     WAITER_S = "1000";
     I_NUM_S = "1140";
     V_NUM_S = "2310";
     break;
    }
    case 350:
    {
```

```
// liteSTAT GUI code for Prototype Model B page 14 of 35

      STEPPER = 14;
      WAITER_S = "1000";
      I_NUM_S = "1040";
      V_NUM_S = "2400";
      break;
     }
     case 400:
     {
      STEPPER = 18;
      WAITER_S = "1000";
      I_NUM_S = "1120";
      V_NUM_S = "2500";
      break;
     }
     case 450:
     {
      STEPPER = 20;
      WAITER_S = "1000";
      I_NUM_S = "1030";
      V_NUM_S = "2550";
      break;
     }
     case 500:
     {
      STEPPER = 22;
      WAITER_S = "1000";
      I_NUM_S = "1020";
      V_NUM_S = "2550";
      break;
     }
     default:
     {
      STEPPER = 4;
      WAITER_S = "1000";
      I_NUM_S = "1550";
      V_NUM_S = "1900";
      break;
     }
    }
  }
  STEPPER += 1000;
  STEPPER_S = str (STEPPER);
}
class Scan {
  Table mainTable, fileTable, methodTable;
  color scanColour;
  int peakStartRow, peakEndRow;
  float peakStartX, peakStartY, peakEndX, peakEndY, baseCurrent, peakHeight;
  float peakVoltage, peakCurrent, peakStartVoltage, peakEndVoltage, peakStartCurrent, peakEndCurrent,
testMaxCurrent, peakX, peakY, baseY;
  float m, c, slope, intercept, peakArea;
  boolean peakFound, peakTrue, tester;
  Scan (color scanColour_) {
    mainTable = new Table ();
    mainTable.addColumn("xVal");
    mainTable.addColumn("yVal");
    mainTable.addColumn("voltage");
    mainTable.addColumn("current");
    mainTable.addColumn("xPlot");
    mainTable.addColumn("yPlot");
    mainTable.addColumn("sg");
    mainTable.addColumn("ySg");
    mainTable.addColumn("firstDiff");
    mainTable.addColumn("firstDiffSmooth");
    mainTable.addColumn("Scan Number");
    scanColour = scanColour_;
    peakFound = false;
    peakTrue = false;
    tester = true;
```

```
// liteSTAT GUI code for Prototype Model B page 15 of 35

   peakCurrent = -100000.0;
  }
  void getPeaks () {
   peakStartVoltage = map (peakStartX, 220, 860, xmin, xmax);
   peakStartCurrent = map (peakStartY, 60, 540, ymax, ymin);
   peakEndVoltage = map (peakEndX, 220, 860, xmin, xmax);
   peakEndCurrent = map (peakEndY, 60, 540, ymax, ymin);
   println ("peakStartX: " + peakStartX + " peakStartY: " + peakStartY + " peakEndX: " + peakEndX + "
peakEndY: " + peakEndY);
   println ("peakStartVoltage: " + peakStartVoltage + " peakStartCurrent: " + peakStartCurrent + "
peakEndVoltage: " + peakEndVoltage + " peakEndCurrent: " + peakEndCurrent);
   for (int i = 0; tester && (i < mainTable.getRowCount()); i++) {
    TableRow testRow = mainTable.getRow (i);
    if ((testRow.getFloat ("voltage")) > peakStartVoltage)
    {
     peakStartRow = i;
     tester = false;
     println ("Starting Row: " + i);
    }
   }
   tester = true;
   for (int i = peakStartRow; tester && (i < mainTable.getRowCount()); i++) {
    TableRow testRow = mainTable.getRow (i);
    if (testRow.getFloat ("voltage") > peakEndVoltage) {
     peakEndRow = i;
     tester = false;
    }
   }
   tester = true;
   for (int i = peakStartRow; i < peakEndRow && (i < mainTable.getRowCount()); i++) {
    TableRow testRow = mainTable.getRow (i);
    if (smooth)
    {
     testMaxCurrent = testRow.getFloat ("sg");
    }
    else
    {
     testMaxCurrent = testRow.getFloat ("current");
    }
    if (testMaxCurrent > peakCurrent)
    {
       peakCurrent = testMaxCurrent;
       peakVoltage = testRow.getFloat ("voltage");
    }
   }
   peakTrue = true;

  }
  int scanMax (float maxStart, float maxEnd)
  {
   int resultMax = 0;
   tester = true;
   for (int i = 0; tester && (i < mainTable.getRowCount()); i++)
   {
    TableRow testRow = scans[scanCounter].mainTable.getRow (i);
    if ((testRow.getFloat ("voltage")) > maxStart)
    {
     peakStartRow = i;
     tester = false;
    }
   }
   tester = true;
   for (int i = peakStartRow; tester && (i < mainTable.getRowCount()); i++)
   {
    TableRow testRow = mainTable.getRow (i);
    if (testRow.getFloat ("voltage") > maxEnd)
    {
     peakEndRow = i;
     tester = false;
```

```
// liteSTAT GUI code for Prototype Model B page 16 of 35

     }
    }
    tester = true;
    resultMax = 0;
    for (int i = peakStartRow; i < peakEndRow; i++)
    {
     TableRow testRow = mainTable.getRow (i);
     if (smooth)
     {
      testMaxCurrent = testRow.getFloat ("sg");
     }
     else
     {
      testMaxCurrent = testRow.getFloat ("current");
     }
     if (testMaxCurrent > resultMax)
     {
        resultMax = int(testMaxCurrent);
     }
    }
    return (resultMax);
  }
}

void makeNewScan (int colourCounter_) {
    if (colourCounter_ == 0) {
      Scan tempScan = new Scan (color (0, 0, 255));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 1) {
      Scan tempScan = new Scan (color (255, 0, 0));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 2) {
      Scan tempScan = new Scan (color (0, 255, 0));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 3) {
      Scan tempScan = new Scan (color (0, 0, 0));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 4) {
      Scan tempScan = new Scan (color (255, 0, 255));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 5) {
      Scan tempScan = new Scan (color (0, 255, 255));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 6) {
      Scan tempScan = new Scan (color (175, 175, 175));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 7) {
      Scan tempScan = new Scan (color (142, 107, 35));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 8) {
      Scan tempScan = new Scan (color (255, 128, 0));
      scans = (Scan[]) append (scans, tempScan);
    }
    if (colourCounter_ == 9) {
      Scan tempScan = new Scan (color (128, 0, 255));
      scans = (Scan[]) append (scans, tempScan);
    }
}
void smoothData ()
{
  /* int[][] coeff = {  {17, 12, -3, 0, 0, 0, 0, 0, 0, 0},
```

```
// liteSTAT GUI code for Prototype Model B page 17 of 35

                {7, 6, 3, -2, 0, 0, 0, 0, 0, 0},
                {59, 54, 39, 14, -21, 0, 0, 0, 0, 0},
                {89, 84, 69, 44, 9, -36, 0, 0, 0, 0},
                {25, 24, 21, 16, 9, 0, -11, 0, 0, 0},
                {167, 162, 147, 122, 87, 42, -13, -78, 0, 0},
                {43, 42, 39, 34, 27, 18, 7, -6, -21, 0},
                {269, 264, 249, 224, 189, 144, 89, 24, -51, -136}  };
 int[] h = {35, 21, 231, 429, 143, 1105, 323, 2261}; */
 smoothLevel = int (cp5.getValue("  "));
 smoothPoints = (2 * smoothLevel) + 3;
 rowFactor = (-1 * smoothLevel) - 1;
 for (int i = lowestActiveScan; i < scans.length; i++)
 {
  if (noDataYet || (scans[i].mainTable.getRowCount() < (smoothPoints + 1)))
  {

