**The Breathalyzer**

Submitted By: Maheshwerie Samaroo, Karandeep Singh and Mohita Prabhakar  
Submitted To: Humber College  
205 Humber College Boulevard,   
Etobicoke, Ontario, M9W5L7  
Discipline: Computer Engineering Technology  
Date Submitted: April 23rd, 2018

# 

# Declaration of Joint Authorship

The team “MKM” which consists of Maheshwerie Samaroo, Karandeep Singh and Mohita Prabhakar confirms that the project of the Breathalyzer is a combined group effort and is a combination of our own thoughts and ideas. The work of this entire project was split as equally as possible. Karandeep Singh is working with calibrating the sensors and was in charge of hardware design and helped with the app layouts. Maheshwerie Samaroo is in charge of mobile application design and maintenance and is handling the database. Mohita Prabhakar is working with the database in terms of setting it up, connecting it with the app and maintaining it. The distribution of testing the hardware and software for bugs and issues will be discussed by the three of us and worked on all together. Before any changes are made on the project, a group consensus has to be made. Links to articles/websites that are being used for guidance and help has been placed in the reference area of this report.

# Approved Proposal

5 February 2018

Proposal for the development of Breathalyzer

Prepared by Karandeep Singh, Maheshwerie Samaroo, Mohita Prabhakar  
Computer Engineering Technology Students  
https://github.com/N01150244/pulsesensor

Executive Summary

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with the following sensors and actuators Heart Rate Educational Starter Kit, MQ3 Alcohol Sensor and Pulse Sensor. The database will store Readings from the heart rate/pulse sensors and MQ3 Alcohol Sensor. The mobile device functionality will include Allowing a user to sign-up or login to an existing account, view their current results as well as past results, call emergency contacts and call an UBER cab (if needed). and will be further detailed in the mobile application proposal. We will be collaborating with the following company/department Prototype lab, Humber Parts Crib and Humber Tech Group. In the winter semester we formed a group, who are also building similar hardware this term and working on the mobile application with group members Karandeep Singh, Mohita Prabhakar and Maheshwerie Samaroo. The hardware will be completed in CENG 317 Hardware

Production Techniques independently and the application was in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

Background

The problem solved by this project is We all want a solution to DUI (Driving under the Influence) or to at least reduce the number of deaths caused by it. Today, we see too many accidents being caused by DUI. Innocent lives are lost due to a drunk driver. Families are emotionally/financially stressed due to the loss of a loved one. We, the MKM Developers, intend to use our project, "The Breathalyzer", to help reduce these occurrences as well as give users a general idea on their BAC (Blood Alcohol Content) and pulse rate prior to them getting behind the wheel of a car.. A bit of background about this topic is The primary focus of our project is to address the issue of DUI (Driving under the Influence). Our target audience is the general public. The main reason for the development of our product, is due to the fact that many lives are being lost as a result of DUI (Driving under the Influence). With our product, we intend to combat this issue and reduce the statistics. We intend to reduce the number of deaths and prevent occurrences of DUI (Driving under the Influence). It is stated that the target audience is the general public. What this means is that, anyone can use this product. This product can be taken along with someone who decides to go to the Bar or to the Club or to any event that involves alcohol consumption. The project involves the integration of the mobile application which was built along with the hardware component. Basically the user will be required to blow into the alcohol sensor as well as use the provided pulse sensor. The sensors will capture the readings which will then be

pushed to a database. The mobile application will then pull the data from the database and display the readings via the application. Once the user is above the legal limit, he/she has the option to either call an Emergency Contact or request an UBER cab. The desired outcome of the overall project would be to assist in reducing DUI (Driving Under the Influence) occurrences..

Existing products on the market include [1]. I have searched for prior art via Humber’s IEEE subscription selecting “My Subscribed Content” [2] and have found and read [3] which provides insight into similar efforts.

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

* Java Docs from CENG 212 Programming Techniques In Java,
* Construction of circuits from CENG 215 Digital And Interfacing Systems,
* Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
* Micro computing from CENG 252 Embedded Systems,
* SQL from CENG 254 Database With Java,
* Web access of databases from CENG 256 Internet Scripting; and,
* Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:  
 Phase 1 Hardware build.  
 Phase 2 System integration.  
 Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16 Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The brief description below provides rough effort and non-labor estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule

details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Raspberry Pi 3 starter kit

XD-58C Sensor from Sparkykit heart rate sensor,MQ3 Alcohol Gas Sensor

Jumper Wires(Male-Female, Female-Female, Male-Male Jumper Wire Cables)

Heart Rate Educational Starter Kit

Concluding remarks

This proposal presents a plan for providing an IoT solution for This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative project which will lower the rate of alcohol consumption and bring us closer to a solution to prevent drinking and driving.

I request approval of this project.. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

References

[1] Heartbeats in Your Project, Lickety-Split ♥. (n.d.). Retrieved February 02, 2018, from https://pulsesensor.com/

#237238, M., #321089, M., #661774, M., O., #721982, M., #554862, M., . . . G. (n.d.). Pulse Sensor. Retrieved February 02, 2018, from https://www.sparkfun.com/products/11574

#637052, M., & O. (n.d.). Alcohol Gas Sensor - MQ-3. Retrieved February 02, 2018, from https://www.sparkfun.com/products/8880

Industries, A. (n.d.). Heart Rate Educational Starter Pack with Polar Wireless Sensors. Retrieved February 04, 2018, from https://www.adafruit.com/product/1077

[2] Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

[3] Wang, X., Jin, J., & Li, S. (2008, September 03). Measurement and analysis of heart signal based on the pressure sensor. Retrieved February 04, 2018, from http://ieeexplore.ieee.org/document/4618175/

Malathi, M., Sujitha, R., & Revathy, M. R. (2018, February 01). Alcohol detection and seat belt control system using Arduino. Retrieved February 04, 2018, from http://ieeexplore.ieee.org/document/8275841/

Kirtana, R. N., & Lokeswari, Y. V. (2017, June 08). An IoT based remote HRV monitoring system for hypertensive patients. Retrieved February 04, 2018, from http://ieeexplore.ieee.org/document/7944086/

# Abstract

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills which we have learned from our program into this Internet of Things themed capstone project. The internet connected hardware includes a custom PCB with the following sensors: Heart Rate Educational Starter Kit, MQ3 Alcohol Sensor and Pulse Sensor. The database stores readings from the heart rate/pulse sensors and MQ3 Alcohol Sensor. The mobile device functionality includes allowing a user to sign-up or login to an existing account, view their current results as well as past results, call emergency contacts and call an UBER cab (if needed) and was further detailed in the mobile application requirements section. We will be collaborating with the following company/department, School of Hospitality, Recreation and Tourism (HRT).

# Table of Contents

[Declaration of Joint Authorship 3](#_Toc512158555)

[Approved Proposal 5](#_Toc512158556)

[Abstract 17](#_Toc512158557)

[Table of Contents 19](#_Toc512158558)

[1. Introduction 23](#_Toc512158559)

[2. Project Description 25](#_Toc512158560)

[2.1 Product Perspective 25](#_Toc512158561)

[2.2 Product Functions 27](#_Toc512158562)

[2.3 User Classes and Characteristics 33](#_Toc512158563)

[2.4 Operating Environment 33](#_Toc512158564)

[2.5 Design and Implementation Constraints 33](#_Toc512158565)

[2.6 User Documentation 35](#_Toc512158566)

[2.7 Assumptions and Dependencies 35](#_Toc512158567)

[3. Mobile Application Requirements 37](#_Toc512158568)

[4. Hardware Requirements 39](#_Toc512158569)

[5. Build Instructions 41](#_Toc512158570)

[5.1 Introduction 41](#_Toc512158571)

[5.2 Bill of Materials 43](#_Toc512158572)

[5.3 Time Commitment 47](#_Toc512158573)

[5.4 Mechanical Assembly 51](#_Toc512158574)

[5.4.1 Raspberry Pi Setup 51](#_Toc512158575)

[5.4.2 PCB Soldering 53](#_Toc512158576)

[5.4.3 Assembling the Project 55](#_Toc512158577)

[5.4.4 Power Up 57](#_Toc512158578)

[5.4.5 Connecting To The Database 59](#_Toc512158579)

[5.4.6 Unit Testing 63](#_Toc512158580)

[5.4.7 Production Testing 63](#_Toc512158581)

[5.4.8 Reproducible? 65](#_Toc512158582)

[5.5 Software Application 67](#_Toc512158583)

[5.5.1 Overview 67](#_Toc512158584)

[5.5.2 Time Commitment 67](#_Toc512158585)

[5.5.3 Detailed Breakdown of Software 67](#_Toc512158586)

[5.5.3.1 Login Pages and Connections to Firebase 67](#_Toc512158587)

[5.5.3.2 Main Page (Breathalyzer Test) 71](#_Toc512158588)

[5.5.3.3 Navigation Drawer and Fragments 73](#_Toc512158589)

[5.5.3.4 Viewing Current Results (Fragment) 75](#_Toc512158590)

[5.5.3.5 Viewing Past Results (Fragment) 79](#_Toc512158591)

[5.5.4 Design Specification 81](#_Toc512158592)

[5.5.4.1 Introduction 81](#_Toc512158593)

[5.5.4.2 Overview 81](#_Toc512158594)

[5.5.4.3 Scenarios for the Usage of the Application 83](#_Toc512158595)

[5.5.4.4 Non-Goals 87](#_Toc512158596)

[5.5.4.5 Screen by Screen Specification 89](#_Toc512158597)

[6. Schedule & Progress Reports 97](#_Toc512158598)

[6.1 Fall Semester 2017 (Phase 1 - Hardware) 97](#_Toc512158599)

[6.2 Winter Semester 2018 (Phase 2 - Hardware) 103](#_Toc512158600)

[6.3 Progress Report A (Week 5 Status Report A - March 5, 2018) 111](#_Toc512158601)

[6.4 Progress Report B ( Week 8 Status Report B (CENG 355) - MKM Developers March 19,2018) 115](#_Toc512158602)

[6.5 Progress Report C ( Week 9 CENG 355 0NB Status Report C - MKM Breathalyzer April 2,2018) 117](#_Toc512158603)

[6.6 Progress made after submitting the three progress reports: 121](#_Toc512158604)

[6.7 Fall Semester 2017 (Phase 1 - Software - Pre Strike) 123](#_Toc512158605)

[6.8 Fall Semester 2017 (Phase 1 - Software - Post Strike) 129](#_Toc512158606)

[6.9 CENG 319 (Software Project) Milestone 1 135](#_Toc512158607)

[7. Challenges 141](#_Toc512158608)

[8. Conclusion 155](#_Toc512158609)

[9. Recommendations 157](#_Toc512158610)

[10. Bibliography 159](#_Toc512158611)

[11. Appendices 163](#_Toc512158612)

[11.1 Python Code for Raspberry Pi 163](#_Toc512158613)

[11.2 Arduino Code for Arduino Mega 2560 171](#_Toc512158614)

# 1. Introduction

This report is a description of the processes followed for the development of a breathalyzer device. The targeted audience for this device will be the general public.

