3K04 Deliverable 1: Documentation

Group 33

Last Updated: 2025-10-25

Contents

List of Figures

List of Tables

1 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

CCS - Cardiac Conduction System

DCM - Device Controller-Monitor

GPIO - General Purpose Input Output

GUI - Graphical User Interface

PWM - Pulse Width Modulation

2.1 Bradycardia Operating Abbreviations

Table 1: 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 Part 1

3.1 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.

3.2 Requirements

- Overall system requirements (summarized from provided specification documents). It can be informal or semi-formal.

- Mode-specific requirements: AOO, VOO, AAI, VVI.

3.2.1 DCM Requirements

The user shall be capable of the following:

- Utilizing and managing windows for display of text and graphics.

- Processing user positioning and input buttons.

- Displaying all programmable parameters for review and modification.

- Visually indicating when telemetry is lost due to the device being out of range or noise.

- Vndicating when a different PACEMAKER device is approached than was previously interrogated

3.3 Design

In this section, you want to expand on the design decisions based on the requirements. You should be specific about your system design and how the various components relate together.

- System architecture (major subsystems, hardware hiding, pin mapping).

- Programmable parameters (rate limits, amplitudes, pulse widths, refractory periods, etc.).

- Hardware inputs and outputs (signals sensed, signals controlled).

- State machine design for each pacing mode (with diagrams if applicable). You can also use a tabular method.

- Simulink diagram

- Screenshots of your DCM, explaining its software structure

You should also be explicit on how your design decisions map directly to the requirements.
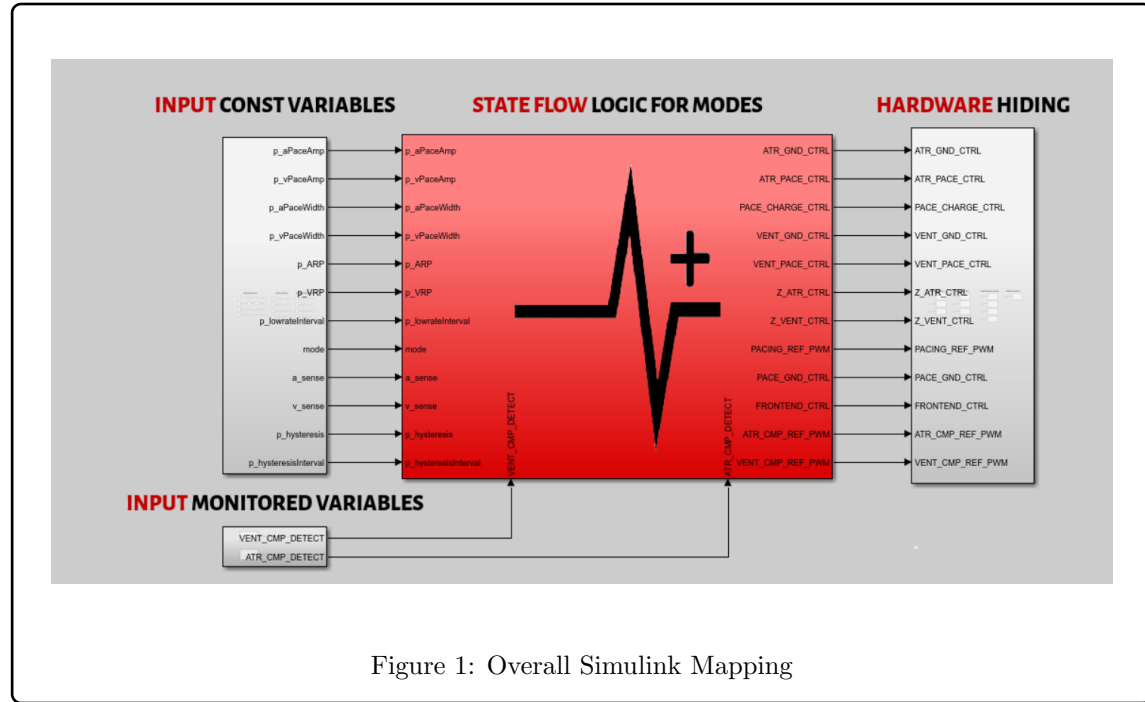
3.4 Simulink Design

3.4.1 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:

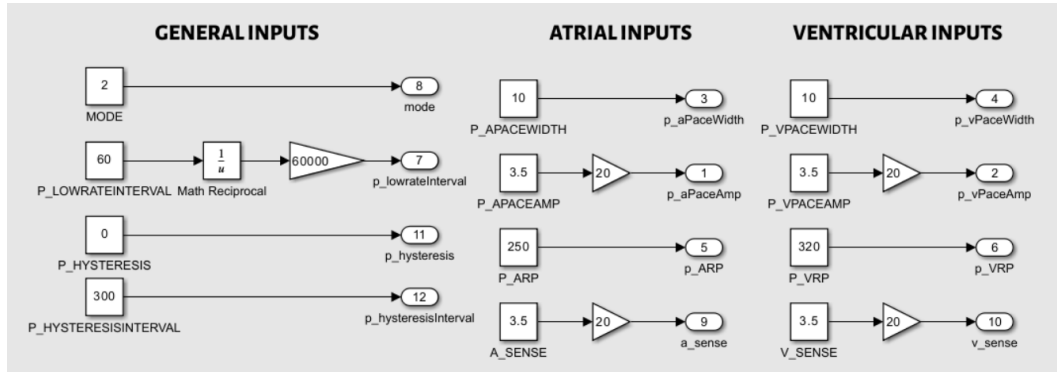### 3.4.2 Input Constant Variables

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.

- **Hysteresis Pace** - When enabled, a longer period is waited before pacing after sensing an event.

- Hysteresis Interval

The modfiable atrial variables are:

- **Pace Pulse Width** - Changes the width of the pace pulse

- **Pace Pulse Amplitude** - Changes the amplitude 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** -

The modifiable ventricular variables are:

- **Pace Pulse Width** - Changes the width of the pace pulse

- **Pace Pulse Amplitude** - Changes the amplitude 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

