3K04 Deliverable 2: Documentation

Group 33

Last Updated: 2025-11-27

Contents

List of Figures

List of Tables

1 Group Members

Table 1: Table of Group Members

Name	MacID	Student Number
Ryan Su	sur21	400507973
Cameron Lin	lin422	400535393
Braden McEachern	mceacb1	400527617
Damian Szydlowski	szydlowd	400512629
Menakan Thamilchelvan	thamilcm	400510755

2 Abbreviations

2.1 General Abbreviations

BPM - Beats Per Minute

CCS - Cardiac Conduction System

DCM - Device Controller-Monitor

GPIO - General Purpose Input Output

GUI - Graphical User Interface

PWM - Pulse Width Modulation

2.2 Bradycardia Operating Abbreviations

Table 2: Bradycardia Operating Abbreviations

Category	**Chambers Paced**	**Chambers Sensed**	**Response to Sensing**	**Rate Modulation**
Letters	O-None	O-None	O-None	R-Rate Modulation
	A-Atrium	A-Atrium	T-Triggered	
	V-Ventricle	V-Ventricle	I-Inhibited	
	D-Dual	D-Dual	D-Tracked	

3 Introduction

It is hard to understate the importance of the human heart. The heart is the core part of the cardiovascular system; supplying nutrients and oxygen to all the cells and removing carbon dioxide, especially to vital organs such as the brain, it is imperative for it to be working flawlessly and harmoniously at all times. Unfortunately, however, cardiovascular diseases are a leading cause of death globally, many of which are caused from complications with abnormal heart rhythms. A pacemaker is an implantable device capable of sending timed electrical impulses causing contractions at appropriate intervals. Understanding the operation and design of this life saving device will aid in developing more efficient and reliable cardiac assistive technology.

The purpose of this project is to design and implement a system that operates a cardiac pacemaker under specified modes. This project will be accomplished through an understanding of embedded systems and through engineering principles of software development.

The scope of this deliverable is to design and implement the embedded pacemaker software, driver software and user interface for the DCM while updating and maintaining documentation.

4 Requirements

4.1 System Requirements

Table 3: System Requirements

ID	Requirement	Rationale
SR1	System Modes	The system shall implement AOO, VOO, VVI, and AAI pacing modes as the basic bradycardia modes required by the spec.
SR2	Hardware Hiding	The system shall use a hardware abstraction layer that maps logical control signals to GPIO pins, which improves maintainability and supports future hardware changes.
SR3	Simulink and DCM Separation	The pacing logic shall be implemented in Simulink and the DCM as a separate GUI so that embedded logic and clinical interface can evolve independently.

4.2 Programming Requirements

Table 4: Programming Requirements

ID	Requirement	Description / Rationale
PR1	Programmable Pulse Amplitude	The pacemaker shall generate atrial and ventricular pacing signals with amplitudes configurable by the user. Adjustable amplitudes allow tuning of pacing strength.
PR2	Programmable Pulse Width	The pacemaker shall generate atrial and ventricular signals with pulse widths configurable by the user so timing of stimulation can be adjusted.
PR3	Programmable Rate Timing	The pacemaker shall have programmable lower rate limit (LRL) and upper rate limit (URL) that control the minimum and maximum pacing rates, preventing bradycardia and tachycardia.
PR4	Programmable Refractory Periods	Atrial and ventricular pulse modes shall implement refractory periods during which sensed events are ignored to avoid double sensing and ringing.
PR5	Programmable Sensitivity	A programmable sensitivity threshold for event detection shall be adjustable from the DCM so that sensing can be adapted to patient signals and noise.
PR6	Pacing Responses	Each pacing mode shall follow the response implied by its letters: O for asynchronous pacing that ignores sensing, I for inhibited pacing, and T for triggered pacing from sensed pulses.

4.3 Hardware Requirements

Table 5: Hardware Requirements

ID	Requirement	Description / Rationale
HR1	Hardware Hiding Layer	A hardware abstraction layer shall map digital logic signals to analog front end hardware pins so that the pacing model does not reference physical pins directly, improving readability and portability.
HR2	Front End Enable Control	A dedicated signal shall enable or disable the front end sensing circuitry so ADCs and amplifiers can be powered only when needed, reducing noise and power consumption.

4.4 DCM Requirements

Table 6: DCM Requirements

ID	Requirement	Description / Rationale
DCM1	User Authentication	The DCM GUI shall provide user registration and login functionality supporting up to ten stored users to control access to the programming interface.
DCM2	Parameter Display and Editing	The DCM shall display and allow editing of pacemaker parameters such as LRL, URL, atrial and ventricular amplitude, pulse width, and sensitivity so that clinicians can program the device.
DCM3	Status Indicators	The DCM shall provide visible indicators of device connection status and communication loss so the user remains aware of system state and faults.

4.5 Serial Requirements

Table 7: Serial & Data Requirements

ID	Requirement	Description / Rationale
COM1	The DCM and Pacemaker shall communicate via a defined Serial Protocol (UART).	Establishes communication between the pacemaker and DCM, replacing the mock communication.
COM2	The DCM shall verify the integrity of sent parameters by reading them back (Echo Verification) from the device.	Ensures safety by confirming data was stored correctly.
DAT1	The system shall support programmable Pulse Widths from 1ms to 30ms (1ms increment).	Updated requirements from the previous deliverable.
DAT2	The system shall support programmable Pulse Amplitudes from 0.1V to 5.0V (0.1V increment).	Updated requirements from the previous deliverable.

4.6 Egram Requirements

Table 8: Serial & Data Requirements

ID	Requirement	Description / Rationale
EKG1	The DCM shall receive real-time electrogram (egram) data points from the pacemaker via Serial.	Allows the doctor to visualize heart activity.
EKG2	The DCM shall plot two simultaneous graphs (Atrial and Ventricular) in real-time.	Required for diagnostic visibility.

4.7 Rate Adaptivity Requirements

Table 9: Rate Adaptivity Requirements

ID	Requirement	Description / Rationale
EKG1	The DCM shall receive real-time electrogram (egram) data points from the pacemaker via Serial.	Allows the doctor to visualize heart activity.
EKG2	The DCM shall plot two simultaneous graphs (Atrial and Ventricular) in real-time.	Required for diagnostic visibility.

4.8 Bonus Requirements

Table 10: Bonus Requirements

ID	Requirement	Description / Rationale
BON1	The system shall implement DDDR mode with a programmable AV Delay.	Mimics natural dual-chamber heart rhythm.
BON2	The system shall inhibit Ventricular pacing (but not Atrial) when the board pushbutton is held.	Simulates a specific failure or diagnostic state.

5 Design

5.1 Simulink Design - Deliverable 1

5.1.1 Overall Design

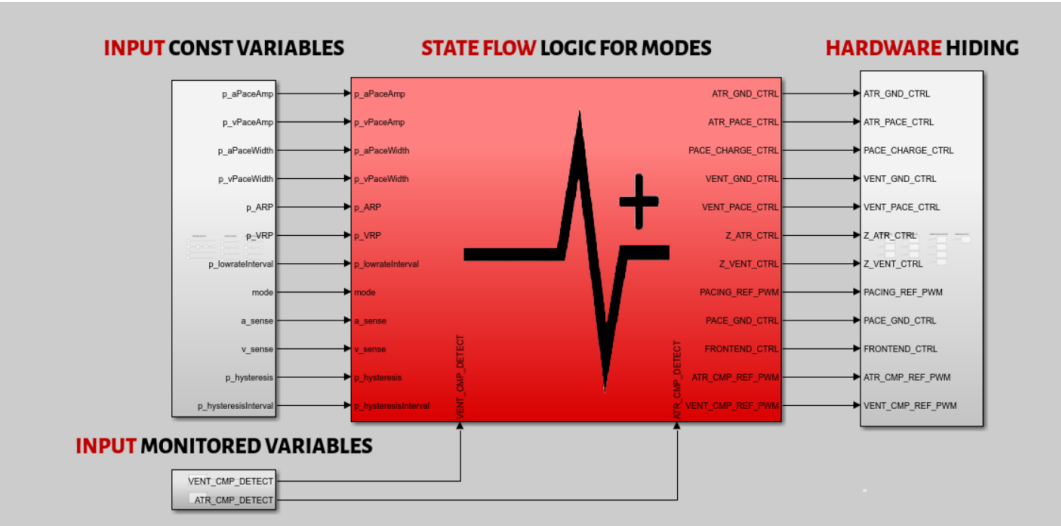

Figure 1: Overall Simulink Mapping

The Pacemaker architecure can be split up into 4 main modules, input constant variables, input monitor variables, stateflow logic for modes, and hardware hiding. Figure 1 below shows the overarcing workflow of the system:

5.1.2 Input Constant Variables

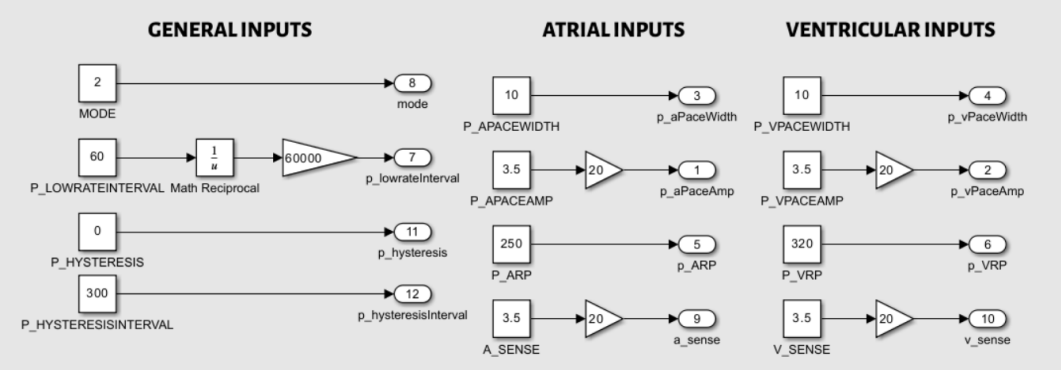

Figure 2: Constant Input Variables

In the above image, Figure 2, we find changeable variables relating to pacemaker operation. For general inputs, the changeable variables are:

- **Mode** - Refers to bradycardia operating modes, e.g AOO, VOO, AAI and VVI.
- **Low Rate Interval** - The number of generated pace pulses per minute, converted from a millisecond time period.
- **Hysteresis Pace** - When enabled, 1, a longer period is waited before pacing after sensing an event to prevent unwanted pacing pulses from ringing from an event.
- **Hysteresis Interval** - Specifies the time interval waited in the hysteresis mode in milliseconds.

The modfiable atrial variables are:

- **Pace Pulse Width** - Changes the width, length of time, of the pace pulse.
- **Pace Pulse Amplitude** - Changes the amplitude, voltage, of the pace pulse.
- **ARP (Atrial Refactory Period)** - The programmed time interval following an atrial event during which time atrial events shall not inhibit nor trigger pacing
- **Sense (Sensitivity)** - Determines the minimum value an atrial signal must be to be considered by the pacemaker.

The modifiable ventricular variables are:

- **Pace Pulse Width** - Changes the width, length of time, of the pace pulse.
- **Pace Pulse Amplitude** - Changes the amplitude, voltage, of the pace pulse.

- **VRP (Ventricle Refactory Period)** - The programmed time interval following an ventricle event during which time atrial events shall not inhibit nor trigger pacing.
- **Sense (Sensitivity)** - Determines the minimum value a ventricle signal must be to be considered by the pacemaker.

5.1.3 Monitored Input Variables

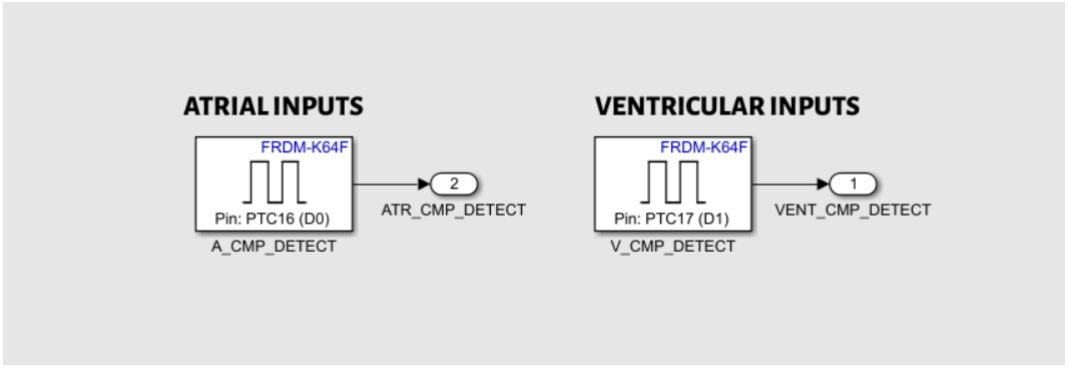

Figure 3: Monitored Input Variables

The monitored input variables can be seen in the above Figure 3. These are the atrial and ventricle detection variables. The pulses are sensed with through GPIO pins connecting to a board simulating heart conditions.

5.1.4 Stateflow Modules

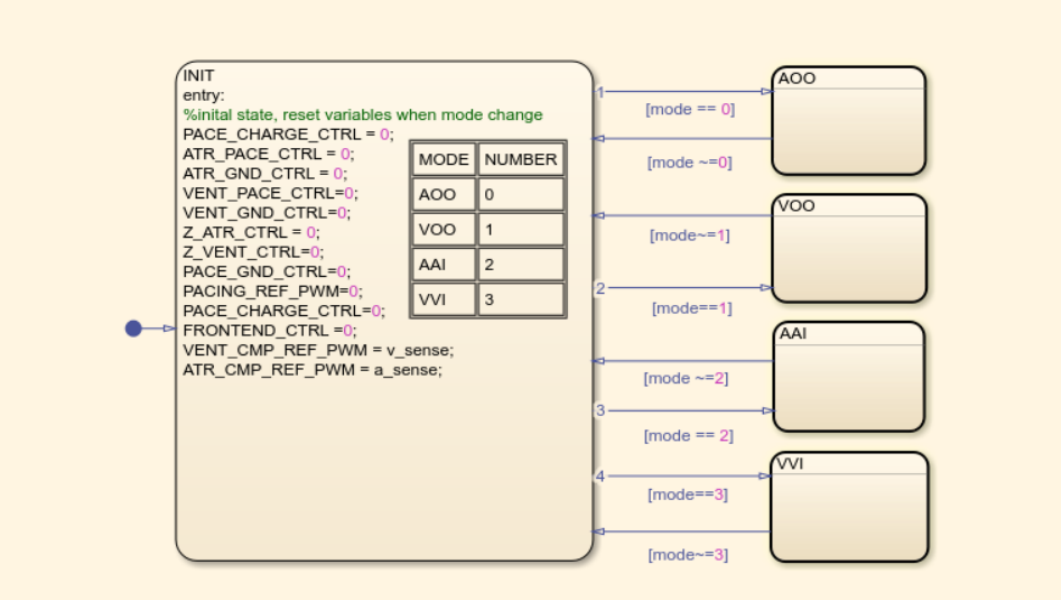

Figure 4: Stateflow Modules

The stateflow diagram in Simulink is shown above. It shows the transition between each mode given the mode input shown in Figure 2. When switching between modes, a core state is returned to that resets variables to their nominal values.

5.1.5 AOO Stateflow Model

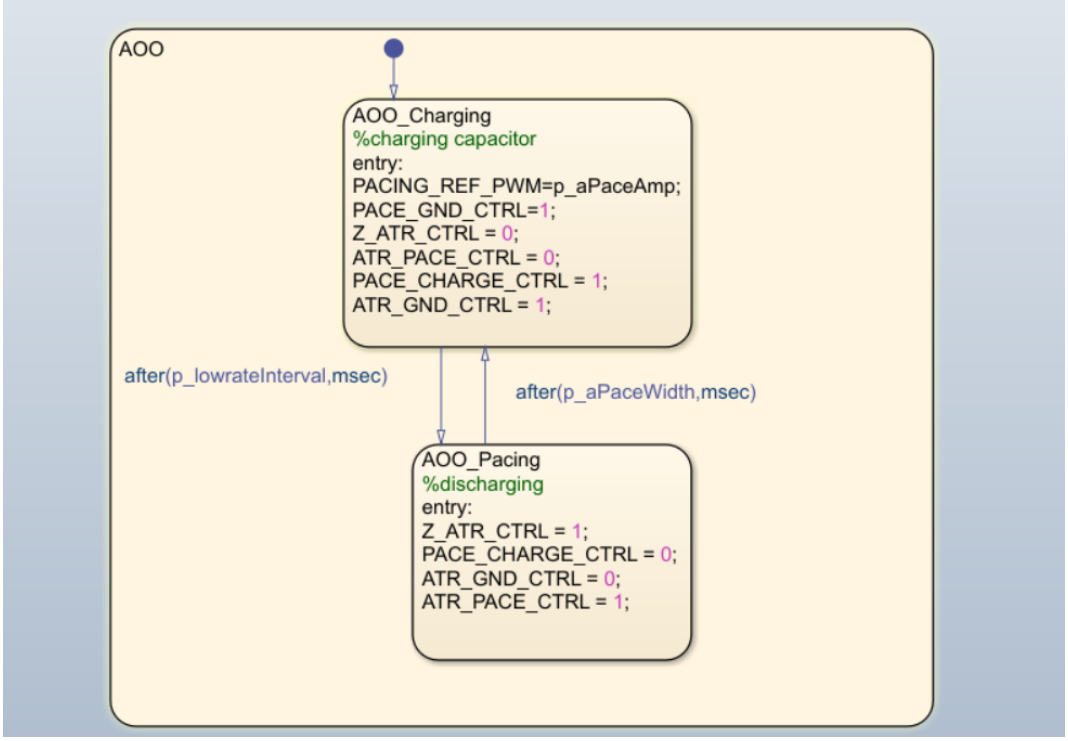

Figure 5: AOO Stateflow Model

The above stateflow, Figure 5, shows the stateflow for the AOO mode. It sets values controlling discharge and charging of the capacitor to specified nominal values.

5.1.6 VOO Stateflow Model

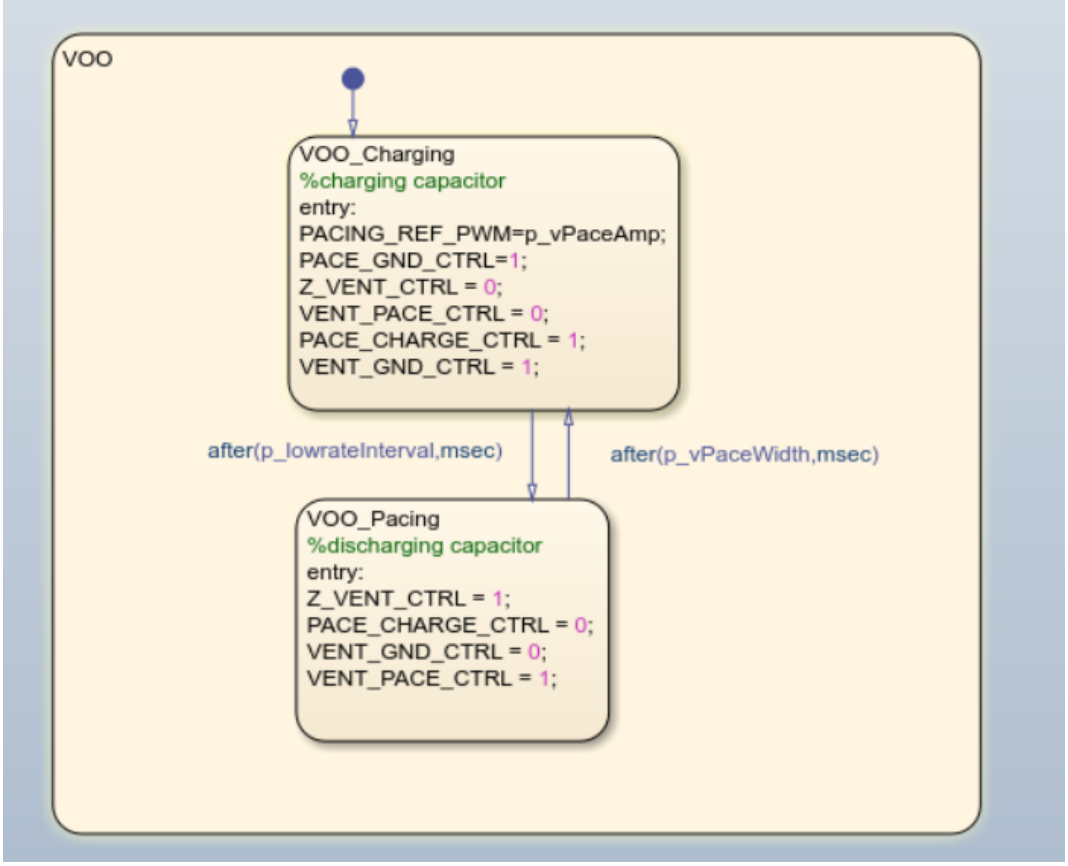

Figure 6: VOO Stateflow Model

Similar to the AOO stateflow, the above figure, Figure 6, shows the states of charging and discharging of the capacitor using specified nominal values.

