The Electrical Properties of Interfacial Double Layers

A thesis

submitted in partial fulfilment

of the requirements for the Degree

of

Doctor of Philosophy

at the

University of Waikato

by

Mark Hedley Jones

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

University of Waikato
2011

Contents

List of Figures

List of Tables

Abstract

test

Acknowledgement

ienrstiesrnitsnr

Part I

Introduction

babies are sick

Chapter 1
Liquids at Scale

Liquids are a vast collection of mobile atoms, ions and molecules. Thinking of liquids as such allows one to better visualise their behaviour at small scales. It is the behaviour of liquids at small scales that this thesis is concerned with.

1.1 Liquids at scale

At the macro scale, the behaviour of liquid is simple and calculable. Modern computers can simulate the movement of liquids using the Navier-Stokes equations with accuracy and speed. This does not hold true when modelling fluid at the scale of individual atoms and molecules, where the behaviour is determined purely by electrical interactions. These interactions are chaotic and therefore simulation of any reasonable sized volume demands prohibitive computing resources.

Water is a liquid comprised of the molecule H_2O. This molecule is polar, meaning one side has a net positive charge and the other is net negatively charged. This simple fact is responsible for very complex behaviour when dealing with a liquid body of reasonable size.

This thesis, in essence, investigates interactions that take place at the boundary between a liquid and a solid. These interactions take place primarily on the liquid side of the boundary within a volume termed the double layer. The

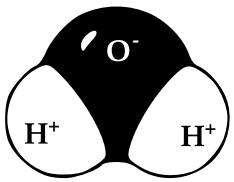

Figure 1.1: A single polar water molecule

electric field of the solid's interior would be affected by the accumulation of charge at its surface, but this should not affect the analysis of the exterior layer. It is important to note that a solid in the context of this thesis is not simply the walls of a container, but may also refer to solid particles, even molecules, within a liquid. This is the case for emulsified substances such as paints where the formed double layer determines the stability of the emulsion.

This thesis focusses on two applications of interfacial double layers with respect to electrical engineering. Firstly, it looks at a method of harvesting power from water by way of a mechanism that has no moving parts; and secondly, it models the electrical impedance that is seen by a medical implant when placed inside a biological system. The second application is where the majority of the research effort can be found.

1.1.1 The Promise of Molecular simulation

Physical simulations involving the properties of materials and interactions with radio-waves are a common tool for any RF engineer. There are many areas in science and engineering that benefit from powerful computer simulation. Many modern-day computer games simulate the position and velocity of thousands of objects, such as bullets and buildings breaking apart, in near real-time.

These kinds of simulations generally involve calculating the state of a system of objects at a specific time, then adjusting the all of the parameters based the physics of that system at a specific time, and repeating this process for as long as the simulation need continue.

Finite-difference time-domain (FDTD) simulation is a common technique for calculating electrical fields and resulting currents and voltages in the field of electrical engineering. Such simulations can involve hundreds of thousands, *sometimes millions*, of objects. Generally these objects are elements of a 3D mesh that has been created to represent the geometry of the system to be simulated. In such a simulation, each mesh element at every time step needs its various parameters (e.g. temperature, electromagnetic fields, electrical current and voltage) to be calculated based on the parameters of neighbouring objects in the mesh. Because of the dependence on neighbouring parameter values, each time-step may take many passes over each element in a system to calculate the final state before moving on. These sorts of simulations are extremely valuable for the radio frequency engineer as systems of a practical size can be simulated relatively quickly on modest computing hardware.

At this point it seems natural to assume that we could model the behaviour of atoms and molecules in a liquid. Doing this would allow simulation of the interactions that take place at the fluid-soild boundary. It is possible to run

such molecular dynamics (MD) simulations, however the number of atoms and molecules in any practical sized volume of fluid is astronomical.

The molar mass of water is defined as $18.0528\,g/Mol$, which represents the amount of water in moles per gram. One gram of water is defined as one millilitre so we can say that for every millilitre of water we have we have an eighteenth of a mole of water. Avagadro's constant, the number of constituent particles per mole of a given substance, is 6.0221×10^{23}. Therefore we have one eighteenth of Avagadro's constant in water molecules, which is about 3.3333×10^{22} molecules. That is 33 333 333 333 333 333 333 333 molecules! The behaviour of double layers in the applications of this research require a far greater volume than $1\,ml$ to be useful, putting molecular simulation out of the picture for the applications discussed in this thesis for the foreseeable future.

Chapter 2
Interfacial Double Layers

Double layers appear to varying degrees almost everywhere an electrolytic fluid and a solid are in contact. They are very small Todo: quantify typical double layer sizes, existing in the micro scale, and are responsible for a number of macro scale effects. Such effects are the stability of emulsions Todo: find instances where double layers play a role, the something else and even... They are a natural response when two systems of arrangement are brought to contact - fixed and fluid arrangements of charged particles.

2.1 Physical Description

Double layer formation is the result of a liquid system responding to an imbalance of charge distribution. Atoms within a solid domain are static, they cannot repel each other or rearrange themselves. In a fluid the opposite is true, each freely moving charged entity will attempt to distance itself from others of like charge and move toward those of opposite charge. When these two domains are brought into contact with one another, the one whose elements are free to move (the liquid), moves to accommodate the imbalance in charge between the two domains. The effect of which is that an electrolytic liquid attracts atoms and molecules with a net opposite charge to the surface of the solid.

2.1.1 Illustrated formation of a double layer

Figures 2.2 and 2.3 illustrate the formation of such a layer around a charged object that, for the sake of illustration, instantaneously appears in a settled electrolyte solution. In these figures, the spacing of depicted fluid elements do not represent the density of a liquid system and the smooth surface of the charged object in no way resembles the surface of a solid at the scale of individual atoms.

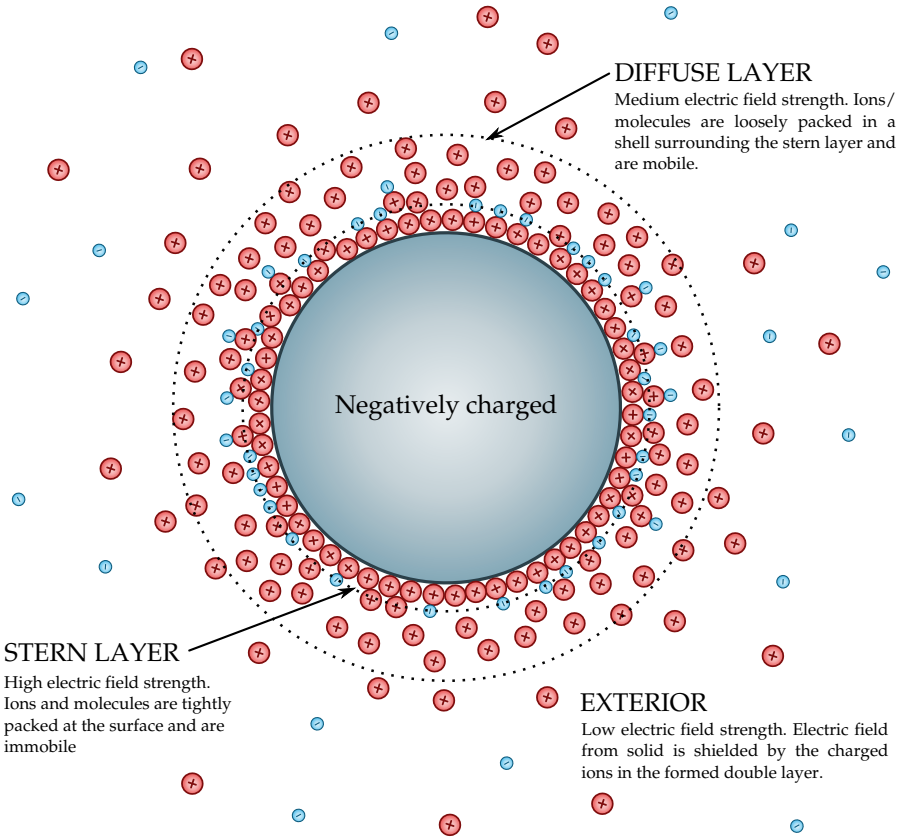

Figure 2.1: Anatomy of the double layer

The first set of images, figure 2.2, depict charged elements in the fluid responding to the presence of the solid by aligning themselves with the induced field and migrating toward or away from the object. The second set of images, figure 2.3, shows continued migration of charged particles and the formation of the Stern layer at the solid-liquid interface. Members of this layer are bound so strongly to the object that they are effectively immobile, i.e., they are adsorbed to the solid object. A substantial proportion of the initial electrical field from the solid is shielded by the Stern layer.

Outside the Stern layer, as the field strength continues to diminish with increasing radial distance, ions and molecules are attracted with decreasing strength. This reduction in attraction forms a layer that is thicker than the stern layer and mobile; the ions are free to move locally within this secondary layer. This secondary layer is called the diffuse layer and it is convenient to visualise it as a fluffy cloud of charged particles attracted to, but not trapped upon, the thin Stern layer immediately adjacent to the surface.

2.1.2 Criteria for formation

It is important to note that in order for a double layer to form, the liquid phase must contain ions, charged molecules, or polar molecules. This means of course that there are likely to be non-participating elements within the fluid, that is, elements not affected by the charge Todo: check affected and effected of surrounding elements. The reverse case is also true Todo: is this correct?, I just made it up where an electrolyte solution comes in contact with a electrically neutral and stable solid. Although this case is not as obvious as one may think. For example, placing pure water in a glass flask, it may be assumed, would not cause the formation of a double layer. Glassware in used throughout chemistry laboratories as it reacts with very little and pure water with no salt ions should be fairly stable. This is not the case however as glass in contact with water undergoes a process of deprotonation at the boundary. This deprotonation is where H^+ ions are torn from the surface of the glass and enter the liquid phase. As this happens, the surface of the glass is left negatively charged. Subsequently, a double layer is formed from polar water molecules and hydrated protons.

2.2 Electrical Behaviour

2.2.1 Capacitive Action

2.2.2 Methods of Modelling

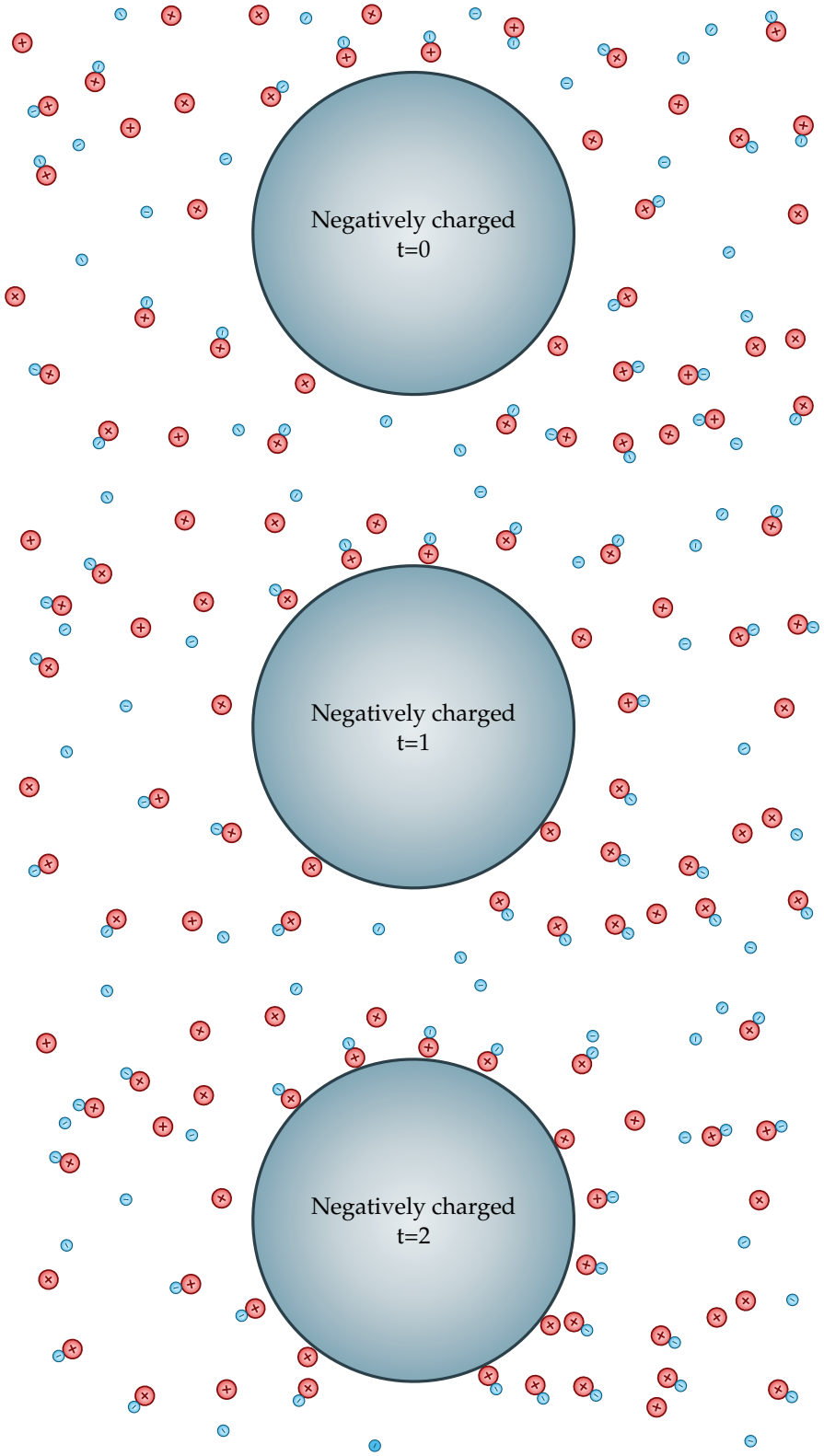

Figure 2.2: Creation of a double layer (time 0-2)

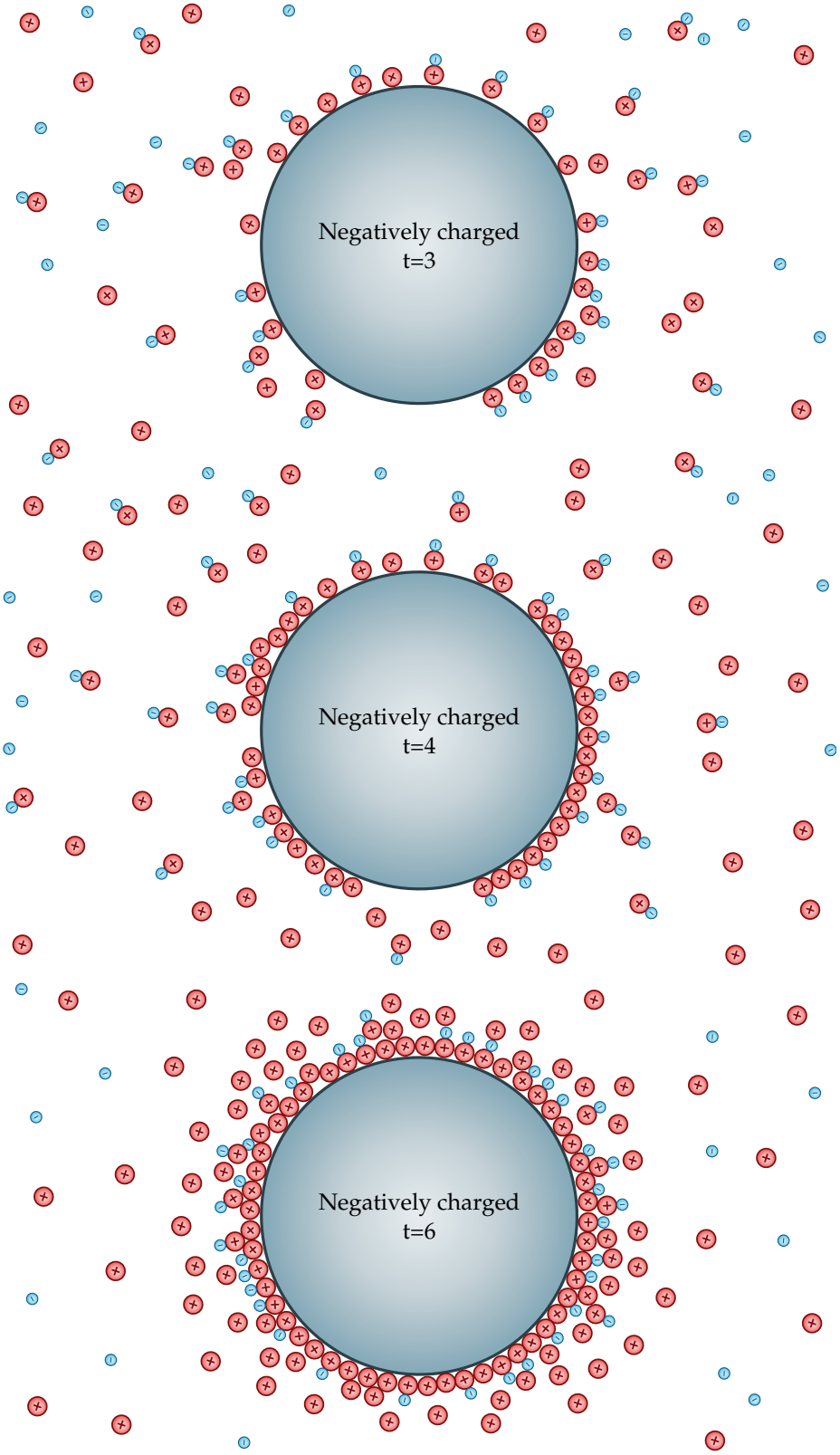

Figure 2.3: Creation of a double layer (time 3-6)

Part II

Double Layers On Insulators

Chapter 3
Harvesting Energy

One practical application of the interfacial double layer is that of electrical energy harvesting. As was discussed in the introduction, a double layer is a naturally formed of a layer of like-charged ions and molecules surrounding the surface of a charged solid. The previous sentence is especially important, it essentially states: *free charge naturally accumulates in certain predictable places*. Now we ask the question "can we collect this charge and put the charge to use?"

Part II of this thesis explains how harvesting energy from double layers may be done, quantifies how much energy one may want to generate and concludes by stating what such a harvester would look like and its feasibility.

3.1 The streaming potential cell

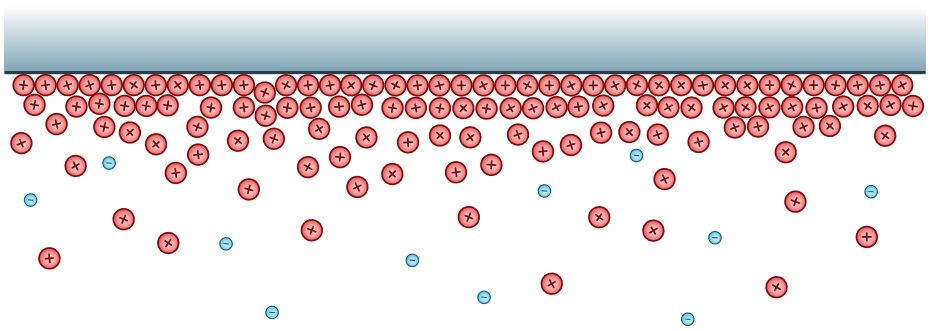

Figure 3.1: A double layer formed along the edge of a container

A streaming potential cell is an apparatus that enables charge separation simply by forcing water through narrow cavities. This phenomenon is understood and is used in colloidal science to determine what's called the zeta potential of an interface [**?**, **?**, **?**, **?**, **?**]. Although this phenomenon isn't new, using it to generate useful amounts of electrical energy is. This chapter will investigate

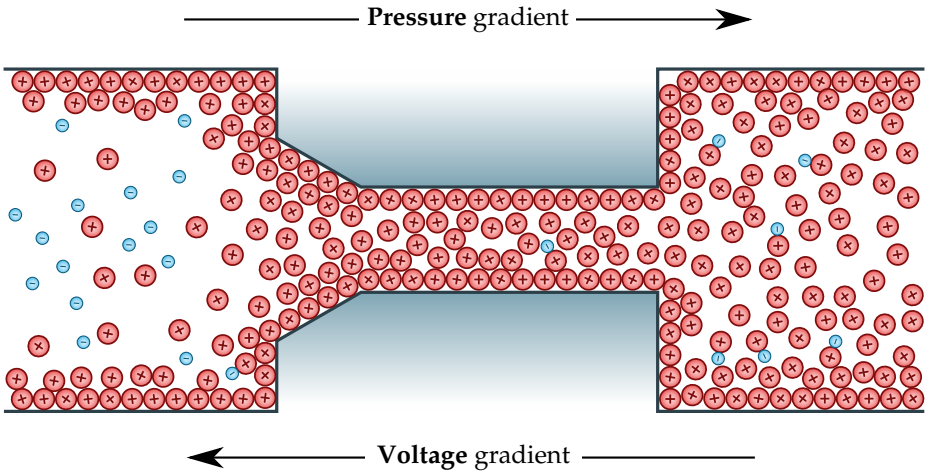

Figure 3.2: The general principle of operation of the streaming potential cell

the application of streaming potentials to energy harvesting for the purposes of running a smart water meter.

