SFWR 3K04: Design Document Revision 3

Group Alpha

Jain, Tarun: jaint

Puertsen, Mitchell: puerstmw

Ponce Morales, Gustavo: poncemg

Khawaja, Wajih: Khawajwm

Jianpeng Liu: liu433

Hurtado Carreon, Andres: hurtada

Professor: Dr. Alan Wyssang

Assignment 3

Contents

1 REVISION HISTORY 4

2 GENERAL INFORMATION AND INTRODUCTION 5

2.1 Motivation 5

2.2 Summary of Project 5

2.3 Summary of Design Choices 5

3 ANTICIPATED CHANGES IN REQUIREMENTS AND DESIGN 6

4 MODULE HIERARCHY 7

6 MODULE DECOMPOSITION 8

6.1 Main Module 8
Purpose: 9
Secrets: 9
Behaviour: 9

6.2 Mode Module 9
Purpose: 9
Secret: 9
Public functions with parameters: 10
Black box behavior: 10
Global variables: 11
Private functions and descriptions: 11
Internal behavior of each function: 11

6.3 Parameters Module 13
Purpose: 13
Secrets: 13
Public functions with parameters: 13
Black-box behaviour description: 14
Private functions and variables: 14
Internal behaviour of each function: 15

6.4 paceNow Module 15
Purpose: 15
Secret: 15
Public functions with parameters: 15

6.5 GUI Display Module 16
Purpose: 16
Secret: 16

Public functions with parameters: 16
Black-Box behaviour description: 16
Global variables: 16
Private functions: 16
The parent module: 17
The child module: 17

6.5.1 GUI Serial communication: 17
Purpose: 17
Global variables: 18
Public functions with parameters: 18

Table 1: Table of Revision History 4
Table 2: Requirement and Module Trace 8

Figure 1: Module Hierarchy Overview 7
Figure 2: Visual Representation of Main Module Implementation 8
Figure 3:Visual Representation of Mode Module Implementation 9
Figure 4:: Visual Representation of Rate Adaptive Pacing 12
Figure 5: Visual Representation of Parameters Module Implementation 13
Figure 6: Visual Representation of paceNow Module Implementation 15
Figure 7:: Visual Representation of GUI Display Module 16

1 Revision History

Date	Revision	Author	Comment
10/10/2016	0.0	All	Created document and populated fields
11/16/2016	0.1	All	Added points on Egram production, rate adaptation, and accelerometer use Section 3 - AC6 added Section 6.3 - Second section on serial communication added Section 6.4 – E-gram information added
11/25/2016	1.0	All	• Added communication methods between GUI and code within the board, as well as parameter updating methods. • Finished designing and implementing closed process within the board. • Increased modularity of the software by separating the closed loop from the pacing options. Section 2.3 - Communication info added Section 4 – Module Hierarchy Figure Section 5 – Design Requirements Connection Section 6.1 – Implementation Section 6.2 – Implementation Section 6.3 – paceNow Section 6.4.1 – Addition of Serial Communication AC 6 – Tolerance Intervals for Egram AC 7 – Change in Hardware
12/11/16	2.0	All	• Implemented accelerometer data printout Section 6.2: Pacing modes • VVI mode implementation Section 6.5: GUI • Black box functionality through the GUI and E-gram data display in real time • Serial Communication • Child modules updated to align with current implementation

Table 1: Table of Revision History

2 General Information and Introduction

The purpose of this design document for the Pacemaker Project is for the potential readers of the document. The following are the stakeholder in the project as well as their respective stakes:

- Developers: This document allows developers to communicate and document module specifics. It also outlines the trace for the requirements thus aiding in helping future designers of the project to be able to determine and alter product design according to the requirements and to satisfy them.
- New Project Members: The document also helps new project members quickly understand the entire scope of the project
- Maintainers: Individuals who would be maintaining the system in the future will be guided as to where the anticipated design and requirement changes so future maintenance can take place without causing the system to "crash". Additionally, by referring to the hierarchy and relationships within the system it can allow an insight to the risks for future changes.

2.1 Motivation

A large system should be decomposed into smaller modules and thus smaller sub- modules is crucial for large system design. Additionally, developing in a multiple modular design allows for many developers to simultaneously add and edit the program and further the overall development of the design. Furthermore, modularity allows for changes and updated in the future to be done without re-writing the overall process.

