Deliverable H - Prototype III and Customer Feedback

GNG 1103
Introduction to Engineering Design

To:
Professor David Knox

By: Group 03 - AllTheyDoIsWin
Alec Plourde
Ashley Garofalo
Gabrielle Graceffa
Thomas Johnson Band
Emmett van den Broek

Due Date: November 27th, 2022
Submission Date: November 25th, 2022

University of Ottawa
Faculty of Engineering

Table of Content

Wrike Link: https://www.wrike.com/open.htm?id=969517657

1.0 - Prototyping Test Plan - Part 3

Test ID #3: Testing the volumetric flow sensor

Goal: The goal of this test was to ensure that the volumetric flow sensor could read the flow value of water in ml/s and display the results in the serial monitor.

Process: The sensor was wired to the Arduino breadboard and the test was performed using a kitchen sink to create the water flow (see Figure 1.1)

Code used: The practice code used to ensure the volumetric flow sensor would run can be seen below:

Results: The sensor was able to read the value of water flowing through it while simultaneously printing it onto the serial monitor (see Figure 1.2)

Volumetric flow sensor code:

```
//attachInterrupt(interrupt number, the function you would like to run
when triggered, what you would like to set as trigger) this is how you can
interpret this function
// I am using interrupt 0 to trigger "Flow" function when a pin changes
low ot high = pulse arrives from sensor
int PinFlow = 2; // This is the flow meter input pin on the arduino

double VolFlow; // this is the volumetric flow we want to find
volatile int sum; // intger is set as volatile for correct updates during
interrupt = important as it will store #of pulses each second

void setup() {

  pinMode(PinFlow, INPUT); // sets the Flow pin from arduino as an input,
pin needs to be set as an input before it can interrupt
  attachInterrupt(0, Flow, RISING); // makes the interrupt 0 (pin #2 on
arduino mega) to run the function "Flow" so the increment function
  Serial.begin(9600);

}
```

```
void loop() {

  sum = 0; // resets the counter so we start counting from 0 again since loops runs over and over again
  interrupts(); // allows interups on the arduino to run, so start counting # of pules from sensor
  delay(1000); // delay reading for 1 second
  noInterrupts(); // Disable the interrupts on the arduino so stop counting pulses

  // Now we must do the math to find the volumetric flow since we know approx 2.25mL fluid/pulse (everytine it rotates)
  VolFlow = (sum*2.25); // #pulse in a second by the amount of 2.25mL of fluid
  VolFlow = VolFlow / 1000; // convert to L/s
  VolFlow = VolFlow / 1000; //convert L/s to m^3/s

  Serial.println(VolFlow);
}

void Flow() { // new function created that the interrupt will run every time a pulse is found

  sum++; // everytime this function is called an increment, ++ will ad 1 to that variable everytime the program runs

}
```

