Assignment 2 – DCM

SOFTWARE DEVELOPMENT 3K04

GROUP 14: PIKA

Table of Contents

Table of Figures

Introduction

The goal of the DCM for the second assignment was to expand the existing DCM model to include the required modes and their programmable parameters. The most important iteration in this section was the implementation of the serial communication to establish a connection between the DCM and Pacemaker which becomes important for the egram and information transmission. The following sections outline the expected changes in the design and requirements as well as breaking down the functions used to create the functioning model.

Requirement Changes That are Likely

In the upcoming design for assignment three, the DCM will be required to build on the current framework. The current DCM covers a portion of the modes outlined in the assignment and will be required to expand to all modes and their given programmable parameters. Some of these will include the DDD and DDDR modes with 20 and 25 parameters respectively within the design space. These modes will require the use of serial communication and the on-board accelerometer to sense and pace within the pacemaker. Serial communication will also integrate in the DCM to display and interchange data. The DCM will require a check to ensure programmable parameter data is set, stored, transmitted and retrieved correctly, implying that the serial communication is vital. Displaying the egram data when the user prompts and successfully receiving data from the serial communication link will be an ongoing requirement for all modes. The future model will also require a display of both the egram data and egram for the DDD mode.

Design Decisions Likely to Change

The DCM design will change in a few aspects to account for the requirements discussed above. The main design changes include the addition of the DDD and rate adaptive DDDR modes with all their programmable parameters included in the user's profiles and stored according to the previous modes. The serial communication functions will be altered to account for constant updates with the Pacemaker model to make sure that stored values are accurate and that the information is synchronous between systems (data required found in Table 7 of the 'PACEMAKER' file on Avenue to Learn). Included in the DCM will be an option to show the egram data and egram on screen (both required for DDD mode) and actively change according to the Pacemaker input sensors and triggered functions. The transmittable parameter data for the past modes and newly implemented modes will be similar to the write() functions explained below. Therefore, the design will require the addition of new functions within serial communication to account for the checking of synchronous data, a function to display egram data and the addition of the mentioned pacing modes with the respective writing and storing functions.

Module 1: Register Stage

The first module, which is the left section of the register/log-in screen, allows a new user to register with a username and a password, which is starred out when typed for privacy. There is no "secret" to this module since it only has one function, to store new usernames and passwords into two text files. The public functions used in this stage are: Label(), Grid(), Entry(), Delete(), Button() and After().These are functions were used from the tkinter library to display the screen, error messages and the boxes where the usernames and passwords are entered. The black box diagram for this module takes two inputs, the username and the password of a new user. It then checks if both follow the requirements and either stores the inputs or displays an error message depending on the error type.

Below is a description of each public function used:

- The Label function has inputs (window, text, colour) and outputs a text in the window specified. The Grid function has inputs (row, column) and is used to place a text created by Label().
- The Entry function takes inputs (window, text) and outputs a textbox that can be written in.
- The Delete function takes inputs (textbox, space) and deletes characters from the textbox based on the space provided.
- The Button function which takes inputs (window, text, command) and outputs a button that runs a function when clicked.
- The After function which takes inputs (time, command) which performs a command after a specific amount of time.

The black box diagram for registering:

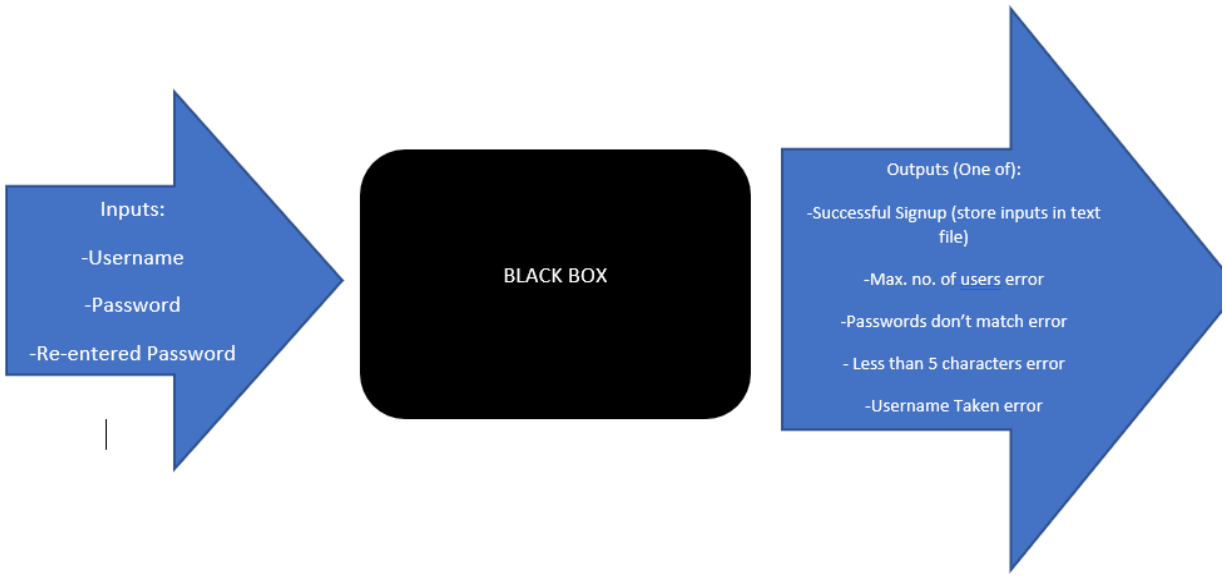

Figure 1: Module 1 Black Box Diagram

The User inputs a username and password and then re-enters the password to make sure there are no typing errors and if the inputs follow the requirements the username and password will be stored in two text files for log in or an error message will be displayed if the inputs do not follow the requirements. The only private function used is the function storing() which runs after

the “Register” button is clicked to store the username and password into the two text files. The storing function uses get() to read the inputs from the buttons name1, pw1 and repass1 created using Entry() and stores them into the variables user1, password1 and repass which are then appended into the two text files, and are then checked by the logging() function to log-in the user. If the password does not match the re-entered password an error message will be displayed saying “Passwords do-not match!”. If the username is used by another user a message will be displayed saying “Username taken!”. If there are 10 users registered and a new user tries to register an error message will be displayed saying “Max amount of users registered!”. If the username or password do not include a number or are less than 5 character long an error message will be displayed saying “Username and Password must be minimum 5 characters and long and include at least 1 number”. All errors display their own error text and does not save the username or the password.

Module 2: Login Stage

The second module serves as the introduction to the program and greets the user with the name of the pacemaker system, Pika Pacemaker. In this module the user can login to their profile with their saved personal parameters and information. The module does not contain any secrets and provides the user with a login screen to access their profile. The main public function is the ‘logging’ function where the user can sign into a profile they just created or a previously stored profile. Inside the logging function are the locally defined functions ‘clear’ and ‘error’ used to clear the text boxes and display error messages respectively. The functions imported from the ‘tkinter’ file include the Tk, title, button, label, grid and entry functions to create a window, text boxes and text needed to create the GUI.

Below is a description of each function used:

- The logging function takes no inputs and is called when the submit button is pressed on the tkinter window, logging()
    - The clear function clears the text boxes and destroys the labels. It is called when the login is incorrect. clear() takes no inputs

- The error function takes no inputs and opens a window to reiterate the registration rules, error()

- The tkinter functions:
  - Label(Window Name, Text, Color).pack() uses a previously defined tkinter window to display a name or title and takes in the window, text to show and the text color
  - Label.grid(row, column, column span) orients the location of the text popup and takes in coordinates on the windows grid pattern
  - After(time, function) uses the tkinter function to destroy all labels and clear the screen from any inputs
  - Button (Window Name, Text, command) is the tkinter function to create a button and delegate a command which in this case opens the globally defined logging function
- Python Library Functions:
  - Open(txt file name, operation 'read') This function from the python library opens the text file containing the user accounts to check for validity and uses line.strip on each line to check for a valid profile combination

