Deliverable #1 Documentation

SFWRENG 3K04

PeaceMakers

Haseeb Shaikh - shaikh22 - 400521659

Azaan Khan - Khana421 - 400399089

Hasan Asim - asimh6 - 400512182

Ayaan Hussain - 400540174

Nabil Memon - memonn - 400506502

Omar Bayari - bayario - 400523052

October 27th, 2025

Part 1

Introduction:

The purpose of this project is to design and implement a Pacemaker system through a combination of software and hardware implementation. The system uses two main components to function: the implantable pulse generator, which is the hardware aspect, and the device controller-monitor (DCM), which is the software aspect. The hardware acts like an actual pacemaker system that detects changes to a heartbeat, while the software is an application that allows users to program, monitor and adjust the pacemaker device accordingly. Deliverable 1 focuses on the development of the DCM interface and on being able to create real-time software on the hardware platform. Through user account management, parameter configuration and simulated serial communication, we were able to complete the deliverable, which brought both components to work in tandem.

The deliverable establishes the foundation of the DCM software architecture. This includes:

- A Tkinter-based graphical user interface (GUI)
- User login and registration system
- A dashboard for configuring pacemaker parameters
- Data storage for patient-specific configurations
- Serial connection functionality for identifying and verifying hardware devices

The Stateflow control software develops the foundational states of the Simulink core functions. This includes:

- Implemented a state machine dispatcher
- Completed the FSM implementation of the asynchronous pacing logic (AOO, VOO)
- Defined control logic I/O to receive programmable parameters and output direct pace control signals
- Initial FSM structure developed for AAI and VVI, including the framework for sensing and refractory period logic.

All these are part of the first deliverable scope, and will be expanded on in deliverable 2 to include more functions and address additional situations.

Requirements:

The pacemaker system shall:

1. Maintain and regulate cardiac rhythm according to permanently programmed pacing modes.
2. Provide programmable parameters to control pacing characteristics such as rate, amplitude, and pulse width.
3. Operate in a permanent pacing state for Deliverable 1.
4. It will be implemented using Simulink with Stateflow, abstracting hardware via hardware hiding (subsystems and pin mapping).
5. Provide a Device Controller Monitor (DCM) application for user interaction, mode selection, and parameter configuration.
6. Allow storage and retrieval of programmable parameters locally (in the DCM).
7. Include documentation that traces each requirement to its corresponding design elements and test cases.

The pacemaker shall implement the following modes in Simulink, as specified in **Deliverable 1**:

Mode	Chambers Paced	Chambers Sensed	Response to Sensing	Required Functionality
AOO	Atrium	None	None (asynchronous)	Deliver atrial pacing pulses at the programmed rate regardless of sensing
VOO	Ventricle	None	None (asynchronous)	Deliver ventricular pacing pulses at the programmed rate regardless of sensing
AAI	Atrium	Atrium	Inhibited	Deliver pacing only in the absence of intrinsic atrial activity
VVI	Ventricle	Ventricle	Inhibited	Deliver pacing only in the absence of intrinsic ventricular activity

General Mode Requirements:

- The pacemaker shall generate pacing pulses according to the programmable Lower Rate Limit.
- In inhibited modes (AAI, VVI), pacing should not occur if intrinsic cardiac activity is sensed during the escape interval.
- Pulse amplitude and width shall be delivered according to programmable parameters defined in Appendix A.

The DCM user interface shall:

1. Provide a graphical user interface capable of displaying text and graphics
2. Allow user interaction via buttons, dropdowns, or input fields.
3. Display all programmable parameters for review and modification
4. Visually indicate when communication with the pacemaker is active
5. Display an alert when telemetry is lost due to the device being out of range
6. Display an alert when telemetry is lost due to signal noise
7. Display a visual warning when a different pacemaker is detected than previously interrogated (e.g., new patient/device)

Additionally, per Deliverable 1:

- The DCM shall support user login and registration (max 10 users stored locally).
- The DCM shall support mode selection for AOO, VOO, AAI, and VVI modes.
- The DCM shall allow users to enter and validate programmable parameters

The system shall allow input, validation, and storage of the following programmable parameters:

Parameter	**Description**	**Range (programmable)**
Lower Rate Limit (LRL)	Minimum pacing rate	30-175 ppm
Upper Rate Limit (URL)	Maximum tracking rate	50-175 ppm
Atrial Pulse Amplitude	Atrial pacing voltage	0.5-7.0 V
Ventricular Pulse Amplitude	Ventricular pacing voltage	0.5-7.0 V
Atrial Pulse Width	Duration of pacing pulse	0.05-1.9 ms
Ventricular Pulse Width	Duration of pulse	0.05-1.9 ms
ARP (Atrial Refractory Period)	Ignore atrial sensing during refractory	150-500 ms
VRP (Ventricular Refractory Period)	Ignore ventricular sensing during refractory	150-500 ms

Design

Pacemaker Design:

Taking a birds eye view of the pacemaker architecture, it may be observed that the system is comprised of three major components. There are two subsystems that encapsulate all information pertaining to programmable parameters, hardware inputs, and hardware outputs, all of which are used by the separate permanent pacing modes of the pacemaker. Additionally, there is for each pacing mode, a finite state machine block that controls the rate of pulsing the heart. The subsystem that hides hardware pin inputs includes two inputs that sense when an intrinsic beat has occurred, either in the atria (D0) or the ventricles (D1). The subsystem that encapsulates the hardware outputs contains pins responsible for controlling the charging of the primary capacitor (D2), setting a baseline voltage against which the electrical pulses generated by intrinsic heartbeats are compared for validity (D3 for ventricles, D6 for the atria), setting a voltage that the electrical pulses will be sent with (D5), discharging the primary capacitor through the atria (D8), or ventricles (D9), and activating the sensing circuitry (D13).

Additionally, there are a multitude of pins responsible for providing grounds that allow the current to be paced through the heart to return to - namely, pins D10, D11, and D12. Within the input subsystem are also programmable parameters which a user may directly control through the DCM in future iterations of this project. The lower and upper rate limit parameters decide the bounds within which the pacemaker must keep its pulsing rate. The ventricle and atrium pulsing amplitudes decide the signal strength of the pacing that is provided to the heart. The pulse width parameters determine how long the electrical signal sent to the heart is paced for. Moreover, the refractory periods for the respective chambers of the heart set an interval of time, after either the heart is paced or sensed from, where all sensing or pacing is ignored. This filters out misleading electrical data that may indicate to the pacemaker that the heart has intrinsically pulsed, for example. The atrial and ventricular sensitivity parameters decide what sensing threshold needs to be crossed, in terms of electrical voltage, so that a heart pulse can be registered as valid. Last, the hysteresis limit parameter sets a pulsing rate, typically longer than the lower rate limit, which encourages the heart to beat on its own without pacemaker assistance. It should be noted that the subsystems are set in place so that programmable parameters can be readily modified - ideally by the DCM in the future deliverable, and that pin mappings can easily be altered. Additionally, many parameters control the signals that are sent to hardware output pins. For example, the atrial/ventricular sensitivity parameter is processed and sent to the pins responsible for setting threshold voltages for electrical sensing (pins D3 and D6).

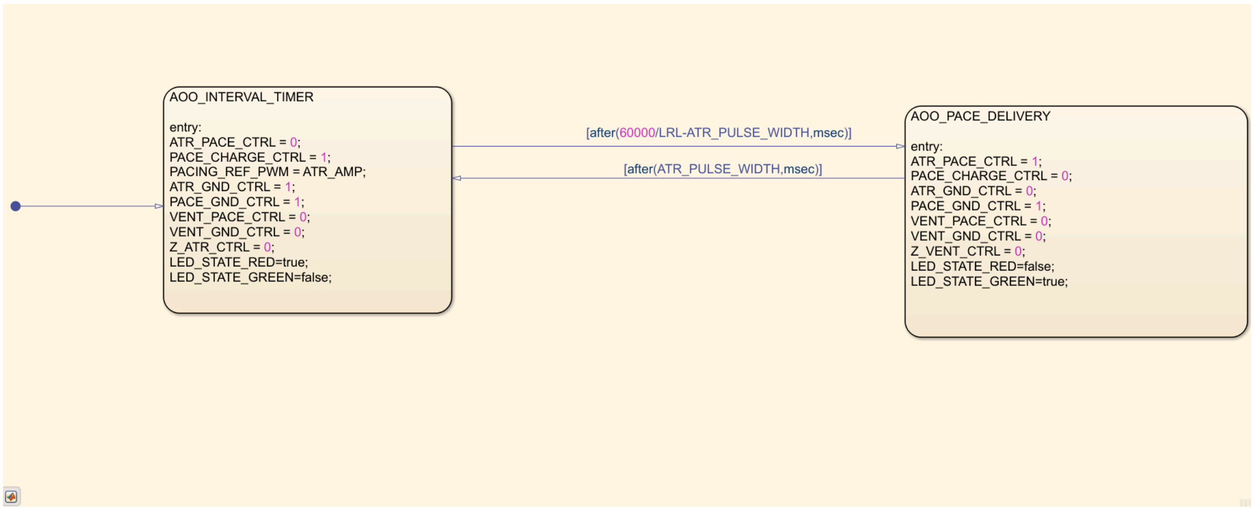
The design implementation follows all requirements extremely closely. Parameters are designed to be readily programmable by DCM communication later on. A regular cardiac rhythm is maintained, namely by the AAI and VVI modes. The AOO and VOO modes asynchronously pace the heart regardless of chamber intrinsic sensing. The pacemaker operates in a permanent state, and last, the hardware is abstracted/hidden via Simulink subsystems.

