Keyless Ignition Shutoff with Brethalyzer

ECE230: Intro to Embedded Systems

Final Project

Tom Kirchhoffer and Derick Miller

Instructor: Dr. Michael Jo

# Table of Contents

[**Table of Contents**](#_6m3ygxllc7uo) **1**

[**Introduction**](#_qssusonrto4d) **2**

[Overview of project features](#_iih4dsl1ul97) 2

[How the project meets the required specifications:](#_9qaxx92s8mil) 2

[Design Specification](#_i6xlto1hk6tn) 3

[**User Manual**](#_alvx9s4zlhj2) **5**

[**Hardware design and implementation**](#_dghldi5tarr6) **10**

[Circuit Diagram](#_2yzqre50v6ns) 10

[Module and Pin Hardware Explanations](#_lndvfprt5z0l) 11

[LCD](#_p6u2dhbu0113) 11

[LEDS](#_btk5tcjvr71z) 11

[Button](#_6084hik4adz7) 11

[Ethanol Sensor](#_es58vk54dr7q) 11

[Servo](#_tfr0kjzzxgd) 11

[Passive Buzzer](#_b5gavoa4gdq5) 11

[RFID/Arduino](#_j7kom7nm24ln) 11

[Sensor Hood](#_1y5ofo8d2260) 12

[**Software design and implementation**](#_uibkc39h2cf) **13**

[Code Organization](#_vq5kq5i46szp) 13

[BAC Floating Point Calculations and Conversions](#_ewrnutuzo9fm) 13

[**Test Plan and Test Results**](#_iecewnsq7s7b) **15**

[**Bill of Materials**](#_97hujex49b2e) **23**

[**References and Acknowledgement**](#_ssf48o8y002g) **24**

# Introduction

## Overview of project features

We created a module to simulate a keyless ignition shutoff attached to a breathalyzer. The purpose of this project is to demonstrate our understanding of the material taught to us in ECE230. We have implemented code from multiple class examples as well as code from prior projects we worked on for class.

The system features an RFID reader for identifying the RFID tag in a car key then activating the rest of the system. Once activated an LCD, LED, and Speaker provides the driver with various forms of feedback. This feedback will direct the user on when to breathe into the ethanol sensor.

Once the user has breathed into the sensor and we have an accurate reading, the system processes the voltage data from the sensor into a useful and equivalent BAC reading. The system will then show the user their reading and indicate if they are allowed to drive. If they are, the ignition is triggered.

## How the project meets the required specifications:

The specifications were as follows:

1. The project must use the MSP432P401R (or other microcontroller with approval)
2. The embedded circuit must include at least one sensor (input)
3. The embedded circuit must include at least one actuator (output)
4. The embedded project must make use of at least one timer
5. The embedded project must make use of at least one interrupt
6. Includes at least 6 unique sensors/actuators
7. Uses at least two peripherals that were not implemented in a lab exercise OR uses them in a significantly different configuration (e.g. SPI configuration rather than I2C)

Description of how we meet each specification in order

1. The MSP432P401R was used for everything except the RFID which was run off the arduino
2. An ethanol sensor was used to measure the alcohol in breath
3. An LCD screen used to provide the user with instructions and feedback
4. The project uses Systick to create a delay function that is used multiple places in the code. TimerA is also used to create a PWM signal for the speaker and servo motor
5. An interrupt is used to detect when the RFID tag has been read
6. List of Sensors and actuators:
   1. LED (actuator)
   2. LCD display (actuator)
   3. Speaker (actuator)
   4. Servo Motor (actuator)
   5. Ethanol Sensor (sensor)
   6. Button (sensor)
   7. Input data from arduino
7. Timer32 is used which was never required for a lab project. We also used the Power controller to set our device to low power mode then trigger a wake up in an interrupt.

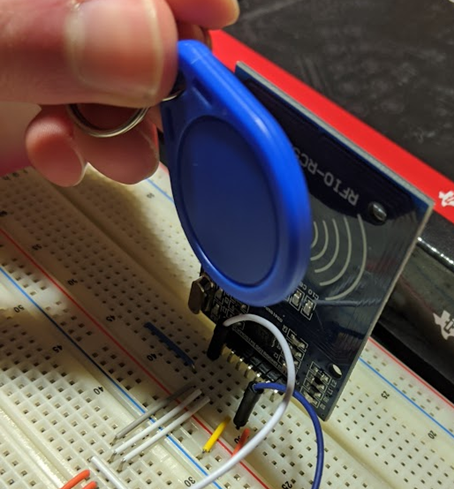
## Design Specification

The following specification defines the general end product we wanted to achieve. They are written based on the chronological order of what we expect to happen, In the following specs multiple forms of feedback refers to (text, color, sound, ect.)

