SE 3K04
Dr. Bokhari
Group 10
October 25, 2020

SE 3K04: Assignment 1
Pacemaker Development

Hamza Ashraf
Sumer Karir
Abigail Nevo
Graham Power
Stephanie Ralph

Revision History

Rev.	Description	Date (Y-M-D)
P1	Preliminary internal release for October 24, 2020, Internal Team Review. **Authors:** Hamza Ashraf, Sumer Karir, Abigail Nevo, Graham Power, Stephanie Ralph	2020-10-15
A	External document release for November 1, 2020, Assignment Deadline **Authors:** Hamza Ashraf, Sumer Karir, Abigail Nevo, Graham Power, Stephanie Ralph	2020-11-01

Related Documents

Designator	Name	Revision
[1]	3K04-Pacemaker DCM (Group 10) HTML	-
[2]	3K04-Pacemaker DCM Testing (Group 10) HTML	-

Contents

On Academic Honesty

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by:

[Hamza Ashraf, ashrah4, 400201553]:

[Sumer Karir, karirs, 400186680]: Sumer K

[Abigail Nevo, nevoa, 400170276]:

[Graham Power, powerg, 400198753]:

[Stephanie Ralph, ralphs, 400171292]: Steffi Ralph

Introduction

A pacemaker is a device used to pace, sense, and provide electrical response to the heart where needed. It generates electrical impulses delivered through electrodes to make the heart chambers contract and pump blood. This device in turn regulates the electrical conduction system of the heart to maintain a safe and healthy body.
We were instructed to design and develop the basis for our pacemaker using Simulink and any trouble shooting was done on Heartview to make sure that the stateflow diagrams were operating correctly.

A Device Controller-Monitor (DCM) Is a type of Graphical User Interface (GUI) capable of displaying information about the device it controls, as well as communicating with that device to pass control instructions. For this project, the DCM needs to store relevant pacemaker parameters for up to ten registered users, and allow logged in users to view and modify these parameters.

Planning and Research

Pacemaker

Prior to any coding and development, research was key to a successful design. Understanding all of the documents and making notes of the key takeaways was crucial from the starting. This made referencing and implementation of the design a lot easier. Once we understood what was asked of us, we prepared some requirements and specifications, in the following section, to have some clear tasks which needed to be completed. Assignment 1 was then divided amongst the group where three members would work on the pacemaker design and two on the DCM.
In the following sections, we go over what our initial design and thought process was when we were starting out and how it evolved as we progressed throughout this project.
We came up with a high level flowchart to provide an understanding of how our pacemaker would work and where the different segments we worked on would fit.

Device Controller-Monitor

Given the lenience in the DCM Requirement Specifications as to what technologies are to be used, all options must be considered before choosing an implementation to ensure the chosen technologies allow all requirements to be met. To assist in choosing these technologies, the DCM was split into three categories; GUI display, data management, and serial communication. Possible implementations of each category were compared and scored to determine their implementations.

Table #1: GUI Display Technology Comparison

Technology	**Supported Implementation Languages**	**Comments**	**Overall Score (/10)**
Tkinter	Python	● Language both developers are comfortable with ● Only allows for basic GUI's (not visually pleasing)	4
Web (HTML/CSS)	Python - Flask JavaScript Java - Spring MVC	● Allows for modern GUI design (visually pleasing) ● Using flask allow developers to use a language both are comfortable with ● Easily scalable	8
Swing	Java	● Platform agnostic ● Uses MVC paradigm, allowing for a more flexible UI ● Developers are not a string in Java as Python	5
Qt	C++	● Qt webkit allows for JavaScript to be connected to C++ code ● Comes with a GUI designer ● DB integration will be harder in C++ than Python	7

Table #2: Data Management Technology Comparison

Technology	Supported Implementation Languages	Comments	Overall Score (/10)
MongoDB	Python - MongoPy JavaScript - Mongoose	• Non-relational database with excellent documentation and an easy learning curve. • Supporting packages allow for convenient integration with Flask applications. • Developers have more experience using Mongo	5
Firebase	JavaScript - Firebase SDK	• Non-relational database with excellent documentation. • Requires registering an account with Google Apps alongside payment information.	2
SQL	Postgres Sqlite3 - Python	• Relational database seems a better data representation for the data we need to store in this project • Using sqlite3 allows for direct integration with a Python project	7

Table #3: GUI Serial Communication Comparison

Technology	Supported Implementation Languages	Comments	Overall Score (/10)
Node.js SerialPort	JavaScript - Node.js	• Allows access to serial ports • Not widely used - issues are not well-documented on programming forums such as Stack Overflow.	4
PySerial	Python	• Popular python library with long term support and lots of documentation • Very easy to implement	10

Using the scores from the above tables, the technologies to use in DCM development can be chosen. The highest scoring implementation from every table shared the Python programming language in common. So the technologies that will be used to create the DCM will be; Python, Flask, SQLite3, PySerial.

Requirements and Specifications

Here we have listed some of the key tasks which our design must accomplish and under which restraints it must operate under.

Table #4: Implementation Requirements and Specifications

<table>
<tr><th colspan="2">Pacemaker Implementation</th></tr>
<tr><th>Requirements</th><th>Specifications</th></tr>
<tr><td>- Pacemaker must be able to emit an electrical stimuli which can pulse the heart safely
- Pacing modes to implement:
    - AOO
    - VOO
    - AAI
    - VVI
- The ‘heart’ and ‘pacemaker’ hardware must be able to communicate with each other and provide the appropriate output where needed.</td><td>- Pacing modes implemented must be precise in design and follow all safety guidelines (size of impulse, frequency of impulse, etc.)
- Pacemaker modes must be designed using Simulink software and downloaded to NXP FRDM-K64F hardware where design is tested</td></tr>
<tr><th colspan="2">DCM Implementation</th></tr>
<tr><th>Requirements</th><th>Specifications</th></tr>
<tr><td>- The user interface shall be capable of utilizing and managing windows for display of text and graphics
- The user interface shall be capable of processing user positioning and input buttons
- The user interface shall be capable of displaying all programmable parameters for review and modification
- The user interface shall be capable of visually indicating when the DCM and the device are communicating
- The user interface shall be capable of visually indicating when a different pacemaker device is approached than was previously interrogated</td><td>- The DCM shall include a welcome screen, including the ability to register a new user and log in as an existing user
- Develop interfaces to present all of the pacing modes mentioned in ‘Pacemaker Implementation’
- Make provisions for storing programmable parameter data and checking inputs</td></tr>
</table>

Design and Specification

After understanding the requirements and specifications and having the appropriate amount of research done, we believed we were ready to begin designing the different components of the pacemaker.

Pacemaker Design

Understanding the anatomy of the heart and it's electrical conduction system was a key starting area, as we had to understand where the electrical stimuli would be applied. We also had to learn of the different pacing modes and what their particular specifications are so that they can be programmed accordingly. For assignment 1 we were required to implement 4 different modes which were AOO, VOO, AAI, and VVI. Each has a distinctly different operation which it conducts, as described below.

AOO:
- Atrial pacing, no sensing, atrial asynchronous pacing at lower programmed pacing rate

VOO:
- Ventricular pacing, ventricular sensing, sensed intrinsic QRS inhibits ventricular pacing

AAI:
- Atrial pacing, atrial sensing, intrinsic P wave inhibits atrial pacing

VVI:
- Ventricular pacing, ventricular sensing, sensed intrinsic QRS inhibits ventricular pacing

DCM Design

To aid in the development of the DCM, it was split into four different modules. Each module was designed to be independent of one another and handle different tasks for the DCM. The description of the four modules are the following. The main module or the FlaskApp was responsible for the rendering of the DCM and the state machine implementation. Its secret was the state machine that allowed for the transition between GUI states. The User module was responsible for user login and account creation as well as managing the users pacemaker parameters. Its secret was the users ability to log in/out and modify its own parameters. The database module was responsible for storing the required data in a database and providing quick and easy access to the database's contents. Its secret was its ability to create, access, and modify a single file database. Finally the configuration manager was responsible for loading the applications configuration and providing the configuration variables to all other modules, as well as creating and managing a logger to allow every module to log their actions for debugging purposes. Its secret was its ability to read custom configuration files and log all DCM actions to a log file. Each module, and their corresponding capabilities are also outlined in the diagram below. For a full description of all code implementations please refer to the Code Documentation [1].

