ECB INCOPERATIONS
USER MANUAL
EMERGENCY CALL BELL

Authors: Adeeb Ali Urfani, Daniel Azza, Walter Wong, Punit Daga

Course: GNG 2101

Table of Contents

Abstract

The problem our group was faced with was to build an emergency call bell that can detect fall an assault and quickly be manually triggered to call the emergency services. The solution to this problem was to modify a Fitbit device using a form of code so that it could carry the services of an emergency call bell. The software we used to implement this was comprised of the following:

Programming Languages:
- Javascript / CSS / SVG (Fitbit companion)
- Java (Android App)

SDKs:
- Fitbit SDK
- Android Studio SDK

Libraries:
- GSON
- SmartLocation
- Java-Websockets
- AndroidFallDetectionApp

Introduction

To begin with it is important to solve this issue because assault and fall are dangerous to our community especially the elderly. We at ECB believe that we have the right to guarantee them their safety. The basic user of such a device requires it be attachable and hard to take off, be able to charge via a USB, have a smartphone application alongside the fitbit, be able to add secondary contact, be able to detect fall/abuse, be able to have bluetooth paring and lastly be able to call 911 and the emergency contacts once fall or abuse is detected.

The difference in our design compared to the competition out there is that there are no hardware costs associated with our product which is software-based and free. We believe our product is better as compared to the competition out there because people should get a feasible solution to a problem they should not be facing.

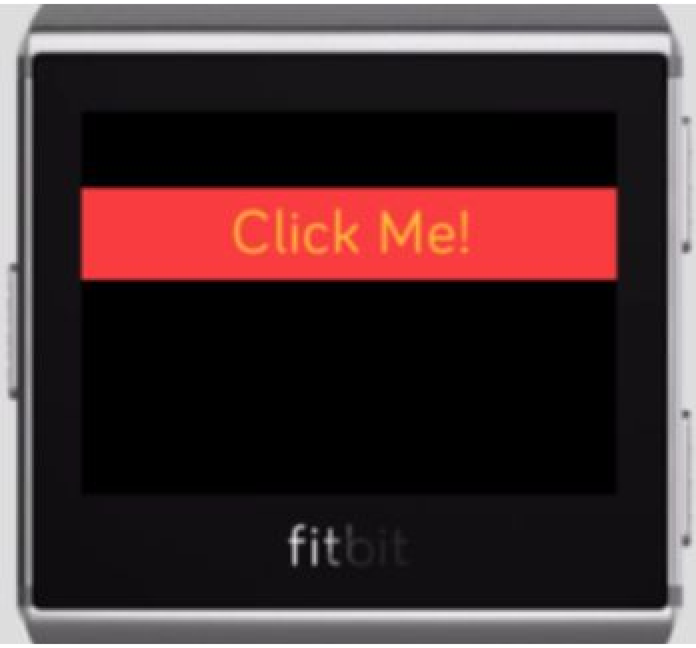
Features

- Fall detection
- Automatic calling to emergency services
- Location sending to secondary contact
- Fitbit compatibility
- Accessing the phone’s GPS to identify the user’s current location
- Monitoring the heartbeat of client