6

3.4.3   Monitored Input Variables

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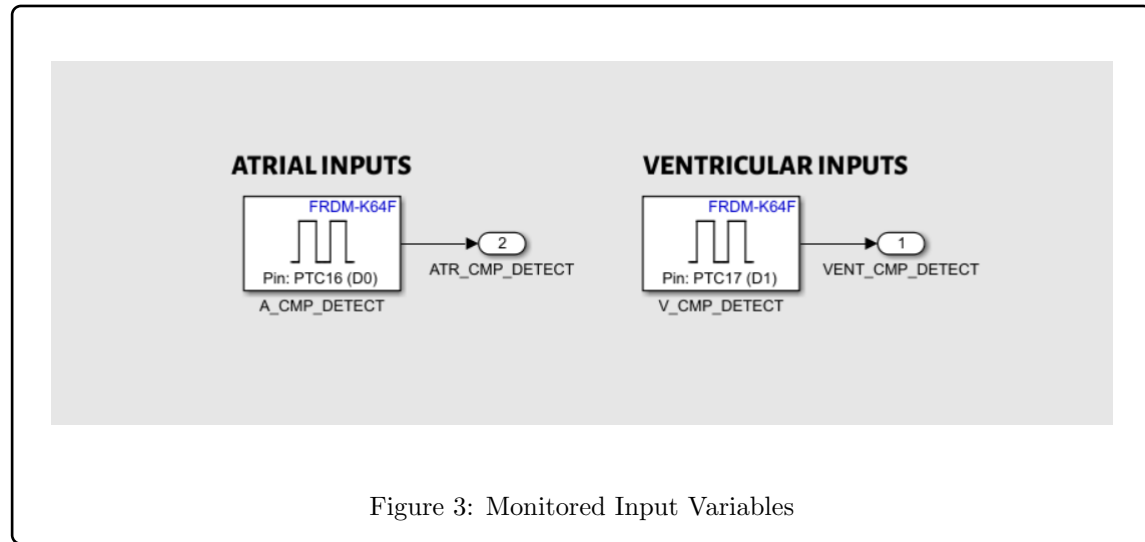
The monitored input variables can be seen in the above Figure 3. These are the atrial and ventricle detection variables. The pulses are sensed with through GPIO pins connecting to a board simulating heart conditions.
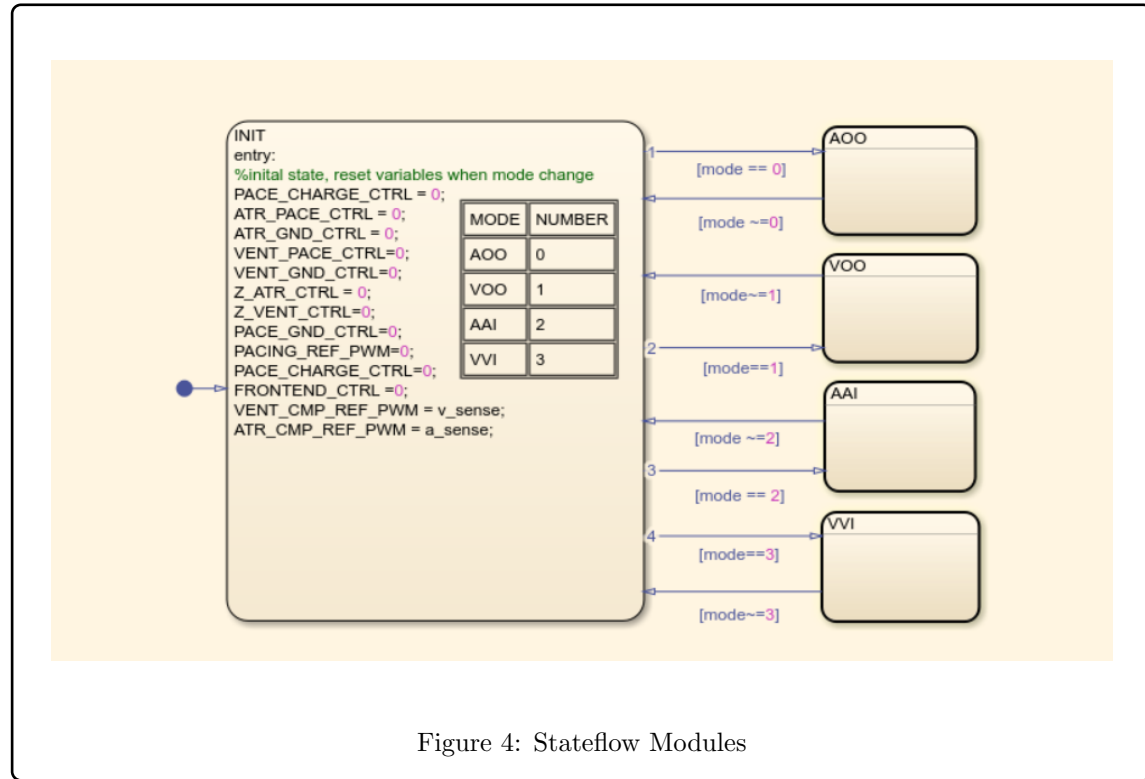
3.4.4 Stateflow Modules

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.
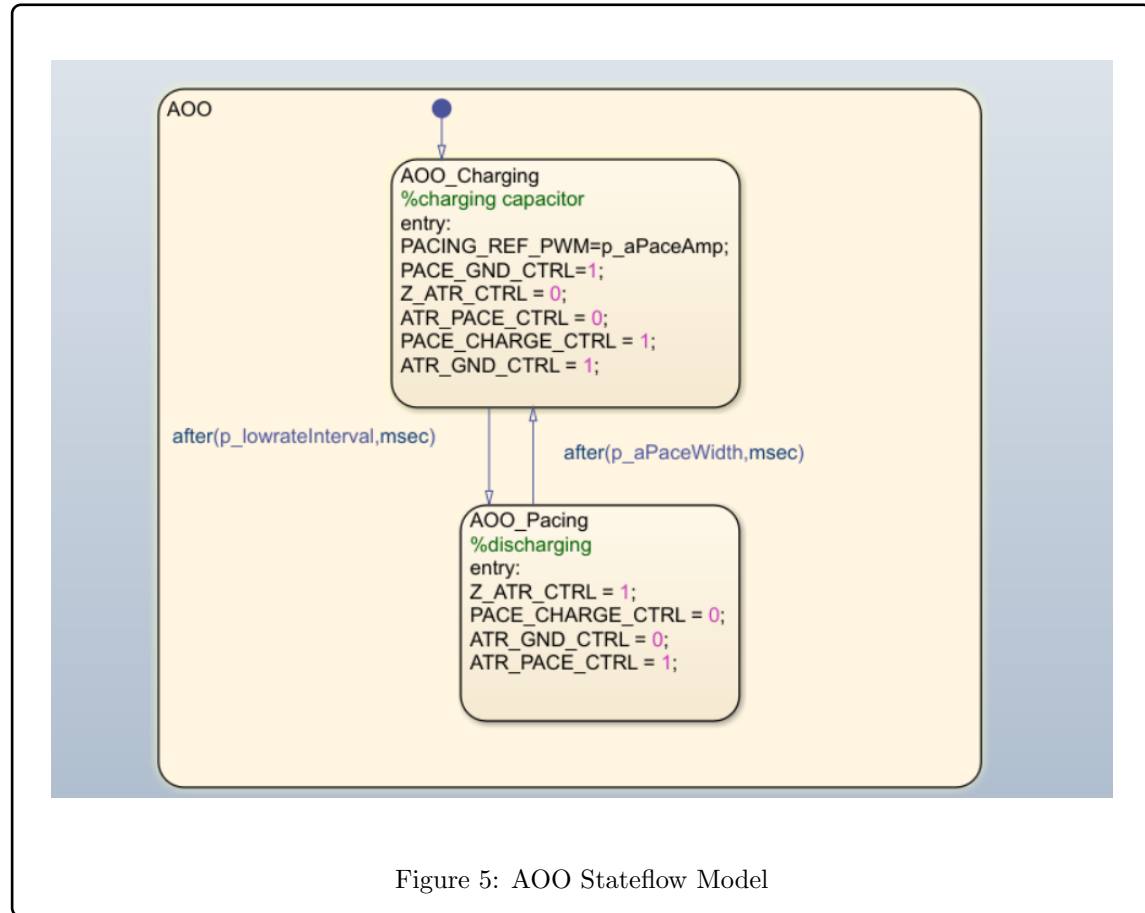
3.4.5 AOO Stateflow Model

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.