5.1.7 VII Stateflow Model

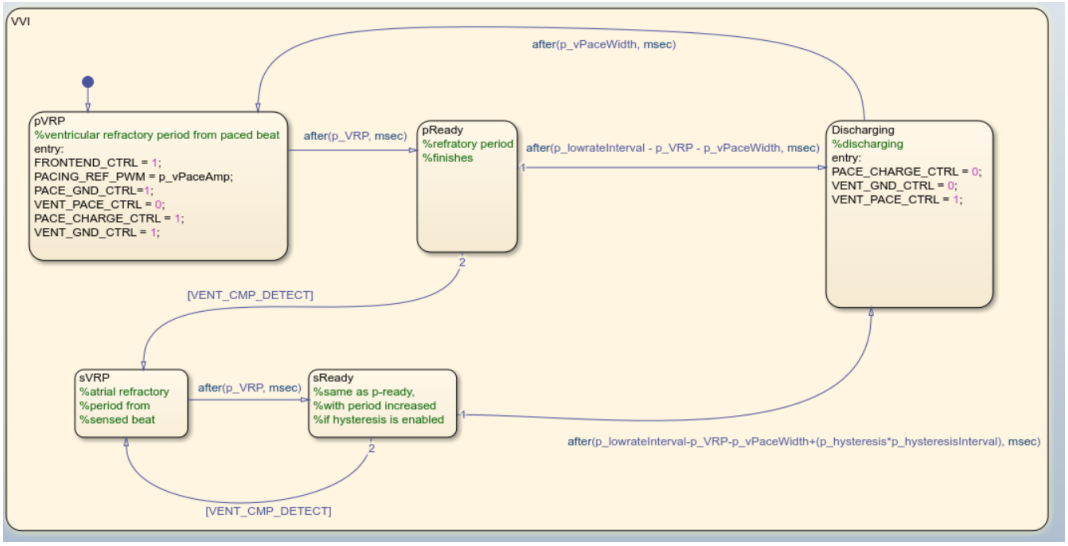

Figure 7: VVI Stateflow Model

The above stateflow, Figure 7 shows the FSM for the VVI model. The initial state, pVRP, is the state that occurs right after the pacemaker delivers a ventricle pacing pulse. A refactory period occurs during this time, where the pacemaker ignores sensor inputs to prevent sensing of its own produced pulse or electrical ringing. After a certain amount of time, the pReady state is transitioned to where sensor inputs are allowed. This allows for the pacemaker to detect natural heart rhythms and correct heart rhythms accordingly, or solely deliver pacing pulses. The last state before going to the initial state is the discharging state where the pacing pulse is produced.

5.1.8 AAI Stateflow Model

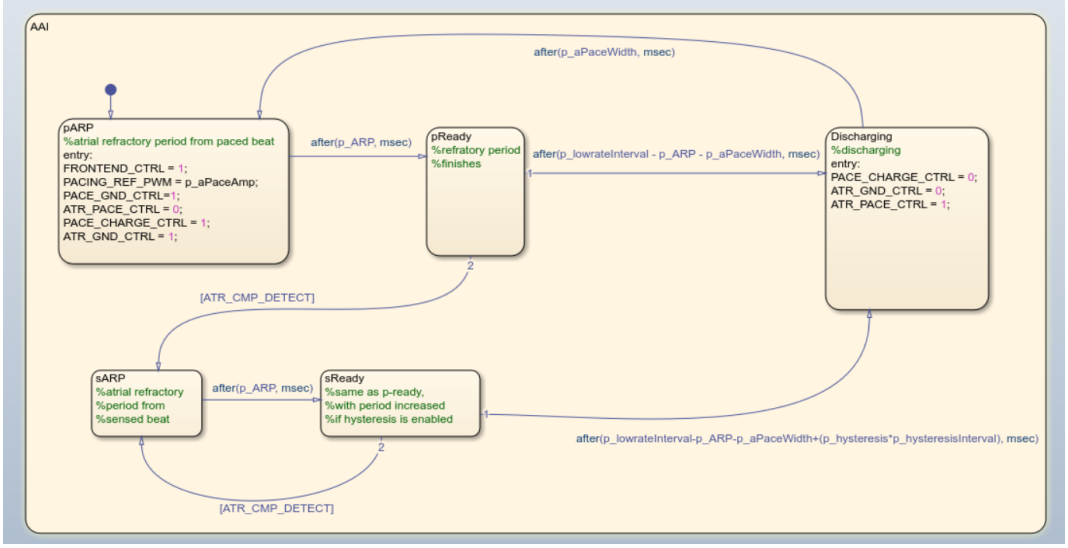

Figure 8: AAI Stateflow Model

The above stateflow, Figure 8 shows the process for the AAI mode. The initial state, pARP, is the state occurs right after the pacemaker delivers an atrial pacing pulse. During this time, the pacemaker ignores sensing events to prevent it from sensing its own delivered pulse. The next state is transitioned to after a set time period, the refactory period, if no natural heartbeat is detected after a certain time inverval, the discharging state is then transitioned to. However, if a natural heartbeat is detected, another refactory period occurs to prevent the pacemaker from sensing ringing. This state then transitions to the discharging state once an appropriate time has passed.

5.1.9 Hardware Hiding

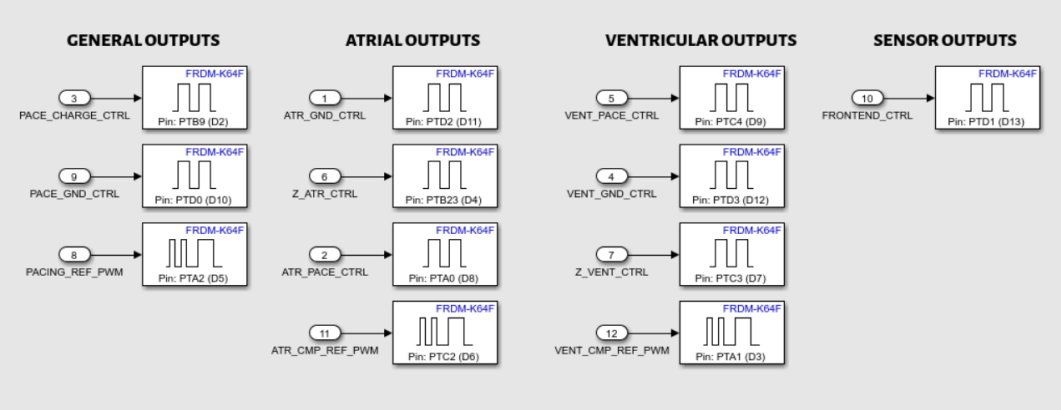

Figure 9: Hardware Hiding of Model

The above image, Figure 9 shows abstraction of the GPIO pin functions. It connects logical output signals to the pins on the FRDM-K64F. This allows for better readability, thus making the code easier to maintain, debug and safer.

5.2 Simulink Design - Deliverable 2

5.2.1 General Overview

The below image shows a high-level overview of the Simulink model with all subsystems. The newly added modules include the serial communication and rate adaptive logic modules for the newly added serial communication with DCM and pacing modes. The input monitored variables now also includes accelerometer values for adaptive rates based on accelerometer values.

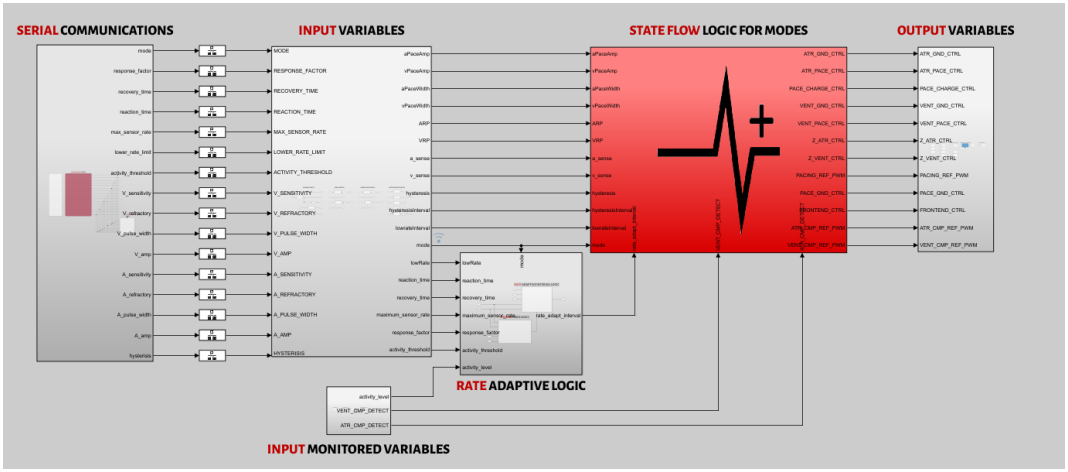

Figure 10: General Overview of Simulink Model - Deliverable 2

5.2.2 Serial Communication Module

The serial communication module can be seen in the figure below. The communication input block is stateflow logic that handles how the rxdata being transmitted from the serial recieve block is being processed and unpacked and the echo function. The communication output is how data is outputted from the pacemaker back to the DCM via the send_params function packing all data back and sending via the serial transmit block. For simplicity, we chose to use a stateflow logic simuilar to that of tutorial 3.

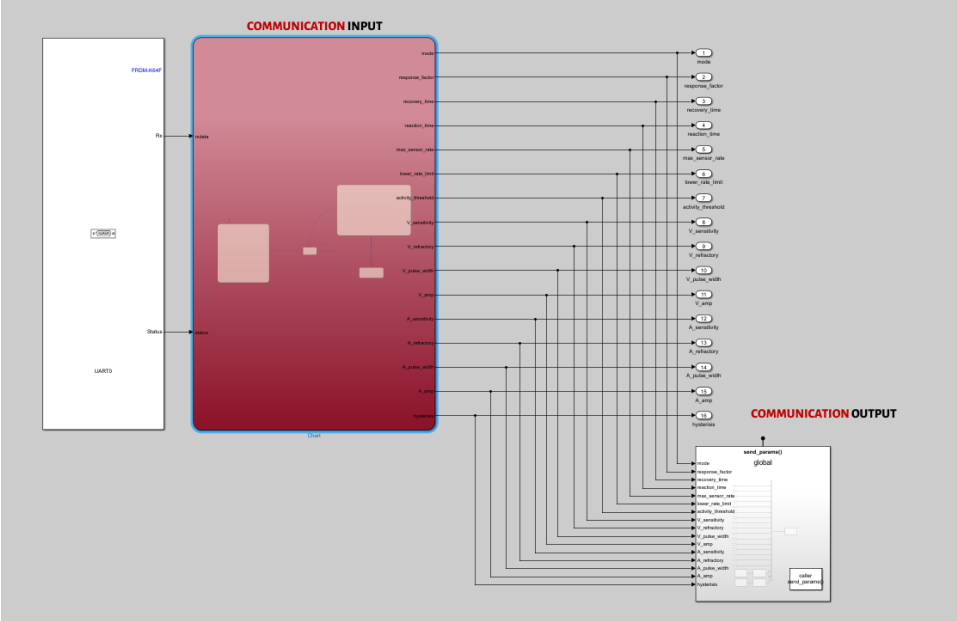

Figure 11: Serial Communication Module Simulink - Deliverable 2

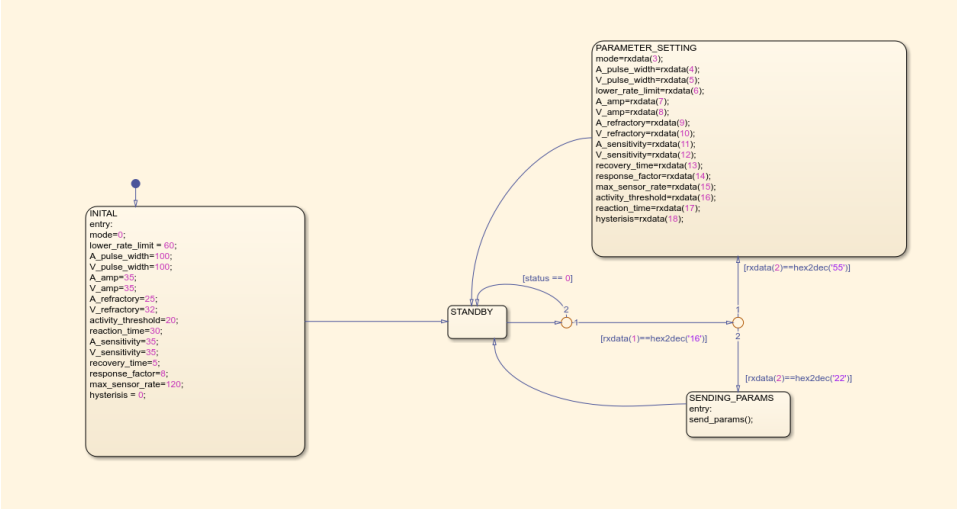

Figure 12: Communication Input Block - Deliverable 2

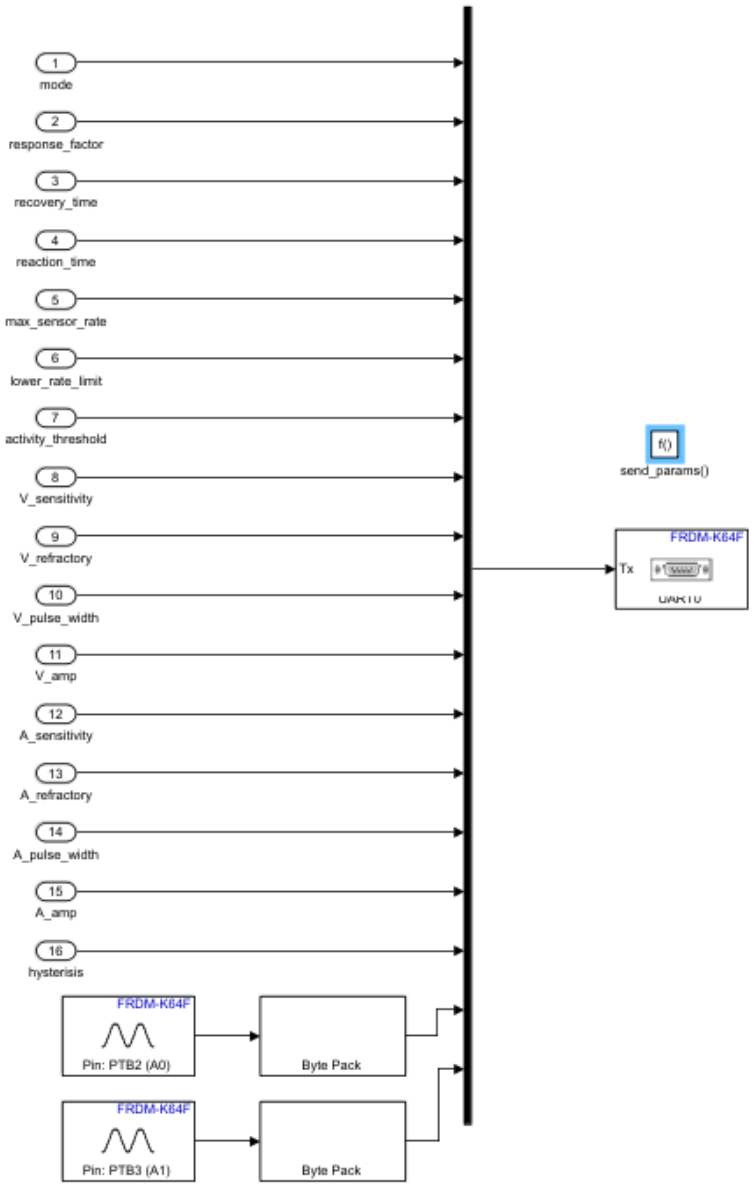

Figure 13: Send Parameters Block - Deliverable 2

5.2.3 Rate Response

5.2.3.1 All Modes

The requirements for the newly added modes were fulfilled and can be seen in the below images. Building off the stateflow logic in deliverable 1, the addition of modes was easily added to the default reset state block.