2.2 Summary of Project

The main purpose of the pacemaker project is in accordance to the system requirements to develop a dual chamber, rate adaptive bradycardia pacing support with varying pacing modes in addition to historical device performance data along with user diagnostics to help healthcare practitioners assess and help the patients. This will be implemented through the use of a FRDM K64F board with the online mBed compiler.

2.3 Summary of Design Choices

The design was based on the modularity of the modes in which this device operates. The device can choose from multiple modes, which in turn take different parameters. This allows for all the pacing modes to be encapsulated within one module that defines the general behavior for all of them. Additionally, it was decided that a separate module would contain the parameters accessible to the mode functions. This was chosen to simplify the transition of the behavior of the system due to changes in the parameters values. In practice, the separation of parameters and modes in their respective modules provides a simplistic approach to anticipated changes in the requirements for both. This design also allows for easy communication between the user GUI and K64F, giving each portion of code one small task to handle at a time. This has made the system significantly more robust, and more effective overall.

3 Anticipated Changes in Requirements and Design

During the composition of the Pacemaker system design changes were taken into consideration. Anticipated changes are changes that can be identified such that these elements are hidden within the system module, as such they are easy to change without affecting the overall functionality of the program.

The following are the anticipated changes as well as how we have integrated them into the design to account for these changes.

AC1: Addition of several more permanent pacing modes

a) Set the simplest modes as functions with parameters for easy calling
b) Set all parameters to their default values in main for easy access
c) Applies a modular design with all modes under a modes module

AC2: Updates to the GUI

a) GUI designed separately
b) C++ code used to design the GUI for best compatibility
c) GUI changes values in a separate file as opposed to the program directly

AC3: Adjustments to the modes as well as the parameters

a) All modes set as functions with parameters to easily change the values applies
b) Functions allow them to be called in other, more complex modes for reuse
c) An extra parameter added to parameters to allow for arrays of N length
d) All parameters are saved separately within a .txt file for easy access as well as adjustment

AC4: Changes to the circuit

a) Used multi-purpose PTD output pins that can be used as inputs for sensing
b) Used one set of pins, close together, to minimize access wiring
c) Used breadboard for easy changes as needed

AC5: Connectivity Module

a) Highly modular design with minimal dependency
b) Main characteristics of program for outputs designed into functions
c) All code in C++ to allow for high compatibility

AC6: Tolerance Intervals for Egram signal interpretation

a) Sampling of heart beat and recording converted values
b) Graphing recorded converted values using saved values
c) Using values to help orient the program as to which section of the heartbeat it is looking at

AC7: Change in Hardware

a) Input is taken via the mBed library for the k64f to utilize onboard accelerometer
b) Use of board features maximized to minimize loss of capability and maximize economy
c) Main code written externally to the mbed compiler (in C++), making it board-independent.