  }
  else
  {

   switch (smoothLevel)
   {
    case 1:
    {
     for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
     {
      TableRow rowl3 = scans[i].mainTable.getRow (j-3);
      TableRow rowl2 = scans[i].mainTable.getRow (j-2);
      TableRow rowl1 = scans[i].mainTable.getRow (j-1);
      TableRow row = scans[i].mainTable.getRow (j);
      TableRow rowh1 = scans[i].mainTable.getRow (j+1);
      TableRow rowh2 = scans[i].mainTable.getRow (j+2);
      TableRow rowh3 = scans[i].mainTable.getRow (j+3);
      tempsg = ((-2.0 * rowl3.getFloat("current") + 3.0 * rowl2.getFloat("current") + 6.0 *
rowl1.getFloat("current") + 7.0 * row.getFloat("current")
       + 6.0 * rowh1.getFloat("current") + 3.0 * rowh2.getFloat("current") -2.0 *
rowh3.getFloat("current")  )/21.0);
      row.setFloat ("sg", tempsg);
     }
     break;
    }
    case 2:
    {
     for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
     {
      TableRow rowl4 = scans[i].mainTable.getRow (j-4);
      TableRow rowl3 = scans[i].mainTable.getRow (j-3);
      TableRow rowl2 = scans[i].mainTable.getRow (j-2);
      TableRow rowl1 = scans[i].mainTable.getRow (j-1);
      TableRow row = scans[i].mainTable.getRow (j);
      TableRow rowh1 = scans[i].mainTable.getRow (j+1);
      TableRow rowh2 = scans[i].mainTable.getRow (j+2);
      TableRow rowh3 = scans[i].mainTable.getRow (j+3);
      TableRow rowh4 = scans[i].mainTable.getRow (j+4);
      tempsg = ((-21.0 * rowl4.getFloat("current") + 14.0 * rowl3.getFloat("current") + 39.0 *
rowl2.getFloat("current") + 54.0 * rowl1.getFloat("current") + 59 * row.getFloat("current")
       + 54.0 * rowh1.getFloat("current") + 39.0 * rowh2.getFloat("current") + 14.0 *
rowh3.getFloat("current") - 21.0 * rowh4.getFloat("current") )/231.0);
      row.setFloat ("sg", tempsg);
     }
     break;
    }
    case 3:
    {
     for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
     {
      TableRow rowl5 = scans[i].mainTable.getRow (j-5);
      TableRow rowl4 = scans[i].mainTable.getRow (j-4);
      TableRow rowl3 = scans[i].mainTable.getRow (j-3);
```

```
          TableRow rowl2 = scans[i].mainTable.getRow (j-2);
          TableRow rowl1 = scans[i].mainTable.getRow (j-1);
          TableRow row = scans[i].mainTable.getRow (j);
          TableRow rowh1 = scans[i].mainTable.getRow (j+1);
          TableRow rowh2 = scans[i].mainTable.getRow (j+2);
          TableRow rowh3 = scans[i].mainTable.getRow (j+3);
          TableRow rowh4 = scans[i].mainTable.getRow (j+4);
          TableRow rowh5 = scans[i].mainTable.getRow (j+5);
          tempsg = ((-36.0 * rowl5.getFloat("current") + 9.0 * rowl4.getFloat("current") + 44.0 *
rowl3.getFloat("current") + 69.0 * rowl2.getFloat("current") + 84.0 * rowl1.getFloat("current") + 89 *
row.getFloat("current")
          + 84.0 * rowh1.getFloat("current") + 69.0 * rowh2.getFloat("current") + 44.0 *
rowh3.getFloat("current") + 9.0 * rowh4.getFloat("current") - 36.0 * rowh5.getFloat("current"))/429.0);
          row.setFloat ("sg", tempsg);
         }
         break;
        }
        case 4:
        {
         for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
         {
          TableRow rowl6 = scans[i].mainTable.getRow (j-6);
          TableRow rowl4 = scans[i].mainTable.getRow (j-4);
          TableRow rowl3 = scans[i].mainTable.getRow (j-3);
          TableRow rowl2 = scans[i].mainTable.getRow (j-2);
          TableRow rowl1 = scans[i].mainTable.getRow (j-1);
          TableRow row = scans[i].mainTable.getRow (j);
          TableRow rowh1 = scans[i].mainTable.getRow (j+1);
          TableRow rowh2 = scans[i].mainTable.getRow (j+2);
          TableRow rowh3 = scans[i].mainTable.getRow (j+3);
          TableRow rowh4 = scans[i].mainTable.getRow (j+4);
          TableRow rowh6 = scans[i].mainTable.getRow (j+6);
          tempsg = ((-11.0 * rowl6.getFloat("current") + 9.0 * rowl4.getFloat("current") + 16.0 *
rowl3.getFloat("current") + 21.0 * rowl2.getFloat("current") + 24.0 * rowl1.getFloat("current") + 25 *
row.getFloat("current")
          + 24.0 * rowh1.getFloat("current") + 21.0 * rowh2.getFloat("current") + 16.0 *
rowh3.getFloat("current") + 9.0 * rowh4.getFloat("current") - 11.0 * rowh6.getFloat("current"))/143.0);
          row.setFloat ("sg", tempsg);
         }
         break;
        }
        case 5:
        {
         for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
         {
          TableRow rowl7 = scans[i].mainTable.getRow (j-7);
          TableRow rowl6 = scans[i].mainTable.getRow (j-6);
          TableRow rowl5 = scans[i].mainTable.getRow (j-5);
          TableRow rowl4 = scans[i].mainTable.getRow (j-4);
          TableRow rowl3 = scans[i].mainTable.getRow (j-3);
          TableRow rowl2 = scans[i].mainTable.getRow (j-2);
          TableRow rowl1 = scans[i].mainTable.getRow (j-1);
          TableRow row = scans[i].mainTable.getRow (j);
          TableRow rowh1 = scans[i].mainTable.getRow (j+1);
          TableRow rowh2 = scans[i].mainTable.getRow (j+2);
          TableRow rowh3 = scans[i].mainTable.getRow (j+3);
          TableRow rowh4 = scans[i].mainTable.getRow (j+4);
          TableRow rowh5 = scans[i].mainTable.getRow (j+5);
          TableRow rowh6 = scans[i].mainTable.getRow (j+6);
          TableRow rowh7 = scans[i].mainTable.getRow (j+7);
          tempsg = ((-78.0*rowl7.getFloat("current")-
13.0*rowl6.getFloat("current")+42.0*rowl5.getFloat("current")+87.0*rowl4.getFloat("current")+122.0*rowl
3.getFloat("current")+147.0*rowl2.getFloat("current")+162.0*rowl1.getFloat("current")+167*row.getFloat(
"current")

+162.0*rowh1.getFloat("current")+147.0*rowh2.getFloat("current")+122.0*rowh3.getFloat("current")+87.
0*rowh4.getFloat("current")+42.0*rowh5.getFloat("current")-13.0*rowh6.getFloat("current")-
78.0*rowh7.getFloat("current"))/1105.0);
          row.setFloat ("sg", tempsg);
```

```
        }
        break;
       }
       case 6:
       {
        for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
        {
         TableRow rowl8 = scans[i].mainTable.getRow (j-8);
         TableRow rowl7 = scans[i].mainTable.getRow (j-7);
         TableRow rowl6 = scans[i].mainTable.getRow (j-6);
         TableRow rowl5 = scans[i].mainTable.getRow (j-5);
         TableRow rowl4 = scans[i].mainTable.getRow (j-4);
         TableRow rowl3 = scans[i].mainTable.getRow (j-3);
         TableRow rowl2 = scans[i].mainTable.getRow (j-2);
         TableRow rowl1 = scans[i].mainTable.getRow (j-1);
         TableRow row = scans[i].mainTable.getRow (j);
         TableRow rowh1 = scans[i].mainTable.getRow (j+1);
         TableRow rowh2 = scans[i].mainTable.getRow (j+2);
         TableRow rowh3 = scans[i].mainTable.getRow (j+3);
         TableRow rowh4 = scans[i].mainTable.getRow (j+4);
         TableRow rowh5 = scans[i].mainTable.getRow (j+5);
         TableRow rowh6 = scans[i].mainTable.getRow (j+6);
         TableRow rowh7 = scans[i].mainTable.getRow (j+7);
         TableRow rowh8 = scans[i].mainTable.getRow (j+8);
         tempsg = ((-21.0*rowl8.getFloat("current")-
6.0*rowl7.getFloat("current")+7.0*rowl6.getFloat("current")+18.0*rowl5.getFloat("current")+27.0*rowl4.g
etFloat("current")+34.0*rowl3.getFloat("current")+39.0*rowl2.getFloat("current")+42.0*rowl1.getFloat("cu
rrent")+43*row.getFloat("current")

