Raul Bertone
Elis Haruni
Muyassar Kokhkharova
Saidar Ramazanov
Xhoni Robo
Total Team Effort: 28h

Weekly Report 1

Weekly Plan (Xhoni Robo, 1h)

After the project proposal, the group members decided to meet twice during the following week. The meeting days are Tuesday and Friday. The meeting on Tuesday lasted **2 hours**, and the team members confirmed what they worked on during the weekend. Each member has already started on a specific task, following the already provided schedule with minor adjustments. On the meeting, the following topics were discussed:

- What each member was already working on
- General plan for the week
- Re-evaluation of the schedule, considering current progress
- Handing out of the weekly tasks for each member
- Getting started on the practical aspects of the application

The following is what each team members role for the next split, as decided on the meeting:

- **Raul Bertone and Xhoni Robo**: SensorTag implementation in C
- **Elis Haruni**: Bluetooth I/O
- **Saidar Ramazanov**: Fall Recognition Algorithm
- **Muyassar Kokhkharova**: Application GUI

During this week, until Thursday, 10.05.2018, the following was accomplished by each team member:

- **Raul Bertone**: Component Diagram, 7h
- **Elis Haruni**: Bluetooth I/O, 5h
- **Saidar Ramazanov**: Literature Research, Analysis and Sharing, 4h
- **Muyassar Kokhkharova**: Use Case Diagrams, 4h
- **Xhoni Robo**: Putting together the Weekly Report, 4h

Component Diagram (Raul Bertone, 7h)

Introduction

The diagram shows the main modules that make up our system: the Java application running onto the PC and the C application running on the two Sensortags.
The communication between the PC and the Sensortag happens over Bluetooth, with the former acting as central device and the latter as peripherals. The Sensortags don't communicate directly with each other.

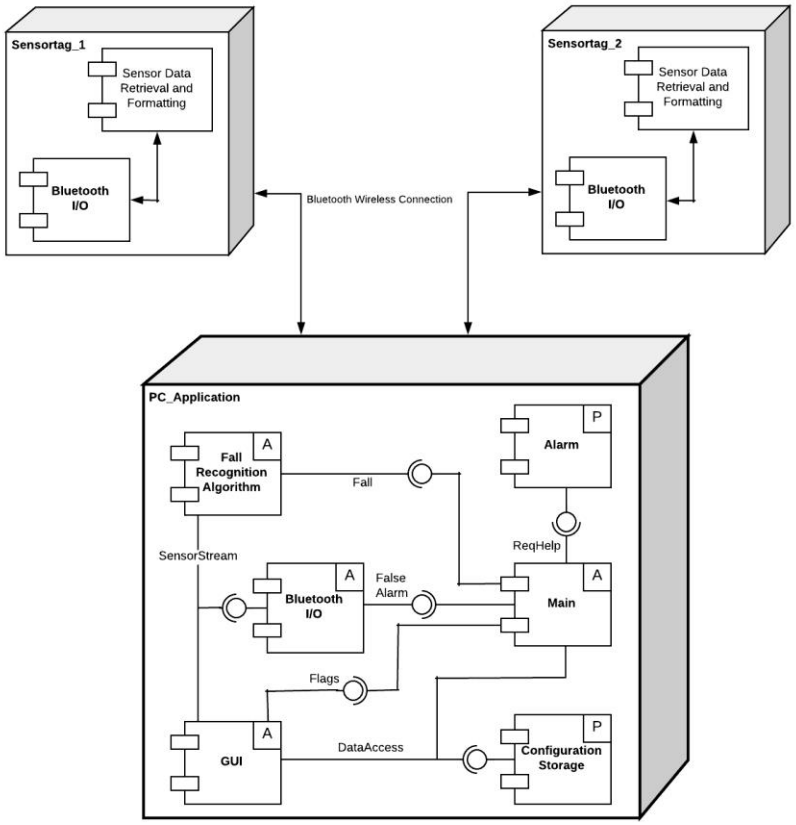


Figure 1: Component Diagram

Sensortags

There are two logical modules in this application:

- *Sensor Data Retrieval and Formatting*. It activates and configures the necessary sensors and actuators, receives the measured values from the sensors and formats them appropriately before passing in to the second module;
- *Bluetooth I/O.* Sets up the Bluetooth profile, services and characteristics; receives data from the previous module and forwards it to the PC_Application.

