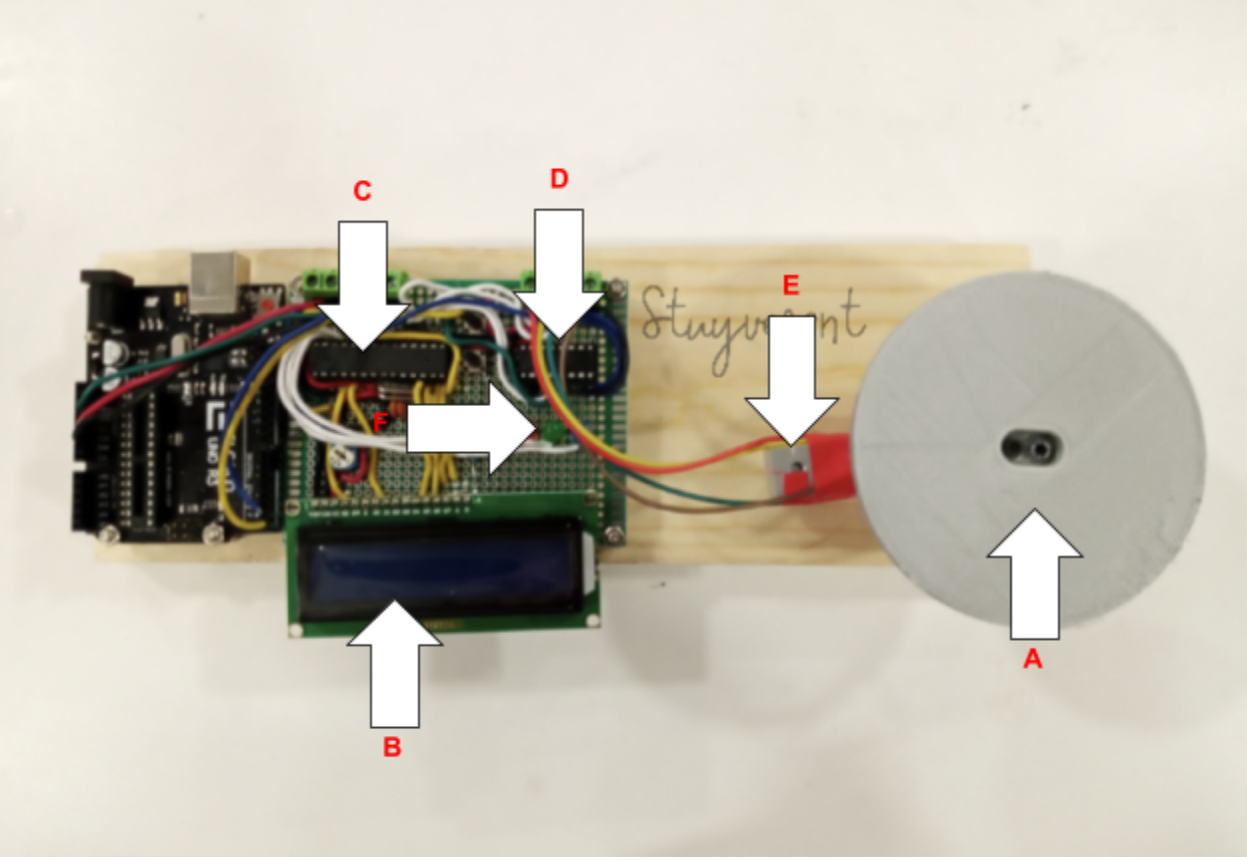
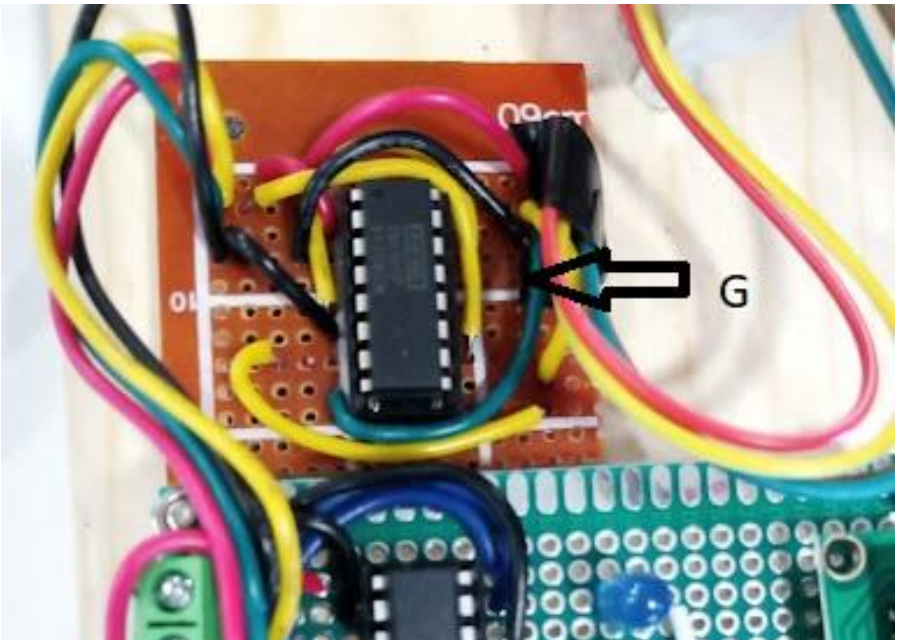
Design Log for Detector Building