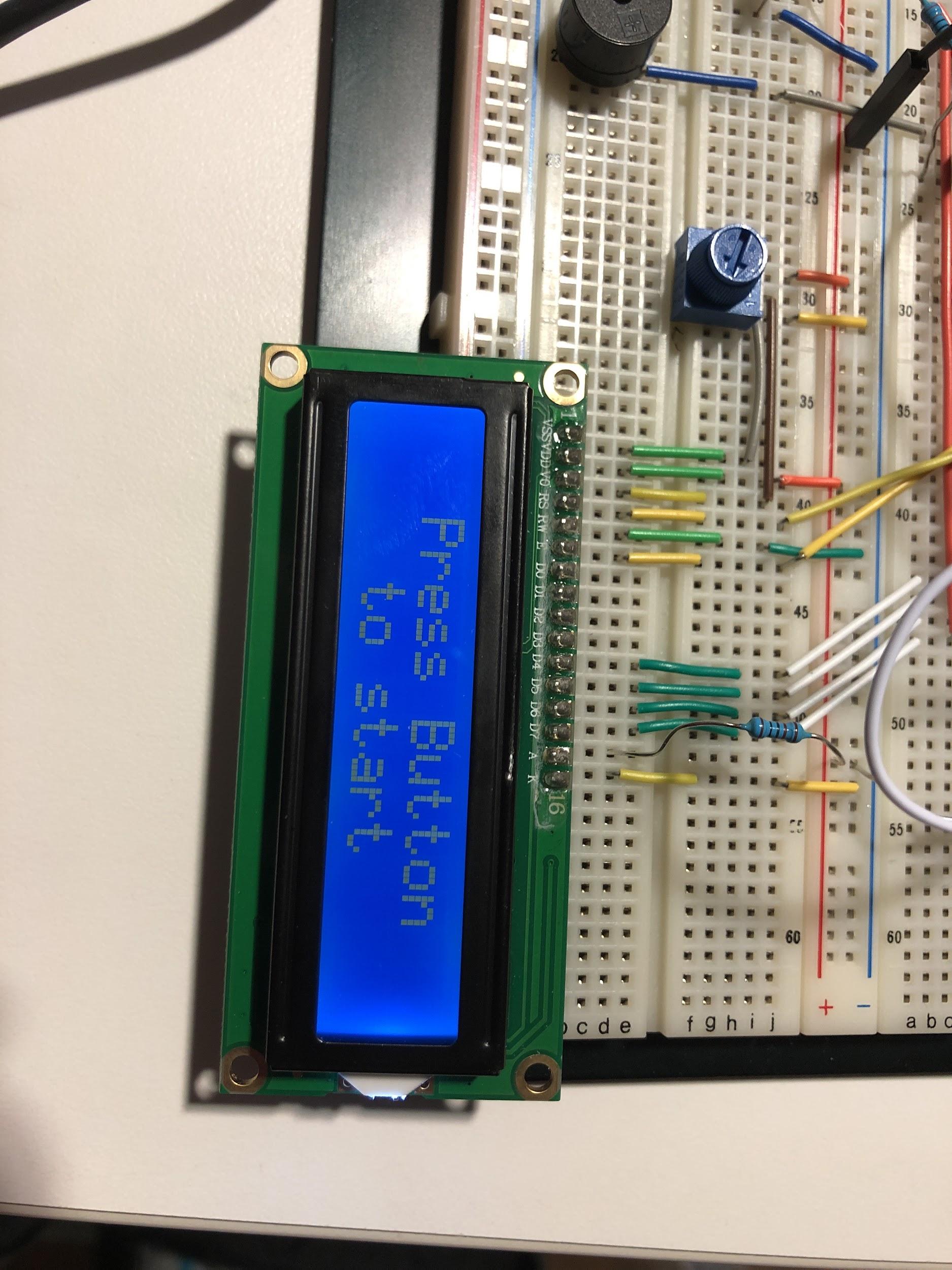
1. Will begin in low power mode
2. Will exit low power mode when key is detected
   1. Key will be ID from a single RFID Tag
   2. Reading a tag should always give the same result
   3. Key should be checked on the arduino so the main device doesn’t need to exit low power mode.
3. When the device exits low power mode gives the user an indication the device has turned on (something like “warming up”, “welcome”, “turning on” ).
4. Give the user a button to start the breathalyzer test when they are ready
5. Device must activate sensor then wait at least 6sec for the sensor to warm up
   1. Give the user some feedback that they should wait and let them know what will happen to show they should start blowing
6. Indicate that the user should start blowing using multiple forms of feedback
7. Have the user blow for 5 seconds before collecting sensor data. Then collect data every 2 milliseconds for 1 second followed by a second of blowing with no data collection before telling the user to stop blowing
8. Tell the user to stop blowing with multiple forms of feedback
9. Average the sensor data collected and process this voltage value into an equivalent BAC value.
   1. Use the correct equations - should never be less than 0%
   2. Reading will be inaccurate above 2% BAC.
10. Display BAC to 2 decimal places
11. If BAC less than .08% turn servo 60 degrees

# User Manual

Begin by placing the a RFID tag next to the RFID Reader this will cause the device to read the identification information from the tag. If the ID matches the expected ID on the key then the other systems will begin to power on.