We all want a solution to DUI (Driving under the Influence) or to at least reduce the number of deaths caused by it. Today, we see too many accidents being caused by DUI. Innocent lives are lost due to drunk drivers. Families are emotionally/financially stressed due to the loss of a loved one.The purpose of our Breathalyzer project is to make sure people are off the roads if they are intoxicated or not safe to drive. Drinking and Driving is a huge issue in our world today, and by using this technology to check if there is alcohol is someone’s body, we can keep lives out of danger. We, the MKM Developers, intend to use our project, "The Breathalyzer", to help reduce these occurrences as well as give users a general idea on their BAC (Blood Alcohol Content) and pulse rate prior to them getting behind the wheel of a car.

|  |
| --- |
|  |

# 2. Project Description

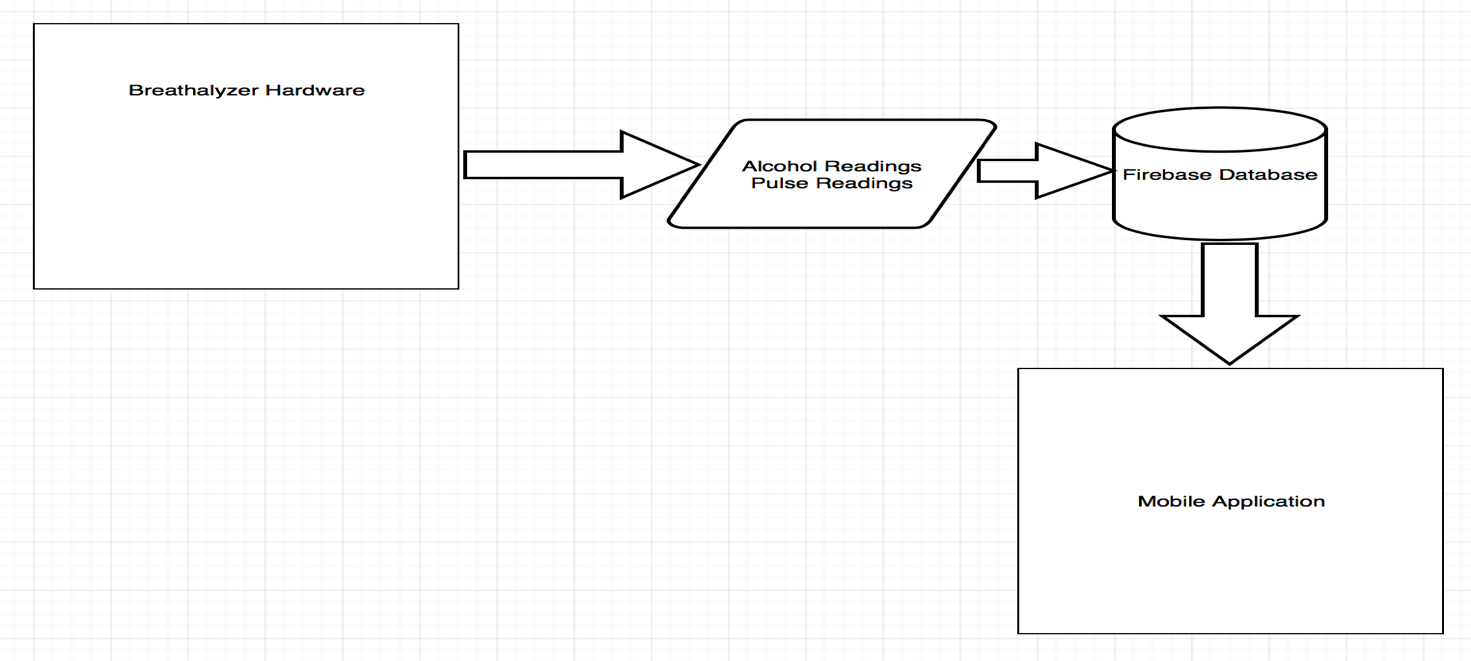
The work described in this technical report was undertaken because we wanted to try to create a solution where you can track your Blood Alcohol Concentration (BAC) as well as your heart rate and present the user with options depending on what range their BAC falls under. Our breathalyzer connects to an android phone where our mobile app displays readings and stores them into a database providing past history recorded to lookup in the future.

## 2.1 Product Perspective

This project encompasses the development of a new, self-contained product, “The Breathalyzer”. The product which is in development, requires software and hardware in order to perform its desired function. The software is in the form of a simple mobile application which is currently live on the Google Play Store as a simulator. The application serves as a display or medium through which the user can view the end result of using the entire product. For our purposes of explanation, we will refer to the Hardware as the “Larger System” and the Software as the “Smaller System”. The Larger System consists of three sensors which all collects/measures data which is then pushed to a Firebase Database. The function of the Smaller System is to display this data and give the user some additional options as well (as shown in the case below). Case 1: John goes to the bar with some friends to have some drinks. It is now time to go home and John decides to take a Breathalyzer test to determine what his next decision should be. John takes the test by blowing into his personal Breathalyzer and the results show up on his mobile application. John is not safe to drive; he is over the legal limit. John now has the option to either call for an UBER cab or to Call two of his

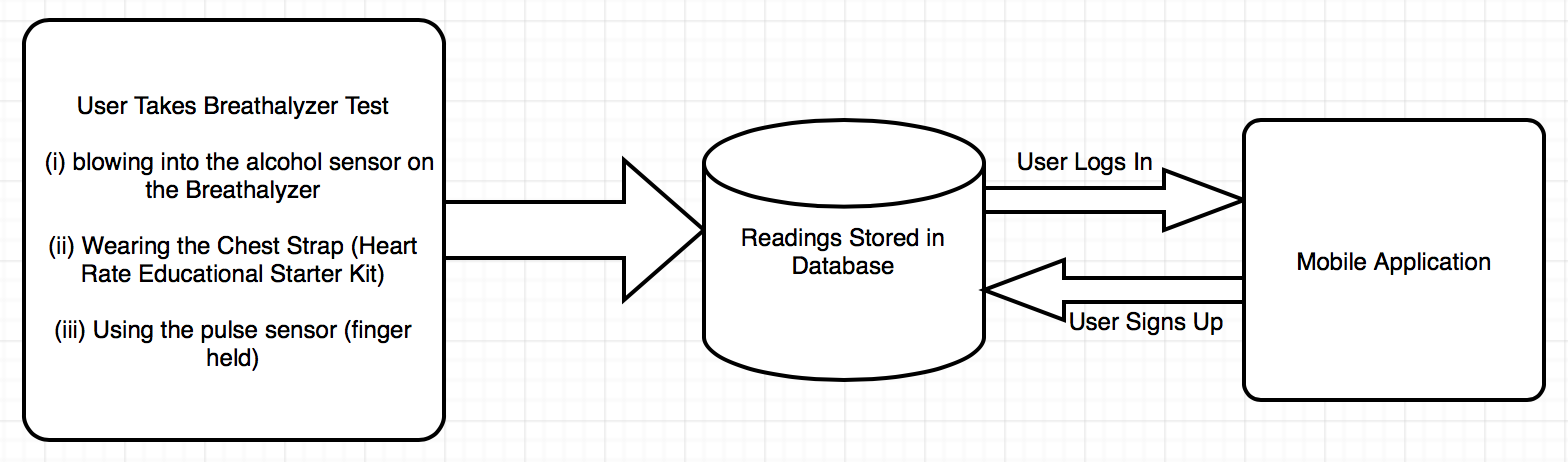
emergency contacts which he himself had stored previously. John calls a cab and gets home safely.

High Level Diagram Showing Data Flow



## 2.2 Product Functions

* Mobile Application – For our project, there will be a mobile android application which will be able to show data collected from our Breathalyzer. Once the application is open the User must login to his/her account (one time login unless logged out)to be able to see past results which were read from our two sensors on our hardware, MQ-3 Gas Sensor and XD-58C.Without logging in/signing up, the past results button would be hidden, which leaves a button to start the alcohol test or heartbeat test. After these tests, you will be brought to a results page that presents the readings. If the reading from the alcohol test detects alcohol in your body, it will give you options to take such as calling a friend, calling a taxi. The last option you would have is to look at past results (if you are logged in). This is where the data is taken from the database. The user will be able to look results stored in the database which will show them their past BAC
* and BPM readings that were done previously, along with their average BAC and BPM readings. If you are logged in, once you run a test it would be added and updated to the database.
* Database – We will be using Google’s Firebase mobile and web application platform for user authentication on logins and our real-time database. Users will be able to sign up for an account and login to access their personal previous alcohol and heart rate results, as well as the averages for both blood alcohol content and heart rate. The database will be a JSON database which will contain the user’s unique ID (UID) as well as user information that is taken from when the user signs up for an account. Their UID is automatically generated by Firebase Auth when they sign up for an account. When they perform an alcohol test or heart rate test, the resulting data is stored under their UID. The data is then fetched and displayed showing the past results when they navigate to the past results page in the mobile application.
* Hardware – The hardware portion of our project is a breathalyzer. This breathalyzer will be used to help eliminate drunk driving by giving the user the ability to test their blood alcohol content (BAC) level as well as their beat per minute (BPM). The device tells the user if it is safe to drive home or gives the user alternative options if their unable to drive. The user must provide a breath sample (by blowing into the alcohol sensor of the breathalyzer) as well as taking two pulse tests.(one will be done using a chest strap and the other done by using a sensor which needs to be held by the user’s fingers). The sensors are connected to a Raspberry Pi that runs a program that gathers a reading from the
* sensor. The device itself is inside a small acrylic case that holds the contents inside as well as a small rechargeable battery to make the device portable. Data from the sensor are also sent to a connected device and displayed on an application as well as sent to a database.



## 2.3 User Classes and Characteristics

This product is meant to be used by the general public. Basically anyone from the age of 13 up is supposed to be able to use our mobile application and device. (The legal age for drinking alcohol in the province of Ontario is 19 years old and may vary from province to province. The age used in the statement above is just for reference purposes.)

## 2.4 Operating Environment

Raspberry Pi 3 with Raspbian along with Android mobile application, Firebase Database are the components of operating environment for the project.

## 2.5 Design and Implementation Constraints

With this project, there’s some implementation constraints. The mobile application for example, requires an active and steady internet connection in order for results to be pushed from the hardware to the database. An implementation constraint is the

accuracy of the sensors. The accuracy of the sensors depends on whether or not they have been properly calibrated.