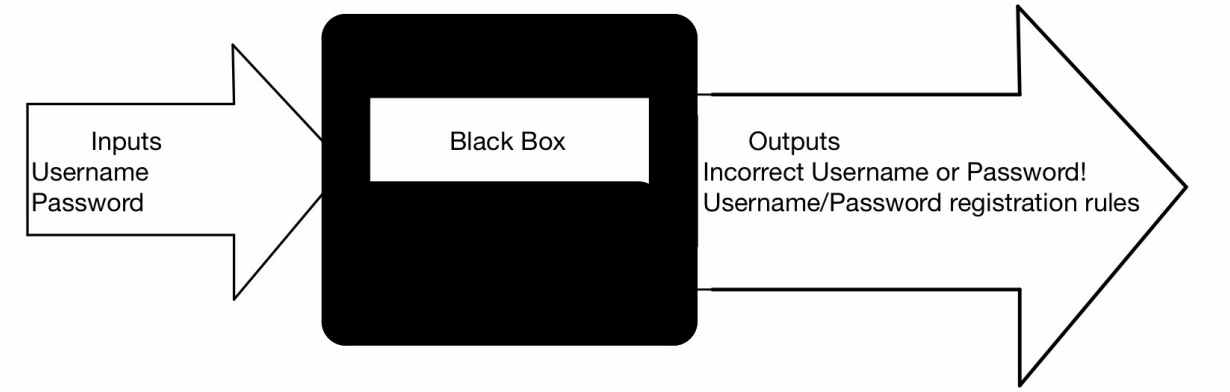

Figure 2: Module 2 Black Box Diagram

Analyzing the program for error and case handling we find there are checks for interface specifications and whether the user enters a valid value. The two inputs into the module are the user's username and password and are the two variables used in testing the robustness of the logging function. The system takes the inputs and assigns them to user1 and password1 to check for the length of the credentials, minimum of 5 characters, and the presence of a number. If this

case is found to not be satisfied the program displays the default error message "*Username and Password must be minimum 5 characters long with minimum 1 number in both" in red text and prompts the user to re-enter. The second case checks for the presence of the username and password within the text file to confirm the identity and notifies the user with the message "Incorrect Username or Password" if the account credentials are not found or entered incorrectly. Thus, the program efficiently handles the error cases for the login screen and notifies the user of any issues encountered during the login process.

The module begins after the user enters a username and password into the provided entry boxes and the globally defined 'Login' button is pressed. The button press defined by the tkinter file calls the logging command which completes the checks for the values and call the user's profile information and segue into the next screen. Within the logging function, the program extracts the globally defined username and password and assigns them to variables to be locally defined within the scope of the function. The function continues by checking Boolean cases for the user credentials by looking for a number in each and assigning 'true' to the Boolean labels for each entry. After creating these Boolean cases, the function then checks the length of the user credentials, if they don't meet the minimum length of 5 characters or one of the previously mentioned Boolean cases comes back as true the locally defined 'error' function is called. The error function displays a screen stating the registration rules and the red warning text explained in the black box analysis of the module. Once the user enters a valid username and password, the module continues to check for the username in the 'Users.txt' file and uses an if statement to open and check the contents of the file. If the username is not found, an error message is displayed saying "Incorrect Username and Password!" and prompting the user to re-enter the credentials or go back to the registration process. Once the function finds the username in the text file, the file's contents are put into an array where each username or password are put into an index in the newly defined local array. The for-statement loops through the array to find the matching index line for the username input and then checks the same index in the passwords array created using the same method with the 'passwords.txt' text file. Once the password is found in the index, the password entered, and the indexed password are compared,

if they match the function continues and if not the ‘error’ function is called restating the registration rules. If all these checks are passed the function, then calls the ‘win2’ profile window function and passes the username to the second window.

Module 3: Patient Information and Pacemaker Data

The third module provides the user with the interface to apply new parameters, view previously saved states and modify any of the values specific to their profile. The module has no secrets and stored data for each user is created in keeping with the 10-user limit provided in the program specifications. Each user has the same code but applied to their specific profile and therefore the code being analyzed will be an example of one sample user and is assumed similar for all users. The text files, where all the values are read from, are used to create the specialized GUI attached to that user. The main function is the ‘win2’ function that is called from the ‘logging’ function and is passed through the username entered and renamed ‘user’ within the win2 function. The function imports ‘font’, ‘geometry’ and ‘configure’ from the ‘tkinter’ file used in the GUI and uses the same tkinter file to create a new window welcoming the user by name. The other main functions used within the win2 function such as ‘VOO’, ‘AOO’, ‘refresh’, ‘inc’, ‘func’ and many others that will be described in more detail below in the module design.