It's important to note that the modules described above don't necessarily represent different files or libraries, but only different logical functions of the application, which might very well be implemented in the same file.

PC_Application
The application is composed by six modules. The modules marked with an "A" are active modules, that is, they are executed within their own thread. "P" modules are instead passive, and their code is executed only when accessed by an active module.

- *Bluetooth I/O.* Manages the Bluetooth connection and receives the data from the Sensortags; sends an alarm when a Fall Event is detected;
- *Fall Recognition Algorithm.* Fetches sensor data from the Bluetooth Input module and examines it to look for Fall Events. When one is detected, it informs the GUI module and sends an alarm to the Bluetooth I/O module;
- *GUI.* Displays the graphical user interface; fetches sensor data from the Bluetooth I/O module to produce the graphs; retrieves and saves configuration data in the Configuration Storage module;
- *Main.* Contains the business logic of the application. When a Fall Event is detected by the Fall Recognition Algorithm module, raises the *Fall Detected* flag in the GUI and sends a false alarm request to the Sensortags; if the false alarm request is not answered, it reads the help data from the Configuration Storage, requests help through the Alarm module and raises *the Help Requested* flag in the GUI;
- *Alarm.* Sends an "alarm", simulated here by sending an email;
- *File Access.* Reads and writes the configuration files.

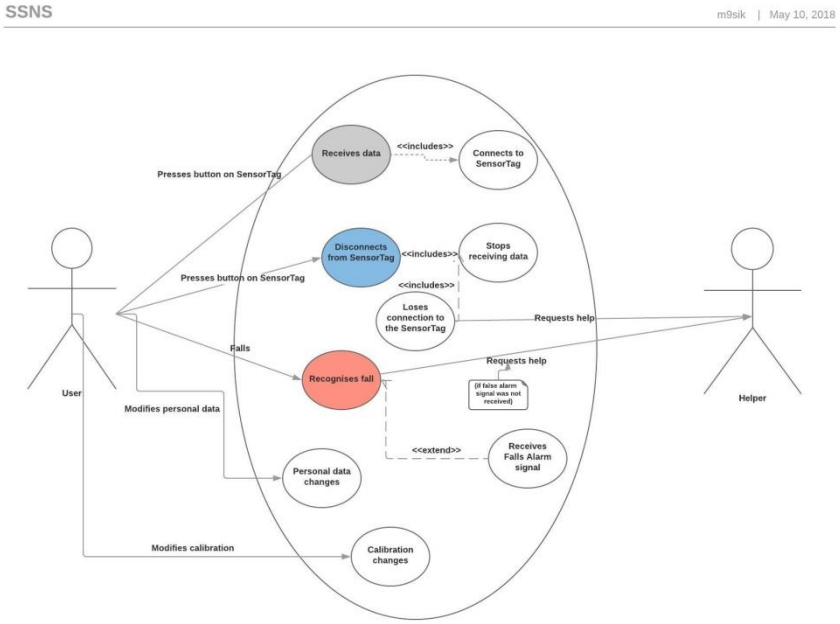
On one hand, the separation into modules will allow the team to subdivide the development tasks and work in parallel, while on the other, the definition of formal interfaces will simplify coordination between this development efforts.
Like noted for the Sensortag application, the modules represent only a logical subdivision of the features, and don't necessarily correspond to specific packages. However, is reasonable to expect that several of them will be implemented as independent packages.

Interfaces
FalseAlarm
Module: Bluetooth I/O
Input: control over the buzzer in the Sensortags
Output: events (presses) of button_2 on the Sensortags
DataAccess
Module: Configuration Storage
Input: write access to the configuration file(s)
Output: read access to the configuration file(s)
SensorStream
Module: Bluetooth I/O
Input: none
Output: one data stream for each sensor

Fall
Module: Main
Input: none
Output: fall event detected
ReqHelp
Module: Alarm
Input: help message (email body); message address (email address)
Output: none
Flags
Module: GUI
Input: “fall detected” status (Boolean); “help requested” status (Boolean)
Output: none