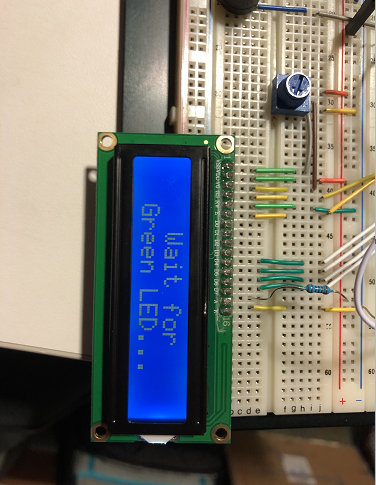
**Figure 1.** RFID tag and reader

The other sensors then turn on providing the user with direction. At this point you should see the LCD display “Device ready: press start button.”

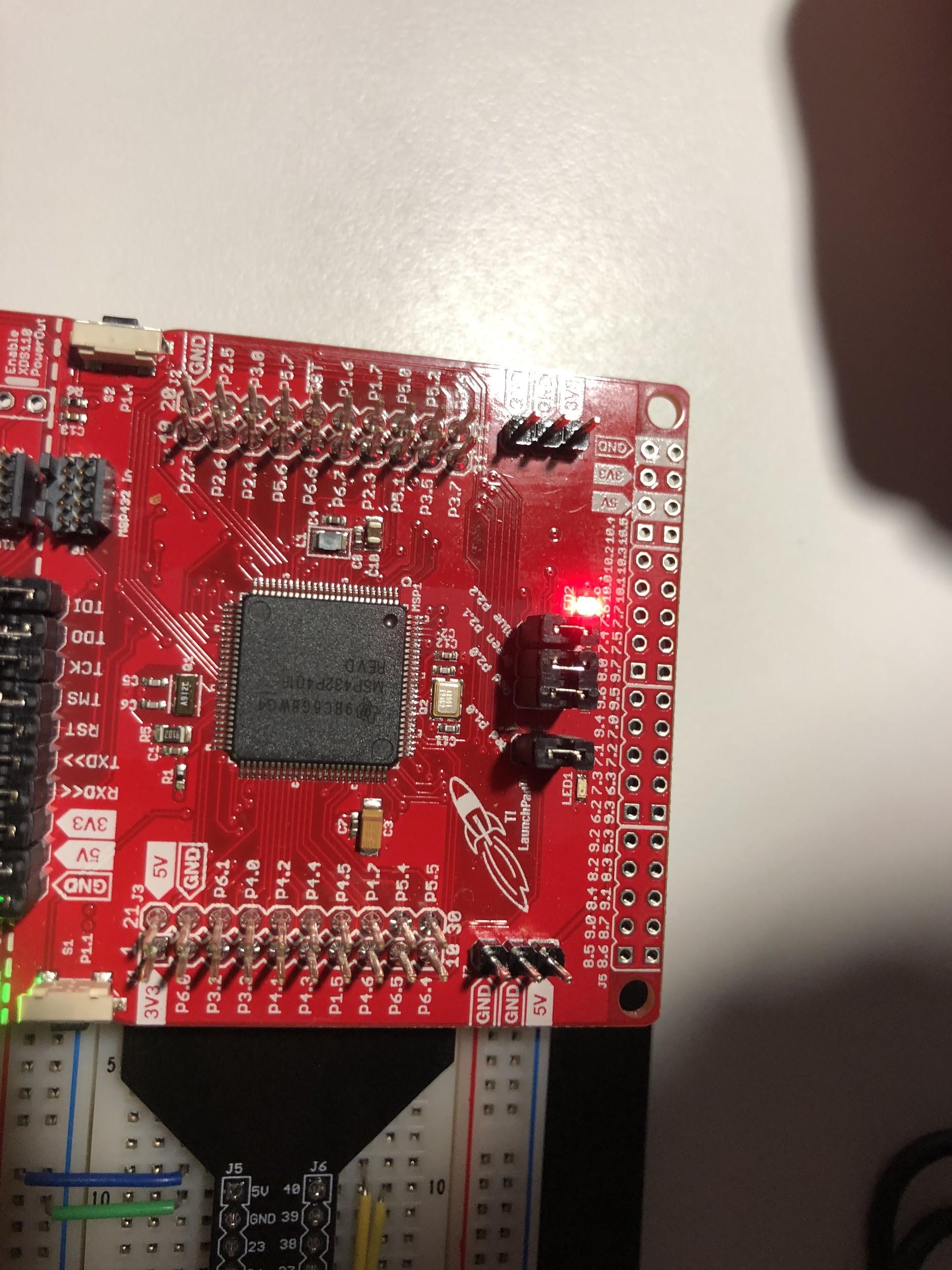


**Figure 2.** Push Start Button Message

Upon press of the button the ethanol sensor will turn on, the LCD will tell the user to breath in and wait for the beep, the LED will turn on RED, and the device will wait 6 sec for the sensor to warm up and the user to breath in.



