Wearable Computing Devices for the Visually Impaired

Shakil Bin Karim (13201068)

Department of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

Nayyar Mustafa (12221056)

Department of Electrical and Electronic Engineering

BRAC University

Dhaka, Bangladesh

Supervisor:

Amitabha Chakrabarty, Ph.D

Assistant Professor

Department of Computer Science and Engineering

BRAC University

Dhaka, Bangladesh

Submitted on: 10.8.2016

Declaration

We, hereby declare that this thesis is based on results we have found ourselves. Materials of work from researches conducted by others are mentioned in the reference. This thesis, neither in whole nor in part, has been previously submitted for any other degree or any other publication.

Date: 10.8.2016

Signature of Supervisor

_____

Amitabha Chakrabarty, Ph.D
Assistant Professor
Department of Computer Science and Engineering
BRAC University

Signature of Authors

_____

Shakil Bin Karim (13201068)

_____

Nayyar Mustafa (12221056)

Abstract

This thesis is about creating a prototype of wearable computing devices that possess the ability to aid the mobility of the visually impaired. The devices are a belt and a pair of shoes which are connected via Bluetooth to a smartphone running an application built specifically for these devices. The belt consists of ultrasonic and infrared sensors for obstacle and hole detection from waist level, and coin vibration motors for providing haptic feedback. When in a difficult situation the user can send for help at the press of a switch on the belt which will make use of its Bluetooth connectivity to the smartphone to transmit the signal to dear ones in the form of a phone call and a push notification. The GPS of the phone will enable the location of the sender to be sent as well along with the help signal in the form of a push notification. The shoes, like the belt, contains ultrasonic sensors for obstacle detection and coin vibration motors for the haptic feedback. But it also comprises of a pulse rate sensor that can measure pulse rate of the user (as taken from the dorsalis pedis artery underneath the top mid surface of the foot) and via Bluetooth connection with the smartphone have it transmitted live to dear ones. If the pulse rate becomes abnormal the application will be able to send help signals to dear ones even if the user did not have the opportunity to press for help.

Acknowledgement

This work was suggested to us by Dr. Amitabha Chakrabarty (Assistant Professor), Department of Computer Science and Engineering, BRAC University, Dhaka, as a Graduation thesis after he listened to what we wanted to work with. This is the work of Shakil Bin Karim and Nayyar Mustafa, student of the SECS, BRAC University.

We would like to express our gratitude to Almighty ALLAH (SWT) who gave us the opportunity, determination, strength and intelligence to complete this thesis.

A very big thank you, goes to our supervisor Dr. Amitabha Chakrabarty who has constantly believed in us and has been there for us through the thicks and thins of the thesis and continuously pushed us to complete our work in time. We are extremely fortunate and grateful to be able work under his supervision.

Our family members have been a great support throughout the thesis work and so we would like to thank them as well. Without their cooperation this thesis would not have been complete.

Lastly, our gratitude goes towards the faculty members of Department of Computer Science and Engineering & Department of Electrical and Electronics Engineering, BRAC University from whom we gained all the knowledge that aided us to successfully complete our thesis work.

Contents

List of Figures

Chapter 1 - Introduction

There are an estimated 285 million visually impaired people worldwide, of whom 39 million are blind, i.e. have complete loss of vision and 246 million have low vision according to a report by World Health Organization in August 2014 [1]. Given such a figure, the need for assistive technology for this population is a critical need. Furthermore, WHO reports that about 90% of the visually impaired population is in developing countries. For instance in Bangladesh, where this project is carried out, there are around 800,000 blind people, of whom 40,000 are children below the age of fifteen [2]. In this developing country 75% of the population live in remote areas with few basic facilities, therefore blind or semi blind residents of these areas face greater challenges in accomplishing day to tasks. Owing to the low income status of these regions, such a significant portion of the visually impaired population does not have access to technology that could otherwise aid in mobility and increase their productivity.

For decades, white canes and trained dogs have been common assistive technologies for the visually impaired. In recent times, there have been many attempts to make the white cane more functional with added transducers and lasers that detect surrounding obstacles but these technologies still lack in aiding better movement. Maintaining trained dogs is itself quite costly and at most they can be of service for 10 years at a time [8]. Human assistance for visually impaired persons also takes additional resources. Our work aims to extend the research already done in this field, and improve technologies available for visually impaired people.

We wanted to provide an affordable, dependable and easy to learn convenient solution using wearable computing devices paired with a smartphone and the Internet and the Cloud to aid the navigation of blind people. While the solution we wanted to provide can work absolutely alone, we also wanted it to be able to work alongside the current solutions like the white cane to make the adoption of the newer solution easier and also so that the visually impaired people can navigate with enhanced perception. In short we wanted the newer solution to be able to make the lives of blind people easier without it getting in the way of what they are already used to. In our thesis work, we have used technologies like android, Arduino, Azure Cloud services, etc. to achieve just that.

1.1 Objective

- Use data provided by sensors in wearables to warn the visually impaired person about obstacles
- Connect multiple wearables to an android phone via Bluetooth so that the phone can read vital data regarding the blind person
- Constantly monitor the blind person's heartrate
- Make two android apps, one for the blind person and the other for the blind person's guardian
- Connect both apps to a common cloud service, which would enable the blind person's guardian to constantly monitor the blind person if he is commuting alone and also would enable notifications to be sent to the guardian if the blind person's heart rate increases too much or if the blind person explicitly asks for the guardian's help.
- Enable blind person to call a specified person with just the push of a button in case of emergencies.
- Implement solution in such a way that it does not disrupt the blind person's normal life.
- Make solution affordable.

1.2 Motivation

Our main motivation throughout this project was the desire to build something that helped the visually impaired population of this country by incorporating modern technology into their lives. We also wanted to implement such a solution as cheaply as possible in order to make it affordable for the general public. The idea of a smart shoe sparked in our minds after we saw the recent rise in the popularity of wearable computing devices. That in conjunction with the increasing availability of Internet and cheap smart phones, convinced us that we could in fact, combine smart phones, Internet and wearable computing devices to build a system that can help blind people navigate in a reliable, affordable and effective way without obstructing their current lifestyle. At the same time we wanted this system to have features that would enable the blind person's dear ones to monitor him/her remotely.

1.3 Scope of the Thesis

The target audience of this thesis are blind or otherwise visually impaired population of this country. We have designed the system so that it can work in harmony with the current tools used for navigation by the visually impaired community (tools like trained dogs and canes). Our device can also work beside other devices built to aid the blind that are currently available in the local and international market. This should make our system easy to adopt and adapt to. We also focused on making our devices cheap and easily affordable to appeal to a wide customer base.

1.4 Features of Our System

- Uses data provided by sensors in wearables to warn the visually impaired person about obstacles
- There are three wearable devices
- 2 shoes and a belt
- The shoes use sensors to detect obstacles on the ground level and notify the wearer using vibrations in the respective direction. It also reads the wearer's heart rate.
- The belt has sensors that detect holes in front of the wearer and waist height obstacles and notify the wearer accordingly. It also has a button that is used for sending out a call to a specified phone number and a notification to the Guardian.
- There are two android apps, Guardian and Dependent.
- The Dependent App reads data from the shoe and belt and uploads it to Azure.
- It also gets the users location and uploads the data to Azure.
- The Dependent App also connects multiple wearables to the android phone via Bluetooth so that the phone can read vital data regarding the blind person
- The Guardian App enables guardians to monitor multiple dependents.
- The user can locate any of his/her dependents by pressing the “Showmap” button in the Guardian App.
- Both Apps connect to Azure Cloud Service, which enables the blind person’s guardian to constantly monitor the blind person if he is commuting alone and also enables notifications to be sent to the guardian if the blind person’s heart rate increases too much or if the blind person explicitly asks for the guardian’s help by pushing the button on the belt.

- The blind person can instantly make a call to a number by pushing the button on the belt.
- Implement can easily be used beside existing solutions for blind people in Bangladesh.
- The solution is relatively cheap.

1.5 Thesis Outline

Chapter 1 is the formal introduction of the thesis. We have discussed our motivation, scope, features and objectives.

Chapter 2 is the background study that covers the literature review and all the research work we have done.

Chapter 3 is where we discuss about the design of our solution. First, we talk about the overall design of the project, later we further discuss the designs of the hardware and software aspects of the project and finally we talk about cloud services.

Chapter 4 is where we discussed about the hardware and software specifications of our prototype.

Chapter 5 is an account of the implementation of our prototype according to the designs in Chapter 3.

Chapter 6 is where we discuss about the results of tests done on our implemented prototype and some analysis of the results.

Chapter 7 is the conclusion.

Chapter 8 is a discussion about the future aspects of our thesis project.

In the References section we have cited the references that we have made use of.

Chapter 2 – Literature Review

Research on using technology to aid mobility of the visually impaired has been ongoing for more than four decades. This research has produced many different types of electronic travel aids that make use of sensors for obstacle detection and employ signal processing techniques such as image processing via a camera implanted in the wearer's clothing. The aim of these devices is to scan the surrounding environment and familiarize the user with it as per their design.

Yuan et al. have developed a proof of concept prototype of a white cane that contains a laser range sensing device [4]. The cane can be pointed like a flash light and moved around to get 15 measurements per second of the space in front of the user. The cane is also capable of detecting surface discontinuities such as a step, drop-off or obstacles that protrude from the ground up to a certain height.

Abu-Faraj et al. have developed an ETA (Electronic Travel Aid) prototype comprising of a pair of glasses and shoes [3]. The glasses have ultrasonic transducers installed above the nose bridge to detect obstacles at head level and a buzzer to one side of the glasses to warn the user when any obstacle is detected. The shoes also have ultrasonic sensors that detect obstacles of different heights at ground level on three sides of the user. Vibration motors are installed at the collar of the shoes to warn the user. The shoes and glasses are controlled by a belt pack unit [3].

Cardin et al. developed a jacket fitted with ultrasonic sensors at shoulder level with a field view of about 60° for detecting obstacles in the user's path. The vibration motors fitted on the jacket to warn users about obstacles [5].

Borenstein in 1990 discussed a navigation system for the blind in the form of the NavBelt which consists of a wearable belt consisting of an array of ultrasonic sensors [6]. The sensors scan the environment ahead of the user, covering about a 120° field and gives feedback via earphones in the form of different frequencies of acoustic stereo signals depending on the distance of detected obstacles in front of the user.

In 2013 Al-Fahoum et al. discussed their design of a microcontroller based navigation system for the blind. The system comprises of a hat and a mini hand stick fitted with IR sensors to detect obstacles up front [8]. For the feedback system is auditory with the left and right speakers indicating the direction of the obstacle by producing a sound of certain frequency. Furthermore a vibration motor works with the speakers to guide the user [8].

Baranski et al. proposed a more interactive approach to an ETA for the blind by using a GSM and GPS module in the blind user's wearable pack. A camera fitted to this pack feeds a remote user with live video from the surroundings of the blind user via the GSM module of the mobile phone carried by the user. Furthermore, the blind user is able to communicate by this module with the remote user who can guide him to his destination based on the location coordinates sent by the GPS module and live video feed [7].

Given the above research done in this field, we identified several areas where our prototype would improve upon earlier technology for visually impaired people. For instance some of the wearable devices mentioned above such as jackets, glasses and hats are able to detect obstacles ahead but are not capable of identifying drop-offs or holes. We took this into consideration and tried to incorporate this feature in our prototype. There have also been many works on digitizing the cane, but we wanted to be able to ease the mobility of the user without having him carry any cane or extra weight.

In some of the aforementioned works, we have seen that the actuating signals used were different frequencies of sound, which meant that to use these devices for navigation the user requires considerable training to distinguish the different sounds that he would have to hear depending on the terrain ahead. For a blind user with auditory difficulties, using these devices would be even more challenging. To make it easier for such users we designed our project to utilize haptic feedback as actuating signals. As for the communication aspect of the device, we have observed that Baranski's use of live video feed is not reliable in all places, as the video communication requires uninterrupted network service in places traversed by the blind user, which is not yet possible with present communication infrastructure especially in the context of Bangladesh where this project is being carried out. So with existing infrastructure the video feeds could be delayed and blurry, as a result of which the guide at the control center will not be able to aid the user's navigation properly. Coming to the guide, we incorporated a push notification and call –at-will system so that the user is assured of safe travel but at the same time not have to worry that someone is always having to keep an eye on his movement, thereby also building on his self-confidence.

Chapter 3 – System Design

In this chapter we discuss about the design of the solution we came up with. We shall describe the design of the hardware and software aspects of the project. We will also justify certain design choices. This chapter will include schematics for hardware, complete user interface designs, pseudo codes of client side and server side codes, activity diagrams, ER diagram for the project, etc.

We have divided this chapter into four sections. They are as follows:

- Over all System
- Hardware of the System
- Software of the System
- Cloud Services

3.1 Overall System

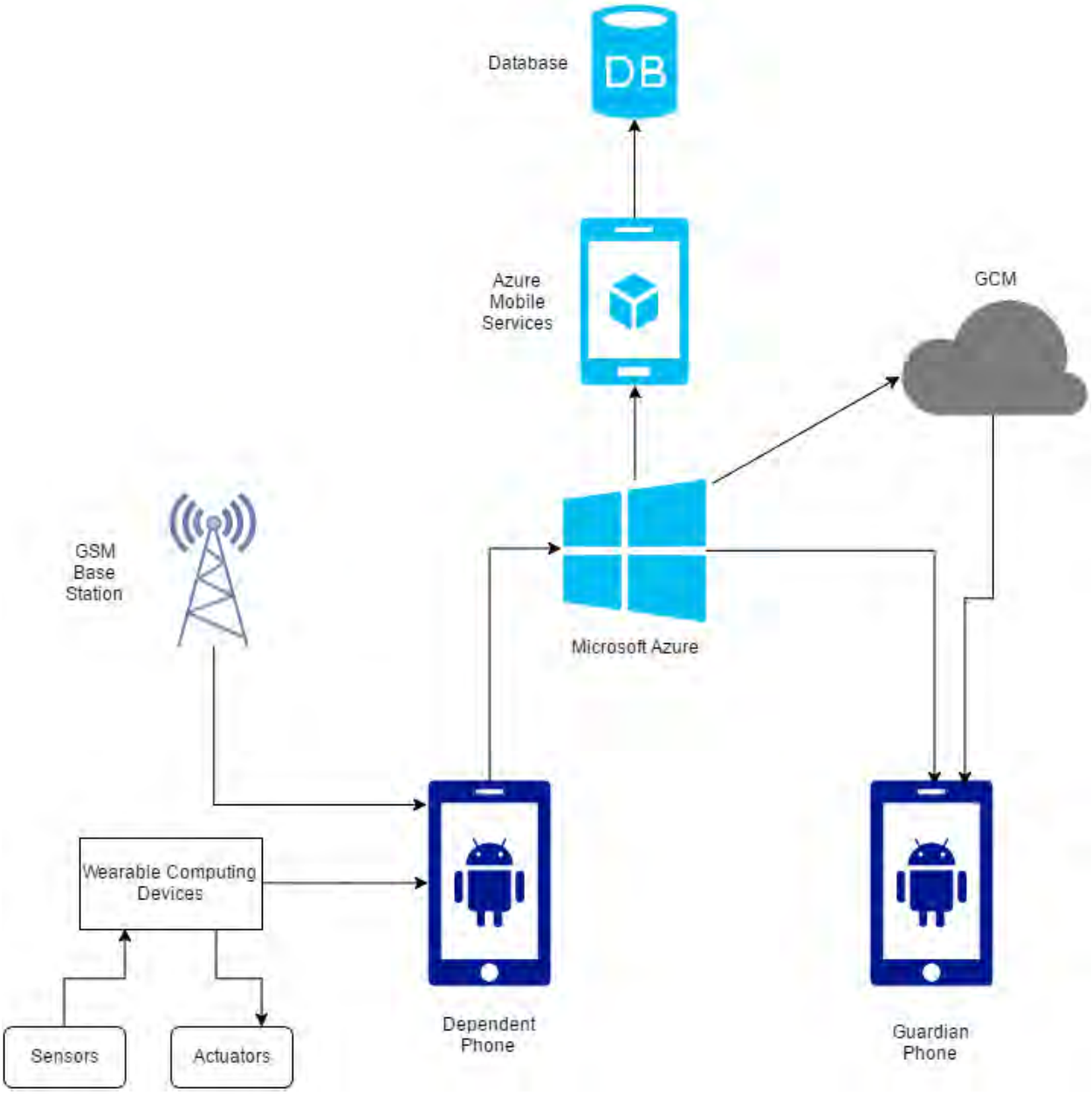


Figure 1 Diagram showing how the overall system works

3.2 Hardware of the System

The hardware in the system involves three components. Two shoes and one belt. Since the hardware is in prototype phase we decided to use Arduino to work as the brains of all the individual hardware components. Arduino is an ideal prototyping platform since it is open source and very easy to learn. Over that it is cross-platform and inexpensive. We could have used Raspberry Pi instead, but for our particular project we felt the Raspberry Pi would be an overkill and it might add unnecessary complexity to the project.

In the following subsections we will discuss the individual hardware components in terms of block diagrams and brief descriptions.

The Belt

The belt consists of an Arduino Uno as the microcontroller, two Sonar sensors to detect obstacle at waist level, an IR Sensor to detect holes in the ground, three vibration motors to make the blind person aware of obstacles, a Bluetooth module to communicate with a smart phone, a two pin push button to signal an emergency to the smart phone and finally a power source which in the case of our prototype are standard 9V battery packs.

When the IR sensor detects a hole or drop off, the back vibration motor vibrates at a frequency of about 5Hz to warn the user. When the left ultrasonic sensor detects an obstacle within its field view at a minimum distance of 10cm ahead of the user then the left vibration motor vibrates and when the right ultrasonic sensor senses an obstacle at a minimum distance of 10cm ahead of the user then the right vibration motor vibrates. If the obstacle spans the front side of the user, as covered by both ultrasonic sensors then all three vibration motors warn the user. The frequency is set at 5Hz for all vibration motors. The two pin push button will be used to send signal via Bluetooth to the smartphone by user to signal an emergency.

The block diagram for the belt is given on the next page.

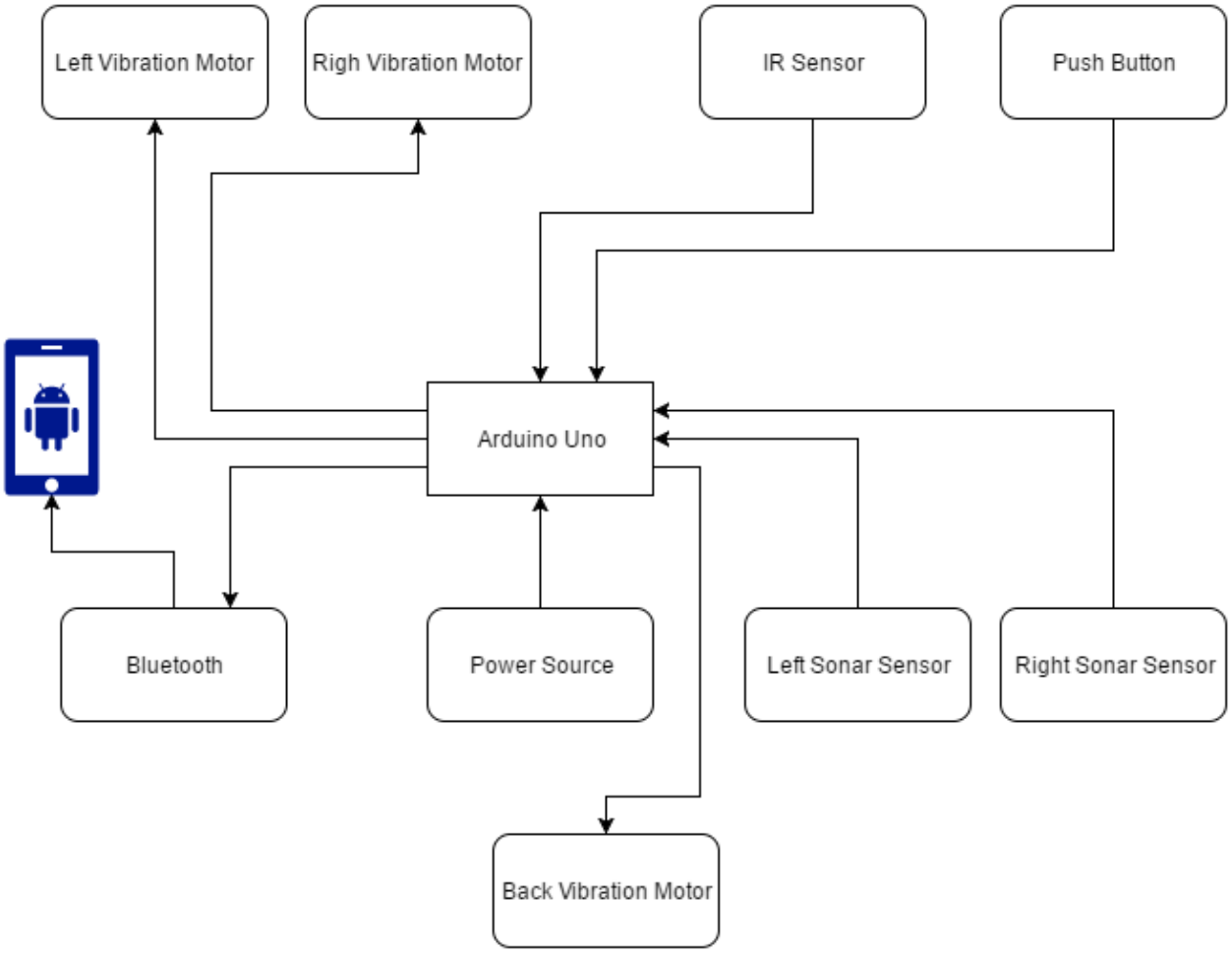


Figure 2 Block Diagram for the Belt

The Shoes

Both shoes consists of an Arduino Uno as the microcontroller, two Sonar sensors to detect obstacle at feet level, two vibration motors to make the blind person aware of obstacles and a power source which in the case of our prototype are standard 9V battery packs.

