HARDWARE MODEL OF CARDIAC CELL

A Design Project Report
Presented to the School of Electrical and Computer Engineering of Cornell University

in Partial Fulfillment of the Requirements for the Degree of Master of Engineering, Electrical and Computer Engineering

Submitted by:
Adarsh Jayakumar
Samir Durvasula

MEng Field Advisor: Bruce Land
MEng Outside Advisor: Bruce Johnson

Degree Date: May 2018

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Our project is a novel teaching tool designed to help students learn about cardiac neurophysiology. This paper highlights the iterative design decisions and model choices that we made on our way to our end product. We begin with the needs-based criterion that prompted our design as well as the materials/methods that we used to create the project. We then provide a brief literature review, with a background into how action potentials work and how they differ from cardiac action potentials. Additionally, we give an overview of how neurons are modeled as per Hodgkin Huxley, how cardiac neurons are modeled as per Dennis Noble, and how human ventricle cells are modeled as per Priebe-Beuckelmann.

This literature review leads directly to the Matlab implementations of these three models as well as the design choices for our final reduced Priebe-Beuckelmann model. The implementation of each of these three models in C on a PIC32 microcontroller are detailed, with more emphasis on the design decisions for our final model implementation. Particular attention is given to how we implemented complex differential equations on a time-sensitive system and how the implementation of these three models differed from each other. All of the equations and state machine logic for each of the models are provided through the text and appendices.

The user interface is detailed according to the three *Protothreads* threads within our code: *protothread_tft, protothread_update, protothread_button*. Each section and design decision is aimed at ensuring our user has an experience that is educational, user-friendly, and computationally accurate. Explanations include both block diagrams and real-life images for the reader's convenience. The *protothread_tft* section highlights the navigation of the first four screens while *protothread_update* focuses on the final TFT oscilloscope screen. Each screen was created to be as intuitive as possible with considerable time spent beta-testing with random strangers to ensure robustness. The *protothread_button* thread contains our debouncing state machine.

Finally, integration is briefly highlighted through an overview of our code structure as well as pictures of our schematic and PCB design. The appendices primarily provide the equations within each of our models. However, a "lab handout" is provided in Appendix 4 as an example of how students can interface with our product and learn about cardiac arrhythmia, cardiac arrest, and cardiac neurophysiology. We hope that this paper alongside our code and final PCB will allow this product to be used as a teaching tool for cross-listed courses within the ECE/Neurobiology curriculum.

ABSTRACT

Our project is a novel teaching tool designed to help students learn about cardiac neurophysiology. More specifically, we have created a microcontroller-based "cardiac cell" that can be used in the lab sections of ECE/BME/BIONB 4910. The "cardiac cell" is based on a reduced Priebe-Beuckelmann model and uses differential equations to describe Potassium,

Sodium, and Calcium ionic currents. Our end product is designed such that students are able to manipulate the ion concentrations/conductance across the cell membrane, choose which ion currents they would like to view, and then watch the voltage/current traces vs. time both on-board as well as through an oscilloscope. Examples of how students can interface with our product and learn about cardiac arrhythmia, cardiac arrest, and cardiac neurophysiology are highlighted. From a teacher's perspective, this project will be useful for any ECE/Neurobiology cross-listed courses and can also be used as a teaching tool for when Cornell's Neurobiology Department travels internationally.

INTRODUCTION

For students in biology classes, one of the most valuable experiences is hands-on work with biologic systems. In neurophysiology classes such as ECE 4910, students dissect, stimulate, and record from invertebrates such as crayfish or snails in order to understand the ionic basis of action potentials (Wyttenbach, et al, 2011). While this experience is valuable in teaching students the process of neural recordings, it is also time-consuming and expensive. Students are often unable to gather reliable data without guidance from experienced instructors. Additionally, hands-on exploration of cardiac neurophysiology requires the use of vertebrate animals, an option which is currently not available for students in the class.

Within the aforementioned scope, we defined several needs for our product: educational, user-friendly, and computationally accurate. Firstly, our product needed to be designed to showcase and highlight educational information. As such, all of our model and user-interface choices were based upon ensuring that students could learn the most about cardiac neurophysiology. Three ion current traces (Sodium, Potassium, and Calcium) can be displayed on the on-board TFT alongside the voltage trace. The traces can be paused at any time and cursors used to record differences in voltage, current, and time. Additionally, the total current or membrane voltage can output to an off-board oscilloscope for further analysis through Labchart or other analysis software. A sample lab handout containing the information students could learn about with our product can be seen in Appendix 4 and 5.

Secondly, our product needed to be user-friendly. There are six variables that the user can change within the model: concentrations and conductance for the Sodium, Potassium, and Calcium ion channels respectively. Each screen for the on-board input is intuitive with four-arrow scrolling, an enter button for selection, and a slider for changing values. The addition of both on board and off board data output allows the user to view and analyze the data in whichever manner most convenient for them. This is especially critical for international cases where access to an oscilloscope may not be as convenient as within the ECE 4910 class setting.

Thirdly, our product needed to be computationally accurate to human neurophysiology. The data provided by our product will be analyzed computationally by students in order to learn about cardiac neurophysiology. As such, our differential equation model needs to accurately

match true physiology while still being implementable on a time-sensitive microcontroller platform.

A picture of the labeled final product can be seen in Figure 1 below

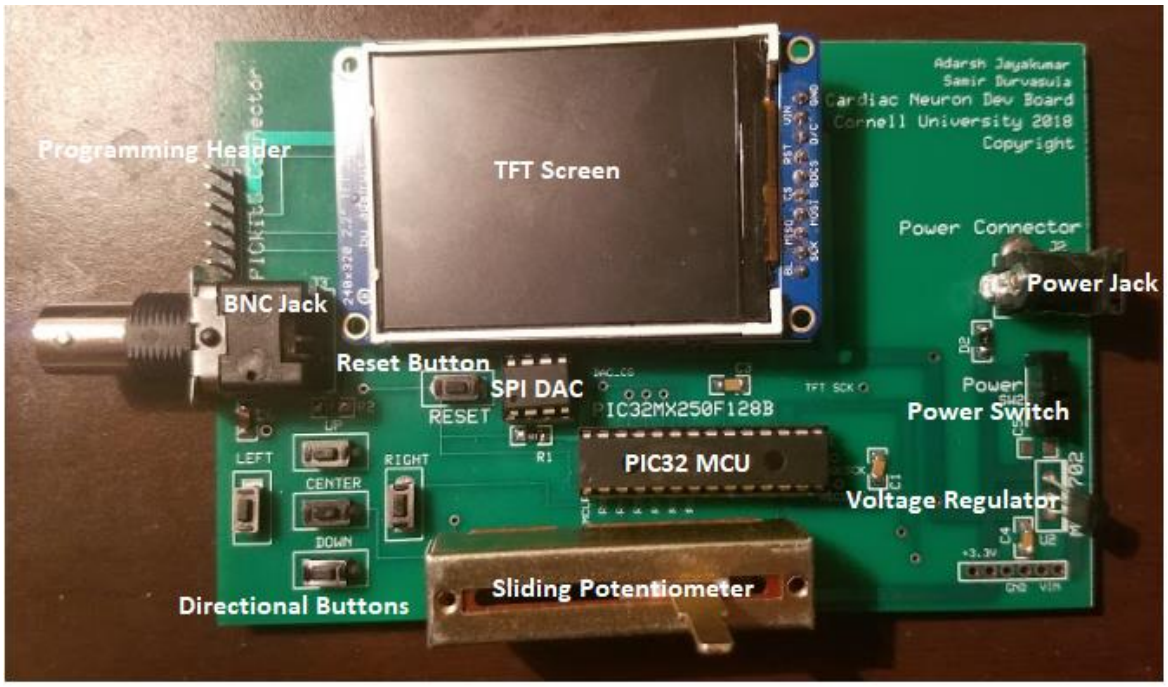

Figure 1: Labeled Printed Circuit Board

DESIGN STEPS & MATERIALS

Our design process had three facets: creation of the cardiac cell model, creation of the user interface, and integration of the total product. The cardiac cell model was initially implemented in Matlab and required significant testing to realize the most accurate and stable system. Once finalized, the cardiac cell model was then implemented in C on a PIC-32 microcontroller through the MPLAB IDE. Testing of this C implementation was accomplished through the external oscilloscope to check for stability and accuracy. The user interface was immediately implemented in C on a PIC-32 microcontroller using the MPLAB IDE. Publicly available libraries for threading and TFT display were integrated into the user interface from the beginning (Mahbub, 2014 & Dunkels, 2006). Finally, integration of the total product involved considerable timing analysis and debugging to ensure proper functionality of the user interface and cardiac cell model together. An initial prototype used for debugging as well as a completed printed circuit board (PCB) were created as the final steps towards integration.

The final product PCB requires several simple components and takes approximately 30 minutes to solder. A list of these materials can be seen in Table 1 below.

Component	Number
Microcontroller (PIC32MX250F128B)	1
Adafruit TFT LCD Screen	1
100 KΩ Sliding Potentiometer	1
100 nF Surface-mount Ceramic Capacitor	3
10 uF Surface-mount Ceramic Capacitor	1
1 uF Surface-mount Ceramic Capacitor	2
10 KΩ Surface-mount Resistor	2
Single-pole Double-throw Switch	1
Push Button	6
1 Amp Rectifying Diode (1N4007)	1
12-bit Digital-Analog Converter (MCP4822)	1
3.3 V LDO Voltage Regulator (MCP1702)	1
BNC Connector	1
Power Jack Connector	1
28 Pin DIP Header	1
8 Pin DIP Header	1

Table 1: List of materials on final PCB

BACKGROUND

Action Potentials

Action potentials are the cellular impulses that control messaging in the nervous system. These action potentials are shaped by a series of steps involving the opening and closing of ion channels.

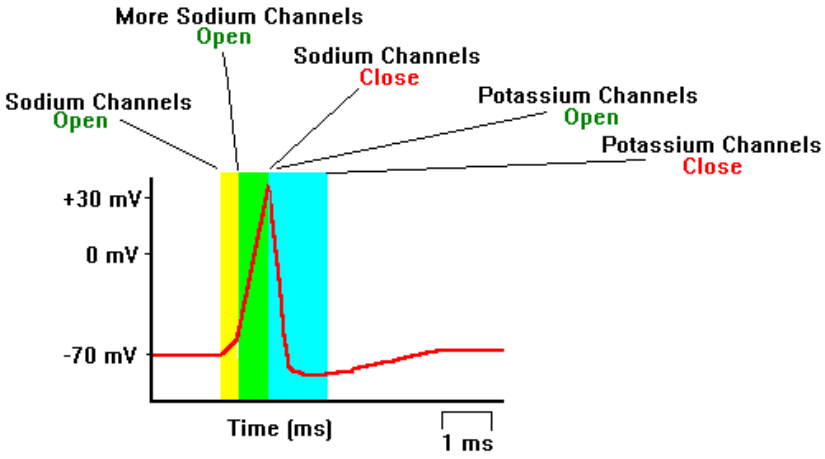

Figure 2: Ionic basis for action potentials (Chudler)

In a typical action potential, when a neuron is stimulated, sodium channels begin to open. Given the concentration gradient, sodium ions begin to rush into the cell and increase its voltage. This process is called depolarization. If the cell is depolarized to a threshold level, many voltage gated sodium channels open, leading to a sharp spike in the cell's voltage. Once a peak voltage is reached, the sodium channels close, and the open potassium channels take over. Given the potassium concentration gradient, open potassium channels lead to positive potassium ions leaving the cell and lowering the cell's voltage (repolarizing/hyperpolarizing). Since these channels are slow to close, the voltage eventually dips below the resting potential. Finally, when the potassium channels close, the cell returns to resting potential. (Silverthorn, 2013)

Cardiac Action Potentials

Cardiac cells have similar behavior as neuron action potentials with two key differences. Firstly, cardiac cells do not require a stimulus to fire. Instead, cardiac cells spontaneously fire at a regular pace in order to control the heartbeat (Luo et al, 1991). Secondly, cardiac action potentials have a significantly different shape from neuron action potentials. The depolarizing spike is extremely fast with a plateau phase causing repolarization to take far longer than in a neuron action potential. This increased repolarization time allows the cell to maintain a constant heartbeat on a second timescale (Ramanathan et al, 2006)

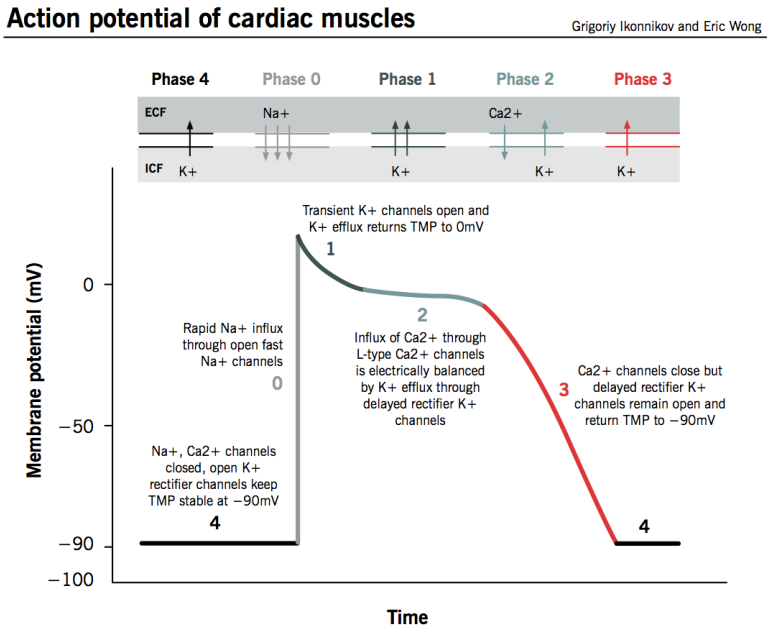

Figure 3: Ionic Basis of Cardiac Action Potential

A cardiac action potential labeled with the five phases can be seen in Figure 3 above (Ikonnikov, 1993). The five phases showcased in Figure 3 above are briefly explained below:

Phase 0: An action potential from a neighboring pacemaker cell causes the membrane potential to rise above -90 mV. Fast Sodium channels start to open, leaking Sodium ions into the cell and further increasing the membrane potential. As the membrane potential crosses the “threshold potential”, there are enough fast Sodium channels to generate a self-sustaining inward current. The membrane potential crosses above 0 mV briefly at which point fast Sodium channels start to close.