In this diagram, a the negatively charged solid is surrounded by layers of positively charged ions that are electrostaticly attracted to the surface of the solid. Near the surface of the solid, tightly bound, positively charged ions are packed together and are largely immobile. These ions reside in what is called the Stern Layer. After this layer is a shell of less densely packed and somewhat mobile ions that are still bound to the solid. At the edge of this shell is what is called the slipping plane, which is the point at which ions are no-longer bound to the solid. The electrical potential at this slipping plane is termed the ζ potential (zeta potential). "This zeta potential is directly related to the interaction between the solid particles themselves when they are suspended in a liquid and thus determines the stability of the suspension solutions" [**?**].

Now lets look at the case where the geometry is such that the surface charge resides on the walls of a channel through which the liquid can flow, as is shown in Figure **??**. By setting the width of the channel such that the double layer formed on each of the sides partially overlap, the channel will be predominantly occupied by ions of the opposite polarity to that of the surface charge at the solid to liquid interface. This configuration has the effect of promoting the transport of those ions across the channel with the application of a pressure differential across the channels length.

3.2 Replicating an experiment

There are many papers describing experiments with streaming potential cells [**?**, **?**, **?**, **?**] of which we have chosen one by Yongan Gu and Dongqing Li [**?**] to attempt to replicate. This paper was chosen for its detailed description of the experimental procedure used, including the brands and model numbers of the equipment used. Our experimental procedure is based upon this paper however we have modified certain areas of the experiment to accommodate for the resources available to us. For a list of the materials used to construct the channels, see Table 3.6.

3.2.1 Experimental Procedure

3.2.1.1 Construction

First, glass microscope slides where cut in half to give panels of glass that where approximately 26 x 38mm. Once such glass panel can be seen in Figure 3.3. Next, two pieces of shim material where placed on either side of the glass panel with a small amount of epoxy, on top of which another glass panel was placed. Pressure was applied to the panels pressing them together while the epoxy set to create the channel that can be seen in Figure 3.4. Each channel was photographed under microscope once set to determine the height of the channel created. Finally, acrylic couplers where mounted over each end, again by the use of epoxy. These couplers allowed the water to be forced through the channel while providing mounting for the voltage sensing wires. The final assembly is shown in Figure 3.5.

3.2.1.2 Measurement

Each assembly was connected between the tap and drain of a lab basin. To facilitate measurement of the pressure gradient across the channel, a pressure transducer was placed across the input and output ports of the channel assembly. Single core copper wire was used to measure voltage gradients across the channel. These wires where epoxied into the couplers to reduce leaks. To conduct the measurements an Agilent E5270B Precision Measurement Mainframe was used, which has an input impedance of $10\,T\Omega$. The high input impedance of this machine ensures that the measurements are not disturbed by current leaking through the measurement device. The E5270 was responsible for measuring the output of the pressure transducer and the voltage across the channel.

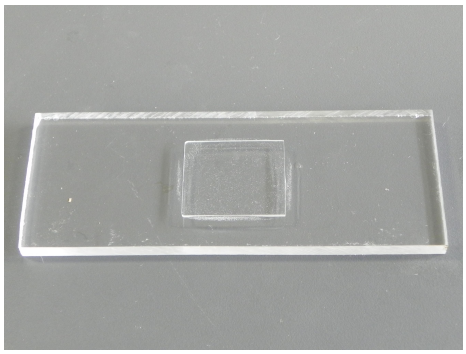

Figure 3.3: Photo showing half of a glass slide glued to acrylic base plate

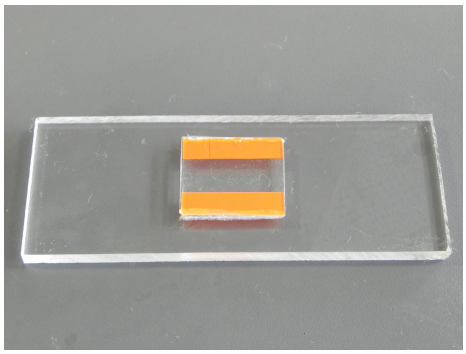

Figure 3.4: Photo showing shims sandwiched between two slide halves

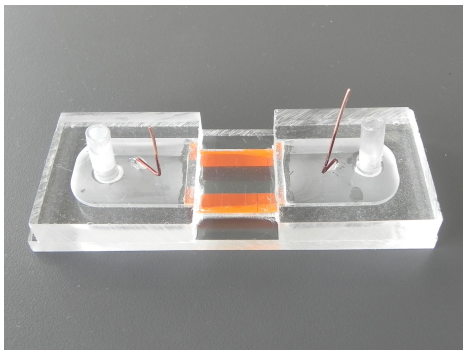

Figure 3.5: Photo showing final assembly

Microscope slides	Sail Brand	JIA 7101WT - 26 x 76mm
Shims	Garlock	Colorplast - 50 μm, 80 μm, 120 μm and 250 μm
Epoxy	Selleys	Araldite - Ultra Clear Resin
Pressure Sensor	Honeywell	24PC15SMT - 0 – $\pm$15 PSI

Figure 3.6: Table of materials used to construct the streaming potential cells

3.2.2 Results

Ten cavities ranging in height from 26 μm to 245 μm where built and measured. Individual plots showing pressure applied versus voltage developed are shown in Appendix C.1. For convenience, a graph with the steepest voltage per pressure gradient is shown in Figure 3.7, which resulted from using a cell with a height of 52 μm.

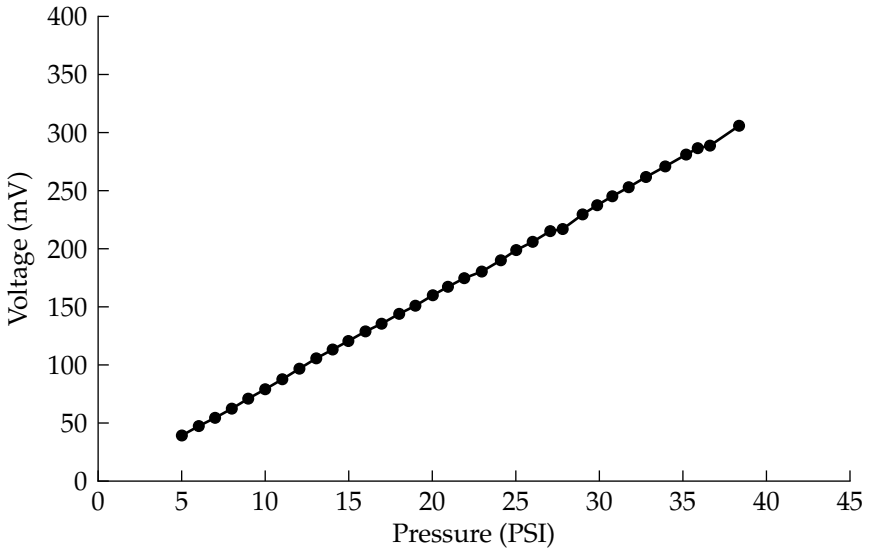

Figure 3.7: Line graph showing voltage output versus pressure for a 52 μm glass micro-channel.

From the results obtained it is clear that there is a definite - linear - relationship between the amount of pressure applied and the voltage developed across each channel. Figure 3.8 compares the gradient of voltage developed per PSI of pressure for each of the channels. The graph shows a trend toward larger voltage development as channel height is reduced to around 50 μm. Below 50 μm it is unclear what to expect from the channel which will require further investigation. Spread in the data is expected to be the result of epoxy (used to glue the slides together) seeping into the cavity, thereby slightly altering each cavity's dimensions.

3.2.3 Conclusion

Power generation from streaming potential cells is possible, this is evident from the voltages developed in this experiment. The streaming cell is capable of producing output when operated using standard tap water at standard tap

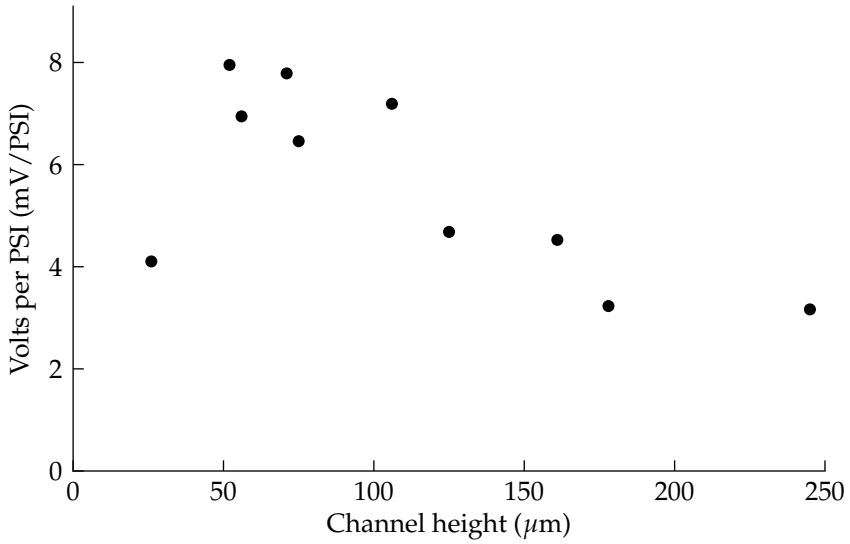

Figure 3.8: Scatter plot of voltage/pressure gradient versus channel height for each of the measured channels.

pressures. As yet, the amount of current that can be drawn from such a channel has not been ascertained, and therefore the amount of available power remains unknown (although it is estimated to be in the very small). An initial estimate of the ideal channel lies between 25-75 μm in height.

3.3 Governing model

3.3.1 Charge separation analysis

As mentioned at the beginning of this chapter, for a streaming potential cell to operate there must be an electrical potential at the interface between solid and liquid matter. In the experiment that Wayne Crump and myself replicated, the solid/liquid interface was between glass and standard Hamilton tap water, so where does this electrical potential come from?

Quoting Tandon et al. in [**?**]: Many microfluidic substrates behave as weak acids in aqueous solutions. In glass substrates, surface silanol groups can be deprotonated in aqueous solutions leaving a negative surface charge:

$$SiOH \overset{K_a}{\rightleftharpoons} SiO^- + H^+ \tag{3.1}$$

It is this deprotonation that is responsible for generating the charge difference at the glass/liquid surface of the microchannel and therefore establishing

an electrical double layer (EDL). Since the charge imbalance in this case arises from the fact that the glass behaves as a weak acid, the process itself is affected by the pH of the liquid, something we have no control over when using tap water.

Once an ionic double layer within the channel has been established, "the streaming potential is engendered by the flow of the ions relative to the stationary charged wall of the channel" [**?**]. This flow is accomplished in our case by applying a pressure gradient across the channel. "The ratio of streaming potential to pressure gradient depends on the zeta potential"[**?**]

3.3.2 Mathematics

The first step toward optimisation is to define all the parameters that describe the device. The streaming current, which is defined as the electrical current that flows in the direction of water flow, is defined in [**?**] to be:

$$I_s = \frac{A\,\varepsilon_0\,\varepsilon_r}{\eta\,l}\Delta P\,\zeta \tag{3.2}$$

where:

ε_0 is the permittivity of free space

ε_r is the relative permittivity of the liquid

η is the viscosity of the liquid

l is the length of the channel

ΔP is the pressure difference across the channel

ζ is the zeta-potential of the of the glass-liquid interface

A is the cross-sectional area of the channel, or combined area of multiple channels

This equation indicates that the cell will produce a current that is proportional to the amount of pressure across it and that the output is affected by the geometry of the channel itself. To help model the situation it is convenient to break

Equation 3.2 formula down in the following manner:

$$\begin{aligned} I_s &= \Delta P \left(\frac{A}{\eta\, l}\right) (\varepsilon_0\, \varepsilon_r\, \zeta) \\ I_s &= \frac{\Delta P}{\left(\frac{\eta\, l}{A}\right)} (\varepsilon_0\, \varepsilon_r\, \zeta) \\ I_s &= \frac{\Delta P}{R_h} (\varepsilon_0\, \varepsilon_r\, \zeta) \\ \frac{I_s}{\Delta P} &= \frac{\varepsilon_0\, \varepsilon_r\, \zeta}{R_h} \\ g_m &= \frac{\varepsilon_0\, \varepsilon_r\, \zeta}{R_h} \end{aligned} \tag{3.3}$$

where R_h is the hydrodynamic resistance caused by the channel, g_m represents the transconductance between pressure applied and current produced and I_s is the streaming current. The hydrodynamic resistance (R_h) will be affected by the geometry of the channel so R_h is considered an approximation of the hydrodynamic resistance.

Once the cell begins to separate charge a current in the reverse direction is established called the conduction current (I_c).

Ohms law states:

$$I = \frac{V}{R}$$

In this situation the resistance is determined by the conductivity of the water (σ), as well as the cross sectional area (A) and length (l) of the channel.

$$R = \frac{l}{\sigma\, A}$$

Therefore, the conduction current back through the channel is determined by:

$$I_c = A\sigma\frac{V_s}{l} \tag{3.4}$$

Where σ is the conductivity of the liquid and V_s is the voltage developed across the channel. This is also shown in

At equlibrium the streaming current (I_s) and the conduction current (I_c) are equal and opposite in direction.

3.3.3 Electrical model

Figure 3.9 depicts schematically how a streaming cell would operate when connected to an external load. This model is based upon that found in [**?**]

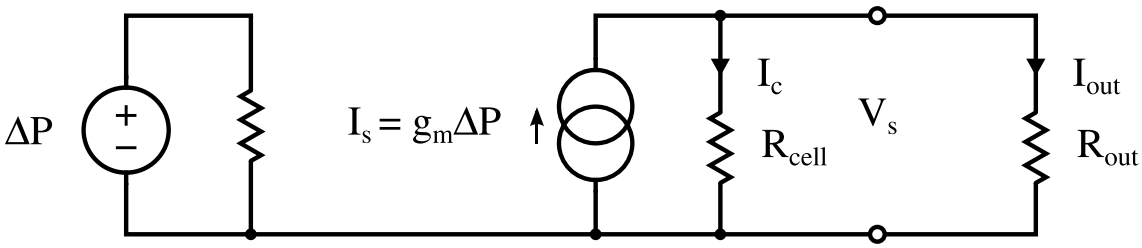

Figure 3.9: Schematic representation of a streaming cell with attached output resistance

by Olthuis et al. but has been modified to show the pressure applied (ΔP) as the voltage source, instead of the zeta potential (ζ). It is the pressure that responsible for creating the streaming current; a zeta potential on its own simply isn't enough. This has been shown through the results obtained in Appendix C.1, where V_s was directly proportional to ΔP; as V_s is directly related to I_s as per Ohm's law.

3.3.4 Output analysis

The electrical model described in 3.3.3 shows how the cell would operate with a load placed across the cell. By combining this model with the equations from 3.3.2 it should be possible to see the effect that altering various cell parameters has on the output.

$$\begin{aligned} P &= V \times I \\ V &= I \times R \\ P &= I^2 \times R \\ P_{out} &= I_{out}^2 \times R_{out} \\ I_{out} &= I_s \times \frac{R_{cell}}{R_{cell} + R_{out}} \\ P_{out} &= \left[I_s \times \frac{R_{cell}}{R_{cell} + R_{out}} \right]^2 \times R_{out} \end{aligned} \tag{3.5}$$

3.3.4.1 Finding an optimum value for R_{out}

Using Ohm's Law it is possible to find an equation that give the output power as a function of the electrical resistance of the cell ($R_{cell)}$) and the output resistance (R_{out}) as shown in Equation 3.5. In order to optimise the output power it is necessary to find the value of output resistance (R_{out}) that maximises the output power.

Next I take Equation 3.5 and treat the streaming current (I_s) as a constant, differentiate with respect to R_{out} and find the condition that gives a maximum/minimum power output.

$$\begin{aligned}
\frac{\partial P_{out}}{\partial R_{out}} &= \frac{R_{cell}^2}{(R_{cell}+R_{out})^2} - \frac{2 \times R_{cell}^2 \times R_{out}}{(R_{cell}+R_{out})^3} \\
0 &= \frac{R_{cell}^2}{(R_{cell}+R_{out})^2} - \frac{2 \times R_{cell}^2 \times R_{out}}{(R_{cell}+R_{out})^3} \\
\frac{2 \times R_{cell}^2 \times R_{out}}{(R_{cell}+R_{out})^3} &= \frac{R_{cell}^2}{(R_{cell}+R_{out})^2} \\
\frac{2 \times R_{cell}^2 \times R_{out}}{R_{cell}+R_{out}} &= R_{cell}^2 \\
2 \times R_{cell}^2 \times R_{out} &= R_{cell}^2 \times (R_{cell}+R_{out}) \\
2 \times R_{out} &= R_{cell}+R_{out} \\
R_{out} &= R_{cell}
\end{aligned}$$

This shows that there is either maximum or minimum power transfer to R_{out} when the value of R_{out} matches that of R_{cell}, which is shown to be a maximum as per Figure 3.10. This result is consistant with that of the Maximum Power Thereom, which I had trouble finding a reference of for a current source and resistances in parallel. It shows that in order to achieve the maximum possible output power, that the load should have a resistance equal to that of the electrical resistance of the cell itself. Additionally, it was found that the absolute magnitudes of both R_{out} and R_{cell} have no effect on the output power, so long as they are equal to one another.

This can now be used to calculate the maximum available power as a function of I_s by substituting R_{out} and R_{cell} for R , as $R_{out} = R_{cell} = R$:

$$\begin{aligned}
P_{out} &= \left[I_s \times \frac{R_{cell}}{R_{cell}+R_{out}} \right]^2 \times R_{out} \\
P_{max} &= \left[I_s \times \frac{1}{2} \right]^2 \times R \\
P_{max} &= \frac{I_s^2 R}{2}
\end{aligned} \tag{3.6}$$

And as $P = I^2R$ (via the power equation and Ohm's Law) this indicates that at most we can hope to havest half of the total power, while the other half is dissipated back through R_{cell}. Combining Equations 3.6 and 3.3 gives:

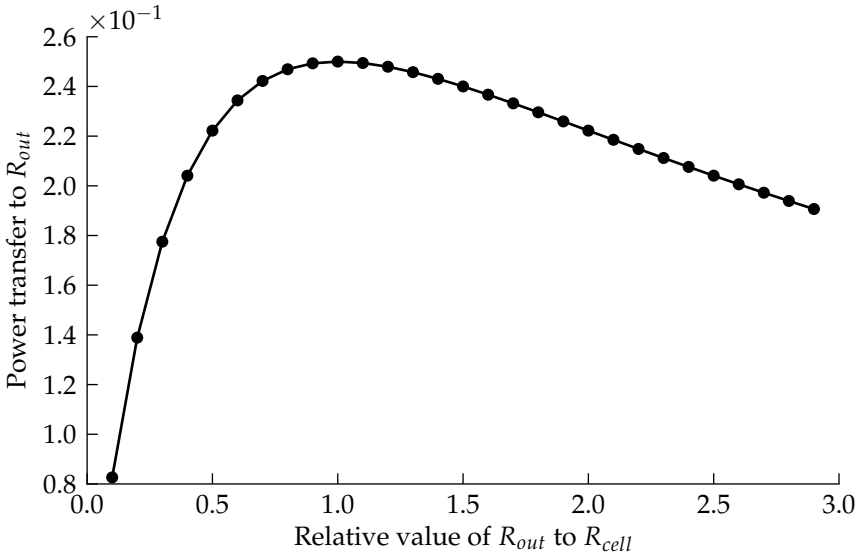

Figure 3.10: Plot of Equation 3.5 when $I_s = 1A$ and $R_{cell} = 1\Omega$

$$
\begin{aligned}
P_{max} &= \frac{\left(\frac{\Delta P}{R_h}\ (\varepsilon_0\,\varepsilon_r\,\zeta)\right)^2 R}{2} \\
P_{max} &= \frac{\left(\frac{\Delta P\,\varepsilon\,\zeta}{R_h}\right)^2 R}{2} \\
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2 R}{2R_h^2}
\end{aligned}
$$

where $\varepsilon = \varepsilon_0\,\varepsilon_r$ and $R = R_{cell} = R_{out}$. If we now substitute R and R_h for their respective functions we end up with:

$$
\begin{aligned}
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2 R}{2R_h^2} \\
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2\,\left(\frac{l}{\sigma A}\right)}{2\left(\frac{\eta l}{A}\right)_h^2} \\
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2\,l}{\sigma A} \times \frac{1}{\frac{2\eta^2 l^2}{A^2}} \\
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2\,l\,A^2}{\sigma A\,2\eta^2 l^2} \\
P_{max} &= \frac{\Delta P^2\,\varepsilon^2\,\zeta^2\,A}{2\,\sigma\,\eta^2\,l} \\
P_{max} &\propto \frac{\Delta P^2\,\zeta^2\,A}{l} \\
P_{max} &\propto \frac{\Delta P^2\,\zeta^2}{R_h}
\end{aligned} \tag{3.7}
$$

3.3.4.2 Optimisation of R_h and ΔP

Knowing the optimum value of output resitance leaves only the streaming current (I_s) and it's constituants as optimisation parameters. Equation 3.3 indicates that in order to further optimise the cell, one must reduce hydrodynamic resistance (R_h) and increase both the Zeta potential (ξ) and pressure difference (ΔP) placed across the cell.

We can model the water supply, harvester and a consumer as a basic electrical circuit as shown in Figure 3.11. In this model P_{supply} represents the total water pressure available, Flow represents the flow rate of water, R_h and $R_{consumer}$ are the hydrodynamic resistances of the harvesting cell and a water consumer (e.g. a house) respecively and ΔP is the pressure developed difference across the harvester, where $R_h \propto \frac{l}{A}$.