Both the left shoe and the right shoe is fitted with two ultrasonic sensors. These sensors detect obstacles at the feet level with vibration motors placed under the shoe soles. The coin vibration motors at the front part of the sole of both shoes will vibrate when the ultrasonic sensors in the front detect obstacles at a distance of 10cm or less from the shoes, thereby warning the user. Similarly the side vibration motors will vibrate when the ultrasonic sensors at the sides detect obstacles at a distance of 10 cm or less.

Beneath the upper padded flap of the right shoe is the pulse rate sensor which takes about 12 pulse rate readings per minute. The pulse sensor is placed in such a way that it is in contact with the top surface of the wearer's foot under which the dorsalis pedis artery is located. It is the contraction and dilation of this artery from which the sensor gets its readings. The right shoe is also equipped with a Bluetooth module which enables it to send data collected from the pulse rate sensor to a smartphone.

Block diagrams of the Shoes are given on the next page.

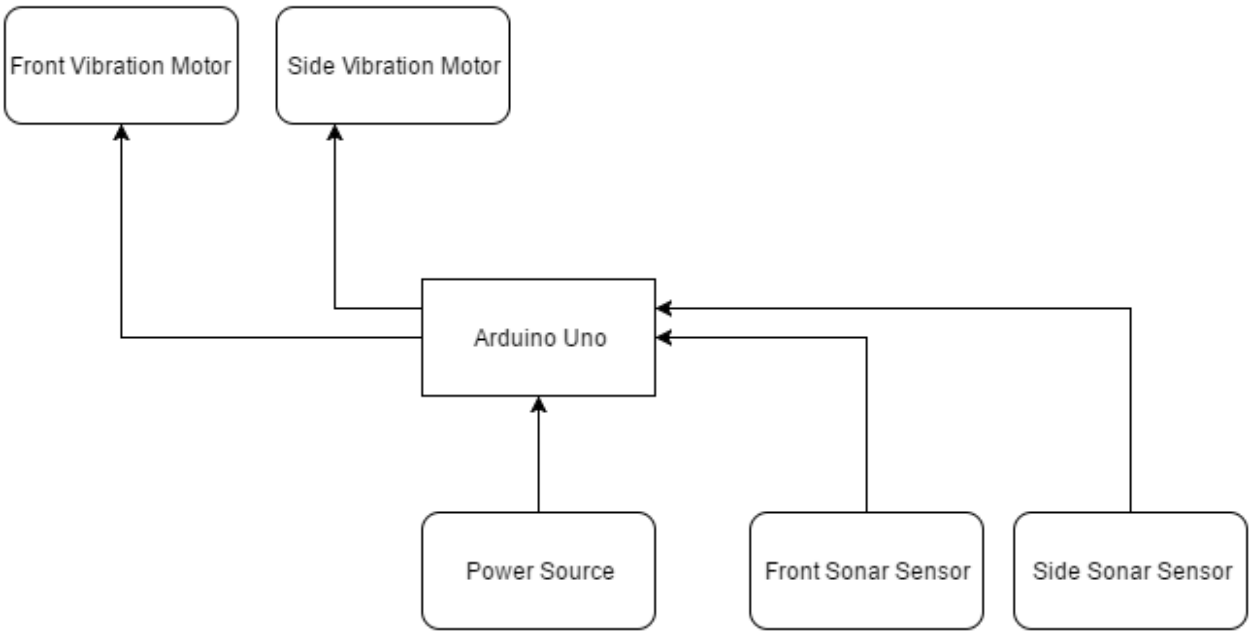


Figure 3 Both the Right and Left Shoes have the above in common

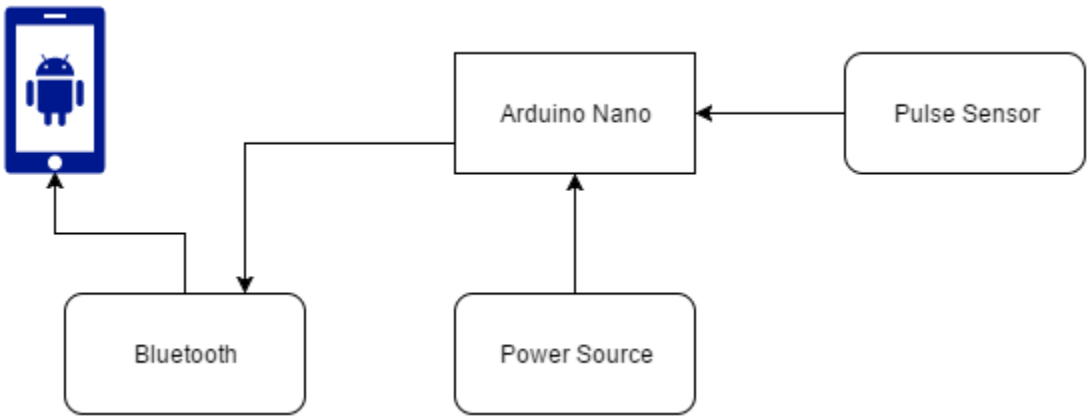


Figure 4 Block diagram for extra pulse sensor and Bluetooth capabilities of the right shoe

3.3 Software of the System

For the software side of the project, we decided to make two android apps. One app would be for the blind person and the other app will be for his/her guardian. The details of each app shall be discussed in the following subsections. We chose to make android apps as android is the most popular mobile OS currently and its open source nature makes it an ideal platform for tinkering developers. The market share for phones and tablet as of June 2016 according to the website https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1, is given on the next page. Over that development tools for android are cross platform and free, so anyone on any platform be it Windows, Linux or Mac, can develop for android. Moreover, android phones can be found in a wide range of prices for customers of almost all backgrounds.

For the cloud services used in our project, we chose Microsoft Azure. The reason for this being it is fully supported in Bangladesh. Azure has very good cross platform support i.e. it supports the web, android, iOS, windows, kindle and even more. Also, big companies like EA and Respawn Entertainment are running their AAA games like Titanfall completely on Microsoft Azure and it is performing very well. So we can be quite certain of Azure's reliability.

In the following subsections we shall discus about the design of the two apps in more details.

Figure 5 Mobile market share as of June 2016

Operating System	**Total Market Share**
Android	65.58%
iOS	27.24%
Windows Phone	3.26%
Java ME	1.81%
Symbian	1.08%
BlackBerry	0.97%
Samsung (Feature Phones)	0.03%
Kindle	0.01%
LG (Feature Phones)	0.01%
Bada	0.01%

The Dependent App

The dependent app is developed for the blind person. The main function of the dependent app is to constantly collect data regarding the patient from the wearables via Bluetooth and uploading them onto the cloud. It does this via running a background sticky service constantly so that the app keeps on doing its work even if the user turns the app off. If the OS closes down the service run by the app for any reason, the service is automatically restarted. This ensures the app is constantly monitoring the visually impaired person.

The app constantly gets the blind person's heart rate from the shoe. It constantly checks for a signal from the belt for the emergency button being pressed. The app gets the location of the blind person constantly from the GPS in the blind person's phone.

The app also enables authentication via Google OAuth. The Guardian can authenticate the blind person from his/her account when the blind person and his/her guardian first starts using the services provided by our project. *Using Azure table permissions we have made sure that only authenticated users can gain access to read/delete/insert/update data from the table. This ensures that the user data is secure and that random request to read their data or change or manipulate their data will be rejected.* After signing in once the user does not need to sign in every time they use the app as the app stores the sign in credentials of the user in the shared preferences. The credentials are loaded from the shared preferences every time the user later wants to use the app for authentication. This is similar to the Gmail app, Facebook app, and other popular apps that require authentication. Since we are using Google OAuth, our app does not directly deal with passwords and user ids. This makes our app extremely secured. It also makes things convenient for the user since they do not need to make a separate account to use our services. They need to make a Google account anyway to use Google play services and download our app or any app as

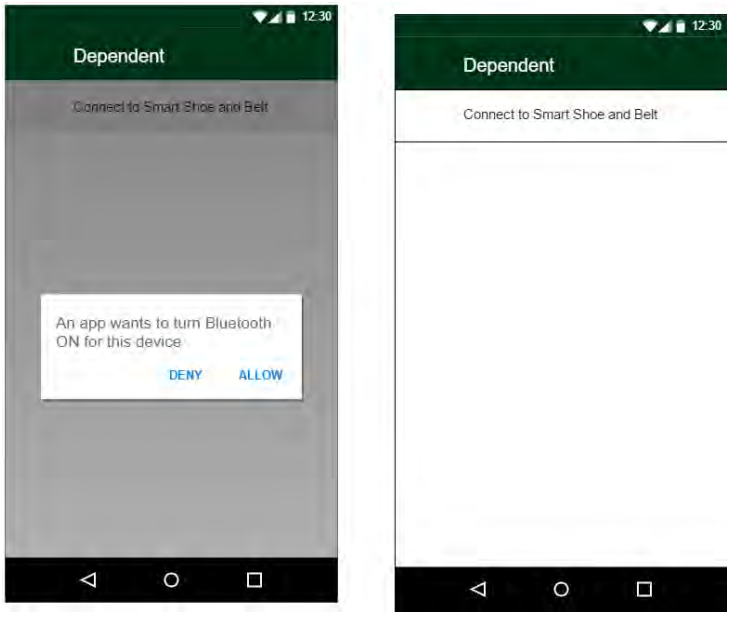
a matter of fact from the play store. The user can use their google account to authenticate themselves in our app.

The dependent app also allows the blind person to make phone calls to a person by just the push of a single button on the belt in case of emergencies. The number the blind person calls to can be set by the blind person's guardian in the app.

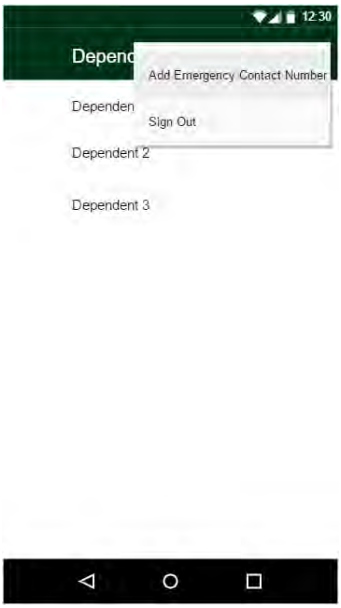
The layouts for the app and Activity Diagrams for the app are provided in the following pages.