4 Module Hierarchy

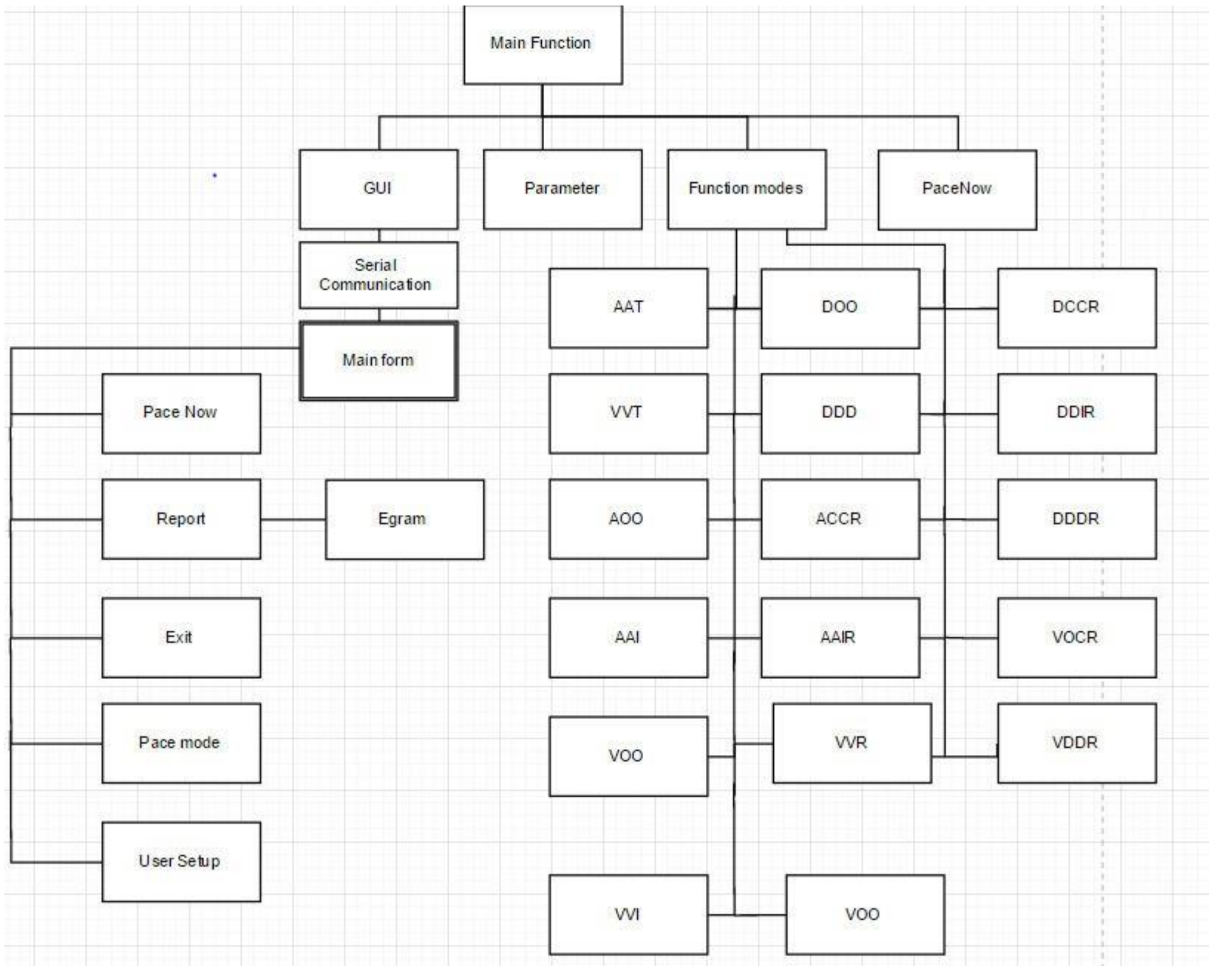


Figure 1: Module Hierarchy Overview

5 Design and Requirements Connection

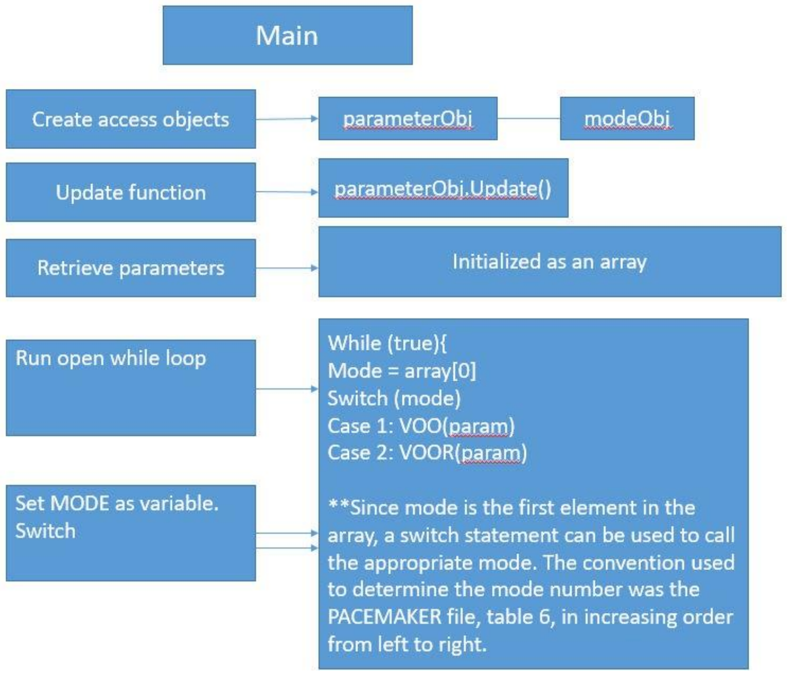
The system is modularized and analyzed so that it is in accordance with the system requirements. The relationship between the modules and requirements is shown below as follows:

Requirements	**Modules**
3.1-3.2 DCM	GUI Module Connectivity Module
3.4 Pacing Pulses	Main Module
3.5-3.6 Modes and States	Mode Module
4.7 Real Time Electrograms	GUI Module Mode Module
5 Modes and their programmable parameters	Mode Module Parameter Module
Appendix A Programmable Parameters	Main Module Mode Module

Table 2: Requirement and Module Trace

6 Module Decomposition

6.1 Main Module



```
While (true){
Mode = array[0]
Switch (mode)
Case 1: VOO(param)
Case 2: VOOR(param)
```

**Since mode is the first element in the array, a switch statement can be used to call the appropriate mode. The convention used to determine the mode number was the PACEMAKER file, table 6, in increasing order from left to right.

