HIS SSNS - Fall Detection based on Accelerometer and Gyroscope Data

Raul Bertone Elis Harruni Muyassar Kokhkharova Saidar Ramazanov Xhoni Robo

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[3].**

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 7 weeks, with the latest possible delivery date being June 22nd.

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) The graph should depict the average value of received sensor data within the offset time.
6) Possibility to enter Users general information.
7) Possibility to calibrate system thus differentiate between sudden movements like walking the steps and free fall.
8) 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 Person1 (In case of fall this Person will be contacted.)
   - Contact Number1 (Phone number of a Contact Person1.)
   - Contact Person2 (In case of fall and if Contact Person1 is not replying this Person will be contacted.)
   - Contact Number2 (Phone number of Contact Person2.)
   - Contact Person3 (In case of fall and if Contact Person1 and Contact Person2 are

not replying this Person will be contacted.)
- Contact Number 3 (Phone number of Contact Person3.)
- Save button to save changes

2) Application Settings is a pop up window containing the following information:
    a) Offset with default value of 0.25 seconds.
    (Graphs of gyroscope and accelerometer will be updated every offset time with average values of gyroscope and accelerometer received during offset time.)
    b) Delta for fall detection
    (While falling values of accelerometer will gain acceleration and then will be equal to null. If delta of accelerometer received from sensor is bigger than configured delta in settings it will mean that the person has probably fallen. The duration of free fall is also important because we have to differentiate between sudden movements like walking the steps and free fall, the duration will be different although delta values might be the same. While falling values of gyroscope will change and after reaching the ground values will be in horizontal position.)
    c) Set Default button
    (After clicking on it, all the default values will be set and Settings pop up window will remain open.)
    d) Save button
    (After clicking on save button, all the changes will be saved and Settings pop up window will remain open.)
    e) Cancel Button
    (Cancel changes)
    f) Close button (Clicking on close button will close Application Settings pop up window)
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) Buttons start/stop receiving gyroscope and accelerometer data.
    e) Button Clear graph.
    f) Label for Sensor Status (Connected to the sensor or No connection).
    g) Label for Fall Detection.
    (If fall is detected inscription about falling will be displayed. If not inscription ”Fall was not detected” will be displayed.)
    h) Label for ”help requested”
    (If fall was detected and user did not pressed the button on SensorTag to report about False alarm inscription on requesting help will be displayed.)
    i) When the Sensor is not connected to the PC Application(e.g. when Data is not being received), than Buttons Start, Stop and Clear should be deactivated*
    j) Stop button is Activated when Start is pressed.
    k) ”False Alarm”
    (When device senses a fall it beeps first and if button on SensorTag is not pressed quickly it calls for help)

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) Software correctly notifies user when one or both sensor tags do not work
3) Software or SensorTags correctly notify user when connectivity is low
4) If transmission stops abruptly during an activity similar to a fall, count that as a fall

2) Security Requirements:

1) Data is collected from the associated SensorTags
2) Other devices cannot send data to the application
3) If any interferences are detected, notify the user

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) Software notifies user when the SensorTags are low on battery
4) 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 [4], which was based on the value of Function Points [1].

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 standalone systems, 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.

2) Transactions: In the following table Transaction (External Input, External Output, External Inquiry) are listed.
They are subdivided according to the Actor that is responsible for them.

Table I
TRANSACTIONS

	External Input	**External Output**	**External Inquiry**
User	Insert system calibration(3FP) Insert user general information(3FP) Insert helper contact information(3FP) Load defaults(3FP)	Accelerometer graph(4FP) Gyroscope graph(4FP) Label ”Fall detection”(4FP) Label ”Help requested”(4FP)	Open application settings(3FP) Start/Stop(3FP) Connect/Disconnect(3FP) Close button(3FP) Clear graph button(3FP)
Helper		Send email(4FP)	
Sensors	Gyroscope(3FP) Accelerometer(3FP) Snooze Alarm(Button 2)(3FP) Bluetooth connection PC(4FP) Bluetooth connection Sensortags(4FP)		
Actuators		Buzzer ”false alarm”(4FP)	

3) Internal Logical Files: In the following table, ILFs are listed. They are subdivided according to the software module they belong to.
Total (unadjusted) Function Points: 89

4) Estimation of Effort: The estimation of effort was conducted with COCOMO II. Considered the early stage of development of the project, the Early

Table II
INTERNAL LOGICAL FILES

Module	**Internal Logical Files**
Sensortags	None
PC Application	User Information(7FP) System calibration values(7FP) Helper contact data(7FP)

Design Model was selected.