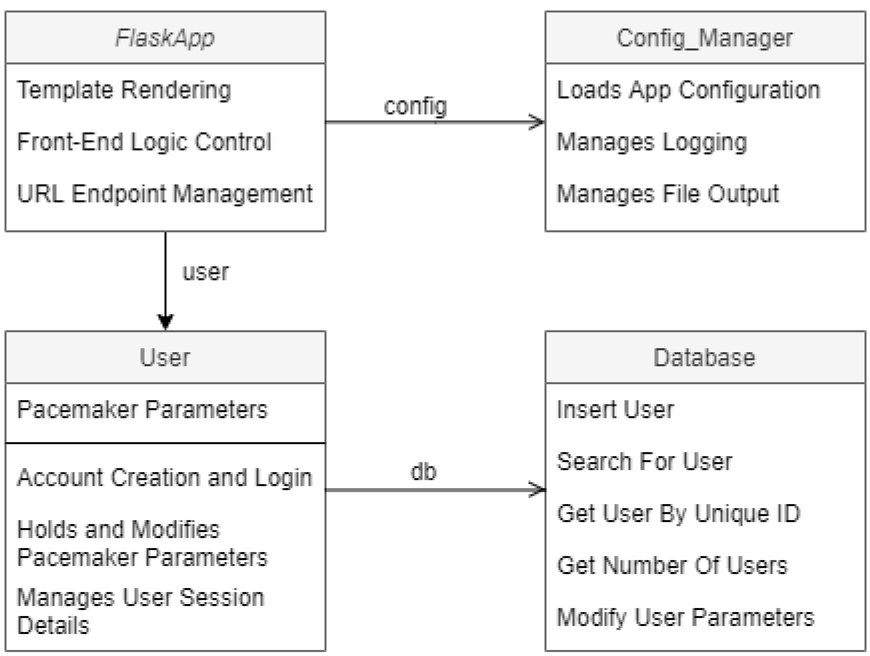


Figure #1: DCM Modular Design

Monitored Variables and Controlled Quantities

AOO

Table #5: AOO Controlled Quantities

Name	Function	Initialization Values	Limitations
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	Amp = 100	Is set to a constant value to make sure that the pacemaker is functioning under all conditions.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Is set to low if the pace control (pin D8) is high, meaning that a pulse is being emitted to the atrium
ATR_PACE_CTRL_D8	Discharges the primary capacitor through the atrium	0	Set to low in charging state, and high in pacing state. This indicates when the capacitors charge will be released to emit a pulse.
ATR_GND_CTRL_D11	Stops charge buildup during primary capacitor discharge	1	Capacitor has a charge limit.
PACE_GND_CTRL_D10	Controls current flow to the tip of the atrium from the ring	0	Set to low in charging state, and high in pacing state to prevent charge overflow to heart.
Z_ATR_CTRL_D4	Controls and analyzes impedance at the atrial electrode, and the connection between the atrial electrodes and the atrium	0	Always kept to low to prevent the pacemaker circuit from short circuiting.
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000ms/45 (45 BPM)	Inputted into larger state flow. Dictates transition timing between charging and pacing state.

PulseWidth	Length of pulse, how long the capacitor is discharging in msec	20	

VOO

Table #6: VOO Controlled Quantities

Name	Function	Initialization Values	Limitations
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Is set to high in charging and low in pacing. This prevents the capacitor from charging while it is emitting the charge as a pulse to the ventricle.
VENT_PACE_CTRL_D9	Discharges the primary capacitor through the ventricle	0	Set to low in charging and high in pacing state. This allows the capacitor to build up sufficient charge.
VENT_GND_CTRL_D12	Stops charge buildup during primary capacitor discharge	1	Set to high in charging, and low in pacing state. Prevents over charging of capacitor.
PACE_GND_CTRL_D10	Controls current flow to the tip of the ventricle from the ring	0	Set to low in charging and high in pacing. Prevents the ventricle from receiving too much charge from the pulse.
Z_VENT_CTRL_D7	Connects the impedance circuit and the ventricle ring electrode	0	Always set to low to keep the circuit closed. Prevents it from short circuiting.
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/60 (60BPM)	Inputted into larger state flow. Dictates transition timing between charging and pacing state.

PulseWidth	Length of pulse, how long the capacitor is discharging in msec	20	

AAI

Table #7: AAI Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
ATR_CMP_DETECT_D0	Atrium Sensing	1	
PushButton	Pushed outputs true to inhibit the pulse Not pushed outputs false and the pace is not inhibited	0	
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.
PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Cannot be high if ATR_PACE_CTRL_D8 is high or the patient's atrium could be connected directly to the PWM signal
ATR_PACE_CTRL_D8	Discharges the primary capacitor through the atrium	0	Cannot be set to high or the patient's atrium could be connected directly to the PWM signal
ATR_GND_CTRL_D11	Stops charge buildup during primary capacitor discharge	1	Capacitor has a charge limit.
PACE_GND_CTRL_D10	Controls current flow to the tip of the atrium from the ring	0	Has to be high as it controls the switch which follows the tip
Z_ATR_CTRL_D4	Controls and analyzes impedance at the atrial electrode, and the connection between the atrial electrodes and the atrium	0	Always kept to low to prevent the pacemaker circuit from short circuiting.
FRONTEND_CTRL_D13	Starts sensing circuitry	1	

ART_CMP_REF_PWM_D6	Limit for when atrial action potential should be sensed	S_Amp	
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/45	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Constant, represents length of pulse, how long the capacitor is discharging in msec	20	
P_Amp	Constant, represents the pacing PWM output that controls the amplitude of the pace	100	0 to 100
S_Amp	Constant, represents the sensing PWM threshold in the sensing circuit	80	0 to 100
ARP	Constant, represents the Atrial Refractory Period in msec	150	

VVI

Table #8: VVI Monitored Variable and Controlled Quantities

Name	Function	Initialization Values	Limitations
Monitored Variables			
VENT_CMP_DETECT_D1	Ventricular Sensing	1	
PushButton	Pushed outputs true to inhibit the pulse Not pushed outputs false and the pace is not inhibited	0	
Controlled Quantities			
PACING_REF_PWM_D5	Charges Primary Capacitor	P_Amp	Set to a constant value to ensure the pacemaker is functioning.

PACE_CHARGE_CTRL_D2	Starts and Stops PACING_REF_PWM_D5 ie. the charging of the capacitor	1	Cannot be high if VENT_PACE_CTRL_D9 is high or the patient's ventricle could be connected directly to the PWM signal
VENT_PACE_CTRL_D9	Discharges the primary capacitor through the ventricle	0	Cannot be set to high or the patient's ventricle could be connected directly to the PWM signal
VENT_GND_CTRL_D12	Stops charge buildup during primary capacitor discharge	1	Set to high in charging, and low in pacing state. Prevents over charging of capacitor.
PACE_GND_CTRL_D10	Controls current flow to the tip of the ventricle from the ring	0	Has to be high as it controls the switch which follows the tip
Z_VENT_CTRL_D7	Connects the impedance circuit and the ventricle ring electrode	0	Always set to low to keep the circuit closed. Prevents it from short circuiting.
FRONTEND_CTRL_D13	Starts sensing circuitry	1	
VENT_CMP_REF_PWM_D3	Establishes the ventricular action threshold for when sensing should occur	S_Amp	
PacePeriod	Converts BMP to total time from end of one pulse to the end of another pulse in msec	60000/60 = 1000	Inputted into larger state flow. Dictates transition timing between charging and pacing state.
PulseWidth	Constant, represents length of pulse, how long the capacitor is discharging in msec	20	
VRP	Constant, represents time interval, to signify a length of time after an ventricular event, when ventricular sensing will not inhibit or trigger pacing in msec	150	

P_Amp	Constant, represents the pacing PWM output that controls the amplitude of the pace	100	0 to 100
S_Amp	Constant, represents the sensing PWM threshold in the sensing circuit	80	0 to 100

The LED States were used purely for testing but are not connected to the functionality of the pacemaker so they are neither monitored variables or controlled quantities.