**Figure 3.**  Wait for Green LED message

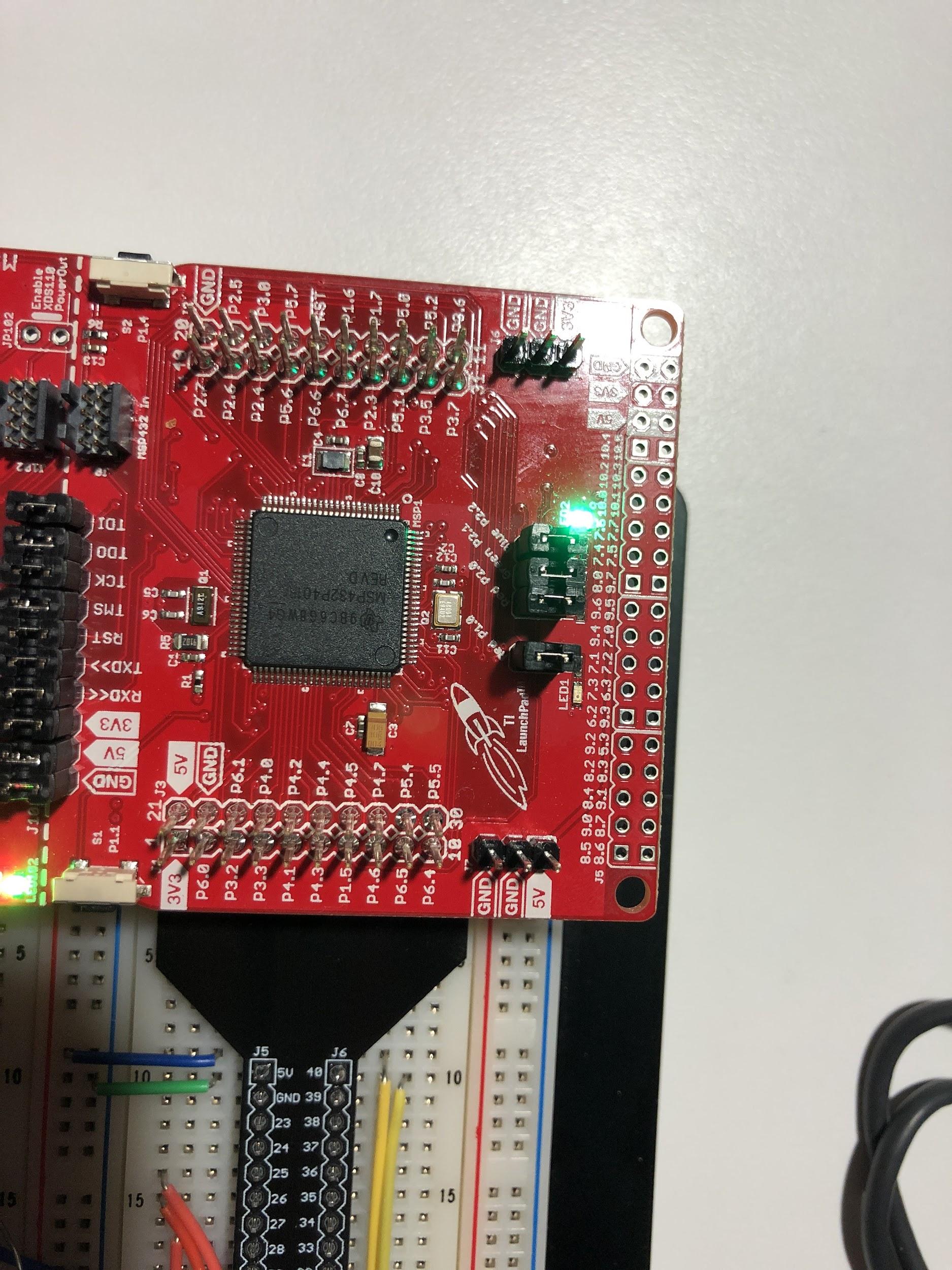


**Figure 4.** Red LED Toggled

At the 6 second mark the LCD will tell the user to breath out into the sensor, a tone will be heard from the speaker, and the LED will turn GREEN.



