



University of Central Florida
Department of Electrical Engineering and Computer Science
Dr. Lei Wei

Group 31

Nam Ngo	nqn1001@knights.ucf.edu	EE
Nicholas Fraser	NFraser@knights.ucf.edu	EE
Charles Taylor	CharlesATaylor@knights.ucf.edu	CpE

Contents

1. Executive Summary	1
2. Project Description	2
2.1 Project Motivation	2
2.2 Project Goals	3
2.3 Objectives	4
2.3.1 Alcohol Sensing	4
2.3.2 Biometric User Verification	4
2.3.3 Key FOB Integration	4
2.3.4 Bluetooth Communication	5
2.3.5 Cellular Device Application	5
2.4 Requirements Specifications	6
2.4.1 Physical Specifications	6
2.4.2 Power specifications	7
2.4.3 Performance specifications	7
2.5 Quality of House Analysis	8
3. Research related to Project Definition	9
3.1 Existing Similar Projects and Products	9
3.1.1 SAAB Alcokey	9
3.1.2 ALCOLOCK V3	9
3.1.3 ALCOLOCK's DRIVESAFE elan	9
3.1.4 BACKtrack Mobile Pro	9
3.1.5 Daniel Andrade's Arduino MQ-3 breathalyzer project	10
3.1.6 Nootropic design's Arduino MQ-3 breathalyzer project	10
3.2 Relevant Technologies	10
3.2.1 Biometric fingerprint scanner	10
3.2.2 Gas sensors	12
3.2.3 Bluetooth	13
3.2.4 Battery	13
3.3 Strategic Components and Part Selections	14
3.3.1 Microcontroller	14
3.3.1.1 Arduino Uno	15
3.3.1.2 Arduino ProMicro	16

3.3.1.3	Arduino Mini.....	17
3.3.1.4	Raspberry Pi 3 model B	17
3.3.1.5	MSP430G2211IN14.....	18
3.3.2	Bluetooth.....	19
3.3.2.1.	Adafruit Bluefruit LE UART Friend.....	19
3.3.2.2	Phantom YoYo JY-MCU Arduino Bluetooth Wireless Serial.....	20
3.3.2.3	BLE Nano - nRF51822.....	20
3.3.3	Fingerprint Scanner.....	21
3.3.4	Breathalyzer sensor	22
3.3.4.1	MQ-3 Sensor.....	22
3.3.4.2	MR513 Sensor.....	23
3.3.5	Power supply.....	24
3.3.5.1	Power option 1.....	24
3.3.5.2	Power option 2.....	25
3.3.6	Key remote.....	25
3.3.6.1	300-0247ES Universal Car Remote	25
3.3.6.2	KPT1306 key remote	25
3.3.6.3	K410 Car Remote Central Lock Locking Entry System.....	25
3.4	Architectures and Related Diagrams	26
3.4.1	Microcontroller Architecture.....	26
3.5	Parts Selection Summary.....	27
4.	Related Standards and Realistic Design Constraints.....	29
4.1	Standards	29
4.1.1	Bluetooth Standard.....	29
4.1.1.1	Core System Architecture	30
4.1.1.2	Bluetooth Security.....	32
4.1.2	Health standard	33
4.1.3	Design impact of relevant standards.....	34
4.2	Realistic Design Constraints.....	34
4.2.1	Economic and Time constraints	34
4.2.2	Environmental, Social, and Political constraints.....	34
4.2.3	Ethical, Health, and Safety constraints	34
4.2.4	Manufacturability and Sustainability constraints.....	34

5. Project Hardware and Software Design Details	35
5.1 Initial Design Architectures and Related Diagrams	35
5.1.1 Project Hardware Components	35
5.1.2 Hardware Wiring.....	36
5.2 Bluetooth Subsystem	38
5.2.1. Bluefruit LE UART Friend (BLE).....	38
5.2.2. Sensor Specific Operation.....	38
5.2.3 Subsystem Implementation.....	38
5.3 Biometric Subsystem	39
5.3.1 Sensor Overview	39
5.3.2 Sensor Specific Operation.....	39
5.3.3 Subsystem Implementation.....	40
5.4 Breathalyzer Subsystem	40
5.4.1 MQ-3 Alcohol Gas Sensor.....	40
5.4.2 Sensor Specific Operation.....	41
5.4.3 Subsystem implementation.....	41
5.5 Software Design	42
5.5.1 Programming Languages	42
5.5.1.1 Assembly	42
5.5.1.2 Python.....	43
5.5.1.3 Java (Android API).....	44
5.5.1.4. C	45
5.5.1.5 C++ (avr-g++ toolchain).....	46
5.5.2 Integrated Development Environments	48
5.5.2.1Eclipse	48
5.5.2.2 Arduino IDE	50
5.5.2.3 Android Studio	52
5.5.3 Functional Requirements	55
5.5.3.1 Main Functionality	55
5.5.3.2 Technical Functionality.....	56
5.5.3.3 Software Requirements.....	58
5.5.3.4 Interface Requirements.....	59
5.6 Summary of Design	60

6. Project Prototype Construction and Coding	61
6.1 Integrated Schematics	61
6.2 Parts Acquisition	61
6.2.1 Adafruit	61
6.2.2 Sparkfun	62
6.2.3 Digi-Key	62
6.2.4 UCF	62
6.3 PCB Design	62
6.3.1 EAGLE.....	62
6.3.2 National Instruments Ultiboard	63
6.3.3 AutoCAD	63
6.4 PCB House	63
6.4.1 PCBWay.....	63
6.4.2 Elecrow	64
6.4.3 Seed Studio	64
6.5 Construction	64
6.5.1 Hand Soldering.....	64
6.5.2 Reflow Oven	64
6.5.3 Types of Mounting	65
6.5.4 TI Innovation Lab.....	65
6.6 Final Coding Plan PERT chart	66
7. Project Prototype Testing Plan	68
7.1 Hardware Test Environment	68
7.1.1 Power Supply	68
7.1.2 Car Access.....	68
7.1.3 Cellular Devices.....	68
7.2 Hardware Specific testing	69
7.2.1 MQ-3 Alcohol Sensor	69
7.2.2 Adafruit Fingerprint Sensor	70
7.2.3 Bluefruit UART Friend Bluetooth Module	74
7.2.4 Key FOB.....	75
7.2.5 Linear Voltage Regulator	76
7.3. Software Test Environment	76

7.4 Software Specific Testing	77
7.4.1 Introduction	77
7.4.2 Overall Objective for Software Test	77
7.4.3 Stopping Criteria & Testing Method	77
7.4.4 Description of Individual Test Cases	78
8. Demonstrations	82
8.1 Initial Activation and Setup	82
8.2 Standalone Operation	82
8.3 Bluetooth Pairing	82
8.4 Connected Operation	83
9. Administrative content	84
9.1 Milestone Discussion	84
9.2 Budget and Finance Discussion	89
9.2.1 Finance option 1	89
9.2.2 Finance option 2	90
9.3 Group management	91

1. Executive Summary

The BreathaLock is a new device idea that is intended to take the place of a car key fob and prevent drunk driving. With the help of this device we hope that car owners will not only be stopped from being able to drive under the influence of alcohol but also be more conscience of the dangers of driving drunk by the reminder of having to use the device repeatedly. Our systems core features include ease of use, accuracy, and effectivity to prevent drunk driving without being an inconvenience to car owners. It is intended that this device will be cross platform, operational to multiple car manufacturers and with the aid of Bluetooth available to connect to a cell phone to display additional information. The BreathaLock with be fully contained inside of a handheld device that will not be excessively large.

The BreathaLock with operate under the following sequence. First, the user when approaching the vehicle will take out the BreathaLock and initialize the device to begin scanning for a specific user. A biometric fingerprint sensor on the back of the BreathaLock will take a reading to confirm the correct user is operating the device. Once passed, the BreathaLock with prompt the user to blow into the alcohol sensor located at the top of the device. In the event that the user is sober the BreathaLock will allow the user to unlock the vehicle remotely. If the user does not pass such criteria, then the BreathaLock will deny access until these tests are repeated yielding passing behavior.

To successfully implement a device to meet these expectations requires considerable amounts of research and design. We must effectively determine the most accurate way to measure and decipher the blood alcohol level of a user through the use of an analog alcohol sensor. This is extremely important when considering the consequences of any chance of error, primarily allowing a user to drive under the influence of alcohol. Secondly it is important that we investigate the speed and accuracy of the biometric sensor and integrate it as a productive feature to ensure that the correct driver of the vehicle is going to be tested for blood alcohol content. Finally, this device needs to connect quickly and effortlessly to the user's cell phone and also be extremely user friendly. We intent to extend these features with additional options such as allowing the user to add additional drivers through the app.

With the help of existing technologies and our own engineering the BreathaLock will be a fully functioning device displaying the previously discussed features. This device could have commercial opportunity for legal repercussions or could be a recreational device used strictly for voluntary preventative care.

2. Project Description

Before getting into depth of what the BreathaLock system is, a brief overview is given below of the overall motivation and specifications that come with this design. To create a compelling and relevant device it is important to research and consider many factors, both in design and consumer desires, which will go into this design.

2.1 Project Motivation

While pursuing a college degree and living in a college town every student encounters many challenges inside and outside of the classroom. One of the primary challenges that affect many students and adults around the country is driving under the influence of alcohol. Most adults know that it is an unwise decision to get behind the wheel at an impaired state but despite the fact that there are many consequences, both legal and life altering, many adults and teens fail to avoid driving under the influence.

According to Mothers against Drunk Driving (MADD), on average two in three people will be involved in a drunk driving crash in their lifetime. This means that the majority of people will be affected by drunk driving personally without considering the effects of loved ones and peers that may be affecting them as well. Drunk driving is a very serious issue that needs to be addressed more effectively and possibly more aggressively.

When considering the legal consequences of getting a driving under the influence (DUI) charge it is scary to see how drastic they can be. Best listed by dui.drivinglaws.org legal punishment of DUI in the state of Florida can include vehicle impoundment, fines, probation, community service, license suspension, ignition interlock devices, and even jail time. As college students, we encounter many distractions and the last thing that we need is legal consequences. Even more severe is that some universities will suspend students or even expel students if the offence is campus related. It is our intention that with the use of a BreathaLock system in a preventative way, less students will legal consequences of DUI by preventing driving under the influence.

Even more important than preventing legal consequences from DUI we seek to prevent the physical dangers that can result from driving at an impaired state. The National institute on Alcohol Abuse and Alcoholism states that "About 1,825 college students between the ages of 18 and 24 die from alcohol-related unintentional injuries, including motor vehicle crashes." It is tragic to see a fellow family member, classmate, or even friend make the mistake of driving under the influence and getting injured or killed. It is even more tragic to see someone that is sober and innocent be negatively affected by the decisions of others under the influence.

Currently there are various ways to avoid this situation: carpooling with a sober driver, using alternate transportation, or waiting till your body processes alcohol enough to be within the legal limit. Some of these are costly, some are inconvenient, and some are hard to measure. Although it may seem like an easily avoidable situation when under the

influence of alcohol bad decisions can be made. With the help of BreathaLock there may be another method to prevent drunk driving more effectively.

To combat this ever present issue we propose to implement a device that is easy to use and cost affective to help college students and adults around the county avoid driving under the influence of alcohol. In our society and specifically in Orlando it is simply too easy for a student to decide to drive under the influence with the option of getting in their car in the driveway and making a regrettable decision. Our device is intended to make the user take an additional step before making this decision and in the event of failing to prove sobriety the user will be unable to make this regrettable decision.

2.2 Project Goals

The goal of BreathaLock is simple: to prevent drunk driving to all vehicle owners. We intend to create a device that is so simple and user friendly that it could be accepted and used by all vehicle owners. To do so we must investigate the most important characteristics of a device that could be acceptable to the public. If the BreathaLock exists in the hands of all vehicle owners, we could completely eliminate drunk driving and consequently eliminate the negative affects the come along with it. Although this may seem like a very large goal it is important to strive to create a device that is best suited for the consumer in efforts to achieve a compelling product.

The technology and implementation of the components that will be incorporated into the BreathaLock device have been available for quite some time. The only reason this device has not been available yet is the lack of seamless implementation and price point. With that said, we recognize that doing this affectively will not be an easy task considering that vehicle owners are currently using very small key FOBs and will probably not want to digress to a large bulky FOB. Additionally, to have this product work cross platform there would have to be allowance from car manufactures to allow the use of this device. Despite this obstacle, our team will be designing the densest and size effective device possible with limited resources. In the implementation of the BreathaLock it is not expected to match the size of a current key FOB but in efforts of proof of concept we hope that it may spark further investigation to get to that size with later revisions.

By the end of this project we hope to have a concept and working model of what a professionally manufactured product could look like. We will be designing a custom fit housing, PCB board, and all wiring which gives us the luxury of designing this product to look and feel the way that we want it to. This device with the help of Bluetooth technology and a cellular device will educate and inform vehicle owner how to avoid and learn more about how they react to alcohol.

2.3 Objectives

In order to allow vehicle owners to appreciate and accept this product we will discuss our core project objectives. These will highlight the most important features that will make the BreathaLock a powerful and useful device. There are many features and extensions to this project that we would like to achieve. The most important features to the BreathaLock system are listed below.

2.3.1 Alcohol Sensing

Our main objective with the alcohol subsystem is to gather an extremely accurate and reliable reading each time we sample. Throughout our implementation it is going to be very important to research existing breathalyzer technology and what the best way to get an accurate reading. Our alcohol sensor provides us with a simple analog reading of the amount of alcohol based on the conductivity of a piece of tin oxide. This is a very primitive sensor that is going to require testing and logic to create repeatability and accurate readings. How long should we sample? Do we throw out the highest reading or average all values taken per sample? These are some questions that we are going to need to ask ourselves and investigate when creating a dependable breathalyzer subsystem. It is imperative that we implement the breathalyzer subsystem to be just as accurate as a police grade breathalyzer to demonstrate reliability.

2.3.2 Biometric User Verification

One of the main advantages of the BreathaLock as opposed to any other breathalyzer on the market is that it offers user recognition to ensure the driver of the vehicle is blowing into the breathalyzer. With that advantage in mind it is our main objective to make sure that our fingerprint sensor work repeatedly and accurately as well. Some research and testing will need to go into what precautions we will need to make to keep our fingerprint sensor working well. Because we are using a prepackaged fingerprint sensor that contains an internal DSP chip and processing capability we anticipate it working well with our systems but it is still important to test all use cases for repeatability. The objective of the fingerprint sensor is to read quickly and accurately to determine if the user is the correct owner of the vehicle or not.

2.3.3 Key FOB Integration

Because our product is interfacing with a vehicle and security system that is out of our control we will need to use the design and parts from the existing key remote access. It is our intent that the BreathaLock interface to the vehicle locking system not be hindered at all compared to working native key fob without being tampered with. Ideally we hope to reverse engineer the key FOBs components, recreate the circuitry and then simply desolder the preprogrammed chip from the FOB and solder it onto our own PCB. Currently we are investigating the complexity of this task and assessing whether or not it is achievable. In the event that we cannot do we will resort to manually wiring to the switches of the remote access control board and control it from there.

2.3.4 Bluetooth Communication

The Bluetooth connection between the BreathaLock and cell phone needs to be implemented in a simple easy to use way. When many people think of Bluetooth connections they immediately think of the headache that sometimes comes with device pairing conflicts. To avoid this problem, the BreathaLock must immediately search and try to connect with the cell phone and, as long as the cell phone has paired previously and has Bluetooth on, will connect automatically. With this model by the time the user goes to use the biometric fingerprint sensor and breathalyzer it will already have connected to the cellphone. Depending on how fast and effectively the implementation of other features go we will decide on how many added features to the cell phone app. Ideally we could have results post processing to give the user an idea of how soon they will be under the legal limit of alcohol.

2.3.5 Cellular Device Application

In order to display more information on the status of the BreathaLock and the results of the sensors we will be designing a cellular application for android platform. This addition to the BreathaLock system will open the door to many added features and post processing data. Although adding an extraordinary application to this device would be great there is a large time constraint with this course and project. For this reason, we will investigate several goals that we will strive to achieve to make a great app.

Level 1: At the very least, we intent to create a viewer on the device that will display real time data and instruction on how and what to do on the BreathaLock handheld device. This will include outputting the current blood alcohol content of the user at the time of sample. This feature would allow for an easier experience in the event that the fingerprint sensor does not recognize the user and would like to prompt for a second reading or breathalyzer test is inconclusive

Level 2: The next available step to the BreathaLock application possible would be to have data logging and the ability to inform the user information on how long to wait or how much the user could approximately drink based on their age, weight, height, etc. Under the data logging of the BreathaLock including how often the device was used, who attempted to use the device, and whether or not the user passed given tests and was enabled to use the vehicle.

Level 3: The final level possible if we are given ample time to implement to the BreathaLock system would be calling other android applications from within the BreathaLock app to do other functions. This could include calling for an alternate means of transportation from a 3rd party app such as Uber or contacting a family member after a certain amount of failures. These features would be great to have as a part of our app but will need considerable amount of time to develop and therefor will be determined based on how quickly we can implement the rest of the features.

2.4 Requirements Specifications

2.4.1 Physical Specifications

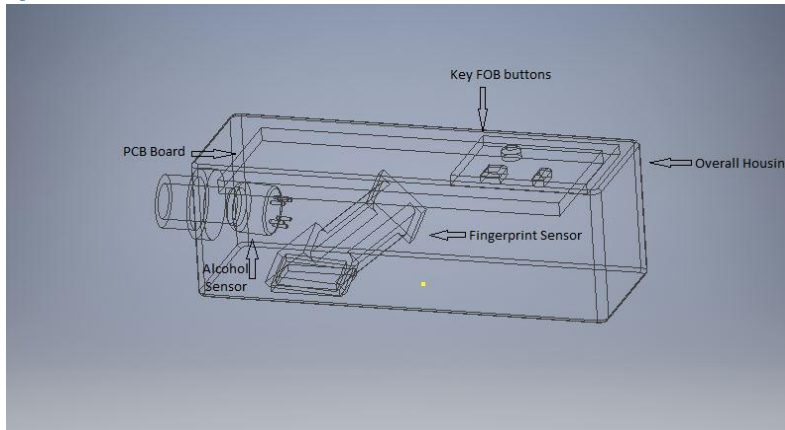


Figure 1: 3D Wireframe model of BreathaLock

To implement a compelling and attractive device it is important to consider the appearance and size goals of the BreathaLock system. In efforts to create the most concise and compact device possible we are creating the entire device no larger than 3"x3"x7". This size must include the whole system including all wires, sensors, and enclosure. Although it would be more desirable to implement this device much smaller we are limited by the size of our designed PCB, native key FOB, and other sensors that will take up much more space than a key FOB without added features that the BreathaLock system will encompass.

Following size constraints, we will investigate construction materials. The bulk of the device will come from the multiple sensors, PCB, and power supply, in this case an interchangeable battery. Because we are limited to the prefabricated sensors, PCB material, and battery the only options we have physically are whether or not to create a custom case out of any material. The main options of materials are to have a plastic or metal enclosure and physically to create a custom enclosure, or to retrofit a store bought housing for our implementation. As far as metal enclosures are concerned the main advantage would be rigidity and durability. Although these are important characteristics, designing and implementing a metal enclosure will be much heavier and costly. If we choose to use a plastic enclosure we will save money, weight, and open the door to ease of custom enclosure design with 3D printing technology. Our team is fully capable of a simple enclosure design and execution which makes a clear decision to go with the custom 3D printed enclosure.

In addition, we intend to have a device that will be no heavier than 1lb and be able to hang on a key ring. These added physical specifications are important to keep this device relevant and comparable to exiting key FOBs. It is important that we not forget the motivation and goals of the implementation of the BreathaLock system. We must do our best to compete physically with the current devices we are replacing.

2.4.2 Power specifications

When talking about the BreathaLock power specifications the only power specification noted is to be able to power on the whole project within 20V. This 20 V should be sufficient to power on a few sensor and a microcontroller. In addition, the BreathaLock project is a device that is meant to be comfortably portable, anything more than 20V would be cumbersome in batteries.

2.4.3 Performance specifications

When talking about the performance of the project, the BreathaLock project must operate with a delay of less than ten seconds. The term “operate” implies that for each action the BreathaLock is taking, there should be no delay of more than ten seconds. For example, waking up from low-power mode should take no more than ten seconds. Taking an alcohol sample then processing it should take no more than ten seconds. Reading the user’s fingerprint then processing it correctly should take no more than ten seconds. This ten-second rule ensures the user has a level of comfort in using the device.

The next performance specification involves user recognition. The BreathaLock device must be able to store the fingerprint data of registered user. This specification ensures the user who is using the BreathaLock system has a level of security. If the device were to get in the hands of someone else then the person who is not registered will not be able to unlock the car, however the lock signal is unmodified. Anybody can send out a lock signal.

The last performance specification involves the alcohol gas sensor. When a blood alcohol content is .08 or above, the device must not be able to send an unlock signal. In the state of Florida, anyone who has a blood alcohol content of 0.08% or above is considered over the legal limit for operating a motor vehicle. If the gas sensor reads a blood alcohol content of below 0.08% then the user is able to send out an unlock signal to the motor vehicle.

Specification conclusions

- The system should be no larger than 3” x 3” x 7”.
- The system should be no heavier than 1lb.
- Both sensors must operate with a delay of less than 10 seconds.
- The system must be battery powered within 20V.
- The system must be able to hang onto a key ring.
- The device must be able to store the fingerprint data of registered user.
- The device must incorporate a sanitary breathalyzer.
- When a lock signal is sent, the automotive should be locked.
- When a blood alcohol content is at .08 or above, the device must not be able to transmit an unlock signal.

2.5 Quality of House Analysis

			(+/-) Direction of Improvement Engineering Requirements					
			Power Consumption	Size	Battery Life	Accuracy	Cost	
			-	-	+	+	-	
Marketing Requirement	1) Ease of Use	+					■	
	2) Reliability	+	●		●			▲
	3) Accuracy	+	■	▲	▲	●	●	●
	4) Cost	-			■	●		●
			1W	3"x3"x7"	1 Week	25%	60\$	

Table 1: House of quality trade off table

To display the engineering and marketing requirements, the house of quality trade off table above is used. By using the different shapes listed in the key we show the correlation between the engineering and marketing requirements individually. First we chose to focus on the power consumption and battery life. Specifically, for an everyday handheld device it is extremely important to implement a device that is going to be ready to use for long periods of time without changing batteries or charging the device. Our goal power consumption for power consumption is 1W and battery life of 1 week to ensure that the user will be able to go extended periods of time repeatedly using the device without having to worry about changing a battery. Secondly we investigate the size of the device. Currently most car owners have a small key FOB that allows them to have on them at all times and is pocket size. We are shooting for something of comparable size due to the fact that BreathaLock will be replacing the key FOB. The BreathaLock must be a reasonably small size so the user can carry the device with them at all time without being overly bulky. Finally, we analyze the cost of the device cost. As a preventative device that is intended to be used for any and all adults it is important to market to a low cost so that anyone can afford this product.

As far as marking requirements are concerned it is extremely important to implement a device that is simple and can be used by anyone. The BreathaLock is intended to be a head-ache free device that can be used with very little inconvenience to the user because the user will have to use the device every time they enter their vehicle.

3. Research related to Project Definition

3.1 Existing Similar Projects and Products

3.1.1 SAAB Alcokey

In the mid-2000s SAAB , the automotive company from Sweden, was working on a remote vehicle lock and doubles as a breathalyzer. The remote was named “Alcokey” and it features breathalyzer mouthpiece at the end of the remote. When the remote takes a sample of the user’s blood alcohol content it takes 3 second until the result are shown. If the user is over the legal limit then a red LED light will appear on the remote indicating that the engine cannot be started, however if a green light is indicated then the engine’s electronic immobilizer is release and the vehicle can be started. In addition, the breathalyzer sensor that is integrated within the key remote is semiconductor based and therefore monitors the temperature of the breath sample in the case if the user tries to bypass the device with say a balloon. The Alcokey also comes with a battery indicator and flashes and amber LED when there’s twenty percent of the battery left, in which the user much come to the SAAB dealership to replace the battery. In terms of the range of the device, the key remote is operable at roughly ten meters or thirty-three foot to the vehicle. If a sample is taken outside the vehicle’s range, then the vehicle remote has a three to four second clearance process.

3.1.2 ALCOLOCK V3.

The ALCOLOCK V3 is an in breathalyzer device for private and commercial use. The breathalyzer is interlocked based and is installed within the vehicle’s dashboard and connected through the engine’s ignition system. In terms of operation, before the engine is able to start a sample of the user’s must be taken. If the sample is over the legal limit then engine will not start, however if the user passes the test then the engine can start. Once the engine start’s, ALCOLOCK V3 can be programed to ask for user samples at random times while the engine is running. The device features a tri-colour LRD display to relay the information back to user. Also, the device’s breathalyzer sensor is electrochemical based that allows the device to operate at twelve volts or twenty-four dc volts.

3.1.3 ALCOLOCK’s DRIVESAFE elan

ALCCOLOCK’s DRIVE elan is a breathalyzer that connected through your android device via USB cable. Once connected, the device communicates to an app on the android market to display the user’s sample results. In addition, through the app, the user can make phone calls or be able to tweet results. The breathalyzer sensor is electrochemical based which gives the device a battery life a roughly one thousand samples. In terms of operation, the user must give a continues and moderate sample in which the device returns a result in less than 10 seconds.

3.1.4 BACKtrack Mobile Pro

BACtrack mobile Pro is a police grade breathalyzer that communicates to an app on a mobile device via Bluetooth. The breathalyzer utilizes a fuel cell based sensor that gives

higher level of accuracy. This technology is used by law enforcements, hospitals and clinics. Furthermore, the physical device has a solenoid base air pump inside to ensure the user's breath sample gets to the sensor. In terms of the mobile app, the app can be download on IOS or android. The app saves and stores all the blood alcohol content results over time and also integrate with Uber to make calls for a ride. Further, the app has an estimation feature that predicts when the user's blood alcohol content will reach zero percent. In terms of device operation, the device first be turned on for about 10 seconds to warm up. Lastly, the user blow time is around five seconds.