+42.0*rowh1.getFloat("current")+39.0*rowh2.getFloat("current")+34.0*rowh3.getFloat("current")+27.0*r
owh4.getFloat("current")+18.0*rowh5.getFloat("current")+7.0*rowh6.getFloat("current")-
6.0*rowh7.getFloat("current")-21.0*rowh8.getFloat("current"))/323.0);
         row.setFloat ("sg", tempsg);
        }
        break;
       }
       case 7:
       {
        for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
        {
         TableRow rowl9 = scans[i].mainTable.getRow (j-9);
         TableRow rowl8 = scans[i].mainTable.getRow (j-8);
         TableRow rowl7 = scans[i].mainTable.getRow (j-7);
         TableRow rowl6 = scans[i].mainTable.getRow (j-6);
         TableRow rowl5 = scans[i].mainTable.getRow (j-5);
         TableRow rowl4 = scans[i].mainTable.getRow (j-4);
         TableRow rowl3 = scans[i].mainTable.getRow (j-3);
         TableRow rowl2 = scans[i].mainTable.getRow (j-2);
         TableRow rowl1 = scans[i].mainTable.getRow (j-1);
         TableRow row = scans[i].mainTable.getRow (j);
         TableRow rowh1 = scans[i].mainTable.getRow (j+1);
         TableRow rowh2 = scans[i].mainTable.getRow (j+2);
         TableRow rowh3 = scans[i].mainTable.getRow (j+3);
         TableRow rowh4 = scans[i].mainTable.getRow (j+4);
         TableRow rowh5 = scans[i].mainTable.getRow (j+5);
         TableRow rowh6 = scans[i].mainTable.getRow (j+6);
         TableRow rowh7 = scans[i].mainTable.getRow (j+7);
         TableRow rowh8 = scans[i].mainTable.getRow (j+8);
         TableRow rowh9 = scans[i].mainTable.getRow (j+9);
         tempsg = ((-136.0*rowl9.getFloat("current")-
51.0*rowl8.getFloat("current")+24.0*rowl7.getFloat("current")+89.0*rowl6.getFloat("current")+144.0*rowl
5.getFloat("current")+189.0*rowl4.getFloat("current")+224.0*rowl3.getFloat("current")+249.0*rowl2.getFl
oat("current")+264.0*rowl1.getFloat("current")+269*row.getFloat("current")

+264.0*rowh1.getFloat("current")+249.0*rowh2.getFloat("current")+224.0*rowh3.getFloat("current")+189
.0*rowh4.getFloat("current")+144.0*rowh5.getFloat("current")+89.0*rowh6.getFloat("current")+24.0*row
h7.getFloat("current")-51.0*rowh8.getFloat("current")-136.0*rowh9.getFloat("current"))/2261.0);
         row.setFloat ("sg", tempsg);
        }
```

```
// liteSTAT GUI code for Prototype Model B page 20 of 35

       break;
      }
      case 8:
      {
       for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
       {
        TableRow rowl10 = scans[i].mainTable.getRow (j-10);
        TableRow rowl9 = scans[i].mainTable.getRow (j-9);
        TableRow rowl8 = scans[i].mainTable.getRow (j-8);
        TableRow rowl7 = scans[i].mainTable.getRow (j-7);
        TableRow rowl6 = scans[i].mainTable.getRow (j-6);
        TableRow rowl5 = scans[i].mainTable.getRow (j-5);
        TableRow rowl4 = scans[i].mainTable.getRow (j-4);
        TableRow rowl3 = scans[i].mainTable.getRow (j-3);
        TableRow rowl2 = scans[i].mainTable.getRow (j-2);
        TableRow rowl1 = scans[i].mainTable.getRow (j-1);
        TableRow row = scans[i].mainTable.getRow (j);
        TableRow rowh1 = scans[i].mainTable.getRow (j+1);
        TableRow rowh2 = scans[i].mainTable.getRow (j+2);
        TableRow rowh3 = scans[i].mainTable.getRow (j+3);
        TableRow rowh4 = scans[i].mainTable.getRow (j+4);
        TableRow rowh5 = scans[i].mainTable.getRow (j+5);
        TableRow rowh6 = scans[i].mainTable.getRow (j+6);
        TableRow rowh7 = scans[i].mainTable.getRow (j+7);
        TableRow rowh8 = scans[i].mainTable.getRow (j+8);
        TableRow rowh9 = scans[i].mainTable.getRow (j+9);
        TableRow rowh10 = scans[i].mainTable.getRow (j+10);
        tempsg = ((-171.0*rowl10.getFloat("current")-
76.0*rowl9.getFloat("current")+9.0*rowl8.getFloat("current")+84.0*rowl7.getFloat("current")+149.0*rowl6
.getFloat("current")+204.0*rowl5.getFloat("current")+249.0*rowl4.getFloat("current")+284.0*rowl3.getFlo
at("current")+309.0*rowl2.getFloat("current")+324.0*rowl1.getFloat("current")+329*row.getFloat("current
")

+324.0*rowh1.getFloat("current")+309.0*rowh2.getFloat("current")+284.0*rowh3.getFloat("current")+249
.0*rowh4.getFloat("current")+204.0*rowh5.getFloat("current")+149.0*rowh6.getFloat("current")+84.0*ro
wh7.getFloat("current")+9.0*rowh8.getFloat("current")-76.0*rowh9.getFloat("current")-
171.0*rowh10.getFloat("current"))/3059.0);
        row.setFloat ("sg", tempsg);
       }
       break;
      }
      case 9:
      {
       for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
       {
        TableRow rowl11 = scans[i].mainTable.getRow (j-11);
        TableRow rowl10 = scans[i].mainTable.getRow (j-10);
        TableRow rowl9 = scans[i].mainTable.getRow (j-9);
        TableRow rowl8 = scans[i].mainTable.getRow (j-8);
        TableRow rowl7 = scans[i].mainTable.getRow (j-7);
        TableRow rowl6 = scans[i].mainTable.getRow (j-6);
        TableRow rowl5 = scans[i].mainTable.getRow (j-5);
        TableRow rowl4 = scans[i].mainTable.getRow (j-4);
        TableRow rowl3 = scans[i].mainTable.getRow (j-3);
        TableRow rowl2 = scans[i].mainTable.getRow (j-2);
        TableRow rowl1 = scans[i].mainTable.getRow (j-1);
        TableRow row = scans[i].mainTable.getRow (j);
        TableRow rowh1 = scans[i].mainTable.getRow (j+1);
        TableRow rowh2 = scans[i].mainTable.getRow (j+2);
        TableRow rowh3 = scans[i].mainTable.getRow (j+3);
        TableRow rowh4 = scans[i].mainTable.getRow (j+4);
        TableRow rowh5 = scans[i].mainTable.getRow (j+5);
        TableRow rowh6 = scans[i].mainTable.getRow (j+6);
        TableRow rowh7 = scans[i].mainTable.getRow (j+7);
        TableRow rowh8 = scans[i].mainTable.getRow (j+8);
        TableRow rowh9 = scans[i].mainTable.getRow (j+9);
        TableRow rowh10 = scans[i].mainTable.getRow (j+10);
        TableRow rowh11 = scans[i].mainTable.getRow (j+11);
```

```
          tempsg = ((-42.0*rowl11.getFloat("current")-21.0*rowl10.getFloat("current")-
2.0*rowl9.getFloat("current")+15.0*rowl8.getFloat("current")+30.0*rowl7.getFloat("current")+43.0*rowl6.
get
// liteSTAT GUI code for Prototype Model B page 21 of 35

Float("current")+54.0*rowl5.getFloat("current")+63.0*rowl4.getFloat("current")+70.0*rowl3.getFloat("curr
ent")+75.0*rowl2.getFloat("current")+78.0*rowl1.getFloat("current")+79*row.getFloat("current")

