Assignment 2 - DCM

SFWRENG 3K04: Group 10

Rutvi Patel - 400260481
Arnav Arora - 400227929
Priya Patel - 400262251
Shiv Thakar - 400247588
Jil Shah - 400252316
Yuvraj Bal - 400247720

Table of Contents

1.0 Requirements Likely To Change

1.1 Assignment #1 Requirements

The requirements for the DCM design were with respect to 3.2.2 in PACEMAKER, where we had to design the essential aspects of the user interface. Our team decided to design the DCM in Python 3.8.9 using Python GUI and the tkinter library.

We were required to develop a user interface that includes a welcome screen that allows the user to login using an existing username and password. The UI should also have the ability to register new users and store them locally.

The user interface should also be able to using and managing windows for text and graphics, processing user positioning and input buttons, displaying all programmable parameters for review and modification, visually indicating when the DCM and the pacemaker device are communicating and lastly able to visually indicating when a different pacemaker device is approached/connected than was previously interrogated.

The feedback given in Assignment #1 Demo was implemented in this assignment

1. When the device gets out of range, the user interface shall be capable of visually indicating when telemetry is lost
2. The user interface shall be capable of visually indicating when telemetry is lost due to noise
3. Defining constraints and limits for the programmable parameters to match realistic conditions.
4. A communication between the DCM and Pacemaker
5. Save parameters for each user

1.2 Assignment #2 Requirements

The requirements for the DCM design for this assignment required us to expand the DCM to include all the required modes and parameters. The parameters also needed to be updated to reflect the specifications of the system.

This assignment required us to implement serial communication to transmit and receive information between the DCM and the Pacemaker device. The system is also able to set, store and transmit these parameters using serial communication and the use of the UART COM ports. The parameters are stored within a unique text file that is saved for each user, which can be opened at any time.

After submitting the parameters, the DCM is able to plot the electrocardiogram and display it to the user depending on what kind of graph was requested (Ventricle, Atrium or Both).

The requirements that were not implemented for Assignment 2 include the following:

1. Scaling the egrams to 0.0 to 5.0 Votls instead of the existing 0.0 to 1.0 Volts.

2.0 Design Decisions Likely to Change

Based on the feedback given in Assignment #1 Demo were implemented in this assignment:

- Using separate modules for each new window/section of the user interface
- Making the programmable and pacemaker parameters into one section on the main user window
- Create a section that outputs information to the doctor/user. This will include the parameters that have been set, the connection of the pacemaker, etc.
- All user information will be stored in separate text files to access information at another time.
- Print the parameters in the window so that it is accessible to the doctor/user.

The feedback that we received from Assignment #2 Demo were not implemented in this assignment, but will likely change our DCM design:

- Current e-gram plot has 0-1 V range , the data can be scaled to 0-5 V range
- Remove the parameters which are not used for a particular mode

3.0 Modules

The python module was designed to create a user interface for a Device Controlled-Machine (DCM) for a pacemaker. Currently in the design of our DCM we have only used one python module to hold all our functions, variables and statements, in the future we intend to create multiple modules to hold our program.

3.0.1 Modules

1. CommonFunctions
    - This module holds functions that are used commonly throughout other modules in the DCM design.
2. LoginData
    - This module is used to process data that is needed to run the LoginGUI modules. This includes reading the file of acceptable logins, or validating the username and password given. It also allows users to submit new usernames and passwords.
3. LoginGUI
    - This module controls all the different aspects of the GUI for the login menu, using python's Tkinter Library
4. MainUserData

- This module is used to process data that is submitted on the Main User Window. This module holds functions that can save and store all the parameters, check errors within the parameters and determine the indices of the parameters that are stored in python lists.

5. MainUserGUI
    - This modules controls all the different aspects of the GUI for the Main User Window, using python's Tkinter Library
6. Reports
    - This module returns printed reports in windows for the user to view
7. SerialCom
    - This module has all functions related to Serial Communication that is needed to interact with the Pacemaker Device and Simulink.
8. Util
    - This module contains functions that allow the user to open the utilities menu, using Buttons, and labels in the Tkinter library.
9. Main
    - This module is the main file that calls on functions from other modules to initiate the DCM and allow it to be started for the user.

3.1 Public Functions in Modules

3.1.1 CommonFunctions

- Libraries Used:
    - struct
- rgb_picker(r,g,b)
    - **Parameters:** int r, int g, int b
    - **Black Box Behaviour:** this function will take three integers representing red, green and blue and will convert them into the hexadecimal number for the colour needed from those values and returns a string.
    - **Internal Behaviour:** this function will take in the parameters, and find the hexadecimal conversion using the built-in hex() function. After doing this for all three values, they are concatenated into a string in the format " #RRGGBB" (RR = hex(r), GG = hex(g), BB = hex(b)). The string is converted to all uppercase using the upper() function and then returned.
- Plus(x,length)
    - **Parameters**: x,length
    - **Black Box Behaviour**:This function takes an integer x and length of array as input and returns new value of x which is incremented by 1
    - **Internal Behaviour**:This function increments x by 1 only if the current value of x is less than the length of the array. Essentially this function is designed to

increment through the array because the values of the pulse width for A and V are stored as doubles in array

- minus(x)
    - **Parameters**: x
    - **Black Box Behaviour**:This function takes an integer x as input and returns new value of x which is decremented by 1
    - **Internal Behaviour**:This function decrements x by 1 only if the current value of x is greater than 1. Essentially this function is designed to decrement through the array because the values of the pulse width for A and V are stored as doubles in array

- find(var,list)
    - **Parameters:** var,list
    - **Black Box Behaviour**: This function takes a variable and a list as input and returns the index of variable in the array
    - **Internal Behaviour**: The function uses a for loop to traverse through the list and when the variable matches the element from the list it returns the index

3.1.2 LoginData

- Libraries Used:
    - tkinter
- readF()
    - **Black Box Behaviour:** This function reads the login.txt file and returns the usernames and passwords in separate lists
    - **Internal Behaviour:** The function reads the text file and stores each line in separate locations of a list. Then elements of the list are then split at the comma and appended into the user and passw lists respectively.
- validateLogin(login, username, password)
    - **Parameter:** login, username, password
        - The parameter passed is the window which is type tk() from the python library tkinter
        - username and password are type StringVar()
    - **Black Box Behaviour:** This function takes the username and password lists and opens up the main window if the login details match otherwise it outputs access denied
    - **Internal Behaviour:** The function iterates through the entire list of username and passwords and compares them with what the user entered using the get() method. If the login details are the same then the main window is called otherwise it displays access denied

- submitNew(login, username, password)
    - **Parameter:** login, username, password
        - The parameter passed is the window which is type tk() from the python library tkinter
        - username and password are type StringVar()
    - **Black Box Behaviour:** There are 3 inputs , login window and the new username and password strings**.** The functions appends the new user login details into the global list , if the input by the user is not acceptable then a warning message is displayed
    - **Internal Behaviour:** The function first checks to see if the user already exists in the records, the next condition checks if the username and password entered by the user are invalid entries like blank spaces. Once all of the above conditions fail the user is successfully registered by appending the username and password into the global lists.

3.1.3 LoginGUI

- functools.partial
- tkinter
- quitNewUser(w)
    - **Parameter:** w
        - The parameter passed is the window which is type tk() from the python library tkinter
    - **Black Box Behaviour:** The input parameter is the window and the function closes that window when called
    - **Internal Behaviour:** Destroy function is used to close the window which needs to be destroyed.