App Layout Design for Dependent App

Log in Screen



Permission for Bluetooth and Connecting to Wearables Screens



Select Dependent Screen

Service Running in the background Screen

Figure 6 Layout Designs for Dependent App

Activity Diagrams and Block Diagrams for Dependent App

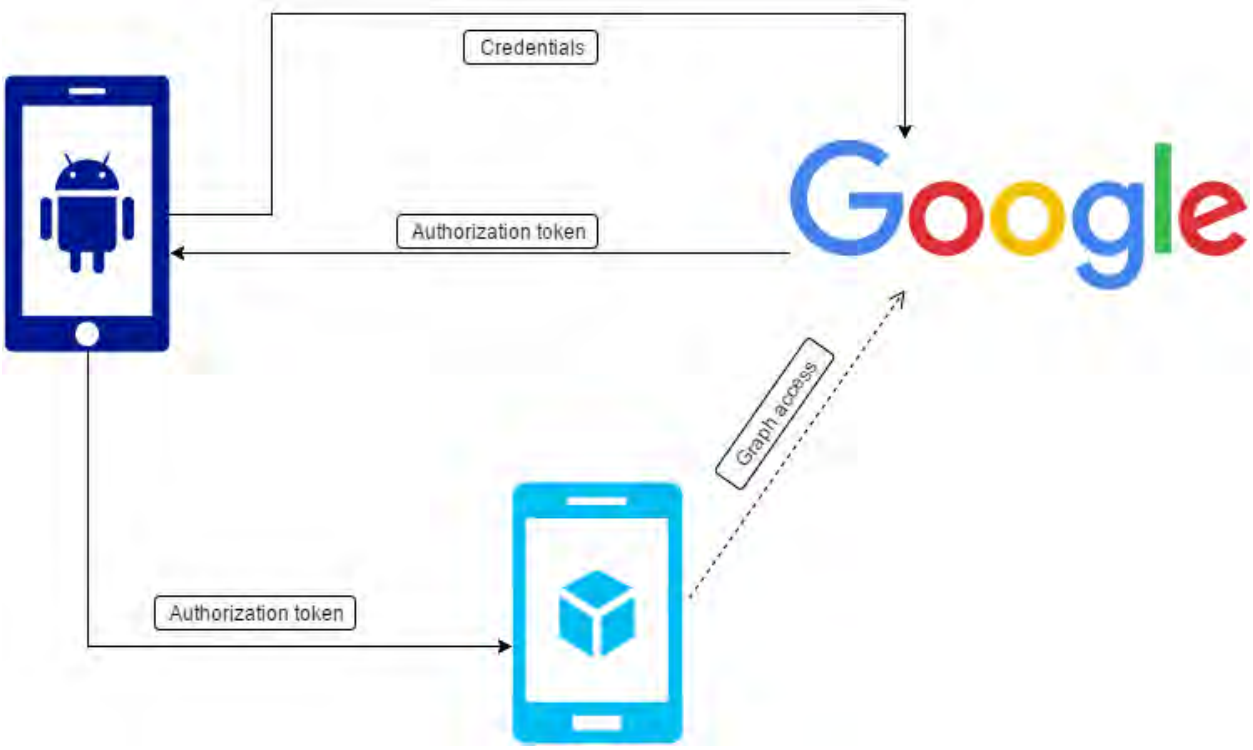


Figure 7 Signing in

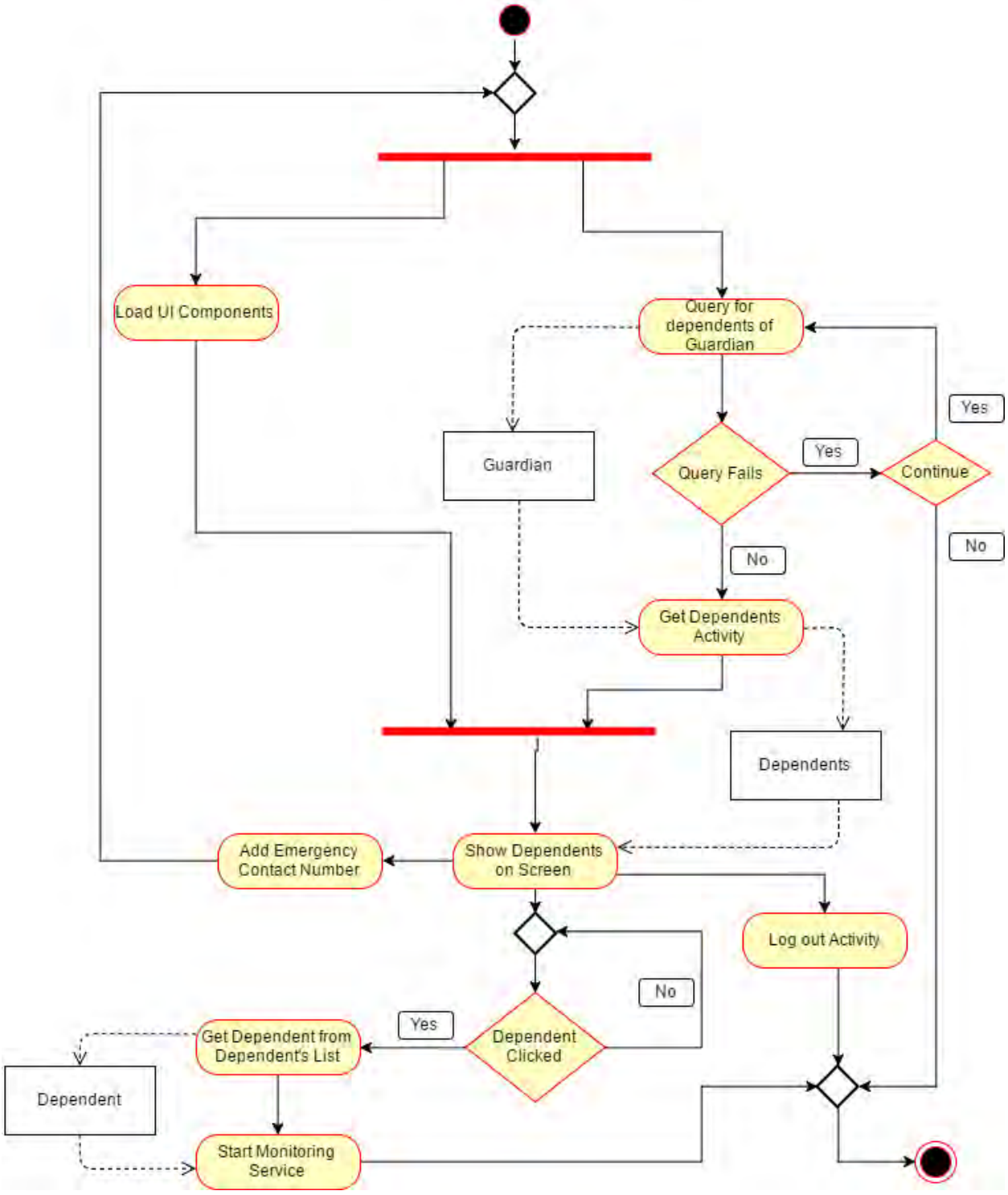


Figure 8 Dependent Dashboard Activity

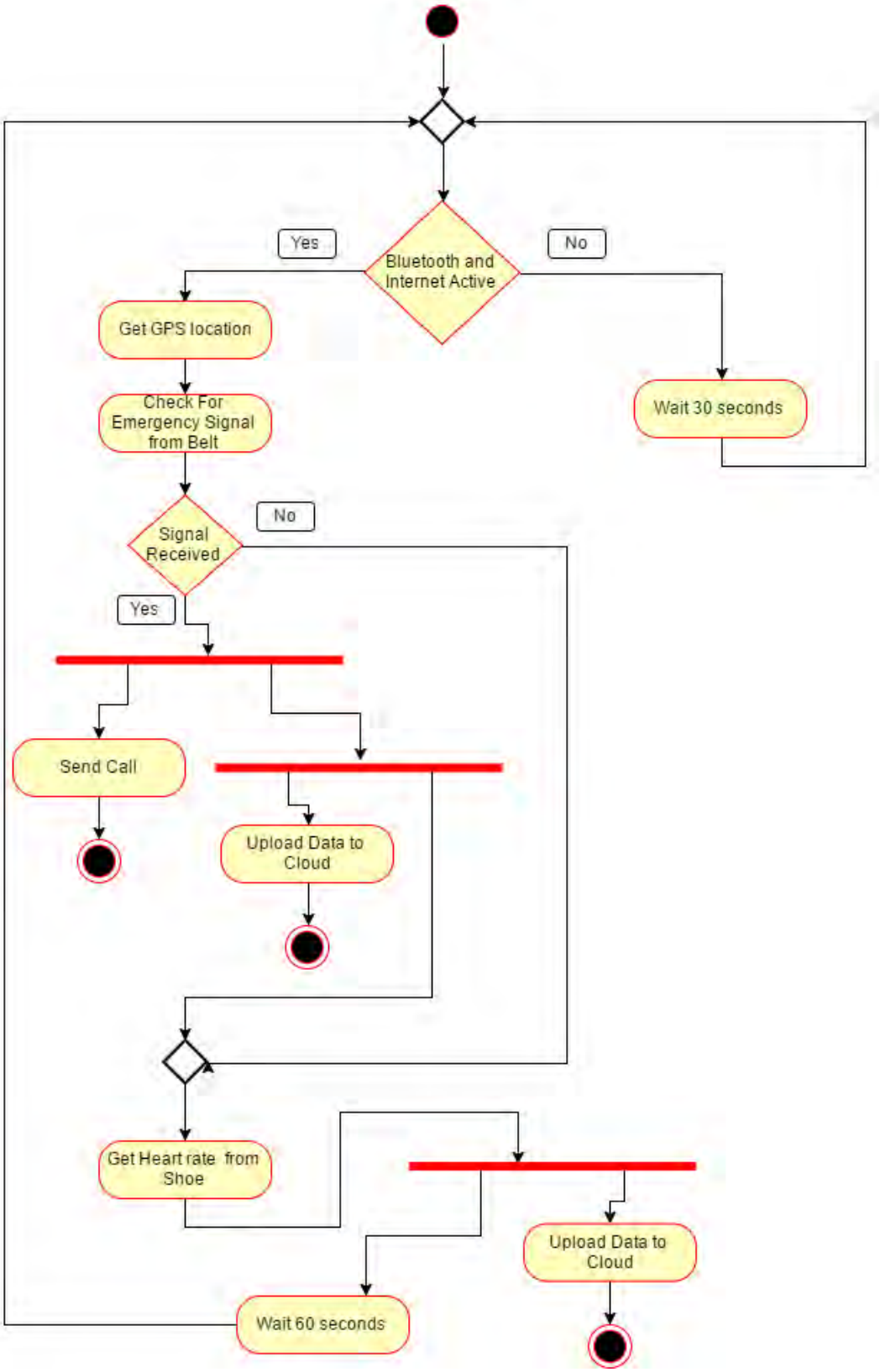


Figure 9 Monitoring Service Activity

Guardian App

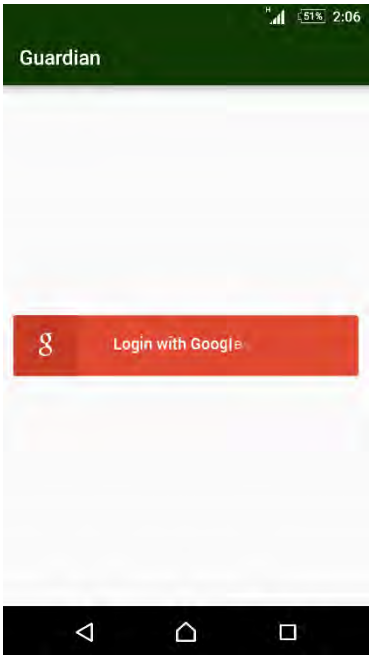
The guardian app would let guardians of blind people to monitor them. It would enable the guardian to add dependents, edit dependents' names later on, delete dependents, and monitor their location and heart rate. The guardian can view the location of their dependents on a map from the app.

Authentication is done by Google OAuth similar to the dependent app. The access to the user table is also limited to authenticated users only, just like the dependent app.

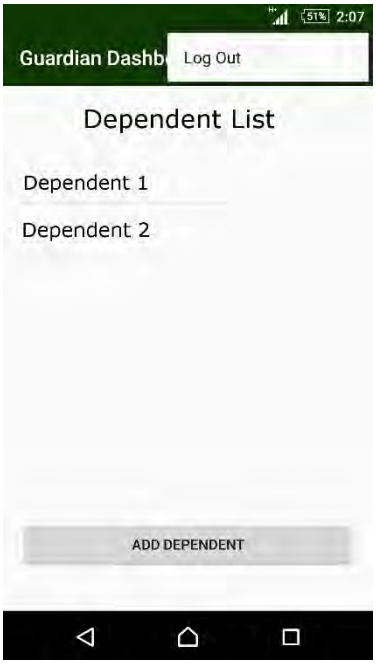
The app enables the guardian to receive notifications if the heart rate of the blind person becomes too high or too low. Clicking on the notification would turn on the guardian app if it is closed and take the guardian to the map activity and show the current location of the dependent along with his/her heartrate.

The layouts for the app and Activity Diagrams for the app are provided in the following subsections.