AOO:

States	**Transition Condition (to enter state)**	**Outputs**	**Notes**

AOO_INTERVAL_TIMER	- Wait atrium pulse width duration	ATR_PACE_CTRL = 0;
		PACE_CHARGE_CTRL = 1;
		PACING_REF_PWM = ATR_AMP;
		ATR_GND_CTRL = 1;
		PACE_GND_CTRL = 1;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		Z_ATR_CTRL = 0;
			- The primary C22 capacitor is charged by the setting of the pace charge control pin
			- The signal amplitude is set with parameter ATR_AMP
			- Grounds are set to complete the charging circuit

AOO_PACE_DELIVERY	- Wait for a lower rate limit pacing interval with the atrium electrical pulse width parameter subtracted	ATR_PACE_CTRL = 1;
		PACE_CHARGE_CTRL = 0;
		ATR_GND_CTRL = 0;
		PACE_GND_CTRL = 1;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		Z_VENT_CTRL = 0;
			- The atrium pace control pin is set (current is released into the heart)
			- The main ground is set active for the circuit return path

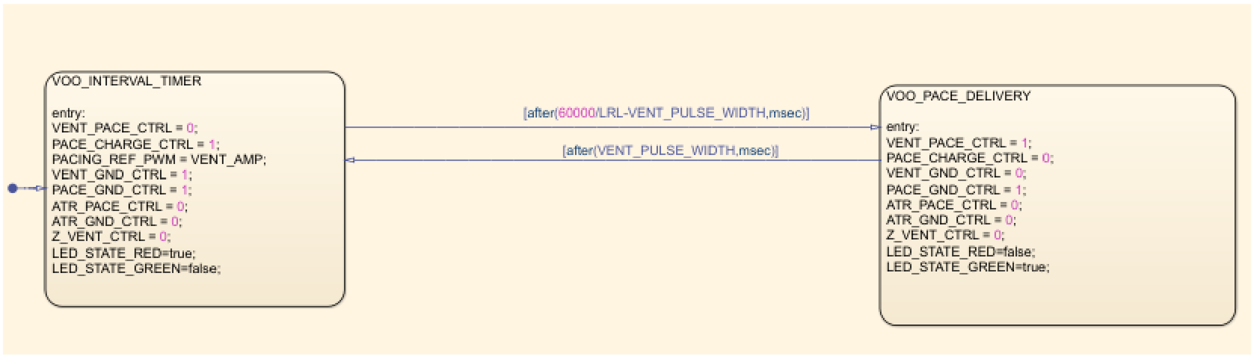

AOO_INTERVAL_TIMER
[after(60000/LRL-ATR_PULSE_WIDTH,msec)]
[after(ATR_PULSE_WIDTH,msec)]
AOO_PACE_DELIVERY

```
AOO_INTERVAL_TIMER

entry:
ATR_PACE_CTRL = 0;
PACE_CHARGE_CTRL = 1;
PACING_REF_PWM = ATR_AMP;
ATR_GND_CTRL = 1;
PACE_GND_CTRL = 1;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
Z_ATR_CTRL = 0;
LED_STATE_RED=true;
LED_STATE_GREEN=false;
```

```
AOO_PACE_DELIVERY

entry:
ATR_PACE_CTRL = 1;
PACE_CHARGE_CTRL = 0;
ATR_GND_CTRL = 0;
PACE_GND_CTRL = 1;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
Z_VENT_CTRL = 0;
LED_STATE_RED=false;
LED_STATE_GREEN=true;
```

VOO:

States	**Transition Condition (to enter state)**	**Outputs**	**Notes**
VOO_INTERVAL_TIMER	- Wait ventricle pulse width duration	VENT_PACE_CTRL = 0; PACE_CHARGE_CTRL = 1; PACING_REF_PWM = VENT_AMP; VENT_GND_CTRL = 1; PACE_GND_CTRL = 1; ATR_PACE_CTRL = 0; ATR_GND_CTRL = 0; Z_VENT_CTRL = 0;	- The primary C22 capacitor is charged by the setting of the pace charge control pin - The signal amplitude is set with parameter VENT_AMP - Grounds are set to active to complete the charging circuit
VOO_PACE_DELIVERY	- Wait for a lower rate limit pacing interval with ventricle pulse width subtracted	VENT_PACE_CTRL = 1; PACE_CHARGE_CTRL = 0; VENT_GND_CTRL = 0; PACE_GND_CTRL = 1; ATR_PACE_CTRL = 0; ATR_GND_CTRL = 0; Z_VENT_CTRL = 0;	- Venticle pace control pin is set (current is released into the heart) - Main ground is set active for the circuit return path



```
VOO_INTERVAL_TIMER

entry:
VENT_PACE_CTRL = 0;
PACE_CHARGE_CTRL = 1;
PACING_REF_PWM = VENT_AMP;
VENT_GND_CTRL = 1;
PACE_GND_CTRL = 1;
ATR_PACE_CTRL = 0;
ATR_GND_CTRL = 0;
Z_VENT_CTRL = 0;
LED_STATE_RED=true;
LED_STATE_GREEN=false;
```

```
[after(60000/LRL-VENT_PULSE_WIDTH,msec)]
[after(VENT_PULSE_WIDTH,msec)]
```

```
VOO_PACE_DELIVERY

entry:
VENT_PACE_CTRL = 1;
PACE_CHARGE_CTRL = 0;
VENT_GND_CTRL = 0;
PACE_GND_CTRL = 1;
ATR_PACE_CTRL = 0;
ATR_GND_CTRL = 0;
Z_VENT_CTRL = 0;
LED_STATE_RED=false;
LED_STATE_GREEN=true;
```

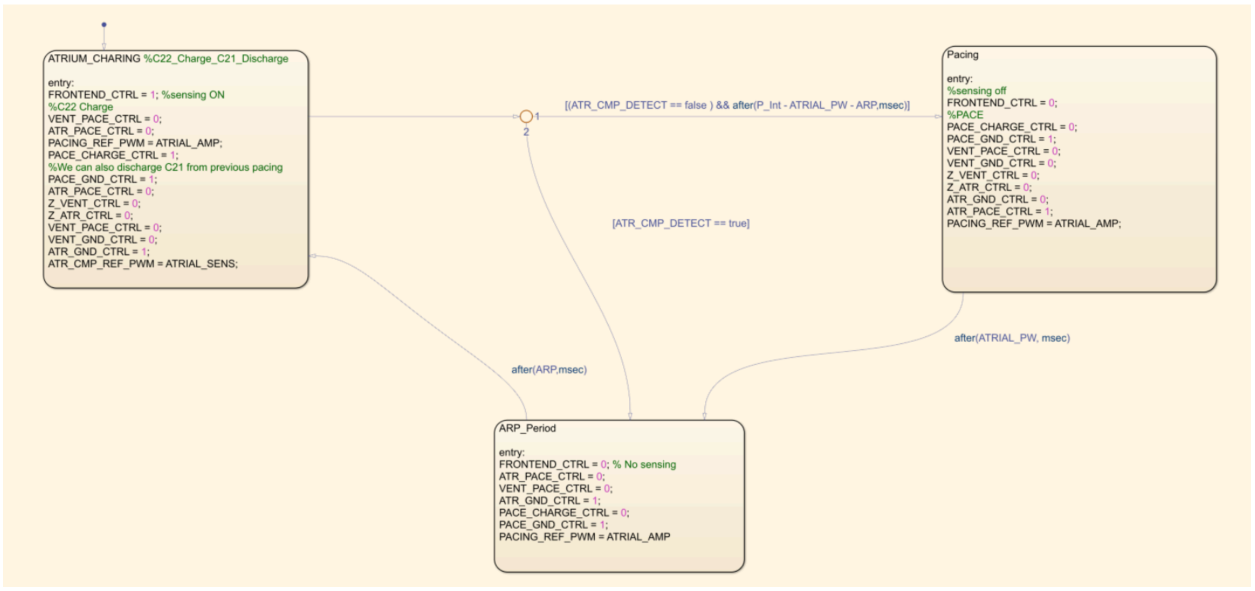
AAI:

States	**Transition Condition (to enter state)**	**Outputs**	**Notes**

C22_Charge_C21_Discharge	- Wait for the atrial refractory period	FRONTEND_CTRL = 1;
		
		VENT_PACE_CTRL = 0;
		ATR_PACE_CTRL = 0;
		PACING_REF_PWM = ATRIAL_AMP;
		PACE_CHARGE_CTRL = 1;
		
