HIS SSNS - Fall Detection based on Accelerometer and Gyroscope Data

Raul Bertone 1224898 Elis Harruni 1231496 Muyassar Kokhkharova 1248560 Saidar Ramazanov 1231658 Xhoni Robo 1248434

Abstract—**This is the final report for the group project for the High Integrity Systems M.Sc. Smart Sensor Network Systems for the Summer Semester 2018, lead by Prof. Dr. Matthias F. Wagner and his associates Luigi La Blunda, Olaf Reich and Kristiyan Balabanov. In this report we will present the design and implementation of an application for detecting falls based on accelerometer and gyroscope data[6].**

The set-up has two main parts: the first consists of two Sensortags, which have to be worn around the waist of the test subject, that will gather the sensor data and send it over a Bluetooth connection to the Base Station for elaboration; the second part is the Base Station, a Bluetooth equipped PC which will run the application that will elaborate the sensor data, try to identify falls, and if necessary request help.

The project span was 9 weeks, with the latest possible delivery date being the 5th of July.

This is a standalone project, with no interaction with other groups or organizations, and no dependencies on other projects by this or other teams.

As the project is intended to develop the understanding of smart sensor networks and the technical understanding of their development process, the software itself is not the sole product. Every relevant document produced during the development, including but not limited to, this document, weekly individual reports by the team members, and a final report and presentation, will be part of the delivered artifacts.

The following elements do not fall within the scope of this project and will not be included in the finished product:

- **considerations on the hardware design of the wearable part**
- **a user manual**
- **maintenance and support of the product after initial delivery**

I. Functional and Non-Functional Requirements

A. Functional Requirements

1) Two CC2650 SensorTags are acting as peripherals and sending periodically acceleration and gyroscope data to the PC.
2) A PC application should be able to connect simultaneously to multiple peripherals via BLE.
3) A PC application should be able to receive sensor data (accelerometer and gyroscope) in real time.
4) Data visualization. Line graphs with accelerometer and gyroscope data received from SensorTag.
5) Possibility to enter Users general information.
6) Possibility to calibrate system thus differentiate between sudden movements like walking the steps and free fall.
7) Main UI with basic control functions for operator working with a PC application.

B. Non-Functional Requirements

1) User general Information is a pop up window and it should contain:
   - First Name
   - Last Name
   - Date of Birth
   - Gender
   - Address
   - Mobile phone number of the User
   - Blood Type
   - Contact Person (In case of fall this Person will be contacted.)
   - Contact Email (Email of a Contact Person.)
   - Save button to save changes
2) Application Settings is a pop up window containing the following information:
   a) Values of Impact (in g)
   b) Measurements after Impact
   c) Laying Acceleration (in g)
   d) Skipped Measurements
   e) Fall Angle (in °)

f) Measurements per Second
g) Lower Laying Limit (in g)
h) Upper Laying Limit (in g)
i) Accelerometer Scale
j) Help Request Delay (in mS)

3) In Main UI:
   a) Graphs with Accelerometer and Gyroscope data.
   b) Accelerometer and Gyroscope data in the graphs will be updated at the same offset time.
   c) Buttons connect/disconnect(to establish the bluetooth connection).
   d) Button Start receiving gyroscope and accelerometer data.
   e) Button Scan.
   f) List of Discovered Devices.
   g) Label for Fall Detection.
   h) Label for ”help requested”
   i) ”False Alarm” Button
   j) Fall Button
   k) I/O Service Button

C. Safety, Security and Reliability Requirements

For an application whose sole purpose is the detection of a person falling, it is important to ensure that it, at the very least, does what is required of it. However, it becomes of critical importance when paired with the fact that the user may be in danger following this fall. If this were a project made to be used for actual cases by hospitals for example, failure to correctly assess when a person is in danger may even be fatal.

Safety in this project translates to the protection of the hardware as well as software mechanisms that allow the fall to be detected. Preventing physical damage of the sensors is pretty self explanatory, and can be done by simply changing the design so that the fall itself would not be enough to break the sensors. As far as software is concerned, we need to ensure that the final application not only receives the data and correctly uses it, but also ensure that there are no interferences by other devices. In the case of low connectivity, the application should immediately notify the user. Lastly, the application should also ensure that it picks only the data received from the sensor tags. That way, there will be no issues with interference. Should anything not work as intended, the user should be notified immediately.

Once safety and security is ensured, the final application needs to also be reliable. The most basic reliability requirement is to prevent the application of notifying us of events that are similar to a fall, but that provide no danger to the user. This includes physical activities such as walking, running and even jumping. Below is the full list of safety, security and reliability requirements:

1) Safety Requirements:
1) Software correctly notifies when a person has fallen
2) User can press the SensorTag Button to signify a False Alarm

2) Security Requirements:
1) Data is collected from the associated SensorTags
2) Other devices cannot send data to the application

3) Reliability Requirements:
1) Software differentiates between falls and other similar activities
2) Software does not crash during long sessions where a lot of data is streamed
3) SensorTags only send the necessary data. Other sensors should be disabled

II. Project Planning

A. Project Estimation

For the estimation of effort, the COCOMO II model was used[7], which was based on the value of Function Points[3]. For a complete overview see [*Estimation.pdf*]. Below we present a summary of our estimation process.

1) Function Points: The Function Points calculation process was conducted only until the Unadjusted Function Points values where obtained, because it is these values which are employed by the COCOMO II model. In identifying the Application Boundary, we considered the two Sensortag devices and the PC application not as being each a standalone system, but as two of three modules that make up the complete application. As a consequence, the internal communication between the modules does not constitute a transaction; also, the complete system results stand-alone, and does not therefore possess External Interface Files.

Table I
FUNCTION POINTS CALCULATION

	FP
External Input	29
External Output	24
External Inquiry	21
Internal Logical Files	7
Total	81

The results of the estimation of effort obtained with the COCOMO II model are as follows.

Table II
FUNCTION POINTS CALCULATION

Person-Months	7.4
Schedule Months	1.5
SLOC	4293

The result above reflect our latest estimation, based on updated requirements. For a comparison with the original estimation, please see [*HIS SSNS - Project Proposal - Fall Detection Using Gyroscope and Accelerometer.pdf*].

B. Group Organization

In consideration of the short time span of this project, and of the prototype nature of the final system, the team decided to employ agile development techniques, specifically Scrum. Sprints had a duration of one week. Two scrum meeting were held each week, on Tuesdays at 10:30 and Fridays at 15:00.
Official communication was organized through two channels:

- for short or urgent messages and general coordination, the Slack[19] ”SSNS” group chat;
- for communication with the project owner, the forum ”Group A” on Moodle[16], or per email

All common artifacts (source code, documentation, reference sources, etc.) are to be uploaded on the teams GitHub repository[8].

C. Responsibilities of the Team Members

All team members will assume several roles during the project. However, each person has been assigned a main role, making him or her the coordinator of all the individual efforts for a specific subject.

