PACEMAKER DOCUMENTATION

3K04 Assignment 1

Asheel Dowd - 400470770
Anna Naumova - 400458754
Daniel Chirackal - 400452897
Jaavin Mohanakumar - 400465588
Jacob Foster - 400474753
Tyler Smith - 400435424

Table of Contents

System Overview

Pacemakers are safety critical systems that provide electrical stimulation to the chambers of the heart to treat arrhythmias or irregular heartbeats. Our system consists of the pacemaker hardware kit and the Device Controller-Monitor (DCM). The DCM acts as the human interface for the physical pacemaker hardware. This allows for care to be prescribed to the patient by designated healthcare professionals, such as selecting different pacing rates and modes. The pacemaker implements the care prescribed through the DCM by performing functions such as pacing the atrium and ventricle of the heart, and sensing natural pulses.

At this stage, the DCM and pacemaker do not communicate, however the structure is in place to be able to integrate the two of them moving forward. The pacemaker is programmed in Simulink, allowing a graphical representation of blocks/modules to be obtained. This emphasizes the concepts discussed in class involving developing modules with high cohesion and low coupling. Additionally, with the chart Simulink type, an FSM for the operation of the pacemaker logic is implemented. These software concepts are also applied to the DCM design, allowing for modular and reusable functionality.

Currently, the pacemaker is able to provide 4 modes of operation, allowing for ventricular or atrial pacing with or without inhibition due to sensed pulses. The DCM allows for user login, parameter selection, and basic data visualization. This base of functionality will continue to be iterated and developed upon as the second part of the assignment commences.

Design

Requirements

DCM

The DCM must handle user sign-ins and registrations but not allow more than ten users and their data to be stored locally. The system must protect (ie. hashing) user passwords in case of a compromised storage device. The DCM must hold the data entered by the user for the four pacing modes. The interface must display the programmable data inputted by the user for each pacing mode and project it on the corresponding electrogram. The front end must also inform the user when the DCM and Pacemaker communicate and their status. The DCM must convey if a new Pacemaker (different than the one previously used) approaches the device. The UI must be operable by any healthcare professional, regardless of tech literacy.

Pacemaker

At this stage the pacemaker is to provide pacing to the heart in four modes as specified throughout. These are AOO, VOO, AAI, VVI. The pacemaker must have the ability to sense pulses on both the atrium and ventricle. The pacemaker must be able to provide pacing to both the atrium and ventricle of varying amplitudes, pulse widths, and rates. The pacemaker must be able to utilize a variable refractory period to disregard subsequent pulses.

AOO

In AOO mode, the atrium is paced at a constant rate that is externally determined, regardless of heart activity. No sensing, ventricular pacing, or rate adaptation is performed.

VOO

In VOO mode, the ventricle is paced at a constant rate that is externally determined, regardless of heart activity. No sensing, atrium pacing, or rate adaptation is performed.

AAI

In AAI mode, the atrium is paced at a rate that is externally determined. While a constant rate is set, generated pulses are inhibited if a (presumably naturally generated) pulse is detected in the atrium. No ventricle pacing or rate adaptation is performed. An atrial refractory period allows subsequently detected pulses to be ignored in the case that artificially generated pulses are detected.

VVI

In VVI mode, the ventricle is paced at a rate that is externally determined. While a constant rate is set, generated pulses are inhibited if a (presumably naturally generated) pulse is detected in the ventricle. No atrium pacing or rate adaptation is performed. A ventricular refractory period allows subsequently detected pulses to be ignored in the case that artificially generated pulses are detected.

Implementation – Pacemaker

Hardware + Variables

Our input variables are listed below.

Name	Source	Functional Description
Mode	User specified (constant)*	The mode is an enumeration used to specify which operating mode the pacemaker should be operating in. 1 = AOO, 2 = VOO, 3 = AAI, 4 = VVI.
ATR_pulseWidth	User specified (constant)*	This specified the pulse width in milliseconds for pulses of the atrium. Utilized in AOO and AAI modes.
VENT_pulseWidth	User specified (constant)*	This specified the pulse width in milliseconds for pulses of the ventricle. Utilized in VOO and VVI modes.
ARP	User specified (constant)*	The atrial refractory period specifies the period of time in milliseconds that follows a sensed pulse on the atrium, in which a detection will not result in an inhibition of the next pulse. Used in the AAI mode.
VRP	User specified (constant)*	The ventricular refractory period specifies the period of time in milliseconds that follows a sensed pulse on the ventricle, in which a detection will not result in an inhibition of the next pulse. Used in the VVI mode.
ATR_pulseAmplitude	User specified (constant)*	The atrial pulse amplitude specified the amplitude of atrial pulses in volts. Used in AOO and AAI modes.
VENT_pulseAmplitude	User specified (constant)*	The atrial pulse amplitude specified the amplitude of atrial pulses in volts. Used in AOO and AAI modes.
LRL	User specified (constant)*	The lower rate limit doubles as the specified BPM for pacing in this iteration of the pacemaker. Used in all modes.
ATR_CMP_DETECT	Hardware (Pin D0)	The atrial comparator detect signal is a rising edge on detection of an atrial pulse. Used for sensing in AAI mode.
VENT_CMP_DETECT	Hardware (Pin D1)	The ventricle comparator detect signal is a rising edge on detection of an ventricular pulse. Used for sensing in VVI mode.

Table 1: Input variables and their uses.

*: This will be an input from the DCM in the future.

A note on URL: The upper rate limit is not specified in this current iteration of the code. This is because all variables marked with an asterisk above are specified by us when testing the code.

Because a separate BPM rate is not specified, comparing to LRL and URL is not necessary. We understand that it will need to be incorporated into future design decisions.

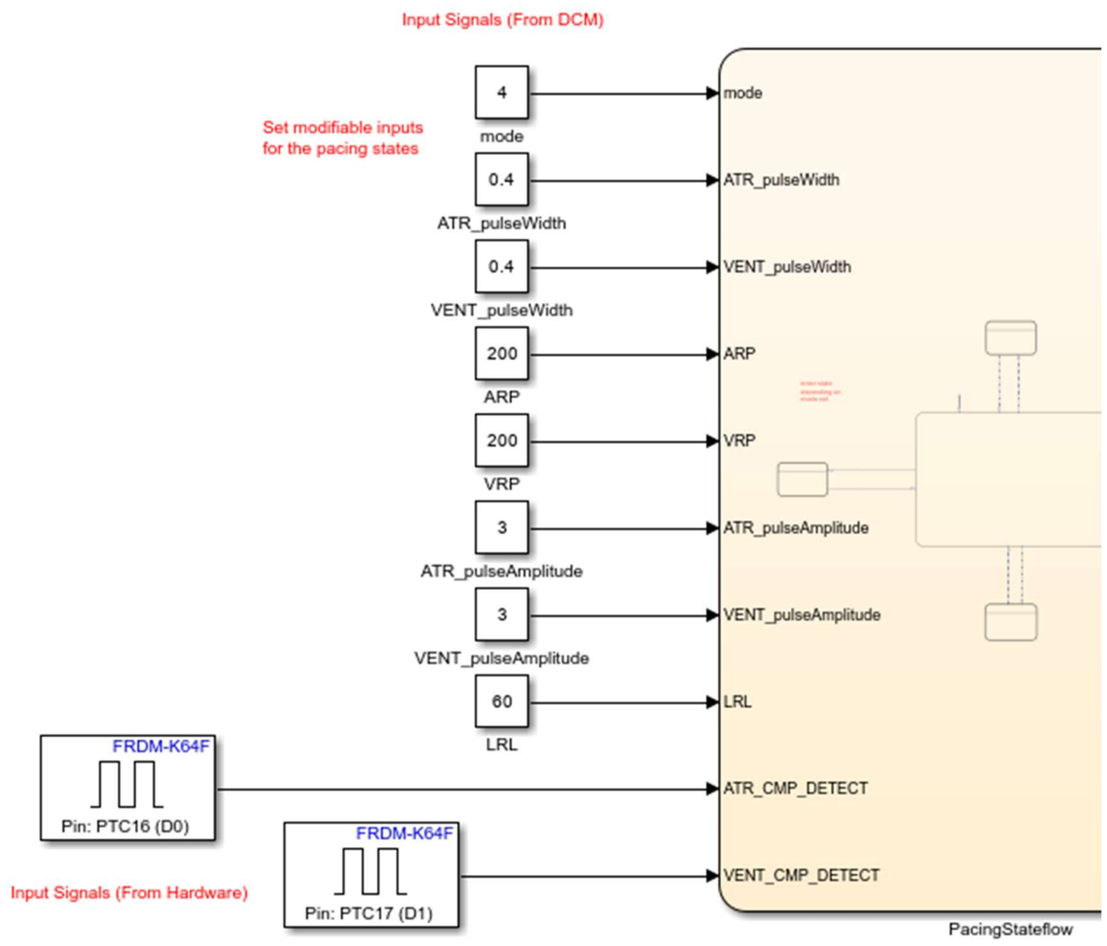


Figure 1: Pacemaker input variables

All constants and hardware inputs currently feed directly into the stateflow chart. This is because no pre-processing is performed on or with any of these variables. This setup allows us to quickly and easily test with different input values. Please see the future considerations section for how this will change going forward.

Our output variables are listed below.

Name	Destination	Functional Description
Z_VENT_CTRL	Hardware (Pin D7)	Controls the connection of the ventricle to the impedance circuit.
ATR_PACE_CTRL	Hardware (Pin D8)	This signal is used to discharge the charging capacitor into the atrium, used in pacing the atrium (modes AOO and AAI).

PACE_CHARGE_CTRL	Hardware (Pin D2)	This signal is used to charge the charging capacitor. The charging capacitor is charged in between paces for all states.
PACE_GND_CTRL	Hardware (Pin D10)	This controls the connection from the heart ground rind to ground. Used to pace either chamber of the heart. Used in all modes.
VENT_PACE_CTRL	Hardware (Pin D9)	This signal is used to discharge the charging capacitor into the ventricle, used in pacing the ventricle (modes VOO and VVI).
VENT_GND_CTRL	Hardware (Pin D12)	Used to discharge the blocking capacitor back into the ventricle in conjunction with charging, used in VOO and VVI.
ATR_GND_CTRL	Hardware (Pin D11)	Used to discharge the blocking capacitor back into the atrium in conjunction with charging, used in AOO and AAI.
Z_ATR_CTRL	Hardware (Pin D4)	Controls the connection of the atrium to the impedance circuit.
FRONTEND_CTRL	Hardware (Pin D13)	This signal enables sensing of pulses on both the atrium and ventricle. Used in all modes.
PACING_REF_PWM	Hardware (Pin D5)	This is an output PWM signal that is used to charge the charging capacitor to the proper voltage level before discharging into the heart.
VENT_CMP_REF_PWM	Hardware (Pin D3)	This is an output PWM signal that sets the comparator voltage that is used to sense whether a pulse has been detected on the ventricle.
ATR_CMP_REF_PWM	Hardware (Pin D6)	This is an output PWM signal that sets the comparator voltage that is used to sense whether a pulse has been detected on the atrium.

Table 2: Output variables and their uses.