3.1.5 Daniel Andrade's Arduino MQ-3 breathalyzer project

In 2010 Daniel Andrade built an Arduino based breathalyzer utilizing the MQ-3 gas sensor. The project uses the Arduino Uno, a few red, green, and yellow LEDs, a potentiometer, a few resistors and the MQ-3 sensor. The project had each individual LED in series with a resistor, then each LED that in series with a resistor grounded at one end and the other end connected to the digital pins two up until digital pin eleven on the microcontroller. Once the potentiometer and the breathalyzer sensor was connected to the analog and digital converter within the Arduino, the project is hooked up. The project works by taking the user's breath sample and outputting it to the LEDs. The higher the blood alcohol content the more LEDs light up from green, then yellow, then red.

3.1.6 Nootropic design's Arduino MQ-3 breathalyzer project

Nootropic's circuit utilizes an Arduino Uno, a resistor and a MQ-3 gas sensor. The circuit is powered by the Arduino's onboard 5V regulator and which is connected to the Arduino's ATmega328 analog pin0 that is in series with the resistor. To ensure the breathalyzer got a uniform breath sample, the sensor was place in a small glass jar. In terms of calibrating the device, the method of correlation was. Nootropic designs took voltages readings from the Arduino analog pins at given blood alcohol content levels, after enough samples was taken the device was calibrated. In terms of the output of the project, the output was displayed on the computer using the Serial.print() function.

3.2 Relevant Technologies

3.2.1 Biometric fingerprint scanner

When talking about talking about biometrics, the term refers to the process in which a person's physical trait is detected processed via electronic device. In the case of fingerprint scanners there is a universal two-step process that every sensor operates on: storing the fingerprint pattern of a user and then detecting if a fingerprint pattern matches with the one that was previously stored.

There are many ways a fingerprint scanner can detect and store fingerprint pattern. One common way is optical in which the process involves digitizing finger patterns via visible light. Commonly, an optical sensor is made up of a clear surface to place the finger on. Underneath this clear surface there is a source of light that shines on to the finger in which it is then reflected on to an imaging array which captures the visual image of the fingerprint. Usually the imaging array is either a charge-coupled device (CCD) or a CMOS

based optical imager. For charge-coupled devices the imager is not low light sensitive, in addition the fabrication process is much more complex and thus more expensive. CMOS based optical imagers however, are more easily made thus making the optical scanner much less expensive. The disadvantage of this type of sensor is that when the clear surface is smudge or if the finger is dirty then the optical scanner cannot properly process the image.

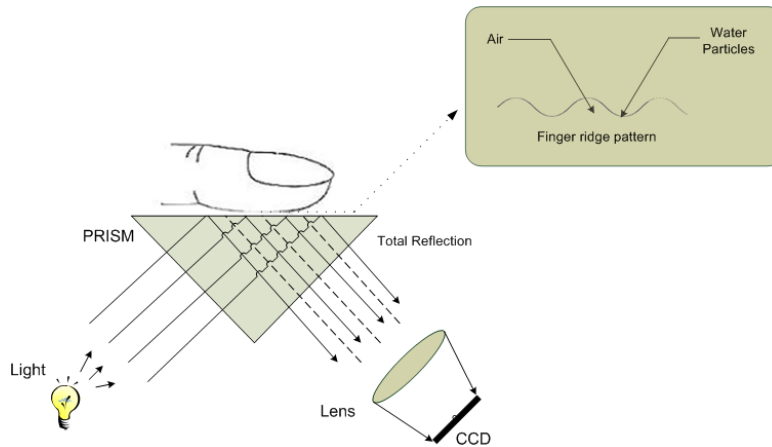


Figure 2: Optical Fingerprint Diagram

Another common method a fingerprint scanner can detect and store a fingerprint pattern is through capacitive touch. Capacitive touch fingerprint scanners are categorized in two categories: passive and active. Passive touch scanner works by having each pixel of the image processor acting as one side of a parallel plate capacitor and a user's finger as the other plate to the capacitor. Since the capacitive values between the image and the dermal layer of the skin are known, the whole array of pixels can map out the valleys and ridges of a user finger, thus making each finger distinguishable between one another.

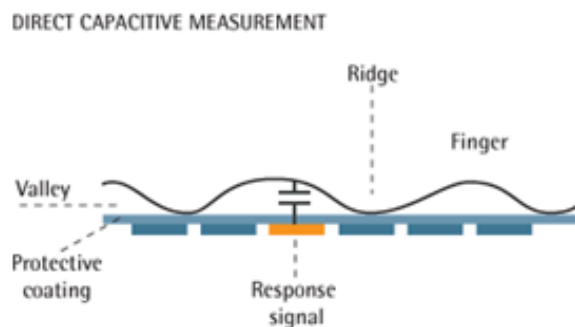


Figure 3: Passive capacitive touch Diagram

Active capacitive touch finger print scanners work by a creating charge onto the skin before sampling takes place. After the charging process, the effective capacitor is

charged thus creating an electric field between the finger and sensor that follows the ridges and valleys of a user's finger. On the discharging process, the voltage between the skin and the sensor is measured and compared to the charged value as a reference to compute the capacitance. After computing the capacitance, the scanner mathematically calculates the distance between the finger and scanner. Upon applying the charge and discharge process to an array, the valleys and ridges are mapped out on a person's finger.

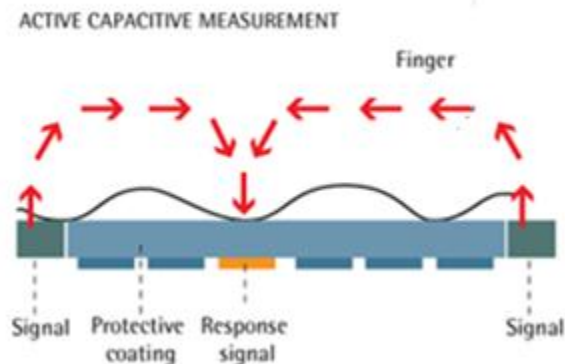


Figure 4: Active capacitive touch Diagram

After scanning for valleys and ridges, then the analog values are converted to a string of binary values in which it is stored and compared to the next set binary values.

3.2.2 Gas sensors

In the case of a foreign gas there is a need for detection. There are many types of gas detectors but through different types technology. Commonly gas sensors include electrochemical and semiconductor.

When referring to electrochemical gas sensors, the structure of the device must be clearly understood. Commonly, electrochemical gas sensors contain two or more electrodes in contact with an electrolyte. The electrodes themselves are a high surface area metal that is covered in a hydrophobic membrane. The sensor allows certain gases to pass through the porous membrane in which it is then chemically oxidizes or reduced. The amount of current generated is determined by the amount of gas that passes through the membrane and oxidizes. Since the size of the membrane can be manipulated during the fabrication process, then the type of gas the sensor can detect can be tailored to the desired gas. One advantage of this technology is that the membrane that surrounds the electrode acts as a physical barrier then this allows the detector to be more stable thus requiring less maintenance over time. However, a disadvantage of electrochemical gas sensors is that is susceptible to corrosion. Since the device is subject to any gas to come in contact with the porous membrane, the membrane is subject to contamination and deterioration.

The other common type of gas sensor is based on semiconductor technology. The principle behind semiconductor based gas sensors is that when the desired gas of detection comes into direct contact with the sensor itself a chemical reaction occurs.

Since a reaction occurs on the surface of the semiconductor itself, it is common for the resistance through semiconductor to either drop or increase depending on the anatomy of the semiconductor. When once the resistance has drop the change in electric current the device is detected and analyzed from which the concertation of gas is recognized.

In addition to common sensor types, another important property of gas sensors that is essential to the device itself is calibration. All gas sensors regardless of the type of technology that it is based off needs to be calibrated routinely. If the gas sensor is more mobile or exposed to many other elements upon taking samples, then the routine will check to see if the device is calibrated properly is more frequent in contrast to a device that stays in one place or only takes samples containing fewer elements. One of the simplest way to calibrate any sensor is to expose the sensor to a known concentration of the desired gas of detection. If the sensor isn't reading the correct concentration, then the difference between the output and the controlled sample can be taken and added or subtracted to correctly offset the device. To improve the accuracy of the device, it is quite common to repeat the sensor correction test, this will result in multiple values of offset in which the average value can be taken and used. The more offset values the more accurate the device is.

3.2.3 Bluetooth

One of the most common methods of device communication is Bluetooth connection. Bluetooth is a global wireless communication standard that is implemented through radio waves. Usually within the bandwidth of 2.4 GHz to 2.485 GHz, the radios wave allows multiple devices to be connected at once, the master Bluetooth device can have up to seven devices be connected all at once. In terms generations of Bluetooth that are most commonly found in devices today, there are three: Bluetooth 3.0, Bluetooth 4.0, and most recently Bluetooth 5. Of those three types of generations there are two branches of types of Bluetooth: Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) and Bluetooth with low energy (LE). Typically, Bluetooth Basic Rate/Enhanced Data Rate are found in stereo speakers, headsets, and computers, while Bluetooth with low energy, as the name implies, are found in device that operate on low power such as wearables and other IoT devices.

3.2.4 Battery

When talking about battery technology we must define what a battery does. A battery is a device that has energy stored in the form of chemical energy and then converts it to electrical energy. Every battery has two terminals: a positive terminal known at the cathode and the negative terminal known as the anode. Separating the cathode and anode is the electrolyte. The role of the electrolyte is to provide a means of a chemical reaction to build up electron at the anode. Since there is a buildup of electrons at the anode, there is a potential difference between the anode and the cathode thus when the battery is under load there will be electron flow from anode to cathode.

There are two classifications of batteries rechargeable and non-rechargeable. Non-rechargeable refers to the types of battery that cannot be reused over again. These types

of batteries aren't rechargeable because the chemical reaction that is provided by the electrolyte cannot be reversed. Typically, when a non-rechargeable is used, a brand new one replaces it.

The other type of battery classification is the rechargeable battery. Rechargeable batteries are able to recharge because the chemical reaction that is provided by the electrolyte is able to reverse its process thus restoring the anode and cathode back to its original state. This original state can again provide full power.

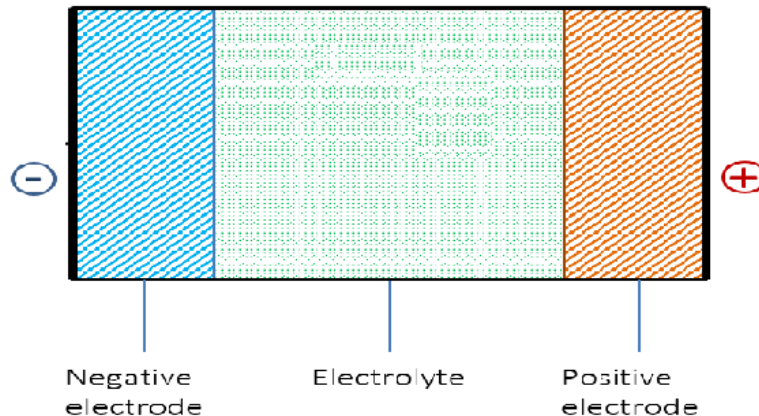


Figure 5: Battery Diagram

3.3 Strategic Components and Part Selections

3.3.1 Microcontroller

When defining a microcontroller, a microcontroller is a computer system that typically has memory, a processor, input/output ports, serial ports, timers, analog to digital converters, and digital and analog converters to perform a task. Usually in the form of a single integrated circuit, a microcontroller is much smaller than a computer as its main uses are found embedded within a main system. However, the term "microcontroller" can easily be confused with the term "microprocessor". The microprocessor itself does not have its memory and other peripherals contained on board with the chip, but rather placed externally, leaving room for upgrades.

Before talking about comparing and strategically selecting microcontrollers, there are categories that microcontrollers are divided into that needs to be defined: bits, memory, instruction sets, architecture. When referring to the number of "bits" a microcontroller can process at once, we are referring to most amounts of ones and zeros the processor can read at one moment. Usually the processor's bit numbers are 8bits, 16bits, 32bits, and 64bits. The higher the bit number, the faster the processor can perform tasks. In terms of memory, memory is used to store data and programs to fetch later on. Typically, a microcontroller has a fixed amount of RAM and ROM or other flash memories because the microcontroller itself comes in a single IC package. The instruction set of a microcontroller is the communication between the microcontroller's software to hardware. Lastly, a feature of a microcontroller that needs to be noted is the microcontroller's

architecture. Typical architectures found on most microcontrollers today are the 8051, PIC, AVR, and ARM.

3.3.1.1 Arduino Uno

The Arduino Uno is a microcontroller assembly that is based on the through hole version of the ATmega328P. The Uno operates at 5V with a maximum input tolerance between 6-20V. In addition, the microcontroller has 28 pins of which there are 14 digital pins, where 6 of those pins can be used for PWM; also there are 6 analog pins. In terms of memory, the Arduino Uno has 2 Kilobytes of volatile SRAM memory, 1 kilobyte of nonvolatile EEPROM memory, and 32 kilobyte of nonvolatile flash memory of which 0.5 kilobytes is allocated for the bootloader.

Additionally, the ATmega328P features three timers/counters, internal and external interrupts, a programmable USART, a 6 channel 10-Bit analog and digital converter, a programmable watchdog timer, and five software selectable power saving modes

MODE	Functionality
Idle mode	Stops CPU, but keeps SRAM, Timers/counters, USART, 2-wire serial interface on
Power-down mode	Save contents inside register and freezes everything else until interrupt occurs
Power-save mode	asynchronous timer stays on, everything else is off
Standby mode	crystal oscillator is on, everything else is off
ADC Noise Reduction mode	only asynchronous timer and ADC is on, everything else is off

Table 2: Arduino Uno Low power mode table

The serial interface uses Universal Asynchronous Receiver / Transmitter (UART) where the pins RX and TX in which UART typically has are connected through to a USB–UART converter circuit. The ATmega328P also has Serial Peripheral Interface(SPI). Besides using it as another option for serial interfacing, it can also be used to program the microcontroller using a standalone programmer.

In terms of power, the Arduino Uno can be powered on either by USB or a DC power jack. The Arduino Uno also has a regulated 5-volt power supply and a 3.3-volt power supply that can supply up to 50mA. The 5V power supply comes from the NCP1117ST50T3G regulator where the input voltage is from the DC power jack that is then connected to a surface mount diode to provide circuit protection. The output of the regulator is then connected to the rest of the 5V circuit where the user can access the power and also to the input of the 3.3V voltage regulator: LP2985-33DBVR. If however, the user decides to power on the Arduino Uno via the USB rather than the DC power jack, then the 5V line of

the USB is connected to the drain of the an FDN340P, a P-channel MOSFET. Furthermore, the source terminal of the MOSFET is connected to the 5V network where the user can access the regulated power and the gate terminal of the MOSFET is connected to an output of a LMV358 comparator. The comparator acts as a switch to turn the MOSFET on and off.

Advantage	Disadvantage
- removable microcontroller	- Size
- Female Connectors	- Video/audio peripherals
- Shield capability	
- Pre-existing Female Pin connectors	

Table 3: Arduino Uno Advantage/Disadvantage table

3.3.1.2 Arduino ProMicro

When looking at the Arduino ProMicro, there are similar features to that of any microcontroller to that of the Arduino family. The Arduino ProMicro is based on the ATmega32U4 microcontroller. This microcontroller has 32 pins, of which 12 of those pins are allocated for analog inputs while the other 27 pins are for digital input/output channels. In terms of memory, the Arduino ProMicro has 2.5 kilobytes of SRAM, 1kilobyte of EEPROM, and 32 kilobytes of flash memory of which 4 kilobytes are used by the bootloader. In terms of power, the Arduino ProMicro operates at 5V however, the input tolerance limit is 5V-12V. Once again like most Arduino microcontrollers, the device can be powered on through the USB or an external power supply. If the user is using an external power supply then connections must be made through the Vin and Gnd pins since there are no other terminals for external power source.

In terms of device communications, the Arduino ProMicro has many ports for communications. The device has TTL serial communication in which pins 0 and 1 are used as RX and TX respectively to receive and transmit data. Also, there is TWI in which pins 2 and pins 3 are used. Implementations can be done by using the Wire library. Lastly, the Arduino ProMicro allows for CDC communication through USB where the device act as an open com port to other software on the computer.

Advantage	Disadvantage
- Size	- nonremovable microcontroller
- Extra Digital input pins	- Not shield capable
- Extra analog input pins	- Video/audio peripherals
	- No pre-existing connectors

Table 4: Arduino ProMicro Advantage/Disadvantage table

3.3.1.3 Arduino Mini

Lastly we take a look at the Arduino Mini. The Arduino Mini, similar to the Arduino Uno, runs on the ATmega328. Since the two microcontroller have similar processor, the peripherals will be similar as well. The Arduino Mini has 22 pins, of which 14 are used for digital input/output, and of those 14 digital pins, 6 can be used for pulse width modulations(PWM). In terms of device memory, the available memory is the exact same as to that of the Arduino Uno.

The advantage to using the Arduino Mini comes from the size of the device. The device itself is 30mm x 18mm. This allows the device to be portable, and easy to be stored away in much tighter place compared to the previous Arduino Microcontrollers mentioned before.

Advantage	Disadvantage
- Size	- Non-removable microcontroller
	- Video/audio peripherals
	- Not shield capable
	- Needs FTDI board to program
	- UART capability
	- No pre-existing connectors

Table 5: Arduino Mini Advantage/Disadvantage table

3.3.1.4 Raspberry Pi 3 model B

Raspberry Pi 3 model B is the third generation of Raspberry Pi. The Raspberry Pi tends to be more of a mini-computer whereas the Arduino family is a microcontroller. For the purpose of this paper, the Raspberry Pi family will be treated as a special kind of microcontroller. The device runs on the 1.2 GHz 64-bit quad-core Armv8 processor with 1 gigabyte of RAM. In terms of peripherals, the device has:

- 40 General Purpose input/output pins (GPIO)
- 802.11n Wireless LAN
- Bluetooth Low Energy
- 4 USB ports
- 1 HDMI port
- 1 Ethernet port
- Camera interface
- Display interface
- MicroSD card slot
- 3.5mm Arduino port

The advantage to using the Raspberry Pi is through the video/audio capability readily available peripheral the device has to offer compared to the of the Arduino.

Advantage	Disadvantage
- Video/audio peripherals	- Non-removable microcontroller
- Number of GPIO	
- Communication peripherals	

Table 6: Raspberry Pi 3 Model B Advantage/Disadvantage table

3.3.1.5 MSP430G2211N14

The MSP430 is a 16-bit microcontroller from Texas Instrument that follows RISC architecture. In terms of powering on the device, the MSP430 operates between 1.8V to 3.6V. Furthermore, the MSP430 has 5 power saving mode in addition to a wakeup time from standby mode of less than 1 μ S. In terms of memory, the MSP430 features 128 kilobytes of RAM.

Advantage	Disadvantage
- Ultra Lower power	- Not user friendly in group31
- Fast wake-up time	

Table 7: MSP430 Advantage/Disadvantage table

Upon reviewing over the advantages and disadvantages of each microcontroller, a decision table is made to furthermore strategically pick the right microcontroller for the BreathaLock project. In the decision table, criterions were carefully picked and weighted according to the project's needs, from 1 being the lowest weight possible to 5 being the highest weight. The criterion: Dimension refers to the physical dimensions of the microcontroller package. The criterion: Programmability refers to the ability to access the microcontroller and program the desired task. Maintainability refers to the microcontroller's ability to be replaced if any accidents should that should arise. Cost is the cost of the microcontroller per unit, the higher the cost score the less expensive the microcontroller is.

Decision Table				Microcontroller scores							
Criterion	Value Weight	Uno	Total	Micro	Total	Mini	Total	Pi	Total	MSP430	Total
Dimension	3	3	9	4	12	5	15	1	3	3	9
Programmability	4	5	20	5	20	2	8	4	16	2	8
Maintainability	5	5	25	1	5	1	5	1	5	2	10
Peripherals	2	2	4	2	4	2	4	5	10	3	6
Cost	2	5	10	2	4	2	4	1	2	5	10
Total		68		45		36		36		43	

Table 8: Microcontroller Decision Table

3.3.2 Bluetooth

Choosing a Bluetooth module for our project was difficult. However, we were able to limit our options to two choices.

3.3.2.1 Adafruit Bluefruit LE UART Friend

The Adafruit Bluefruit LE UART Friend has a ARM Cortex M0 core running at roughly 16MHz, has 256kb of memory 32kb of static RAM. The device includes voltage regulation on board which is important for our needs. Adafruit's board also utilizes a UART transport scheme at a 9600 baud rate with hardware flow control such as CTS+RTS a RS-232 standard which can be enabled if necessary. However, this feature seems that it may not be useful. The particular module fits within our size and weight specification at 21mm x 32mm x 5mm (WxLxH) and 3.4g. The module also uses Bluetooth 4.0

Advantages

- Compatible with our logic board
- Very modifiable
- Well documented
- Small

Disadvantages

- Complex Driver
- Expensive cost wise

3.3.2.2 Phantom YoYo JY-MCU Arduino Bluetooth Wireless Serial

The Phantom JY-MCU Bluetooth module has similar specs to the Adafruit module; however, with some key differences. The Bluetooth standard that is utilized is the older 2.0 EDR standard and the size is larger at 4.4 cm x 1.6 cm x 0.7 cm which poses problems for our specification as we would like something smaller. The voltage requirement of this device is 3.3V.

Advantages

- Simple pin layout
- Cheap cost

Disadvantages

- Poorly Documented
- Does not meet size specifications
- Old Bluetooth specification

3.3.2.3 BLE Nano - nRF51822

The BLE nano-nRF51822 features an ARM Cortex-M0 SoC. In addition, the nano-nRF51822 uses Bluetooth 4.1 which is the latest technology in low power Bluetooth communication. In addition, the BLE nano is only 18.5mm x 21.0mm, making it a good candidate for Blue communication as portability is important in our BreathaLock project. In terms of powering on the device the operating voltage is between 1.8V to 3.3V.

Advantages

- Size
- Cheap cost
- ultra Low power consumption
- Comes with headers
- Works with IOS and android

Disadvantages

- no onboard storage

In the decision table below, the highest weight possible for a criterion is 5 while 1 is the lower weight possible.

Decision Table			Bluetooth Module scores				
Criterion	Value (Weight)	Bluefruit LE	Total	YoYo JY-MCU	Total	BLE Nano	Total
Dimension	3	4	12	3	9	4	12
On board storage	4	4	16	0	0	0	0
Bluetooth protocol	5	5	25	3	15	5	25
Cost	3	4	12	2	6	2	6
Total		65		30		43	

Table 9: Decision Table

3.3.3 Fingerprint Scanner

When strategically selecting a biometric fingering print scanner, we needed to check the variety and the availability of the standalone technology that's on the market. It turns out that even though fingerprint sensing is very common amongst technology today, the standalone technology that is available on the market is very low. This is most likely due to the fact that pre-existing fingerprint technology is uniquely designed by companies to be coupled with their existing product. There are only two finger print scanner modules that are available in the market: the TTL(GT-511C3) and Adafruit.com's fingerprint scanner (product ID:751).

The GT-511C3 features an ARM Cortex M3 Core CPU embedded into the package. Additionally, the device can image a size of 202 x 258 pixels with a resolution of 450 dpi. The false acceptance rate is less than 0.001% and a false rejection rate is less than 0.1%. In terms of powering on the device, the operating voltage is between 3.3 to 6V and the operating current is less than, 130mA. The baud rate for this device or rather the maximum amount of bits per second the serial port is capable of transferring is 9600 bits.

When we look at the fingerprint sensor found on Adruit.com, the sensor has the exact same features and specifications as the GT-511C3 however the thing that differentiate it from the GT-511C3 is the baud rate. The baud rate for optical fingerprint sensor found on Adafruit.com is 9600, 19200, 28800, 38400, and 57600. This mean that the number of bits second the serial port can transfer can be varied depending on the user. Regardless, the device has a default baud rate of 57600 bits thus making it much faster and much more efficient than the GT-511C3.

However, when comparing the two devices to strategically select the right fingerprint scanner for the BreathaLock project, the baud rate is not an important criterion to compare by but rather then dimensions of each device. Both devices are both optical sensors, and by the nature of their technology the two devices are much larger than that of capacitive touch fingerprint scanner. In our case, the smaller the device is, the better suited the

device is for the project. In the decision table below the maximum weight possible is 5 and the lowest weight possible is 1

Decision Table		Fingerprint scores			
Criterion	Value(Weight)	GT-511C3	Total	Adafruit.com's fingerprint scanner	Total
Dimensions	5	3	15	4	20
Power	4	4	16	4	16
Data rate	1	3	3	4	4
False acceptance rate	3	5	15	5	15
False rejection rate	3	5	15	5	15
Total		64		70	

Table 10: Fingerprint Scanner Decision Table

3.3.4 Breathalyzer sensor

Once again when strategically selecting a breathalyzer sensor, we needed to check the variety and the availability of the standalone technology that is on the market. It turns out that even though gas sensing technology is common, finding the right sensor for the with the right sensitivity can be difficult. There are only two alcohol sensors available on the market that can detect alcohol on a sensitive level: the MQ-3, and the MR513 alcohol sensor.

3.3.4.1 MQ-3 Sensor

The MQ-3 sensor is a semiconductor-based sensor that can detect concentrations of alcohol within the scope of 25 to 550 parts per million(ppm). In terms of how this device operates, the six terminal device has allocated two sets of pins to power on a heating element, while the remaining four pins acts as 2 sets of leads to a resistor. The heating element dries up the surrounding air to prepare the existence of alcohol gas. Upon the presence of alcohol gas, there will be a differential in conductivity. Then the differential will then be translated into an analog signal in which the analog signal can tell us the amount of alcohol present in the air. One key thing to note on this device is that for the

first time using the device the sensor must be powered on for 48 hours. This ensures the heating element can work properly over time.

