EE333 – Project 2 Final Report

Section 1 Group 10

Aaron Sledge & Mohd Harith Arsyad

December 12, 2022

Table of Contents

1. Introduction

One thing that is always taught in our society is that in general the truth is good and that lying is bad. But it is basically impossible to tell if someone you are talking to is telling the truth or telling a lie without physical evidence or something of that nature. That is why we at first set out to create a lie detector circuit, because lie detectors are an easy way to determine if someone is telling the truth or not. However, this did not meet the requirements of project II so instead we decided to add three additional sensors to our lie detector circuit which were all safety themed. With the collection of safety themed sensors, we decided that our circuit should be a multipurpose home security device instead.

2. Initial Design

As stated, before our initial design consisted of only a lie detector. This type of lie detector used the users galvanic skin response as a way to determine whether someone is lying or not. It is stated that when a person lies the average resistance across their fingers drops. So, our lie detector used the person's skin resistance as a pseudo-resistor in a voltage divider circuit.

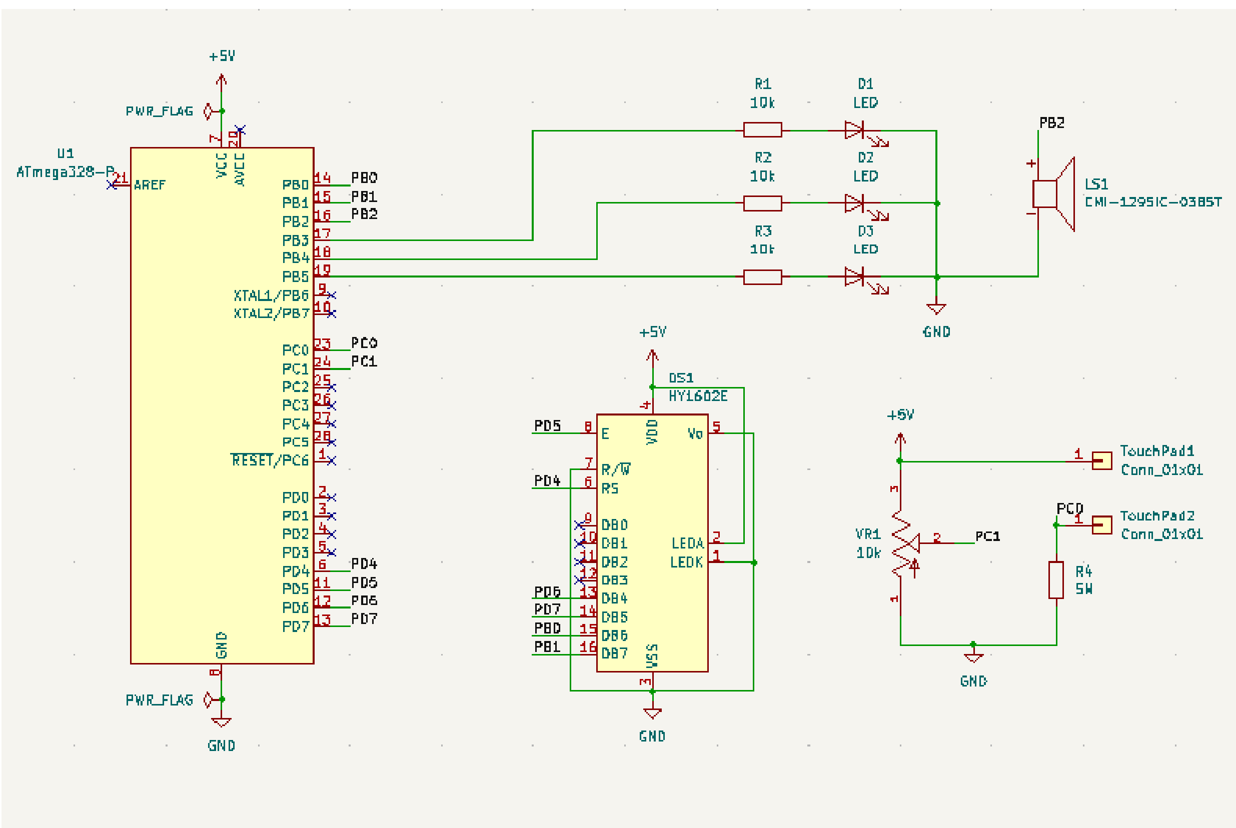

Figure 1. Lie Detector

Our initial design consisted of three LEDs; red, blue, and green to indicate lying, unsure, and telling the truth respectively. In addition to that there is also a buzzer that sounds when the lie detector determines the user is lying. The LCD screen is meant to show messages such as “user lying” or “user not lying”. And finally in the bottom right corner there is a potentiometer whose wiper is being read for a reference voltage in order to calibrate the lie detector for the user. Seeing as how everyone’s skin resistance is not the same . As well as touch pads for the user to place their fingers on that are connected in series with a 5 Mohm resistor. As can be seen in figure 1 the Arduino reads the user voltage (voltage after the touch pad and before the 5 Mohm resistor) and compares that voltage to the reference voltage to determine whether someone is lying or not. That aspect of the circuit will be explained further during the “simulation of initial design” section of the report.

3. Simulation of Initial Design

In the beginning we only had time to simulate and test the lie detector part of the circuit. We were still trying to figure out how to wire the LCD screen to the Arduino in order to make it display messages. We decided to use Tinkercad as our simulation platform due to it being able to be used to program an Arduino and it containing the majority of the parts that we were using in our initial design.