## 2.6 User Documentation

Along with this software user will be equipped with the detailed build instructions and a technical report which will be available online: <https://github.com/N01150244/pulsesensor>

The user can also consult and get assistance from all of the online links mentioned in the references of the technical report.

## 2.7 Assumptions and Dependencies

Both the mobile application and the web application requires internet connection since the firebase database gets updated in real time. The python script will be used to store data in the database and can be retrieved on the application side via a php script.

# 3. Mobile Application Requirements

This product comes with a mobile application with which the user interacts. A more detailed breakdown will be provided within the 5.5 Software Application. Upon launching the application the user will be presented with a Login Screen with the option to create a user account if an account is not already existing. When the user logs in, he/she will be presented with a web view which shows the current news articles related to DUI (Driving Under the Influence). There is also a Navigation Drawer from which the User can choose one out of four options:

1. Start Test
2. View Past Results
3. Call a friend
4. Call Uber

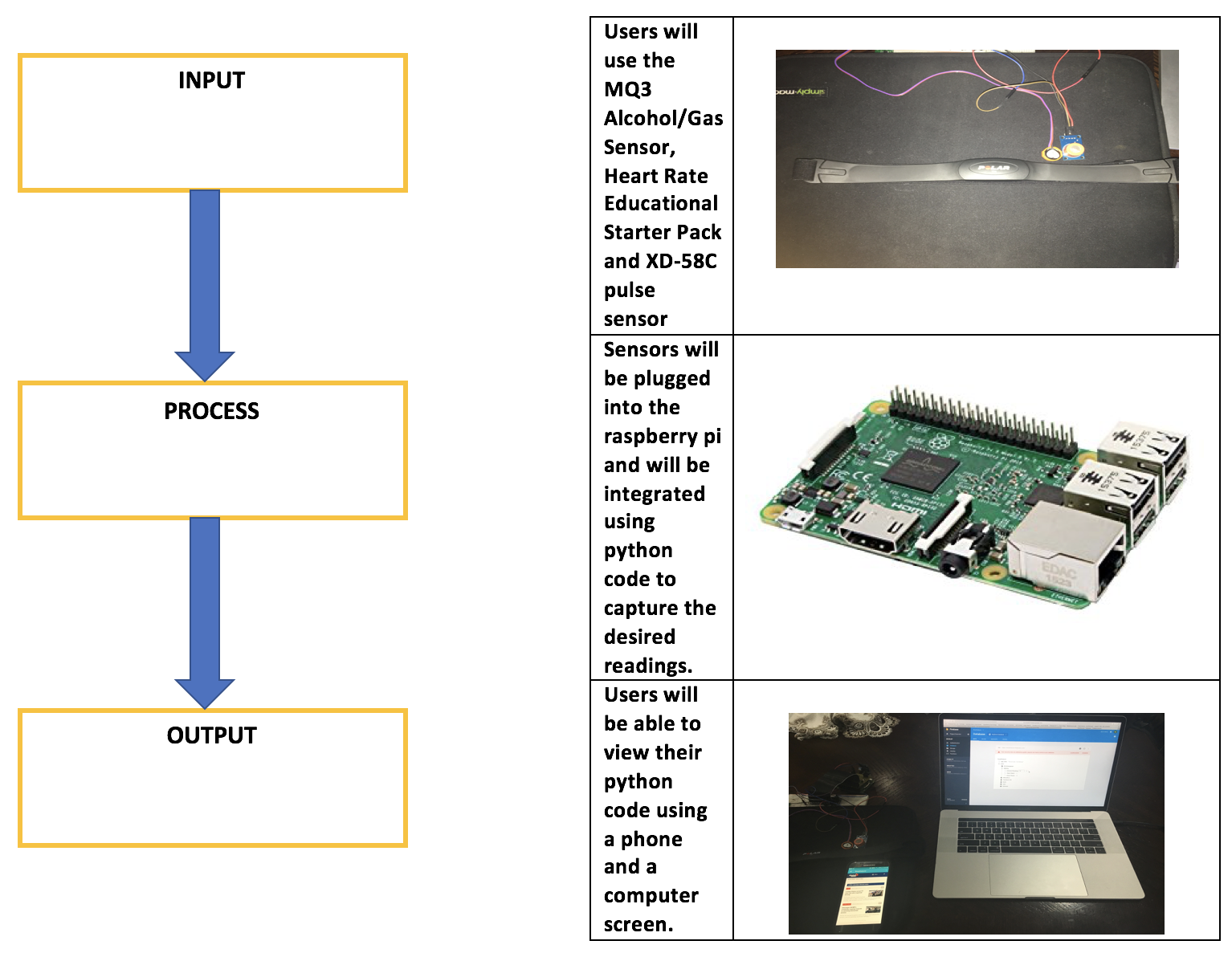
# 4. Hardware Requirements

The Raspberry Pi software can be used through a LCD screen, a mouse and a keyboard. The screen is used to display sensor results as well. When results are transferred on to the application, these can be viewed via a mobile device using Android 5.0 or higher. This project makes use of three sensors. The first sensor is the MQ3 Alcohol Sensor. Currently, this sensor is being worked on by Mohita Prabhakar. There is a python script with which this sensor works. Basically a user can blow into the MQ3 sensor and a reading will be displayed and pushed to a Database. The second sensor is the XD-58C pulse sensor. The user should be able to attach the sensor via a velcro strap around their finger and his/her pulse rate will be captured and pushed to a Firebase Database. The third sensor is the Polar Heart Rate Sensor which is commonly known as the “Heart Rate Educational Starter Pack”. In this project, there are two pulse sensors which improves accuracy of the readings. One sensor may be more accurate than the other.

# 5. Build Instructions

## 5.1 Introduction

This page is created to help you to create your own Heart rate sensor project. In order to fix the problem of drinking and driving, we chose the Heart Rate Educational Starter Kit, Alcohol Gas sensor and Pulse sensor. Instead of people assuming that they are okay to drive with alcohol in their body, they can use our project to see if there is a large amount of alcohol in their body along with seeing their heart rate to determine whether or not they are legally able to drive. It's a cheap alternative, and isn't too hard to make yourself. Using a gas and two heartbeat sensors, users can plug those into a PCB and use a raspberry pi to display the readings on either a computer or phone on our mobile application. You can have your very own Breathalyzer product just by following this section. The only thing that would keep you from doing it much faster would be the delivery dates on items. After putting in the hard work, you can use it to impress people with and hopefully use it yourself to keep you far away from drunk driving. You yourself can help reduce the problem of drinking and driving just by making this.



## 5.2 Bill of Materials

Given below are parts and material required for this project. Most of these parts are pretty cheap which makes this project not too expensive, but that is because we already had our respective raspberry Pis, electronic parts kit, and PCB kits which were paid for as a part of our tuition.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Quantity | Cost | Supplier & Part Number |
| Raspberry Pi 3 Starter Kit | 1 | $79.99 | (Canakit) Amazon - B01CCF9BYG |
| XD-58C Sensor | 1 | $18.99 | (JutaTech) Amazon.ca - B01AUVMFIS |
| MQ3 Alcohol/Gas Sensor | 1 | $8.80 | (Creatron) creatroninc.com - SPEDE-000030 |
| Heart Rate Educational Starter Pack | 1 | $65.00 | (AdaFruit) adafruit.com - 1077 |
| Electronics Parts Kit | 1 | $119.99 | Humber - SKU #163 |

|  |  |  |  |
| --- | --- | --- | --- |
| Jumper Wires (120 pack) | 1 | $19.99 | (Elegoo) Amazon.ca - B01EV70C78 |
| Solder Kit | 1 | ~$40.00 | Humber |
| Soldering Iron | 1 | ~$20.00 | Humber |
| Power Cables/Connectors | 1 | included | Humber/Amazon |

Again, these are just the parts and prices for the things we bought. Prices may change over time, but our total comes to around $360.

## 5.3 Time Commitment

If you work on this continuously with no other tasks in your way, it shouldn't take that long considering you do everything correctly! In this chart below, we break down how much time was taken on each main task of the project.

|  |  |
| --- | --- |
| Things you need to do | Time Taken To Complete (Approx.) |
| Looking for and Purchasing Parts + Delivery | 1.5 hours + 1 week |
| Assembling case and setting up Raspberry Pi | 1 hour |
| Editing your custom PCB | 30 minutes |
| Soldering/Testing/Troubleshooting your PCB | 3.5 hours |
| Creating a case for your project | 1 hour |
| Testing/calibrating the sensors | 2 hours |
| Setting up the project | 10 minutes |

After breaking down the parts of the project, it is pretty easy to tell that it's not a very time consuming project to complete. If you are very committed to this project then it shouldn't be very difficult to complete this in these time frames.

## 5.4 Mechanical Assembly

To keep things simple, for mechanical assembly, everything will be broken into sections. First, there will be a description on how to setup your raspberry pi, what is needed to be done in order for it to work properly. Next, steps to create your own PCB will be detailed. After that, steps to connect and power up will be detailed.

### 5.4.1 Raspberry Pi Setup

Once you’ve acquired your Raspberry Pi, the first thing to do would be to get a keyboard and a mouse connected. Please note that whenever connecting any new device or making any new connections to the Raspberry Pi, please ensure that you Pi is powered off completely. For the Pi that was used for this project, what was noticed, was that when connections were made while the pi was powered on, the pi would freeze and become unresponsive. The next thing to do would be to insert the SD card which comes with the Pi. There’s many ways in which to connect the Pi to a display but in this case, a HDMI cable was used to connect to a monitor. The final thing would be to plug the Pi in. A black screen will be displayed at first, which is fine and absolutely normal. You will need to run an update which can be done by issuing the “sudo apt-get update” command. Please note that your Raspberry Pi must have an active internet connection in order for this to work. You can purchase a Wi-Fi dongle separately for this or use a LAN cable. In this case, we had neither so using one of our mobile phones, we used a feature called “Mobile Tethering” which enables the phone to share its internet connection when it is connected via USB. The final step would be to download the python code needed for this project which can be found on the GitHub. (https://github.com/N01150244/pulsesensor/blob/master/Breathalyzer.py)

### 5.4.2 PCB Soldering

The next thing to start working on would be the PCB (Printed Circuit Board). The PCB which was provided by the Humber Prototype Lab is what was used to facilitate majority of the project. First you need to download the software EAGLE and then open the appropriate files. The schematic and board files were provided to us by our professor.

Assistance was given to us when needed by Kelly and Vlad who both work at the Prototype Lab. They were more than willing to assist whenever students needed it. Most of the soldering was done by looking at a reference model. Tools were provided by the lab, free of cost for the soldering of the PCB. Here’s what your PCB should look like when you’ve finished.