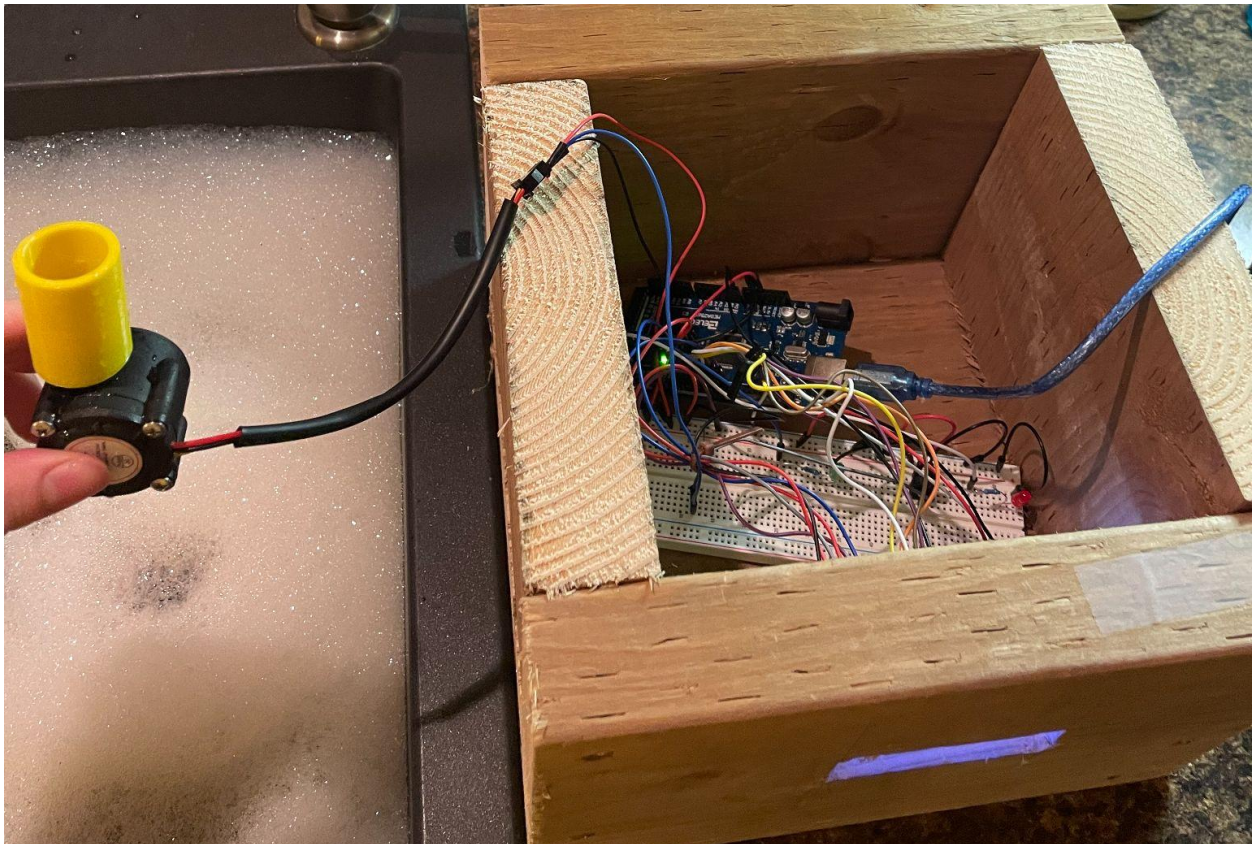

Figure 1.1 - Flow Meter Testing Layout

```
0.00
2.30
3.11
2.97
3.24
3.38
```

Figure 1.2 - Flow meter reading results during testing in m^3/s

Test ID #8: Finishing the 3D printing

Goal: The goal of this test was to build the 3D model to begin the testing phase of the design

Process: The printing process was completed in the maker space on November 16th. The assembly process for testing was done on November 19th.

Results: The 3D printing process was successfully completed with the same blockage being present in the pressure tap designs (see figure 2.1). A few errors were present during the building phase which will be explained in the errors section. The built for parts 2,3, and the orifice plate can be seen in figure 2.2