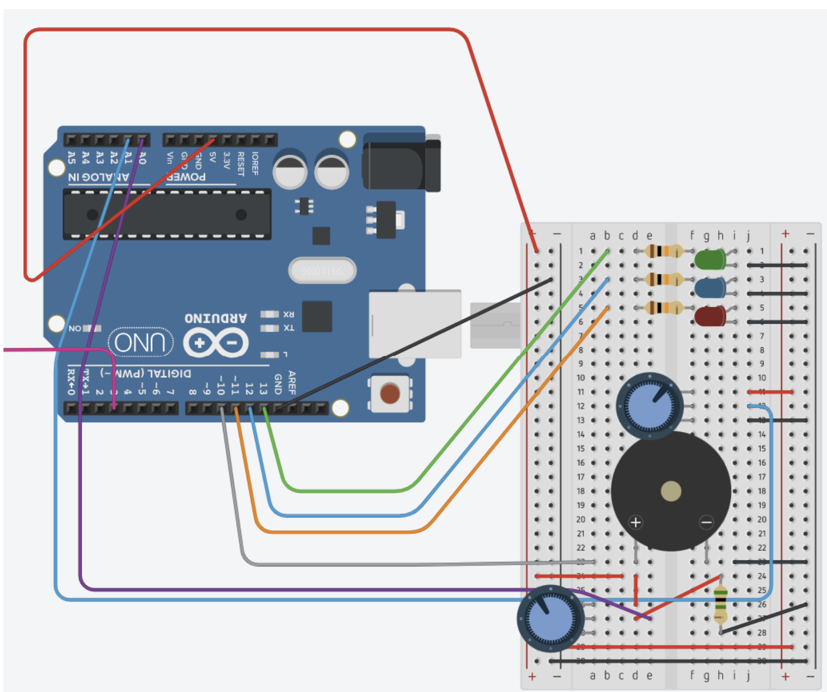

Figure 2. TinckerCad simulation setup

Figure 2 shows the initial design of the lie detector circuit inside of Tinkercad. The potentiometer closest to the LEDs provides the adjustable reference voltage for the Arduino to read. The second potentiometer represents the users skin resistance which is connected directly to 5 volts. And is also connected in series with a 5 Mohm resistor connected to ground.

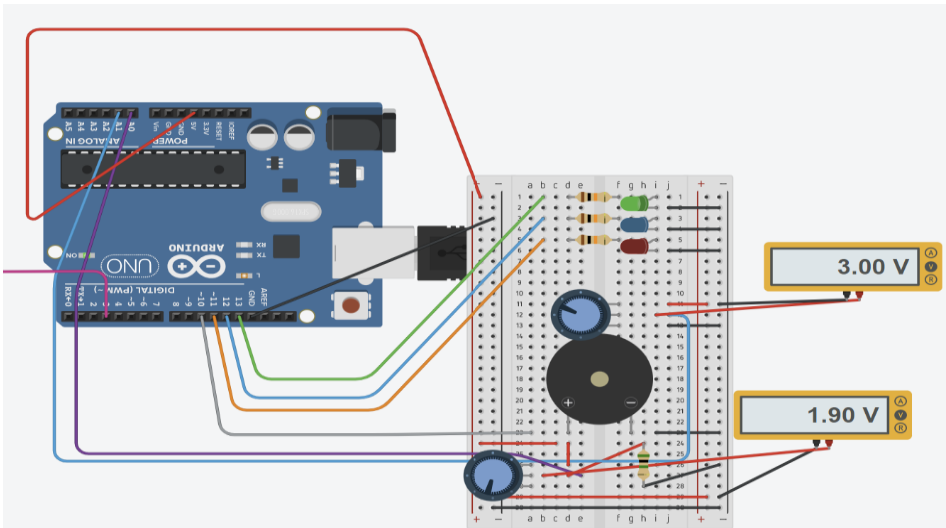

Figure 3. Simulation of user telling the truth

The Arduino reads the user voltage and compares it to the reference voltage. In figure three you can see that the user voltage is lower than the reference voltage which means that the user is telling the truth and the green LED lights up.