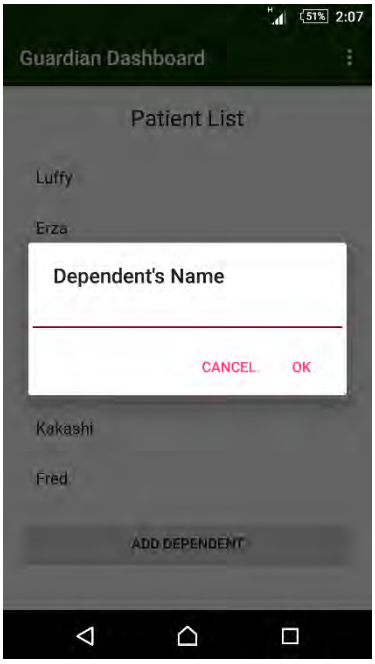
App Layout Design for Guardian App



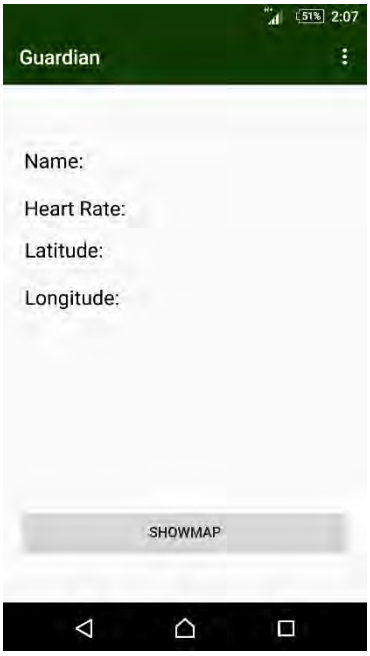
Login Screen



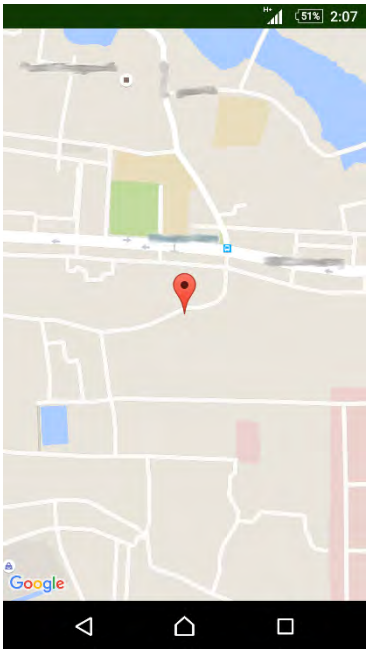
Guardian Dashboard



Add Dependent



Dependent Info Screen



Dependent Location on Map

Figure 10 Layout Designs for Guardian App

Activity Diagrams and Block Diagrams for Guardian App

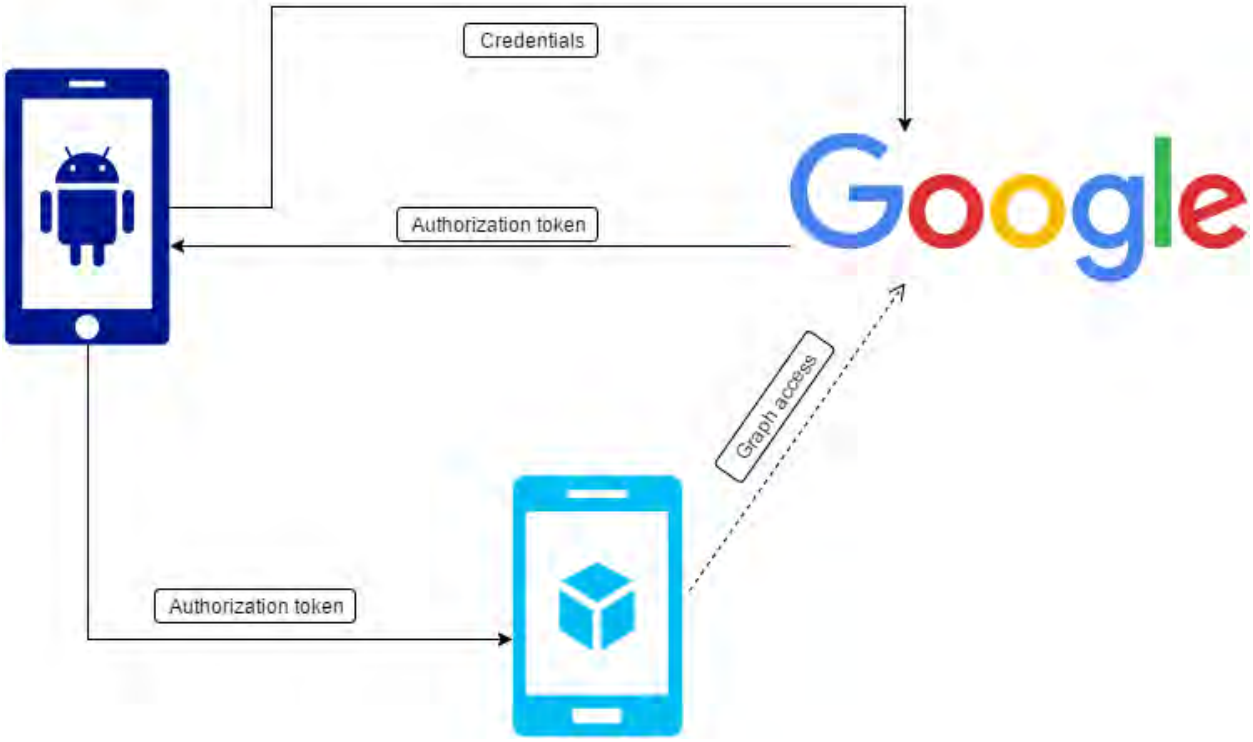


Figure 11 Guardian Login

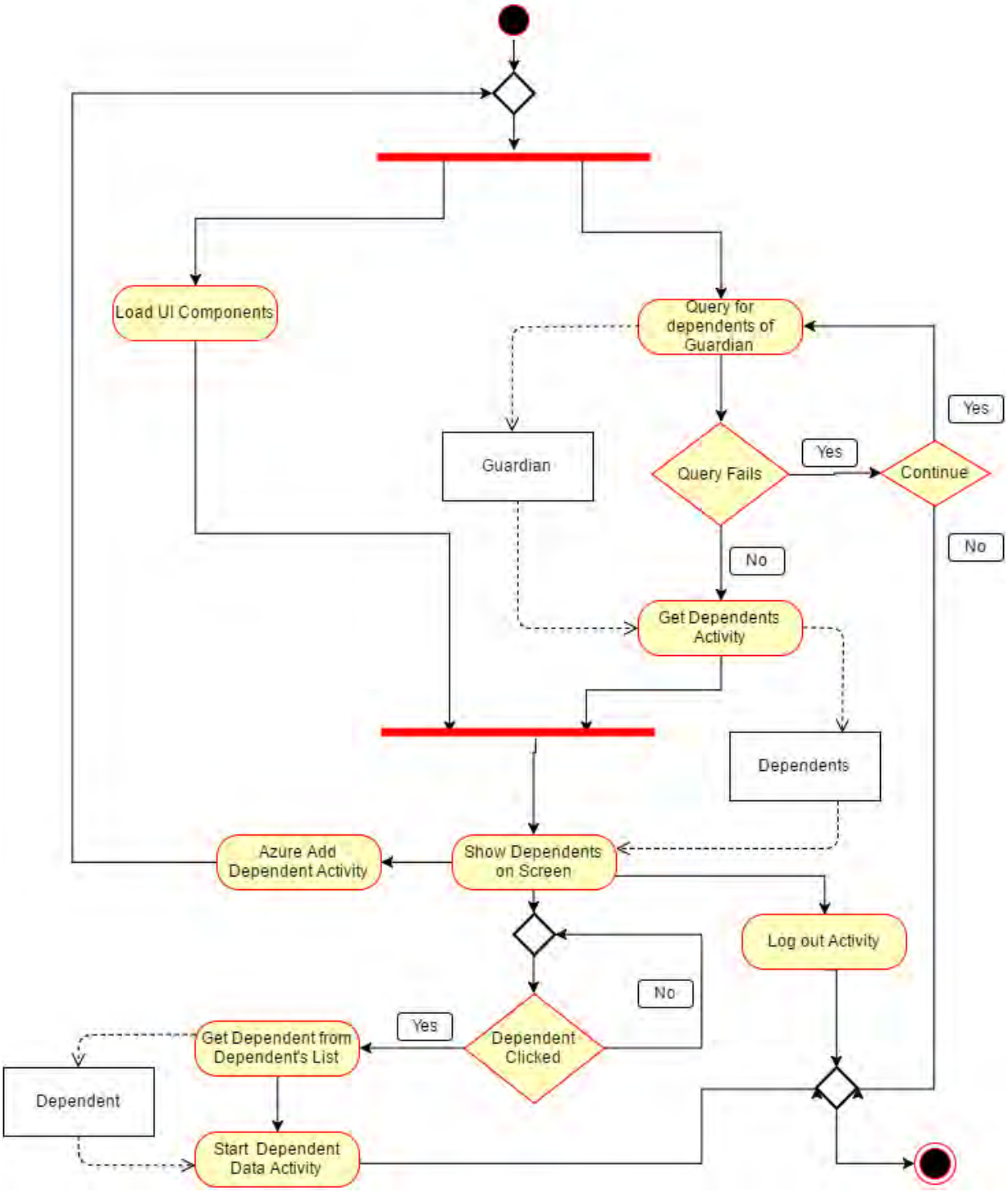


Figure 12 Guardian Dashboard Activity

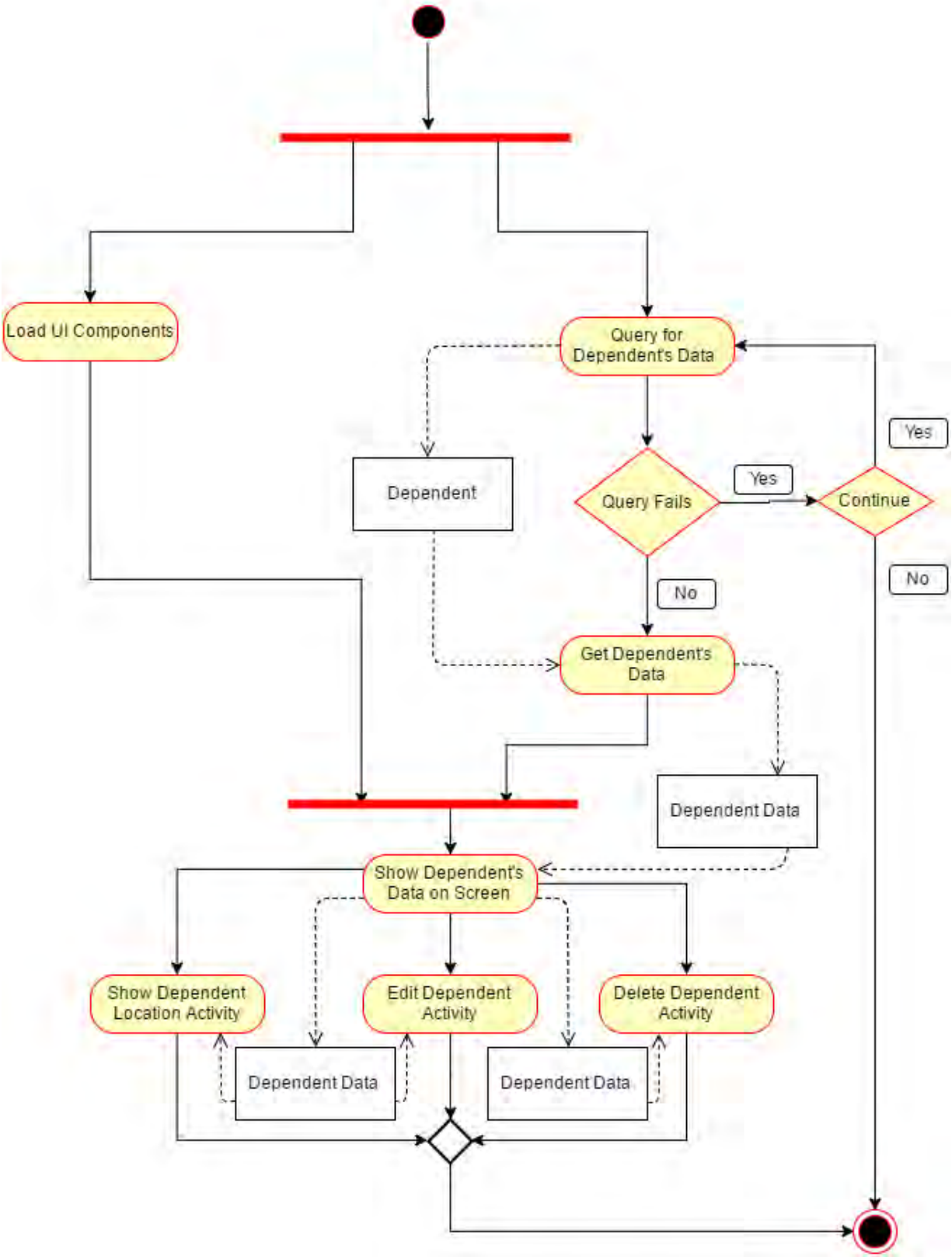


Figure 13 Dependent Info Activity

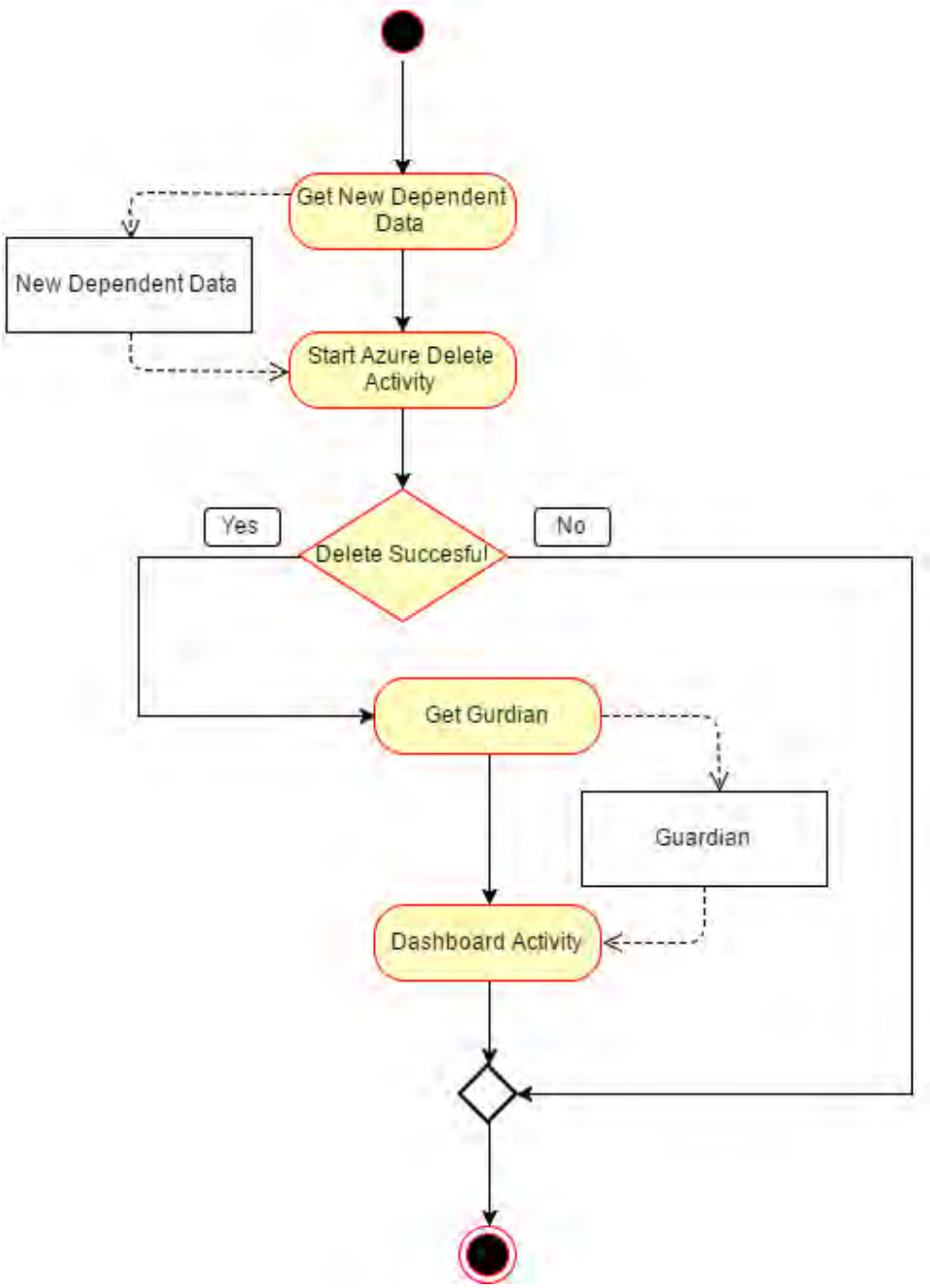


Figure 14 Delete Dependent Activity

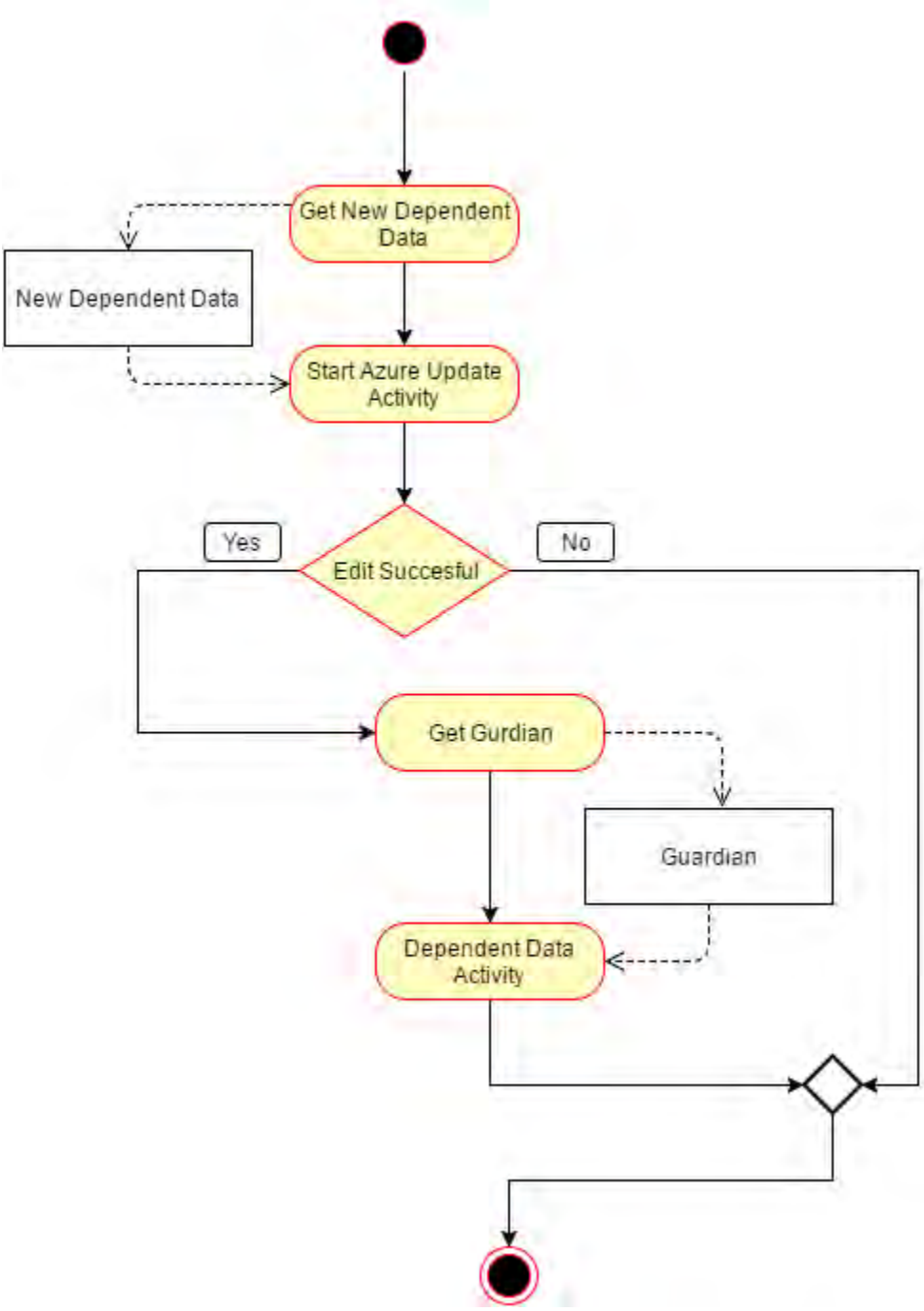


Figure 15 Edit Dependent Activity

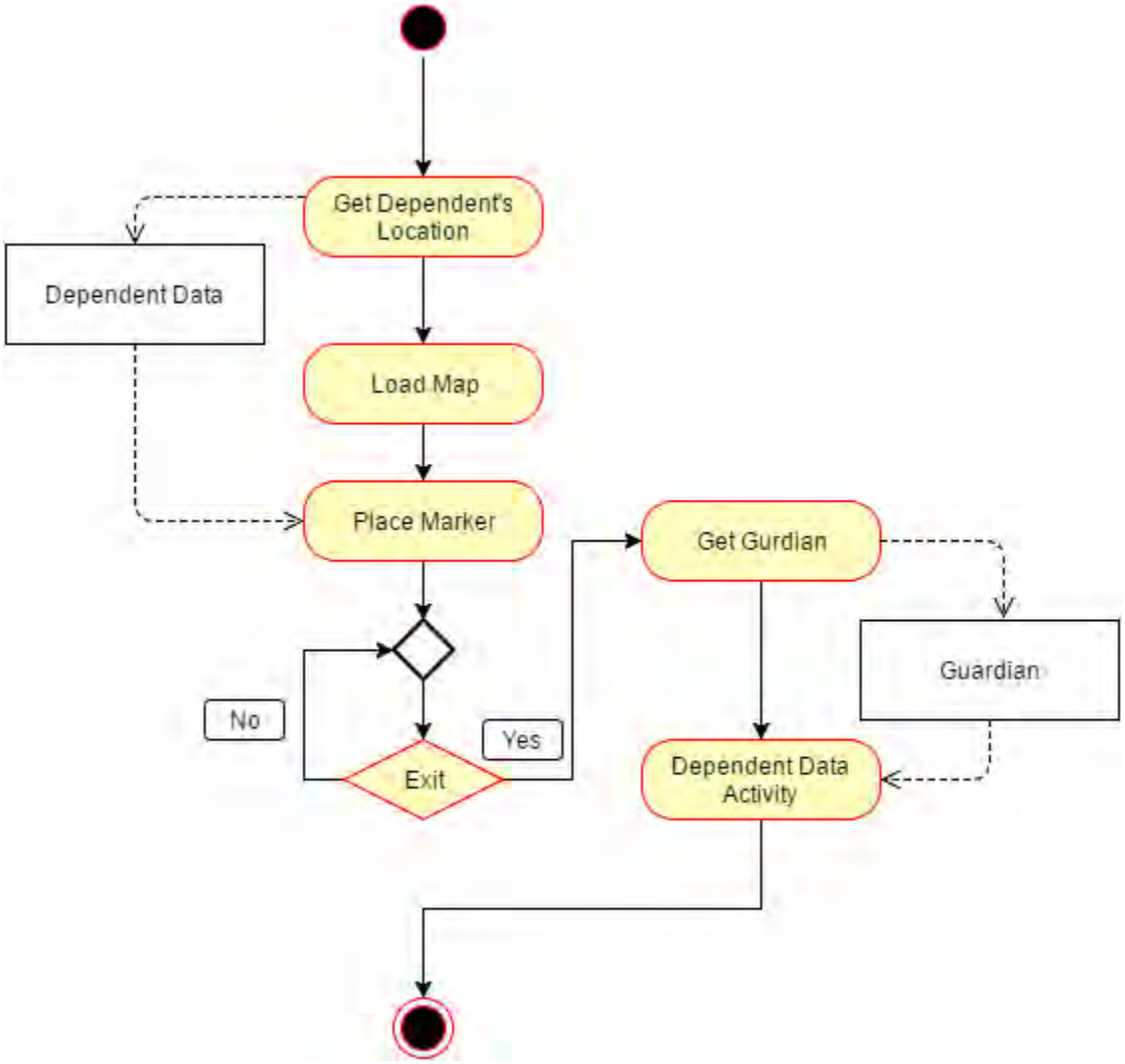


Figure 16 Show Dependent's Location Activity

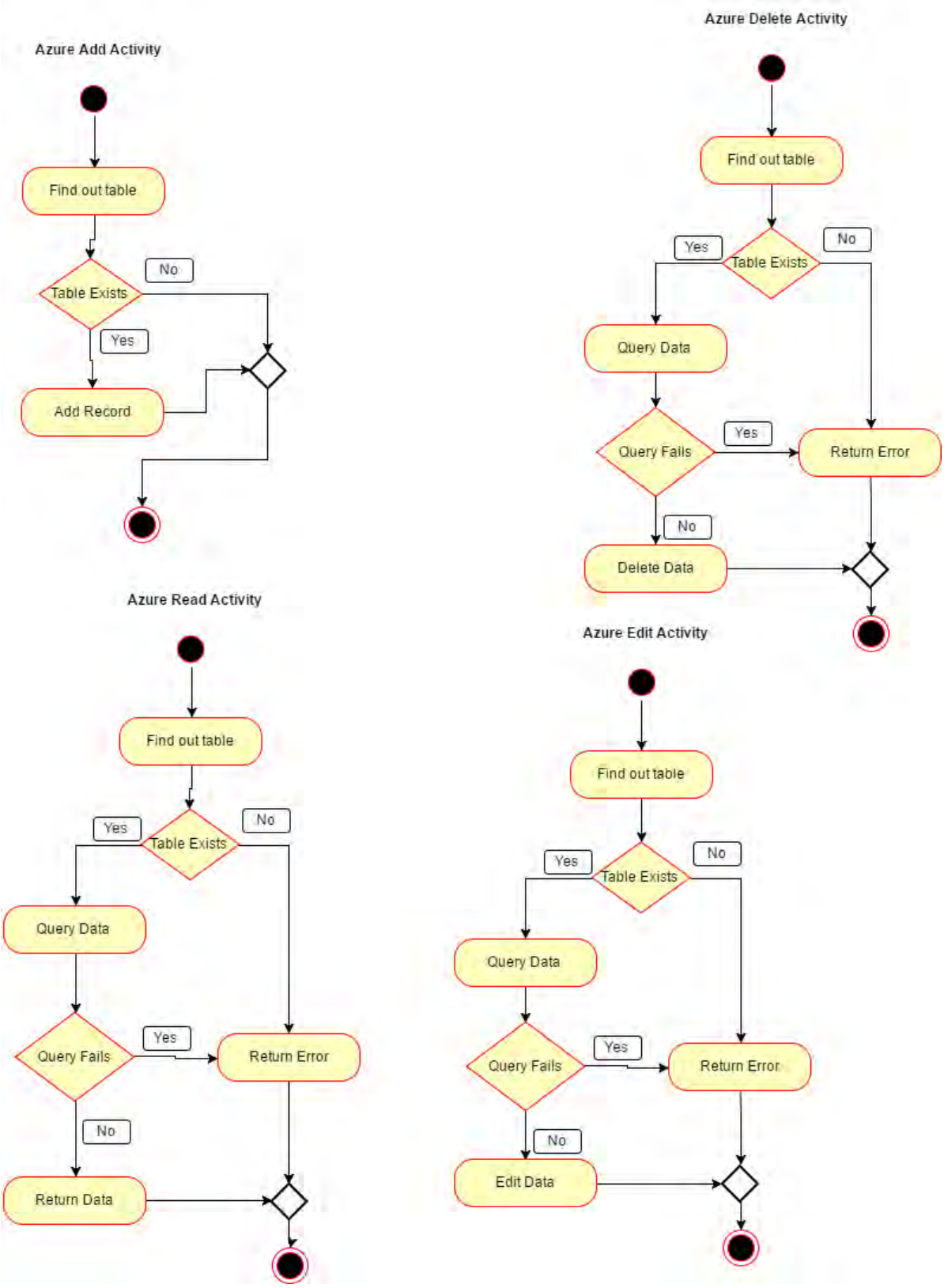


Figure 17 Activities of Microsoft Azure Functions

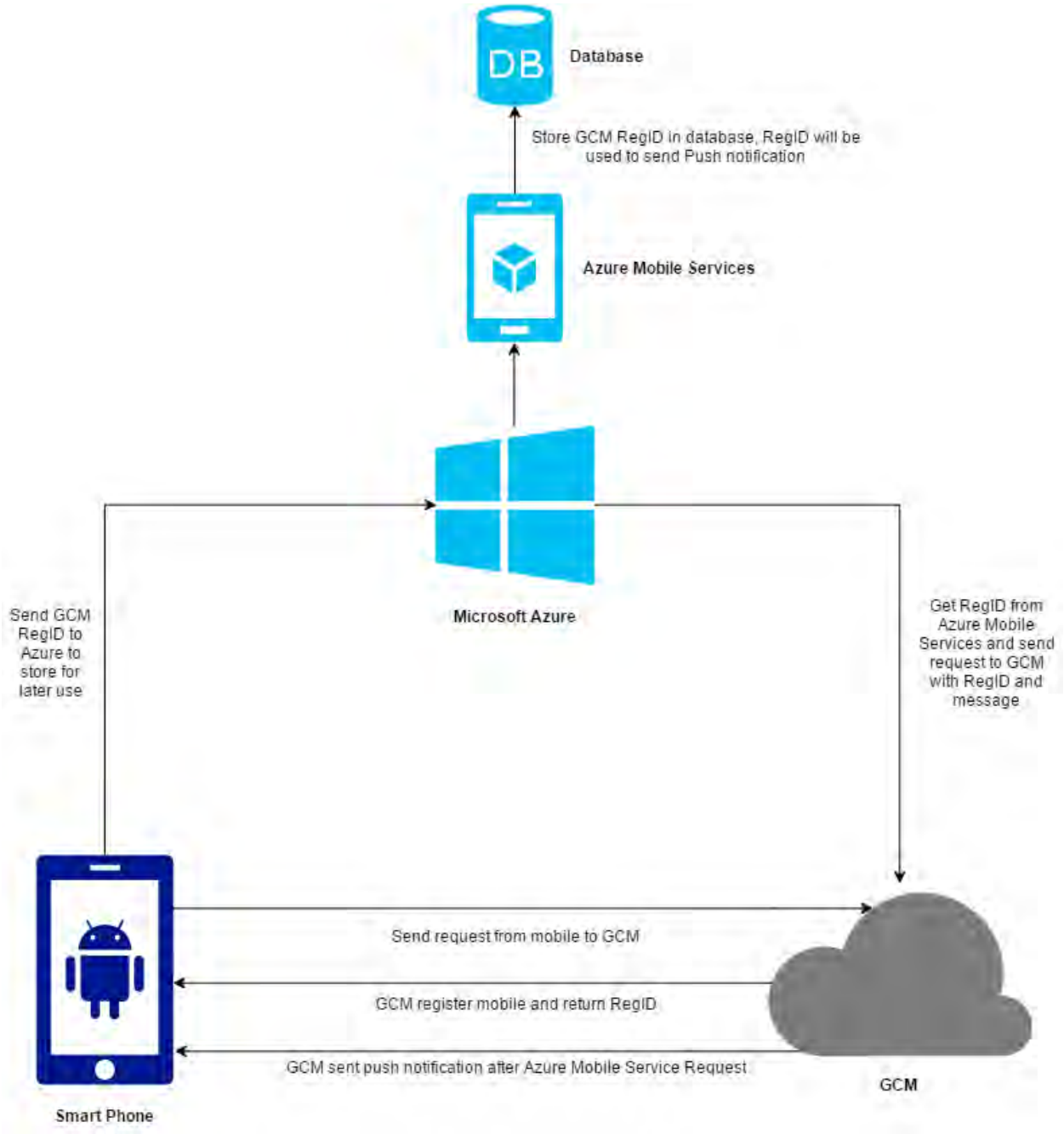


Figure 18 How notification works in the App

3.4 Cloud Services

In this section we shall discuss about the design of the cloud services we used in our project. We mainly needed the cloud services to authenticate the users (description of this is already provided in earlier sections) and to process data and requests made to the cloud service. Activity diagrams for processes on the cloud side are given as follows:

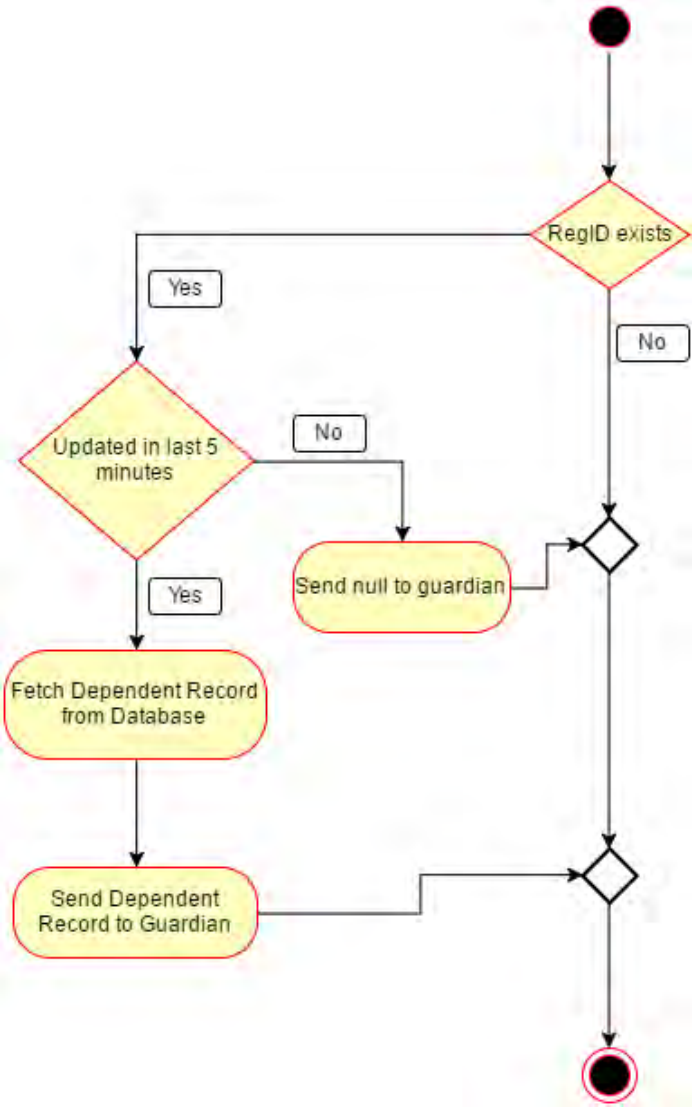


Figure 19 Data Read Activity from cloud

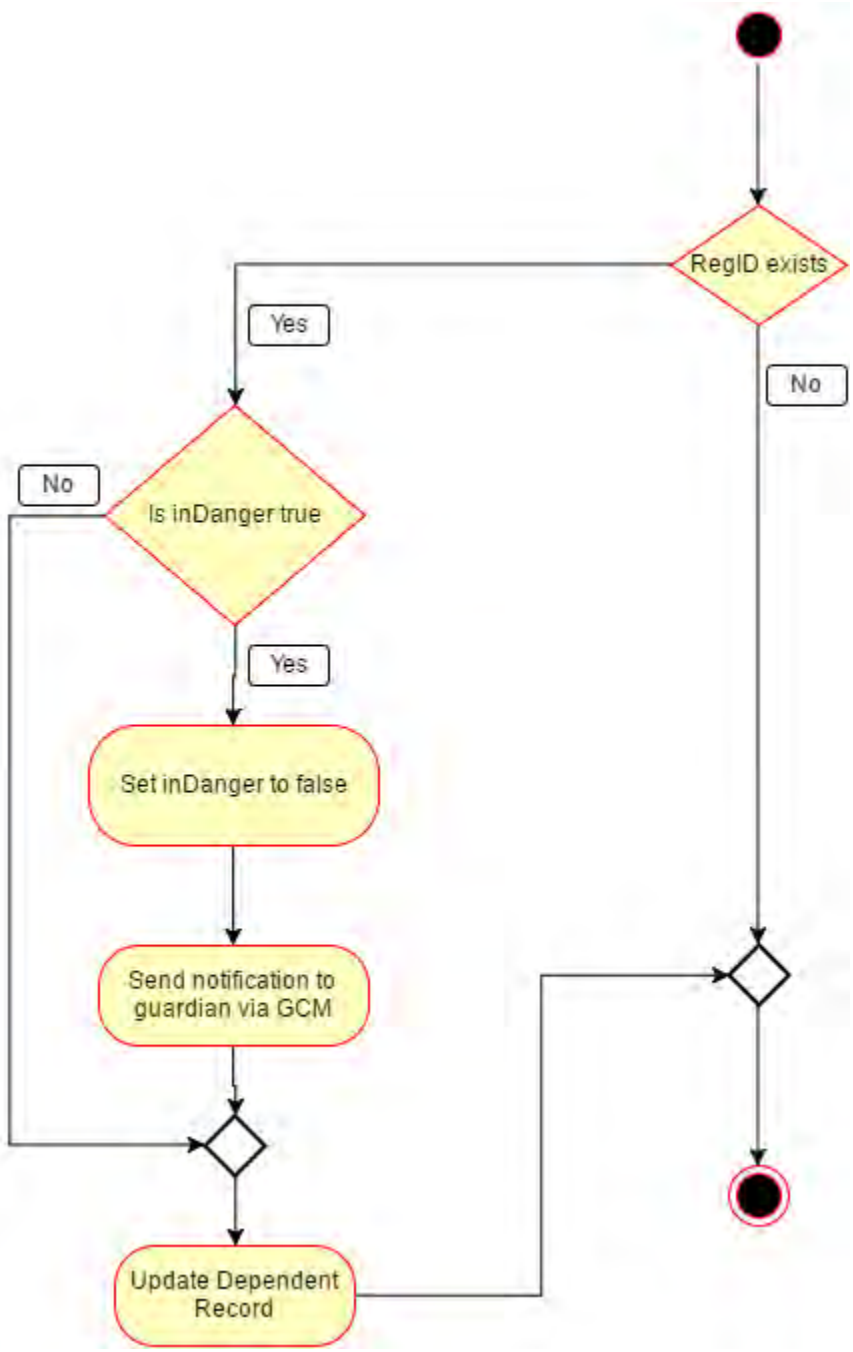


Figure 20 Data Update Activity in cloud

Chapter 4 System Specifications

4.1 Hardware Specifications

Arduino Uno

One of the two types of microcontroller boards we used for controlling the sensors and actuators in our devices is the Arduino Uno. The board contains 14 digital input/output pins among which 6 of them can be used as PWM (Pulse Width Modulation) outputs, 6 analog inputs, a USB connection, power jack, reset button and ICSP header. Its clock speed is 16MHz. Each digital pin receives a DC current of 20mA which makes it suitable for the sensors controlled by the board. Due to the availability of the USB connection, programming the microcontroller is a hassle free task as it can be done straight from the personal computer. The power jack further helped make our project complete as it allowed us to power the devices through the board with standard 9V battery packs. The weight of an UNO board is around 25 grams which helped keep our devices light and easy to carry.

Figure 21 An Arduino Uno board

Arduino Nano

The other type of microcontroller we used is the Arduino Nano. This microcontroller is from the same manufacturer, Arduino but is a smaller version of the Uno. The Nano comprises of the same number of digital pins and the same clock speed as the Uno. However it has 8 analog pins. The digital pins are provided about 40mA of current. Arduino Nano requires a breadboard for connection (it does not have jumper cable connectors on it like the Uno). It weighs only 5 grams and comes with a Mini-B USB connection. For programming Nano requires the same IDE (Integrated Development Environment) as the Uno. We used the Nano to control the pulse rate sensor in the right shoe.

Figure 22 An Arduino Nano board

HC-SR04

HC-SR04 is an ultrasonic ranging module that can measure distances of 2cm-400cm using sound waves in the range of 40 kHz, which is well above the range of human audibility. When supplied with an electric pulse for about 10µs the HC-SR04 by default triggers an 8 cycle burst of ultrasound. Whenever the ultrasound encounters an obstacle ahead it reflects back to the sensor. The sensor starts timing when an incident wave is triggered and stops timing when the reflected wave is received. The timing diagram is given below.

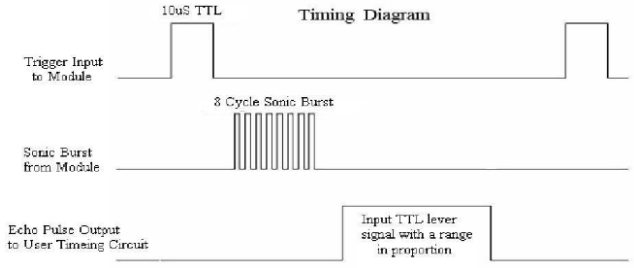


Figure 23 HC-SR04 timing diagram

For distance measurement by utilizing the time kept by the sensor we programmed the following formula in the microcontroller (for the purpose of signaling the actuators):

$$distance = \frac{duration/2}{29.1}$$

The formula above enables us to get the distance in centimeters. For the formula the velocity of sound is taken to be 343.5 m/s (at about a temperature of 20°C). When converted into cm/µs we get:

$$\frac{343.5 \times 100cm}{1{,}000{,}000\mu s} = 0.03435cm/\mu s$$

Now the numerator consisting of the variable duration is the amount of time taken for the ultrasound to reach the obstacle and bounce back to the sensor. Since this covers the two way trip, so for calculating the distance we need time for a one way trip which is multiplied by the velocity. This is given as:

$$distance = \frac{duration}{2} \times 0.03435$$

When written as denominator 0.03435 becomes:

$$\frac{1}{0.03435} = 29.1$$

Hence distances are calculated and ranges are set for obstacle detection in the belt and both shoes.

Figure 24 HC-SR04 Ultrasonic Sensor

SHARP GP2Y0A21YK

Infrared proximity sensors such as the SHARP GP2Y0A21YK, are able to sense objects ahead of them using electromagnetic radiation. The sensor emits an electromagnetic wave with a wavelength belonging to the Infrared part of the electromagnetic spectrum. This incident wave will reflect back after it hits an object in its way. The reflected wave's intensity will be affected depending on the distance of the sensor from the object (the closer the object, the stronger the reflected wave's intensity). It is this change in intensity that is sensed by the sensor and a corresponding analog output voltage is produced which can be manipulated by a microcontroller unit such as an Arduino to give the distance of the object. For detecting drop-offs, surface discontinuities or holes that our blind subject may encounter, we have used a SHARP GP2Y0A21YK long range proximity sensor. Its range is 10cm-80cm. Since it consumes a current of about 30mA and requires a supply voltage of 4.5V to 5.5V, it was suitable for use with our Arduino UNO board.

Figure 25 The SHARP GP2Y0A21YK IR sensor

Coin vibration motor

Coin vibration motors are small DC motors of diameters ranging from 0.5cm to as high as 1.2cm with flattened surfaces on both sides. These motors provide vibration when a DC voltage equivalent to the nominal voltage (3V in most cases) or higher is applied to the motor. The commutator, motor shaft are all packed within the round packaging. The vibration of these motors are barely audible, and are generally used in cell phones. Due to their non-audible vibrations they are suitable for devices that rely on haptics as indication. We have used motors of 0.5cm diameter in our devices which are able to give us the desired frequency of vibration as programmed in the Arduino Uno at a voltage of 5V and a current of about 20mA.

Figure 26 Coin vibration Motor

Bluetooth Module

The Bluetooth modules we used in both the belt and the right shoe are HC-05 modules. These are Serial Port Protocol modules that are made for transparent wireless connection setup. HC-05 has a dimension of 1.5cm x 3cm, making it small and very light weight, and has the potential to communicate up to a distance of 5m. However our devices are in closer proximity to the smartphone hence HC-05 is apt for them. This module is a master/slave module which means that it allows for bidirectional communication. The Arduino boards have pins especially for Bluetooth module connections, which we were able to utilize with the HC-05.

Figure 27 HC-05 Bluetooth Module

Pulse Rate Sensor

The pulse rate sensor we used is an open source project which is apt for use when coupled with an Arduino microcontroller. Its function is based on a very basic principle of optoelectronics. The sensor uses an LED (Light Emitting Diode) and an LDR (Light Dependent Resistor) to detect a pulse. When placed above a skin area, underneath which is an artery, the sensor's LED emits a light. When heart pumps blood into the artery then more blood cells are available to absorb the light. Therefore the reflected light that is received by the LDR will have less intensity. It is the change in intensity that the LDR looks out for at each heartbeat. When the light intensity falls, the resistance of the LDR decreases, varying the voltage across it. The voltage variation is amplified by the operational amplifier in the sensor to a value that is detectable by the Arduino. The Arduino program that comes with the pulse rate sensor enables the board to display the voltage variation at each heartbeat and it also keeps track of the number of beats per minute (as calculated from the voltage values). We have altered the program according to our design so that the sensor takes 12 readings per minute (1 reading each 5 seconds in a total of 60 seconds).

Figures are on next page.

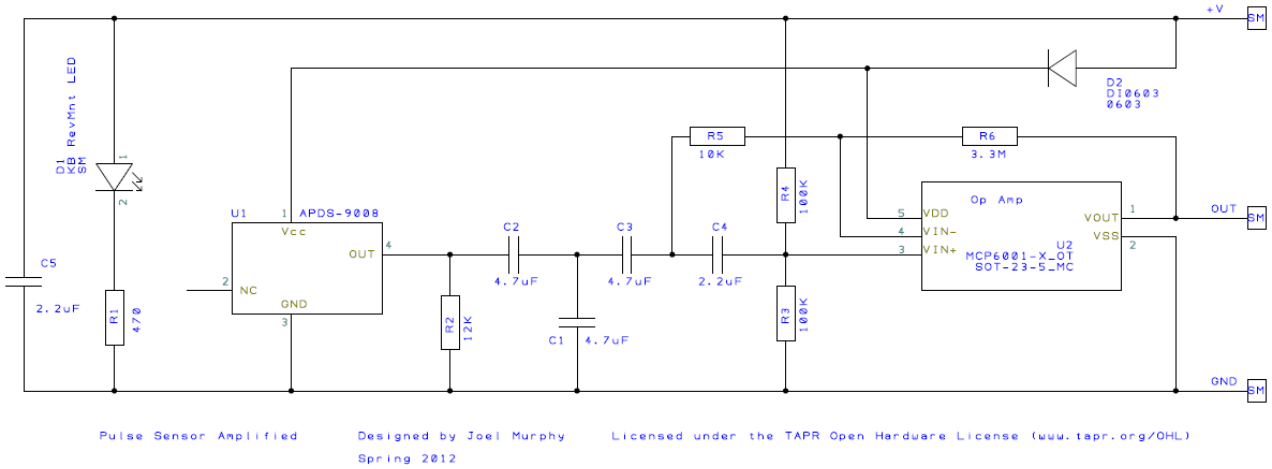


Figure 28 Schematic for the pulse rate sensor

Figure 29 Pulse Rate Sensor

4.2 Software Specifications

Android Studio

To build our software we used Android Studio 2.1. Android Studio is the official cross-platform Integrated Development Environment (IDE) promoted by Google for android development. It free and easy to use. It has better code completion and has code refactoring which allows for faster work flow than Eclipse Android Development Tools (ADT) (which has been replaced by Android studio), while still allowing the users to write android applications with Java. It uses Gradle as the build engine. It has an intuitive layout editor where it is possible to drag and drop elements to create the user interface. The user can also define the layout using xml.

Figure 30 Android Studio

Microsoft Azure

Microsoft Azure is a growing collection of Cloud services provided by Microsoft. They are the second largest cloud service provider after Amazon Web Services. Microsoft provides a wide range of easy to use and reliable services ranging from Azure App Services (Platform as a Service, PaaS) to hosting virtual machines (Infrastructure as a service, Iaas). Microsoft Azure has a flexible payment model which allows companies to "pay as they go" that is, scale up or scale down the resources they are using from Azure (like the number of virtual machines for example) according to their own user demands and pay Azure for only what they used.

For our project we used Azure Mobile Services which provides a well-tested and documented Sdk to use common services required Mobile Apps such as sending and receiving push notifications, authenticating users and storing and processing the user data. The Android Sdk for Azure Mobile Services blankets a SQL database in the back end and allows the user to carry out all kinds of operations on it. For the server side code, Azure allows both Node.js and .NET Framework. We decided to use Node.js as it is more versatile than C# (used in .NET) for this purpose. The permissions for every table can be altered individually. This allows us to regulate access to each tables in our database using Application keys or Master Keys or even only allow authenticated users to access or use the data in a certain table. Users can be authenticated by connecting the mobile service with an authentication provider such as Google OAuth.

Figure 31 Azure Mobile Services Logo

Figure 32 Microsoft Azure Logo

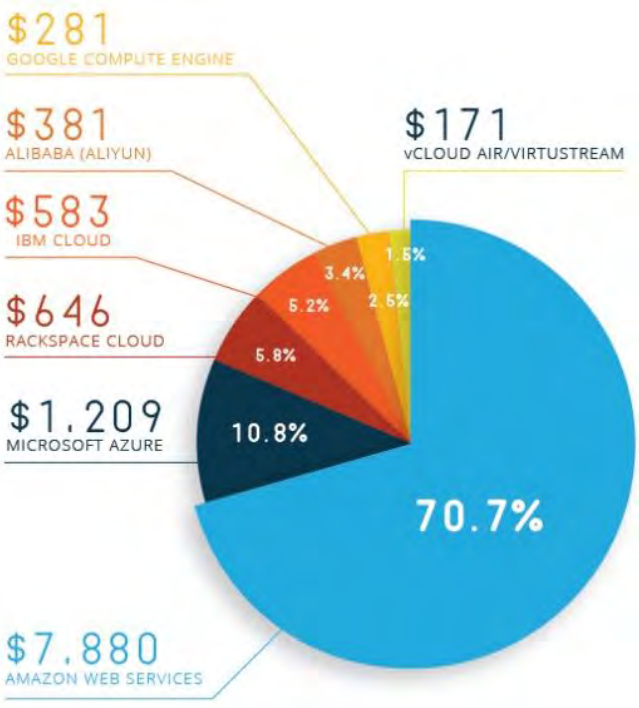


Figure 33 Cloud Services Market Share

NodeJs

Node.js is an open source cross platform runtime environment for developing server-side parts of Applications. It supports Asynchronous Input / Output to enable high frequency transfer of data which is needed for Web based applications. Node.js is open source and allows the users to add new modules written in JavaScript to it as needed. Node.js executes commands in parallel and using callbacks to notify a failure or success on execution completion. It is ideal for frequent input and read operations that is required by the apps in our project. We have used it to write our server side code in Azure.

Figure 34 NodeJs Logo

Arduino IDE:

Arduino IDE is an open-source Integrated Development environment that allows users to write code for Arduino boards in C and C++ and upload the program into the Arduino using a Universal Serial Bus (USB) port. The IDE runs on Linux, Mac OS X and Windows, and is compatible with all Arduino boards. We used this to write our Arduino code as it is lightweight and easy to use.

Figure 35 Arduino IDE Logo on Windows

Google Play Services

Google play services is Google's copyrighted background service. It includes APIs for a large number Google services including Google OAuth 2.0, Google location Service, Google Maps and Google Cloud Messaging services, etc. Internally, all the services are collectively referred to as Google Mobile Services (gms).

Our Apps make use of Google OAuth 2.0 to authenticate our users. It is an easy and secure way of authenticating users as the authentication data is completely handled by a Google and we do not need to explicitly store the user's login data.

We have also used Google location API to get dependent's location using GPS, and Google Maps Android API to show the dependent’s location on the map.

We have also used Google Cloud Messaging Services API to send push notifications to the Guardian App.

Figure 36 Google Play Services Logo

Git

We used Git as a version controlling tool for our app. Git is a very popular, freely distributed under the terms of the GNU General Public License version 2, version control system that is used for software development and other version control tasks. It is aimed at speed, data integrity, and support for distributed, non-linear workflows. Furthermore, Git can be directly integrated with Android Studio making it super easy for us to use it.

Every Git directory on every computer is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.

Figure 37 Git Logo

Bluetooth

Bluetooth is a wireless technology standard which uses UHF radio waves to transfer data between devices over a short distance. It is an easy to use and available technology that is often used to connect wireless devices to phones, computers, etc. We have used Bluetooth to connect our shoe and belt to our Dependent Android App.

Figure 38 Bluetooth Logo

Chapter 5 – Implementation

In this chapter we will discuss about the implementation of our project. We implemented the prototype in three phases according the design we came up with as mentioned earlier. In the first phase we implemented the hardware and in the second phase we implemented the cloud services and in the third phase we implemented the software.

In the following subsections we shall discuss about the two phases of implementation.

5.1 First Phase: Hardware Implementation

As mentioned earlier in Chapter 3, the hardware part of our project consists of two shoes and a belt. The hardware components used to make each wearable prototype is mentioned in the following subsections along with brief descriptions of the implementation.

The Belt

We began the development of the devices with the belt. The belt is made up of a 38.5 inch long, 2.5 inch wide woven Oxford fabric which is 3 to 4 layers thick with Velcro strap covering 11 inches on either side of the belt. The aforementioned layered fabric was chosen for its ability to keep the sensors and coin vibration motors in place which we sewed on the belt. The Velcro strap makes it easy for the wearer to wrap it around the waist or remove it.

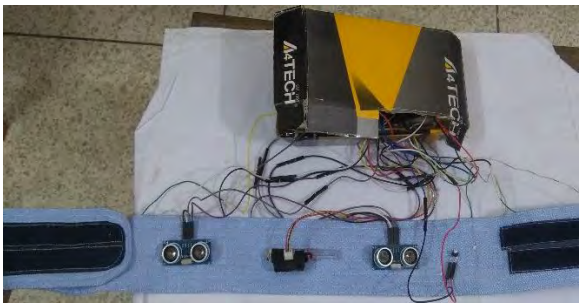


Figure 39 The belt with the sensors and belt pack unit

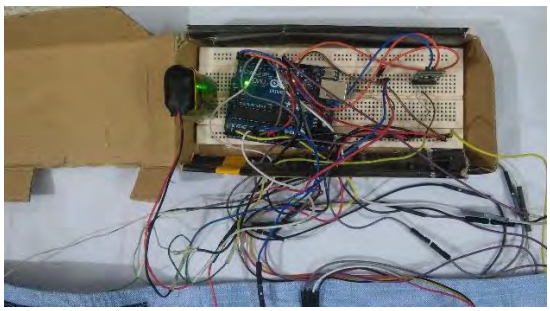


Figure 40 The belt pack unit containing the breadboard with all connections

As seen in the figures on the previous page, we have sewn in a spindle at the center of the belt, on top of which the infrared proximity sensor is tied. The spindle allows us to vary the angle of the infrared sensor from 0°-20° downwards. Varying the angle during initial testing enabled us to find the appropriate positioning of the infrared sensor that enabled drop-off detections from a distance that allowed the user to have adequate reaction time and change course. For our design the infrared sensor (SHARP GP2Y0A21YK) is able to detect a hole, or drop of a minimum depth of 53.34 cm (approximately 1.75 feet) from a distance of 9 inches, when the sensor is positioned at an angle of 15° downward. Whenever a hole ahead of the user is detected the vibration motor sewn onto the back at the inner surface of the belt vibrates to indicate him about it.