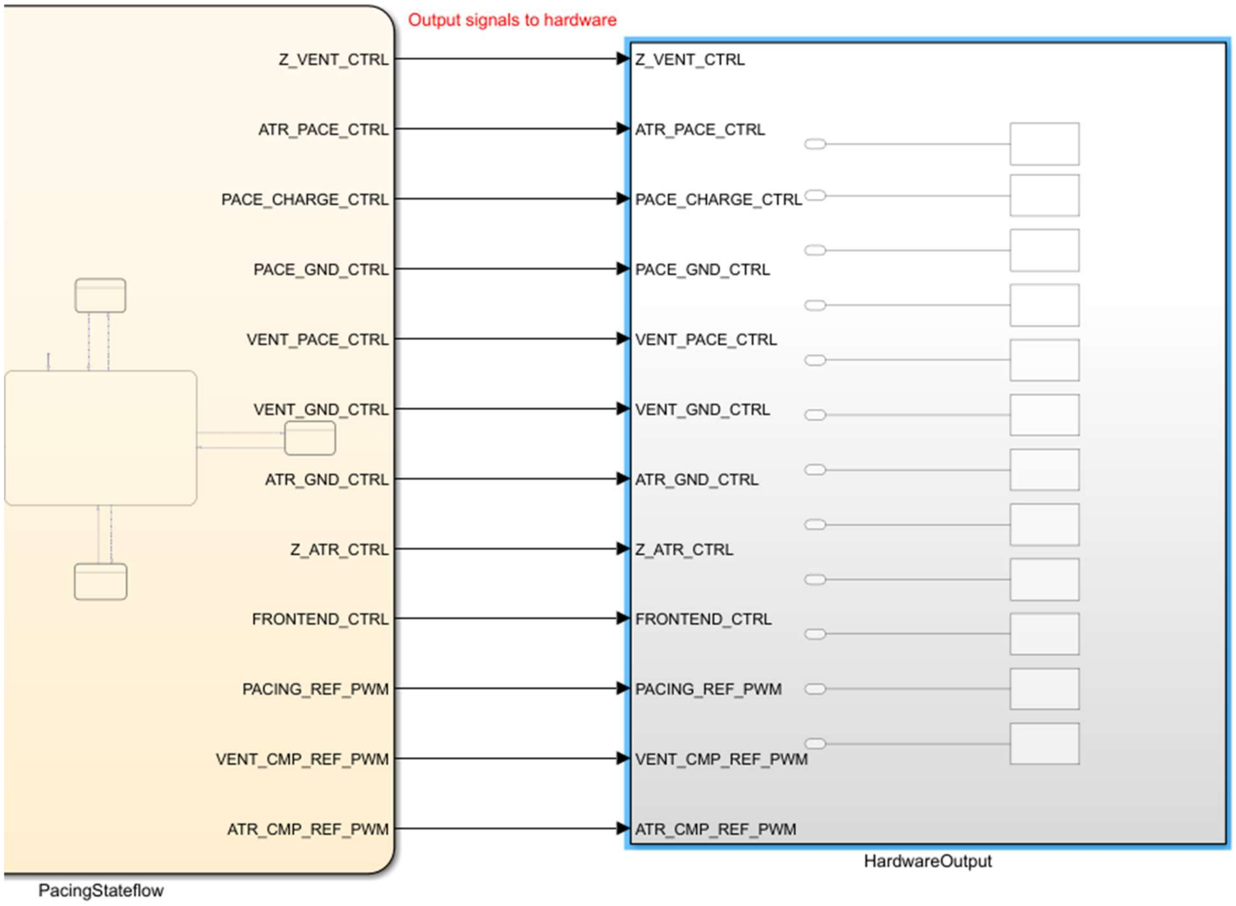
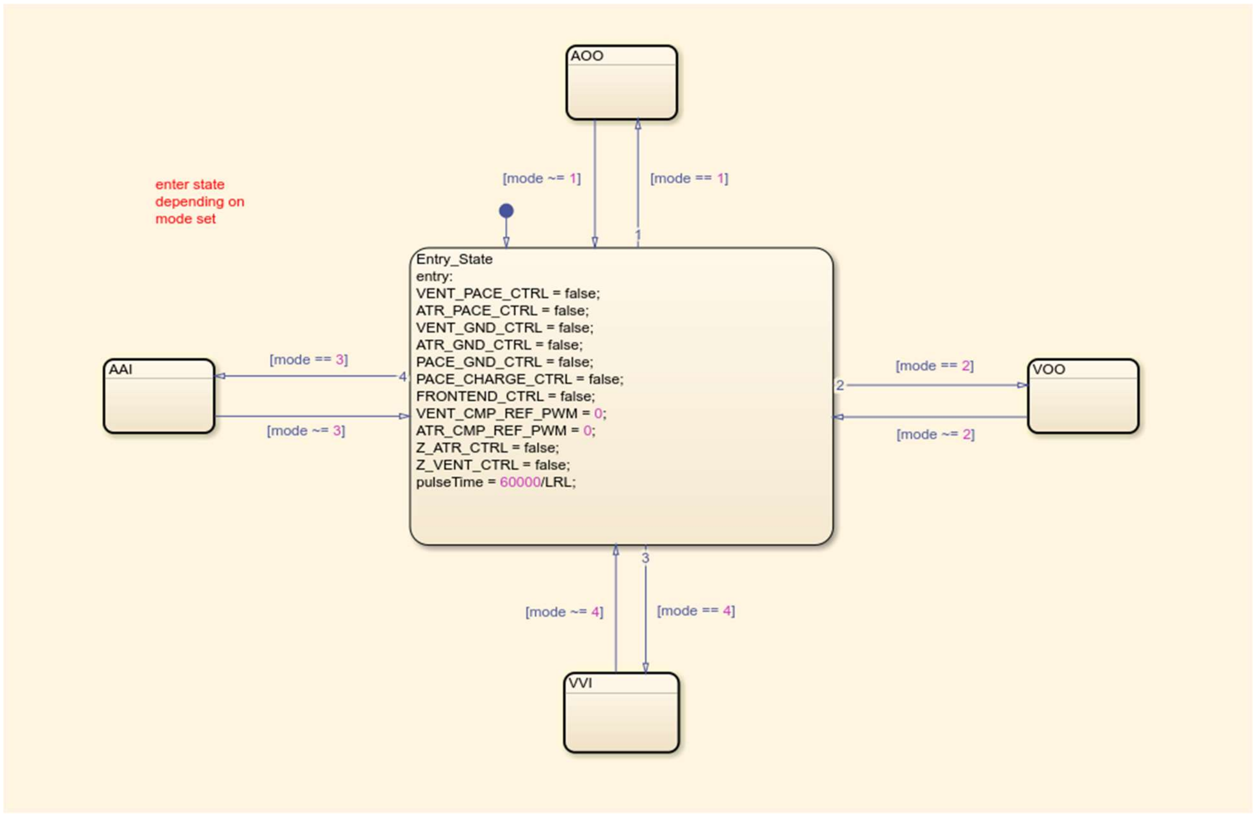


Figure 2: Output variables and hardware subsystem.

The output variables from the stateflow chart are fed into a hardware output subsystem. We implement hardware hiding using this subsystem, allowing the hardware complexities to be abstracted away so we can focus on the signals and their meaning to the Simulink code. This also allows for a modular design. The logic of the pacemaker could be changed completely and written in a new stateflow chart. However, that new logic could simply be connected to this same subsystem without issue. Because the subsystem has one specific function, this allows for it to be decoupled from the rest of the logic, while displaying cohesion by including all of the hardware outputs.

Sensing and Pacing Implementation

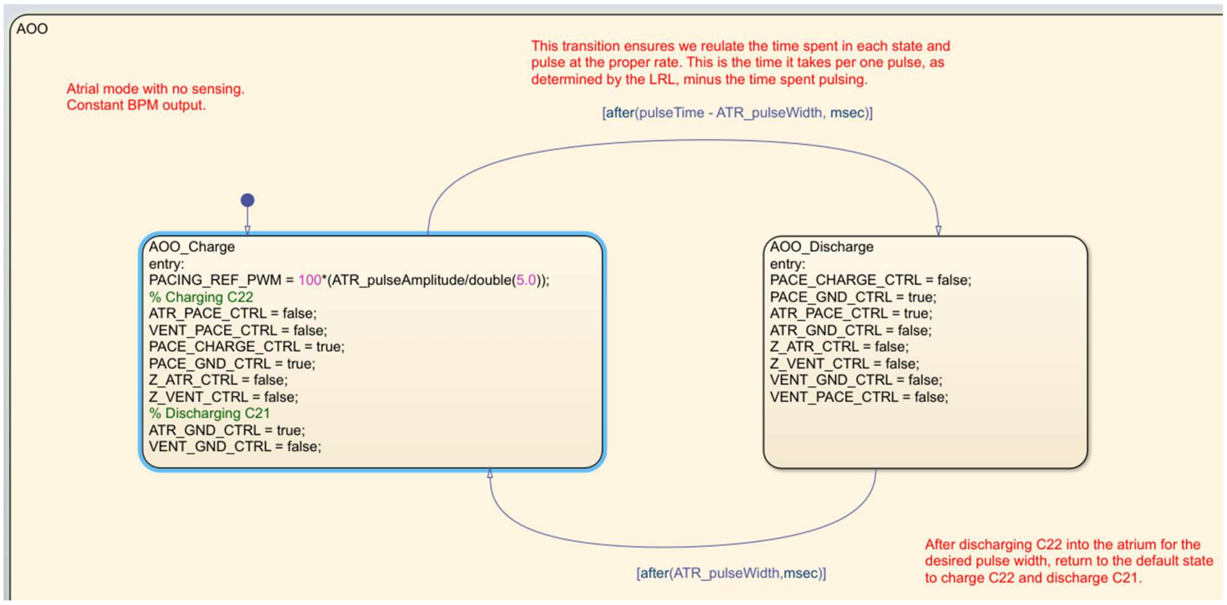
The core logic of the pacemaker is broken up into four blocks corresponding to each of the four states.



```
Entry_State
entry:
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
VENT_GND_CTRL = false;
ATR_GND_CTRL = false;
PACE_GND_CTRL = false;
PACE_CHARGE_CTRL = false;
FRONTEND_CTRL = false;
VENT_CMP_REF_PWM = 0;
ATR_CMP_REF_PWM = 0;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
pulseTime = 60000/LRL;
```

Figure 3: Top-level Siumlink chart.

You can see the behaviour of the mode input variable as an enumeration that selects which mode will be active. All variables are reset in the entry state in order to ensure that all states are starting from a clean slate and changes to outputs in one mode will not affect the operation of subsequent modes. Additionally, the *pulseTime* variable is calculated. This is 60 000 divided by the LRL (specified BPM). This division converts a BPM to the period in milliseconds for a full charge and pace cycle. The logic for each of the modes is described here.



Atrial mode with no sensing.
Constant BPM output.

This transition ensures we reulate the time spent in each state and pulse at the proper rate. This is the time it takes per one pulse, as determined by the LRL, minus the time spent pulsing.

```
[after(pulseTime - ATR_pulseWidth, msec)]
```

```
AOO_Charge
entry:
PACING_REF_PWM = 100*(ATR_pulseAmplitude/double(5.0));
% Charging C22
ATR_PACE_CTRL = false;
VENT_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
PACE_GND_CTRL = true;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
% Discharging C21
ATR_GND_CTRL = true;
VENT_GND_CTRL = false;
```

```
AOO_Discharge
entry:
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
ATR_PACE_CTRL = true;
ATR_GND_CTRL = false;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
VENT_GND_CTRL = false;
VENT_PACE_CTRL = false;
```

```
[after(ATR_pulseWidth,msec)]
```

After discharging C22 into the atrium for the desired pulse width, return to the default state to charge C22 and discharge C21.

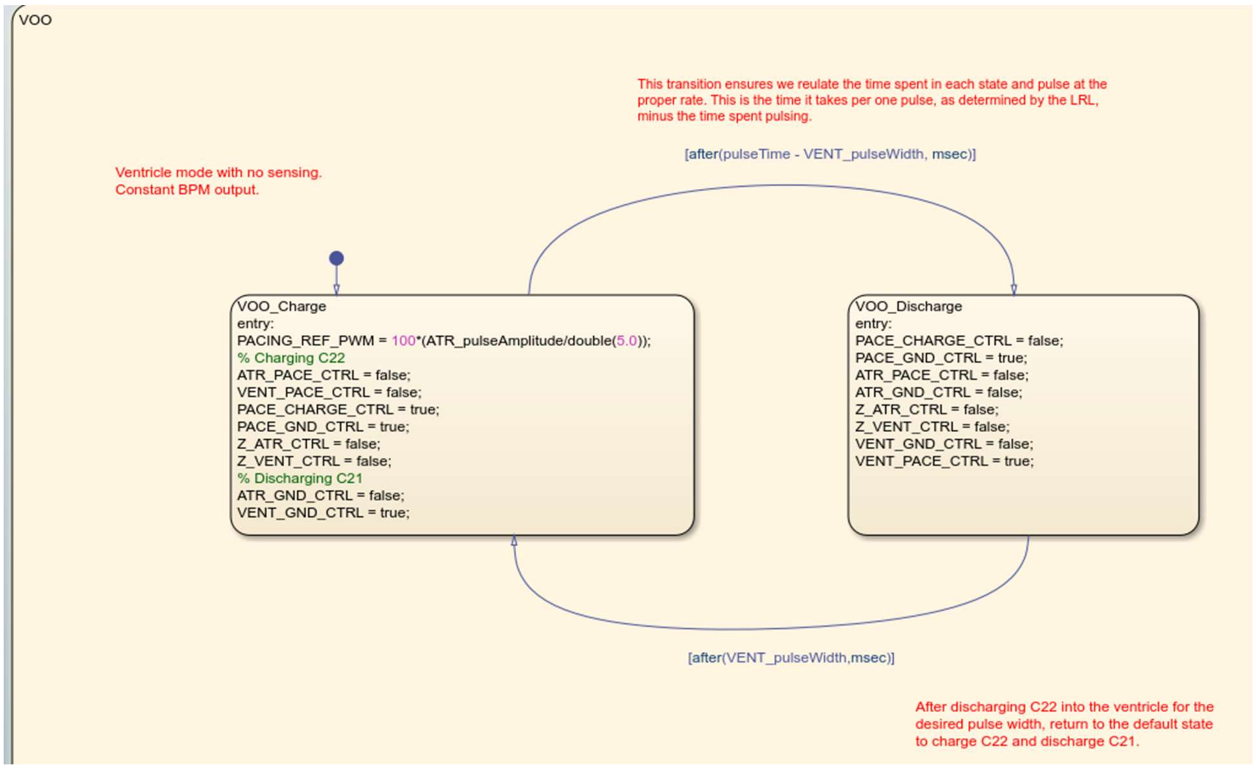
Figure 4: Mode 1 – AOO.