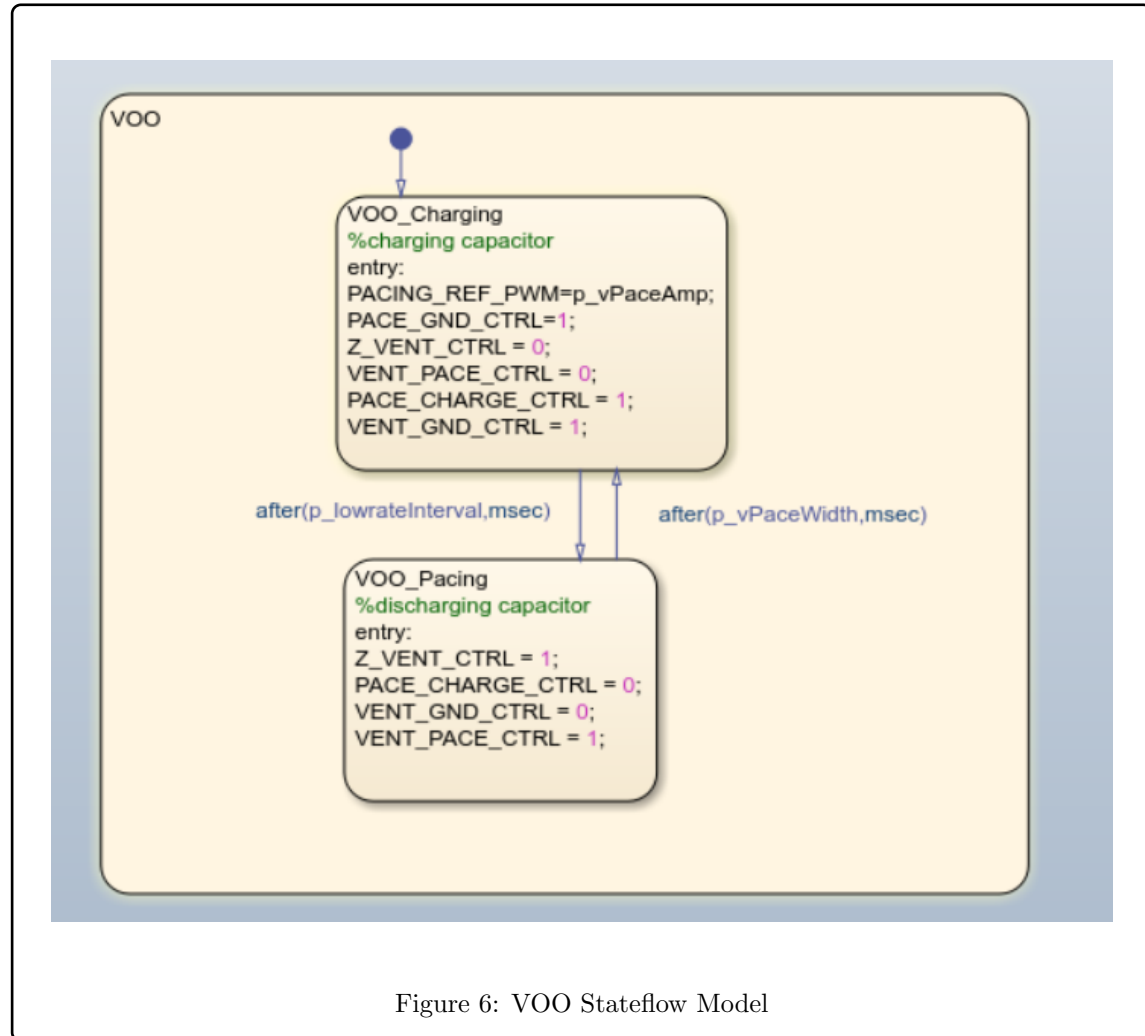### 3.4.6 VOO Stateflow Model

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.
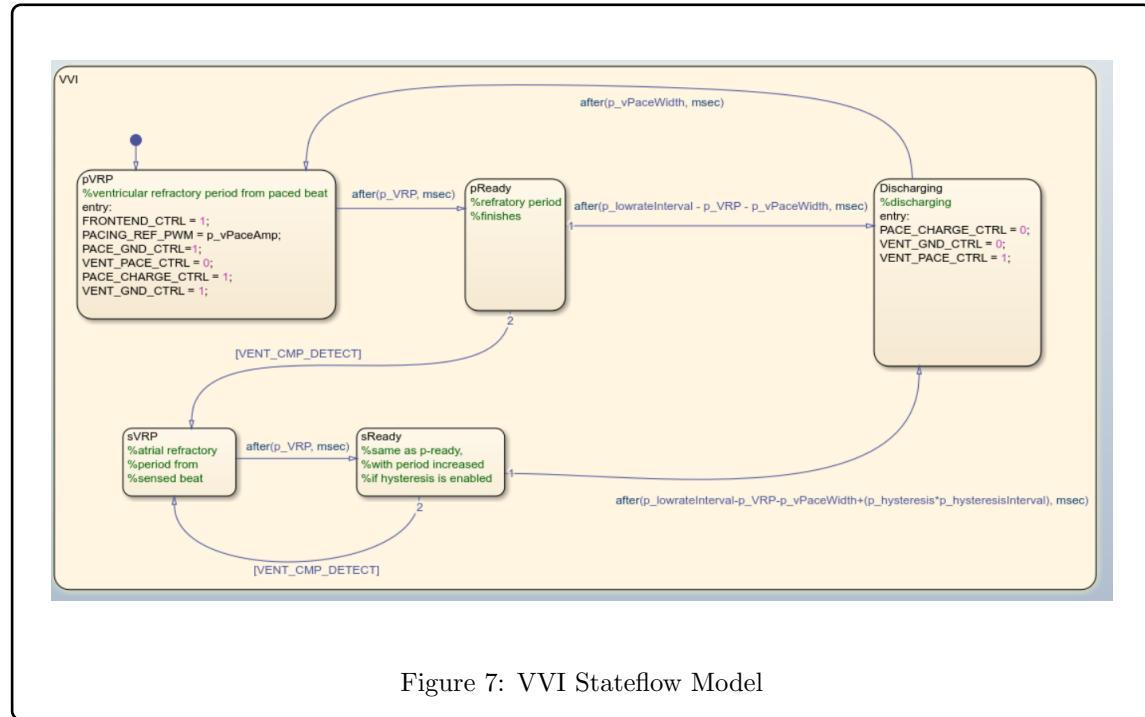
3.4.7 VII Stateflow Model

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.

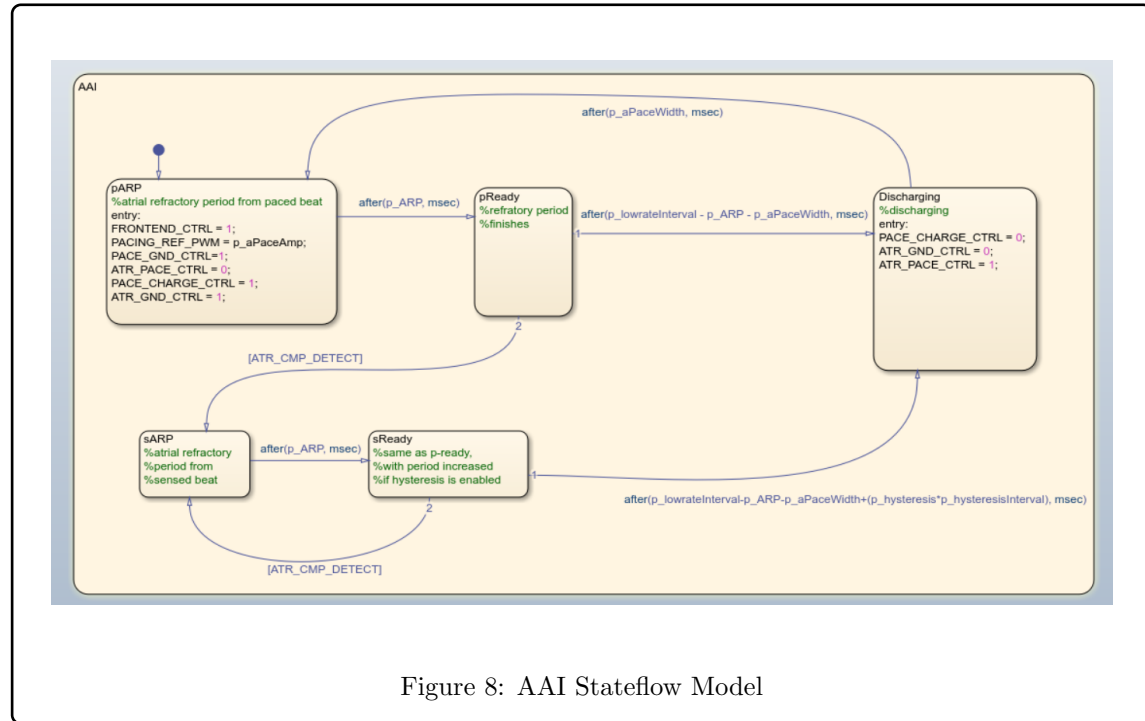### 3.4.8 AAI Stateflow Model

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.

