

**FallAlarm – Fall Detection and Alarm App**

## Daniel Hunegnaw | EE P 523 Mobile Applications for Sensing and Control| 08-015-21

# **OBJECTIVE**

The objective of this paper is present the design, and implementation of an Android mobile App called FallAlarm, which is used to detect when a person carrying the phone falls. The project aims to provide knowledge, and hands-on experience in designing, and programming of android app using Kotlin programming language. The project uses the built-in accelerometer sensor present in android phones as source of data for detecting fall.

# **INTRODUCTION**

In this paper, it will be presented the design and implementation of an android app for detecting when a person carrying the phone falls, alerting an emergency contact when a fall is detected via SMS. The app uses the built-in accelerometer , GPS, SMS messaging system in the phone, SQLite database for storing data and is written in Kotlin.

In section I, a brief description of solution modeling and creating decision thresholds will be presented.

In section II, a high-level design of the app and the interaction of various components will be presented using a block diagram will be described. In addition, the implementation of each component along with the libraries used will be presented

In Section III the issues faced during the project and future enhancements works will be briefly explained.

Finally, in Section IV, we will provide a conclusion and references used in the project

# **SECTION I**

## Modeling thresholds

To detect a fall, a threshold-based algorithm is proposed. Thus, the first phase of the project was to establish thresholds that will enable us to decide if a fall was detected. In this modeling phase of the project, a simple accelerometer data collection app was developed. The app collects the vector magnitude of the three components of the accelerometer readings using the formula:

Am = √ (A2x+ A2y +A2z)

The app collects the data in an array. Upon a click event of a save button, the data (list of numerical values of the acceleration magnitude in a chronological order) is saved in a file.

Sample data collection was done as follows

1. **Walking** : Run the app, put the phone in a pant-pocket and walk around , and save the collected data
2. **Walk-Sit-Stand** : Run the app, put the phone in pant-pocket and walk, sit, stand, and walk again, and save the data collected
3. **Walk-Fall**: Run the app, put the phone in a pocket, and free fall on a mattress. And save the collected data
4. Data collected from each scenario was exported to excel and a liner graph (x-axis is a sequence of numbers representing relative time), and y-axis representing the acceleration in m/s2 was generated as shown in fig 1 below

Fig 1. Graphical representation of sample data collected.

Interestingly, as one can see from the graph, there is not much difference between walking and walk-sit-stand graph (red and gray lines) patterns. However, for the **walk and fall**, during the fall, the acceleration goes very low close to zero (less than 0.3m/s2, three down bursts of the yellow line) and immediately raises about 10m/s2 upon landing on the ground. Thus , after doing several similar experiments, an average lower and upper threshold patterns was developed. During the fall , the acceleration goes below a **lower** **threshold of 2.5m/s2** , and after impact, the acceleration magnitude goes above an **upper threshold of 10m/s2** within a specific during of **time [0.5, 1] seconds.**

## Modeling thresholds

The following algorithm was developed based on the above modeling and analysis of data



# **SECTION II**

## Design the APP

## Diagram Description automatically generated

## Implementation

## Libraries used

# **SECTION III**

## ISSUES

## POTENTIAL Improvement

# **SECTION IV**

## CONCLUSION

## REFERENCES

# FallAlam – The Fall Detection App Project

In this project , I am proposing to design, and implement a fall-detector app, called FallAlarm. The main purpose of the FallAlarm app is to detect when a person falls\* and to call or to text (with location information) an emergency contact that the user sets up in the App. The project will be implemented using Kotlin, Android Studio, Android Phone/Emulator. Since a fall involves an acceleration from a higher position to a lower position, the accelerometer sensor of an android device will be used a source of data for detecting a fall. In addition, the GPS will be used to get a location information , which is needed to alert an emergency contact in text (SMS).

The project will have two phases. In the first phase, a simple data collection app will be developed to gather accelerometer data to be used for modeling various states (or creating a threshold) of a person’s state such as normal activity, walking, falling etc .

In phase two, based on the model developed in phase one, an algorithm will be designed to detect a fall and will be implemented in the main app. In addition, a user’s settings, and a notification system to notify an emergency contact about the fall incident of the user will be implemented. The notification can be directly calling the emergency contact or just texting about the incident along with a location information.

To detect a fall, a threshold-based algorithm is proposed in this project. Thus, the first phase of the project is to establish thresholds that will enable us to detect a fall. In this modeling phase of the project, a simple accelerometer data collection app was developed. The app collects the vector magnitude of the three components of the accelerometer readings using the formula:

Am = √ (A2x+ A2y +A2z)

The app collects the data in an array. Upon a click event of a save button, the data (list of numerical values of the acceleration magnitude) is saved in a file.

Sampling is done as follows

1. **Walking** : Run the app, put the phone in my phone and walk around , and save the collected data
2. **Walk-Sit-Stand** : Run the app, put the phone in my pocket and walk, sit, stand, and walk again, and save the data collected
3. **Walk-Fall**: Run the app, put the phone in my pocket, and free fall on a mattress. I fall three times for this scenario and saved the collected data.
4. Data collected from each scenario was exported to excel and a liner graph (x-axis is a sequence of numbers representing relative time), and y-axis representing the acceleration in m/s2 was generated as shown in fig 1 below

Fig 1. Fall detection modeling

**Conclusion**

Interestingly, with this modeling and graph, there is not much difference between walking and walk-sit-stand graph ((red and gray lines) patterns. However, for the **walk and fall**, during the fall, the acceleration goes very low close to zero (less than 0.3m/s2, three down bursts of the yellow line) and immediately raises about 11m/s2 upon landing on the ground. Thus , we can assume a fall is detected when the acceleration is falls below 0.3m/s2 and followed by a rise above 11m/s2.

**Challenge**

One of the biggest challenges in detecting a fall is false positives. For example, jumping from a platform or just simply jumping up and landing back has the same characteristics as a fall. A way to minimize false positive has yet to be found.