Figures 3.12, 3.13 and 3.14 show how varying R_h will affect ΔP, the flow rate and the value of $\frac{\Delta P}{R_h}$ respectively. Notice in Figure 3.12 that increasing R_h has the effect of increasing ΔP. Optimisation wise, the benifit of increasing I_s by increasing ΔP is opposed by also increasing the value of R_h (referring to Equation 3.3).

Figure 3.13 shows how the value of R_h affects the flow rate through the system. Keeping the flow rate high is important in order to ensure that the consumer does not notice a drop in water pressure due to the addition of the harvester in their water system.

Finally, Figure 3.14 shows the effect that varying R_h has on P_{max} (according

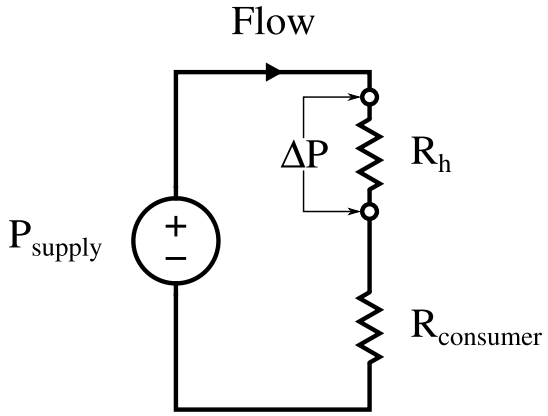

Figure 3.11: Schematic model of the water supply, harvester and consumer

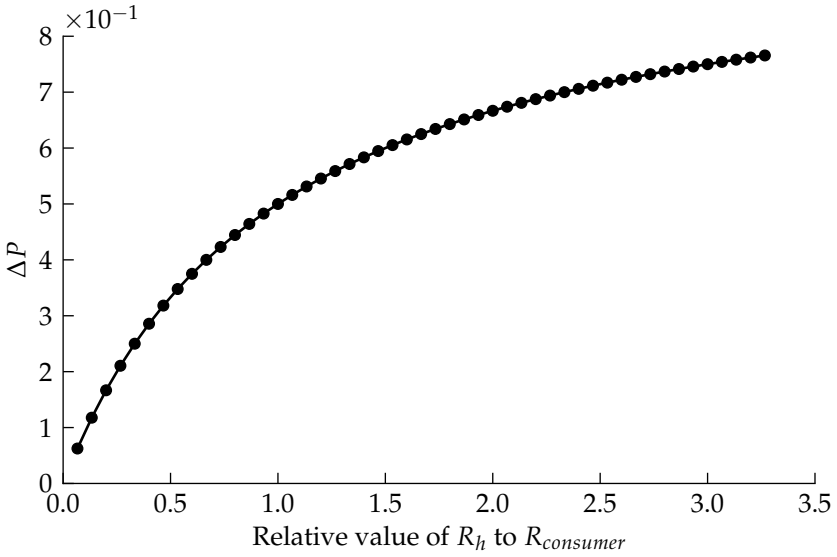

Figure 3.12: Effect of varying R_h on ΔP for the harvester/consumer model shown in Figure 3.11.

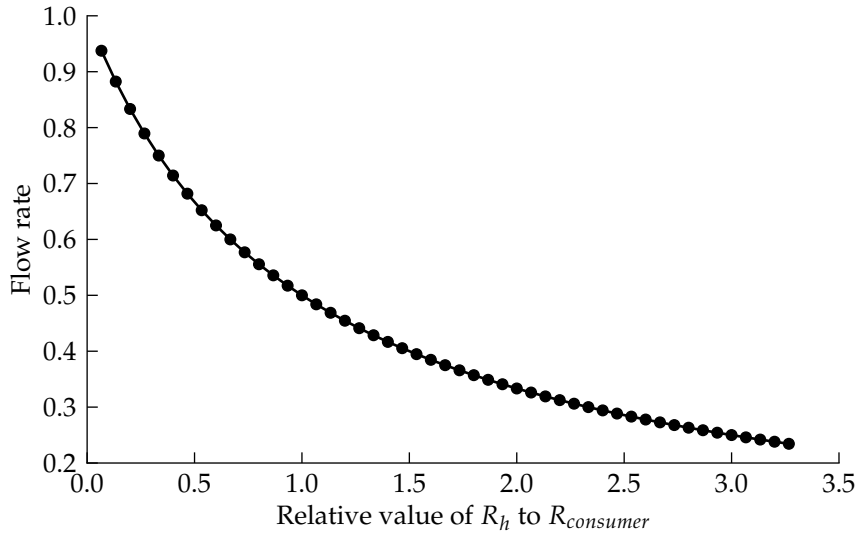

Figure 3.13: Effect of varying R_h on Flow for the harvester/consumer model shown in Figure 3.11.

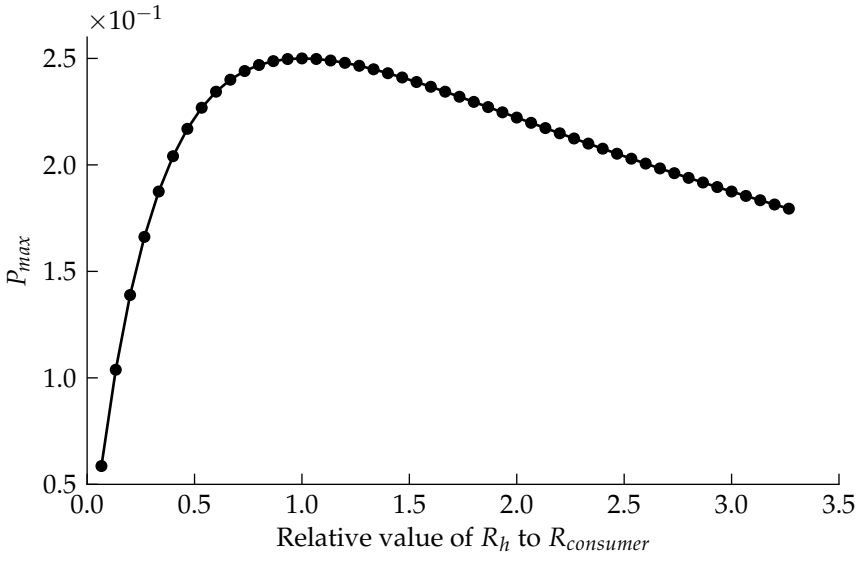

Figure 3.14: Effect of varying R_h on P_{max} for the harvester/consumer model shown in Figure 3.11.

to $P_{max} \propto \frac{\Delta P^2 \zeta^2}{R_h}$ when $\zeta = 1$). The result is the same as that of Equation 3.5, showing that this is again goverend by the maximum power transfer thereom. Practically, a harvester that dropped half of the supply pressure would not be feasable in most domestic situations, so a trade-off will have to be made that meets plumbing standards.

3.3.4.3 Optimisation of the Zeta potential

Referring back to Equation 3.2, I have determined how to increase the total output of the harvester by choosing appropriate values for all of the parameters except ε_r and ζ. As the harvester will be using domestic tap water the value of ε_r will be that of the relative permittivity of water, and is therefore relatively fixed (being only affected by the temperature of the water). This leaves ζ as the remaining parameter.

Chapter 4
Wireless Water Metering

Chapter 5
Energy Requirements

5.1 Microcontroller

At the heart of a smart metering system is the microcontroller (MCU), which among other things will be keeping track of the amount of water consumed. In order to know whether it is possible to extract enough power from a domestic water supply it is necessary to assess how much power is required to run a microprocessor and any associated hardware.

In this chapter I compare power consumption and operational efficiencies of six low power MCUs deemed suitable for running a power harvesting water meter. The microprocessors to be investigated are low power, general purpose 8-bit MCUs from three mainstream manufacturers, Microchip, Atmel and Freescale. Measurement of power consumption will be carried out during processing, sleeping and whilst undertaking two tasks that are essential for smart water metering, analog-to-digital conversion and non-volatile writes to memory.

5.2 Selection

The following microprocessors have been chosen as they represent a well spaced selection from the three mainstream MCU manufacturers.

- Microchip PIC16F1827
- Microchip PIC16F688
- Microchip PIC12F675
- Atmel ATtiny25V
- Atmel ATtiny13V
- Freescale MC9S08QG8

A basic feature comparison of the MCU selection is shown in table

5.3 Power consumption

Each of the MCUs were programmed to carry out various tasks which were performed over their specified range of operating voltages and operating frequencies. While the chips where carrying out these tasks their power consumption was measured by an Agilent E5270B Precision Mainframe via a PC and a Tektronix MSO 4054 Oscilloscope for timing purposes. Details on how each of the tests where carried out, as well as raw data, can be found in appendix

5.3.1 Test conditions

Where possible the chips were configured to have each of their pins set to outputs, held high, whilst being tied to Vdd via 10kΩ resistors. All chips had their peripherals disabled (where appropriate) including any watchdog timers and brownout detection circuitry.

To allow more accurate sleep current measurements, the chips were placed in a chip carrier with 10kΩ resistors soldered between the pins (except ground and Vdd) and Vdd. They chip and carrier was then washed in isopropyl alcohol and dried before being suspended by connections to the the E5270. These steps were taken to reduce any current leakage between pins due to fingerprints or dirt.

5.3.2 Sleeping

A microprocessor in sleep mode is basically switched off, the only difference being that volatile data is preserved provided the input voltage doesn't fall below a minimum threshold voltage. In order to consume as little power as possible an MCU should spend as much time in sleep mode as is feasible. Power consumption during sleep therefore plays a significant role in a chips ability to conserve power over time.

Figure 5.1 on page 40 shows the amount of current consumed by each of the MCUs while in their sleep state, note the log scale on the y-axis. It can be seen that the PIC16F1827 consumes the most current. This is somewhat contradictory to its specified sleep current of 30nA [**?**] (the second lowest of the set) at almost 1000 times higher. The Freescale MC9S08QG8 consumed energy at an average of 11% higher than specified [**?**]. Both the Atmel ATtiny13V and ATtiny25V fell within their specification, [**?**] and [**?**] respectively. There appears to be a trade-off made between the two chips in the way of minimum power consumption and minimum response to Vdd. The ATtiny13V required approximately 2.7 times

	PIC16F1827	PIC16F887	PIC16F688	PIC12F675	ATtiny25V	ATtiny13V	MC9S08QG8
Vdd (min)	1.8	2.0	2.0V	2.0	1.8	1.8	1.8
Vdd (max)	5.5	5.5	5.5V	5.5	5.5	5.5	3.6
I (sleep)	30nA	50nA	50nA	1nA	100nA	<100nA	450nA
CLOCK (min)	31kHz	31kHz	31kHz	31kHz	16kHz	16kHz	1MHz
CLOCK (max)	32MHz	8MHz	8MHz	4MHz	16MHz	9MHz	10MHz
EEPROM	256B	256B	256B	128B	128B	64B	+
Serial	USI	USI	USI	–	USI	–	USI
USART	UART	UART	UART	–	–	–	–
ADC	10bit	10bit	10it	10bit	10bit	10bit	10bit

†Has 8,192 bytes of software programmable flash (16 pages of 512 bytes each).
‡Has 256 bytes of software programmable flash (4 pages of 64 bytes each).

Table 5.1: Feature comparison of selected MCUs.

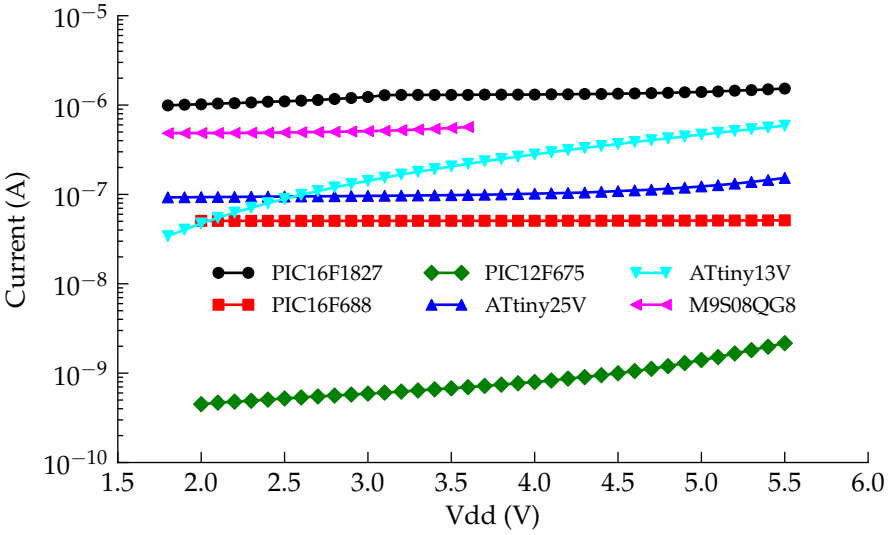

Figure 5.1: Current consumed by investigated MCUs while in sleep mode.

less power than the ATtiny25V at 1.8V, but this advantage rapidly falls away as Vdd moves up from 1.8V (and negated after 2.5V Vdd). The Microchip PIC12F675 fell within specification[**?**] and was the clear winner of the tested MCUs.

As the PIC16F1827 was so far off its specified value, the measurement was repeated numerous times using code written in both assembler and HI-TECH C as well as trying five different chips. All steps in the datasheet to reduce power consumption where followed and all configuration options where set, however no improvement was made.

5.3.3 Processing

Measuring the amount of power it takes to process information is not a simple task. The way each chip carries out processing operations internally can be quite different from one another, even though they all produce the same result. This section will focus on the the energy consumed while processing data and executing code.

5.3.3.1 Behind the scenes

Not all MCUs are created equal, however this doesn't mean that some are simply "better" than others. There are many complex factors that determine the suitability of a particular MCU for a specific task, one of which is the ability to

	Instructions
PIC16F1827	53
PIC16F688	35
PIC12F675	35
ATtiny25V	120
ATtiny13V	120
MC9S08QG8	145

Table 5.2: Instruction set size for each tested MCU.

Algorithm 5.2 Psudo machine-code representation of a branch instruction.

```
1 load 5 into register 001
2 load danger into register 002
3 branch-if-greater-or-equal 001 002 flee_call
4 call-subroutine fight
5 jump-to continue
6 [flee_call]
7 call-subroutine flee
8 [continue]
```

crunch numbers. This section is intended to give some background in plain-english of the relevant inner workings that affect computational performance.

Algorithm 5.1 Simple C-code representation of a branch instruction.

```
1 if (danger >= 5) flight();
2 else flee();
```

Algorithm 5.1 shows a simplified portion of C-code to demonstrate (very simply) how it may be implemented in various ways. To determine whether or not the outcome of the code is to fight or flee, the processor must first evaluate whether or not the variable 'danger' is greater than or equal to five and either branch to the function 'flight' or continue on to execute the function 'flee'.

Algorithms 5.2 and 5.3 demonstrate two slightly different ways of implementing 5.1 in pseudo machine-code. The decision of which is best is made

Algorithm 5.3 Psudo machine-code representation of an alternative branch instruction.

```
1 load danger into register 001
2 subtract-from-register 001 5
3 branch-if-minus 001 fight_call
4 call-subroutine flee
5 jump-to continue
6 [fight_call]
7 call-subroutine fight
8 [continue]
```

by the compiler, which should take the instructional efficiency of the specific MCU into account. This is an overly simplistic example, but its main point is to illustrate that there are multiple paths that lead to the same result. The important thing to note is that not all of those paths require the same amount of effort. This means that the compiler's ability to optimise code efficiently plays a role in determining the overall performance of the chip. This also means that the programmer shouldn't need to worry too much about instructional efficiency as the compiler should transform C-code into machine code that best suits the target MCU.

Another factor in processing efficiency comes down to the number of different things (or instructions) it can do. All instructions for each MCU are defined in its instruction set. Most 8-bit MCUs are based on reduced instruction set computing (RISC) architecture, as opposed to complex instruction set computing (CISC). This means that when compared CISC based CPU, a RISC based chip is simpler and therefore usually cheaper to produce. "Instruction traces from CISC machines consistently show that few of the available instructions are used in most computing environments"[**?**], meaning that a lot of the added complexity in CISC designs is mostly underutilised. To prevent confusion, a processor with a small instruction set can perform all the same calculations that a processor with a larger instruction set can so they are no less capable in a calculation sense. What this means is that the processor with the reduced instruction set may need to execute multiple instructions in order to achieve the same result as an instruction from another processor which it doesn't have.

The final thing to note when comparing performance is that the clock frequency of an MCU isn't necessarily the frequency at which it performs operations, although sometimes it is.

5.3.3.2 Power consumed while processing

The Microchip PIC16F1827 displayed the lowest energy usage with $10\mu A$ while clocking 7.75kIPS (as shown in figure 5.2). It should be noted that the Microchip MCUs complete one instruction cycle for every four clock cycles, so the 7.75kIPS corresponds to a clock frequency of 31kHz. Also, not all instructions take one instruction cycle to complete (as discussed in section 5.3.3.1). Figure 5.3 shows that the PIC16F688 consumes less power than the PIC16F1827 while at low voltages for the same instruction rates (except at 7.75kIPS). However, there appears to be a flatter response in power consumption with respect to Vdd in the PIC16F1827, similar to what was seen in figure 5.1 with the Atmel chips.The PIC12F675 (as shown in 5.3) used the same at the PIC16F688 (figure 5.3) for its 1MIPS trace. Figures 5.5 and 5.6 show both Atmel MCUs having