		PACE_GND_CTRL = 1;
		ATR_PACE_CTRL = 0;
		Z_VENT_CTRL = 0;
		Z_ATR_CTRL = 0;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		ATR_GND_CTRL = 1;
		ATR_CMP_REF_PWM = ATRIAL_SENS;
			- Sensing circuitry is activated via the frontend pin.
			- Electrical signal is set via the load of the atrial amplitude parameter into the pacing ref pin.
			- C22 capacitor charged via setting of pacing charge control pin
			- C21 capacitor discharged
			- Grounds are activated for circuitry completion
			- Atrial electrical sensing pulse threshold is set with the ATRIAL_SENS parameter

Pacing	- Wait for a lower rate limit pacing interval with atrial pulse width and refractory period subtracted	FRONTEND_CTRL = 0;
		
		PACE_CHARGE_CTRL = 0;
		PACE_GND_CTRL = 1;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		Z_VENT_CTRL = 0;
		Z_ATR_CTRL = 0;
		ATR_GND_CTRL = 0;
		ATR_PACE_CTRL = 1;
			- Sensing circuitry is deactivated.
			- Capacitor charging is not set.
			- Grounds are activated for pacing return paths.
			- Atrium pace control signal is set so that the current is released into the heart

ARP_Period	- Either the atrial intrinsic beat comparator is true (heart has beaten)
	- Or the atrial pulse width duration has passed
		FRONTEND_CTRL = 0;
		ATR_PACE_CTRL = 0;
		VENT_PACE_CTRL = 0;
		ATR_GND_CTRL = 1;
		PACE_CHARGE_CTRL = 0;
		PACE_GND_CTRL = 1;
		PACING_REF_PWM = ATRIAL_AMP;
			- Grounds are active for complete circuit paths
			- Sensing and pacing control signals are off/ignored
			- ARP period is waited before returning to the charging state



```
ATRIUM_CHARING %C22_Charge_C21_Discharge

entry:
FRONTEND_CTRL = 1; %sensing ON
%C22 Charge
VENT_PACE_CTRL = 0;
ATR_PACE_CTRL = 0;
PACING_REF_PWM = ATRIAL_AMP;
PACE_CHARGE_CTRL = 1;
%We can also discharge C21 from previous pacing
PACE_GND_CTRL = 1;
ATR_PACE_CTRL = 0;
Z_VENT_CTRL = 0;
Z_ATR_CTRL = 0;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
ATR_GND_CTRL = 1;
ATR_CMP_REF_PWM = ATRIAL_SENS;
```

```
[(ATR_CMP_DETECT == false ) && after(P_Int - ATRIAL_PW - ARP,msec)]
```

```
[ATR_CMP_DETECT == true]
```

```
Pacing

entry:
%sensing off
FRONTEND_CTRL = 0;
%PACE
PACE_CHARGE_CTRL = 0;
PACE_GND_CTRL = 1;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
Z_VENT_CTRL = 0;
Z_ATR_CTRL = 0;
ATR_GND_CTRL = 0;
ATR_PACE_CTRL = 1;
PACING_REF_PWM = ATRIAL_AMP;
```

```
after(ATRIAL_PW, msec)
```

```
after(ARP,msec)
```

```
ARP_Period

entry:
FRONTEND_CTRL = 0; % No sensing
ATR_PACE_CTRL = 0;
VENT_PACE_CTRL = 0;
ATR_GND_CTRL = 1;
PACE_CHARGE_CTRL = 0;
PACE_GND_CTRL = 1;
PACING_REF_PWM = ATRIAL_AMP
```

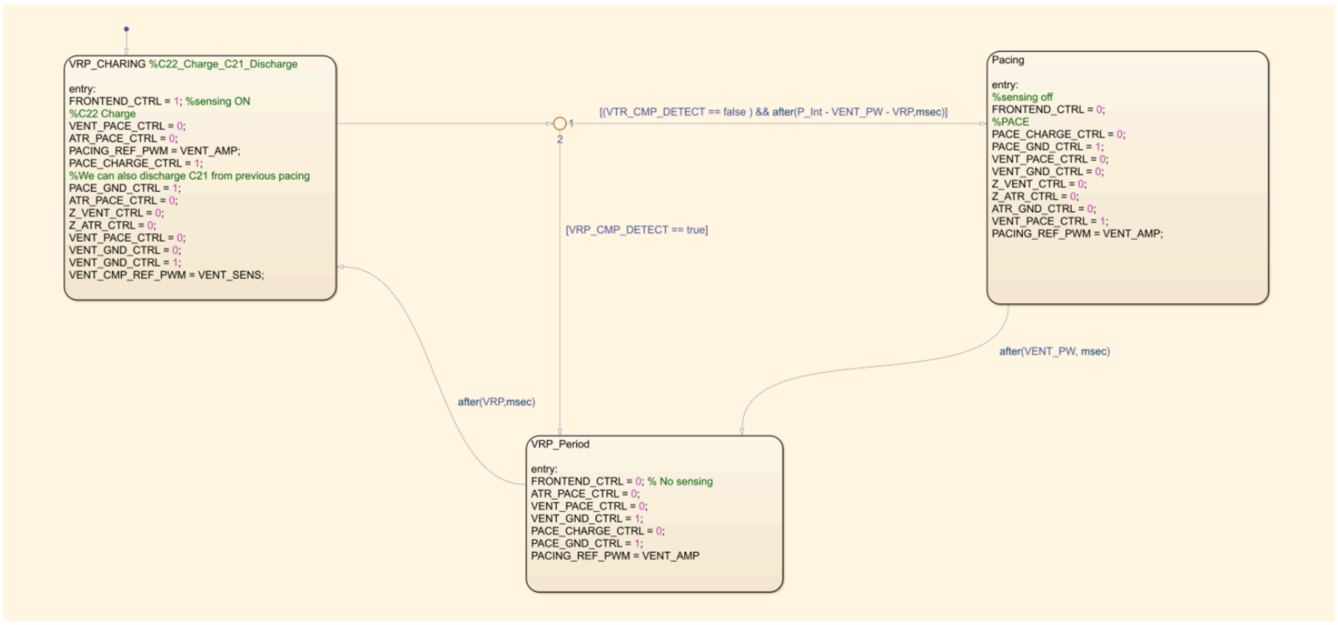
VVI:

States	**Transition Condition (to enter state)**	**Outputs**	**Notes**

C22_Charge_C21_Discharge	- Wait ventricle refractory period	FRONTEND_CTRL = 1;
		VENT_PACE_CTRL = 0;
		ATR_PACE_CTRL = 0;
		PACING_REF_PWM = VENT_AMP;
		PACE_CHARGE_CTRL = 1;
		
		PACE_GND_CTRL = 1;
		ATR_PACE_CTRL = 0;
		Z_VENT_CTRL = 0;
		Z_ATR_CTRL = 0;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		ATR_GND_CTRL = 1;
		ATR_CMP_REF_PWM = ATRIAL_SENS;
			- Sensing circuitry is activated via the frontend pin.
			- Electrical signal is set via the load of the ventricle amplitude parameter into the pacing ref pin.
			- C22 capacitor charged via setting of pacing charge control pin
			- C21 capacitor discharged
			- Grounds are activated for circuitry completion
			- Ventricle electrical sensing pulse threshold is set with the VENTRICLE_SENS parameter

Pacing	- Wait for a lower rate limit pacing interval with ventricle pulse width and refractory period subtracted	FRONTEND_CTRL = 0;
		PACE_CHARGE_CTRL = 0;
		PACE_GND_CTRL = 1;
		VENT_PACE_CTRL = 0;
		VENT_GND_CTRL = 0;
		Z_VENT_CTRL = 0;
		Z_ATR_CTRL = 0;
		ATR_GND_CTRL = 0;
		ATR_PACE_CTRL = 1;
		PACING_REF_PWM = ATRIAL_AMP;
			- Sensing circuitry is deactivated.
			- Capacitor charging is not set.
			- Grounds are activated for pacing return paths.
			- Ventricle pace control signal is set so that the current is released into the heart

VRP_Period	- Either the ventricle intrinsic beat comparator is true (heart has beaten)
	- Or the ventricle pulse width duration has passed
		FRONTEND_CTRL = 0;
		ATR_PACE_CTRL = 0;
		VENT_PACE_CTRL = 0;
		ATR_GND_CTRL = 1;
		PACE_CHARGE_CTRL = 0;
		PACE_GND_CTRL = 1;
		PACING_REF_PWM = ATRIAL_AMP
			- Grounds are active for complete circuit paths
			- Sensing and pacing control signals are off/ignored
			- ARP period is waited before returning to the charging state


```
VRP_CHARING %C22_Charge_C21_Discharge

entry:
FRONTEND_CTRL = 1; %sensing ON
%C22 Charge
VENT_PACE_CTRL = 0;
ATR_PACE_CTRL = 0;
PACING_REF_PWM = VENT_AMP;
PACE_CHARGE_CTRL = 1;
%We can also discharge C21 from previous pacing
PACE_GND_CTRL = 1;
ATR_PACE_CTRL = 0;
Z_VENT_CTRL = 0;
Z_ATR_CTRL = 0;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
VENT_GND_CTRL = 1;
VENT_CMP_REF_PWM = VENT_SENS;
```