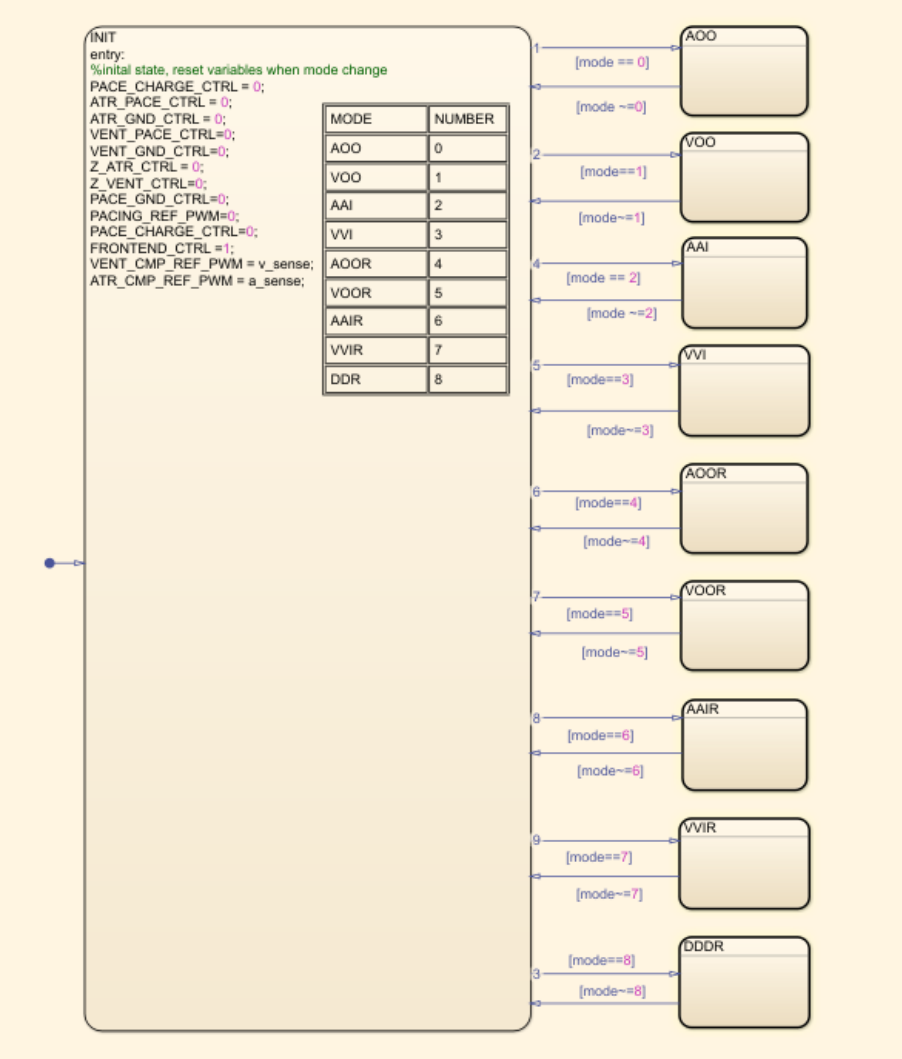

Figure 14: All Modes and Default State Stateflow Diagram - Deliverable 2

5.2.3.2 Mode Stateflow

Below are the stateflows for each rate adaptive modes:

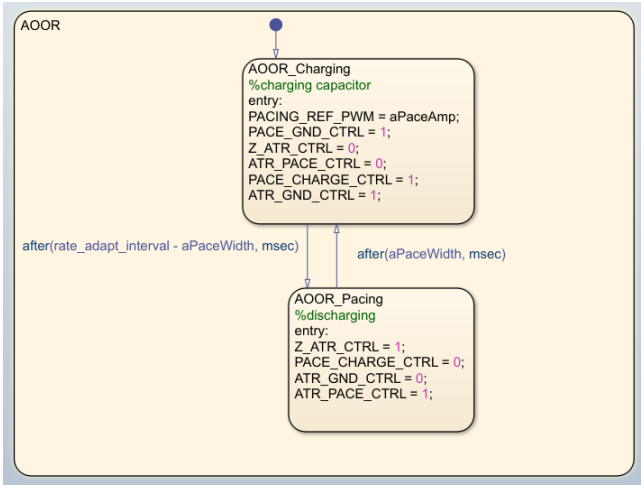

Figure 15: AOOR Stateflow - Deliverable 2

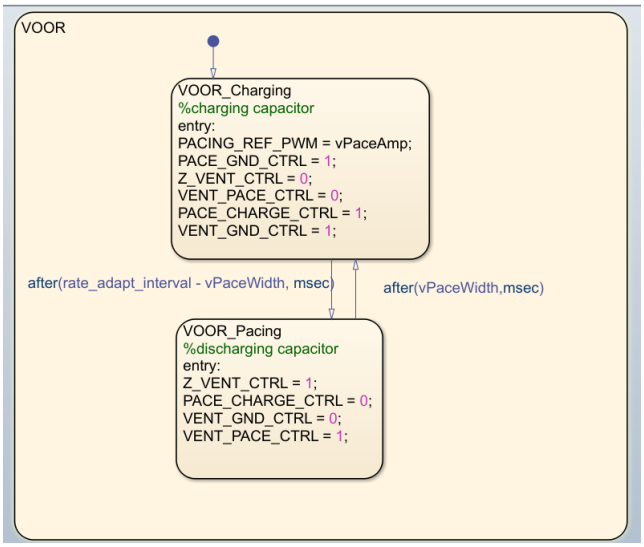

Figure 16: VOOR Stateflow - Deliverable 2

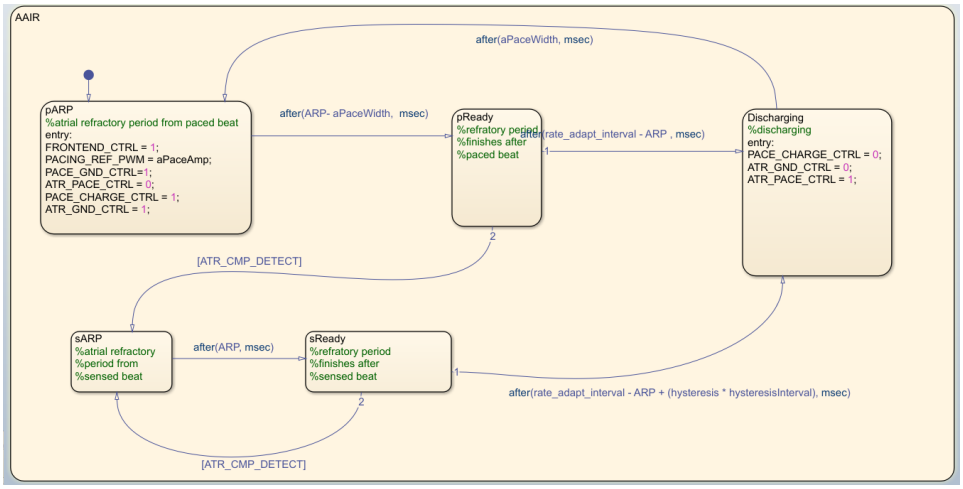

Figure 17: AAIR Stateflow - Deliverable 2

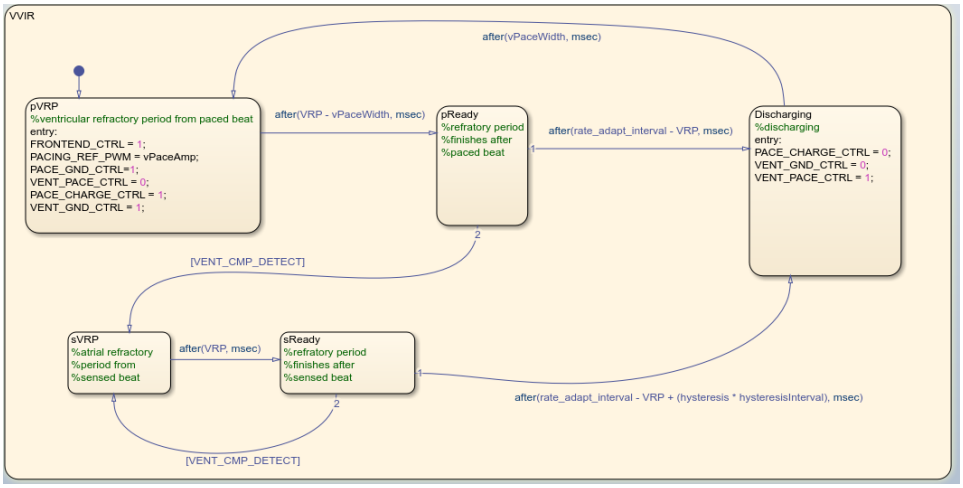

Figure 18: VVIR Stateflow - Deliverable 2

The logic of these states are similar to that of the regular modes e.g AOO, VVI except are governed by the rate_adapt_interval. This changes the effective BPM of pacing. The Simulink calculations for these modes can be seen below:

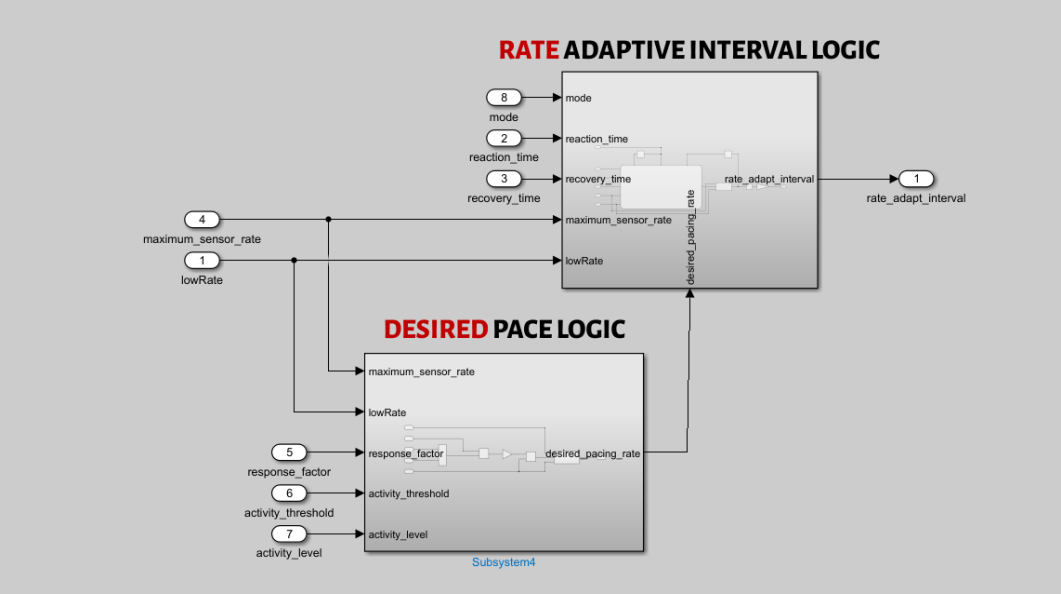

Figure 19: Rate Adaptive Logic - Deliverable 2

It takes in newly added parameters maximum sensor rate, response factor, activity threshold and activity level and outputs a rate adaptive interval value with constraints and parameters. The logic for desired pace is shown below:

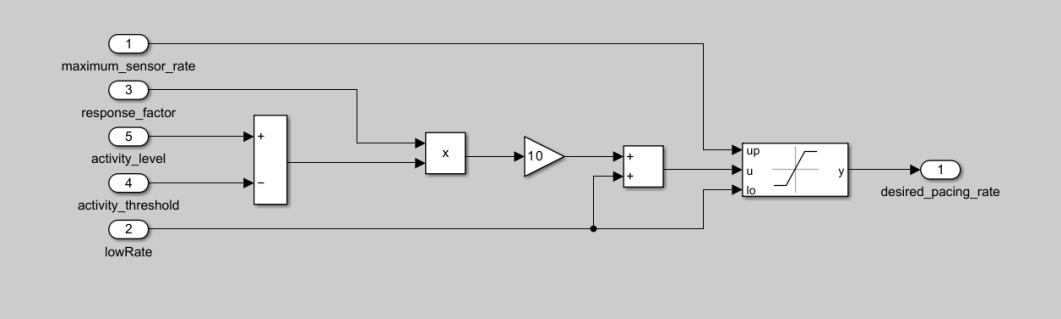

Figure 20: Desired Pace Logic - Deliverable 2

The saturation dynamic block prevents the desired pace rate from being too low, below the low rate, or being too high, above the max sensor rate. This calculates the desired pacing rate which is

fed into the rate adaptive interval block which can be seen below:

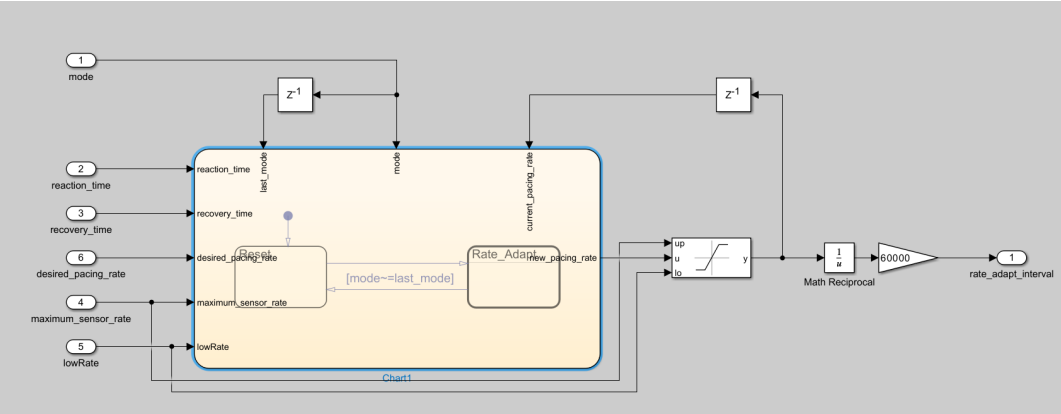

Figure 21: Rate Adaptive Interval Block - Deliverable 2

This block includes more logic within itself that resets the desired pace rates when swapping between different rate adaptive modes. The logic of which can be seen in the stateflow logic below:

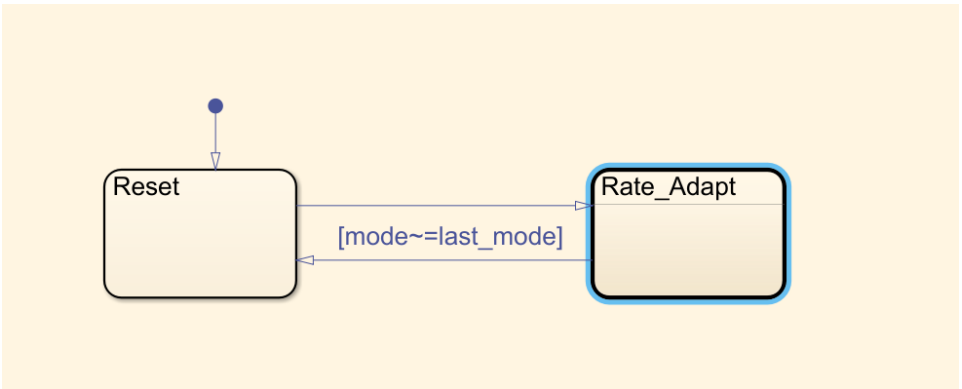

Figure 22: Stateflow for Mode Switching of Rate Adaptive Logic - Deliverable 2

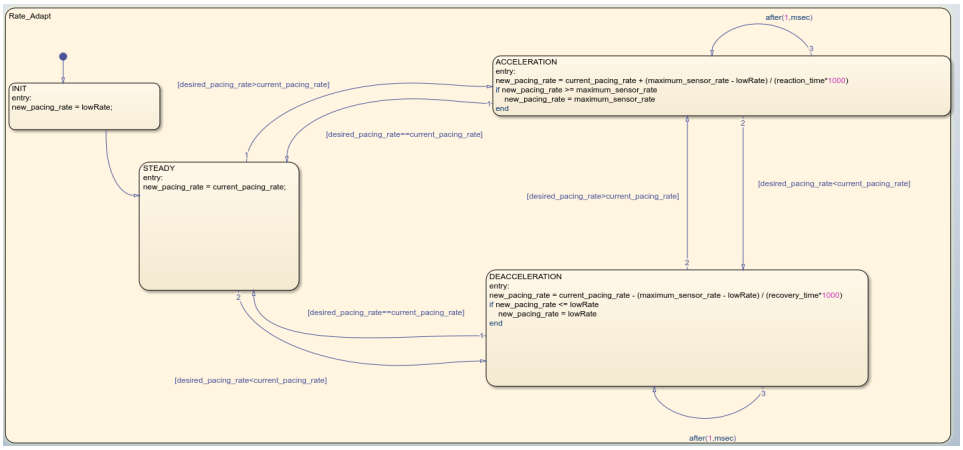

Figure 23: Rate Adaptive Mode Reset When Switching Logic - Deliverable 2

The way activity level for the logic above is measured is through the accelerometer. This is computed with the following block diagram:

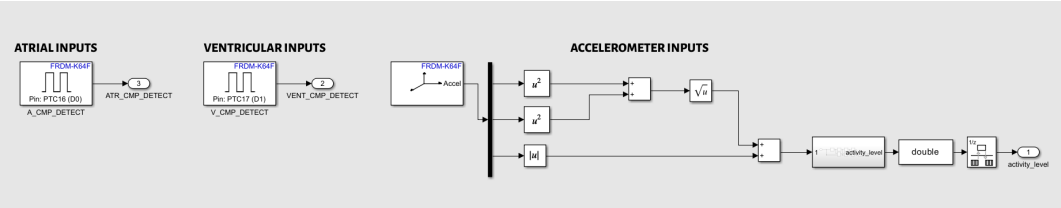

Figure 24: Accelerometer Reading Block Diagram - Deliverable 2

The activity level is calculated by finding the magnitude of x and y values first, then adding the absolute z value, normalizing these vales. The reason is because we want to treat moving diagonally on the xy-plane to be the same as moving straight in x or y, since the magnitude is the same. The z-value is added for the case the patient jumps or walks up stairs.

The sampling rate of the ventricular and atrial inputs is faster, 0.001s, to produce more accurate values in AAI and VVI modes. The accelerometer sampling rate is lower and is unable to sample faster than the value shown, 0.01s.

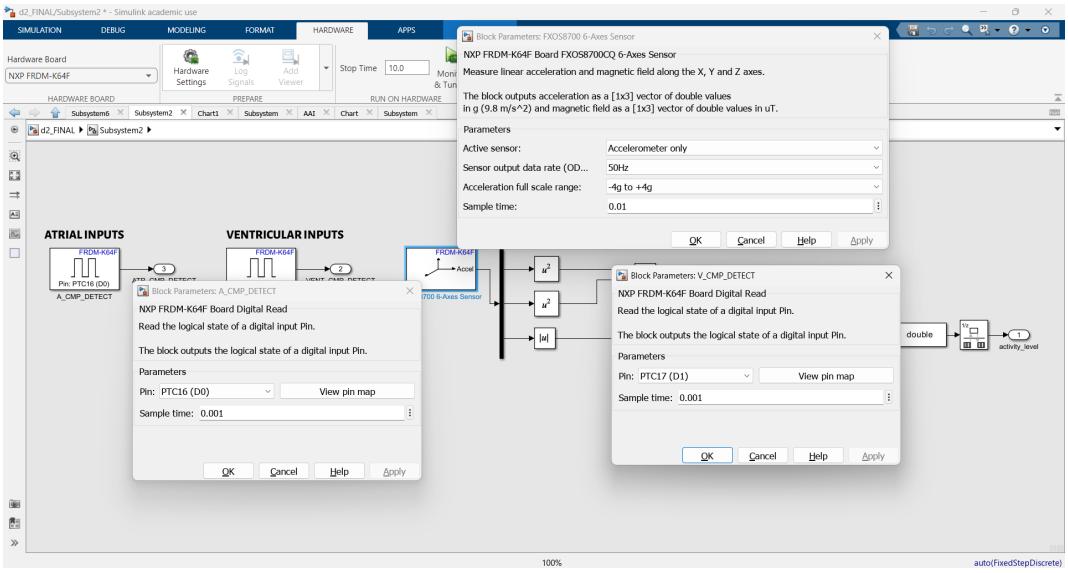

Figure 25: Accelerometer and Inputs Sampling Rates - Deliverable 2

Furthermore, this calculation uses a moving custom moving average block and the calculation of such can be seen below. This is to smooth out the signal from small deviations or noise in the measurements of acceleration.

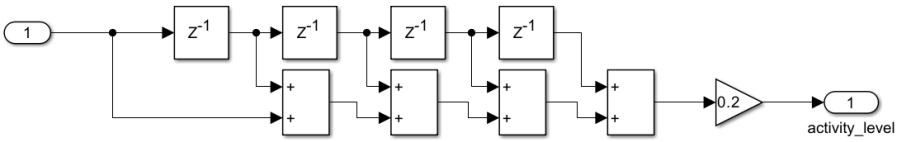

Figure 26: Custom Moving Average - Deliverable 2

5.2.4 Outputs

The ouputs of the Simulink is handled in the outputs subsystem. The subsystem is shown below:

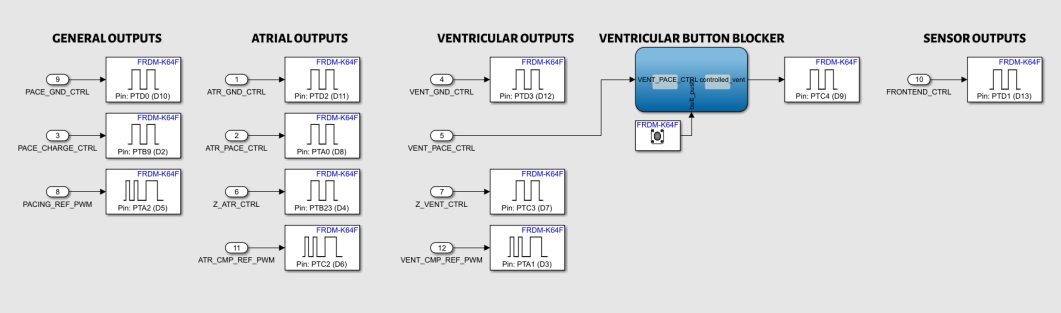

Figure 27: Outputs Block - Deliverable 2

5.2.4.1 Ventricular Inhibiting

This block includes a push button input that inhibits the ventricular pulsing. If held down, it prevents the ventricular part of the pacemaker from pacing. This is a safety feature that allows for quick emergency shutoff incase of malfunction. The logic for this block is shown below:

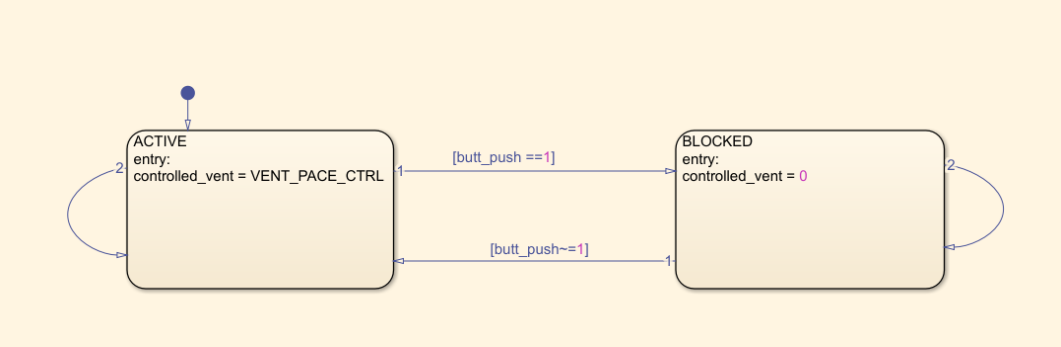

Figure 28: Ventricular Inhibiting Logic - Deliverable 2

5.3 DCM Design

The design of the DCM has 3 seperate sections; the **model, view and controller.** This is to allow modularity and seperation of different functionalities of the DCM from one another. For example, adding functionality can be as simple as making and includiung another class in another file.