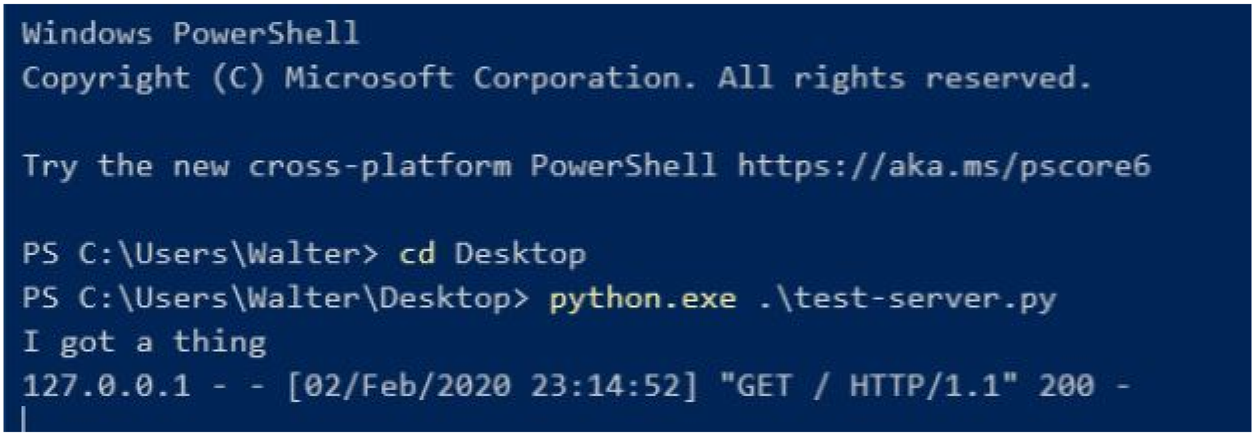
Functions

1. The user falls down, this action is noted by the fall detection system on the Fitbit app
2. The user receives a prompt on their Fitbit asking if they would like to call emergency services
3. If the user fails to respond to the prompt emergency services are automatically contacted via the user’s smartphone and a text detailing the location and time of the fall is sent to an emergency contact.