3.4.9 Hardware Hiding
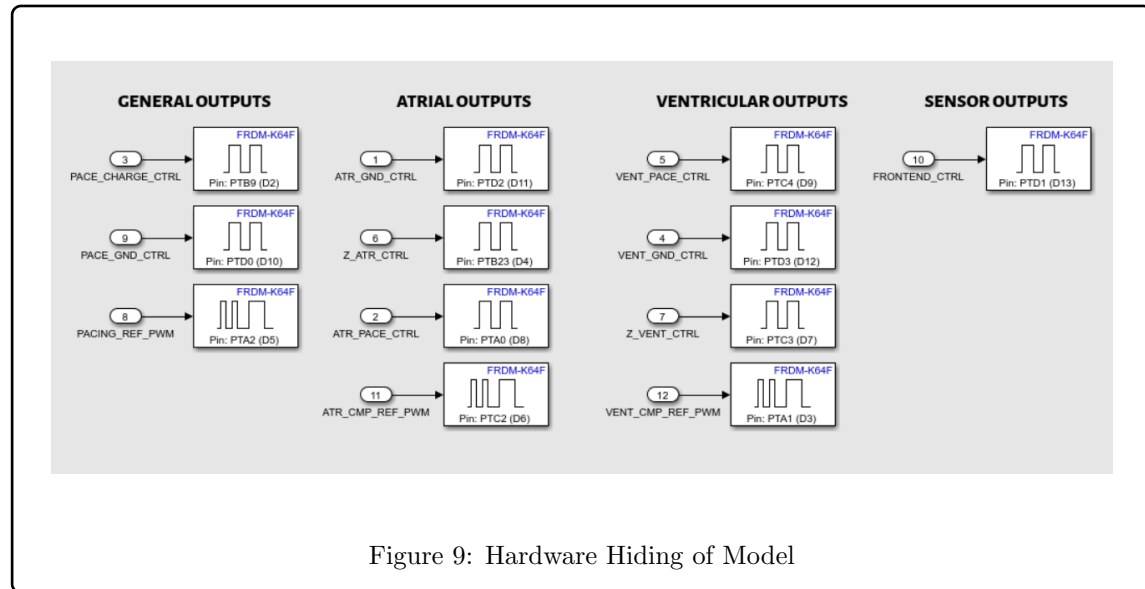
Figure 9: Hardware Hiding of Model

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.

4 Part 2

4.1 Requirements Potential Changes

Identify which requirements may evolve in the next deliverable (e.g., adding more modes, communication, new parameters).

4.2 Design Decision Potential Changes

List design choices that may need revisiting (e.g., choice of libraries, interface design, architecture).

4.3 Module Description

- Purpose of the component
- Key functions/methods (public vs internal)
- Global or state variables (if any)
- Interactions with other components

4.4 Testing

Document test cases for each module. Each test case should include:

1. Purpose of the test

2. Input conditions

3. Expected output

4. Actual output
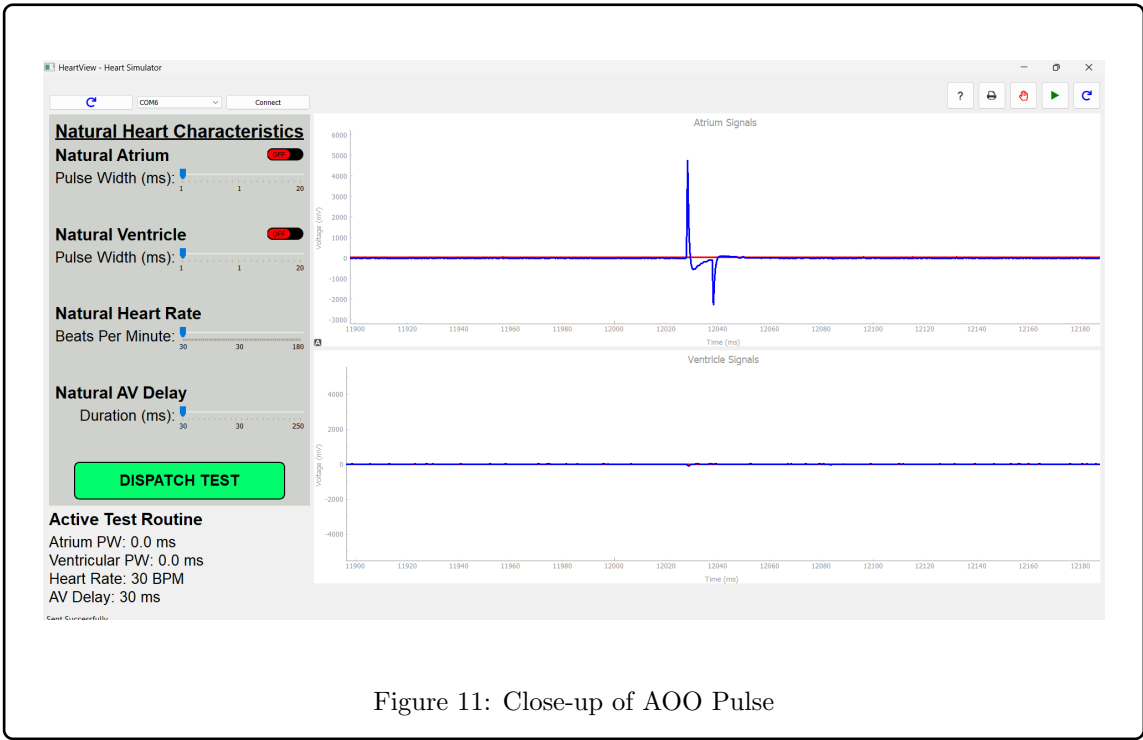
5. Result (Pass/Fail)
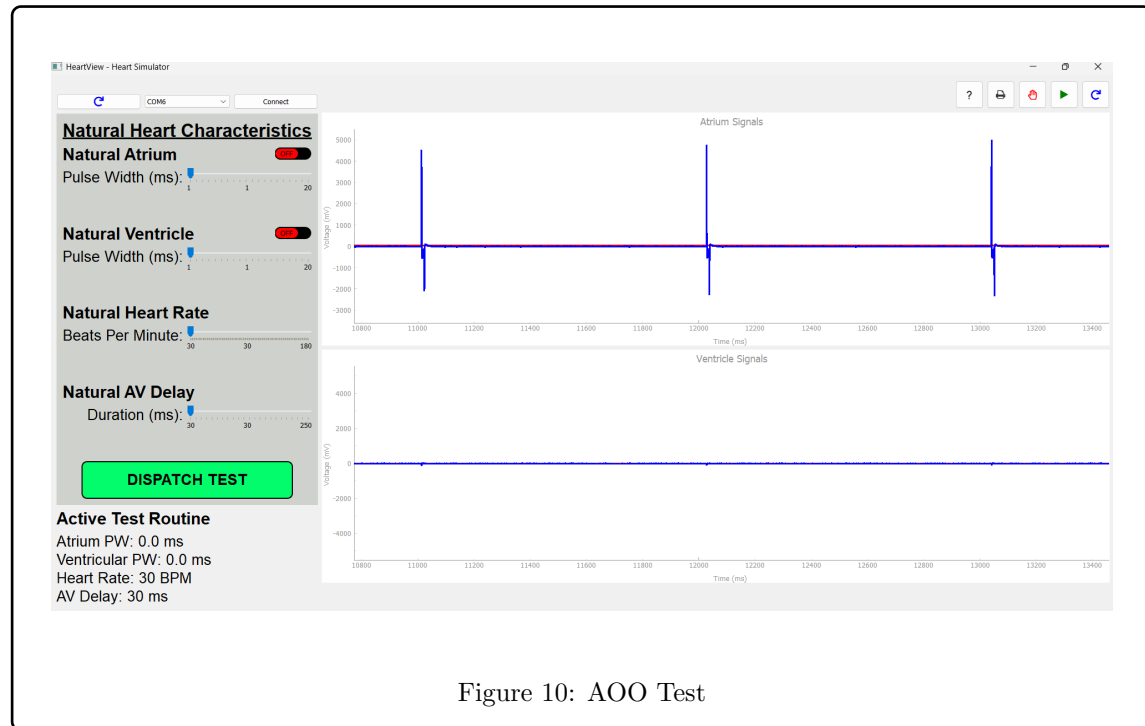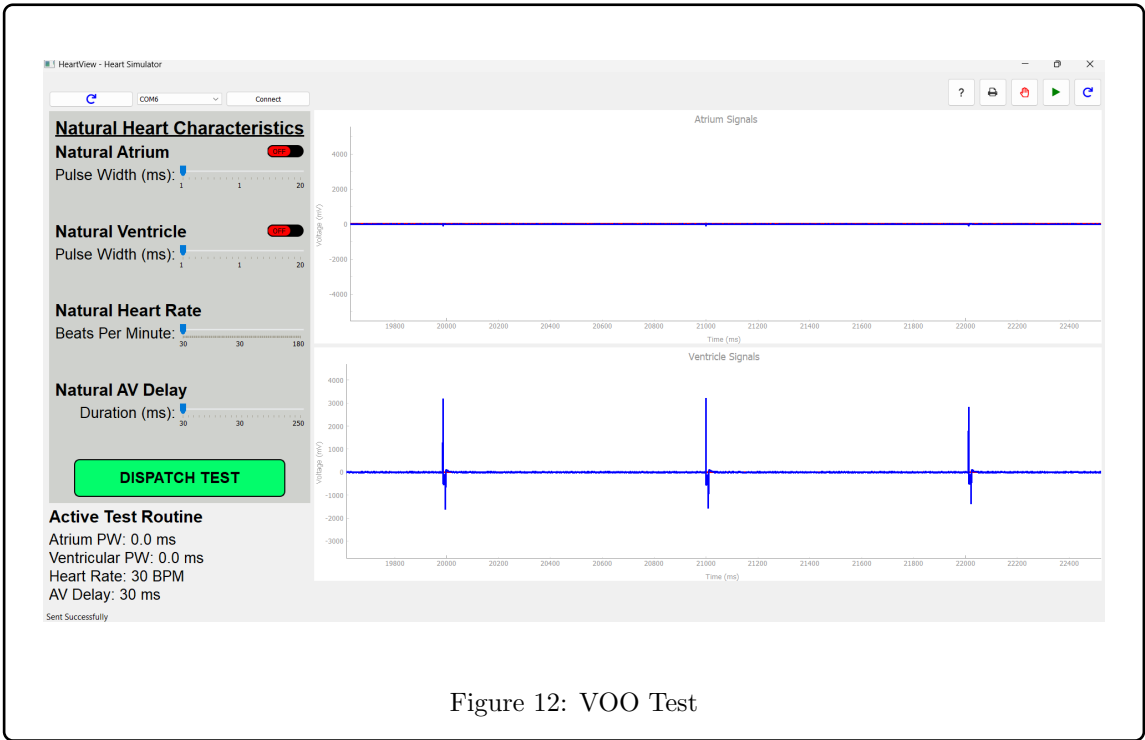