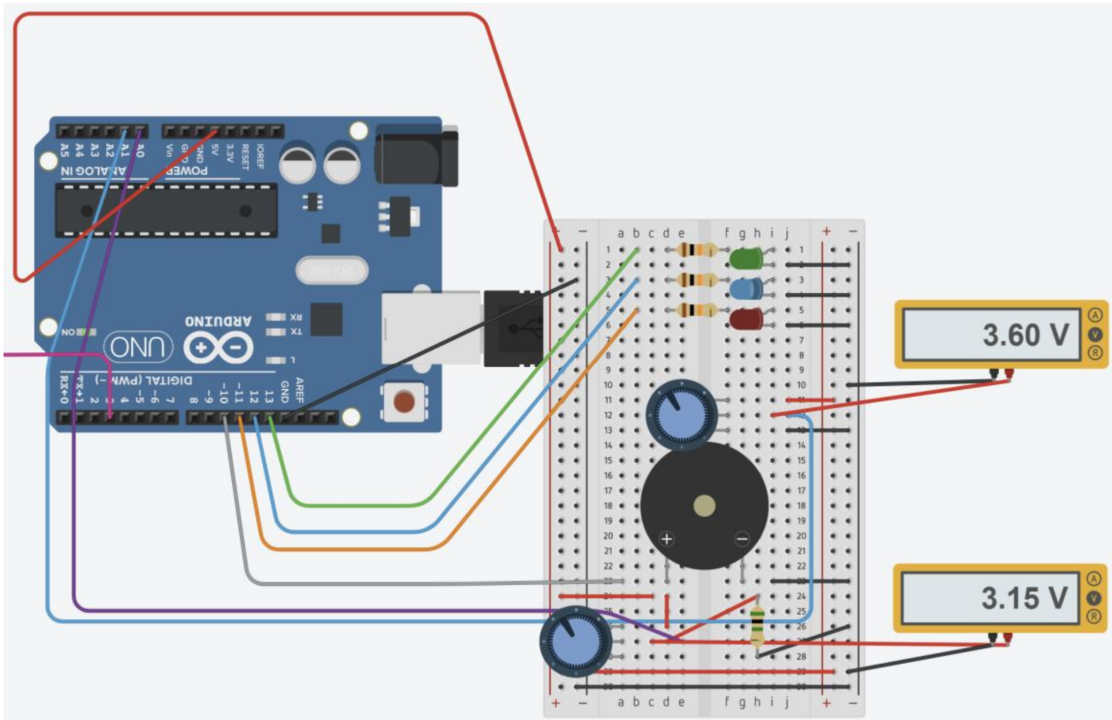

Figure 4. Buffer zone area

For the blue LED it represents when the difference between the user voltage and the reference voltage is less than 500 mV. We decided to give an ample buffer zone due to how quickly a person's skin resistance changes over time. However, a person's skin resistance changes over time going up and down in magnitude even when the user is doing nothing. So, we figured that a buffer zone is necessary so that the system would only detect when someone is actually lying.

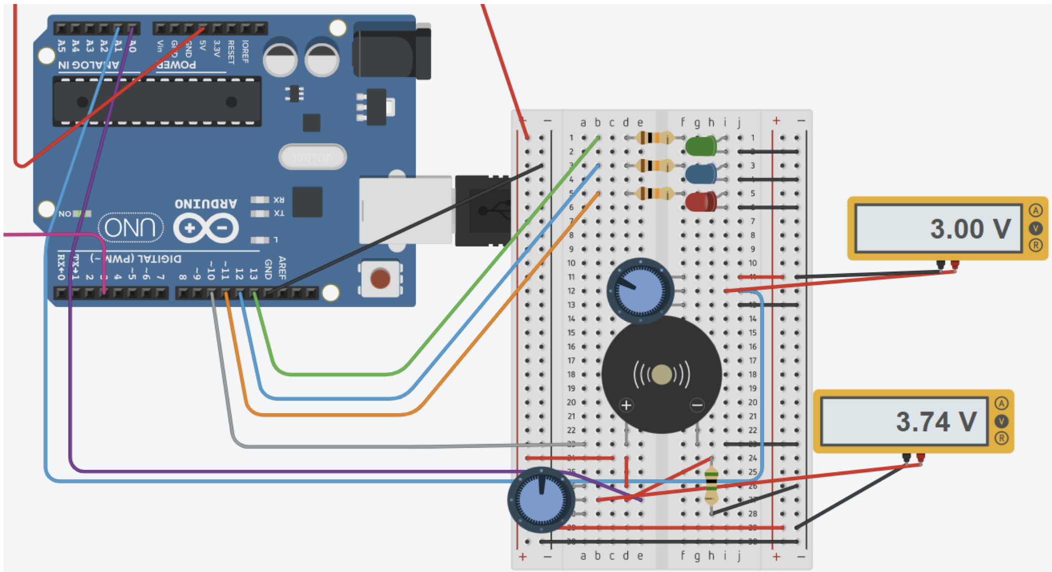

Figure 5. Simulation of user lying

In Figure 5 you will see that the user voltage is greater than the reference voltage by a margin that is greater than 0.5 volts. This indicates that the user is lying at which point the red LED lights up and the buzzer goes off.

4. Additions to initial design

After getting feedback from our first presentation of this project, we decided to add 3 more sensors to our circuit to make it multi-functional. However, we still wanted the sensors we added to be in theme with our lie detector, so we decided to make our circuit a home security system alongside the lie detector. The three sensors we added were a laser detector sensor, vibration sensor, and water sensor.