**Figure 5.** Start Blowing into Brethalyzer message



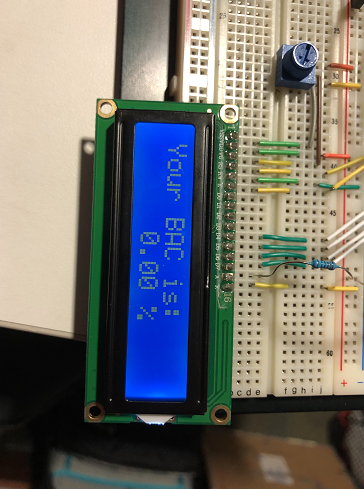
**Figure 6.** Green LED Toggled

The device will wait 5 seconds then begin storing voltage values from the sensor. It waits 5sec as the first 5 seconds of breath comes from the top of the lungs which does not give accurate BAC readings. After the 5 sec period the device will add voltage data taken every 2ms together for 1sec. The speaker then gives off a different tone and the LED changes to BLUE indicating the data has been collected and the user can stop breathing into the device.



**Figure 7.** Blue LED Toggled

We will divide the summed voltage by 50 to get the average. This digital value will then be converted to a BAC(Blood alcohol concentration) value. That will be displayed to the user in a message on the LCD that also indicates if the user is fit to drive.



**Figure 8.** BAC Percentage message

If the user is unfit to drive then a corresponding tone is played and the LED is turned to RED. If the user is fit to drive a corresponding tone is played and the LED is turned to GREEN. The servo motor is also turned 60 degrees simulating an ignition start.

# Hardware design and implementation

## Circuit Diagram

## Module and Pin Hardware Explanations

### LCD

The LCD is set up in 4 pin mode to conserve pins. All pin assignments are entirely arbitrary.

### LEDS

The RGB LED we are using is built into the Launchpad so those pins must be used. We chose to use the LED built into the launchpad because it results in a cleaner circuit.

### Button

The button we are using is built into the Launchpad so those pins must be used. We chose to use the button built into the launchpad because it results in a cleaner circuit.

### Ethanol Sensor

The ethanol sensor is designed as a voltage divider. We measure the voltage across 200kΩ resistor and the 1µF capacitor. The capacitor serves to to average out the voltage values read. The ethanol sensor works as a resistor that decreases in resistance as the ethanol concentration in the air increases. We had to make slight adjustments to our calibration due to the input pin having a pullup resistor that is comparable to the current through our gas sensor. This increases the voltage at point of measurement by a roughly fixed amount.

### Servo

The servo needs to run on pin 2.4 because it is configured through the timerA peripheral.

### Passive Buzzer

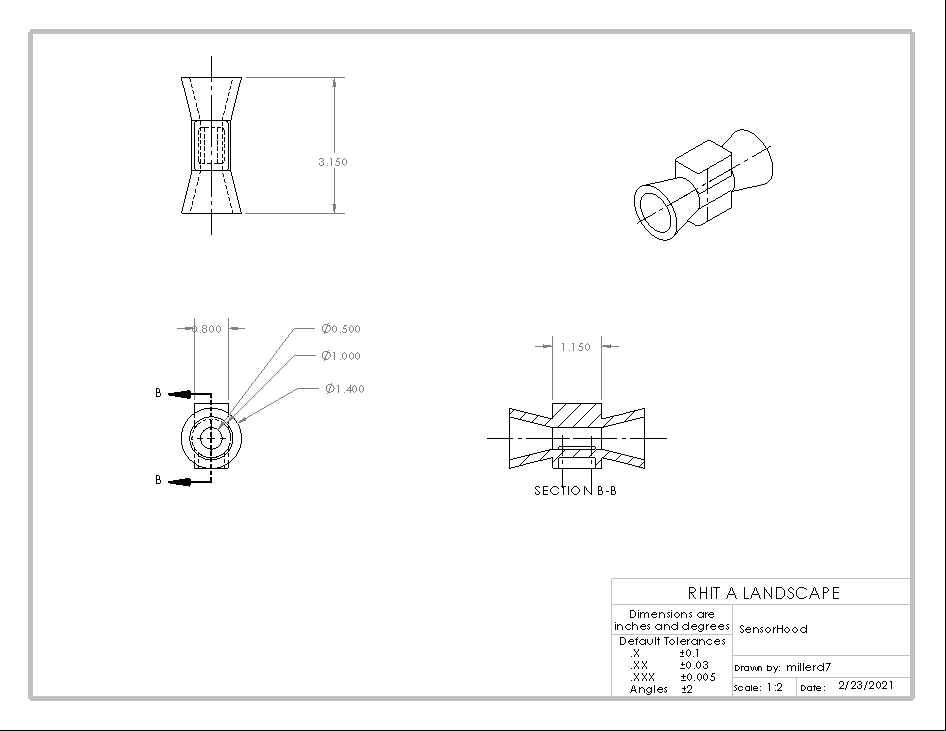
The passive buzzer pin is entirely arbitrary and can be remapped if desired. We also added a potentiometer into the circuit so we could control volume. The pin is attached to a transistor in the circuit rather than directly powering the circuit because it does not provide enough current.