Decision Tables

Pacemaker

AOO & VOO

Table #9: AOO Decision Table

	Pacing	ChargingAndSensing
after(PacingPeriod-Pulse Width): {0,1}	1	*
after(PulseWidth): {0,1}	*	1

AAI

Table #10: AAI Decision Table

	Pacing	ChargingAnd Sensing	ChargingAnd Sensing	ChargingAnd Sensing
ATR_CMP_DETECT: {0,1}	0	1	*	*
PushButton: {0,1}	0	*	1	*
after(PacingPeriod-Pulse Width): {0,1}	1	*	*	*
after(PulseWidth): {0,1}	*	*	*	1

VVI

Table #11: VVI Decision Table

	Pacing	ChargingAnd Sensing	ChargingAnd Sensing	ChargingAnd Sensing
VENT_CMP_DETECT: {0,1}	0	1	*	*
PushButton: {0,1}	0	*	1	*
after(PacingPeriod-Pulse Width): {0,1}	1	*	*	*
after(PulseWidth): {0,1}	*	*	*	1

DCM

Table #12: User Login / Account Creation

Conditions:					
Login Attempt	1	1	0	0	0
Account Creation	0	0	1	1	1
Matching Credentials In Database	1	0	1	0	0
Max Users In Database	*	*	*	1	0
Actions:					
Login Granted	1	0	0	0	1
New Account Created	0	0	0	0	1

Table #13: Attempted Change Of Pacemaker Parameters

Conditions:								
VOO Mode	1	1	0	0	0	0	0	0
VVI Mode	0	0	1	1	0	0	0	0
AOO Mode	0	0	0	0	1	1	0	0
AAI Mode	0	0	0	0	0	0	1	1
VOO Parameter Changed	1	0	1	0	1	0	1	0
VVI Parameter Changed	*	*	*	*	*	*	*	*
AOO Parameter Changed	*	*	*	*	*	*	*	*
AAI Parameter Changed	*	*	*	*	*	*	*	*
Action:								
Change Allowed	1	0	1	0	1	0	1	0