The ultrasonic sensors (HC-SR04) on the belt are used for detecting obstacles. With a field angle of 30° from left to right for each sensor, together the two sensors can sense obstacles for an area of 60° spanned ahead of the user. Each sensor has been sewn on either side of the belt at 3 inches away from the belt, as seen in the Fig 21. The positions of the ultrasonic sensors are such that their fields do not coincide with that of the infrared proximity sensor. The ultrasonic sensors sense and warn the user about obtacles at waist level ahead of the user according to the design discussed in the previous chapter. The push button which enables the user to communicate with a remote guide is attached to the belt beside the right ultrasonic sensor as seen in Fig 21 as well. The left and right vibration motors are each sewn onto the inner surface of the belt as seen in Fig 24. The sensors and vibrating motors are all controlled by he Arduino Uno board. The board and the components are conncted using a breadboard which is kept in a box, which we refer to as the belt pack unit. Whenever the belt is worn the belt back unit is held together at the right side of the user using Velcro straps that wrap around the belt and the box.

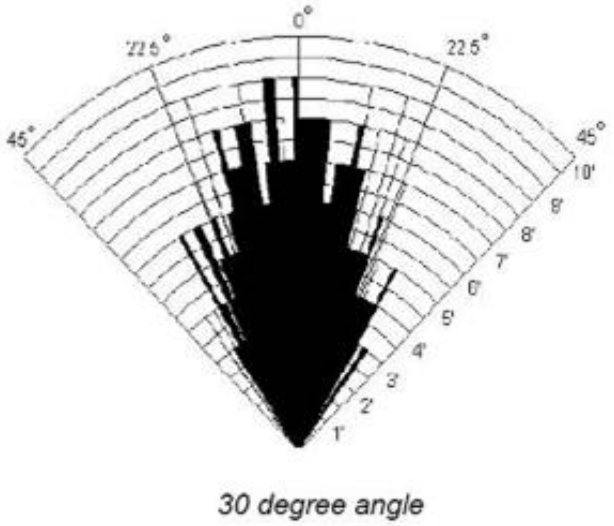


Figure 41 Field angle of HC-SR04

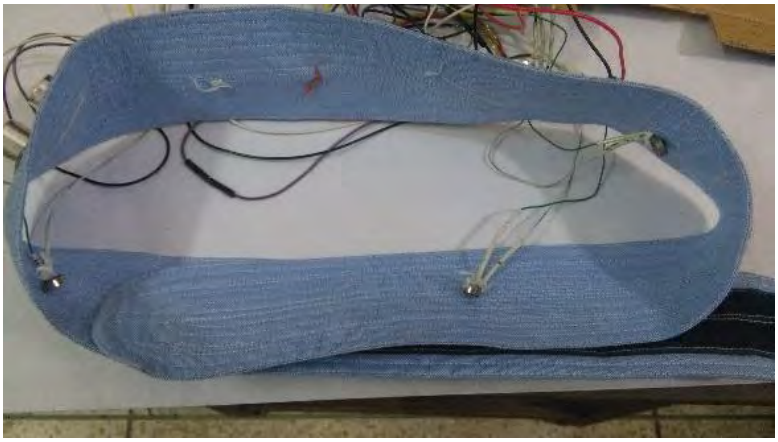


Figure 42 The vibration motors sewn onto the inner surface of the belt

The Shoes

The right shoe has ultrasonic sensors sewn onto the front and the right side. These are able to detect obstacles at the feet level. The vibration motors are attached underneath the sole of the shoes. The pulse rate sensor is sewn onto the flap of the right shoe as seen in the fig. The pulse rate sensor is placed in such a way that the LED of the sensor comes in contact with the area of the feet underneath which is the dorsalis pedis artery. This artery gives a prominent reading for pulses. With wires extended from the sensors and actuators, the connections are made to the Arduino Uno and Nano on a 5.5cm x 17.5cm breadboard which is attached to the left side of the right shoe. The Arduino Nano has been used to control the pulse rate sensor. For the rest of the components including the Bluetooth module the Arduino Uno is used. Both the boards are powered by the same battery.

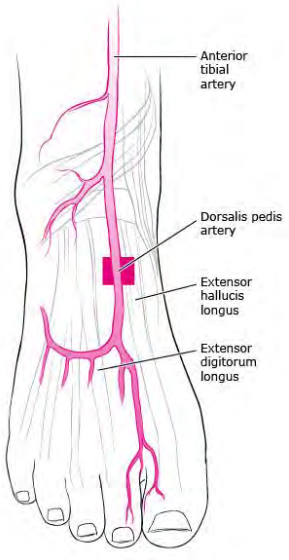


Figure 43 The position of the dorsalis pedis artery

Figure 44 The pulse rate sensor under the flap and the vibration motors on the sole of the right shoe

The left shoe has ultrasonic sensors to the front and left side. Similar to the right shoe the left shoe also contains vibration motors attached to the base of the shoe under the sole. The front vibration motor is positioned under the big toe at the ball of the feet. The left vibration motor is placed at the left of the mid part of the foot's sole. The sensors and motors are connected to the Arduino Uno board attached to the right side of the left shoe using a 5.5cm x 17cm breadboard.

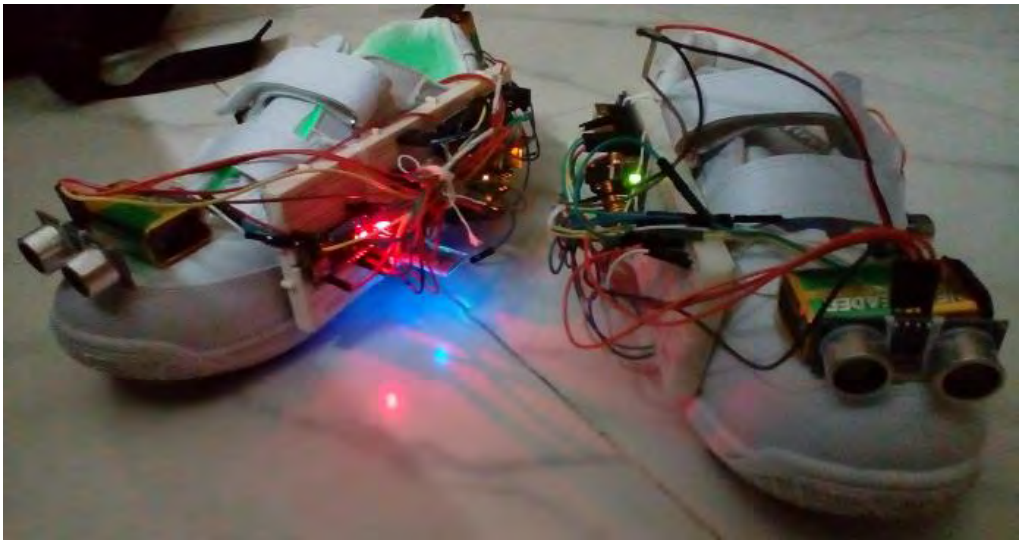


Figure 45 The shoes with the wire connections on the inner side and front ultrasonic sensors

Figure 46 The left shoe with the ultrasonic sensor at the left side

Figure 47 The right shoe with the ultrasonic sensor at the right side

5.2 Second Phase: Cloud Services Implementation

We implemented the system according to the design. Here is a screenshot of our table inside Azure.

First we connected a database and a SQL Server for the database to our mobile service.

Connecting Database

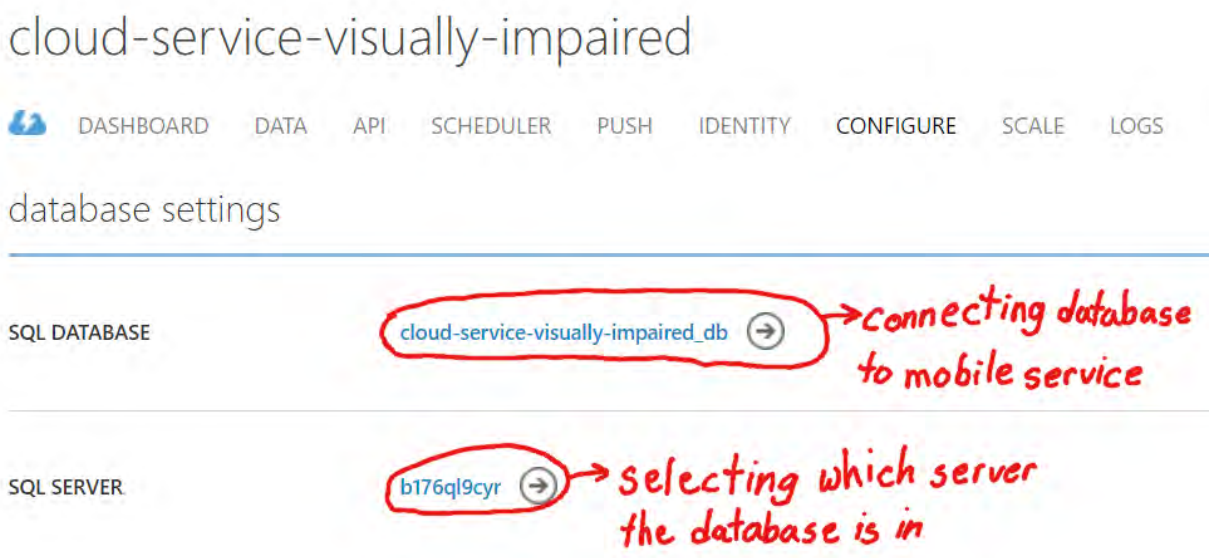


Figure 48 Adding database and SQL Server to the mobile service

Creating Table

We created a table called “People” to hold all our data according to the design.

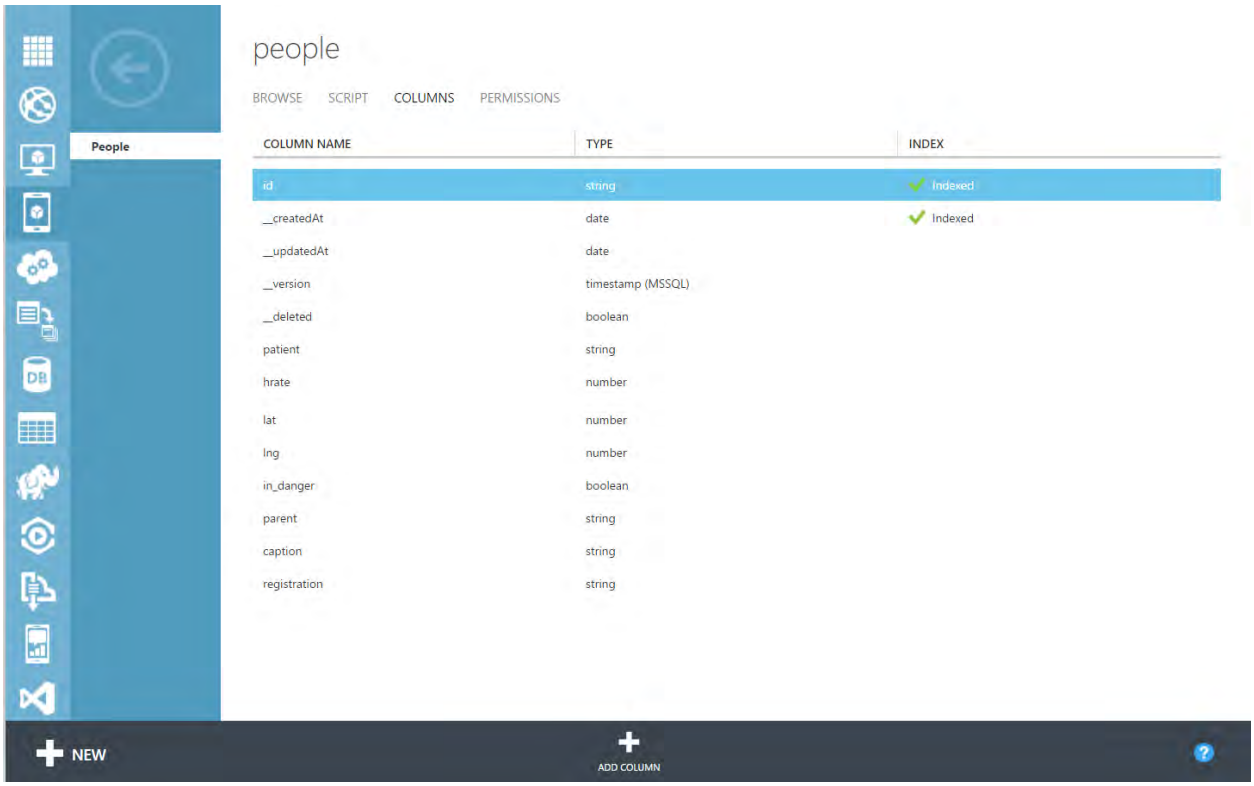


Figure 49 People table in database

Adding server side scripts

Server side scripts were written in NodeJs according to the design and were added to the Azure mobile service.

Connecting to GCM for push notifications

We got an auto generated API key from Google API Manager which we later added to our Azure mobile service, so that it could communicate with GCM to send push notifications the appropriate guardian.

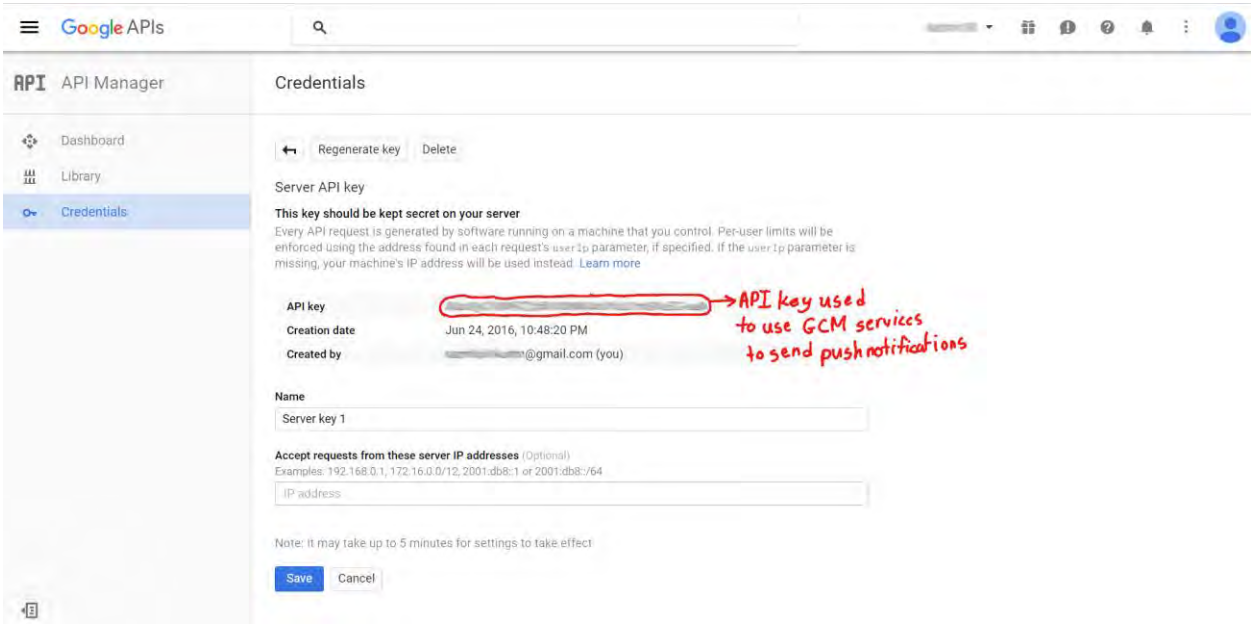


Figure 50 Getting auto-generated API key from Google API Manager

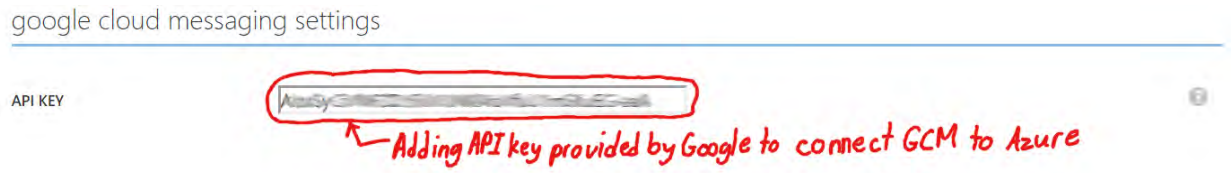


Figure 51 Adding API key to Azure mobile service to connect it to GCM

5.3 Software Implementation

In this section we discuss about how we developed the two apps according to the design we came up with in chapter 3.

Dependent App

In this section we discuss about the various aspects of the implementation of the dependent app. We have built the app using Android Studio v2.1. More details are as follows:

- Build Tool Version: 24.0.0
- Compile Sdk Version: 24
- Minimum Sdk Version: 16
- Gradle Version: 2.10

Dependencies of the app

- testCompile 'junit:junit:4.12'
    - Needed for unit testing
- compile 'com.android.support:appcompat-v7:24.0.0'
    - Needed to support Action bar UI
- compile 'com.microsoft.azure:azure-mobile-services-android-sdk:2.0.3'
    - Needed for using Azure Mobile Services in our App
- compile 'com.google.code.gson:gson:2.4'
    - Needed to convert a Java Object to a JSON representation and vice versa. This is needed to transfer data to and from Azure and make queries without the need of SQL
- compile 'com.google.guava:guava:18.0'
    - It facilitates best coding practices and helps reduce coding errors
- compile 'com.google.android.gms:play-services-auth:9.2.0'
    - Needed for Google OAuth
- compile 'com.android.support:design:24.0.0'
    - Needed for adding material design components and patterns to our App
- compile 'com.google.android.gms:play-services-appindexing:9.2.0'
    - Needed for indexing the app

Java Classes we wrote for the app

We wrote seven Java classes for the app. They are as follows:

- LoginActivity
  - Deals with the Login Screen. Handles the Google OAuth to authenticate the user
- DeviceListActivity
  - Deals with the Screen where the user connects to the wearables via Bluetooth.
- ParentDashActivity
  - Deals with the Dashboard Screen of the app. Connects the app to Azure mobile services in the background.
- DependentDetailActivity
  - Starts the MonitoringService if it is not already running
- MonitoringService
  - Manages connections to the wearables and constantly reads data from them. After every 5s it reads heartrate from the right shoe and then it checks weather the blind person has pushed the emergency button or not. It also gets user's location using the phone's GPS. It then uploads all data to the cloud on a separate thread. If the emergency button is pressed it initiates phone call to the emergency contact number for the patient on a separate thread and it also immediately uploads all data to the cloud on a separate thread. This service also starts running automatically if for some reason the service gets closed by the OS. This ensures this service never stops monitoring the patients.

- ClientManager
    - This class abstracts away all Azure complex functions. This class provides static methods to update, insert, delete and read records from the database with single lines of codes. This made life extremely easy for us later on as we did not need to write long lines of codes later on every time we needed to run a query or insert or update records in the database.
- People
    - This is the data model for our app.