Figure 10: AOO Test

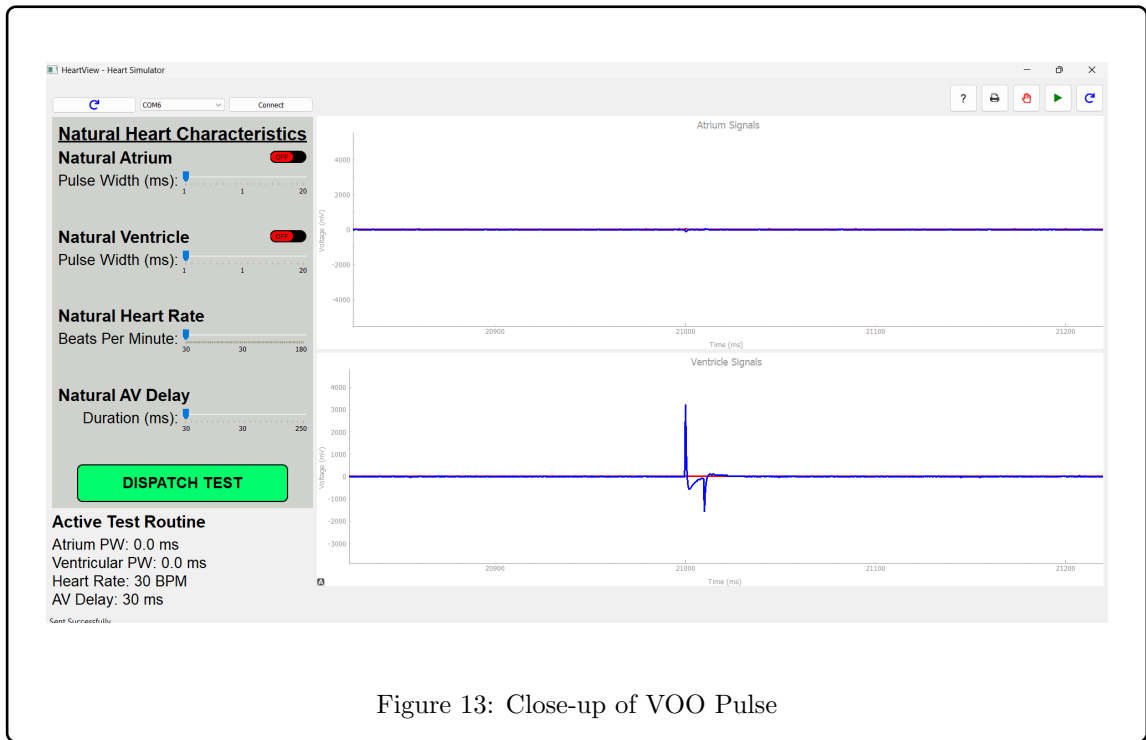Figure 11: Close-up of AOO Pulse
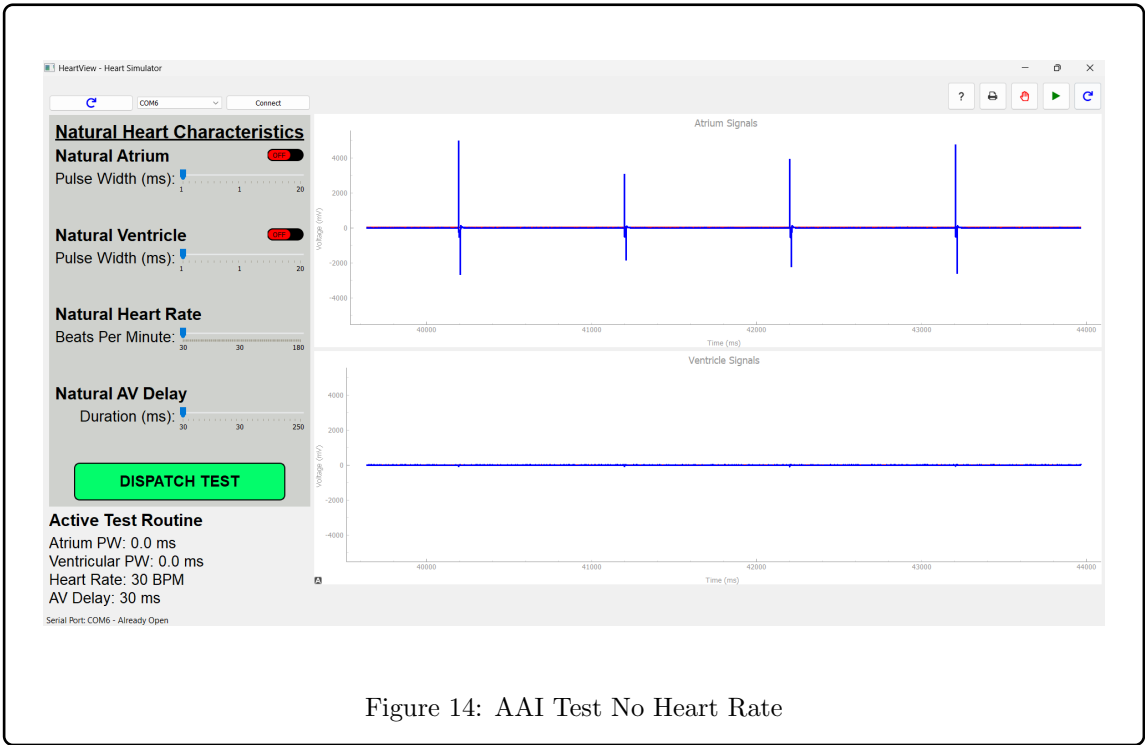
Figure 12: VOO Test