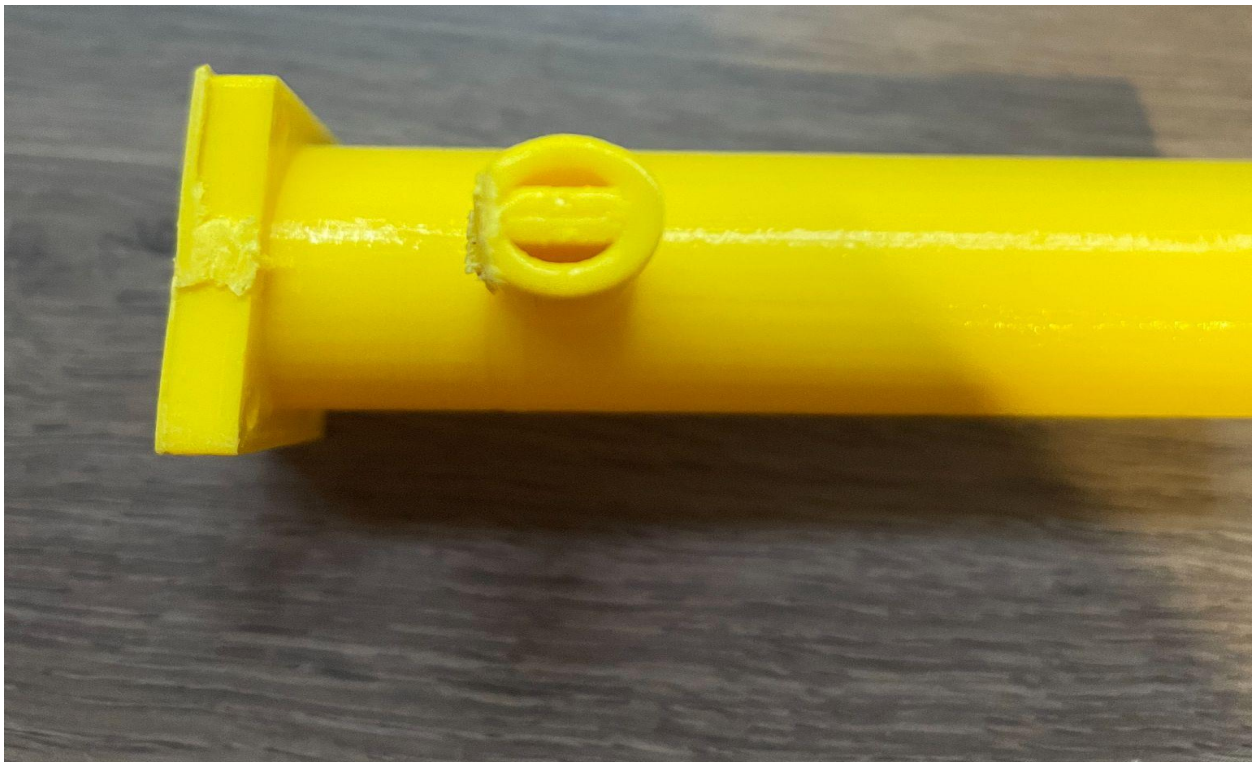

Figure 2.1 - New print for part 2 with the same pressure tap blockage

Figure 2.2 - 3D print assembly for parts 2,3 and the orifice plate

Test ID #9,10,11: Final build and testing of design

Goal: The goal of this test was to finish the building process while ensuring the physical device can be properly wired and connected.

Process: The finishing touches to the design building process were completed on November 19th. The testing phase of the specific gravity measurement occurred on the same day and the testing occurred using a garden hose to simulate the water in-line system (see figure 3.1).

Results: The design was able to successfully read an SG value of the water. However, due to multiple errors that occurred, the value was approximately 0.68 before starting to fluctuate frequently in ranges that did not make theoretical sense. This SG value reading should have theoretically been 1 as the fluid used for testing was the reference fluid in the specific gravity equation. The average error percentage observed was approximately:

$$\%error = \left|\frac{0.68-1}{1}\right| * 100 = 32\%$$

The results were printed into the Serial Monitor and can be seen below in figure 3.2