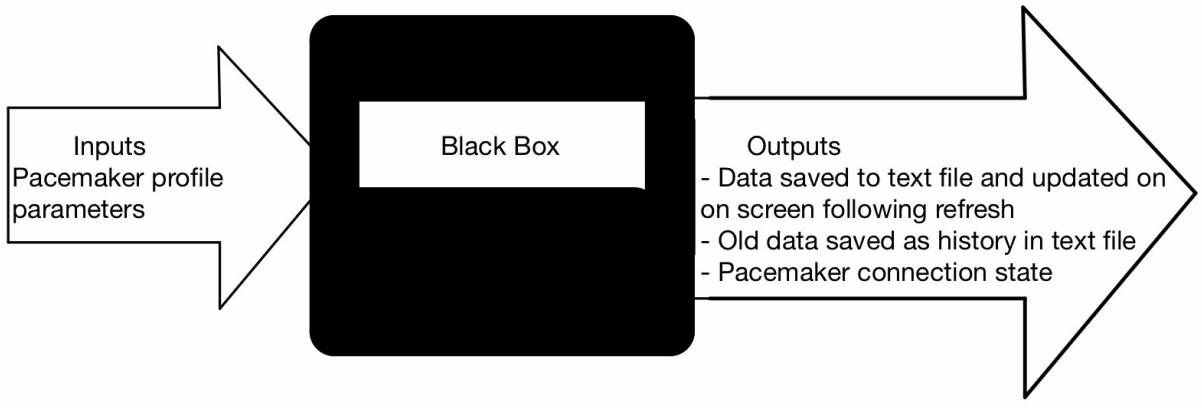

Figure 3: Module 3 Black Box Diagram

The module interface has case handling for a few user inputs and the black box analysis. Once the user logs in and sees their profile parameters they can choose one of the 8 modes from a drop-down menu and enter new parameter values for the first time or edit past inputs. Once the button is pressed, the user chooses the range for the value from one menu and then a value within that range through another menu. The user is only able to choose values within the chosen range and once all the values are submitted, they are then saved to a text file. The saved parameters are not local variables but instead are saved within a text file along with past configurations. The module also has a refresh option to check the text file for the newest parameters and update the screen accordingly. Every time a past input is edited the new values are appended to the text file and when the refresh button is pressed the most recent values are read from the file. To add to the robustness of the module the user can close the program and reopen it with the same numbers and is able to see whether the USB is connected to the microcontroller.

The main function in each file is named after the mode and includes several other functions that create the drop-down menus. There are 8 modes with a total of 20 different parameters, some of which are included in more than one mode. The 8 pacing modes are: VOO, AOO, VVI, AAI, and their rate adaptative versions: VOOR, AOOR, VVIR, AAIR. Some functions have been used for all modes and all parameters, such as “write”, “store”, “print”, “combine”.

Below are descriptions of each type of function in the 8 modes:

- All “write” functions take the file name(user) as an argument to extract the chosen parameter values, then use labels to display them on the screen when the refresh button is pressed.
- All “store” functions write the chosen parameter values into a file called “parameters.txt”, using the “write” functions.
- All “print” functions display the labels for the drop-down menus on the second window.
- All “combine” functions simply call the “print” functions and the “store” functions for each parameter.

- All “inc” functions create a list with all the incremented values between a set range. For example in VOO the inc() creates a list of values between 30 and 50 ppm incremented by 5, inc2() does the same between 50 and 90 ppm and inc() between 90 and 175 ppm.
- All “func” functions check the range chosen and call one of “inc” function based on the range.

VOO-

Parameter: Lower Rate Limit

Functions:

- write1(user, value), combine1(value), printlrl(value), store1(value), func1(value), inc(), inc2(), in3()

Parameter: Upper Rate Limit

Recurring Functions:

- write2(user, value), combine2(value), printurl(value), store2(value), func3(value):

Parameter: Ventricular Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Ventricular Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc(value):

AOO-

Parameter: Lower Rate Limit

Functions:

- write1(user, value), combine1(value), printlrl(value), store1(value), func1(value), inc1(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user, value), combine2(value), printurl(value), store2(value), func3(value), uinc()

Parameter: Atrial Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Atrial Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc()

VVI-

Parameter: Lower Rate Limit

Functions:

- write1(user,value), combine1(value), printlrl(value), store1(value), inc(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user,value), combine2(value), printurl(value), store2(value), func3(), uinc()

Parameter: Ventricular Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Ventricular Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc()

Parameter: Ventricular Sensitivity

Functions:

- write5(user,value), combine5(value), printsense(value), store5(value), sensefunc(value), senseinc(value)

Parameter: VRP

Functions:

- write6(user,value), combine6(value), printvrp(value), store6(value), vrpfunc(value), vrpinc()