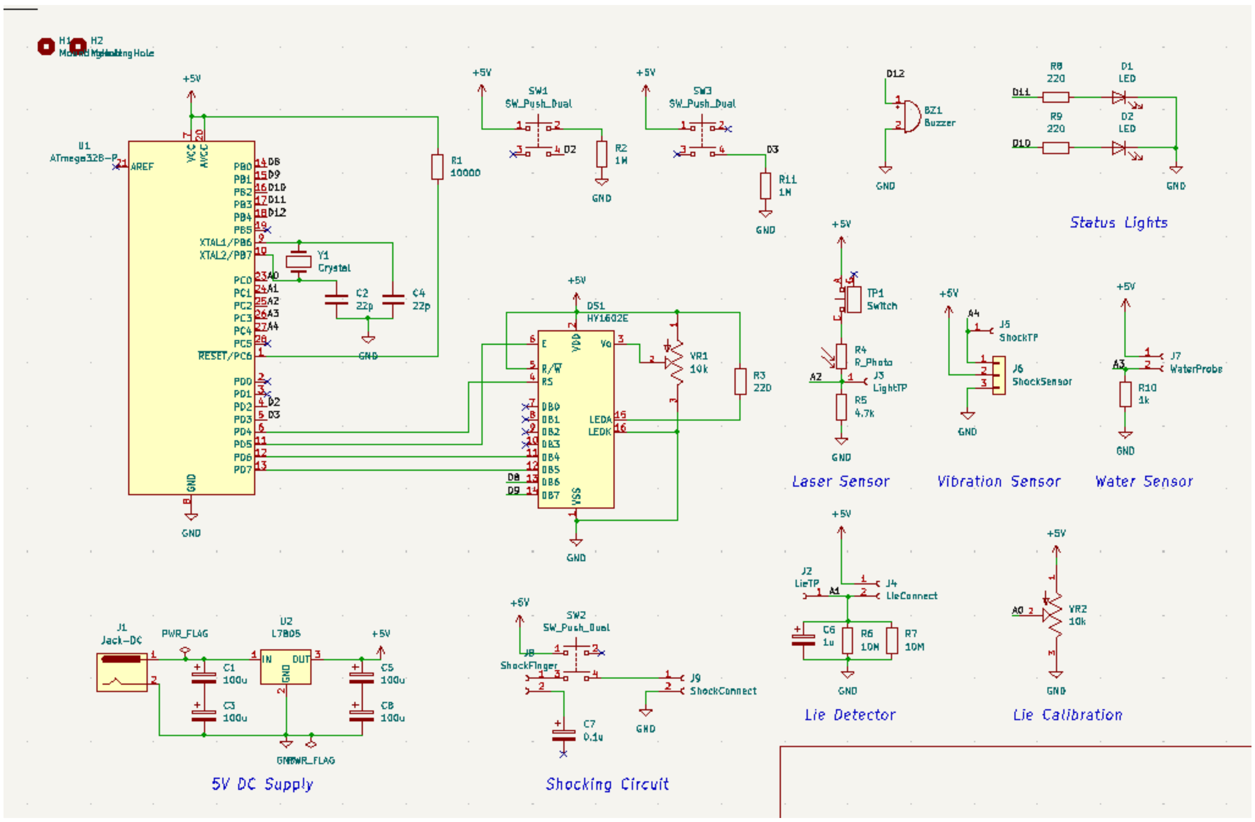

Figure 6. Final Design Schematic

Figure 6 shows the schematic for our final design. It shows how we implemented a lot of the sensors in our circuit and where we added some more functionality. The way we made the laser sensor was with a photoresistor in series with another resistor and the ATMEGA328 analog input is connected to this to detect if the light level drops below a certain threshold to trigger the alarm. The vibration sensor uses an external shock sensor module, so we added a 3 pin connect for the module to fit on our PCB. The water sensor is just two probes that are usually an open circuit, but if the probes come into contact with water, it will short out and we detect this with an analog pin.

A few things we added that were not present in our initial design was a 5V DC Supply which takes in a 12V input from a barrel jack and uses a 5V voltage regulator (L7805) to make a 5V supply. We also added two buttons in our circuit. One button is used to switch the circuit from home security mode to lie detector mode. The other button was added for any other functionality we wanted to add in the future and in the end, we decided to make it activate a calibration mode for all our sensors.

The three LEDs in our initial design that indicated if someone is lying or not was changed to status lights button that shows which mode we are in. We also set these buttons to flash if there is an emergency alert if any of our sensors are triggered or flash if someone is lying. Finally, we added the part of the circuit that shocks you if you're lying. The components we used for this part of the circuit was stripped from a shock gum toy which had a transformer and a DC to AC converter. We also used the metal connects for your finger in the shock gum for our lie detector and shocking circuit.

5. Bread Board Design

One of the advantages we had in our project was that most of our components was already in our lab kit or something we could get from ETG. Apart from the vibration sensor, shocking circuit, and buzzer, we were able to start prototyping and testing our circuit before our other components arrived.