Phase 1: Potassium channels briefly open, causing an outward flow of Potassium ions and returning the membrane potential closer to 0 mV

Phase 2: Calcium channels activated during depolarization cause a constant inward current of Calcium ions. Potassium ions leak out the cell through delayed rectifier Potassium channels. The balance of these currents causes a plateau of the membrane potential.

Phase 3/4: Calcium channels start to inactivate and Potassium ions flow through inward rectifier channels. This causes the membrane potential to come back to its resting potential and prepare for another depolarization. (Silverthorn, 2013)

Modeling Neurons

In 1952, Alan Hodgkin and Alfred Huxley published the model of neurons that is still used as a gold standard for computational models of the nervous system. This model theorizes a neuron as an RC Circuit: the cell membrane is modeled as a capacitor and ion channels are modeled as variable conductors. The conductance depend on cell voltage, cell current, ion

concentrations, among other factors. To define these interactions, Hodgkin and Huxley derived a series of differential equations. In their model, Hodgkin and Huxley included three different conductance: Na, K, and Leak. For each of these conductance, Hodgkin and Huxley defined behavior through the use of gating variables. These gating variables are what change behavior based on the aforementioned cellular factors. (Hodgkin and Huxley, 1952)

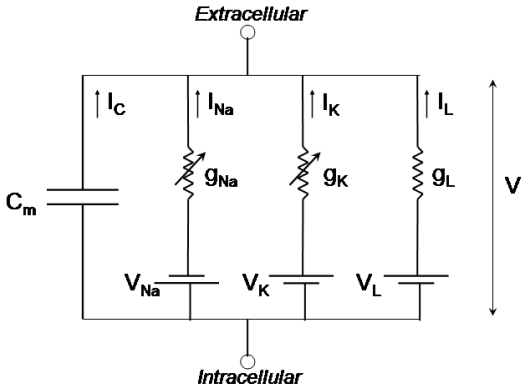

Figure 4: Hodgkin Huxley model of Neuron. Noble Model has the same conductance.

The first mammalian model of cardiac cells was developed in 1962 by Denis Noble. This model was very similar to Hodgkin Huxley in the currents and conductance involved. However, the Denis Noble model introduces a slow acting potassium current. Since this second current acts slowly, the leakage current is able to take over and lead to regular, spontaneous, firing of the cells without an external stimulus (Noble, 1962). The general current equation is very similar, and can be found in (1). In addition, the base conductance and equilibrium potentials had changed. Finally, there were slight changes in the rate functions that controlled the action of the neurons.

$$\sum_k I_k = (g_{Na} m^3 h + .14)(V - E_{Na}) + (g_{K_2} + g_{K_1}) n^4 (V - E_k) + g_{Leak}(V - E_{Leak}) \quad (1)$$

In this model, the initial conditions for conductance and equilibrium potential changed as well (Appendix 2, Table 1). Finally, all of the rate equations changed to represent cardiac neuron dynamics (Appendix 2, Table 2).

In Figure 5, we show the graph of a Denis Noble based neuron. This neuron has a steep spike, and generally lower frequency. However, these neurons do not exhibit the typical calcium channel plateau as would be expected in human heart cells. Hence, the Noble model, while a strong basis for cardiac cells was incomplete in defining cell behavior.

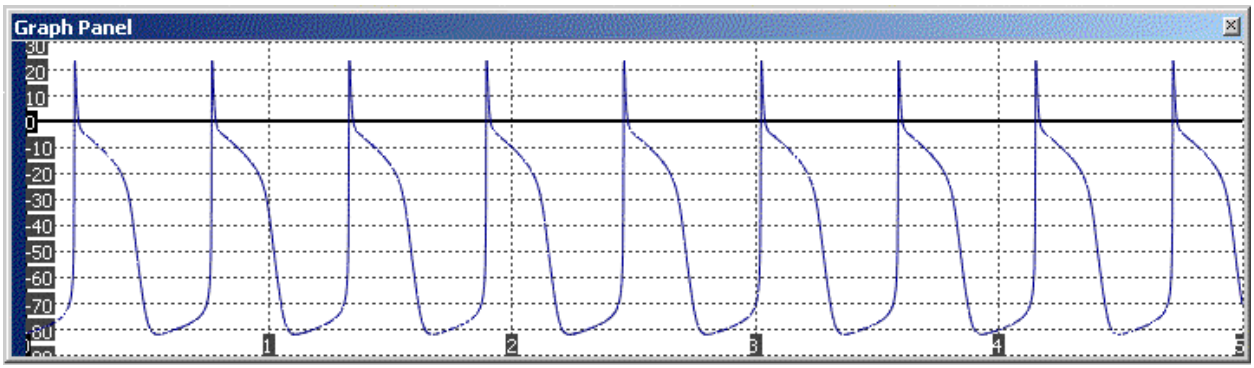

Figure 5: Spontaneous firing of Noble neurons (Fink & Noble, 2006)

Modeling Human Ventricle Cell

In 1998, Leo Priebe and Dirk Beuckelmann developed the first model of the human ventricle cell. While the Noble model was simpler in that it included only 4 currents and 3 gating variables, the Priebe-Beuckelmann model was far more complete. This model, instead includes a total of 10 total currents with 17 total variables. These include the L-type Calcium current, a transient outward potassium current, a fast and slow delayed rectifier potassium current, an inward rectifier potassium current, a transient outward potassium current, a fast sodium current, a sodium/calcium exchange current, a sodium/potassium pump, and background sodium and calcium currents. In total the voltage, 9 gate variables, intracellular sodium and potassium concentrations, and several variable calcium concentrations were involved in the calculation. The interactions between these currents bubbled into many equations to control the shape of the model (Priebe and Beuckelmann., 1988).

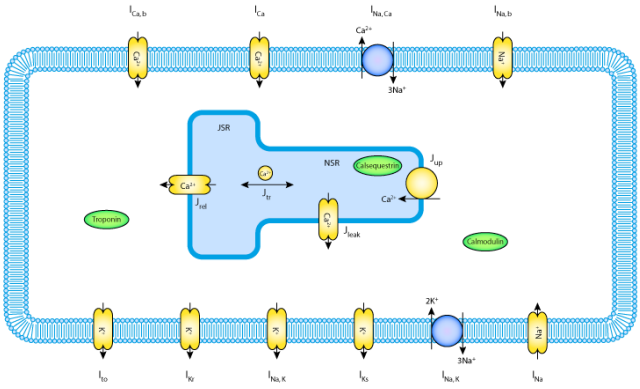

Figure 6: All currents involved in Priebe Beuckelmann

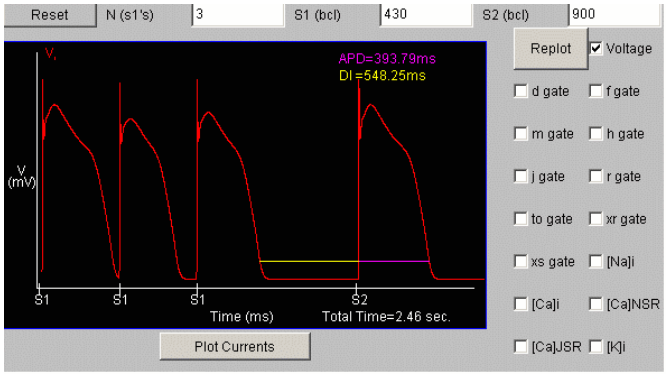

Figure 7: Priebe Beuckelmann Shape (Fenton & Cherry)

MODEL DESIGN

Matlab Implementation of Hodgkin Huxley

The Hodgkin Huxley model neuron was first implemented in Matlab by the following sets of equations. First, each ion has a base conductance and equilibrium potential (Appendix 1, Table 1). The base conductance is a constant coefficient for each term. The current is determined as the sum of products, the gating variables, and the difference between membrane voltage and equilibrium potential. Hodgkin and Huxley updated the gating variables in an exponential sense, dividing the changing of the variables into two rate functions, α andβ. These are changed as in Appendix 1 Table 2, and Appendix 1, Equations 1-5.

Finally, the change in voltage is solved for by dividing the current by membrane capacitance. The capacitor acts as an integrator to the current. To solve for these equations, integration methods such as Euler's method can be used to find the voltages. For each of these potentials a sustained stimulus current is required to fire the neurons.

We directly derived our MATLAB model from Professor Bruce Land's Hodgkin Huxley neuron MATLAB model, which was used for the computational section of BIONB 2220. This program set initial conditions based on ion conductance for a Giant Squid Axon (Appendix 1, Table 1) and used Euler's method to find membrane voltage at any given time for a Hodgkin Huxley Neuron.

In this program, we first set all initial equilibrium potentials and base conductance. Then, we calculate the steady state gate variables based on the resting potential. This steady state value is the initial condition for the gating variables. Since Hodgkin Huxley neurons require stimulation, we also initialize the magnitude of stimulation. During the calculation, we first update the overall conductance based on the gating variables. We then calculate the change in gating variables based on their respective α and β values. The equations for these updates are described in Appendix 1. We then calculate the updated net current of the cell, and find the change in the voltage by dividing by the membrane capacitance. Finally, having calculated the change in the variables, we update the gating variables and voltages using Euler's method. We

iterate these steps for a set time. In figure 8, we show the results of modeled Hodgkin Huxley neurons at various stimulus magnitudes.

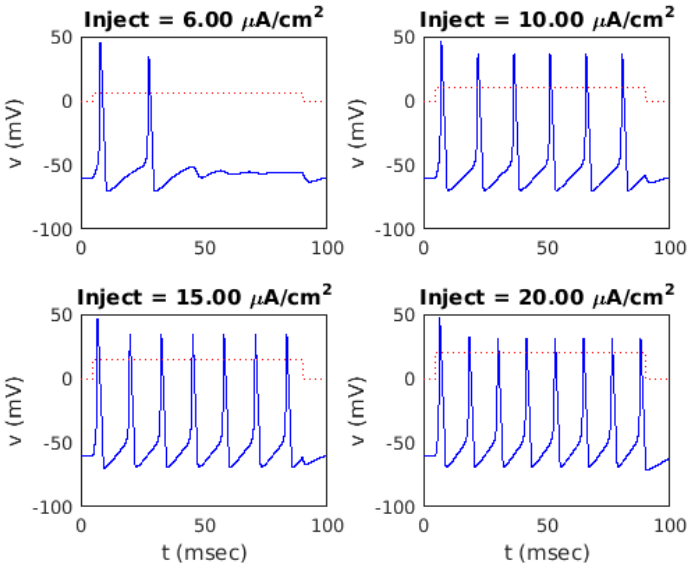

Figure 8: Hodgkin Huxley Neurons at various current injections

Denis Noble Matlab Modeling

Our first pass at implementing a model of a cardiac cell was modeling the Denis Noble model in MATLAB. Having a working Hodgkin Huxley model was very useful because the Denis Noble model is simply a modification of Hodgkin Huxley. Both neuron models make the use of changing ion conductance with gating variables and rate variables. In our modeling of Denis Noble, we used Euler integration as we had previously with Hodgkin Huxley. However, individual equations were changed as in Appendix 2. The final total current calculation is (2).

$$\sum_k I_k = (g_{Na} m^3 h + .14)(V - E_{Na}) + (g_{K_2} + g_{K_1}) n^4 (V - E_k) + g_{Leak}(V - E_{Leak}) \quad (2)$$

After changing currents, all other elements of this program changed. Importantly, there was no current injection. This is because cardiac cells as modeled by Denis Noble do not require external stimulation to fire. This model's firing pattern is displayed in Figure 9.

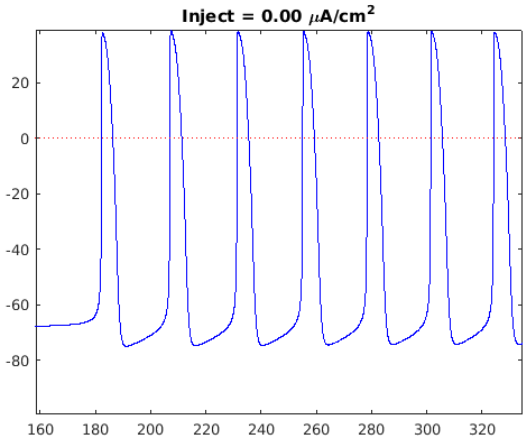

Figure 9: Noble Model action potentials from MATLAB

In order to see the firing, the overall time of the program had to be increased. In addition, there was significant delay time between the beginning and the first potential, although firing occurs regularly after the first potential.