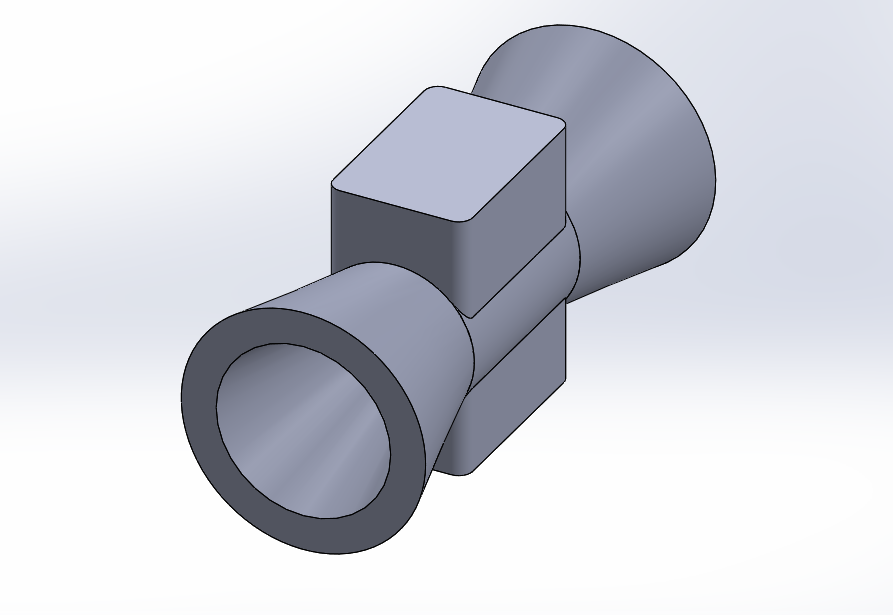
### RFID/Arduino

Originally when designing for the RFID we intended to interface directly with the RFID using the ti SPI library so we had MOSI and MISO attached to P1.6 and 7. This proved to be too challenging as it was difficult to get the RFID reader to communicate with the RFID tags as we found little to no documentation on the structure of the tags. So we had to try and mimic the arduino library on TI. When this proved too much we offloaded the communication to the arduino and had it test for the key we wanted then trigger a wakeup on our main device by setting pin 5.0 to high.

### Sensor Hood

The sensor hood is vital for maintaining a constant airflow over the sensor and not onto the sensor. As direct airflow onto the sensor could damage the sensitive layer of the alcohol sensor and render it useless for reading values accurately in the future. The sensor hood also reduces the chances of moisture accruing on the sensor.