States

Pacemaker

Stateflow Chart

AOO

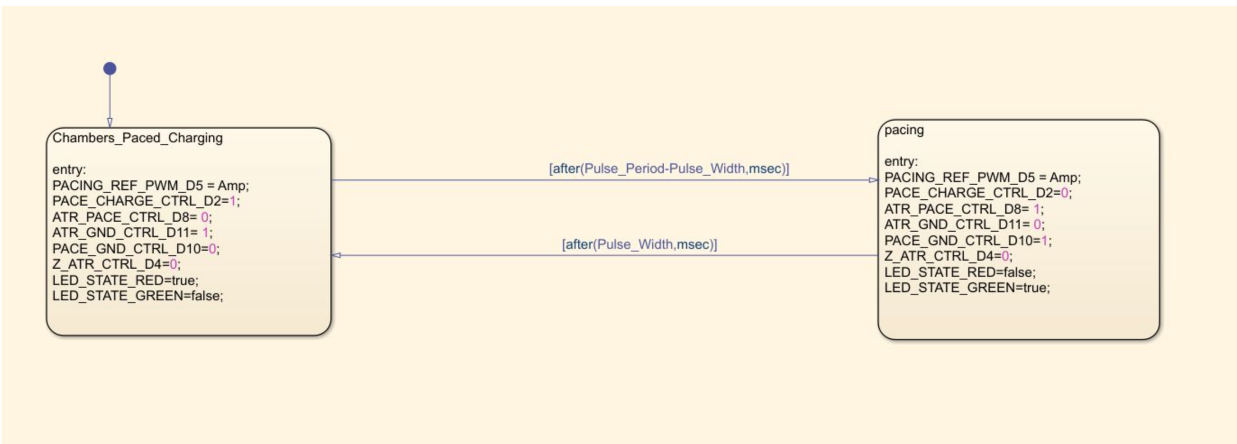


Figure #2: AOO Stateflow Chart

VOO

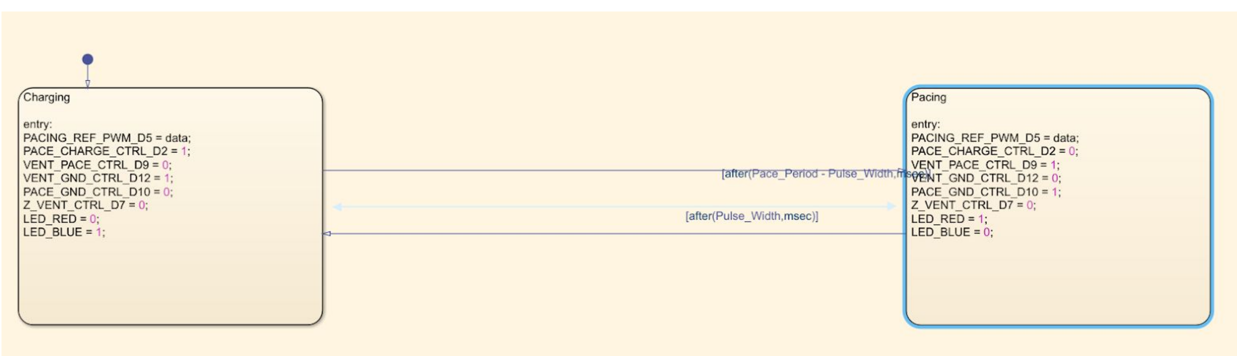


Figure #3: VOO Stateflow Chart

AAI

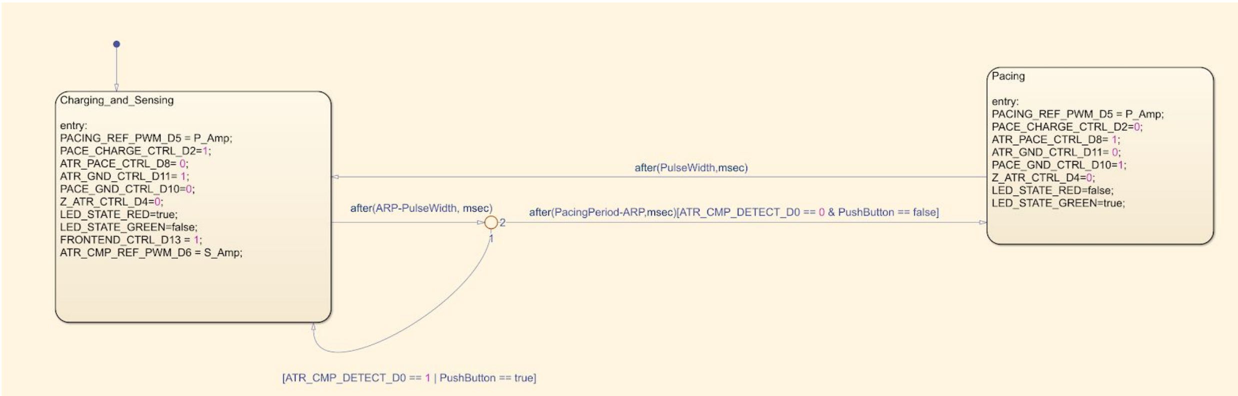


Figure #4: AAI Stateflow Chart

VVI

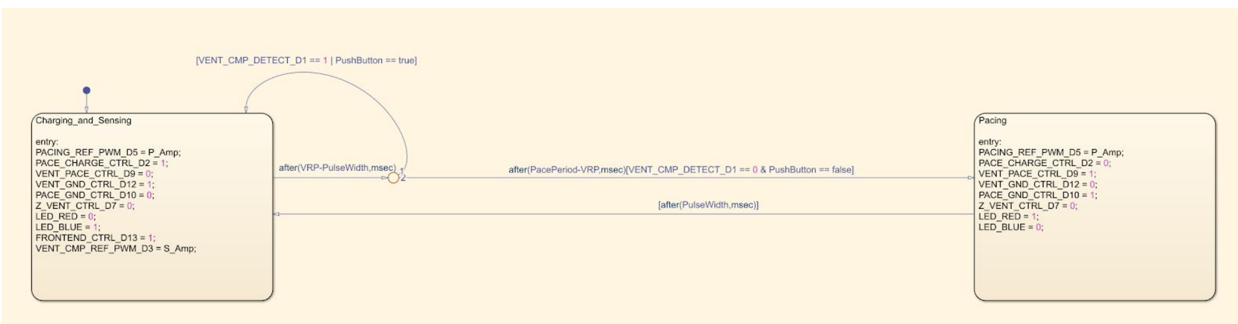


Figure #5: VVI Stateflow Chart

State Diagram

AOO

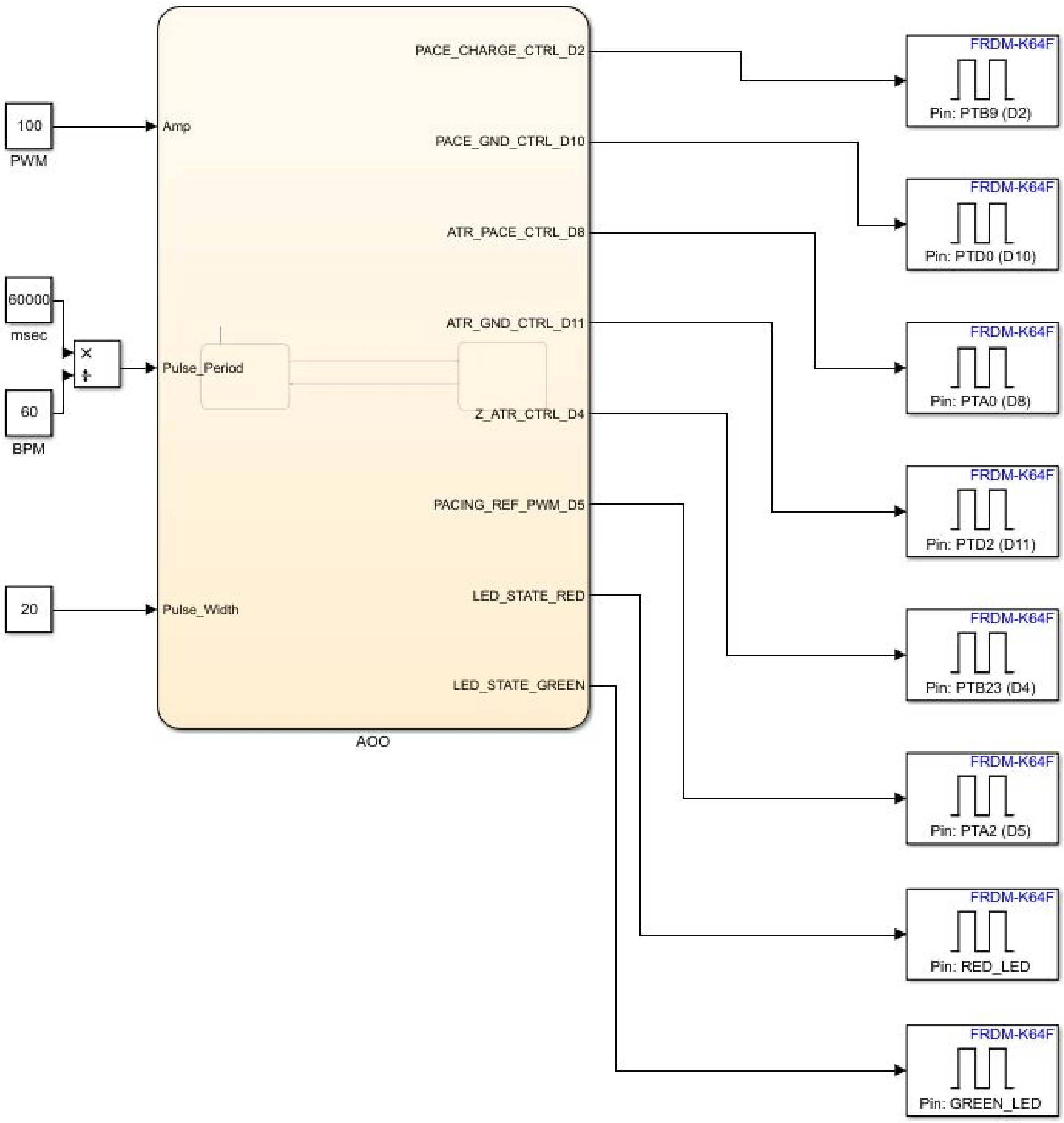


Figure #6: AOO State Diagram

VOO

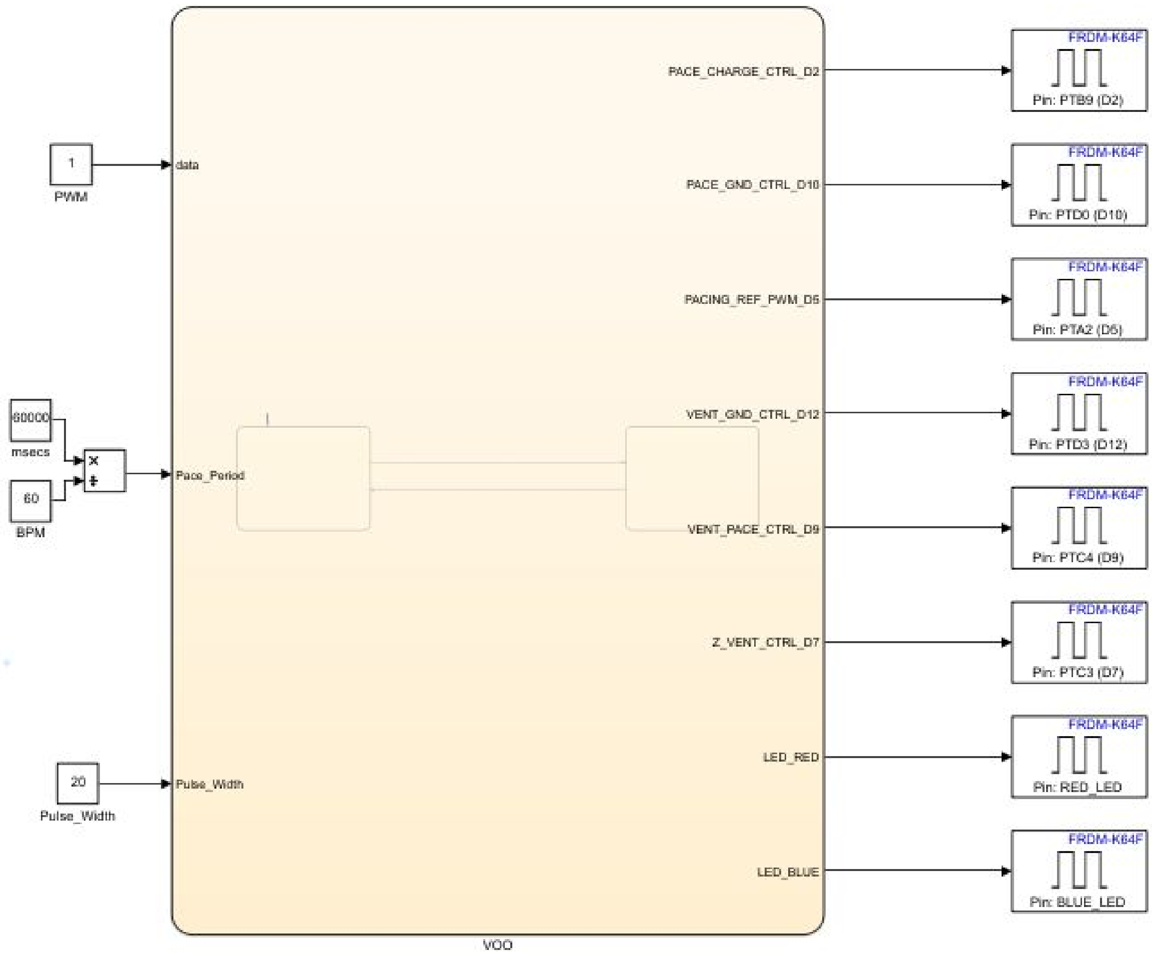


Figure #7: VOO State Diagram

AAI

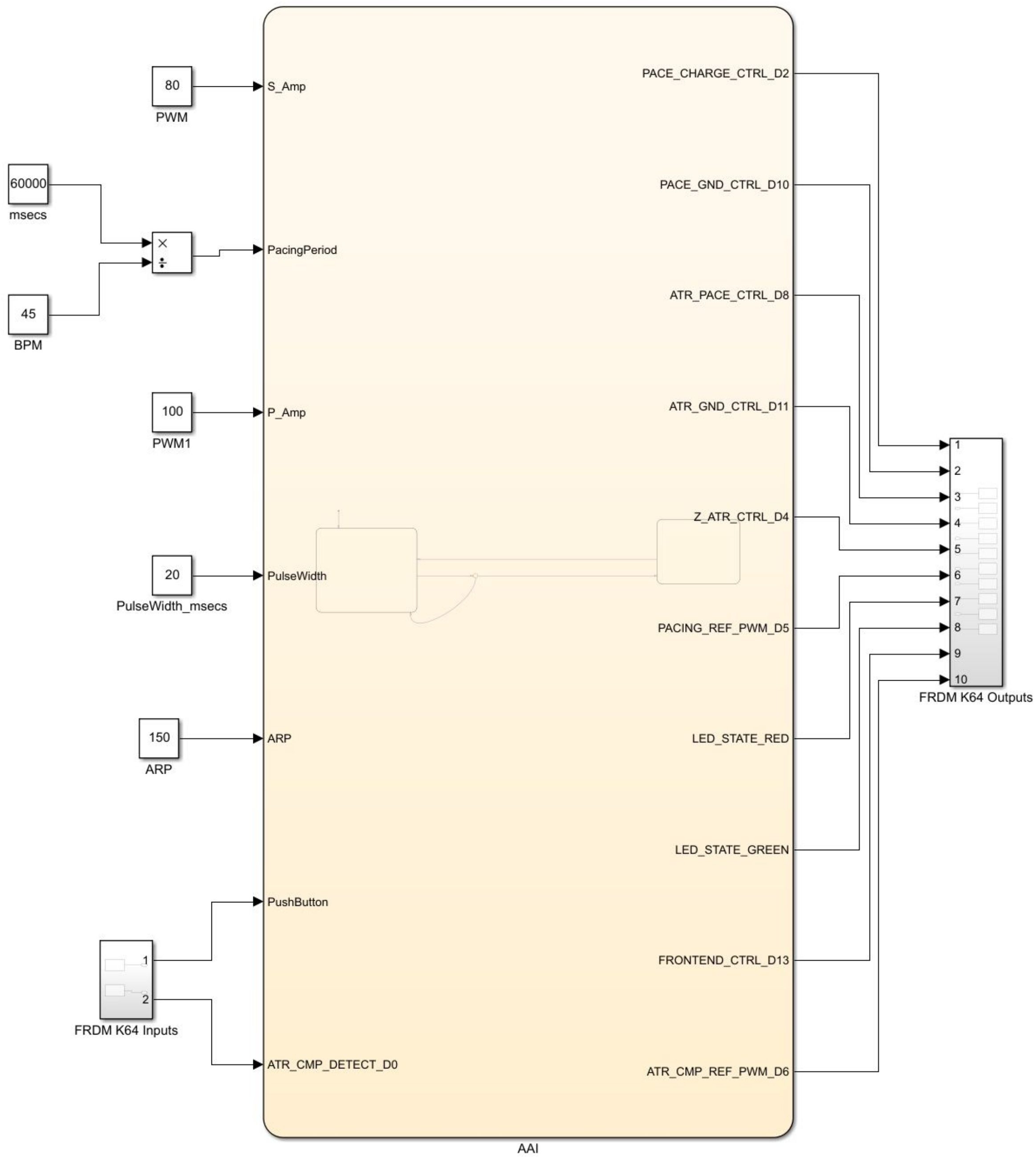


Figure #8: AAI State Diagram

VVI

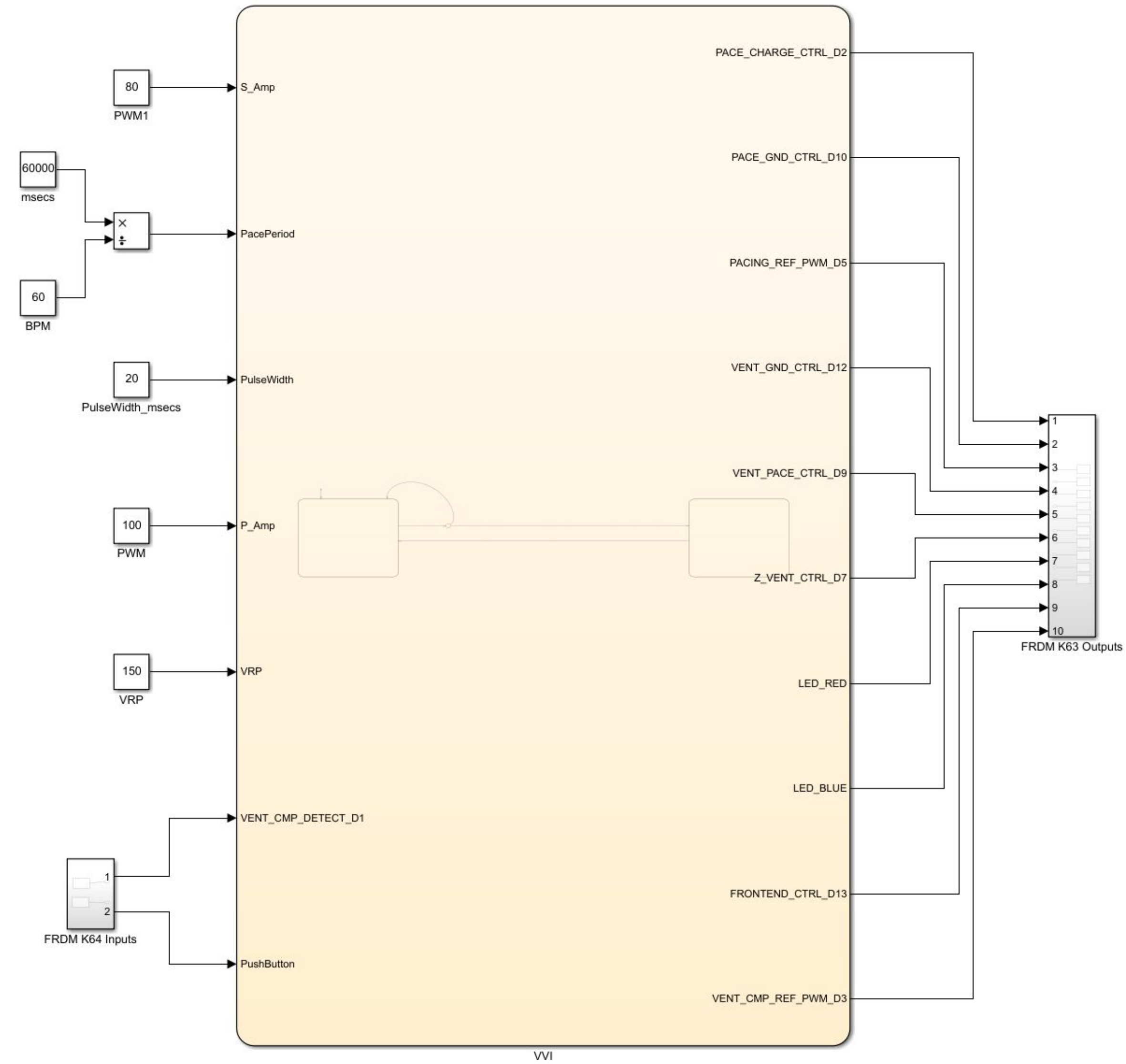


Figure #9: VVI State Diagram

State Transition Table

AOO

Table #14: AOO State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>Charging_and_Sensing
(Default state)</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>1</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>1</td><td>0</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr>
<tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr>
<tr><td>LED_STATE_RED</td><td>true</td><td>false</td></tr>
<tr><td>LED_STATE_GREEN</td><td>false</td><td>true</td></tr>
</table>
</td><td>Capacitor is charged (next state charge from capacitor is released to emit pulse)</td></tr>
<tr><td>Pacing</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>1</td><td>0</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>1</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>0</td></tr>
<tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr>
<tr><td>LED_STATE_RED</td><td>false</td><td>true</td></tr>
</table>
</td><td>Pulse is emitted (once the charge is released, it returns back to charging state)</td></tr>
</table>

	LED_STATE_GREEN	true	false	

VOO

Table #15: VOO State Transition Table

Current State	Variable	Value for Current State	Value for Next State	Output
Charging_and_Sensing (Default state)	PACE_CHARGE_CTRL_D2	1	0	Capacitor is charged (next state charge from capacitor is released to emit pulse)
	VENT_PACE_CTRL_D9	0	1	
	VENT_GND_CTRL_D12	1	0	
	PACE_GND_CTRL_D10	0	1	
	Z_VENT_CTRL_D7	0	0	
	LED_RED	false	true	
	LED_BLUE	true	false	
Pacing	PACE_CHARGE_CTRL_D2	0	1	Pulse is emitted (once the charge is released, it returns back to charging state)
	VENT_PACE_CTRL_D9	1	0	
	VENT_GND_CTRL_D12	0	1	
	PACE_GND_CTRL_D10	1	0	

<table>
<tr><td></td><td>