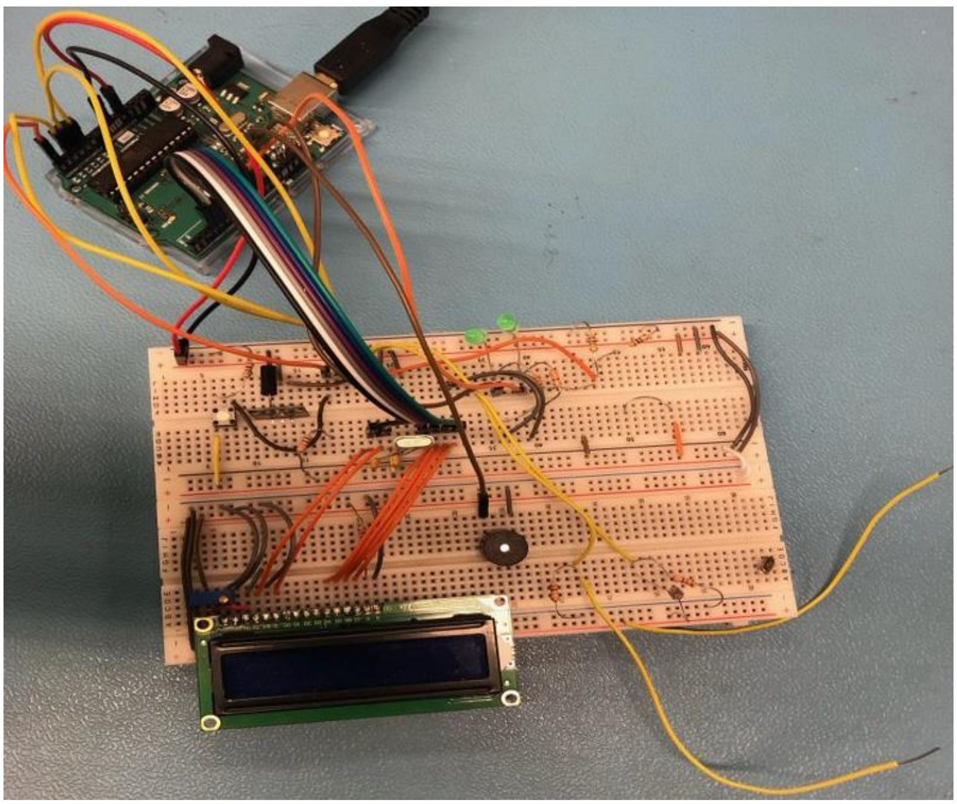

Figure 7. Breadboard Design

Figure 7 shows the breadboard prototype of the circuit and we were mainly testing how to make the ATMEGA328 chip a standalone component outside the Arduino and balancing out our digital and analog pins to see how much functionality we can fit in our circuit. We ended up with 1 more analog and digital pin available in our final design. We did a lot of the testing in TinkerCAD for the sensors and mode change and the prototyping we ran there transitioned smoothly onto the final breadboard, so we focused a lot on making sure our code ran as smoothly as possible and making the user experience better.

6. PCB Schematic

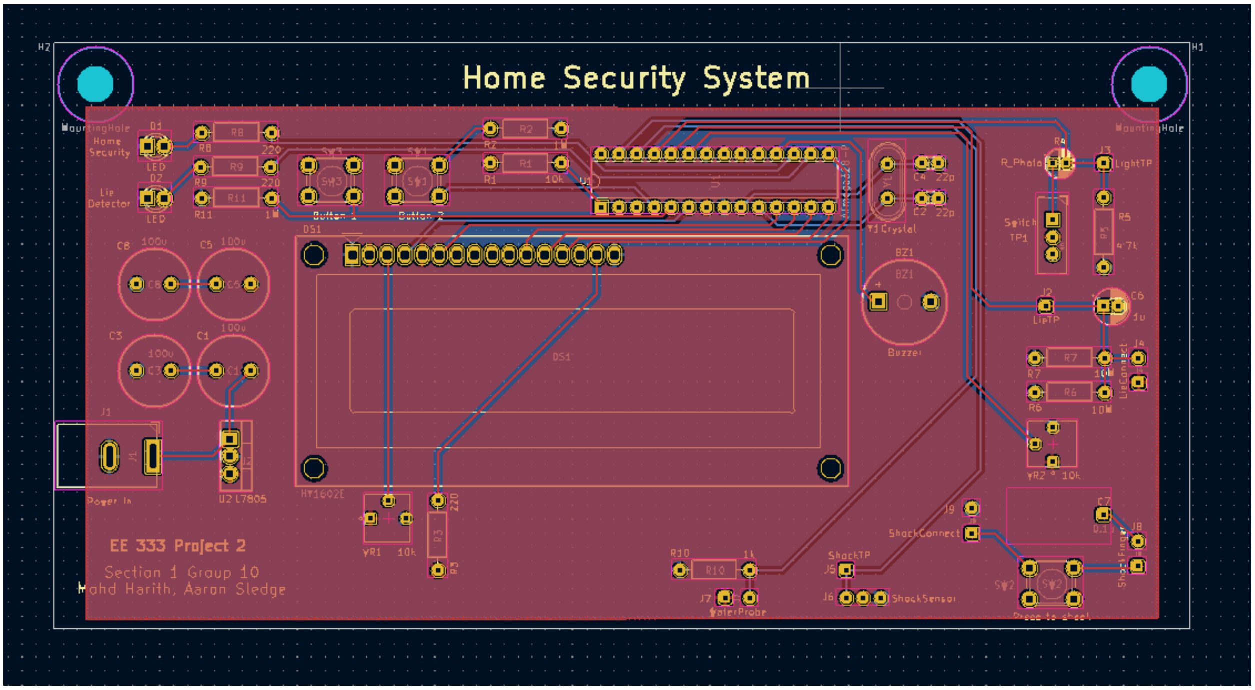

Figure 8. Final PCB Layout

In figure 8 you can see the completed PCB schematic which was designed to be mounted to a wall. In the bottom right is our lie detector circuit and the shocking circuit to shock the user if they are caught lying. To the left of those circuits is our shock sensor as well as our water sensors. We decided to put the water sensor at the bottom of the board so that its leads were close to the ground so that it would be able to detect water before it reaches the PCB. In the top right corner is our laser detection sensor, we put it towards the top and made sure that the photoresistor was facing out so that a laser could be pointed at it at a reasonable height.

The Atmega328 was placed closest to the LCD screen due to LCD occupying such a large number of the Atmega328's pins. That way the traces to the LCD were short and minimized the difficulty with routing the rest of the components on the board. And the other components in our home security device were placed where it made routing the easiest.

Now we did have an issue with routing this board. In the beginning when we attempted to route our board using ground and 5V planes we found that our Atmega 328 wasn't grounded; our traces to the LCD were keeping it from connecting to the ground plane. So we had to go back into Kicad and re-route our Atmega328 to connect to the LCD but also so that the traces were far enough away from the ground pin of the Atmega328 so that it would be able to connect to the ground plane. That taught us that using planes, although good when it comes to boards with a lot of components because it makes the board easier to trace. Can also lead us to believing that all ground and power connections are fine no matter what when in fact that is not the case.