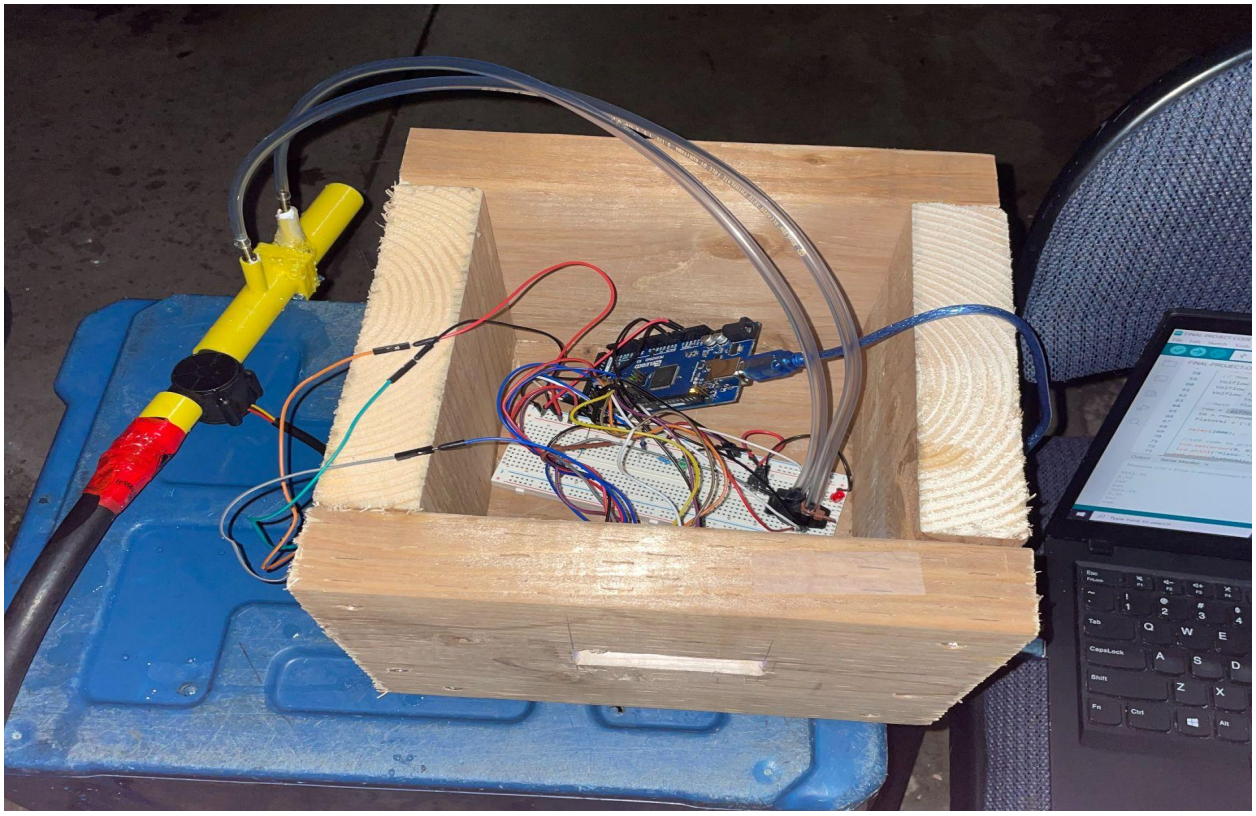

Figure 3.1: Final Testing Layout

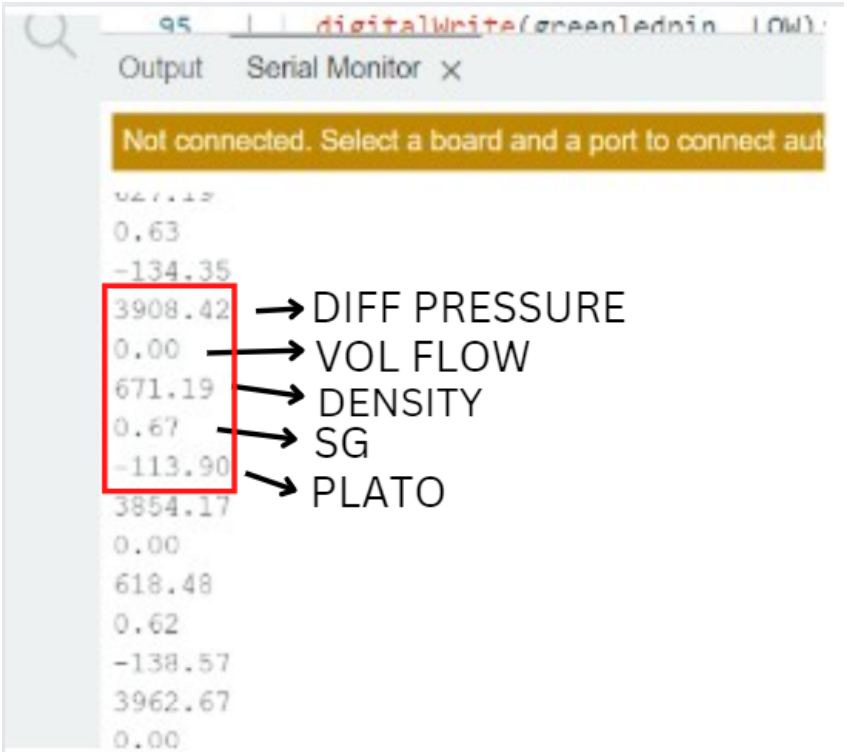

Figure 3.2 - Results on Serial Monitor for SG values

2.0 - Errors and Challenges

During the testing and building phase, there were, unfortunately, multiple errors that arose, leading to inaccuracy in the results. The first error was during the assembly process of all 3D parts. For parts 2 and 3, careful and precise drilling was required to open up the blockage/supports created during the 3D printed process in the pressure taps. However, during the drilling process, one of the pressure taps cracked, causing a major hole in the physical device (see figure 4.1). This hole was a major concern and disappointment due to the high chance of the taps no longer being waterproof/air sealed tight. In an attempt to fix this mistake, waterproof tape and epoxy resin was heavily added to all areas of the crack to try & create a sustainable seal. The next error occurred during the attempted seal of the orifice plate and the 2 plated areas for parts 2 and 3. All 3 rectangular sections were screwed together tightly with the premade slots. In addition to this, epoxy resin was generously added to all surfaces and cracks. Upon testing of the entire assembly, it was seen that both solutions did not sustain in keeping proper air and water seal. The crack area was ejecting large amounts of water while the 3 plates were slightly letting water pass (see figure 4.2). This lack of seal obviously created multiple errors in our specific gravity measurement. Finally, another error that occurred is the accuracy of the volumetric flow meter sensor. As seen in figure 3.2, the flow meter reads a value of 0 during the trial process. This was an error from the sensor itself as the other values depend on the flow volume, thus meaning that the flow meter was reading a value, sending it to equations but not saving/displaying it. This error questions the accuracy of the cheap volumetric flow meter and leads to inaccurate results when calculating the specific gravity.