How Were the Prototypes Made



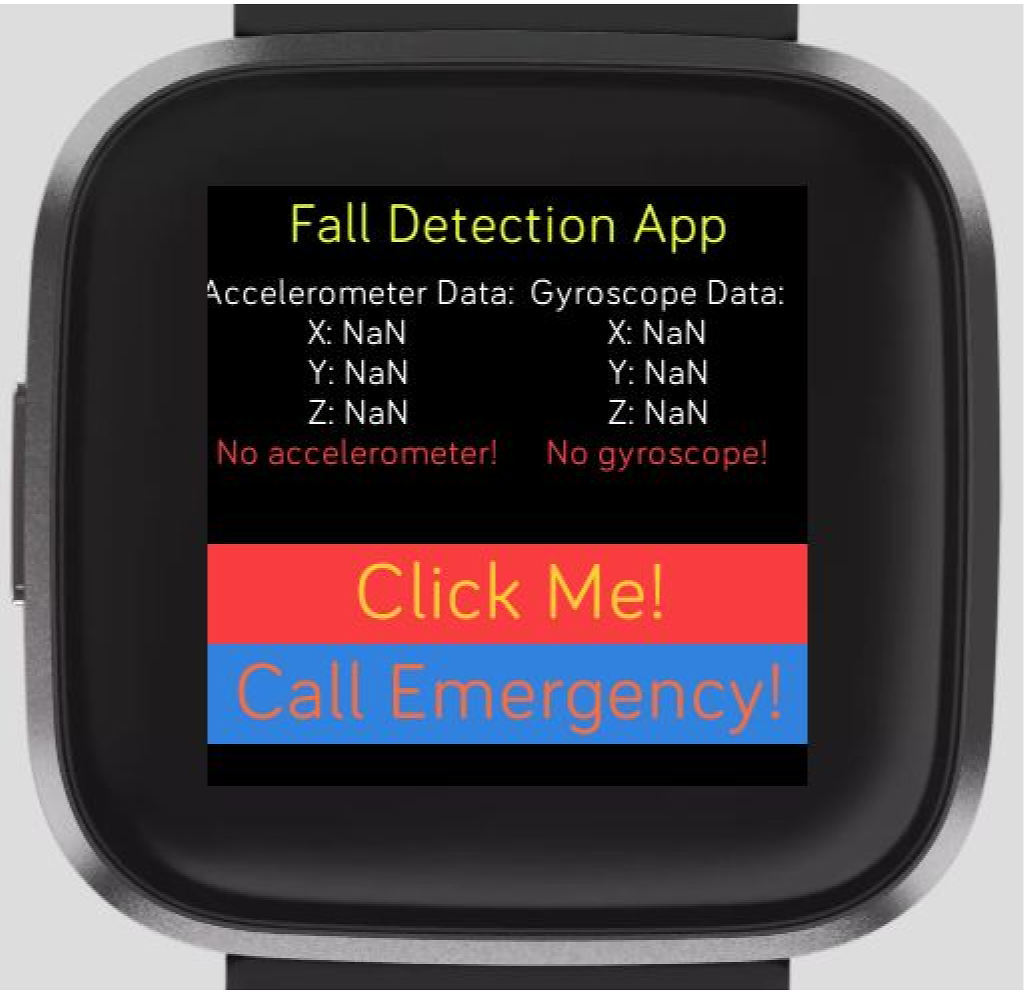
The first prototype was made to check if the Fitbit app would be able to communicate with a remote server on the android phone. This was made using the Fitbit SDK. The android app at this point was just a skeleton with no logic in it yet.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Walter> cd Desktop
PS C:\Users\Walter\Desktop> python.exe .\test-server.py
I got a thing
127.0.0.1 - - [02/Feb/2020 23:14:52] "GET / HTTP/1.1" 200 -
```



The second prototype is able to collect accelerometer data and is able to send a test message to the server (on the phone) by clicking the button on the Fitbit. At this point, the android app has the ability to call and message anyone.

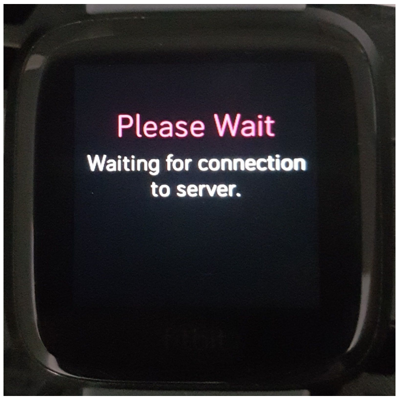
The logic in the android app was added using Java and Google's pre-built libraries in Android Studio.

How Does the Final Prototype Work

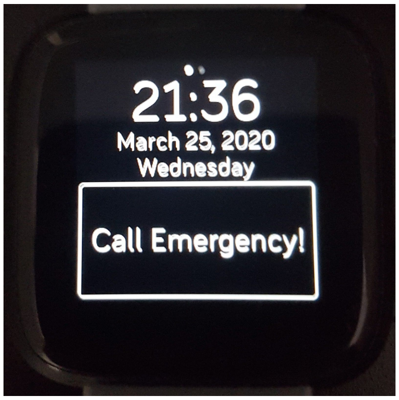
The final prototype works by sending the accelerometer data from the Fitbit directly to the phone. Here, the android companion app checks to see if there was a fall. If a fall is detected, it calls the emergency services and sends a message to the secondary contact along with the address of the client.

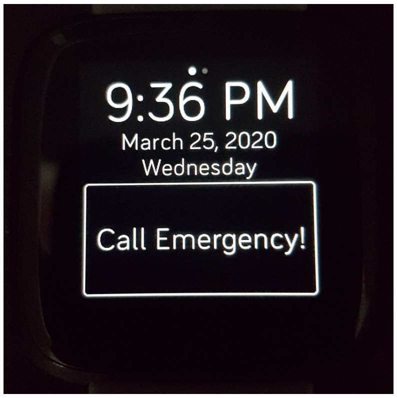
The fall detection is done using the open source fall detection algorithm done by alessandror available on Github.

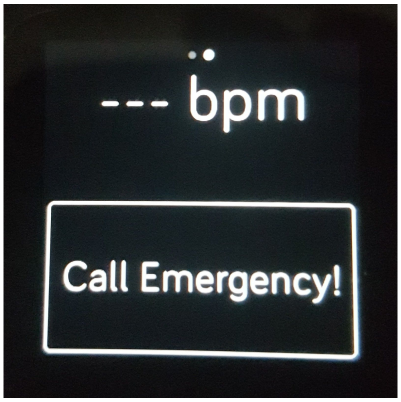
As seen in the images, when launching the app, the first thing it does is establish connection to the server. Once that is done, it shows a watch face that has a button to immediately call the emergency services. The time can also be switched between 24 hours and 12 hours format. On the next screen, you can get a reading of your heart rate. Once a fall is detected or the "Call Emergency!" button is pressed, a screen pops up to inform that the emergency services are being called with an option to cancel it on screen.

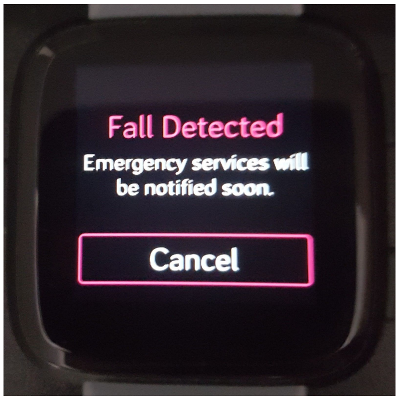


Waiting for connection to server.





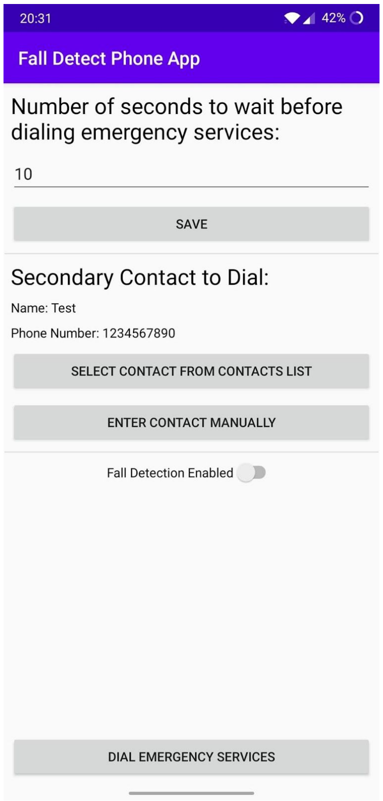




Emergency services will be notified soon.

The Android app on the other hand, is used to customize things like the number of seconds to wait before dialing the emergency services, the secondary contact, and to turn off fall detection in order to let the user sleep and do intensive activities like working out. This app also has an option to directly dial emergency services from the app.

In this setup, the android app works as the backend, being used for fall detection, and to call the emergency services and to message the secondary contact with the location of the user.



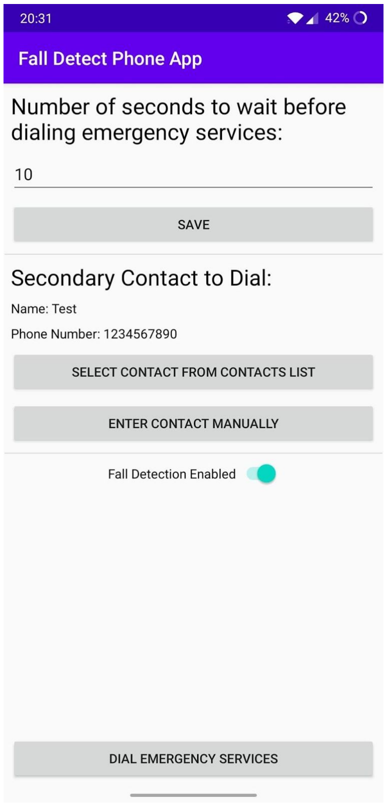
Number of seconds to wait before dialing emergency services:

10

Secondary Contact to Dial:

Name: Test

Phone Number: 1234567890



Number of seconds to wait before dialing emergency services:

10

Secondary Contact to Dial:

Name: Test

Phone Number: 1234567890

Health and Safety Concerns

The nature of these concerns do not correlate directly with the product (itself as an app cannot physically harm you) but with the products needed to run the app.

For the Fitbit (Taken from the Fitbit website) :

- *They may cause skin irritation so keep Fitbit clean, dry, and it should not be worn excessively tight*