# Software design and implementation

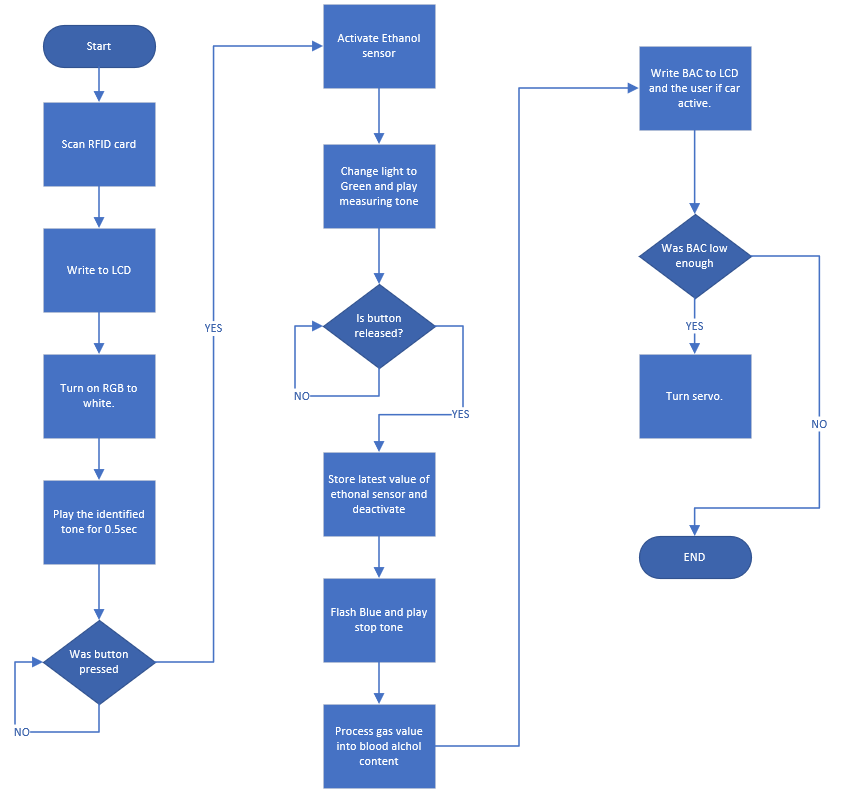
### Code Organization

Our Keyless Ignition Shutoff with Brethalyzer has a lot of separate parts, so to keep each of the separate parts organized we used header files for each of the separate parts. These include header files for the LCD, Servo, Buzzer, and Sensor, this helped make our code modular and easier to debug.

### BAC Floating Point Calculations and Conversions

Analog voltage is calculated in an interrupt that runs 1 time every .2ms, every time the interrupt is called it calculatest the digital value read, and converts it into a voltage with a -1.9V offset as we had measured a 1.9V signal coming from the analog pin. These 50 analog voltage samples from the sensor are then averaged over this one second period and this average is used for the BAC calculations.

The first calculation is the resistance of the sensor module, this is done using a voltage divider. Where our sensor is in series with a 200kΩ resistor, we can find the resistance by using ohm’s law. Where we calculate V over the sensor, and current I running through this series circuit, gives us the resistance of the sensor Rs. Next we use our equation Where the corresponds to the ratio of calculated resistance to the known resistance at 0 ppm (.



# Test Plan and Test Results

| **Requirements** | **What am Testing for** | **Procedure** | **Pass condition: (Failure will be considered anything that does not completely meet the pass condition)** |
| --- | --- | --- | --- |
| Will begin in low power mode | Low power mode is enabled on startup | Analyze code to show that low power mode is enabled | Begins in low power mode after disabling necessary parts |
| Result: | | | |
|  | | | As can be seen in the code we enter into low power mode and don't leave until keyless becomes false. This code is near the very start of the code after breathalyzeroff() so this PASSES |
| Will exit low power mode when key is detected | An interrupt is triggered that causes that the device to exit low power mode | Analyze code to show that low power mode is disabled | Low power mode is exited |
| Result: | | | |
|  | | | As can be seen in the code the power state is set back to Active Mode during the interrupt and keyless is changed to false allowing the program to continue so this PASSES |
| Key should be checked on arduino so the main device doesn’t need to exit low power mode. | Key is checked on the arduino | Analyze arduino code to show that the key is checked on the arduino | Key is checked on the arduino |
| Result: | | | |
|  | | | As seen in the code the if statements check for new tags and retrieve their id information and the if statements check if we have the correct key then set pin2 high if we do. So this PASSES |
| When the device exits low power mode gives the user an indication the device has turned on (something like “warming up”, “welcome”, “turning on” ). | "Warming up" is displayed | Check visually that Warming up is displayed on the LCD | warming up is displayed |
| Result: | | | |
|  | | | As seen from the image warming up is displayed so this PASSES |
| Give the user a button to start the breathalyzer test when they are ready | Message press to start is displayed on LCD and when S1 is press the testing continues | Once the device has reached the "press to start" display wait at least a minute to check that nothing changes then press Select! and confirm the program continues | Nothing changes until S1 is press then the program continues |
| Result: | | | |
|  | | | As you can see from the code nothing is allowed to continue while S1 equals not pressed so this PASSES |
| Device must activate sensor then wait at least 6sec for the sensor to warm up | The device waits 6sec after activating the sensor before telling the user to blow | Use breakpoints to measure clock cycles between activating the sensor and changing the LCD and check that is within 2ms | The clock cycles are within 2ms of 6sec |
| Result: | | | |
|  | | | 72138763 Clock Cycles on a 12Mhz Clock is approximately 6.01156 seconds which is within our requirement of 2 milliseconds thus it PASSES |
| Give the user some feedback that they should wait and let them know what will happen to show they should start blowing | A red led turns on when the button was originally pressed and the user is told to wait for the light to turn green before blowing. | Visually confirm that a reasonable message matching the idea is displayed | The LED is red and the LCD displays a message telling the user to wait for the green light |
| Result: | | | |
|  | | | This tells the user to wait for the green LED and there is a RED light off camera so this PASSES. |
| Indicate that the user should start blowing using multiple forms of feedback | A beep happens at the same time the light turns green and the LCD display changes to "start blowing" | Visually confirm that the light changes green and the LCD reads "start blowing". | The message on the LCD tells the user to start blowing and the LED turned green |
| Result: | | | |
|  | | | As seen from the image Begin Blowing into Brethalyzer is displayed, a green LED can be seen off camera, and tone is playing so this PASSES |
| Have the user blow for 5sec before collecting sensor data. Then collect data every 2ms for 1s followed by a second of blowing with no data collection before telling the user to stop blowing | Data is not read from the sensor until 5 sec after the light turns green | Use breakpoints to measure clock cycles between turning the LED green and the first data collection is within 2ms of 5sec | Clock cycles are within 5ms of 5sec |
| Result: | | | |
|  | | | 60004609 Clock Cycles on a 12Mhz Clock is approximately 5.00038 seconds which is within our requirement of 2 milliseconds thus it PASSES |
| Tell the user to stop blowing with multiple forms of feedback | Light turns blue, a tone is played, the LCD displays "stop blowing" | Visually confirm the LED turns blue, "stop blowing" is displayed. | The LCD displays a message similar to stop blowing and the LED turns blue |
| Result: | | | |
|  | | | Stop blowing is displayed the LED is blue and a tone was heard so this PASSES |
| Average the sensor data collected and process this voltage value into an equivalent BAC value. | Sensor Data is averaged | Analyze code to show data is averaged | Code sums read value then divides by the number of times read |
| Result: | | | |
|  | | | As you can see the analog values are summed at every read then divided at the end to give an average so this PASSES |
| **^** | Voltage is converted to BAC using a reasonable equation | analyze code to show conversion is rational | Conversion is sensible |
| Result: | | | |
|  | | | Conversions are shown, as long as resistance never goes to 0 the value will never be inf. Therefore this PASSES. |
| Display BAC to 2 decimal places | BAC is displayed to 2 decimal places on LCD | Visually confirm the displayed BAC is going to 2 decimal places | BAC is displayed with 2 decimals |
| Result: | | | |
|  | | | LCD message is shown, and BAC percentage is rounded to 2 decimal places therefore this PASSES. |
| If BAC less than .08% turn servo 60 degrees | If BAC less than .08% the servo turns 60 degrees | Visually confirm that servo turns 60 degrees when BAC is less than 0.08% | Servo turns 60 degrees when BAC was less than .08 % |
| Result: | | | |
| This has been visually confirmed but can not be shown as it involves motion so this PASSES | | | |

