Anurag Muglikar, Lindsay Langford, Nishanth Shyamkumar, Ritika Gupta

CIS 541

Final Project Milestone 2

For this milestone we implemented the heart model, pacemaker model, and monitor model in C on the mbed platform, then added IoT functionality using the MQTT protocol. This report will discuss each section in detail and provide instructions for assembling our system.

**System Assembly**

Our system requires two mbeds and one ESP8266. One mbed will act as the heart, and the other one will act as the pacemaker and monitor. The heart mbed and pacemaker mbed communicate over two wires, so connect pin 5 on the heart to pin 19 on the pacemaker, and then connect pin 5 on the pacemaker to pin 19 on the heart. The diagram below shows the connections between the mbeds and ESP8266 (WiFi module).

Pace Maker

5

19 3.3V

28

27

26

WiFi Module

Ch\_PD

Reset

Rx

Tx

Heart Model

5

19

**Heart**

The heart only has a main thread due to its simplicity. It follows the UPPAAL model we submitted except when the Milestone 2 documentation required a different value. The heart generates a random value between 100 and 1900ms and waits until that time interval is done to beat. However, if it receives a pace before its time interval is up it will use that to beat. When the heart beats naturally it toggles LED1 when it sends a signal to the pacemaker and LED 2 when it receives a signal from the pacemaker. After either a beat or a pace the heart waits for the VRP interval, which have defined as a constant 240ms like we did for Milestone 1.

**Pacemaker**

The pacemaker has its own thread. It waits and does nothing either until it receives a natural heartbeart or until the elapsed time has exceeded the pre-determined time value RI and it must send a pace. There is some hysteresis in the value of RI. If the previous heartbeat came from the heart, RI is equal to the value of HRI, which we have as 1500ms to match our model. If the previous heartbeat came from the pacemaker, then the value of RI is the LRI value of 1100ms, again from our UPPAAL model. The pacemaker lights up LED 2 every time it sends a pace to the heart and LED 1 when it receives a beat from the heart. After every beat or pace, the pacemaker waits for the VRP of 240 ms, during which time it cannot pace and does not pay attention to heartbeat signals. The pacemaker also raises flags that the monitor uses whenever there is a heartbeat or a pace.

**Heart Monitor**

The heart monitor has two separate threads, monitor\_observe and monitor\_calculate. The observation thread is our C implementation of the UPPAAL model in part 1. It waits until it receives a signal indicating either a natural heartbeat or a pace. Once it gets a signal it processes it and then immediately returns to waiting. The processing is very short in an attempt to make sure the observer does not miss any signals – the monitor records the time and then adds it to a queue used to calculate the average heartrate in another thread. If the monitor saw a pace instead of a natural heartbeat then it adds the time to a second, pace-only queue as well.

The monitor\_calculate thread is where the queues are processed. We have two queues – one which holds the times of all the paces sent by the pacemaker, and another which stores the times of both the paces and the natural heartbeats. The queue which is only for paces counts the number of paces which have occurred within the averaging interval and is used to determine if the alarm that the heart is too slow needs to sounds. Every averaging interval, the program gets the size of the queue with all of the beat times in it and uses it to determine the BPM of the heart. If the BPM is too high, meaning it exceeds the URL of 100 BPM, the alarm for the heart beating too fast will be activated. Both alarms are indicated by an LED blinking and sending an MQTT message to the cloud; the slow alarm is on LED 3 and the fast alarm is on LED 4. Because the averaging window can be less than 60 seconds, we scale the number of beats by 60 / the averaging window in order to get BPM regardless of the window length.

We store the time the beats and paces are registered into the queues instead of keeping a running count because we are required to have a sliding windowed average where the window is time instead of the number of samples. Every time the monitor needs to calculate the average, it first goes through the queues and removes any times which are too old as a way to slide the averaging window.

**IoT Functionality**

The cloud communication is done using the MQTT library.

It relays the alarm messages RATE\_ALARM (for high heart rate) and PACE\_ALARM (low heart rate). The observer communicates the alarms by publishing to the topic. The published message is sent to the MQTT Broker from where it is relayed to all other MQTT clients that have subscribed on this topic. In this way, the alarms are sent to the cloud application and front end from the pacemaker device. We used a public MQTT broker provided by HiveMQ, and our website can be accessed at [this site](http://propelis.in/pace-maker/Dashboard/dashboard.html)