Table III
SCALING DRIVERS

Driver	*Value*
Precedentedness	High
Development Flexibility	Nominal
Risk Resolution	Nominal
Team Cohesion	High
Process Maturity	Very Low

Table IV
COST DRIVERS

Driver	*Value*
Facilities	Nominal
Personnel Experience	Nominal
Personnel Capability	High
Required Reusability	Low
Platform Difficulty	Nominal
Product Reliability and Complexity	Low
Required Development Schedule	Nominal

Table V
RESULTS

	Value
Person-Months	9.7
Schedule Months	1.9
SLOC	4717

Notes

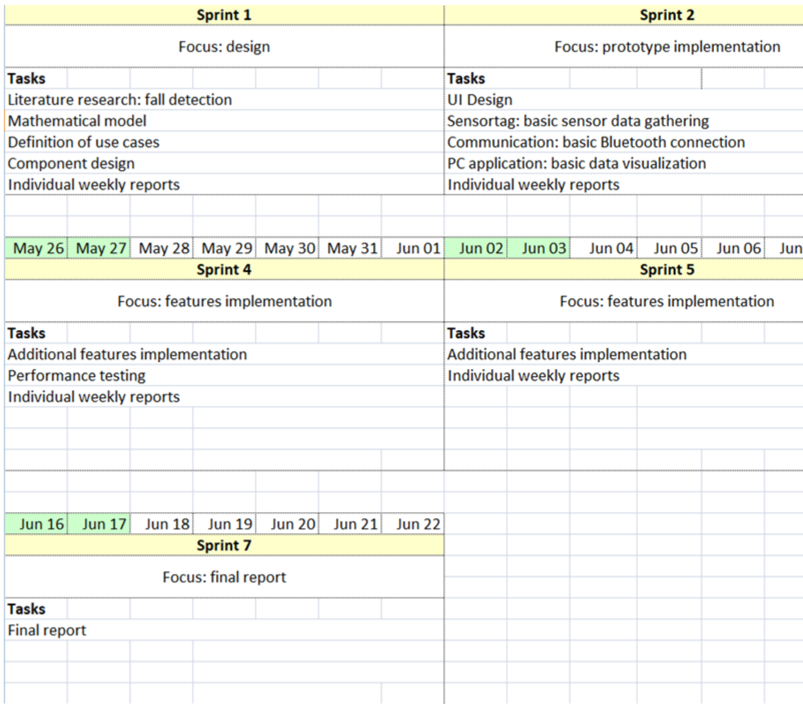
The result of 1.9 months seems at first encouraging. However, our analysis must also consider that, on one hand, the team is working only part-time on the project, and on the other, that the final product will be a prototype, not a production ready system. In light of these facts, we believe the estimation to be generally accurate.

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, tentatively on Tuesdays at 11:00 and Fridays at 15:00.
Official communication was organized through two channels:

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

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



Sprint 1
Focus: design
Tasks
Literature research: fall detection
Mathematical model
Definition of use cases
Component design
Individual weekly reports

May 26	May 27	May 28	May 29	May 30	May 31	Jun 01

Sprint 2
Focus: prototype implementation
Tasks
UI Design
Sensortag: basic sensor data gathering
Communication: basic Bluetooth connection
PC application: basic data visualization
Individual weekly reports

Jun 02	Jun 03	Jun 04	Jun 05	Jun 06	Jun

Sprint 4
Focus: features implementation
Tasks
Additional features implementation
Performance testing
Individual weekly reports

Jun 16	Jun 17	Jun 18	Jun 19	Jun 20	Jun 21	Jun 22

Sprint 5
Focus: features implementation
Tasks
Additional features implementation
Individual weekly reports