The AOO code is a good place to highlight the charging and discharging processes. The AOO_Charge state is responsible for applying voltage across the charging capacitor while also discharging the blocking capacitor back into the atrium to prevent charge buildup. You can see the code for these two steps split by comments within the state. The function for each of these output variables that are being set can be found in the *Hardware + Variables* section.

PACING_REF_PWM is responsible for setting the proper voltage on the charging capacitor, and therefore impacts the amplitude of the output pulses. Because the output is scaled between 0V and 5V, we set a duty cycle percentage to scale that voltage (where 100% = 5V, 50% = 2.5V, 0% = 0V, etc.).

The AOO_Discharge state discharges the charging capacitor into the atrium. The signals being set and their functions can be found in the *Hardware + Variables* section.

The state transitions are responsible for AOO delivering a constant steady pacing to the atrium. A transition from the discharge to charge state occurs after the pulse width (specified in milliseconds) has elapsed. This results in the charging capacitor discharging for the duration specified as the pulsewidth. The transition from charge to discharge occurs after a time specified as the full pacing period minus the pulsewidth has elapsed. This ensures pacing continues occurring at a constant rate specified by the pull pacing period (and indirectly the specified BPM).


VOO
[after(pulseTime - VENT_pulseWidth, msec)]
[after(VENT_pulseWidth,msec)]

This transition ensures we reulate the time spent in each state and pulse at the proper rate. This is the time it takes per one pulse, as determined by the LRL, minus the time spent pulsing.

Ventricle mode with no sensing.
Constant BPM output.

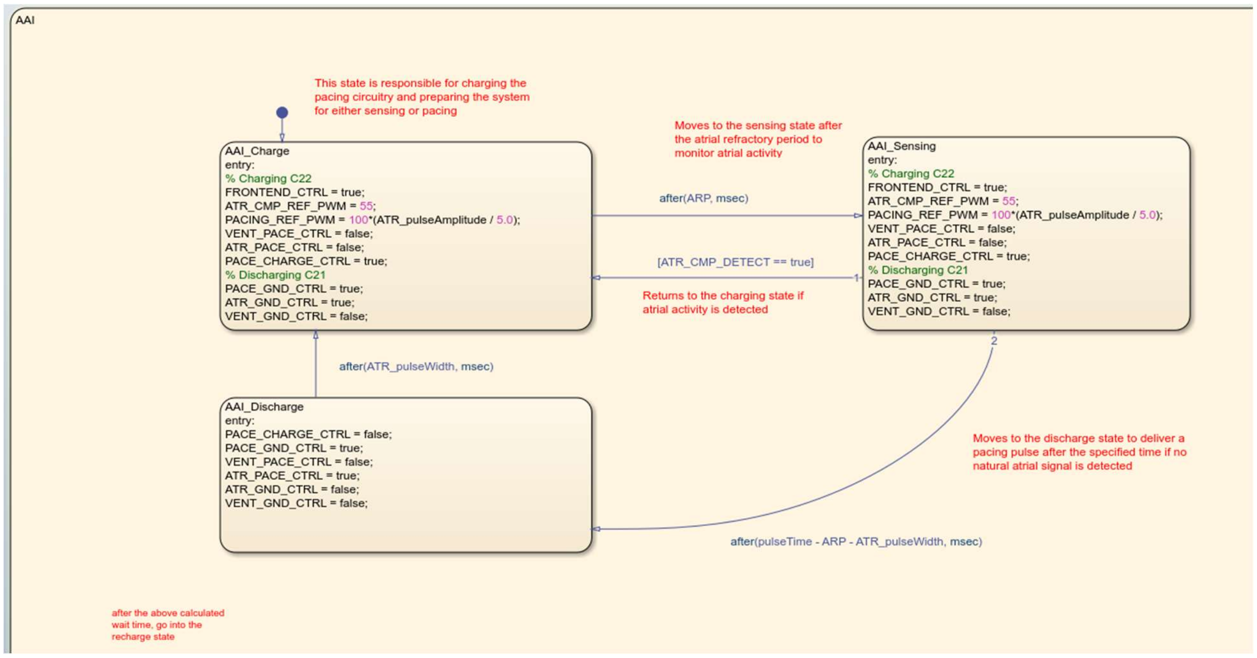
```
VOO_Charge
entry:
PACING_REF_PWM = 100*(ATR_pulseAmplitude/double(5.0));
% Charging C22
ATR_PACE_CTRL = false;
VENT_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
PACE_GND_CTRL = true;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
% Discharging C21
ATR_GND_CTRL = false;
VENT_GND_CTRL = true;
```

```
VOO_Discharge
entry:
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
ATR_PACE_CTRL = false;
ATR_GND_CTRL = false;
Z_ATR_CTRL = false;
Z_VENT_CTRL = false;
VENT_GND_CTRL = false;
VENT_PACE_CTRL = true;
```

After discharging C22 into the ventricle for the desired pulse width, return to the default state to charge C22 and discharge C21.

Figure 5: Mode 2 – VOO.

The VOO mode is identical in terms of logic to the AOO mode. The only difference is that discharging the charging capacitor and blocking capacitor both occur into the ventricle rather than the atrium.


AAI
after(ARP, msec)
[ATR_CMP_DETECT == true]
1
2
after(ATR_pulseWidth, msec)
after(pulseTime - ARP - ATR_pulseWidth, msec)

This state is responsible for charging the pacing circuitry and preparing the system for either sensing or pacing

Moves to the sensing state after the atrial refractory period to monitor atrial activity

```
AAI_Charge
entry:
% Charging C22
FRONTEND_CTRL = true;
ATR_CMP_REF_PWM = 55;
PACING_REF_PWM = 100*(ATR_pulseAmplitude / 5.0);
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
% Discharging C21
PACE_GND_CTRL = true;
ATR_GND_CTRL = true;
VENT_GND_CTRL = false;
```

```
AAI_Sensing
entry:
% Charging C22
FRONTEND_CTRL = true;
ATR_CMP_REF_PWM = 55;
PACING_REF_PWM = 100*(ATR_pulseAmplitude / 5.0);
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
% Discharging C21
PACE_GND_CTRL = true;
ATR_GND_CTRL = true;
VENT_GND_CTRL = false;
```

Returns to the charging state if atrial activity is detected

```
AAI_Discharge
entry:
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = true;
ATR_GND_CTRL = false;
VENT_GND_CTRL = false;
```

Moves to the discharge state to deliver a pacing pulse after the specified time if no natural atrial signal is detected

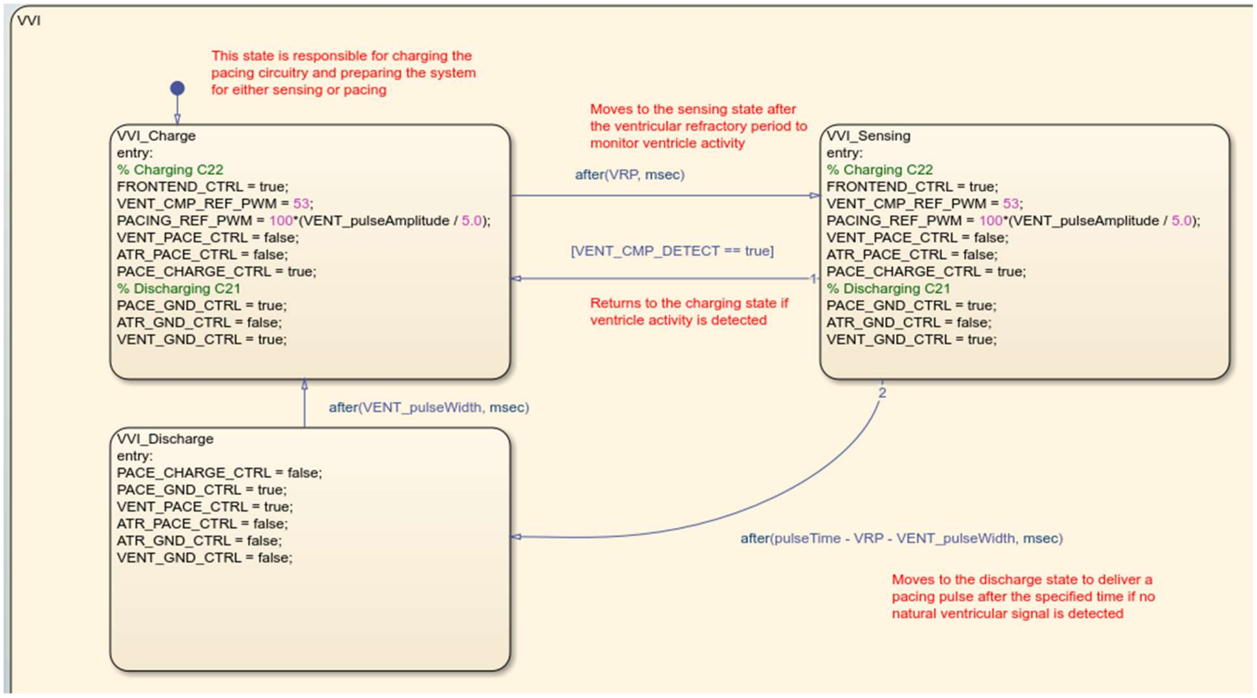
after the above calculated wait time, go into the recharge state

Figure 6: Mode 3 - AAI.

In AAI, the process of discharging the charging capacitor into the atrium is identical to that seen in AOO. The charging process is similarly identical, with the addition of setting FRONTEND_CTRL to true, effectively enabling sensing. The atrial comparison voltage was empirically set to 2.75V to ensure that false reads are mitigated (as would occur with too low of a comparison voltage) and all pulses are detected (which would not occur if with too high of a comparison voltage).

The transitions are altered from the AAI in order to account for pulse inhibition in the event of a pulse detection in the atrium. In the AAI_sensing state, the charging actions from AOO are performed (charging the charging capacitor and discharging the blocking capacitor). However, in the event of a pulse being detected in the atrium, the program transitions into the AAI_charging state. The same actions are performed in this state are the same as the AAI_sensing state, and it remains in this state for a time specified as the atrial refractory period. This prevents subsequent pulse detections from continuously inhibiting the pacing.

After the refractory period has elapsed, the AAI_sensing state is re-entered. If, instead of a pulse being detected, a time equal to the full pacing period minus the refractory period and pulse width elapses, the atrium is paced. This is done using the same logic as AOO.


VVI
This state is responsible for charging the pacing circuitry and preparing the system for either sensing or pacing
Moves to the sensing state after the ventricular refractory period to monitor ventricle activity
after(VRP, msec)
[VENT_CMP_DETECT == true]
Returns to the charging state if ventricle activity is detected
after(VENT_pulseWidth, msec)
1
2
after(pulseTime - VRP - VENT_pulseWidth, msec)
Moves to the discharge state to deliver a pacing pulse after the specified time if no natural ventricular signal is detected

```
VVI_Charge
entry:
% Charging C22
FRONTEND_CTRL = true;
VENT_CMP_REF_PWM = 53;
PACING_REF_PWM = 100*(VENT_pulseAmplitude / 5.0);
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
% Discharging C21
PACE_GND_CTRL = true;
ATR_GND_CTRL = false;
VENT_GND_CTRL = true;
```

```
VVI_Sensing
entry:
% Charging C22
FRONTEND_CTRL = true;
VENT_CMP_REF_PWM = 53;
PACING_REF_PWM = 100*(VENT_pulseAmplitude / 5.0);
VENT_PACE_CTRL = false;
ATR_PACE_CTRL = false;
PACE_CHARGE_CTRL = true;
% Discharging C21
PACE_GND_CTRL = true;
ATR_GND_CTRL = false;
VENT_GND_CTRL = true;
```

```
VVI_Discharge
entry:
PACE_CHARGE_CTRL = false;
PACE_GND_CTRL = true;
VENT_PACE_CTRL = true;
ATR_PACE_CTRL = false;
ATR_GND_CTRL = false;
VENT_GND_CTRL = false;
```

Figure 7: Mode 4 – VVI.

The VVI mode is identical in terms of logic to the AAI mode. The only difference is that discharging the charging capacitor and blocking capacitor both occur into the ventricle rather than the atrium. Additionally, the ventricular refractory period and pulse widths are used, while sensing occurs on the ventricle rather than atrium.