Figure 2: Visual Representation of Main Module Implementation

Purpose:

The main module implements a while loop to determine the required mode and switches the mode based upon the parameters through accessing the mode and parameter modules. It also updates the parameters along with gathering the accelerometer data through the while loop to use for the rate adaptive pacing functionality.

Secrets:

None.

Behavior:

The main module accesses the mode and parameters module through private objects, then runs and updates the parameters to ensure the requested pacing is done.

6.2 Mode Module

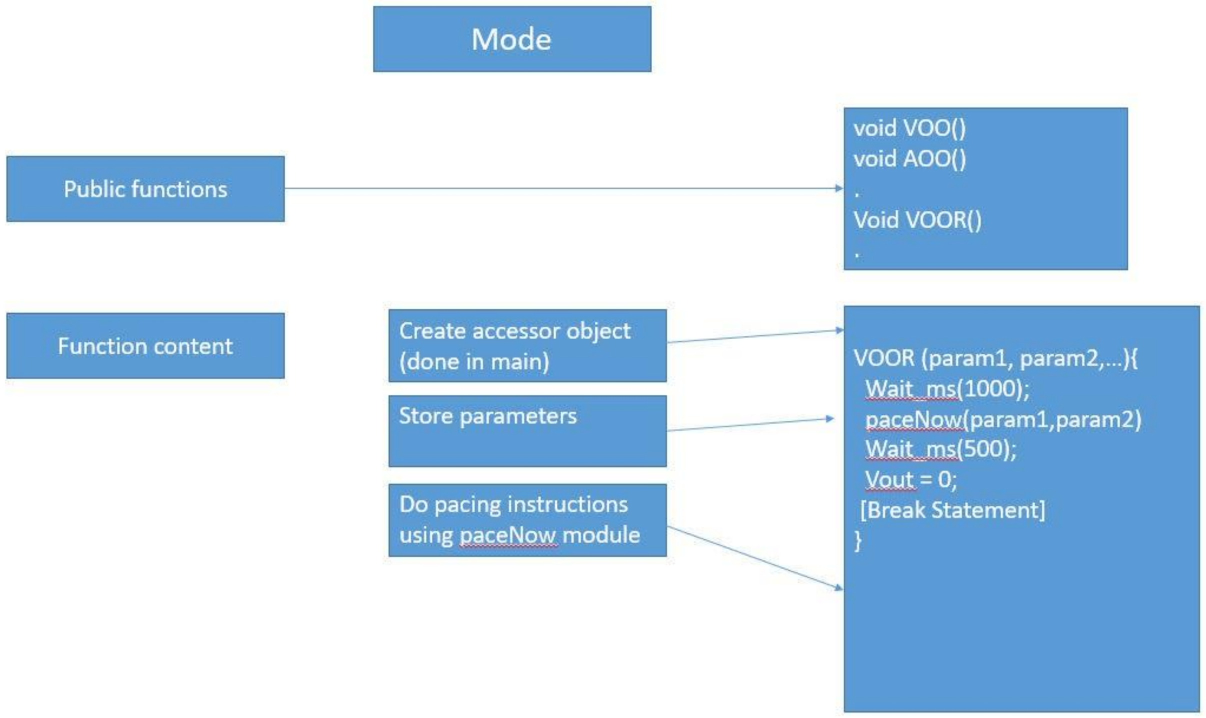


Figure 3:Visual Representation of Mode Module Implementation

Purpose:

Contains all permanent modes of the pacemaker, including VOO, AOO, DOO, VVI, VVIR all designed as functions to allow for reuse in more complex modes.