5.3.1 Model

The model section handles the applications data and user logic. It does not interact directly with the interface but only manages data and enforces user authentication. Below is an explanation of the individual Python files and their functionality:

User Model

Purpose: To manage all user account information.

Functions:

- Loads and saves login information to or from the users.json file.
- Registers new users, checking for duplicates and the max user limit.
- Authenticates logins by verifying user passwords
- Provides a method to get the amount of registered users which is displayed by the DCM.

Pacing Model

Purpose: Manages pacing parameters for all users.

Functions:

- Loads and saves settings to or from pacing_settings.json
- The data is structured as a dictionary with settings as the key and value as the value.
- Saves a dictionary of parameters for a specific user and pacemaker mode
- Retrieves the saved parameters for a user and mode

Egram Model

Purpose: Manages the capture of electrogram data

Functions:

- Adds a single data point to an internal listed
- Returns the list of captured data
- Clears the captured data to start a new session

5.3.2 View

The view section is what the user sees and interacts with. It is built with the customtkinter library for a more aesthetically pleasing interface. They do not have any control logic, only to display data and send user actions such as button clicks to the controller. Below are an explanation of each view:

Login View

Purpose: Provides a screen for users to login

Functions:

- Welcome class displays the login form and a register button.
- Register class displays the new user registration form and a back button.
- When a button is clicked, it collects text from the entry fields and calls a method on the controller that handles the login logic.
- The welcome view calls a function to display the current registered user count.

Main View

Purpose: Provides the main application screen after logging in

Functions:

- MainFrame is the main menu. It shows buttons for each pacing mode AOO, VOO, AAI and VVI as well as mock connection controls. It also displays the current connection status.
- DataEntry is the form for editing parameters. It displays all parameters and handles validation of data such as the input being in the correct range and is a numeric type before sending the data to the controller.

5.3.3 Controller

Controller

Purpose: The main application class that uses instances of the models and view frames.

Functions:

- It initializes itself and creates all view frames, UserModel and PacingModel. It then stores these in a dictionary.
- The navigation is handled by a show_frame function which requests a frame from the dictionary and uses tkinters library function, tkraise(), to bring it into view.
- Event handling is seperated into handling registration and saving settings. For For registration it takes the username and password from the view, calls the UserModel to verify them, then shows a message to confirm or deny entry. The saving of settings takes mode and data from the DataEntry view and passes them to the PacingModel to be saved.

Main

Purpose: Its purpose and sole function is to create an instance of the controller and run it in the main loop, starting the customtkinter application.

5.3.4 Views of DCM GUI

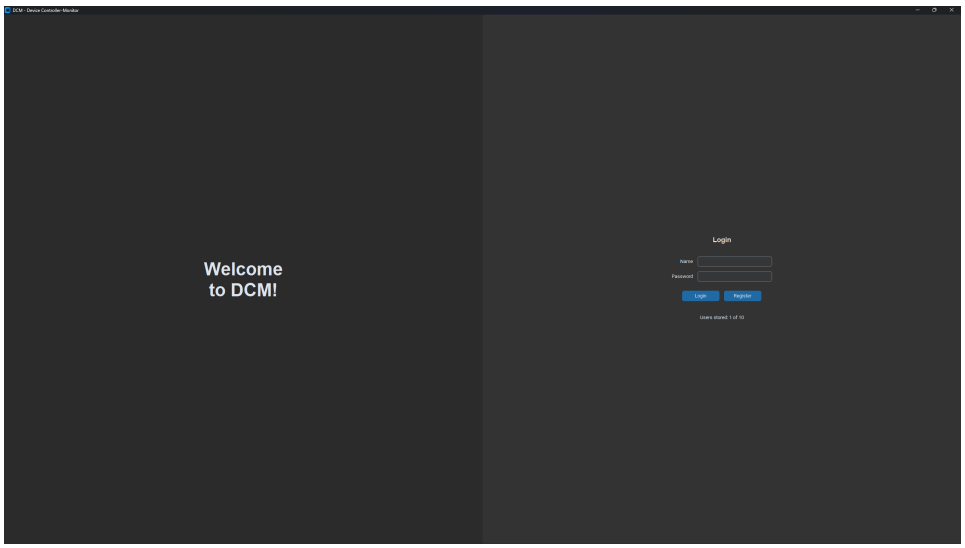

Figure 29: Login Screen of DCM

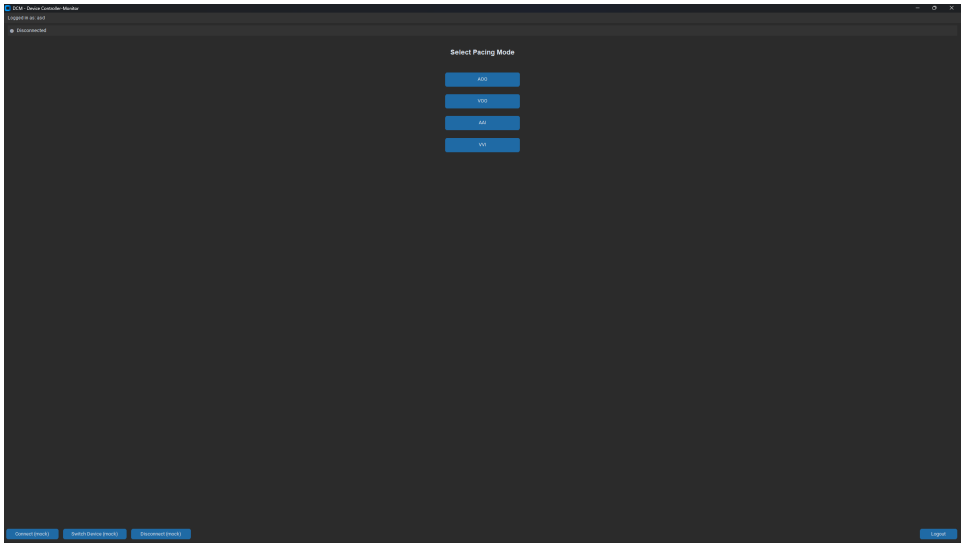

Figure 30: Main Menu of DCM

Figure 31: Data Entry View

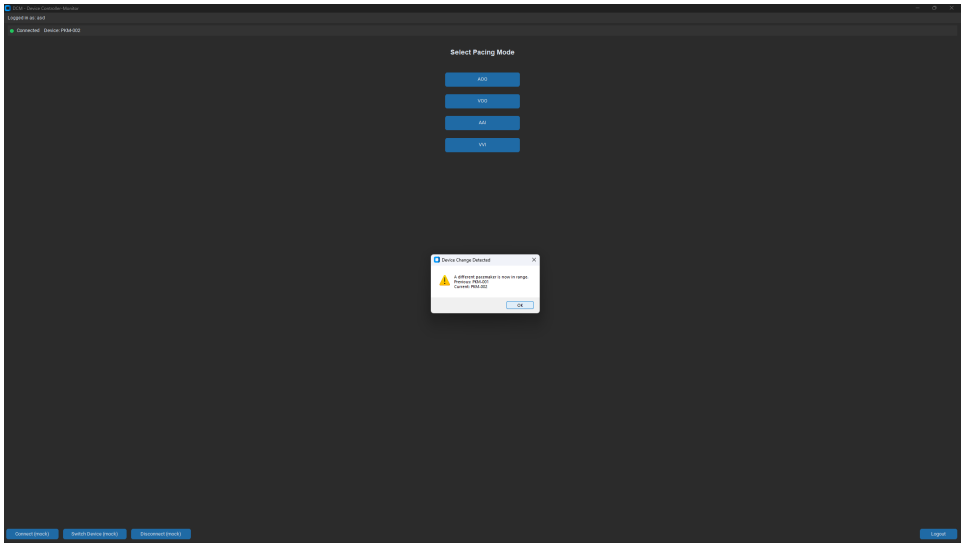

Figure 32: Switch in Device Detection from DCM

5.4 DCM & Serial Design - Deliverable 2

The DCM was changed up in deliverable 2 to allow for serial communication and more modes to be used. This is accompanied by a new model in the structure of the DCM, serial_comms.py.

Serial_Comms

Purpose: Provides serial communication functions to the DCM through serial send and recieve functions.

Functions:

- Serial port management - allows low-level hardware connection by scanning for COM ports through the get_ports function, establishes connection for the DCM by setting baud rate and other parameters, and safely disconnecting devices.
- Binary Data Structure - Uses struct library to pack and unpack data for more efficient data transfer.
- Send_params - The main function that allows for sending of parameter data to the pacemaker. It typecasts and scales data to allow for more efficient data transfer.
- Data Verification - The echo feature allows for data to be verified once sent to the pacemaker.
- Data Streaming - The EKG requires real time data streaming to be effective which is handled through this class.

5.4.1 Serial Packet Structure

With the addition of additional parameters a data conversion table is used to organize the multiple data types being sent. This can be seen in the table below. The chosen variables are due that are sent are due to the fact the increments within each range fit in a Uint8 format. This allows for the parameter data to be scaled in code to fit in a Uint8 format then scaled back and typecasted in the Simulink. This datatype choice reduces the amount of data being transfered while maintaining the same amount of fidelity. Some parameters such as activity threshold and mode are mapped to integers.

Table 11: Parameter Settings

Parameter	**Min**	**Max**	**Inc**	**Unit**	**Type**	**Bytes**
LRL	30	175	5	BPM	Uint8	1
URL	50	175	5	BPM	Uint8	1
Max Sensor Rate	50	175	5	BPM	Uint8	1
Fixed AV Delay	70	300	10	ms	Uint8	1
Dynamic AV Delay	OFF	ON	N/A	N/A	Bool	1
Sensed AV Delay Offset	0 (lowest -10)	-100	-10	ms	Uint8	1
Atrial Amplitude	0.00	5.00	1.25	Volts	Uint8	1
Ventricular Amplitude	0.00	5.00	1.25	Volts	Uint8	1
Atrial Pulse Width	0.05 0.1	- 1.9	- 0.1	ms	Uint8	1
Ventricular Pulse Width	0.05 0.1	- 1.9	- 0.1	ms	Uint8	1
Atrial Sensitivity	0.25 1	0.75 10	0.25 0.5	mV	Uint8	1
Ventricular Sensitivity	0.25 1	0.75 10	0.25 0.5	mV	Uint8	1
VRP	150	500	10	ms	Uint8	1
ARP	150	500	10	ms	Uint8	1
PVARP	150	500	10	ms	Uint8	1
PVARP Extension	0	400	50	ms	Uint8	1
Hysteresis	OFF	ON	N/A	N/A	Bool	1
Rate Smoothing	0	21	3	N/A	Uint8	1
ATR Duration	10 20 100	- 80 2000	- 20 100	cc	Uint8	1
ATR Fallback Mode	OFF	ON	N/A	N/A	Bool	1
ATR Fallback Time	1	5	1	min	Uint8	1
Activity Threshold	V-Low, Low, Med-Low, Med, Med-High, High, V-High			N/A	Uint8	1
Reaction Time	10	50	10	sec	Uint8	1
Response Factor	1	16_{33}	1	N/A	Uint8	1
Recovery Time	2	16	1	min	Uint8	1
Mode	OFF, DDD, VDD, DDI, DOO, AOO, AAI, VOO, VVI, AAT, VVT, DDDR, VDDR, DDIR, DOOR, AOOR, AAIR, VOOR, VVIR			N/A	Uint8	1

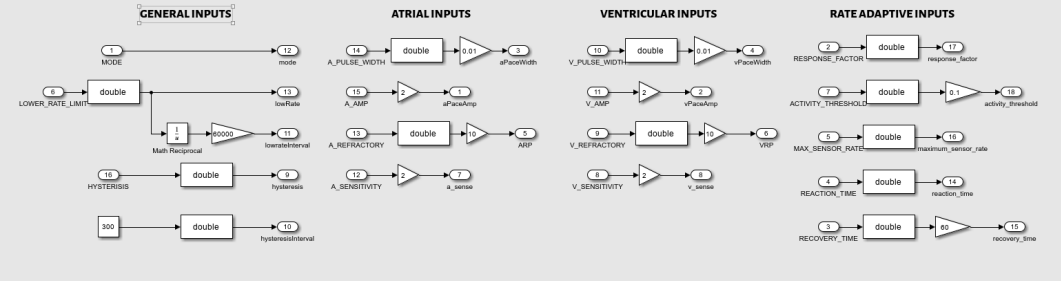

Figure 33: Conversion of Data In Simulink

Code 1: Conversion of Data in DCM

```
    def send_params(self, params: dict):
        if not self.ser or not self.ser.is_open: return False
        try:
            data = struct.pack(self.FMT_18_BYTES,
                               0x16, 0x55,
                               int(params.get("mode", 0)),
                               int(params.get("a_pw", 0) * 100),
                               int(params.get("v_pw", 0) * 100),
                               int(params.get("lrl", 60)),
                               int(params.get("a_amp", 0) * 10),
                               int(params.get("v_amp", 0) * 10),
                               int(params.get("a_ref", 0) / 10),
                               int(params.get("v_ref", 0) / 10),
                               int(params.get("a_sens", 0) * 10),
                               int(params.get("v_sens", 0) * 10),
                               int(params.get("recov", 0)),
                               int(params.get("resp_fact", 0)),
                               int(params.get("msr", 0)),
                               int(params.get("act_thresh", 0)),
                               int(params.get("react_time", 0)),
                               int(params.get("hyst", 0)))
            self.ser.write(data)
            return True
        except Exception: return False
```

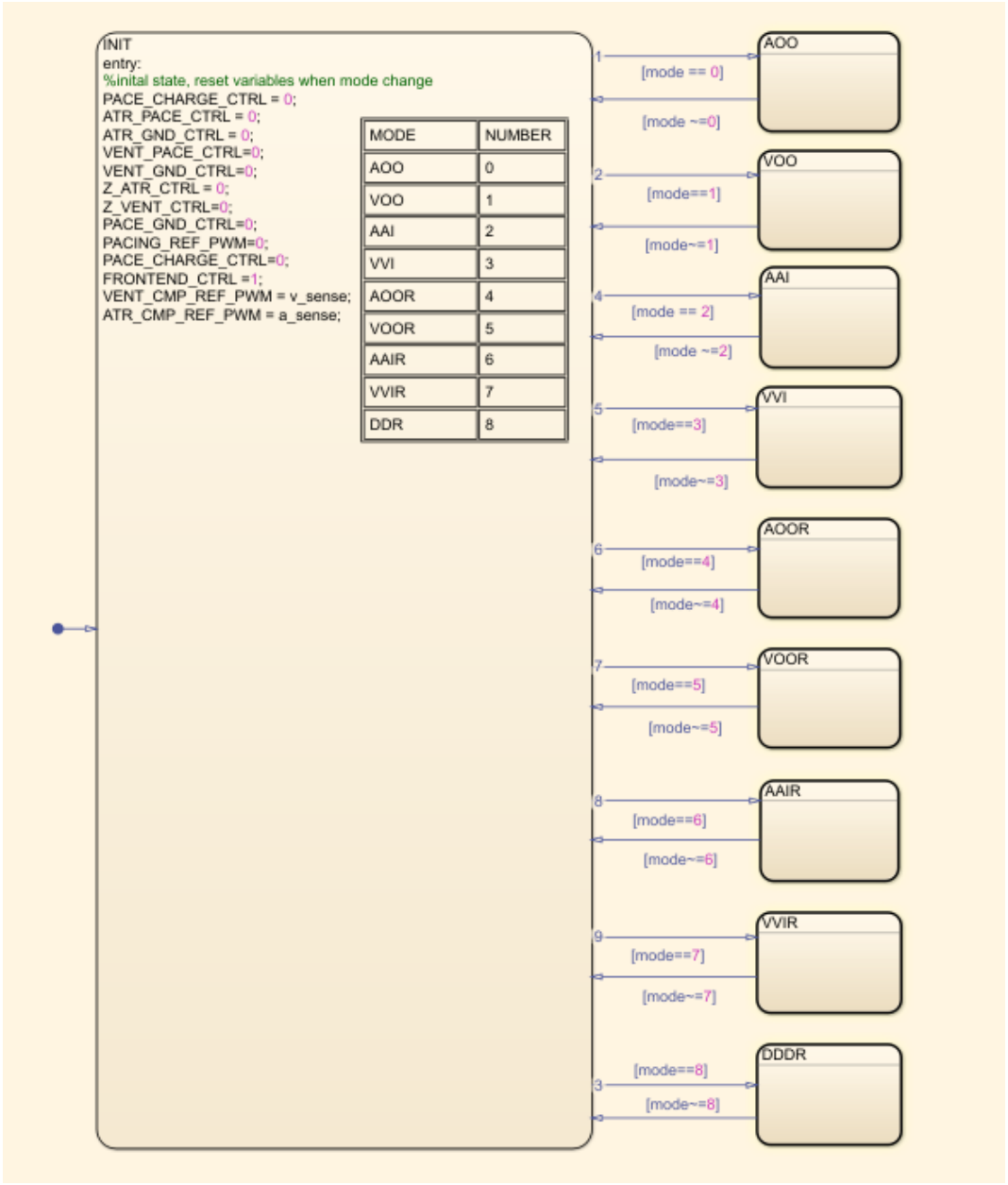

Figure 34: Mapping of Modes to Integers In Simulink

5.4.2 Main View Design & Decisions

Building on top of the previous design the main page was given plenty of new features. The main page can be seen in the screenshot below:

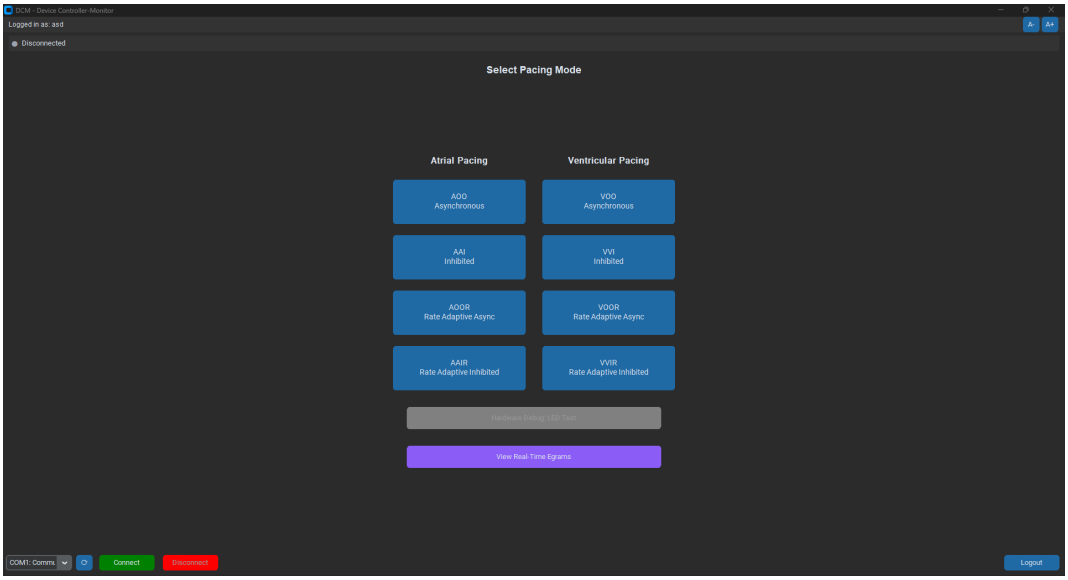

Figure 35: DCM Main Page - Deliverable 2

Changes can be seen in the top left and right corner, the bottom left corner, and the middle. The top left corner now displays COM connections with the computer rather than a mock device.