To further detail on how the device can take the differential in conductivity and translate it into an MR513 analog signal we must note the two sets of pins on the device (4 pins). One set of pins act as one end of a variable resistor and the other set acts as the other end. However, since the variable resistor is made up of tin dioxide (SnO₂) which is a semiconductor, the true conductivity of the device is unknown due to the temperature dependency property of semiconductors. With that being said, the variable resistor can be connected in series with a load in which the output is the voltage across the load. Once alcohol gas comes into contact with the semiconductor a differential in conductivity will occur across the variable resistor, this change can be measured by measuring the voltage across the load before and after the presence of alcohol. Upon measuring the change in voltage, the concentration of alcohol gas can be known.

Advantage	Disadvantage
- low power	- water sensitive
- Size	- vibration sensitive
- Load resistance adjustable	- break-in period
	- Susceptible to corrosion

Table 11: MQ-3 Advantage/Disadvantage table

3.3.4.2 MR513 Sensor

The MR513 Sensor is based on semiconductor technology. The MR513 consist of a detection element and a compensation element placed in Wheatstone configuration. When there is alcohol gas within the sensor, the voltage of the Wheatstone bridge will be change thus telling us how much alcohol gas is present. The Sensitivity for this device is 100 parts per million. In terms of powering on the device, the alcohol sensor run on 3 volts with a working current or around 100 milliamps.

Advantage	Disadvantage
- low power	- only 1 datasheet
- Size	- Data sheet isn't detailed
- only 4 terminal device	

Table 12: MR513 Advantage/Disadvantage table

Decision Table		Gas Sensor scores			
Criterion	Value(Weight)	MQ-3 Sensor	Total	MR513	Total
Dimensions	2	3	6	3	6
Power	3	3	9	4	12
Documentation	5	4	20	2	10
Total		35		28	

Table 13: Gas Sensor Decision Table

3.3.5 Power supply

In terms of powering on the project, we must understand how much power each component needs to operate. From the table below we can see each component can operate under 10V.

	Operating voltage	Peak Current
Arduino Uno	7.0-12V DC	-
Bluetooth	5V	
Fingerprint scanner	3.6- 6.0 DC	150mA
Breathalyzer sensor	5.0V AC or DC	180 mA
Car remote	3V	-

Table 14: Selected Component Operating voltage

Since the Arduino Uno has an onboard 5V and 3V regulator, we can power on the entire BreathaLock project via 9 volts. To achieve this 9V there many options in which we go about.

3.3.5.1 Power option 1

To achieve the desired 9V that powers the entire BreathaLock project, we can use a standard rechargeable 9V battery. The advantage to using a standard rechargeable 9V comes from the storage capacity of the battery and the recharge-ability of the battery. Typically, a standard 9V can hold much power than that of a coin cell battery. The disadvantage of the standard 9V battery is that the size of battery is big and bulky.

3.3.5.2 Power option 2

Power option 2 consists of using three 3V coin cell battery. When connecting batteries in series, the overall voltage is equal to the sum of each individual voltage. In the case of power option 2, three 3V coin cell battery adds up to 9V. The advantage to using three smaller batteries is the amount of space the three-coin cell battery takes up when compared to the size of the standard 9V battery. The disadvantage to from using three 3V coin cell battery is the amount power the batteries can hold compared to a standard 9V battery.

Decision Table		Power Option Scores			
Criterion	Value(Weight)	Option 1	Total	Option 2	Total
Dimensions	5	2	10	4	20
Power storage	3	3	9	4	12
Total		19		32	

Table 15: Power Option Decision Table

3.3.6 Key remote

Before picking a motor key remote to use for the project we must identify the car that will be used for the project. Nicholas Fraser volunteer his 2005 Ford F150 for the project.

3.3.6.1 300-0247ES Universal Car Remote

The 300-0247ES universal car remote is a 6 button device that does have compatibility for the 2005 Ford-FF150. The 6 buttons on the device are for lock, unlock, open trunk, panic alarm, and two auxiliary buttons for vehicle functions. Vehicle functions include, van door, remote start, convertible top etc. In terms of powering on the device, the universal car remote found at Walmart operate using a standard lithium 3V coin cell battery.

3.3.6.2 KPT1306 key remote

The KPT1306 key remote is compatible for the 2005 Ford-F150. The device features 3 buttons for lock, unlock, and panic. In terms of powering on the device once again the device runs on a stand lithium 3V coin cell battery. In terms of programming the device, the ignition must be turned from off to run 8 times within 10 seconds. After the 8th turn, the user has 20 seconds, pressing any key on the keyless remote to enable the device to be programmed. After that, the next button needs to be pressed again to confirm the second programming. Lastly, turning the ignition to off will end the programming phase.

3.3.6.3 K410 Car Remote Central Lock Locking Entry System

Though this product comes with two double button key remotes, it also comes with the remote receiver which is still relevant to consider for the BreathaLock project. In terms

of powering on the whole system the requires 12voltes where the max current draw is 15A. In the decision table below, the highest weight possible for a criterion is 5 while 1 is the lower weight possible.

Decision Table		Keyless remote Scores					
Criterion	Value(Weight)	300-0247ES	Total	KPT1306	Total	K410	Total
Simplicity	5	4	20	5	25	2	10
Power	3	3	9	3	9	2	6
Quantity	2	1	2	4	8	4	8
Total		31		42		24	

Table 16: Power option Decision Table

3.4 Architectures and Related Diagrams

3.4.1 Microcontroller Architecture

In terms of microcontroller architecture, we will be discussing the ATmega328P's architecture. The ATmega328P is an 8bit- AVR RISC-based microcontroller. The term AVR RISC tells us that microcontroller follows a modified version of the Harvard architecture with reduced instruction set computing. In terms the available register the 32 general purpose register that is all directly connected to the Arithmetic Logic Unit (ALU). This direct connection allows for simultaneous access of each register upon an execution of an instruction.

The Harvard architecture says that volatile and nonvolatile memories are treated as 2 separate systems, whereas the popular von Neumann architecture only has a single memory system. One advantage of the Harvard architecture is that the ability to simultaneously access the programs and data elements.

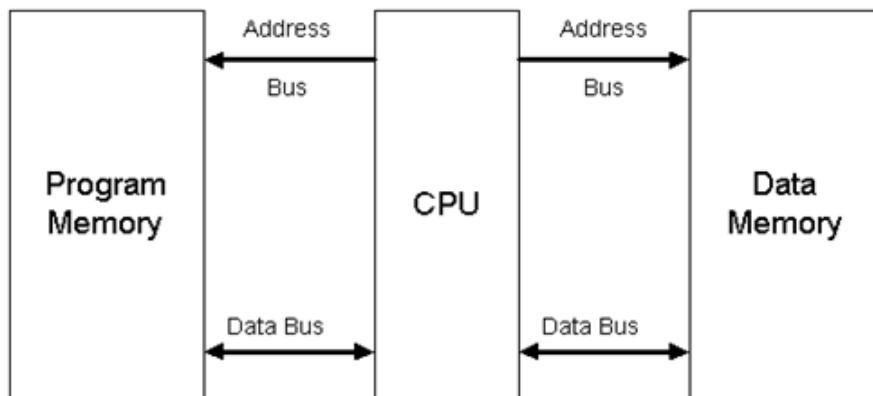


Figure 6: Harvard architecture

Furthermore, the ATmega328P follows reduced instruction set computing. This means that the instruction set for this device is simplified by cutting down on the complexity of each available instruction. This allows the device to process down simple instructions at every single clock cycle thus achieving a high throughput at around 1 MIPS per MHz. If the device is however CISC, the exact opposite of RICS, the instruction set would have a degree of complexity thus requiring more resources to process down each instruction. The focus of CICS is to process down instructions with the fewest lines of code possible at the cost more of time, however, the focus of RISC to process down instructions fast at the cost of more lines of code.

3.5 Parts Selection Summary

When talking about the selection of parts, we must first look at the decision table for each category of parts. For the microcontroller that will be governing the project, we chose to pick the Arduino Uno with a few minor adjustments. For the BreathaLock project we will be cloning our own Arduino Uno. This will give us the freedom to design our project under one PCB package. Additionally, the microcontroller will have its USB bridge removed to save power and board space since our project does not have any use for that component to permanently be on the PCB. In terms of supplying the regulated 5V and 3V, we will be using the LM7805 and the LT1761ES5-3 respectively. Again, the reason why we're building our own Arduino Uno is to be able to tailor our design to be compact and lower power consumption.

In terms of why we chose the ATmega328P, we chose the microcontroller because of its low-power capabilities as having 4 modes of low power is an asset to our BreathaLock project. Furthermore, we chose the ATmega328P because its ability to be placed in a dip socket. This allows us to replace a broken microcontroller without removing any circuit components if need be.

For the Bluetooth module, we chose the Bluefruit LE UART module. We chose this component because of its size. The fact the entire module is 21mm x 32mm x 5mm becomes an advantage to us when we are creating a device that is meant to be portable. Additionally, we chose this component because of its low power capability as the device conveniently runs on 5V which makes it Arduino Uno friendly. Furthermore, the device has 256 kilobytes of flash memory. This allows us to have multiple profiles saved onto the device.

For the fingerprint scanner, we chose to go with the one found on Adafruit.com. We chose this device because of the form fitting factor this device has to offer when compared to the GT-511C3. The GT-511C3 has undesirable hinges protruding off to the sides thus making the device itself awkward to make portable.

For the gas sensor we chose to use the MQ-3 alcohol gas sensor. We chose this device over the MR513 because of the available documentation the MQ-3 has compared to the MR513. The MR513 only has 1 available datasheet that isn't detailed in how the device

operates. However, the MQ-3 has multiple documentations along with meeting the required alcohol sensitivity for the project.

For the keyless car remote, we chose to use the KPT1306 key remote. We chose this remote because it was the simplest keyless remote when compared to the other two options. For the BreathaLock project, since we are only interested in the transmission of the unlock signal and other device with more than two button will complicate the process of modifying the remote. In addition, since we are using 2005 Ford-F150, there is already a preinstalled remote receive on the vehicle thus using the K410 Car Remote Central Lock Locking Entry System is not necessary.

Lastly, for powering on the entire device we chose to go with three 3V coin cell battery (power option 2). We chose power option 2 because of the portability coin cells batteries have to offer. In addition, carrying around a standard 9V battery would not promote portability with the BreathaLock project.

Part selection summary		
Part	Selection	Cost (before tax and shipping)
Microcontroller	Arduino Uno	\$29.99
Bluetooth module	Bluefruit LE UART	\$17.50
Alcohol gas sensor	MQ-3	\$4.95
Fingerprint Sensor	Adafruit.com's	\$49.95
Battery	three 3V battery	\$10.00
Keyless Remote		\$7.95
Total		\$120.34

Table 17: Part selection summary Table

4. Related Standards and Realistic Design Constraints

4.1 Standards

4.1.1 Bluetooth Standard

The Bluetooth specification defines the technology that developers can use to create the devices that communicate between other Bluetooth applicable devices. The Bluetooth specification is overseen by a Special Interest Group (SIG) and is regularly updated to meet new needs.

Summary

The Bluetooth standard specifies two “flavors” of Bluetooth are as follows:

- **Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR)** - which is an older standard adopted as version 2.0/2.1. The entire spectrum of the RF part of the physical layer (see figure XX) operates in an unlicensed industrial, scientific and medical (ISM) radio band at 2.4GHz. In order to tackle interference, the any device may encounter Bluetooth employs a frequency-hop transceiver which assists in identifying “good” frequency by avoiding “bad” frequencies that may be in use, are experiencing selective fading, or perhaps those bands are being actively jammed. BR/EDR includes two data rates Basic Data Rate and Enhanced Data Rate. Basic Rate, supports a bit rate of 1Mbps while Enhanced Data Rate supports gross air bitrate of 2Mbps. Making this an ideal standard for relatively short-ranges, continuous wireless communications, which is ideal for audio streaming.
- **Bluetooth with Low Energy (BLE)** - This is the newest standard also known as 4.0/4.1/4.2. This newer standard was developed with power-efficiency in mind. Devices that utilize small or isolated power sources; such as, button cell batteries or solar power. This platform is more heavily supported for every major operating system and allows for seamless development for a board. Opposite of BR/EDR, BLE offers short burst of long-range radio connections which is ideal for applications in the field of Internet of Things or devices that do not require continuous connection but depend on battery longevity. This version achieves this energy efficient mode by having three modes Ultra-low peak, average and idle mode. BLE offers several security enhancements versus its predecessors such as, digital signing, key generation, encryption as is government-grade security with 128-bit AES data encryption, etc. (see sec. 1.1.1.1.3. *Bluetooth Security* below). These features make it ideal for variety of applications such as security systems, portable devices, fitness monitors, proximity sensors, and breathalyzers.

Each implementation has different use cases and each implementation uses a different chipset to meet essential hardware requirements. dual-mode chipsets are available to support single devices such as smartphones or tablets that need to connect to both BR/EDR devices (such as audio headsets) and LE devices (such as wearables or retail beacons)

4.1.1.1 Core System Architecture

While each implementation has specific requirements that are detailed in the Bluetooth specification, the Bluetooth core system architecture has many consistent elements. The system includes an RF transceiver, baseband and protocol stacks that enable devices to connect and exchange a variety of classes of data.

Bluetooth devices exchange protocol signaling according to the Bluetooth specification. Core system protocols are the radio (RF) protocol, link control (LC) protocol, link manager (LM) protocol and logical link control and adaptation protocol (L2CAP), all of which are fully defined in the Bluetooth specification.

The lowest three system layers—the radio, link control and link manager protocols—are often grouped into a subsystem known as the Bluetooth controller. This is a common implementation that uses an optional standard interface—the Host to Controller Interface (HCI)—that enables two-way communication with the remainder of the Bluetooth system, called the Bluetooth host.

The primary controller may be one of the following configurations, depending on use case:

- BR/EDR controller including the radio, baseband, Link Manager and optionally HCI
- LE controller including the LE PHY, Link Layer and optionally HCI
- Combined BR/EDR controller and LE controller, with one Bluetooth device address shared by the combined controller

The Bluetooth specification enables interoperability between systems by defining the protocol messages that are exchanged between equivalent layers. It also enables interoperability between independent Bluetooth subsystems by defining the common interface between Bluetooth controllers and Bluetooth hosts.

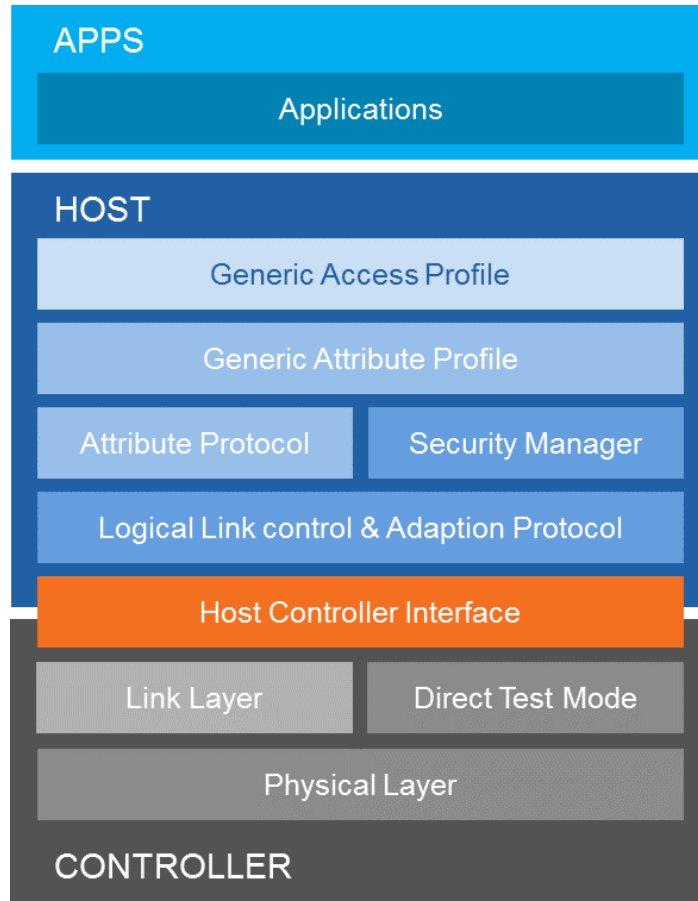


Figure 7: System Architecture

Physical (PHY) Layer:

Controls transmission/receiving of the 2.4Ghz radio with Bluetooth communication channels. BR/EDR provides more channels with narrower bandwidth, while LE uses fewer channels but broader bandwidth.

Link Layer:

Defines packet structure/channels, discovery/connection procedure and sends/receives data.

Direct Test Mode:

Allows testers to instruct the PHY layer to transmit or receive a given sequence of packets, submitting commands to it either via the HCI or via a 2-wire UART interface.

Host to Controller Interface (HCI):

Optional standard interface between the Bluetooth controller subsystem (bottom three layers) and the Bluetooth host.

Logical Link Control and Adaptation Protocol (L2CAP) Layer:

A packet-based protocol that transmits packets to the HCI or directly to the Link Manager in a hostless system. Supports higher-level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information to higher layers.

Attribute Protocol (ATT):

Defines the client/server protocol for data exchange once a connection is established. Attributes are grouped together into meaningful services using the Generic Attribute Profile (GATT). ATT is used in LE implementations and occasionally in BR/EDR implementations.

Security Manager:

Defines the protocol and behavior that manages pairing integrity, authentication and encryption between Bluetooth devices, and provides a toolbox of security functions that other components use to support almost any level of security needed by diverse applications.

Generic Attribute Profile (GATT):

Using the Attribute Protocol, GATT groups services that encapsulate the behavior of part of a device and describes a use case, roles and general behaviors based on the GATT functionality. Its service framework defines procedures and formats of services and their characteristics, including discovering, reading, writing, notifying and indicating characteristics, as well as configuring the broadcast of characteristics. GATT is used only in Bluetooth LE implementations.

Generic Access Profile (GAP):

Works in conjunction with GATT in Bluetooth LE implementations to define the procedures and roles related to the discovery of Bluetooth devices and sharing information, and link management aspects of connecting to Bluetooth devices.

4.1.1.2 Bluetooth Security

To ensure communication via Bluetooth is secure, BLE achieves this by utilizing several security features, the Bluetooth specification gives several features to cover the encryption, trust, data integrity and privacy of the user's data. The processes are described as follows:

- **Pairing** - this mechanism is the process where devices involved in communication exchange their identity information to set up trust and get the encryption keys ready for future data exchange. Bluetooth has a few options in regards to pairing. In version 4.0 and 4.1 of the specification, Bluetooth uses the Secure Simple Pairing model (SSP) a form of public key cryptography this promotes an effective mitigation strategy for Man-In-The-Middle (MITM) attacks. The devices will often choose one method from the following: Just Works, Passkey Entry, Numeric Comparison, and OOB.

- **Key Generation** - Keying BLE is performed by the Host on each device independently. Key generation in BR/EDR is performed in the Controller. By performing this on the Host, the key generation algorithms can be upgraded without changing the device. The following keys are exchanged between primary device and secondary device: Connection Signature Resolving Key (CSRK) for authentication of data, and Identity Resolving Key (IRK) for the devices identity and privacy. The two keys pub and priv key are generated in the host and a SSK is generated by combining information from each device involved in communication.
- **Encryption** - Bluetooth with LE uses AES-CCM cryptography. Both version of Bluetooth perform some level of encryption. The LE Controller performs the encryption function. LE generates 128-bit encrypted data from a 128-bit key and plaintext data using the AES-128-bit block cypher (<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>)
- **Signed Data** - This feature supports the ability to send authenticated data over an unencrypted transport between two devices with a trusted relationship (see pairing). In circumstances where the channel is not encrypted, the device could still ensure the data authentication. To sign the data, BLE utilizes CSRK. The sending device places a signature after the Data Protocol Data Unit (PDU). The receiving side verifies the signature and, if the signature is verified, the PDU is assumed to come from a trusted device. The signature is composed of a Message Authentication Code generated by the signing algorithm and a counter. The counter is used to protect against a replay attack and is incremented on each signed Data PDU sent.

How Bluetooth Utilizes these Features to Protect Your Information The goal of the low energy security mechanism is to protect communication between devices at different levels of the stack. Below are commons types of attacks against various wireless communication protocols, and how Bluetooth addresses them.

4.1.2 Health standard

While there are no standards for health that we are required to maintain, we would want to be cautious considering our project is an indicator for overall health and capability of the driver. So we must adhere to the given law of Blood Alcohol Content levels and ensure those reported values are within a given tolerance. It must also be known that those values are within that tolerance.

The BAC in Florida is 0.08 so in order to avoid errors in tolerance our implementation should aim for a much lower value. By taking a “Better Safe Than Sorry” approach we avoid any potential errors in regards to setting a tolerance. Our team has decided that X.XX is an appropriate value as this will ensure the doors do not unlock and allow the driver to do harm to property, himself, passengers, or pedestrians.

4.1.3 Design impact of relevant standards

The design impact of the Bluetooth standard affects us in some ways especially in regards to the security.

How Bluetooth Utilizes these Features to Protect Your Information The goal of the low energy security mechanism is to protect communication between devices at different levels of the stack. Below are common types of attacks against various wireless communication protocols, and how Bluetooth addresses them.

4.2 Realistic Design Constraints

These are constraints that we may encounter in the real world. We would need to evaluate these domains before proceeding into our design phase. It is important the project be an achievable idea, and it needs to be within the realistic constraints set by these standards.

4.2.1 Economic and Time constraints

Obviously, we are limited by our budget and the time constraints placed on us from the semester; however, we have also set completion milestones for ourselves. These would be soft deadlines as we may encounter trouble with the development. Our team has allocated slack time onto project milestones. Hard deadlines are unavoidable and must be addressed.

4.2.2 Environmental, Social, and Political constraints

The social and political value of our product is enormous. Socially, receiving a DUI has a tremendous impact on one's life. Many people have been arrested for DUI and with very clean records the ramifications of receiving a DUI can stick around for years. Most people are aware of the short term consequences of drinking and driving, which can include a driver license suspension, very high fees and fines, and high insurance premiums. However, there are also long term consequences associated such as, loss of job opportunities, losing government clearance for jobs, loss of scholarship, or even relationships. All of these can tailspin into a social nightmare.

That being said, while our product may be able save your social status long term and short term. There are implications of utilizing the device without the need; such questions may arise such as “why an individual would want to use this device?”, “does this individual have a drinking problem?”, etc. We would like to constrain the device to be unperceivable or at least latent.

4.2.3 Ethical, Health, and Safety constraints

As mentioned in our health standard we should be cautious with setting our limiting value (BAC) too close to the legal limit. This very easily could cause harms to health and safety. Our group also feels it has an ethical obligation to fulfil. If we can stop any potential accidents from occurring, then we will have done a good job.

4.2.4 Manufacturability and Sustainability constraints

Most of the products we are working on are already heavily manufactured and are not difficult to produce

5. Project Hardware and Software Design Details

5.1 Initial Design Architectures and Related Diagrams

To implement the BreathaLock system most effectively we have chosen components based on many factors listed previously. Below we have an image of all of the major components and breakout boards that will make up the bulk of the BreathaLock system. Of course in final implementation we will need many other small discrete components.

5.1.1 Project Hardware Components

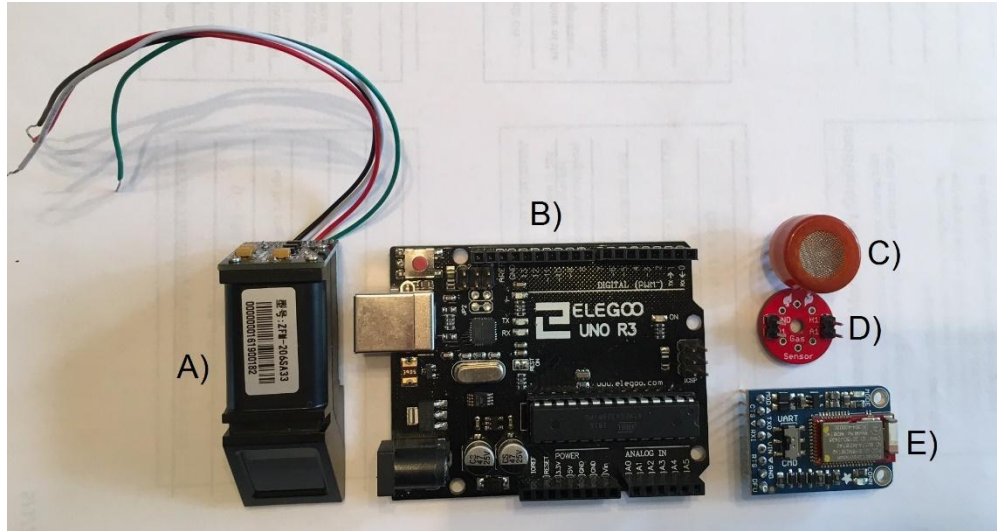


Figure 8: All components by letter

Part Letter	Component	Purpose
A)	Adafruit Fingerprint Sensor	Biometric fingerprint sensor subsystem
B)	Elegoo Uno R3 (Arduino 3 rd Party Board)	Microcontroller for testing
C)	MQ-3 Alcohol Sensor	Alcohol sensor component
D)	Sparkfun Alcohol Sensor Breakout Board	Breakout board to allow for bread boarding
E)	Adafruit Bluefruit UART Friend	Bluetooth component for Bluetooth subsystem

Table 18: Component descriptions

In our final implementation many of these components will be included in PCB which will allow us to create a more concise and dense device. In the above image we include the Elegoo Uno R3. This is because it allows us to interface with the microcontroller that we will be using in our final implementation. Note that we will be using the open source design of Arduino PCB for our project PCB. This will also be the case for the Sparkfun breakout board.

5.1.2 Hardware Wiring

While investigating what our overall hardware will look like we need to investigate how to connect all sensor to the Arduino to interface with the microcontroller effectively. Discussed late in the individual components section we see the wiring of each individual component to the Arduino. To keep our diagrams consistent, we will make most connection point the same to reduce interference with testing and wiring. This will also make it easier for us to debug any problems that we may have with the components later in the prototyping phase.