Secret:

Creation and utilization of pacing modes based on inputs

Public functions with parameters:

Global Variables:
LRL: Lower Rate Limit = 60
URL: Upper Rate Limit = 120
APW: Atrial Pulse Width =
0.4ms
VPW: Ventricle Pulse Width = 0.4ms
FAVD: Fixed Atrial-Ventricular Delay = 150ms

Leads:
VAR = 3.5
AAR = 3.5

VOO: Paces ventricle on repeated required cycle without
sensing Parameters: LRL, VPW, VAR

AOO: Paces atrium on repeated required cycle without
sensing Parameters: LRL, APW, AAR

DOO: Paces ventricle and atrium on repeated required cycle without
sensing Parameters: LRL, URL, APW, AAR, VPW, VAR, FAVD

VOOR: Paces ventricle with rate adaptive pacing
Pacing cycle occurs faster if repeated movement is detected by the accelerometer however
it is limited by the maximum sensor rate.
Parameters: LRL, VPW, VAR, MSR, AT, ReaT, RF, RecT, FC

VVI: Pace ventricle if pace is not sensed within a given time frame
If pace is sensed at the appropriate time, no pace is sent out
Parameters: LRL, VPW, VAR, SENSITIVITY, HYS, sample

VVIR: Pace ventricle if pace is not senses within a given time frame
Increases rate based on movement
Time given between sensing decreased as activity increases
Parameters: LRL, VPW, VAR, MSR, AT, ReaT, RF, RecT, FC, SENSITIVITY, HYS, sample

Black box behavior:

The mode module consists of multiple functions, each with a specific instruction set geared toward each different pacing setting and its respective requirements. By selecting a "mode" using the interactive GUI, the medical practitioner can specify what the patient requires in terms of pacing from functions within the module. If "VOO" were selected from the mode module, the ventricle would be paced at a predetermined cyclic rate without sensing any part of the heartbeat. If required, the mode can easily be changed by selecting a new one using the interactive GUI. Should the patient actually require AOO instead, it could be substituted using the mode module, and the atrium would be paced accordingly instead of the ventricle. These requirements are in accordance with the 3-4 letter mode specified.

This critical mode module has been implemented in order to provide simplicity in choosing which pacing mode to use. It also provides an organized structure that can be easily found, and modified if design specifications change later in the project. With this module designed and out of the way, the only coding left will be in sensing the heartbeat, and sending that to the module for it to handle.

Global variables:

These have been avoided to allow for better modularity

Private functions and descriptions:

No private functions currently implemented

Internal behavior of each function:

VOO:

1. Uses the LHL to calculate time in milliseconds and waits
2. Calculates output voltage form given VAR
3. Applies output voltage
4. Waits VPW
5. Resets output to zero

AOO:

1. Uses the LHL to calculate time in milliseconds and waits
2. Calculates output voltage form given AAR
3. Applies output voltage
4. Waits APW
5. Resets output to zero

DOO: (To be adjusted to apply AOO and VOO)

1. Uses the LHL to calculate time in milliseconds and waits
2. Calculates output voltage form given VAR
3. Calculates output voltage form given AAR
4. Applies output voltage to Ventricle outputs
5. Waits VPW
6. Resets output to Ventricle to zero
7. Waits FAVD
8. Applies output voltage to Atria outputs
9. Waits APW
10. Resets output to zero

VOOR:

1. Behaves similarly to VOO
2. Intakes data from the accelerometer to determine if the pacing rate should be increased.
3. Adjusts the pacing rate accordingly.
4. Increases rate to maximum pre-specified value over a period also pre-specified
5. Detects lack of activity through accelerometer
6. Lowers heart rate to resting value over pre-set time period, approximately 5 min

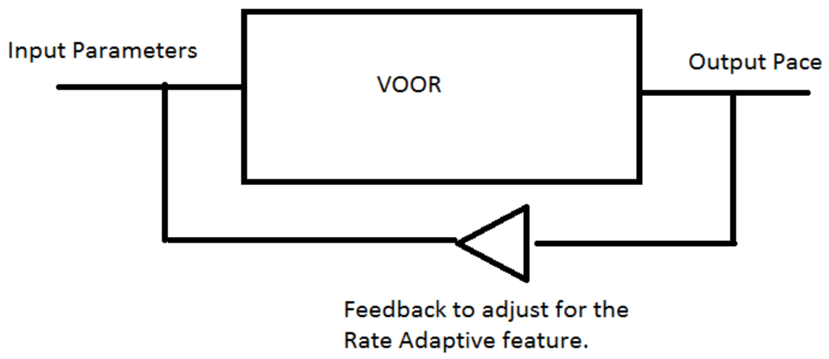


Figure 4: Visual Representation of Rate Adaptive Pacing

VVI:

1. Modified version of VOOR
2. Waits for a ventricular beat
3. If beat is detected within a specified amount of time, no further action is required
4. If no beat is sensed then it will send out a signal to force a beat

VVIR:

1. Combination of VVI and VOOR essentially
2. Detects accelerometer activity
3. If no activity then behaves exactly as VVI by waiting for sensed heartbeat
4. If no beat is detected within a specified time, it sends out a signal to forces a beat
5. If a beat is sensed then nothing happens
6. If it detects accelerometer activity then is begins to decrease the waiting time
7. If no beat is sensed within the reduces waiting time it sends a signal for force a beat
8. If beat is sensed within the time, indicating heartbeat is increasing on its own, then no signal is sent
9. If accelerometer activity is not sensed then waiting between sensing begins to increase
10. More time is given for the heart to beat
11. If no beat is sensed then a signal is sent out to force a beat

6.3 Parameters Module

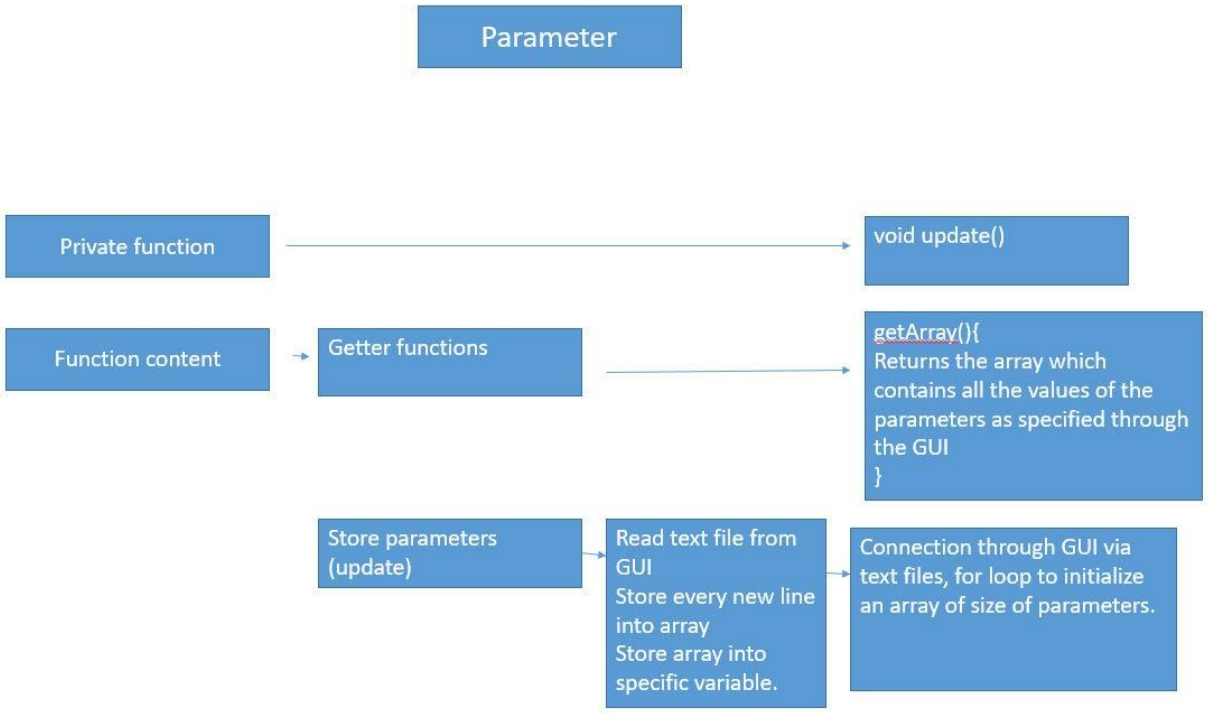

Parameter
Private function
void update()
Function content
Getter functions
getArray(){
Returns the array which contains all the values of the parameters as specified through the GUI
}
Store parameters (update)
Read text file from GUI
Store every new line into array
Store array into specific variable.
Connection through GUI via text files, for loop to initialize an array of size of parameters.

Figure 5: Visual Representation of Parameters Module Implementation

Purpose:

The parameters module holds in itself the variables to be accessed by the Mode and Main modules. This module allows for the separation of concerns with regards to the values of the parameters with which each mode operates.

