DCM Documentation

Part 1: Requirements and design

1.1 Requirements

1. The Device Controller-Monitor (DCM) must be capable of utilizing and managing windows based UI to allow users to monitor and control the pacemaker device.
2. The UI should allow users to interact with the application using their mouse and keyboard.
3. The DCM must include a welcome screen where users have the option to either sign in with a username & password, or to register as a new user. There should be a maximum of ten users stored locally.
4. The user interface must store all programmable parameters for the pacemaker, display them to the user, and allow for their modification. Programmable parameters include:
   Lower Rate Limit, Upper Rate Limit, Atrial Amplitude, Atrial Pulses Width, Ventricular Amplitude, and Ventricular Pulse Width, Ventricular Refractory Period (VRP), Atrial Refractory Period (ARP), Post-Ventricular Atrial Refractory Period (PVARP), and Hysteresis rate limit.
5. Parameters must only be modified according to the following constraints:

Parameter	**Programmable Values**	**Increment**	**Default**
Mode	0,1,2,3 (AOO, VOO, AAI, VVI)	1	0 (AOO)
Lower rate limit	30-50 ppm 50-90 ppm 90-175 ppm	5 1 5	60 ppm
Upper rate limit	50-175 ppm	5	120 ppm
Atrial Pulse Amplitude	2.5V – 5.0 V	0.5 V	2.5 V
Ventricular Pulse Amplitude	2.5V – 5.0 V	0.5 V	2.5 V
Atrial Pulse Width	0.3 – 1.9 ms	0.1 ms	0.4 ms
Ventricular Pulse Width	0.3 – 1.9 ms	0.1 ms	0.4 ms
Atrial Refractory Period	150-500 ms	10 ms	250 ms
Ventricular Refractory Period	150-500 ms	10 ms	320 ms
Post-Ventricular Atrial Refractory Period	150-500 ms	10 ms	320 ms
Hysteresis rate limit	Off or same as lower rate limit	-	Off

6. The user interface should clearly indicate when the device and DCM are communicating.
7. The user interface should indicate when a device is connected that has not previously been connected.
8. The DCM must include interface to present all the pacing modes.

1.2 Design Decisions

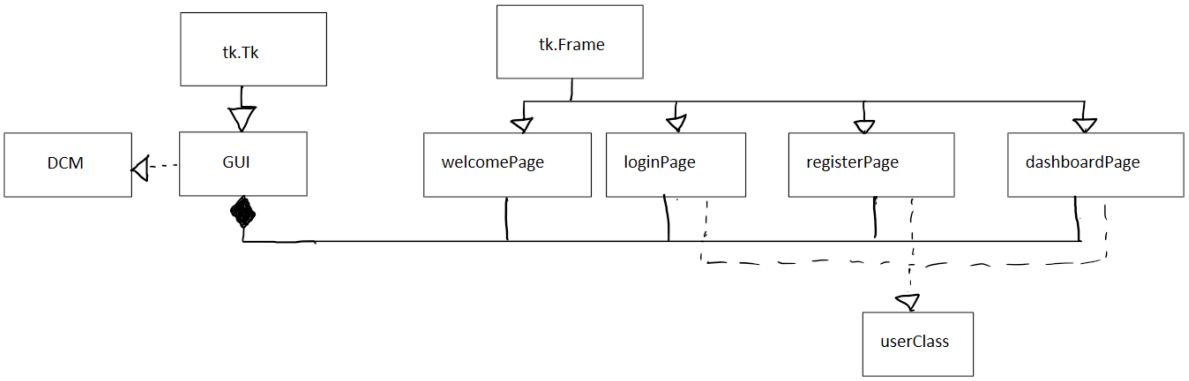
The set of allowed values for both pulse widths and amplitudes was reduced to ensure reliable pacing. Voltages below 2.5V may not be sufficient to induce a pace when the pulse is shorter than 0.3 ms. When voltage and pulse width are greater than 2.5 V and 0.3 ms respectively then reliable pacing is ensured. This was determined from the strength duration curve in section 3.4 of the pacemaker_shield_explained document.

We decided to use the tkinter library to implement our GUI because it is well documented, we had some experience using it, and it was recommended.

We laid out our user interface into welcome, login, register, and dashboard pages because it was the simplest layout that aligned with the third requirement.

We postponed implementing a date structure in our modules because we did not use any date information up to this point and implementing it without sufficient information could lead to future complications.