Iterative Design + Design Decisions

The pacemaker went through a multitude of development steps in order to reach its current state. The development of VOO and AOO was tackled first in order to confirm the functionality of the charging and discharging behaviour. While VOO was quick to come together, AOO was slow going as many hours of debugging were disregarded when it was revealed one hardware kit is faulty as the same AOO code worked properly on another hardware kit.

From there, work began on AAI and VVI. The main hurdle in putting together the sensing logic was determining whether to use 1x or 100x sensing attenuation. 100x was empirically determined as the correct choice.

It was important to maintain a high degree of modularity in the code, striving for low coupling between modules. This is why the top-level states are arranged in operating states rather than subtasks such as pacing and charging. This allows only one input to be required (the mode), while the substates take care of their own computations. As only one state is active at a time, this is not a computational burden.

Safety

This is a safety critical system. A major consideration was to ensure that the atrium or ventricle is never connected to the charging capacitor while it was charging, resulting in a direct connection to the charging PWM signal. Additionally, as we look into the future and the connection of the DCM to the pacemaker it will be vital that we perform checks to confirm the validity of parameters received from the DCM. The most important thing we can do to ensure that the system is safe is to ensure that our modes work properly, as is validated in the *Testing - Pacemaker* section.

Modules – DCM

Window (Backend)

Purpose

The Window class (module) is decomposable into two backend components: signing in (including registering and changing the password) and creating the electrogram charts. The signing-in component consists of three submodules: registering the user, signing in to the DCM, and changing the password. Registering the user involves providing a username and password, then clicking the register button, and signing in requires the user to input their credentials to log in. After signing into the DCM, a user can change their password. The electrogram charts are displayed immediately upon entering the main Pacemaker page.

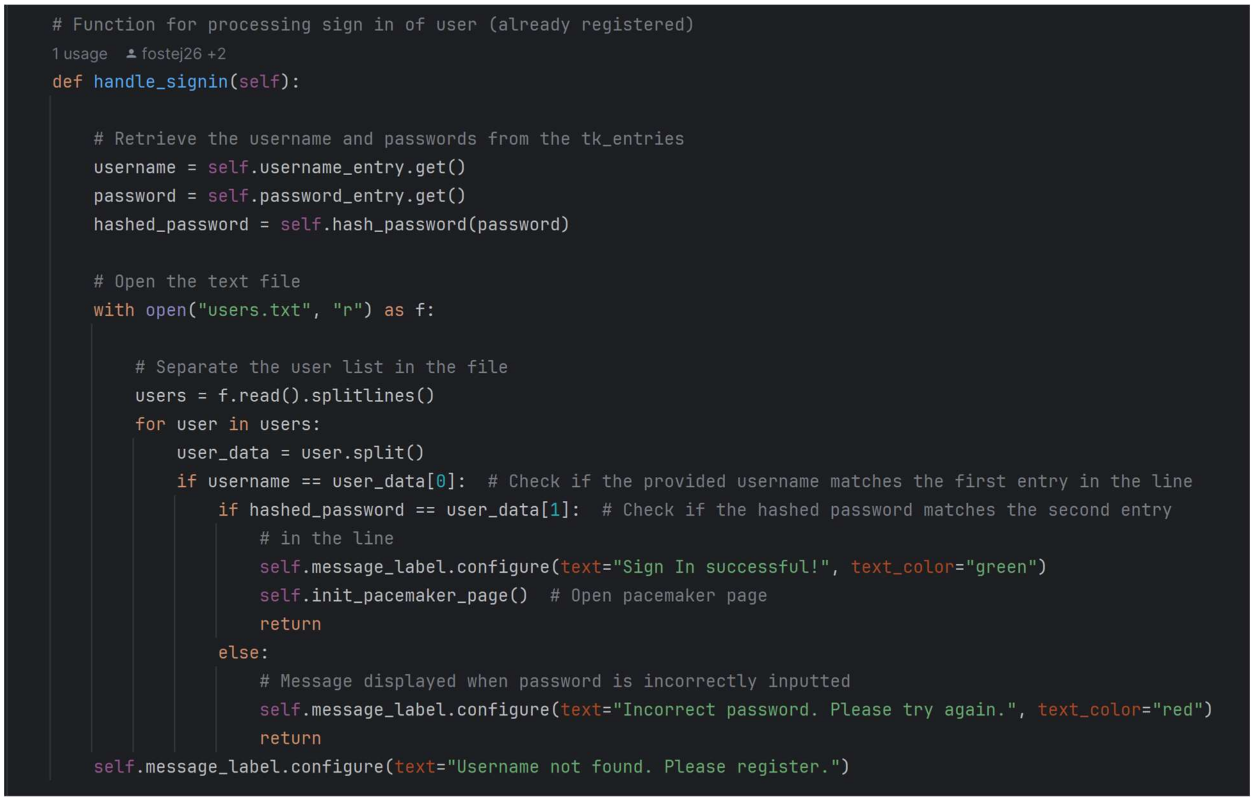
Functions

Table 3: Module Backend Methods

Function Name	**Encapsulation**	**Arguments (type)**	**Black Box Description**	**Return Type**
handle_signin	Public	None	If the entered username and password matches the one in the text file, the Pacemaker page opens. Otherwise, the program notifies the user their credentials don't work.	None
handle_register	Public	None	If the entered username doesn't exist and there are less than 10 users, the program writes the new credentials to the file.	None
hash_password	Public	password (string)	Takes the entered password from the sign-in page and hashes it.	String
handle_ change_password	Public	username (string), newpassword (string), label (tk.label)	If the entered username matches the one in the file, the newly entered password is hashed and written to the file as the user's primary password.	None
ventricular_ electrogram	Public	None	Creates a matplotlib figure and attaches it to a TK canvas.	None
atrium_ electrogram	Public	None	Creates a matplotlib figure and attaches it to a TK canvas.	None

Internal Behaviours

- handle_signin: the method works by retrieving the user input from the TKinter entry widgets and then invoking the hash password method to encrypt the password. Subsequently, the method reads the local file and splits all the lines into a list of strings. The process iterates over each line and splits the lines into another list of strings. If the entered username matches the first index of the split string and the hashed password matches the second index, the Pacemaker page opens. Otherwise, if the wrong credentials are received, the method exits. If the loop completes without finding a matching username, the user is notified (Figure 8).

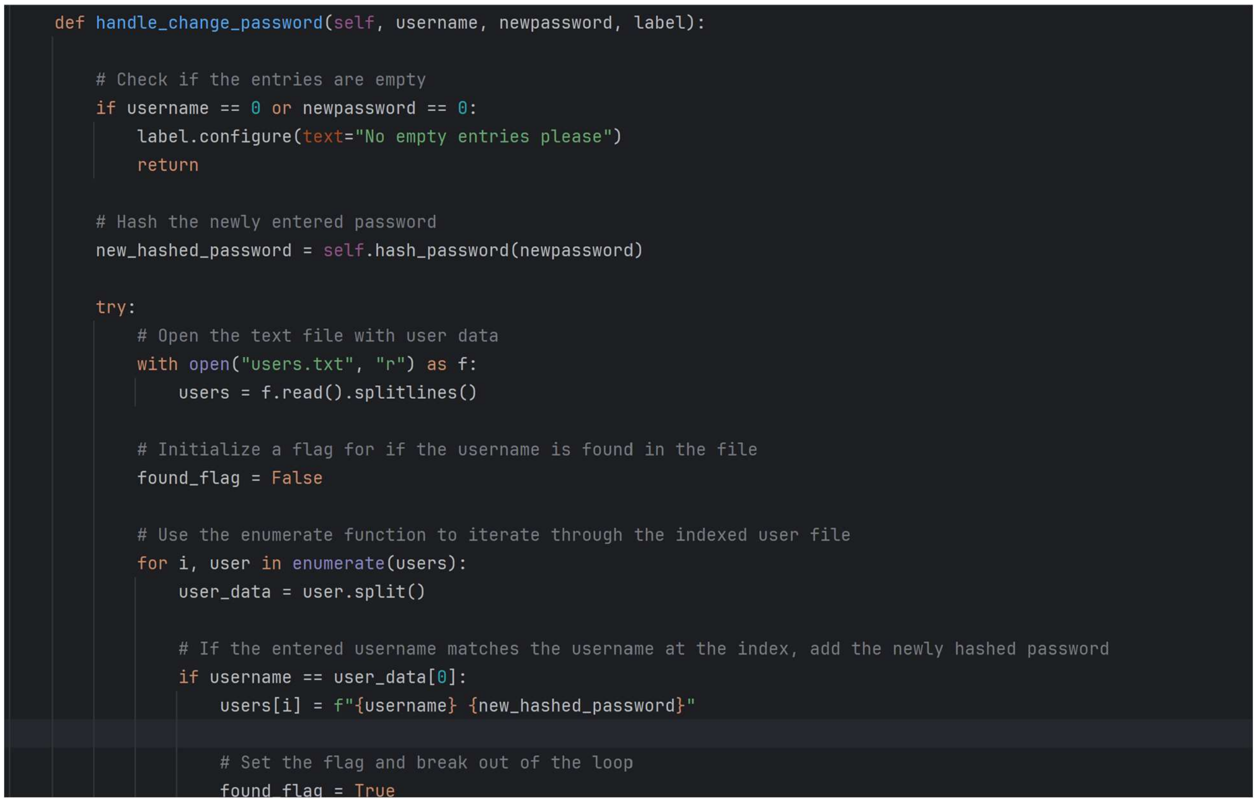


```
# Function for processing sign in of user (already registered)
1 usage  fostej26 +2
def handle_signin(self):

    # Retrieve the username and passwords from the tk_entries
    username = self.username_entry.get()
    password = self.password_entry.get()
    hashed_password = self.hash_password(password)

    # Open the text file
    with open("users.txt", "r") as f:

        # Separate the user list in the file
        users = f.read().splitlines()
        for user in users:
            user_data = user.split()
            if username == user_data[0]:  # Check if the provided username matches the first entry in the line
                if hashed_password == user_data[1]:  # Check if the hashed password matches the second entry
                    # in the line
                    self.message_label.configure(text="Sign In successful!", text_color="green")
                    self.init_pacemaker_page()  # Open pacemaker page
                    return
                else:
                    # Message displayed when password is incorrectly inputted
                    self.message_label.configure(text="Incorrect password. Please try again.", text_color="red")
                    return
    self.message_label.configure(text="Username not found. Please register.")
```

Figure 8: handle_signin function

- handle_registration: the handle_register method manages the user registration process. It starts by retrieving the user input from the Tkinter entry widget and opening the user data file. The method splits the lines into strings and checks the user count. If the number of users exceeds ten, the label notifies the user, and the process exits. If the username exists, the label notifies the user of an existing profile and exits. Otherwise, the method appends the new credentials to the file, clears the entries, and exits.
- hash_password: the method takes the password from the Tkinter entry and uses the SHA-256 hashing function to protect the credentials.
- handle_change_password: the handle-change_password method changes the user's password. It starts by checking empty entries and exiting if the conditions are correct. Next, the process hashes the new password and opens the file to split the lines. The method

initializes a flag to track whether the username has been found. Furthermore, the process iterates through each line and matches the username to the first index of the newly split string. If a username matches an index, the method updates the line, sets the flag to become true, and exits the for loop. If the flag is still false, the process notifies the user and returns. The file finally opens in write mode and appends the updated list of users to the file.



```
def handle_change_password(self, username, newpassword, label):

    # Check if the entries are empty
    if username == 0 or newpassword == 0:
        label.configure(text="No empty entries please")
        return

    # Hash the newly entered password
    new_hashed_password = self.hash_password(newpassword)

    try:
        # Open the text file with user data
        with open("users.txt", "r") as f:
            users = f.read().splitlines()

        # Initialize a flag for if the username is found in the file
        found_flag = False

        # Use the enumerate function to iterate through the indexed user file
        for i, user in enumerate(users):
            user_data = user.split()

            # If the entered username matches the username at the index, add the newly hashed password
            if username == user_data[0]:
                users[i] = f"{username} {new_hashed_password}"

                # Set the flag and break out of the loop
                found_flag = True
```

Figure 9: handle_change_password initialization

- ventricular_electrogram: the method creates an animated plot of a ventricular electrogram. The process starts by initializing a figure and axes for the plot with a specified size alongside the axis's titles. Next, the method initializes the line and sets the plot limits. The animation function updates the plot in every frame with the given equation. Finally, the process creates the animation using the matplotlib library and embeds the figure in a Tkinter frame.
- atrium_electrogram: see ventricular_electrogram.