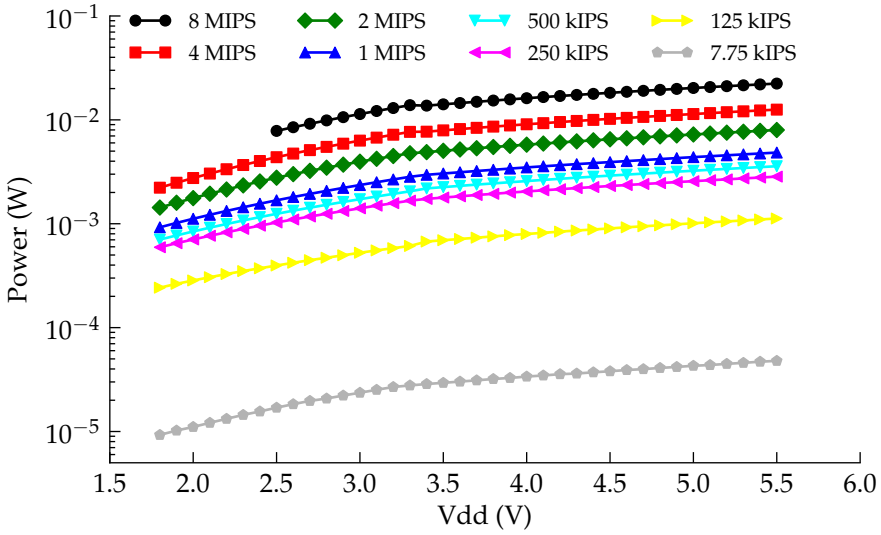

Figure 5.2: Power consumed by the PIC16F1827 while processing

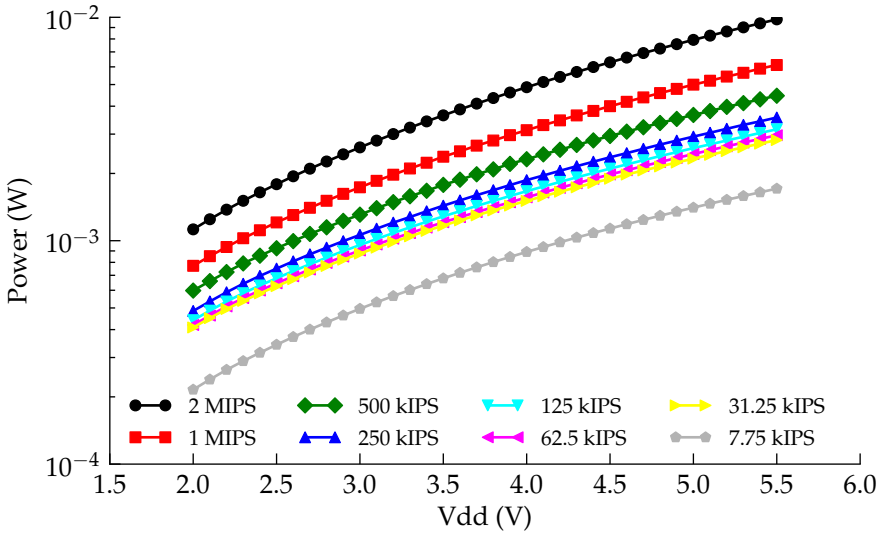

Figure 5.3: Power consumed by the PIC16F688 while processing

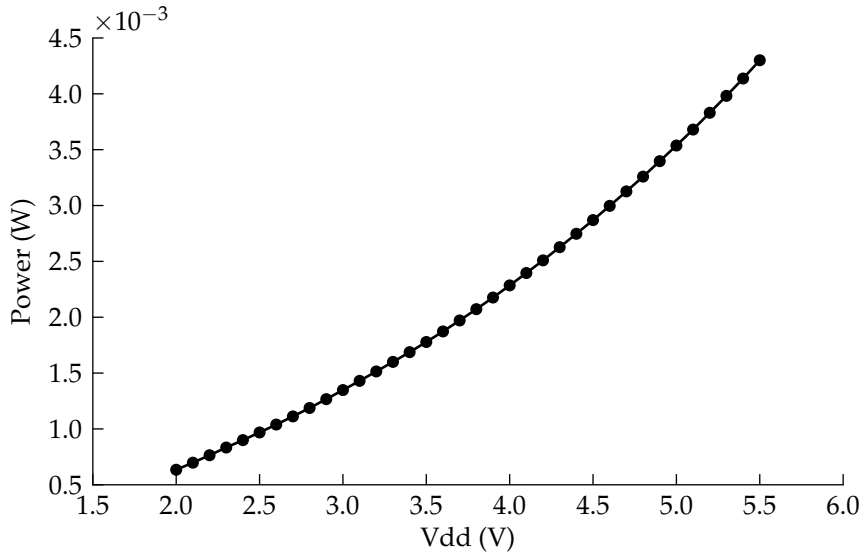

Figure 5.4: Power consumed by the PIC12F675 while processing

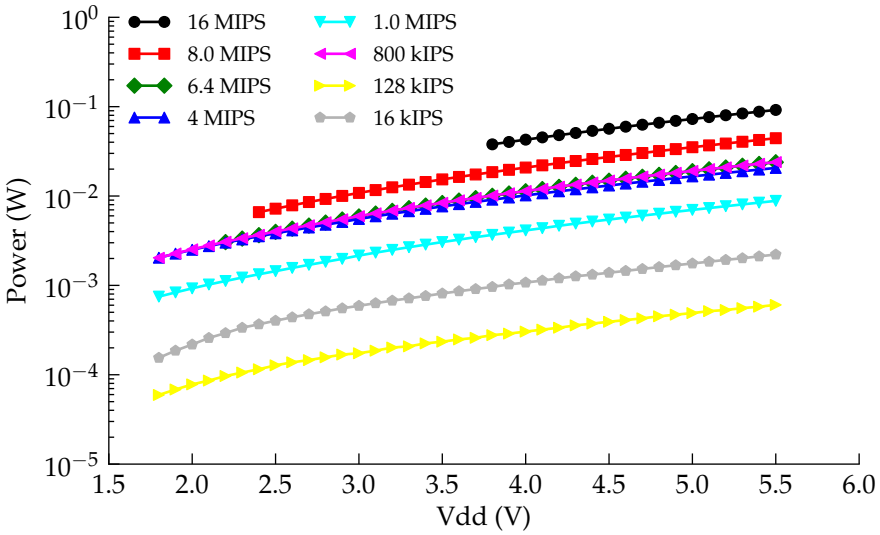

Figure 5.5: Power consumed by the ATtiny25V while processing

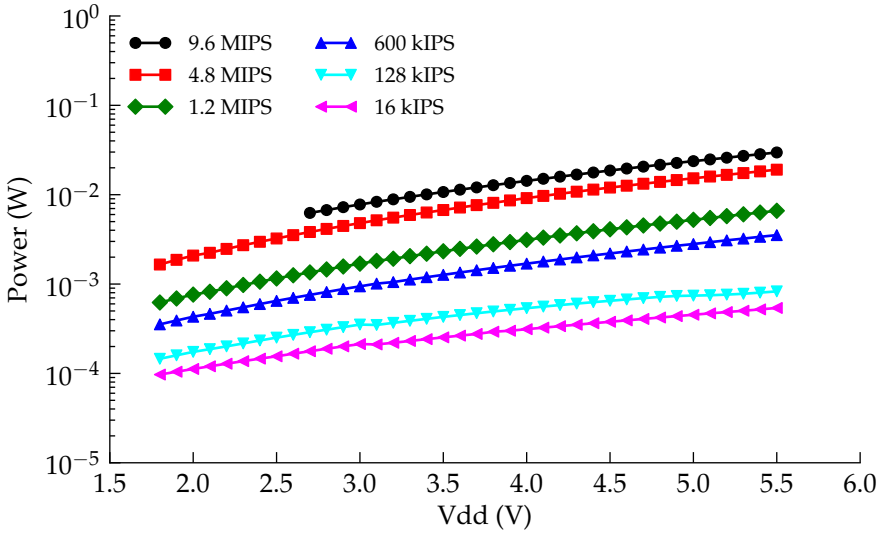

Figure 5.6: Power consumed by the ATtiny13V while processing

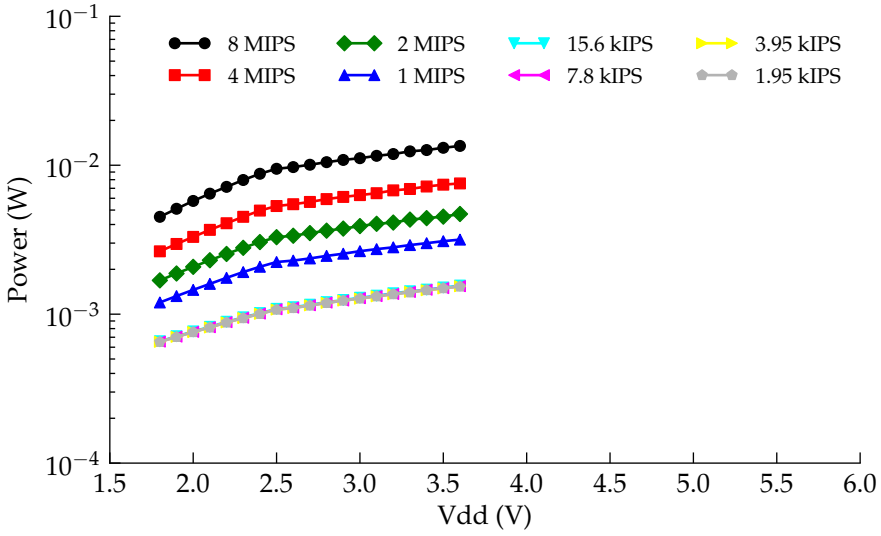

Figure 5.7: Power consumed by the MC9S08QG8 while processing

similar power requirements.The MC9S08QG8, although being able to clock the slowest, performed very poorly at low frequencies. At 1.95kIPS it consumed approximately the same amount of power as the Microchip MCUs operating at 1MIPS.

5.3.3.3 Joules of energy consumed per instruction cycle

A convenient, and more insightful way to interpret the previous processing power consumption graphs is to calculate the energy spent per instruction performed. The energy cost of an instruction cycle can be calculated using equation 5.1.

$$J_i = \frac{I \times Vdd}{f_i} \tag{5.1}$$

where J_i is the number of joules consumed per instruction, I is the current draw, Vdd is the input voltage and f_i is the instruction frequency.

Figure 5.8 compares the most energy efficient operating conditions of each of the tested chips. What is most interesting about this graph is amount of overlap between each of the chips. Also, these greater efficiencies occur at high operating frequencies. A simple rule of thumb for selecting the most power efficient operating frequency based on these results is to choose the highest frequency where the MCU can operate over its full input voltage (Vdd) range. For comparison, figure 5.9 shows the trade-off made when selecting a higher frequency, which is typical across the range of MCUs tested.

5.3.3.4 Performance benchmark

In section 5.3.3.3 I presented a graph showing the amount of energy consumed per instruction cycle for each MCU under ideal operating conditions. But, as discussed in section 5.3.3.1, not all instructions take one instruction cycle to complete. Also, some MCUs have extra instructions that are designed to help speed-up code execution by combining commonly used pairs of instructions. This section will look at how many instruction cycles each MCU takes to complete a benchmarking function. Which will allow a more accurate representation of execution efficiency to be made.

For the benchmarking function I wanted something that would consume a large number of instructions cycles, be well suited to an 8-bit microcontroller (i.e. no complex maths functions and small memory footprint) and have a well defined end point. For this I chose a linear feedback shift register based pseudo-random number generator (as shown in 5.4), for which the code was sourced from [**?**].

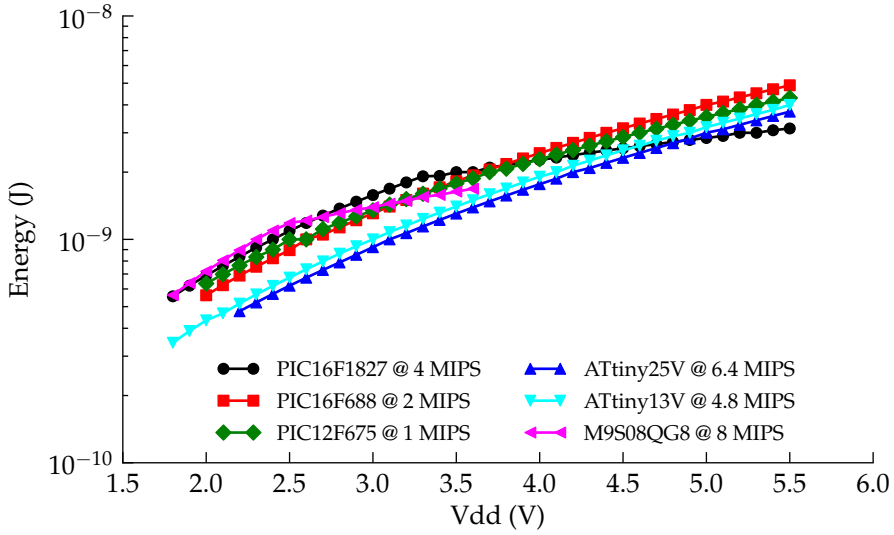

Figure 5.8: Per instruction cycle energy consumption comparison

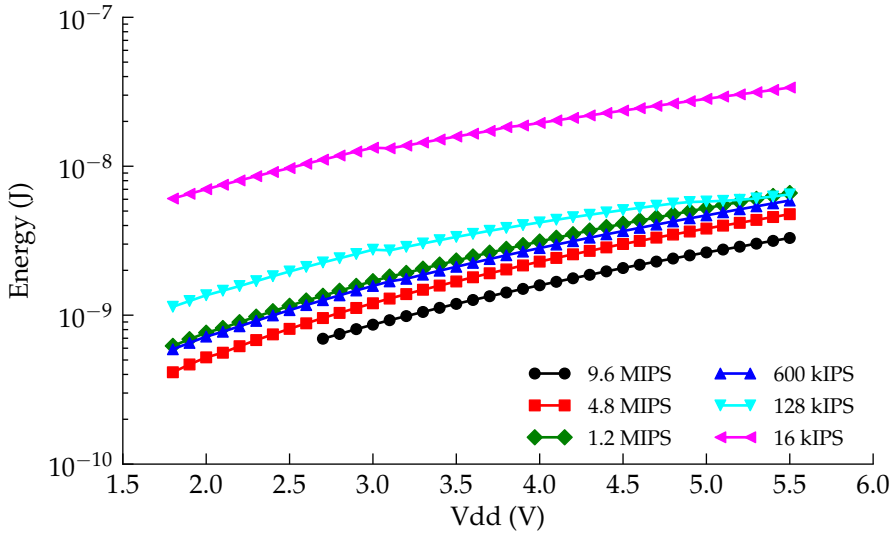

Figure 5.9: Per instruction cycle energy consumption of the ATtiny13V

Algorithm 5.4 Benchmarking algorithm

```
1 unsigned short lfsr = 0xACE1u;
2 unsigned period = 0;
3 do {
4   lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xB400u);
5   ++period;
6 } while(lfsr != 0xACE1u);
```

In short, this function starts with a 16-bit number and runs it through the linear feedback register in a tight loop until the initial value of the 16-bit feedback register is produced again. This function steps through every possible combination of bits possible in a 16-bit number (except zero; 65535) in a pseudo-random order before exiting the loop. The function combines the exclusive-OR (XOR), bit shifting, bitwise AND, increment a value and numerical comparison operations in a tight loop.

The benchmarking function was compiled and run on each of the MCUs operating at a range of frequencies. The amount of time each MCU took to complete the benchmark was timed by watching a pin that toggled on completion of the benchmark function with a Tektronix MSO 4054 Oscilloscope. The number of instruction cycles each chip took to complete the benchmark was deduced by multiplying the time taken to complete the benchmark by the instruction cycle frequency. The results of the benchmark are shown in figure 5.10. It should be noted that to calculate the number of instruction cycles taken by the Microchip family of processors (PIC16F1827, PIC16F688 and PIC12F675) the chip frequency divided by four was used, meaning that the number of clock cycles consumed was four times higher. It is clear from figure 5.10 that the Atmel (ATtiny25V and ATtiny13V) microprocessors are by far the most efficient microprocessor in terms of executing code using a minimum number of instructions of the selection.

5.3.4 Non-volatile memory

In order for a microprocessor to keep information about its current state and recorded data in the event of power loss it must write to non-volatile memory. Non-volatile memory is implemented as either electrically erasable and programmable read only memory (EEPROM) or Flash memory. Flash memory is similar to EEPROM with the exception that it must be erased in large blocks, or pages, before it can be written to. All of the tested MCUs have on-board EEPROM with the exception of the MC9S08QG8 which has flash memory instead. Table 5.1 shows the amount of non-volatile memory space available on each of the chips. The energy consumption of each of the chips with EEPROM memory

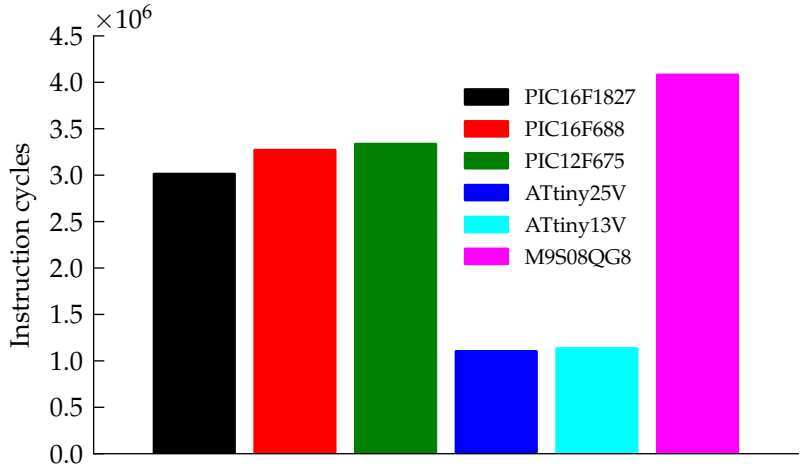

Figure 5.10: MCU comparison of instructions taken to complete a benchmark.

during a 1-byte write operation (which also includes a 1-byte erase operation) is shown in figure 5.11. There appears to be a glitch with the PIC12F675 with respect to it's energy consumption. It's energy consumption was calculated as negative with a Vdd below 4.7V, meaning that it consumed less current while performing write operations than running through the same program loop without performing writes. The measurement was repeated several times and produced the same result but I did not further pursue the cause. I have concluded that this is not representative of the MCU and have disregarded EEPROM measurement data for this chip. In the case of the MC9S08QG8, which has Flash memory instead of EEPROM, the power consumption in the E + W case was calculated per erase/write operation to be $1/512^{th}$ of the page erase energy consumption added to the energy cost of a single write operation, where the W case was the energy cost of a single write operation. In order for the MC9S08QG8 to perform a write operation, the destination byte must have already been pre-erased at an earlier point in time. This may be useful for power harvesting since the energy expensive page erase operation, which consumes an average of 3.02e-4 Joules, can be done performed when available energy is plentiful.

5.3.5 Analog-to-digital conversions.

To measure the flow of water the chosen MCU will likely either count pulses record a voltage level from a flow meter. It is possible that another mechanism such as ultrasonic flow measurement may be used but for now only analog-

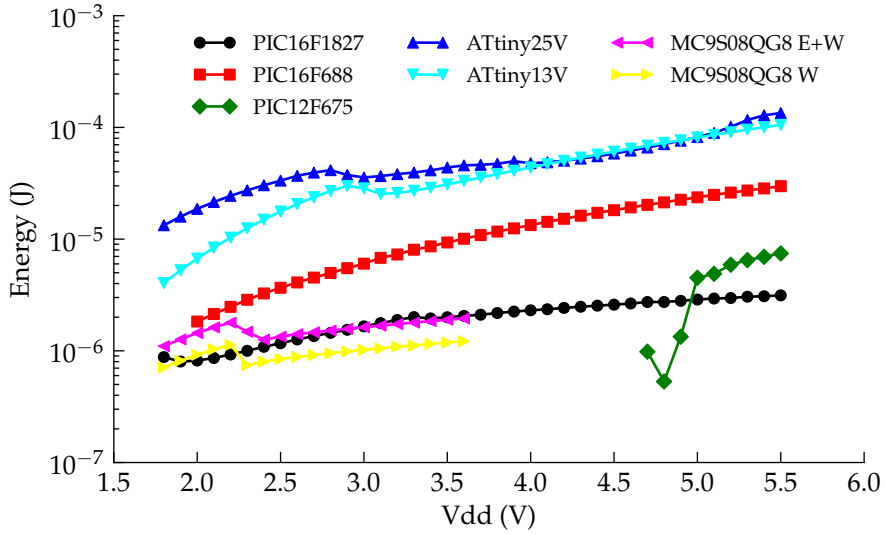

Figure 5.11: Energy consumed by each MCU per non-volatile erase/write operation.

to-digital (ADC) measurements will be measured. For those unfamiliar with an ADC, it is simply a way of converting a voltage (which is generally free to change) into a digital number that a MCU can process and/or store. The power consumption per ADC measurement for each of the tested MCUs is shown in figure 5.12.

5.4 Data Transmission

5.5 Power Consumption

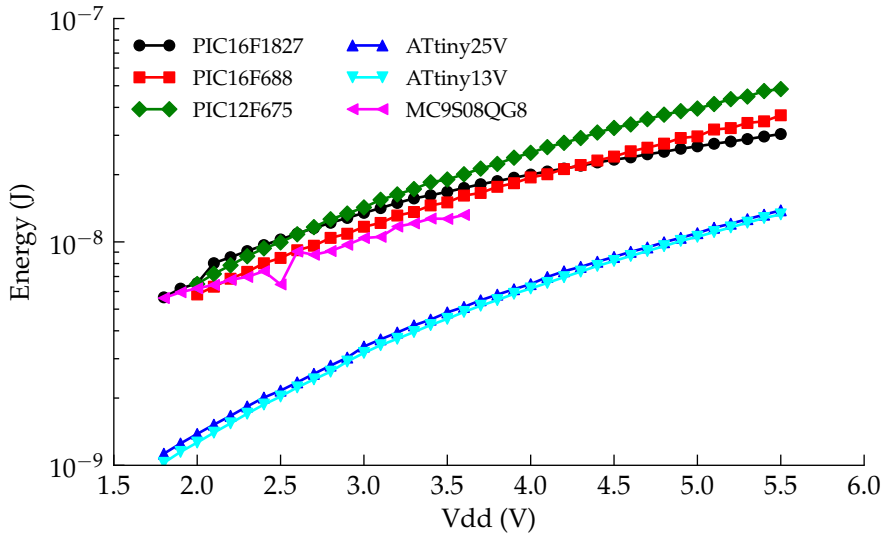

Figure 5.12: Energy consumed by each MCU per ADC measurement.

Figure 5.13: Power consumption of ZigBee

Figure 5.14: Power consumption of RFM12B

Chapter 6
Harvester Design

The main purpose of developing the power harvesting unit outlined in **??** is to operate a wireless sensor node, or mote. In this part I will outline the combinations of hardware and software used to operate the sensing, power management and radio communication elements of the sensor.

6.1 The first prototype

An initial prototype - while not able to energy harvest - provides a starting point for further wireless sensing nodes and sets a baseline for energy performance evaluation. The wireless battery operated sensing node used a Microchip PIC16F886 microcontroller and RFM12B radio transciever, having power requirements as already assesed in **??**. One of the primary purposes of building such a limited prototype was to ensure that I was infact able to build something capable of sensing and relaying data autonomously, which has been achieved. As there is little academic value in the construction of this device, I will talk only briefly about its construction.

Part III

Double layers on conductors

Chapter 7
Electrically Modelling an Interface

7.1 Jonathan Scott's SPICE Model

7.1.1 Resistor Ladder

7.1.2 CPE

7.1.3 Diodes

7.1.4 Memristor

Chapter 8
Determining Interface Parameters

8.1 Scaling of parameters with salinity

8.1.1 Small-signal scaling

8.1.2 CPE scaling

8.1.3 Diode scaling

8.1.4 Discussion

8.2 Biological parameter measurements

8.2.1 CPE in-vivo

8.2.2 CPE butchered sheep

8.2.3 Discussion

8.3 Faradaic Currents

8.3.1 DC Measuement

8.3.2 Discussion

8.4 TO REHOME - RECYCLE THIS

Faradaic processes at an electrode/electrolyte interface are modeled by the Butler-Volmer equation:

$$i_{net} = i_o \left\{ \frac{[O]_{(0,t)}}{[O]_\infty} e^{\frac{-\alpha_c nF\eta}{RT}} - \frac{[R]_{(0,t)}}{[R]_\infty} e^{\frac{(1-\alpha_c)nF\eta}{RT}} \right\} \tag{8.1}$$

where:

i_{net} = net Faradaic current

i_o = current exchange density

$[O]_{(0,t)}$ = concentration of oxidising species at electrode surface $(x = 0)$

$[R]_{(0,t)}$ = concentration of reducing species at the electrode surface

$[O]_\infty$ = bulk concentration of oxidising species in electrolyte

$[R]_\infty$ = bulk concentration of oxidising species in electrolyte

α_c = cathodic transfer coefficient

n = number of moles of electrons per mole of reactant oxidised

F = Faraday's constant $\approx 96,485$

R = the gas constant ≈ 8.314

T = absolute temperature in Kelvin

η = electrode overpontential

Part IV

Appendices

Chapter A
Alternative Energy Harvester

In 1867 William Thomson (Lord Kelvin) described an apparatus that could generate electrostatic charge using drops of water (some of which are shown in Figure A.1) [**?**].