7. PCB Design

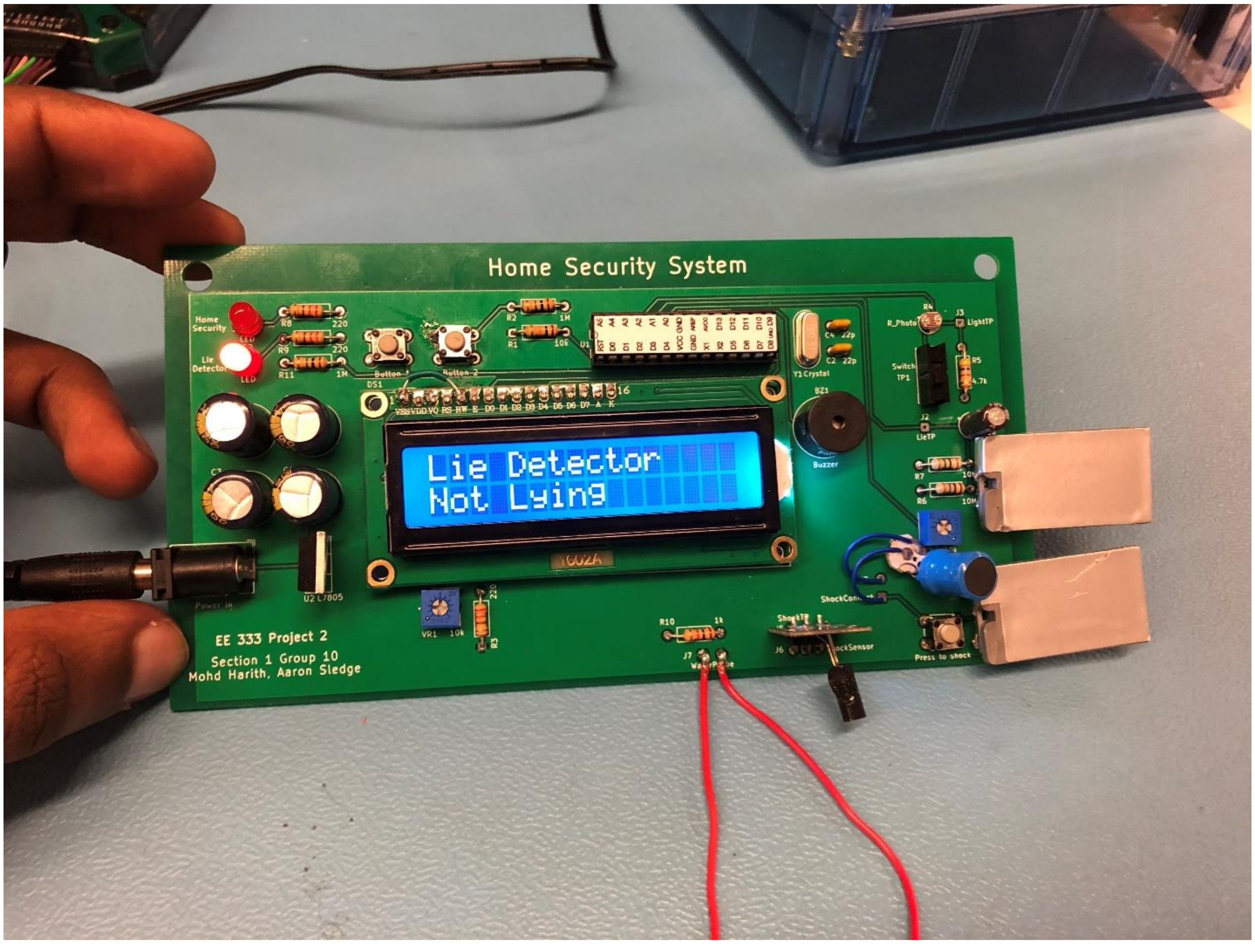

Figure 9. Completed PCB with components soldered to it

Above is a picture of our completed PCB with all of the components soldered to it. In the beginning our LCD screen wouldn't display the messages we coded it to display so we had to trouble shoot. We figured out that it wasn't displaying messages because one of our pins for our LCD was supposed to be connected to ground, but it was in fact connected to the 5V plane instead in our Kicad files. This led to us having to de-solder and cut that pin's connection to the board and solder a wire from that pin to another LCD pin that was connected to ground. If you look closely at the LCD screen you can see that we soldered a green wire across two pins to solve this issue. After that our PCB had no issues with the LCD screen.

We also encountered another issue which was that we laid out two of our push buttons incorrectly. The way we had traced our design led to the Arduino reading the push button as always being pressed, due to us routing to pins that were already shorted together. To fix this we had to de-solder the buttons and bend the pins to fit into the PCB again in a way that allowed for the button to operate how it was designed to operate. After doing so our Arduino only read the buttons as being pressed whenever we pressed the buttons.

Other than that, there were no issues at all with our PCB and it functioned as it was designed to function. Any parts of our circuit that weren't operating as intended weren't being hindered by something to do with our board layout and routing. So overall the PCB turned out fairly good with only a few minor issues that were easily fixed.