- newUser(login, username,password)
    - **Parameter:** login, username, password
        - The parameter passed is the window which is type tk() from the python library tkinter
        - username and password are type StringVar()
    - **Black Box Behaviour:** There are 3 inputs , login window and the new username and password strings**.** The function calls submitNew() function to register new user details
    - **Internal Behaviour:** This function accepts the username and password and calls the submitNew() function to successfully register a user. Partial function is used to call the SubmitNew() function. If the max number of users are registered then we call the quitNewUser function to close the new user registration window
- login()

- **Black Box Behaviour:** The function displays the login window with the username and password textbox and 3 buttons login, New User Registration,quit button which perform the desired actions
- **Internal Behaviour:** Inside the function we create username and password label and entry box and. Partial function is used which calls the validate login and to check whether the username and password match. Three buttons are created at the bottom which are used to login once the user enters the details, register a new user and quit button to close the window

3.1.4 MainUserData

- Libraries Used:
  - _tkinter
  - Tkinter
  - functools.partial
  - Time.strftime
  - datetime.datetime
- submit_Parameters(w, mode, lr, ur, aa, apw, va, vpw, vrp, arp, AVD, react, recov, act, maxSen,rateS, vsen, asen,rf)
  - **Parameters:** w, mode, lr, ur, aa, apw, va, vpw, vrp, arp, AVD, react, recov, act, maxSen,rateS, vsen, asen,rf
  - **Black Box Behaviour:** This function takes in all the parameters that were submitted and saves, and stores them a unique text file for each user.
  - **Internal Behaviour:** The function first opens the unique text file for each user. Then all the parameters are saved to a new variable depending on its entry type (StringVar() -> String, IntVar() -> int, DoubleVar() -> float). Try and Except blocks are used to check if the entry boxes are blank (using blankValueError function), if they are, an error message is sent to the console.
- checkParameters(list)
  - **Parameters:** list
    - A list that contains all the parameters (with each one typecasted to its proper data type)
  - **Black Box Behaviour:** a list of parameters is passed through and returns an error_list that holds a [1] or [0, index, message, default].
    - Error_list[0] = 1 or 0, if 1 there are no errors in the parameter. If 0, there are errors in the parameters and the index will be sent, along with the a unique message and the default value of that parameter.
  - **Internal Behaviour:** The functions initializes variables to hold the message that will be transmitted, the global variables are called that control the index of the amplitude list and sensitivity list. Then the program goes through various conditions to check every parameters validity. If the parameter is valid, it will

check the next one and so on. If all are valid, [1] is returned and the program can proceed. If there is an error, the program will return [0, index, message, default].

- incr(value,typeP)
    - **Parameters:** value, typeP
        - value is type IntVar() or DoubleVar()
        - typeP is an integer
    - **Black Box Behaviour:** This function takes the current value and increments it when the plus button is clicked. The value is incremented and returned back to the entry widget
    - **Internal Behaviour:** Using the value that is passed, the function checks what type of incrementation needs to occur with the typeP parameter. Using decision structures it determines the correct incrementation needed and will return back to the user when the plus button is clicked. Using the set() function and get functions.
- decr(value,typeP)
    - **Parameters:** value, typeP
        - value is type IntVar() or DoubleVar()
        - typeP is an integer
    - **Black Box Behaviour:** This function takes the current value and decreases it when the minus button is clicked. The value is decreased and returned back to the entry widget.
    - **Internal Behaviour:** Using the value that is passed, the function checks what type of decremenation needs to occur with the typeP parameter. Using decision structures it determines the correct decremenation needed and will return back to the user when the minus button is clicked. Using the set() function and get() functions.
- setIndex()
    - **Black Box Behaviour:** This function sets the index of the global counters that keeps track of where the float values occur in the list and sets the index, so the incrementation can happen correctly. This is used because the default values from the file
    - **Internal Behaviour:** The function opens the parameter file of the user, and stores the values in a list. Each global counter is called using the global keyword. Using the find() function from the CommonFunction module, find() will return the index where the preSet value occurs and that index gets stored in the global counter
- blankValueError(index)
    - **Parameter:** index
    - **Black Box Behaviour:** The function receives the index of a specific parameter and returns a list. This list contains [0, index, a unique message, default value]. The function is meant to send an error message if

- **Internal Behaviour:** Using the passed parameter, the functions goes through various decisions statements to determine which parameter is blank.

3.1.5 MainUserGUI

- Libraries Used:
  - tkinter
  - functools.partial
  - matplotlib
    - .backends.backend_tkagg import FigureCanvasTkAgg
    - .figure import figure
  - numpy
  - time
- patient_information(w)
  - **Parameters:** w
  - **Black Box Behaviour:** The function displays patient information to the user on the main user window
  - **Internal Behaviour:** The function presents the patient information as text on the main user window using tkinter Label widgets.
- reportMenu(w)
  - **Parameters:** w
  - **Black Box Behaviour:** This function displays a drop down for selecting the type of report and opens the report in the new window.
  - **Internal Behavior:** The function takes in the window as the parameter and then uses the tkinter external library to display button widgets on the window. It also forms a drop down menu that enables users to choose between different types of reports.
- graph(w)
  - **Parameters:** w
  - **Black Box Behaviour:** The function displays graph information and buttons to the user on the main user window. This allows the user to select the requested graph.
  - **Internal Behaviour:** The function presents the graph information and buttons as text on the main user window using tkinter Label widgets.
- viewGraph(selection)
  - **Parameters:** selection
  - **Black Box Behaviour:** Input(Selection) is the type of signal and the output is the plotted signal
  - **Internal Behaviour:** The function takes selection as input and calls the plot_start function to plot the required signal. It can also plot both signals
- plot_start()

    - **Black Box Behaviour:** The function modifies the global variable cond
    - **Internal Behaviour:** This function increases the counter value and sets the global variable “cond” to true
- plot_stop()
    - **Black Box Behaviour:** The function modifies the global variable cond
    - **Internal Behaviour:** This function increases the counter value and sets the global variable “cond” to false
- _clear()
    - **Black Box Behaviour:** the function clears the tkinter canvas
    - **Internal Behaviour:** The function uses global variables and clears them by initializing again. The function also uses a for loop to traverse items in widgets to delete them and reset it.
- plot_data()
    - **Black Box Behaviour:** When the global variable cond is set to true, the
    - **Internal Behaviour:** If the global variable cond is set to true, the SerialCom.getAVSignal() function is called and is assigned to a variable called signal. If the requested graph is ventricle, only the first index of the signal list is used. If the requested graph atrial, then the second index is used. If both graph are requested, both are used. After the have been set, the signal is appended to its own data list and is plotted on the matplotlib tkinter canvas. This will continue to loop as long as cond is True, if it is False it will stop the graph. The after function so the the function can loop without worrying about getting stuck in an infinite loop.
- showCanvas()
    - **Black Box Behaviour:** Displays the tkinter canvas on the main window
    - **Internal Behaviour:** The function initializes figures, axis, and lines. The canvas is made and set on the main window with the correct height, width, and coordinates.
- prog_para(w)
    - **Parameters:** w
    - **Black Box Behaviour:** Input of this function is the window currently being used and it sets the Programmable parameters values.
    - **Internal Behavior:** The function takes in ‘window’ as the parameter. The function creates labels for different pacing modes and allows the user to set different values for respective pacing modes. The values set by the user can also be incremented or decremented with an appropriate step size. The ‘submit’ label writes and stores information in a separate text file for our records.
- displayError(w,error_list, lr, ur, aa, apw, va, vpw, vrp, arp, AVD, react, recov, maxSen, vsen, asen,rf)
    - **Parameters:** w,error_list, lr, ur, aa, apw, va, vpw, vrp, arp, AVD, react, recov, maxSen, vsen, asen,rf