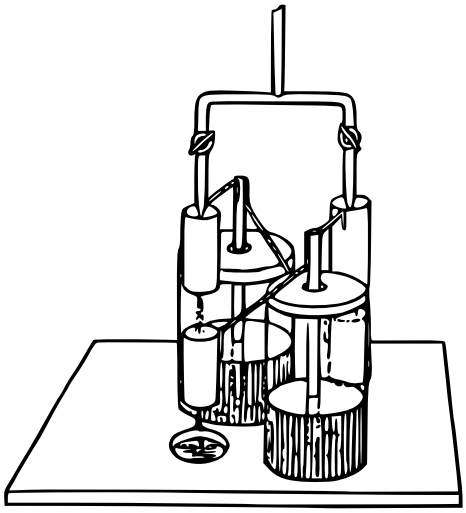

Figure A.1: Drawing of Lord Kelvin's electrostatic generator.

This apparatus works by inducing charge onto drops of water before they detached from the source of the drips. It is this apparatus that forms the basis for the investigation of the of charged drops of water to generate useful amount of power.

A.1 How charge is generated

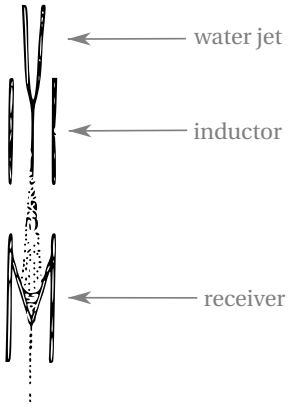

Figure A.2: Drawing of the charging mechanism for Lord Kelvin's electrostatic generator.

Figure A.2 shows one of the charge generating mechanisms of Lord Kelvin's electrostatic generator. This mechanism is comprised of three main components, a jet of water that breaks to form droplets of water, an inductor surrounding the point at which the stream of water breaks and a receiver where the charged water droplets are condensed.

A simplified diagram of the apparatus is shown in Figure A.3 and will be used to explain the concept further.

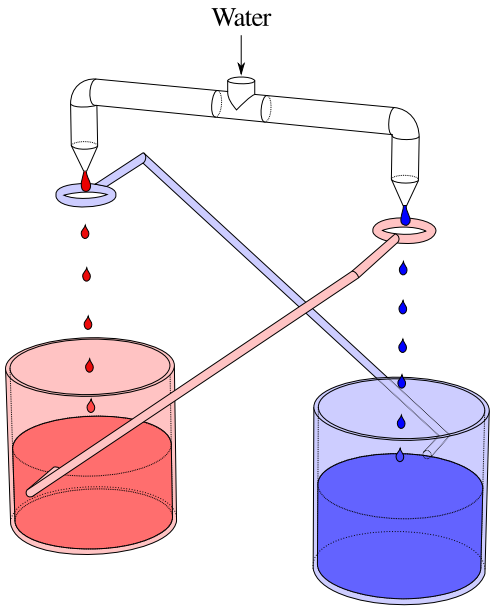

Figure A.3: Simplified diagram of Lord Kelvin's water dropper configuration.

The diagram illustrates the charge of each element by its colour. Equal and opposite amounts of electrical charge are accumulated in each of the containers at the bottom. These containers are insulated from one another but are connected to the rings (near the droppers) of the opposite side. The rings are responsible for inducing excess charge on the drops (equivalent to the cylinder shaped inductor depicted in Figure A.2), which becomes trapped on the drop once it detaches from the dropper. Once detached, the drops are pulled by gravity into the containers below. Because the container below is charged with like charges there is a repulsive force between the falling drop and the container. It is this force acting over the distance through which the drop falls that is responsible for doing work (forcing the drop onto a body that has the same net polarity of as the drop) and thus increasing the voltage of the container/ring.

A.2 Replicating the experiment

The first step toward assessing the electrostatic generators ability to generate energy for the water meter was to replicate the experiment. This work was done by a Summer Research Scholarship student Jonathon McMullen.

A.3 Optimising output

Once the experiment had been replicated, questions arose regarding how to best optimise the design.

A.3.1 Drop volume and frequency

The first optimisation question in its most practical form was "is it better to have lots of small drops or less larger drops?". To answer this question a more simplified experiment was performed with the help of another summer research student, Wayne Crump. The purpose of this experiment was to remove as many variables from the electrostatic generator experiment performed by Jonathan McMullen (see subsection A.2) in an attempt to isolate the effect of varying drop size, induction voltage and flow rate, independently.

A.3.1.1 Experimental setup

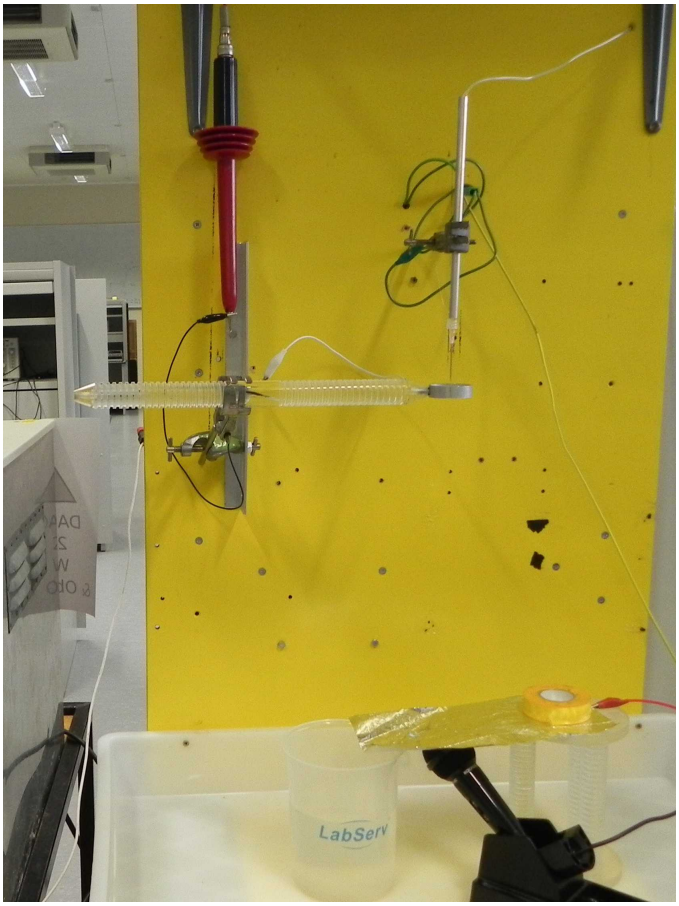

Figure A.4: Photo of experimental setup for charge on drip measurements.

A photo of the measurement setup can be seen in Figure A.4. For clarity, a diagram of the experimental setup is shown in Figure A.5. As can be seen in this diagram, drips are formed from the end of a syringe needle, pass through the induction ring and land on a piece of tin foil before running off into a catchment container. A closeup of the needle and inductor is shown in Figure A.3.1.1. The frequency of drips is determined acoustically by the use of the microphone and the flow of water is set by the syringe pump (as shown in Figure A.7).

The volume of each drip is calculate by dividing the flow rate by the drip frequency. The charge on each drop of water was determined by dividing the average current by the drip frequency. The size of drips was determined largely by the size of needle used but was also affected by the magnitude of the induction voltage placed on the ring.

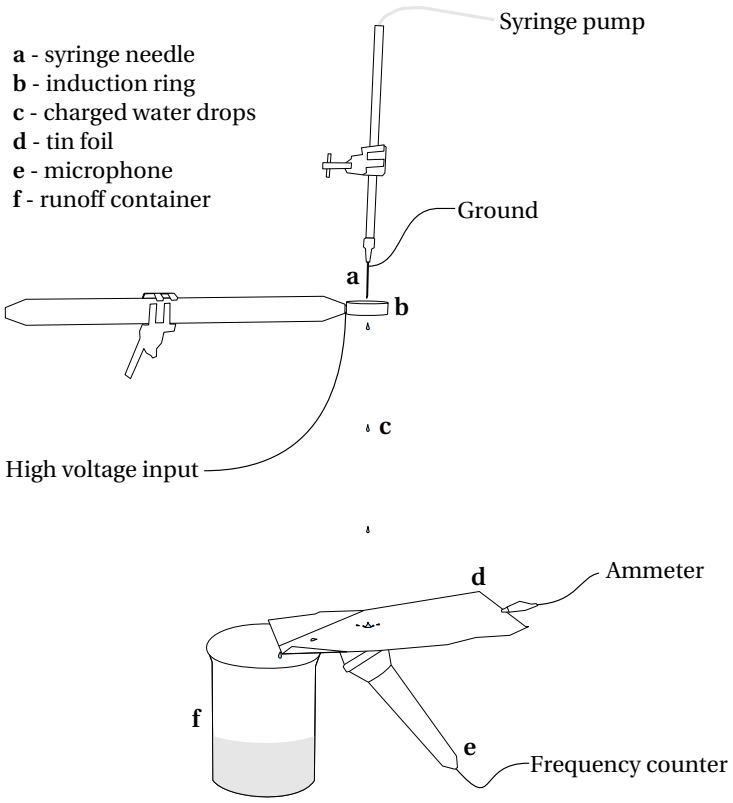

Figure A.5: Diagram of experimental setup for charge on drip experiments.

Because the current being measured was so small (in the nano-amp range), direct measurement with a multimeter was not possible. Instead the multimeter (a Fluke series 115 handheld multimeter) was set to measure voltage in the mV range. In this setting the multimeter has an internal resistance of 10 $M\Omega$ (as measured with another multimeter) over which the voltage was being measured. Ohms law states:

$$V = IR \tag{A.1}$$

Where V is voltage, I is current and R is resistance. Rearranging this equation and substituting in our resistance of 10 $M\Omega$ we get

$$I = \frac{V}{10 \times 10^6} \tag{A.2}$$

Where V is the output of the multimeter and I is the current through the multimeter, through which we assume all charge collected on the tin foil will flow.

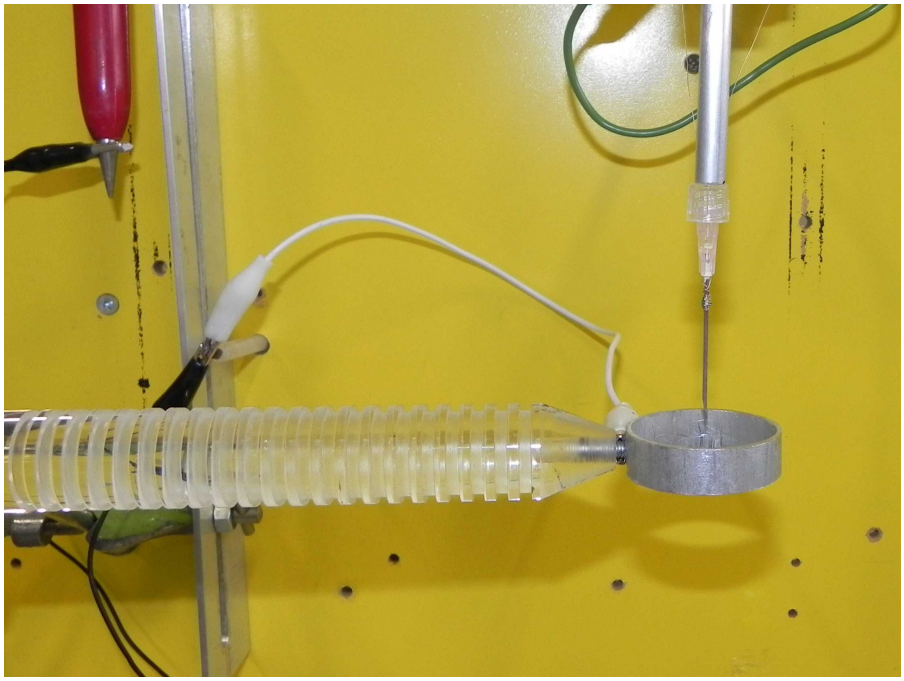

Figure A.6: Photo of the dropper and high voltage inductor.

Figure A.7: Syringe pump used to produce drops and control flow rate.

A.3.1.2 Results

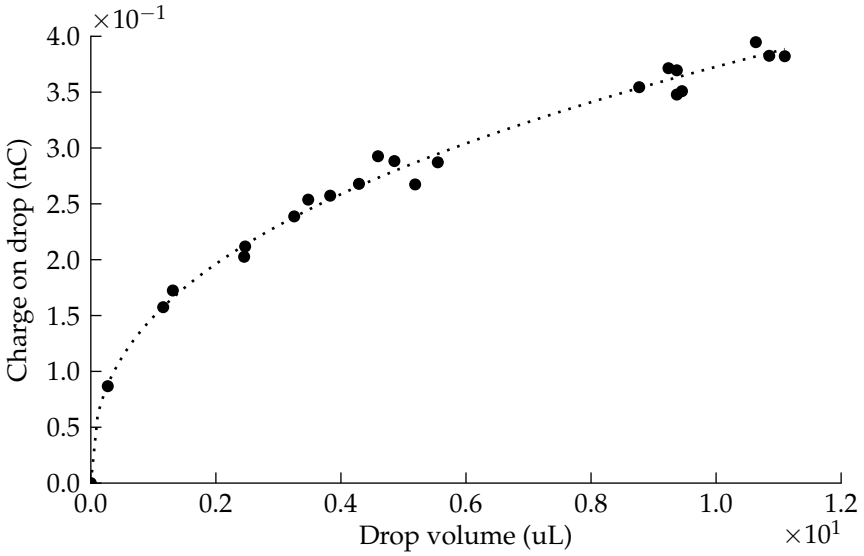

Figure A.8: Charge on drip versus drip volume for a fixed induction voltage of 2.5 kV.

Figure A.8 shows the effect that changing the volume of each drop has on the amount of charge bound to that drop. The shape of this curve is similar to what one gets when plotting the surface area of a sphere against the volume of a sphere. This was expected to be the case since excess charge will position itself evenly over the surface of the drop and hence be proportional to its surface area. If the same data is plotted in terms of charge per volume versus volume of a drip, as is shown in Figure A.9, it is evident that smaller drop sizes equate to a higher charge per volume of water ratio.

Figure A.10 shows the results of changing the induction voltage on the average charge carried per drop. These results indicate that the charge induced on a drop is proportional to the induction voltage. Variation in the measurement data was due to differences in room temperature.

Figure A.11 shows the effect of increasing the flow on the output current of the measurement setup, which is relatively linear.

A.3.1.3 Conclusion & discussion

Increasing the output of the initial apparatus is possible by doing any of the following things:

- Favour smaller drops at higher frequencies.

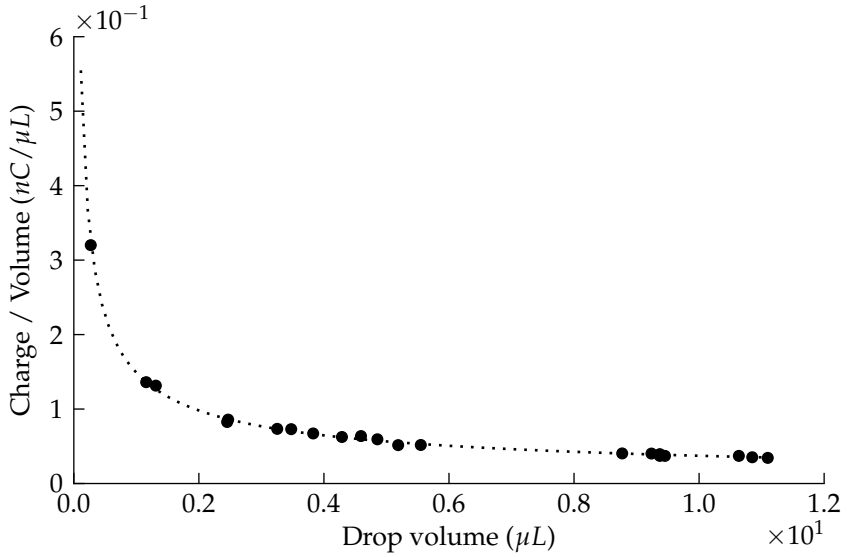

Figure A.9: Charge per volume versus drop volume for a fixed induction voltage of 2.5 kV.

- Increase the total flow of water through the apparatus.
- Increase the induction voltage.

Increasing the total flow through the apparatus is difficult to achieve as this soon has adverse effects on the formation of drops. This usually leads to the formation of a stream of water from the needle, reducing the output as the drips aren't forming within the inductor.

Favouring small drops or increasing the induction voltage both increase the charge to mass ratio of individual drops. As this ratio increases the movement of the drops becomes increasingly dominated by electrostatic forces between the container and the drop. As this ratio increases so to does the tendency for drops to diverge from the path dictated by gravity. Eventually there comes a point where the drops no-longer fall and instead create a stream that loops to the nearest point on the induction ring. For this reason it is important that drop sizes are kept consistent as to maintain a stable charge to mass ratio to which the design can be optimised for.

A.3.2 Scale

Due to the size of the apparatus the fields generated had to be relatively large, requiring equally large voltages before useful amounts of power is generated. There are two problems with generating power at such large voltages:

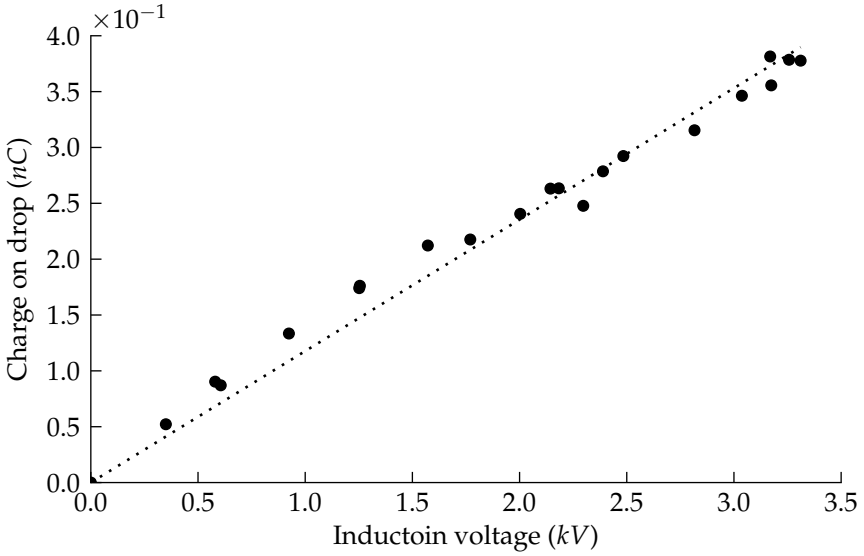

Figure A.10: Charge on drip versus ring induction voltage for a fixed drip volume.

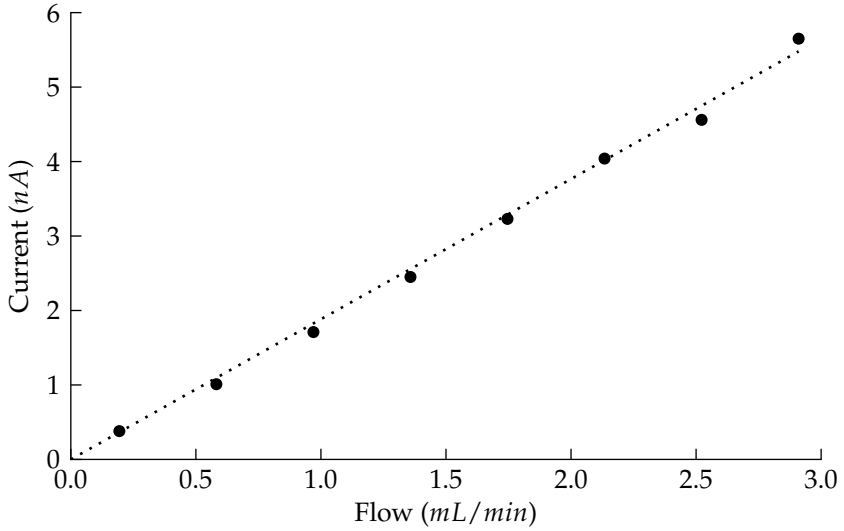

Figure A.11: Current versus flow rate for an induction voltage of 3.8 kV and drop volume of 3.1 μL.

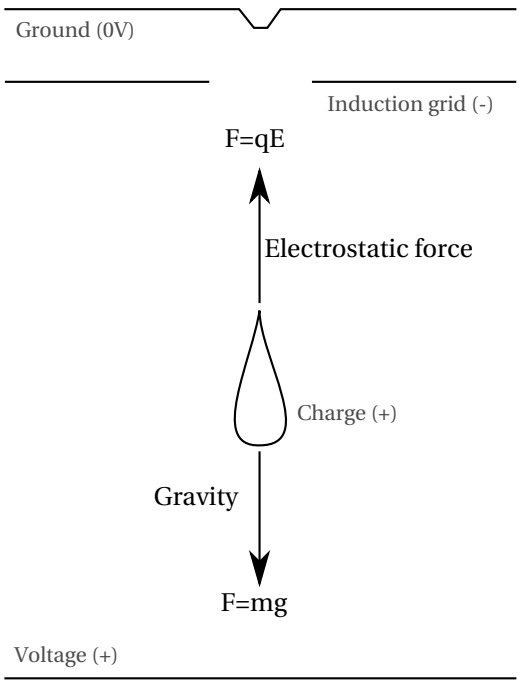

Figure A.12: Diagram of the electrostatic force and gravity acting on a drop of water.