Design Decisions for Priebe Beuckelmann Model

While the Priebe Beuckelman Model is extremely accurate, the amount of calculation required is impractical to run on a microcontroller while still maintaining biologically accuracy to a human heartbeat. It is necessary to therefore find a model that exists between the Noble Model and the Priebe Beuckelmann models of cardiac cells. In order to do so, it is important to realize that several currents and concentrations are relatively constant throughout the process of the cardiac action potentials. For instance, it is appropriate to assume that intracellular concentrations of calcium ions do not change significantly throughout the course of an action potential and can remain constant. The sodium calcium exchange and the sodium potassium pump only change based on these concentrations, and can also be assumed to be constant. Therefore, several of the currents can be combined in order to limit the total number of currents available.

Bernus et al took all of these factors into account and derived the Reduced Priebe Beuckelmann model. In this model, Bernus et al reduced the previous model from 10 currents and 17 variables down to 5 currents and six total variables (Bernus et al, 2002). This was done so by leaving concentrations constant where possible, and eliminating several gating variables. The result of this simplification was the creation of a still biologically accurate model to Priebe Beuckelmann, but also a model that was easier to work with. Although most of the currents still exist in this model, they are essentially constant or exponential with voltage as seen in Figure 10 below.

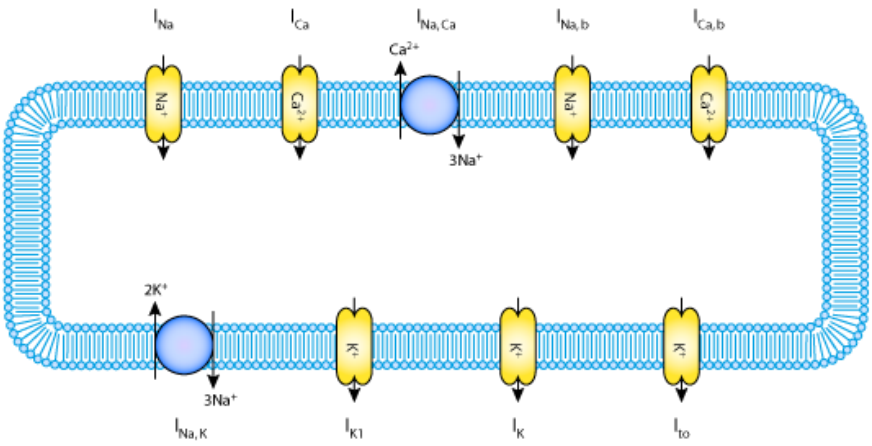

Figure 10: All currents of reduced Priebe Beuckelmann model

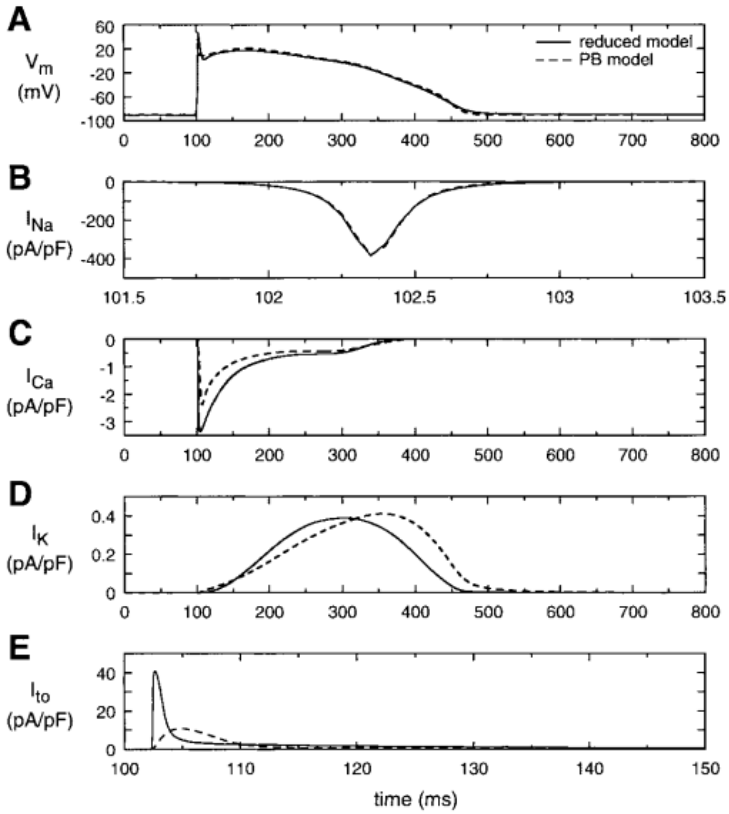

Figure 11: Comparison of Priebe Beuckelmann with Reduced Priebe Beuckelmann

While the shapes of individual currents were affected by the reduced model, the net action potential remained essentially the same (Figure 11 above). Most importantly, because this was the case, we decided to implement this model as our cardiac cell model. The equations used to define this, and all models are contained in the appendices.

In order to implement this model, we followed the logic used in the virtual heart simulation, implemented by Flavio Fenton and Elizabeth Cherry. All of the equations used in this model are described in Appendix 3. To implement this model in MATLAB, much of the initial design had to be remodeled.

Firstly, several new currents have to be implemented, each with their own conductance and gating variables. The total of all currents is in (3):

$$I_{total} = I_{Na} + I_{Ca} + I_K + I_{K,slow} + I_{K,to} + I_{NaK} + I_{NaCa} + I_{Na,b} + I_{Ca,b} \quad (3)$$

These include fast sodium, L-type calcium, potassium, transient outward potassium, sodium/potassium exchange, and sodium calcium exchange.

In addition, gating variables were not updated using the simple Euler's method as was done previously. Instead, if $x_\infty$is the steady state value of gating variable x, and $\tau_x$is the time constant of gating variable x, then $x(t) = x_\infty(t-1) + (x(t-1) - x_\infty(t-1)) * exp(-dt/\tau_x)$. These are voltage dependent in that, in general, each gating variable steady state and time constant values are dependent on rate functions α_x and β_x. This is described in (4) and (5)

$$x_\infty = \alpha_x/(\alpha_x + \beta_x) \quad (4)$$

$$\tau_x = 1/(\alpha_x + \beta_x) \quad (5)$$

(4) and (5) were primarily used for steady state calculation for all gating variables. However, there are some exceptions to these methods of calculations that are included in Appendix 3, Section 4. In addition, some gating variables involved in conductance calculation only use their steady state value. These are also included in the appendix.

As a whole, these resulted in better approximations of a Cardiac cells, with some artifacts due to integration. This model was much more effective in detecting direct behavior changes due to changes in all ion concentrations and conductance. Figure 12 shows an example waveform from the MATLAB model of reduced Priebe Beuckelmann.

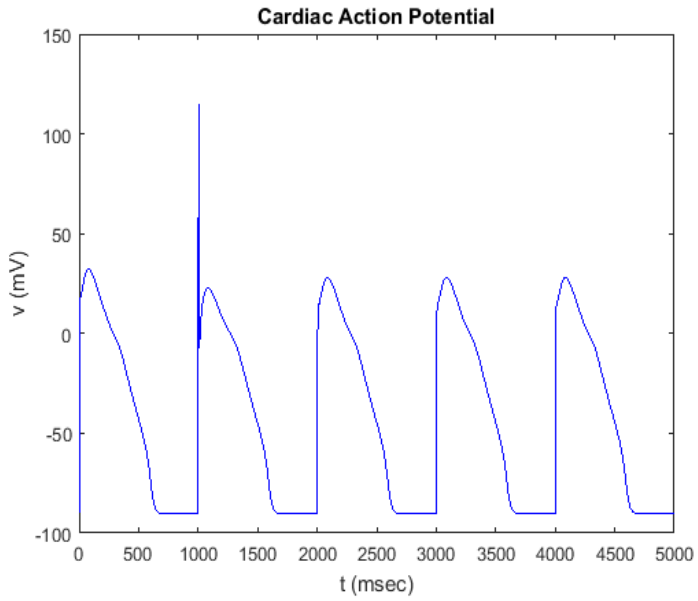

Figure 12: Reduced Priebe Beuckelmann Action Potentials

MICROCONTROLLER IMPLEMENTATION

Hardware Setup

In order to model neurons in hardware, we used the PIC32 microcontroller. In order to obtain the analog signal for our oscilloscope output, we connected the MCP4822 12-bit SPI DAC to the PIC32. At every sample, the current membrane voltage or chosen current is sent to the DAC via SPI. We use SPI channel 1 to write to the DAC. The output is taken at DAC output A. We initially noticed that there was significant SPI noise. As a result, we created a low pass filter with a 1 kΩ resistor and a .1 μF capacitor. This output is probed by the oscilloscope for the final output. The wiring diagram for the DAC can be seen in FIgure 13 below

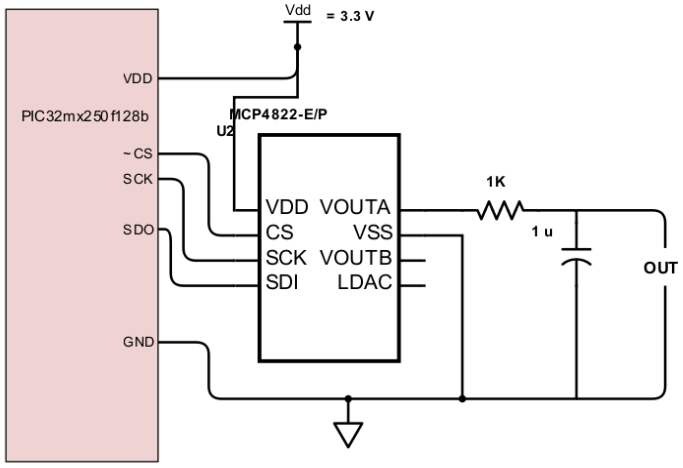

Figure 13: DAC wiring diagram

Hodgkin Huxley & Noble Models

The software on the PIC microcontroller strongly parallels the MATLAB models of the neurons. We first initialized variables representing the base conductance of the ion channels, the reverse potentials for each ion, the current injection value, the gating values, and the membrane voltage. In order to perform Euler integration, we maintained the values of the previous gating values and membrane voltage. All of these values were first initialized based on Hodgkin Huxley. After we finished creating a Hodgkin Huxley neuron, we modified all values for the Denis Noble Model. All of the variables were initially of the *_Accum* data type, which is the PIC32's default 15.16 fixed point type. We used this data type in order to minimize computation and reduce clock cycles. Using fixed point allowed us to remain as true to neurons as possible.

The bulk of the software can be broken down into three parts: the main method, the integration method, and helper functions to calculate the rate values α and β. In the main method, we initialized protothreads, SPI, and set the initial values of the membrane voltage and gating constants. We then called the integration thread.

The integration method uses the variable *delayTime* as the time delta for integration. In this method, all of the ion conductance are changed based on the appropriate equations from MATLAB modeling. The derivatives of the gating values and membrane voltages are then updated based on the updated conductance and the rate functions for each of the gating values. After all of these values are updated, the current membrane voltage and gating values are changed using Euler's method. Once the values are updated, the current membrane voltage is written to the DAC. A 12 bit DAC can only handle values between 0 and 2047. As a result, to get the best display, the membrane voltage is dc offset in order to make all values positive, and scaled to allow for better visualization on the display. The scaling was set in a trial and error fashion, with the offset decided from observation from the MATLAB results. Sending information to the DAC was accomplished through SPI. This loop repeats after every delay of the thread.

All of the rates α and β were calculated using helper functions. These functions directly mapped to the respective functions in the MATLAB model. The bulk of the calculations is done in these functions. In each of these functions, exponentiation is required. This was possible after importing "math.h" and using the *exp()* function. Since we decided to use the math.h header file, exponentiation required to be in floating point. As a result, in all of the helper functions, all exponentiation were done after casting the fixed point values to floating point. In addition, since division takes less clock cycles in floating point than in fixed point, all divisions were done in floating point as well. In the end, each of these functions output a fixed point value representing an α or β rate value. There were six total functions, one each for $\alpha_n, \beta_n, \alpha_h, \beta_h, \alpha_n$, and β_n as shown in the Appendix.

As mentioned before, we first made a working Hodgkin Huxley model. The oscilloscope output can be seen in Figure 14 and directly corresponds to the Hodgkin Huxley spiking in Figure 8.