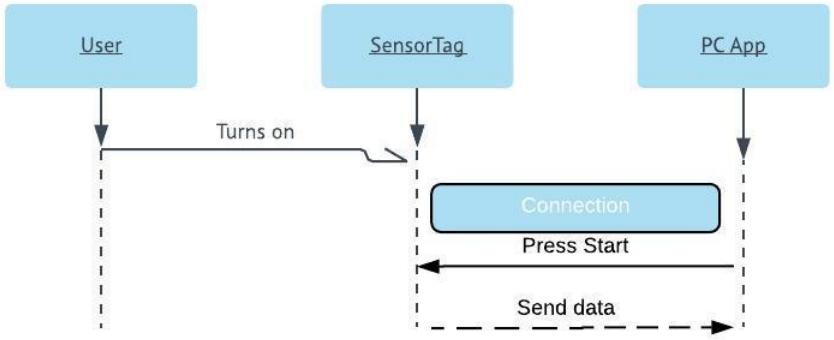
Use case diagram (Muyassar Kokhkharova, 4h)

Activate: User presses button on SensorTag to turn it on. PC app connects to SensorTag and starts receiving data
Deactivate: PC stops receiving data from SensorTag, disconnects from SensorTag and User presses button on SensorTag to turn it off
Out of Range: PC app stops receiving data, loses connection and sends help request to Helper
Fall: User falls, system recognizes fall and requests help
False Alarm: User “falls”, system recognizes fall and receives False Alarm signal
Modify personal data: User modifies his personal data and system saves changes
Modify calibration: User modifies calibration and system saves changes

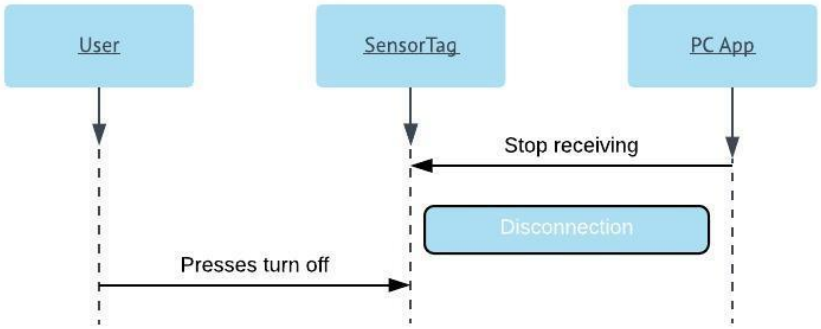


Sequence diagrams

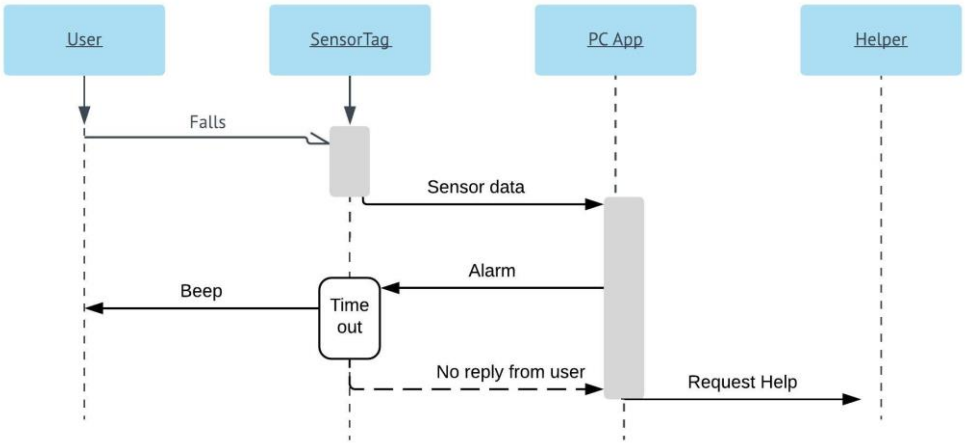
Activate:



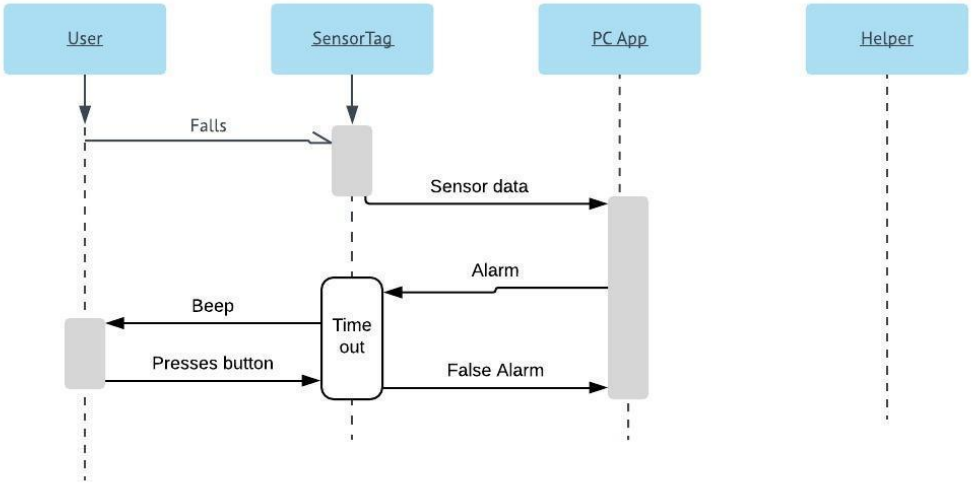
Deactivate:



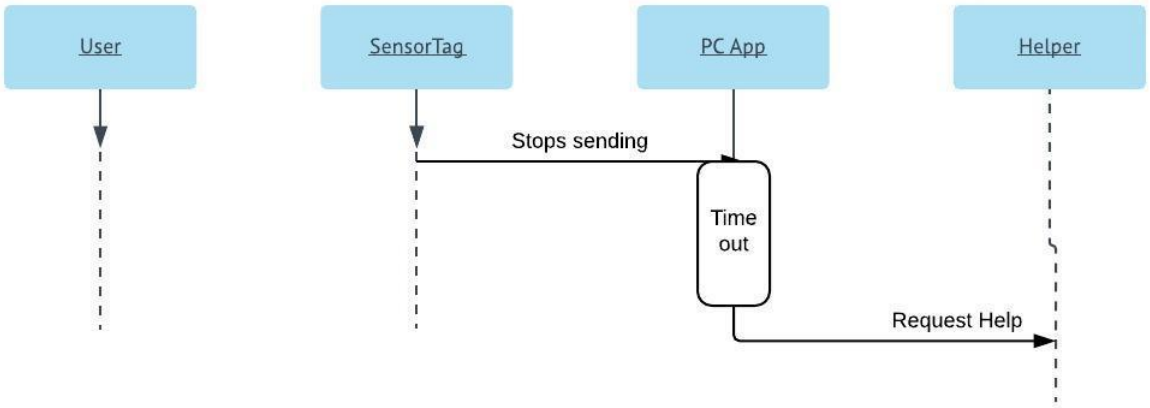
Fall:



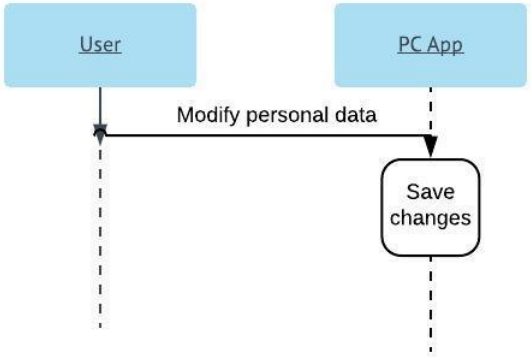
False Alarm:



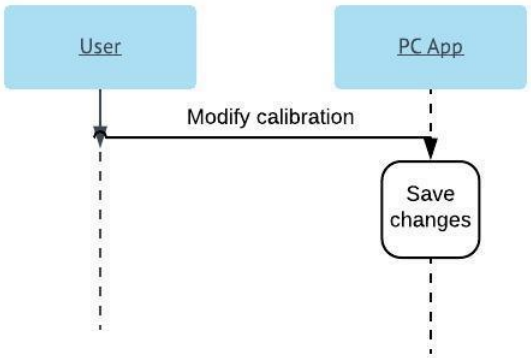
Out of range:



Modify personal data:



Modify calibration:



Conclusion

This is what was accomplished so far in the week, however there is still a group meeting on Friday before class time. So far, there have been no problems with the project. For the next two weeks, the goal is to have a completed application with only the most basic features. More will be discussed before class time.