1. Large voltage differences try very hard to neutralise each other. This means that the geometry and materials used may need to be modified in order to reduce charge migrating over insulating barriers.

2. Stepping these voltages down to a useful level is challenging and would be very inefficient. Electronic components and configurations that would usually be used to change voltage levels no longer work.

Scaling down the design would allow the device to generate its maximum practical output at voltages that are easier to deal with.

A.3.2.1 Investigation:

Figure A.12 shows a simplified diagram of the two dominant forces acting upon a charged drop as it falls towards a like charged plate from a grounded plane. These two forces are given by the following equations:

$$F_{down} = mg \tag{A.3}$$

$$F_{up} = qE \tag{A.4}$$

Where m is the mass of the drop, g is the acceleration due to gravity, q is the charge on the drop and E is the electric field strength.

Chapter B
Microprocessor Power Measurements

B.1 Measurement scripts

Energy consumption measurements were made using an Agilent E5270B 8-Slot Precision Measurement Mainframe, a Tektronix MSO 4054 Mixed Signal Oscilloscope and a desktop PC. Operation of the E5270 was done via GLIB interface using a USB connection and Agilent IO Libraries Version 16.0.1458.0. From here the machine was interfaced using custom Python scripts (appended) and PyVisa (available from http://pyvisa.sourceforge.net/pyvisa/).

B.1.1 E5270 Scripts

Measurement of chip energy consumption was carried out using the following measurement script. This script is written in Python and was executed using PyLab from the Enthought Python bundle.

B.2 Measurement data

It should be noted that the Freescale M9S08QG8 and the Microchip PIC16F1827 were unable to boot reliably into a low power state at 1.8V. To prevent this from happening the chips were booted at 2.0V and lowered to 1.8V to prevent the chips entering a state where current consumption was in the hundreds of microamps range.

B.2.1 Chips sleeping

The following tables list the unprocessed current measurements for a sweep of Vdd from 1.8V to 5.5V. Sweeps have been restricted to the input voltage ranges as specified in each chips datasheet.

Vdd	Tiny13V	Tiny25V	12F675	16F1827	M9S08QG8
1.80	3.41E-08	9.27E-08	N/A	9.914E-07	3.82E-04
1.90	4.04E-08	9.30E-08	N/A	1.008E-06	3.86E-04
2.00	4.73E-08	9.33E-08	4.497E-10	1.023E-06	4.88E-07
2.10	5.47E-08	9.36E-08	4.677E-10	1.038E-06	4.89E-07
2.20	6.26E-08	9.39E-08	4.797E-10	1.054E-06	4.91E-07
2.30	7.10E-08	9.42E-08	4.907E-10	1.068E-06	4.92E-07
2.40	7.99E-08	9.45E-08	5.057E-10	1.087E-06	4.94E-07
2.50	8.92E-08	9.47E-08	5.210E-10	1.102E-06	4.96E-07
2.60	9.90E-08	9.50E-08	5.347E-10	1.119E-06	4.99E-07
2.70	1.09E-07	9.53E-08	5.487E-10	1.137E-06	5.02E-07
2.80	1.20E-07	9.56E-08	5.610E-10	1.166E-06	5.05E-07
2.90	1.31E-07	9.59E-08	5.760E-10	1.195E-06	5.10E-07
3.00	1.42E-07	9.63E-08	5.903E-10	1.234E-06	5.15E-07
3.10	1.54E-07	9.66E-08	6.057E-10	1.290E-06	5.21E-07
3.20	1.67E-07	9.70E-08	6.217E-10	1.300E-06	5.28E-07
3.30	1.79E-07	9.74E-08	6.397E-10	1.301E-06	5.37E-07
3.40	1.92E-07	9.78E-08	6.590E-10	1.302E-06	5.47E-07
3.50	2.06E-07	9.82E-08	6.763E-10	1.301E-06	5.60E-07
3.60	2.20E-07	9.87E-08	6.970E-10	1.304E-06	5.75E-07
3.70	2.35E-07	9.93E-08	7.200E-10	1.307E-06	N/A
3.80	2.49E-07	1.00E-07	7.437E-10	1.309E-06	N/A
3.90	2.65E-07	1.01E-07	7.703E-10	1.311E-06	N/A
4.00	2.80E-07	1.02E-07	7.960E-10	1.312E-06	N/A
4.10	2.97E-07	1.03E-07	8.273E-10	1.317E-06	N/A
4.20	3.13E-07	1.04E-07	8.690E-10	1.319E-06	N/A
4.30	3.31E-07	1.05E-07	9.033E-10	1.330E-06	N/A
4.40	3.48E-07	1.07E-07	9.487E-10	1.334E-06	N/A
4.50	3.67E-07	1.09E-07	9.977E-10	1.342E-06	N/A
4.60	3.86E-07	1.11E-07	1.056E-09	1.351E-06	N/A
4.70	4.05E-07	1.13E-07	1.117E-09	1.360E-06	N/A
4.80	4.25E-07	1.16E-07	1.197E-09	1.373E-06	N/A
4.90	4.46E-07	1.19E-07	1.296E-09	1.385E-06	N/A
5.00	4.67E-07	1.23E-07	1.403E-09	1.401E-06	N/A
5.10	4.90E-07	1.27E-07	1.513E-09	1.422E-06	N/A
5.20	5.13E-07	1.32E-07	1.661E-09	1.445E-06	N/A
5.30	5.37E-07	1.38E-07	1.811E-09	1.469E-06	N/A
5.40	5.62E-07	1.45E-07	1.975E-09	1.497E-06	N/A
5.50	5.88E-07	1.53E-07	2.164E-09	1.529E-06	N/A

Table B.1: Raw sleep measurements (Vdd 1.8V – 5.5V)

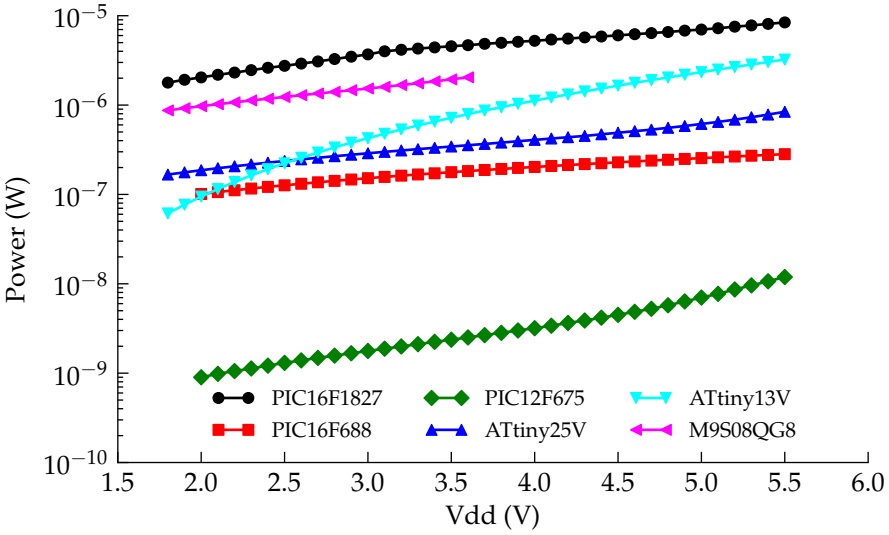

Figure B.1: Power usage of each microprocessor while in sleep mode

Figure 5.1 shows the amount of current drawn by each of the microprocessors in sleep mode. Similarly, figure B.1 shows the power consumption of each of the chips in sleep mode.

B.2.2 Clocking

B.2.2.1 ATtiny13V

B.2.2.2 ATtiny25V

B.2.2.3 Microchip PIC16F1827

B.3 Microprocessor Test Code

B.3.1 ATMEL ATtiny13V and ATtiny25

Code was written in AVRStudio 4.18 (build 684) and compiled using WinAVR (AVR-GCC compiler for windows available from http://winavr.sourceforge.net/. Chip programming was done using an Atmel AVR STK500 demonstration board with serial interface.

B.3.1.1 Sleep

Programming fuses were all disabled (Watchdog, brown-out detect, clock divider) and clock selection was set to 'Int. RC Osc 128kHz; Start-up time; 14 CK

Vdd	9.6 MHz	4.8 MHz	1.2 MHz	600 kHz	128 kHz	16 kHz
1.80	N/A	N/A	3.900E-04	2.148E-04	8.098E-05	5.393E-05
1.90	N/A	N/A	4.121E-04	2.258E-04	8.387E-05	5.499E-05
2.00	N/A	N/A	4.242E-04	2.329E-04	8.708E-05	5.608E-05
2.10	N/A	N/A	4.429E-04	2.421E-04	8.897E-05	5.721E-05
2.20	N/A	N/A	4.634E-04	2.523E-04	9.126E-05	5.839E-05
2.30	N/A	N/A	4.859E-04	2.632E-04	9.415E-05	5.964E-05
2.40	N/A	N/A	5.088E-04	2.748E-04	9.738E-05	6.099E-05
2.50	N/A	N/A	5.323E-04	2.865E-04	1.006E-04	6.214E-05
2.60	N/A	N/A	5.557E-04	2.984E-04	1.038E-04	6.396E-05
2.70	2.759E-03	1.664E-03	5.799E-04	3.108E-04	1.071E-04	6.560E-05
2.80	2.879E-03	1.742E-03	6.065E-04	3.235E-04	1.102E-04	6.743E-05
2.90	3.001E-03	1.834E-03	6.384E-04	3.366E-04	1.136E-04	6.926E-05
3.00	3.116E-03	1.945E-03	6.705E-04	3.540E-04	1.174E-04	7.100E-05
3.10	3.231E-03	2.022E-03	6.876E-04	3.582E-04	1.124E-04	6.809E-05
3.20	3.358E-03	2.109E-03	7.144E-04	3.710E-04	1.150E-04	6.878E-05
3.30	3.486E-03	2.199E-03	7.421E-04	3.843E-04	1.175E-04	6.996E-05
3.40	3.619E-03	2.295E-03	7.700E-04	3.979E-04	1.200E-04	7.115E-05
3.50	3.757E-03	2.394E-03	7.976E-04	4.114E-04	1.226E-04	7.235E-05
3.60	3.905E-03	2.491E-03	8.269E-04	4.251E-04	1.250E-04	7.354E-05

Table B.2: Atmel ATtiny13V clocking current (Vdd 1.8V – 3.6V).

Vdd	9.6 MHz	4.8 MHz	1.2 MHz	600 kHz	128 kHz	16 kHz
3.70	4.046E-03	2.580E-03	8.555E-04	4.392E-04	1.275E-04	7.471E-05
3.80	4.185E-03	2.661E-03	8.854E-04	4.535E-04	1.298E-04	7.706E-05
3.90	4.322E-03	2.740E-03	9.156E-04	4.685E-04	1.321E-04	7.701E-05
4.00	4.464E-03	2.822E-03	9.467E-04	4.834E-04	1.343E-04	7.814E-05
4.10	4.606E-03	2.900E-03	9.780E-04	4.990E-04	1.365E-04	7.929E-05
4.20	4.753E-03	2.932E-03	1.010E-03	5.152E-04	1.386E-04	8.045E-05
4.30	4.901E-03	2.991E-03	1.043E-03	5.313E-04	1.406E-04	8.163E-05
4.40	5.061E-03	3.001E-03	1.077E-03	5.491E-04	1.425E-04	8.283E-05
4.50	5.219E-03	2.987E-03	1.110E-03	5.655E-04	1.444E-04	8.406E-05
4.60	5.389E-03	3.073E-03	1.144E-03	5.833E-04	1.462E-04	8.531E-05
4.70	5.547E-03	3.162E-03	1.178E-03	6.002E-04	1.480E-04	8.661E-05
4.80	5.710E-03	3.257E-03	1.213E-03	6.216E-04	1.498E-04	8.795E-05
4.90	5.872E-03	3.348E-03	1.248E-03	6.395E-04	1.502E-04	8.932E-05
5.00	6.046E-03	3.443E-03	1.285E-03	6.567E-04	1.484E-04	9.072E-05
5.10	6.228E-03	3.548E-03	1.320E-03	6.746E-04	1.470E-04	9.213E-05
5.20	6.391E-03	3.647E-03	1.357E-03	6.940E-04	1.465E-04	9.346E-05
5.30	6.575E-03	3.753E-03	1.394E-03	7.134E-04	1.469E-04	9.482E-05
5.40	6.753E-03	3.857E-03	1.434E-03	7.331E-04	1.485E-04	9.640E-05
5.50	6.955E-03	3.970E-03	1.471E-03	7.526E-04	1.506E-04	9.818E-05

Table B.3: Atmel ATtiny13V clocking current (Vdd 3.7V – 5.5V).

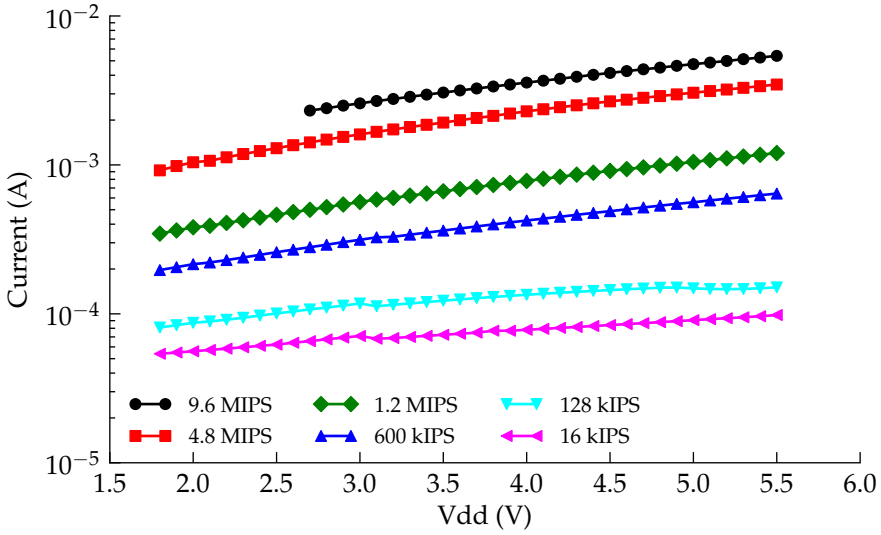

Figure B.2: Current consumption of the Atmel ATtiny13V while clocking

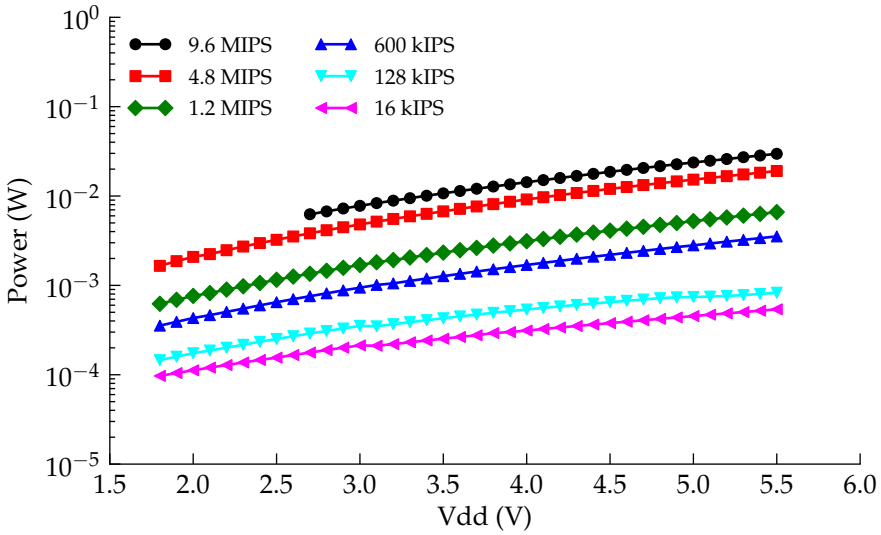

Figure B.3: Power consumption of the Atmel ATtiny13V while clocking

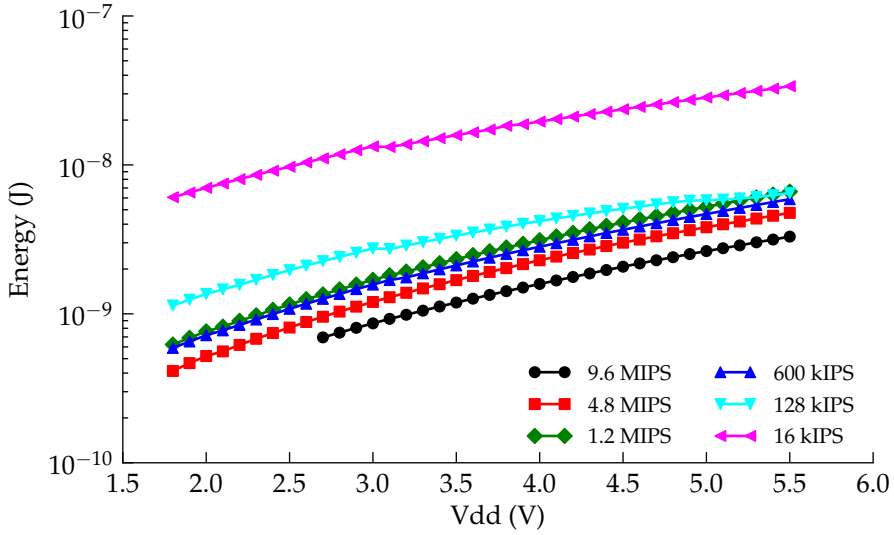

Figure B.4: Energy consumed per instruction for the Atmel ATtiny13V

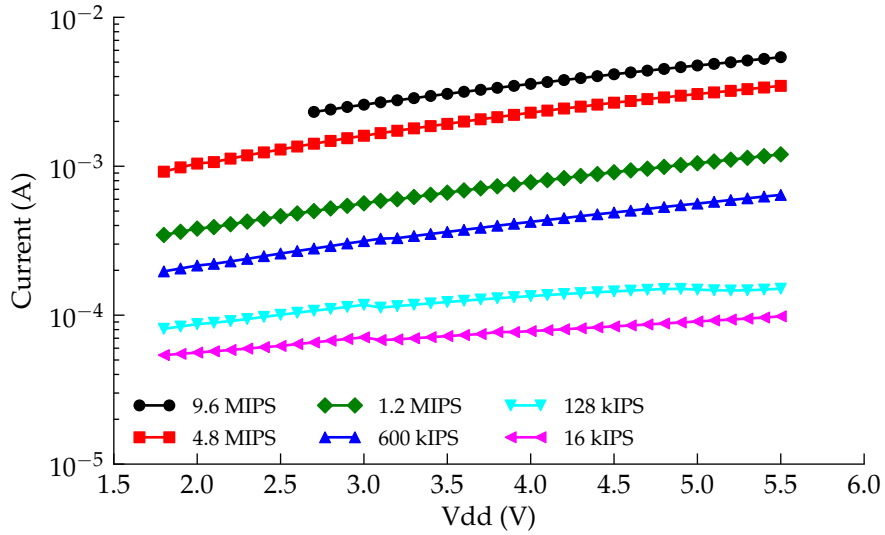

Figure B.5: Current consumption of the Atmel ATtiny25V while clocking

Vdd	16 MHz	8 MHz	6.4 MHz	2 MHz	1 MHz	800 kHz	128 kHz	16 kHz
1.80	N/A	N/A	N/A	1.148E-03	4.106E-04	1.146E-03	7.622E-05	1.666E-04
1.90	N/A	N/A	N/A	1.205E-03	4.319E-04	1.209E-03	7.858E-05	1.737E-04
2.00	N/A	N/A	N/A	1.261E-03	4.531E-04	1.274E-03	8.121E-05	1.813E-04
2.10	N/A	N/A	N/A	1.320E-03	4.754E-04	1.339E-03	8.464E-05	1.903E-04
2.20	N/A	N/A	N/A	1.378E-03	4.984E-04	1.404E-03	8.981E-05	1.994E-04
2.30	N/A	N/A	N/A	1.434E-03	5.195E-04	1.471E-03	9.486E-05	2.066E-04
2.40	N/A	N/A	N/A	1.491E-03	5.413E-04	1.537E-03	9.642E-05	2.126E-04
2.50	N/A	N/A	N/A	1.540E-03	5.671E-04	1.604E-03	9.903E-05	2.193E-04
2.60	N/A	N/A	N/A	1.597E-03	5.912E-04	1.671E-03	1.008E-04	2.257E-04
2.70	6.283E-03	3.124E-03	1.765E-03	1.651E-03	6.163E-04	1.740E-03	1.030E-04	2.320E-04
2.80	6.551E-03	3.256E-03	1.842E-03	1.706E-03	6.425E-04	1.811E-03	1.053E-04	2.378E-04
2.90	6.826E-03	3.392E-03	1.916E-03	1.766E-03	6.683E-04	1.883E-03	1.072E-04	2.414E-04
3.00	7.097E-03	3.537E-03	1.995E-03	1.832E-03	6.963E-04	1.959E-03	1.094E-04	2.434E-04
3.10	7.380E-03	3.682E-03	2.081E-03	1.906E-03	7.284E-04	2.050E-03	1.118E-04	2.440E-04
3.20	7.695E-03	3.825E-03	2.172E-03	1.973E-03	7.623E-04	2.131E-03	1.143E-04	2.469E-04
3.30	7.996E-03	3.975E-03	2.253E-03	2.036E-03	7.925E-04	2.210E-03	1.170E-04	2.521E-04
3.40	8.288E-03	4.122E-03	2.340E-03	2.101E-03	8.216E-04	2.286E-03	1.190E-04	2.587E-04
3.50	8.605E-03	4.271E-03	2.425E-03	2.165E-03	8.516E-04	2.371E-03	1.208E-04	2.657E-04
3.60	8.915E-03	4.421E-03	2.510E-03	2.230E-03	8.824E-04	2.452E-03	1.222E-04	2.732E-04