Sprint 7
Focus: final report
Tasks
Final report

Figure 1. Project Schedule

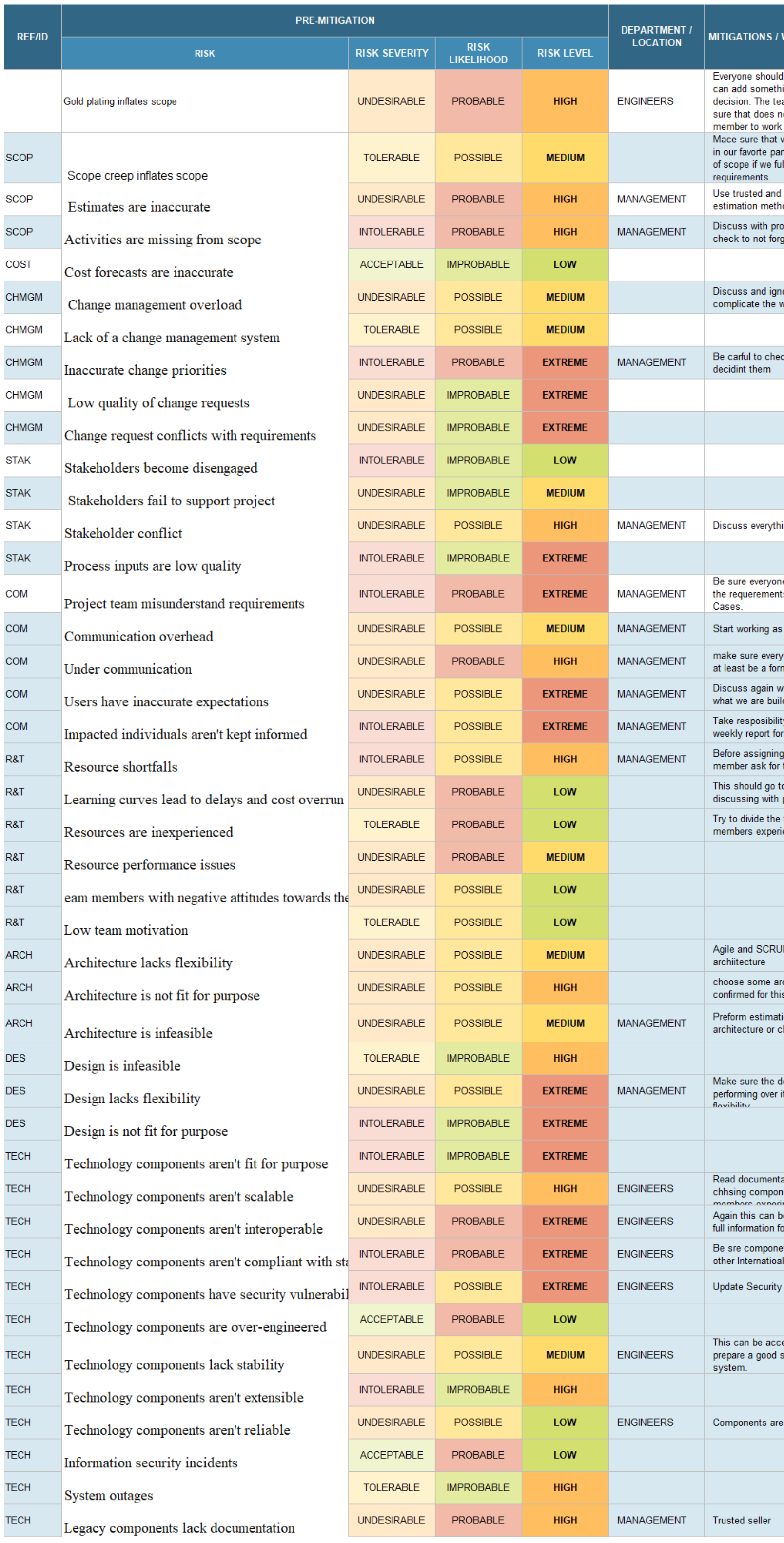
C. *Responsibilities of the Team Members*

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 VI
ROLES OF THE TEAM MEMBERS

