ECE 473: Embedded System Design

Final Project

Human Radar

Mohamed El-Abdallah, Rami Badawi

April 21, 2024

1 Abstract

In this project we proposed the project of making a Human Radar. What that entails is device that is sweeping/scanning the area at a certain degrees of rotation sensing heat signatures near by. If a heat signature is above a certain rating the device will take this as a human. Stopping the scan and grabbing the distance of the human from the device. Then continuing the scan to alert for other humans near by. This report will breakdown the division of work, efforts taken from ECE473, the initial design and scaled back design for this class, testing results and parameters, and learning outcomes from this project.

2 Work Breakdown

The work breakdown has been split evenly depending on the understanding of the necessary tasks to make this system work as intended.

Name	Task
Mohamed El-Abdallah	PID Controller
Mohamed El-Abdallah	Hardware Setup
Mohamed El-Abdallah	Mount CAD
Rami Badawi	Ultrasonic Sensor
Rami Badawi	Analog Communication with Buzzer
Rami Badawi/Mohamed El-Abdallah	IR Sensor Communication
Rami Badawi/Mohamed El-Abdallah	Individual Sensor Performance
Rami Badawi/Mohamed El-Abdallah	Integrated System Performance

Table 1: Work Breakdown of Human Radar

3 Effort Taken From ECE473

This project is a build off of an already completed project from ECE460. In ECE460, Mohamed has made a PID controller that will follow a sinusoidal target using a DC motor with an encoder and an Arduino. The code for this existing project was leveraged however the integration of it was required to be manipulated in order for it to work with the now being used STM32F722ZE microcontroller for this Human Radar.

All other efforts of processing the data from the AK9753 4-channel IR sensor, Ultrasonic, and buzzer together with the PID controller was all effort put in from ECE473.

4 Background

The starting point of our project was to work off of the existing PID controller code that Mohamed made in ECE460. This code was not going to be able to be copy and pasted in the Arduino IDE since the exisiting code was done with an Arduino Uno using a library that handles all atomic functions. This library does not work with the STM32F722ZE microcontroller we are using.

As well as with the Ak9753 4-Channel IR sensor, this sensor comes with an Arduino library that has functions that allow us to easily access register values after the sensor does the conversion of reading to a picoAmp output. Along with the internal conversion from the library, the library has a function that allows us to tell the direction of the humans movement. This is important because in our original design of this project, we wanted the PID to have a changing target when the human is detected so that the IR sensor would track the human when it is in the frame, consistently collecting its distance from the apparatus.

This design was scaled back due to the lack of time available that would be necessary to implement this design feature.

Example code is available on the IR sensor, however it was minimally referenced due to

the ease of the labeled functions in the "hook up guide".

5 Design

5.1 Hardware Used

The following table will list the hardware used for this project to be operational. Along with the last the table will include the purpose of the device in the project

Device Name	Use Case	Link
DC Motor	Spinning the Sensor Apparatus	Amazon
VNH5019 DC Motor Controller	Controlling DC Motor with PID	Pololu
STM32F722ZE	Microcontroller	Digikey
AK9753 Human Presensece Sensor	4-Channnel IR Sesnor	Sparkfun
HC-SRO4 Ultrasonic Sensor	Distance Tracking	Digikey

Table 2: Devices Used In Human Radar

5.2 Math, Science and Principles

Throughout the development of this project, some key understanding in our design, analysis and implementation of the program needed to be solid. The first key understanding required was the PID equation which is shown below.

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_p \frac{de}{dt}$$

Changing the gains of K_p,K_i,K_d will allow us to tweak the controller to the correct response to the target position we are looking for.

Next necessary understanding is how the ultrasonic sensor will be processed. How the ultrasonic works is that it will send a trigger signal and it will record the time between the

trigger and echo received. We can take that pulse width and compute that into a distance to be used. this equation will be below:

$$D = \frac{PW \cdot .000343}{2}$$

5.3 Design Alternatives

During the initial design, we may have been more ambitious on the design than we could have handled in the amount of time given.

We found that getting the PID controller to track to human while moving lateral to the sensor was going to be harder than originally planned. During the development of the system we have decided to scale the project back and instead of having the tracking function be our main goal. We moved to a solution that would still stop the motor spinning, but instead of tracking, it would gather distance data for the user and then when the human is out of detecting range or line of sight (LOS), it would continue on original sinusoidal target.

We still had the intention of developing the tracking function if time was available after the completion of the main goal.

5.4 Requirements

The requirements of this system are going to be listed below based on the goal of the radar, between sweeping stopping then continuing vs sweeping and tracking

Sweeping interrupt:

- Have a PID loop control the Motor with 180 degrees of spin
- Ultrasonic sensor will accurately detect distance on a human body
- The IR sensor will accurately trigger when a human is in front of it

- Buzzer will sound at a different frequency depending on distance from the Ultrasonic sensor

The below requirements will list out what the requirements would have been for the Tracking feature:

- Correctly output the direction of the human when moving lateral to the sensors Field of View (FOV)
- Have a target change based on the humans movement in the IR sensor FOV
- Have the PID controller smoothly move the DC motor shaft based on the movement of target
- Set an interrupt to consistently sense the distance of the human while in tracking mode.