Table B.4: Atmel ATtiny25V clocking current (Vdd 3.7V – 5.5V)

Vdd	16 MHz	8 MHz	6.4 MHz	2 MHz	1 MHz	800 kHz	128 kHz	16 kHz
3.70	9.232E-03	4.582E-03	2.599E-03	2.298E-03	9.122E-04	2.535E-03	1.235E-04	2.805E-04
3.80	9.550E-03	4.754E-03	2.687E-03	2.365E-03	9.436E-04	2.622E-03	1.246E-04	2.867E-04
3.90	9.908E-03	4.921E-03	2.776E-03	2.436E-03	9.746E-04	2.706E-03	1.258E-04	2.950E-04
4.00	1.025E-02	5.077E-03	2.864E-03	2.506E-03	1.006E-03	2.795E-03	1.269E-04	2.951E-04
4.10	1.057E-02	5.245E-03	2.962E-03	2.578E-03	1.040E-03	2.882E-03	1.280E-04	3.015E-04
4.20	1.092E-02	5.422E-03	3.054E-03	2.652E-03	1.074E-03	2.970E-03	1.278E-04	3.085E-04
4.30	1.128E-02	5.607E-03	3.153E-03	2.724E-03	1.108E-03	3.062E-03	1.279E-04	3.159E-04
4.40	1.161E-02	5.783E-03	3.250E-03	2.795E-03	1.143E-03	3.155E-03	1.270E-04	3.235E-04
4.50	1.198E-02	5.958E-03	3.348E-03	2.873E-03	1.177E-03	3.248E-03	1.266E-04	3.313E-04
4.60	1.235E-02	6.139E-03	3.445E-03	2.949E-03	1.210E-03	3.340E-03	1.254E-04	3.390E-04
4.70	1.272E-02	6.312E-03	3.549E-03	3.027E-03	1.248E-03	3.430E-03	1.240E-04	3.468E-04
4.80	1.311E-02	6.501E-03	3.653E-03	3.105E-03	1.284E-03	3.528E-03	1.230E-04	3.547E-04
4.90	1.349E-02	6.689E-03	3.757E-03	3.190E-03	1.319E-03	3.633E-03	1.229E-04	3.625E-04
5.00	1.387E-02	6.898E-03	3.860E-03	3.270E-03	1.358E-03	3.732E-03	1.237E-04	3.703E-04
5.10	1.427E-02	7.099E-03	3.965E-03	3.356E-03	1.396E-03	3.832E-03	1.248E-04	3.781E-04
5.20	1.464E-02	7.302E-03	4.075E-03	3.437E-03	1.432E-03	3.931E-03	1.262E-04	3.857E-04
5.30	1.506E-02	7.502E-03	4.181E-03	3.527E-03	1.471E-03	4.033E-03	1.277E-04	3.930E-04
5.40	1.547E-02	7.713E-03	4.297E-03	3.616E-03	1.512E-03	4.142E-03	1.293E-04	3.999E-04
5.50	1.588E-02	7.908E-03	4.415E-03	3.705E-03	1.552E-03	4.245E-03	1.311E-04	4.063E-04

Table B.5: Atmel ATtiny25V clocking current (VDD 3.7V – 5.5V)

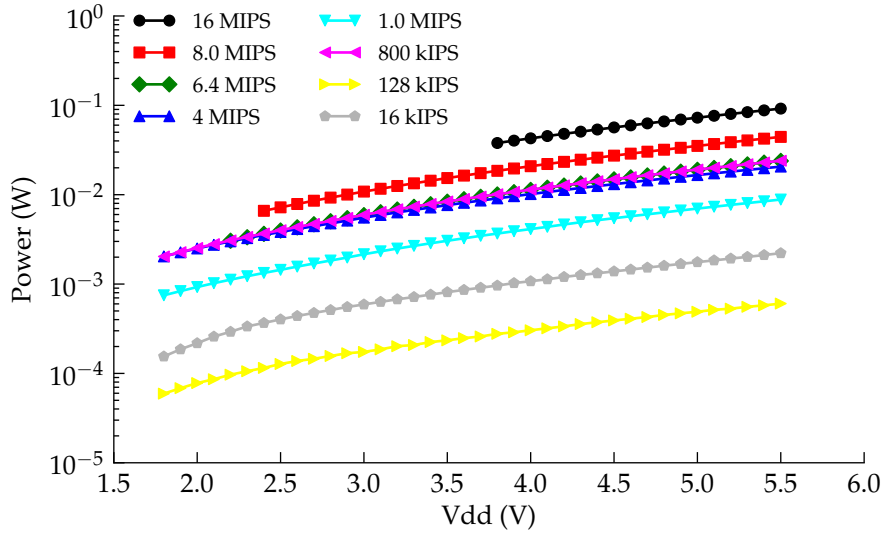

Figure B.6: Power consumption of the Atmel ATtiny25V while clocking

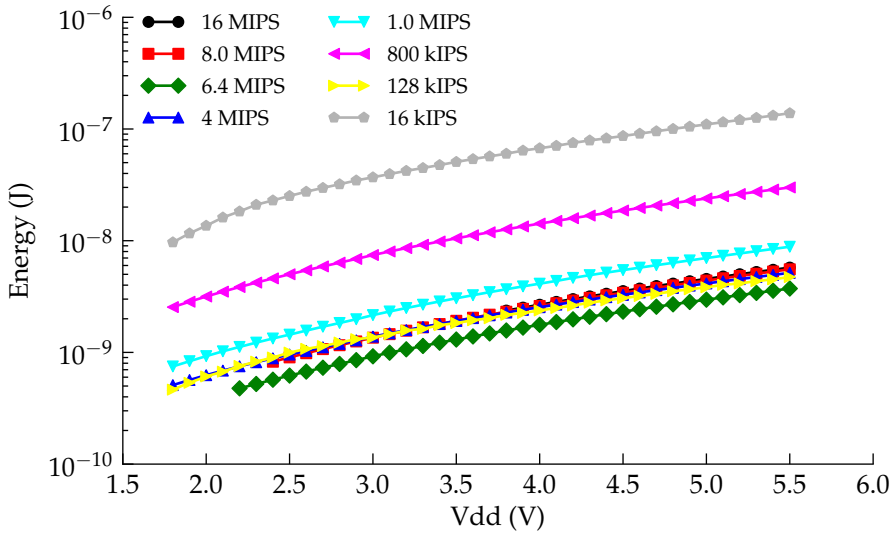

Figure B.7: Energy consumed per instruction for the Atmel ATtiny25V

Vdd	32 MHz	16 MHz	8 MHz	4 MHz	2 MHz	1 MHz	500 kHz	31 kHz
1.80	N/A	1.250E-03	8.043E-04	5.163E-04	3.912E-04	3.290E-04	1.327E-04	4.679E-06
1.90	N/A	1.322E-03	8.483E-04	5.391E-04	4.066E-04	3.405E-04	1.367E-04	4.892E-06
2.00	N/A	1.395E-03	8.930E-04	5.619E-04	4.220E-04	3.520E-04	1.402E-04	5.063E-06
2.10	N/A	1.468E-03	9.382E-04	5.848E-04	4.375E-04	3.637E-04	1.436E-04	5.307E-06
2.20	N/A	1.542E-03	9.841E-04	6.079E-04	4.528E-04	3.754E-04	1.471E-04	5.546E-06
2.30	N/A	1.620E-03	1.031E-03	6.310E-04	4.683E-04	3.870E-04	1.503E-04	5.796E-06
2.40	N/A	1.698E-03	1.078E-03	6.544E-04	4.838E-04	3.987E-04	1.535E-04	5.996E-06
2.50	2.711E-03	1.773E-03	1.126E-03	6.778E-04	4.996E-04	4.105E-04	1.567E-04	6.282E-06
2.60	2.826E-03	1.848E-03	1.172E-03	7.008E-04	5.150E-04	4.223E-04	1.599E-04	6.536E-06
2.70	2.939E-03	1.919E-03	1.214E-03	7.232E-04	5.301E-04	4.337E-04	1.631E-04	6.734E-06
2.80	3.050E-03	1.990E-03	1.255E-03	7.456E-04	5.450E-04	4.449E-04	1.662E-04	6.912E-06
2.90	3.163E-03	2.062E-03	1.297E-03	7.685E-04	5.605E-04	4.565E-04	1.695E-04	7.109E-06
3.00	3.277E-03	2.134E-03	1.338E-03	7.916E-04	5.758E-04	4.683E-04	1.728E-04	7.349E-06
3.10	3.391E-03	2.207E-03	1.380E-03	8.152E-04	5.917E-04	4.802E-04	1.763E-04	7.534E-06
3.20	3.507E-03	2.281E-03	1.422E-03	8.388E-04	6.074E-04	4.919E-04	1.797E-04	7.601E-06
3.30	3.623E-03	2.355E-03	1.465E-03	8.626E-04	6.235E-04	5.039E-04	1.829E-04	7.604E-06
3.40	3.445E-03	2.263E-03	1.438E-03	8.666E-04	6.386E-04	5.273E-04	1.940E-04	7.638E-06
3.50	3.445E-03	2.263E-03	1.439E-03	8.663E-04	6.390E-04	5.276E-04	1.942E-04	7.582E-06
3.60	3.447E-03	2.264E-03	1.439E-03	8.663E-04	6.389E-04	5.275E-04	1.943E-04	7.616E-06

Table B.6: Microchip PIC16F1827 clocking current (Vdd 0V – 3.6V)

Vdd	32 MHz	16 MHz	8 MHz	4 MHz	2 MHz	1 MHz	500 kHz	31 kHz
3.70	3.449E-03	2.265E-03	1.439E-03	8.663E-04	6.392E-04	5.277E-04	1.945E-04	7.617E-06
3.80	3.449E-03	2.265E-03	1.440E-03	8.665E-04	6.393E-04	5.279E-04	1.947E-04	7.649E-06
3.90	3.450E-03	2.265E-03	1.440E-03	8.667E-04	6.397E-04	5.283E-04	1.950E-04	7.603E-06
4.00	3.451E-03	2.267E-03	1.441E-03	8.669E-04	6.396E-04	5.283E-04	1.952E-04	7.634E-06
4.10	3.452E-03	2.267E-03	1.441E-03	8.672E-04	6.401E-04	5.286E-04	1.955E-04	7.633E-06
4.20	3.453E-03	2.268E-03	1.442E-03	8.673E-04	6.401E-04	5.286E-04	1.957E-04	7.667E-06
4.30	3.454E-03	2.269E-03	1.442E-03	8.675E-04	6.404E-04	5.290E-04	1.960E-04	7.620E-06
4.40	3.454E-03	2.269E-03	1.443E-03	8.679E-04	6.405E-04	5.293E-04	1.961E-04	7.660E-06
4.50	3.456E-03	2.270E-03	1.443E-03	8.681E-04	6.410E-04	5.294E-04	1.964E-04	7.663E-06
4.60	3.457E-03	2.271E-03	1.444E-03	8.684E-04	6.412E-04	5.297E-04	1.966E-04	7.704E-06
4.70	3.458E-03	2.271E-03	1.444E-03	8.687E-04	6.416E-04	5.300E-04	1.969E-04	7.664E-06
4.80	3.458E-03	2.272E-03	1.445E-03	8.690E-04	6.417E-04	5.303E-04	1.972E-04	7.710E-06
4.90	3.460E-03	2.273E-03	1.446E-03	8.692E-04	6.421E-04	5.307E-04	1.976E-04	7.721E-06
5.00	3.461E-03	2.274E-03	1.446E-03	8.695E-04	6.424E-04	5.310E-04	1.979E-04	7.757E-06
5.10	3.463E-03	2.275E-03	1.447E-03	8.697E-04	6.428E-04	5.314E-04	1.983E-04	7.733E-06
5.20	3.464E-03	2.276E-03	1.448E-03	8.699E-04	6.433E-04	5.319E-04	1.986E-04	7.762E-06
5.30	3.465E-03	2.277E-03	1.448E-03	8.703E-04	6.437E-04	5.326E-04	1.991E-04	7.797E-06
5.40	3.467E-03	2.278E-03	1.449E-03	8.708E-04	6.444E-04	5.332E-04	1.996E-04	7.777E-06
5.50	3.468E-03	2.279E-03	1.450E-03	8.711E-04	6.449E-04	5.336E-04	2.002E-04	7.833E-06

Table B.7: Microchip PIC16F1827 clocking current (Vdd 3.7V – 5.5V)

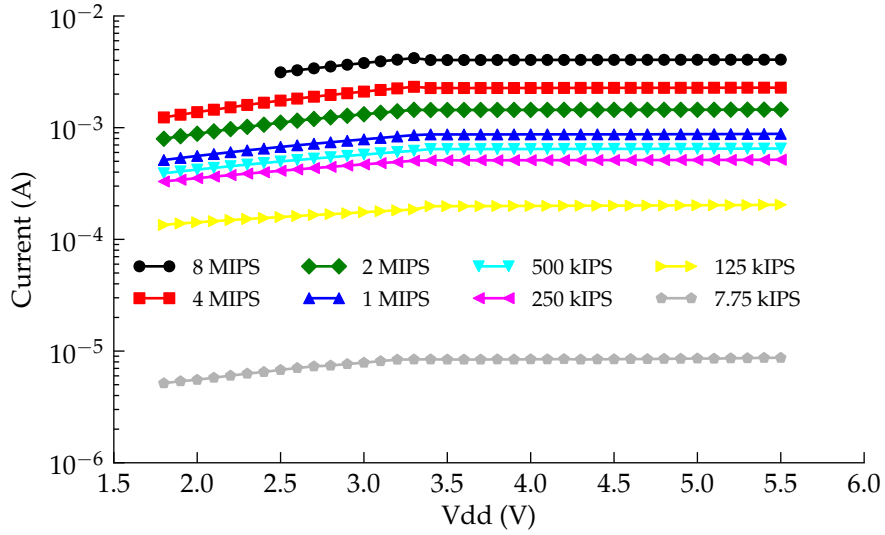

Figure B.8: Current consumption of the Microchip PIC16F1827 while clocking

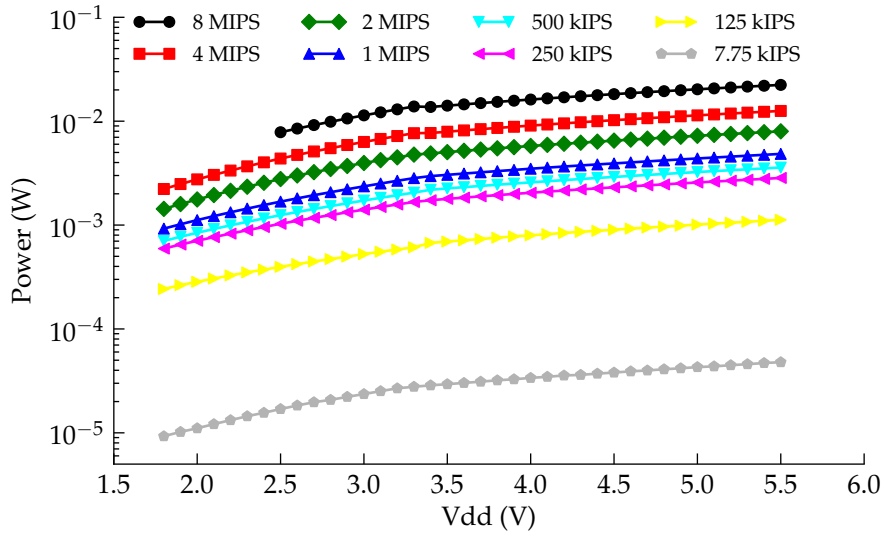

Figure B.9: Power consumption of the Microchip PIC16F1827 while clocking

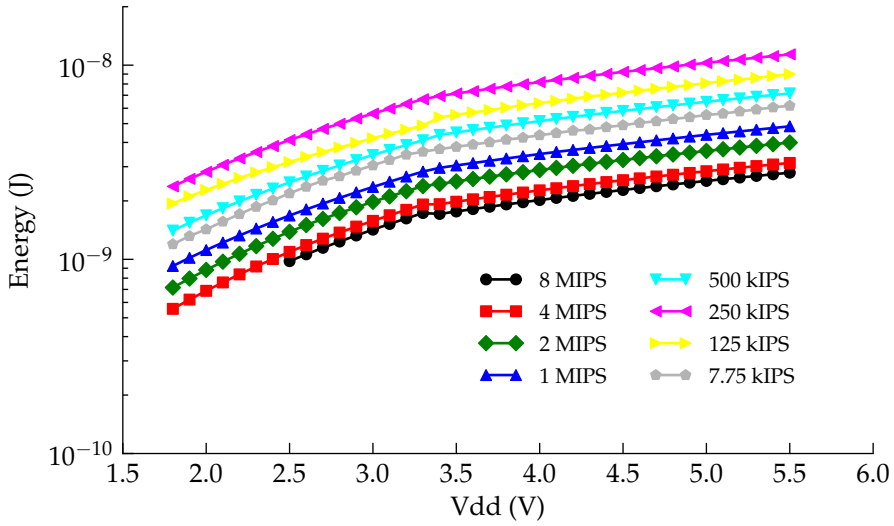

Figure B.10: Energy consumed per instruction for the Microchip PIC16F1827

+ 0ms′

```
1  #include <avr/io.h>
2
3  int main(void)
4  {
5      //POWER REDUCTION TIPS:
6    //   * Disable DWEN fuse
7    //   * Disable BODLEVEL fuse
8
9    //Disable interrups
10     SREG = 0x00;
11
12   //Set PortB as outputs
13     DDRB = 0b00000000;
14
15   //Disable analog comparitor
16   ACSR = 0x80;
17
18   //Disable digital input
19   DIDR0 = 0xFF;
20
21   //Disable ADC before sleep
22   ADCSRA = 0x00;
23
24   //Set GPIOs as high
25     PORTB = 0xFF;
26
27   //Sequence to disable brown-
```

```
28   //out detect while sleeping
29     MCUCR = 0b00110000;
30
31     while(1)
32   {
33     asm("sleep");
34   }
35 }
```

Listing B.1: ATtiny25V Sleep Proceedure

B.3.1.2 Clocking

```
1  #include <avr/io.h>
2
3  int main(void)
4  {
5      unsigned char rand;
6      SREG = 0x00;
7    //Set PortB as outputs
8      DDRB = 0b11111111;
9    //Set pins high
10     PORTB = 0xFF;
11     for(;;)
12     {
13         rand++;
14         asm("nop");
15     }
16
17 }
```

Listing B.2: ATtiny25V Clocking Proceedure

B.3.2 Microchip 12F675

Code was written in MPLAB v8.6 in C and compiled using HI-TECH C v9.60. Chip programming was done using PICkit 2 Programmer software v2.61. [1]

B.3.2.1 Sleep

B.3.3 Microchip 16F1827

Code written in MPLAB v8.6 in C and compiled using HI-TECH C v 9.60. Chip programming was done using PICkit 2 Programmer software v2.61, however in order to program the 16F1827 with the PICkit 2 programmer a patch was

[1] Downloading to the 12F675s with MPLAB led to corruption of the internal oscillators, causing them fail.

applied to the device file list. This patch was retrieved from http://www.uploadarchief.net/files/download/pk2patch16x.zip on the 12th May 2011.