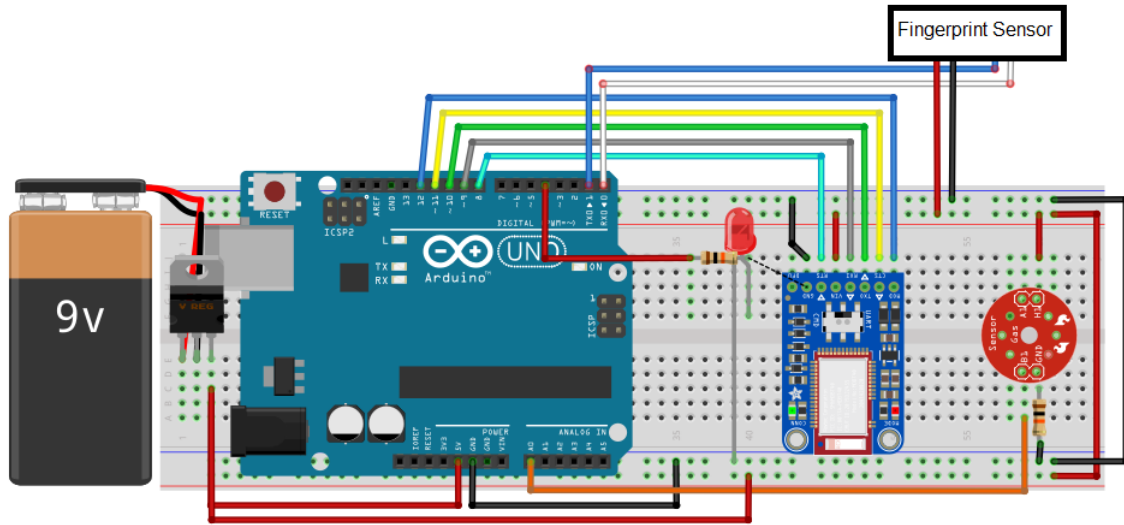


Figure 9: Fritzing breadboard test wiring diagram

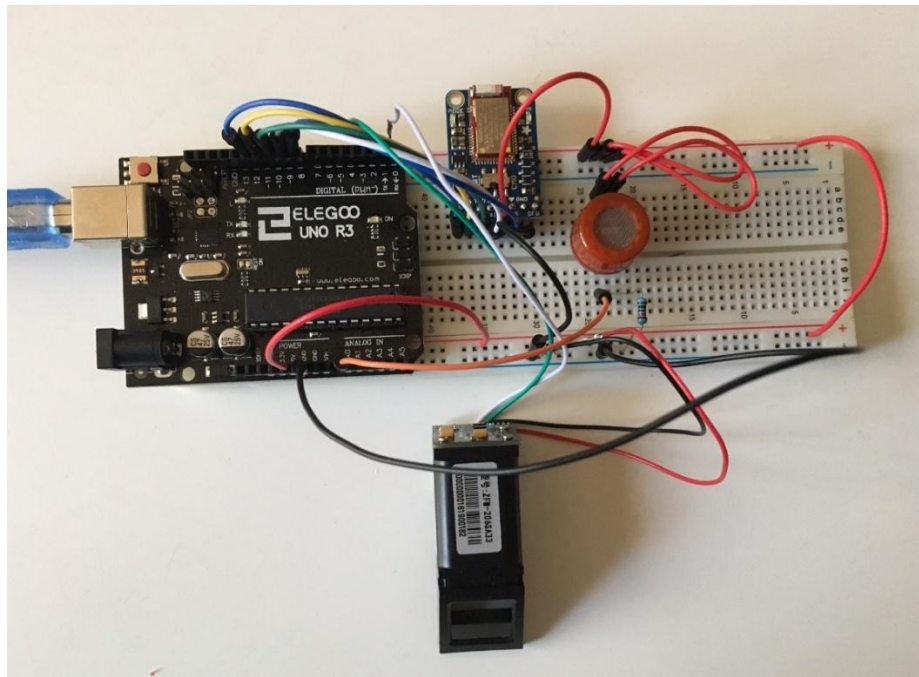


Figure 10: Breadboard testing of all components

Section	Arduino Input	Connection
Power	5V	Breadboard Power Rail
	GND	Breadboard Ground Rail
Analog	A0	Analog Output for MQ-3 Alcohol sensor
Digital	0	RX Fingerprint Sensor
	1	TX Fingerprint Sensor
	8	RTS Bluetooth Module
	9	RXI Bluetooth Module
	10	TXO Bluetooth Module
	11	CTS Bluetooth Module
	12	MOD Bluetooth Module

Table 19: Arduino Specific Connection

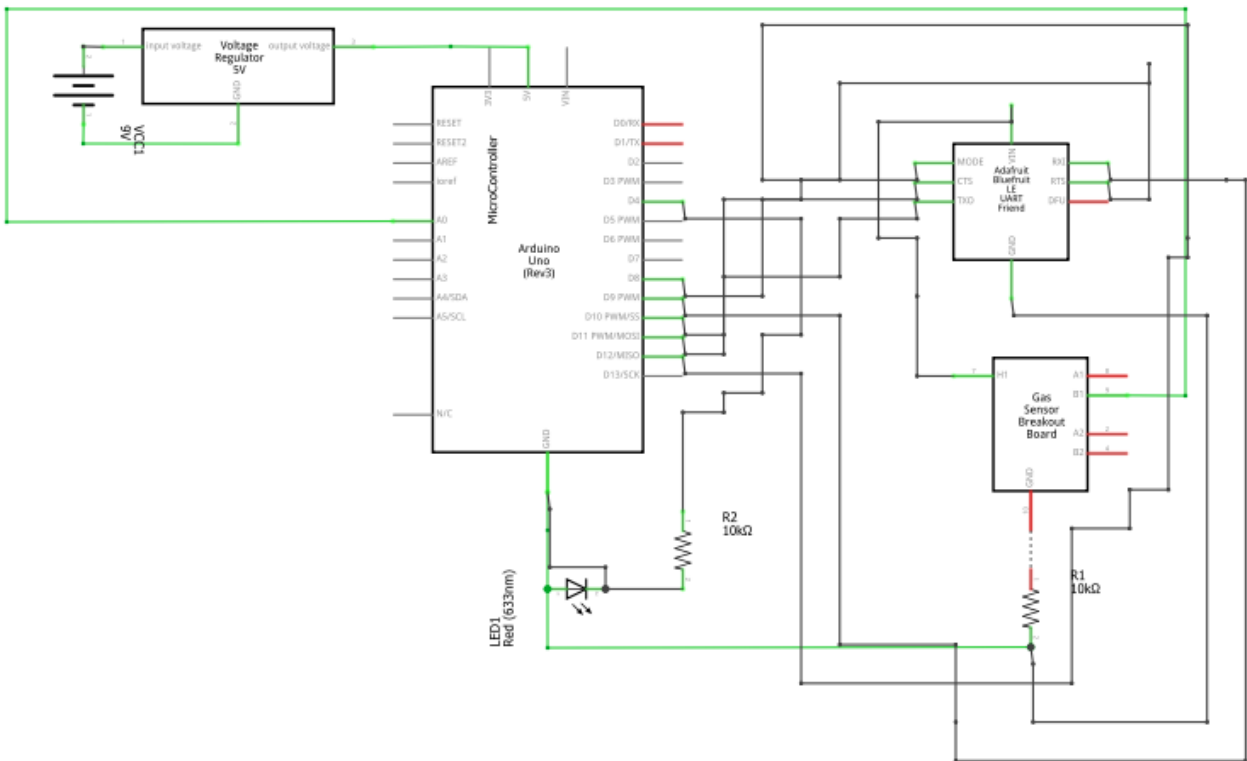


Figure 11: Fritzing wiring schematic

The above table and figure depict a clear representation of how we will interface our microcontroller with the various components and sensors. It is important to test and verify the amount of connection points that we will be using so that when we begin our PCB design we will be able correctly interface. In the above diagrams and schematics, we are making connections with female headers, breadboards, and raw wires. When we get into the PCB design we will be able to eliminate these connection types and use the PCB wire pours to make connections.

5.2 Bluetooth Subsystem

5.2.1. Bluefruit LE UART Friend (BLE)

For our Bluetooth component we choose to use a low cost Bluetooth module sold by Adafruit. We chose this chip for a few reasons, it is cost effective, low energy, and compact. The chip comes with EAGLE files and some instruction on how to test it which will allow us to easily include its functionality onto our own PCB and ensure functionality.



Figure 12: Bluefruit LE UART Friend (BLE) external image

Onboard Processing	ARM Cortex 16Mhz
Flash Memory	256KB
RAM	32KB SRAM
Baud Rate	UART at 9600 Baud
Supply Voltage	5v-safe input with onboard voltage regulation
Dimensions	21 x 32 x 5mm
Weight	3.4g

Table 20: Bluefruit LE UART Specifications

5.2.2. Sensor Specific Operation

The Adafruit Bluefruit UART Friends is a BT 4.0 and BT 4.1 stack which was specifically designed for low energy use. We intend to use Bluetooth simply as a gateway to communicate between the BreathaLock system and a cellular device. Specifically, in hardware implementation we will be including these components into our PCB design and will have the BreathaLock constantly attempting to connect to a cellular device. Once connected the BreathaLock will be controlled with both the buttons on the BreathaLock but also Bluetooth commands from the cellular device.

5.2.3 Subsystem Implementation

To include Bluetooth as an asset to the BreathaLock we will need to include it first into our hardware realization. This will be done by including the components and processing into our PCB. This should be done without much stress aside from the connections to the MCU to transfer data and to bring power to the sensor. Because the sensor operates comfortably with a 5v input we will be bringing power to the sensor from the same output of the 5v voltage regulator.

5.3 Biometric Subsystem

5.3.1 Sensor Overview

Adafruit Fingerprint Sensor

We chose a prepackaged fingerprint sensor simply because the focus of this design project is to implement a device to ensure ease of use and accuracy to prevent drunk driving, not to investigate image and digital signal processing of fingers. The Adafruit fingerprint sensor has an onboard DSP chip and onboard flash memory to allow for multiple finger prints to be stored.



Figure 13: Fingerprint sensor external image

Component	Fingerprint Sensor
Supply voltage	3.6-6.0VDC
Operating Current	120mA max
Peak Current	120mA
Imaging time	Less than 1 second
Interface	TTL Serial
Dimension	56 x 20 x 21.5mm
Weight	20 grams

Table 21: Fingerprint sensor technical characteristics

5.3.2 Sensor Specific Operation

By understanding how our fingerprint sensor works we will be able to better implement its full capabilities into the BreathaLock system. The fingerprint sensor operates in two settings: enrolling and searching. Because the system has onboard processing and memory it has the ability to read and save a fingerprint and register it into an onboard database of users. Once the fingerprint is read and saved it will remain in memory until deleted. Secondly the sensor can operate in the searching state. When the sensor receives a search command it will read the finger on the sensor and cross reference it to fingerprints already in the memory. The sensor will either confirm user with a confidence score or fail the finger all together. These two operations combined with some other logic can be very powerful.

5.3.3 Subsystem Implementation

To implement the fingerprint sensor successfully we will need to take some time investigating the best way to save and read fingerprints for the application of drunk driving prevention. The main operation is quite simple: read a fingerprint at the time the user wants to get into their car and if the user is the correct owner then move onto the BAC test. The main obstacle is regarding to when and how to enroll new users. We cannot allow the BreathaLock to enroll new users at any time at ease because it would leave a huge loop hole for users to enroll any user at the time they are drunk and need access to their car. The best option would be to have to connect the BreathaLock to a computer and enroll users as an administrative user. Another option would be possibly to create an administrative password that can be entered into the android application to allow for new user enrolling. Our final consideration would be if we want there to be multiple users enrolled at a time or only 1. Either option would still result in user enrolling to be performed at a time before trying to enter the car.

5.4 Breathalyzer Subsystem

5.4.1 MQ-3 Alcohol Gas Sensor

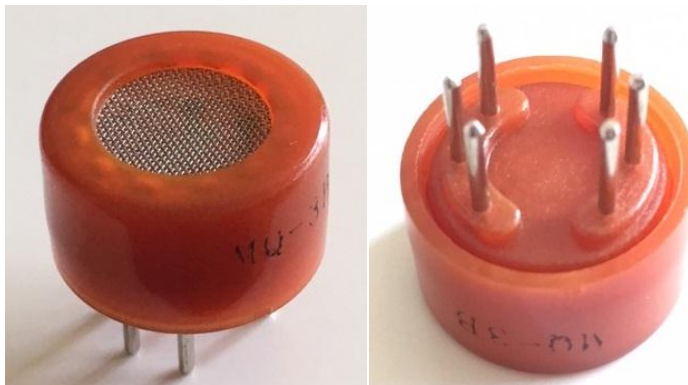


Figure 14: MQ-3 sensor external image and pin reference

Component	MQ-3
Sensor Type	Semiconductor
Target Gas Type	Alcohol
Detection Range	25-500ppm alcohol
Heater Voltage	5.0V
Output Voltage	2.5-4.0V

Table 22: MQ-3 sensor technical characteristics

We chose to use the MQ-3 alcohol sensor because of its simplicity, sensitivity, and fast response time. This sensor provides an analog output that can be read and analyzed by a microcontroller to decipher the sobriety of the user.

5.4.2 Sensor Specific Operation

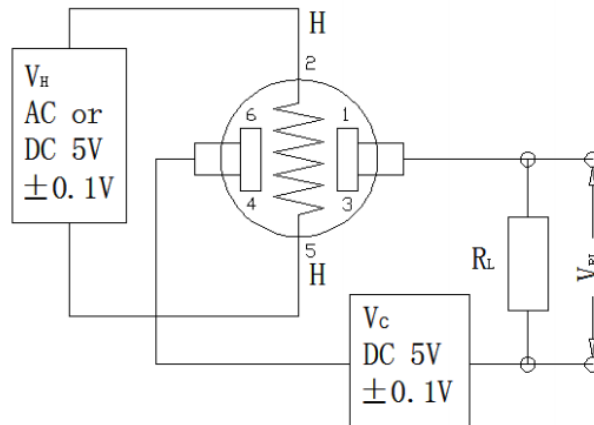


Figure 15: Sensor operation schematic

To implement a breathalyzer using the MQ-3 sensor it is important to fully understand how the sensor operates. This sensor works by using a conductivity sensitive material, particularly S_nO_2 or tin dioxide. The tin dioxide changes conductivity with the presence of alcohol in the air. When higher concentrations of alcohol exist in the air the conductivity of the air gets higher and so does this sensitive material. By placing the tin dioxide in series with a load resistor, the sensor is able to have an analog voltage read across this load resistor that is dependent on the conductivity of the tin dioxide. After calibrating what read values relate to blood alcohol levels we will have an accurate way to measure blood alcohol concentration (BAC).

5.4.3 Subsystem implementation

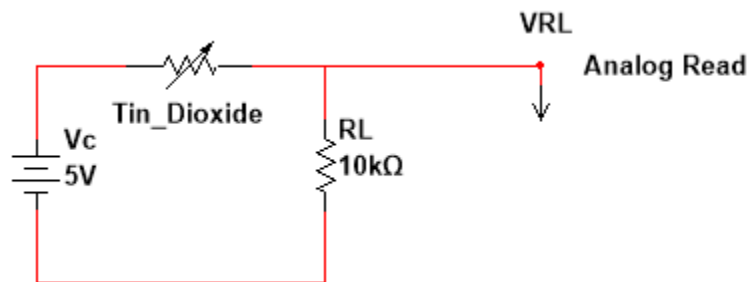


Figure 16: Circuit analysis of sensor

$$V_{RL} = \frac{V_C * R_L}{R_L + R_{Tin\ Dioxide}}$$

Equation 1: Output Voltage Analog read

To properly implement the breathalyzer subsystem, it is necessary to allocate some time and effort into layout, placement and design to ensure we are reading accurate data. First we need to make sure that we effectively power the sensor and choose the correct resistor value to receive expected analog output and make sense of it. To power the chip, we will

need to bring 5 volts into pins 2, 4 and 6 in the figure x. This will be taken from the output of a voltage regulator in the PCB design. Secondly we need to include a load resistor of whichever value we choose to use. It is not important because it will only affect the analog output which will need to be converted into a relative BAC but for the purpose of early design we will choose $R_L = 10k\Omega$. The V_{RL} pin from 1&3 to ground will be connected to an analog read pin accessed by the microcontroller.

To read and make sense of this sensor we will need read an analog output that is relative to the voltage across the load resistor. Unfortunately, because the conductivity of the tin dioxide layer is unknown, due to variability among components and the dependence on temperature, it would be impossible to calculate a perfect output based on the equation above. With that said we will be able to see that our output will be dependent on the resistor value chosen and that the value will be inversely proportional to the amount of alcohol in the air. Our code should break this range of values into two cases: more than 0.08 and less than 0.08 BAC. Depending on which case the sensor reads will determine if the user is over the legal limit and must be denied access.

5.5 Software Design

Software is an important component to an embedded system. It is not only about the hardware required but it is also heavily about the software decided. Choosing the right software for different stages of the development is not easy. There are various phases in a project that determine the software naturally. However, with our existing framework we were presented with several options for that incorporate various design methods. If we do not correctly choose the right software approach progress will be stalled, or the team will have to re-evaluate our projects requirements.

5.5.1 Programming Languages

One of the most important parts of a software solution is choosing the programming language. Each language has its own unique characteristics that may come with their own pros and cons based on the goals of the project. Choosing a language takes knowledge of your requirements and goals. Through this section we will cover various languages that we have considered for usage in the final deliverable.

5.5.1.1 Assembly

Assembly language (ASM) is a low-level programming language (bare-metal) that used for embedded devices, or other devices that can be programmable. It is comparable to computer architecture machine code instruction sets. Different assembly languages correspond to various computer architecture; for example, the ARM processors would use the ARM assembly language which utilizes sixteen user registers. They are all 32-bits wide. Only two are dedicate; the others are general purpose and are used to store operands, results and pointers to memory. Of the two dedicated registers, only one of these is permanently used for a special purpose (it is the PC). Although this is useful to know for ARM these instruction sets vary widely by architecture. It is not uncommon to encounter assembly especially in devices that require minimal code footprint (processing and memory).

Advantages

- Performant
- Minimal (memory wise)

Disadvantages

- Lacks readability
- Machine dependent
- Long code for simple programs

Conclusion

We decided that we will not use assembly for our project. The platform we are utilizing does not require we use something so low level! We also fear that the code would become quite unmanageable.

5.5.1.2 Python

Python might be at its strongest when used as a communication middleman between the user and the embedded system they're working with. Sending messages through Python to or from an embedded system allows the user to automate testing. Python scripts can put the system into different states, set configurations, and test all sorts of real-world use cases. Python can also be used to receive embedded system data that can be stored for analysis. Programmers can then use Python to develop parameters and other methods of analyzing that data and may further be used as a tool to assist developers.

Currently the main debate about the merits of Python comes down to what's more important to your team: development speed or runtime speed.

In regards to the embedded libraries support for the embedded world is quite limited python may still have some applicability in the world of scripting tools for development purposes.

Advantages

- Readability
- Platform independent
- Multiparadigm and supports OO, procedural, and functional programming styles

Disadvantages

- Slow
- Limited embedded support
- Global Interpreter Lock (only one thread may access python data)

Conclusion

Python does not seem like a good fit for the final deliverable solution. However, there may be room for helper scripts. Helper scripts may include test case scripts, installation, building, compiling and various developer task.

5.5.1.3 Java (Android API)

While most Android applications are written in Java language, there is some fundamental differences between the Java API and the Android API. Android does not run java bytecode by the traditional JVM but by ART (android runtime). This code compiles the code that ART runs to ELF (Executable and Linkable Format) executables which contain the machine code. Java bytecode in JAR-files is not executed by the Android operating system. Instead Java classes are compiled into bytecode which are executed on top of the ART framework (see below)

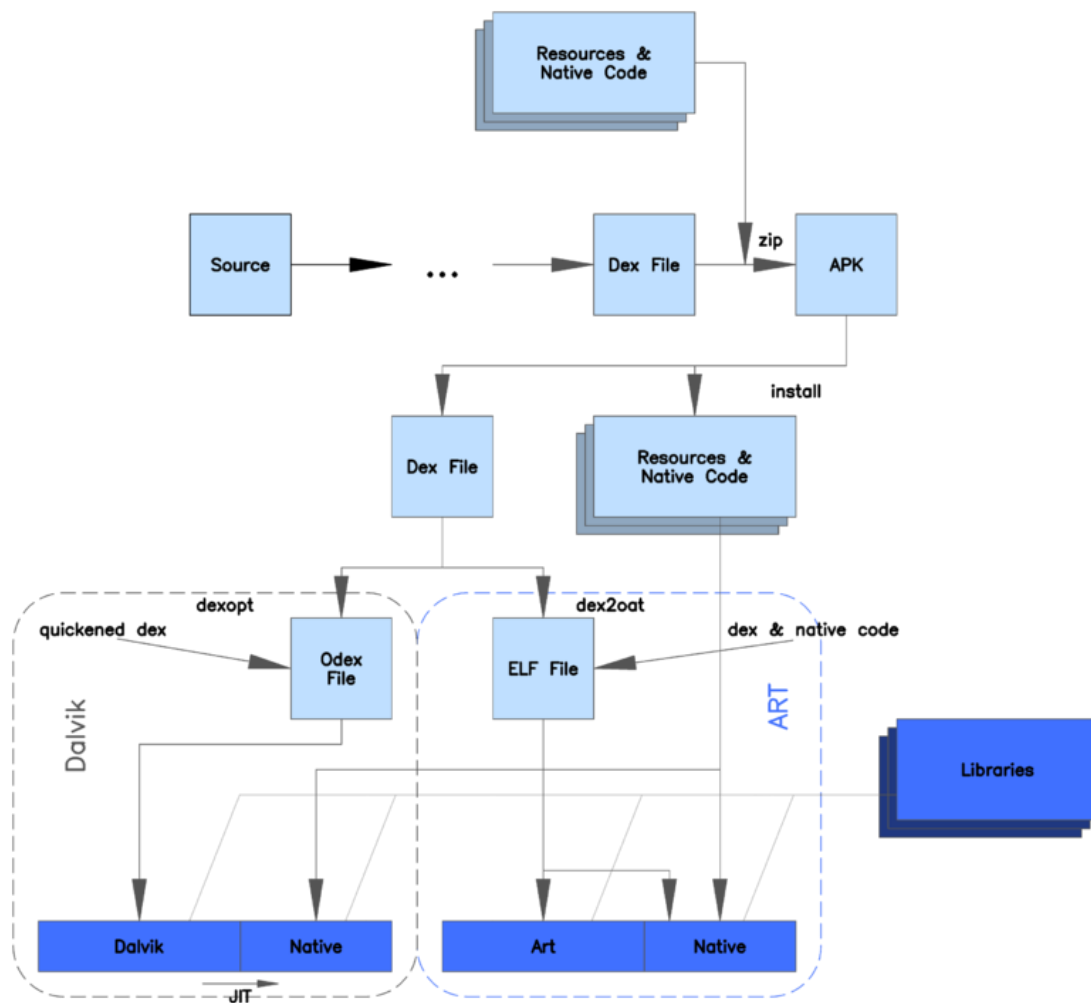


Figure 17: Java ART framework

Dalvik has some specific characteristics that differentiate it from other standard VMs:

- The VM was designed to use less memory.
- The constant pool has been modified to use only 32-bit indexes to simplify the interpreter.

- Standard Java bytecode executes 8-bit stack instructions. Local variables must be copied to or from the operand stack by separate instructions. Dalvik instead uses its own 16-bit instruction set that works directly on local variables. The local variable is commonly picked by a 4-bit "virtual register" field.

Advantages

- Syntax is easy
- Readability
- Platform independent
- Garbage Collection via Memory Manager
- Only officially supported language for Android
- Very good documentation

Disadvantages

- Large memory footprint
- Slow (often slower by a factor of 20-50x compared to other languages)
- Consistently burdened with security issues in the JVM

Conclusion

Java will be utilized for our project as it is the go-to solution for any android application engineers wish to build. There are a few stacks we will utilize for this project with regards to the Java toolchain:

- Bluetooth
- Android API

5.5.1.4 C

C is a general-purpose, imperative (changes a program's state) computer programming language. By design, C provides constructs that map efficiently to typical machine instructions, and it is the reason it is so heavily used right now. Developed by Dennis Ritchie an engineer at Bell Labs during 1970's, and used to re-implement the Unix operating system. Then embedded C is a subset of the C language, and the embedded C world requires a new set of libraries that vary across architecture similar to assembly. However, the overlying syntax of the language remains and maintained the same and introduces some more higher-level concepts as well. Such as, the main() function, conditional statement, loops, strings, arrays, bit operations, etc. these operations remain unspecific to the architecture. C is one of the most widely accepted and follows several standards (see standards) such as C has been standardized by the American National Standards Institute (see ANSI C) and subsequently by the International Organization for Standardization (ISO). This ensures the language is portable.

C also has a wide variety of tools that can be used to build software around it. There are many different support tools; such as, compilers and cross-compilers, IDE's and hardware.

Advantages

- Use standard C syntax
- Higher-level than assembly but close enough to hardware languages to be efficient
- Portable
- No VM (like Java)

Disadvantages

- No garbage collection (possibly a positive if done correctly)
- Steep learning curve (similar to assembly)

Conclusion

For this project C will be use less frequently in the traditional sense. As we will be primarily using C++ for the Arduino board (see next section x.x.x.5).

5.5.1.5 C++ (avr-g++ toolchain)

C++ in general is just “C with Classes” in the traditional sense following a object-oriented paradigm. Created by Bjarne Stroustrup during his Ph.D thesis, Bjarne set out to create a language that was C and supported features of objects, classes, inheritance and subclasses.

The Arduino platform fortunately utilizes C++ with some domain specific libraries, that is built with the avr-g++ toolchain. These add on various features such as functions that allow you to map to specific features on the board. Without these functions the layers of abstraction would need to be manually written with special registers.