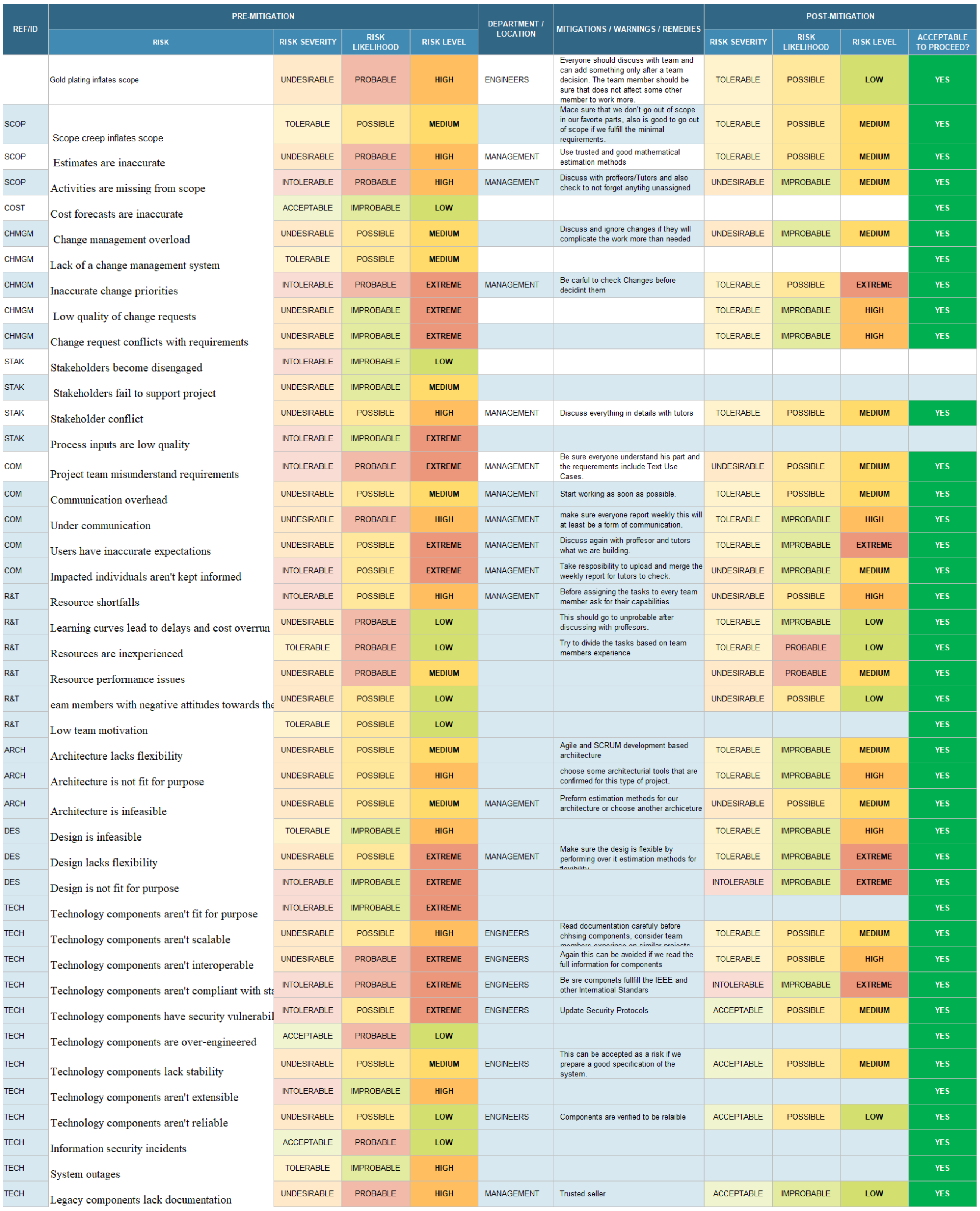
Table III
ROLES OF THE TEAM MEMBERS

Name	**Main Role**
Raul Bertone	Project Manager, Scrum Master
Elis Harruni	Lead Java Developer
Muyassar Kokhkharova	Statistics, UI Designer
Saidar Ramazanov	Mathematical Model
Xhoni Robo	Lead C Developer

D. Schedule

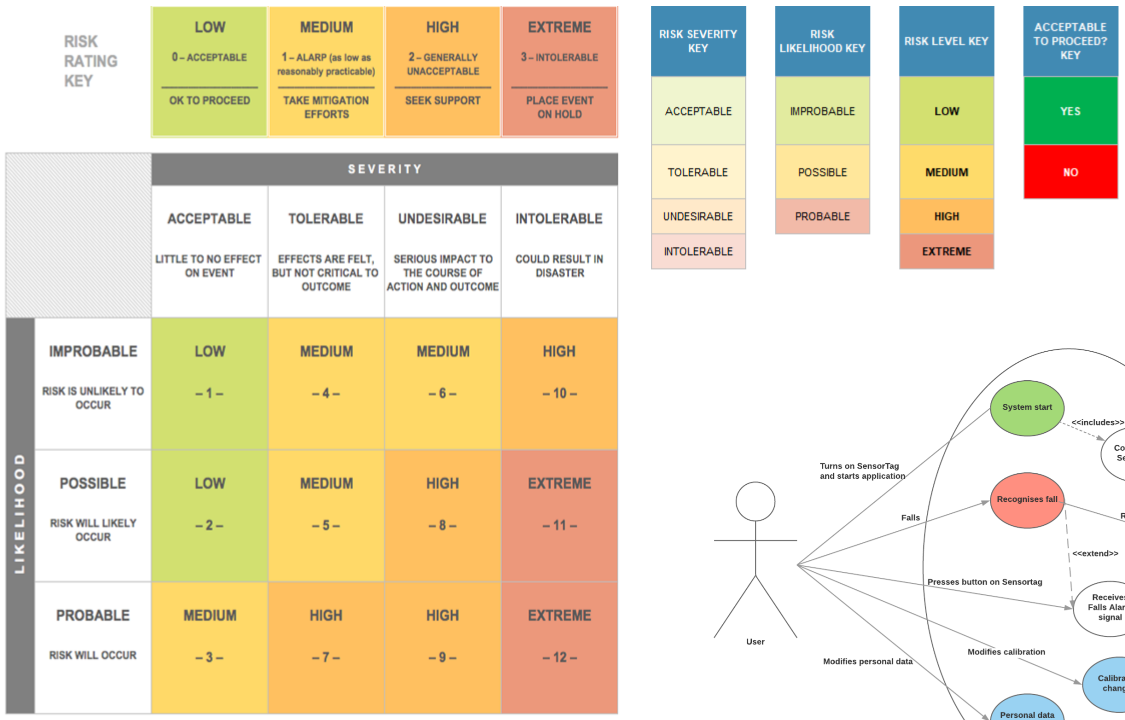
The original project schedule spanned 7 weeks, the latest delivery date being June 22th. However, during the course of the project, the project owner postponed it to July 5th, adding 2 weeks to it. Most of the initial work happened on schedule, up to the end of week 3. Then, the group started to face what turned out to be the hardest of the projects challenges, namely connecting more than one Sensortag at the same time to a Bluetooth central device. From that moment on, the project started running late while many independent attempts were being made at solving the problem. In week 7 a solution was finally found, and a scramble begun to complete the remaining tasks on time, tasks which fortunately didnt present any other unexpected obstacle.