5.5 Test Plan

The test plan will be in a list of items that we have used to check off certain milestones during the analysis of this project.

1. Test Ultrasonic Sensor
   (a) Set up a tape measure that stretches beyond the maximum detecting distance, specified in the data sheet, 4m
   (b) Move a block of wood at different distances measured on the tape measure and gather readings from the ultrasonic and determine accuracy
2. Test IR Sensor
   (a) Determine the distance at which the IR sensor can detect a human accurately while directly in front of the sensor
   (b) Determine how wide of a FOV the sensor can detect in

3. Test PID Controller

   (a) How accurately can the PID follow a sinusoidal target

   (b) Can the PID stop the DC with no drift

   (c) Can the PID resume motion accurately based on its last position and direction

6 Development

6.1 Hardware Set Up

The hardware set up will cover the necessary connections from the sensor pins to the headers of the microcontroller

Sensor	Pin	STM32 Headers
Ultrasonic	Trig	PE2
Ultrasonic	Echo	PE3
IR Sensor	SDA	PB9
IR Sensor	SCL	PB8
Buzzer	PWM	PG2
Motor Controller	INA	PE5
Motor Controller	INB	PE3
Motor Controller	PWM	PD15
DC Motor	Encoder A	PF8
Dc Motor	Encoder B	PF7

Table 3: Hardware Connections to STM32

6.2 PID Controller on STM32

6.2.1 Atomic Library

To start off this project, I first wanted to test the already existing PID code from an Arduino on the STM32F722ZE. When I copy and pasted the code I found that the code was using a library called

```
#include <util/atomic.h>
```

This library makes the entire code atomic. this allows for the code to not be interrupted at different function calls. Eliminating shared data problems throughout the code. This was not going to work on the STM as this library is not compatible with the hardware architecture.

what had to change in the program in order for this to work is of the following:

```
noInterrupts(); // disable interrupts temporarily while reading
pos = posi;
interrupts(); // turn interrupts back on
```

Rather than using a function called $ATOMIC\_BLOCK$. Which does the same as the above, not allowing an interrupt to be called when the operation is being executed. However the function $ATOMIC\_BLOCK$ does it in a very fast secure way rather than disabling and enabling interrupts. Unfortunately this is the only solution we could find for the replacement of this function.

6.2.2 Tuning PID

How the controller is set up is values for K_p,K_i,K_d and the values of each will dictate the reaction of the PID. AS I was testing the PID on the STM I found that the STM had a hard time digitizing the sinusoidal target with a very small sampling time making the wave look like a step function in a sinusoidal shape as seen below in the images:

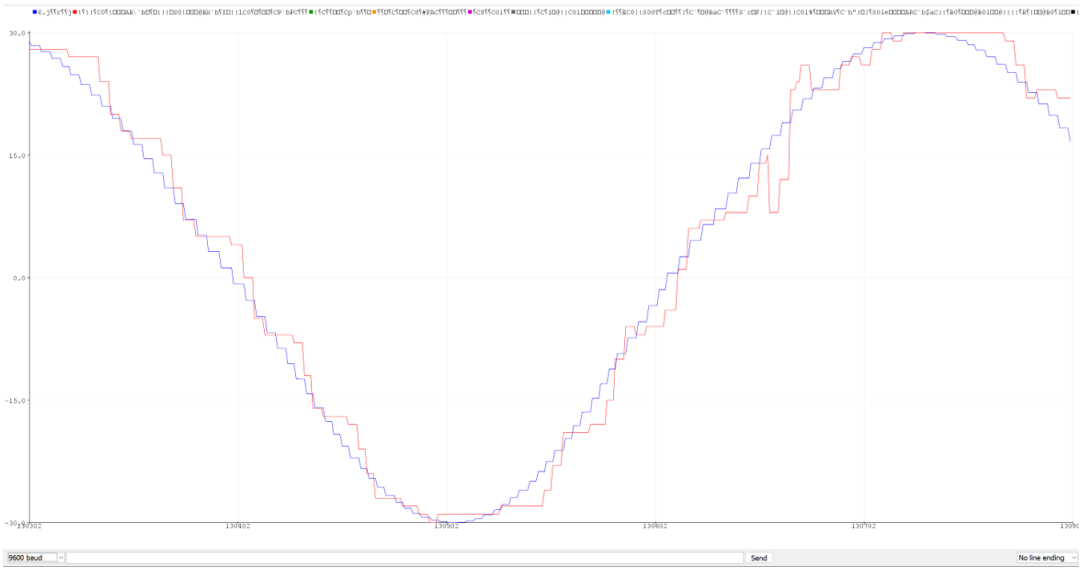

Figure 1: Target and Position of DC Motor at 9600 Baud

One thing that I found that would help make the steps a lot smaller on the target sinusoid is by increasing the baud rate. However with this solution, it does not affect the position of the dc motors smoothness. It will only affect the target wave.