### 5.4.3 Assembling the Project

Once the Raspberry Pi is properly set-up, and your PCB is finished soldering, the rest of the project can now be assembled. The PCB needs to be “plugged in” directly to the raspberry pi. In this case, we plugged the PCB into the Raspberry Pi and then connected a pi cobbler plus breakout cable which we then connected to a breadboard. This makes it easier for you to make your connections. Note that it isn’t a requirement for you to buy the pi cobbler and breakout cable. The PCB must be connected to the header which is labelled “PFC-ADC”. Please remember that the Raspberry Pi must be completely powered off. The first sensor which we will speak about will be the XD-58C sensor (pulse rate sensor). The XD-58C sensor has three cables, which are colored purple, red and black. The red and black cables must be connected to 3.3V and GND respectively using the DTC header on the PCB. The purple cable must be connected to the AIN1 pin on the PCB. Please make sure that the Raspberry Pi is powered off completely before making these connections or else you will be at a risk of shorting a connection on the board. The second set of sensors which were used were the Polar T34 Heart Rate Transmitter and the Polar Heart Rate Receiver. Connecting the Heart Rate Receiver is simple.The third and final sensor which was used was the MQ3 Alcohol/Gas Sensor. The Raspberry pi is connected with the Mq3 sensor in such a way that VCC, D out and A out of the MQ3 Sensor is connected to the pin Vcc (4),Ground(6) and GPIO14(8) respectively as given below in the picture.

### 5.4.4 Power Up

Once all the connections have been made, the last thing left to do would be to plug the Raspberry Pi in. A quick check can be done to ensure that the sensors are in proper working order. The command “i2cdetect -y 1” can be issued in the terminal and 48 will be returned. The next step will be to run the code previously downloaded from the GitHub. First you must navigate to the location of the code via the Terminal and then issue the command “python Breathalyzer.py”. If the sensors are correctly connected, then the code should work perfectly.

|  |
| --- |
|  |

### 5.4.5 Connecting To The Database

Originally, when starting the project, our biggest question was how to push data from our hardware to the database and pull that in our web page and mobile application. We were thinking of using Shared Preferences for our mobile application or SQL. We had heard about Firebase but never thought about using it. We took the opinion of one of our professors from our Software Project class and he suggested that we use Firebase since it is relatively free and cheap to use.   
  
Luckily for us, Firebase has built in functionalities in Android Studio. Firebase works with certain rules for reading and writing data from and to the database which we had to specify.

Connecting firebase to our hardware wasn’t that difficult as well. We made use of Firebase’s REST API to make our connection. These are the steps to getting your raspberry pi setup for interaction with Firebase:

1. Install python-firebase using pip:

pip install -e git://github.com/mikexstudios/python-firebase.git#egg=python-firebase

1. Then simply import firebase at the top of your python script:

from firebase import Firebase

1. and then instantiate Firebase, passing in your root url:

f = Firebase('https://SampleChat.firebaseIO-demo.com/')

1. Now call the different methods of the Firebase class (see the Firebase REST API page: <http://www.firebase.com/docs/rest-api.html> and the source of firebase/\_\_init\_\_.py for what methods are available and how to call them). For example, to push a list of data:

f = Firebase('https://SampleChat.firebaseIO-demo.com/message\_list')

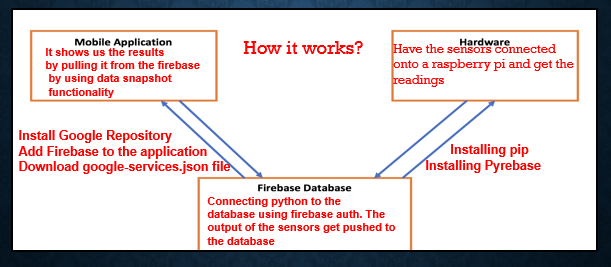
r = f.push({'user\_id': 'wilma', 'text': 'Hello'})

### 5.4.6 Unit Testing

For unit testing, we individually tested each sensor with individual pieces of code before combining into one piece of code. For the XD-58C sensor, the sensor should detect a heartbeat and it’s reading should be shown on the terminal screen. The same conditions are applicable to the other two sensors.

### 5.4.7 Production Testing

This is the final step. At this stage everything should be in perfect working order. We’ve tested our code and sensors based on our Firebase Database. Basically if everything is in working order, then results from the sensors should be posted to our real time Firebase database. Also, this project works hand in hand with a mobile application which can be downloaded from the Google Play Store. Below is a simple diagram for you to understand the process flow of our project. The Webpage is a current work in progress and will be updated on our GitHub.



### 5.4.8 Reproducible?

If this guide is followed step by step, then you shouldn’t have any issues with the set-up. As long as you have your parts and components, this project can be completed within a week. The best advice especially if you’re trying to meet a deadline, is to get started as soon as you receive your parts and be sure to track and slot your time wisely.

## 5.5 Software Application

### 5.5.1 Overview

For the software aspect of this project, a mobile application was created within which the user can view his/her past and current results as well as make calls and arrange an UBER cab if needed. This aspect of the project was completed in the CENG 319 (Software Project) class during the fall semester (Fall 2018).

### 5.5.2 Time Commitment

If you work on the software aspect of this project with undivided attention, then the application can be done within two weeks or less depending on your personal time allotment. The software aspect of this project was spread out across the time span of an entire semester (approximately 11 weeks).

### 5.5.3 Detailed Breakdown of Software

The software aspect of this project is in the form of a mobile application. Over the next few headings, the functionality and know-hows of the application will be broken down.

#### 5.5.3.1 Login Pages and Connections to Firebase

The mobile application is heavily dependent on firebase which is a mobile and web application development mainly used for authentication and storage. It is cloud based and is the backbone of the entire Breathalyzer project. The application makes use of a Signup page as well as a Login page. In order to get started, you must first have a firebase account.

* After gaining access, you should create your first project.
* The package name of your android application with which you are integrating firebase must be provided.
* Ensure that you download the *google-services.json* file (press add app button).
* Go to the dashboard of your project (on the firebase website).
* Select Authentication from the panel on the left side then click set-up sign-up method. Choose email & password and enable it.
* Open Android Studio, select tools and then firebase. You will then be walked through step by step on how to integrate firebase with your mobile application.

#### 5.5.3.2 Main Page (Breathalyzer Test)

This page holds the basic UI (User Interface) and makes use of a navigation drawer along with a web view which pulls news articles based on DUI (Driving Under Influence) from globalnews.ca

Basically you need to create a new activity and edit the xml file to suit whatever layout you deem fit for your main page. Here is an example of what the xml file for the application for this project looks like. Also, another thing to note, is that multiple layouts were used.



#### 5.5.3.3 Navigation Drawer and Fragments

The main page uses a navigation drawer in which the user has various options. The user has the option to either:

· View his/her current results

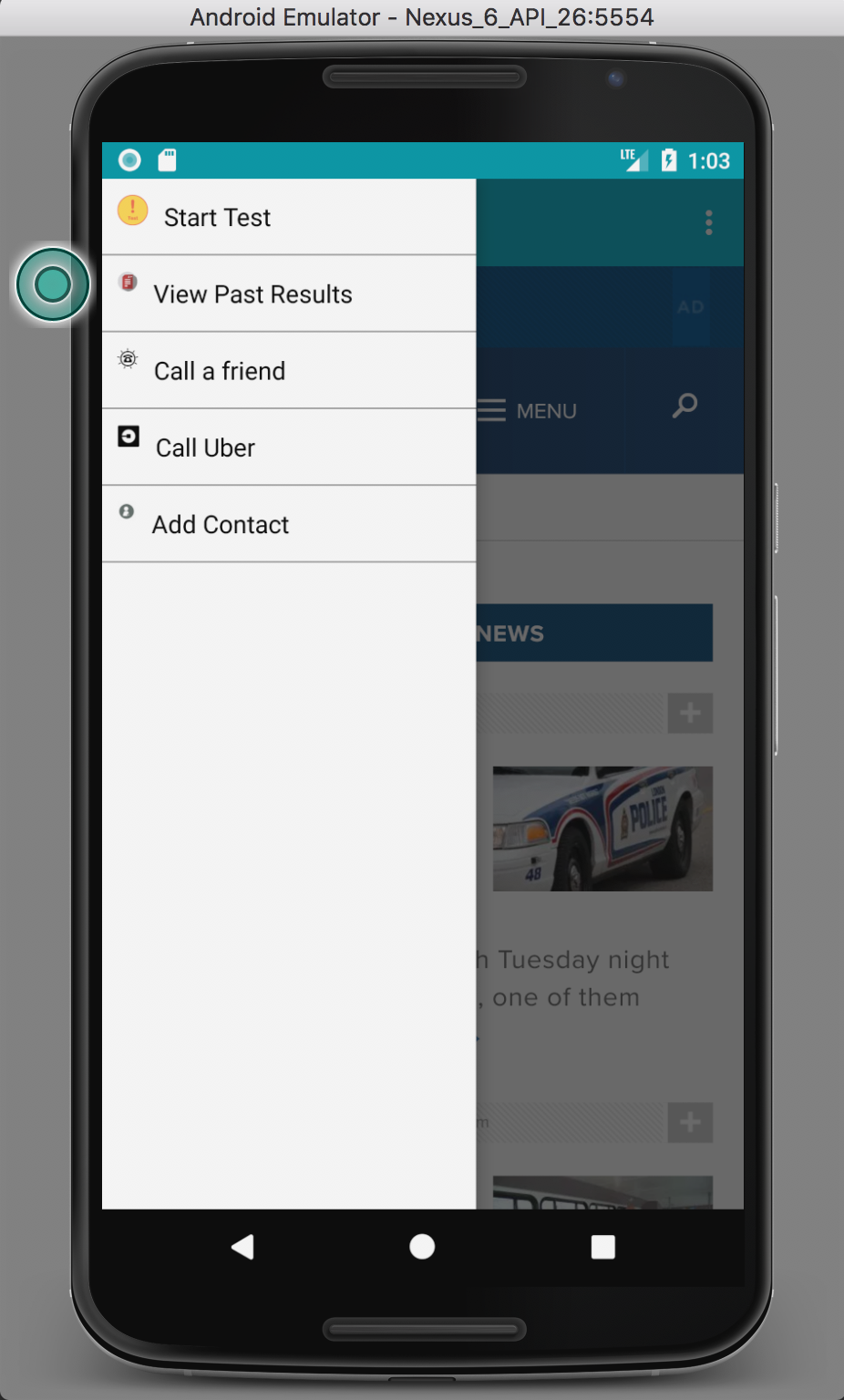
· View his/her past results

· Call a Friend

· Call an UBER cab

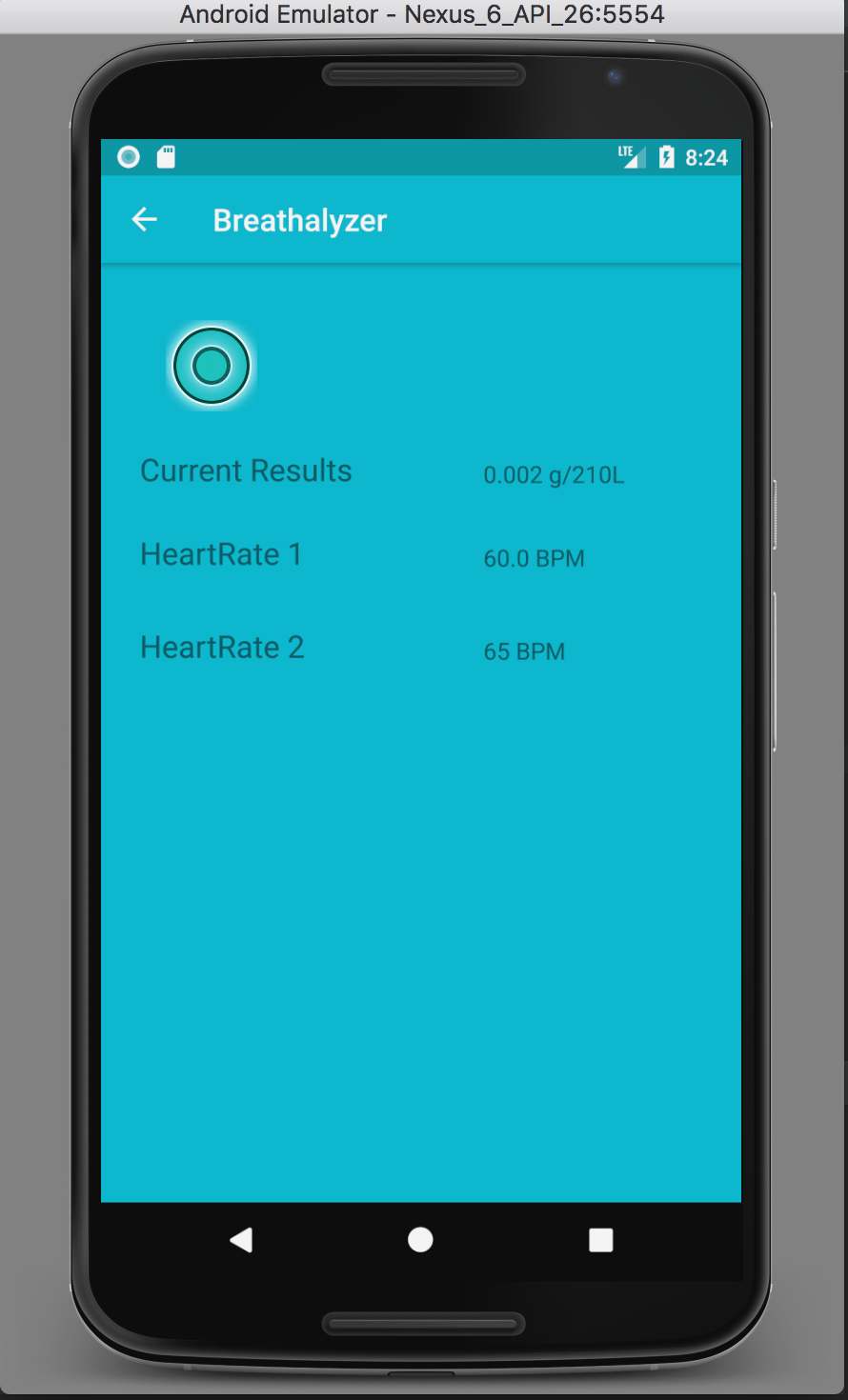
· Update Emergency Contacts

Below is a screenshot of the Navigation Drawer.



#### 5.5.3.4 Viewing Current Results (Fragment)

As a part of using Navigation Drawers, for each option in the navigation drawer, a corresponding fragment must be created. In this case, a fragment for viewing the current results of a user was created. Within this fragment, a call to the firebase database had to be made by using the “DataSnapshot” functionality. Below is a screenshot of the current results fragment.



One thing that is important to note, within the “OnDataChange” function, you are supposed to assign the value retrieved from “DataSnapshot” to a variable that matches the same data type of the data being stored in your firebase database. For Example, in

the application that was created for this project, whenever users registered for a new account, their user information gets stored on the database and dummy values are assigned in the format of strings. When the hardware runs the tests, the results from the sensors are pushed to the database in the form of floats. At first, within the application, the data type that “DataSnapshot” was assigned to was a string. This caused the application to crash repeatedly. After about two weeks of research and changing code, it was finally fixed.

#### 5.5.3.5 Viewing Past Results (Fragment)

Similarly, a fragment had to be created for this option since it was a part of the Navigation Drawer. Within this fragment, a call to the firebase database was made by using the “DataSnapshot” functionality. The past results screen basically looks the same as the current results screen with the only difference being the actual values that are displayed.  
This fragment pulls the data from firebase after it is pushed to the firebase database from the python code that interacts with the hardware. Basically, the python code takes the current results and then pushes it to the firebase database where each value is stored with a unique (timestamped) key. This fragment (Viewing Past Results) then pulls the second most recent time-stamped data. (The code will be put in the Appendices of this document)

### 5.5.4 Design Specification

#### 5.5.4.1 Introduction

The aim of this document is to basically explain what The Breathalyzer application does/entail when it’s in use by a user. Please not that this document will not contain any lines of code (to be done in another document) but will contain useful diagrams to illustrate the process flow from the splash screen all the way to the end of the application.

#### 5.5.4.2 Overview

This app is known as The Breathalyzer app and is basically supposed to do what it’s name says which is to interface with the hardware to assist in the Breathalyzer testing process. The app basically allows the user to view his/her past results as well as perform a breathalyzer test after which they can have the options to call for a taxi or emergency contacts.

#### 5.5.4.3 Scenarios for the Usage of the Application

Basically when desigining software or applications, it is usually good to provide two or three use cases to show how actual people can make use of the application.

**Scenario 1: Cindy**

Cindy is an off duty police officer who happens to be out having drinks with some friends. She has had one too many drinks to drive home.

**Cindy without the application**

Cindy lives two blocks away and assumes she has the legal limit to drive home. Cindy gets pulled over 2 minutes into her drive. Cindy is found to be over the legal limit. Cindy gets a ticket and a fine.

**Cindy with the application**

Cindy lives two blocks away and assumes she has the legal limit to drive home. Cindy decides to check with her breathalyzer and application. Cindy sees that her BAC (Blood Alcohol Content) level is well over the legal limit. Cindy’s app has redirected her to the UBER app to call for a taxi. Cindy meets home safely without any injunctions.

**Scenario 2: Matthew**

Matthew is a new father out having drinks with some friends and his wife to celebrate the birth of his new daughter.

**Matthew without the Application**

Matthew and his wife have had too many to drink but assume that they are both okay to drive. Matthew decides to take the wheel and goes onto the highway. Matthew vision becomes clouded due to inebriation and veers off the highway killing him and his wife instantly. Matthew’s baby daughter has to now go through without parents.

**Matthew with the Application**

Matthew and his wife have had too many to drink but assume that they are both okay to drive. Matthew decides to take a breathalyzer test with his breathalyzer and mobile application to boot. Matthew sees that his BAC (Blood Alcohol Content) level is over the limit as well as his wife. The application redirects Matthew to Emergency Contacts with which he calls his sister to get a ride home. Matthew and his wife reaches home safely to their baby daughter and grandmother.

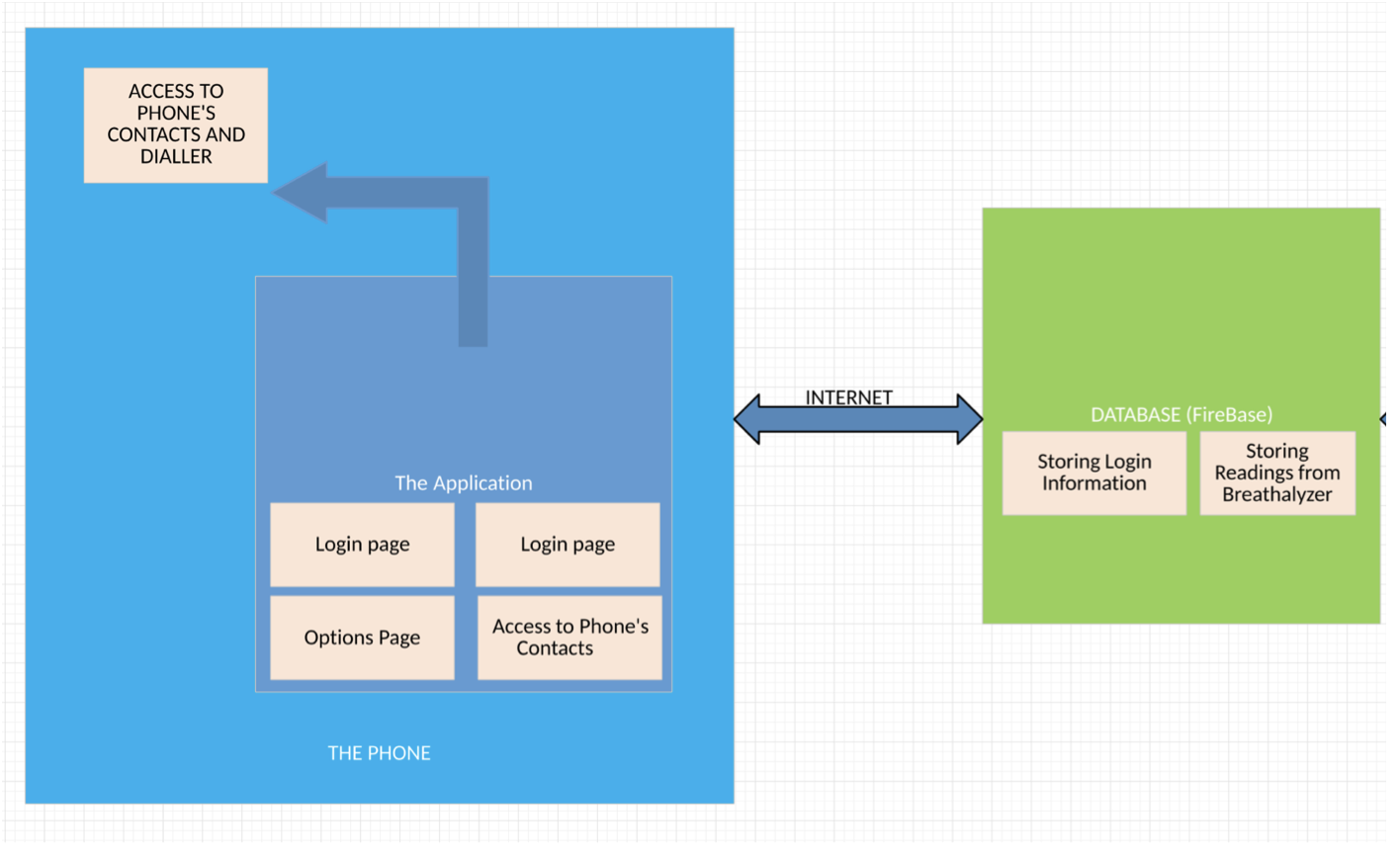
#### 5.5.4.4 Non-Goals

This version of the application will not support the following:

1. Multiple Users on the Same Device (To be Fixed)

2. Changing of Passwords (To be Fixed)

**FlowChart**

****

#### 5.5.4.5 Screen by Screen Specification

The Breathalyzer application uses quite a few different screens and layouts. All layouts are created with the XML Language within Android Studio. Each Screen will be named with a name that refers to which activity it is associated with. For example, calling a taxi will be named Taxi Activity etc.