Window (Frontend)

Purpose

The frontend of the DCM follows a 3-page design, with the pages being: Login, Pacemaker, and Change Password. The pages are compiled and transitioned between via the use of 6 methods:

Function Name	**Encapsulation**	**Arguments (type)**	**Black Box Description**	**Return Type**
Init	Public	None	Initialize common window components such as window size and name. Calls upon Init_Login_Page to begin stateflow	None
Init_Login_Page	Public	None	Compiles the logo, username and password textboxes, and sign-in/register buttons. Buttons call upon handle methods.	None
Init_Pacemaker_Page	Public	None	Compiles navbar, pacemaker connection status, pacemaker modes, and electrograms. Contains buttons that direct to Login and Change password pages.	None
Change_Password_Page	Public	None	Clears previous window. Compiles the logo, username and password textboxes, and change password button and return button. Buttons call upon handle methods.	None
Show_Pacemaker_Page	Public	None	Deletes current window and calls upon Init_Pacemaker_Page	None
Show_Login_Page	Public	None	Deletes current window and calls upon Init_Login_Page	None

Design and Considerations

- Login Page:

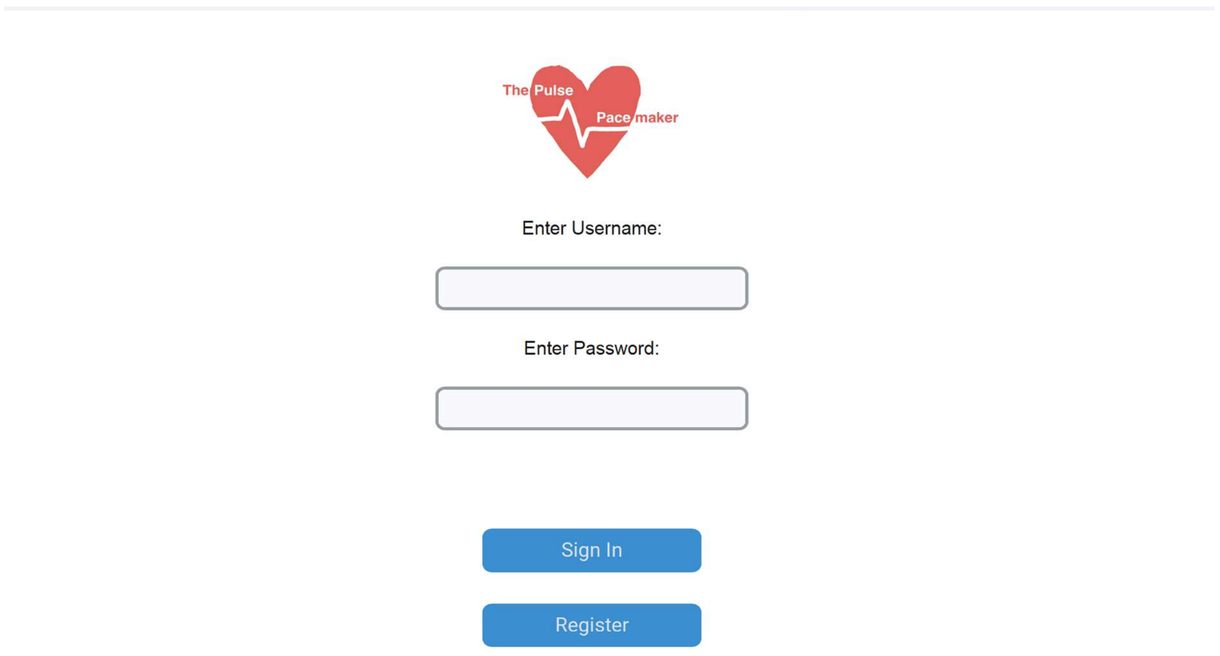


Figure 10: Login Page

The login page allows the user to enter their username and password and either sign in to their existing account or create a new one. In the event an error occurs, the message will be displayed above the sign in button. These error messages provide insight on why they were unable to login/register.

- Pacemaker Page:

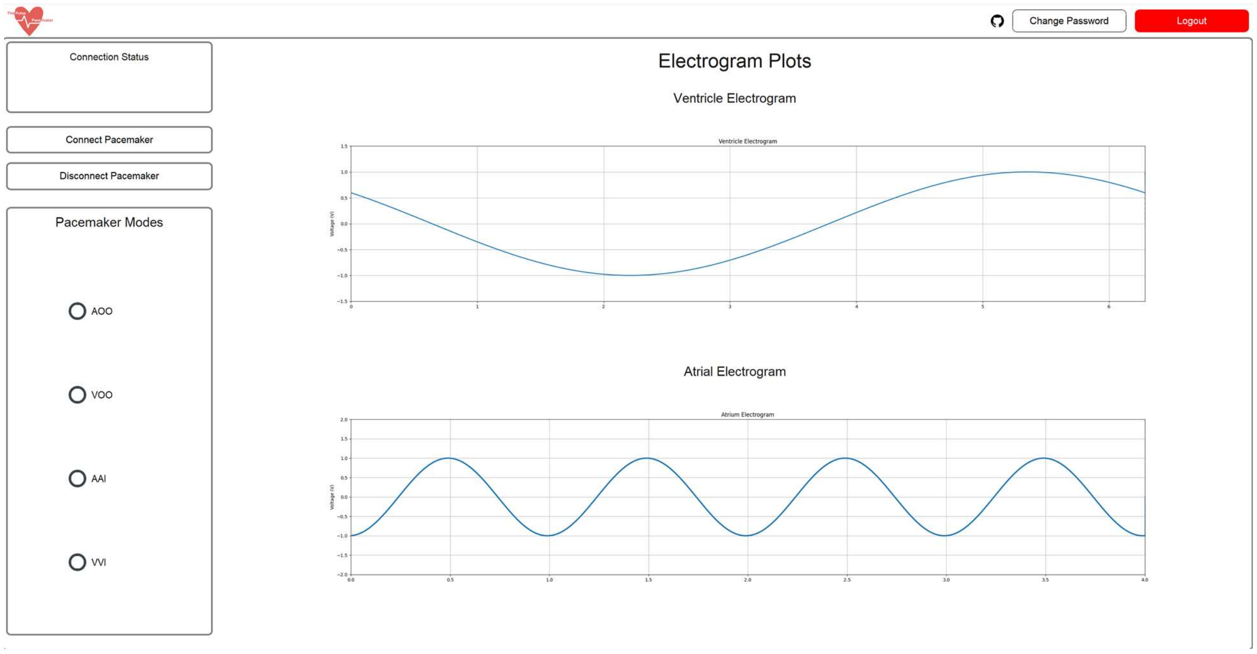


Figure 11: Pacemaker Page

The pacemaker page allows users to connect/disconnect their pacemaker, alter any parameters, and view both atrial and ventricular electrograms. Additionally, they are also able to log out and return to the sign-in page, change their password, or view the source code. A save button was added to the parameters to ensure that the correct values are applied to the pacemaker for safety reasons.

- Change Password Page:

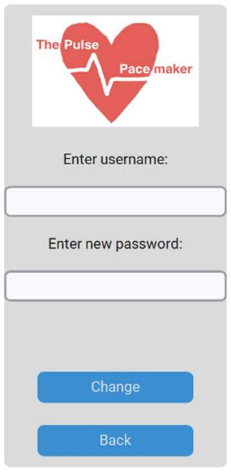


Figure 12: Change Password Page

The login page allows the user to enter their current username and a new password. In the event an error occurs, the message will be displayed above the change button. These error messages provide insight on why they were unable to login/register. After a successful password change, a message will inform the user that their password has been reset and prompt them to return to the pacemaker screen.

Testing

The DCM was tested by iterating through the following state flow:

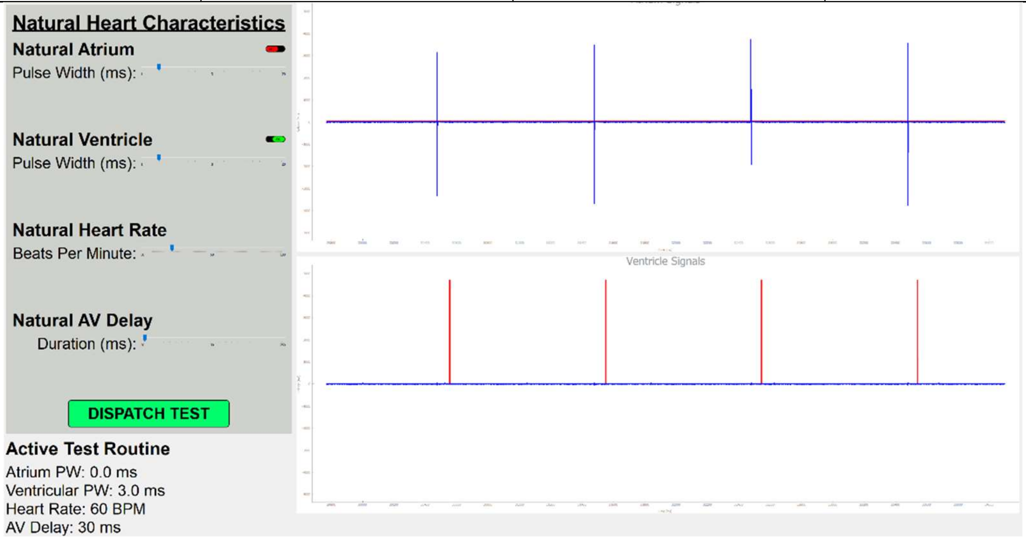
$$Register \rightarrow Log\ In \rightarrow Pacemaker\ Page \rightarrow Change\ Password \rightarrow Pacemaker\ Page \rightarrow Log\ Out \rightarrow Repeat$$

This state flow was repeated until the maximum of 10 users was reached.

Testing – Pacemaker

AOO

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
AOO	60	0.4	3

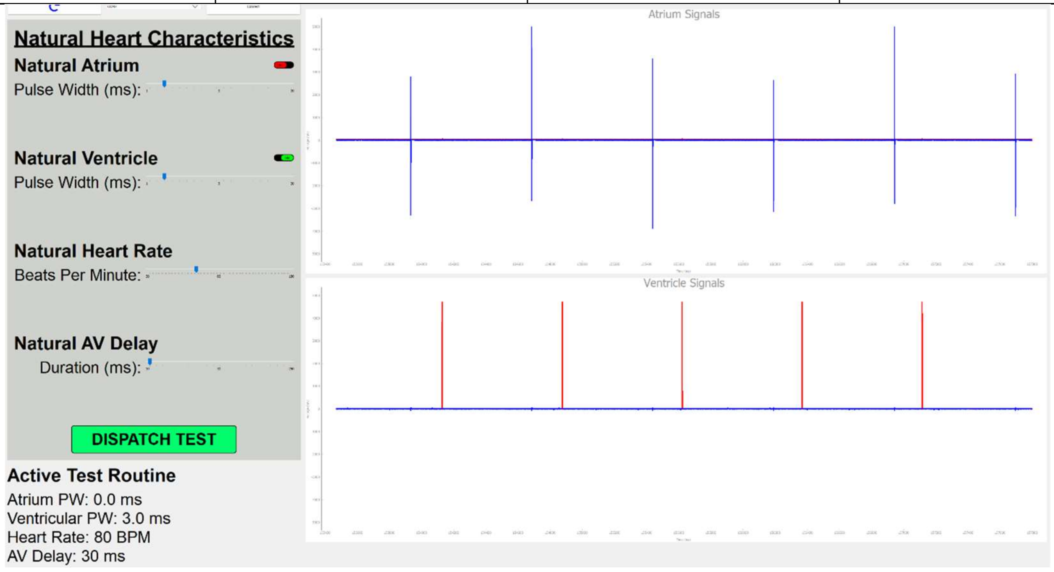


Active Test Routine
Atrium PW: 0.0 ms
Ventricular PW: 3.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 13: AOO 60 BPM Test.