<table>
<tr><td>Z_VENT_CTRL_D7</td><td>0</td><td>0</td></tr>
<tr><td>LED_RED</td><td>false</td><td>true</td></tr>
<tr><td>LED_BLUE</td><td>true</td><td>false</td></tr>
</table>
</td><td></td></tr>
</table>

AAI

Table #16: AAI State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>Charging_and_Sensing
(Default state)</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>0</td><td>1</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>1</td><td>0</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr>
<tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr>
<tr><td>ATR_CMP_DETECT_D0</td><td colspan="2">Will only move to the next state (pacing) if it remains 0 after PacingPeriod-PulseWidth amount of time (in msec)</td></tr>
<tr><td>PushButton</td><td colspan="2">Will only move to the next state (pacing) if is still false after PacingPeriod-PulseWidth amount of time (in msec)</td></tr>
</table>
</td><td>Charging the Capacitor</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td>Pacing</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>0</td><td>1</td></tr>
<tr><td>ATR_PACE_CTRL_D8</td><td>1</td><td>0</td></tr>
<tr><td>ATR_GND_CTRL_D11</td><td>0</td><td>1</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>1</td><td>0</td></tr>
<tr><td>Z_ATR_CTRL_D4</td><td>0</td><td>0</td></tr>
</table>
</td><td>Setting the Pace

(next state is charging_and_sensing)</td></tr>
</table>

These are the only two states and they will move back and forth between then if the conditions are met.

VVI

Table #17: VVI State Transition Table

<table>
<tr><th>Current State</th><th>Variables to get to Next State</th><th>Output</th></tr>
<tr><td>Charging_and_Sensing (Default state)</td><td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr>
<tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>1</td></tr>
<tr><td>VENT_GND_CTRL_D12</td><td>1</td><td>0</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr>
<tr><td>Z_VENT_CTRL_D13</td><td>0</td><td>0</td></tr>
<tr><td>VENT_CMP_DETECT_D0</td><td colspan="2">Will only move to the next state if it is</td></tr>
</table>
</td><td>Charging the Capacitor</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td></td><td>0</td></tr>
<tr><td>PushButton</td><td>Will only move to the next state if is still false after period of time</td></tr>
</table>
</td>
<td></td>
</tr>
<tr>
<td>Pacing</td>