Parameter: Hysteresis

Functions:

- write7(user,value), combine7(value), printhys(value), store7(value), hysfunc(value), hyinc(), hyinc2(), hyinc3()

Parameter: Rate Smoothing

Functions:

- write8(user,value), combine8(value), printsmooth(value), store8(value)

AAI-

Parameter: Lower Rate Limit

Functions:

- write1(user,value), combine1(value), printlrl(value), store1(value), inc(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user,value), combine2(value), printurl(value), store2(value), func3(), uinc()

Parameter: Atrial Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Atrial Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc():

Parameter: Atrial Sensitivity

Functions:

- write5(user,value), combine5(value), printsense(value), store5(value), sensefunc(value):, senseinc(value):

Parameter: ARP

Functions:

- write6(user,value), combine6(value), printvrp(value), store6(value), arpfunc(value), arpinc()

Parameter: PVARP

Functions:

- write7(user,value), combine7(value), printvarp(value), store7(value), pvarpfunc(value), pvarpinc(value)

Parameter: Hysteresis

Functions:

- write8(user,value), combine8(value), printhys(value), store8(value), hysfunc(value), hyinc(), hyinc2()

Parameter: Rate Smoothing

Functions:

- write9(user,value), combine9(value), printsmooth(value), store9(value)

VOOR-

Parameter: Lower Rate Limit

Functions:

- write1(user, value), combine1(value), printlrl(value), store1(value), func1(value), inc(), inc2(), in3()

Parameter: Upper Rate Limit

Recurring Functions:

- write2(user, value), combine2(value), printurl(value), store2(value), func3(value):

Parameter: Ventricular Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Ventricular Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc(value):

Parameter: Maximum Sensor Rate

Functions:

- write5(user,value),combine5(value),printMaxSenRate(value),store5(value), func5(value), uinc2()

Parameter: Activity Threshold

Functions:

- write6(user,value), combine6(value), printActThresh(value), store6(value)

Parameter: Reaction Time

Functions:

- write7(user,value), combine7(value), printRT(value), store7(value)

Parameter: Response Factor

Functions:

- write8(user,value), combine8(value), printRF(value), store8(value)

Parameter: Recovery Time

Functions:

- write9(user,value), combine9(value), printRecTime(value), store9(value)

AOOR-

Parameter: Lower Rate Limit

Functions:

- write1(user, value), combine1(value), printlrl(value), store1(value), func1(value), inc1(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user, value), combine2(value), printurl(value), store2(value), func3(value), uinc()

Parameter: Atrial Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Atrial Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc()

Parameter: Maximum Sensor Rate

Functions:

- write5(user,value),combine5(value),printmsr(value),store5(value),func4(value), msrinc()

Parameter: Activity Threshold

Functions:

- write6(user,value), combine6(value), printthreshold(value), store6(value)

Parameter: Reaction Time

Functions:

- write7(user,value), combine7(value), printreaction(value), store7(value)

Parameter: Response Factor

Functions:

- write8(user,value), combine8(value), printfactor(value), store8(value)

Parameter: Recovery Time

Functions:

- write9(user,value), combine9(value), printrecovery(value), store9(value)

VVIR-

Parameter: Lower Rate Limit

Functions:

- write1(user,value), combine1(value), printlrl(value), store1(value), inc(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user,value), combine2(value), printurl(value), store2(value), func3(), uinc()

Parameter: Ventricular Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Ventricular Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc()

Parameter: Maximum Sensor Rate

Functions:

- write5(user,value),combine5(value),printMaxSenRate(value),store5(value), func5(value), uinc2()

Parameter: Ventricular Sensitivity

Functions:

- write10(user,value),combine10(value),printsense(value),store10(value), sensefunc(value), senseinc(value)

Parameter: VRP

Functions:

- write11(user,value), combine11(value), printvrp(value), store11(value), vrpfunc(value), vrpinc()

Parameter: Hysteresis

Functions:

- write12(user,value), combine12(value), printhys(value), store12(value), hysfunc(value), hyinc(), hyinc2(), hyinc3()

Parameter: Rate Smoothing

Functions:

- write13(user,value), combine13(value), printsmooth(value), store13(value)

Parameter: Activity Threshold

Functions:

- write6(user,value), combine6(value), printActThresh(value), store6(value)

Parameter: Reaction Time

Functions:

- write7(user,value), combine7(value), printRT(value), store7(value)