In this test the desired BPM is set at 60. This causes the atrium to be paced by the pacemaker at a constant rate corresponding to 60BPM. This pacing correctly occurs regardless of any other heart activity, such as the ventricle activity seen in the lower half of the figure. The ventricle BPM is set to 60BPM to show the correct pacing rate of the atrium.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
AOO	80	0.6	4V

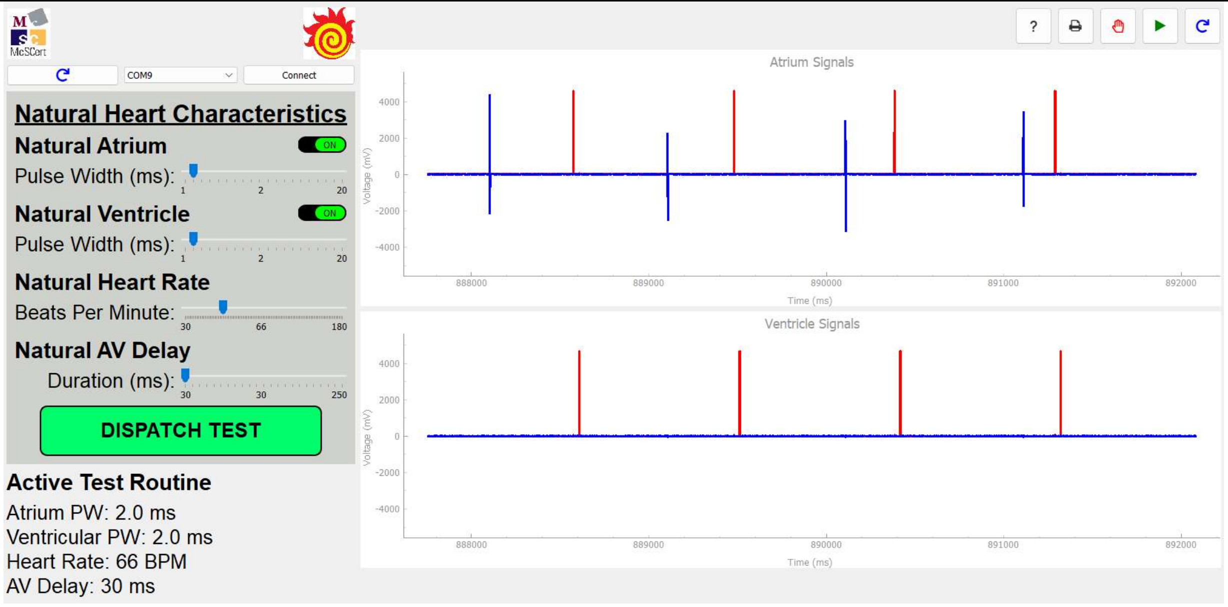


Active Test Routine
Atrium PW: 0.0 ms
Ventricular PW: 3.0 ms
Heart Rate: 80 BPM
AV Delay: 30 ms

Figure 14: AOO 80 BPM Test.

In this test the desired BPM is set at 80. This causes the atrium to be paced by the pacemaker at a constant rate corresponding to 80BPM. This pacing correctly occurs regardless of any other heart activity, such as the ventricle activity seen in the lower half of the figure. The ventricle BPM is set to 80BPM to show the correct pacing rate of the atrium.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
AOO	60	0.4	3V



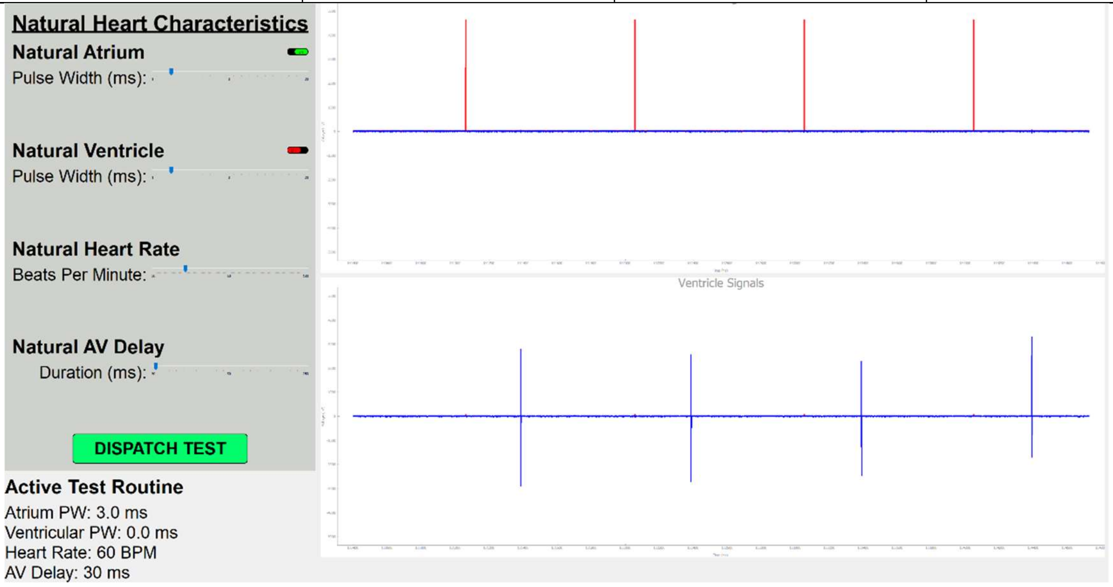
Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 66 BPM
AV Delay: 30 ms

Figure 15: AOO Test with other atrium signals.

In this test the desired BPM is set at 60. This causes the atrium to be paced by the pacemaker at a constant rate corresponding to 60BPM. This pacing correctly occurs regardless of any other heart activity, as highlighted by the fact that the correct pacing is maintained from the pacemaker (blue pulses) despite the 66 BPM natural pacing of the heart.

VOO

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
VOO	60	0.4	3

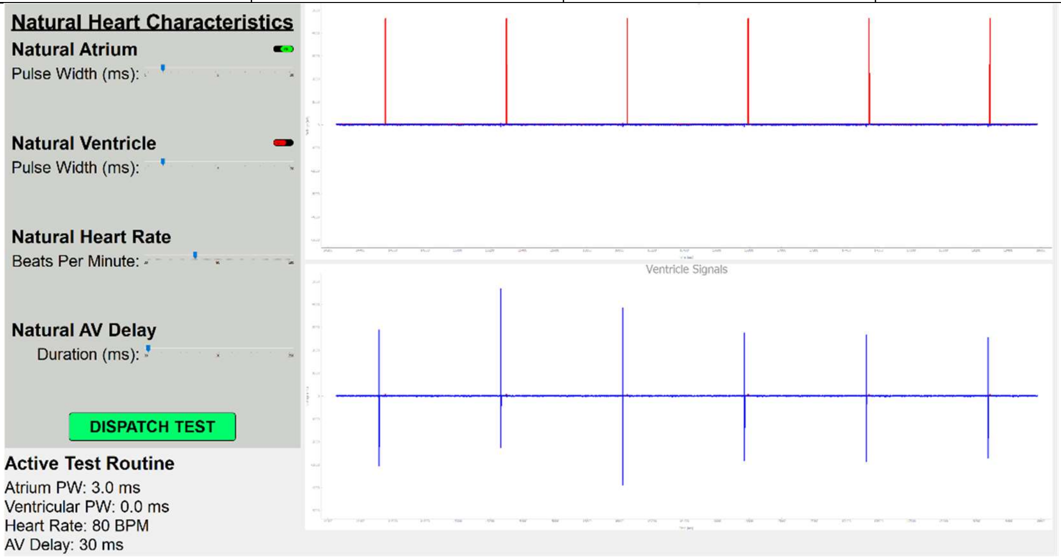


Active Test Routine
Atrium PW: 3.0 ms
Ventricular PW: 0.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 16: VOO Test 60BPM.

In this test the desired BPM is set at 60. This causes the ventricle to be paced by the pacemaker at a constant rate corresponding to 60BPM. This pacing correctly occurs regardless of any other heart activity, such as the atrium activity seen in the upper half of the figure. The atrium BPM is set to 60BPM to show the correct pacing rate of the atrium.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
VOO	80	0.6	4


Natural Heart Characteristics

Natural Atrium
Pulse Width (ms):

Natural Ventricle
Pulse Width (ms):

Natural Heart Rate
Beats Per Minute:

Natural AV Delay
Duration (ms):

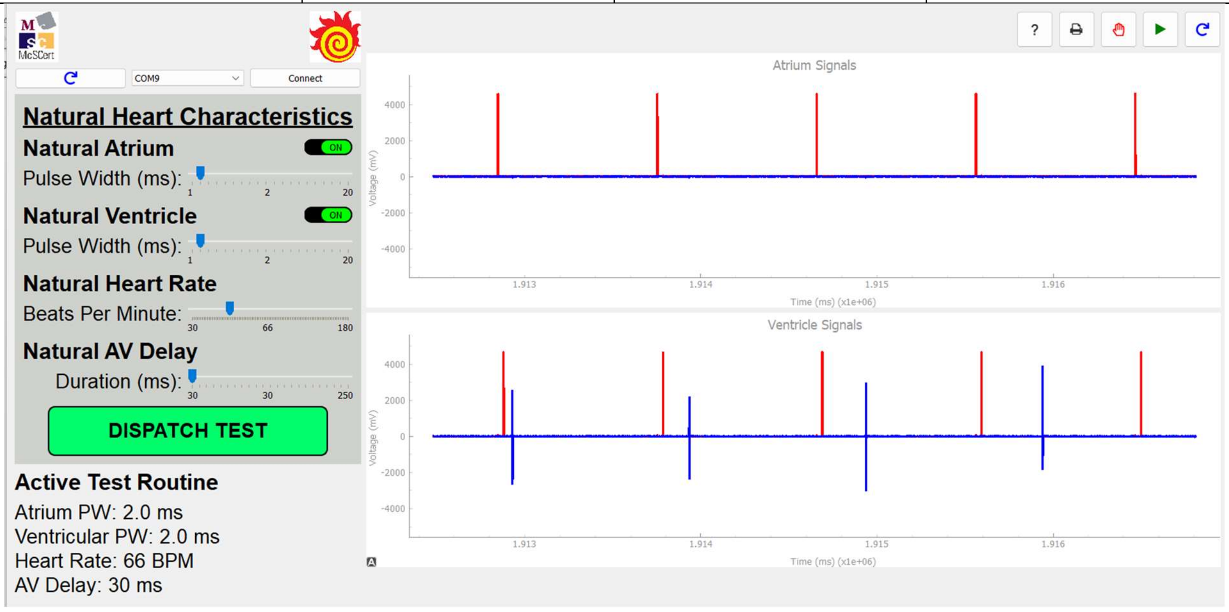
DISPATCH TEST

Active Test Routine
Atrium PW: 3.0 ms
Ventricular PW: 0.0 ms
Heart Rate: 80 BPM
AV Delay: 30 ms

Figure 17: VOO Test 80 BPM.

In this test the desired BPM is set at 60. This causes the ventricle to be paced by the pacemaker at a constant rate corresponding to 60BPM. This pacing correctly occurs regardless of any other heart activity, such as the atrium activity seen in the upper half of the figure. The atrium BPM is set to 60BPM to show the correct pacing rate of the atrium.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude (V)
VOO	60	0.4	3V



Natural Heart Characteristics

Natural Atrium

Pulse Width (ms):

Natural Ventricle

Pulse Width (ms):

Natural Heart Rate

Beats Per Minute:

Natural AV Delay

Duration (ms):

Active Test Routine

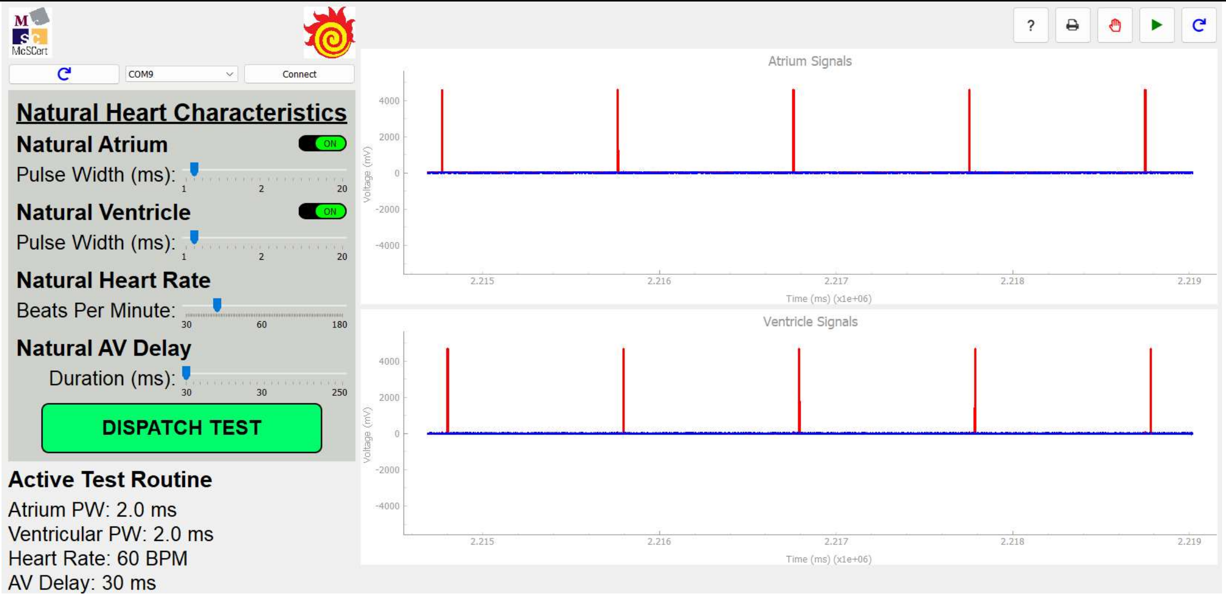
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 66 BPM
AV Delay: 30 ms

Figure 18: VOO Test with other ventricle signals.