    - **Black Box Behaviour:** The function clears the console and identifies errors and sets it back to the default value on the console.
    - **Internal Behavior:** The function takes in the parameters as stated above and then it sets all the parameters to default values after an error has been encountered and the error message has been displayed on the DCM interface.
- set_message(w, pacingMode, lower_rate, upper_rate, atr_amp, atr_pw, ven_amp, ven_pw, vrp_, arp_, avd_, react_,recov_, act_, max_sen, rate_s, v_sen, a_sen,res_fact)
    - **Parameters:** w, pacingMode, lower_rate, upper_rate, atr_amp, atr_pw, ven_amp, ven_pw, vrp_, arp_, avd_, react_,recov_, act_, max_sen, rate_s, v_sen, a_sen,res_fact
    - **Black Box Behaviour:** This function takes the above parameters as input and displays the parameters in the console window
    - **Internal Behavior:** The inputs are converted to strings to display the parameters in the display window
- clearConsole(w)
    - **Parameters:** w
    - **Black Box Behaviour:** This function takes the current window as input and clears the console window
    - **Internal Behavior:** The function clears the console window when the user sets new parameters or enters invalid parameters.
- window_Quit(w)
    - **Parameters:** w
    - **Black Box Behaviour:** The input is the current window and the output is the login window
    - **Internal Behavior:** The function calls the destroy function and quits the window to open the login screen where user can login again
- main_window()
    - **Black Box Behaviour:** The function creates a window
    - **Internal Behavior:** This function uses labels to create the sub-windows in the main window

3.1.6 Reports

- Libraries Used:
    - datetime
    - time
    - tkinter
    - functools.partial
    - matplotlib
        - .backends.backend_tkagg import FigureCanvasTkAgg
        - .figure import figure

    - numpy
    - time
- report_Submit(w, selection)
    - **Parameters:** w, selection
        - w is a tkinter window
        - selection an entry box variable of type StringVar()
    - **Black Box Behaviour:** The function takes in a tkinter window and returns the selected printed report.
    - **Internal Behaviour:** This function opens a unique report file that will write the report that was requested. This function first writes and displays the header of the report. After the header is saved, the function calls the specified report type function (parameterReport(r) and egramReport(r))
- parameterReport(r):
    - **Parameters:** r
        - r is a tkinter window created within the report_Submit() function
    - **Black Box Behaviour:** This function writes the parameters to the window and saves within a text file.
    - **Internal Behaviour:** This function first sets the size of the window, and then opens the patient's parameters file. Two lists are initialized to hold the title of the parameter and the units of the parameter (this lists are parallel to the parameter file information list. Using a for loop, the parameters are written to the file and to the window.
- egramReport(r)
    - **Parameters:** r
        - r is a tkinter window created within the report_Submit() function
    - **Black Box Behaviour:** This function writes the parameters and outputs an egram sampled within 1 second.
    - **Internal Behaviour:** This function first sets the size of the window, and then opens the patient's parameters file. Two lists are initialized to hold the title of the parameter and the units of the parameter (this lists are parallel to the parameter file information list. Using a for loop, the parameters are written to the file and to the window. Using the matplotlib library and tkinter canvas we can sample the AV signals and they can be show in the graph

3.1.7 SerialCom

- Libraries Used:
    - serial
    - struct
    - Serial.tools.list_ports
    - time

    - tkinter
- sendParameters(pacingMode, lower_rate, upper_rate, atr_amp, atr_pw, ven_amp, ven_pw, vrp_, arp_,avd_,react_,recov_, act_,max_sen, rate_s,v_sen, a_sen)
    - **Parameters**:pacingMode, lower_rate, upper_rate, atr_amp, atr_pw, ven_amp, ven_pw, vrp_, arp_,avd_,react_,recov_, act_,max_sen, rate_s,v_sen, a_sen
    - **Black Box Behaviour:** The function takes pacingMode as input and prints the hex value corresponding to the pacing mode. Similarly the hex value for all the different inputs is stored and printed to carry out serial communication with simulink
    - **Internal Behaviour:** The function uses the struct.pack() to calculate the hex value of all the different inputs to transmit data to simulink . For pacing modes , we used an int variable called mode which assigns a value to each pacing mode. The int value is then passed to the struct.pack() function to determine the decimal value in bytes. Similarly for the act_ input we assign a unique integer to the act_threshold which is then converted to hex value. This is done to all the parameters. Once the parameters are converted, these converted values are sent to the send() function which will pack and send all the values and automatically send the parameters. After they are send, the echoParameters() function is called and will return the values back to check if they are valid.
- connection(window)
    - **Parameter:** window
    - **Black Box Behaviour**: The function takes the tkinter window as input and displays whether the pacemaker is connected/disconnected on the window. This function also will display an icon if there is a new pacemaker device connected to the DCM
    - **Internal Behaviour:** The function checks if the COM port connected to the pacemaker is in the list of all active COM ports. If the active port is in the list that means that the pacemaker is connected to the correct COM port and will display that the device is ready and connected.
- send(toSend, sequence)
    - **Parameters:** toSend, sequence
        - toSend is a list of all parameters including the bits required to initiate the sending process. [22,18, …(all parameters)]
        - sequence is the order of the data types the parameters need to be packed as. “<BBHHHfHfHHHHHHdHHffH”
    - **Black Box Behaviour:** Sends the input list through serial communication in bytes
    - **Internal Behaviour:** the function first checks if toSend list and sequence is the same length, if this is true, the data is packed into bytes and is appended to a new list which is sent one by one through serial communication function s.write()
- echoParameters()

    - **Black Box Behaviour:** Should return the same list as the one sent except for the two last value
    - **Internal Behaviour:** Initializes a zero list, with the first to elements set to 22 that will let simulink know that we want to receive the information back. The start sequence is set as "<BBHHHfHfHHHHHHdHHffH" so the bytes get packed and sent. And an unpack sequence is also set "<HHHfHfHHHHHHdHHffHdd". Once the first sequence and list is sent, the function can now use s.read(66) to read 66 bytes of date, the bytes are unpacked and saved.
- getAVSignal()
    - **Black Box Behaviour:** returns the last two elements in a list to get the AV Signal
    - **Internal Behaviour:** The function calls the echoParameters function and saves the last two values and returns them as a list.

3.1.8 Util

- Libraries Used:
    - tkinter
- About:
    - **Black Box Behaviour:** The function creates a new window which displays the pacemaker information, for example the team name, model number of the device, application version and DCM serial number
    - **Internal Behaviour:** The function is used to display text on the window using the Label widget which creates a display box.
- Util_Button()
    - **Black Box Behaviour:** This button is used to open a new window called the utilities menu
    - **Internal Behaviour:** This function opens up a new window which has different buttons for actions. When the about button is clicked the about() function is called. This window also has return to DCM window which destroys the utility menu.

3.2 Global Variables

3.2.1 LoginGUI

- **user[] and passw[] -** These two lists are used to store the username and password of the user. These lists include all usernames and passwords for already registered users. When a new user is registered , the lists are appended with new user entries
- **name_Flag -** this global variable saves the index of the logged in user so the first name and last name can be displayed to the user in the main window

3.2.2 MainUserData

- **AtrAmp_counter, VenAmp_counter, AtrSens_counter, VenSens_counter** - The global variables are used as a counter for incrementing the values through an array. The array holds the values of the pulse width for A and V, this is useful as the values are incremented by the exact amount required
- **setFlag -** a global flag is used to confirm that parameters have been sent and saved to the text file, list, and device.
- **amp[], sens[]** - holds the value of all the possible values for A/V amplitude and A/V sensitivity as floats. This is done so we can account for floating point errors when the user increments a float value on the main window.