E. Project Risk Analysis



REF/ID	PRE-MITIGATION				DEPARTMENT / LOCATION	MITIGATIONS / WARNINGS / REMEDIES	POST-MITIGATION			
	RISK	RISK SEVERITY	RISK LIKELIHOOD	RISK LEVEL			RISK SEVERITY	RISK LIKELIHOOD	RISK LEVEL	ACCEPTABLE TO PROCEED?
	Gold plating inflates scope	UNDESIRABLE	PROBABLE	HIGH	ENGINEERS	Everyone should discuss with team and can add something only after a team decision. The team member should be sure that does not affect some other member to work more.	TOLERABLE	POSSIBLE	LOW	YES
SCOP	Scope creep inflates scope	TOLERABLE	POSSIBLE	MEDIUM		Mace sure that we don't go out of scope in our favorte parts, also is good to go out of scope if we fulfill the minimal requirements.	TOLERABLE	POSSIBLE	MEDIUM	YES
SCOP	Estimates are inaccurate	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	Use trusted and good mathematical estimation methods	TOLERABLE	POSSIBLE	MEDIUM	YES
SCOP	Activities are missing from scope	INTOLERABLE	PROBABLE	HIGH	MANAGEMENT	Discuss with proffeors/Tutors and also check to not forget anytihg unassigned	UNDESIRABLE	IMPROBABLE	MEDIUM	YES
COST	Cost forecasts are inaccurate	ACCEPTABLE	IMPROBABLE	LOW						YES
CHMGM	Change management overload	UNDESIRABLE	POSSIBLE	MEDIUM		Discuss and ignore changes if they will complicate the work more than needed	UNDESIRABLE	IMPROBABLE	MEDIUM	YES
CHMGM	Lack of a change management system	TOLERABLE	POSSIBLE	MEDIUM						YES
CHMGM	Inaccurate change priorities	INTOLERABLE	PROBABLE	EXTREME	MANAGEMENT	Be carful to check Changes before decidint them	TOLERABLE	POSSIBLE	EXTREME	YES
CHMGM	Low quality of change requests	UNDESIRABLE	IMPROBABLE	EXTREME			TOLERABLE	IMPROBABLE	HIGH	YES
CHMGM	Change request conflicts with requirements	UNDESIRABLE	IMPROBABLE	EXTREME			TOLERABLE	IMPROBABLE	HIGH	YES
STAK	Stakeholders become disengaged	INTOLERABLE	IMPROBABLE	LOW						
STAK	Stakeholders fail to support project	UNDESIRABLE	IMPROBABLE	MEDIUM						
STAK	Stakeholder conflict	UNDESIRABLE	POSSIBLE	HIGH	MANAGEMENT	Discuss everything in details with tutors	TOLERABLE	POSSIBLE	MEDIUM	YES
STAK	Process inputs are low quality	INTOLERABLE	IMPROBABLE	EXTREME						
COM	Project team misunderstand requirements	INTOLERABLE	PROBABLE	EXTREME	MANAGEMENT	Be sure everyone understand his part and the requerements include Text Use Cases.	UNDESIRABLE	POSSIBLE	MEDIUM	YES
COM	Communication overhead	UNDESIRABLE	POSSIBLE	MEDIUM	MANAGEMENT	Start working as soon as possible.	TOLERABLE	POSSIBLE	MEDIUM	YES
COM	Under communication	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	make sure everyone report weekly this will at least be a form of communication.	TOLERABLE	IMPROBABLE	HIGH	YES
COM	Users have inaccurate expectations	UNDESIRABLE	POSSIBLE	EXTREME	MANAGEMENT	Discuss again with proffesor and tutors what we are building.	TOLERABLE	IMPROBABLE	EXTREME	YES
COM	Impacted individuals aren't kept informed	INTOLERABLE	POSSIBLE	EXTREME	MANAGEMENT	Take resposibility to upload and merge the weekly report for tutors to check.	UNDESIRABLE	IMPROBABLE	MEDIUM	YES
R&T	Resource shortfalls	INTOLERABLE	POSSIBLE	HIGH	MANAGEMENT	Before assigning the tasks to every team member ask for their capabilities	UNDESIRABLE	POSSIBLE	HIGH	YES
R&T	Learning curves lead to delays and cost overrun	UNDESIRABLE	PROBABLE	LOW		This should go to unprobable after discussing with proffesors.	TOLERABLE	IMPROBABLE	LOW	YES
R&T	Resources are inexperienced	TOLERABLE	PROBABLE	LOW		Try to divide the tasks based on team members experience	TOLERABLE	PROBABLE	LOW	YES
R&T	Resource performance issues	UNDESIRABLE	PROBABLE	MEDIUM			UNDESIRABLE	PROBABLE	MEDIUM	YES
R&T	eam members with negative attitudes towards the	UNDESIRABLE	POSSIBLE	LOW			UNDESIRABLE	POSSIBLE	LOW	YES
R&T	Low team motivation	TOLERABLE	POSSIBLE	LOW						YES
ARCH	Architecture lacks flexibility	UNDESIRABLE	POSSIBLE	MEDIUM		Agile and SCRUM development based archiitecture	TOLERABLE	IMPROBABLE	MEDIUM	YES
ARCH	Architecture is not fit for purpose	UNDESIRABLE	POSSIBLE	HIGH		choose some architecturial tools that are confirmed for this type of project.	TOLERABLE	IMPROBABLE	HIGH	YES
ARCH	Architecture is infeasible	UNDESIRABLE	POSSIBLE	MEDIUM	MANAGEMENT	Preform estimation methods for our architecture or choose another archiceture	UNDESIRABLE	POSSIBLE	MEDIUM	YES
DES	Design is infeasible	TOLERABLE	IMPROBABLE	HIGH			TOLERABLE	IMPROBABLE	HIGH	YES
DES	Design lacks flexibility	UNDESIRABLE	POSSIBLE	EXTREME	MANAGEMENT	Make sure the desig is flexible by performing over it estimation methods for	TOLERABLE	IMPROBABLE	EXTREME	YES
DES	Design is not fit for purpose	INTOLERABLE	IMPROBABLE	EXTREME			INTOLERABLE	IMPROBABLE	EXTREME	YES
TECH	Technology components aren't fit for purpose	INTOLERABLE	IMPROBABLE	EXTREME						YES
TECH	Technology components aren't scalable	UNDESIRABLE	POSSIBLE	HIGH	ENGINEERS	Read documentation carefuly before chhsing components, consider team	TOLERABLE	POSSIBLE	MEDIUM	YES
TECH	Technology components aren't interoperable	UNDESIRABLE	PROBABLE	EXTREME	ENGINEERS	Again this can be avoided if we read the full information for components	TOLERABLE	POSSIBLE	HIGH	YES
TECH	Technology components aren't compliant with sta	INTOLERABLE	PROBABLE	EXTREME	ENGINEERS	Be sre componets fullfill the IEEE and other Internatioal Standars	INTOLERABLE	IMPROBABLE	EXTREME	YES
TECH	Technology components have security vulnerabil	INTOLERABLE	POSSIBLE	EXTREME	ENGINEERS	Update Security Protocols	ACCEPTABLE	POSSIBLE	MEDIUM	YES
TECH	Technology components are over-engineered	ACCEPTABLE	PROBABLE	LOW						YES
TECH	Technology components lack stability	UNDESIRABLE	POSSIBLE	MEDIUM	ENGINEERS	This can be accepted as a risk if we prepare a good specification of the system.	ACCEPTABLE	POSSIBLE	MEDIUM	YES
TECH	Technology components aren't extensible	INTOLERABLE	IMPROBABLE	HIGH						YES
TECH	Technology components aren't reliable	UNDESIRABLE	POSSIBLE	LOW	ENGINEERS	Components are verified to be relaible	ACCEPTABLE	POSSIBLE	LOW	YES
TECH	Information security incidents	ACCEPTABLE	PROBABLE	LOW						YES
TECH	System outages	TOLERABLE	IMPROBABLE	HIGH						YES
TECH	Legacy components lack documentation	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	Trusted seller	ACCEPTABLE	IMPROBABLE	LOW	YES