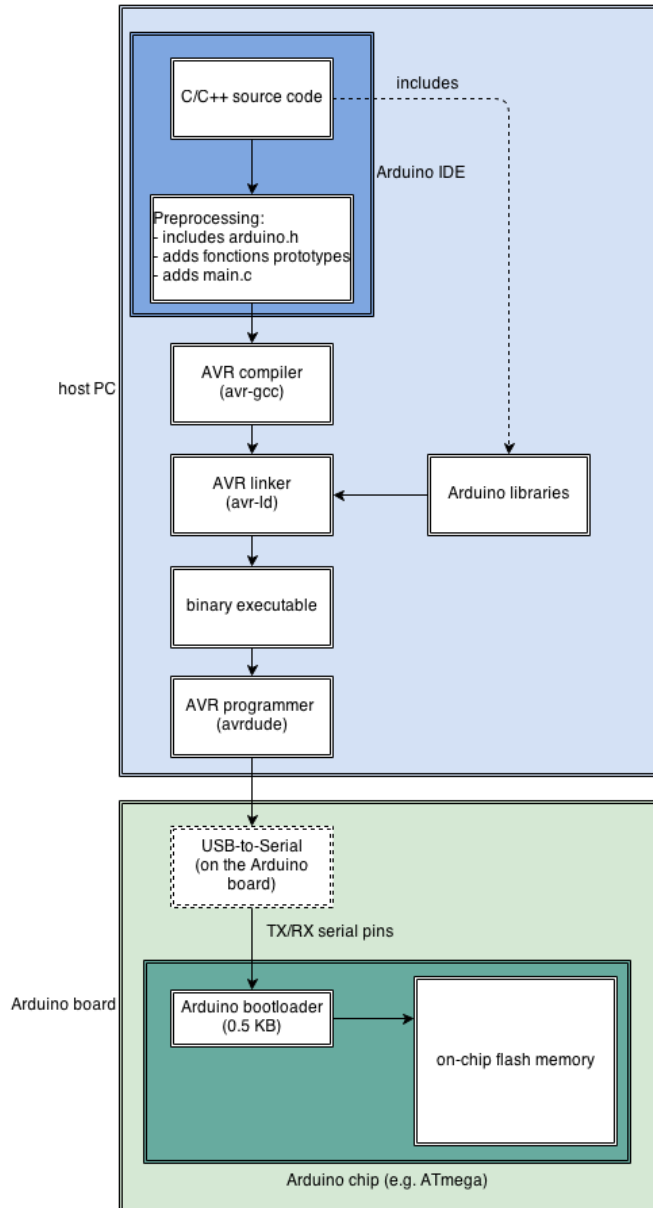


Figure 18: C++ Flow Chart

Advantages

- Object Oriented
- Readability
- All the positives of C
- Garbage Collection (contributes to bigger footprint)

Disadvantages

- Bigger footprint than C
- Doesn't provide strong type-checking. The codes are prone to errors.

Conclusion

Since out of the box Arduino supports the libraries that are based on C++ this will be our primary workhorse for this project's logic. We will be heavily utilizing the Arduino avr-g++ compiler. We also feel this is a good fit as C++ and Arduino are both documented quite well.

5.5.2 Integrated Development Environments

An Integrated Development Environment (IDE) are designed to assist the developer by increasing productivity, efficiency and accuracy. An IDE provide many components often presented in the form of graphical user interface (GUI) where all development can be accomplished. These environments often provide features such as, modifying, compiling, deploying, debugging, code completion, code folding and much more. It was integral our team chose a development environment that suited our needs so that we could become an effective team of engineers and increase our productivity. Contrasting this by using standard text editing software and using a compilers mentioned above with long winded build systems that have many command line parameters to get the desired results.

However, deciding on an IDE is not an easy task as there is a lot of noise on the market for various ones; however, we managed to dwindle this down to two tools that will help support us in our final deliverable.

5.5.2.1 Eclipse

Eclipse is one of the most versatile IDE's that currently exist; albeit, old the IDE is still one of the most used IDE's for a wide variety of supported languages and fortunately for us all of the languages mentioned in section *2.2.1 Programming Languages* are supported.

Eclipse has some very well thought out refactoring capabilities that work well, and great documentation capabilities. The IDE has several features that makes it attractive to us such as code completion, templating, integration with version control and build systems. Its code formatting and cleanup tools are very well done. We also found that its build system works well for our needs.

Eclipse has a few add-ons that we should be able to utilize for this project. They are the Android ADT Plugin and the AVR-Eclipse plugin with both of these we will be able to develop for both Java Android and Arduino.

AVR-Eclipse Plugin

The AVR-Eclipse plugin includes CDT which provides a fully functional C and C++ integrated development environment built on to Eclipse platform. The CDT plugin has many of the same features that eclipse does; such as, project creation, managed build for various toolchains, standard MAKE build, source navigation, call graphs, browser, and macro definitions, code folding and hyperlink navigation, visual debugging tools like memory, registers, and disassembly viewers.

The AVR plugin itself is a cross platform code builder. It nearly platform independent and supports our platform so this is not a flaw. The AVR plugin includes the required toolchains, debuggers, and frameworks that work on most popular platforms. Below is an image of the Eclipse IDE utilizing the AVR plugin with code involved.

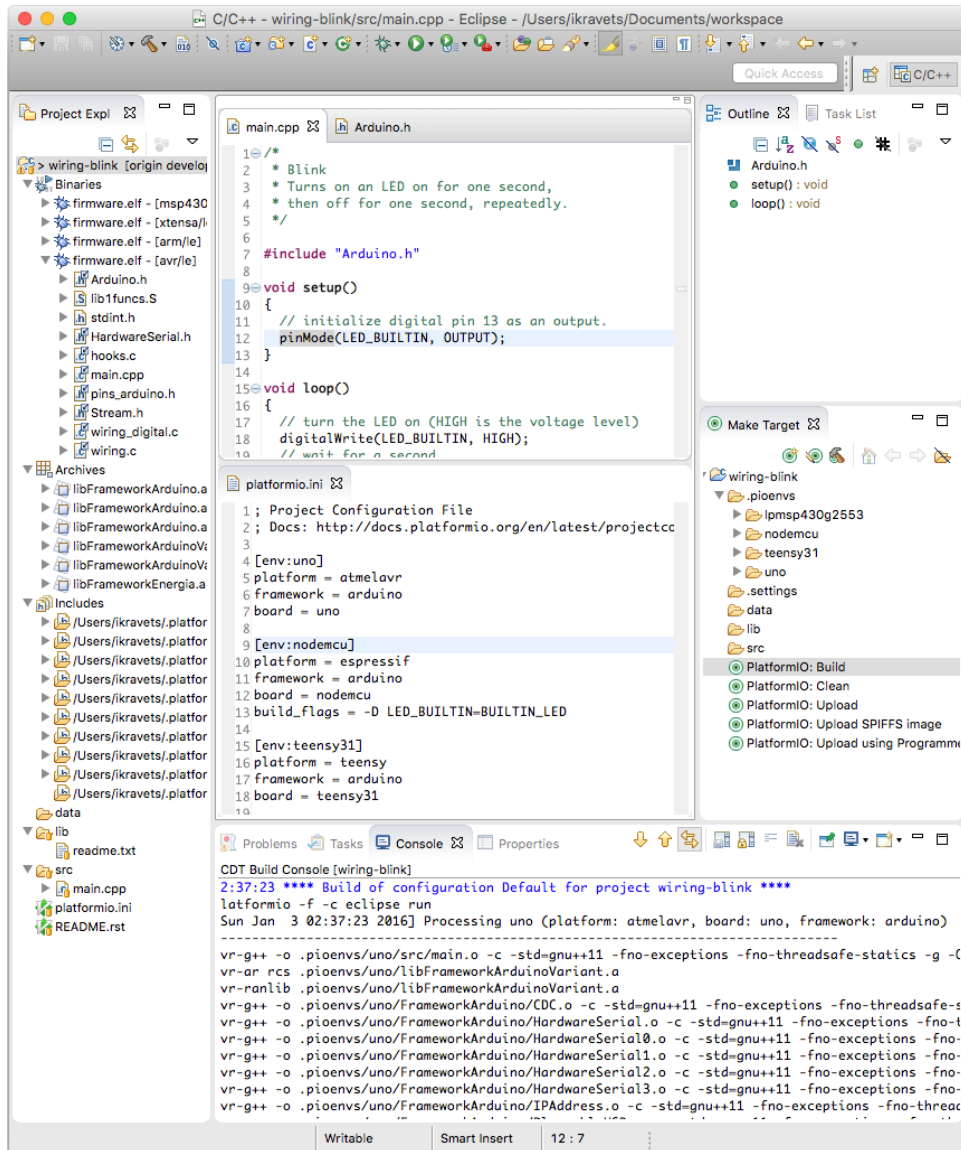


Figure 19: Eclipse IDE

ADT Android Plugin

Android Studio is the official Integrated Development Environment (IDE) for android app development, is for the eclipse IDE and provides. It was created to give developers a one stop shop for a development environment in which to build Android applications. It extends the current capabilities (listed above) and allows developers to build android projects with a user interface. Developers can also add libraries from the ADT toolchain. It extends the capabilities of Eclipse to let you quickly set up new Android projects, build

an app UI, debug your app, and export signed (or unsigned) app packages (APKs) for distribution.

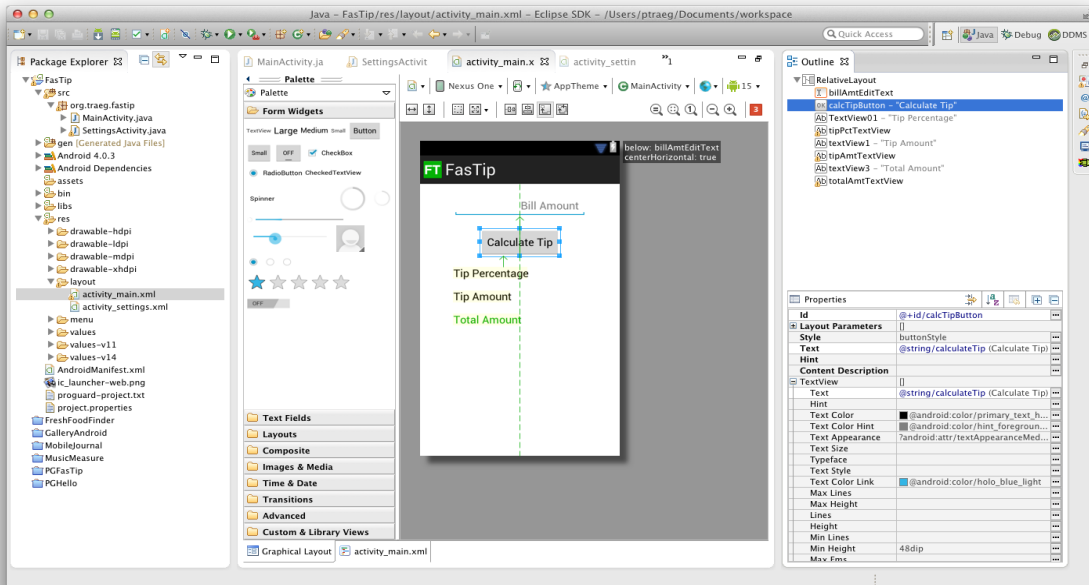


Figure 20: Eclipse ADT Plugin

Advantages

- Cross Compatible
- Multiple Languages Supported
- Many Modern IDE Features

Disadvantages

- Documentation is lacking
- Preferences overload
- No out-of-the-box configuration
- Outdated
- Unsupported

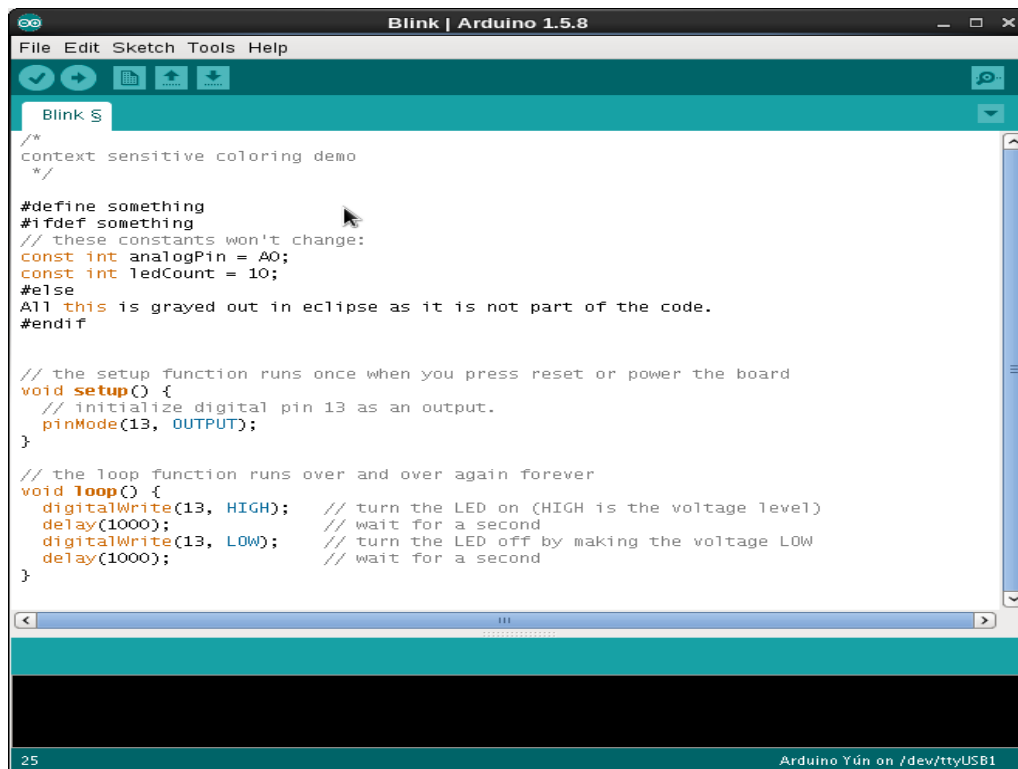
Conclusion

The Eclipse IDE overall is great, however, it has been outdated for a while now especially in regards to Android Development. Android Development has been usurped by Android Studio created by Google built on top of IntelliJ IDE (discussed later)

5.5.2.2 Arduino IDE

The Arduino IDE contains a text editor for writing the codes, a text console, toolbar with common methods and view details. It connects to the Arduino hardware in order to upload and flash programs onto the board. The IDE can also simultaneously debug and communicate with the platform.

Programs written in this IDE are called sketches. The developer composes these sketches in the text editor and are saved as with the file extension.ino. The editor has very basic features such as, cutting/pasting and for searching and replacing text. The Message are connected with the Arduino platform and gives feedback regarding errors and warnings. These error messages are also displayed in the console windows. This all gives useful feedback for development.



```
Arduino IDE - Blink | Arduino 1.5.8
File Edit Sketch Tools Help
Blink S
/*
context sensitive coloring demo
*/

#define something
#ifdef something
// these constants won't change:
const int analogPin = A0;
const int ledCount = 10;
#else
All this is grayed out in eclipse as it is not part of the code.
#endif

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}

25 Arduino Yun on /dev/ttyUSB1
```

Figure 21: An opened sketch

Advantages

- Default Development Tool
- Simple to use

Disadvantages

- Limited to single sketch
- No project viewer

Conclusion

For most simple projects we will be probably utilize this it is a very light and easy to use IDE. This would be ideal for debugging and testing small programs.

5.5.2.3 Android Studio

Android Studio is currently the official Integrated Development Environment (IDE) for developing Android applications, the platform is built on IntelliJ IDEA. The IDE has some very powerful features. The interface is one of the cleanest and most user friendly IDE we have seen. The Android Studio IDE run very quickly and offers a responsive interface. The IDE also offers a variety of analytical tools that help the developer with analyzing code before delivery. The Android API, is included out of the box with the IDE so there is little to no configuration in setup and installation. Android Studio also highlights potential bugs you may experience in your code at runtime or compile-time. This streamlines the development process.

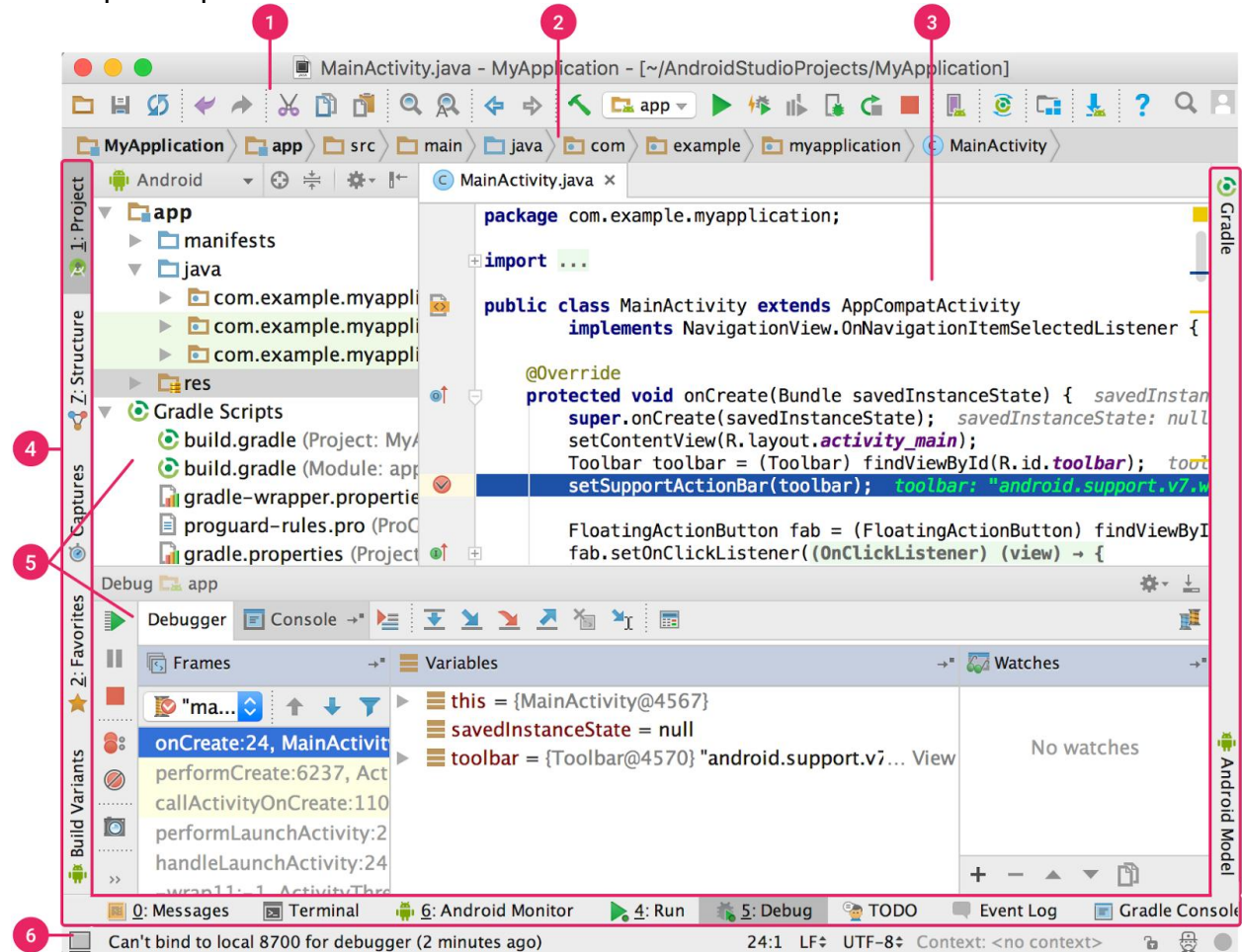


Figure 22: Android Studio IDE

1. Toolbar
2. Navigation toolbar browse hierarchically
3. Main editor window
4. View toolbar can modify the user experience
5. Navigate various tools such as debugger or console
6. Status bar giving valuable information regarding startup

The Android studio has a very flexible means of prototyping with an emulator called Android Virtual Device (AVD) this virtual device allows you to run your android app on a variety of different android platforms such as android TV to android phones

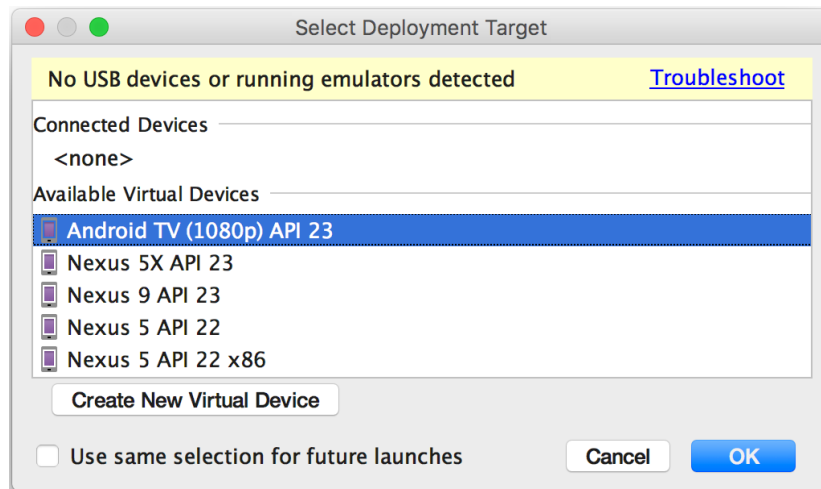


Figure 23: Selection screen for Android devices



Figure 24: AVD emulating an android device

The Android Virtual Device simulates a device and displays it on your development environment. This allows the developer to quickly prototype, develop, and test Android apps. This is done without the using a physical device. As mentioned above AVD supports Android phones, wear, Android TV, and tablets. It comes with all the device metadata required to begin rapidly prototyping. However, while this feature is useful it does suffer from performance issues and is not comparable to the physical device performance wise. Android Virtual Device also has a very slow startup time, this can be mitigated by having the emulator running in the background and pushing your development apps to the existing/running emulator.

Advantages

- Included Build System
- Feature Rich Emulator
- Unified environment where you can develop for all devices
- Develop and Prototype without the hardware device
- Out-Of-Box setup no configuration

Disadvantages

- Emulator is very slow

Conclusion

Android Studio seems to fit our needs very well for our Android development purposes.

5.5.3 Functional Requirements

5.5.3.1 Main Functionality

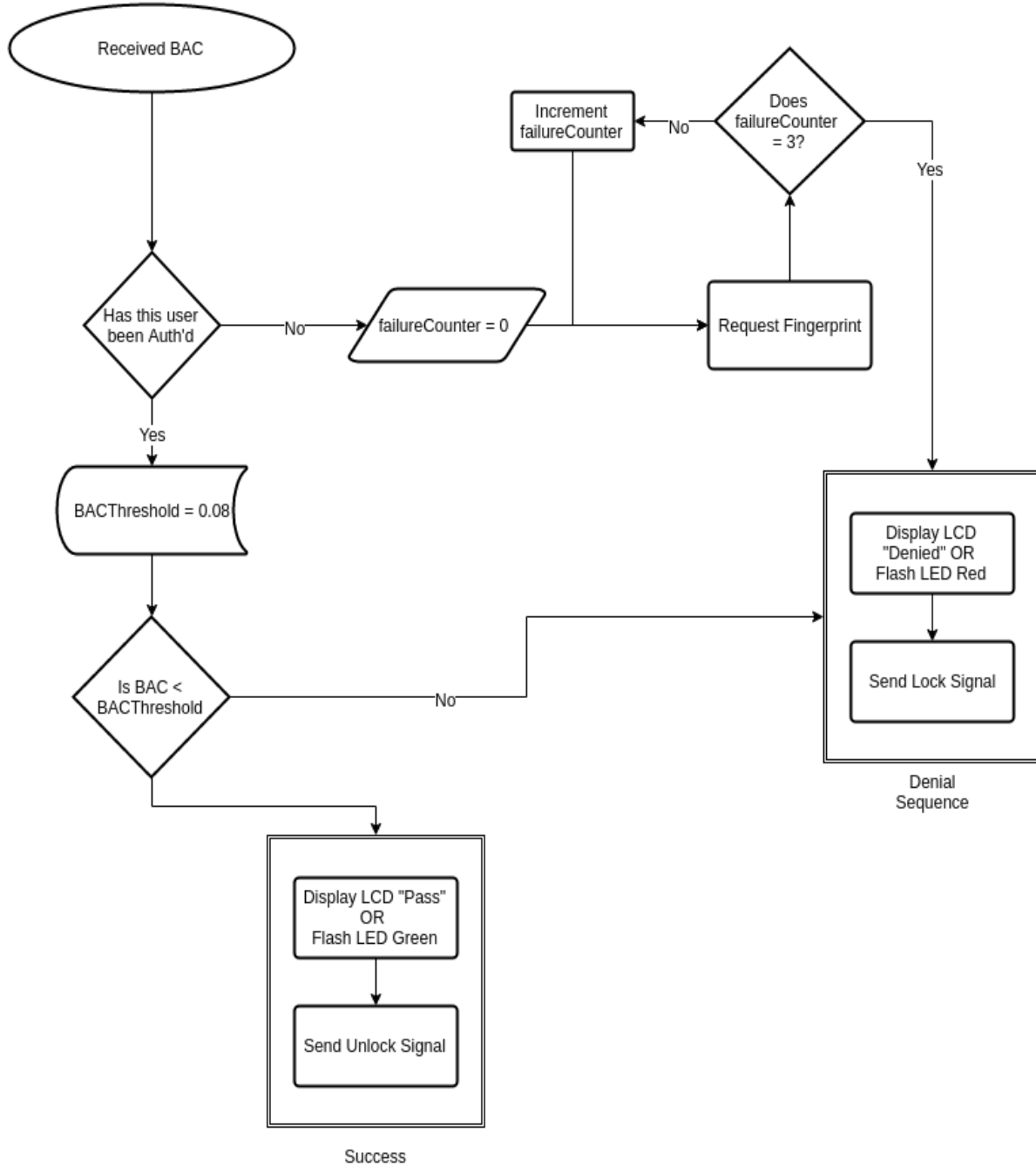


Figure 25: Handheld Device Functionality

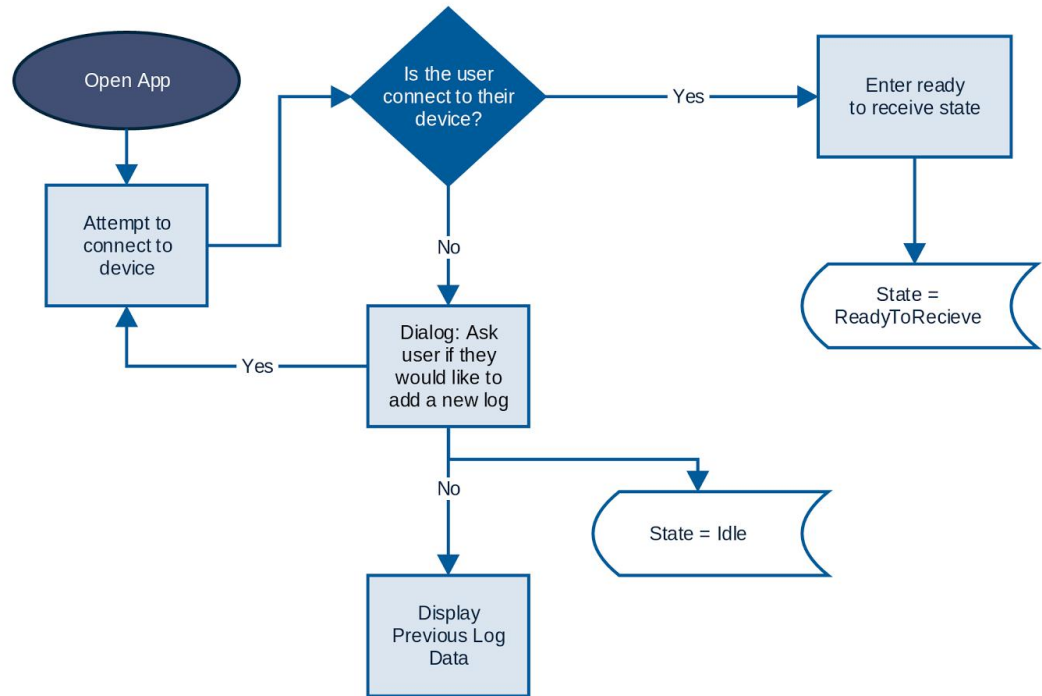


Figure 26: Android Device Functionality

5.5.3.2 Technical Functionality

No: 1
Statement: The user should be able to connect to the Breathalyzer with their phone
Source: Team
Dependency: Two devices (breathalyzer and android)
Conflicts: None
Supporting Materials:
Evaluation Method: 3.2.4. Description of Individual Test Cases
Revision History: C.Taylor (initial requirement)