Stuyvesant High School
Team C35



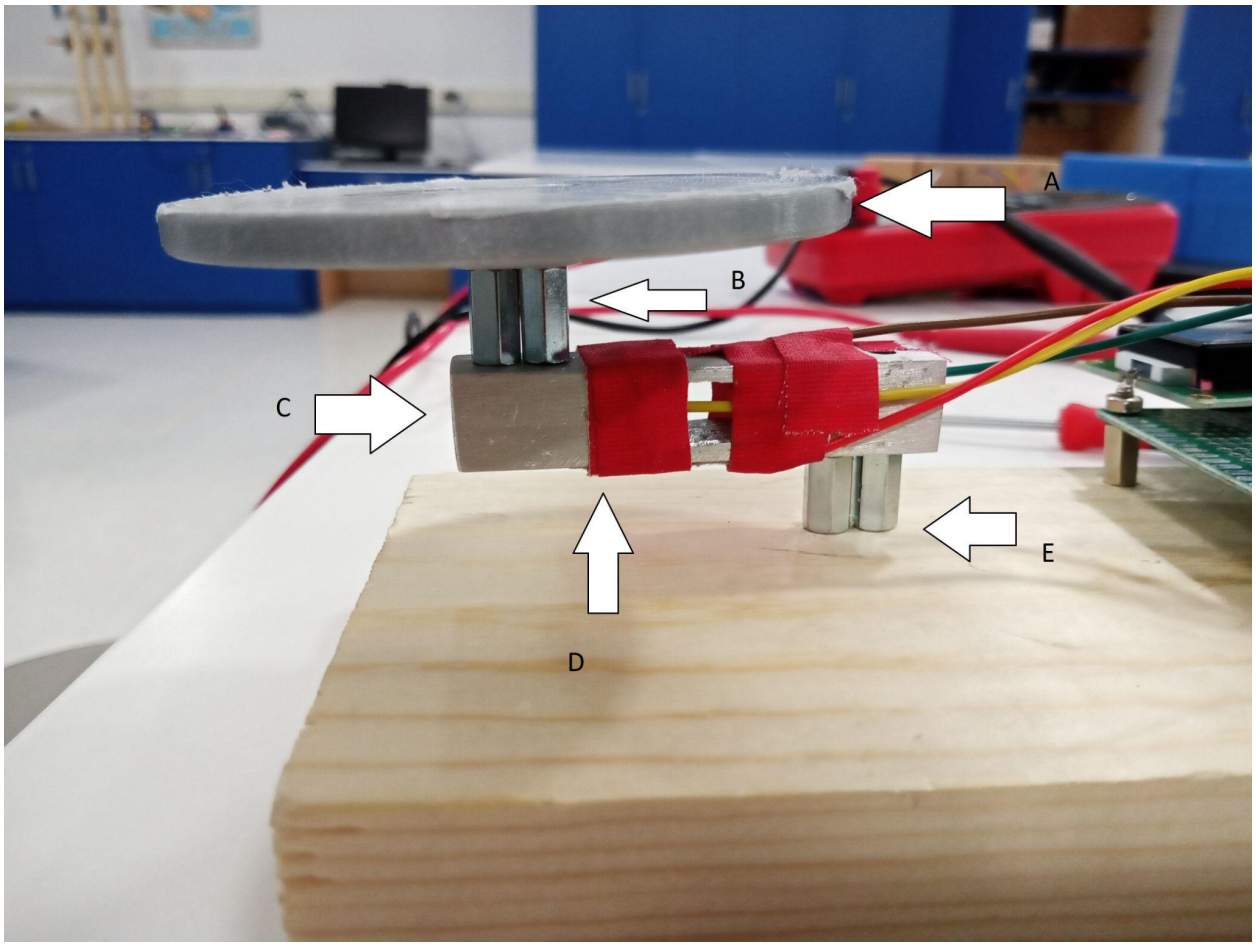


Key:

A) 3d printed platform for unknown mass

B) HD44780 LCD - 16x02 LCD display used to display data.
C) Atmega328p microcontroller (Arduino)
D) MAX1416 16-bit ADC - Converts analog signals to digital signals to get voltage
E) Aluminium block with strain gauges
F) Red, Green and Blue LEDS
G) INA125P - instrumentation amplifier

The circuit was first prototyped on a breadboard, and then soldered to a perfboard, which provided more consistent readings from the sensor. PCB mounts were used for easy connections to our strain gauges and to our Arduino. We used a MAX1416 ADC to get the voltage reading from our strain gauges and used the internal amplifier to give it a more noticeable difference when pressure was being applied. We then used M3 hex mounts to attach the build onto a wooden block to keep everything in place.



FORCE/MASS SENSOR:

A) 3D printed disc to hold the unknown mass
B) Standoff
C) Drill Press Cut aluminum bar using a Baileigh DP-4016B - 16" drill press and a hacksaw
D) Strain Gauges, wrapped under tape
E) Measurement mechanism

½ by ½ inch aluminium bar stock was cut to size, and using a hand drill, the slot in the middle was made. The mounting holes for the standoffs were also drilled by hand and tapped to fit the screws.
Strain gauges were glued to the top and bottom of the aluminum block and were wired together in a wheatstone bridge configuration. In order to protect the wires, red tape was wrapped around the strain gauges and wires.

We designed our 3d printed disc platform in a CAD software called onshape and a slicer called Cura to print it on our school's Ultimaker S3s with PLA filament.

Printer: Ultimaker S3
Material: PLA
CAD programs: Onshape to design, and Cura to slice it.
Exported File Type: STL used to model components needed for the final build.
Design: Used to act as the plate for weights.