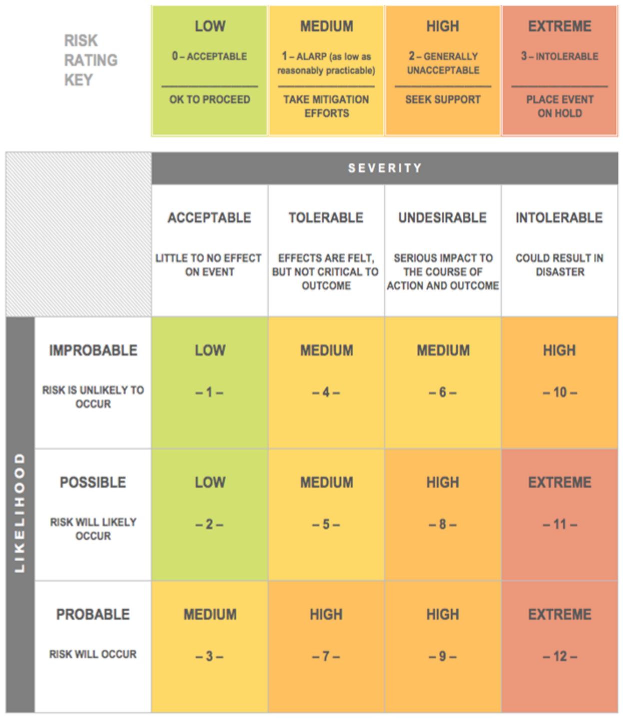
Name	**Main Role**
Raul Bertone	Project Manager, Scrum Master
Elis Harruni	Lead Java Developer
Muyassar Kokhkharova	Statistics, UI Designer
Saidar Ramazanov	Mathematical Model
Xhoni Robo	Lead C Developer