8. Conclusion

In the end, most of our circuit ran as expected although we did have a few issues. One of the issues is that we had to remove the vibration sensor from our code because the sensor itself wasn't very reliable and we couldn't get stable readings to make it work. We had around 6 of the vibration sensors to test with and each of them behaved differently. Some of them would just give us an open circuit and never actually output anything on the analog pin showing a constant 0 reading while others would give us very unstable readings that ranged from 100 – 400 on the analog pin of the Arduino.

Another issue we found was that sometimes our button would activate and change the mode even though no one pressed the button. We suspect that this has to do with how most of our sensors are in parallel with each other and with the button, so voltage changes in the sensors might actually cause voltage changes in the button and trigger the interrupts we used in our button pin. When we print out the readings of the button onto the serial monitor, most of the time our readings would be correct showing 0 when the button is not pressed, but it will randomly spike up to a reading of 800 in an analog reading from time to time which is enough to trigger an interrupt and change the mode. Adding switches to sensors that are otherwise always on such as the laser sensor has reduced the number of times this happens, but future iterations of this design might want to include a transistor to automatically switch the sensors on and off depending on which mode we're in.

The last major issue with this circuit is that our lie detector circuit is pretty unreliable. It gives stable readings if the user is relaxed, but any sudden movements or changes such as breathing in deeply would cause drops in the readings and trigger that someone is lying. If we want to improve this in the future, we will have to make the system smarter and add in more factors such as someone's heartbeat so it doesn't have to only rely on the galvanic skin response which can change spontaneously. Because of this, we decided that our shocking circuit should be manually triggered instead of automatically by the ATMEGA328 chip since the system wasn't reliable enough to tell if someone is lying or not.

9. Bill of Materials

Part Number	Part Description	Link	Quantity	Cost	Sourced
107KXM050M	100uF Capacitor	107KXM050M Cornell Dubilier / Illinois Capacitor | Capacitors | DigiKey	4	$1.88	Yes
C317C220J1G5TA	22pF Capacitor	C317C220J1G5TA KEMET | Capacitors | DigiKey	2	$0.86	Yes
UPS1H010MDD1TD	1uF Capacitor	UPS1H010MDD1TD Nichicon | Capacitors | DigiKey	1	$0.29	Yes
C317C104M5U5TA7301	0.1uF Capacitor	C317C104M5U5TA7301 KEMET | Capacitors | DigiKey	1	$0.34	Yes
294-220-RC	220 Ohm Resistor	294-220-RC Xicon | Resistors | DigiKey Marketplace	3	$0.45	Yes
CFR-50JT-52-1M	1M Ohm Resistor	CFR-50JT-52-1M YAGEO | Resistors | DigiKey	2	$0.22	Yes
CFR-50JB-52-10M	10M Ohm Resistor	CFR-50JB-52-10M YAGEO | Resistors | DigiKey	2	$0.26	Yes
CFR100J10K	10k Ohm Resistor	CFR100J10K TE Connectivity Passive Product | Resistors | DigiKey	1	$0.34	Yes
294-4.7K-RC	4.7k Ohm Resistor	294-4.7K-RC Xicon | Resistors | DigiKey Marketplace	1	$0.15	Yes
CFR200J1K0	1k Ohm Resistor	CFR200J1K0 TE Connectivity Passive Product | Resistors | DigiKey	1	$0.47	Yes
PDV-P9200	10k - 50k Photoresistor	PDV-P9200 Advanced Photonix | Sensors, Transducers | DigiKey	1	$1.96	Yes
3362P-1-103	10k Potentiometer	3362P-1-103 Bourns Inc. | Potentiometers, Variable Resistors | DigiKey	2	$3.94	Yes
L513SRD-B	Red LED	L513SRD-B American Opto Plus LED | Optoelectronics | DigiKey Marketplace	2	$0.36	Yes
71600	ATMEGA328P	71600 Microchip Technology | Integrated Circuits (ICs) | DigiKey Marketplace	1	$7.50	Yes
L7805CV	5V Voltage Regulator	L7805CV STMicroelectronics | Integrated Circuits (ICs) | DigiKey	1	$0.69	Yes
FC4STCBMF16.0	16MHz Crystal Oscillator	FC4STCBMF16.0 Fox Electronics | Crystals, Oscillators, Resonators | DigiKey	1	$0.34	Yes
FSMH	Pushbutton Switch	FSMH TE Connectivity ALCOSWITCH Switches | Switches | DigiKey	3	$0.84	Yes
CMI-1295-0585T	Piezo Buzzer	CMI-1295-0585T CUI Devices | Audio Products | DigiKey	1	$1.18	Yes
MD21605G12W3-BNMLW-VE	16x2 LCD Display	MD21605G12W3-BNMLW-VE Midas Displays | Optoelectronics | DigiKey Marketplace	1	$4.77	Yes
EG1218	Slide Switch	EG1218 E-Switch | Switches | DigiKey	1	$0.76	Yes
RAPC742X	12V Barrel Jack	RAPC742X Switchcraft Inc. | Connectors, Interconnects | DigiKey	1	$1.56	Yes
KY-002	Shock Sensor	uxcell KY-002 SW-18015P Shock Vibration Switch Sensor Module for Arduino 2pcs: Amazon.com: Industrial & Scientific	1	$3.25	Yes
Shocking Gum	DC-AC Converter and Transformer	Amazon.com: 3 Pack of Shocking Gums - Funny Shock Gag : Toys & Games	1	$2.00	Yes
			Total Cost	$34.41	