+78.0*rowh1.getFloat("current")+75.0*rowh2.getFloat("current")+70.0*rowh3.getFloat("current")+63.0*r
owh4.getFloat("current")+54.0*rowh5.getFloat("current")+43.0*rowh6.getFloat("current")+30.0*rowh7.get
Float("current")+15.0*rowh8.getFloat("current")-2.0*rowh9.getFloat("current")-
21.0*rowh10.getFloat("current")-42.0*rowh11.getFloat("current"))/805.0);
          row.setFloat ("sg", tempsg);
         }
         break;
       }
       case 10:
       {
         for (int j = smoothPoints; j < scans[i].mainTable.getRowCount () - smoothPoints; j++)
         {
           TableRow rowl12 = scans[i].mainTable.getRow (j-12);
           TableRow rowl11 = scans[i].mainTable.getRow (j-11);
           TableRow rowl10 = scans[i].mainTable.getRow (j-10);
           TableRow rowl9 = scans[i].mainTable.getRow (j-9);
           TableRow rowl8 = scans[i].mainTable.getRow (j-8);
           TableRow rowl7 = scans[i].mainTable.getRow (j-7);
           TableRow rowl6 = scans[i].mainTable.getRow (j-6);
           TableRow rowl5 = scans[i].mainTable.getRow (j-5);
           TableRow rowl4 = scans[i].mainTable.getRow (j-4);
           TableRow rowl3 = scans[i].mainTable.getRow (j-3);
           TableRow rowl2 = scans[i].mainTable.getRow (j-2);
           TableRow rowl1 = scans[i].mainTable.getRow (j-1);
           TableRow row = scans[i].mainTable.getRow (j);
           TableRow rowh1 = scans[i].mainTable.getRow (j+1);
           TableRow rowh2 = scans[i].mainTable.getRow (j+2);
           TableRow rowh3 = scans[i].mainTable.getRow (j+3);
           TableRow rowh4 = scans[i].mainTable.getRow (j+4);
           TableRow rowh5 = scans[i].mainTable.getRow (j+5);
           TableRow rowh6 = scans[i].mainTable.getRow (j+6);
           TableRow rowh7 = scans[i].mainTable.getRow (j+7);
           TableRow rowh8 = scans[i].mainTable.getRow (j+8);
           TableRow rowh9 = scans[i].mainTable.getRow (j+9);
           TableRow rowh10 = scans[i].mainTable.getRow (j+10);
           TableRow rowh11 = scans[i].mainTable.getRow (j+11);
           TableRow rowh12 = scans[i].mainTable.getRow (j+12);
           tempsg = ((-253.0*rowl12.getFloat("current")-138.0*rowl11.getFloat("current")-
33.0*rowl10.getFloat("current")+62.0*rowl9.getFloat("current")+147.0*rowl8.getFloat("current")+222.0*r
owl7.getFloat("current")+287.0*rowl6.getFloat("current")+343.0*rowl5.getFloat("current")+387.0*rowl4.g
etFloat("current")+422.0*rowl3.getFloat("current")+447.0*rowl2.getFloat("current")+462.0*rowl1.getFloat(
"current")+467*row.getFloat("current")

+462.0*rowh1.getFloat("current")+447.0*rowh2.getFloat("current")+422.0*rowh3.getFloat("current")+387
.0*rowh4.getFloat("current")+343.0*rowh5.getFloat("current")+287.0*rowh6.getFloat("current")+222.0*ro
wh7.getFloat("current")+147.0*rowh8.getFloat("current")+62.0*rowh9.getFloat("current")-
33.0*rowh10.getFloat("current")-138.0*rowh11.getFloat("current")-
253.0*rowh12.getFloat("current"))/5175.0);
           row.setFloat ("sg", tempsg);
         }
         break;
       }
     }


     for (int j = 1; j < smoothPoints; j++)
     {
       TableRow row = scans[i].mainTable.getRow (j);
       row.setFloat ("sg", (row.getFloat ("current")));
     }
     for (int j = scans[i].mainTable.getRowCount () - smoothPoints; j < scans[i].mainTable.getRowCount ();
j++)
     {
```

```
      TableRow row = scans[i].mainTable.getRow (j);
      row.setFloat ("sg", (row.getFloat ("current")));
     }
// liteSTAT GUI code for Prototype Model B page 22 of 35

   }
  }
  for (int i = 0; i < scans.length; i++)
   {
    for (int j = 0; j < scans[i].mainTable.getRowCount (); j++)
    {
     TableRow row = scans[i].mainTable.getRow (j);
     row.setFloat ("ySg", map (row.getFloat ("sg"), ymin, ymax, 540.0, 60.0));
    }
   }
}
void actionPerformed(GUIEvent e)
{
   if (e.getSource() == maxB)
   {
      findMax();
  }
  if (e.getSource() == screenGrabCB)
  {
   if (screenGrabCB.isSelected()) {
    capture = true;
   }
   else
   {
    capture = false;
   }
  }
  if (e.getSource() == holdCB)
  {
   if (holdCB.isSelected()) {
    holdGain = true;
   }
   else
   {
    holdGain = false;
   }
  }
  if (e.getSource() == scanNumberB)
  {
   chosenScan = int (scanNumberT.getValue());
   singleScanOnly = true;
   for (int z = 0; z < 100; z++)
   {
    for (int j = 0; j < scans[(scans.length - 1)].mainTable.getRowCount (); j++)
    {
     TableRow row = scans[(scans.length - 1)].mainTable.getRow(j);
     if (row.getInt("Scan Number") != chosenScan) scans[(scans.length - 1)].mainTable.removeRow(j);
    }
   }
  }
  if (e.getSource() == cvB) {
   if (notConnected)
   {
    removeAllTandL ();
    draw ();
   }
   else
   {
   removeAllTandL ();
   setupCV ();
   }
  }
  /*if (e.getSource() == testB) {
   testProcedure();
  }*/
  else if (e.getSource() == lsvB) {
```

```
  if (notConnected)
  {
   removeAllTandL ();
// liteSTAT GUI code for Prototype Model B page 23 of 35

   draw ();
  }
  else
  {
  removeAllTandL ();
  setupLSV ();
  }
 }
 else if (e.getSource() == exportScanB) {
  selectOutput ("", "exportScan");
 }
 else if (e.getSource() == swvB) {
  if (notConnected)
  {
   removeAllTandL ();
   draw ();
  }
  else
  {
  removeAllTandL ();
  setupSWV ();
  }
 }
 else if (e.getSource() == runB)
 {
  println ("runB clicked");
  sweepCounter = 0;
  dontSmooth = false;
  noDataYet = false;
  if (holdGain)
  {
   HOLD_S = "2";
  }
  else
  {
   HOLD_S = "1";
  }
  println ("After choosing hold");
  if (notConnected)
  {
   println ("went back to draw");
   draw ();
  }
  else
  {
   println ("After checking connection");
   pointCounter = 0;
   scanCounter++;
   colourCounter = scanCounter % 10;
   makeNewScan (colourCounter);
   showData = true;
   potI = int (cp5.getValue(" "));
   potI += 999;
   potS = str (potI);
   println ("Inside method setup");
   if (cv || lsv)
   {
    setRate ();
    println ("Before setCurrent ()");
    setCurrentFactors ();
    println ("After setCurrent ()");
    DEPOSITION_V_S = "5096";
    DEPOSITION_T_S = "1001";
    EQUILIBRIUM_T_S = "1001";
    PULSE_S = "1001";
   }
```

