# SE 3K04 - Pacemaker

*Assignment 3 - Report*


**Group 18 : Names / Student Numbers**

**Liuyin Shi 1427387**
**Amar Bayat 1228292**
**Richard Zhang 1407090**
**Hanxi Huang 1303296**
**Peijie He 1328918**
**Zuge Gao 1204754**

Instructor : Alan Wassyng


Date: Dec 11th, 2016

# Table of Contents

**Part 1 - MBED Board**

**Functions and Behaviors** **2**

**Use Relationship** **3**

**Global Variables** **3**

**Likely Requirement Changes** **3**

**Future Decisions Changes** **3**

**Modes of the Micro-Controller** **4**

**Part 2 - DCM**

**Functions and Behaviors** **5**

**Global Variables** **6**

**Likely Requirement Changes** **6**

**Future Decisions Changes** **6**

# Part 1

For part 1 of assignment 3, we developed the working hardware for change modes, transmitting/save data based on the general module from the given code. The purpose of part one is to make a functional hardware that can pace making the VOOR or VOO mode and that can be controlled using the DCM developed in part 2 since the last assignment.

In this part we used the serial class from the mbed library, and the code provided on avenue to learn.

# Functions and Behaviors

**Public void PaceModule: transmit(int sel )**

This function can be used by every component of the hardware program. It sends the rate and amplitude from the pacing module in real time. Transmission of the data can also be switched ON or OFF by the user through the DCM developed in part 2. An integer value of 0 switches it OFF , and an integer value of 1 switches it ON, which is inputted by the user through the user interface.

**Void thePace(PaceModule a)**

In the ***thePace()*** we wrote a series of if-statements for pacing, transmission of egram data and control for states from the instructions sent from the DCM. When the user sends instructions the pace module would decide whether to pace, send data or not. Also depending on the user input, the pacemaker hardware would change pacing modes and decide when to display the data when users require it. It calls all the required functions for pace making and hide the internal details/info to the user.

The internal behavior of the function is that it first decides to updates the egram data and the data comes from the sensing thread. Then the function would detect the instructions from the DCM to turn the pacing module on/off with a if statement. The following if-else statement would decide whether to pace VOOR or VOO based on the modeID variable. At the end a if statement decision for deciding whether to display egram data is made. After that one complete pace cycle is completed. This function is designed for information hiding and ease of use when doing pacing in main function.

## Use Relationship

The relationship between the functions in part 1 is not tightly connected with each other.For the transmit() function of the pace module, it is only dependent on the Switch, pc.rate, pc.amp and the data from the motion sensing, which are public and available to every other functions. *thePace()* function uses the functions from the other classed as given from the avenue and the mbed library, like VOOR which is implemented inside the 'pulsegenerator.cpp.' For the variables, all the parameters-related variables are defined in the main.cpp and are public for future development.

## Global Variables

1. **Public Int Switch**: The switch for control the on/off of pacing module. When it is 0 the module would not send egram data and stop pacing. When it is 1 it will send data and start pacing.
2. **Public Float amp,rate**: The egram data packet(amplitude and the rate) that would be sent to the GUI
3. **Serial poi**: The serial port that handles the receiving data/ instructions and send real time data packet. The main functions is heavily based on serial class from mbed library.
4. **Extern float magtd:** the float comes from the motion sensing and would be sent to GUI for display
5. **Public Int Display:** the display switch for control of displaying the egram data. The value can be modified by the user to turn display on and off by sending '4' in the UI.
6. **Public Int modeID:** the mode variable for controlling the pacing mode of the pacemaker hardware. The value can be modified by the user through GUI.

## Likely Requirement Changes

1. The pacemaker can pace in different modes and customize the parameters of pacing.
2. The pacemaker can have customized modes of pacing dependent on the DCM instructions.
3. Smoother analog voltage output with signal processing.

## Future Decisions Changes

1. We may modify the current main function to make it general so it can be used by every mode of pacing.
2. The egram data can be more in details and the

## Modes of the Micro-Controller

| Current Mode of Switch | Input | Transmit Data | Pacing |
|---|---|---|---|
| 0 | 0 | No | No |
| 0 | 1 | Yes | Yes |
| 1 | 0 | No | No |
| 1 | 1 | Yes | Yes |

| Current modeID | Input | modeID next |
|---|---|---|
| 0 | 2 | 0 |
| 0 | 3 | 1 |
| 1 | 2 | 0 |
| 1 | 3 | 1 |

| Current Display | Input | Display Data Next |
|---|---|---|
| 0 | 4 | yes |
| 1 | 4 | no |

# Part 2

For the pacemaker, Assignment 3; we developed a working GUI that can transmit , receive and save data. It is able start transmission of data from the MBED board from sending an integer of “1” and stop the transmission by send an integer “0” from the interface. The DCM was built using Visual Studios. It was built with high cohesion and low coupling of modules. Each private function was interconnected as much as possible with each other.

# Functions and Behaviors

**(Connect Button)-private void send_btn_Click(object sender, EventArgs e)**

This method includes instructions about what happens when the “Connect” button is clicked on the user interface . When the “Connect” button is clicked, it also activates serial communication between the mbed board and the user interface. This methods takes two user inputs which is , baud rate and the port name.

**(Disconnect Button)-private void button3_Click(object sender, EventArgs e)**

This methods disables serial communication between the mbed board and the user interface. This is accomplished by closing the com port that the user inputted.

**(Save Button)-private void save_btn_Click(object sender, EventArgs e)**

This method saves to the desktop, the egram data that is being received from the board and being displayed in the user interface. Basically , in this private method , you give it a path to where you want to save the egram data that is being transmitted from the serial port to the user interface. The file saved on the desktop , or wherever the user chooses to save the data is named “egram data.txt”.

**(Send Data Button)-private void transmit_btn_Click(object sender, EventArgs e)**

This method transmits an integer to the MBED board. Firstly, it takes an input from the user (written in textbox), then it takes the value inputted and converts the string to an integer. Afterwards , it writes the specified integer ( anInteger, in the visual studio code) and it writes it to the serial port , if the serial port is open . If the serial port is not open , then the DCM would throw an exception. It writes the integer to the serial port by using the command , myport.WriteLine().

## Global Variables

1. **Private SerialPort myport** - Name of the serial port to be accessed (User input)
2. **Private DateTime datetime** - Current date and time to see at what time the data is transmitted
3. **Private string in_data** - Saves the data that is being transmitted from the mbed board in order to be displayed on the user interface

## Likely Requirement Changes

1. DCM can change the mode in which the pacemaker is in
2. Customizable settings, multiple languages, various modes to filter data
3. Flexible diagnostics, user can choose what data to see from the serial port connection

## Future Decisions Changes

1. Display the egram data visually through a plot/graph
2. Include an about section in the DCM with instruction on how to use the DCM
3. Being able to generate individual profiles for each user of the pacemaker, that includes the history of his performance.