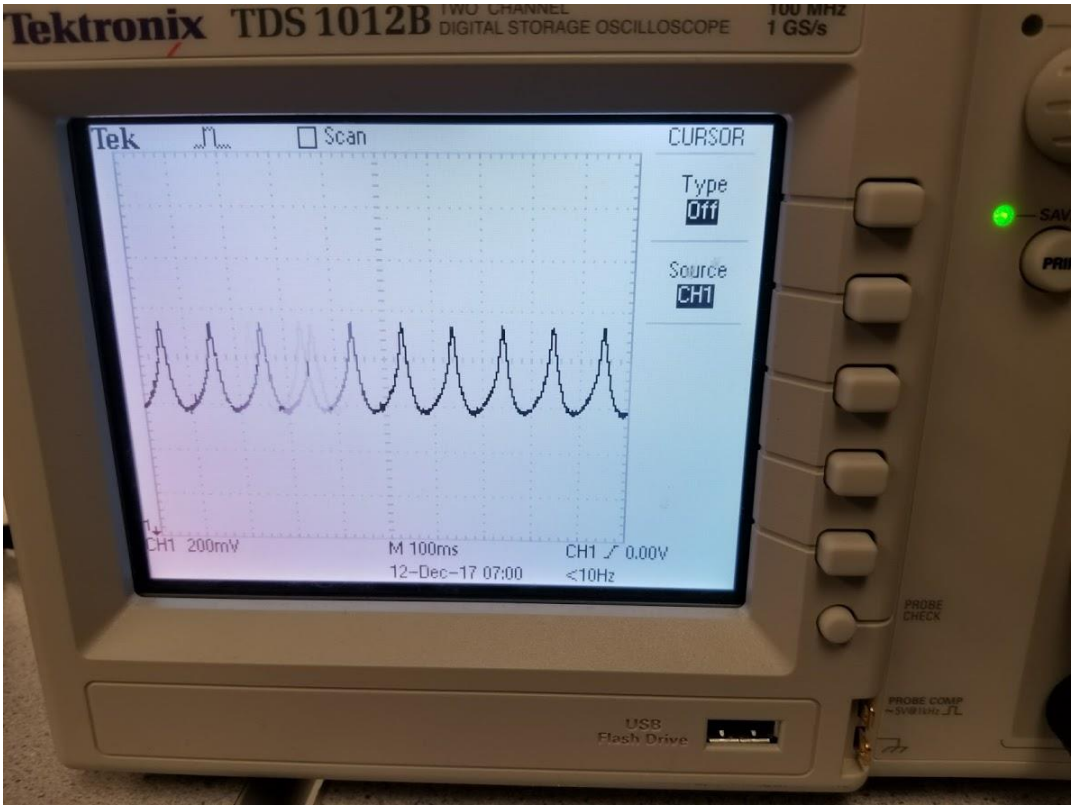

Figure 14: Hardware Model of Hodgkin Huxley Neurons

Once Hodgkin Huxley was working and tested, we implemented the Denis Noble model neurons. The output of these neurons is in Figure 15 and corresponds to the MATLAB model in Figure 9. In this model, we still do not have the tall spike or plateau as we would ideally want. However, it is clear that there is still a strong spike for depolarization.

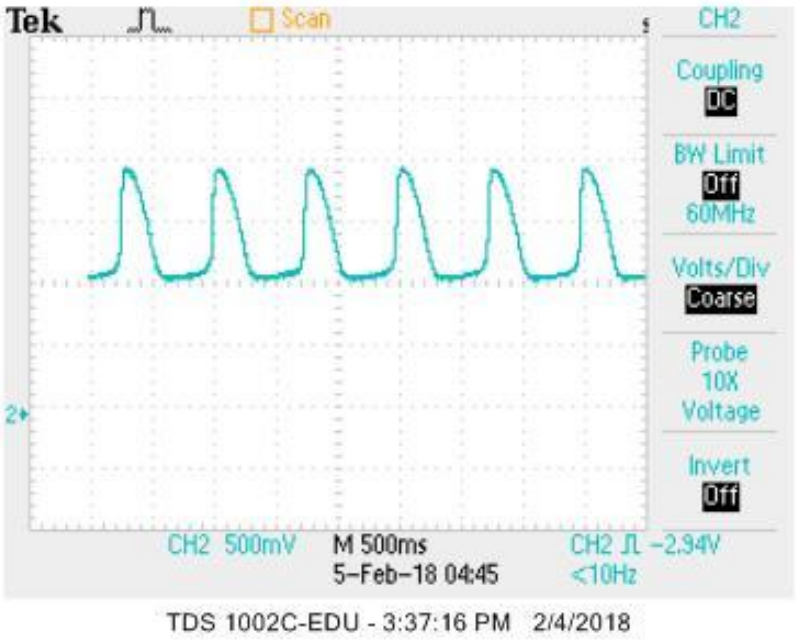

Figure 15: Hardware Model of Denis Noble Neuron

Reduced Priebe Beuckelmann Model

In this section, we primarily describe the implementation of the differential equations into C. Base conductance and concentrations are initially set from the user interface input. Constants as summarized in (Appendix 3, Section 1, Table 1) are initialized and calculated before running the simulation. After initializing constants, initial steady state values of gate variables are calculated. Afterwards, differential equations are solved based on Figure 16 below.

In the differential equation loop, we first store previous values of gating variables and voltage. These previous voltages are used in all calculations for updates. We then update all conductance using the equations in Appendix 3 for g_{Na}, g_K, g_{K1}, g_{Ca}, and g_{to}. We then calculate the rate functions α and $\beta$for all gating variables.Then, at the previous voltage, we calculate all steady state values for the gating variables. Next, we calculate the time constants for all gating variables whose steady state values are not explicitly used to calculate conductance (v_{rate}, *X, m, f,* and *to)*. Using the updated conductance, we calculate the current from each ion and ion exchange (sodium, fast potassium, slow potassium, transient outward, calcium, sodium-potassium exchange, and sodium-calcium exchange). After calculating currents, we sum the currents to calculate total current across the membrane, and integrate the current across the membrane to find the voltage. The updated membrane voltage is calculated as $V_m = V_{prev} - I_{tot} * dt/C_m$, where $V_m$is the membrane voltage, $I_{tot}$is the total current, dt is the change in time per iteration, and $C_m$is the membrane capacitance. At this point, the calculated membrane voltage and selected current is sent to the user interface in order to display on the TFT screen. In addition, since the minimum voltage is known to be -90 mV, the membrane voltage is offset by 90 and multiplied by 16 in order to send to the SPI DAC. This is necessary as the DAC does not handle negative values, and will show minute differences at the scale of the membrane voltages. As a result, the membrane voltage must be scaled by 16 to appear accurate.

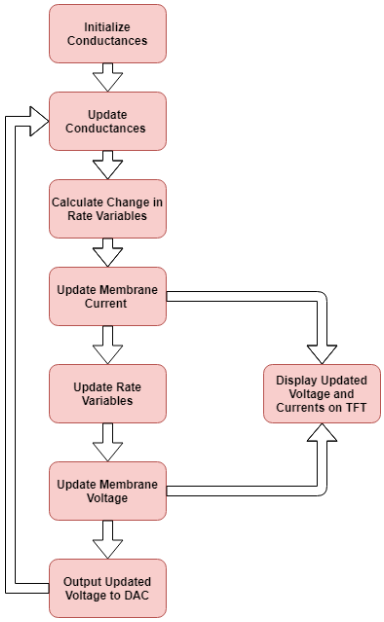

Figure 16: Flow of differential equation software

In comparison to the previous implementations of cell models, this model required far more sub-functions in order to implement in an organized fashion. While the Denis Noble model only required functions to calculate $\alpha_m, \beta_m, \alpha_n, \beta_n, \alpha_h$, and β_h, this implementation used functions for all α, β, steady state calculation, and time constant calculations, in addition to functions for other intermediate calculations. These functions are all described in Table 2 below. Between iterations, the update thread yielded. However, the thread did not yield with a set wait time. This ensured minimizing delay between calculations, and led to the update thread having an implicitly higher priority than all other threads.

Function Header	**Purpose**
alpha_d(voltage)	Calculate α_d for *d* gating variable based on current voltage
alpha_f(voltage)	Calculate α_f for *f* gating variable based on current voltage
alpha_k1(voltage)	Calculate α_{k1} for *k* gating variable based on current voltage
alpha_m(voltage)	Calculate α_m for *m* gating variable based on current voltage
alpha_r(voltage)	Calculate α_r for *r* gating variable based on current voltage
alpha_to(voltage)	Calculate α_{to} for *to* gating variable based on current voltage
beta_d(voltage)	Calculate β_d for *d* gating variable based on current voltage
beta_f(voltage)	Calculate β_f for *f* gating variable based on current voltage

beta_k1(voltage)	Calculate β_{k1} for *k1* gating variable based on current voltage
beta_m(voltage)	Calculate β_{m} for *m* gating variable based on current voltage
beta_r(voltage)	Calculate β_{r} for *r* gating variable based on current voltage
beta_to(voltage)	Calculate β_{to} for *to* gating variable based on current voltage
calcNaCaCurrent(voltage,temperature, g_{NaCa}, Km_{Na}, $[Na]_e$, $[Na]_i$, Km_{Ca}, k_{sat}, $[Ca]_e$ η)	Calculate sodium-calcium exchange current, based on sodium permeability, sodium intra and extracellular concentrations, extracellular calcium concentration and constant η
calcNaKCurrent(voltage, sigmaNaK, temperature, g_{NaK})	Calculate sodium-potassium exchange current based on current voltage, constant σ_{NaK}, current temperature, and sodium-potassium exchange conductance
E_to($[Na]_e$, $[K]_e$, $[Na]_i$ $[K]_i$, temperature)	Calculate equilibrium potential of transient outward currents based on sodium and potassium concentrations as well as temperature
equlibriumPotential($[ion]_i$, $[ion]_e$, ion charge, temperature)	Calculate equilibrium potential based on ion concentrations, charge, and temperature
expDTChange(dt, τ)	Calculate change in variable exp(-dt/τ) as mentioned in MATLAB section
f_Ca($[Ca]_i$, K_{Ca})	Calculate calcium constant gating variable f_{Ca}
fNaK(voltage, sigmaNaK, temperature)	Calculate sodium-potassium pump gating variable f_{NaK} based on voltage, σ_{NaK}, and temperature
f_primeNaK($[K]_e$, $[Na]_i$)	Calculate intermediate step gating variable for sodium potassium pump f'_{NaK}based on extracellular potassium concentration and intracellular sodium concentration
gateSteadyState(α, β)	Calculate steady state value for gating variable given its $\alpha$and β
gateTau(α, β)	Calculate time constant for gating variable given its $\alpha$and β
getRateofChangeK(voltage, DT, τ_X)	Potassium rate of change is an exception to all other ions. This function calculates its change
sigma_nak($[Na]_e$)	Calculate the constant σ_{NaK}based on sodium concentration
tau_v(voltage)	Calculate time constant for gate variable v_{rate}
tau_x(voltage)	Calculate time constant for gate variable X
v_inf(voltage)	Calculate steady state value for gate variable v_{rate}
X_inf(voltage)	Calculate steady state value for gate variable X
fixedPointExp(n)	Calculate *long _Accum* fixed point exponentiation of n
fixedPointTanh(n)	Calculate *long _Accum* fixed point hyperbolic tangent of n

Table 2: All functions involved in Reduced Priebe Beuckelmann implementation. Note that function headers use symbols to imply definition and are written out based on appropriate data types

In Table 2, we do not include data types since we ended up implementing all numbers in two different data types. In our first pass we implemented all numbers using the *long _Accum* data type. This is the PIC32's 31.32 fixed point data type. This was chosen instead of *_Accum* because this model required greater precision in order to be accurate and stable. However, none of the exponentials were solved in fixed point. Instead functions *fixedPointExp* and *fixedPointTanh* were used to handle exponential functions. These functions casted the input to a *double*, used the math library to calculate exp() or tanh() respectively, casted the value back to *long _Accum*, and returned the result. This method of exponentiation was effective in the previous models to speed up calculation.

However, in the newer model, there is a total of 55 exponentials that have to be calculated. As a result, the time to cast values as *double* and to cast back to *long _Accum* eclipsed the benefits of clock cycles saved using fixed point. For this reason, we replaced all values of the *long _Accum* data type with *double* data types. Calculations now only used the *exp()* and *tanh()* functions inbuilt into *math.h*. This design choice led to the benefit of spikes appearing on the oscilloscope once every 4 seconds to once every second, which is much more accurate to biology. More importantly, this design choice allowed the frequency of firing to be able to completely be set by the sampling interval *dt*. The final *dt* chosen was .03 ms, which while coarse in calculation, allowed for an accurate shape of action potential without an unstable differential equation. The final oscilloscope output of the reduced Priebe-Beuckelmann model of a cardiac cell at standard human cardiac cell settings can be seen in Figure 17 below.

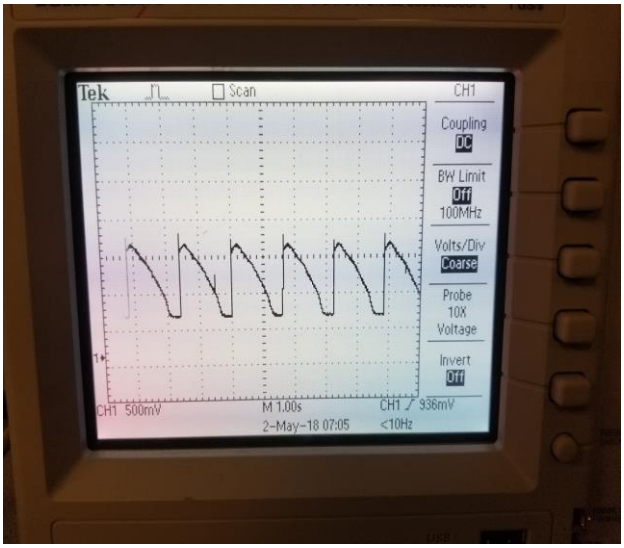

Figure 17: Oscilloscope capture of reduced Priebe Beuckelmann model of cardiac potentials.

USER INTERFACE

The user interface was designed to be as intuitive as possible, with directional buttons used to navigate the screen and a sliding potentiometer used to change on-screen values. The overall block diagram of the user interface can be seen in Figure 18 below.