D. *Project Risk Analysis*



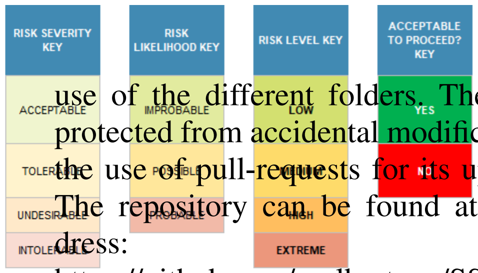
REF/ID	PRE-MITIGATION				DEPARTMENT / LOCATION	MITIGATIONS / V
	RISK	RISK SEVERITY	RISK LIKELIHOOD	RISK LEVEL		
	Gold plating inflates scope	UNDESIRABLE	PROBABLE	HIGH	ENGINEERS	Everyone should can add somethi decision. The tea sure that does n member to work
SCOP	Scope creep inflates scope	TOLERABLE	POSSIBLE	MEDIUM		Mace sure that v in our favorte par of scope if we ful requirements.
SCOP	Estimates are inaccurate	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	Use trusted and estimation metho
SCOP	Activities are missing from scope	INTOLERABLE	PROBABLE	HIGH	MANAGEMENT	Discuss with pro check to not forg
COST	Cost forecasts are inaccurate	ACCEPTABLE	IMPROBABLE	LOW		
CHMGM	Change management overload	UNDESIRABLE	POSSIBLE	MEDIUM		Discuss and ign complicate the w
CHMGM	Lack of a change management system	TOLERABLE	POSSIBLE	MEDIUM		
CHMGM	Inaccurate change priorities	INTOLERABLE	PROBABLE	EXTREME	MANAGEMENT	Be carful to chec decidint them
CHMGM	Low quality of change requests	UNDESIRABLE	IMPROBABLE	EXTREME		
CHMGM	Change request conflicts with requirements	UNDESIRABLE	IMPROBABLE	EXTREME		
STAK	Stakeholders become disengaged	INTOLERABLE	IMPROBABLE	LOW		
STAK	Stakeholders fail to support project	UNDESIRABLE	IMPROBABLE	MEDIUM		
STAK	Stakeholder conflict	UNDESIRABLE	POSSIBLE	HIGH	MANAGEMENT	Discuss everythi
STAK	Process inputs are low quality	INTOLERABLE	IMPROBABLE	EXTREME		
COM	Project team misunderstand requirements	INTOLERABLE	PROBABLE	EXTREME	MANAGEMENT	Be sure everyone the requerements Cases.
COM	Communication overhead	UNDESIRABLE	POSSIBLE	MEDIUM	MANAGEMENT	Start working as
COM	Under communication	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	make sure every at least be a forr
COM	Users have inaccurate expectations	UNDESIRABLE	POSSIBLE	EXTREME	MANAGEMENT	Discuss again w what we are buil
COM	Impacted individuals aren't kept informed	INTOLERABLE	POSSIBLE	EXTREME	MANAGEMENT	Take resposibility weekly report for
R&T	Resource shortfalls	INTOLERABLE	POSSIBLE	HIGH	MANAGEMENT	Before assigning member ask for t
R&T	Learning curves lead to delays and cost overrun	UNDESIRABLE	PROBABLE	LOW		This should go to discussing with p
R&T	Resources are inexperienced	TOLERABLE	PROBABLE	LOW		Try to divide the members experie
R&T	Resource performance issues	UNDESIRABLE	PROBABLE	MEDIUM		
R&T	eam members with negative attitudes towards the	UNDESIRABLE	POSSIBLE	LOW		
R&T	Low team motivation	TOLERABLE	POSSIBLE	LOW		
ARCH	Architecture lacks flexibility	UNDESIRABLE	POSSIBLE	MEDIUM		Agile and SCRU archiitecture
ARCH	Architecture is not fit for purpose	UNDESIRABLE	POSSIBLE	HIGH		choose some ar confirmed for this
ARCH	Architecture is infeasible	UNDESIRABLE	POSSIBLE	MEDIUM	MANAGEMENT	Preform estimati architecture or c
DES	Design is infeasible	TOLERABLE	IMPROBABLE	HIGH		
DES	Design lacks flexibility	UNDESIRABLE	POSSIBLE	EXTREME	MANAGEMENT	Make sure the d performing over it
DES	Design is not fit for purpose	INTOLERABLE	IMPROBABLE	EXTREME		
TECH	Technology components aren't fit for purpose	INTOLERABLE	IMPROBABLE	EXTREME		
TECH	Technology components aren't scalable	UNDESIRABLE	POSSIBLE	HIGH	ENGINEERS	Read documenta chhsing compon
TECH	Technology components aren't interoperable	UNDESIRABLE	PROBABLE	EXTREME	ENGINEERS	Again this can b full information fo
TECH	Technology components aren't compliant with sta	INTOLERABLE	PROBABLE	EXTREME	ENGINEERS	Be sre compone other Internatioal
TECH	Technology components have security vulnerabil	INTOLERABLE	POSSIBLE	EXTREME	ENGINEERS	Update Security
TECH	Technology components are over-engineered	ACCEPTABLE	PROBABLE	LOW		
TECH	Technology components lack stability	UNDESIRABLE	POSSIBLE	MEDIUM	ENGINEERS	This can be acce prepare a good s system.
TECH	Technology components aren't extensible	INTOLERABLE	IMPROBABLE	HIGH		
TECH	Technology components aren't reliable	UNDESIRABLE	POSSIBLE	LOW	ENGINEERS	Components are
TECH	Information security incidents	ACCEPTABLE	PROBABLE	LOW		
TECH	System outages	TOLERABLE	IMPROBABLE	HIGH		
TECH	Legacy components lack documentation	UNDESIRABLE	PROBABLE	HIGH	MANAGEMENT	Trusted seller

Figure 2. 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 –
LIKELIHOOD: POSSIBLE RISK WILL LIKELY OCCUR	LOW – 2 –	MEDIUM – 5 –	HIGH – 8 –	EXTREME – 11 –
LIKELIHOOD: 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 3. Risk Management Matrix Key

E. *Setup and Description of Development Environments*

SensorTags
The development for the SensorTags (and if necessary for the Launchpad) will be in C. For it we will use Texas Instruments own Code Composer Studio as well as the toolchain provided by Texas Instruments.