Table 23: Technical Functionality 1

No: 2
Statement: The handheld Breathalyzer device should be able to toggle car locks
Source: Team
Dependency: Two devices (breathalyzer and android)
Conflicts: None
Supporting Materials:
Evaluation Method: 3.2.4. Description of Individual Test Cases
Revision History: C.Taylor (initial requirement)

Table 24: Technical Functionality 2

No: 3
Statement: Android app will collect data from device and log it for personal or liability use
Source: Team
Dependency: Two devices (breathalyzer and android)
Conflicts: None
Supporting Materials:
Evaluation Method: 3.2.4. Description of Individual Test Cases
Revision History: C.Taylor (initial requirement)

Table 25: Technical Functionality 3

No: 4
Statement: The breathalyzer register as pass based on +/- tolerance
Source: Team
Dependency: Two devices (breathalyzer and android)
Conflicts: None
Supporting Materials:
Evaluation Method: 3.2.4. Description of Individual Test Cases
Revision History: C.Taylor (initial requirement)

Table 26: Technical Functionality 4

No: 5
Statement: Placing the fingerprint give the user access to the device
Source: Team
Dependency: Two devices (breathalyzer and android)
Conflicts: None
Supporting Materials:
Evaluation Method: 3.2.4. Description of Individual Test Cases
Revision History: C.Taylor (initial requirement)

Table 27: Technical Functionality 5

5.5.3.3 Software Requirements

During running of the Android application

- The user is prompted for more
- Data should be stored on the android device
- If the android application fails to connect to the BrethaLock device it should

During the running of the BrethaLock device

- The user must be able to toggle the car door locks
- Should be independent of the Android device in that it does not require an android device in order to work it is merely supplementary

- A location must be provided and the location must either be a 5 numerical digit zip code or in city, state/province, country form.
- If there are data integrity issues we should prompt the user to retry

5.5.3.4 Interface Requirements

- The Android user interface should be simple to use
- The list of data should be displayed in a ListView (Figure XX)
- The queries asked of the users will be presented with Dialog boxes or module windows (Figure XX)
- In order to maintain aesthetic appeal we should utilize Google's Material Design standards.

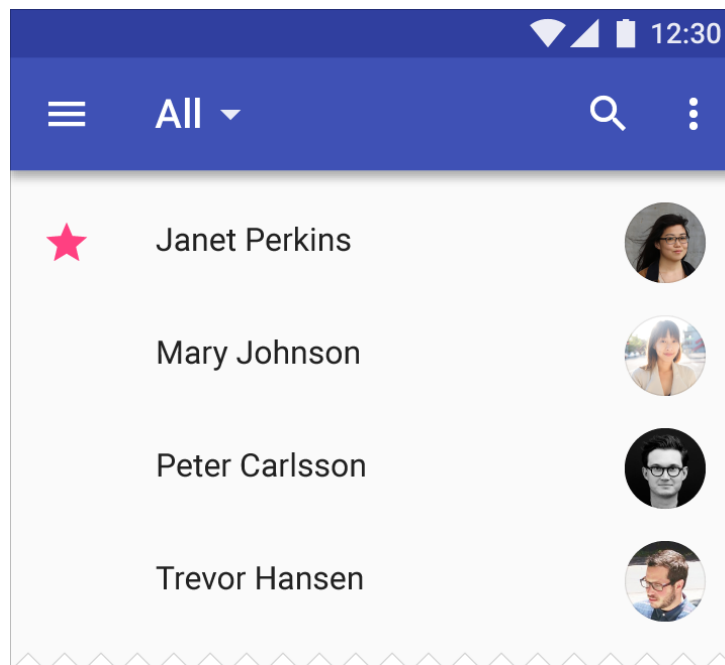


Figure 27: List View

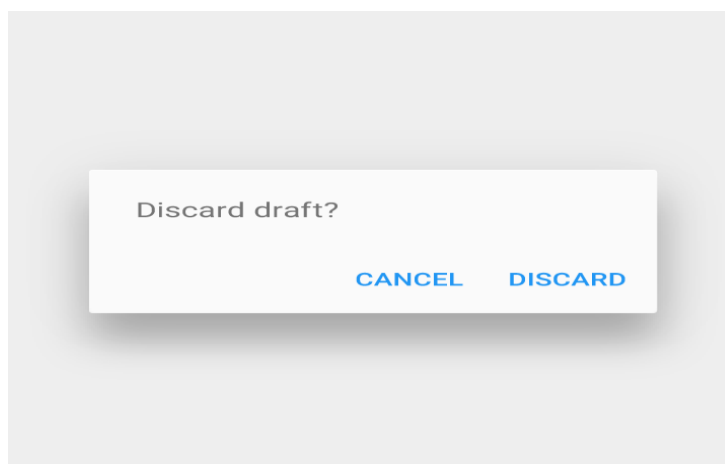


Figure 28: Dialog window example

5.6 Summary of Design

The overall design summary includes the use of all sensors collaboratively to create the decision of whether or not the correct user is sober enough to drive a vehicle in the legal limit. To do so our microcontroller will first be taking readings from our fingerprint subsystem and alcohol sensor subsystem. Once these two input are taken, using our own code, we will determine what we will output to the user through the use of the android device and also allow the ability to unlock the vehicle.

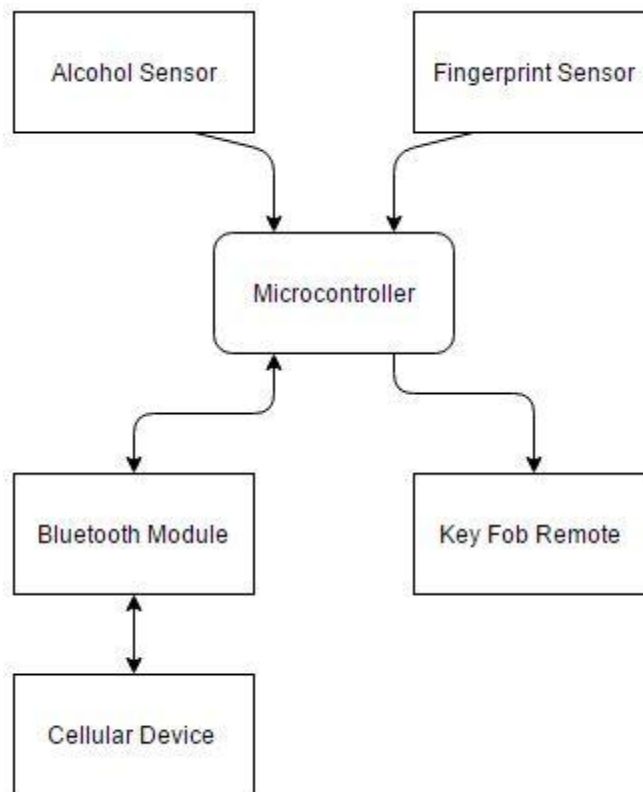


Figure 29: Flowchart of system hardware implementation

Overall implementation will follow the above designs and components connected as shown. We intend to prototype and design this device seamlessly with little to no complications.

6. Project Prototype Construction and Coding

6.1 Integrated Schematics

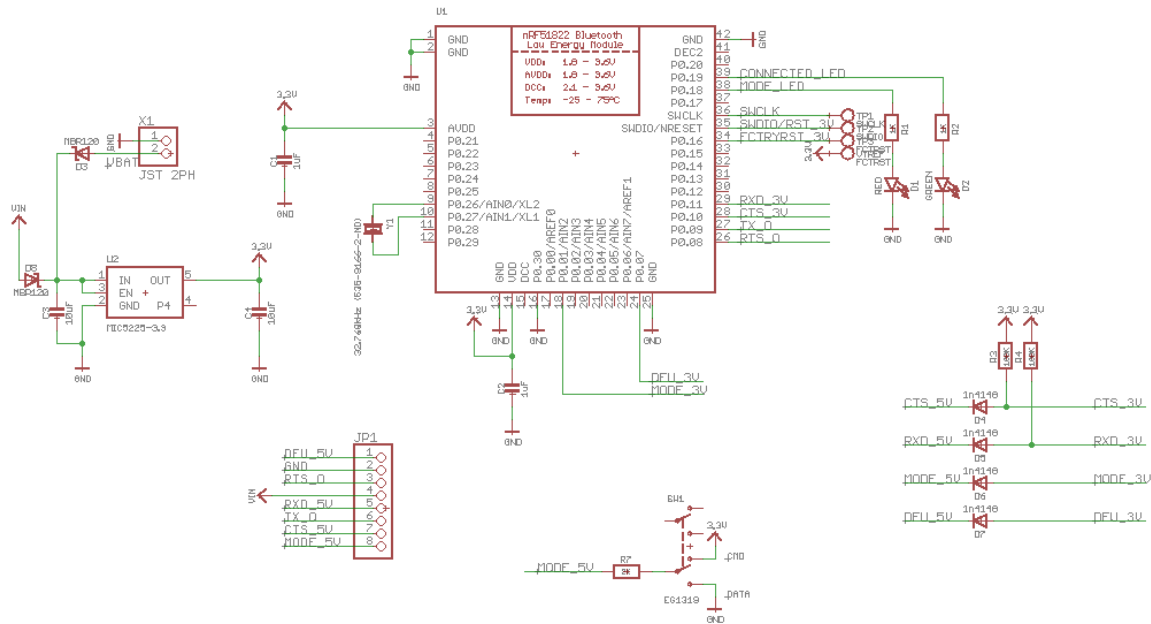


Figure 30: Eagle Schematic of Bluetooth Module

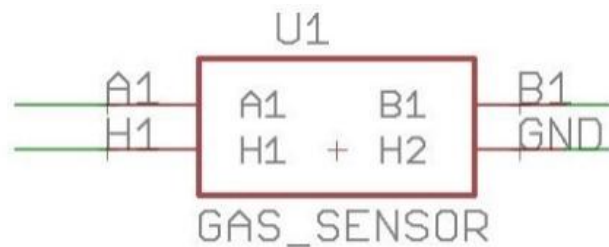


Figure 31: Eagle Schematic of alcohol sensor breakout board

6.2 Parts Acquisition

During the prototyping process it is necessary for us to have components to test and verify our design is functional. Initially we will be working with mostly connection of already working chips and manually connecting together to work as a system. Once our initial breadboard tests are complete we will order our PCB and begin verifying that our PCB works the same way by populating the board with all of the various components.

6.2.1 Adafruit

For many of our larger chips and components Adafruit will be very useful as they offer components at a good price with full support including schematics and many time tutorials and test designs. Adafruit is the supplier that we chose to order our fingerprint sensor and also our Bluetooth chip from. We also value Adafruit because their name is very large and therefore they many times have libraries, header files, and even component diagrams

for many programs that we will be using during our design. Aside from electrical components Adafruit also offers tools to populate our boards and test including soldering equipment, wires, wire strippers etc.

6.2.2 Sparkfun

Sparkfun is also another great option for larger components and build support. Sparkfun is a very large seller that has everything from components to books and even a blog section. The alcohol sensor that we chose to use is only sold by Sparkfun. They also include test support, a break-out board add on, and some comments on other user's experience. Another great advantage to using Sparkfun is their inclusion in Fritzing software. Fritzing has an entire library on components sold by Sparkfun that is very useful for schematic creation.

6.2.3 Digi-Key

Digi-Key is the fourth largest electronic component distributor in North America and a very organized marketplace for PCB and breadboard testing shopping. Digi-Key will most likely be our largest supplier of surface mount and through-hole components to populate our PCB and also do breadboard testing. Many other component companies that sell surface mount components sell only very large quantities of particular components whereas Digi-Key will allow us to purchase relatively low quantities for our PCB population.

6.2.4 UCF

Our final resource for parts is our very own school. UCF has many labs on campus and also has the TI innovation lab that carries many components. In addition to components we will need to gather as we design the BreathaLock schematic there are many components that we have already acquired throughout our courses with lab sections. For example, we already have an LM7805 voltage regulator, some resistors, capacitors, and various IC components that may prove to be useful in the breadboard prototyping stage.

6.3 PCB Design

To create the densest and reliable device we will need to design and have a printed circuit board made. Not only is this a requirement of this project but it displays full competence and understanding of electrical components working together in a system. Designing a printed circuit board is very time consuming and requires a lot of attention to detail. One incorrect wire trace can leave a circuit open and leave part of your board without use, on the contrary a short can burn and destroy components. To successfully design a functional PCB we will use the help of a PCB board software that will give us the best experience and best PCB design.

6.3.1 EAGLE

Eagle PCB design is a commercial software used for schematic creation and board layout used by many PCB designers. It allows for an extensive list of features to assist in prototyping and production level PCB boards. Fortunately, as students we are able to use this great tool with a slightly limited student version for free. In addition to the cost there are extensive video tutorials, workshops, and learning tools to get started with Eagle.

Finally, Eagle is our preferred platform because most all open source hardware supplies Eagle schematics free of charge allowing us to combine many of the component boards without having to reverse engineer to design our own circuits for chips like our Bluetooth communication chip. In addition to PCB schematic and layout editing EAGLE has recently added a feature that allow the designer to go through parts included in the schematic and add them to a virtual shopping cart to then be linked to a distributor.

6.3.2 National Instruments Ultiboard

Another PCB layout software considered is National instruments Ultiboard. One advantage to Ultiboard is that at UCF NI Multisim is the preferred circuit analysis software and therefore our team has extensive experience with Multisim. Together Multisim and Ultiboard are a complete circuit design solution that would be capable of PCB layout and routing for our project. For this particular project we find it to be unattractive because although circuit design is very fluent for our team PCB design is quite new and Ultiboard is not as user friendly and simple as we would like. In addition, getting Multisim and Ultiboard is not free and even to get a student version would require purchase of the student software.

6.3.3 AutoCAD

Our final consideration for PCB design software is to use AutoCAD. The primary advantage of this option is that Autodesk is extremely generous with their software to students. Once creating a student account, Autodesk offers a student limited version of almost all of their software options including AutoCAD. In addition to the availability, there are extensive tutorials on using AutoCAD which would make adapting to their platform relatively easy. The largest drawback to this option is that AutoCAD is not the industry standard for PCB design and may not allow us to design as complex and intricate PCB layouts.

6.4 PCB House

Depending on the complexity of the PCB and the size we will need to find the most cost effective and timely PCB house to implement the BreathaLock. It is important that we plan to print multiple boards in case of print error or hardware testing and modifications. Unfortunately, it is not likely for us to need more than 10-15 boards which is considered low order quantity which forces us to pay a premium for each board. For our particular design and hardware specifications we are considering PCBWay, Elecrow, and Seed Studio as our primary PCB house options.

6.4.1 PCBWay

PCBWay is a Chinese manufacturer that seems to offer reasonable products at a very low price point. This is good for us because we are trying to prototype and implement this device as cost effectively as possible. In addition to a very low price point PCBWay is most likely our fastest option to receive our PCBs from the time we submit an order. Although some poor reviews in regards to sloppy silk screens and some sloppy vias they seem to be a good low cost option.

6.4.2 Elecrow

Elecrow is also a Chinese manufacturer that is popular for PCB fabrication. Although none of our team has experience testing products from Elecrow after some research and past customer reviews we have concluded that Elecrow has good customer support, and supplies a slightly higher quality product and a slightly higher cost.

6.4.3 Seed Studio

Seed Studio, although widely used for low price PCB fabrication, is our least attractive PCB house due to its poor customer reviews and wait time to receive product after order submission. We find Seed Studio to be not as impressive as Elecrow or PCBway although further consideration will be taken once final PCB schematic and layout are finalized.

6.5 Construction

Soldering is an extremely powerful tool that almost all engineering disciplines have some experience with. As far as electrical engineering is concerned soldering is the process of joining two or more pieces together with a filler metal composed of metals with lower melting points than the components to create an electrical passage.

Following the design and receipt of etched PCB boards it will come time for us to populate our board with our components. There are two primary methods that could be used to populate our PCB board: manually hand soldering individual components one at a time, or reflow oven. Both have their advantages and disadvantages so it is important to investigate which option we choose to populate our PCB with.

6.5.1 Hand Soldering

The most straightforward and cheapest way for us to populate our boards is to manually solder each individual component at a time using a soldering iron and the filler metal previously mentioned commonly referred to as solder. This method is great because it can be done anywhere that you have a standard wall outlet and a soldering iron, which most members already own. We can work in pieces, take breaks and work at our own pace. The drawback to this is that it is extremely time consuming since every component requires its own attention. Also this process is quite messy and depending on the experience of the person making solder joints can lead to variable error. Although this method may not be used to populate the entire board it is a complete necessity to be used when creating the BreathaLock in the event that we need to change components or add features for hardware testing. Fortunately, our team has some experience soldering and also has many tools to assist in soldering and populating our PCB board.

6.5.2 Reflow Oven

Another option to populate a PCB board is with the use of a reflow oven. The advantages of a reflow oven are that it is less time consuming and many times results in a cleaner more professional looking solder joints. To effectively use a reflow oven the PCB is laid out and components are placed onto desired pads with solder paste, a mixture of solder and flux. Once the board has components placed correctly the entire board and components are heated and once the optimal temperature is reached the solder will melt

creating electrical paths for all components. The largest downfall to this method for populating the BreathaLock PCB is that we do not have access to a professional solder oven and would most likely have to resort to hacking a toaster oven and hoping that it works effectively. Although this may give us a more professional appearance and might save time the risk of failure and additional parts required to create an over and solder components make this option quite unattractive.

6.5.3 Types of Mounting

When considering how we plan on designing our layout and how we would like to populate our PCB we need to decide as to whether we want to use through-hole mounting or surface mount technology (SMT). Through-hole mounting was standard until the 1980's when SMT became the standard. The primary advantage to through-hole mounting is durability for mechanical stress and reliability. Although BreathaLock does not need to withstand very much mechanical stress we will most likely chose to use through-hole mounting for any connectors and our power connections. As far as the other small components (i.e. resistors, capacitors, LEDS) we will chose to use surface mount to save time, space, and save cost in PCB manufacturing.

	Through-Hole	Surface Mount
Strength	Great	Good
Assembling Speed	Very Slow	Fast
Fabrication Cost	High	Low
Size	Large	Small

Table 28: Component mounting comparison

6.5.4 TI Innovation Lab

One great resource that UCF student have access to for the PCB construction in the Texas Instruments innovation lab located on the first floor of Engineering II on UCF main campus. This lab provides students with equipment to create prototype boards such as a good soldering station, oscilloscopes, DMMs, various components and much more. Not only does it offer various tangible goods that will assist in PCB construction but professionals including professors and trained advisors also monitor it. This lab will become increasingly important as we come to build and populate our PCB.

6.6 Final Coding Plan

PERT chart

As shown previously this was the agreed upon timeline

Spring 2017, EEL 4915L: Senior design 2	
Date	
1-9 to 1-20	Class begins, start building prototype
1-21 to 3-27	Test prototype
3-28 to 4-4	Order PCB
4-5 to 4-21	Troubleshoot and finalize design
4-22 to 5-02	Prepare final documentation and presentation

Table 29: Tentative Deadline for Spring 2017

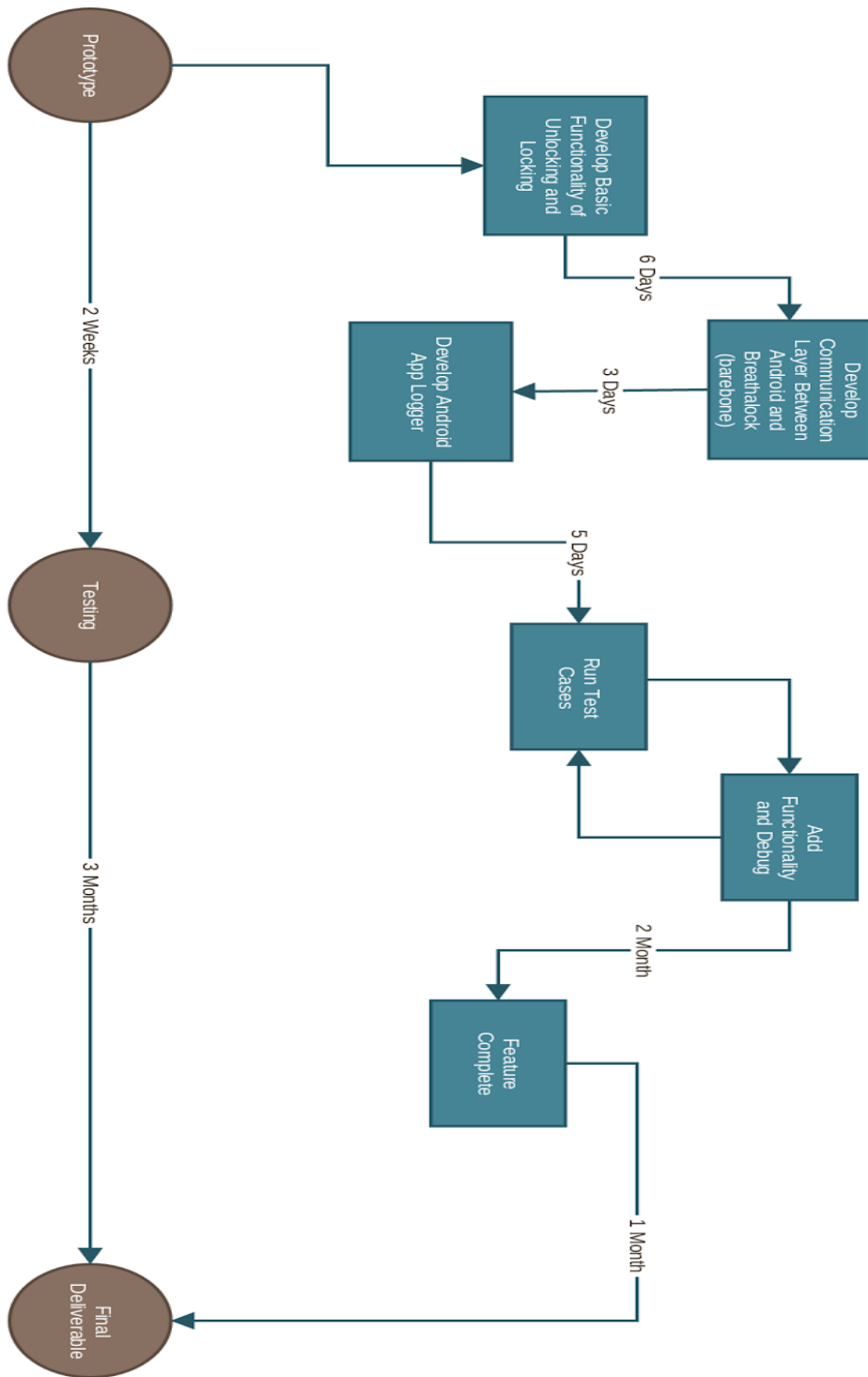


Figure 32: Tentative Deadline for Spring 2017

7. Project Prototype Testing Plan

After researching and obtaining our desired devices it is important to test them individually and as a final prototype before moving into the PCB design and final design of the BreathaLock system.

7.1 Hardware Test Environment

Creating a hardware test environment with specific constraints and controls is necessary to ensure repeatability and debug problems we may run into during the testing process. We specify this environment based on our access to limited resources and also on how strict and important a certain test it.

7.1.1 Power Supply

While testing individual components and systems working together it is extremely important to have a controlled power supply and other controls to eliminate as much error as possible. Our team will be designing our subsystems under certain conditions to perform to each power level. For the purpose of hardware testing we will be using a power supply provided by UCF labs for official data and will use batteries or USB power for small scale testing at our homes.