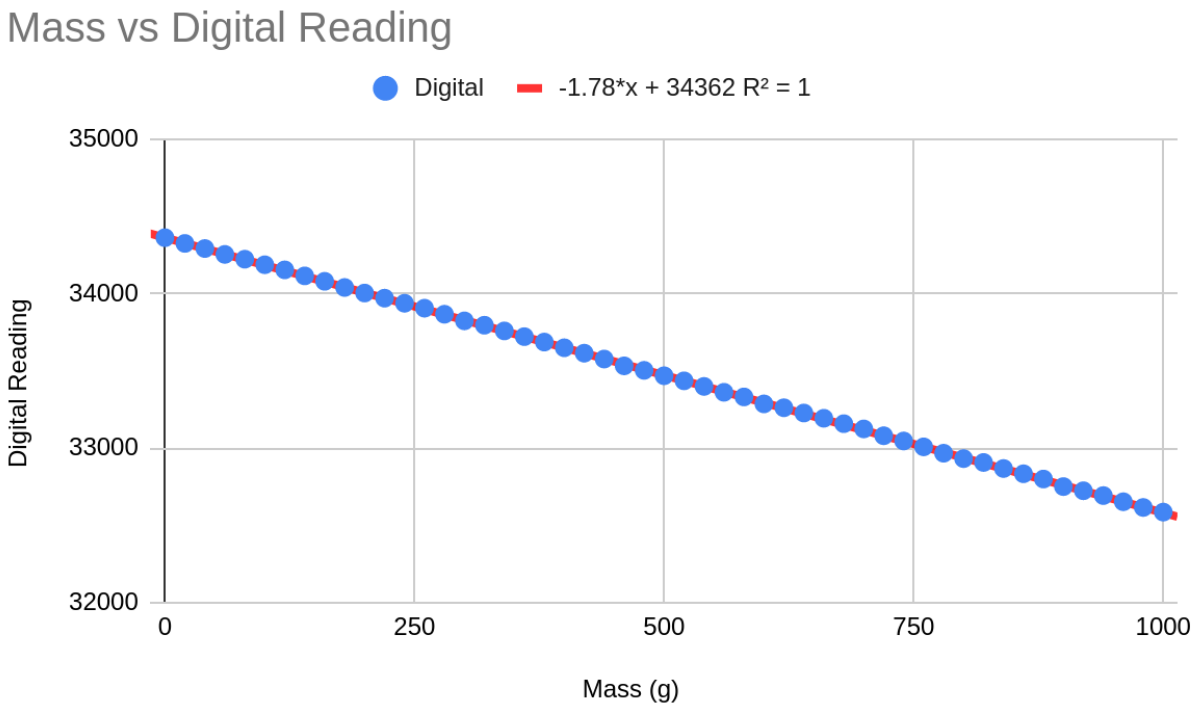
DATA

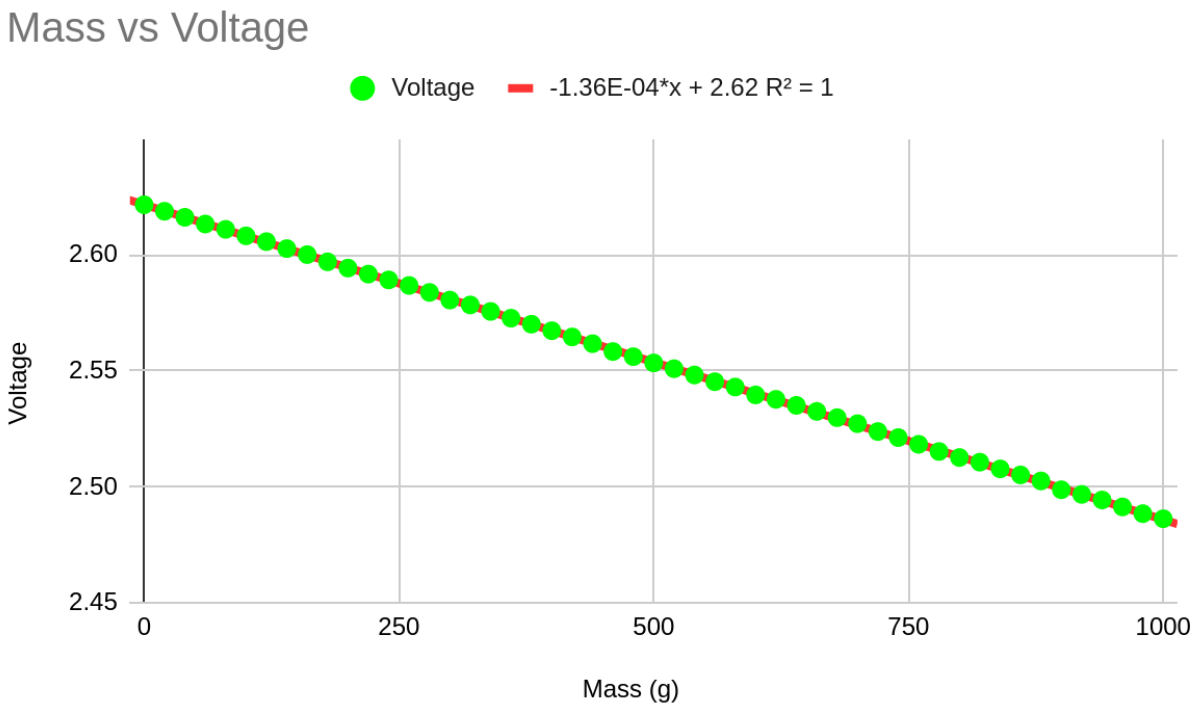
mass in grams	digital	voltage
0	34361	2.6216
20	34324	2.6188
40	34291	2.6162
60	34253	2.6133
80	34222	2.611
100	34186	2.6082
120	34153	2.6057
140	34114	2.6027
160	34079	2.6001
180	34039	2.597
200	34003	2.5943
220	33970	2.5917
240	33937	2.5892
260	33905	2.5868
280	33866	2.5838
300	33823	2.5805
320	33795	2.5784
340	33758	2.5756
360	33721	2.5727
380	33686	2.5701
400	33649	2.5673
420	33614	2.5646
440	33576	2.5617
460	33532	2.5583
480	33503	2.5561
500	33468	2.5534
520	33435	2.5509
540	33399	2.5482
560	33361	2.5453
580	33331	2.543
600	33286	2.5396
620	33261	2.5377