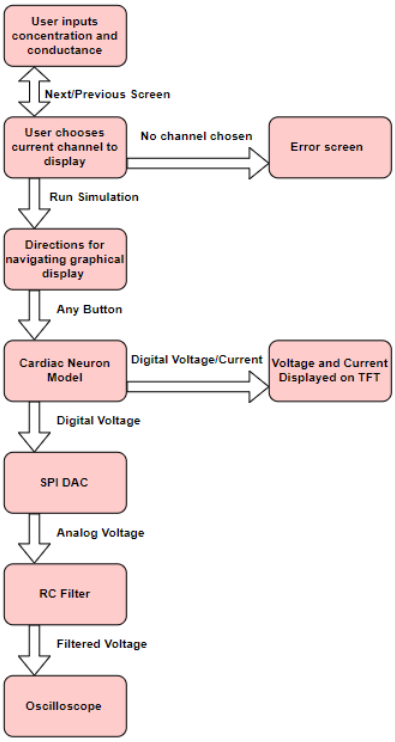

Figure 18: Block diagram of user interface

The first screen allows the user to input the concentrations and conductance for the Sodium, Potassium, and Calcium ion channels. When satisfied with their values, the second screen allows the user to choose which current channel (Sodium, Potassium, or Calcium) to display on the TFT as well as what should display on the oscilloscope (voltage or current). If no choice is presented for both the TFT and the oscilloscope, an error screen appears (the 'fourth' screen) and the user is prompted to return back and provide options. Once a current channel is chosen, a Lab Directions screen is shown, explaining basic functionality of the buttons (the third screen). When the user is ready, any button press will lead them to the TFT Oscilloscope screen, displaying the membrane voltage and chosen current channel over time with the given concentrations and conductance. Simultaneously, the voltage or current is sent through an SPI DAC and RC filter to an oscilloscope for data analysis.

Protothreads, a lightweight stackless threading library, was used to provide linear code execution with the help of a round-robin scheduler. The use of Protothreads allowed the sequential flow of control without the use of complex state machines or full multi-threading.

The user interface code is separated into three separate Protothreads: *protothread_button, protothread_update,* and *protothread_tft*.

Protothread_tft

Protothread_tft handles the first four screens of the program.

The first screen is designed for the user to set the Sodium, Potassium, and Calcium concentrations and conductance values. The up, down, left, and right buttons allow the user to navigate the screen while the sliding potentiometer allows the user to change any value. The in-built analog-digital converter on the PIC-32 is used to sample the values received from the sliding potentiometer. When on a certain variable (i.e. Sodium concentration), that text will highlight red with all the other text on the screen being yellow. The center button locks in the chosen number for a given variable which is then used for the differential equation model. The 'Next screen' on-screen button at the bottom of the screen proceeds to the second screen. A sample screenshot of the first screen can be seen in Figure 19 below.

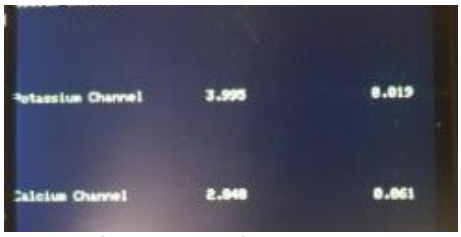

Figure 19: First Screen

The second screen is designed for the user to choose the current channel they wish to display (Sodium, Potassium, or Calcium) and what the oscilloscope should output (voltage or current). The up, down, left, and right arrows allow the user to navigate the screen with the center button locking in the user's choice. When a choice is made, that text will highlight red or cyan with all the other text on the screen being yellow. The 'Previous Screen' and 'Run Simulation' buttons allow the user to either return to the first screen or move onwards with their choices. A sample screenshot of the second screen can be seen in Figure 20 below.

Figure 20: Second screen

The third screen simply displays text involving the functionality of the fifth screen (i.e. the TFT oscilloscope). In the future, this screen could involve more complex lab directions as well. If no current channel is chosen within the second screen, a fourth screen exists as an 'error screen', simply telling the user to return back and choose a current channel to display. In either case, any button press will lead the user to the correct screen. A sample screenshot of the third and fourth screens can be seen in Figure 21 below.

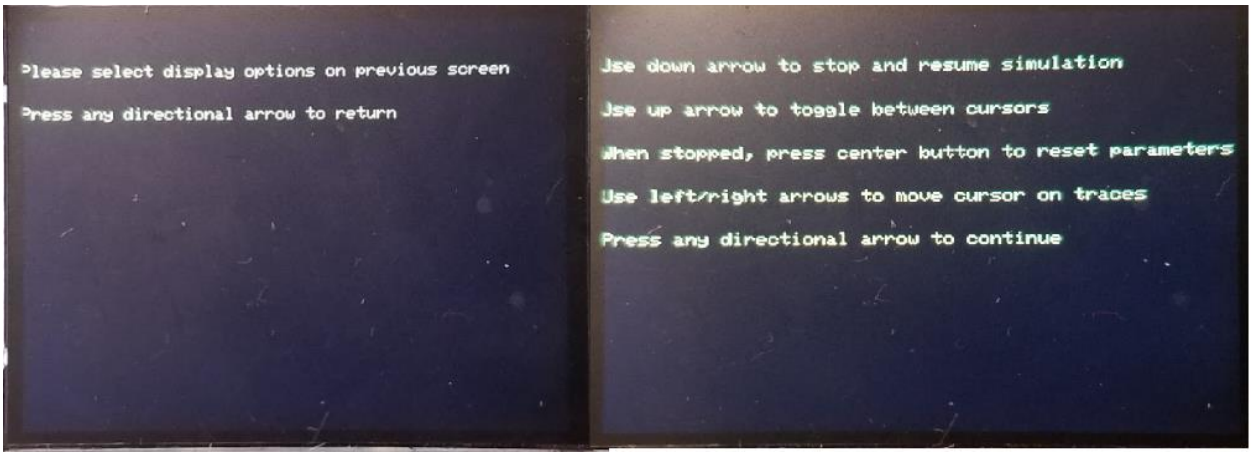

Figure 21: Error Screen (Left). Directions Screen (Right)

Protothread_button

Protothread_button reads the bits from the GPIO pins connected to the left, right, up, down, and center buttons. If a button seems to be pressed, the below finite state machine (FSM) is run for that button. The FSM below ensures that double button presses are not recorded and the "button experience" is well-tuned for the user.

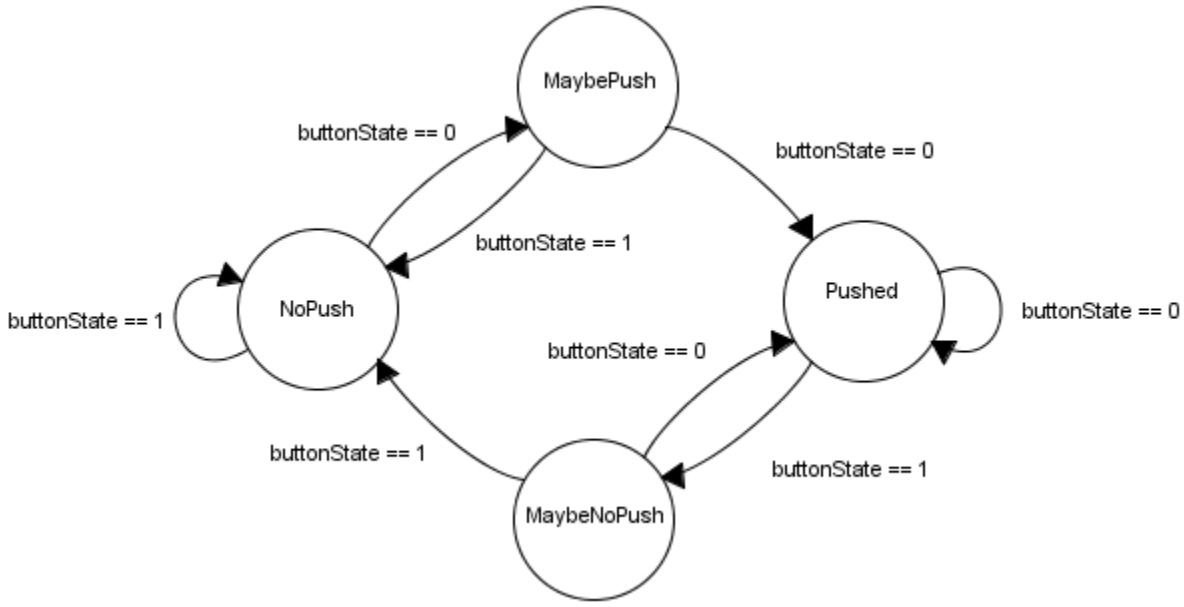

Figure 22: Finite state machine for debouncing buttons

The finite state machine begins in the 'NoPush' state by checking to see if the *buttonState* variable is equal to 0 (i.e. a button may have been pushed). If this is true, the *pushState* variable is set to 'MaybePush'. If it is not true, the *pushState* stays as 'NoPush'. Within the 'MaybePush' state', the *buttonState* is once again checked. If *buttonState* is still equal to 0, we know the button has been pushed and *pushState* becomes 'Pushed'. If *buttonState* is not 0, we go back

to ‘NoPush’. We do another check of *buttonState* within the ‘Pushed’ state to see if the button is being held down or released. If *buttonState* is still 0, we stay in ‘Pushed’, otherwise we go to ‘MaybeNoPush’. Regardless of the *buttonState* in the ‘Pushed’ state, the button is recorded as having been pushed at least once through the *buttonValue* variable. Within ‘MaybeNoPush’, we simply check *buttonState* to return to either ‘NoPush’ or ‘Pushed’ depending on whether the button has been released or held down respectively.

Protothread_update

Protothread_update runs only during the fifth screen (i.e. the main TFT ‘oscilloscope’ screen). This thread checks the down buttons and pauses the screen if it is pressed. If the screen is paused, two blue vertical cursors are placed on the left hand-side of the screen and the voltage, current, time value, and difference in time values of the cursors’ position are displayed on the top right hand side of the screen (see Figure 23 below). When on the pause screen, the cursor location can be moved with the left and right buttons. Time is recorded as 0 at the left-hand side of the screen and is in milliseconds**.** The up button can be used to toggle between cursors. When paused, the center button can be used to exit the screen and return to the second screen (choosing which ion current channel to display).

Figure 23: Paused TFT Oscilloscope Screen

When not paused, all of the previous and next differential equation model parameters are updated. A current stimulus is provided and a new membrane voltage and total current is calculated. The voltage and current are plotted on the screen every 25th iteration of the thread through the variable *sampling*. This variable can be adjusted in the code to allow more or less accuracy to the model. More accuracy will lead to less waveforms on the screen and vice versa.

In order to obtain the analog signal for the oscilloscope, we connected the MCP4822 12-bit SPI DAC to the PIC32. At every iteration of the update thread, the current membrane voltage or total current is sent to the DAC via a Serial-Peripheral Interface (SPI) bus. We use SPI channel 2 on the microcontroller to write to the DAC with the output taken out of DAC output A. Initial testing showed a significant amount of noise on the output from the SPI DAC. As a result, we created a low pass filter with a 10 kΩ resistor and a 1 μF capacitor to reduce this noise. This output is probed by the oscilloscope for the final output.

INTEGRATION

Code Structure

The first step of integration was combining each element of our code together. The differential equation model was placed into *differential_equations.c* with a corresponding *differential_equations.h* acting as a header file for function prototypes. *Cardiac_neuron.c* and *Cardiac_neuron.h* acts as the main file containing the various threads, the round-robin scheduler, and all high-level screen logic. This file was designed to be as sparse and clean as possible so future engineers can understand the structure of the code without delving into the details. Those details are placed in the *Helper_functions.c* and *helper_functions.h* files with contain the function definitions called by *Cardiac_neuron.c*. Several variables such as the membrane voltage and ion channel currents are externed within header files so that they may be accessed globally.

Printed Circuit Board

PCBExpress software was used to create a schematic and PCB of our product. The final schematic and PCB design can be seen in Figures 24 and 25 below. A labeled image of the physical image can be seen in Figure 1.