3.2.3 MainUserGUI

- **isConnected, isNewDevice -** these global variables are boolean values to check if the device is connected and if a new device is integrated than before.
- **time_ms[], vData[], aData[]** - holds the current values of the time, ventricle signal, and atrial signal. These are cleared and appended to while the egram is running.
- **t -** holds the current values of the time. This values gets incremented every loop so the time_ms can be updated and plotted every 1 ms.
- **cond -** this is a boolean value that controls the start and stop of the electrocardiogram plot. When this variable is set to true (the view button was pushed) it will start the diagram. Otherwise, it won't show the electrocardiogram. If the variable is false it was stopped and or just initially off.
- **counter -** this counter counts every time the canvas is displayed, this is used so the _clear() function won't result in an error if there is no diagram present.
- **window -** this variable is initalized as NONE (tkinter constants) and later assigned to the main window.
- **canvas, ax, fig -** these variables hold the elements need to construct the canvas and display it. This is set to a global variable so that all the graphing and plotting functions can have access to them
- **graphType -** holds a string depending on what kind of graph is requested by the user. This is saved and used in the plot_data() so that the function knows exactly what kind of graph to display

3.2.4 SerialCom

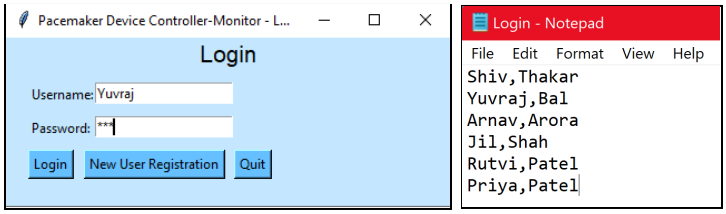
- **s -** this global variable hold are serial communication port and baud-rate to transmit data to the device.

4.0 Testing

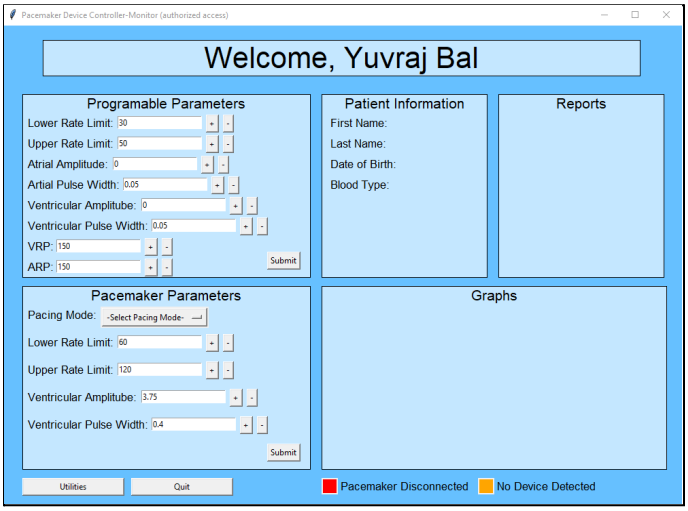
4.1 Assignment #1 Test Cases

4.1.1 Existing User Login Test

The username “Yuvraj” and password “Bal” exists in the text file

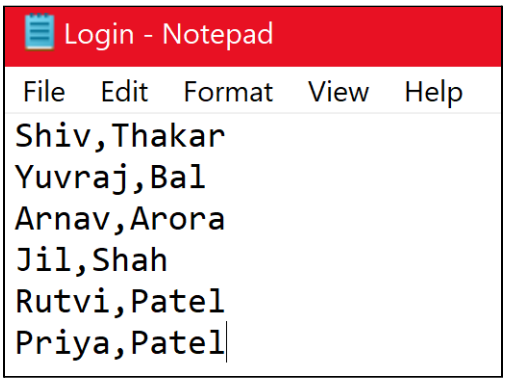


Once the login button is pressed.



4.1.2 New User Registration Test

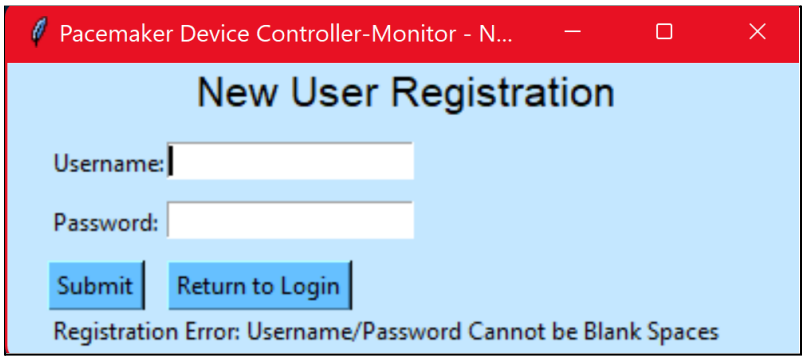
Starting Text File.

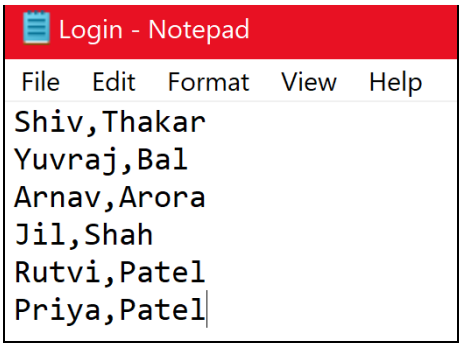


When adding a new user, the input should not be blank, same as an existing user, and have a maximum of 10 users in memory.

4.1.2.1 Blank Entry

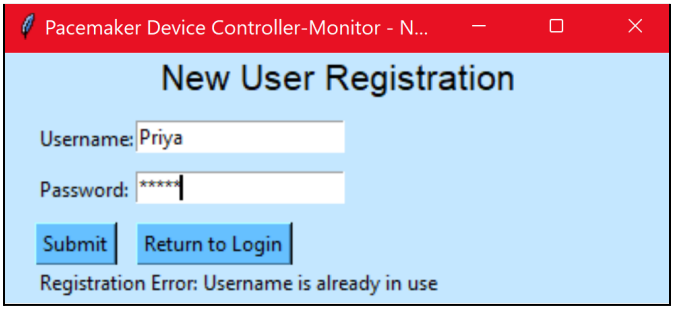
This test should show an error message on the window since the a new user cannot have a blank username or password

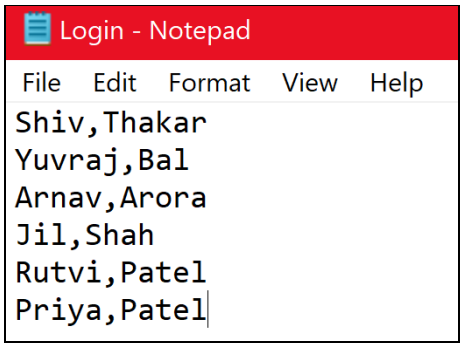




4.1.2.2 Existing User

This test should show an error message on the window since the user Priya already exists, and it will not register the user.