640	33227	2.5351
660	33193	2.5325
680	33158	2.5298
700	33124	2.5272
720	33080	2.5238
740	33046	2.5212
760	33008	2.5183
780	32967	2.5152
800	32932	2.5126
820	32907	2.5106
840	32868	2.5077
860	32834	2.5051
880	32800	2.5025
900	32751	2.4987
920	32724	2.4967
940	32693	2.4943
960	32653	2.4913
980	32616	2.4884
1000	32586	2.4862

For each weight measurement, 2 trials were taken and averaged out. Digital is the raw digital values from our ADC, and Analog is the converted digital value into an analog one. Analog value = Digital value * (5/65535). Each data measurement is in itself the average value of 3000 individual measurements from the sensor.

SCATTER PLOTS

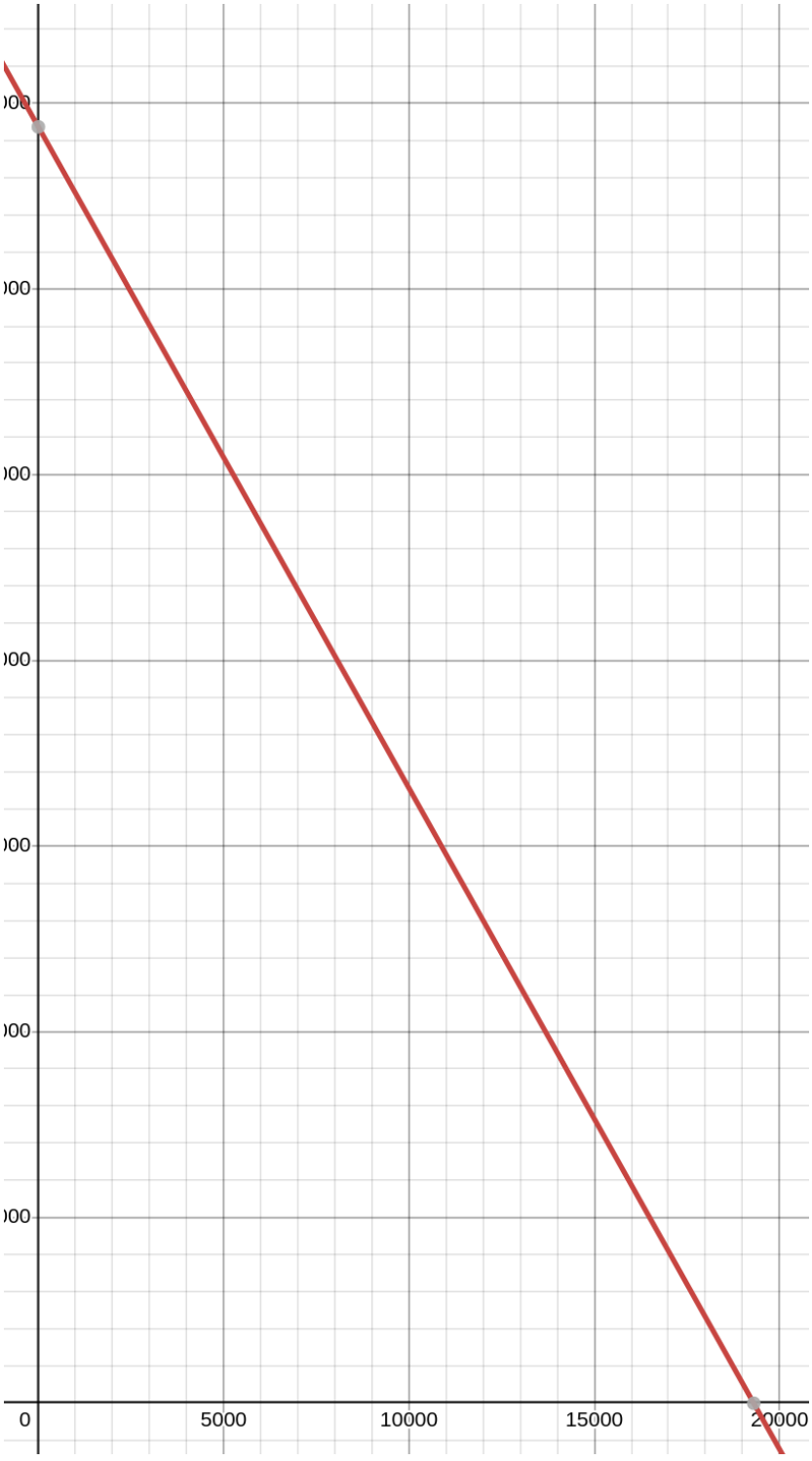




Mathematical model:

-1.78x + 34362

Graph:



Code:

```
#include "SPI.h"
#include "LiquidCrystal.h"



// Set weight Ranges in grams, R1 <= R2*/
const double R1 = 50;
const double R2 = 500;



//Amount of samples for the average
#define N_AVG 250



//LEDS use analog pins
#define RED A3
#define GRN A4
#define BLU A5



//Buttons
#define B1 A1
#define B2 A2

#define DRDY A0
const int ADCRST = 9, SPICLOCK = 13; //MAX1416 PINS
const int RS = 2, E = 3, D4 = 5, D5 = 6, D6 = 7, D7 = 8; //LCD PINS
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
const int ss=10;
unsigned int adcValue, offset = 0;
```

```
void setup()
{
 pinMode(B1, INPUT);
 pinMode(B2, INPUT);
 pinMode(RED, OUTPUT);
 pinMode(GRN, OUTPUT);
 pinMode(BLU, OUTPUT);
 digitalWrite(RED, LOW);
 digitalWrite(GRN, LOW);
 digitalWrite(BLU, LOW);
 pinMode(DRDY, INPUT);
 pinMode(ADCRST, OUTPUT);
 pinMode(ss, OUTPUT);
 lcd.begin(16, 2);
 lcd.print("INIT...");
 digitalWrite(ss,HIGH);
 SPI.begin();
 SPI.setBitOrder(MSBFIRST);
 SPI.setDataMode(SPI_MODE3);
 SPI.setClockDivider(SPI_CLOCK_DIV16);
 digitalWrite(SPICLOCK,HIGH);
 digitalWrite(ADCRST,HIGH);
 delay(1000);
 digitalWrite(ADCRST,LOW);
 delay(1000);
 digitalWrite(ADCRST,HIGH);
 delay(1000);
 MAX1416_Config();
 delay(100);
 MAX1416_ReadSetupReg();
```

```
 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("READY!");
 delay(100);
}



void MAX1416_SerialInit()//You can modify it for handle channels
{
 //series of commandbit
 digitalWrite(ss,LOW); // Enable ADC SPI

 SPI.transfer(0xFF);
 SPI.transfer(0xFF);
 SPI.transfer(0xFF);
 SPI.transfer(0xFF);



 digitalWrite(ss,HIGH); // Disable ADC SPI
}

void MAX1416_Config()//You can modify it for handle channels
{
 //series of commandbit
 digitalWrite(ss,LOW); // Enable ADC SPI

 //Write OP
 SPI.transfer(0x20);//command for comm reg to select ch1 and write to clock register
 delay(100);
 SPI.transfer(0xA7);//command for clock reg to set 2,4576Mhz
 //End Write OP
 delay(100);
```

```
 //Write OP
 SPI.transfer(0x10);//command for comm reg to write setup register
 delay(100);
 SPI.transfer(B01100000);//command for setup reg to self calibration,unipolar,unbuffered,
 //End Write OP



 digitalWrite(ss,HIGH); // Disable ADC SPI
}

void MAX1416_WaitForData_Soft()
{
        char DataNotReady = 0x80;

        digitalWrite(ss,LOW); // Enable ADC SPI

        while(DataNotReady) // wait for end of conversion
        {
        // Read OP
        SPI.transfer(0x08);//command for comm reg to read  (dec   8)
        DataNotReady =SPI.transfer(0x00); // Read comm register
        // End Read OP
        Serial.println(DataNotReady,BIN);
        DataNotReady &= 0x80;
        }

        digitalWrite(ss,HIGH); // Disable ADC SPI
}


void MAX1416_WaitForData_Hard()
```

```
{
        char DataNotReady = 1;
        char value;



        while(DataNotReady) // wait for end of conversion
        {