```
    startS = startT.getValue();
    stopS = endT.getValue ();
// liteSTAT GUI code for Prototype Model B page 24 of 35

    startF = float (startS);
    stopF = float (stopS);
    startF = startF * 4096.0 / 5.0;
    startF += 4096.0;
    stopF = stopF * 4096.0 / 5.0;
    stopF += 4096.0;
    startS = str (int (startF));
    stopS = str (int (stopF));
    if (cv)
    {
     methodS = "1";
     cyclesS = cyclesT.getValue ();
     cyclesI = int (cyclesS);
     cyclesI += 100;
     cyclesS = str (cyclesI);
     toSendS = methodS + complianceS + potS + startS + stopS + cyclesS + WAITER_S + V_NUM_S +
I_NUM_S + STEPPER_S + DEPOSITION_V_S + DEPOSITION_T_S + EQUILIBRIUM_T_S
      + PULSE_S + HOLD_S + ALPHA_S + SHIFT_S + "!";
     port.write (toSendS);
     println (toSendS);
    }
    else if (lsv)
    {
     methodS = "2";
     cyclesS = "101";
     toSendS = methodS + complianceS + potS + startS + stopS + cyclesS + WAITER_S + V_NUM_S +
I_NUM_S + STEPPER_S + DEPOSITION_V_S + DEPOSITION_T_S + EQUILIBRIUM_T_S
     + PULSE_S  + HOLD_S + ALPHA_S + SHIFT_S + "!";
     port.write (toSendS);
     println (toSendS);
    }
    else if (swv)
    {
     methodS = "3";
     cyclesS = "101";
     WAITER_I = int (waitT.getValue());
     WAITER_I = 500 / WAITER_I;
     V_NUM_S = "9999";
     I_NUM_S = "9999";
     STEPPER_I = int (stepT.getValue());
     if (STEPPER_I < 2)
     {
      STEPPER_I = 2;
     }
     DEPOSITION_V_F = float (depPotT.getValue());
     DEPOSITION_T_I = int (depTimeT.getValue());
     EQUILIBRIUM_T_I = int (equiTimeT.getValue());
     PULSE_I = int (pulseT.getValue());
     if (PULSE_I < 2)
     {
      PULSE_I = 2;
     }
     STEPPER_F = STEPPER_I / 1.22;
     STEPPER_I = int (STEPPER_F);
     STEPPER_F = STEPPER_I * 1.22;
     PULSE_F = PULSE_I / 1.22;
     PULSE_I = int (PULSE_F);
     PULSE_F = PULSE_I * 1.22;
     WAITER_I += 1000;
     STEPPER_I += 1000;
     DEPOSITION_V_F = DEPOSITION_V_F * 4096.0 / 5.0;
     DEPOSITION_V_F += 4096.0;
     DEPOSITION_T_I += 1000;
     EQUILIBRIUM_T_I += 1000;
     PULSE_I += 1000;
     WAITER_S = str (WAITER_I);
```

```
      STEPPER_S = str (STEPPER_I);
      DEPOSITION_V_S = str (int (DEPOSITION_V_F));
      DEPOSITION_T_S = str (DEPOSITION_T_I);
// liteSTAT GUI code for Prototype Model B page 25 of 35

      EQUILIBRIUM_T_S = str (EQUILIBRIUM_T_I);
      PULSE_S = str (PULSE_I);
      toSendS = methodS + complianceS + potS + startS + stopS + cyclesS + WAITER_S + V_NUM_S +
I_NUM_S + STEPPER_S + DEPOSITION_V_S + DEPOSITION_T_S + EQUILIBRIUM_T_S
        + PULSE_S + HOLD_S + "1073!";
      port.write (toSendS);
      println (toSendS);
      //pulseT.setValue (str(PULSE_F));
      //stepT.setValue (str(STEPPER_F));
     }
    }
   }

   else if (e.getSource() == clearB) {
    lowestActiveScan = scans.length;
    noDataYet = true;
    for (int i = 0; i < scans.length; i++) {
     scans[i].mainTable.clearRows();
     outRange = false;
     smooth = false;
     dontSmooth = true;
     sgB.setLabel ("Smooth Data");

    }
    peakDetection = false;
    showData = false;
    scanning = false;
    scans[scanCounter].peakTrue = false;
    makeMins ();
    showPeaks = false;
    area = 0.0;
     }
   else if (e.getSource() == sgB) {
    smooth = !smooth;
    if (smooth) {
     sgB.setLabel ("Unsmooth");
    }
    else {
     sgB.setLabel ("Smooth Data");
    }
   }

    else if (e.getSource() == voltB)
    {
     compliance = !compliance;
     if (compliance)
     {
      voltB.setLabel ("Compliance: High");
      complianceS = "2";
     }
     else
     {
      voltB.setLabel ("Compliance: Low");
      complianceS = "1";
     }
    }
    if (e.getSource () == loadMethodB)
    {
     smooth = false;
     sgB.setLabel ("Smooth Data");
     scanCounter++;
     if (scanCounter == 0) dontSmooth = true;
     showData = true;
     scanning = true;
     colourCounter = scanCounter % 10;
     makeNewScan (colourCounter);
```

```
    selectInput("", "openSelectedMethod");
    lowestActiveScan = scanCounter+1;
   }
// liteSTAT GUI code for Prototype Model B page 26 of 35

   else if (e.getSource () == loadDataB) {
    smooth = false;
    sgB.setLabel ("Smooth Data");
    scanCounter++;
    dontSmooth = false;
    showData = true;
    scanning = true;
    colourCounter = scanCounter % 10;
    makeNewScan (colourCounter);
    selectInput("", "openSelectedData");
    noDataYet = false;
    }
   else if (e.getSource () == peaksB) {
    area = 0.0;
    showPeaks = true;
    peakDetection = true;
    baseCounter = 0;
   }
   else if (e.getSource () == portB) {
    portChoice = true;
   }

}

void setupCV () {
   cv = true;
   lsv = false;
   swv = false;
   startT = new IFTextField ("", 90, 118, 70);
   endT = new IFTextField ("", 90, 143, 70);
   rateT = new IFTextField ("", 90, 168, 70);
   cyclesT = new IFTextField ("", 90, 193, 70);
   c.add(startT);
   c.add(endT);
   c.add (rateT);
   c.add (cyclesT);
   startT.addActionListener(this);
   endT.addActionListener (this);
   rateT.addActionListener (this);
   cyclesT.addActionListener (this);
   startL = new IFLabel ("", 20, 120);
   endL = new IFLabel ("", 20, 145);
   rateL = new IFLabel ("", 20, 170);
   cyclesL = new IFLabel ("", 20, 195);
   c.add(startL);
   c.add(endL);
   c.add (rateL);
   c.add (cyclesL);
   c.add (voidL);

   startL.setLabel ("Start (V):");
   endL.setLabel ("End (V):");
   rateL.setLabel ("Rate (mV/s):");
   cyclesL.setLabel ("Cycles:");
}

void setupLSV () {
  cv = false;
  lsv = true;
  swv = false;
  startT = new IFTextField ("", 90, 118, 70);
  endT = new IFTextField ("", 90, 143, 70);
  rateT = new IFTextField ("", 90, 168, 70);
  c.add(startT);
  c.add(endT);
  c.add (rateT);
```

```
  startT.addActionListener(this);
  endT.addActionListener (this);
// liteSTAT GUI code for Prototype Model B page 27 of 35

rateT.addActionListener (this);
  startL = new IFLabel ("", 20, 120);
  endL = new IFLabel ("", 20, 145);
  rateL = new IFLabel ("", 20, 170);

  c.add(startL);
  c.add(endL);
  c.add (rateL);
  c.add (voidL);
  startL.setLabel ("Start (V):");
  endL.setLabel ("End (V):");
  rateL.setLabel ("Rate (mV/s)");

}

void setupSWV () {
    cv = false;
    lsv = false;
    swv = true;
    startT = new IFTextField ("", 90, 118, 70);
    endT = new IFTextField ("", 90, 143, 70);
    depPotT = new IFTextField ("", 90, 168, 70);
    depTimeT = new IFTextField ("", 90, 193, 70);
    equiTimeT = new IFTextField ("", 90, 218, 70);
    pulseT = new  IFTextField ("", 90, 243, 70);
    stepT = new IFTextField ("", 90, 268, 70);
    waitT = new IFTextField ("", 90, 293, 70);
    c.add (startT);
    c.add (endT);
    c.add (depPotT);
    c.add (depTimeT);
    c.add (equiTimeT);
    c.add (pulseT);
    c.add (stepT);
    c.add (waitT);
    startT.addActionListener (this);
    endT.addActionListener (this);
    depPotT.addActionListener (this);
    depTimeT.addActionListener (this);
    equiTimeT.addActionListener (this);
    pulseT.addActionListener (this);
    stepT.addActionListener (this);
    waitT.addActionListener (this);
    startL = new IFLabel ("Start (V):", 20, 120);
    endL = new IFLabel ("End (V):", 20, 145);
    depPotL = new IFLabel ("Dep (V):", 20, 170);
    depTimeL = new IFLabel ("Dep t (s):", 20, 195);
    equiTimeL = new IFLabel ("Equ t (s):", 20, 220);
    pulseL = new IFLabel ("Ampl (mV):", 20, 245);
    stepL = new IFLabel ("Step (mV):", 20, 270);
    waitL = new IFLabel ("Freq (Hz):", 20, 295);
    c.add (startL);
    c.add (endL);
    c.add (depPotL);
    c.add (depTimeL);
    c.add (equiTimeL);
    c.add (pulseL);
    c.add (stepL);
    c.add (waitL);
    c.add (voidL);
}
boolean rollOver (float tempX, float tempY) {
  if (tempX >= 180 && tempX <= 900 && tempY >= 20 && tempY <= 560) {
    return true;
  }
  else if (tempX >=915 && tempX <= 1002 && tempY >=500 && tempY <= 582 && mousePressed)
```