We have included the screenshot of the classes from the IDE as follows:

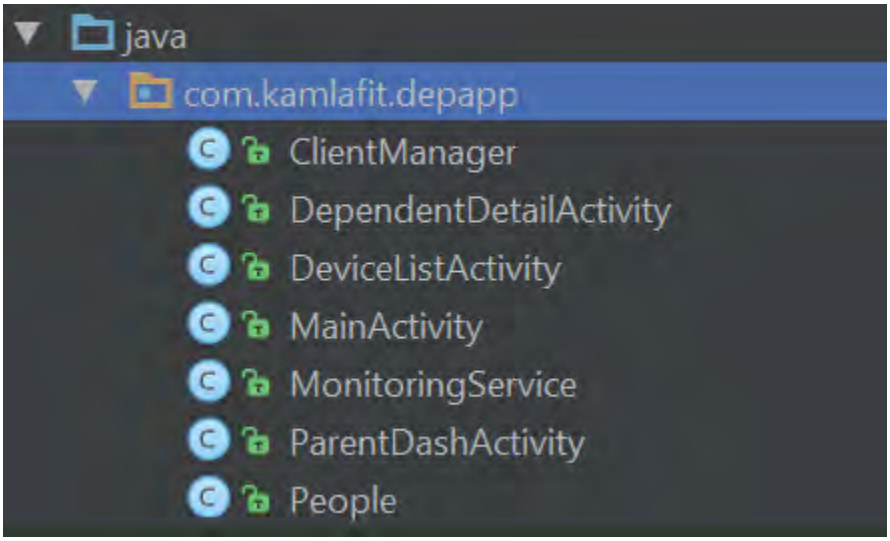


Figure 52 Java Classes we wrote for the dependent app

Permissions

- BLUETOOTH
  - This grants the permission to use the Bluetooth of the device
- CALL_PHONE
  - This grants the permission to make calls from the device
- INTERNET
  - This grants the permission to use the Internet on the device
- GET_ACCOUNTS
  - Grants permission to read Google account (needed for android 3.0 and lower)
- READ_PROFILE
  - This grants permission to query user profile data
- READ_CONTACTS
  - Grants permission to access user's Contacts data (needed for phone calls)
- ACCESS_FINE_LOCATION
  - This grants permission to access and use the devices fine location using GPS
- ACCESS_COARSE_LOCATION
  - This grants permission to access and use the devices coarse location using GPS

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Figure 53 Screenshot of permissions from the IDE

Guardian App

In this section we discuss about the various aspects of the implementation of the guardian app. Similar to the dependent app we have built this app using Android Studio v2.1. More details are as follows:

- Build Tool Version: 24.0.0
- Compile Sdk Version: 24
- Minimum Sdk Version: 16
- Gradle Version: 2.10

Dependencies of the App

- testCompile 'junit:junit:4.12'
    - Needed for unit testing
- compile 'com.android.support:multidex:1.0.1'
    - Since our app is relatively large i.e. it has more than 65,536 methods (including all methods in libraries used in the app), we needed this library to compile our project
- compile 'com.android.support:appcompat-v7:24.0.0'
    - Needed to support Action bar UI
- compile 'com.microsoft.azure:azure-mobile-services-android-sdk:2.0.3'
    - Needed for using Azure Mobile Services in our App
- compile 'com.google.code.gson:gson:2.4'
    - Needed to convert a Java Object to a JSON representation and vice versa. This is needed to transfer data to and from Azure and make queries without the need of SQL
- compile 'com.google.guava:guava:18.0'
    - It facilitates best coding practices and helps reduce coding errors
- compile 'com.google.android.gms:play-services-auth:9.2.0'
    - Needed for Google OAuth
- compile "com.google.android.gms:play-services-maps:9.2.0"
    - Needed for displaying blind person’s location on a map
- compile 'com.android.support:design:24.0.0'
    - Needed for adding material design components and patterns to our App

- compile 'com.google.android.gms:play-services-gcm:9.2.0'
    - Needed for receiving notifications via GCM
- compile 'com.microsoft.azure:notification-hubs-android-sdk:0.4@aar'
    - Needed for using Azure notification hub
- compile 'com.microsoft.azure:azure-notifications-handler:1.0.1@aar'
    - Needed to handle notifications when they arrive

```
dependencies {
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:multidex:1.0.1'
    compile 'com.android.support:appcompat-v7:24.0.0'
    compile 'com.microsoft.azure:azure-mobile-services-android-sdk:2.0.3'
    compile 'com.google.code.gson:gson:2.4'
    compile 'com.google.guava:guava:18.0'
    compile 'com.google.android.gms:play-services-auth:9.2.0'
    compile "com.google.android.gms:play-services-maps:9.2.0"
    compile 'com.android.support:design:24.0.0'
    compile 'com.google.android.gms:play-services-gcm:9.2.0'
    compile 'com.microsoft.azure:notification-hubs-android-sdk:0.4@aar'
    compile 'com.microsoft.azure:azure-notifications-handler:1.0.1@aar'
}
```

Figure 54 Screenshot of Dependencies in the app from the IDE

Java Classes we wrote for the app

We wrote seven Java classes for the app. They are as follows:

- LoginActivity
  - Deals with the Login Screen. Handles the Google OAuth to authenticate the user
- ParentDashActivity
  - Deals with the Dashboard Screen of the app. Connects the app to Azure mobile services in the background.
- DependentDetailActivity
  - Starts the MonitoringService if it is not already running
- ClientManager
  - This class abstracts away all Azure complex functions. This class provides static methods to update, insert, delete and read records from the database with single lines of codes. This made life extremely easy for us later on as we did not need to write long lines of codes later on every time we needed to run a query or insert or update records in the database.
- People
  - This is the data model for our app.
- MapsActivity
  - This class deals with displaying the location of the blind user on a map
- MyHandler
  - This class receives the notification, retrieves data from it and actually displays it on the device. It also controls which activity is opened when the notification is clicked.

- MyInstanceIDService
    - Refreshes the GCM registration token if it has expired
- NotificationSettings
    - Holds connection string to connect app to relevant Notifications Hub in Azure and the Google Cloud messaging Service project used for this project.
- RegistrationIntentService
    - Gets a Registration Id from Google Cloud Messaging Service and sends it Azure to be stored in a table.

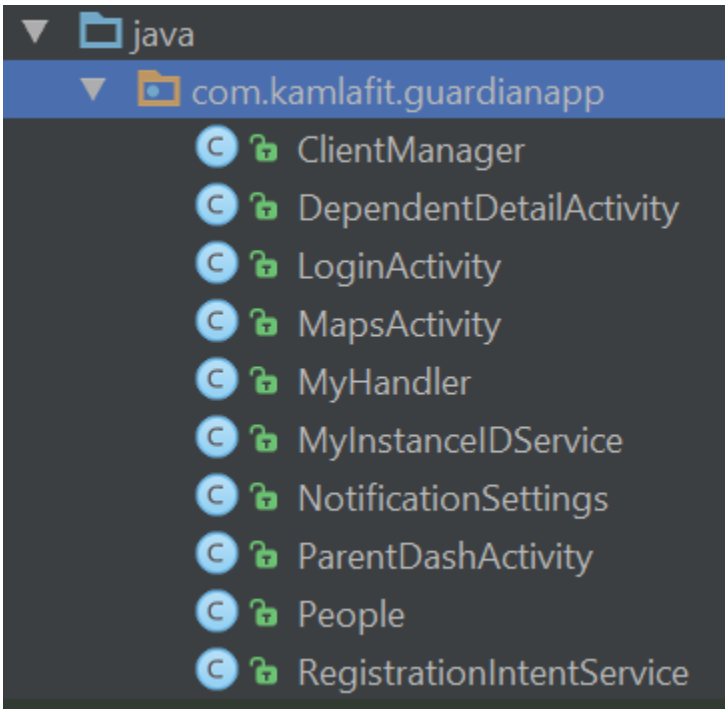


Figure 55 Screenshot of Java Classes we wrote for the guardian app from the IDE

Permissions

- INTERNET
    - This grants the permission to use the Internet on the device
- GET_ACCOUNTS
    - Grants permission to read Google account (needed for android 3.0 and lower)
- READ_PROFILE
    - This grants permission to query user profile data
- READ_CONTACTS
    - Grants permission to access user's Contacts data (needed for phone calls)

```
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_PROFILE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.INTERNET"/>
```

Figure 56 Screenshot of permissions for the Guardian App from the IDE

Chapter 6 Result and Analysis

In this particular chapter we will discuss about the cost estimation of our thesis project. Then we shall talk about the performance of the Dependent app while being paired to the wearables. After that we will discuss about the performance of the Guardian app. We will also talk about performance tests of the cloud services used in our app.

6.1 Cost Estimation

Trying to reduce the cost margin to a minimum while at the same time making sure that no compromises were made to the safety of the user and accuracy of the data output, was one of the biggest challenges we faced while working on our thesis project. In the following we have provided a breakdown of the estimated costs of our system based on the prototype we built which include hardware costs, software costs and costs of cloud services.

Hardware Costs:

Device	Parts	Cost
Belt	Arduino Uno x 1	Tk 650.00
	HC-SR04 (ultrasonic sensors) x 2	Tk 300.00
	SHARP GP2Y0A21YK (Infrared distance measuring sensor) x 1	Tk 630.00
	Coin vibration motor (as found in cell phone units) x 3	Tk 750.00
	Bluetooth Module x 1	Tk 540.00
	Battery x 1	Tk 200.00
	Push Button	Tk 20.00
	Total	**Tk 3090.00**
Right Shoe	Arduino Uno x 1	Tk 650.0
	Arduino Nano x 1	Tk 900.00
	HC-SR04 (ultrasonic sensors) x 2	Tk 300.00
	Pulse Rate Sensor x 1	Tk 900.00
	Coin vibration motor (as found in cell phone units) x 2	Tk 500.00
	Bluetooth Module x 1	Tk 540.00
	Battery x 2	Tk 400.00
	Total	**Tk 4190.00**

Left Shoe	Arduino Uno x 1	Tk 650.00
	HC-SR04 (ultrasonic sensors) x 2	Tk 300.00
	Coin vibration motor (as found in cell phone units) x 2	Tk 500.00
	Battery x 1	Tk 200.00
	Total	**Tk 1650.00**

Android Phone	Not Applicable	Price depends on Consumer Choices and affordability with prices of phones varying from below Tk 5,000.00 to above Tk 80,000.00

Software costs:

The apps developed using free and open source tools are going to be free, so the user do not have to pay anything for them.

Costs of cloud services:

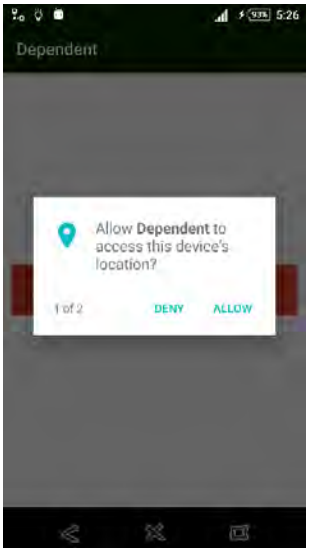
Assuming each user uses the cloud services for 10 hours every day, the cloud service cost for that user per month would be around US$ 0.3 to US$ 0.7 which we could easily earn back through advertisements on the Guardian app. So the end users of our app can use it free of charge at all times.

6.2 Performance analysis of dependent app along with wearables

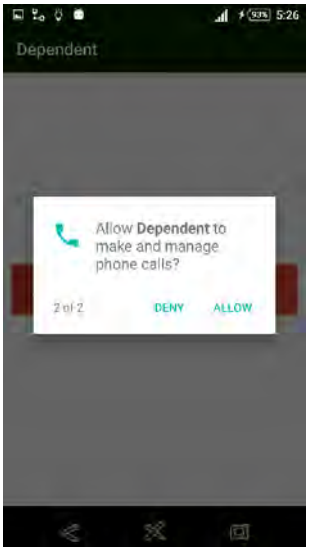
We have made the dependent app and the wearables go through rigorous testing to ensure the system works reliably. The following subsections discuss about several performance analysis about different aspects of the system that we made.

Authentication

Authentication via Google OAuth worked smoothly, effectively and reliably in all our testing. Before the app even authenticates the user, the app will ask the user to turn on GPS if it is not already turned on and will ask for permission to make phone calls.

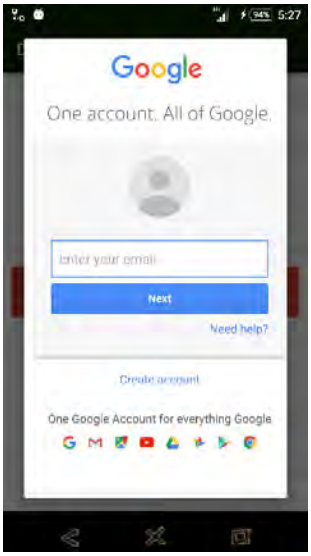


If permission is not granted, the app asks for permission to access device's location



If permission is not granted, the app asks for permission to make phone calls

The App enables authentication via Google OAuth



User can login with their Google ID

Entering Password and Signing In

Figure 57 Testing the Dependent App

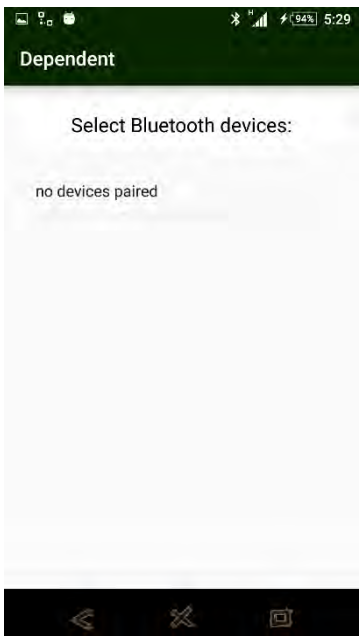
Connection with wearables via Bluetooth and the Cloud via the Internet

In all our testing, we have found out that connecting to wearables via Bluetooth is also very reliable. If the device's Bluetooth is turned off, the app asks for permission to turn on Bluetooth from the user. If the shoe and the belt are not turned on, the app displays a message to the user to turn them on. Once both the shoe and the belt are turned on, the user can connect to them. While connected to the wearables, the app can read data from them quite reliably. In all our testing with an Asus Zenfone Go, we have not noticed any delays in getting readings from both the wearable prototypes. The phones response to pressing the emergency button on the belt has always been instant. It makes a call to the emergency number instantly. It also sends push notification to the guardian instantly. It also reads heart rate from the right shoe without any flaws. The heartrate sensor takes about half a minute to stabilize before it starts providing reliable readings. Connection with the cloud has also worked flawlessly in all our testing as long as the phone had a decent Internet connection.

The wearables also worked flawlessly in our testing. Both the shoes and the belt were able to detect obstacles very well and provide feedback to the user. We even tested our prototype in a rural area as well as urban streets where it functioned quite reliably.

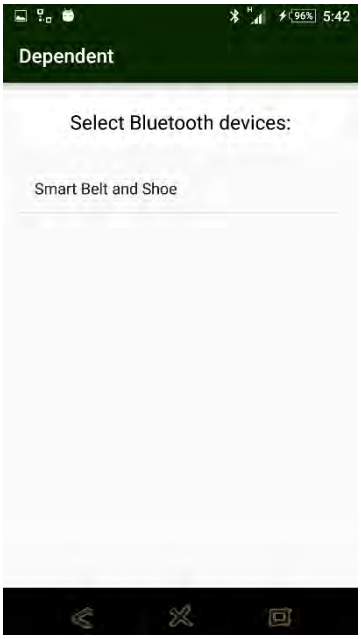
More Screenshots of the app are given on the following pages.

Asking for permission to turn on Bluetooth on user's device if it is not already turned on



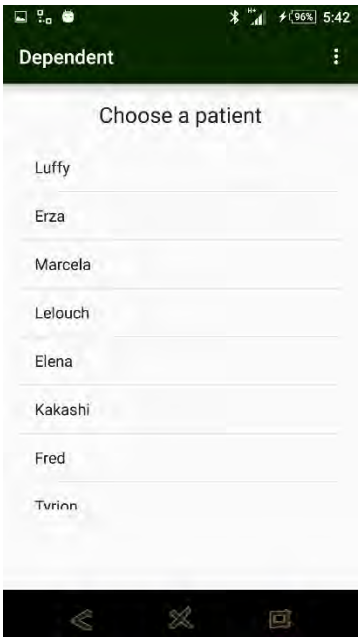
If the shoe and the belt are not paired to the device, the app displays this message

If the shoes and belt are not turned on

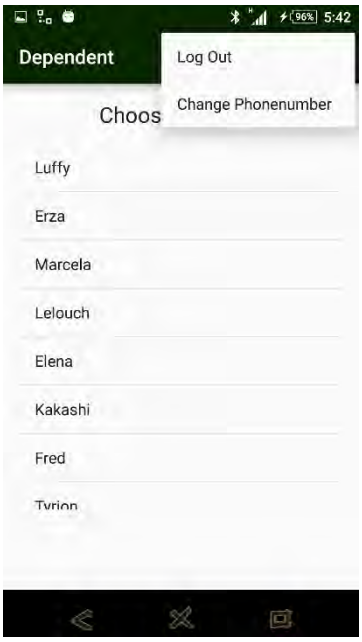


If both the shoes and the belt are turned on

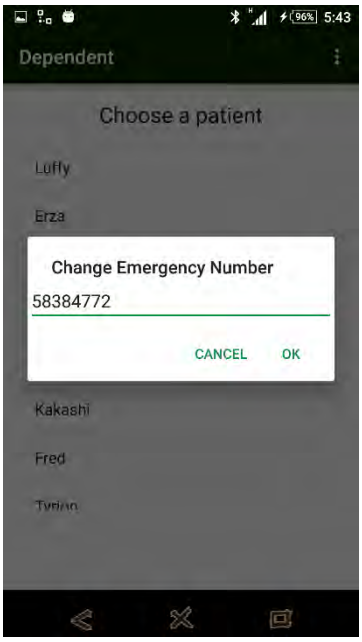
Figure 58 Checking Bluetooth Connections of the Dependent App



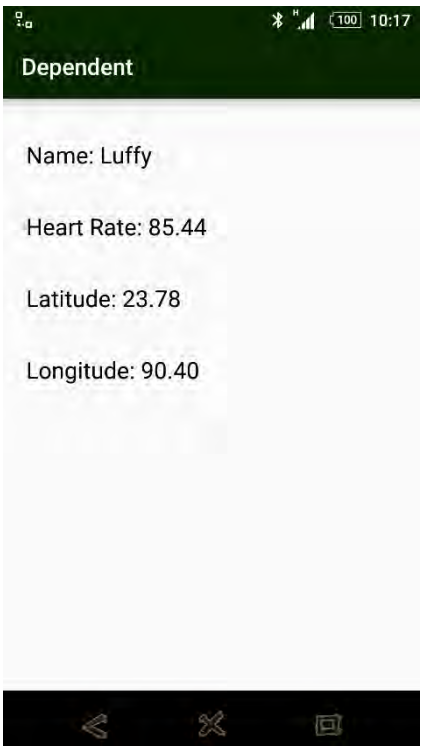
Dashboard from where Guardian can select the dependent who will be using the device



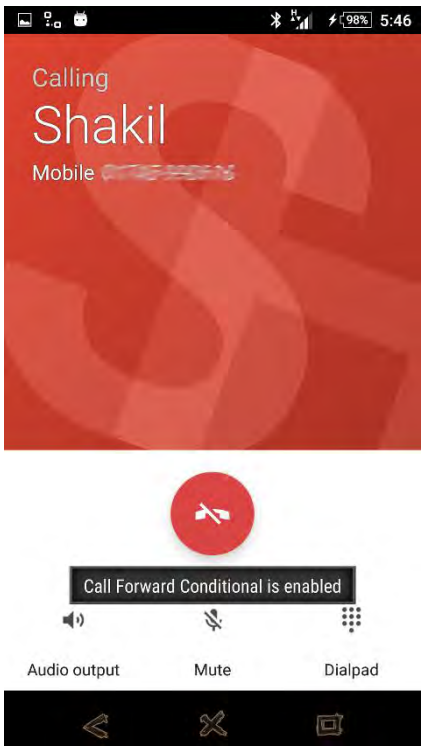
When the action bar button is pressed



Changing emergency phone number



Displaying data read from heart rate sensor and GPS of the phone



Phone call being made when the emergency button is pressed

Figure 59 Checking Connection with Azure of the Dependent App

Battery Consumption

Both the app's and the devices' power consumption have been tested. We have used standard 9V battery packs for the devices. Initially the belt was tested for 3 hours when it was assembled and the Arduino Uno in it was able draw power from the battery without any difficulty. Afterwards it was made to stay on for 11 hours, among which it was used for 6 hours. The same procedure was followed for the shoes. Both shoes were assembled and initially tested with new 9V batteries for about 2 hours. Later after 6 hours of use and a combined 12 hours of keeping on, the right shoe was observed to have consumed all battery power in 9 hours and the left shoe went through for 3 more hours. The right shoe and belt consumed more power as the right shoe contains two Arduino boards, one of which has a Bluetooth connection and the belt's Arduino board also contains a Bluetooth connection dedicated to send a signal from the push button to the phone.