        // Read OP
        value = digitalRead(DRDY); // Read comm register
        if (value == LOW)
        DataNotReady = 0;
        else
        DataNotReady = 1;
        // End Read OP
        //Serial.println("NOT READY");
        }
}



byte MAX1416_ReadSetupReg() //You can modify it to read other channels
{



        byte myByte;


        digitalWrite(ss,LOW); // Enable ADC SPI


        // READ Data OPERATION
        SPI.transfer(0x18);//command for the comm to read register register 00011000
        //read 8bit of data
        myByte = SPI.transfer(0x00);
        // End Read Data
```

```
        Serial.print(myByte,BIN);
        //delay(2000);
        digitalWrite(ss,HIGH); // Disable ADC SPI



        return myByte;
}

unsigned int MAX1416_ReadCH0Data() //You can modify it to read other channels
{
        unsigned int uiData;
        byte highByte;
        byte lowByte;

        digitalWrite(ss,LOW); // Enable ADC SPI

        // READ Data OPERATION
        SPI.transfer(0x38);//command for the comm to read data register for channel 1 (dec  56)
        //read 16bit of data ADC
        highByte = SPI.transfer(0x00);
        lowByte = SPI.transfer(0x00);
        // End Read Data

        digitalWrite(ss,HIGH); // Disable ADC SPI



        uiData = highByte;
        uiData <<= 8;
        uiData |= lowByte;

        return uiData;
}
```

```
double readavg(int n) {
  double a = 0;
  for (int i = 0; i < n; i++) {
              MAX1416_WaitForData_Hard();
              a += MAX1416_ReadCH0Data();
  }
  return a/n;
}



void tare() {
  offset = 34361 - readavg(N_AVG); //Voltage decreases with load applied
}



//Inverse of the model
double gx1(double x) {
 return (x - 34362)/(-1.78);
}



void loop()
{
  double mass = 0;
  double voltage;
  if (digitalRead(B1) == HIGH) {
              lcd.clear();
              lcd.print("READING...");
              adcValue = readavg(N_AVG) + offset;
```

```
        voltage = double(adcValue)*5/65535;
        mass = gx1(adcValue);
        lcd.clear();
        lcd.print(mass, 4);
        lcd.print("g");
        lcd.setCursor(0, 1);
        lcd.print(voltage, 4);
        lcd.print(" v");
        //LED Mass range code:
        if (mass < R1) {
        digitalWrite(RED, HIGH);
        digitalWrite(GRN, LOW);
        digitalWrite(BLU, LOW);
        }
        if ((mass > R1) && (mass < R2)){
        digitalWrite(RED, LOW);
        digitalWrite(GRN, HIGH);
        digitalWrite(BLU, LOW);
        }
        if (mass > R2) {
        digitalWrite(RED, LOW);
        digitalWrite(GRN, LOW);
        digitalWrite(BLU, HIGH);
        }
 }
 if (digitalRead(B2) == HIGH) {
        lcd.clear();
        lcd.print("TARE...");
        tare();
        lcd.clear();
        lcd.print("TARE COMPLETE");
 }
```

```
  digitalWrite(ss,HIGH); // Disable ADC SPI
}
```