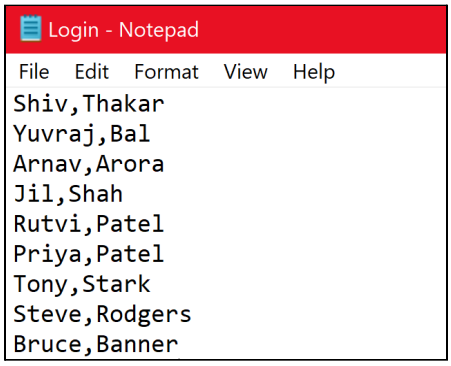




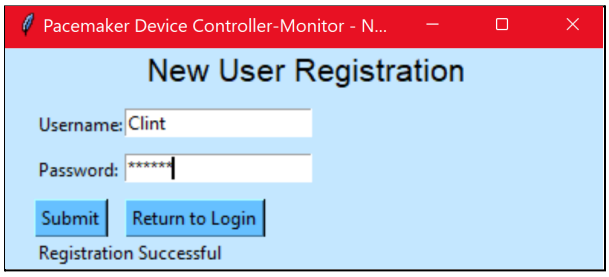
4.1.2.3 Enter Eleventh User

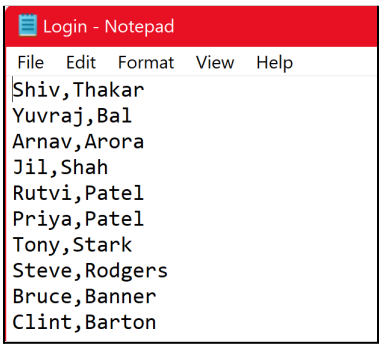
This test should show an error message on the window since the maximum number of users had already been reached.

Starting text file contains 9 usernames and passwords.

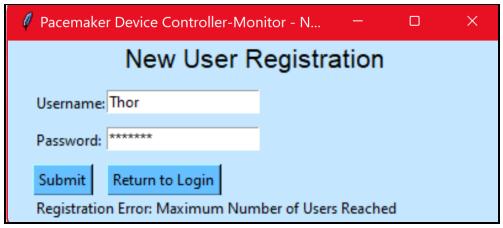


10th username and password registration successful and stored into the file.

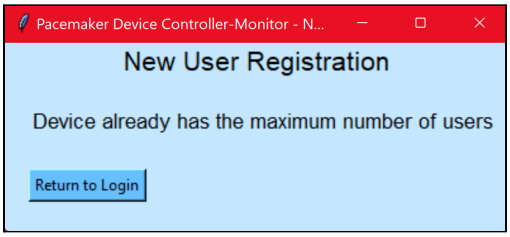




Now if an 11th username is entered an error message will be displayed.



When closing the program and running it again, if the user tries to open the registration menu when there are already 10 users the program will not show the username and password entry widget but will give a message telling the user it has reached the maximum number of users.



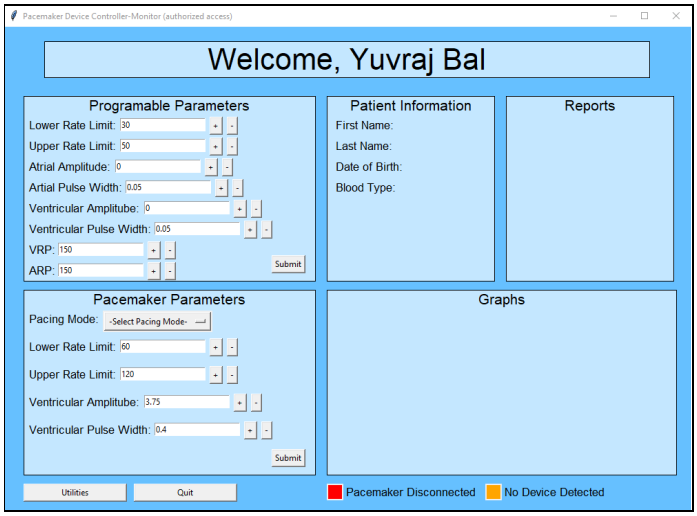
4.1.3 Device Connections Test

Using the global boolean variables and decision structures will output a different colour and message on the main window.

```
#Device Connection
if(isConnected == True):
    colour = "green"
    connect = "Pacemaker Connected"
elif(isConnected == False):
    colour = "red"
    connect = "Pacemaker Disconnected"

canvas = Canvas(window, height = 20, width = 20, bg = colour)

canvas.place(x = 470, y = 660)

canvas.create_rectangle(0,0, 200,200, outline = "black", fill = colour)

connection = Label(window,
                   text = connect,
                   font = "Airal 13",
                   bg = bg_colour).place(x = 495, y = 660)

#New Device
if(isNewDevice == False and isConnected == True):
    colour = "green"
    connect = "Previously Interrogated Device"
elif(isNewDevice == True and isConnected == False):
    colour = "red"
    connect = "Newly Interrogated Device"
elif(isConnected == False):
    colour = "orange"
    connect = "No Device Detected"

canvas = Canvas(window, height = 20, width = 20, bg = colour)

canvas.place(x = 700, y = 660)

canvas.create_rectangle(0,0, 200,200, outline = "black", fill = colour)

connection = Label(window,
                   text = connect,
                   font = "Airal 13",
                   bg = bg_colour).place(x = 725, y = 660)
```

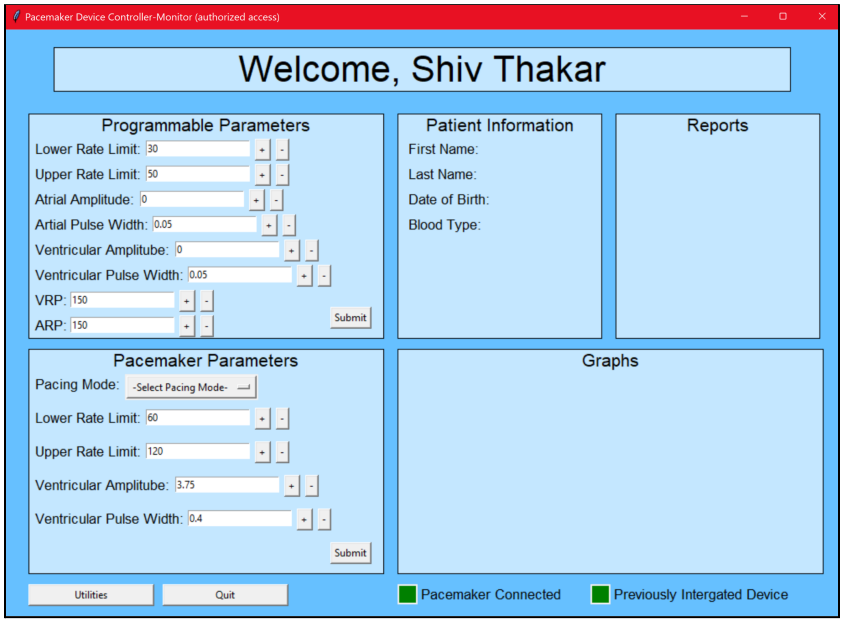
4.1.3.1 isConnected = False, isNewDevice = False

isConnected will output “Pacemaker Disconnected” with a red icon, isNewDevice will output “No Device Detected” with an orange icon.