To check on the feasibility of incorporating the prototypes in day to day use we have used 9V, 250mAh rechargeable batteries for the devices. These served the devices for a combined average of 3 hours of use till they required charging. Testing was further done with 5V 20,000mAh power bank, to establish the possibility of a more reliable and sustainable source of power. This testing was possible as Arduino boards have a USB connection in their design which can serve as a power input.

We have tested the app's power consumption by running the Monitoring Service of the dependent app continuously for about 6 hours and observing drops in battery level. During this period we used a brand new sim card on the phone and so during the testing we received no phone calls. We also did not run any other app. After about six hours we found the battery to drop from 100% to 78% with our app consuming 1% of the charge.

This shows that our app is very battery efficient. The screenshot after this test is given below.

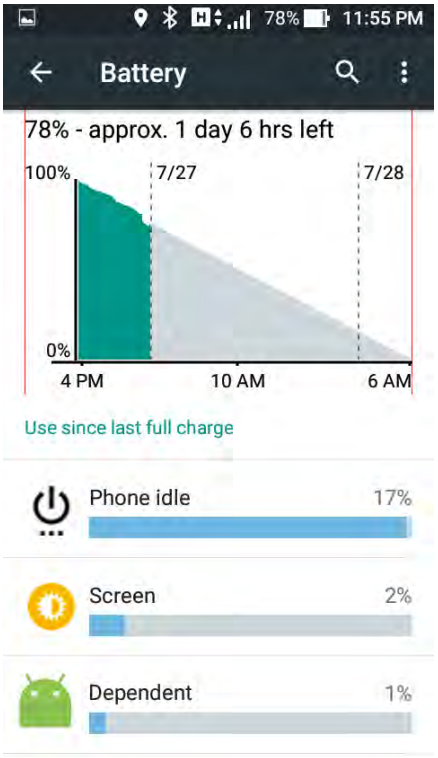


Figure 60 Battery Status after running our app for about six hours continuously

We did all our testing on an Asus Zenfone Go, which has a 1540mA battery.

Data Consumption

After continuously using the app for about six hours the dependent app consumed only 3.24MB data. So in a month where the user uses the app for 10 hours every day our app would consume about 162MB of data which really isn't much for an app that is continuously connected to the Internet. So this shows that our app is also very data efficient which is very important for a country like Bangladesh where Internet data is expensive.

Screenshot after the test is provided below.

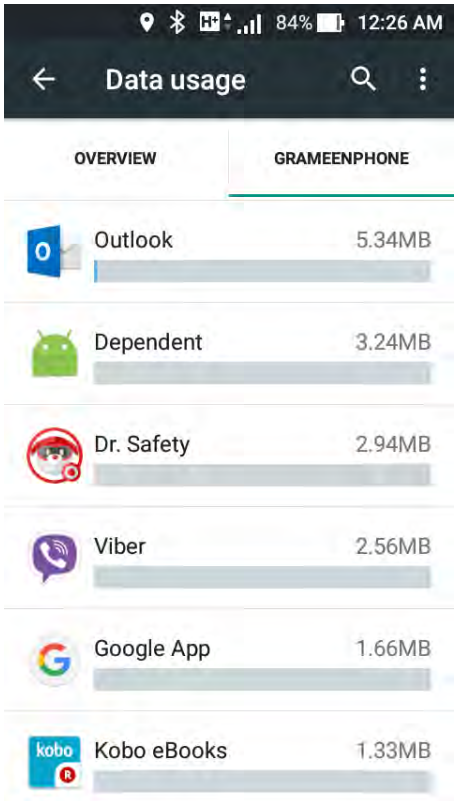


Figure 61 Data usage after running our app for about six hours continuously

We did all our testing on an Asus Zenfone Go, on Grameen Phone 3G data.

6.3 Performance analysis of guardian app

We have made the guardian app also go through rigorous testing to ensure the system works reliably. The following subsections discuss about several performance analysis about different aspects of the system that we made.

Authentication

Just like the dependent app authentication via Google OAuth worked smoothly, effectively and reliably in all our testing in the guardian app as well.

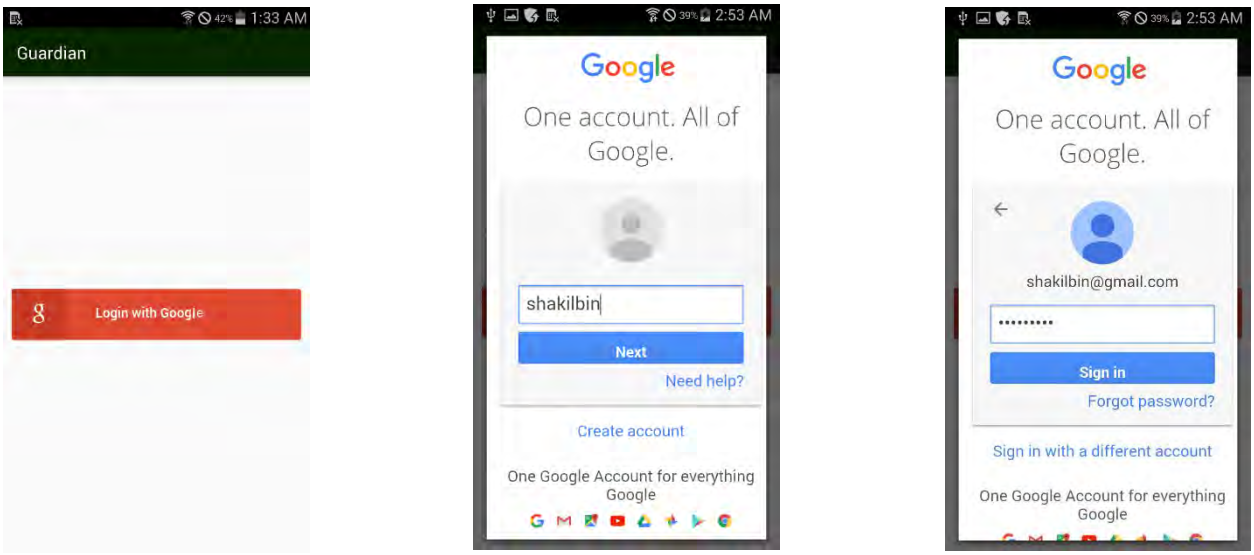
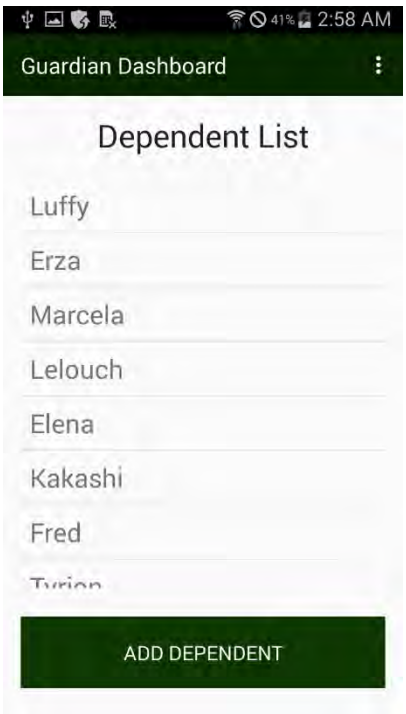


Figure 62 Authentication

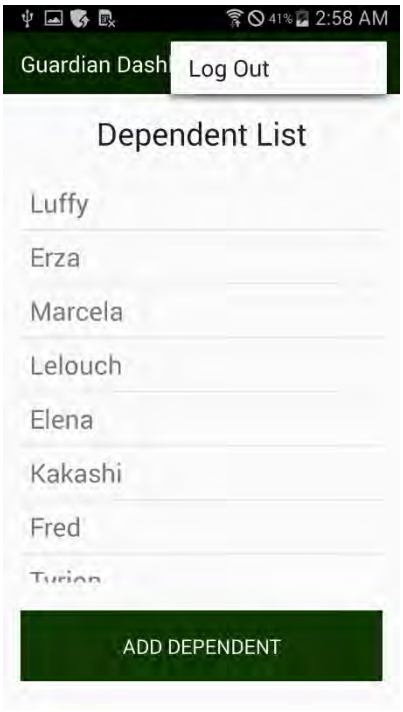
Connection with the Cloud via the Internet

In all our testing with a Samsung Galaxy S3, connection with the cloud has worked flawlessly in all our testing as long as the phone had a decent Internet connection.

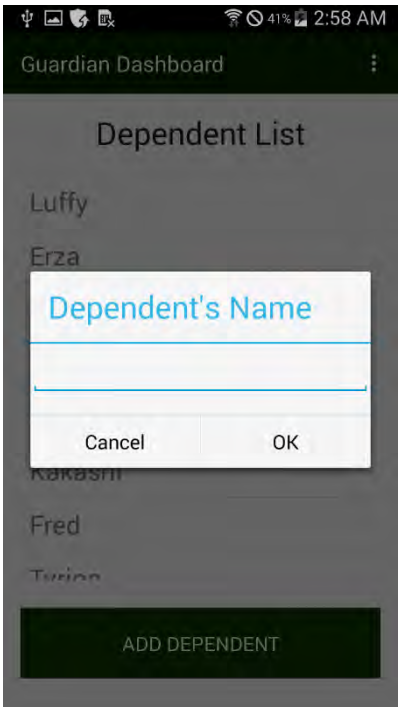
More Screenshots of the guardian app are below.



Guardian Dashboard loads all dependents from the cloud reliably on all tests

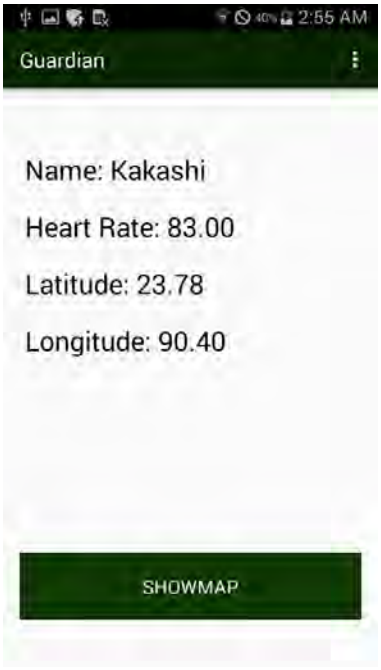


Log Out function worked properly on all tests

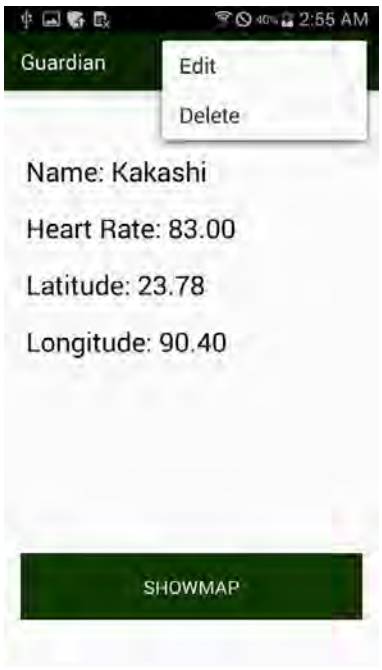


Add dependent function worked on all tests

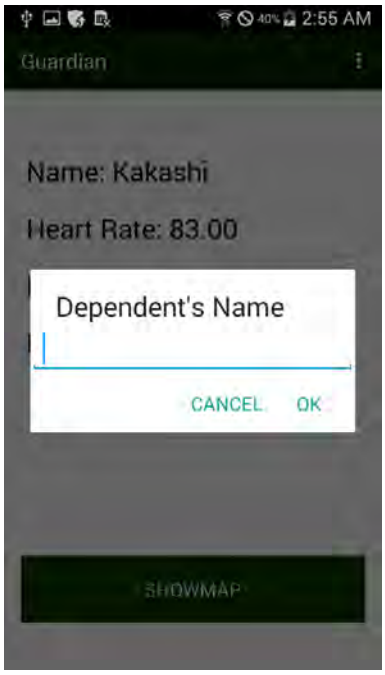
Figure 63 Checking Connection with Azure of the Guardian App



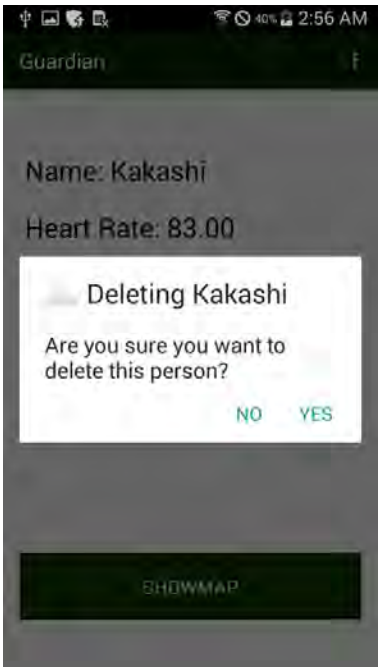
The app is able to load data properly from the cloud and display them to the guardian



The action bare button opens up options to edit or delete the current dependent

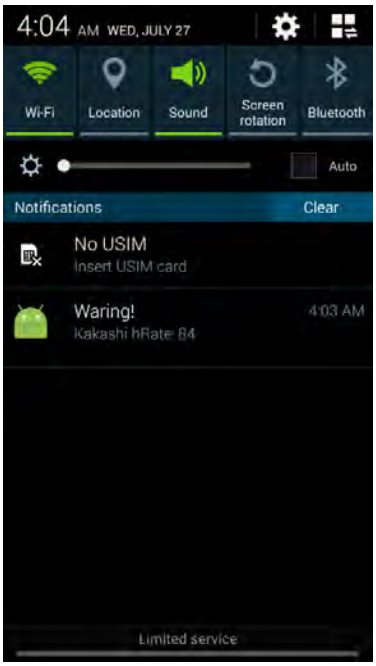


The edit function worked flawlessly in all our tests



The delete function worked flawlessly in all our tests

The show map function worked flawlessly in all our tests



The guardian's phone received notifications in all our tests when the emergency button was pushed or the heart rate of the dependent became too high as long as the guardians phone was connected to the Internet. The response was always almost instant.

Figure 64 Further Checking of Functionality of the Guardian App

Data and Power Consumption

We have tested the battery consumption of this app by running it throughout a day, checking the condition of the test patients every hour. During this time we did not use any other apps.

By the end of the day the percent of battery consumed by our app was still negligible. To further test this, we ran the same test for another entire day, but instead of running the Guardian App every hour, we simply unlocked the screen. The battery usage on both days was almost the same. On The day we used the app the battery level dropped to 66% and on the second day, without the app running it dropped to 64%.On both days we were connected to the Wi-Fi continuously. Both tests were carried out on the same Samsung Galaxy S3 device. This shows that the effect of our app on the charge level is negligible and the variation in the charge is due to other random factors.

At the end of the day when we tested the Guardian App, we checked the data usage by the app.

It had only used 1.38MB. Which means that, on a monthly basis the app will use 41.4MB in total, assuming that the user checks his/her patients using the app every hour.

Chapter 7 Conclusion

With the recent widespread availability of the Internet and affordable prices of smartphones in Bangladesh, it has become possible to combine these technologies with wearable computing devices to give life to our project in an efficient and cost-effective fashion. With this system, blind people can move more efficiently and independently.

The shoe and belt will work together to alert the blind person of incoming obstacles and holes in real time, thus enabling him/her to read his environment faster and make him/her more agile.

The people responsible for them do not constantly need to be in close proximity as our system lets the guardians monitor and locate their dependents from anywhere. The blind person can also feel safer while travelling as they know that their guardian is only a push of a button away. One button press, and the person's guardian will be notified of their dependents condition and location and will also immediately get a call from the blind person.

Even if the blind person is in a state of extreme distress and forgets to or cannot press the emergency button on their belt, our system will notify the person responsible immediately as the blind person's heart rate reaches dangerous levels. And there is hardly a chance for us to miss such an event as we are reading the blind persons heart rate every 5 seconds.

Thus both the guardian and the blind person can be more at ease when the blind person is travelling alone. Both the guardian and their dependent have more freedom of movement and peace of mind.

Chapter 8 Future Works

The scope of this system can be greatly expanded by further modifying it. For Example, our solution can be modified for use in a hospitals so that the doctor or nurse can keep track of their patients when the patient is moving inside or outside of the hospital. Thus eliminating the need for someone to constantly be with the patient and giving the patient more freedom. The system can be expanded so that more of the wearer's vitals are recorded to be sent to the doctor or nurse.

Furthermore we can also add another button to the belt to launch Google now, thereby allowing the blind person to access the voice commands more easily

However our main aim for the future will be to design our own PCB, and make the shoes and belt light weight and comfortable to wear while giving the user audio feedback about their environment along with vibrations.

References

[1] Anonymous. Visual Impairment and Blindness. Retrieved from http://www.who.int/mediacentre/factsheets/fs282/en/

[2] Alech, A. (28 September, 2010). Bangladesh fights to end blindness. *The Guardian.* Retrieved from: https://www.theguardian.com/world/2010/sep/28/bangladesh-volunteers-childhood-blindness-treatment

[3] Abu-Faraj, Z.O., Jabbour, E., Ibrahim, P. & Ghaoui. A. (2012*). Design and Development of a Prototype Rehabilitative Shoes and Spectacles for the Blind*. Paper presented at the 5th International Conference on BioMedical Engineering and Informatics, Chongqing, China. doi: 10.1109/BMEI.2012.6513135

[4] Yuan, D. & Manduchi, R. (2005). *Dynamic Environment Exploration Using a Virtual White Cane*. Paper presented at 2005 IEEE Computer Society Conference on Computer Vision and Pattern recognition (CVPR'05). doi: 10.1109/CVPR.2005.136

[5] Cardin, S., Thalmann, D., Vexo, F. (n.d). *Wearable Obstacle Detection System for visually impaired people.* Ecole Polytechnique Fédérale de Lausanne (EFPL) CH-1015 Lausanne, Switzerland, p.p 1-6.

[6] Borenstein, J. (1990). *The NavBelt- A Computerized Multi-Sensor Travel Aid for Active Guidance of the Blind.* Paper presented at the CSUN's Fifth Annual Conference on Technology and Persons with Disabilities, Los Angeles, California.

[7] Baranski, P., Polanczyk, M., Strumillo, P. (2010). Paper presented at 2010 12th IEEE International Conference on e-Health Networking Applications and Services, Lyon, France. doi: 10.1109/HEALTH.2010.5556539

[8] Al-Fahoum, A. A., Al-Hmoud, H. B., Al-Fraihat, A. A. (2013). A Smart Infrared Microcontroller-Based Blind Guidance System. *Active and Passive Electronic Components*. http://dx.doi.org/10.115/2013/726480

[9] Arduino. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Arduino

[10] Node.js. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Node.js

[11] Gailey, G. (n.d.). Mobile Services Concept. Retrieved from https://azure.microsoft.com/en-us/documentation/articles/mobile-services-concepts-links/

[12] Microsoft Azure. (n.d.). Retrieved from https://azure.microsoft.com/en-us/overview/what-is-azure/

[13] Microsoft Azure. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Microsoft_Azure#Timeline

[14] Sverdlik, Y. (7 June 2016). Top Cloud Providers Made $11B on Iaas in 2015, but It's Only the Beginning. Retrieved from http://www.datacenterknowledge.com/archives/2016/06/07/top-cloud-providers-made-11b-on-iaas-in-2015-but-its-only-the-beginning/

[15] Git. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Git_(software)

[16] Android Studio. (n.d.). Retrieved from https://en.wikipedia.org/wiki/Android_Studio

[17] Pulse Sensor. (n.d.). Retrieved from http://pulsesensor.com/pages/open-hardware

[18] Serial Port Bluetooth Module (Master/Slave): HC-05. (n.d.). Retrieved from https://www.itead.cc/wiki/Serial_Port_Bluetooth_Module_(Master/Slave)_:_HC-05

[19] Pujar, R. (31 October 2014). Learn how a heart beat sensor works. Retrieved from http://www.raviyp.com/embedded/140-learn-how-a-heart-beat-sensor-works

[20] Ultrasonic Ranging Module HC-SR04. [pdf datasheet on web]. Retrieved from http://www.micropik.com/PDF/HCSR04.pdf

[21] Dorsalis pedis arterial puncture site. (n.d.). Retrieved from http://s0www.utdlab.com/contents/image.do?imageKey=EM%2F74882

[22] Coin Vibration Motors. (n.d.). Retrieved from https://www.precisionmicrodrives.com/vibration-motors/coin-vibration-motors