Figure 1. Risk Management Matrix



RISK RATING KEY	LOW	MEDIUM	HIGH	EXTREME
	0 – ACCEPTABLE	1 – ALARP (as low as reasonably practicable)	2 – GENERALLY UNACCEPTABLE	3 – INTOLERABLE
	OK TO PROCEED	TAKE MITIGATION EFFORTS	SEEK SUPPORT	PLACE EVENT ON HOLD

		SEVERITY			
		ACCEPTABLE LITTLE TO NO EFFECT ON EVENT	TOLERABLE EFFECTS ARE FELT, BUT NOT CRITICAL TO OUTCOME	UNDESIRABLE SERIOUS IMPACT TO THE COURSE OF ACTION AND OUTCOME	INTOLERABLE COULD RESULT IN DISASTER
LIKELIHOOD	IMPROBABLE RISK IS UNLIKELY TO OCCUR	LOW – 1 –	MEDIUM – 4 –	MEDIUM – 6 –	HIGH – 10 –
	POSSIBLE RISK WILL LIKELY OCCUR	LOW – 2 –	MEDIUM – 5 –	HIGH – 8 –	EXTREME – 11 –
	PROBABLE RISK WILL OCCUR	MEDIUM – 3 –	HIGH – 7 –	HIGH – 9 –	EXTREME – 12 –

RISK SEVERITY KEY	RISK LIKELIHOOD KEY	RISK LEVEL KEY	ACCEPTABLE TO PROCEED? KEY
ACCEPTABLE	IMPROBABLE	LOW	YES
TOLERABLE	POSSIBLE	MEDIUM	NO
UNDESIRABLE	PROBABLE	HIGH	
INTOLERABLE		EXTREME	

Figure 2. Risk Management Matrix Key

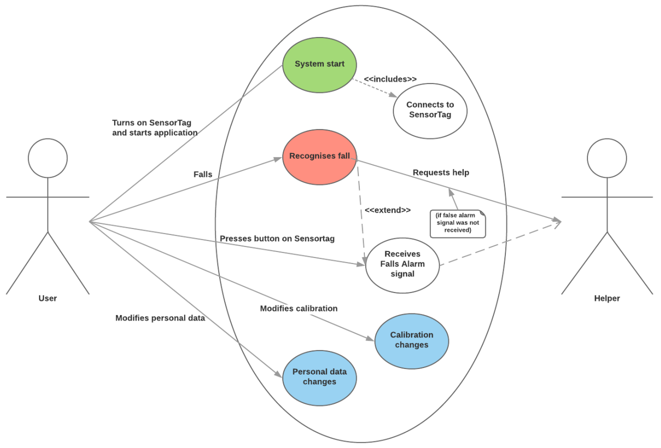


Figure 3. Use Case Diagram[9]

III. Design

A. Use Case Diagram

System Start: User presses button on SensorTag to turn it on. PC app connects to SensorTag and starts receiving data
Recognises Fall: User falls, system recognizes fall and requests help
Receiving False Alarm: User falls, system recognizes fall and receives False Alarm signal
Personal data changes: User modifies his personal data and system saves changes
Calibration changes: User modifies calibration and system saves changes

B. Sequence Diagrams taken from the Use Cases

To better clarify the Use Case diagram, we included below the following sequence diagrams concerning the most complicated parts.

Figure 4. System Start

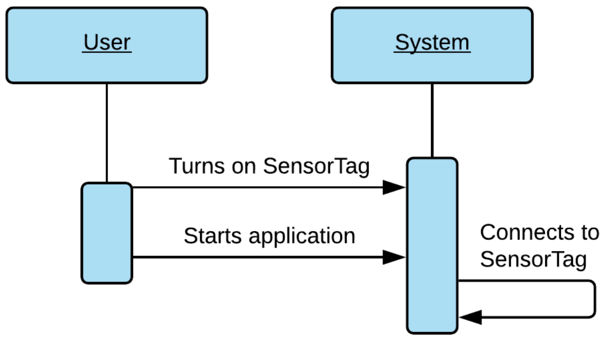
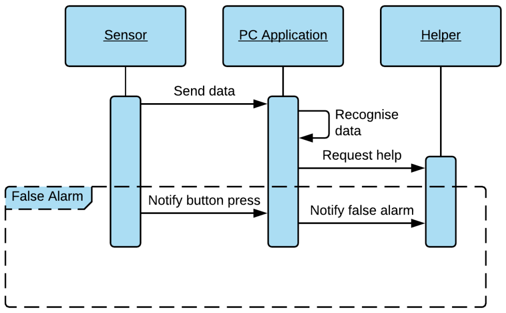


Figure 5. Recognize Fall



C. Architecture

The system consists of two TI Sensortags connected via Bluetooth LE to a TI Launchpad, which is in turn connected via USB to a PC. Initially, in an effort to simplify the development, we intended to connect the Sensortags directly to a BLE enabled PC. However, that turned out to be more difficult than expected and it was therefore decided to use a Launchpad to interface the two.