<td>
<table>
<tr><th>Variable</th><th>Value for Current State</th><th>Value for Next State</th></tr>
<tr><td>PACE_CHARGE_CTRL_D2</td><td>1</td><td>0</td></tr>
<tr><td>VENT_PACE_CTRL_D9</td><td>0</td><td>1</td></tr>
<tr><td>VENT_GND_CTRL_D12</td><td>1</td><td>0</td></tr>
<tr><td>PACE_GND_CTRL_D10</td><td>0</td><td>1</td></tr>
<tr><td>Z_VENT_CTRL_D13</td><td>0</td><td>0</td></tr>
</table>
</td>
<td>Sending the Pace

(next state is charging_and_sensing)</td>
</tr>
</table>

These are the only two states and they will move back and forth between then if the conditions are met.

Simulink Stateflow Tabular Expression

AOO

Table #18: AOO Stateflow Tabular Expression

Source State	Event	Conditions	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
Chambers_Paced_Charging	TRUE	after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	Pacing	PACING_REF_PWM_D5 = Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false;
		~after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	No Change- Chambers_Paced_Charging	Remain in present state until the condition is satisfied.
Pacing	TRUE	after(Pulse_Width,msec)	none	none	none	none	Chamber_Paced_Charging	PACING_REF_PWM_D5 = Amp; PACE_CHARGE_CTRL_D2=0; ATR_PACE_CTRL_D8= 1; ATR_GND_CTRL_D11= 0; PACE_GND_CTRL_D10=1; Z_ATR_CTRL_D4=0; LED_STATE_RED=false; LED_STATE_GREEN=true;
		~after(Pulse_Width,msec)	none	none	none	none	No Change- Pacing	Remain in present state until the condition is satisfied.

VOO

Table #19: VOO Stateflow Tabular Expression

Source State	Event	Conditions	During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
Charging	TRUE	after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	Pacing	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2 = 1; VENT_PACE_CTRL_D9 = 0; VENT_GND_CTRL_D12 = 1; PACE_GND_CTRL_D10 = 0; Z_VENT_CTRL_D7 = 0; LED_RED = false; LED_BLUE = true;
		~after(Pulse_Period - Pulse_Width ,msec)	none	none	none	none	No Change-Charging	Remain in present state until the condition is satisfied.
Pacing	TRUE	after(Pulse_Width,msec)	none	none	none	none	Charging	PACING_REF_PWM_D5 = data; PACE_CHARGE_CTRL_D2 = 0; VENT_PACE_CTRL_D9 = 1; VENT_GND_CTRL_D12 = 0; PACE_GND_CTRL_D10 = 1; Z_VENT_CTRL_D7 = 0; LED_RED = true; LED_BLUE = false;
		~after(Pulse_Width,msec)	none	none	none	none	No Change-Pacing	Remain in present state until the condition is satisfied.