The top left features font size changing buttons, this was added for accessibility features as a previous issue was that the font size can be too small or too large when the DCM window was shurnk or expanded.

The bottom left corner now shows all COM connections on the PC, selecting the pacemaking device is typically identified as the J-Link Segger and connecting to a device that is not identified using these key words, a warning is shown as seen below:

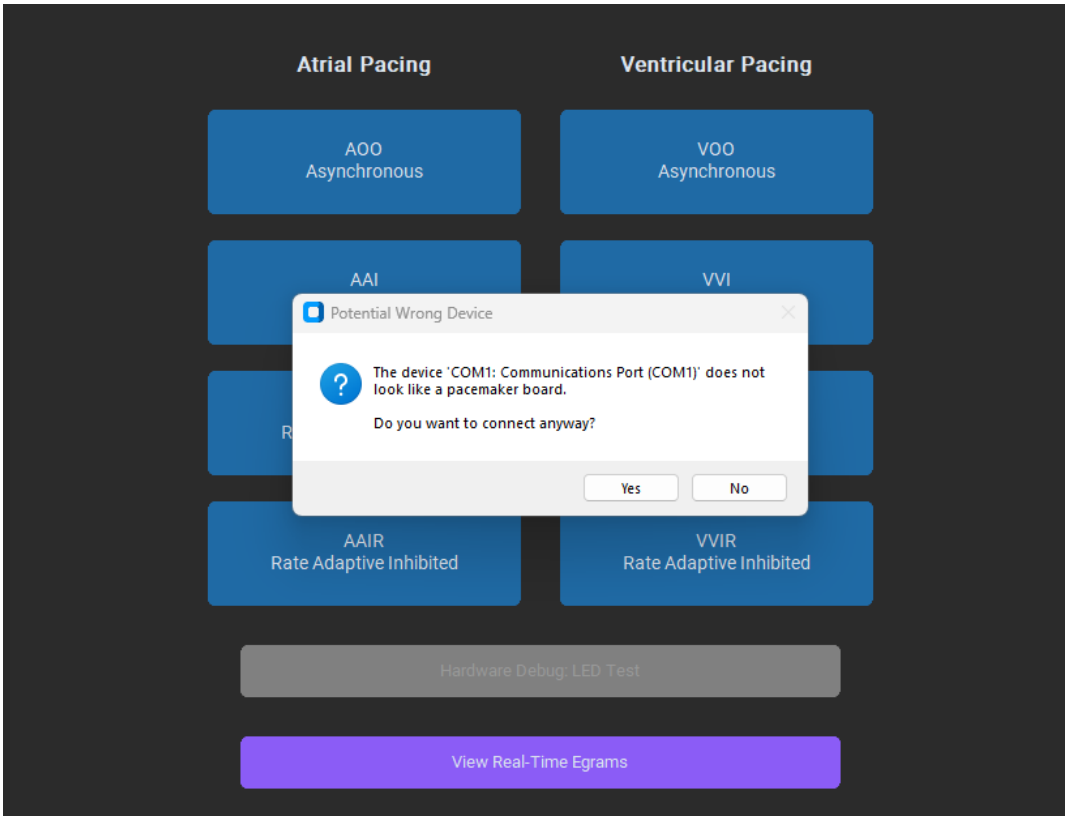

Figure 36: Warning of Potentially Wrong Device

This is done through this code snippet in the controller.py file:

Code 2: Correct Device Check

```
# --- SAFETY CHECK ---
safe_keywords = ["mbed", "OpenSDA", "NXP", "DAPLink", "JLink", "Segger"]

is_likely_safe = any(keyword.lower() in port_name_display.lower() for
    keyword in safe_keywords)

if not is_likely_safe:
    response = messagebox.askyesno(
        "Potential Wrong Device",
        f"The device '{port_name_display}' does not look like a pacemaker
            board.\n\n"
        "Do you want to connect anyway?"
    )
```

Connecting anyway will not allow you to send data through as a safety feature for unrecognized devices.

If the pacemaking device is connected however, it will prompt confirming connection before establishing serial communication.

Lastly, the middle section has been expanded to both allow for expansion when font size is increased and to support all the new modes alongside the EKG view.

5.4.3 Parameter Pages Design & Decisions

The parameter page has been changed to account for all new parameters. Previously, we greyed out non-used parameters for each mode however, with the increase of parameters the view will get cluttered and more complicated to navigate. Thus, only changeable parameters for each mode is shown:

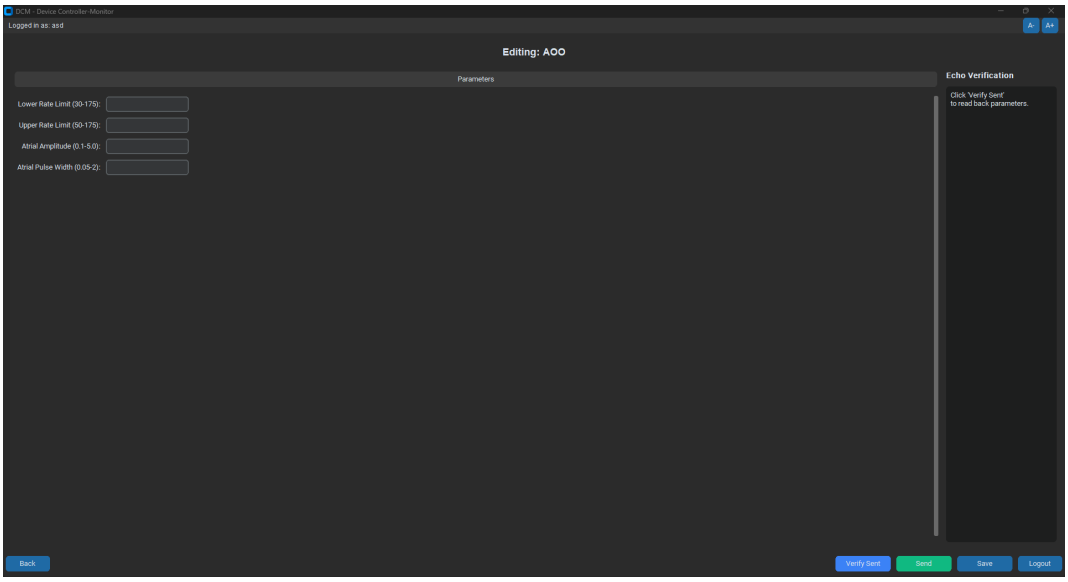

Figure 37: Updated Parameter Page for AOO Deliverable 2

This view also contains more buttons for the serial communication side of things. These features are shown on the left side of the screen with buttons verify send, send, save and the bar for echo verification.

The save button saves the parameters locally for convenience so you do not have to remember all parameters and set them all again each time you want to modify values.

The verify send sends a packet of data to the simulink that asks for the parameter values and is sent back. This is a safety feature to ensure that the right data and parameters were sent to the simulink and nothing was improperly sent or corrupted.

The send button sends a packet of data containing all the parameters to the simulink. This is in compliance with Table 11. To ensure simplicity the non-used values are just replaced with 0.0s. The simulink handling the packet unpacking is shown in below where rxdata and status are provided by the serial recieve block:

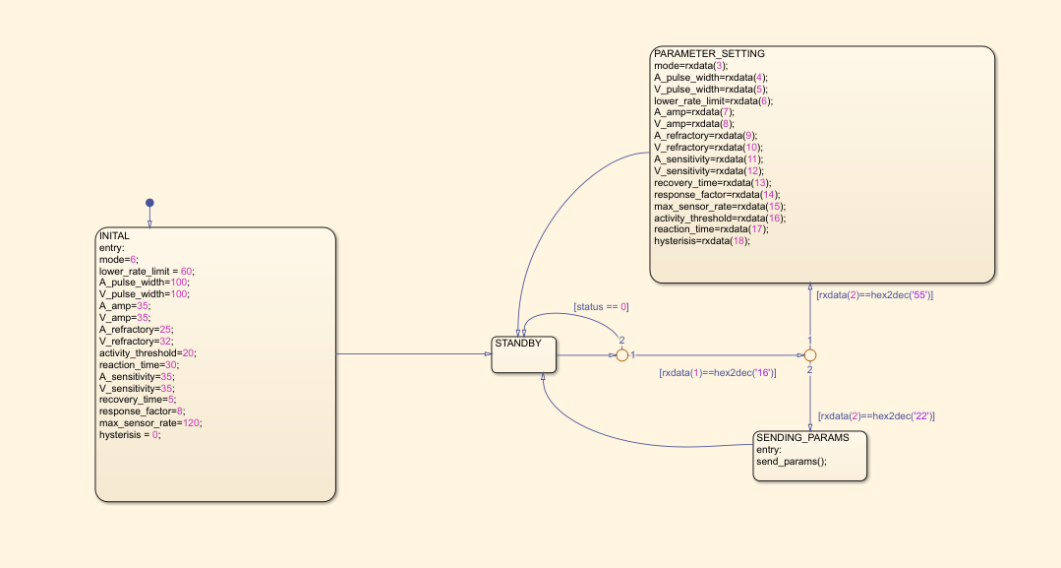

Figure 38: Serial Communication Simulink - Deliverable 2

From this, we see as well how the echo works by sending a second byte of 22 rather than 55. The send_params() function sends all parameters back by MUXing all data and sending it back via a serial transmit block.

For some pages the format is slightly different such as the rate adaptive modes. As the parameter is a word with an approximate number value a drop down menu seen below is used to select.

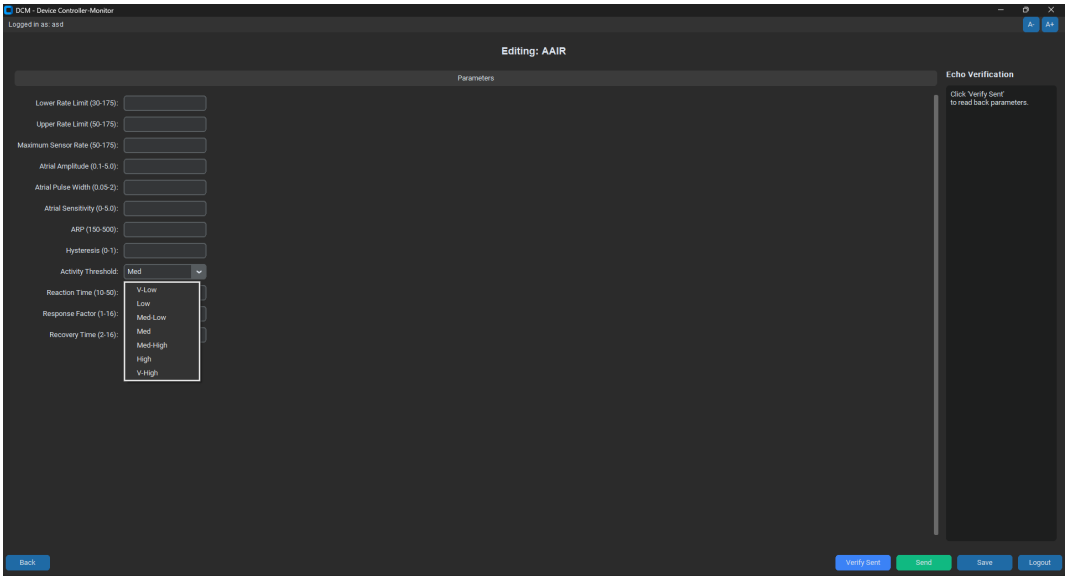

Figure 39: DCM View of AAIR - Deliverable 2

This corresponds to numerical values in the code of 1.5 at V-LOW with increments of 0.5 with higher levels of activity threshold. These values were done through testing with the use of the accelerometer as a nominal value of 1 through stationary action and a max value of 7 was found.

5.4.4 EKG View Design & Decisions

The EKG is handled through the ekg_model.py and ekg_view.py. It takes analog data from the pins relating to the atrium and ventricle. It samples the data and plots it on a graph using the matplotlib python library. This was used as it gave many useful features such as the ability to scroll and zoom on the EKG data. The view for the EKG can be seen below:

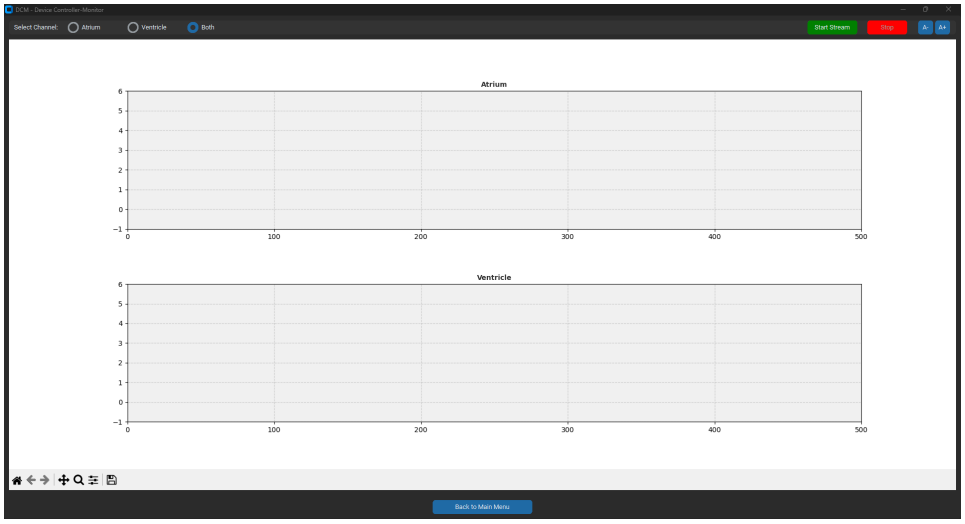

Figure 40: View of EKG Page

This page also allows you to isolate whether or not you want to look at just the atrial pulses, ventricular pulese or both with functions on the bottom left to zoom, scroll save and configure the subplots. The view of the EKG can be seen below with dummy data:

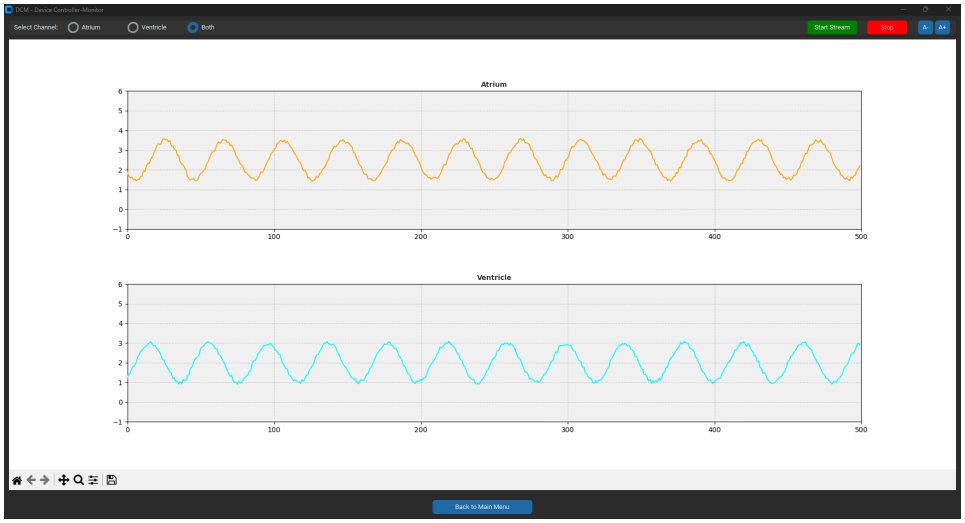

Figure 41: EKG View with Data

5.5 Assurance Cases

Some features of safety were covered in the design section of this documentation, however this section will be more comprehensive.

5.5.1 Clamping of Heart Rate For Rate Adapative Pacing

Safety Goal: For rate adaptive pacing modes, the heart rate must not be lower than the lower rate limit and the max sensor limit.

Rationale: To prevent the heart from beating too slow, lower than lower rate limit, or beating too fast, higher than max sensor limit, to ensure healthy heart operation.

Evidence: The clamp is hard locked from via how its data is processed in simulink seen in the images below. However, this can be seen live through testing as shown in the demo. The clamp is used in two spots to ensure redundancy as extra signal processing occurs.

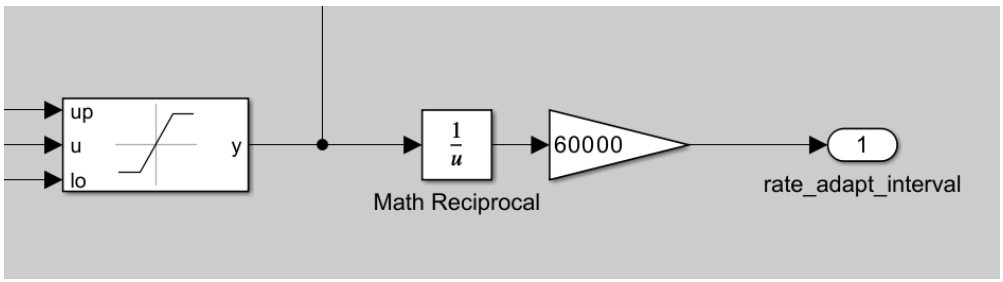

Figure 42: Clamping of Heart Rate 1

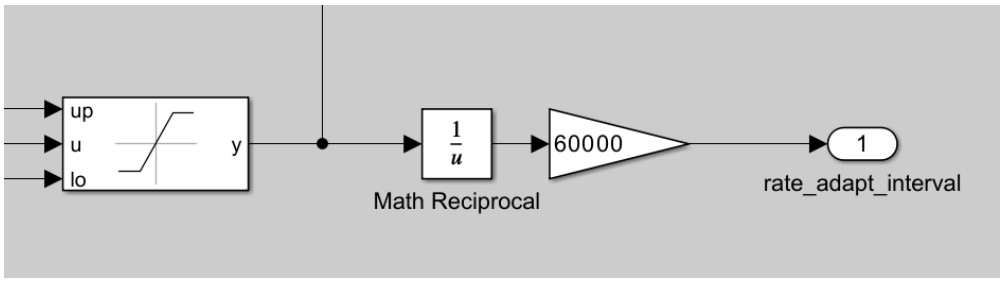

Figure 43: Clamping of Heart Rate 2

5.5.2 Echo Parameter Verification

Safety Goal: Ensuring that the data transmitted to the pacemaker by the DCM is correct.

Rationale: The ability to verify the data sent with an echo feature is imperative to ensure data was not corrupted or mismanaged when being sent to the pacemaker. Without this verification it can potentially cause issues by setting unsafe limits or incorrect parameters causing harm long term.

Evidence: The below images are of the Simulink sending the parameters back as well as the DCM sending back the sent parameters.