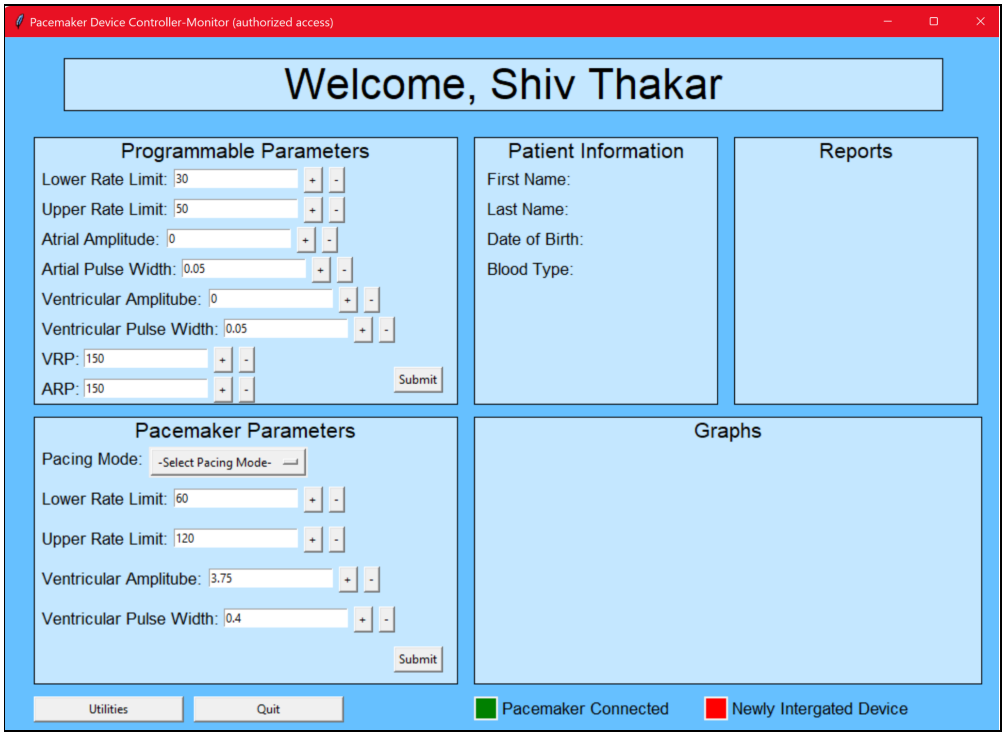
4.1.3.2 isConnected = True, isNewDevice = False

isConnected will output “Pacemaker Connected” with a green icon, isNewDevice will output “Previously Integrated Device” with an green icon.



4.1.3.3 isConnected = True, isNewDevice = True

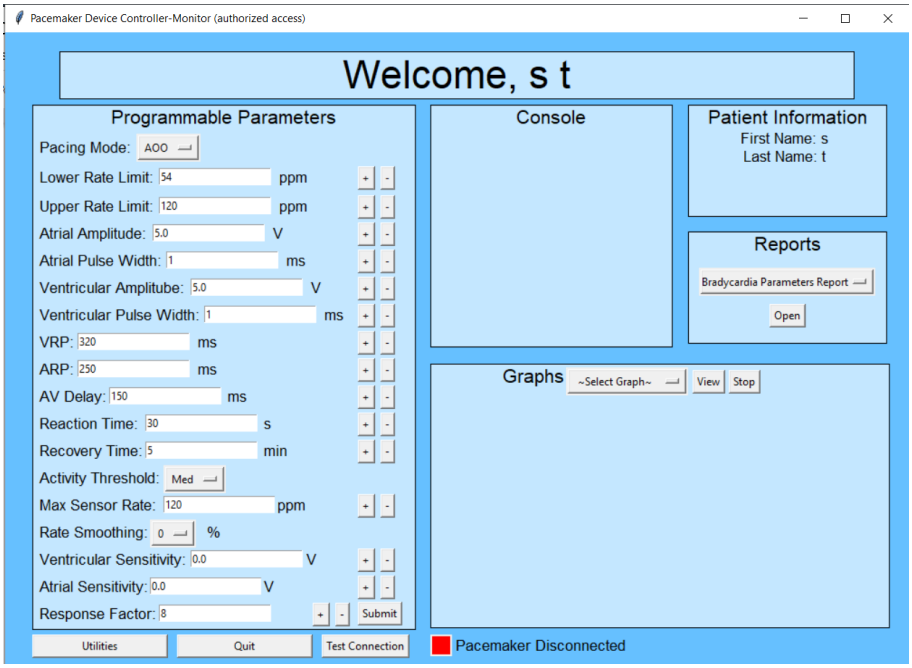
isConnected will output “Pacemaker Connected” with a green icon, isNewDevice will output “Newly Integrated Device” with an red icon.



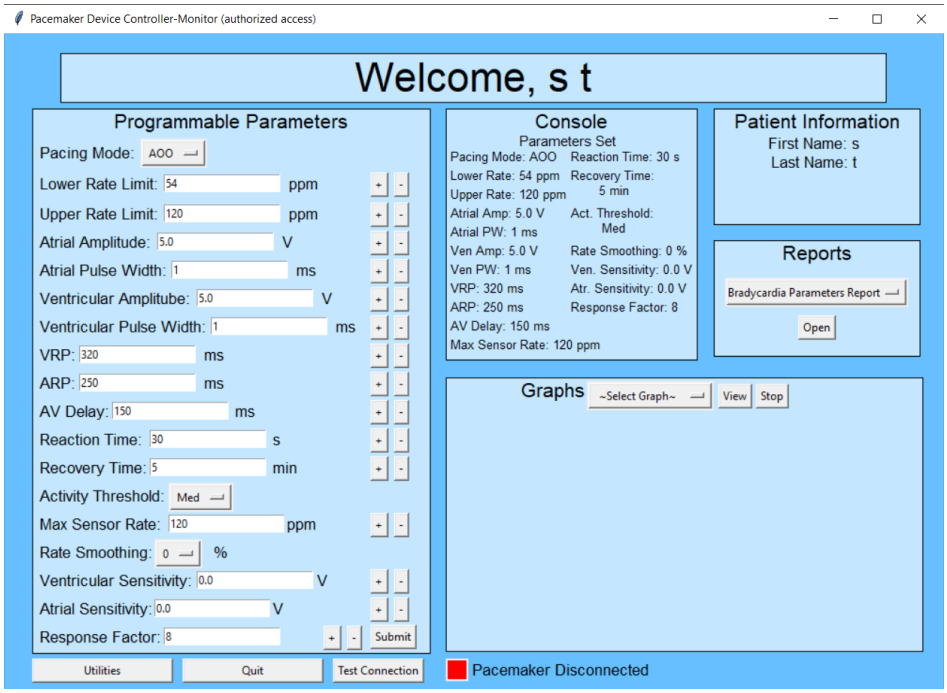
4.2 Assignment #2 Test Cases

4.2.1 Setting Parameters Test

The default parameters are shown after log in.