- *If you feel soreness, tingling, numbness, burning or stiffness in your hands or wrists while or after wearing the product, please immediately discontinue useDo not attempt to replace the battery, open the enclosure or disassemble your Fitbit product. Doing so will void the warranty and can result in a safety hazard.*
- *Do not use your Fitbit product if the display is cracked.*
- *Substances in a Fitbit product and its battery may harm the environment or cause injury if handled and disposed of improperly.*
- *Do not expose your Fitbit product to extremely high or low temperatures.*

- *Do not use your Fitbit product in a sauna or steam room.*
- *Do not use abrasive cleaners to clean your Fitbit product.*
- *Remove your Fitbit product immediately if it feels warm or hot.*
- *Charge the battery in accordance with the instructions provided during setup. Only use a power supply that is certified by a recognized testing laboratory and an authorized Fitbit charging cable.*
- *Do not wear your Fitbit product while charging it.*
- *Do not charge your Fitbit product while it is wet*

Technical Issue and Solutions

- The Fitibit watch app doesn't connect to the phone app!
    - Make sure Bluetooth and location services are enabled on the smartphone.
    - Try restarting the Fitbit smartwatch app.
        - Use the left button on the Fitbit to exit the application, and tap the application icon to launch the app again.
    - Try relaunching the Fitbit smartphone app.
        - Go to the recent app menu (usually the rightmost button on the navigation bar), find the "Fitbit" app, and swipe it to remove it from the recent app menu. Then relaunch the "Fitbit" app.
    - Try relaunching the Android companion app.
        - Go to the recent apps menu (usually the rightmost button on the navigation bar, represented by a square), find the "Fall Detect Phone App" app, and swipe it to remove it from the recent app menu. Then relaunch the "Fall Detect Phone App" app.
    - Try restarting your smartphone.

Design Files

Due to the nature of how the project is implemented, it is split into three parts: the Fitbit smartwatch app, the Fitbit smartphone companion app, and a companion Android app.

- Fitbit smartwatch app & smartphone companion app (Javascript, CSS, SVG)
    - *app/index.js*: This file contains the code that runs on the smartwatch. It is responsible for driving the UI and sending commands and accelerometer data to the Android companion app for fall detection.
    - *common/constants.js*: This file contains constants that define a message's type.

  - *companion/index.js*: This file contains all the code that runs under the Fitbit smartphone app. The primary purpose of this code is to act as a bridge between the Android companion app and the Fitbit smartwatch.
  - *resources/index.gui*: This file defines the UI for the smartwatch app. It uses SVG.
  - *resources/widgets.gui*: This file defines external UI components that are imported to be used in the UI.
  - *resources/styles.css:* This file defines CSS rules that change how elements of the UI appear.
  - *resources/icon.png*: This is the icon of the application.
- Android companion app (Java)
  - *AndroidManifest.xml*: This file defines the properties, permissions, and the activities of our application.
  - *build.gradle:* This file defines how the app gets compiled and built, in addition to the libraries used.