PC Application
The PC application will be implemented in Java. For its development we will use Eclipse as an IDE, Junit for unit testing. All mentioned software are available for both Linux and Windows, so we will leave the choice of an OS open for each team member (none makes use of MacOS). Additionally, the choice of Java as the implementation language allows the deployment and testing of the PC application on all OSs.
As a version control system software we selected git, and, based on this, GitHub as an online shared repository. All common artifacts (source code, documentation, etc.) will be uploaded there. In the root folder a readme file will describe the intended use of the different folders. The master branch is protected from accidental modifications by requiring the use of pull-requests for its update.
The repository can be found at the following address:
https://github.com/raulbertone/SSNS
We integrated GitHub into our Slack group to be timely informed about commits performed by other team members.

III. Design

A. *Use Case Diagram*

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

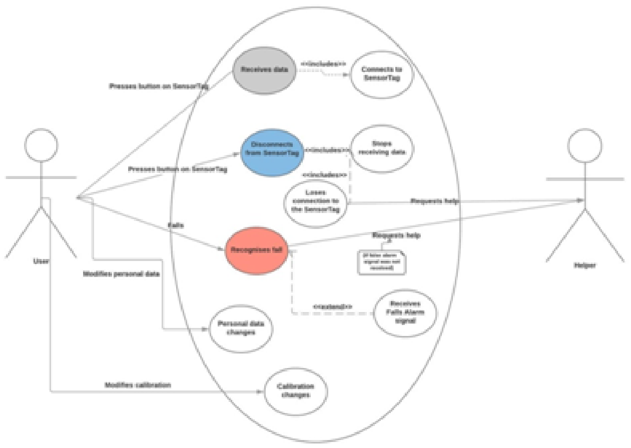
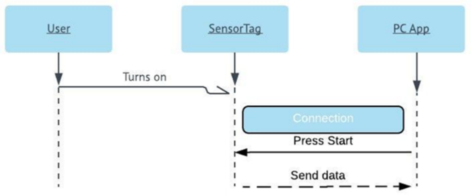


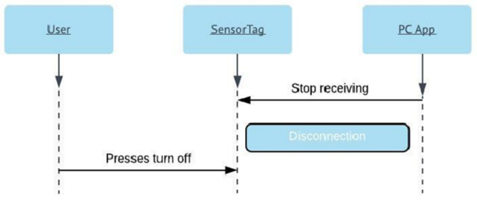
Figure 4. Use Case Diagram

B. Sequence Diagrams

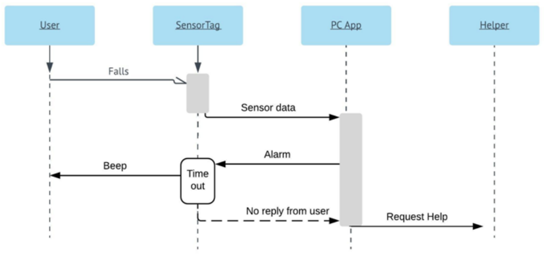
Activate:



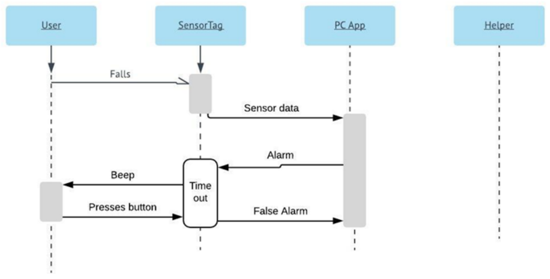
Deactivate:



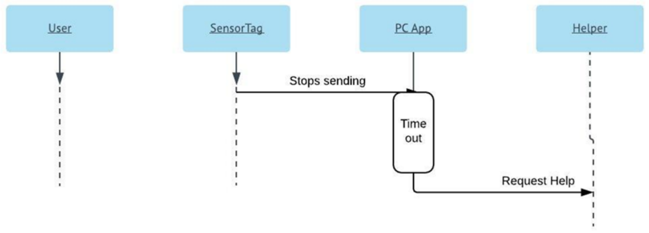
Fall:



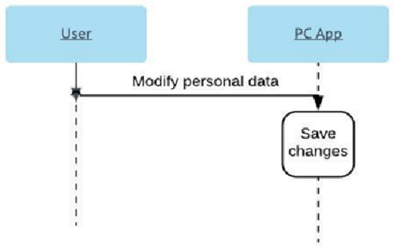
False Alarm:



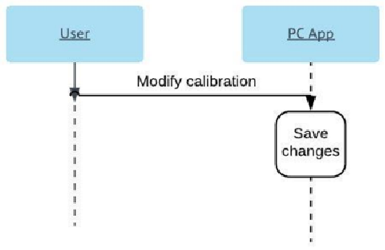
Out of range:



Modify personal data:



Modify calibration:



C. Architecture

1) Component Diagram: 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 dont communicate directly with each other.