**Splash Screen**

This screen will basically initialize the application. It has no functionality in the sense that it doesn’t store input data or accepts input of any kind.

**Home Screen**

The home screen is displayed when the splash screen has finished being displayed. The home screen has two purposes:

1. Allow existing users to Login to their Account

2. Allow new users to Create an Account (Still has to be debated upon)

**Login Screen**

The login screen is displayed if the user happens to select the login option from the home page. It is used by current account holders to login to their accounts. The username will be made to spec to allow an input of upto 60 characters while the password field will allow for upto 12 characters to be typed and will be disguised with astericks to assist in security and prevention of hacking. Hints will also be used on each Edit Text so that users can differentiate between the UserName and Password fields. The username and password will then be authenticated with entries which were stored in a firebase database.

Technical Note:

In order to disguise the password entries with astericks, within the edit text declarations in the “activity\_login.xml” file, **<android:INPUT\_TYPE=PASSWORD>** will be used.

When the user clicks login, the following checks are performed:

(i) If the user inputs an email address which either exceeds the 60 character quota or is in an incorrect format, (example doesn’t have the @ sign), then a toast will be displayed saying **incorrect email has been entered** and the user will have to re-enter the correct format.

(ii) If the user inputs an email address which isn’t associated with an account in the database, then a toast is displayed which **says “The**

**email address isn’t associated with a user account. Please click on the register button to create an account”**.

(iii) If an email address was provided but no password was entered, a toast will be displayed which says, **“Please input your password”**.

(iv) If the email address was provided, and it belongs to a registered account in our database but the password is incorrect, a toast will be displayed which **says “Invalid Password entered. Please double check your entry”**.

(v) If the email address was provided and the password and they both match what was stored in the database, then the user is redirected to the menu screen.

**Register Screen**

The Register Screen will be launched if the Register option is selected from the Home Page. This screen will make use of various edit text fields to store the user’s information. The user’s information will be stored within a firebase database. The “Full Name” edit text will allow the user to input upto a maximum of 70 characters (Letters Only). The “Email” edit text will allow the user to input upto a maximum of 60 characters. The password fields will be validated to ensure that the user enters the same password twice. The Gender Field will be a dropdown list and DOB will make use of a calendar.

Technical Notes:

Limiting the Edit Text Fields to 60/70 chars will use the same format:

android:maxLength = “60” or android:maxLength=“70”

Limiting the Full Name field to letters only:

android:digits= “abcdefghijklmnopgrstuvwxyz”

**Menu Screen**

The Menu Screen is launched upon the user’s successful login or account creation. Many options are presented to us. We have the option to Start the test, Book a cab (Redirected to UBER app), Call a Friend(Emergency Contacts)

# 6. Schedule & Progress Reports

## 6.1 Fall Semester 2017 (Phase 1 - Hardware)

* Project Selection and Project Proposal Due

Start : Monday 9/11/17

End: Sunday 9/17/17

Due: Monday 9/18/17

* Project Schedule

Start: Monday 9/18/17

End: Sunday 9/24/17

Due: Monday 9/25/17

* Discussion/Meeting with Group Members

Start: Tuesday 9/26/17

End: Tuesday 9/26/17

* Individual Budget

Start: Wednesday 9/20/17

End: Tuesday 9/27/17

Due: Monday 10/02/17

* Progress Report (Acquisition of Components)

Start: Tuesday 10/03/17

End: Thursday 10/12/17

Due: Monday 10/16/17

**College on Strike**

OCT 16, 2017 – NOV 21, 2017

* **Progress Report (Mechanical Assembly)**

Start: Tuesday 10/17/17

End: Saturday 10/21/17

Due: Monday 11/27/17

* **Individual PCB due**

Start: Saturday 09/05/17

End: Monday 10/02/17

Due: Monday 12/04/17

* **Placard and 30 Second Script**

Start: Tuesday 12/05/17

End: Sunday 12/10/17

Due: Monday 12/11/17

* **Build Video**

Start: Tuesday 12/12/17

End: Saturday 12/17/17

Due: Monday 12/18/17

* **Progress Report and Hardware Demonstration**

Start: Tuesday 12/19/17

End: Saturday 12/24/17

Due: Monday 01/08/18

* Presentation

Start: Tuesday 01/09/18

End: Saturday 01/13/18

Due: Monday 01/15/18

* **Build Instructions**

Start: Tuesday 01/16/18

End: Saturday 01/21/18

Due: Monday 01/22/18

## 6.2 Winter Semester 2018 (Phase 2 - Hardware)

* **Proposal**

Start: Tuesday 01/16/18

End: Saturday 01/21/18

Due: Monday 02/05/18

* **Requirements**Start: Tuesday 02/06/18

End: Sunday 02/11/18

Due: Monday 02/12/18

* **Technical Report Structure and Mechanics (Abstract, Introduction and Declaration of Authorship)**

Start: Tuesday 02/13/18

End: Saturday 02/24/18

Due: Monday 02/26/18

* **Group Status Report A**

Start: Tuesday 02/27/18

End: Saturday 03/03/18

Due: Monday 03/05/18

* **Build Instructions Merged into Technical Report, App, Web/Hardware, and Database Independent Demonstration, Project Photo Releases**

Start: Tuesday 03/06/18

End: Saturday 03/10/18

Due: Monday 03/12/18

* **Integration Status Report**

Start: Tuesday 03/13/18

End: Saturday 03/17/18

Due: Monday 03/19/18

* **OACETT Checklist**

Start: Tuesday 03/18/18

End: Saturday 03/24/18

Due: Monday 03/26/18

* **Troubleshooting Status Report**

Start: Tuesday 03/27/18

End: Sunday 04/01/18

Due: Monday 04/02/18

* **Demonstrations**

Due: Monday 04/09/18

* **Presentations**

Start: Tuesday 04/10/18

End: Saturday 04/14/18

Due: Monday 04/16/18

* **Technical Report**

Start: Tuesday 02/06/18

End: Saturday 04/21/18

Due: Monday 04/23/18

As seen in the project schedule above, everything is pretty laid out. Planning ahead was crucial to get this project organized and properly done. The project would not have been done as well as it did if we did not follow this schedule as much as possible. Having all these days written down so we can follow them allowed us as a group see what needs to be done, and when it needs to be done by so we are able to see when we have extra time to work on something that will take a lot more work. For example, in phase one We had set our schedule so that we start our build video a month before it was actually due. Knowing that this task would take a long time we allocated more time to it, and was able to work on other tasks while working on the build log slowly and not all at once.

Although the schedule might not have been followed to the exact dates, we believe that the main benefit from this schedule is for time management and making sure you're able to follow tasks and schedules set. Even though you might not be able to do things exactly and always be on time, it is more beneficial if you're more disciplined enough to put in the extra effort and make up for the lost time. This schedule that we had to make also helped us in the long term and towards the end where we saw how long it took us to do some tasks and see how we could improve our time management.

## 6.3 Progress Report A (Week 5 Status Report A - March 5, 2018)

This is our status update on the project MKM BREATHALYZER. We have already created the template for the Database, Software Requirements Specification, Abstract and introduction last week. Now we are actually working on implementing the design, hardware specification and writing the code essential for the working Breathalyzer.

Karan is taking the lead for the hardware implementation, first he tested the individual sensor on the pi, and then connected it together on one pi. The connection was successful and the sensors XD-58C pulse sensor, Mq3 sensor and Polar Heart Rate Sensor worked successfully. But still there is some issue with the Polar Heart Rate sensor as it shows results sometimes even when it is not being used to sense anything.

The use of two pulse sensors would improve the accuracy of the readings.

Maheshwerie is taking the lead on the database and she is trying to write a python script that pushes the result of the testing directly into firebase database. For now, she wrote a script that connects the database with the python script and send random values. Our next approach is to combine the code where it filters the data and pushes the output values into the database.

For the Mobile application, I am trying to retrieve the random values added to the database through python in the field called “View Results” by writing a java code that makes pull request. Our next step is to be able to have a fully implemented interface, in

which we are able to store the results in the firebase and display that information on the application.

There was no change in our budget as everything was available to us from the previous semester. We are using the same raspberry pi which we used in the last semester for remote connections, and the extra wires and the breadboard for connecting the sensors were already available in Parts kit. Here are some of the links that were helpful in our research.

Using firebase with python: <https://pypi.python.org/pypi/python-firebase/1.2>

Using java with firebase: <https://github.com/firebase/firebase-admin-java>

Forum that helped in troubleshooting the hardware connections: <https://www.raspberrypi.org/forums/viewtopic.php?t=42628>

## 6.4 Progress Report B ( Week 8 Status Report B (CENG 355) - MKM Developers March 19,2018)

I'm writing you this email to update you on the progress of our project: The Breathalyzer.

Currently we have combined our sensors into one working project. Karandeep has taken the lead for this. This is being done by using the Raspberry Pi breakout board and ribbon cable.I have been working on modifying the python code to facilitate multiple sensors as opposed to one. I have also been working on fixing the mobile application. What I've noticed is that the Firebase Database component of the mobile application works with devices that have API 26 or higher but not with devices that have API 24. Upon doing some research, I found out that as of August 2018, android applications will be required to have a minimum target API level of 26.0 (Which in this case will solve our issue of our app not being compatible with API 24)Mohita has been working on the webpage. The basic layout of the page is finished. She is working on adding the functionality to the webpage which would be to pull and display information for an inputted user onto the webpage. Currently we are all trying to troubleshoot one of our sensors. (Polar Heart Rate Sensor) We've noticed that the sensor picks up a reading/stray signal even when the chest strap isn't being worn.We have plans to make a change to our budget by purchasing a USB wireless adapter which will eliminate our use of "Mobile Tethering" in our project.

