HIS SSNS - Bi-Weekly Report 2

Raul Bertone Elis Harruni Muyassar Kokhkharova
Saidar Ramazanov Xhoni Robo

May 31, 2018

Abstract

This report covers the progress made by the group on their project during the time period 19th of May to 31st of May 2018. Individual and total group effort is at the end of this report.

1 SensorTag Development

Function activation and deactivation: we were able to deactivate the unused functions in the proper way, by adding the relevant pre-processor options: EXCLUDE_OAD, EXCLUDE_REG, EXCLUDE_OPT, EXCLUDE_BAR, EXCLUDE_HUM, EXCLUDE_TMP. In the same way we also activated the buzzer, with the option Board_BUZZER.

Movement sensor configuration: first, using the appropriate bitmask in the *mpuconfig* variable, we were able to configure the movement sensor. In particular we set the sensitivity of the accelerometer to +/- 4G and we turned off the magnetometer.

mpuconfig = 0x1BF

Additionally, we increased the update rate of the whole sensor to 20Hz (instead of the standard 1Hz).

I/O service: to be able to control the buzzer (and LEDs) remotely via Bluetooth, we modified the I/O service configuration.

ioMode = 1

At the moment, this concludes the development on the SensorTags.

1.1 Dead SensorTags

We did some investigation on why the SensorTags often stopped working after flashing. We found out that, on some computers, flashing by connecting the Debugger module via USB 3.0 would kill the Sensortag (it would then only work while powered through the Debugger itself, but not on battery alone). To avoid this, it was enough to flash via a USB 2.0 port instead of 3.0. However, to bring back a killed Sensortag it is first necessary to flash the hex file using the Flash Programmer 2 (again, only via USB 2.0).

1.2 Git Lecture

Because of the different levels of programming experience in the team, it was deemed necessary to hold a short meeting where the basics of using *git* were introduced. This included everything from the basic ideas of how git works, how to effectively use branches for smart programming as well as the command lines that are required to work on different computers.

1.3 Desktop Application Development

During the past two weeks since the last report, the team has completed a primary version of the *Alarm Module* and of the *Configuration Module*. The first is able to send an email to one or more contacts to request assistance. The second stores all the information needed by the application. This information is of two kinds:

1. user data and contact information for the helpers
2. calibration and settings information for the sensors

2 Implementation of Math Model

On the previous week, the Mathematical logic of the accelerometer fall detection was implemented. The architecture of the implementation was easy and reliable for the first look. Program starts a thread when the function of the maths model is triggered by main package and it gets new measurements from sensors. However, it was found that the amount of measurements per second is too high. It means that there will be 25 threads each moment. Also, the minimum fall time is 450 ms, consequently the program can skip calculations for 10 measurements. Finally, the mathematical model will not care about calculations for 500 ms, during that time the Mathematical class will only add new measurements into the array of data. It is obvious that there will be only two threads of calculations in one moment. This brings less memory usage and increases of speed of the application.

Another point was made, that the thread should be started inside Mathematical class, instead of starting it outside. It brings more flexibility for the program, because the application will just trigger functions of this class, and is not worried with what happens inside this class.

3 Gyroscope

To use only the accelerometer is not enough to detect the fall. The application should include protection from making wrong decisions. That is why it was important to figure out the logic of additional part in Mathematical package.

First, the application should store data from sensors for the Gyroscope the same way as for the Accelerometer part in parallel. Secondly, the main mathematical part (accelerometer) should detect a fall. If it happens, the main part should trigger the method of the Gyroscope class and gives time stamps of the start and end of the fall. Then, the Gyroscope part is going to the previous measurements, when falling begins and the accelerometer has approximately -1 g on OZ axis. Between these two timestamps, the application started to measure the angle of the fall on two axis. If the composition angle of the two axis is equal to:

$$90 \pm 35 \pm 180 * n, where\ n \in Z,$$

then it means that the patient is laying and the Gyroscope class proofs the fall.

The main problem of it is to understand from what point of time the application should start calculations of the patient tilt.

4 Machine Learning

On one of the previous team meetings, it was stated that our team would implement Machine Learning only if we will have enough time at the end of the project. On the previous week, it was decided that the application should be more reliable for decisions. That is why it was figured out that the application should also calculate angle of the patient during the fall.

To minimize risks, a decision was made to check how Machine Learning can be implemented in our app. If ML is faster to implement it makes more sense to change the whole strategy of the Mathematical part. However, after research, it revealed that in order to implement ML, the team faces with several problems:

- State a model for ML;(It is easy to do)
- Convert the data from sensors to data, which can be understandable by ML (it will be the biggest problem, because data should be preprocessed. It means that all data that the application has should be converted to boolean data. Consequently, ML cannot decide if it's a fall or not depending on exact data from sensors)
- Implement learning techniques for ML.

Model for ML can contain the following boolean fields:

- isImpact absolute value of acceleration more than **2g**
- isLayng absolute value of acceleration after 400 ms on OX and OY axis is equal to **1g**
- isMoving absolute value of acceleration still more than **1,5 g**
- isRotating angular speed is more than normal

isImpact	isLayng	isMoving	isRotating	Fall
1	1	0	0	1
1	1	0	1	1
0	0	1	1	0
0	1	0	1	0

As the result, ML in fall detection can do only a small part of the work that is why it is not effective to implement it at the field of study.

5 Graphical User Interface

To implement GUI based on the mock up made earlier, JavaFX Scene Builder[1] was used. JavaFX Scene Builder is a tool that lets users design a JavaFX application's UI. UI components can be draged and dropped to a work area, modify their properties and at the end we will have FXML code for the created layout generated automatically. The result is a FXML file that can be combined with a Java project by binding the UI to the applications logic.

In the first version of GUI we have main window with area for Accelerometer and Gyroscope graphs and buttons: Connect, Disconnect, Start(receiving data), Stop(receiving data), Clean. In the menu bar we have User General Information, Settings and Close buttons. In User General Information the user should fill in the form with his data and the contact person‘s data. We are planning to modify the Settings window and we will discuss it on our weekly group meeting.

6 Effort Hours

- **Raul Bertone:** 16h
- **Elis Harruni:** 12h
- **Muyassar Kokhkharova:** 10h
- **Saidar Ramazanov:** 15h

- **Xhoni Robo:** 9h

Total Effort: 62h

References

[1] *JavaFX Scene Builder*. http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html. Accessed: 2018-05-30.