```
  {
    delay (250);
    link("http://bbc.co.uk/news");
// liteSTAT GUI code for Prototype Model B page 28 of 35

    return false;
  }
  else {
    return false;
  }
}
void finishedScan () {
  if (scanCounter >= 0) {
    buildFileTable ();
    int y = year ();
    int m = month ();
    int d = day ();
    int h = hour ();
    int mi = minute ();
    int s = second ();
    String filename = ("Data/" + y + "_" + m + "_" + d + "/");
    filename += saveT.getValue();
    filename += ("_" + h + "_" + mi + "_" + s);
    filename += (".csv");
    for (int i = 0; i < scans[scanCounter].mainTable.getRowCount (); i++) {
      TableRow main = scans[scanCounter].mainTable.getRow (i);
      TableRow data = scans[scanCounter].fileTable.addRow ();
      data.setFloat ("Voltage (V)", main.getFloat ("voltage"));
      data.setFloat ("Current (uA)", main.getFloat ("current"));
      data.setInt ("Scan Number", main.getInt ("Scan Number"));
    }
    storeMethodValues ();
    saveTable (scans[scanCounter].fileTable, filename);
    if (capture) {
      String filenameP = ("Screen Captures/" + y + "_" + m + "_" + d + "/");
      filenameP += saveT.getValue();
      filenameP += ("_" + h + "_" + mi + "_" + s);
      filenameP += (".png");
      save (filenameP);
    }
  }
}
void storeMethodValues () {
  if (cv) {
    TableRow row0 = scans[scanCounter].fileTable.getRow(0);
    row0.setString ("Parameter", "Method:");
    row0.setString ("Value", "Cyclic Voltammetry");
    TableRow row1 = scans[scanCounter].fileTable.getRow(1);
    row1.setString ("Parameter", "Start (V):");
    row1.setString ("Value", startT.getValue());
    TableRow row2 = scans[scanCounter].fileTable.getRow(2);
    row2.setString ("Parameter", "End (V):");
    row2.setString ("Value", endT.getValue());
    TableRow row3 = scans[scanCounter].fileTable.getRow(3);
    row3.setString ("Parameter", "Rate (mV/s):");
    row3.setString ("Value", rateT.getValue());
    TableRow row4 = scans[scanCounter].fileTable.getRow(4);
    row4.setString ("Parameter", "Cycles:");
    row4.setString ("Value", cyclesT.getValue());
  }
  if (lsv) {
    TableRow row0 = scans[scanCounter].fileTable.getRow(0);
    row0.setString ("Parameter", "Method:");
    row0.setString ("Value", "Linear Sweep Voltammetry");
    TableRow row1 = scans[scanCounter].fileTable.getRow(1);
    row1.setString ("Parameter", "Start (V):");
    row1.setString ("Value", startT.getValue());
    TableRow row2 = scans[scanCounter].fileTable.getRow(2);
    row2.setString ("Parameter", "End (V):");
    row2.setString ("Value", endT.getValue());
    TableRow row3 = scans[scanCounter].fileTable.getRow(3);
```

```
    row3.setString ("Parameter", "Rate (mV/s):");
    row3.setString ("Value", rateT.getValue());
   }
// liteSTAT GUI code for Prototype Model B page 29 of 35

   if (swv) {
    TableRow row0 = scans[scanCounter].fileTable.getRow(0);
    row0.setString ("Parameter", "Method:");
    row0.setString ("Value", "Square Wave Voltammetry");
    TableRow row1 = scans[scanCounter].fileTable.getRow(1);
    row1.setString ("Parameter", "Start (V):");
    row1.setString ("Value", startT.getValue());
    TableRow row2 = scans[scanCounter].fileTable.getRow(2);
    row2.setString ("Parameter", "End (V):");
    row2.setString ("Value", endT.getValue());
    TableRow row3 = scans[scanCounter].fileTable.getRow(3);
    row3.setString ("Parameter", "Deposition (V):");
    row3.setString ("Value", depPotT.getValue());
    TableRow row4 = scans[scanCounter].fileTable.getRow(4);
    row4.setString ("Parameter", "Deposition (s):");
    row4.setString ("Value", depTimeT.getValue());
    TableRow row5 = scans[scanCounter].fileTable.getRow(5);
    row5.setString ("Parameter", "Equilibrium (s):");
    row5.setString ("Value", equiTimeT.getValue());
    TableRow row6 = scans[scanCounter].fileTable.getRow(6);
    row6.setString ("Parameter", "Pulse Height:");
    row6.setString ("Value", pulseT.getValue());
    TableRow row7 = scans[scanCounter].fileTable.getRow(7);
    row7.setString ("Parameter", "Step Height:");
    row7.setString ("Value", stepT.getValue());
    TableRow row8 = scans[scanCounter].fileTable.getRow(8);
    row8.setString ("Parameter", "Frequency:");
    row8.setString ("Value", waitT.getValue());
   }
}

void exportScan (File selection) {
  if (selection == null) {
  }
  else {
    filename = (String) selection.getAbsolutePath();
    filename += ".csv";
    scans[scanCounter].fileTable = new Table ();
    buildFileTable ();
    for (int i = 0; i < scans[scanCounter].mainTable.getRowCount (); i++) {
      TableRow main = scans[scanCounter].mainTable.getRow (i);
      TableRow data = scans[scanCounter].fileTable.addRow ();
      data.setFloat ("Voltage (V)", main.getFloat ("voltage"));
      data.setFloat ("Current (uA)", main.getFloat ("current"));
    }
    storeMethodValues ();
    saveTable (scans[scanCounter].fileTable, filename);
  }
}

void openSelectedData (File selection) {
  if (selection == null) {
  }
  else {
    println("Well got to data load 1");
    scans[scanCounter].fileTable = loadTable (selection.getAbsolutePath(), "header");
    println("Well got to data load 2");
    for (int i = 0; i < scans[scanCounter].fileTable.getRowCount (); i++) {
       TableRow rowL = scans[scanCounter].fileTable.getRow (i);
       TableRow rowT = scans[scanCounter].mainTable.addRow ();
       rowT.setFloat ("voltage", rowL.getFloat("Voltage (V)"));
       rowT.setFloat ("current", rowL.getFloat("Current (uA)"));
       rowT.setInt ("Scan Number", rowL.getInt ("Scan Number"));
     }
     println("Well got to data load 3");
     //saveTable (scans[scanCounter].fileTable, filename);
```

```
    println("Well got to data load 4");
  }
  println("Well got to data load 5");
// liteSTAT GUI code for Prototype Model B page 30 of 35