AAI

Table #20: AAI Stateflow Tabular Expression

Source State	Event	Condition		During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Charging_and_ Sensing	TRUE	after(ARP-PulseWidth,msec)	ATR_CMP_DETECT_ D0 == 1 OR PushButton == true	none	none	none	none	Chargin_and_ Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
			after(PacingPeriod-ARP,msec) AND ATR_CMP_DETECT_ D0 == 0 AND PushButton == false]	none	none	none	none	Pacing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=0; ATR_PACE_CTRL_D8= 1; ATR_GND_CTRL_D11= 0; PACE_GND_CTRL_D10=1; Z_ATR_CTRL_D4=0; LED_STATE_RED=false; LED_STATE_GREEN=true;
		~after(ARP-PulseWidth,msec)		none	none	none	none	No change -- Charging_and _Sensing	[have already occurred]
State == Pacing	TRUE	after(PulseWidth, msec)		none	none	none	none	Charging_and _Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; ATR_PACE_CTRL_D8= 0; ATR_GND_CTRL_D11= 1; PACE_GND_CTRL_D10=0; Z_ATR_CTRL_D4=0; LED_STATE_RED=true; LED_STATE_GREEN=false; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
		~after(PulseWidth,msec)		none	none	none	none	No change -- Pacing	[have already occurred]

VVI

Table #21: VVI Stateflow Tabular Expression

Source State	Event	Condition		During Actions	Condition Actions	Exit Actions	Transition Actions	Destination State	Entry Actions
State == Charging_and_Sensing	TRUE	after(VRP-PulseWidth,msec)	VENR_CMP_DETECT_D1 == 1 OR PushButton == true	none	none	none	none	Chargin_and_Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; VENT_PACE_CTRL_D9= 0; VENT_GND_CTRL_D12= 1; PACE_GND_CTRL_D10=0; Z_VENT_CTRL_D7=0; LED_STATE_RED=0; LED_STATE_GREEN=1; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
			after(PacingPeriod-ARP,msec) AND ATR_CMP_DETECT_D0 == 0 AND PushButton == false]	none	none	none	none	Pacing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=0; VENT_PACE_CTRL_D9= 1; VENT_GND_CTRL_D12= 0; PACE_GND_CTRL_D10=1; Z_VENT_CTRL_D7=0; LED_STATE_RED=1; LED_STATE_GREEN=0;
		~after(VRP-PulseWidth,msec)		none	none	none	none	No change -- Charging_and_Sensing	[have already occurred]
State == Pacing	TRUE	after(PulseWidth, msec)		none	none	none	none	Charging_and_Sensing	PACING_REF_PWM_D5 = P_Amp; PACE_CHARGE_CTRL_D2=1; VENT_PACE_CTRL_D9= 0; VENT_GND_CTRL_D12= 1; PACE_GND_CTRL_D10=0; Z_VENT_CTRL_D7=0; LED_STATE_RED=0; LED_STATE_GREEN=1; FRONTEND_CTRL_D13 = 1; ATR_CMP_REF_PWM_D6 = S_Amp;
		~after(PulseWidth,msec)		none	none	none	none	No change -- Pacing	[have already occurred]

States Descriptions

AAI & VVI

Table #22: AAI and VVI State Descriptions

State	Function
Charging_and_Sensing	Charges the capacitor and is continuously sensing when in this state.
Pacing	Discharges the capacitor, sets the pace.

DCM

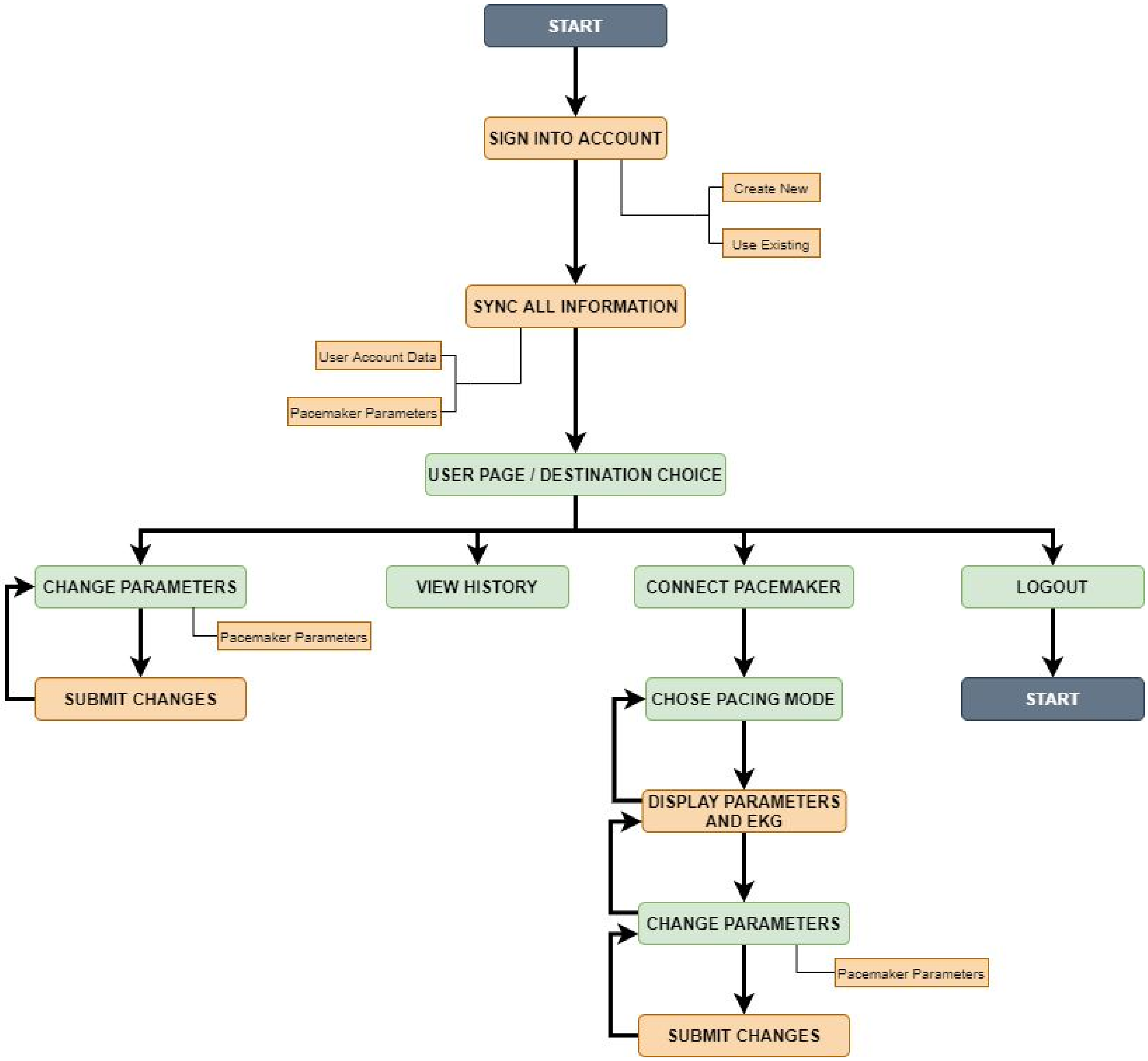


Figure #10: DCM State Transition Chart

Design Decisions

There was no change in information from the supplied information. Our design follows all laid out parameters. In the coming project alterations will be made in order to improve the design.