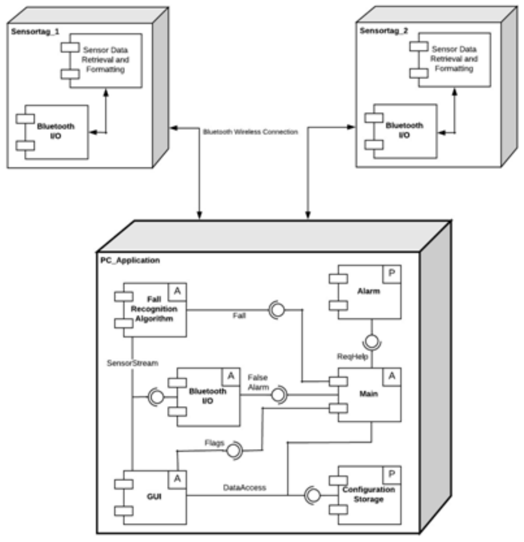


Figure 5. Component Diagram

2) 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.

Its important to note that the modules described above dont necessarily represent different files or libraries, but only different logical functions of the application, which might very well be implemented in the same file.

3) 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 dont necessarily correspond to specific packages. However, is reasonable to expect that several of them will be implemented as independent packages.
4) Interfaces: False Alarm
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

IV. 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.

A. 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).

B. 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

C. 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.

D. 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 pre-processed. 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.

E. Graphical User Interface

To implement GUI based on the mock up made earlier, JavaFX Scene Builder[7] 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.

F. Desktop Application Development

[8][5][2] First a recap of what I was doing I am working all the time with establishing BLE connection and creating a network to read the required data from the sensors.
I started directly with multi_role project that was

suggested to me I could connect tow Slaves to the lunchpad (Master) but to discover Characteristics and read/write I could not do. That happened because the Write/read function of multi role project was not implemented. Considering to implement that function and solve the problem I decided not to do it because of my restricted knowledge on C and C++ from this decision starts the work of this 4 weeks.

First 2 weeks using host_test application with lunchpad again as a master I created a network for 1 sensor as a slave reading data and saving them in queue to print on console after that just to see the messages.
The implementation was done by the help of Java library to read and write to a serial COM port called jssc.jar.

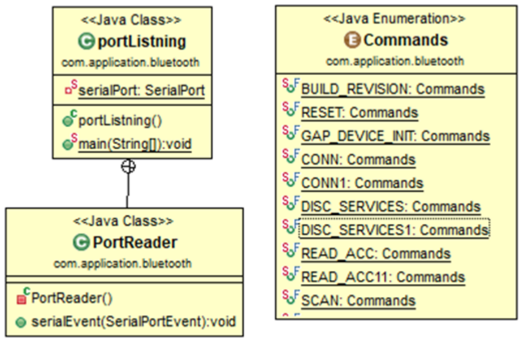


Figure 6. UML diagram for the first prototype of Network system in JAVA

First I needed only this three classes portListning PortReader that is a private inner class of port listning and an Enumeration for some commands that I need to send to lunchpad.
In the port listning Class the main method search for available COM ports and print them after that it ask you what port want to connect and open. After that activating a SerialPortEvent to the opened port will notify this thread every time that an event is happening to the port write/read. On the PortReader Class that implements SerialPortEvent Interface implementing the required method serialEvent I only catch read events so when the lunchpad is sendig me data and I just print the hexadecimal Sring in console to check the data.
The time consuming part was Creating the so called Commands I started to build commands using TI BLE Vendor Specific HCI Reference Guide BLE5 Version 1.0.0.
Creating commands like that was taking a lot of time and I figured out another method sniffing the port with a port sniffer application while I used BLE device monitor for getting data. Then o got all the writing commands with the help HCI Reference Guide I could translate all messages so I Go only the commands that I need to read from Accelerator. Here I was trying to get data only from Accelerator then later will think for Gyroscope.
A the end I could get data for my Application but I need to translate those data and to parse in the way I get a use of them.