10. Arduino Sketch

```
#include <LiquidCrystal.h>
const int rs = 4, en = 5, d4 = 6, d5 = 7, d6 = 8, d7 = 9; //LCD Screen Pins

const byte statusPin1 = 10, statusPin0 = 11, buzzerPin = 12; //LED and Buzzer Pins
const byte buttonPin0 = 2, buttonPin1 = 3; //Button Pins
const byte potPin = 0, fingerPin = 1, lightPin = 2, waterPin = 3, shockPin = 4; //Sensor Pins

volatile byte state = LOW; //Button States
int band = 50; //adjust value for sensitivity, aka buffer zone to tell difference between truth and a lie

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  //LCD Setup
  lcd.begin(16, 2);
  lcd.clear();

  //LED Pins Setup
  pinMode(statusPin0, OUTPUT);
  pinMode(statusPin1, OUTPUT);

  //Button Interrupt Setup
  pinMode(buttonPin0, INPUT);
  attachInterrupt(digitalPinToInterrupt(buttonPin0), modeChange, RISING);

  //Serial.begin(9600); //Used for Testing Pin Readings
}

void loop() {

  modeSetup();

  if (state == LOW){
    sensorDetect();
  }else{
    lieDetector();
  }

  delay(200);
}

//Interrupt function for changing modes
```

```
void modeChange() {
  state = !state;
}

//Sets up LCD Screen and LED for the two modes (Home Security and Lie Detector)
void modeSetup() {
  lcd.clear();
  if (state == LOW){
  lcd.print("Home Security");
  digitalWrite(statusPin0, HIGH);
  digitalWrite(statusPin1, LOW);
  }else{
  lcd.print("Lie Detector");
  digitalWrite(statusPin0, LOW);
  digitalWrite(statusPin1, HIGH);
  }
}

//Function while in Home Security Mode
//Reads the analog inputs from the three sensors and trigger an alarm if they go past a certain threshold
void sensorDetect() {
  lcd.setCursor(0, 1);
  int waterValue = analogRead(waterPin);
  int lightValue = analogRead(lightPin);
  int shockValue = analogRead(shockPin);

  if (waterValue > 15){ //Detects if the water probes short circuit
    lcd.print("Water Alert     ");
    emergency(1);
  }else if ((lightValue < 700 )&(lightValue > 50)){ //Detects if the light level drops below a certain 
threshold and that the sensor is not turned off
    lcd.print("Intruder Alert  ");
    emergency(1);
  }//else if ((shockValue > 50 )){ //Detects if there is any sudden vibrations (DISABLED BECAUSE READINGS 
ARE UNRELIABLE)
    //lcd.print("Earthquake Alert    ");
    //emergency(2);
  //}
  else //If none of the sensors trigger
    lcd.print("Home is Safe    ");

  //Serial.println(waterValue+"\t"+shockValue); //Used for threshold readings
}

//Emergency Function if sensors are triggered
```

```
//Sets the LED to flash and buzzer to sound periodically
void emergency (int type)
{
  switch (type){
    case 1: //Emergency case for Water and Light sensors. Runs continuosly while sensors are triggered
      digitalWrite(statusPin0, HIGH);
      digitalWrite(statusPin1, HIGH);

      tone(buzzerPin, 800, 500);
      delay(500);
      noTone(buzzerPin);

      digitalWrite(statusPin0, LOW);
      digitalWrite(statusPin1, LOW);

      delay(500);
      digitalWrite(statusPin0, HIGH);
      digitalWrite(statusPin1, HIGH);
      break;
    case 2: //Emergency Case for vibration sensor. Runs 5 times before turning off
      for (int i = 0; i < 5; i++){
        digitalWrite(statusPin0, HIGH);
        digitalWrite(statusPin1, HIGH);

        tone(buzzerPin, 800, 500);
        delay(500);
        noTone(buzzerPin);

        digitalWrite(statusPin0, LOW);
        digitalWrite(statusPin1, LOW);

        delay(500);
        digitalWrite(statusPin0, HIGH);
        digitalWrite(statusPin1, HIGH);
      }
      break;
    default:
     break;
  }

  //Runs final tone before turning off the LEDs to display the correct mode again
  tone(buzzerPin, 800, 500);
  delay(500);
  noTone(buzzerPin);
```

```
  digitalWrite(statusPin0, LOW);
  digitalWrite(statusPin1, LOW);

  delay(500);
}

//Function while in Lie Detector Mode
//Takes reading from skin resistance and calibration potentiometer to detect if someone is lying
void lieDetector(){
  lcd.setCursor(0, 1);
  int VoltageDivider = analogRead(fingerPin);
  int pot = analogRead(potPin);

  if (VoltageDivider > pot + band)//if voltage divider is greater that means that users skin resistance
has dropped and is lying
  {
    lcd.print("Lying               ");
    beep();
  }
  else if(VoltageDivider < pot - band) //If voltage devider is less that means that persons skin
resistance is high and is telling the truth
  {
    lcd.print("Not Lying           ");
  }
  else //There is a buffer zone to account for natural fluctuations in skin resistance that arent drastic
such as when you lie
  {
    lcd.print("Maybe               ");
  }
  delayMicroseconds(500);
}

//Makes the buzzer beep
void beep()
{
  for (int i = 0; i < 1000; i++)
  {
  digitalWrite(buzzerPin, 1);
    delayMicroseconds(500);
    digitalWrite(buzzerPin, 0);
     delayMicroseconds(500);
   }
}
```