We structured our design into the following conceptually related modules:



Part 2: Future flexibility and modules

2.1 Requirements likely to change

In the future, more modes will need to be added and the DCM will interface with the Simulink model using serial communication. Electrocardiogram data may also need to be collected, stored, and displayed. When serial communication is implemented, we will need to show the status of the connection.

2.2 Design decisions likely to change

Our decision to delay implementation of a date structure will change when we begin working with electrocardiogram data and it is required.

We will likely need to add more pages when more features are added so that the dashboard does not become overcrowded and unclear.

The storage of electrocardiogram data may require data for each user to be stored in a folder instead of a single text file.

2.3 MIS and MID

2.3.1 Module Interface Specification

2.3.1.1 DCM Module Interface Specification

The DCM must provide a user interface that presents and allows the modification of pacemaker parameters according to requirements listed in 1.1. The DCM's input and output will be entirely graphical. This interface will be handled by the GUI module, which will display all other pages. Other pages will each have their own module. In addition to a module for each page, the DCM will store each user's data separately.

2.3.1.2 GUI module Interface Specification

The GUI module will display each page to the user. The welcome page will be displayed by default. Transitions between pages will be handled in the module of the page currently being displayed. The GUI module will update the dashboard module on login to set the current user and load their dashboard.

2.3.1.3 Welcome Page Module Interface Specification

The welcome page module will display options for the user to login or register. These will lead respectively to the login page or register page module being displayed by the GUI.

2.3.1.4 Login Page Module Interface Specification

The login page module will allow the user to enter a previously registered username and password. If the username and password match an existing user, then that user's data will be loaded, and the GUI will show the dashboard module. If the username is not found or the password is incorrect this will be communicated to the user, and they will be allowed to try again. There will be no limit on the number of login attempts allowed. A back button that returns the user to the welcome page will be included.

2.3.1.5 Register Page Module Interface Specification

The register page module will allow the user to enter their username and password. It will ensure their username has not already been taken, in which case it will communicate this and allow them to try again. If their username is unique a new user file will be created, and default values will be stored for each parameter. After a new user file is created, its data will be loaded and the dashboard for that user will be shown by the GUI. A back button that returns the user to the welcome page will be included.

2.3.1.6 User Module Interface Specification

The user module allow access to and verify the modification of each user's data. Each user will have a username, password, pacing rate, pacing mode, ventricular pulse width and amplitude, atrial pulse width and amplitude, upper and lower pacing rate limits, atrial refractive period, ventricular refractive period, post-ventricular atrial refractive period, and hysteresis rate limit. Excluding a user's username and password, which will be immutable, each parameter will have an associated method that allows its modification to a value meeting these required conditions:

Parameter	**Programmable Values**	**Increment**	**Default**
Mode	0,1,2,3 (AOO, VOO, AAI, VVI)	1	0 (AOO)
Lower rate limit	30-50 ppm 50-90 ppm 90-175 ppm	5 1 5	60 ppm
Upper rate limit	50-175 ppm	5	120 ppm
Atrial Pulse Amplitude	2.5V – 5.0 V	0.5 V	2.5 V
Ventricular Pulse Amplitude	2.5V – 5.0 V	0.5 V	2.5 V
Atrial Pulse Width	0.3 – 1.9 ms	0.1 ms	0.4 ms
Ventricular Pulse Width	0.3 – 1.9 ms	0.1 ms	0.4 ms
Atrial Refractory Period	150-500 ms	10 ms	250 ms
Ventricular Refractory Period	150-500 ms	10 ms	320 ms
Post-Ventricular Atrial Refractory Period	150-500 ms	10 ms	320 ms
Hysteresis rate limit	Off or same as lower rate limit	-	Off

The user module will also enable values to be stored after their modification.

2.3.1.7 Dashboard Module Interface Specification

The dashboard module will display all current parameter values to the user. It will allow users to enter a new value for each of them and attempt to modify it using the associated method in the user module. If the modification is allowed it will notify the user and update displayed values, otherwise it will display a message that the entry was invalid. It will also display conditions for each parameter to be valid as described in 2.3.1.6. The dashboard module will also have a back button to log out and return to the previous screen.

2.3.2 Module Internal Design

2.3.2.1 DCM Module Internal Design

The DCM Module will create an instance of the GUI module and show the GUI.

2.3.2.2 GUI Module Internal Design

The GUI module uses the Tk interface (tkinter) library included with python. It inherits from the main Tk () class. This is the class that creates a window on the user's screen. The GUI module will contain a dictionary with instances of each page module. On initialization the GUI will show the welcome page.