  - *java/FallDetection.java*: This file contains the fall detection algorithm.
  - *java/MainActivity.java*: This file contains the code to drive the UI of the application. It also contains the code to start the parts of the application responsible for handling input from the smartwatch.
  - *java/Message.java*: This file defines the Message class, which represents a message from the Fitbit smartwatch.
  - *java/MonitorService*: This file contains the code that forms the server that listens for input from the smartwatch and responds accordingly. It is also responsible for driving the fall detection algorithm and processing the output from it.
  - *java/Settings.java*: This file defines the Settings class, which holds the settings for this application.
  - *java/Telephone.java*: This file is a helper class for accessing the telephony functions of the smartphone.
  - *res/drawable/**: Contains graphics resources used by other parts of the application.
  - *res/layout/**: Contains the UI layouts and elements used by different components of the application.
  - *res/mipmap/**: Contains the app’s icon.
  - *res/values/**: Contains other miscellaneous values used by the app.

Conclusion

Over the course of the project our group learned some valuable lessons such as the importance of how project management will lead to better results. It is also important to note that a group should always work to everyone’s strengths. Daniel and I were not the best with regards to coding so we focused on management while Punit and Walter worried about the coding side of it. Part of working on a project also means you have to take in some form of criticism, when we started working on this project we were not working to the best of our abilities. Due to that fact we received constructive criticism from our professor and project managers. At the end of the day, that critism helped and made our group work better and harder. Lastly, we learned how crucial time management really is. With good time management, we accomplish more in a shorter period of time, which leads to more free time for everyone in our group.

Additionally, our group wanted to configure the heart rate monitor of the Fitbit in such a way where it would let the user of the fitbit know if their heart rate was high. However, since this was not a customer need we did not put time into it since we only had a finite amount of time for this project.

For future work, our group would recommend that all group members learn the coding languages that would be used at a basic level. That way more people could work on the prototype resulting in the prototype being completed quicker. The earlier the prototype is done the more you as a group can test it and see where more work needs to get done.

Bibliography

Fall Detection Algorithm:
https://github.com/alessandror/AndroidFallDetectionApp/blob/master/FallDetection/app/src/main/java/com/alexr/falldetection/FallDetectAlgo.java

Man Falling Icon:
https://www.flaticon.com/free-icon/falling-man_11015

Fitbit Health and Safety Concerns:
https://www.fitbit.com/us/legal/safety-instructions

Libraries Used

GSON (Apache-2.0):
https://github.com/google/gson
Java-Websockets (MIT):
https://github.com/TooTallNate/Java-WebSocket

SmartLocation (MIT):
https://github.com/mrmans0n/smart-location-lib