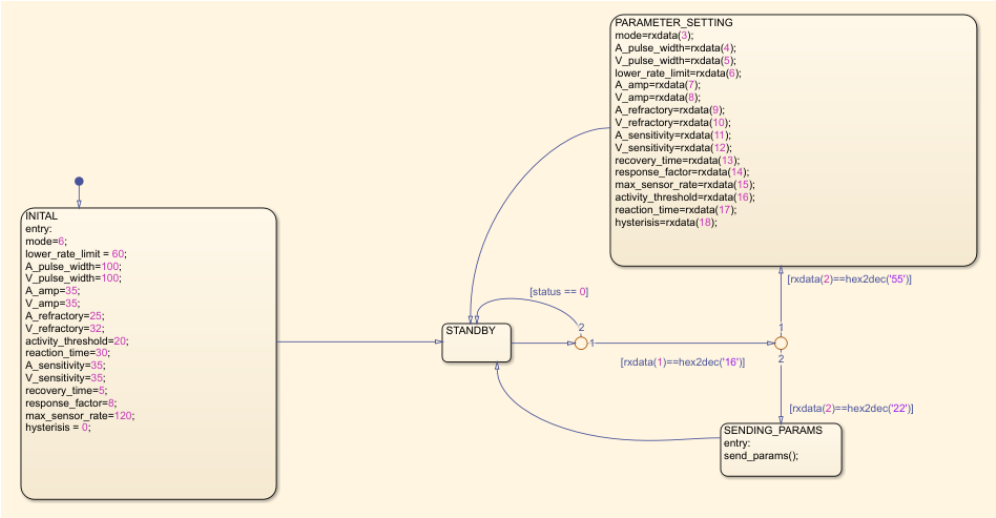

Figure 44: Echo Stateflow in Simulink

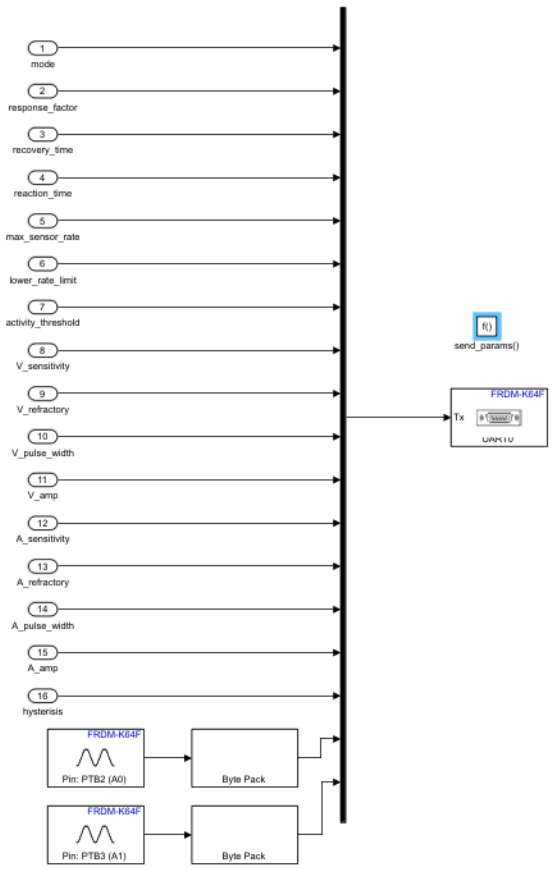

Figure 45: Send Params Function in Simulink

Code 3: Echo Function in DCM

```
        def get_echo(self):
        if not self.ser or not self.ser.is_open: return None
        try:
            self.ser.reset_input_buffer()
            self.ser.write(struct.pack(self.FMT_11_BYTES, 0x16, 0x22,
                0,0,0,0.0,0))
            resp = self.ser.read(9)
            if len(resp) != 9: return None
            u = struct.unpack('<BBBHf', resp)
            return {"red": u[0], "green": u[1], "blue": u[2], "
                switch_time": u[3], "off_time": u[4]}
        except Exception: return None
```

5.5.3 Reset Between Mode Switching

Safety Goal: Prevent unwanted data when switching between pacing modes of the pacemaker.

Rationale: When switching between modes a default state is returned to before switching to a new mode. This mode grounds all values to prevent erratic and unwanted behaviour between pacing such as switching to another mode mid pace which may cause complications

Evidence: This is soft-locked by the stateflow diagram shown below:

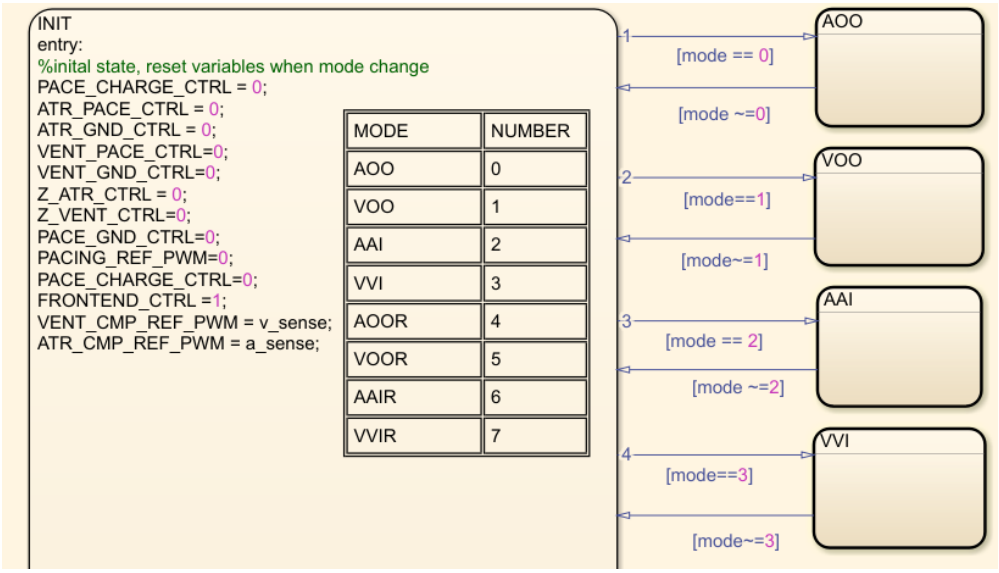

Figure 46: Default State in Simulink

5.5.4 Incorrect Device Handling

Safety Goal: When trying to use an unrecognized device the DCM will not allow serial communication and warn the user.

Rationale: To prevent the DCM from harming an unrecognized device by sending data and to warn the user that the pacemaker is currently not the device being configured.

Evidence: A warning message pops up when an unrecognized device is connected to with code to check the COM for recognizable identifiers.

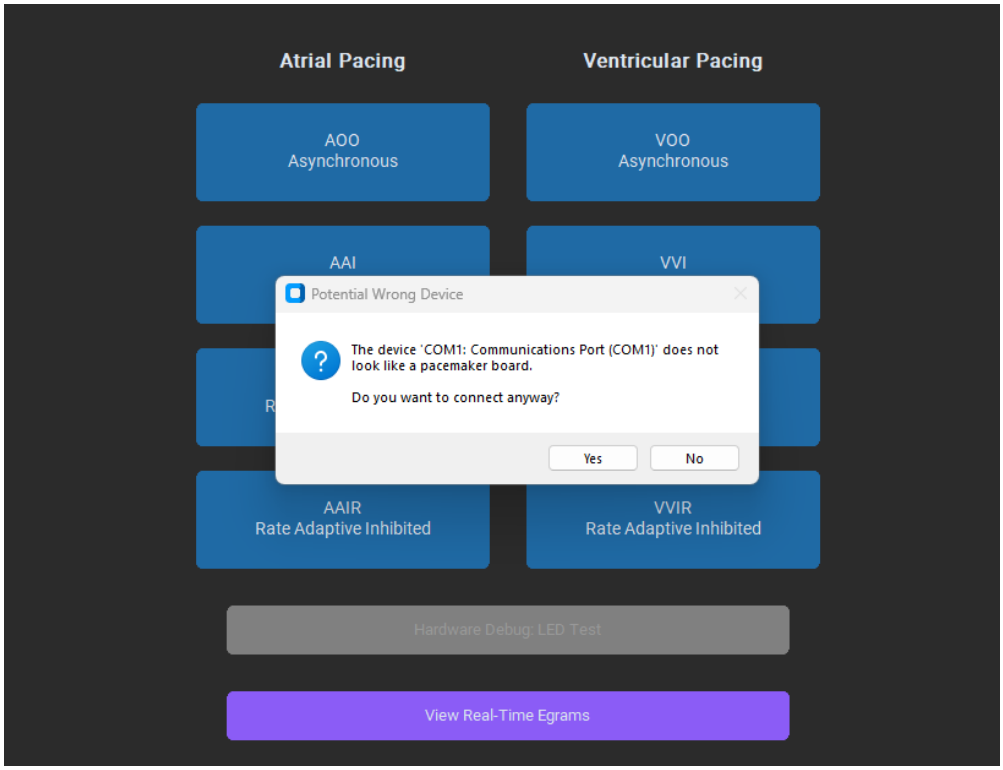

Figure 47: Potentially Wrong Device Warning in DCM

Code 4: Safety Check of Device in Code

```
        # --- SAFETY CHECK ---
        safe_keywords = ["mbed", "OpenSDA", "NXP", "DAPLink", "JLink",
            "Segger"]

        is_likely_safe = any(keyword.lower() in port_name_display.lower
            () for keyword in safe_keywords)

        if not is_likely_safe:
            response = messagebox.askyesno(
                "Potential Wrong Device",
                f"The device '{port_name_display}' does not look like a
                    pacemaker board.\n\n"
                "Do you want to connect anyway?"
            )
```

6 Requirements Potential Changes

Pacing Modes: Addition of all pacing modes such as dual chamber modes.

7 Design Decision Potential Changes

Expansion of Libraries:

8 Module Description

Although breifly highlighted in the design section of this documentation. This section will go into more detail about the purpose of each module, key functionality, variables, and how each module interacts with one another.

As shown in Figure 1, there are 3 primary modules operating the pacemaker; **input constant variables, stateflow logic for modes, and hardware hiding.**

8.1 Input Constant Variables Module

This module is highlighted in the design section. It goes through all the state variables that are changeable parameters of the pacemaker. These inputs include general inputs such as the pacing mode, low rate interval, and hysteresis settings, as well as parameters for atrial and ventricle pacing. These parameters are then used in the stateflow logic module to complete pacing to user specifications.

8.2 Stateflow Logic

This module has many submodules which are also covered in the design section. The overall state machine controlling mode selection and reseting of variables to prevent unwanted pacing behaviour is shown in Figure 4. This module converts input parameters into raw data that can be further converted to electrical signals. This module recieves data from the input constant variables module as well as from the monitored input variables. This module then feeds into hardware hiding.

8.3 Hardware Hiding

Hardware hiding is also briefly covered in the design section. The purpose of this module is to convert the signals from the stateflow logic module into outputted electrical signals through pins. As shown in Figure 9, pins are mapped to certain electrical signals such as atrial outputs, ventricle outputs, front end signals, and general pin configurations such as grounding pins. This is designed to ensure coding and modules are easier to debug with higher level identification and function of each GPIO.

9 Testing

9.1 SimuLink Mode Testing

9.1.1 Testing of AOO

Purpose: The purpose of this test is to test basic AOO functionality.

Input Conditions: General inputs of mode = 0, AOO, and hysteresis = 0, off, and standard atrial inputs.

Expected Output: Consistent, evenly spaced pulses in the output with disregard for natural heart beats.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 10.

Result: Pass

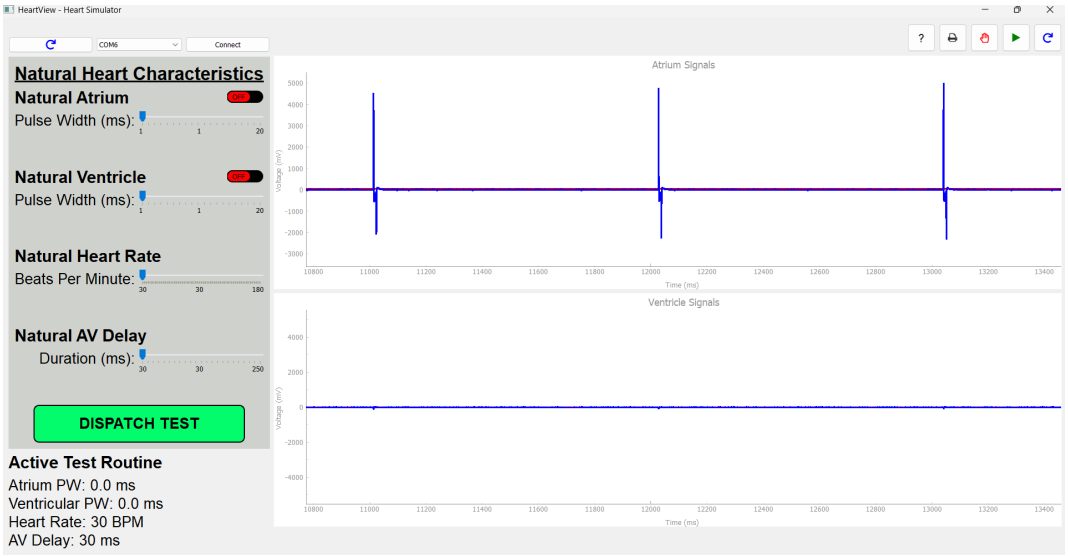

Figure 48: AOO Test

Figure 11 below shows a zoomed in view of one of the pulses in the AOO test above, Figure 10.

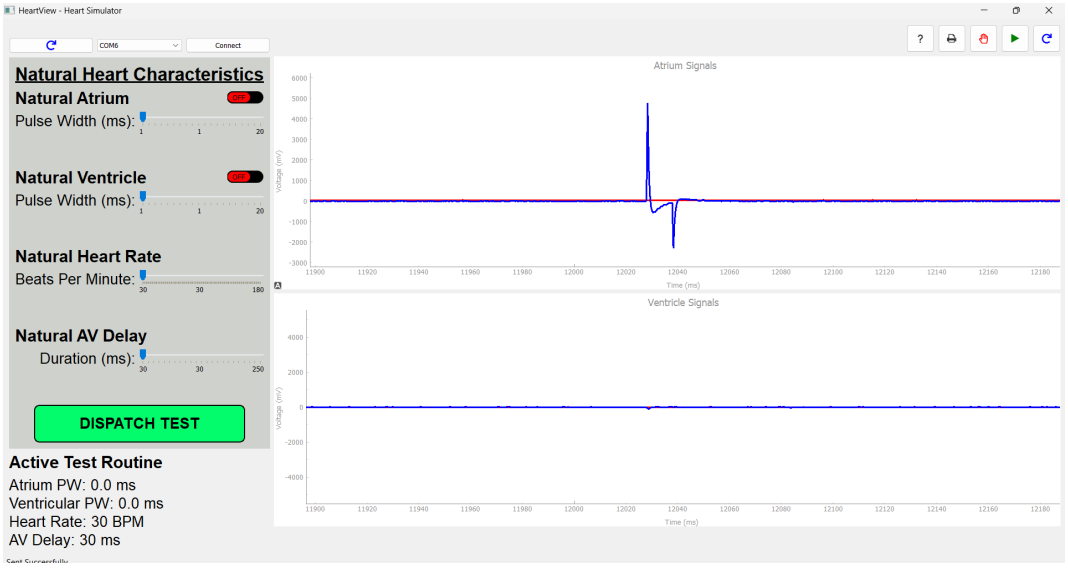

Figure 49: Close-up of AOO Pulse

9.1.2 Testing of VOO

Purpose: The purpose of this test is to test basic VOO functionality.

Input Conditions: General inputs of mode = 1, VOO, and hysteresis = 0, off, and standard ventricle inputs.

Expected Output: Consistent, evenly spaced pulses in the output with disregard for natural heart beats.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 12.

Result: Pass

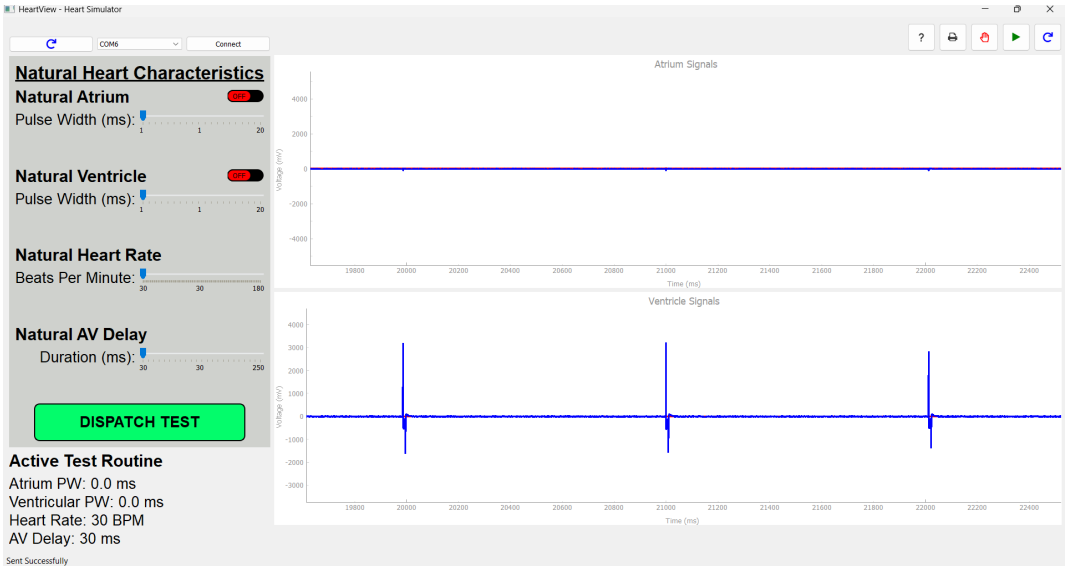

Figure 50: VOO Test

Figure 13 below shows a zoomed in view of one of the pulses in the VOO test above, Figure 12

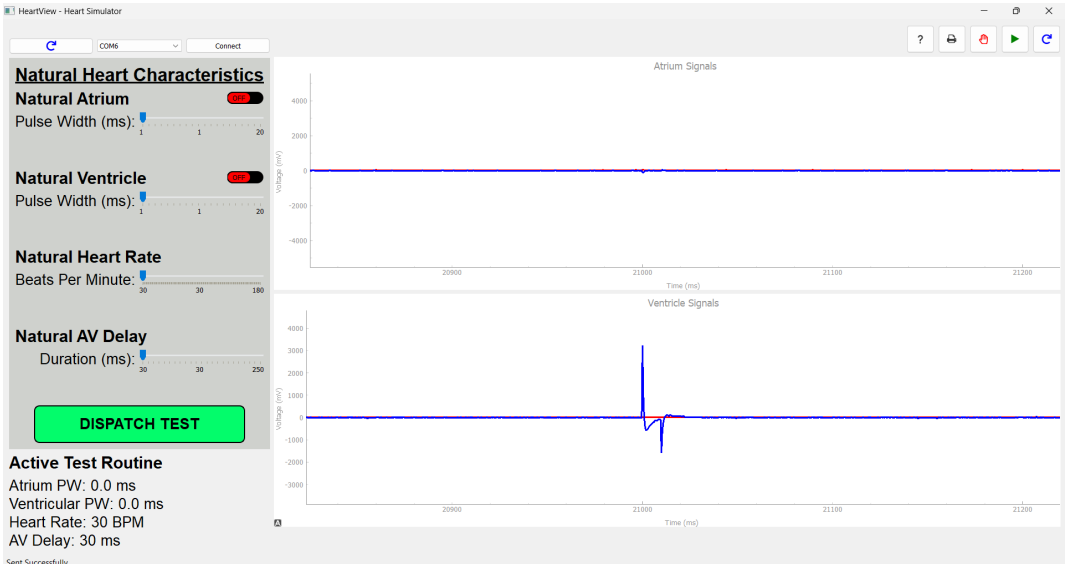

Figure 51: Close-up of VOO Pulse

9.1.3 Testing of AAI

No Natural Heart Rate

Purpose: The purpose of this test is to test basic AAI functionality with no natural heart rhythms.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 0, off, and standard artial inputs.