Attribute	Description
frames: Frame []	Dictionary containing instances of each page class.
Container: Frame	Parent frame to contain pages

Method	Description
show_frame(frame)	Show the requested frame from frames using tkraise() from tkinter.
load_dashboard(User)	Calls the setUser then the load_user_info methods of the dashboard module.
Init ()	Calls parent init function, sets window geometry, packs Container to the main window, and populates frames with an instance of each page.

2.3.2.3 Welcome Page Module Internal Design

The welcome page module inherits from tkinter's Frame module. It has a button to login that shows the login page frame, and a register button that shows the register page frame. It also has a label that says welcome.

Attribute	Description
controller	A reference to the GUI, allowing control from one object & decoupling the modules.

Method	Description
Init (parent, controller)	Calls the parent init function, creates the two buttons which use the controller to call show_frame from the GUI module for their respective pages.

2.3.2.4 Login Page Module Internal Design

The login page module inherits from tkinter's Frame module. It displays two entry boxes, labelled username and password, a button to login, and a label to display error messages. It also has a label that says login page.

Attribute	Description
controller	A reference to the GUI, allowing control from one object & decoupling the modules.

Method	Description
Init (parent, controller)	Calls the parent init function, creates the two entries, label, a button that calls attempt_login with the values in the two entries, and a back button.
Attempt_login(username, password)	Compares the username to user files in the Users folder. If username is found then that user's data is read, and passwords are compared. If the password matches the one stored, then the controller is used to call GUI's load_dashboard and show_frame(DashboardClass). If the username or password, do not match then the label's text is updated to tell the user.
backButtonCommand()	Hide the login page to bring the user back to the welcome page.

2.3.2.5 Register Page Module Internal Design

The register page inherits from tkinter's Frame module. It displays two entry boxes for the user to enter their username and password, a button to register, and a label to display error messages. When the button is pressed, values are read from the two entries. If the username is not already a user in the Users file, then a new user is created with the given username, password, and default values for all other parameters.

Attribute	Description
controller	A reference to the GUI, allowing control from one object & decoupling the modules.

Method	Description
Init (parent, controller)	Calls the parent init function, creates the two entries, label, and button that calls attempt_register with the values in the two entries, and a back button.
Attempt_register(username, password, message_box)	Compares the username to user files in the Users folder. If username is found, then the label will show the user that the user already exists. The number of the user will also be check. If it exceeds 10, the label's text is updated to tell the user that maximum of 10 Users already created. If these two conditions mentioned above do not happen, this method will call createUserFile(). If createUserFile returns false an error message is

	shown, otherwise the new user's dashboard is loaded and shown.
createUserFile(username, password)	Attempts to create a new file for the new user with their username and password in the Users folder. It also writes all the default parameters to the file according to the chart in section 2.3.1.6. It returns True on success and false if a file could not be created with the given username.
backButtonCommand()	Hide the register page to bring the user back to the welcome page.

2.3.2.6 User Module Internal Design

Attribute	Description
controller	A reference to the GUI, allowing control from one object & decoupling the modules.
File_found: bool	Boolean storing whether or not user data was successfully read from the user file on initialization
Username: str	The user's name
Password: str	The user's password
pacingRate: int	The number of paces to be delivered per minute
Mode: int	The index of the selected mode in the list [AOO, VOO, AAI, VVI]
ventPulseWidth: float	The duration in milliseconds of the pulse to the ventricle
ventAmplitude: float	The amplitude in volts of the pulse to the ventricle
atrialPulseWidth: float	The duration in milliseconds of the pulse to the atrium
atrialAmplitude: float	The amplitude in volts of the pulse to the atrium
upperRateLimit: int	The highest number of beats per minute
lowerRatelimit: int	The lowest number of beats per minute
ARP: int	The duration in milliseconds after an atrial event where any further atrial events should not trigger or inhibit pacing (for single chamber pacing modes only).
VRP: int	The duration in milliseconds after a ventricular event where any further ventricular events should not trigger or inhibit pacing (for single chamber pacing modes only).
PVARP: int	The duration in milliseconds after a ventricular event where any atrial event should not inhibit atrial pacing or trigger a ventricular pace (for

	ventricular pacing atrial sensing modes, not used in assignment one)
hysteresisRateLimit: int	The rate above which hysteresis pacing will be activated (not used in assignment one)