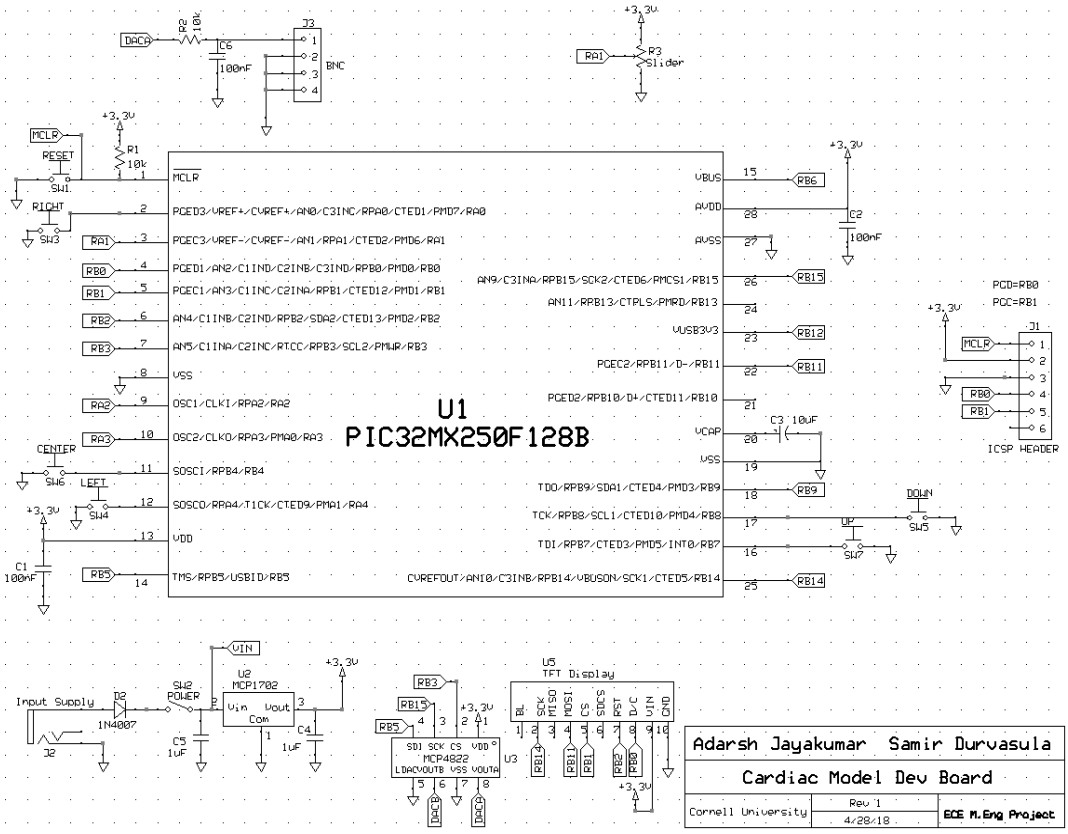

Figure 24: Schematic of Final Board

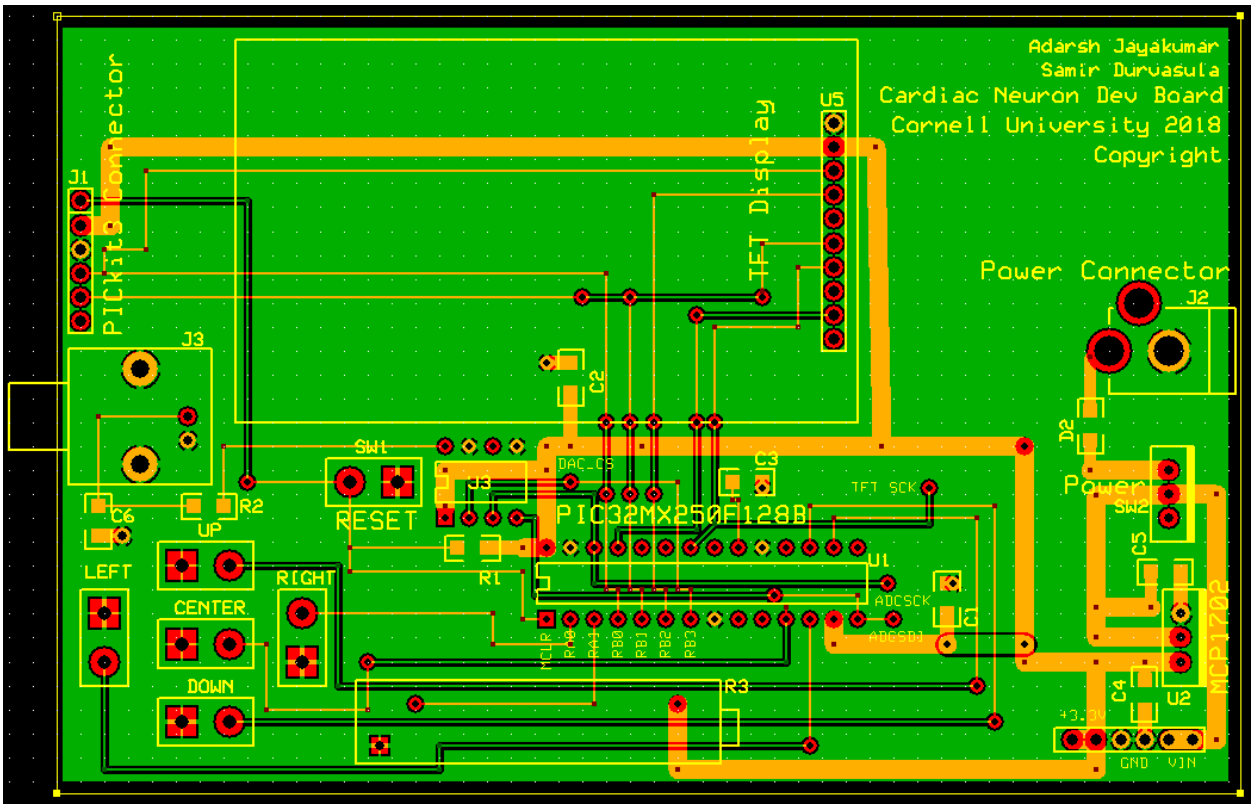

Figure 25: PCB Design

FUTURE WORK

Several pieces of future work remain to be explored. Firstly, minor user interface errors still exist including occasionally “sticky” buttons as well as a sliding potentiometer that does not fit flat on the board. However, more needs-based design problems may surface when students and instructors interface with the product. Secondly, there are a large amount of variables that could be user programmable that are currently not set to be in the code. For example, making the frequency of current stimulus tunable may be an interesting avenue as it would change the frequency of the heartbeat. Thirdly, extensive testing needs to be done to ensure that all of the bounds for voltage and current produce reasonable waveforms on the oscilloscope and TFT screens. Fourthly, unnecessary long delays exist in the code for button debouncing and TFT displaying. These delays could certainly be reduced to their absolute minimum for optimal performance. Finally, more lab handouts could be created in order to provide students and teachers the breadth of knowledge that our product can showcase.

CITATIONS

Bernus, O., Wilders, R., Zemlin, C. W., Verschelde, H., & Panfilov, A. V. (2002). A computationally efficient electrophysiological model of human ventricular cells. *American Journal of Physiology-Heart and Circulatory Physiology*, *282*(6), H2296-H2308.

Chudler, E. (n.d.). Neuroscience For Kids. Retrieved from https://faculty.washington.edu/chudler/ap.html

Electrolytes and the Heart. (2015, February 10). Retrieved from https://www.resus.com.au/2015/02/10/electrolytes-and-the-heart/

Dunkels, Adam, et al. "Protothreads." *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems - SenSys 06*, 2006, doi:10.1145/1182807.1182811.

Fenton, F. H., & Cherry, E. M. (n.d.). Models of cardiac cell. Retrieved from http://www.scholarpedia.org/article/Models_of_cardiac_cell

Fink, M., & Noble, D. (2006). Noble model. Retrieved from http://www.scholarpedia.org/article/Noble_model

Grandi et al. "Theoretical investigation of action potential duration dependence on extracellular Ca2+ in human cardiomyocytes" Journal of Molecular and Cellular Cardiology. *Journal of Molecular and Cellular Cardiology.* Volume 46, Issue 3, March 2009, 332-342

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, *117*(4), 500-544.

Ikonnikov, Greg and Yelle, Dominique. "Physiology of cardiac conduction and contractility." *Clin Anat*, Can J Anaesth. 40 Nov. 1993

Kazama I. High-calcium exposure to frog heart: a simple model representing hypercalcemia-induced ECG abnormalities. *The Journal of Veterinary Medical Science*. 2017;79(1):71-75. doi:10.1292/jvms.16-0413.

Luo, and Y Rudy. "A Model of the Ventricular Cardiac Action Potential. Depolarization, Repolarization, and Their Interaction." *Circulation Research*, American Heart Association, Inc., 1 June 1991

Mahbub, Tahmid Syed. "Interfacing a Color TFT Display with the PIC32MX250F128B." *Tahmid's Blog: Microcontroller and Power Electronics*, tahmidmc.blogspot.com/2014/10/interfacing-color-tft-display-with.html.

McCullough, Peter et al. "Acute and Chronic Cardiovascular Effects of Hyperkalemia: New Insights Into Prevention and Clinical Management" *Cardiovascular Medicine*. 2014. Vol 15, Issue 1, 11-23**.**

Noble, D. (1962). A modification of the Hodgkin—Huxley equations applicable to Purkinje fibre action and pacemaker potentials. *The Journal of physiology*, *160*(2), 317-352.

Parham WA, Mehdirad AA, Biermann KM, Fredman CS. Hyperkalemia Revisited. *Texas Heart Institute Journal*. 2006;33(1):40-47.

Priebe, L., & Beuckelmann, D. J. (1998). Simulation study of cellular electric properties in heart failure. *Circulation research*, *82*(11), 1206-1223.

Ramanathan, Charulatha, et al. "Activation and Repolarization of the Normal Human Heart under Complete Physiological Conditions." *PNAS*, National Academy of Sciences, 18 Apr. 2006

Silverthorn, D. U. (2013). *Human physiology*. Harlow: Pearson Education.

Wyttenbach RA, Johnson BR, Hoy RR. (2011). Stretch Reception: A Model for Muscle Spindle Organs. Crawdad: An Online Lab Manual for Neurophysiology.

APPENDICES

Appendix 1: Hodgkin Huxley Constants and Equations (Hodgkin and Huxley, 1952)

x	$E_x\ (mV)$	$g_x\ (ms/cm2)$
Na	115	120
K	-12	36
$Leak$	10.6	0.3

Table 1: Base conductance and equilibrium potentials in Hodgkin Huxley

$$\sum I_k = g_{Na} m^3 h(V - E_{Na}) + g_K n^4 (V - E_K) + g_L (V - E_L) \quad (1)$$

$$\frac{dm}{dt} = \alpha_m(V)(1 - m) - \beta_m(V) * m \quad (2)$$

$$\frac{dn}{dt} = \alpha_n(V)(1 - n) - \beta_n(V) * n \quad (3)$$

$$\frac{dh}{dt} = \alpha_h(V)(1 - h) - \beta_h(V) * h \quad (4)$$

x	α_x	β_x
m	$\frac{.1(25 - V)}{exp(\frac{-V - 48}{15}) - 1}$	$4exp(\frac{-V}{18})$
h	$.07exp(\frac{-V}{20})$	$\frac{1}{1 + exp(\frac{30 - V}{10})}$
n	$\frac{.01(10 - V)}{exp((10 - V)/10) - 1}$	$0.125exp(\frac{-V}{80})$

Table 2: Change of rate functions

Appendix 2: Denis Noble Model Constants and Equations (Noble, 1968)

x	Ex	gx
Na	40	400
$K1$	-100	1.2
$K2$	-100	$1.2 * exp((-V-90)/50) + 0.015 * exp((V+90)/60)$
$Leak$	-60	.75

Table 1: Reversal Potential and Initial conductance for ion channels in Denis noble model

x	α_x	β_x
m	$\frac{.1(-V-48)}{exp(\frac{-V-48}{15})-1}$	$\frac{.12(V+8)}{exp(\frac{V+8}{5})-1}$
h	$0.17 * exp(\frac{-V-90}{20})$	$\frac{1}{1+exp(\frac{-V-42}{10})}$
n	$\frac{.1(-V-50)}{exp(\frac{-V-50}{10})-1}$	$2 * exp(\frac{-V-90}{80})$

Table 2: Rate variables changed for Denis Noble model

Appendix 3: Reduced Priebe Beuckelmann Equations (Bernus et. al, 2002)

Section 1: Base Conductances and Concentrations

Ion	Conductance (mS/cm^2)
Na(g_{Na})	16
Ca (g_{Ca})	0.064
Fast K (g_K)	0.4
Slow K (g_{K1})	0.019
Transient Outward K (g_{to})	3.9
Background Na ($g_{Na,b}$)	0.001
Background Ca ($g_{Ca,b}$)	0.00085
Na K Exchange (g_{NaK})	1.3
Na Ca Exchange (g_{NaCa})	1000

Table 1: Base Conductances

Ion	**Intracellular (mM)**	**Extracellular (mM)**
Na	0.0004	2
Ca	10	138
K	140	4

Table 2: Base Concentrations

Constant	**Value**
K_{Ca}	0.0006
Km_{Ca}	1.38
Km_{Ke}	1.5
Km_{Na}	87.5
Km_{Nai}	10.0
k_{sat}	.1

Table 3: Permeabilities

Section 2: Overall Current and Gate Variable Calculations:

x	i_x
Na^+	$g_{Na} * v^2_{gate} * m^3(V_m - E_{Na})$
K^+	$g_{K1} * X^2 * (V_m - E_K)$
K_1^+	$g_{k2} * k_1(V_m - E_K)$
K^+_{To}	$g_{TO} * r_\infty * to(V_m - E_{TO})$
Ca^{2+}	$g_{Ca} * d_\infty * f * f_{Ca}(V_m - E_{Ca})$
Ca^{2+}, b	$g_{Ca,b} * (V_m - E_{Ca})$
Na^+, b	$g_{Na,b} * (V_m - E_{Na})$
$Na - K$	$g_{NaK} * f_{NaK} * f'_{NaK}$
$Na - Ca$	$g_{NaCa} * f_{NaCa}$

Table 4: Calculations for all currents

For a given gate variable, x

$$x(t) = x(t-1) + (x(t-1) - x_\infty(t-1)) * exp(-dt/\tau_x) \quad (1)$$

Section 3: Equilibrium Potentials

$$E_{ion} = \left(\frac{RT}{NF}\right) ln\left(\frac{[Ion]_e}{[Ion]_i}\right) \quad (2)$$

$$E_{to} = \left(\frac{RT}{F}\right) ln\left(\frac{0.043 * [Na^+]_e + [K^+]_e}{0.043 * [Na^+]_i + [K^+]_i}\right) \quad (3)$$