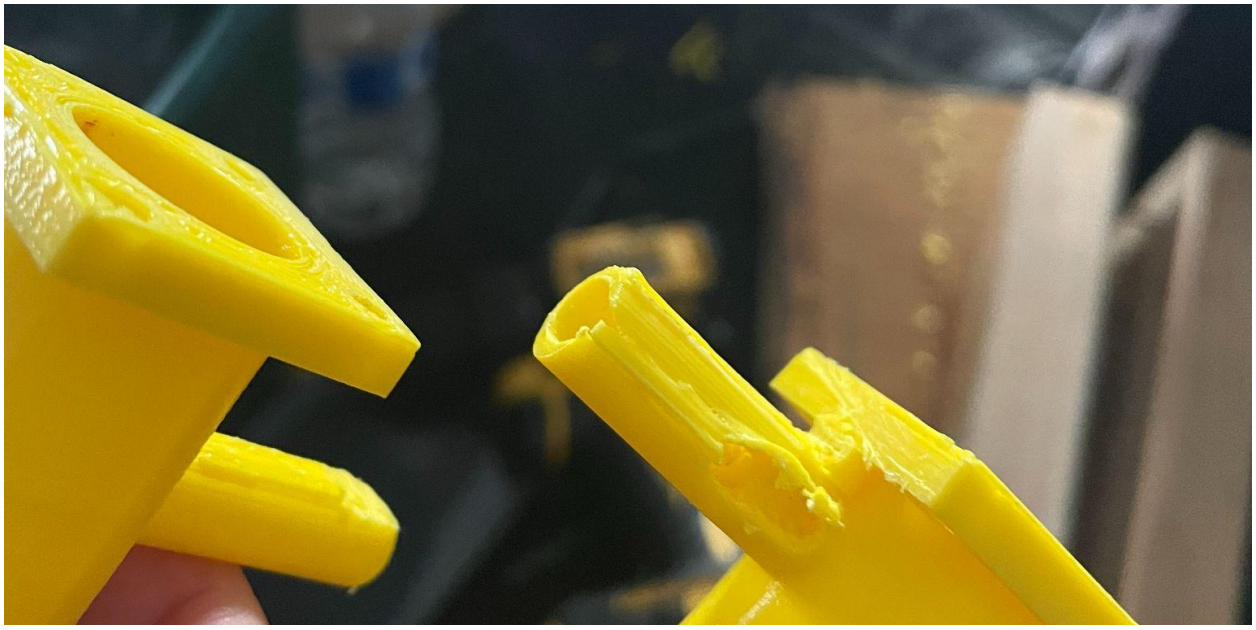

Figure 4.1 - Cracking in the pressure tap during drilling

Figure 4.2 - Water leakage when testing

3.0 - Final Code

```
#include <LiquidCrystal.h> // include library for the LCD Screen

//Variables for the Differential Pressure Sensor
const float ADCtomV = 4.8828125; // Conversion multipler needed from
Arduino ADC value at pin to voltage in mV
const float SensorOff = 200; // units in mV taken from data sheet for
MX5050DP
const float sensitiv = 90; // units in mV/Kpa taken from datasheet
float diffpressure; // The value in Kpa of the differential pressure that
we need
int PIN_P1 = A0; // If the diff pressure sensor is plugged into analog pin
A0

//Variables for the Volumetric Flow Meter
int PinFlow = 2; // This is the flow meter input digital pin 2 on the
arduino
double VolFlow; // this is the volumetric flow we want to find
volatile int sum; // intger is set as volatile for correct updates during
interrupt = important as it will store #of pulses each second

//Variables for LED lights
int redledpin = 24; // define where red led pin is placed 24 digital pin
on arduino
int greenledpin = 25; // define where green led pin is placed 25 digital
pin on arduino

//Variables for the LCD screen
const int rs=1, en=8, D4 = 4, D5=5, D6=6, D7=7;  //rs = rs pin on LCD, en
= enable pin on LCD, D4 = register pin D4, same for D5-D7
LiquidCrystal lcd(rs,en,D4,D5,D6,D7);   //Initialize the library by
associatting LCD interface pin with arduino pin number it is conected to

//Plato Calculations Variables
const float CD = 0.6; // coefficient of discharge for orifice plate finish
(for flat edge is typically 0.6)
const float Beta = 0.33333333333; // ratio between orifce diameter and
inside pipe diamter = 1/3 so 0.333333
```

```
const float Do = 0.00635; // orifice diameter in meters
const float pi = 3.141592654; // pi constant
const float rowWater = 1000; // constant for row of water
float row; // density of fluid constant
float SG; // density of specific gravity constant
float PlatoVal;  // define Plato Value, in float so it can display
Decimals

void setup() {

  lcd.begin(16,2); // setup LCD screen dimensions (16 x2)
  lcd.print("This is Good");

  pinMode(PinFlow, INPUT); // sets the Flow pin from arduino as an input,
pin needs to be set as an input before it can interrupt
  attachInterrupt(0, Flow, RISING); // makes the interrupt 0 (pin #2 on
arduino mega) to run the function "Flow" so the increment function

  pinMode(redledpin, OUTPUT); // setup red and LED as output on arduino
  pinMode(greenledpin, OUTPUT); // setup green LED pin as output on