AOO, VOO, AAI & VVI

Table #23: AOO, VOO, AAI & VVI Design Decisions

Requirement being Changed	Resulting Change in Parameter + Assigned Values
PacePeriod	PacePeriod will be changed from a controlled quantity to a monitored variable, so that the input can be controlled from the user
BPM	Beats per minute have been restricted to be between 30 and 180. This is a restriction implemented due to heartview since these are the minimum and maximum BPM values.
PulseWidth	PulseWidth will be changed from a controlled quantity to a monitored variable, so that the input can be controlled from the user. The constant value is also restricted to be between 0ms and 20ms, so no quantity larger can be entered, otherwise design does not run.
P_Amp	P_Amp will be changed from a controlled quantity to a monitored variable, so that the amplitude of the pace can be controlled from the user
S_Amp	S_Amp will be changed from a controlled quantity to a monitored variable, so that the amplitude of the pace can be controlled from the user
Amp	PWM will be changed from a controlled quantity in AOO and VOO to manipulate the amplitude of the pace.
ARP	ARP will be changed from a controlled quantity to a monitored variable, so that the time interval after an atrial event can be controlled from the user
VRP	ARP will be changed from a controlled quantity to a monitored variable, so that the time interval after a ventricular event can be controlled from the user

Hardware Hiding

In order to implement hardware hiding, subsystems were made for the hardware inputs and outputs. These subsystems can be seen highlighted in the image below. These subsystems are essentially folders which contain all of the pins. They are named in consistently with the inputs and outputs from the board in order for the ease of understanding for the programmers, however the names could be altered as to further add to the hardware hiding.

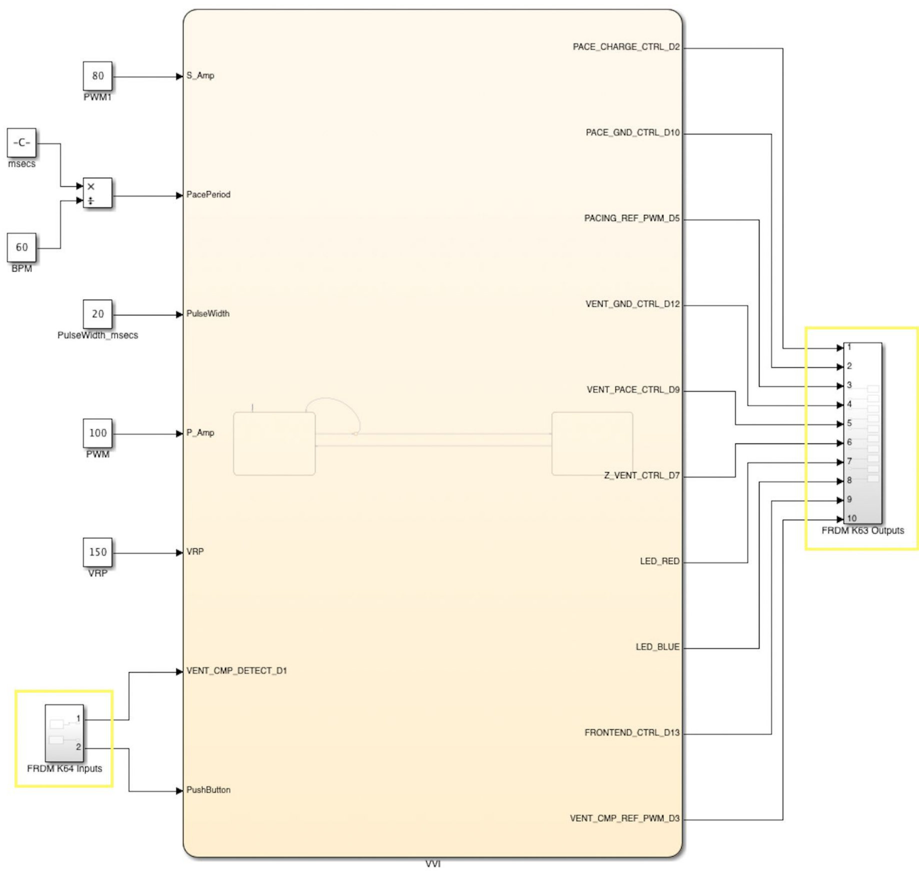
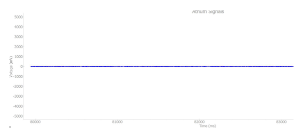
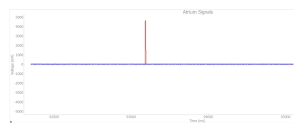
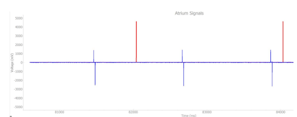
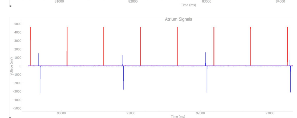
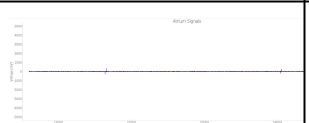
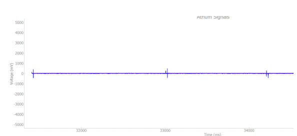
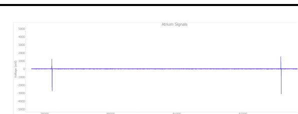


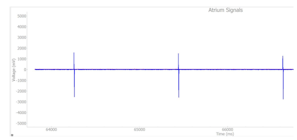
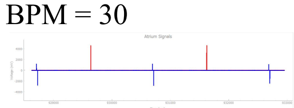
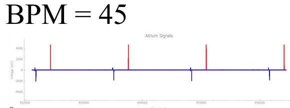
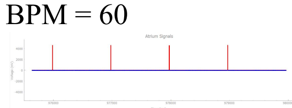
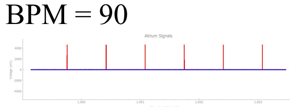
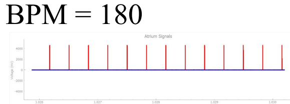
Figure #11: VVI State Diagram Highlighted for Hardware Hiding

Testing

Simulink

Table #24: Simulink Test Cases

Test Cases	Expected Outcome	Actual Outcome	Pass/Fail
Decrease S_Amp (S_Amp = 50)	Sensing threshold too low → pacemaker is too sensitive to signals from the heart, so will interpret even small signals as a pulse → will not send any paces	 	Pass
Increase S_Amp (S_Amp = 100)	Sensing threshold is too high → pacemaker will not register any electrical signal from the heart as a pulse → will continue to send paces no matter the heart's pulse	  	Pass
P_Amp = 10	Pace amplitude should be 90% (100-10) smaller than default		Pass
P_Amp = 50	Pace amplitude should be 50% (100-50) smaller than default		Pass
Push button many times quickly and consecutively	Inhibit pace until button is no longer being pressed & once pace period has passed (assuming no natural heartbeat)		Pass
Make PulseWidth longer than PacingPeriod	A pulse should not be emitted because it is not possible in reality. A restriction was put on Pulse Width variable so	Pulse Width = 1500ms Matlab Diagnostics: "Inconsistent numeric values for parameter 'Value' in 'AOO/Constant': Quantized parameter value (1500) is	Pass

	that it can only be between 0 and 20ms.	greater than maximum (20)”	
Shorten PulseWidth to 5 msec	Pulse width should become 5 ms (still within the boundary of 1-20 msec)		Pass
Vary BPM less than 30, greater than 180	Simulink should give an error since the BPM values not within 30 and 180 cannot be supported by heartview. Simulink prevents these settings from being registered to the pacemaker.	BPM = 20 Matlab Diagnostic: “Inconsistent numeric values for parameter 'Value' in 'AOO/BPM': Quantized parameter value (20) is less than minimum (30)” BPM = 190 “Inconsistent numeric values for parameter 'Value' in 'AOO/BPM': Quantized parameter value (190) is greater than maximum (180)”	Pass
BPM (less than, equal to and greater than heartbeat): - 30 - 45 - 60 - 90 - 180	Pacemaker is set to 60BPM. When the heart rate is less than 60BPM, pulses will be emitted by the pacemaker. When the heart rate is greater than 60BPM, pulses will be inhibited.	BPM = 30  BPM = 45  BPM = 60  BPM = 90  BPM = 180 	Pass

Python

Unit testing for python was completed using the PyTest library. Test results were output to both html, the generally accepted viewing method for python documentation and testing, and pdf for those unable to open and html document. To view these test results please refer to the DCM Testing document [2].