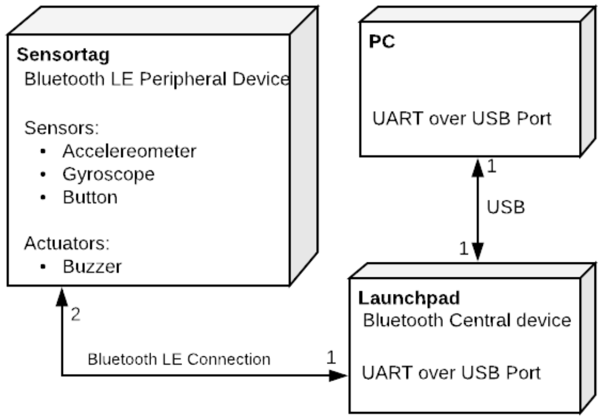
Each Sensortag runs a slightly modified version of the original TI application ”*sensortag_cc2650stk_app*”: they gather data through their built-in sensors and send it over BLE to the Launchpad. In this BLE connection, the Sensortags act as peripheral devices, while the Launchpad as the central device.

The Launchpad runs an unmodified version of the TI firmware[5]. It receives the sensor data and makes it available to the PC application through a USB COM port.

The PC application analyses the sensor data stream to detect fall events. Then, if necessary, it notifies the helpers that the user has fallen. The PC application possesses a GUI where the sensor data is used to display graphs. Additionally, the GUI can be used to calibrate the fall detection algorithm, and to change the stored user and helper personal data.

The diagram below shows the main system component:

Figure 6. Component Diagram



D. Software Architecture

Our application has six main classes, which will be explained further below.

The Server class represents the open COM port operations. This class not only opens the port, in order to allow the application to perform operations, it also handles the reads and writes to the port. This is a Singleton class, so we do not create a new instance by mistake and try to open the port again, as that will result in an exception.

The Mathems class is the Fall Detection Algorithm. It is also a Singleton class, because it is a different thread that tries to access resources on the Server. If we create two instances, we risk running into deadlocks.

Server class and ProcessMessage read and translate the messages for our application. FallNotification-Service is the class that will send the email and notify the helper for a fall or a false alarm.

How these classes communicate and work together is described by these sequence diagrams.

Figure 7. UML Diagram

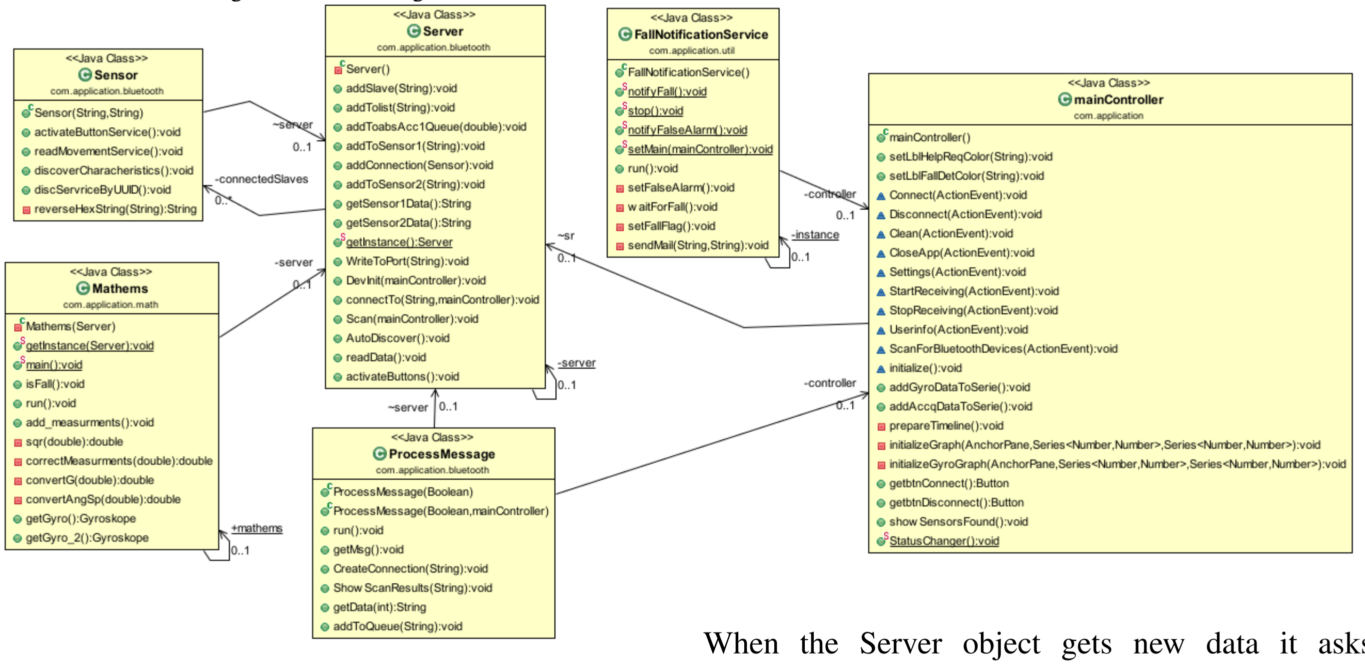
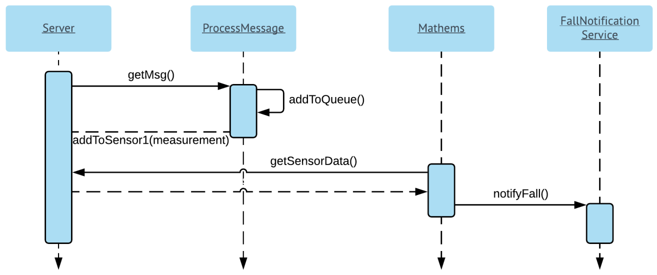
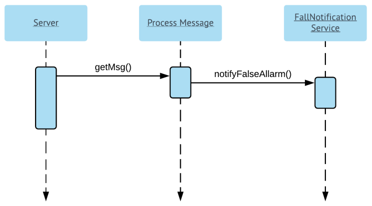


Figure 8. Fall Detection Sequence Diagram



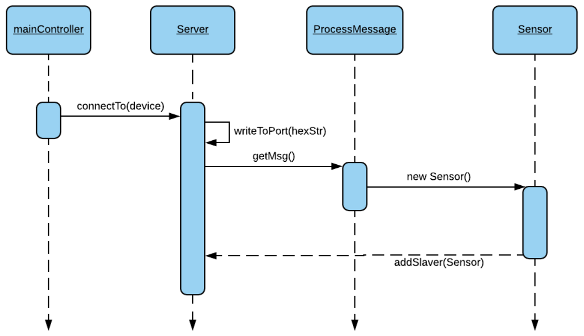
Server will tell the ProceessMessage that we have a new data, ProcessMessage will check if it measurement and if yes will save them to the Server. Mathems will analyze the measurements and if it is a Fall will tell FallNotificationService to send for help.