This 2 week I continue on building the NS (Network System) part of SSNS.

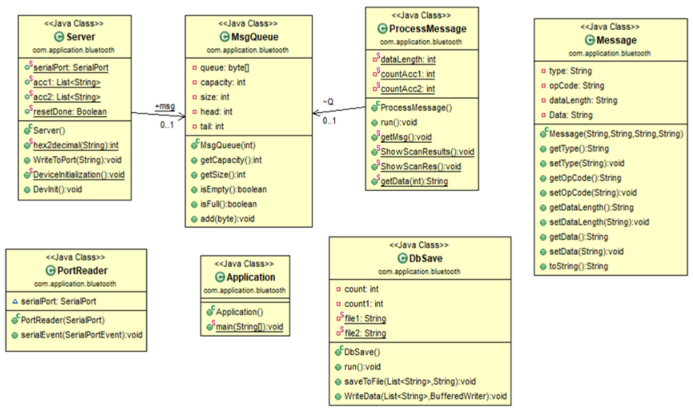


Figure 7. New current Application UML diagram.

This is the auto generated UML diagram from an Eclipse plugin that's why there are some relations missing without a detailed explanation this UML is not understandable because of missing elements. Also here the Commands Enumeration is missing because is very big for the figure.
Here we have now 7 classes this is developed from the first Application in previous weeks with 3 classes PortReader is the same but instead

of a private inner class now I use Aggregation and Inheritance to implement the same thing. portListning was not named good for Java convention by mistake and with the new ideas I got a new name for it that is Server and the main() method now is in a new class Called Application that is somehow the user interface for now(Consloe view). Now I dont print the message on the Console when I get an event but insert the byte array to a Byte Queue I implemented and to process the Messages and to parse and translate the messages I use a new thread that gets the data from Queue so I get lower chances to interrupt or to put the main Thread in a lot of work and risk losing data.
I have defined a structure of the every message with the help of TI BLE Vendor Specific HCI Reference Guide BLE5 Version 1.0.0.
DbSave class is just a class to save Accelerator data in 2 different files for different sensors. So with this Structure I can connect 2 sensortags and get data from them in the same time. Apart for the successful work I was working on developing a python script to read the data from sensor and also with android code from TI but I then gave another try to this before finding a solution on one of those 2.

V. Measurement Chain[6]

For our application we make use only of the MPU-9250 sensor pack, 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 would require adaptation of the fall detection algorithms to many different user/clothing combinations.
The measured quantities are: the total resulting acceleration (gravity plus motion); and angular velocity.
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 transmission induced errors.

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:

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

VI. Fall Detection Algorithm

On previous weeks was implemented additional functional of Gyroscope method of Fall Detection. As it was mentioned on the last report, the logic of work is:

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

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

then it is mean that patient is laying and Gyroscope class proofs the fall.
First problem was to state the timeframe when the fall was. In that point, algorithm looking for the last measurements of **-1 g** on OZ axis and states it as beginning of fall. Then the end of fall is stated by time when the impact was proceed. By this, we can state the timeframe of the fall for the Gyroscope algorithm. However, if the measurement of OZ axis is 0 g it means that the user was laying before the fall.

One of the main problem during the implementation was that the program could not change array of measurements during the computations. That is why there was implemented footprint of the measurements. It brings some disadvantages, as the program should store a lot of information for computations. Nevertheless, it is one solution, which can be done in case to do not lose important measurements from SensorTag.

Finally, the architecture of the Maths logic now contains three classes:

- Mathematics 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;
- Gyro contains measurements from sensor, functions off adding new measurements, thread of Gyro computations;
- Accelerometer - contains measurements from sensor, functions off adding new measurements, can trigger Gyro method to start its thread.

References

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

[2] *BLE User Guide*.

[3] 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].

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

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

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

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

[8] *TI BLE Vendor Specific HCI Reference Guide BLE5 Version 1.0.0*.