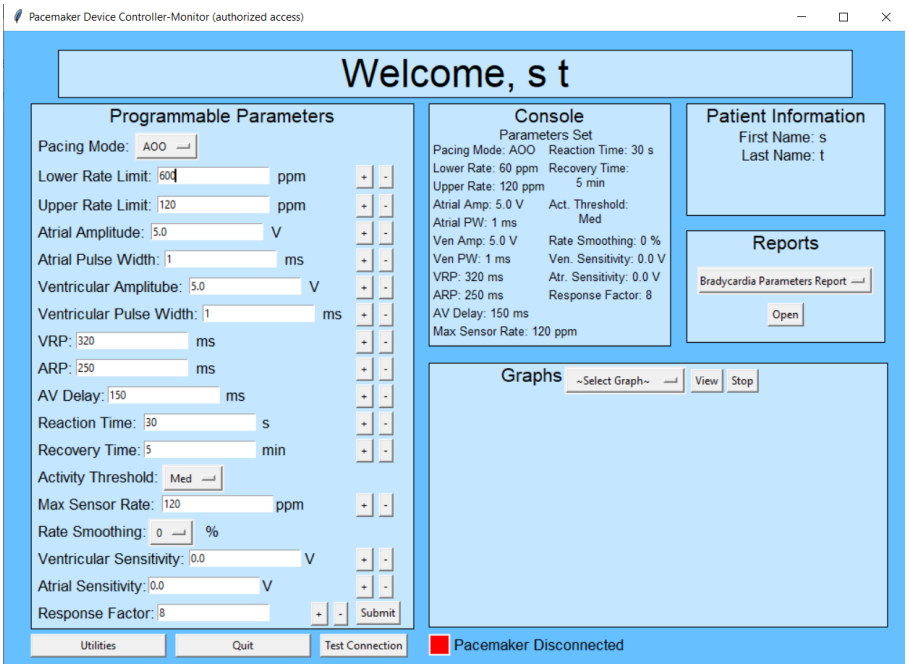


After changing the Programmable Parameters and pressing submit

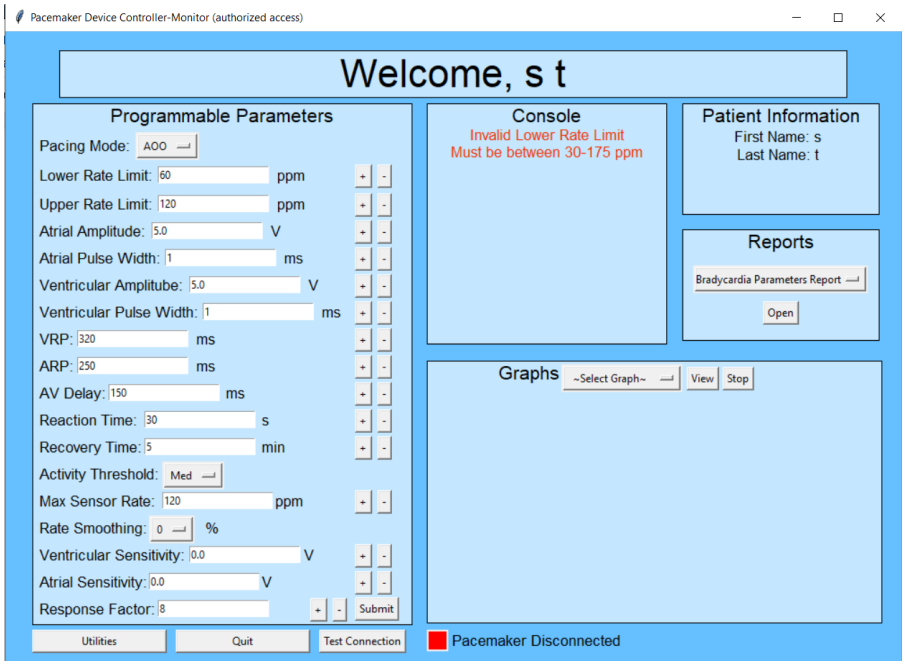


4.2.2 Invalid Parameters (Lower rate limit out of range)

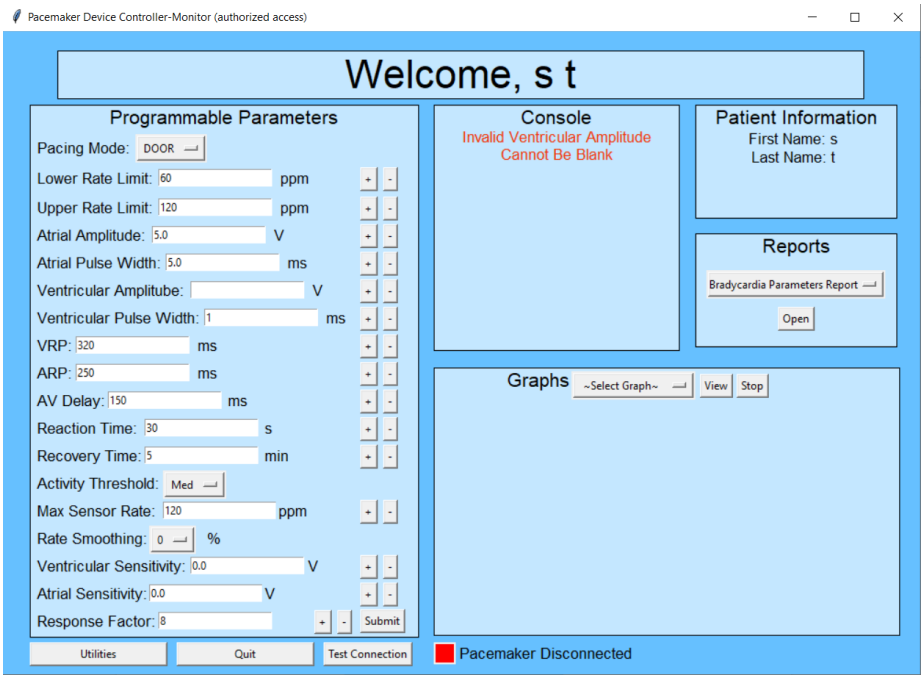
Setting Lower rate limit to 600



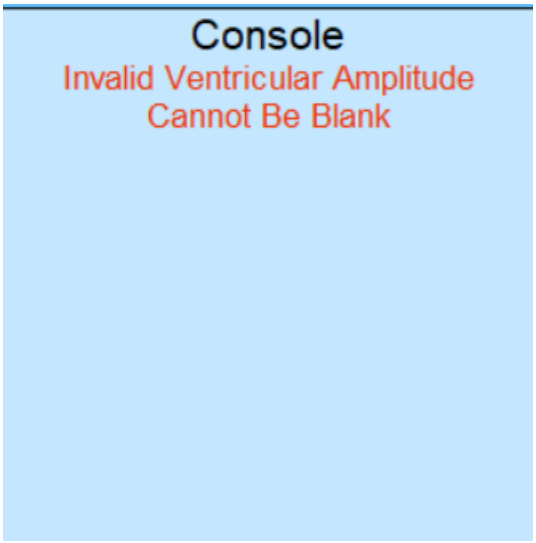
Console displaying the error after pressing submit



4.2.3 Blank Parameter (Ventricular Amplitude)



Console displaying the error after pressing submit

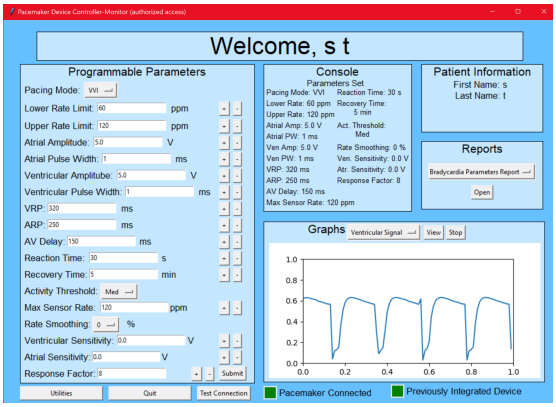


4.2.4 Electrocardiogram Test Video

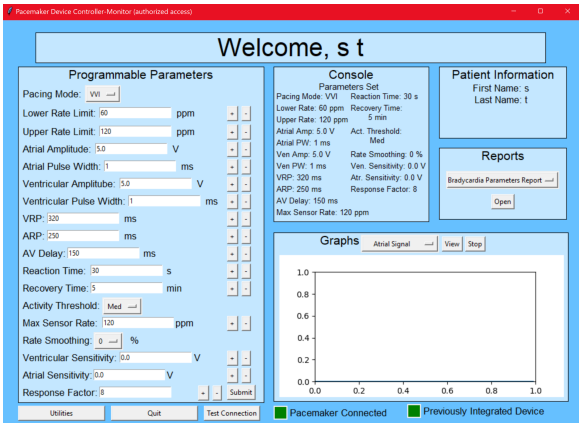
[DOO_AAIR_EGramTest.mp4]

The test below shows the electrogram for VVI pacing mode.