Secrets:

Hides all parameter values after reading form specialized .txt file

Public functions with parameters:

Public functions and descriptions:

update: No Parameters

Stores lines of data in a temporary file
Checks if update file already exists
Loops through an array to store relevant data points

parameterSet: No Parameters

Initializes variables for mode parameters
Stores two integers that are required for VOOR time scaling for accelerometer

getArray: No Parameters

Returns the parameter from the update function

getLength: No Parameters

Returns the number of parameters used

Black-box behavior description:

Black box behavior for all getter functions: Once it is called (in the main, through an object) it returns the number stored in the variable, to be used as parameters in the modes.

Private functions and variables:

Function: Update

Updates the current values for the private variables to the ones changed by the user through the GUI.

No parameters. (void function which changes values).

Variables

LRL

URL

APW

AAR

VPW

VAR

Internal behavior of each function:

Private Function:	Internal Behavior
Update:	Updates the current values for the private variables to the ones changed by the user

6.4 paceNow Module

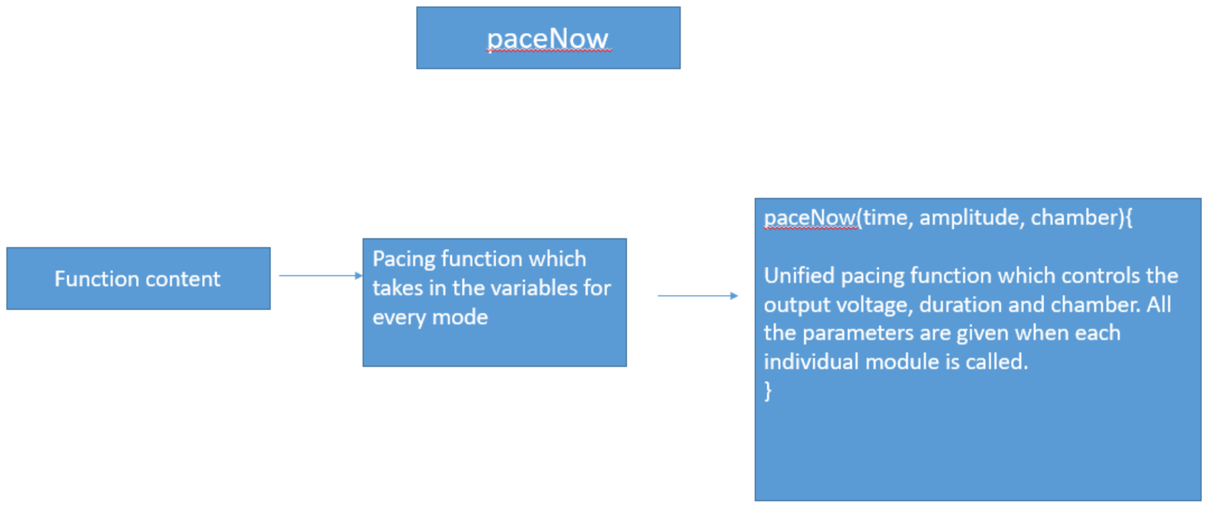

paceNow
Function content
Pacing function which takes in the variables for every mode
paceNow(time, amplitude, chamber){
Unified pacing function which controls the output voltage, duration and chamber. All the parameters are given when each individual module is called.
}

Figure 6: Visual Representation of paceNow Module Implementation

Purpose:

To initial values for pacing modes.

Secret:

Currently none.

Public functions with parameters:

Function	Parameters:
paceNow	PW
	AR2
	Chamber

6.5 GUI Display Module

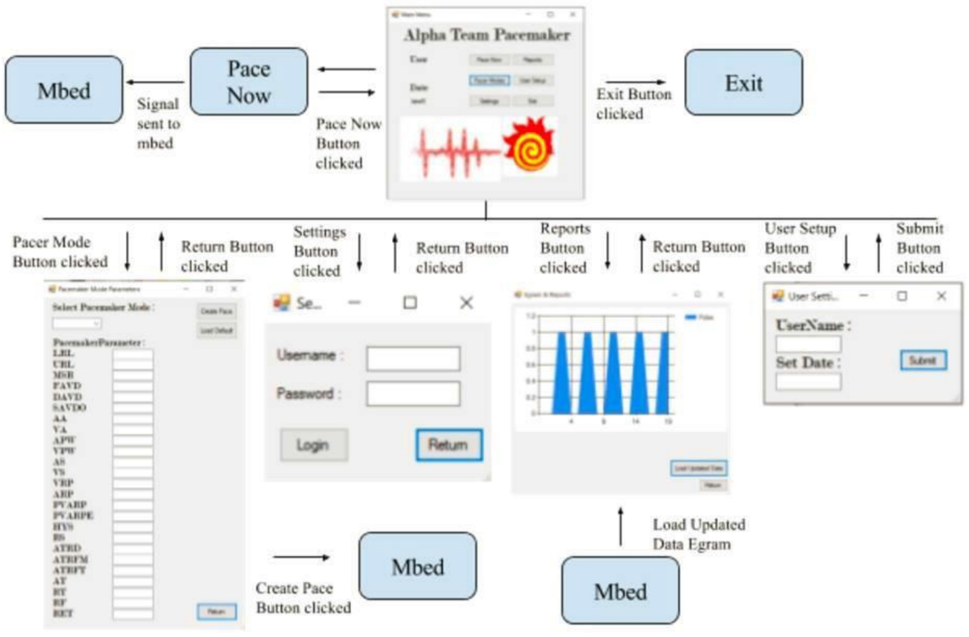


Figure 7:: Visual Representation of GUI Display Module

Purpose:

The purpose of this module is to display the GUI on the DCM which in our case is the computer, and allowing for interaction with the mBed to change the various parameters within the predetermined design requirements for the Pacemaker system.

Secret:

There are no secrets to the module as of yet.

Public functions with parameters:

There are no public functions in this module

Black-Box behavior description:

The GUI module's black box behavior is as follows. The GUI receives inputs from the user and outputs a text file with updated parameters that the mBed's main program will read and execute through serial communication with the computer.

Global variables:

The GUI uses no global variables due to the possibility of the program misinterpreting the variables and not executing properly.

Private functions:

Each of the child forms in the GUI have their own private functions that include button clicks, loading of forms, reading text boxes and other GUI components.

The GUI for the Pacemaker project has two main parts the parent module and the child modules.

The parent module:

Composed of a Main Menu module that has different buttons that lead to the child modules. Such buttons include Pacer Modes, Serial COM, User Setup, and Reports. Each of these child modular forms have their respective functions in the User Interface of the Pacemaker. Furthermore, it also displays important information about the Pacemaker including date and username of the patient.

The child module:

Pacer Modes: The following form allows the user to select different Pacer modes as well as updating the multiple parameters for the mBed to give the correct pace. This child module is composed of one form that has a dropdown menu with all pace modes and textboxes next to the respective parameters where a pace can be generated and transmitted to the pacemaker through serial communication.

Serial COM: The following module allows the user to initialize serial communication with the mBed board. Furthermore, it can send and read required data to produce paces and proper reports. Finally, it has safety features implemented to it where the user can initialize the serial communication port once and if the need to initialize a new port arises the user is forced to close the previous port communication and initialize the new one. This prevents data from being corrupted and for communication between the device and GUI to be proper.

Settings: The following form allows for only a Doctor/Nurse to change parameters in the pacemaker of the patient given the uniqueness of the treatment required. Furthermore, a security module form will be added to prevent the patient or other subjects to enter this form.

User Setup: Allows the user to do a required first and only time setup to initialize the pacemaker.

Reports: Allows for the user, as well as any other interested parties, to check reports and real time graphs of the Pacemaker. In order to produce such a diagram, the input signal is translated through the serial communication from the heart to the DCM, and updated into a data server used by the GUI to generate an approximate representation of the real time pace.

Pace Now: A child module that directly sends a signal to the Mbed to send a pulse for the patient.

6.5.1 GUI Serial communication:

Purpose:

The purpose of using the serial communication is to realize the data transmission between DCM and PG.

1. Read the heartbeat data, which is in analog form, into the DCM using the serial communication protocol.
2. Convert analog signal from the heart to digital form, therefore DCM could load the specific mode to the pulse generator based that signal.
3. Transmit the output signal from DCM to the pulse generator to pace the heart in certain mode.
4. Interpret the magnitude of the amplitude of the pulse.

Secret:

To allow data transfer between GUI and the K64F, both temporary and permanent text files are generated, written, and read from. Temporary files are written by the K64F with EKG data and passed to the GUI. The GUI then creates the EKG image, and stored the data within a log file for M.P. use. The GUI also writes semi-permanent text files with the current pace.

Global variables:

Serial PC (USBTX, USBRX);

Public functions with parameters:

read_USBPort (Port* name); which takes one of the serial port as the parameter, and then read the data from the port into the DCM.

write_USBPort (Port* Name); which takes one of the serial port as the parameter, and then write the data from the port into a local text file.