7.1.2 Car Access

For the purpose of testing and demonstration we will be using an untampered Ford F150 provided by a group member that will be tested regularly with a stock Ford key FOB to ensure that the vehicle is performing as expected to native instruction. Due to the fact that we don't have access to any of the car computer or software we will need to rely on simple pass/fail when commands are made by BreathaLock according to if the door unlocks or not when signaled.

In addition to pass or fail or fail base on the entry of the vehicle we will also need to test and make sure that the range of the key FOB is unaffected by our implementation. It is expected that the only thing we will be tampering with is the power supply of the key FOB itself and therefore we should not affect the range but we need to keep in mind that we are trying to maintain the same quality of industry standards that are expected by customers.

7.1.3 Cellular Devices

In efforts to avoid running into issues with our Bluetooth subsystem, as Bluetooth can be quite tricky sometimes, we will also test the cellular devices before interfacing with the BreathaLock system. Two of three group members use android phones which we will be using for initial and also final testing. Before any testing with the BreathaLock system we will be routinely connecting and disconnecting to other Bluetooth devices to confirm that the cellular devices are working properly and will create a good connection to the BreathaLock.

7.2 Hardware Specific testing

To create the best quality and functioning device we will need to verify and check that all individual subsystems and components are functioning properly. To do so the following section discussed how to connect each individual component to an Arduino and test it for functionality.

7.2.1 MQ-3 Alcohol Sensor

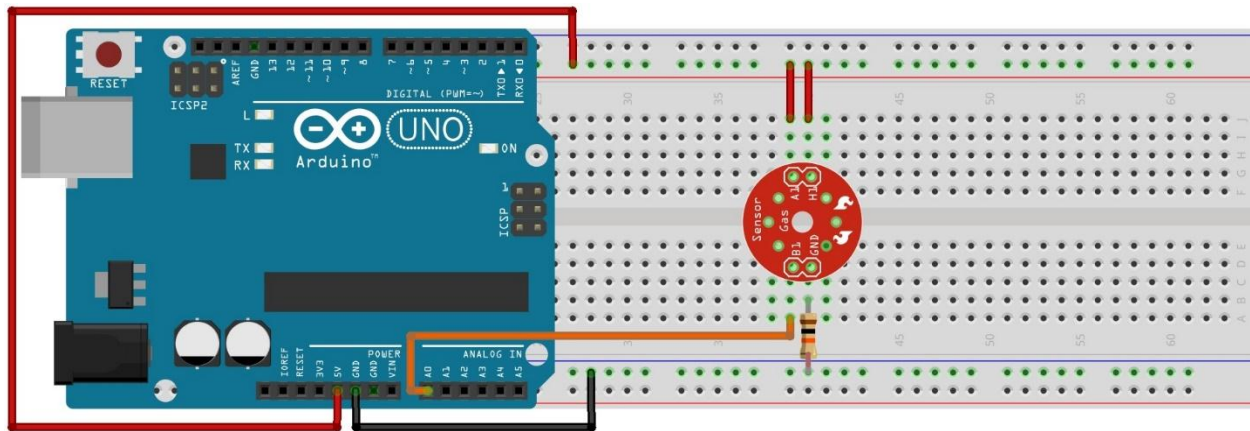


Figure 33: Alcohol sensor test circuit

To successfully ensure that our alcohol sensor is functioning properly we will need a specific test environment for the sensor alone. By connecting the MQ-3 to an Arduino UNO board we can simply read values by printing to a serial monitor on the computer to view the analog values discussed previously. Specifically, in the code provided below we are interested in the analog read pin to the Arduino that we set to pin A0. This pin is connected to the output of the alcohol sensor previously discussed that is a voltage read on the load resistor.

```
int analogPin = A0;           // Set arduino read pin to A0
void setup() {
  Serial.begin(9600);         // Set serial baud rate to 9600 bps
}
void loop() {
  int mq3_output = analogRead(analogPin);
  Serial.println(mq3_output);
  delay(200);                 //Set print delay time
}
```

Figure 34: Alcohol sensor test code writing in Arduino IDE

The above image displays example code to test the alcohol sensor by viewing the output in the serial monitor. Using Arduino IDE this simple set of code will read the analog values

coming into pin0 and print them into the serial plotter continuously. When we introduce the presence of alcohol into the sensor we expect to see the values in the serial plotter to decrease and increase again once the alcohol is removed. Attached below is a screen capture of the serial plotter output of the alcohol sensor when alcohol is introduced and taken away.

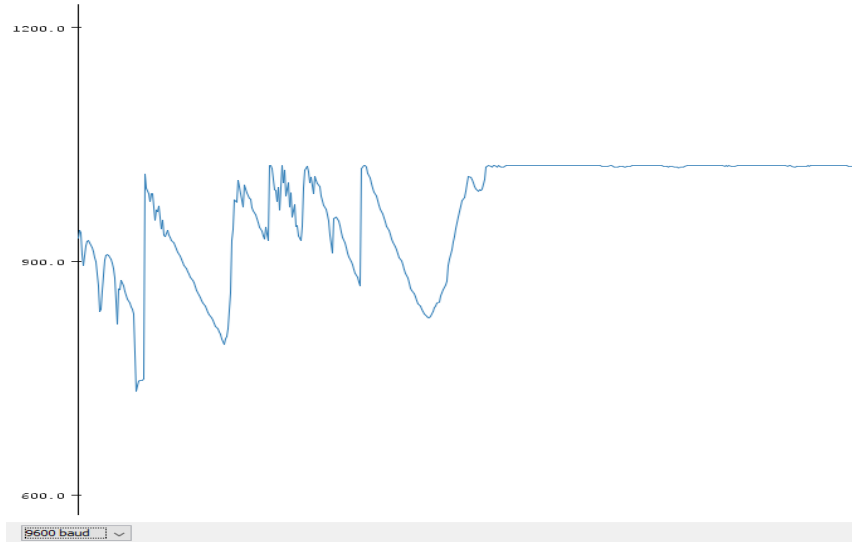


Figure 35: Alcohol sensor test output

7.2.2 Adafruit Fingerprint Sensor

Although the fingerprint sensor that we are using is prepackaged it is important to test and verify expected functionality. There are two ways that we can test the fingerprint sensor: via its native interface supplied by the manufacturer (SFG Demo), and controlling the sensor with Arduino commands. To successfully test this sensor it will be easier to first ensure functionality with the native software and secondly manipulate the sensor through the Arduino.

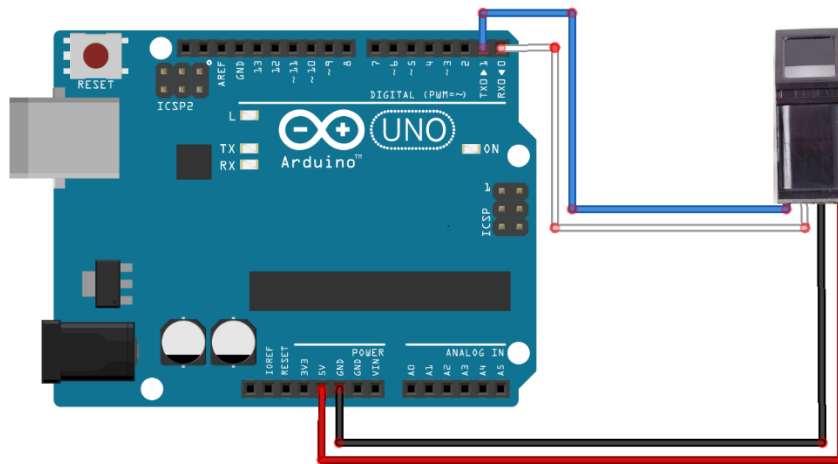


Figure 36: Fingerprint sensor test circuit

As previously mentioned in the description of the fingerprint sensor there are two ways to interface with the Adafruit fingerprint sensor. To test the sensor for our application we will be interfacing with it first with the native software provided by the manufacturer to initially verify its functionality and secondly with Arduino code to verify its functionality and customizability. We will ultimately be controlling the module without use of a computer so the native application will not be user officially but it is the simplest way to quickly check functionality.

SFG Demo:

SFG demo is the native windows application that allows interfacing with the fingerprint sensor. This application is simple and nice to use. First we must connect the fingerprint sensor to the Arduino as show in figure x above and tell the Arduino to use these pins as communication directly to the computer through USB using the following code sequence:

```
// Uploading this code will bypass the Atmega chip
// and connect the sensor directly to USB/Serial
// Red --> +5V
// Black --> Ground
// White --> Digital 0
// Green --> Digital 1
void setup() {}
void loop() {}
```

Figure 37: Arduino IDE code to bypass the Atmega328P Chip

After bypassing the ATmega328P chip on the Arduino we can connect the Arduino to the computer and launch the SFG Demo application to connect to the fingerprint sensor. Once the application is open simply select open device and select the COM port the Arduino is connected to (for this case COM 4).

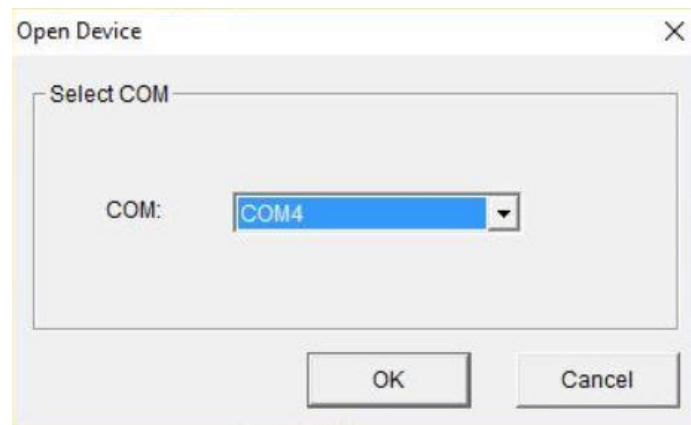


Figure 38: SFG Demo 1

Once confirmed SFG Demo should alert you with “Open Device Success”. The device is now connected and interfacing with the desktop computer.

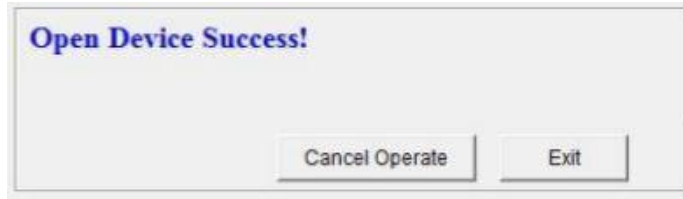


Figure 39: SFG Demo 2

Once this is complete we can match, add, search, and delete users.



Figure 40: SFG Demo 3

Arduino IDE:

After verifying that the fingerprint sensor can interface with the windows software, SFG Demo, we will test that it can interface through the Arduino. Adafruit has example code library and libraries written available and once we include them into the Arduino IDE we can call them with ease. We first conned the fingerprint sensor as shown above. Once we connect the Arduino we can open various sketches provided by Adafruit and run them to verify the sensor is working. Attached are some images of the serial monitor outputs:

We first open the “enroll” sketch provided by Adafruit to verify that the sensor if found and test to see if it can enroll a new fingerprint

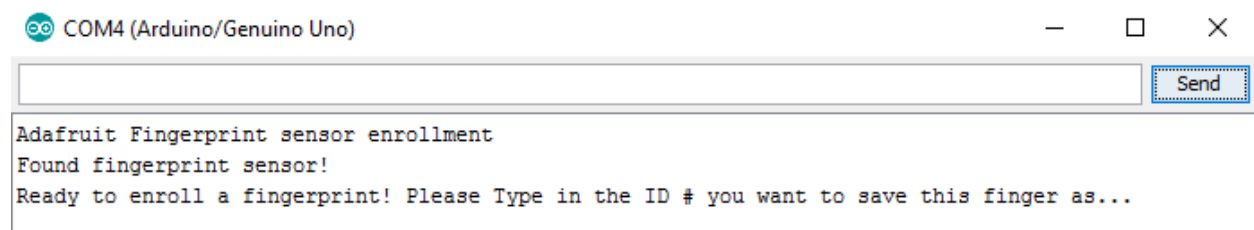
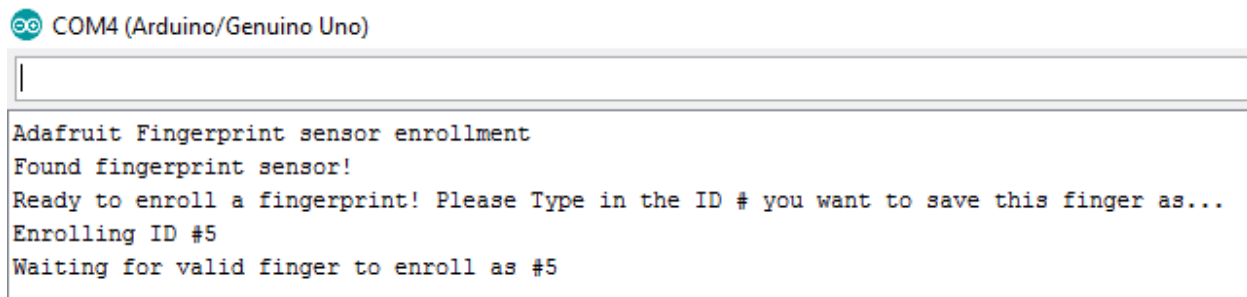


Figure 41: Arduino IDE Output 1

We will enter the number 5 to test if the sensor will save a model

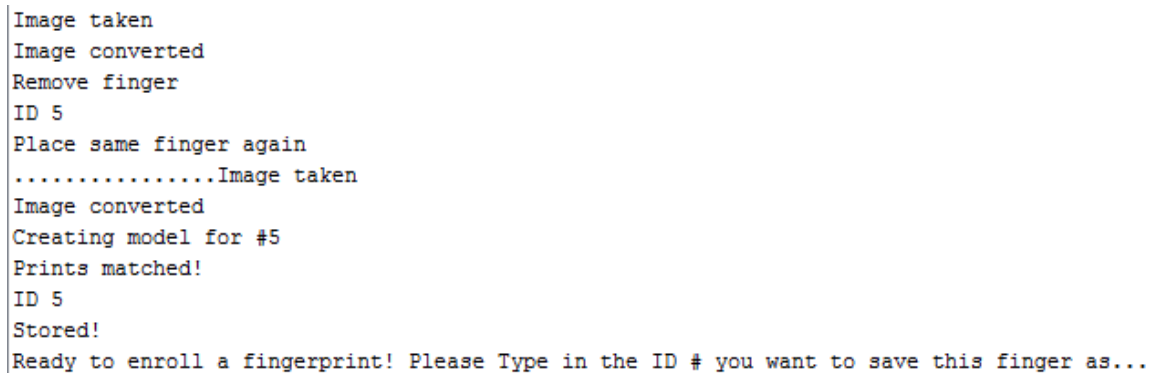


```
COM4 (Arduino/Genuino Uno)

Adafruit Fingerprint sensor enrollment
Found fingerprint sensor!
Ready to enroll a fingerprint! Please Type in the ID # you want to save this finger as...
Enrolling ID #5
Waiting for valid finger to enroll as #5
```

Figure 42: Arduino IDE Output 2

Once entered in the serial monitor we can place a finger on the sensor and wait for the sensor to get a reading and save the input.

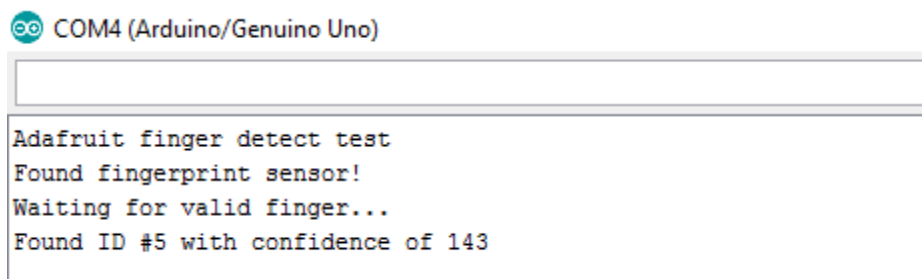


```
Image taken
Image converted
Remove finger
ID 5
Place same finger again
.....Image taken
Image converted
Creating model for #5
Prints matched!
ID 5
Stored!
Ready to enroll a fingerprint! Please Type in the ID # you want to save this finger as...
```

Figure 43: Arduino IDE Output 3

If the sensor is working correctly the above output will be shown. The sensor has now enrolled that fingerprint to model #5.

Finally, we upload and run the “fingerprint” sketch to verify that the sensor can read and reference a fingerprint.



```
COM4 (Arduino/Genuino Uno)

Adafruit finger detect test
Found fingerprint sensor!
Waiting for valid finger...
Found ID #5 with confidence of 143
```

Figure 44: Arduino IDE Output 4

The above image shows that the sensor read and confirmed that the fingerprint read is indeed #5 and with a confidence level of 143.

7.2.3 Bluefruit UART Friend Bluetooth Module

An essential part of our project is the use of Bluetooth. Because it requires the connection between two devices it is the most important device for us to test before continuing with our design procedure and implementing it into our final PCB design. We will need to apply slightly more focus and time into testing the Bluetooth component because to confirm that it is working as intended we will need to power the device, compile and run Arduino code, and then find and connect to the Arduino using an additional device. The company that we purchased this device from supplies some support and instruction on how to use and test this device. We intend do reference their materials to make the process easier and more efficient keeping in mind that we don't want to waste time developing ways to test; we simply want to verify its functionality. We choose the wiring diagram seen below and then specify which pins are to be considered later in the Arduino code.

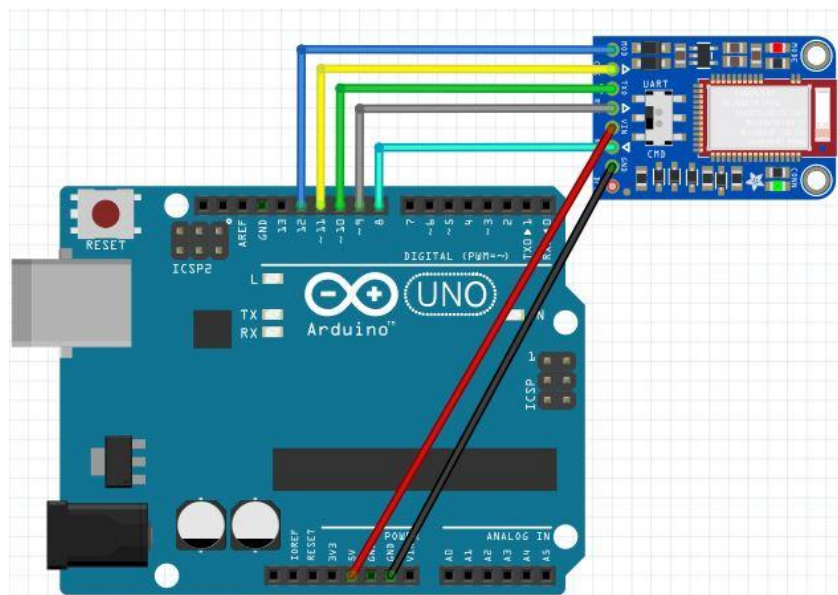


Figure 45: Bluetooth component test circuit

The Bluefruit UART Friend Bluetooth chips has a few different pin that need to be connected to the Arduino to work properly described in depth previously in the component section; here we are mostly interested in only the TX and RX pins. The TX and RX allows for UART transmission and receipt between the Arduino and Bluetooth component. We will define these in our test code as pins 9 and 10.

Test Procedure: Once wiring the Bluetooth breakout board we will first test it for basic functionality to help us understand the operation of the chip and also verify that the module is functioning properly. As previously discussed, Adafruit supplies support for most all components and particularly this component. We will first load example code in Arduino IDE and make sure that works before writing our own source code to have the module do what we need for the BreathaLock.

7.2.4 Key FOB

Unfortunately, there is no open source hardware designs or many resources for how and what the key FOB uses to unlock the car. We do instead have 3rd party working key FOBs that we can attempt to reverse engineer to work for our project. We plan to visually analyze the PCB, following traces and hoping that there are no hidden vias. Once we can create a schematic with confidence we will program the FOB to the car, de-solder the IC from the FOB and connect it to a breadboard with the schematic build on it. If we can re-create this FOB we will include this schematic into our final PCB design.



Figure 46: Image of internal PCB of key fob

As far as main functionality of the key FOB is concerned we will need to program the FOBs that we will be testing with to the specific car that we will be using. Part of the reason that we choose this remote and vehicle is because of its ability to be programmed quickly and effectively. The test procedure below is provided to explain the sequence to do so with this particular vehicle.

	Procedure
Step 1	Unlock all of the doors using the power lock switch.
Step 2	Cycle the ignition from OFF to RUN 8 times within a period of 10 seconds. The doors should lock at the end of the 8 th cycle to confirm programming mode
Step 3	In a period of 20 seconds press any button on each remote that will be programmed. The door locks will cycle each time to confirm.

Table 30: Procedure for programming key FOBs

After complementing the above sequence our key fobs will be programmed to work with our test vehicle. We will now move forward with testing and designing the PCB to function the same as this remote. Our main goal would be to replicate the PCB design into our own PCB and then simply de-solder the IC from the remote and solder it to our Breathalock system which will be pre-programmed to the car.

7.2.5 Linear Voltage Regulator

To supply power to the overall BreathaLock system we will be using a 9v battery because of its size and capacity. In order to power our processor and sensors we will need to have an output voltage of 5v and also 3v. To do this we will be using a LM7805 linear voltage regulator, and LT1761ES5-3 linear voltage regulator. Before including these components, it is important to test it to verify expected characteristics. To illustrate specific testing, we will be using NI Multisim and also a physical test with a portable digital multi-meter.

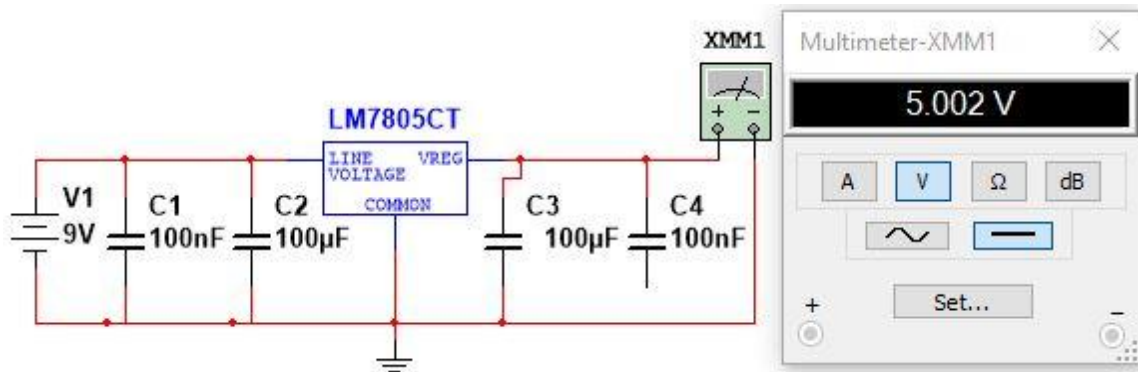


Figure 47: LM7805 linear voltage regulator test circuit

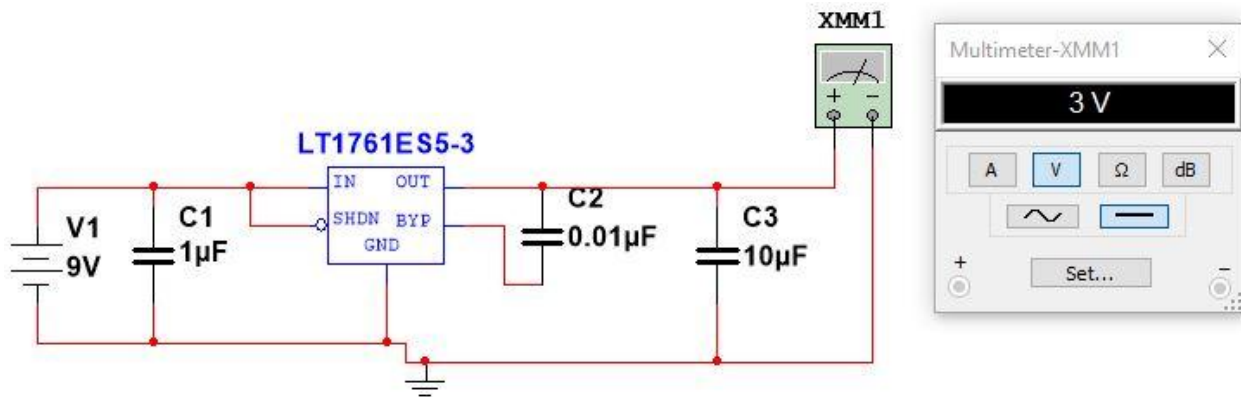


Figure 48: LT1761ES5-3 linear voltage regulator test circuit

7.3. Software Test Environment

Both the functionality of the hardware (Bluetooth, transceiver, and breathalyzer) and software (Android OS, application, and underlying code for previously stated hardware) will be included in testing. As we are not guaranteed ideal conditions with various forms of communication. Our testing environment will include a vehicle (to test the unlock and lock functionality) an Android device, and our BreathaLock and we will need to make sure they work in various conditions of connectivity (outside, indoors, etc). The engineers will handle the responsibility of testing the devices

7.4 Software Specific Testing

7.4.1 Introduction

It is imperative that we test the software for this device. Therefore, we have developed a testing plan that we feel meets the needs for the rigor of this project. This section will focus on the testing plan for the software side of our project BreathaLock. Our group felt it was important to identify the overall objective of the testing environment, the stopping conditions, and the individual test cases for BreathaLock.

7.4.2 Overall Objective for Software Test

We expect the test plan to allow the engineers to deliver a successful product that works in various conditions of connectivity and provides a way of mitigating DUI's and traffic accidents that involve individuals being under the influence.

7.4.3 Stopping Criteria & Testing Method

If errors are determined during testing, these bugs will be noted in the developer's weekly activity log. If the developer is assigned to the component that failed, the developer may fix the component immediately. Otherwise, the developer should inform, in a timely manner, the developer(s) responsible for the component of the test conditions and test results.