Link for API: <https://android-developers.googleblog.com/2017/12/improving-app-security-and-performance.html>

## 6.5 Progress Report C ( Week 9 CENG 355 0NB Status Report C - MKM Breathalyzer April 2,2018)

I'm writing you this email to update you on the progress of our project: The Breathalyzer.

In the last two weeks, we made some significant progress. We were focused on making all three of ours sensors work properly and rectify the issues with our Mobile Application.

Earlier mq3 sensor was giving only digital output i.e. 0 or 1 so we made some changes to retrieve the analog output (percentage of alcohol). Mohita and I, with the help of our professor, checked the Mq3 sensor on the multimeter for voltage changes upon detecting alcohol. This made sure that the sensor was working properly.Then we removed all the redundant lines of code from our python script .We looked up the pcf8591 module data-sheet and checked all the analog pins to understand their functionality On the basis of our finding we connected the mq3 sensor on pin 2. The python script was also modified to display the analog readings for mq3 and heartrate sensor. Later on we changed the firebase coding to display the alcohol percentage instead of a boolean value.

Meanwhile, Maheshwerie made her Wireless Pulse sensor work properly on Arduino. She changed the pin connections from digital to analog and then fixed a tiny error in the code which caused a beat to always be detected. She also fixed the problem which was causing application to crash.There was an error in one of the activities. She kept on testing the application based on an account that was already signed in. When testing on a newly logged in account, the application would crash. It was due to a missing piece of code upon Login. She changed the signup activity so that when an email is entered, a

user profile is created under the first 9 letters of the email address. That way, upon login, the first 9 letters of the input email is captured and used to access the user profile within the firebase and display the results of the user.

We will demo the above mentioned work to our professor in the class. This coming week we will combine the python scripts and all the three sensors onto one Raspberry Pi.There were no changes to the budget.

Links that were helpful

To check the pin layout of pcf8591 module

<http://wiki.sunfounder.cc/index.php?title=PCF8591_8-bit_A/D_and_D/A_converter_Module>

To get more information about login activity

<https://www.youtube.com/watch?v=uV037mLG_Ps>

## 6.6 Progress made after submitting the three progress reports:

In order to make our project compact, we found a way to remove the breadboard component from it. We found the ground pins on the PCB board itself.

Other than this, our professor wanted us to store and display the past readings of the users also. Earlier we could see the present test readings only. In order to achieve our end goal, we had to make changes to the python script and mobile application so that both the interfaces can comply to one another.

The past results were not able to be stored as they were being overwritten by the new ones. As a result, we ended up using the database’s feature of generating a unique key for each result once a reading is being captured. Due to this, we were able to see the current as well as the past results.

Finally, we had to present our project to the class.

# 

## 6.7 Fall Semester 2017 (Phase 1 - Software - Pre Strike)

A schedule was also made in Software Project class where we had to make the mobile application. Again, this helped us prioritize our application when we needed to, but also work on labs and assignments from that class at the same time.

* **Project Topic Selection and Interview**

Start: Tuesday 09/12/17

End: Sunday 09/17/17

Due: Tuesday 09/19/17

* **Final Team Member Selection**

Start: Wednesday 09/20/17

End: Saturday 09/23/17

Due: Tuesday 10/03/17

* **Working on First Draft of Application /Simulation**

Start: Wednesday 10/04/17

End: Saturday 10/14/17

Due: Tuesday 10/17/17

* **Draft Test Plan**

Start: Wednesday 10/18/17

End: Sunday 10/22/17

Due: Tuesday 10/24/17

* **Beta Release / Simulate Application to Show Progress / Progress Report**

Start: Wednesday 10/25/17

End: Monday 10/30/17

Due: Tuesday 10/30/17

* **Connect Application to Server (Firebase Database)**

Start: Wednesday 10/25/17

End: Monday 10/30/17

Due: Tuesday 10/30/17

* **Source Code Control Through Github. JUNIT Testing**

Start: Wednesday 09/20/17

End: Continuous (No end date)

Due: Tuesday 11/07/17

* **Work Progress and Storing Data to Firebase**

Start: Wednesday 09/20/17

End: Continuous (No end date)

Due: Tuesday 11/07/17

* **Test cases. Portrait and landscape support**

Start: Wednesday 11/08/17

End: Friday 11/24/17

Due: Tuesday 11/28/17

* **Testing and Demo**

Start: Wednesday 11/29/17

End: Sunday 12/03/17

Due: Tuesday 12/05/17

* Presentation, Interview Preparation and Final Submission

Start: Wednesday 12/06/17

End: Sunday 12/10/17

Due: Tuesday 12/12/17

## 6.8 Fall Semester 2017 (Phase 1 - Software - Post Strike)

* **Project Topic Selection and Interview**

Start: Tuesday 09/12/17

End: Sunday 09/17/17

Due: Tuesday 09/19/17

* **Final Team Member Selection**

Start: Wednesday 09/20/17

End: Saturday 09/23/17

Due: Tuesday 10/03/17

* **Working on First Draft of Application /Simulation**

Start: Wednesday 10/04/17

End: Saturday 10/14/17

Due: Tuesday 10/17/17 (School Strike)

* ***Draft Test Plan***

*Start: Wednesday 10/18/17*

*End: Sunday 10/22/17*

*Due: Tuesday 10/24/17*

* ***Beta Release / Simulate Application to Show Progress / Progress Report***

*Start: Wednesday 10/25/17*

*End: Monday 10/30/17*

*Due: Tuesday 10/30/17*

* ***Connect Application to Server (Firebase Database)***

*Start: Wednesday 10/25/17*

*End: Monday 10/30/17*

*Due: Tuesday 10/30/17*

* ***Source Code Control Through Github. JUNIT Testing***

*Start: Wednesday 09/20/17*

*End: Continuous (No end date)*

*Due: Tuesday 11/07/17*

* ***Work Progress and Storing Data to Firebase***

*Start: Wednesday 09/20/17*

*End: Continuous (No end date)*

*Due: Tuesday 11/07/17*

**Classes Resumed (November 21st, 2017)**

* **Test cases. Portrait, Landscape Support, Source Code Control**

Start: Wednesday 10/18/17

End: Friday 11/24/17

Due: Tuesday 11/28/17

* **JUNIT Testing and Firebase Connection**

Start: Wednesday 11/29/17

End: Sunday 12/03/17

Due: Tuesday 12/05/17

* **Test Plan and Application Simulation**

Start: Wednesday 12/06/17

End: Sunday 12/10/17

Due: Tuesday 12/12/17

* **Progress Report and Interview**

Start: Wednesday 12/13/18

End: Tuesday 01/09/18

Due: Tuesday 01/16/18

* **Final Submission and Interview**

Start: Wednesday 01/17/18

End: Monday 01/22/18

Due: Tuesday 01/23/18

## 6.9 CENG 319 (Software Project) Milestone 1

**1.** **Present the Topic and Interview**

**The Breathalyzer**

**Outline ->** This app is intended to be coupled with hardware which will be made in the CENG 317 class. The app which will be known as the breathalyzer/enforcement app, can be used by members of law enforcement. This project will be targeting specifically law enforcement individuals and making their work easier. Basically the app will provide a login screen with which a user name and login will need to be provided. The user is then redirected to a page in which they can search for individuals or make a new entry. The “search for individuals” option is intended to show the user results of an individual’s BAC (Blood Alcohol Level) as well as their heart rate at the time of testing with the hardware. The “new entry” can be used to input other offences, example, burglary or other minor offences such as disruptive behaviour etc. Most of this data will be stored within a firebase database (decision still pending). The whole aim of this app is to make the work of law enforcement officers easier while promoting efficiency in terms of environmentally wise (Paper reduction) etc.

**2.** **Team members and tasks**

**Mohita Prabhakar**

* 3 test cases
* Work on Add new individual Activity
* Powerpoint for Presentation
* Help with Gantt Chart
* Brainstorming on connection to firebase database

**Maheshwerie Samaroo**

* Working on Layouts and Java Classes
* Work on Login Activity
* 3 Test Cases
* Powerpoints for Presentation
* Excel Sheet dedicating tasks
* Brainstorming on connection to firebase database

**Karandeep Singh**

* Working on Layouts and Java Classes
* Work on Search for Individuals Activity
* Test Cases
* Work on Gantt Chart
* Brainstorming on connection to firebase database

# 7. Challenges

During the development of this project, we faced several challenges which we overcame through research, brain-storming and the help of our professors. The first challenge was the uploading of the application that was developed for this project to the Google Played Store. The upload was initially tried but got rejected by the Play Store due to the violation of the Misleading Claims Policy. Basically, the play store stated that the application wasn’t declared as a simulator.

This is the failed email that was received from the Google Play Store.

*Hi Developers at MohitaSandy,*

*After review, BreathAlyzer, prabhakar2.mohita, has been suspended and removed from Google Play as a policy strike because it violates the****misleading claims****policy.*

*The title and/or description of your app has been determined to misrepresent its function by not clearly marking it as a "prank," "simulated," “joke,” or "fake" app.*

***Next Steps***

1. *Read through the*[*Misleading Claims*](https://play.google.com/about/ip-deception-spam/deceptive-behavior/misleading-claims/)*article for more details and examples of policy violations.*
2. *Make sure your app is compliant with the*[*Deceptive Behavior*](https://play.google.com/about/ip-deception-spam/deceptive-behavior/)*policy and all other policies listed in the*[*Developer Program Policies*](https://play.google.com/about/developer-content-policy.html)*. Remember additional enforcement could occur if there are further policy issues with your apps.*
3. [*Sign in to your Developer Console*](https://play.google.com/apps/publish)*and submit the policy compliant app using a new package name and a new app name.*

*Additional suspensions of any nature may result in the termination of your developer account, and investigation and possible termination of related Google accounts. If your account is terminated, payments will cease and Google may recover the proceeds of any past sales and/or the cost of any associated fees (such as chargebacks and transaction fees) from you.*

*If you’ve reviewed the policy and feel this suspension may have been in error, please reach out to our policy support team. One of my colleagues will get back to you within 2 business days.*

*Regards,*

*The Google Play Review Team*

The Description was not clear enough for the Application and google protects the users from being pranked or misled.

Here is what Google’s Misleading Claims Document says.

[**Google’s Misleading Claims**](https://play.google.com/about/privacy-security-deception/deceptive-behavior/misleading-claims/)

Google doesn’t allow apps that contain false or misleading information or claims, including in the description, title, icon, and screenshots.