Parameter: Response Factor

Functions:

- write8(user,value), combine8(value), printRF(value), store8(value)

Parameter: Recovery Time

Functions:

- write9(user,value), combine9(value), printRecTime(value), store9(value)

AAIR-

Parameter: Lower Rate Limit

Functions:

- write1(user,value), combine1(value), printlrl(value), store1(value), inc(), inc2(), inc3()

Parameter: Upper Rate Limit

Functions:

- write2(user,value), combine2(value), printurl(value), store2(value), func3(), uinc()

Parameter: Atrial Amplitude

Functions:

- write3(user,value), combine3(value), printvamp(value), store3(value)

Parameter: Atrial Pulse Width

Functions:

- write4(user,value), combine4(value), printvpulse(value), store4(value), pulsefunc(value), pinc()

Parameter: Maximum Sensor Rate

Functions:

- write10(user,value),combine10(value),printmsr(value),store10(value),func4(value), msrinc()

Parameter: Atrial Sensitivity

Functions:

- write5(user,value), combine5(value), printsense(value), store5(value), sensefunc(value), senseinc(value)

Parameter: ARP

Functions:

- write6(user,value), combine6(value), printvrp(value), store6(value), arpfunc(value), arpinc()

Parameter: PVARP

Functions:

- write7(user,value), combine7(value), printvarp(value), store7(value), pvarpfunc(value), pvarpinc(value)

Parameter: Hysteresis

Functions:

- write8(user,value), combine8(value), printhys(value), store8(value), hysfunc(value), hyinc(), hyinc2(), hysinc3()

Parameter: Rate Smoothing

Functions:

- write9(user,value), combine9(value), printsmooth(value), store9(value)

Parameter: Activity Threshold

Functions:

- write11(user,value), combine11(value), printthreshold(value), store11(value)

Parameter: Reaction Time

Functions:

- write12(user,value), combine12(value), printreaction(value), store12(value)

Parameter: Response Factor

Functions:

- write13(user,value), combine13(value), printfactorvalue), store13(value)

Parameter: Recovery Time

Functions:

- write14(user,value), combine14(value), printrecovery(value), store14(value)

Every mode has its own file with one function that has the name of the mode and is then imported into the file win2 to group them all in one window. The module begins by passing the valid username to the win2(user) function where the user is greeted by their name and their profile data is shown on screen. The program continues to add each line in the user's text file to an array 'para' where their past entries are shown on screen and printed using one of the "write" functions. If this is the first login on the screen or the user presses the one of the pacing mode buttons shown on screen, they are taken to another window to edit the parameters for the mode selected using several drop-down menus. These drop-down menus consist of menus for selecting ranges and separate menus for setting the parameter values the user wishes to choose within the function. The global imported function "OptionMenu" is used to create drop down menus for the user as part of the configuration GUI. Once the user goes back to the profile data screen the new data is appended to the end of the text file to provide them with a history of their older parameter entries. The functions display (user, mode), opens the text files to read and write values. If the program is closed and the user returns to the program with the same login credentials their profile is loaded with the most recent parameters. Within the 8 mode files, local functions 'write' and 'combine' are passed the username and value to be updated. The user does not have locally defined parameters but rather they are included in the text file. The same method is applied to all modes' information and a press of the refresh button calls the 'refresh' function which opens the user text file and looks for the parameter keywords and extracts the most recent values.

Module 4: Serial Communication

Serial communication is done through the public function upload( ) which opens the text file for the user logged in, and imports the data to the pacemaker. It uses several private functions listed as follows:

- serial.Serial(), takes 2 arguments, the port number and the number, and uses them to identify which USB port to access the pacemaker.
- ser.isOpen() simply opens the port

- ser.write() uploads the values passed onto the port opened
- ser.close() closes the port opened with ser.isOpen()

The upload function opens the text file and inputs and extracts the parameter values selected then appends them into a list called “para”. Every element in the list is then converted into an integer value and uploaded to the pacemaker using the function “ser.write()”. The global upload function uses the try and except error handling method as an exception handler to create a new tkinter window s1 and display an upload failure message. The public function egram(), is used to display the voltage graph that would be displayed on an oscilloscope if connected to the board. The function animate(i), refreshes the plot created by egram(), and the function streamdata() uses the private functions listed above to get the data from the pacemaker and pass it to egram() to graph on the DCM.