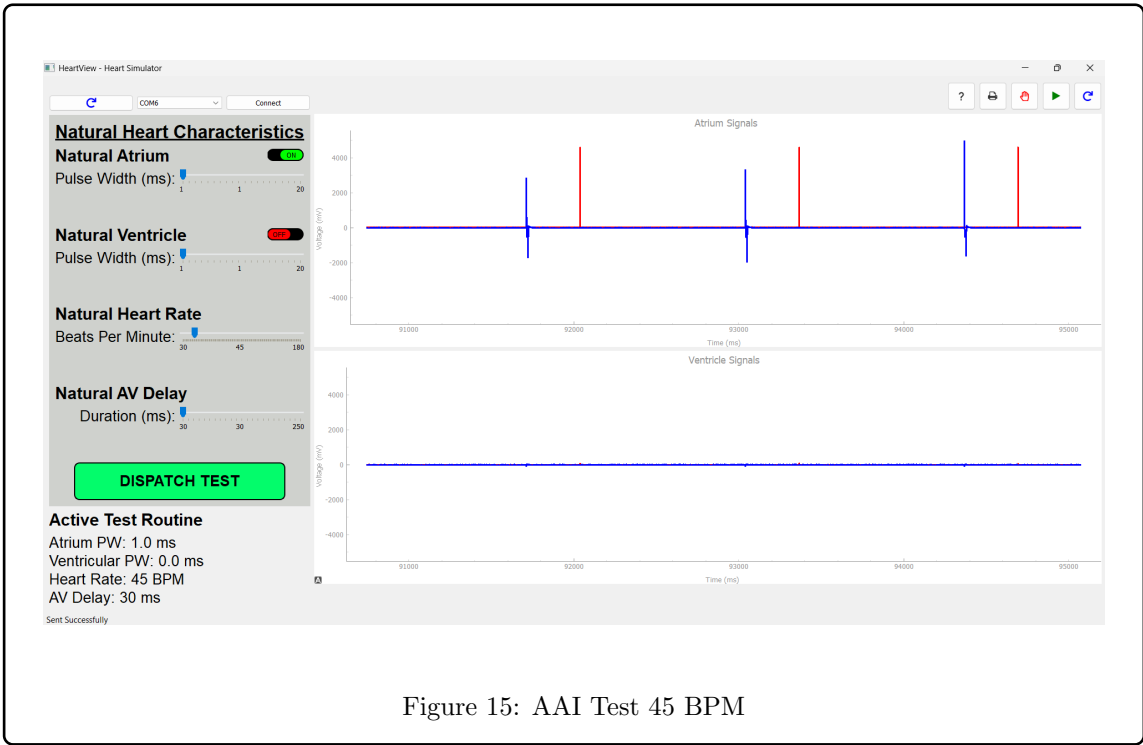Figure 13: Close-up of VOO Pulse

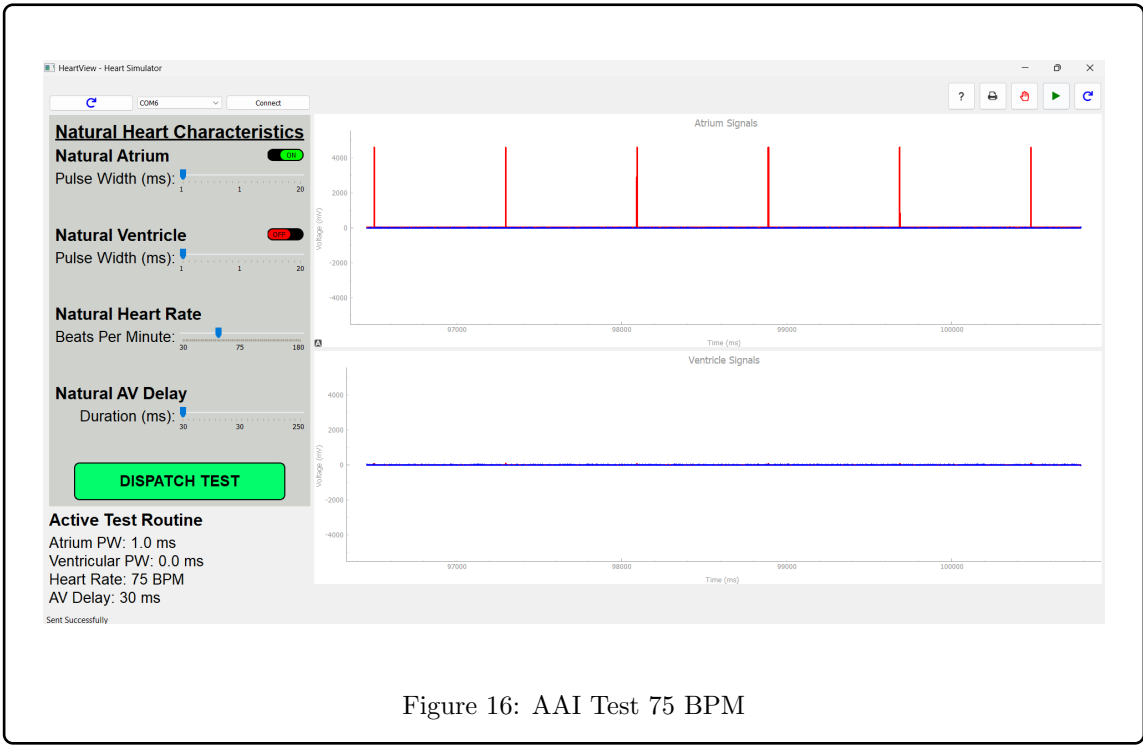Figure 14: AAI Test No Heart Rate
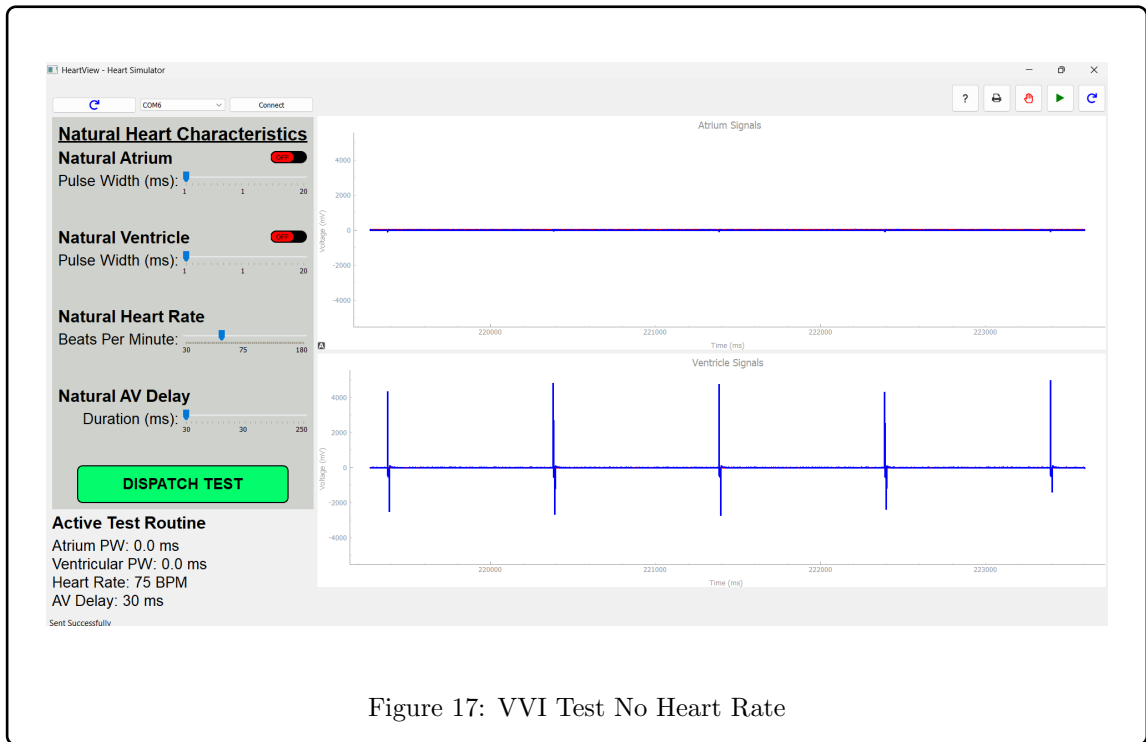
Figure 15: AAI Test 45 BPM