Expected Output: Consistent, evenly spaced pulses in the output.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 13.

Result: Pass

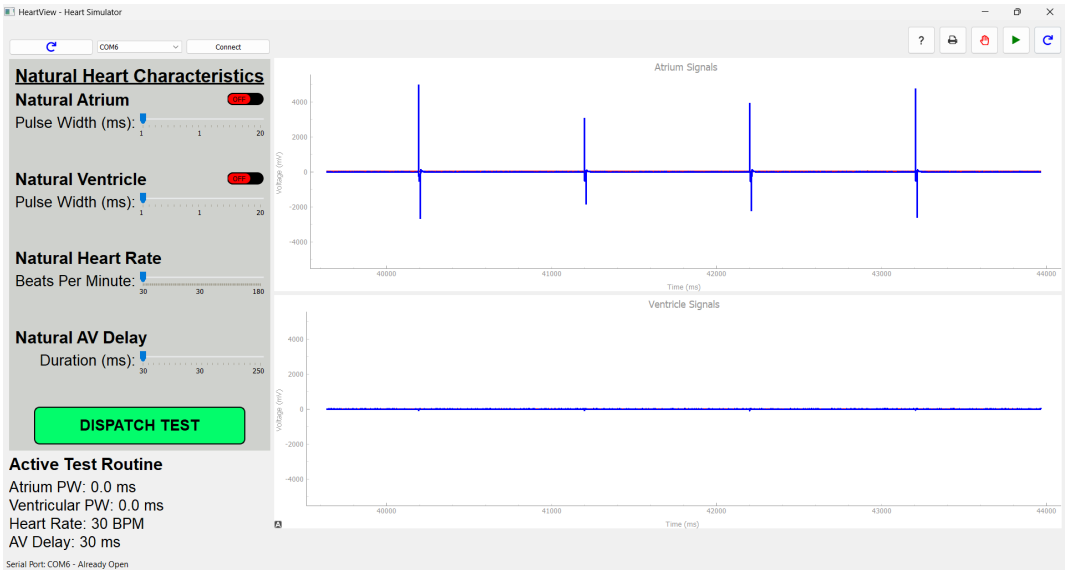

Figure 52: AAI Test No Heart Rate

Natural Heart Rate of 45 BPM

Purpose: The purpose of this test is to test basic AAI functionality at a low heart rate.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 0, off, standard artial inputs, and monitored atrial pulses at 45 BPM.

Expected Output: The pacemaker should pulse after the refactory period expires. An output of blue pulses right before the natural red pulses is expected.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 15.

Result: Pass

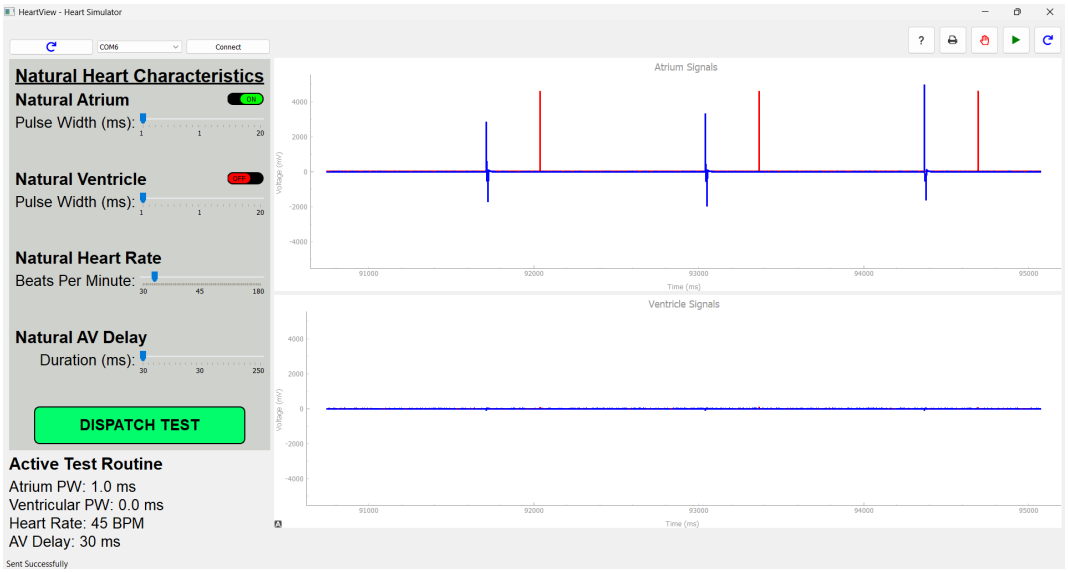

Figure 53: AAI Test 45 BPM

Natural Heart Rate of 75 BPM

Purpose: The purpose of this test is to test basic AAI functionality at a nominal heart rate.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 0, off, standard artial inputs, and monitored atrial pulses at 75 BPM.

Expected Output: As a nominal heart reate is being inputted, the pacemaker should not be delivering pacing pulses as the simulated heart rate is nominal and healthy.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 16.

Result: Pass

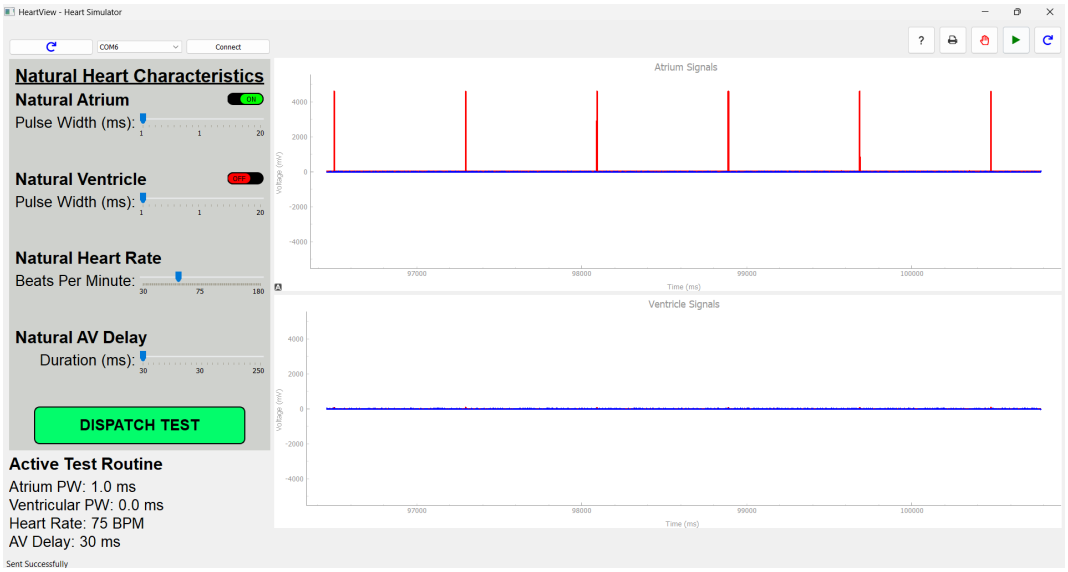

Figure 54: AAI Test 75 BPM

9.1.4 Testing of VVI

No Natural Heart Rate

Purpose: The purpose of this test is to test basic VVI functionality with no inputted heart rate.

Input Conditions: General inputs of mode = 3, VVI, and hysteresis = 0, off, standard ventricle inputs, and monitored ventricle pulses.

Expected Output: Consistent, evenly spaced pulses in the output.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 17.

Result: Pass

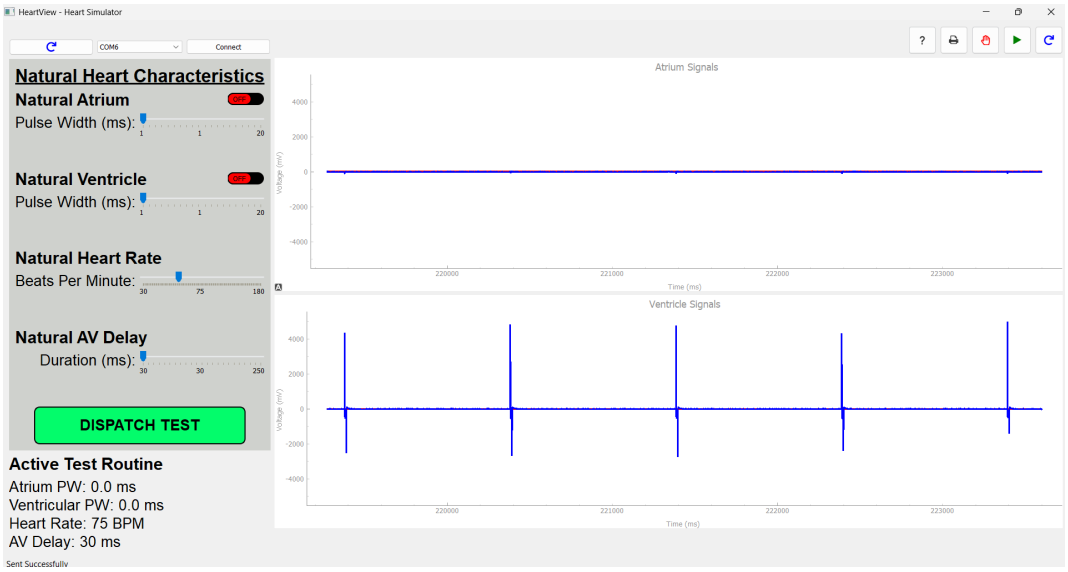

Figure 55: VVI Test No Heart Rate

Natural Heart Rate at 45 BPM

Purpose: The purpose of this test is to test basic VVI functionality with no inputted heart rate.

Input Conditions: General inputs of mode = 3, VVI, and hysteresis = 0, off, standard ventricle inputs, and monitored ventricle pulses.

Expected Output: The pacemaker should pulse after the refactory period expires. An output of blue pulses right before the natural red pulses is expected.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 18.

Result: Pass

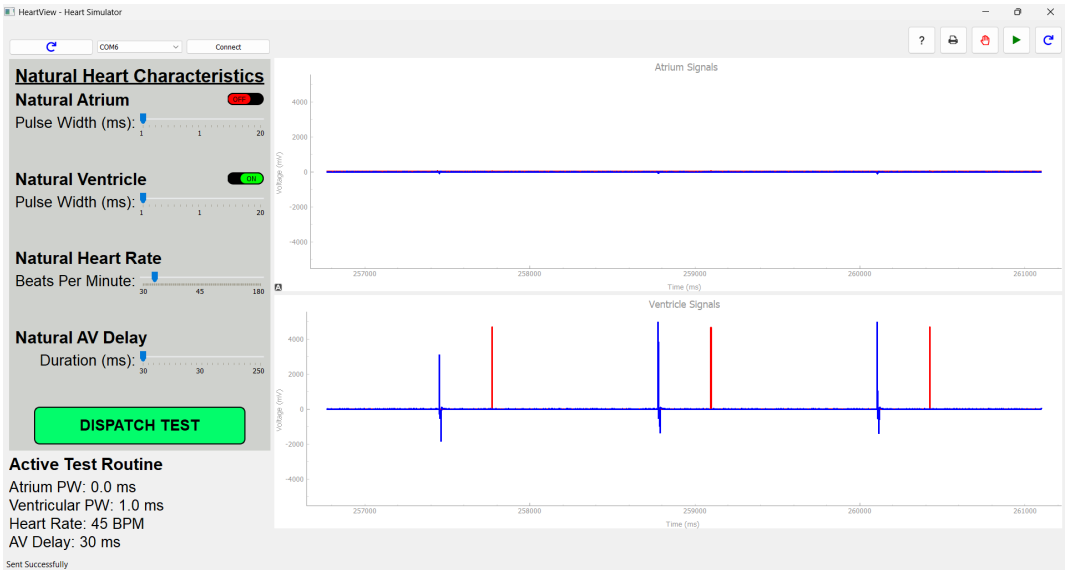

Figure 56: VVI Test 45 BPM

Natural Heart Rate at 75 BPM

Purpose: The purpose of this test is to test basic VVI functionality with no inputted heart rate.

Input Conditions: General inputs of mode = 3, VVI, and hysteresis = 0, off, standard ventricle inputs, and monitored ventricle pulses.

Expected Output: As a nominal heart reate is being inputted, the pacemaker should not be delivering pacing pulses as the simulated heart rate is nominal and healthy.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 18.

Result: Pass

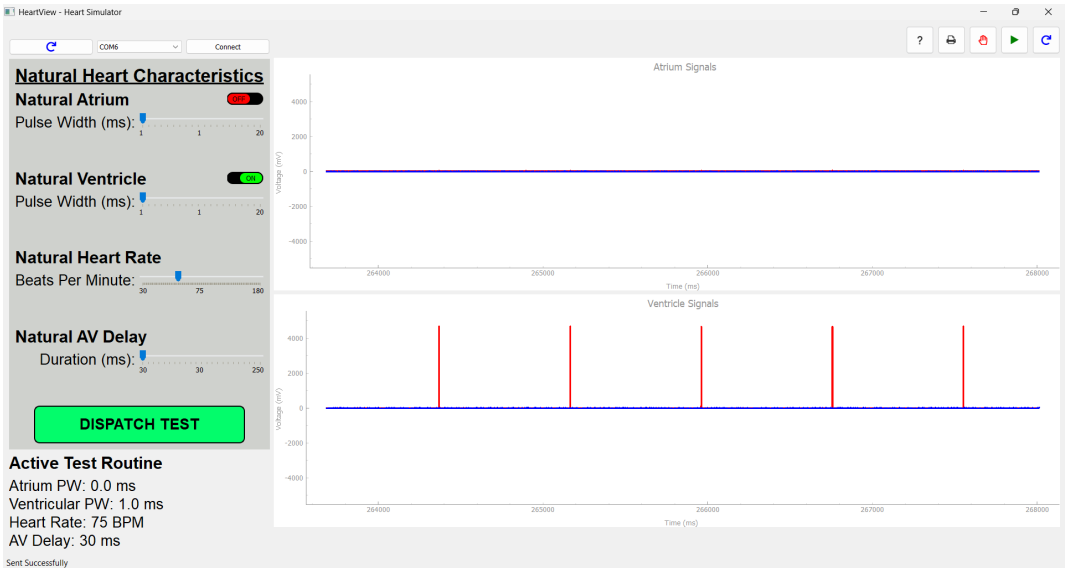

Figure 57: VVI Test 75 BPM

9.1.5 Hysteresis Testing

Hysteresis Test 1 (60 BPM)

Purpose: The purpose of this test is to test basic hysteresis mode functionality.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 1, on, standard atrium inputs, and monitored atrial pulses at 50 BPM.

Expected Output: No output of the pacemaker.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 20.

Result: Pass

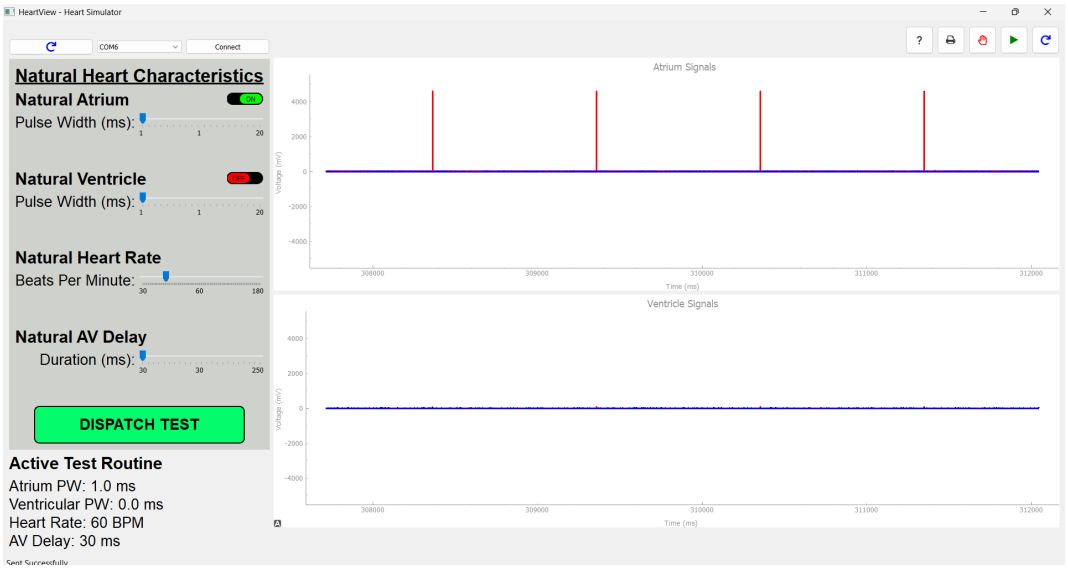

Figure 58: Hysterisis Test 1

Hysteresis Test 2 (50 BPM)

Purpose: The purpose of this test is to test basic hysteresis mode functionality.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 1, on, standard atrium inputs, and monitored atrial pulses at 50 BPM.

Expected Output: No output of the pacemaker.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 21.

Result: Pass

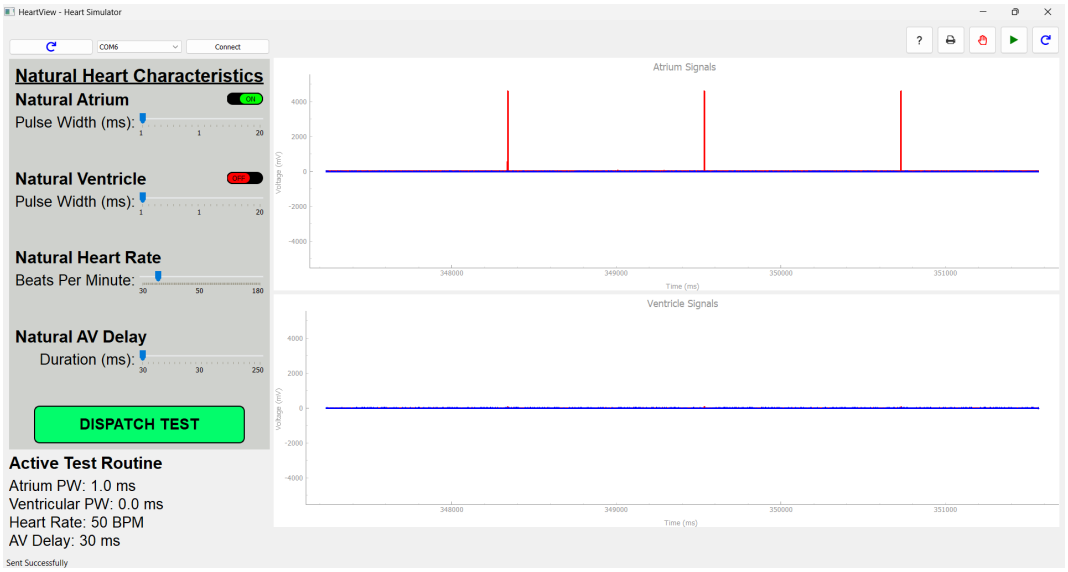

Figure 59: Hysterisis Test 2

Hysteresis Test 3 (40 BPM)

Purpose: The purpose of this test is to test basic hysteresis mode functionality.

Input Conditions: General inputs of mode = 2, AAI, and hysteresis = 1, on, standard atrium inputs, and monitored atrial pulses at 50 BPM.

Expected Output: Delayed output signals from the pacemaker.

Actual Output: Output of testing is exactly that of expected, as shown below in Figure 22.