Figure 9. False Alarm Sequence Diagram



When the Server object gets new data it asks the ProcessMessage to find out what message that was. If it was a message from I/O service it means that the button on the SensorTag was pressed and will notify that the previous alarm was a false one.

Figure 10. Connecting to Sensor



Here we see also the main controller class that works like a bridge to send commands from the GUI to the backend. From GUI we ask the Server to connect to a device. The Server object will write to the COM port, which is the bridge between our application and Launchpad. After receiving a response from the Launchpad, the Server will ask the ProcessMessage to translate the message received, and if it is the comfirmation message for a new slave connected to the master, it will create a new object of type Sensor and will add the reference to the Server.

E. Technologies, Libraries, IDEs, Tools

Technologies used for this appliation are Java as a programming language for backend development, JavaFX for GUI Development. Gradle as a build tool, java-simple-serial-connector[14] library for reading a serial port in java. The ObjectAid UML Explorer for Eclipse[18], Eclipse IDE. C programming Language for Sensor Tags. LaTeX for Documentation.

IV. Measurement Chain[13]

The application makes use only of the MPU-9250 sensor pack present on the Sensortags, which includes a three-axis accelerometer, a three-axis gyroscope and a three-axis magnetometer (the last is not used and was deactivated).
The *measuring object* is the sensor itself (the acceleration and orientation of it). Since it is rigidly mounted on the PCB, we can assume that corresponds to measuring the movement of the whole Sensortag. However, we cannot ignore the soft connection of the Sensortag to the belt and of the belt to the user. This requires adaptation of the fall detection algorithms to many different user/clothing combinations (which is achieved by means of the calibration constants accessible through the GUI).
The *measured quantities* are: the total resulting acceleration (gravity plus motion) on each axis; the angular velocity on each axis.
The *measurement method* is indirect (see individual sensor description below).

The complete *measurement chain* consists of the sensors themselves, the ADCs built into the MPU-9250, and the number conversion that takes place in the desktop part of the application. Since the signals are transmitted only after the analog to digital conversion takes place, we dont have to worry about signal degradation due to analog transmission.

A. Accelerometer

”The accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially.” The analog value is digitalized by three 16bit sigma-delta ADC (one per axis). The resulting value may be written as:

$$a = val \pm e_i \pm e_T \pm e_l \pm e_{cal} \pm e_n \quad g$$

Where *val* is the real value, e_i is the intrinsic error (3%), e_T is the temperature induced error (0.026%/°C), e_l is the non-linearity error (0.5%), e_{cal} is the initial calibration error (80 mG) and e_n is the noise (8 mG). At room temperature and for the threshold values used by our fall detection algorithm, the above formula can be approximated with:

$$a = val \pm 6\% \quad g$$

B. Gyroscope

The gyroscopes sensors consist of 3 vibratory MEMS. ”When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff.” The resulting value may be written as:

$$v_{ang} = val \pm e_i \pm e_T \pm e_l \pm e_{cal} \pm e_n \quad {}^\circ/s$$

Where val is the real value, e_i is the intrinsic error (3%), e_T is the temperature induced error (4%), e_l is the non-linearity error (0.1%), e_{cal} is the initial calibration error (5 $^\circ/s$) and e_n is the noise (0.1 /s). At room temperature and for the threshold values used by our fall detection algorithm, the above formula can be approximated with:

$$v_{ang} = val \pm 9\% \quad {}^\circ/s$$

V. Fall Detection Algorithm

The Fall Detection Algorithm is the back end logic of the application, where the application handles decisions about a fall. The logic is divided into several parts, each of which is responsible for its own goals and tasks. It contains three classes:

- Mathems class - contains main thread of computations, creating stamp of Gyro and Accelerometer objects inside the thread, and contains main static variables like impact power, laying acceleration;
- Gyroscope class - contains measurements from sensor, functions of adding new measurements, thread of Gyro computations;

- Accelerometer class - contains measurements from sensor, functions for adding new measurements, can trigger Gyro method to start its thread.

It is better to start discovering parts of data from end to beginning. The ”Gyroscope class” stores queues for every axis of its measurements, the logic of the calculation degrees, which were made during the fall, the functionality of adding new measurements into the queue and a link to the ”Mathems class”. The ”Accelerometer class” contains queues for every axis of its measurements, the logic of calculation impact, meaning the impact which is made at contact with the floor, the functionality of adding new measurements into the queue, finding the beginning of the fall, checking if the patient is still laying on the floor and a link to the ”Mathems class”. The main part, which is called ”Mathems class”, contains the main thread of mathematical computations, with two pairs of objects: Gyro and Accelerometer for each SensorTag, the object Server, and several flags to inform about the fall. Also, it has the functionality of adding, parsing and converting measurements which come from the SensorTags.

A. Configuration Storage

The class of Configuration Storage was made to store all the static variables, which are used through all the application. It is an abstract and static class, which is why the application can have access to the variables from everywhere without having an instance of its object. The main aim of this class is to store main static values in one place. Furthermore, all static variables in the class have get()/set() methods and are connected to the UI, which gives it the advantage of changing them without restarting the application. For example, if an expert user wants to change the value of the impact, they can change it in ”Settings” window.

B. Main Thread and Work Logic

The ”Mathems class” was implemented as a class, which extends the Thread class, and it uses the Singleton pattern to have only one instance of the object. It was done this way because the object should be created only when the SensorTag sends values. In the constructor of the class, the application starts the main thread of the Mathematical model. The main tasks of the mathematical model execute only in this thread.

First of all, the main thread asks the Server for new measurements from SensorTags. If there is no new values then the thread will wait for $10ms$. Then, the thread has a fork: either add new measurements if the queue is less than the $ConfigurationStorage.getSKIP\_MEASURE()$ static variable, or make computations of the fall if there is enough measurements.

On the first few weeks of the project, it was calculated by the kinematic equation:

$$\triangle\ x = v_0 t + \frac{1}{2} a t^2$$

minimum amount time which is need for the fall. The sensor falls from *1m* high. This means that the minimum fall time will be *450ms*. In addition, Raul Bertone increased the speed of the output measurements from the SensorTag; now we get 20 measurements per second, which means that the application gets measurements every *50ms*. Consequently, for calculations it should store at least 10 measurements per one timeframe. Unfortunately, after several tests it was stated that the application should store more values because the amount is too small to distinguish the fall. Now there are 20 values for calculations and 10 from the past timeframe.

The function $add\_measurments$ asks the Server for new measurements, then converts them for G and Degree velocity scale. Finally, it adds them into the graph and the corresponding queues for the Gyroscope and the Acclerometer[17].

Furthermore, the class has two flags for the fall: $isAclrFall\&\&isGyroFall$. The Gyroscope and the Accelerometer classes change their values. They are checked by the thread every pass. If they are both ”*true*” the thread calls $FallNotificationService.notifyFall()$. It creates a notification of the fall.