B.3.3.1 Sleep

The specified sleep current of 30nA was not achievable. In an attempt to reach the specified current, the program was written in assembler. The assembler version of the code was used in the measurement data. The assembler version gave a lower sleep current than the C version, probably not as a result of the compiler but as due to a more thorough initialisation routine.

```
1  #include <htc.h>
2
3  __CONFIG(FOSC_INTOSC & WDTE_OFF & MCLRE_OFF & PWRTE_OFF
4    & BOREN_OFF & FCMEN_OFF & IESO_OFF & CLKOUTEN_OFF
5    & CP_OFF & CPD_OFF & LVP_ON & BORV_19 & STVREN_ON
6    & PLLEN_OFF & WRT_OFF);
7
8
9  void main(void)
10 {
11   //Set system clock to Internal osc block
12     SCS0 = 0;
13     SCS1 = 1;
14
15   //Set internal osc freq = 31kHz
16     IRCF3 = 0;
17     IRCF2 = 0;
18     IRCF1 = 0;
19     IRCF0 = 0;
20
21   //Disable interrupts
22     GIE = 0;
23
24     //Set all pins high (Tied to VDD via 10k)
25     PORTA = 0xFF;
26     PORTB = 0xFF;
27
28   //Put to sleep
29     SLEEP();
30     while(1)
31     {
32
33     }
34 }
```

Listing B.3: 16F1827 Sleep Proceedure - HI-TECH C version

```
1  LIST    P=PIC16F1827
2  #include <P16F1827.INC>
3
4    __CONFIG _CONFIG1 , _FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF &
         _MCLRE_OFF & _CP_OFF & _CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF &
         _IESO_OFF & _FCMEN_OFF
5    __CONFIG _CONFIG2 , _WRT_OFF & _PLLEN_OFF & _STVREN_OFF & _BORV_19 &
         _LVP_ON
6
7
8  ORG 0x0000 ; Specifies where to place the following code (which in this
       case is at the beginning of memory space )
9
10 START
11   ;Disable interrupts
12   BANKSEL INTCON
13   CLRF INTCON
14   ;Disable watchdog
15   BANKSEL WDTCON
16   CLRF WDTCON
17   ;Disable capacitive sensing
18   BANKSEL CPSCON0
19   CLRF CPSCON0
20   ;Disable modulation control
21   BANKSEL MDCON
22   CLRF MDCON
23   ;Disable perhiperal interrupts
24   BANKSEL PIE1
25   CLRF PIE2
26   ;Disable timer 1
27   BANKSEL T1CON
28   CLRF T1CON
29   ;Disable DAC
30   BANKSEL DACCON0
31   CLRF DACCON0
32   ;Disable ADC
33   BANKSEL ADCON0
34   CLRF ADCON0
35   BANKSEL ADCON1
36   CLRF ADCON1
37   ;Disable timers
38   BANKSEL T2CON
39   CLRF T2CON
40   BANKSEL T4CON
41   CLRF T4CON
42   BANKSEL T6CON
43   CLRF T6CON
44   ;Init PortA
45   BANKSEL PORTA ;
```

```
46    CLRF PORTA ;Init PORTA
47    BANKSEL LATA ;Data Latch
48    CLRF LATA ;
49    COMF LATA ;
50    BANKSEL ANSELA ;
51    CLRF ANSELA ;digital I/O
52    BANKSEL TRISA ;
53    CLRF TRISA ;SET AS OUTPUT
54    ;Init PortB
55    BANKSEL PORTB ;
56    CLRF PORTB ;Init PORTB
57    BANKSEL LATB
58    CLRF LATB ;
59    COMF LATB ;
60    BANKSEL ANSELB
61    CLRF ANSELB ;Make RB<7:0> digital
62    BANKSEL TRISB ;
63    ;and RB<3:0> as outputs
64    CLRF TRISB ;
65
66  LOOP ; Label this position (forms the start of a loop)
67    SLEEP
68    GOTO LOOP
69  END
```

Listing B.4: 16F1827 Sleep Proceedure - MPASM assembler version

B.3.3.2 Clocking

```
1  #include <htc.h>
2
3  __CONFIG(FOSC_INTOSC & WDTE_OFF & MCLRE_OFF & PWRTE_OFF
4    & BOREN_OFF & FCMEN_OFF & IESO_OFF & CLKOUTEN_OFF
5    & CP_OFF & CPD_OFF & LVP_ON & BORV_19 & STVREN_ON
6    & PLLEN_ON & WRT_OFF);
7
8  unsigned int count;
9
10 void main(void)
11 {
12   //Set system clock to Internal osc block
13     SCS0 = 0;
14     SCS1 = 0;
15
16   //Set internal osc freq = 31kHz
17     IRCF3 = 1;
18     IRCF2 = 1;
19     IRCF1 = 1;
20     IRCF0 = 0;
```

```
21
22   //Disable interrupts
23     GIE = 0;
24
25     //Disable serial ports
26     SSP1CON1 = 0x00;
27     SSP2CON1 = 0x00;
28
29     ADCON0 = 0x00;
30
31     //Disable modulation
32     MDSRC = 0b10000000;
33     MDCARH = 0b10000000;
34     MDCARL = 0b10000000;
35
36     //Add pins as outputs
37     TRISA = 0x00;
38     ANSELA = 0x00;
39     TRISB = 0x00;
40     ANSELB = 0x00;
41
42     //Set all pins high (Tied to VDD via 10k)
43     PORTA = 0xFF;
44     PORTB = 0xFF;
45
46   //Put to sleep
47     while(1)
48     {
49       count++;
50     }
51 }
```

Listing B.5: PIC16F1827 Clocking Proceedure

B.3.4 Freescale M9S08QG8

Code was written using Freescale's bundled IDE, CodeWarrior v5.90, and downloaded using a supplied USB demo board (DEMO9S08QG8E).

B.3.4.1 Sleep

```
1 #include <hidef.h> /* for EnableInterrupts macro */
2 #include "derivative.h" /* include peripheral declarations */
3
4 void main(void) {
5
6
7   SOPT1 = 0b00100000;
```

```
8   //          |||    ||
9   //          |||    |\- RESET Pin Enable (0)
10  //          |||    \-- Background Debug Mode Pin Enable (1)
11  //          ||\------- Stop Mode Enable (0)
12  //          |\-------- COP Watchdog Timeout (1)
13  //          \--------- COP Watchdog Enable (1)
14
15  //Enable low power bit
16  ICSC2 = 0b01001000;
17
18  //Disable Low Voltage Detect
19  SPMSC1 = 0x00;
20
21  //Enable power down control
22  //Disable partial power down
23  SPMSC2 = 0x02;
24
25  for(;;)
26  {
27    //Enter sleep mode
28    _Stop;
29  }
30 }
```

Listing B.6: M9S08QG8 Sleep Proceedure

Chapter C
Streaming Cell Mesaurements

C.1 Streaming Potential Cell Measurements

In this section the results of individual streaming cell measurements are presented. Each of the following graphs represent measurements of various channel heights. The construction of the channels is details in Section 3.2.1

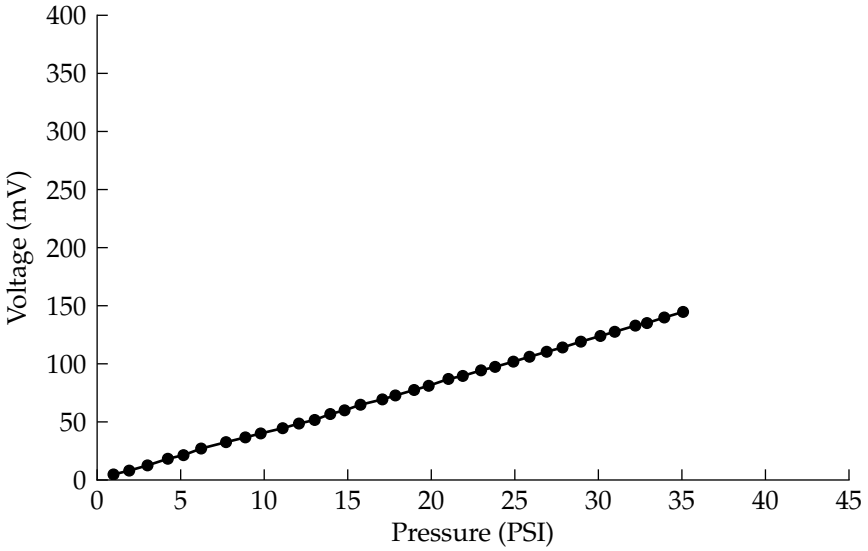

Figure C.1: Line graph showing voltage output versus pressure for a 26 μm glass micro-channel

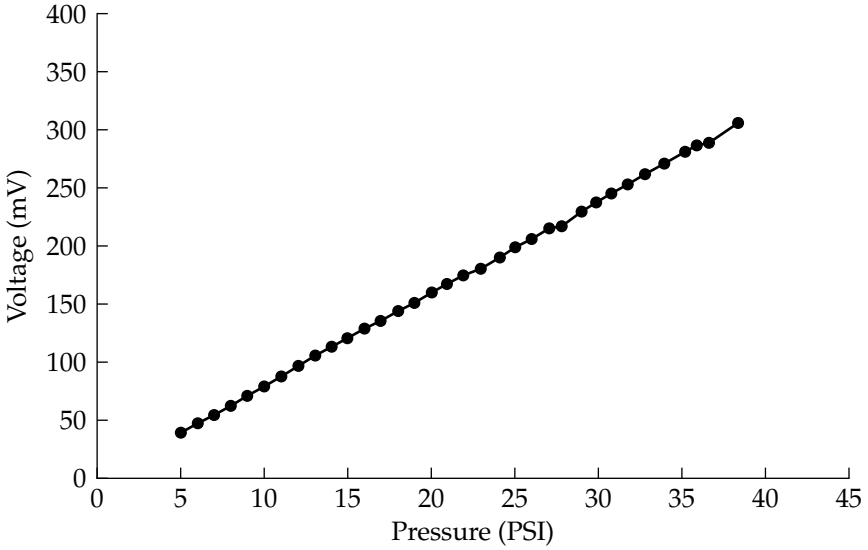

Figure C.2: Line graph showing voltage output versus pressure for a 52 μm glass micro-channel

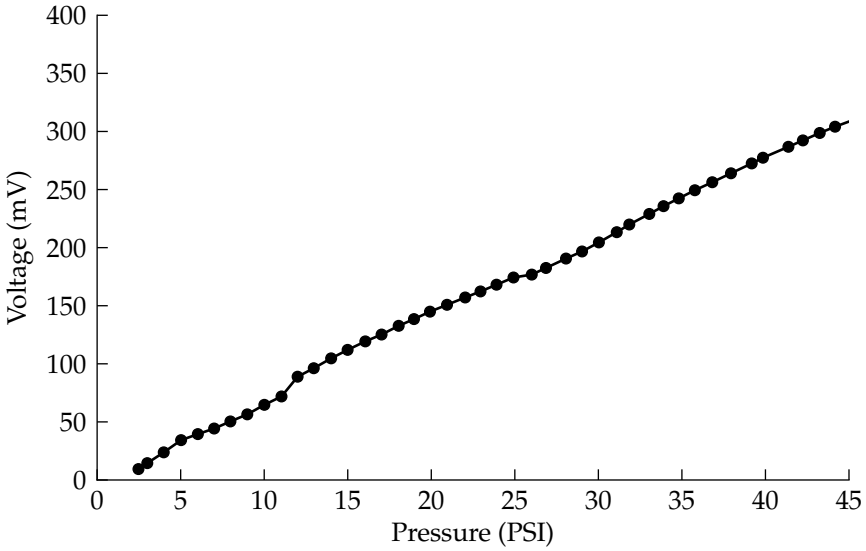

Figure C.3: Line graph showing voltage output versus pressure for a 56 μm glass micro-channel

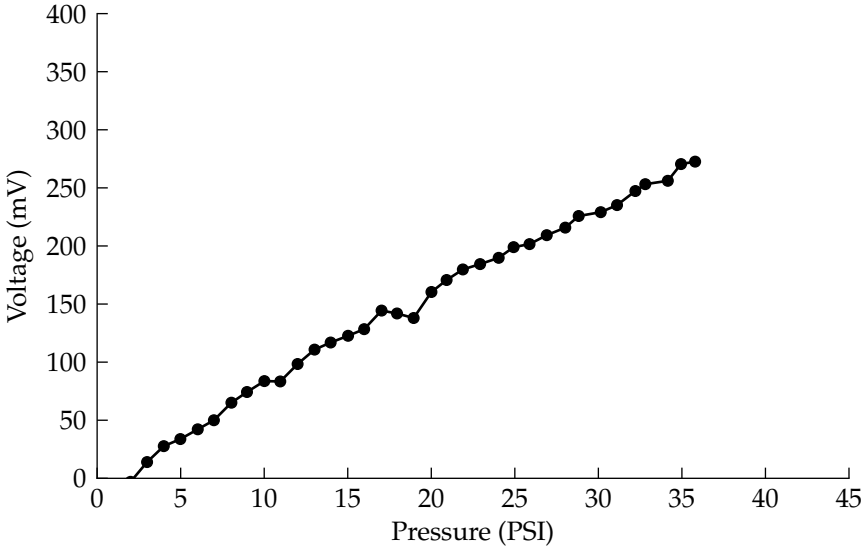

Figure C.4: Line graph showing voltage output versus pressure for a 71 μm glass micro-channel

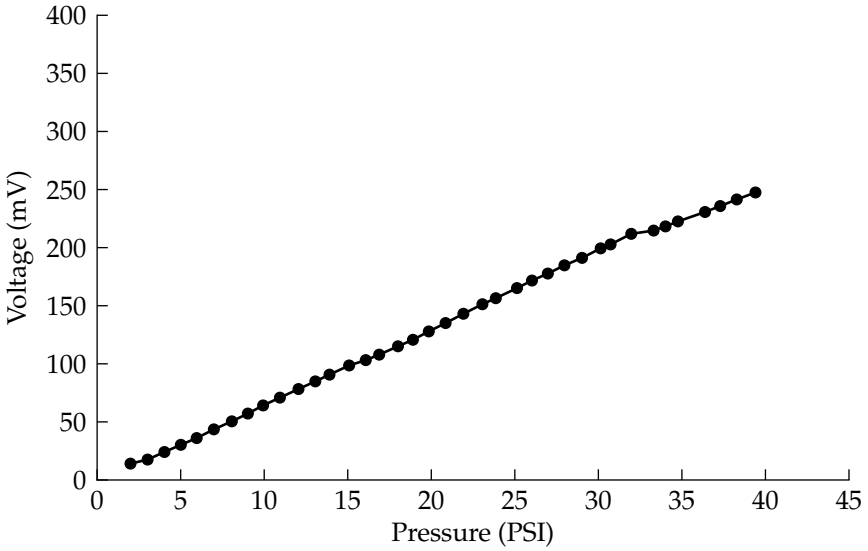

Figure C.5: Line graph showing voltage output versus pressure for a 75 μm glass micro-channel

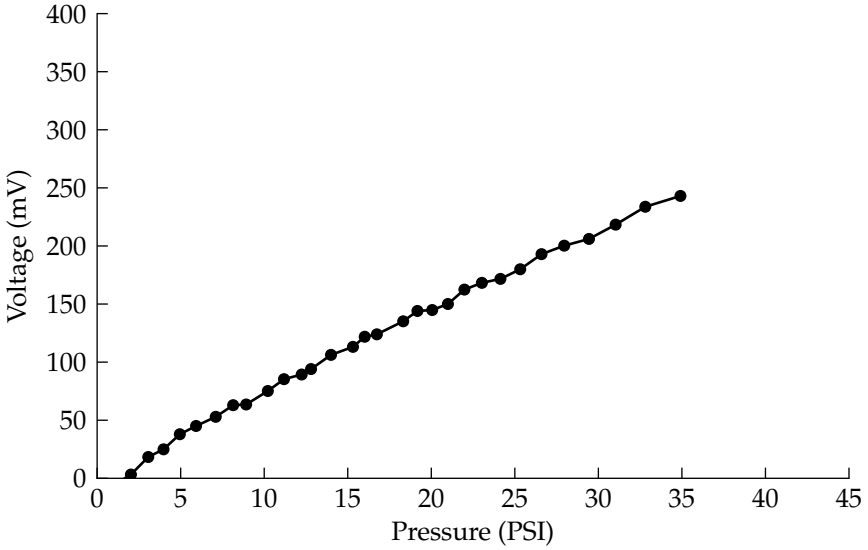

Figure C.6: Line graph showing voltage output versus pressure for a 106 μm glass micro-channel

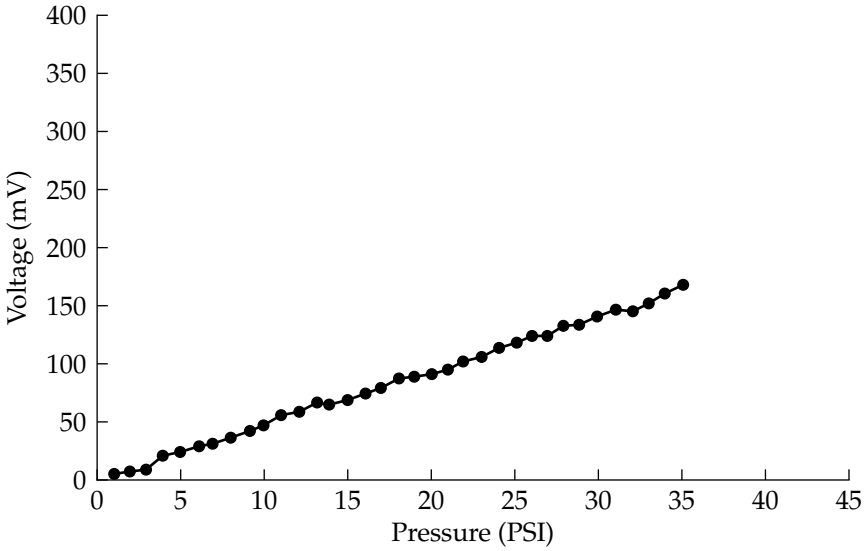

Figure C.7: Line graph showing voltage output versus pressure for a 125 μm glass micro-channel

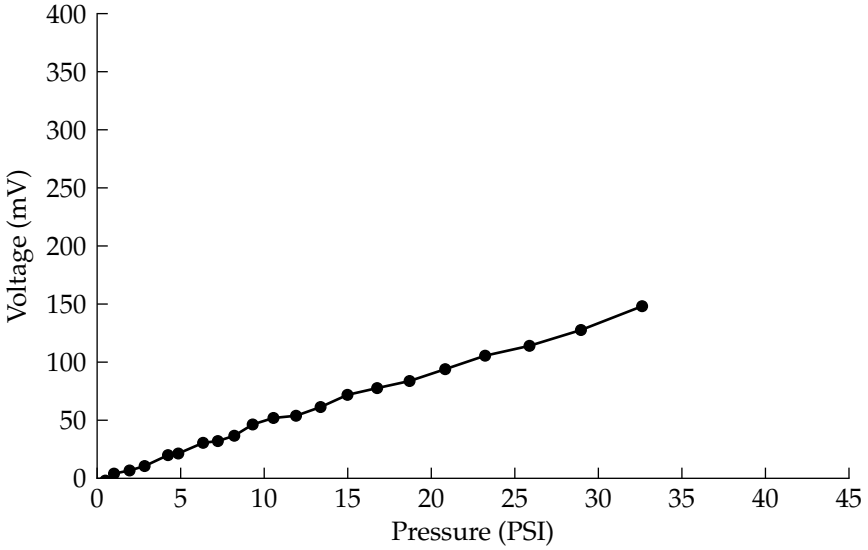

Figure C.8: Line graph showing voltage output versus pressure for a 161 μm glass micro-channel

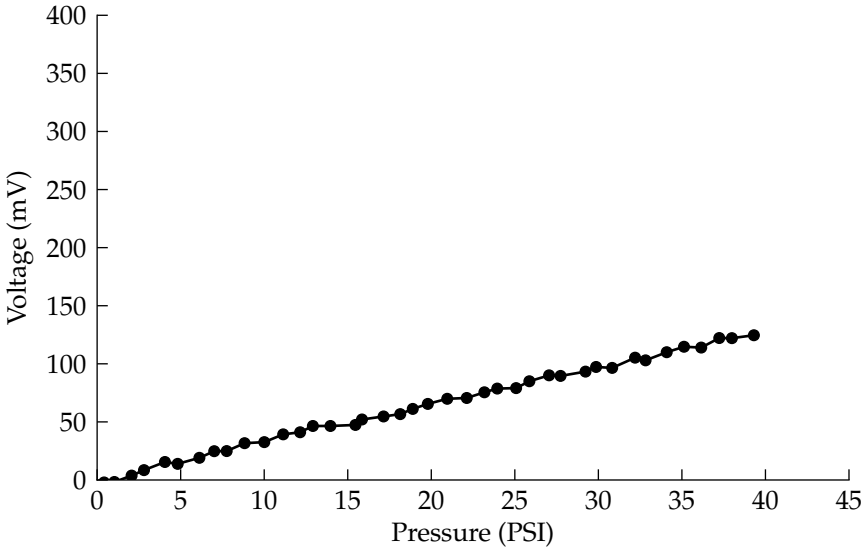

Figure C.9: Line graph showing voltage output versus pressure for a 178 μm glass micro-channel

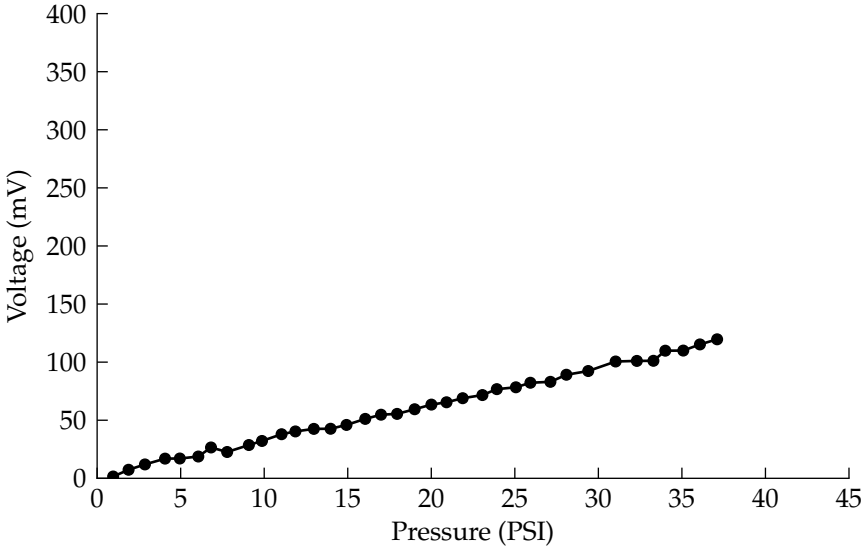

Figure C.10: Line graph showing voltage output versus pressure for a 245 μm glass micro-channel