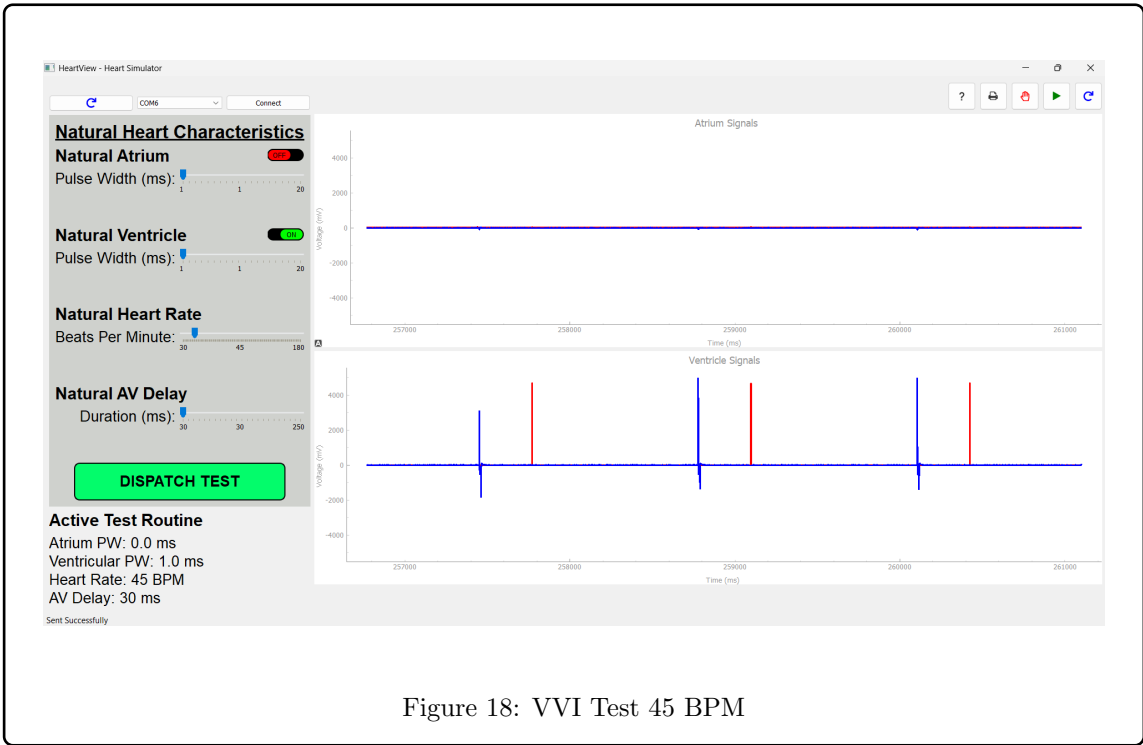Figure 16: AAI Test 75 BPM

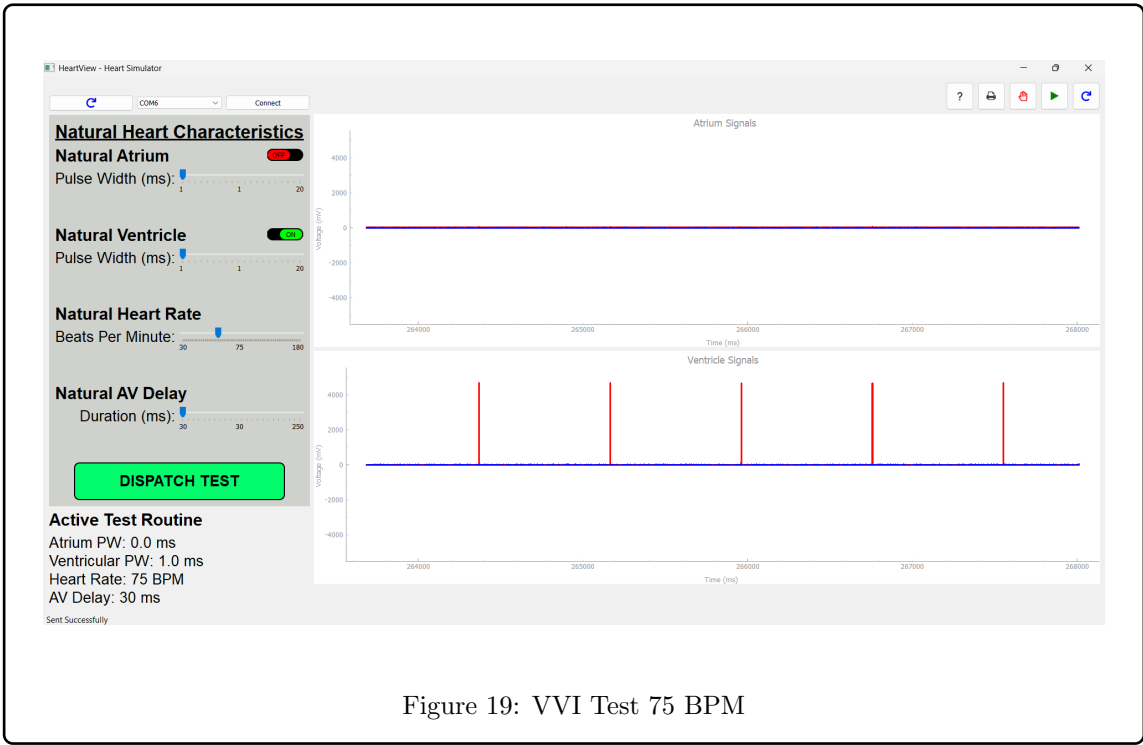Figure 17: VVI Test No Heart Rate
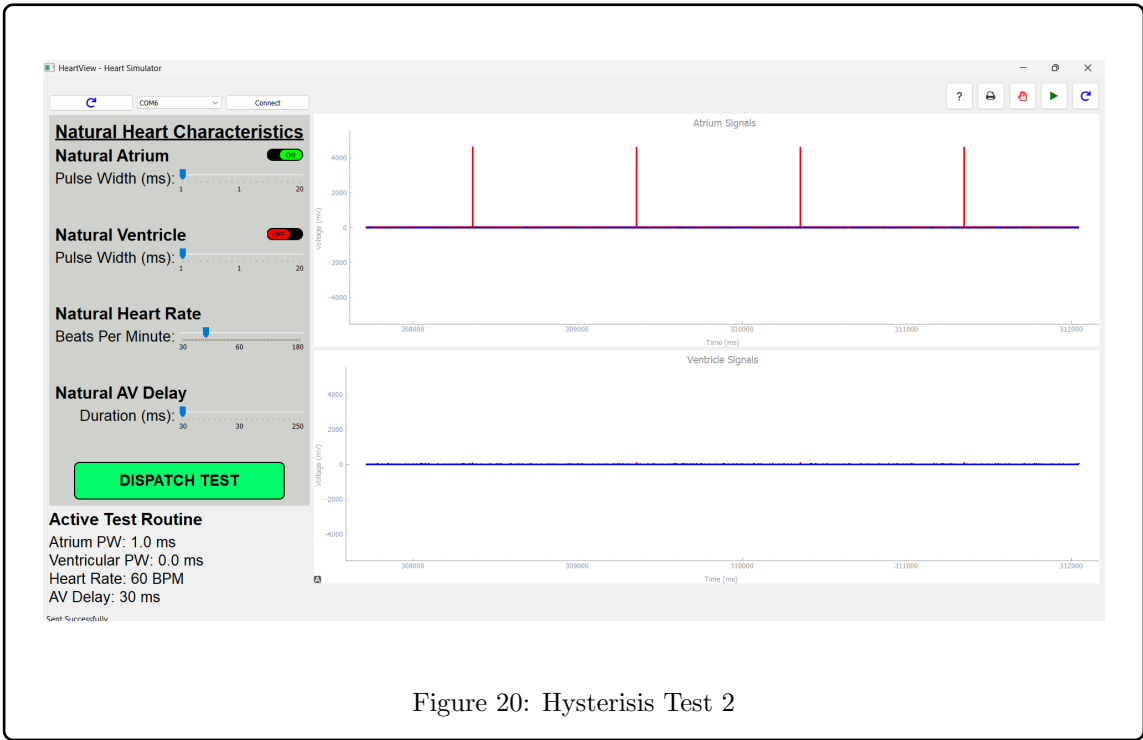
Figure 18: VVI Test 45 BPM