C. Accelerometer

Every pass of the tread triggers the method $isArclFall$ to start calculations of the Accelerometer. The method makes a copy of objects every time it is triggered and starts in a loop to calculate the absolute value of the impact of all the axes together for every measurement.

$$impact = \sqrt{Gx^2 + Gy^2 + Gz^2} > 2, \\ where \quad \text{Gx, Gy, Gz} \quad are \quad axis \\ values;$$

If the impact is more than $2g$ it is going to check if the patient is laying to be sure that it was a fall. The Laying function skips several measurements after impact to be sure that the patient is in a static position. If the absolute value on OX and OY axis is between 0.8 and 1.3 it means that patient is laying.

$$0.8 < \sqrt{Gx^2 + Gy^2} < 1.3$$

If the patient is laying, the application changes the first flag that a fall was detected from the Accelerometer. Then the class calls the method of the Gyroscope class to calculate the change in degrees that was registered by patient during the fall. To do this, the Accelerometer class should calculate when the fall starts. This is calculated by a function $getFallStart$. The function takes the index of the impact values, subtracts five measurements back and starts one by one to check when the last absolute value on the OZ axis between 0.8 and 1.3 was. This way the application can understand the last index of the measurements when the patient was standing.

$$0.8 < |Gz| < 1.3$$

D. Gyroscope

To use only the Accelerometer is not enough to prove that it is a fall. The application should include protection from making wrong decisions. That is why it was important to implement the additional logic of the Mathematical package.

The application should store data from the sensors for the Gyroscope the same way as for the Accelerometer class in parallel. Then, the Gyroscope starts to calculate the degrees on OX and OY axes, which were registered by the patient between the beginning of the fall and the impact indexes.

The class has only the degrees' velocity (degrees per second), which is why the application should convert them into actual degrees. The conversion for every measurement can be made by dividing the speed with the amount of values per second. Then the application should sum up all the values from the beginning of the fall till the impact.

$$angle_i = \left|\frac{Gx_i}{CS}\right| + \left|\frac{Gy_i}{CS}\right|, \\ where\, Gx,\ Gy\, is\, the\, velocity\, on\, OX\, and\, OY\, axis; \\ CS\, is\, the\, amount\, of\, measurements\, per\, second;$$

Then this value should be divided by 180 and the application should take the remainder of the division and subtract from it 90 degrees. If this value is less than 30 degrees the application changes another flag for "true".

$$\textstyle\sum_i angle_i - 90 \pm 180 * n < 30, \\ where \quad n \in Z;$$

E. Problems

The main thread work continuously that is why during computations the application could link on the values in the queue, which have been changed. It can bring errors in logic of work. That is why the application make copies of objects of the Accelerator and Gyroscope classes.

Another one problem was connected to the creating only one instance of the "Mathems class". It was stated that the main thread should be created only when the first values come from the SensorTags. That is why instance of the class is implemented by Singleton pattern and the application creates it in the Server class.

One of the main problems was with detecting a fall when the patient falls on the first measurements of the queue. Because the application cannot go back in measurements to detect when the fall was started. That is why the Accelerator and Gyroscope objects store 30 measurements. However, first 10 measurements do not use in impact calculations. They are used in calculation of finding last horizontal position of the patient.

Finally, the application skips first five measurements when it is trying to find last horizontal position. It is made because after fall patient can have some noise. That noise can be detected as last horizontal position but it is not true.

VI. Development

A. Sensortags

As a base for development, we made use of the "*sensortag_cc2650stk_app*" TI project, to which we made the modifications described below.

Feature activation and deactivation: since our project only makes use of the accelerometer and

gyroscope, we choose to turn off all the other sensors. To do this, we used the TI preferred method, i.e. the ”*EXCLUDE*” pre-processor directives. For a complete list of the employed directives, please see [*HIS SSNS - Group 1 - Bi-Weekly Report 1*]. In the same way we also activated the buzzer.
Movement sensor configuration: using the appropriate bitmask in the *mpuconfig* variable, we were able to configure the set the sensitivity of the accelerometer to +/- 4g and turn off the magnetometer.
Update rate: 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 remotely via Bluetooth, it was necessary to modify the I/O service configuration, setting it to its ”remote” mode.
1) Problems with Sensortags: Sometimes our Sensortags would stop working after flashing. We found out that, on some computers, after flashing by connecting the debugger module via USB 3.0 the Sensortag 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 repair a Sensortag that already been flashed over USB 3.0, it was first necessary to flash the hex file using the Flash Programmer 2 (again, only via USB 2.0).

With an early version of our modified firmware, the data stream coming from the Sensortag would freeze every few seconds, only to resume shortly thereafter. After talking with Group 2, which made similar modifications but didnt use the ”EXCLUDE” directives for the pre-processor, we were mislead into thinking that the directives themselves were at fault. This made us lose some time chasing a red herring. Debugging revealed however that the bug was actually a feature we inadvertently turned on, namely Wake-On-Motion. This interrupts the sending of sensor data when there is no movement for more than a few seconds. Turning the feature back off, neatly solved the problem.

B. Desktop Application Development

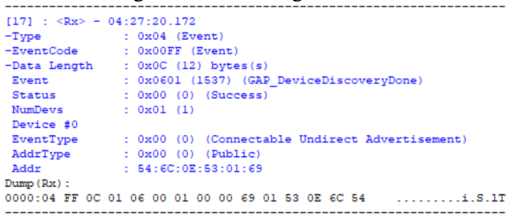
1) Problem with Connecting 2 SensorTags to the Application: Since SensorTags were configured and implemented as slaves we need a bluetooth profile as a server. For that we had different possibilities. First, we decided to develop a bluetooth profile directly on a Laptop device but there was no support or lib for that in Windows. All we could find was the Tinyb[20] project, which would work only under Linux and could be adapted for Windows implementing the Bluetooth profile in C++ then using JNI to use the code on our Java application.

The other possibilities were using a smart phone as a master device. We tried to do that in Python but from the lack of experience of our team in this programming language we decided to use a Launchpad, something that until then we were trying to avoid using because the Launchpad was a black box for us. Nevertheless, that was the final solution we decided on. Now the problem was how the Launchpad would communicate with the computer.
Windows recognizes the Launchpad as a serial port and we need to use that serial port to transfer messages between our application and Launchpad. For that we use jssc[14] library to open port and to Read and Write from that. This way we could write and read byte arrays to the port perfectly. We could also exhange messages with the master Bluetooth profile.
2) Deciding What to Send to the Launchpad: We used the help of BTool from the core TI applications. From there we could find the hexadecimal string to advertise the Launchpad, scan for discoverable bluetooth devices and connect to a desired one.
The problem now became how to read and translate the messages, the byte arrays we get from COM serial port.
From the documentation of TI Bluetooth profile[12] we found out that a new message from the Launchpad to the serial port always starts with the hexadecimal value of 0x04FF, then we used the TI Documentation to find out what the rest of the message means.