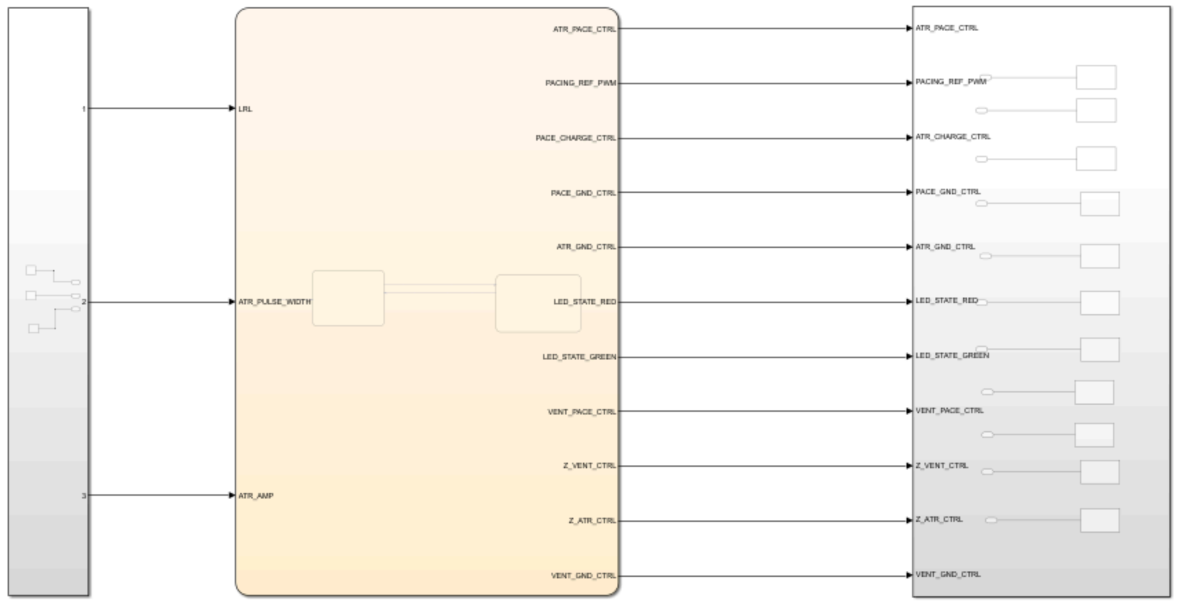
```
Pacing

entry:
%sensing off
FRONTEND_CTRL = 0;
%PACE
PACE_CHARGE_CTRL = 0;
PACE_GND_CTRL = 1;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 0;
Z_VENT_CTRL = 0;
Z_ATR_CTRL = 0;
ATR_GND_CTRL = 0;
VENT_PACE_CTRL = 1;
PACING_REF_PWM = VENT_AMP;
```

```
VRP_Period

entry:
FRONTEND_CTRL = 0; % No sensing
ATR_PACE_CTRL = 0;
VENT_PACE_CTRL = 0;
VENT_GND_CTRL = 1;
PACE_CHARGE_CTRL = 0;
PACE_GND_CTRL = 1;
PACING_REF_PWM = VENT_AMP
```

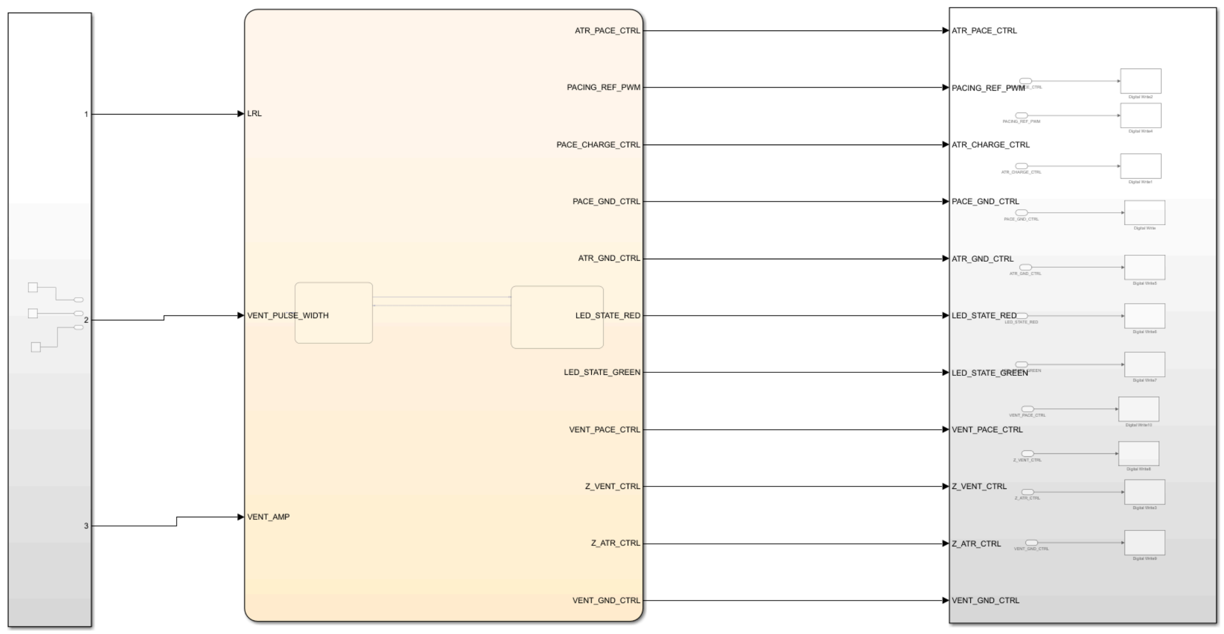
1
2
[(VTR_CMP_DETECT == false) && after(P_Int - VENT_PW - VRP,msec)]
[VRP_CMP_DETECT == true]
after(VENT_PW, msec)
after(VRP,msec)

Simulink Diagrams:

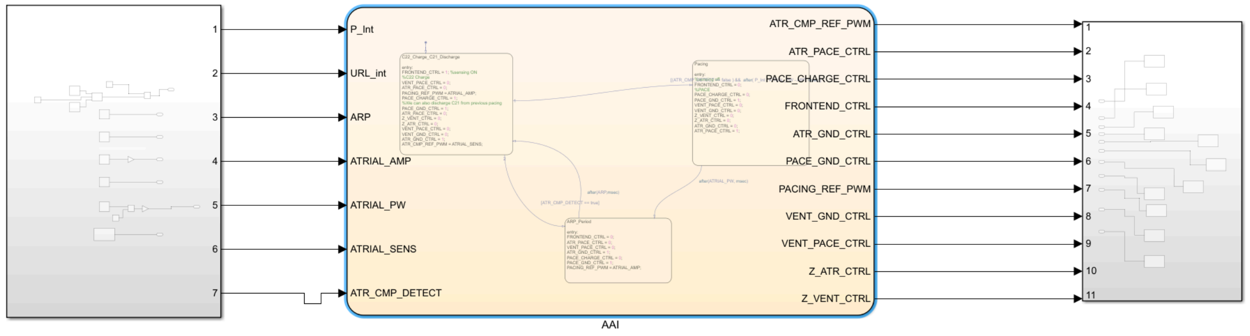
AOO



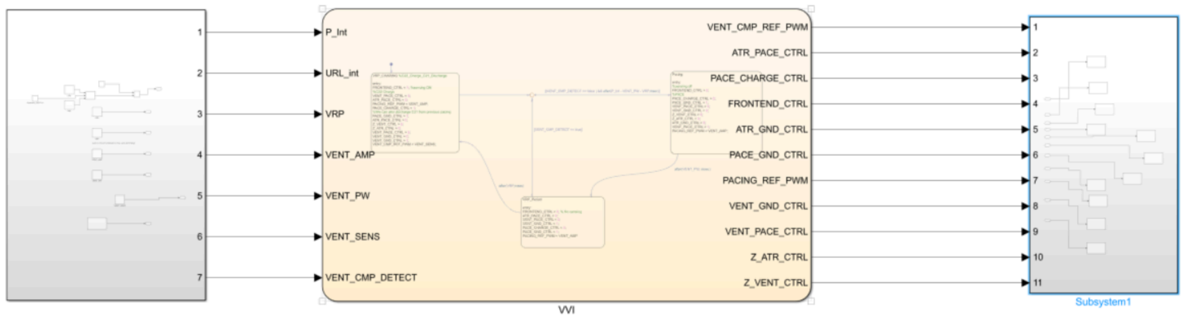
VOO



AAI



VVI

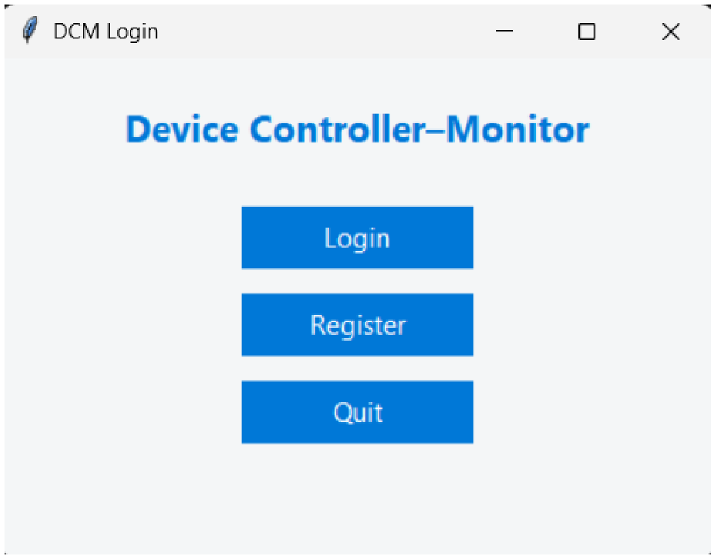


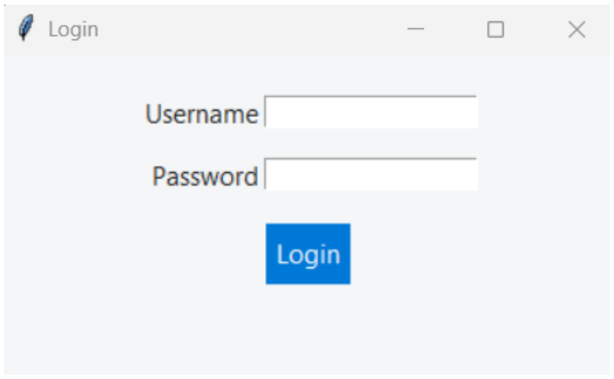
DCM Software:

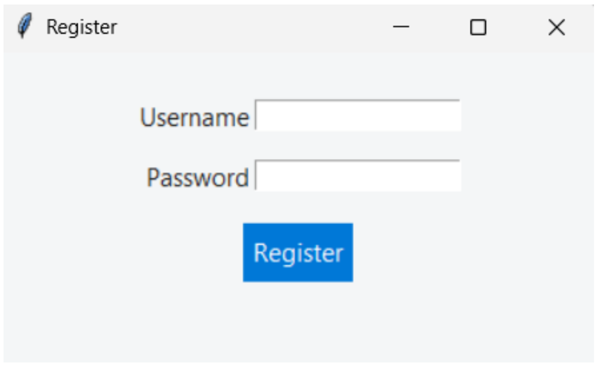
The DCM software application is a Python-based interface used by the operator to configure and communicate with the pacemaker. The interface is made up of three layers: a Tkinter GUI, a controller module that works with the parameters, and a serial communication aspect that exchanges data with the hardware. The programmable parameters include rate limits, pulse amplitudes, pulse widths, and refractory periods. The inputs for every parameter are checked to fall within a specific range to ensure that the system runs within the safety limits before transmission to the hardware. The parameters are saved locally and verified with the pacemaker during the communication.

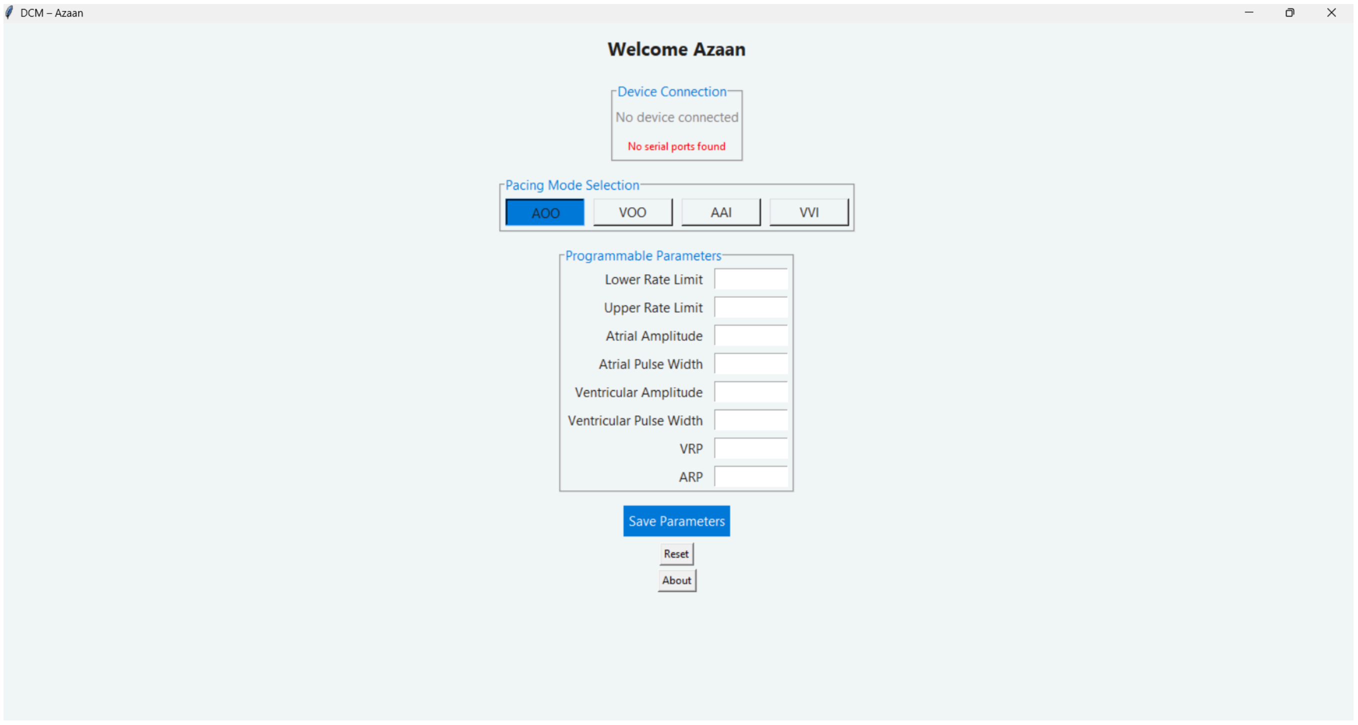
The DCM receives telemetry and egram signals from the device and displays them in real time, and also identifies the hardware that has been connected through a unique device ID to prevent mismatched sessions or deal with different settings or parameters set. The hardware includes serial data for parameters, egram streaming and visual alerts or updated settings. Each pacing mode, of the four implemented, is represented by a simple state-machine logic within the DCM and Simulink model.

The interface provides the user with a login screen, which allows users to register or login, a dashboard which allows for parameter configuration and telemetry status, and the egram view, which is the real-time signal display from the hardware. This design allows us the space to improve and make changes in the future, so that other features and functions can be added, while maintaining appropriate and instant communication with the pacemaker.


DCM Login
Device Controller–Monitor
Login
Register
Quit


Login
Username
Password
Login


Register
Username
Password
Register


DCM – Azaan
Welcome Azaan
Device Connection
No device connected
No serial ports found
Pacing Mode Selection
AOO
VOO
AAI
VVI
Programmable Parameters
Lower Rate Limit
Upper Rate Limit
Atrial Amplitude
Atrial Pulse Width
Ventricular Amplitude
Ventricular Pulse Width
VRP
ARP
Save Parameters
Reset
About

Part 2

Requirements/Design Decision Potential Changes

In regard to the potential changes, we note that there are several requirements that we will expand on in the second deliverable.

1. We implemented the four basic modes: AOO, VOO, AAI, VVI. We will be working off of this to implement four more parameters that build off of these: AOOR, VOOR, AAIR, VVIR.
2. We will work on the rate adaptive modes through the on-board accelerometer, which will track the activity of the user. This will also affect the lower rate limit (LRL) depending on the activity being undertaken, so increased intensity will have a greater LRL. This will be checked through shaking the pacemaker hardware to mimic different activities.
3. We will make sure that all the modes and functions are integrated into one whole model system, so that the overall project is fully functional.
4. We will generate an assurance case for the safety requirements and specifications that are implemented. We will justify and demonstrate how our safety requirements are met, whereas deliverable one required just basic testing without much emphasis on safety.
5. We will work on expanding the DCM to include all required modes and parameters, which we will implement.
6. We will fully adapt our system to ensure that serial communication works appropriately to transmit any and all relevant info from the DCM to the pacemaker and vice versa. This relates to the setting, storing, transmitting and verifying programmable parameter data on both ends.
7. We will display the egram data on the DCM, which is part of the data sent from the hardware to the user interface.
8. We will work on our documentation to mention any and all changes that we have implemented, and give an explanation in the updated document.

Module Description

Module 1: ui_login.py

(a) **Purpose of the Module:** This module manages all user authentication and registration functions for the Device Controller-Monitor (DCM). It provides the user interface for logging in and creating new users before accessing the main pacemaker control dashboard.

(b) Public Functions

Function	Parameters	Description
hash_pw(pw)	pw (string): plaintext password	Returns SHA-256 hash of the given password string.
LoginWindow.__init__()	none	Initializes the main login window and displays primary interface buttons.
LoginWindow.register_screen()	none	Opens the registration window, allowing new users to register.
LoginWindow.login_screen()	none	Opens the login window, allowing existing users to authenticate.
LoginWindow.load_users()	none	Loads user credentials from users.json.
LoginWindow.save_users(users)	users (list of dicts): user data to save	Writes user credential list to users.json.