In this test the desired BPM is set at 60. This causes the ventricle to be paced by the pacemaker at a constant rate corresponding to 60BPM. This pacing correctly occurs regardless of any other heart activity, as highlighted by the fact that the correct pacing is maintained from the pacemaker (blue pulses) despite the 66 BPM natural pacing of the heart.

AAI

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
AAI	40	0.6	3	100

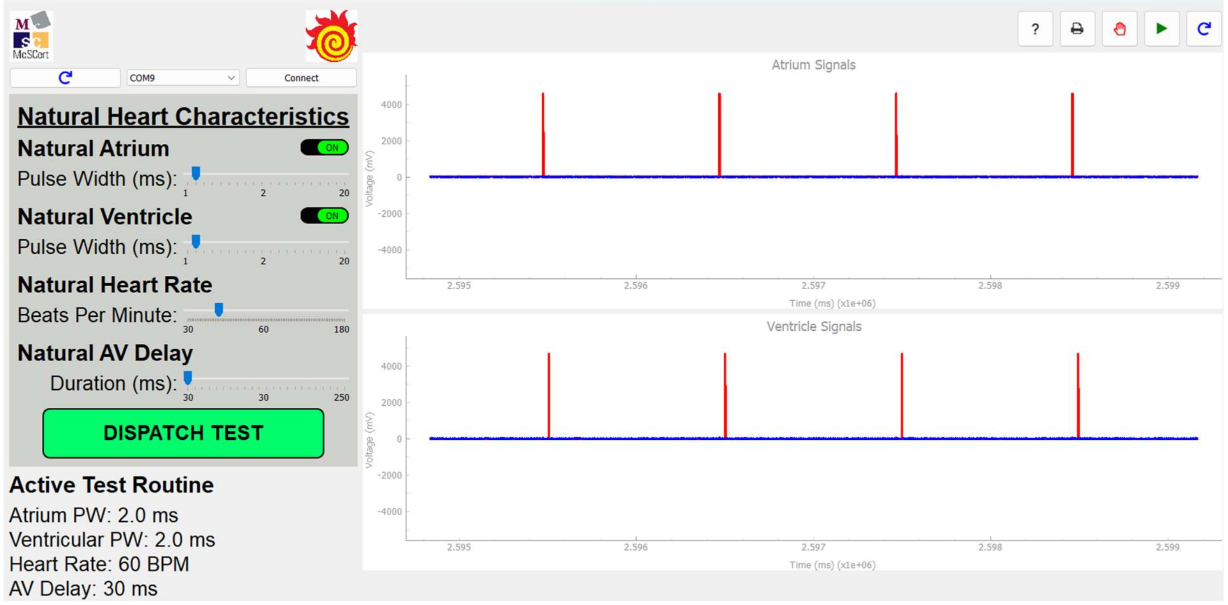


Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 19: AAI Test at 40BPM setting.

In this test the desired BPM is set to 40. However, because the natural heart rate is continuously beating at 60BPM, the output pulses are continuously inhibited. This means that the pacemaker never provides any pacing as none is required due to the natural heart rate being greater than the desired pacemaker pacing rate.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
AAI	60	0.5	3	100

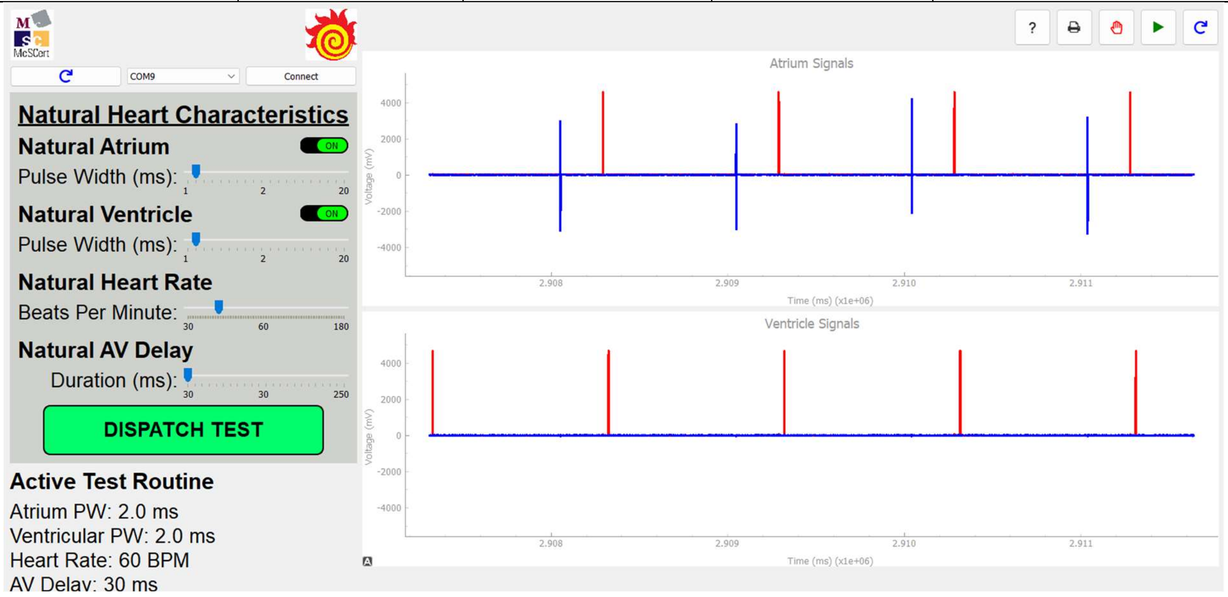


Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 20: AAI Test at 60 BPM setting.

In this test the desired BPM is set to 60. However, because the natural heart rate is continuously beating at 60BPM, the output pulses are continuously inhibited. This means that the pacemaker never provides any pacing as none is required due to the natural heart rate being equal to the desired pacemaker pacing rate.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
AAI	80	0.5	3	100



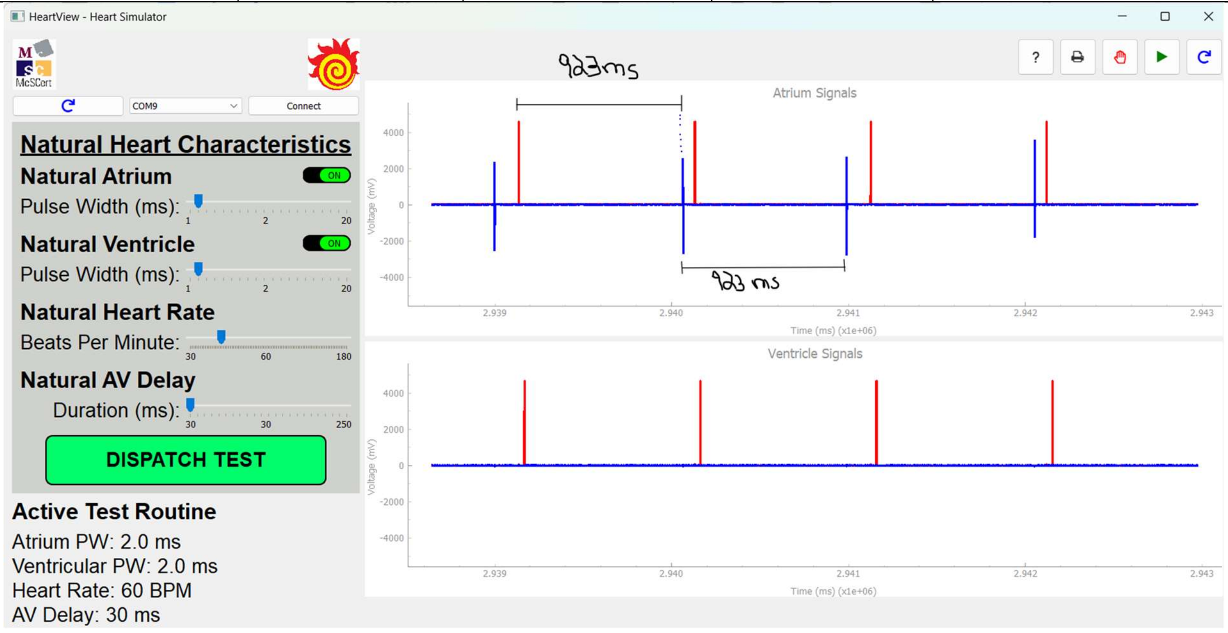
Natural Heart Characteristics
Natural Atrium: ON
Pulse Width (ms): 1 – 2 – 20
Natural Ventricle: ON
Pulse Width (ms): 1 – 2 – 20
Natural Heart Rate
Beats Per Minute: 30 – 60 – 180
Natural AV Delay
Duration (ms): 30 – 30 – 250

Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 21: AAI Test 80 BPM setting.

In this test the desired BPM is set to 80, and the natural heart is beating at 60 BPM. 80BPM pacing requires a pulse every 750ms. Using the scale at the bottom of the atrium chart (where each numerical marking denotes a second), it is clear that the heart is being paced by the pacemaker following a period of inactivity totalling 750ms after a natural pulse as desired.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
AAI	65	0.5	3	100



Natural Heart Characteristics
Natural Atrium: ON
Pulse Width (ms):
Natural Ventricle: ON
Pulse Width (ms):
Natural Heart Rate
Beats Per Minute:
Natural AV Delay
Duration (ms):

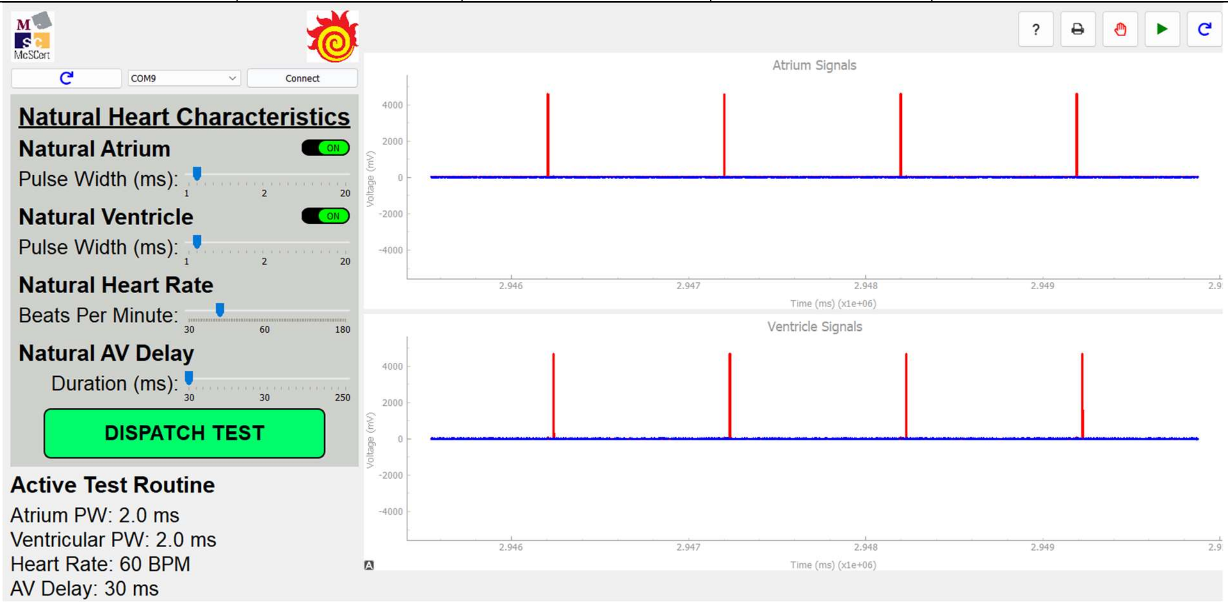
Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 22: AAI Refractory Period Test.

In this test the desired BPM is set to 65, and the natural heart is beating at 60 BPM. 65BPM pacing requires a pulse every 923ms. It is clear that there is an alternating pattern in gaps between given blue and red pulses. This occurs due to the refractory period. When the blue and red pulse are further apart, the red pulse inhibits the pacemaker, so the next blue pacemaker pulse occurs 923ms later. However, then the natural heart beats 77ms after that blue pulse. Because the refractory period is 100ms, this inhibition is ignored and the next blue pulse comes 923ms after the last blue pacemaker pulse. This repeats continuously, matching the desired refractory period design.