When testing functionality of a module test cases are written before the code itself; at that point, they are impassable. Code is written specifically to pass a given test case. When the written code successfully passes the test, the passing code is refactored into a more elegant module – without introducing any new functional elements.

By using this Test Driven Development (TDD) strategy we can improve our iterative build process in the following:

- It facilitates easy maintenance and helps alleviate scope creep
- Encourages granularity in testing; it is guaranteed that every standalone piece of logic can be tested
- Since test cases are written first, other programmers can view the tests as usage examples of how the code is intended to work

If a component is deemed complete, the developer(s) responsible for the component should notify the project manager in a timely manner. If the project manager deems necessary, a component may be sent back for further testing or development.

At the end of the project's lifecycle, all test cases will be run against the total code base to verify the functionality of the app. Communication errors will take priority over any cosmetic errors, these are more defined test cases, and the final project depends on full functionality of communication. The Software should complete all of the test cases for each module to ensure no functionality was loss by changes.

7.4.4 Description of Individual Test Cases

Test No.	1
Test Objective	Connectivity (BreathaLock to Android)
Test Description	Does the Bluetooth module (hardware) connect to the android app
Test Conditions	<ol style="list-style-type: none"> 1. Open app 2. Connect 3. Receive Bluetooth address
Expected Results	Bluetooth module successfully communicates its 48 bit address to the android device

Table 31: Description of Individual Test Case 1

Test No.	2
Test Objective	Connectivity (BreathaLock to Car) (w/o fingerprint or breathalyzer)
Test Description	Does the BreathaLock device communicate to the car and locks the door
Test Conditions	<ol style="list-style-type: none"> 1. Activate the device 2. False-Positive on BAC pass 3. Observe if door locks
Expected Results	BreathaLock successfully toggles the car door locks.

Table 32: Description of Individual Test Case 2

Test No.	3
Test Objective	Connectivity (BreathaLock to Car) (w/o fingerprint or breathalyzer)
Test Description	Does the BreathaLock device communicate to the car and the door unlocks
Test Conditions	<ol style="list-style-type: none"> 1. Activate the device 2. False-Positive on passing BAC 3. Observe if door locks unlocks
Expected Results	BreathaLock successfully unlocks the car door locks.

Table 33: Description of Individual Test Case 3

Test No.	4
Test Objective	BreathaLock Breathalyzer Sensor (Passing) (w/o fingerprint)
Test Description	Does the breathalyzer register as pass based on +/- tolerance
Test Conditions	<ol style="list-style-type: none"> 1. Receive False-Positive for fingerprint 2. Blow into Breathalyzer 3. IF BAC is lower than acceptable value 4. Register a PASS
Expected Results	This will be indicated on the device as a PASS

Table 34: Description of Individual Test Case 4

Test No.	5
Test Objective	BreathaLock Breathalyzer Sensor (Failing) (w/o fingerprint)
Test Description	Does the breathalyzer register as pass based on +/- tolerance
Test Conditions	<ol style="list-style-type: none"> 1. Receive False-Positive for fingerprint 2. Blow into Breathalyzer 3. IF BAC is higher than acceptable value 5. Register a FAIL
Expected Results	This will be indicated on the device as a FAIL

Table 35: Description of Individual Test Case 5

Test No.	6
Test Objective	Fingerprint Access (passing)
Test Description	Does placing the fingerprint give the user access to the device
Test Conditions	<ol style="list-style-type: none"> 1. Add User fingerprint to memory as passing 2. User apply fingerprint to reader 3. Register a pass
Expected Results	The device should allow the user to proceed to the next step (breathalyzing)

Table 36: Description of Individual Test Case 6

Test No.	7
Test Objective	Fingerprint Access (failure)
Test Description	Does placing the fingerprint give the user access to the device
Test Conditions	<ol style="list-style-type: none"> 1. Add user A to memory 2. Allow user B to apply finger to reader 3. Device should register as a failure.
Expected Results	The device should NOT allow the user to proceed to the next step (breathalyzing)

Table 37: Description of Individual Test Case 7

Test No.	8
Test Objective	Collect statistics to android app
Test Description	Android app will collect data from device and log it for personal or liability use
Test Conditions	<ol style="list-style-type: none"> 1. Commit an action on the device (fingerprint) 2. Send completed action to android device via Bluetooth 3. Check log on android phone
Expected Results	The logs should accurately reflect the action committed by user.

Table 38: Description of Individual Test Case 8

Test No.	9
Test Objective	More to be added
Test Description	Android app will collect data from device and log it for personal or liability use
Test Conditions	<ol style="list-style-type: none"> 4. Commit an action on the device (fingerprint) 5. Send completed action to android device via bluetooth 6. Check log on android phone
Expected Results	The logs should accurately reflect the action committed by user.

Table 39: Description of Individual Test Case 9

Test No.	10
Test Objective	Connectivity (Android to Device)
Test Description	To ensure the device is able to receive will the android app be able to communicate to the device.
Test Conditions	<ol style="list-style-type: none"> 1. Connect the android and devices 2. Once connected the android device should send a signal 3. Turn on debug light to activate signal
Expected Results	A debug light will turn on

Table 40: Description of Individual Test Case 10

8. Demonstrations

Following the design and testing phase, the BreathaLock system will go into a demonstration phase. During this time, we will need to prove that our concept has come to full realization and that specifications and design requirements have been met. As a part of the University of Central Florida curriculum, every student in the ECE department must pass senior design 1 & 2 which include demonstration of their project.

8.1 Initial Activation and Setup

The first demonstration that will be performed will be the initial activation and user setup to the device. Because our device involves the biometric user verification, when the users receive the device they will need to enroll themselves into the system so that the BreathaLock can verify their identity during regular use. To make this demonstration we will power on the device and put it into a calibration mode. In this mode the BreathaLock will request that the user place their finger onto the fingerprint sensor screen using a red background light. The system will request that the user places the same finger 2-3 times to make sure a confident reading has been taken. Once complete the user will be enrolled and saved into the fingerprints memory.

8.2 Standalone Operation

Following the initial activation and setup we will test the basic functionality of the handheld device without the use of Bluetooth. To complete this demo, we will need to turn on the BreathaLock device by button press. Once the device is on and responsive we expect to have indicators to request for identity verification. As long as the users in the demonstration have previously been enrolled, we should be able to simply light up the red LED indicator on the fingerprint sensor and read the finger upon contact with the sensor. Once this is complete the sensor will accept or deny the user. In the event that the incorrect user is attempting to use the device it will continuously loop until the correct finger is recognized. Once the correct user is verified the BreathaLock will request a sample on the alcohol sensor to test for sobriety. Similar to the fingerprint sensor, an LED will alert the user when to blow and the BreathaLock will take a reading to verify the user is under the legal limit of alcohol. Once these two tests have been passed an LED will blink alerting the user that they now have the option to unlock their vehicle.

8.3 Bluetooth Pairing

An additional feature to the BreathaLock system is the ability to connect with a cellular device over a Bluetooth connection to assist the process and also display additional information. Before entering the unlock sequence we must demonstrate the BreathaLock's ability to pair to a cellular device. This will be done by holding down a button on the BreathaLock to turn on and search for a Bluetooth link. Once on, the user will open the application developed by our team and instruct the cellular device to pair to the BreathaLock. The user will be confirmed that the BreathaLock is paired on the android application.

8.4 Connected Operation

The final demonstration will be the operation of the BreathaLock device while it is connected to an android device. During this demonstration the android application will be the most prominent feature to focus on. The demonstration will mostly follow the same procedure as the “standalone operation” but in this demonstration most of the indicators and requests will be interactive and more visual through the application. The user will first open the app and be confirmed that the BreathaLock is connected and ready. Once confirmed the user will be prompted to place their finger onto the fingerprint sensor for user verification. The application will walk the user through the process with visual indicators as to whether they need to place the finger again or if they pass or fail. Assuming the correct user is operating the device, the application will welcome the user into the next operation of testing the blood alcohol content of the user. In this step the application will indicate when to blow, how long, and when the BreathaLock has enough sampling to decipher what the users BAC is. One main feature of the android application is that we will be able to display the exact BAC of the user instead of an LED indicator of whether or not above or below 0.08. Concluding the alcohol sensor testing the application will display the exact BAC of the user and whether or not they can or cannot drive. In the event that the user is above the legal limit the BreathaLock application will advise the user to wait and blow again or to wait a long time and rest. Finally, the last feature will be the ability for the BreathaLock to act as a general breathalyzer to any user with the intent to inform other drivers on their sobriety.

9. Administrative content

9.1 Milestone Discussion

The milestone for the BreathaLock project is show in the figures below.

AUGUST 2016

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22 School Begins	23	24	25	26 Seniors design project idea assignment	27	28
29	30	31				
		NOTES:				

SEPTEMBER 2016

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
			1	2	3	4
5	6	7	8	9 Initial Project document divide and Conquer Due	10 Research design concepts and Parts	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25 Write updated Divide and Conquer paper
26	27	28	29	30 Updated Divide and Conquer Paper Due		
		NOTES:				

OCTOBER 2016

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
					1 Individual Research and prototype development and code design	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29 Write table of contents	30
31	NOTES:					

NOVEMBER 2016

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
	1	2	3	4 Table of Contents due	5 Write initial draft	6 _____
7 _____	8	9	10 →	11 Initial Draft Due	12 Finish prototyping and code	13 _____
14 _____	15	16	17	18	19	20 _____
21 _____	22	23	24	25	26	27 _____
28 →	29 Write final documentaion	30				
		NOTES:				

DECEMBER 2016

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
			1	2	3	4
5	6 Final documentation due , Order parts	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	
		NOTES: Parts include, microcontroller, fingerprint scanner , alcohol sensor, batteries, bluetooth module				

JANUARY 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
						1
2	3	4	5	6	7	8
9 Class begins, Start building prototype	10	11	12	13	14	15
16	17	18	19	20	21 Test Prototype	22
23	24	25	26	27	28	29
30	31	NOTES: Prototype should be on breadboard				

FEBRUARY 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28					
		NOTES: Whole month of february is allocated to test the prototype				

MARCH 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22 Design PCB	23	24	25	26
27	28 ORDER PCB	29	30	31		
		NOTES: First half of March is for testing the prototype				

APRIL 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
					1	2
3	4	5 Troubleshoot and finalize design	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22 Prepare final documentation and presentation	23
24	25	26	27	28	29	30
		NOTES:				

MAY 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				
		NOTES:				

From the figures above we can see that the first five months of the project consist of declaring a project idea. Once a project idea has been established, we research further into the idea to start designing /develop a physical prototype. This design develop and research must be documented for the Senior Design 1 paper. In addition, the first five months consist of ordering parts. In our case, parts were ordered and breaded boarded way before the milestone deadline thus the BreathaLock project is ahead of schedule.

The last five months of the project consist of troubleshooting our developed design. Once troubleshooting is complete, we then design our PCB to order. Once PCB has come in and populated, there is roughly 2 weeks allocated to make sure that the populated PCB is working properly. Once everything is finalized, there is about a week and a half to prepare for final presentation.

9.2 Budget and Finance Discussion

Since the BreathaLock project does not have any sponsorship there are two available options to finance the project

9.2.1 Finance option 1

Finance option 1 consist being crowd funded. Crowd funding mediums such as Kickstarter and Gofundme are a great place to get funded for any project. However, there are some advantages and disadvantages to budget option 1.

Advantage	Disadvantage
-No money out of personal funds	- Can take a long time to reach desired amount
	- Project needs awareness
	- Project may never reach desired funding

Table 41: Finance Option 1 Advantage/Disadvantage table

9.2.2 Finance option 2

Finance option 2 consist of paying for the project ourselves. There are also advantages and disadvantages from paying the project ourselves

Advantage	Disadvantage
- Project is paid for fast	-money comes from person funds
- Freedom of when to buy	
- Project is funding independent of outside factors	

Table 42: Finance Option 2 Advantage/Disadvantage table

We chose to go with Finance option 2. We chose to pay for the project ourselves because we have decided as a team that the total cost of the project is not unreasonable. Also, Finance option 2 allows the project to be paid for fast, rather than wait on donations as we only have two semesters to finish the entire project. Additionally, we have to decide to split the project cost between each member. This ensures that each member will pay for the project equally. The table below shows the maximum the BreathaLock project may cost. Extra quantities are considered to account for broken parts during prototyping.

Part	Quantity	Cost	Total
Microcontroller	2	\$10-\$20	\$20-\$40
Fingerprint Sensor	2	\$30-\$45	\$60-\$90
Bluetooth Module	2	\$20-\$35	\$40-\$50
Blood alcohol Sensor	2	\$30-\$45	\$60-\$90
Battery	6	\$3-\$5	\$6-\$10
Programmable Key fob	2	\$20-\$30	\$40-\$60
PCB	3	\$20-\$30	\$60-\$90
Breathalyzer	1	\$20	\$20
Estimated Grand Total			\$316-\$450

Table 43: Parts Cost table

9.3 Group management

In terms of group management, Charles Taylor, the computer engineer, is in charge of programming the microcontroller to process the data that are coming from all the sensors and relaying that information to the mobile app. The microcontroller will need to be able to process the right user fingerprint and reject users who aren't register. In addition, the microcontroller will need to be programmed to process the blood alcohol content data and respond according if the data is above or below the legal limit.

Nam Ngo, the electrical engineer, is in charge of powering on the key remote, the microcontroller, the alcohol gas sensor, the fingerprint sensor and interfacing the Bluetooth communication module with the microcontroller. The key remote and the microcontroller should be powered on by battery while the alcohol gas sensor, fingerprint sensor, and the Bluetooth module should get its power through the regulated power supply from the microcontroller.

Nicholas Fraser, the electrical engineer, is in charge of connecting the alcohol gas sensor and the fingerprint sensor with the microcontroller. In addition, he is also in charge of electrically interfacing the microcontroller to allow and block the unlock signal. The key remotes unlock system should be electrically connected and controlled by the microcontroller. If the microcontroller reads that the user is below legal limit, then the unlock signal is able to be sent to the car. However, if the microcontroller reads that the user is above the legal blood alcohol content then electrically, the key remote cannot send out an unlock signal.

Though each member is assigned a task within the project, we as a group mutually agreed that we work as a team thus if any group member is struggling with their respective tasks

then the roles of each members are subject to change accordingly. For example, if Nicholas Fraser is having trouble connecting the sensors to the microcontroller then Nam Ngo can take on that task in exchange for his contribution of powering on the sensors.

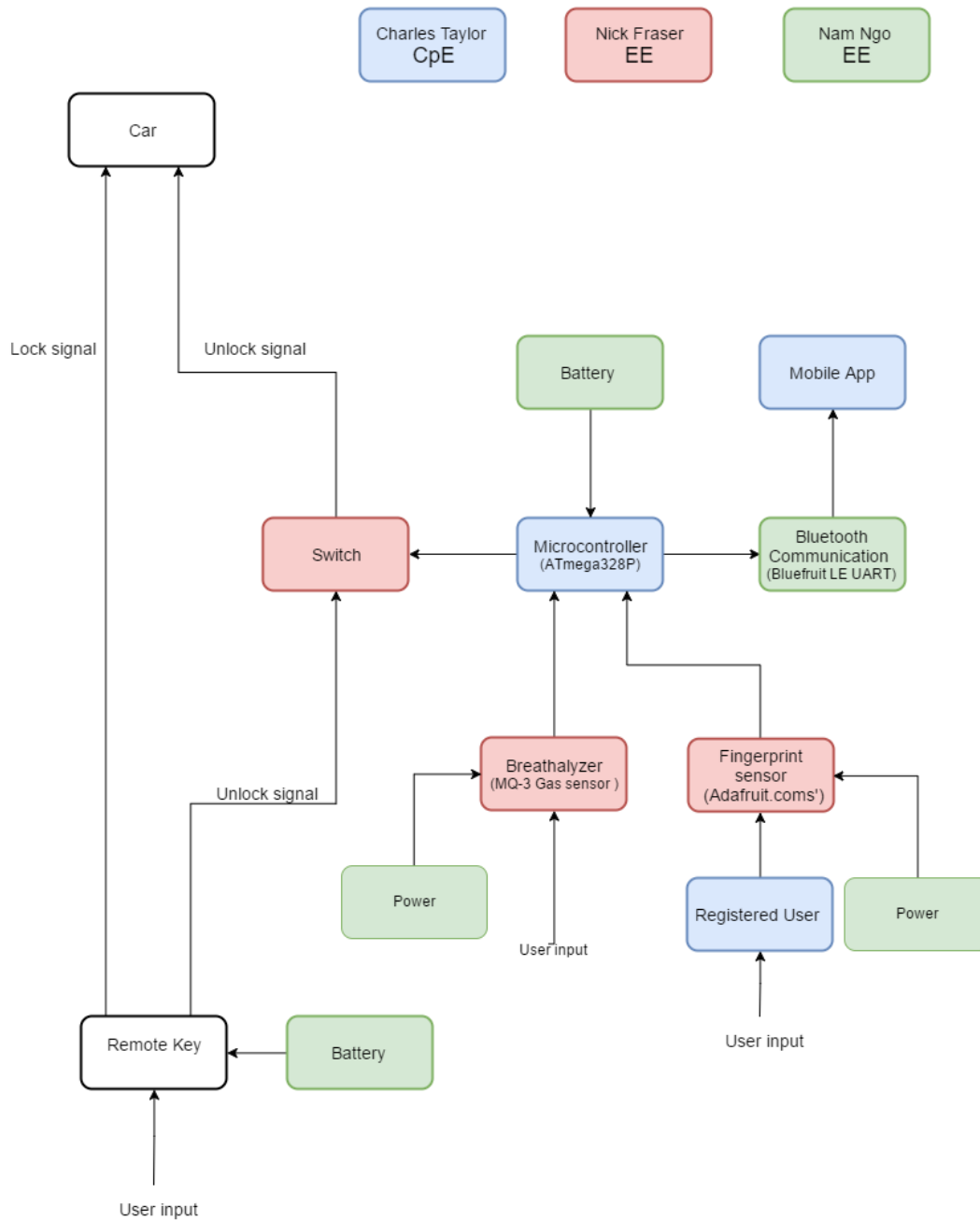


Figure 49: Project management flowchart

Appendix A List of Figures

Figure 1: 3D Wireframe model of BreathaLock	6
Figure 2: Optical Fingerprint Diagram	11
Figure 3: Passive capacitive touch Diagram.....	11
Figure 4: Active capacitive touch Diagram.....	12
Figure 5: Battery Diagram.....	14
Figure 6: Harvard architecture	26
Figure 7: System Architecture	31
Figure 8: All components by letter	35
Figure 9: Fritzing breadboard test wiring diagram	36
Figure 10: Breadboard testing of all components	36
Figure 11: Fritzing wiring schematic.....	37
Figure 12: Bluefruit LE UART Friend (BLE) external image.....	38
Figure 13: Fingerprint sensor external image	39
Figure 14: MQ-3 sensor external image and pin reference.....	40
Figure 15: Sensor operation schematic.....	41
Figure 16: Circuit analysis of sensor	41
Figure 17: Java ART framework	44
Figure 18: C++ Flow Chart	47
Figure 19: Eclipse IDE.....	49
Figure 20: Eclipse ADT Plugin	50
Figure 21: An opened sketch	51
Figure 22: Android Studio IDE	52
Figure 23: Selection screen for Android devices.....	53
Figure 24: AVD emulating an android device	53
Figure 25: Handheld Device Functionality.....	55
Figure 26: Android Device Functionality.....	56
Figure 27: List View	59
Figure 28: Dialog window example.....	59
Figure 29: Flowchart of system hardware implementation	60
Figure 30: Eagle Schematic of Bluetooth Module.....	61
Figure 31: Eagle Schematic of alcohol sensor breakout board	61
Figure 32: Tentative Deadline for Spring 2017.....	67
Figure 33: Alcohol sensor test circuit	69
Figure 34: Alcohol sensor test code writing in Arduino IDE.....	69
Figure 35: Alcohol sensor test output.....	70
Figure 36: Fingerprint sensor test circuit	70
Figure 37: Arduino IDE code to bypass the Atmega328P Chip.....	71
Figure 38: SFG Demo 1	71
Figure 39: SFG Demo 2	72
Figure 40: SFG Demo 3.....	72
Figure 41: Arduino IDE Output 1	72
Figure 42: Arduino IDE Output 2	73

Figure 43: Arduino IDE Output 3	73
Figure 44: Arduino IDE Output 4	73
Figure 45: Bluetooth component test circuit.....	74
Figure 46: Image of internal PCB of key fob	75
Figure 47: LM7805 linear voltage regulator test circuit	76
Figure 48: LT1761ES5-3 linear voltage regulator test circuit	76
Figure 49: Project management flowchart.....	92

Appendix B List of Tables

Table 1: House of quality trade off table	8
Table 2: Arduino Uno Low power mode table.....	15
Table 3: Arduino Uno Advantage/Disadvantage table	16
Table 4: Arduino ProMicro Advantage/Disadvantage table.....	16
Table 5: Arduino Mini Advantage/Disadvantage table	17
Table 6: Raspberry Pi 3 Model B Advantage/Disadvantage table	18
Table 7: MSP430 Advantage/Disadvantage table	18
Table 8: Microcontroller Decision Table	19
Table 9: Decision Table.....	21
Table 10: Fingerprint Scanner Decision Table	22
Table 11: MQ-3 Advantage/Disadvantage table	23
Table 12: MR513 Advantage/Disadvantage table	23
Table 13: Gas Sensor Decision Table	24
Table 14: Selected Component Operating voltage.....	24
Table 15: Power Option Decision Table	25
Table 16: Power option Decision Table.....	26
Table 17: Part selection summary Table	28
Table 18:Component descriptions.....	35
Table 19: Arduino Specific Connection.....	37
Table 20: Bluefruit LE UART Specifications	38
Table 21: Fingerprint sensor technical characteristics	39
Table 22: MQ-3 sensor technical characteristics	40
Table 23: Technical Functionality 1	56
Table 24: Technical Functionality 2.....	57
Table 25: Technical Functionality 3.....	57
Table 26: Technical Functionality 4.....	58
Table 27: Technical Functionality 5.....	58
Table 28: Component mounting comparison.....	65
Table 29: Tentative Deadline for Spring 2017	66
Table 30: Procedure for programming key FOBs	75
Table 31: Description of Individual Test Case 1.....	78
Table 32: Description of Individual Test Case 2.....	78
Table 33: Description of Individual Test Case 3.....	78
Table 34: Description of Individual Test Case 4.....	79
Table 35: Description of Individual Test Case 5.....	79
Table 36: Description of Individual Test Case 6.....	79
Table 37: Description of Individual Test Case 7.....	80
Table 38: Description of Individual Test Case 8.....	80
Table 39: Description of Individual Test Case 9.....	80
Table 40: Description of Individual Test Case 10.....	81
Table 41: Finance Option 1 Advantage/Disadvantage table	90
Table 42: Finance Option 2 Advantage/Disadvantage table	90

Table 43: Parts Cost table 91

Appendix C Citations and Permissions

Citations

- [1] K. Townsend. (2016, September 30). *Introducing the Adafruit Bluefruit LE UART Friend* (1st ed.) [Online]. Available: <https://cdnlearn.adafruit.com/downloads/pdf/introducing-the-adafruit-bluefruit-le-uart-friend.pdf>
- [2] L. Ada. (2015, May 4). *Adafruit Optical Fingerprint Sensor* [Online]. Available: <https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/overview>
- [3] M. S. (2016, August 08). *Getting Started with Arduino and Genuino UNO* [Online]. Available: <https://www.arduino.cc/en/Guide/ArduinoUno>
- [4] H. Barragan. *Power Regulator 5v: LM7805* [Online]. Available: <http://wiring.org.co/learning/topics/power5lm7805.html>
- [5] Sparkfun. *Alcohol Gas Sensor - MQ-3* [Online]. Available: <https://www.sparkfun.com/products/8880>
- [6] Sparkfun. *Gas Sensor Breakout Board* [Online]. Available: <https://www.sparkfun.com/products/8891>
- [7] Digi-Key. *Linear Technology LT1761ES5-3#TRMPBF* [Online]. Available: <http://www.digikey.com/product-detail/en/linear-technology/LT1761ES5-3-TRMPBF/LT1761ES5-3-TRMPBFCT-ND/1629845>

Datasheets:

- [8] Texas Instruments, “ μ A7800 Series Positive-Voltage Regulators” LM7805 datasheet, May. 1976 [Revised May. 2003].
- [9] Linear Technology, “LT1761 Series 100mA, Low Noise, LDO Micropower Regulators in TSOT-23” LT1761ES5-3 datasheet

Permissions

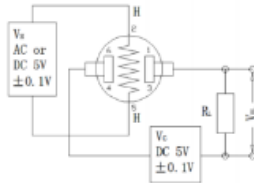


Nicholas Fraser

Today, 7:13 PM

sales@winsensor.com

👍 Reply all | ▾



Download Save to OneDrive - University of Central Florida - UCF

To whom this may concern,

I'm writing to request permission to use an image from a datasheet that you have created. I am specifically interested in Fig2 from "MQ-3 ver1.3 - Manual.pdf". I am an electrical engineering student at the University of Central Florida and would like to use this image as a part of a senior design paper that I am working on. The datasheet will be cited and the paper will give your company all credit for this image. Please let me know if there is any further questions or information you would need to grant me permission. I've attached the image for ease of reference.

Thank you for your time,

Nicholas Fraser

MQ-3 Ver1.3 Datasheet Permission



Alyssa Rong <sales@winsensor.com>

Today, 12:15 AM

Nicholas Fraser

👍 Reply all | ▾

Dear Nicholas,

Thanks for kind inquiry.

It is okay for you to use the figure, but pls kindly mark the source. Many thanks!

Best regards,
Alyssa Rong

=====

Zhengzhou Winsen Electronics Technology Co., Ltd

No.299, Jinsuo Road, National Hi-Tech Zone, Zhengzhou 450001, China

Tel: +86-371-67169097

Fax: +86-371-60932988

Skype: Alyssa Rong

Email: sales@winsensor.com