Figure 19: VVI Test 75 BPM
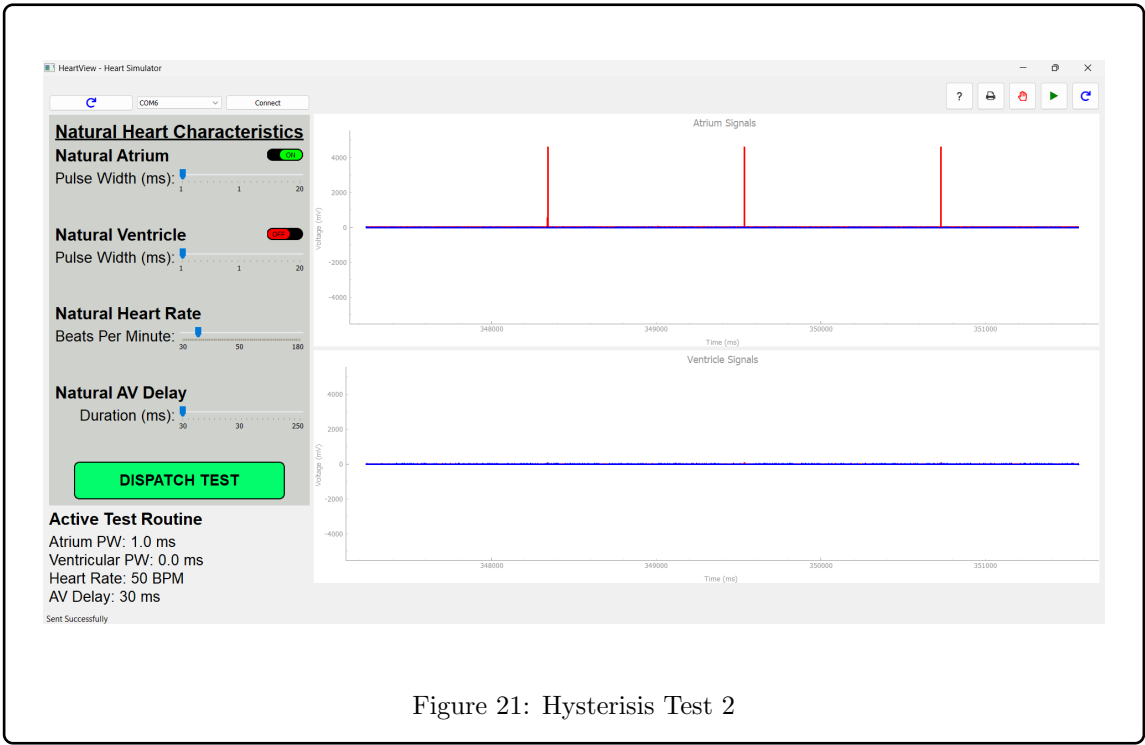
Figure 20: Hysterisis Test 2
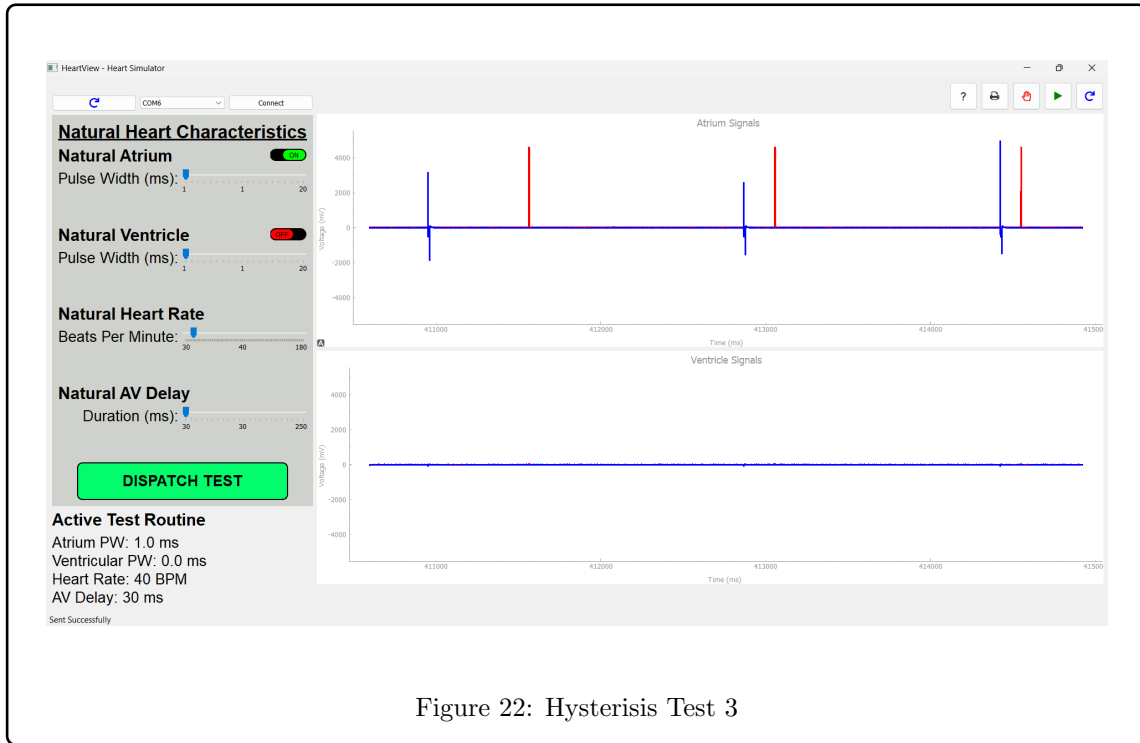
Figure 21: Hysterisis Test 2

Figure 22: Hysterisis Test 3

For instance, on the DCM side, you should test registration and login, parameter input validation, and mode selection and data storage/retrieval. This is not a complete list, depending on your system, you will need to test other components.

4.5 GenAI Usage

Provide a summary of any usage of GenAI in developing the model, DCM or writing this section. If you did not use GenAI tools at all, state that.

5 General Notes

- This is a general outline based on the Deliverable 1 handout. You should make sure everything listed in the handout it is included.

- Use screenshots of Simulink diagrams and DCM interface where appropriate.

- Ensure the requirements are traceable to design and test cases.

- Be concise and make things clear.

- You can add other sections, and you can also decide not to use this structure, however, I am including the main general sections we will expect to see.