(c) Black-Box Behaviour

Function	Inputs	Outputs	Description
hash_pw(pw)	Plaintext password	Hashed password (hex string)	Performs one-way password hashing for secure storage.
LoginWindow.__init__()	none	Creates and displays the main login UI	Displays a window with Login, Register, and Quit buttons.
register_screen()	none	Popup registration window	Prompts user for username/password and validates input.
login_screen()	none	Popup login window	Prompts user for credentials and launches the Dashboard upon success.
load_users()	none	Returns the user list from JSON	Reads user data; returns an empty list if no file exists.
save_users(users)	Updated list	Saves to file	Overwrites users.json with new credential data.

(d) Global/State Variables

Variable	Type	Description
THEME_BG, THEME_ACCENT, THEME_TEXT, THEME_FONT	Constants	Define UI theme (colour palette and font).
users.json	File	Persistent JSON file storing all registered users as a list of dictionaries.

(e) **Private Functions:** None explicitly defined; however, register() and login() are nested callback functions inside their respective screen creation methods.

(f) Internal behaviour of each public and private function:

(i) __init__()

   (1) Creates a root Tk window with title, geometry, and background.

   (2) Adds header label and three buttons.

   (3) Each button triggers its respective method: login_screen(), register_screen(), or window destruction.

(ii) register_screen()

   (1) Creates a secondary Toplevel window centred on the screen.

   (2) Contains username and password entry fields.

   (3) Defines inner function register() that:

      a) Reads entries, trims whitespace.

      b) Validates non-empty inputs and password length ≥ 4.

      c) Calls load_users() to read users.json.

      d) Rejects duplicates and enforces a maximum of 10 users.

      e) Hashes the password using hash_pw().

      f) Calls save_users() to update the JSON file.

      g) Displays success/failure messages via the messagebox.

(iii) login_screen()

   (1) Creates a secondary Toplevel window for login.

   (2) Prompts for username and password.

   (3) Defines inner function login() that:

      a) Loads all users.

      b) Compares the entered username and hashed password.

      c) On success: shows success message, closes main window, launches Dashboard(username).

      d) On failure: shows an error dialog.

(iv) load_users() / save_users()

   (1) Handle file read/write operations safely using JSON.

   (2) Gracefully handle FileNotFoundError by returning an empty list.

Module 2: ui_dashboard.py

(a) Purpose of the Module: Provides the main Device Controller-Monitor (DCM) interface after login. Allows users to view and edit programmable pacemaker parameters, select pacing modes, connect to the hardware device over serial, and verify the device's identity.

(b) Public Functions

Function	Parameters	Description
Dashboard.__init__(username)	username (string): current user	Initializes DCM GUI, loads saved settings, and sets up hardware connection panel.
Dashboard.connect_device(port)	port (string): COM port name	Connects to the pacemaker hardware via serial and retrieves its unique ID.
Dashboard.check_device_id()	none	Compares the new device ID with the previously connected one and updates the status label.
Dashboard.load_previous()	none	Loads previously saved pacing parameters for the current user and mode.
Dashboard.save_params()	none	Validates and saves pacing parameters to patients.json.
Dashboard.reset_fields()	none	Clears all parameter fields in the GUI.
Dashboard.about()	none	Displays project/course information.
find_ports()	none	Detects available serial ports on the system and returns a list.

(c) Black-Box Behaviour

Function	Input	Output	Description
connect_device(port)	Port name	Connection status via GUI update	Attempts to open a serial connection and identify the device.
check_device_id()	none	Updates GUI text	Detects if the same, new, or different pacemaker is connected.
save_params()	User input fields	JSON file update + GUI message	Validates all parameter entries and saves them persistently.
load_previous()	none	Populates UI fields	Loads previous parameter values for the same user/mode.
reset_fields()	none	Clears UI	Removes text from all entry boxes.
about()	none	Popup	Displays a static info window.

(d) Global/State Variables

Variable	Type	Purpose
THEME_BG, THEME_ACCENT, THEME_TEXT, THEME_FONT	Constants	Control UI visual style.
LIMITS	Dictionary	Defines a safe numeric range for all pacing parameters.
MODES	List	List of pacing modes (AOO, VOO, AAI, VVI).

Variable	Type	Purpose
THEME_BG, THEME_ACCENT, THEME_TEXT, THEME_FONT	Constants	Control UI visual style.
patients.json	File	Stores user-specific parameter configurations.
device_id.json	File	Stores the last connected hardware ID for verification.
self.serial	Serial object	Active serial port connection to pacemaker.
self.device_id	String	Stores the current device's unique identifier.
self.entries	Dict[str, tk.Entry]	Maps parameter names to input entry fields.

(e) Private Functions:

- **(i)** _register() and _login() (nested functions from ui_login.py) act as private callbacks.
- **(ii)** In this module, no formally private functions exist, but check_device_id() and load_previous() behave as internal helpers.

(f) Internal behaviour of each public and private function:

- (i) __init__()
  - (1) Initializes GUI with sections for:
    - a) Device Connection (lists COM ports, handles connection)
    - b) Pacing Mode Selection (radio buttons)
    - c) Programmable Parameters (auto-generated from LIMITS)
    - d) Control Buttons (Save, Reset, About)
  - (2) Automatically loads previous parameters for this user. 2.
- (ii) connect_device(port)
  - (1) Opens serial port (baud = 115200).
  - (2) Sends ID. command to hardware; expects response DEVICE_ID.

(3) If the response is missing, assign a simulated ID.

(4) Calls check_device_id() to verify consistency.

(5) Updates the GUI status label colour (green for same/new, red for different).

(iii) check_device_id()

(1) Reads device_id.json for the last known ID.

(2) If none exists, it marks a new connection.

(3) If matches: “Same device connected”.

(4) If mismatch: “Different device detected”.

(5) Updates the stored ID and displays the result.

(iv) save_params()

(1) Reads all entry field values.

(2) Validates that:

a) Each field is numeric.

b) Each value lies within its defined range.

(3) Loads existing patients.json (or creates new).

(4) Updates the user’s mode data and writes JSON.

(5) Shows a success message.

(v) load_previous()

(1) Loads previously saved parameter values (if available) for this user and current pacing mode.

(2) Populates each GUI entry box with stored values.

(vi) reset_fields()

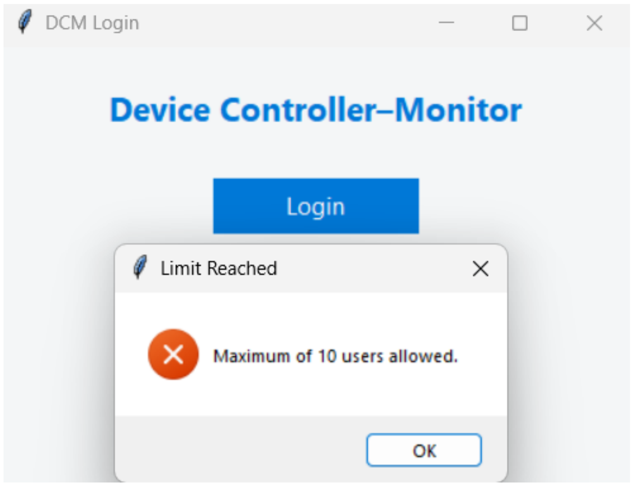
(1) Clears all Tkinter Entry widgets (sets to empty).

(vii) about()

(1) Displays a pop-up with the project title, course, and group number.

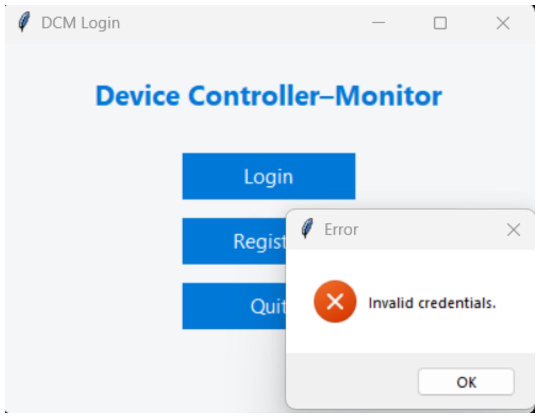
Testing

Test Case: User Registration (Max 10 Users)

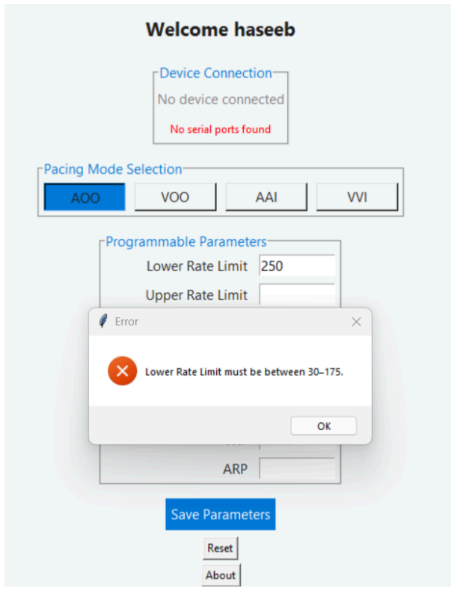
Field	**Description**
Purpose	To verify that the system allows user registration and enforces a maximum of 10 stored users.
Input Conditions	Enter a new username and password. Repeat until 10 users are registered. Attempt to register an 11th user.
Expected Output	Users 1-10 are successfully registered. On the 11th attempt, the system displays an error message: "Maximum of 10 users allowed."
Actual Output	 DCM Login Device Controller–Monitor Limit Reached Maximum of 10 users allowed.
Result	Pass

Test Case: User Login (Correct and Incorrect Credentials)

Field	**Description**
Purpose	To verify that user authentication only succeeds with valid credentials.
Input Conditions	Case A: Correct username and password. Case B: Incorrect credentials.
Expected Output	Case A: DCM proceeds to the Mode Selection Screen. Case B: Display message "Invalid credentials."