Method	Description
Init(user_file)	Creates an instance of User from data stored in user_file. If the file is not found or data cannot be read then the attribute file_found is set to false, otherwise true. Attributes for username, password, and all other parameters are assigned from the user file, line by line, in the order listed in the table above.
getUsername	Return username
OverwriteUserData(self)	Set the correct path for username in the User file with writing mode to change each parameter for the following username.
setPacingRate (self, rate)	SetPacingRate calls rate and Pacingrate be between lowerRatelimit and upperRatelimit which are set in setUpperRateLimit and setLowerRateLimit functions.
setMode (self, rate)	SetMode will determine the mode among “AOO”,” VOO”,” AAI,” “VVI” by entering the mode number that we assigned in the function. (mode 1 = “AOO”, mode 2 = “VOO”, mode 3 = “AAI”, mode = “VVI”)
setVentPulseWidth (self, width)	This setVentPulseWidth calls width and set the values of the Width between range of the allowed values from 0.3 to 1.9 (increasing by 0.1)
setVentAmplitude (self, amp)	This setVentAmplitude calls amp and set the values of the amplitude between the range of the amplitudes from 2.5 to 5.0 by increasing every 0.5
setAtrialPulseWidth (self, amp)	This setAtrailPulseWidth calls amp and set the value of the Atrialamplitude between the range from 2.5 to 5.0, increasing by 0.5
SetUpperRateLimit(self, upper)	This SetUpperRateLimit calls upper and set the value of the upper limitation that should be between 50 and 175 or bigger than current pacing rate. The value should be divisible by 5.
SetLowerRateLimit(self, lower)	This SetLowerRateLimit calls lower and set the value of the lower limitation that should be between 30 and 175 or less than current pacing rate. The values should be also between 50 and 90 and divisible by 5.

SetARP(self,val)	This setARP calls val and this makes val be set the ValInt between 150 and 500, and ValInt should be divisible by 10.
setVRP(self, val)	This setVRP calls val and this makes val be set between 150 and 500. Also, this val should be divisible by 10.
SetPVARP(self,val)	This PVARP calls val and this makes Val be set between 150 and 500 and this value should be divisible by 10.
SetHysteresisRateLimit(self, limit)	This SetHysteresisRateLimit calls limit, and this limit should be between 30 and 175. The limit also should be between 50 and 90 and divisible by 5.

2.3.2.7 Dashboard Page Module Internal Design

Attribute	Description
controller	A reference to the GUI, allowing control from one object & decoupling the modules.
Message_box	A label whose text can be updated to notify users when an entry is invalid
nameLabel, rateLabel, modeLabel, etc	All labels are attributes so their text can be modified when a user logs in or modifies a parameter.

Method	Description
Init (parent, controller)	Call the parent init function. Create a label to show which user is logged in. Create a back button. Create a table displaying each parameter's name, value, an entry box, a button to modify it, and an explanation of allowed inputs. Set the current user to None (One will be set when the user logs in/registers).
changeMode(mode: str, message_box)	Calls the current User's setMode method. If successful, updates the shown values with load_user_info and displays a success message in the message box, otherwise shows an error message.
changeRate(rate: str, message_box), changeVentPW(width: str, message_box), etc	Same as changeMode, but calling the appropriate method from the User module.

backButtonCommand()	Hide the dashboard frame and set the current User to none.
Load_user_info()	Updates all parameter labels to show their current value

2.4 Testing

2.4.1 GUI Testing

From each screen all buttons linked to functions that showed or hid frames were tested. The only bug encountered was that the dashboard was shown before a user's data had been loaded to it. This was fixed by calling the load_user_data in load_dashboard.

2.4.2 Login & Register Testing

File path errors led to login and register modules failing. This was fixed by running the DCM module from the correct working directory (3K04-PACEMAKER-PROJECT not DCM).

Newlines added when writing user data led to errors when comparing entered password to stored password. Fixed by stripping newlines when reading from user file.

Tested logging in, out, and back in with the same and different users.

Tested registration with the new users ,and the existing users cannot create the same user account in the registration.

When the number of users become 10 users, the new users cannot be registered in the registration and shows the comments "Maximum 10 Users already created". This satisfies the all the Login and Register requirements. No errors were encountered.

2.4.3 Dashboard (and User) Testing

Tested each modification with correct, incorrect, and absent values.

Encountered type errors when testing parameter modification. Fixed by adding type conversion of entered values in the try-catch section of the User module's change parameter methods.

Values did not initially change on dashboard even when success message was shown. Fixed by adding calls to load_user_data.