}

void openSelectedMethod (File selection) {
  if (selection == null) {
  }
  else {
    notConnected = false;
    println("Well got to method load 1");
    scans[scanCounter].methodTable = loadTable (selection.getAbsolutePath(), "header");
    println("Well got to method load 2");
    if (scans[scanCounter].methodTable.getString(0, "Value").equals("Cyclic Voltammetry"))
    {
      cv = true;
      lsv = false;
      swv = false;
      println("Well got to method load 3");
      removeAllTandL ();
      setupCV ();
      startT.setValue (scans[scanCounter].methodTable.getString (1, "Value"));
      endT.setValue (scans[scanCounter].methodTable.getString (2, "Value"));
      rateT.setValue (scans[scanCounter].methodTable.getString (3, "Value"));
      cyclesT.setValue (scans[scanCounter].methodTable.getString (4, "Value"));
    }
    if (scans[scanCounter].methodTable.getString(0, "Value").equals("Linear Sweep Voltammetry")) {
      cv = false;
      lsv = true;
      swv = false;
      removeAllTandL ();
      setupLSV ();
      startT.setValue (scans[scanCounter].methodTable.getString (1, "Value"));
      endT.setValue (scans[scanCounter].methodTable.getString (2, "Value"));
      rateT.setValue (scans[scanCounter].methodTable.getString (3, "Value"));
    }
    if (scans[scanCounter].methodTable.getString(0, "Value").equals("Square Wave Voltammetry")) {
      cv = false;
      lsv = false;
      swv = true;
      removeAllTandL ();
      setupSWV ();
      startT.setValue (scans[scanCounter].methodTable.getString (1, "Value"));
      endT.setValue (scans[scanCounter].methodTable.getString (2, "Value"));
      depPotT.setValue (scans[scanCounter].methodTable.getString (3, "Value"));
      depTimeT.setValue (scans[scanCounter].methodTable.getString (4, "Value"));
      equiTimeT.setValue (scans[scanCounter].methodTable.getString (5, "Value"));
      pulseT.setValue (scans[scanCounter].methodTable.getString (6, "Value"));
      stepT.setValue (scans[scanCounter].methodTable.getString (7, "Value"));
      waitT.setValue (scans[scanCounter].methodTable.getString (8, "Value"));
    }
  }
}
void buildFileTable () {
    scans[scanCounter].fileTable = new Table ();
    scans[scanCounter].fileTable.addColumn ("Voltage (V)");
    scans[scanCounter].fileTable.addColumn ("Current (uA)");
    scans[scanCounter].fileTable.addColumn ("Scan Number");
    scans[scanCounter].fileTable.addColumn (" ");
    scans[scanCounter].fileTable.addColumn ("Parameter");
    scans[scanCounter].fileTable.addColumn ("Value");
    scans[scanCounter].fileTable.addColumn ("  ");
    scans[scanCounter].fileTable.addColumn ("Peak Voltage (V)");
    scans[scanCounter].fileTable.addColumn ("Peak Current (uA)");
}
void positivePeak ()
{
  if (noDataYet)
  {
```

```
 }
 else
 {
// liteSTAT GUI code for Prototype Model B page 31 of 35

   cursor (HAND);
   if (mousePressed && baseCounter < 2)
   {
    scans[scanCounter].peakStartX = mouseX;
    scans[scanCounter].peakStartY = mouseY;
    baseCounter++;
    stroke (0);
    strokeWeight (3);
    line (scans[scanCounter].peakStartX, scans[scanCounter].peakStartY, mouseX, mouseY);
    strokeWeight (1);
    }
    if (mousePressed && baseCounter == 2)
    {
     scans[scanCounter].peakEndX = mouseX;
     scans[scanCounter].peakEndY = mouseY;
    }
    if (baseCounter == 2)
    {

      scans[scanCounter].getPeaks ();
      stroke (0);
      strokeWeight (3);
      line (scans[scanCounter].peakStartX, scans[scanCounter].peakStartY, scans[scanCounter].peakEndX,
scans[scanCounter].peakEndY);
      if (scans[scanCounter].peakEndX - scans[scanCounter].peakStartX != 0)
      {
       scans[scanCounter].m = (scans[scanCounter].peakEndY - scans[scanCounter].peakStartY) /
(scans[scanCounter].peakEndX - scans[scanCounter].peakStartX);
       scans[scanCounter].c = scans[scanCounter].peakEndY - (scans[scanCounter].peakEndX *
scans[scanCounter].m);
       scans[scanCounter].peakX = map (scans[scanCounter].peakVoltage, xmin, xmax, 220.0, 860.0);
       scans[scanCounter].peakY = map (scans[scanCounter].peakCurrent, ymin, ymax, 540.0, 60.0);
       scans[scanCounter].baseY = scans[scanCounter].m * scans[scanCounter].peakX +
scans[scanCounter].c;
       line (scans[scanCounter].peakX, scans[scanCounter].peakY, scans[scanCounter].peakX,
scans[scanCounter].baseY);
       scans[scanCounter].peakHeight =scans[scanCounter].peakCurrent - map
(scans[scanCounter].baseY, 540.0, 60.0, ymin, ymax);
       text ("Peak Start Voltage: " + scans[scanCounter].peakStartVoltage, 700, 40);
       text ("Base End Voltage: " + scans[scanCounter].peakEndVoltage, 700, 55);
       text ("Start Row " + scans[scanCounter].peakStartRow, 700, 70);
       text ("End Row: " + scans[scanCounter].peakEndRow, 700, 85);
       text ("Total Rows: " + scans[scanCounter].mainTable.getRowCount(), 700, 100);
       text ("Intercept: " + scans[scanCounter].c, 700, 115);
       scans[scanCounter].slope = (scans[scanCounter].peakEndCurrent -
scans[scanCounter].peakStartCurrent) / (scans[scanCounter].peakEndVoltage -
scans[scanCounter].peakStartVoltage);
       scans[scanCounter].intercept = scans[scanCounter].peakEndCurrent - (scans[scanCounter].slope *
scans[scanCounter].peakEndVoltage);
       area = 0.0;
       for (int i = scans[scanCounter].peakStartRow; i < scans[scanCounter].peakEndRow - 1; i++)
       {
        TableRow rowLo = scans[scanCounter].mainTable.getRow (i);
        TableRow rowHi = scans[scanCounter].mainTable.getRow (i+1);
        dx = rowHi.getFloat ("voltage") - rowLo.getFloat ("voltage");
        if (smooth)
        {
         currentLo = rowLo.getFloat ("sg") - (scans[scanCounter].slope * rowLo.getFloat("voltage") +
scans[scanCounter].intercept);
         currentHi = rowHi.getFloat ("sg") - (scans[scanCounter].slope * rowHi.getFloat("voltage") +
scans[scanCounter].intercept);
        }
        else
        {
         currentLo = rowLo.getFloat ("current") - (scans[scanCounter].slope * rowLo.getFloat("voltage") +
scans[scanCounter].intercept);
```

```
          currentHi = rowHi.getFloat ("current") - (scans[scanCounter].slope * rowHi.getFloat("voltage") +
scans[scanCounter].intercept);
        }
// liteSTAT GUI code for Prototype Model B page 32 of 35

        area = area + (0.5 * (currentLo + currentHi) * dx);
        rectMode (CORNERS);
        areaRectXpos1 = map (rowLo.getFloat("voltage"), xmin, xmax, 220.0, 860.0);
        areaRectXpos2 = map ((rowHi.getFloat("voltage")), xmin, xmax, 220.0, 860.0);
        if (smooth)
        {
          areaRectYpos1 = map (((rowLo.getFloat("sg") + rowHi.getFloat ("sg")) / 2), ymin, ymax, 540.0,
60.0);
          areaRectYpos2 = map ((rowHi.getFloat("sg") - currentHi), ymin, ymax, 540.0, 60.0);
        }
        else
        {
          areaRectYpos1 = map (((rowLo.getFloat("current") + rowHi.getFloat ("current")) / 2), ymin,
ymax, 540.0, 60.0);
          areaRectYpos2 = map ((rowHi.getFloat("current") - currentHi), ymin, ymax, 540.0, 60.0);
        }

        noStroke ();
        fill (200, 100, 200, 75);
        rect (areaRectXpos1, areaRectYpos1, areaRectXpos2, areaRectYpos2);
      }
      scans[scanCounter].peakArea = area;
      strokeWeight (1);
      rectMode (CORNER);