Section 4: Steady State Equations and Exceptions

In general, for a given gating variable x

$$x_\infty = \frac{\alpha_x}{\alpha_x+\beta_x} \quad (4)$$

$$\tau_x = \frac{1}{\alpha_x+\beta_x} \quad (5)$$

The following equations are exceptions to the above steady state and time constant values.

$$v_{rate\infty} = 0.5[1 - tanh\,(7.74 + 0.12V_m)\,] \quad (6)$$

$$\tau_{v_{rate}} = 0.25 + 2.24\left(\frac{[1-tanh\,(7.74+0.12V_m)\,]}{\{1-tanh\,[0.07(V_m+92.4)]\,\}}\right) \quad (7)$$

$$X_\infty = \frac{0.988}{1+exp\,(-0.861-0.0620V_m)} \quad (8)$$

$$\tau_X = 380 * exp\,[-\frac{(25.5+V_m)^2}{156}] + 182 * [1 + tanh\,(0.154 + 0.0116V_m)] + 40 * [1 - tanh(160 + 2 * V_m)] \quad (9)$$

Section 5: Rate Functions

x	α_x
m	$\frac{[0.32(V_m+47.13)]}{\{1-exp\,[-0.1(V_m+47.3)]\,\}}$
d	$\frac{14.98exp\,\{-0.5[\frac{V_m-22.36}{16.68}]^2\}}{16.68\sqrt{2\pi}}$
f	$\frac{6.87e-3}{1+exp\,[-\frac{6.1546-V_m}{6.12}]}$
r	$\frac{0.5266exp\,[-0.0166(V_m-42.2912)]}{1+exp\,[-0.0943(V_m-42.2912)]}912)t$
to	$\frac{5.612e-5*V_m+0.0721exp\,[-0.173(V_m+34.2531)]}{1+exp\,[-0.1732(V_m+34.2531)]}$
$K1$	$\frac{0.1}{1+exp\,[0.06(V_m-E_K-200)]}$

Table 5: Calculation of for all gate variables

x	β_x
m	$0.08exp\,(-\frac{V_m}{11})$
d	$0.1471 - \frac{(5.3exp\,\{-0.5[\frac{V_m-6.27}{14.93}]^2\})}{14.93\sqrt{2\pi}}$
f	$\frac{0.069exp\,[-0.11(V_m+9.825)]+0.011}{(1+exp\,[-(0.278(V_m+9.825))])} + 5.75e-4$
r	$\frac{(5.186e-5*V_m+0.5149exp\,[-0.1344(V_m-5.0027)])}{1+exp\,[-0.1348(V_m-5.186e-5)]}$
to	$\frac{1.215e-4*V_m+0.0767exp\,[-1.66e-9*(V_m+34.0235)]}{1+exp\,[-0.1604(V_m+34.0235)]}$
$K1$	$\frac{3exp\,[2e-4(V_m-E_K+100)]+exp\,[0.1(V_m-E_K-10)]}{1+exp\,[-.5(V_m-E_K)]}$

Table 6: Calculation of $Beta$ for all gate variables

Section 6: Exchanger Current Gating Variables

$$f_{NaK} = \frac{1}{1+0.1245exp\,(-0.0037V_m)+0.0365\sigma_{NaK}exp\,(-0.037V_m)} \quad (10)$$

$$f'_{NaK} = \frac{1}{1+\sqrt[3]{(\frac{Km_{Na_i}}{[Na^+]_i})}} * \frac{[K^+]_e}{K_{m_{K_e}}[K^+]_e} \quad (11)$$

$$f_{NaCa} = \frac{[Na^+]_i^3[Ca^{2+}]_e exp\,(0.013V_m) - [Na^+]_e^3[Ca^{2+}]_i exp\,(-0.024V_m)}{(Km_{Na}^3+[Na^+]_e^3)(K_m+[Ca^{2+}]_e)(1+k_{sat}exp\,(-0.024V_m))} \quad (12)$$

Section 7: Voltage Independent Variables

$$f_{Ca} = \frac{1}{1+\frac{[Ca^{2+}]_i}{0.0006}} \quad (13)$$

$$\sigma_{NaK} = 0.1428[exp\,(\frac{[Na^+]_e}{67.3}) - 1] \quad (14)$$

Appendix 4: Possible Lab Handout 1

Lab Handout

Introduction

We have learned through the course of the previous labs that neurons exhibit excitable behavior resulting from opening and closing of ion channels. However, excitability is not limited to cells in the nervous system. Cells elsewhere in anatomy contain ion channels and exhibit action potential behavior as well. One such system is the cardiac system. In this lab, we will explore the ionic basis of cardiac action

potentials. Cardiac action potentials are unique in that they act like the central pattern generators of invertebrates. These cells spontaneously fire and are directly related to mammalian EKG.

Invertebrates do not have hearts. For this reason, we will not be able to use a dissection or recordings to understand cardiac cells. Instead, we will be using an electrical simulation device to act is we have a heart available. This system allows us to explore the ionic basis of Cardiac Action potentials by being able to adjust ion concentrations and conductance in the cell.

How to use the device:

Before use, plug in the device using a 5 V DC adaptor. Connect a BNC cable from the device into a neural recording device as well as into an oscilloscope.

On the first screen, use the up, down, left, and right buttons to navigate. Use the center button to select the appropriate conductance or concentration. These include sodium, potassium, and calcium. To change the value, use the slider. Once completed, press the "down" arrow until you reach the "Next Screen" option. Press the center button to continue.

On the second screen, use the up and down buttons to choose whether to display and record sodium, potassium, or calcium currents. Select as in the first screen. Additionally, use the directional arrows to select whether membrane voltage or current will display on the oscilloscope. Once selected, scroll down to the "Next Screen" option. Press the center button continue.

On the third screen, you will be given a set of instructions on how to interpret the final output. Press any button to continue.

On the final screen, you will see the resulting waveform. The red waveform represents the membrane voltage and the green waveform represents the selected ion current. Press the down button to pause. To move the cursors, use the right and left arrows. The current time, voltage, and current will be displayed in yellow letters. To toggle between cursors, use the up button. When paused, use the center button to exit the screen and re-enter values.

Experiments:

Human Ventricle Cell:

For the first experiment, adjust settings to the default values for the human ventricle cells (Bernus et al, 2002).

Ion	Concentration (mM)	Conductance (mS)
Na	138	16
K	4	0.018
Ca	1.5	0.64

Table 1: Human Ventricle Cell Conductance

Record the membrane voltage, and record the current contribution from each ion. How does each ion contribute to the final waveform?

Change Concentrations:
For each ion, gradually increase and decrease concentrations for each ion. How does each ion change and affect action potential shape?

Example Solution for Sodium Concentration

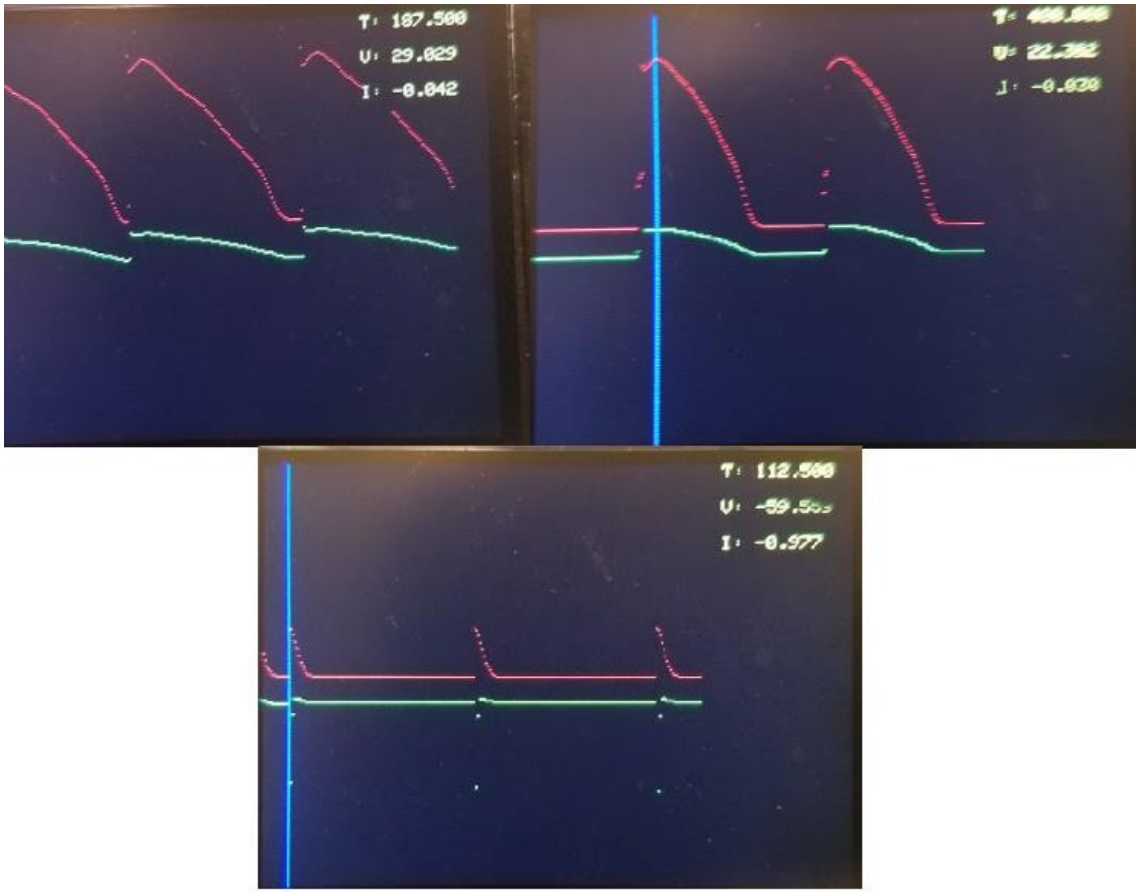

Figure 1: TFT screenshots with decreasing Sodium concentration
Top left: 138. Top right: 70. Bottom: 67

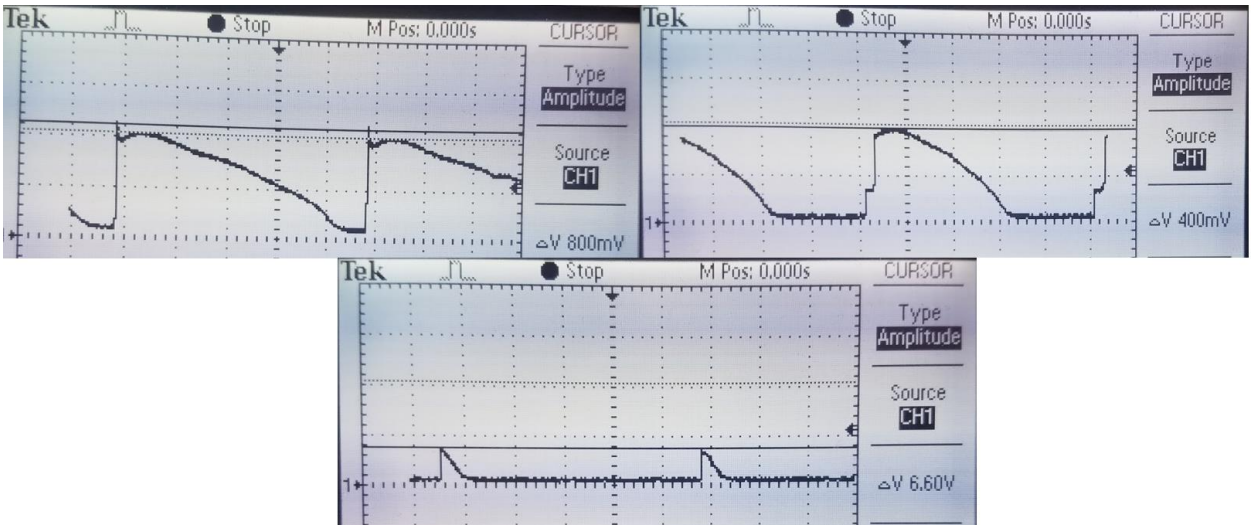

Figure 2: Oscilloscope screenshots with decreasing Sodium concentration
Top left: 138. Top right: 70. Bottom: 67

Figures 1 and 2 above show the membrane voltage and Sodium current waveforms as the Sodium concentration is decreased from 138 to 70 to 67 mM. As the Sodium concentration decreases, the depolarizing spike gets smaller until it can no longer fire. On the TFT, the spike decreases from 29 mV to 22 mV to -59 mV.

This behavior occurs due to another phase within the heartbeat. Action potentials are triggered in a neighboring pacemaker cell causes the membrane potential to rise above -90 mV. Fast Sodium ion channels start to open as Sodium leaks into the cell, raising the membrane potential to the “threshold potential” (~59 mV). This point is where enough fast Sodium channels have opened to generate a self-sustaining inward Sodium current, leading to a depolarization spike.

The value of 67 mM is the concentration at which there are too few Sodium channels to reach the “threshold potential”. Thus, at this Sodium concentration, the cell reaches -59 mV but cannot generate the self-sustaining inward current and subsequent depolarization spike. Of particular interest is the stark difference in action potentials between the 70 mM and 67 mM Sodium concentration.

Change Conductance:
For each ion, gradually increase and decrease concentrations for each ion. How does each ion change and affect action potential shape?