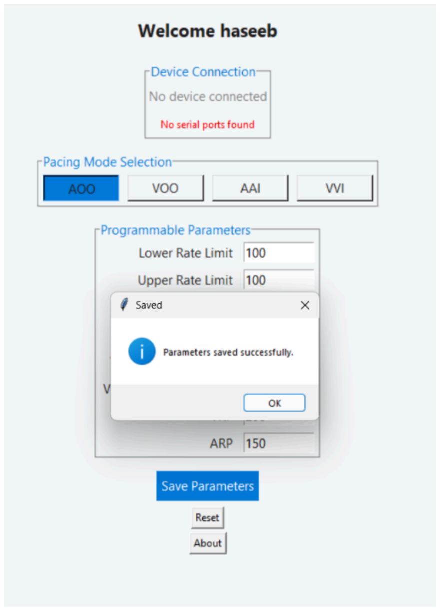
Actual Output	 DCM Login Device Controller–Monitor Error Invalid credentials.
Result	Pass

Test Case: Parameter Input Validation

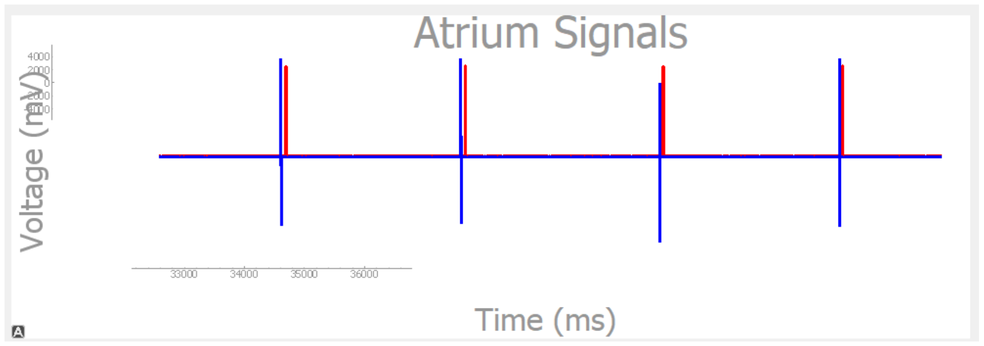
Field	**Description**
Purpose	Ensure input validation prevents unsafe values.
Input Conditions	Enter “250” for Lower Rate Limit (outside allowed range).
Expected Output	System displays error message: “Lower Rate Limit must be between 30-175” and does not proceed.
Actual Output	 Welcome haseeb Device Connection No device connected No serial ports found Pacing Mode Selection AOO Programmable Parameters Lower Rate Limit 250 Upper Rate Limit Error Lower Rate Limit must be between 30–175. ARP

Result	Pass

Test Case: Save Parameters

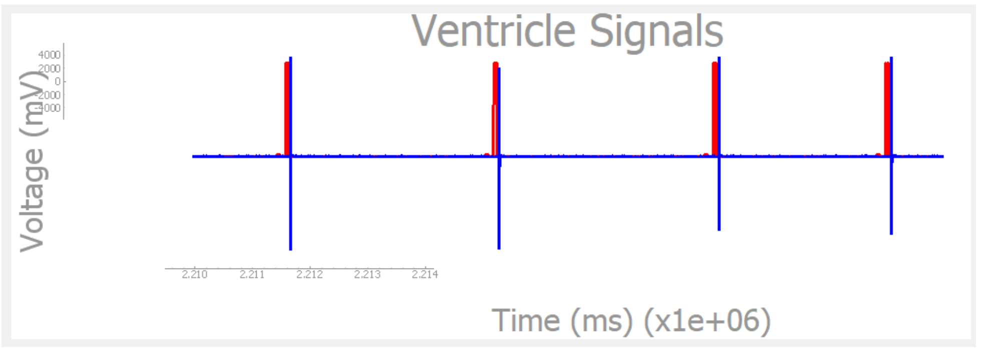
Field	**Description**
Purpose	To verify that the selected pacing mode and parameters are stored locally.
Input Conditions	The user selects AOO mode and enters valid parameter values, then presses “Save.”
Expected Output	System confirms: “Parameters saved successfully.”
Actual Output	 **Welcome haseeb** Device Connection No device connected No serial ports found Pacing Mode Selection AOO VOO AAI VVI Programmable Parameters Lower Rate Limit 100 Upper Rate Limit 100 Saved Parameters saved successfully. OK ARP 150 Save Parameters Reset About
Result	Pass

Test Case: AOO

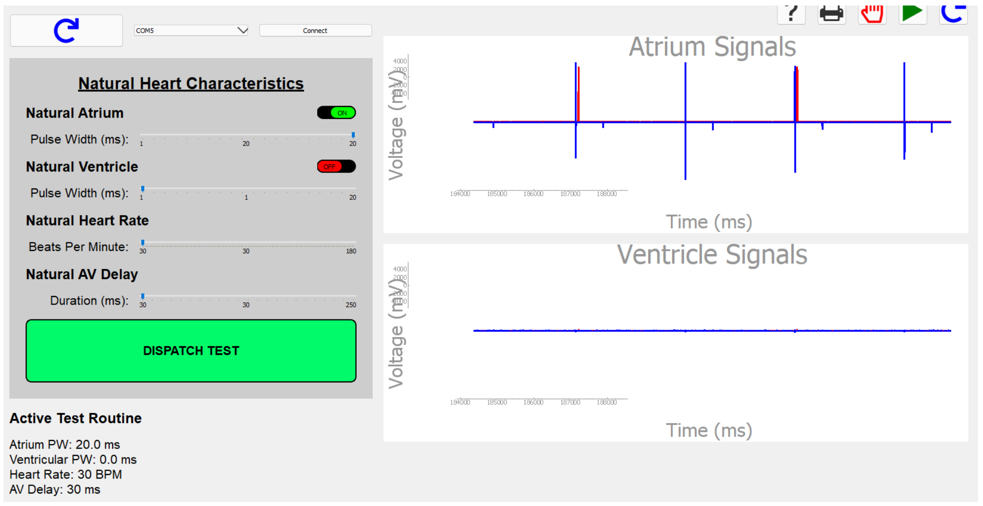
Field	Description
Purpose	Verify the fixed-rate, asynchronous delivery of atrial pacing pulses. Independent of the real heart rate.
Input Conditions	The input for the AOO is a LRL = 60bpm. We compared it with a natural heart rate of 60bpm to ensure that the pulses are at a fixed rate.
Expected Output	Pulses are at a fixed rate.
Actual Output	
Result	Pass

Test Case: VOO

Field	Description
Purpose	Verify the fixed-rate, asynchronous delivery of ventricle pacing pulses. Independent of the real heart rate.
Input Conditions	The input for the VOO is a LRL = 60bpm. We compared it with a natural heart rate of 60bpm to ensure that the pulses are at a fixed rate.
Expected Output	Pulses are at a fixed rate.