    }
   }

   if (showPeaks)
   {

     int j = 440;
      if (scans[scanCounter].peakTrue)
      {
        stroke (scans[scanCounter].scanColour);
        strokeWeight (5);
        line (20, j-4, 40, j-4);
        noStroke ();
        fill (255);
        rect (50, j-15, 120, 20);
        fill (0);
        //scans[scanCounter].peakArea = scans[scanCounter].peakArea * 1000000.0;
        scans[scanCounter].peakVoltage = round(scans[scanCounter].peakVoltage * 100.0f)/100.0f;
        scans[scanCounter].peakHeight = round(scans[scanCounter].peakHeight * 100.0f)/100.0f;
        scans[scanCounter].peakArea = round (scans[scanCounter].peakArea * 1000.0f)/1000.0f;
        text (scans[scanCounter].peakVoltage + " V, " + scans[scanCounter].peakHeight + " uA", 53, j);
        text ("Area: " + scans[scanCounter].peakArea + " uW", 53, j +20);
        strokeWeight (1);
        j -= 25;
      }
     }
  }
}
void findMax ()
{
  float maxStartF, maxEndF;
  int maxFound;
  maxStartF = float (peakStartT.getValue());
  maxEndF = float (peakEndT.getValue());
  maxFound = scans[scanCounter].scanMax (maxStartF, maxEndF);
  peakMaxT.setValue (str(maxFound));
}
void addInitials () {
  c = new GUIController(this);
  cvB = new IFButton ("CV", 20, 20, 40, 18);
  lsvB = new IFButton ("LSV", 70, 20, 40, 18);
```

```
  swvB = new IFButton ("SWV", 120, 20, 40, 18);
  chronoB = new IFButton ("Chronoamp", 20, 48, 140, 18);
  runB = new IFButton ("Run Scan", 20, 76, 140, 18);
// liteSTAT GUI code for Prototype Model B page 33 of 35

  //testB = new IFButton ("Test Button", 20, 470, 140, 18);
  voltB = new IFButton ("Compliance: Low", 20, 510, 140, 18);
  saveL = new IFLabel ("Filename:", 920, 22, 80);
  saveT = new IFTextField ("", 920, 40, 80);
  loadMethodB = new IFButton ("Load Method", 920, 72, 80, 18);
  loadDataB = new IFButton ("Load Data", 920, 100, 80, 18);
  clearB = new IFButton ("Clear Data", 920, 128, 80, 18);
  exportScanB = new IFButton ("Export Scan", 920, 156, 80, 18);
  sgB = new IFButton ("Smooth Data", 920, 184, 80, 18);
  peaksB = new IFButton ("Find Peaks", 920, 242, 80, 18);
  scanNumberB = new IFButton ("Scan #", 920, 270, 44, 18);
  holdCB = new IFCheckBox ("Hold Gain", 20, 458);
  portB = new IFButton ("Choose Port", 20, 480, 140, 18);
  highL = new IFLabel ("HI", 161, 549, 18);
  lowL = new IFLabel ("LO", 3, 549, 18);
  cRangeL = new IFLabel ("Current Range:", 20, 530, 18);
  scanNumberT = new IFTextField ("", 970, 267, 32);
  maxB = new IFButton("<MAX>", 920, 322, 80, 18);
  peakStartL = new IFLabel ("<", 920, 348, 18);
  peakEndL = new IFLabel (">", 920, 374, 18);
  peakStartT = new IFTextField ("", 950, 348, 51);
  peakEndT = new IFTextField ("", 950, 374, 51);
  peakMaxL = new IFLabel ("Max", 920, 400, 18);
  peakMaxT = new IFTextField ("", 950, 400, 51);
  voidL = new IFLabel ("", -10, -10);
  debugL = new IFLabel ("", width/2, height/2);
  screenGrabCB = new IFCheckBox ("Include png", 920, 300);

  cp5 = new ControlP5(this);
  cp5.addSlider(" ")
    .setDecimalPrecision(0)
    .setRange(1,8)
    .setValue(4)
    .setPosition(20,550)
    .setSize(140,18)
    .setNumberOfTickMarks(8)
    .setSliderMode(Slider.FIX);
  cp5.addSlider("  ")
    .setDecimalPrecision(0)
    .setRange(1,10)
    .setValue(5)
    .setPosition(920,212)
    .setSize(80,18)
    .setNumberOfTickMarks(10)
    .setSliderMode(Slider.FIX);
  c.add (cvB);
  c.add (lsvB);
  c.add (swvB);
  c.add (chronoB);
  c.add (voltB);
  c.add (runB);
  c.add (scanNumberB);
  //c.add (testB);
  c.add (clearB);
  c.add (sgB);
  c.add (exportScanB);
  c.add (loadDataB);
  c.add (loadMethodB);
  c.add (peaksB);
  c.add (saveT);
  c.add (scanNumberT);
  c.add (saveL);
  c.add (highL);
  c.add (lowL);
  c.add (cRangeL);
  c.add (screenGrabCB);
```

```
  c.add (holdCB);
  c.add (portB);
  c.add (maxB);
// liteSTAT GUI code for Prototype Model B page 34 of 35

  c.add (peakStartL);
  c.add (peakEndL);
  c.add (peakStartT);
  c.add (peakEndT);
  c.add (peakMaxL);
  c.add (peakMaxT);

  rectMode (CORNER);
  noStroke ();
  fill (255);

  portB.addActionListener (this);
  cvB.addActionListener(this);
  swvB.addActionListener(this);
  lsvB.addActionListener (this);
  chronoB.addActionListener (this);
  voltB.addActionListener (this);
  runB.addActionListener (this);
  //testB.addActionListener (this);
  clearB.addActionListener (this);
  sgB.addActionListener (this);
  loadDataB.addActionListener (this);
  exportScanB.addActionListener (this);
  saveT.addActionListener (this);
  scanNumberB.addActionListener (this);
  scanNumberT.addActionListener (this);
  holdCB.addActionListener (this);
  maxB.addActionListener (this);
  peakStartT.addActionListener (this);
  peakEndT.addActionListener (this);
  peakMaxT.addActionListener (this);

  loadMethodB.addActionListener (this);
  peaksB.addActionListener (this);
  screenGrabCB.addActionListener (this);

  makeMins ();

  gridAL = new IFLabel ("", 200, 565);
  gridBL = new IFLabel ("", 280, 565);
  gridCL = new IFLabel ("", 360, 565);
  gridDL = new IFLabel ("", 440, 565);
  gridEL = new IFLabel ("", 520, 565);
  gridFL = new IFLabel ("", 600, 565);
  gridGL = new IFLabel ("", 680, 565);
  gridHL = new IFLabel ("", 760, 565);
  gridIL = new IFLabel ("", 840, 565);
  gridJL = new IFLabel ("", 182, 525);
  gridKL = new IFLabel ("", 182, 445);
  gridLL = new IFLabel ("", 182, 365);
  gridML = new IFLabel ("", 182, 285);
  gridNL = new IFLabel ("", 182, 205);
  gridOL = new IFLabel ("", 182, 125);
  gridPL = new IFLabel ("", 182, 45);
  currentL = new IFLabel ("Current (uA)", 180, 8);
  voltageL = new IFLabel ("Voltage (V)", 840, 583);
  c.add (gridAL);
  c.add (gridBL);
  c.add (gridCL);
  c.add (gridDL);
  c.add (gridEL);
  c.add (gridFL);
  c.add (gridGL);
  c.add (gridHL);
  c.add (gridIL);
  c.add (gridJL);
```

```
  c.add (gridKL);
  c.add (gridLL);
  c.add (gridML);
// liteSTAT GUI code for Prototype Model B page 35 of 35

  c.add (gridNL);
  c.add (gridOL);
  c.add (gridPL);
  c.add (currentL);
  c.add (voltageL);
}

void removeAllTandL () {

  singleScanOnly = false;
  c.remove (startT);
  c.remove (endT);
  c.remove (rateT);
  c.remove (cyclesT);
  c.remove (depPotT);
  c.remove (depTimeT);
  c.remove (equiTimeT);
  c.remove (pulseT);
  c.remove (waitT);
  c.remove (stepT);
  c.remove (startL);
  c.remove (endL);
  c.remove (rateL);
  c.remove (cyclesL);
  c.remove (depPotL);
  c.remove (depTimeL);
  c.remove (equiTimeL);
  c.remove (pulseL);
  c.remove (stepL);
  c.remove (waitL);
  c.remove (voidL);

}

void splashScreen () {
  int x = 0;
  int y = 0;
  int z = 0;
    while (y < height) {
      while (x < width) {
        float r = random (255);
        float g = random (255);
        float b = random (255);
        fill (r, g, b);
        rect (x, y, 15, 15);
        x += 15;
      }
      x = 0;
      y += 15;
    }
    image (splashLogo, 200, 200);
    noStroke ();
    fill (255, 255-frameCount);
    rect (200, 200, 608, 200);
}
```