Example Solution for Calcium Conductance

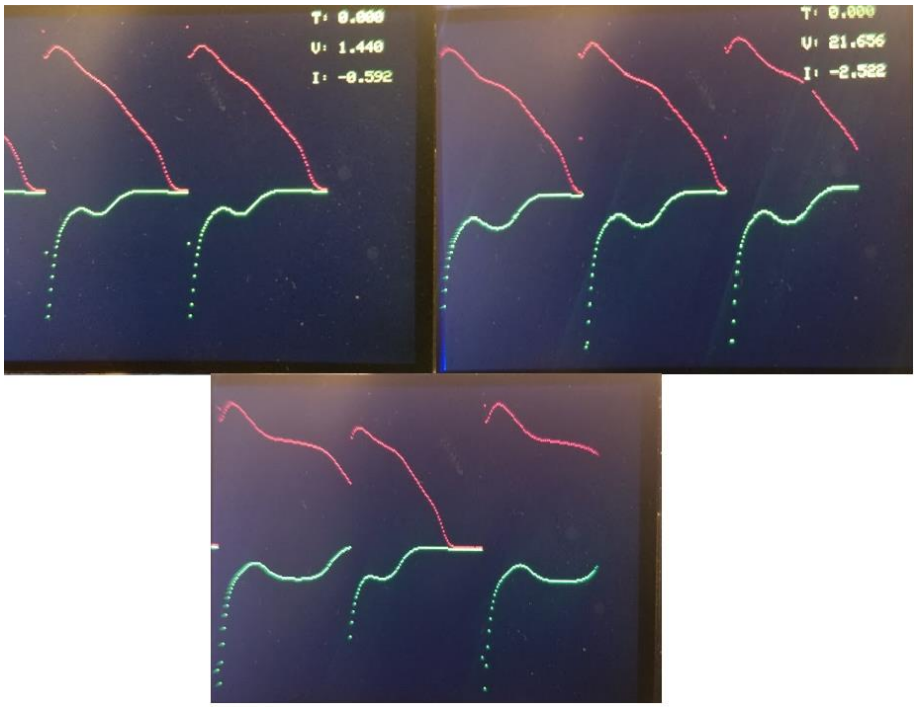

Figure 3: TFT Screenshots with increasing calcium conductance.
Top left: 0.64. Top right: 0.84. Bottom: 1.0

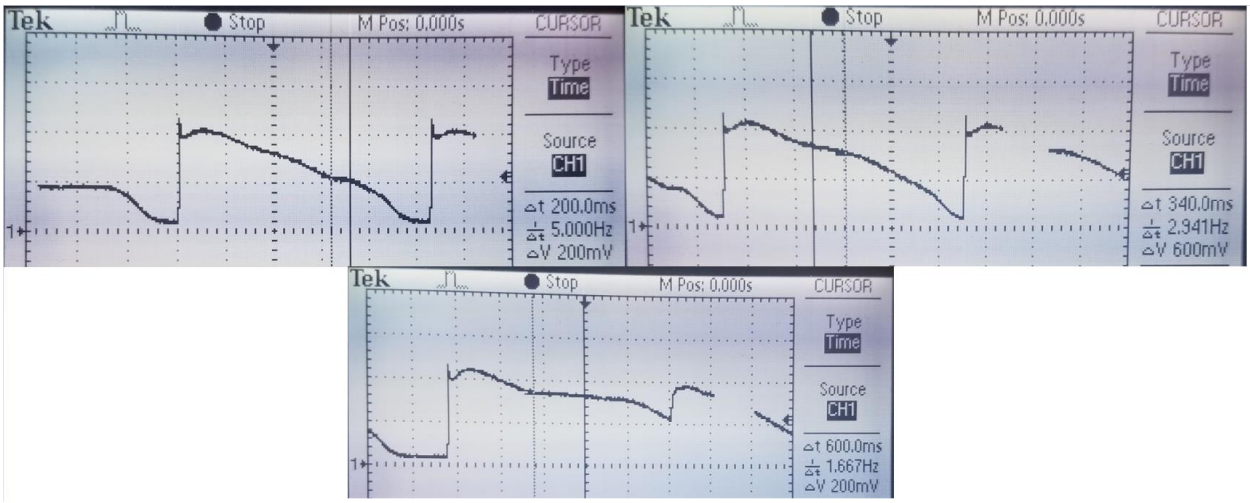

Figure 4: Oscilloscope screenshots with increasing calcium conductance.
Top left: 0.64. Top right: 0.84. Bottom: 1.0

Figures 3 and 4 above show the membrane voltage and calcium current waveforms as the Calcium conductance is increased from 0.64 to 0.84 to 1.0 nS on the TFT and oscilloscope respectively. As the Calcium conductance increases, the plateau of the action potential clearly elongates. On the TFT, the plateau increases from approximately 50 ms to 105 ms to 230 ms. The oscilloscope shows the exact same behavior.

This behavior occurs due to the physiology of cardiac action potentials. During the plateau, L-type Calcium ion channels are open, allowing a small and constant inward current of Calcium ions. This initial influx of Calcium ions into myocytes through the L-type Calcium channels is insufficient to trigger the contraction of myofibrils. However, the signal is amplified by the Calcium-induced Calcium release leading to a larger release of Calcium from the sarcoplasmic reticulum. Ultimately, this large release of intracellular Calcium acts on tropomyosin complexes to induce myocyte contraction and the contraction of cardiac muscle cells (Ikonnikov et al, 1993)

During the plateau phase, Potassium ions are also leaking out down the concentration gradient through delayed rectifier Potassium channels. The Calcium and Potassium currents are electrically balanced and the total membrane potential is maintained at a plateau. Over time, Calcium channels are gradually inactivated leading to eventual repolarization after the plateau (Grandi et al., 2009). However, with increased Calcium conductance, these Calcium channels take longer to become inactivated. Thus, with increased Calcium conductance, we also see a longer plateau. Additionally, with an extremely large Calcium conductance like 1.0 nS, the plateau elongates so far as to disrupt the next action potential (i.e. the next heartbeat of the cardiac cell).

This behavior mimics conditions like Vitamin D deficiency that result in hypocalcemia (low Calcium levels in the blood serum).

Example solution for Potassium Conductance

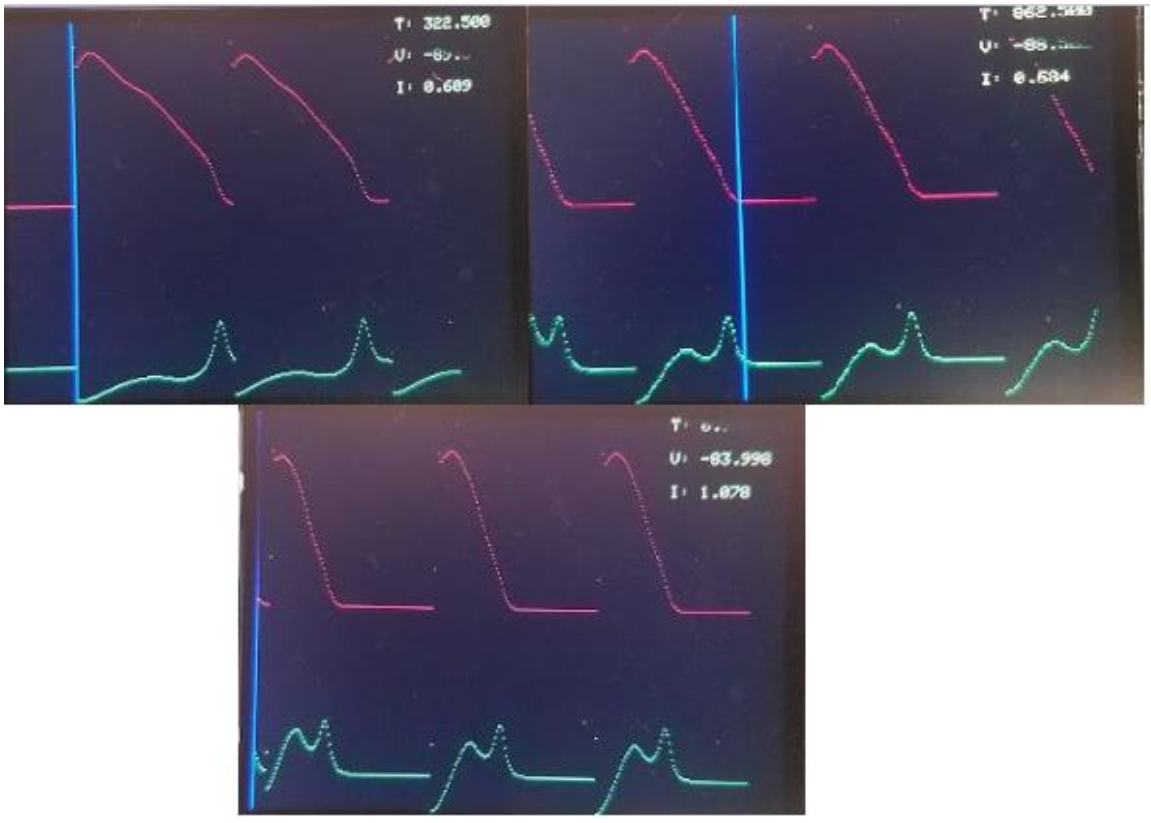

Figure 5: TFT screenshots with increasing Potassium conductance
Top left: 0.015. Top right: 0.06. Bottom: 0.045

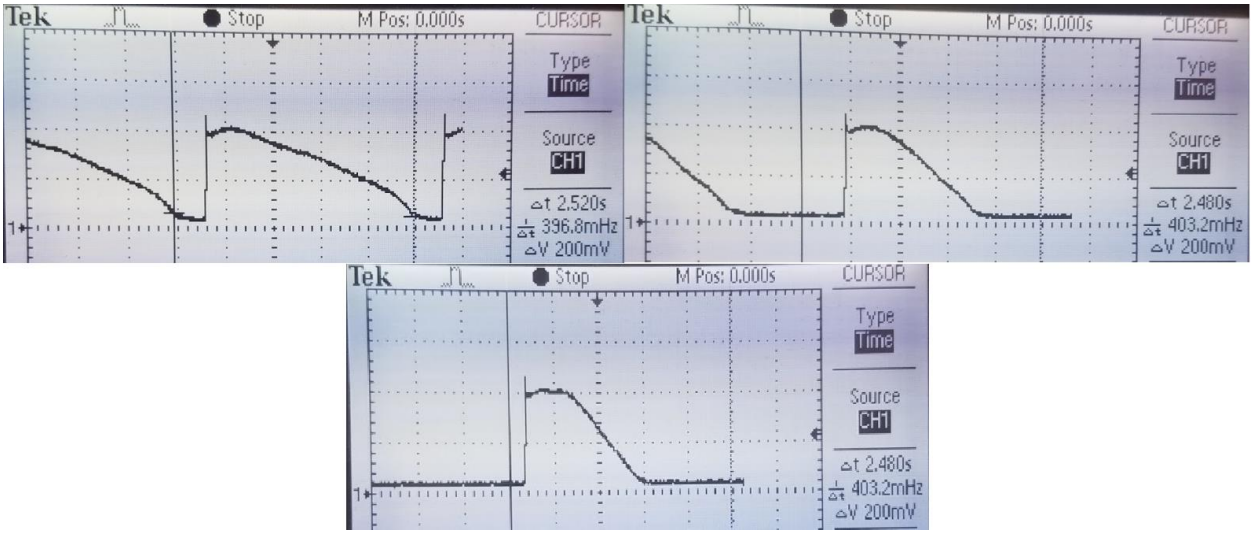

Figure 6: Oscilloscope screenshots with increasing Potassium conductance
Top left: 0.015. Top right: 0.06. Bottom: 0.045

Figures 5 and 6 above show the membrane voltage and Potassium current waveforms as the Potassium conductance is increased from 0.015 to 0.06 to 0.045 nS. As the Potassium conductance increases, the amount time on the TFT from depolarization spike back to the resting potential decreases from 675 to 435 ms to 300 milliseconds.

This behavior occurs due to the final phase of the cardiac action potential. As mentioned earlier, the plateau phase is caused by the balance between Potassium and Calcium currents. As Calcium channels inactivate, the continuing outward flow of Potassium ions brings the membrane potential back towards the resting potential of -90 mV. However, with increased Potassium channel conductance, the ions flow outward faster, and allow the membrane potential to reach the resting potential even faster.

This behavior is characteristically similar to chronic kidney disease or a crush injury. Both of these problems cause hyperkalemia, an increase in the amount of Potassium ions. Left untreated, this fast repolarization spike can result in cardiac death (McCullough et al, 2014)

Arrhythmias and Heart Attacks
Can you make an arrhythmia or heart attack occur? Change conductance and concentrations until you detect significant abnormalities. How does this relate to physiology?

Questions:
In your lab report, report and explain your results from the above experiments. Also, answer the following questions.

1. How do cardiac action potentials differ from muscle cells and other neurons previously recorded? In addition to behaviorally, what is different about the shape of these action potentials?

2. What is the role of each ion in the Cardiac action potential? What happens to the shape of the action potential when the concentration of each ion is increased or decreased?
3. Explain the nature of arrhythmias and heart attacks in terms of the cardiac cell action potential. How can abnormalities in each ion lead to disruptions in heart behavior?
4. What poisons exist that affect the action potential of cardiac cells? Describe their action.
5. Cardiac cells directly affect mammalian EKG. Explain how each ion affects individual parts of an EKG recording.

Appendix 5: Distribution of Work

All work was evenly distributed between the two team members. Samir primarily worked on the model implementations in Matlab and C. Adarsh primarily worked on the TFT user interface code, PCB design, and product/code integration.