VVI

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
VVI	40	0.6	3	100



Natural Heart Characteristics

Natural Atrium ON

Pulse Width (ms): 1 2 20

Natural Ventricle ON

Pulse Width (ms): 1 2 20

Natural Heart Rate

Beats Per Minute: 30 60 180

Natural AV Delay

Duration (ms): 30 30 250

DISPATCH TEST

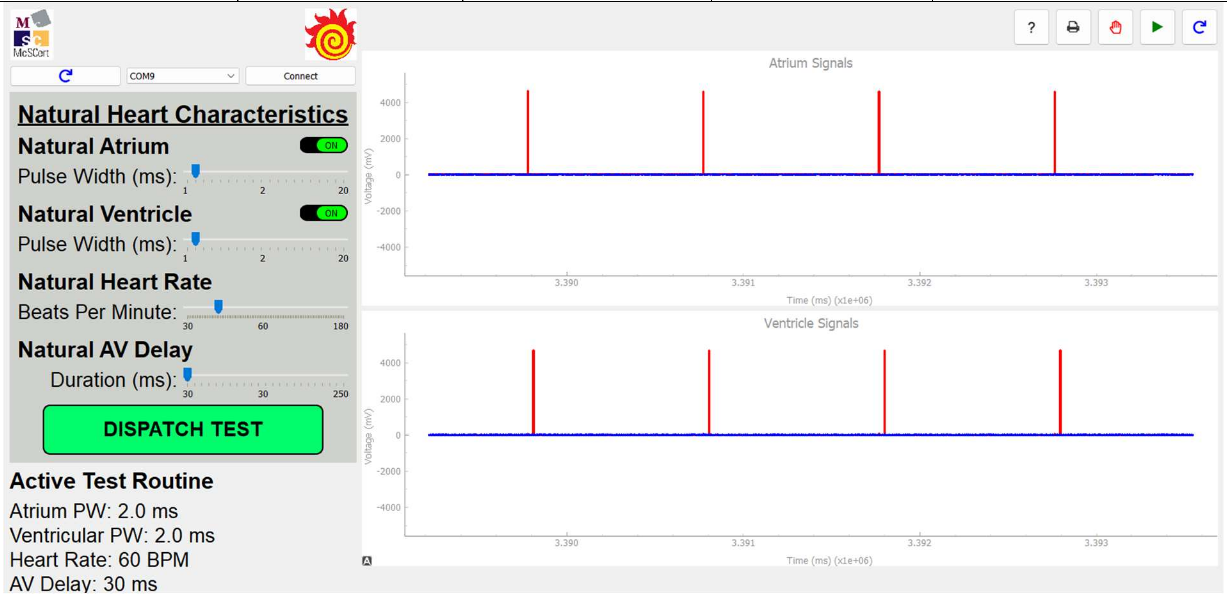
Active Test Routine

Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 23: VVI Test 40 BPM setting.

In this test the desired BPM is set to 40. However, because the natural heart rate is continuously beating at 60BPM, the output pulses are continuously inhibited. This means that the pacemaker never provides any pacing as none is required due to the natural heart rate being greater than the desired pacemaker pacing rate.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
VVI	60	0.5	3	100

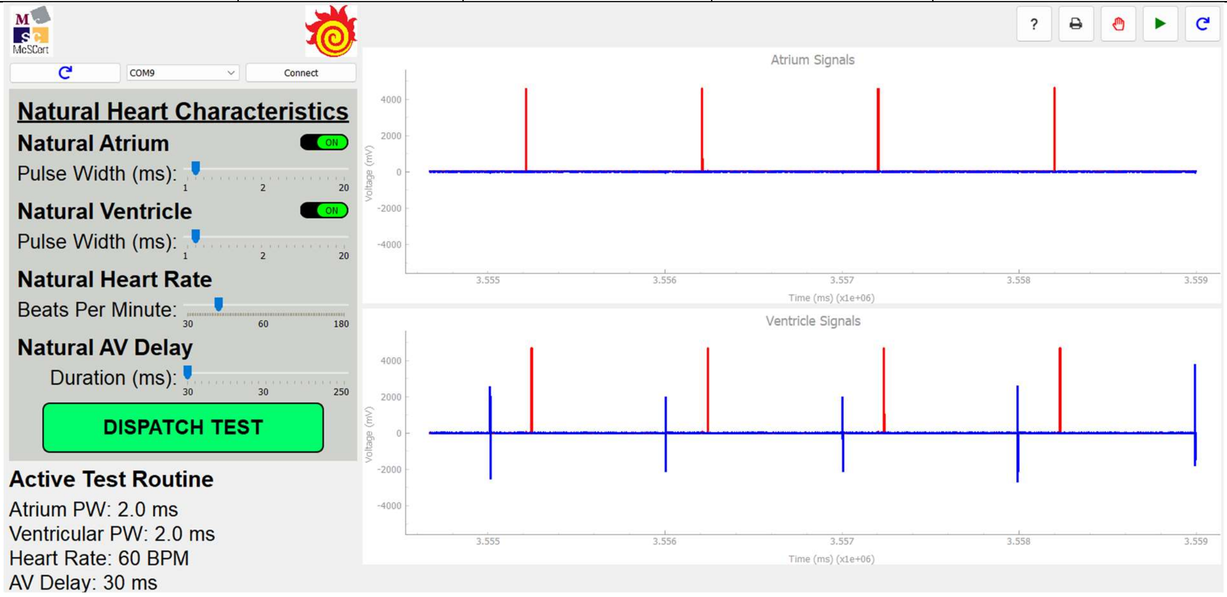


Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 24: VVI Test 60BPM setting.

In this test the desired BPM is set to 60. However, because the natural heart rate is continuously beating at 60BPM, the output pulses are continuously inhibited. This means that the pacemaker never provides any pacing as none is required due to the natural heart rate being equal to the desired pacemaker pacing rate.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
VVI	80	0.5	3	100

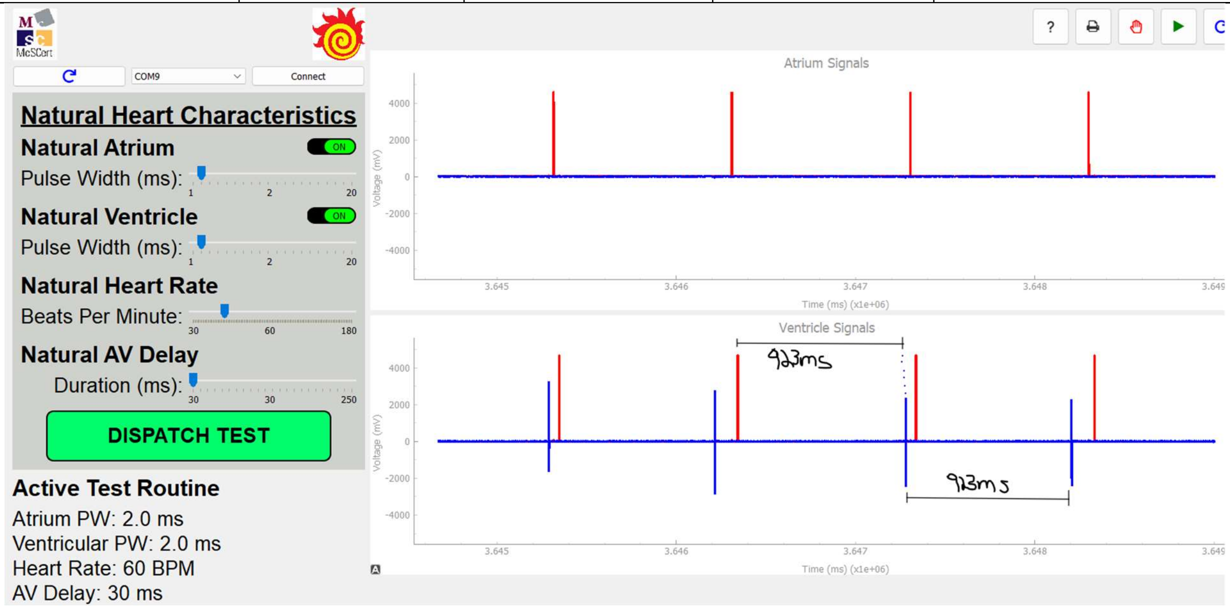


Active Test Routine
Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 25: VVI Test 80BPM setting.

In this test the desired BPM is set to 80, and the natural heart is beating at 60 BPM. 80BPM pacing requires a pulse every 750ms. Using the scale at the bottom of the ventricle chart (where each numerical marking denotes a second), it is clear that the heart is being paced by the pacemaker following a period of inactivity totalling 750ms after a natural pulse as desired.

Mode	LRL (BPM)	Pulse Width (ms)	Pulse Amplitude	Refractory Period (ms)
VVI	65	0.5	3	100



Natural Heart Characteristics

Natural Atrium ON

Pulse Width (ms):

Natural Ventricle ON

Pulse Width (ms):

Natural Heart Rate

Beats Per Minute:

Natural AV Delay

Duration (ms):

Active Test Routine

Atrium PW: 2.0 ms
Ventricular PW: 2.0 ms
Heart Rate: 60 BPM
AV Delay: 30 ms

Figure 26: VVI Refractory Period Test.

In this test the desired BPM is set to 65, and the natural heart is beating at 60 BPM. 65BPM pacing requires a pulse every 923ms. It is clear that there is an alternating pattern in gaps between given blue and red pulses. This occurs due to the refractory period. When the blue and red pulse are further apart, the red pulse inhibits the pacemaker, so the next blue pacemaker pulse occurs 923ms later. However, then the natural heart beats 77ms after that blue pulse. Because the refractory period is 100ms, this inhibition is ignored and the next blue pulse comes 923ms after the last blue pacemaker pulse. This repeats continuously, matching the desired refractory period design.

Considerations

Future Requirement Changes – DCM

The next step for the DCM is communicating serially with the Pacemaker. In real-life applications, the DCM allows physicians to change the operating mode or parameters of the device remotely and non-invasively after implantation. Therefore, since the current iteration of the DCM uses Python, one can implement the PySerial library to interface with the Pacemaker shield. The Pacemaker shield uses Simulink to induce signals, so the project's next stage requires the DCM to receive the output. Furthermore, the pacing modes stimulate the heart chambers individually in the current DCM state. Another implementation must induce signals simultaneously (dual mode).

Given the requirements are likely to change depending on the project's scale, the DCM must be modifiable. For example, the system should handle more than ten users, especially in healthcare settings. If there are more than ten individuals afflicted with bradycardia in one region, cardiologists and other practitioners would have difficulty monitoring them all if the user count was limited. Also, with heart disease becoming more common, medical professionals may call for more sophisticated treatments and, therefore extra pacing modes. The programmable data will change and there must be room for new metrics.

Affected Future Design Decisions – DCM

Currently, the DCM provisions UI space for the Pacemaker status connection but does not display any data. Once the DCM can communicate with the Pacemaker seamlessly, the frame must convey more intricate details. For example, if a new Pacemaker approaches the DCM (it has a different serial number), the DCM should prompt the user of the new device detection. Furthermore, the user data file (users.txt) will have the programmable data, such as the upper rate limit, lower rate limit, atrial amplitude, atrial pulse width, ventricular amplitude, ventricular pulse width, VRP, and ARP, extracted from the TK entries, so iterating through the text file and enumerating all lines will become a lengthy process. Instead, separating the files and printing them in different areas will improve the organization of the folders, therefore avoiding confusion for the developers/users. Even though the primary users of the DCM will be medical professionals, patients will still use the device to view (not alter) their electrograms. Therefore, one future design consideration will be accessibility options for the visually impaired and new languages other than English.

Future Requirement Changes – Pacemaker

There will be additional requirements for the pacemaker in the future. Most notably, the addition of more modes and rate adaptive pacing will be required. This means that the pacemaker must implement the ability to determine the required rate based on accelerometer data, as well as implementing additional operational mode.

Another upcoming major change is the need for input parameters to the pacemaker to be sent serially over USB by the DCM. This means that the data from the DCM will need to be able to receive this serial data, extract parameters from it, and then set those parameters to be used in the core logic of the pacemaker.

Affected Future Design Decisions – Pacemaker

The structure of the design that comes before the *PacingStateflow* chart will need to be adapted to accommodate the serial data being received from the DCM. This will likely consist of an input subsystem being implemented which will perform necessary extraction of data from the serial input data, as well as performing data validation checks (such as ensuring that all parameters are within a valid range). This makes the proposed input subsystem an immensely important piece of the software for maintaining proper operational safety of the pacemaker.

The additional modes would need to be added, meaning the addition of multiple top level states as well as values to the mode enumeration. In order to facilitate the development of some of these new modes, a new hardware input would be required that allows for input from the on-board accelerometer. Two separate input subsystems may be useful: the one mentioned above for DCM inputs as well as a subsystem for hardware inputs.