Another problem was that the serial port in Windows can read only 8 bytes at a time and the messages for Launchpad were much longer, so there we had a problem where we receive a

new 8 byte array, and we check if the first 2 bytes are 0x04FF. After that next bit was the length of the message, which is mainly where the problem lay. Lets suppose we receive the message 04FF0C010600010000 690153036C53
What this message means is displayed in Figure 11.
What we were reading from our application is the following:
04FF0C010600010000
690153036C53..next message.
Whenever we have a new array we check if it starts with 0x04FF. Then we need 0x0C =12 bytes for the data length but we have only 5 bytes left from the array before we get the next array so the message can be misinterpreted.

Figure 11. Message Structure

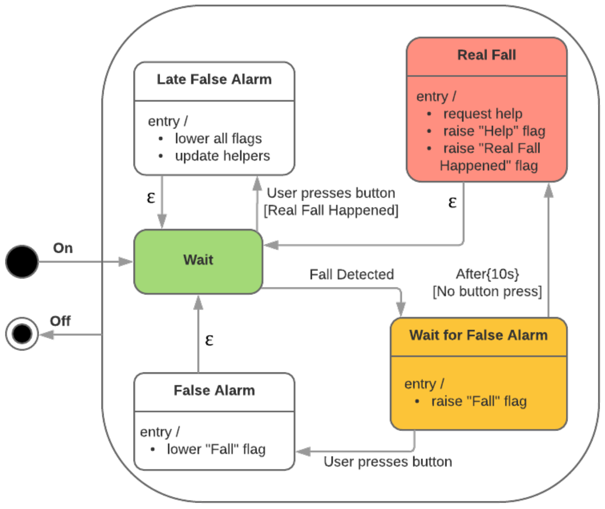


```
------------------------------------------------------------------------
[17] : <Rx> - 04:27:20.172
-Type            : 0x04 (Event)
-EventCode       : 0x00FF (Event)
-Data Length     : 0x0C (12) bytes(s)
 Event           : 0x0601 (1537) (GAP_DeviceDiscoveryDone)
 Status          : 0x00 (0) (Success)
 NumDevs         : 0x01 (1)
 Device #0
 EventType       : 0x00 (0) (Connectable Undirect Advertisement)
 AddrType        : 0x00 (0) (Public)
 Addr            : 54:6C:0E:53:01:69
Dump(Rx):
0000:04 FF 0C 01 06 00 01 00 00 69 01 53 0E 6C 54    .........i.S.lT
------------------------------------------------------------------------
```

To solve this we put everything that we get from the Launchpad in a byte queue then we create a new thread just to process that queue and read the messages. We read only the messages that we need and the others we just ignore.

3) Fall Notification Service (FNS): This is the service that handles the notifications triggered by the detection of a fall. Its code interacts with each and every other part of the desktop application and offers a good overview of its main features.
On start the FNS idles while waiting for a trigger. When the fall detection algorithm thinks the user has fallen, it alerts the FNS which turns on the buzzer in the Sensortag and raises the ”Fall Detected” flag in the GUI and then waits for a preset amount of time (default 10 seconds) for feedback from the user in the form of a button press on the Sensortag. If no such feedback comes, help is requested (an email is sent to the helpers address) and the ”Help Requested” flag is raised in the GUI. Instead, in case a button is pressed before the preset time has elapsed, the buzzer is turned off and the ”Fall Detected” flag lowered, after which the cycle repeats itself. The user can also signal a false alarm after help has been already request (that is, when in the ”Wait” state): in this case, the buzzer is turned off, all flags are lowered and a new message is sent to the helpers explaining that it was indeed a false alarm.
Below you see a state machine diagram illustrating the service operation:

Figure 12. Fall Notification Service



The content of the messages sent to the helpers, as well as the addresses of the helpers themselves, can be changed through the GUI. The waiting time after a fall is detected and before help is requested, can also be modified in the same way.

C. Graphical User Interface

To implement GUI based on the mock up made earlier, JavaFX Scene Builder[15] was used. JavaFX Scene Builder is a tool that lets users design a JavaFX applications 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 to establish and control the connections to the SensorTags.
Through the File menu the user can access the User

General Information and Settings windows. In User General Information the user should fill in the form with his data and the contact persons data.
In Settings menu the calibration values for the fall detection algorithm can be inserted.

References

[1] SSNS Group 1. *HIS SSNS - Group 1 - Bi-Weekly Report 1*. [Release Date: 14.05.2018].

[2] SSNS Group 1. *HIS SSNS - Project Proposal - Fall Detection Using Gyroscope and Accelerometer.pdf*. [Release Date: 03.05.2018].

[3] A. J. Albrecht. *Measuring Application Development Productivity*. in Proceedings of the Joint SHARE, GUIDE, and IBM Application Development Symposium, Monterey, California, 1979.

[4] Raul Bertone. *Estimation.pdf*. [Release Date: 05.07.2018].

[5] *BLE Host Test Project*. http://dev.ti.com/tirex/content/simplelink_academy_cc13x2sdk_1_15_03_10/modules/projects/ble_hosttest/information.html. [Accessed: 03.05.2018].

[6] M.Sc. L. La Blunda. *Project Proposal*. https://moodle.frankfurt-university.de/pluginfile.php/512521/mod_forum/attachment/94672/Project1.pdf. [accessed 27.04.18].

[7] B. Boehm. *COCOMO II Model Definition Manual*. http://csse.usc.edu/csse/research/COCOMOII/cocomo2000.0/CII_modelman2000.0.pdf. [accessed 27.04.18].

[8] *Github Repository*. https://github.com/raulbertone/SSNS. [Accessed: 05.07.2018].

[9] Object Management Group. *UML 2.5.1 specification*. p. 643. December 2017.

[10] Elis Haruni. *IC-Risk-Management-Matrix-SSNS-Project.xlsx*. [Release Date: 03.05.2018].

[11] Texas Instruments. *CC26x0 SimpleLink Bluetooth low energy Software Stack 2.2.x Developer's Guide*.

[12] Texas Instruments. *TI BLE Vendor Specific HCI Reference Guide BLE5*. Version 1.0.0.

[13] InvenSense. *MPU-9250 Product Specification*. Document Number: PS-MPU-9250A-01, Revision 1.1. [Release Date: 6.20.2016].

[14] *Java Simple Serial Connector*. https://code.google.com/archive/p/java-simple-serial-connector/. [Accessed: 05.07.2018].

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

[16] *Moodle Website*. https://moodle.frankfurt-university.de/mod/forum/view.php?id=172434.

[17] *Movement Sensor*. http://processors.wiki.ti.com/index.php/CC2650_SensorTag_User. [Accessed: 05.07.2018].

[18] *ObjectAid UML Explorer for Eclipse*. http://www.objectaid.com/home. [Accessed: 05.07.2018].

[19] *Slack Main Website*. https://slack.com.

[20] *TinyB Project*. https://tiny-project.com/. [Accessed: 05.07.2018].