# Bill of Materials

| Qty | Part | Description | Source | Unit Cost |
| --- | --- | --- | --- | --- |
| 1 | MSP432P401R | Primary computing unit (microcontroller) | ECE230 | Preowned |
| 1 | Arduino Uno | Secondary computing unit (microcontroller) | ECE160 | Preowned |
| 1 | Adafruit MiCS5524 | Ethanol Sensor | Amazon | $22 |
| 1 | Passive Buzzer | Passive Buzzer | ECE160: Elegoo Kit | Preowned |
| - | Wires, Resistors, and Capacitors | Connecting Elements | ECE160: Elegoo Kit | Preowned |
| 1 | SG90 Servo Motor | Servo Motor | ECE160: Elegoo Kit | Preowned |
| 1 | Breadboard | Breadboard | ECE parts room | Preowned |
| 1 | MFRC522 RFID Module | RFID Reader | ECE160: Elegoo Kit | Preowned |
| 1 | Sensor Hood | Air flow control element | 3D printed by Treven Ritko-Siros | FREE |
| 1 | Booze | For Testing and Calibration | Undisclosed | $30 |
| 1 | Test tube | For Testing and Calibration (solution vessel) | ChemE Stockroom | Borrowed |
| 1 | Rubber stopper with bubbler | For Testing and Calibration (Reactant) | ChemE Stockroom | Borrowed |
| - | Ethanol/Distilled Water | For Testing and Calibration (solution) | ChemE Stockroom | FREE |
|  |  |  | TOTAL: | $52 |

# References and Acknowledgement

We like to acknowledge the following.

Arduino RFID Library for MFRC522 from github (miguelbalboa/rfid)

[GitHub - miguelbalboa/rfid: Arduino RFID Library for MFRC522](https://github.com/miguelbalboa/rfid)

We used this library extensively when trying to develop our own code for the MSP432P401R to interface with the RFID reader. When this ended up being too troublesome we switched to actually using this library on an arduino.

Texas Instruments MSP432 Driver Library

The TI driver library was used extensively throughout the code. Almost every line is written utilizing a function from the library.

ECE230 Header Files

The Delays.c, Delays.h, lcd.c, lcd.h were already written for us in example code from the course. We wrote one or two functions in each class as well.

MFRC522 datasheet

Created by NXP Semiconductors, was referenced when developing our MSP432P401R for interfacing directly with the RFID reader.

The MiCS-5524 is a compact MOS sensor.

Written by SGX Sensortech, used for calibrating the ethanol sensor

Adafruit MiCS5524 CO / Alcohol / VOC Gas Sensor Breakout

Written by Adafruit Industries Used when calibrating the ethanol sensor