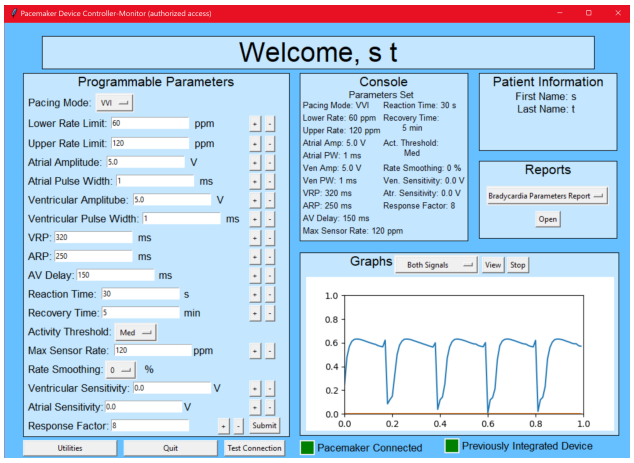
When the user requests the Ventricle Signal, only the ventricle graph will be displayed.



When the user requests the Atrial Signal, only the atrial graph will be displayed. The graph shows that atrial is zero since in this pacing mode, there should not be a Atrial signal being outputted.

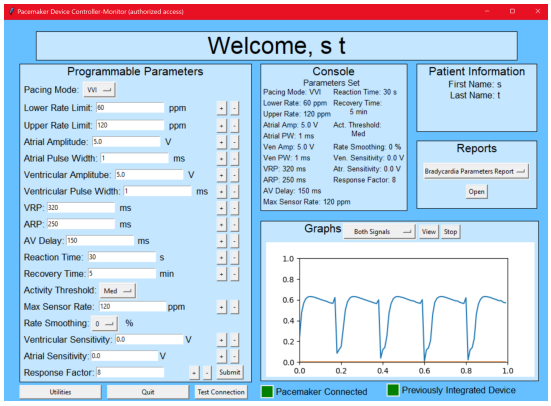


When the user requests the Bothl Signals, ventricle and atrial signals will be will be displayed. Ventricle Signal is blue while Atrial is set to orange.

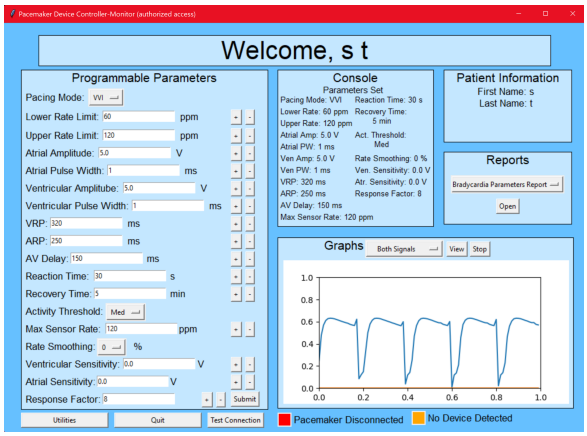


4.2.5 Device Connection with Serial Communication

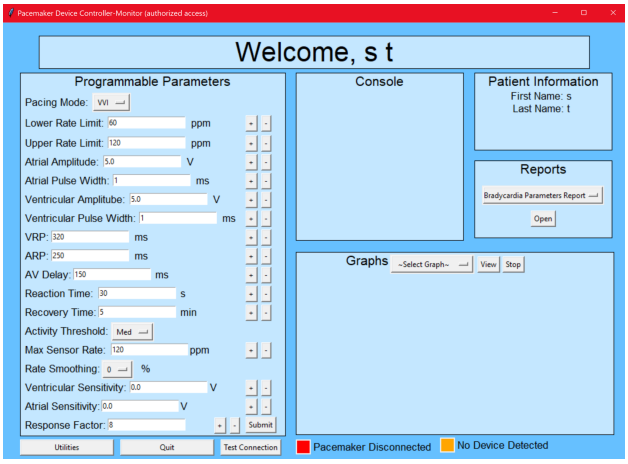
When the device is connected to the computer, the program will check if it is connected by checking all the COM ports that are active. If the program determines that COM8 (specific to my hardware) is active, the DCM will show that the device is ready.



If the Device is not connected or gets disconnected while the DCM is running, this will be displayed on the window. Connection is tested with the “Test Connection” button, this button uses the same function as above, but will give a different message.

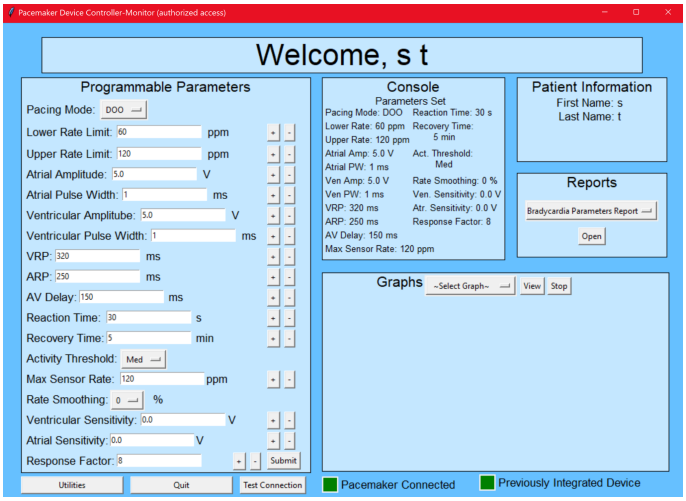


Disconnected while the DCM is active



4.2.6 Sending Data

For testing purpose, we used print statements to view what is being sent and received. These parameters were sent, and on the console they will show so the doctor knows that the parameters were set



In the Python Shell:

----------Sending Data----------

To Send:

[22, 18, 4, 60, 120, 5.0, 1, 5.0, 1, 320, 250, 150, 30, 5, 0.08, 120, 0, 0.0, 0.0, 8]

Send in Bytes

[b'\x16', b'\x12', b'\x04\x00', b'<\x00', b'x\x00', b'\x00\x00\xa0@', b'\x01\x00', b'\x00\x00\xa0@', b'\x01\x00', b'@\x01', b'\xfa\x00', b'\x96\x00', b'\x1e\x00', b'\x05\x00', b'{\x14\xaeG\xe1z\xb4?', b'x\x00', b'\x00\x00', b'\x00\x00\x00\x00', b'\x00\x00\x00\x00', b'\x08\x00']

----------Receive Data----------

Send for Echo:

[22, 22, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Send in Bytes

[b'\x16', b'\x16', b'\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00\x00\x00', b'\x00\x00', b'\x00\x00\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00\x00\x00\x00\x00\x00\x00', b'\x00\x00', b'\x00\x00', b'\x00\x00\x00\x00', b'\x00\x00\x00\x00', b'\x00\x00']

ECHO:

[4, 60, 120, 5.0, 1, 5.0, 1, 320, 250, 150, 30, 5, 0.08, 120, 0, 0.0, 0.0, 8, 0.4637216754406043, 0.3495384145876249]

As shown above, the value being sent is echoed back. The first two bytes is [22,18], these are the decimal values of the hexadecimal value that is needed to allow simulink to read data. For

echo, the first two bytes must be equivalent to hexadecimal 16 to allow simulink to send data. This allows us to confirm that the right parameters are being sent to the pacemaker device and are being stored. The last two elements in the echo list hold the ventricle and atrial signals