arduino

  Serial.begin(9600); // start serial communication

}

void loop() {

  //Pressure in Kpa Calculation
  diffpressure = ((analogRead(PIN_P1) * ADCtomV - SensorOff) / sensitiv) *
1000; // result will be in Kpa since [mv * Kpa/mV] = Kpa so we then
multiply by 1000 to get Pa


  //Volumetric flow in m^3/s Calculations
  sum = 0; // resets the counter so we start counting from 0 again since
loops runs over and over again
  interrupts(); // allows interups on the arduino to run, so start
counting # of pules from sensor
  delay(1000);
```

```
  noInterrupts(); // Disable the interrupts on the arduino so stop
counting pulses
  // Now we must do the math to find the volumetric flow since we know
approx 2.25mL fluid/pulse (everytine it rotates)
  VolFlow = (sum*2.25); // #pulse in a second by the amount of 2.25mL of
fluid
  VolFlow = VolFlow / 1000; // convert to L/s
  VolFlow = VolFlow / 1000; //convert L/s to m^3/s

  //Math to find Plato Value
  row = (diffpressure *
(CD*CD)*(Do*Do*Do*Do)*(pi*pi))/(8*(VolFlow*VolFlow)*(1-(Beta*Beta*Beta*Bet
a))); // equation when soving for density
  SG = row/rowWater; //  finding SG value
  PlatoVal = (-1*616.868) + (1111.14* SG) - (630.272 * (SG*SG)) + (135.997
* SG * SG * SG);  // equation found converting SG to Plato

  //LCD code to print resulting Plato Value to Screen
 lcd.clear();
  lcd.setCursor(0, 0); // set the cursor to column 0, line 0 which means
1st row, 1st column
  lcd.print("Plato: ");
  lcd.print(PlatoVal);
  lcd.setCursor(0, 1); // set cursor to print SG value 2nd row, column 0
  lcd.print("SG: ");
  lcd.print(SG);
  delay(500);

  Serial.println(diffpressure);
  Serial.println(VolFlow);
  Serial.println(row);
  Serial.println(SG);
  Serial.println(PlatoVal);

  // Code for LED lights depending on PlatoVal
  if(PlatoVal >0 && PlatoVal <30) { // if the plato value is between 0 and
30 then show green light as it is good

   digitalWrite(redledpin, LOW); // digital write (LOW) = turn Red led off
```

```
   digitalWrite(greenledpin, HIGH); // digitalWire (HIGH) = turns green
LEd on and shows plato range is good

 }

  else if(PlatoVal <0 || PlatoVal > 30){ // if plato value is less then 0
or larger then 30 , show that plato value is bad with red light

    digitalWrite(redledpin, HIGH); // red light is turned on
    digitalWrite(greenledpin, LOW); // green light is turned off
  }


}

void Flow() { // new function created that th einterrupt will run every
time a pulse is found

  sum++; // everytime this function is called an increment, ++ will ad 1
to that variable everytime the program runs

}
```

-