Result: Pass

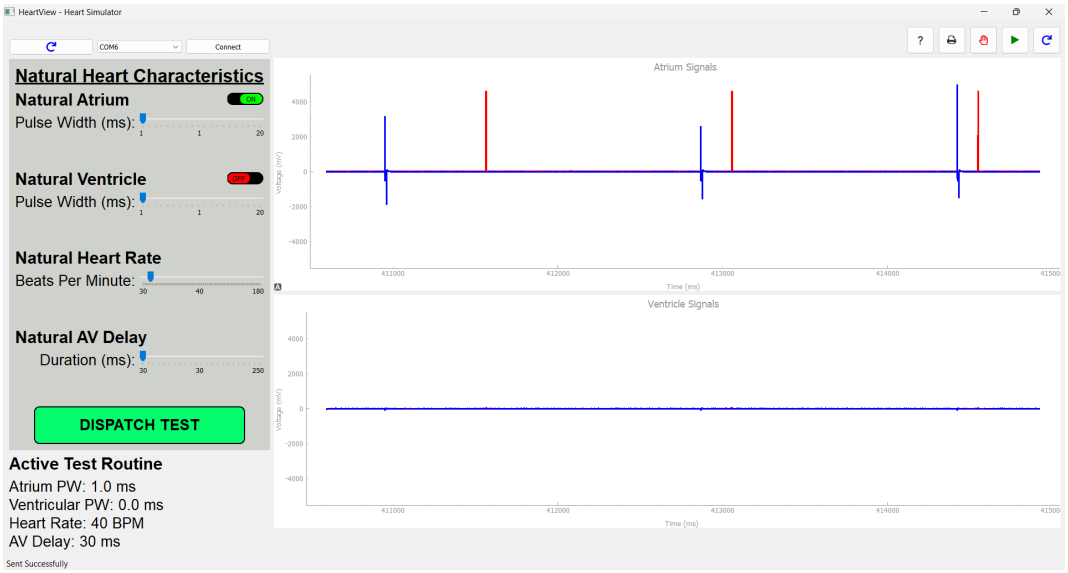

Figure 60: Hysterisis Test 3

9.2 DCM Testing

9.2.1 Login and Registration

Purpose: The purpose of this test is to verify correct storage of newly registered user data and allow login.

Input Conditions: A random username and password. This will be used again in the login screen to access the DCM.

Expected Output: A window should pop up notifying the user an account has been registered. The DCM controls should be accessible after the user logs in.

Actual Output: Dialogue is shown and the file is updated to include new user data. The user is then brought to the

Result: Pass

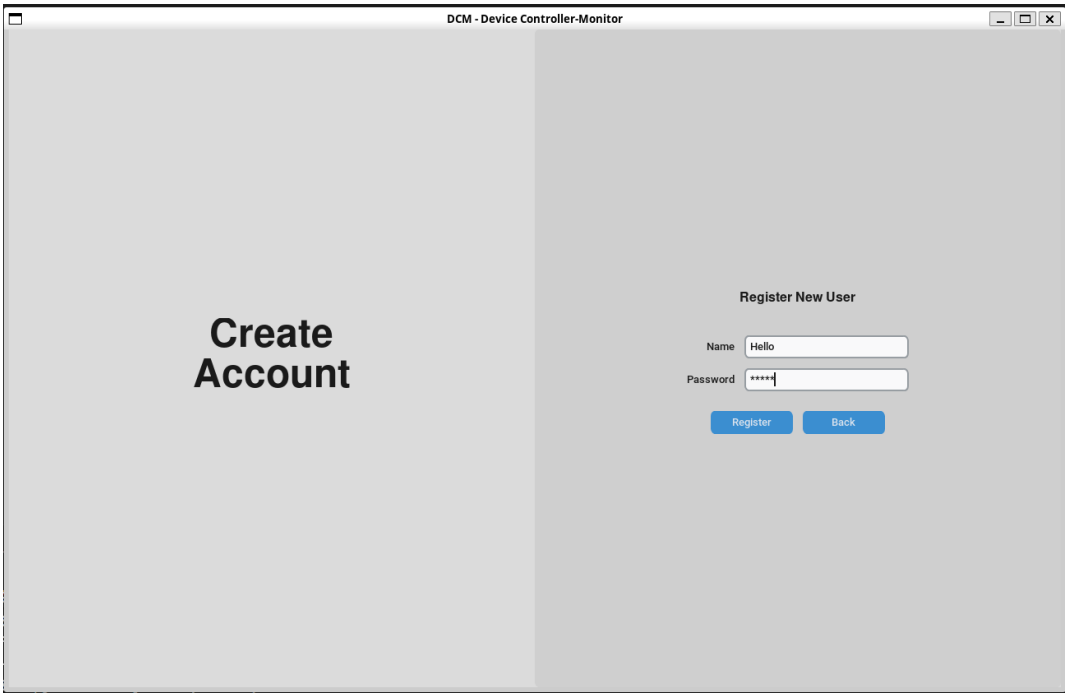

Figure 61: Registration Test

Figure 62: Registration Result

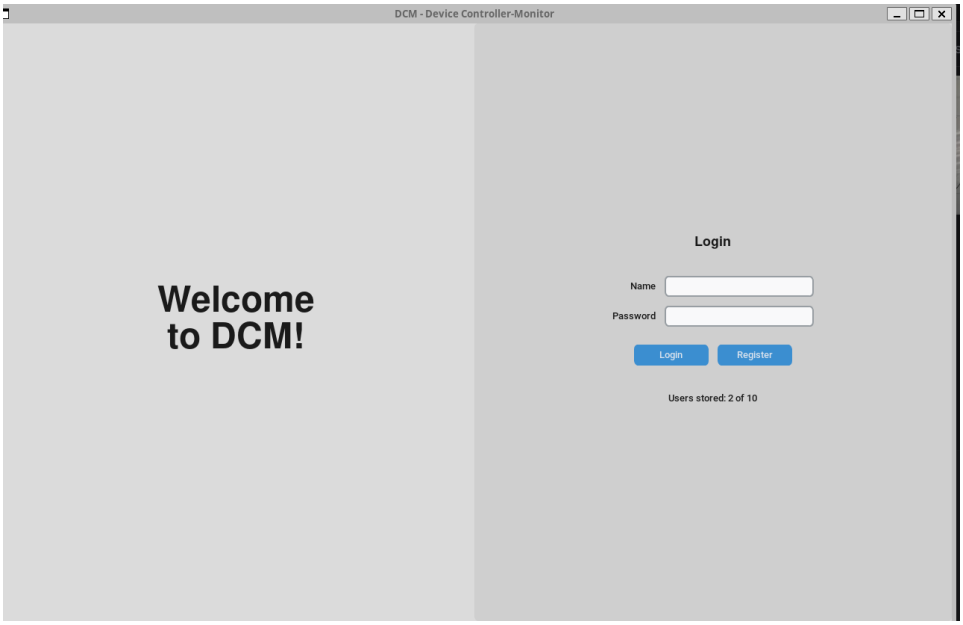

Figure 63: Login Test

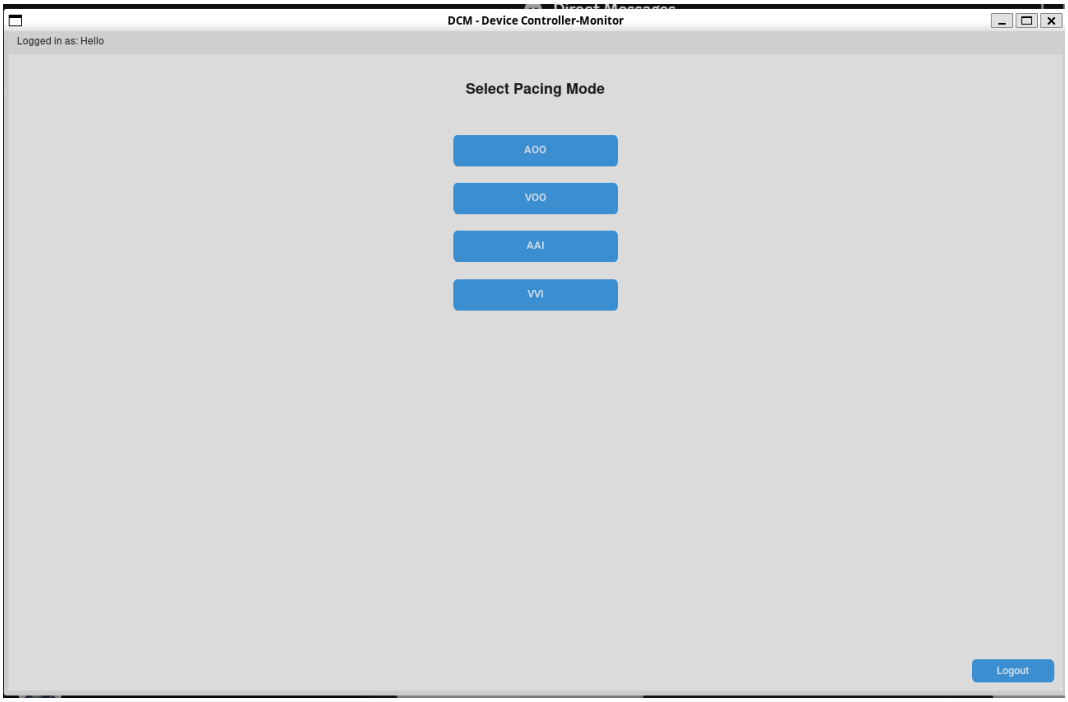

Figure 64: Login Result

9.2.2 Parameter Input Validation

Purpose: To enforce numeric types within an allowed range and to ensure the upper rate interval is greater than lower rate interval.

Input Conditions: Entering a non-numeric, an out of range number and an upper rate interval that is greater than lower rate interval in parameter settings.

Expected Output: Invalid input dialogue is shown and parameter changes are not saved.

Actual Output:

Result: Pass

The below figures show the general parameter page with some values inputted as well as the results of putting invalid values into parameter page.

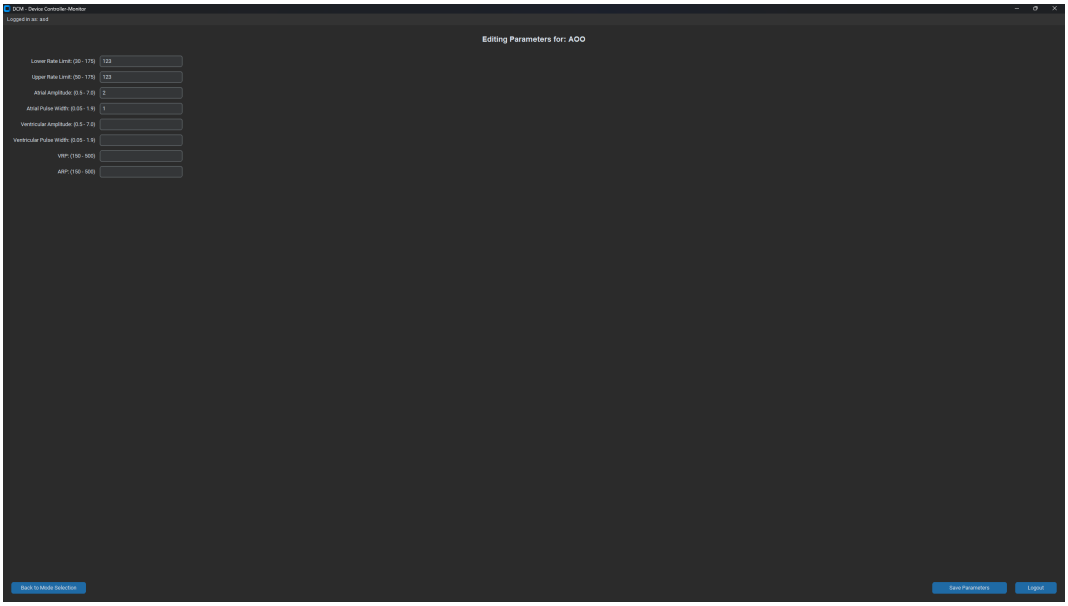

Figure 65: DCM Parameter Page with Values Filled

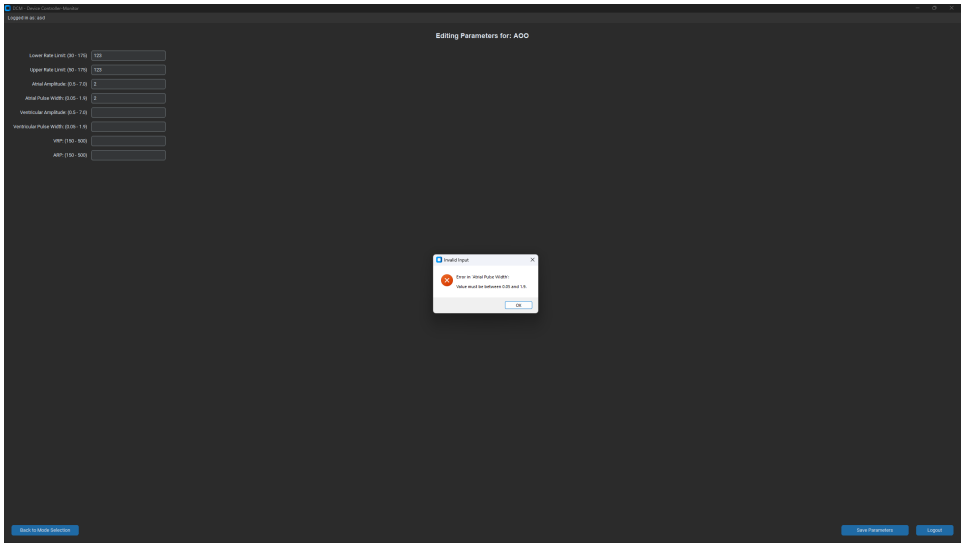

Figure 66: DCM Parameter Error When Number Inputted is Out of Range

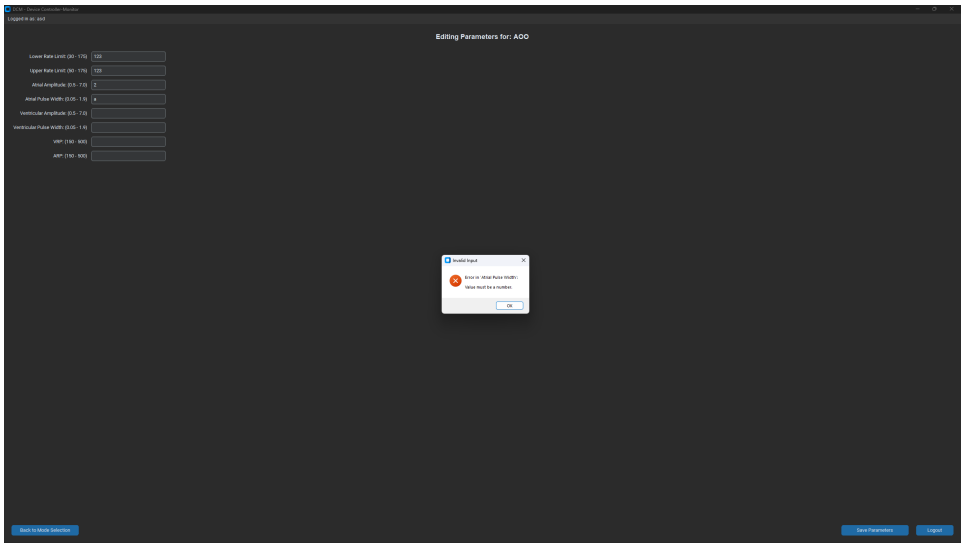

Figure 67: DCM Parameter Error When Non-Number is Inputted

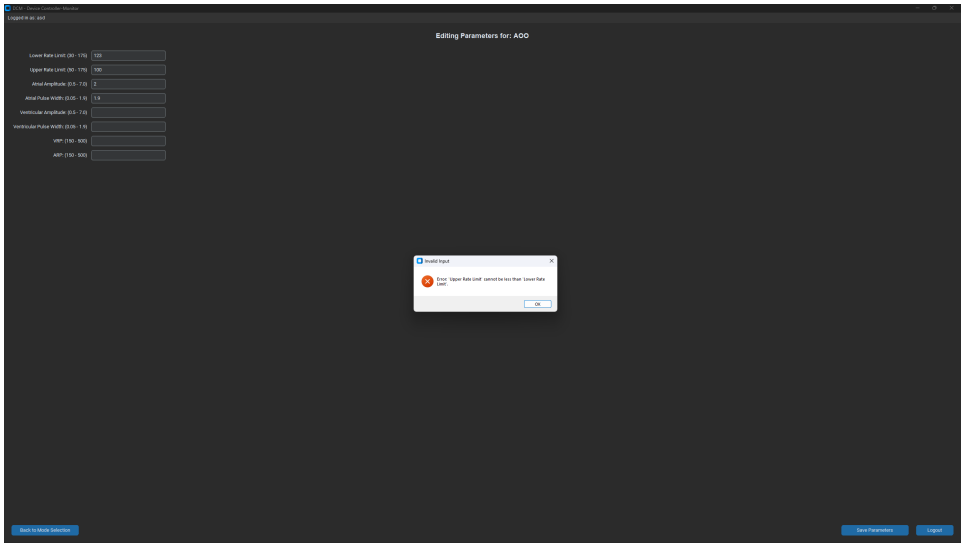

Figure 68: DCM Parameter Error When URL is Larger Than LRL

9.2.3 Mode Selection and Data Retrieval

Purpose: To test data storage, ensuring proper saving of user data

Input Conditions: Registering account, logging in, and saving parameters.

Expected Output: User data is now found in the associated JSON files.

Actual Output: Registered user data and parameters are found in their respective JSON files.

Result: Pass

This test was done in conjunction to previous tests except with a different user registered. A user "asd" with password "asd" was used for faster log ins. The following images are of the JSON files and the saved parameters from the previous test.

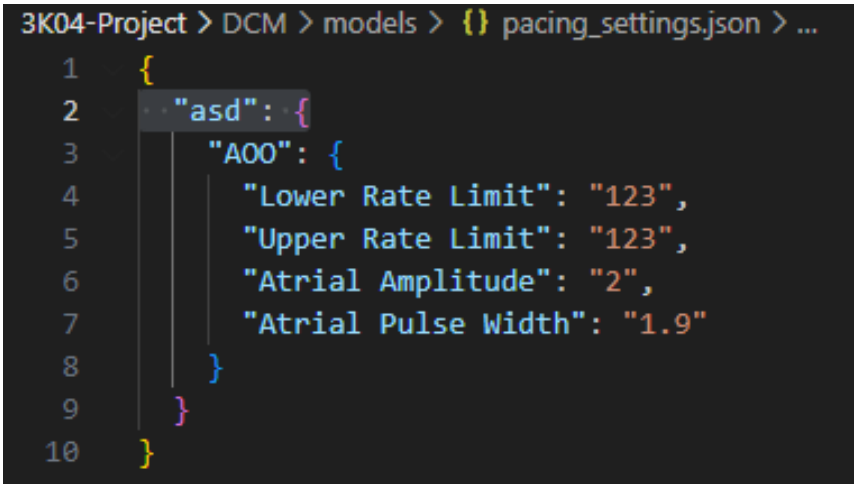

Figure 69: Stored Parameter File

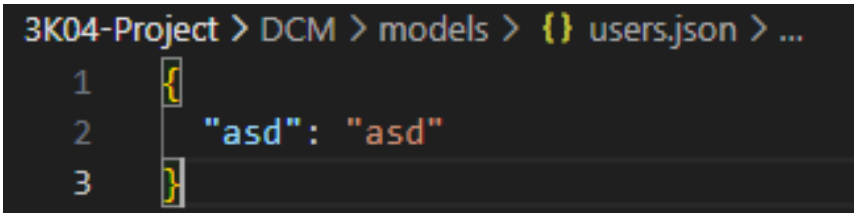

Figure 70: Stored Users File

9.3 Serial Testing

9.4 Rate Adaptive Test

10 GenAI Usage

We used a Generative AI assistant to support development of the DCM. It provided starter boilerplate for a Tkinter app with a welcome screen, registration and login, JSON storage capped at ten users, which we then adapted and tested. We also used it to clarify Python functions and libraries such as Tkinter, JSON, etc. and to troubleshoot installing tkinter. We had AI to clarify comments within the code as well. All design decisions, requirements, and validation were done by our team, and we reviewed and verified all AI outputs before inclusion.