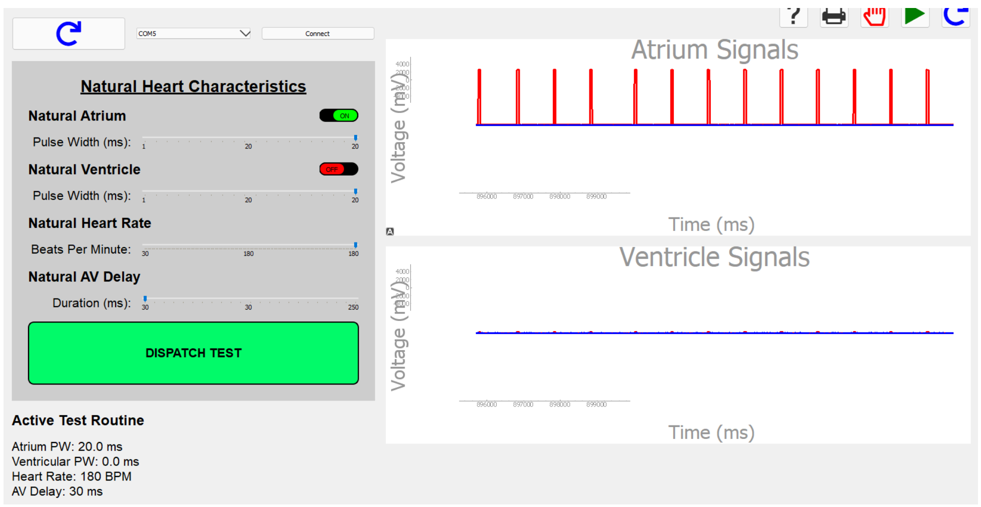
Actual Output	
Result	Pass

Test Case: AAI Low BPM

Field	Description
Purpose	Verify the inhibited pacing function in the Atrium. The test must prove that the pacemaker inhibits the scheduled pace pulse when a natural atrial beat is sensed, and correctly delivers a pace pulse only if the heart fails to beat before the LRL interval expires.
Input Conditions	The input for the test is the natural heartbeat at 30 BPM (Below the LRL of 60 BPM), with a natural atrium at a pulse width of 20 ms.
Expected Output	Frequent paces (in blue) due to the extremely low natural heart rate, with the frequency of the paces decreasing as the natural heart rate gets closer to the LRL
Actual Output	 COM5 Connect DISPATCH TEST Atrium Signals Ventricle Signals Voltage (mV) Time (ms) Natural Heart Characteristics Natural Atrium ON Pulse Width (ms): 20 Natural Ventricle OFF Pulse Width (ms): 1 Natural Heart Rate Beats Per Minute: 30 Natural AV Delay Duration (ms): 30 Active Test Routine Atrium PW: 20.0 ms Ventricular PW: 0.0 ms Heart Rate: 30 BPM AV Delay: 30 ms There is a high frequency of paces due to the low heart rate (lower than LRL) as expected

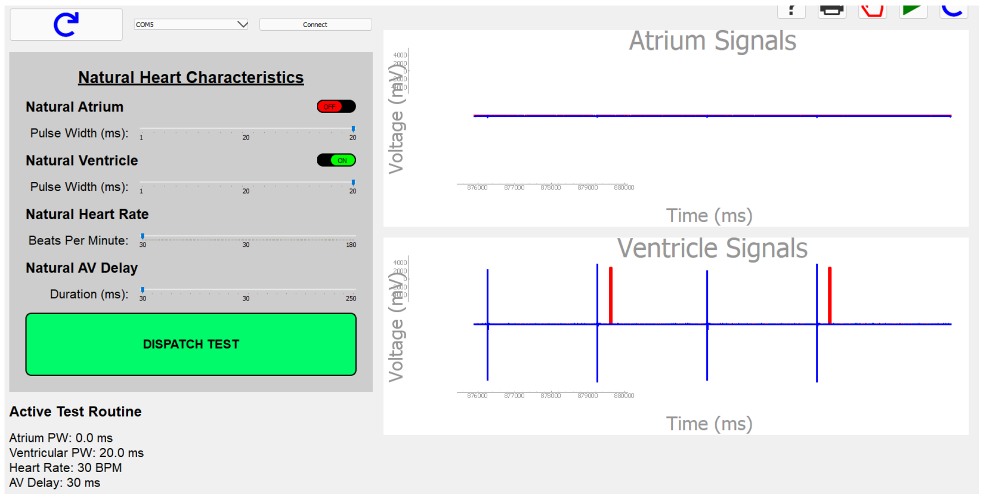
Result	Pass

Test Case: AAI High BPM

Field	Description
Purpose	Verify the inhibited pacing function in the Atrium. The test must prove that the pacemaker inhibits the scheduled pace pulse when a natural atrial beat is sensed, and correctly delivers a pace pulse only if the heart fails to beat before the LRL interval expires.
Input Conditions	The input for the test is the natural heartbeat at 180 BPM (Above the LRL of 60 BPM), with a natural atrium at a pulse width of 20 ms.
Expected Output	With a natural heart rate above LRL, there should be no paces produced. (No blue paces).
Actual Output	 COM5 Connect Atrium Signals Voltage (mV) Time (ms) Ventricle Signals Voltage (mV) Time (ms) DISPATCH TEST **Natural Heart Characteristics** Natural Atrium: ON Pulse Width (ms): 20 Natural Ventricle: OFF Pulse Width (ms): 20 Natural Heart Rate Beats Per Minute: 180 Natural AV Delay Duration (ms): 30 **Active Test Routine** Atrium PW: 20.0 ms Ventricular PW: 0.0 ms Heart Rate: 180 BPM AV Delay: 30 ms There is no pace generated as the natural heart rate is higher than the LRL as expected.
Result	Pass

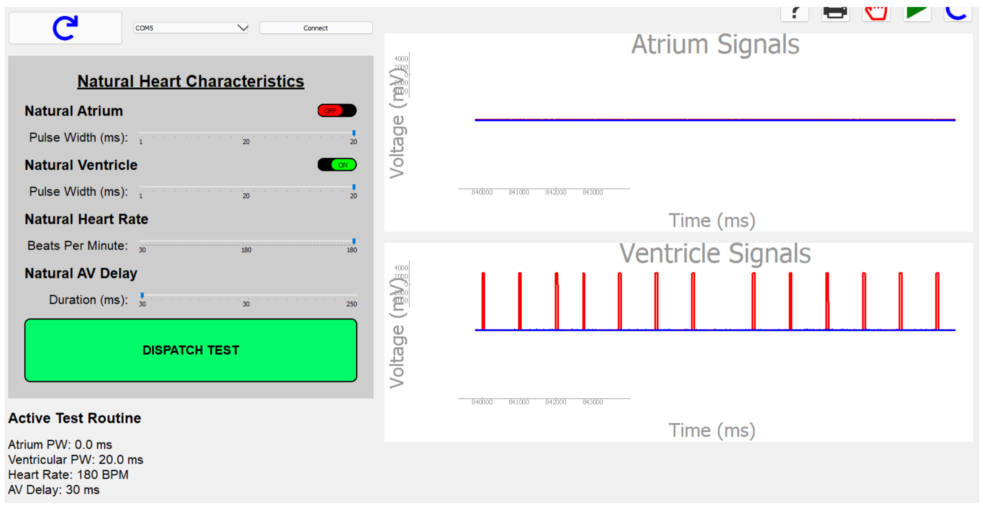
Test Case: VVI Low BPM

Field	Description
Purpose	Verify the inhibited pacing function in the Atrium. The test must prove that the pacemaker inhibits the scheduled pace pulse when a natural atrial beat is sensed, and correctly delivers a pace pulse only if the heart fails to beat before the LRL interval expires.

Input Conditions	The input for the test is the natural heartbeat at 30 BPM (Below the LRL of 60 BPM), with a natural atrium at a pulse width of 20 ms.
Expected Output	Frequent paces (in blue) due to the extremely low natural heart rate, with the frequency of the paces decreasing as the natural heart rate gets closer to the LRL
Actual Output	 COM5 Connect Atrium Signals Voltage (mV) Time (ms) Ventricle Signals Voltage (mV) Time (ms) OFF ON DISPATCH TEST Natural Heart Characteristics Natural Atrium Pulse Width (ms): 1 20 20 Natural Ventricle Pulse Width (ms): 1 20 20 Natural Heart Rate Beats Per Minute: 30 30 180 Natural AV Delay Duration (ms): 30 30 250 Active Test Routine Atrium PW: 0.0 ms Ventricular PW: 20.0 ms Heart Rate: 30 BPM AV Delay: 30 ms There is a high frequency of paces due to the low heart rate (lower than LRL) as expected
Result	Pass

Test Case: VVI High BPM

Field	Description
Purpose	Verify the inhibited pacing function in the Atrium. The test must prove that the pacemaker inhibits the scheduled pace pulse when a natural atrial beat is sensed, and correctly delivers a pace pulse only if the heart fails to beat before the LRL interval expires.
Input Conditions	The input for the test is the natural heartbeat at 180 BPM (Above the LRL of 60 BPM), with a natural atrium at a pulse width of 20 ms.
Expected Output	With a natural heart rate above LRL, there should be no paces produced (No blue paces).

Actual Output	 COM5 Connect Atrium Signals Voltage (mV) Time (ms) Ventricle Signals Voltage (mV) Time (ms) DISPATCH TESTNatural Heart Characteristics Natural Atrium OFF Pulse Width (ms): 1 20 20 Natural Ventricle ON Pulse Width (ms): 1 20 20 Natural Heart Rate Beats Per Minute: 30 180 180 Natural AV Delay Duration (ms): 30 30 250 Active Test Routine Atrium PW: 0.0 ms Ventricular PW: 20.0 ms Heart Rate: 180 BPM AV Delay: 30 ms There is no pace generated as the natural heart rate is higher than the LRL as expected.
Result	Pass

GenAI Usage

One roadblock we faced, which we used GenAI for, was figuring out how to implement the serial communication and identification function of the hardware. We did not quite understand how to go about this, as we did not expect the board to have a special identification tag. For this, we used AI to explain the concept of how we can differentiate between hardware that belongs to us and one that belongs to another. We were able to get the gist of the matter, and using the steps outlined by the AI model, we were able to successfully implement the serial communication and identification process, which was needed.