**Here are some examples of common violations:**

* Apps that misrepresent or do not accurately and clearly describe their functionality:
  + An app that claims to be a racing game in its description and screenshots, but is actually a puzzle block game using a picture of a car.
  + An app that claims to be an antivirus app, but only contains a text guide explaining how to remove viruses.
* Developer or app names that misrepresent their current status or performance on Play. (E.g. “Editor’s Choice,” “Number 1 App,” “Top Paid”).
* Apps that feature medical or health-related functionalities that are misleading or potentially harmful.
* Apps that claim functionalities that are not possible to implement.
* Apps that are improperly categorized

The application was then re-uploaded by another group member but the issue was that the package name was similar to the first upload. Similarly, while working with databases as well, it makes use of the same package name as that used in the Android Studio. Also, we had to be very specific about the description of the application this time. Therefore, within one day another database was required and the other content was copied to the new application. There was a lot of effort done to upload an application to the google play store. It almost required more effort as a whole new working database and description was required. We made use of the words “simulated” and “fake” in the title so that we didn’t need to go through another termination. This is what our description looks like:

**Title**: Simulated Breathalyzer App (Fake)

**Description**: This app has been developed under my university project. The app is supposed to work as a companion app with Breathalyzer hardware device. The device is under construction under the same project. The purpose of uploading the app currently is to demonstrate User Interface and other features as part of my current work assignment. In the current state the app is returning Sample data preprogrammed into the app, it cannot connect to a hardware device yet. There will be subsequent updates with newer functionality as the project progresses. The final goal is to have the app work in real time with 'Breathalyzer hardware device' in a manner that it will return accurate information while connecting to Breathalyzer device.

An email was sent to the Google-developer-Support team in which it was explained to Google developer reviewers that the application was solemnly based for University/college project with GUI functionalities.

After uploading the application again, within the wait time of an hour, the application was successfully uploaded to the Play Store

This is how version one of the application MKM BREATHALYZER was released. Version 2 is in production and will be released soon. Below is the link for version 1 of the application.

<https://play.google.com/store/apps/details?id=ram.ram>

Another challenge that was faced, was the combination of our hardware components. As was stated earlier, this project is a combination of three sensors; two pulse sensors and one mq3 (Alcohol and Gas) sensor. One of the two pulse sensors made use of a wireless interface. Basically, when the receiver was connected, it kept receiving readings even though the chest strap which is referred to as the transmitter wasn’t being worn. The first thing that was done was to isolate that component from the rest of the project.

The wireless pulse sensor was tested by using the Arduino Mega 2560. The testing was just to ensure that there was not a broken component and to verify that everything was in working order. After that was done, the sensor was re-connected to the rest of the project but the same issue was still encountered. It took approximately a week to figure out what was wrong with the connections. Basically, the wireless receiver was connected to Digital pins on the Raspberry Pi. The wire that was connected to the Digital Pin, acted like a switch. Basically, connecting the wire acted like a closed circuit and hence a reading was always captured by the receiver. The other option that was considered was by using an DAC (Digital to Analog Converter) but that was quickly ruled out due to time and budget constraints. Eventually a solution was found for this problem. The Arduino was connected to the Raspberry Pi via a serial connection. The code that interacts with the wireless sensor was pushed to the Arduino and a connection was made via the python code on the Raspberry Pi. This fix was quick and simple but if this project was to go into consumer production, it wouldn’t be recommended.

Another challenge faced was with integrating the mobile application and the hardware. Whenever the hardware was run, results would be pushed into the firebase database. Whenever those results were retrieved by the application, it would cause the application to crash. At first it was thought of to be an API version issue. What was realized was that there was an issue in the code. Initially when an entry is created in the firebase database for a user, the results are pushed in the String format. Whenever those results are updated after using the hardware, the results are pushed in the Float format. Initially the application retrieved data from the firebase in the String Format which was fine but upon the data captured by the sensors being pushed to the database, the application crashed. A simple line of code had to be changed to use the Float format instead of the String format.

# 8. Conclusion

This project has been created with the hope of reducing the drinking and driving rates and to lower the death rates caused due to this. This breathalyzer is intended to be a personal device with prospects to be introduced in bars or locales that serve alcoholic beverages. It is our sincere hope that this project will be used for the greater good. The final version of our project meets all of the specifications that were required to be completed.

# 9. Recommendations

The hardware that we’ve created can be improved upon in so many ways. Firstly, beginning with the wiring; it can be done a little neater especially if the project was to be reproduced for consumer use. Currently we are using an adapter as our power source. That can be swapped with a portable charger. Another thing would be the cost of the sensors. The most expensive sensor in this project was the Polar Heart Rate Receiver and Transmitter. Most of the websites typically have these at the same prices but the other sensors can be bought in bulk for a much cheaper price. In the long run, it will cut your shipping cost. Once you’ve completed the project and it’s in working order, you can feel free to make any tweaks or improvements that you feel fit.

# 10. Bibliography

[1] Heartbeats in Your Project, Lickety-Split ♥. (n.d.). Retrieved February 02, 2018, from https://pulsesensor.com/

[2] #237238, M., #321089, M., #661774, M., O., #721982, M., #554862, M., . . . G. (n.d.). Pulse Sensor. Retrieved February 02, 2018, from https://www.sparkfun.com/products/11574

[3] #637052, M., & O. (n.d.). Alcohol Gas Sensor - MQ-3. Retrieved February 02, 2018, from https://www.sparkfun.com/products/8880

[4] Industries, A. (n.d.). Heart Rate Educational Starter Pack with Polar Wireless Sensors. Retrieved February 04, 2018, from https://www.adafruit.com/product/1077

[5] Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

[6] Wang, X., Jin, J., & Li, S. (2008, September 03). Measurement and analysis of heart signal based on the pressure sensor. Retrieved February 04, 2018, from http://ieeexplore.ieee.org/document/4618175/

[7] Malathi, M., Sujitha, R., & Revathy, M. R. (2018, February 01). Alcohol detection and seat belt control system using Arduino. Retrieved February 04, 2018, from http://ieeexplore.ieee.org/document/8275841/

[8] Kirtana, R. N., & Lokeswari, Y. V. (2017, June 08). An IoT based remote HRV monitoring system for hypertensive patients. Retrieved February 04, 2018, from <http://ieeexplore.ieee.org/document/7944086/>

[9] Enabling Dishonest Behavior | Deceptive Behavior | Privacy, Security, and Deception - Developer Policy Center. (n.d.). Retrieved April 22, 2018, from <https://play.google.com/about/privacy-security-deception/deceptive-behavior/dishonest-behavior/>

[10] Instructables. (2017, October 03). Raspberry Pi - Arduino Serial Communication. Retrieved April 22, 2018, from <http://www.instructables.com/id/Raspberry-Pi-Arduino-Serial-Communication/>

# 11. Appendices

## 11.1 Python Code for Raspberry Pi

|  |
| --- |
| import pyrebase |
|  | import time |
|  | import serial |
|  | import RPi.GPIO as GPIO |
|  | #GPIO.setup(11,GPIO.OUT) |
|  | #GPIO.VERSION |
|  | GPIO.setmode(GPIO.BOARD) |
|  | GPIO.setwarnings(False) |
|  | GPIO.setup(16, GPIO.IN) |
|  |  |
|  | ser=serial.Serial("/dev/ttyACM1", 9600) |
|  | ser.baudrate=9600 |
|  | def blink(pin): |
|  |  |
|  | GPIO.output(pin,GPIO.HIGH) |
|  | time.sleep(1) |
|  | GPIO.output(pin, LOW) |
|  | time.sleep(1) |

|  |  |
| --- | --- |
|  | return |
|  |  |
|  | from smbus import SMBus |
|  | bus = SMBus(1) |
|  |  |
|  | def read\_ain(i): |
|  | global bus |
|  | #bus.write\_byte\_data(0x48, 0x40 | ((i) & 0x03), 0) |
|  | bus.write\_byte(0x48, i) |
|  | bus.read\_byte(0x48)#first 2 are last state, and last state repeated. |
|  | bus.read\_byte(0x48) |
|  | return bus.read\_byte(0x48) |
|  |  |
|  | config = { |
|  | "apiKey": "your api key here", |
|  | "authDomain": "example.firebaseapp.com", |

|  |  |
| --- | --- |
|  | "databaseURL": "https://example.firebaseio.com", |
|  | "projectId": "exampler", |
|  | "storageBucket": "example.appspot.com", |
|  | "messagingSenderId": "your messaging Sender Id here" |
|  | } |
|  |  |
|  | firebase = pyrebase.initialize\_app(config) |
|  |  |
|  |  |
|  | applicant = input("Enter your username:") |
|  |  |
|  | i = 0 |
|  | count = 0 |
|  | j = 0 |
|  | while(i<5): |
|  | heartrate1 = read\_ain(1) |
|  | alcohol = read\_ain(2)\*0.001 |
|  | read\_ser=ser.readline() |
|  | hr2 = read\_ser.decode('utf-8') |
|  | print ("\n----Testing----\n") |

|  |  |
| --- | --- |
|  | print("HeartRate Sensor: {0:.0f} BPM".format(heartrate1)) |
|  | print ("Alcohol Reading: {0:.2f}%\n".format(alcohol)) |
|  | print ("Wireless Heart Rate : %s\n" %hr2) |
|  | time.sleep(3) |
|  | i+=1 |
|  |  |
|  |  |
|  | db = firebase.database() |
|  | result = db.child("user").child(applicant).update({'Heart Rate1':heartrate1}) |
|  | result2 = db.child("user").child(applicant).update({'Alcohol Reading':alcohol}) |
|  | result3 = db.child("user").child(applicant).update({'Heart Rate2':hr2}) |
|  | result4 = db.child("user").child(applicant).child("Past Results HR1").push(heartrate1) |
|  | result5 = db.child("user").child(applicant).child("Past Results Alcohol Reading").push(alcohol) |
|  | result6 = db.child("user").child(applicant).child("Past Results HR2").push(hr2) |

## 11.2 Arduino Code for Arduino Mega 2560

//Definitions

//const int HR\_RX = 7;

byte oldSample, sample;

int i = 0;

int b = 0;

unsigned long currentmillis;

unsigned long millisecs[20] = {};

unsigned long startmillis;

void setup() {

Serial.begin(9600);

pinMode (A5, INPUT); //Signal pin to input

Serial.println("Waiting for heart beat...");

//Wait until a heart beat is detected

while (analogRead(A5)) {};

Serial.println("Heart beat detected!");

Serial.println(analogRead(A5));

startmillis = millis();

}

void loop() {

static int count = 0;

static bool counted = false;

static uint32\_t ts = millis();

int val = analogRead(A5);

if ((val > 550) && (counted == false)) {

counted = true;

count++;

} else if (val < 500) {

counted = false;

}

if (millis() - ts >= 10000) {

ts += 10000; // You could assign ts the millis() value, but

// by adding 10 seconds it cancels any drift

// caused by other delays in your sketch

int BPM = count \* 6;

count = 0;

Serial.println(BPM);

}

}