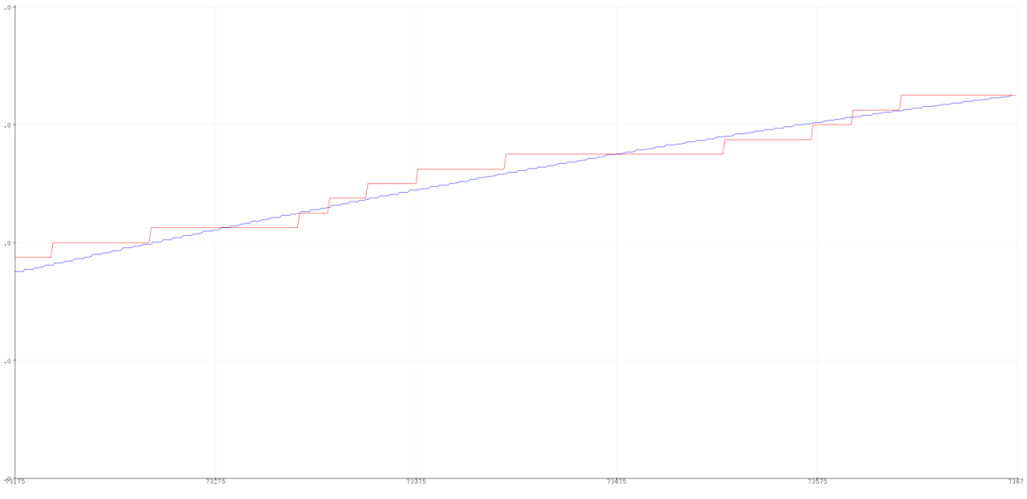

Figure 2: Target and Position of DC Motor at 115200 Baud

I found that using a baud rate of 9600 was going to be the best for the entire system as it will not push the other sensors to its limits. As well as tuning the gains of the PID accordingly allows for minimal random spikes in the response. The values I found the best reactions were of the following:

$$K_p = 5$$

$$K_d = .05$$

$$K_i = 0$$

6.3 Human Presence Sensor

6.3.1 Initialization and Reading Registers

Through the Arduino library interfacing with the sensor was very streamlined. I first has to initialize a library called $Wire.h$ and the sensors library. From there the initialization of the sensor was completed for us since wire.h handles majority of your initialization as long as the microcontroller can recognize what is connected to it. After that I had to set up an object called *movementsensor* from the AK975x class.

```
#include <SparkFun_AK975X_Arduino_Library.h>
#include <Wire.h>
.
.
.
AK975X movementSensor;
```

After that, reading the registers are just as easy as calling the function:

```
var = movementSensor.getIRx();
```

what this does is get the value from the individual register from the IR sensor, this sensor has 4 registers so we would replace the x with the desired number register was want.

The reason we opted for this simple solution is because, we first tested this all on a the TIVA board using Keil. We shortly after found that we need to initalize each register as an I2C device with their own address, which would not have been a problem, if we were able to call in the library for the sensor into Keil μVision. With out the library the value output of the sensor is a constant 255 using Putty to view the serial output. This is because the value output is then converted into a picoAmp value that we can use to detect a human. The table below is shown and pulled from the data sheet on values its converted to:

Measurement Data IR Sensor [15:0]			Output Current of IR Sensor	Unit
Two's Complement	Hex	Decimal		
0111 1111 1111 1111	7FFF	32767	14286.4	
⋮	⋮	⋮	⋮	
0101 1001 1001 1000	5998	22936	10000.1	
⋮	⋮	⋮	⋮	
0100 0000 1000 0010	4082	16514	7000.1	
⋮	⋮	⋮	⋮	
0000 1000 1111 0110	08F6	2294	1000.2	
⋮	⋮	⋮	⋮	
0000 0000 0010 0000	0020	32	14.0	
⋮	⋮	⋮	⋮	
0000 0000 0000 0000	0000	0	0	pA
⋮	⋮	⋮	⋮	
1111 1111 1110 0000	FFE0	-31	13.5	
⋮	⋮	⋮	⋮	
1111 0111 0000 1001	F709	-2294	-1000.2	
⋮	⋮	⋮	⋮	
1011 1111 0111 1101	BF7D	-16514	-7200.1	
⋮	⋮	⋮	⋮	
1010 0110 0110 0111	9667	-22936	-10000.1	
⋮	⋮	⋮	⋮	
1000 0000 0000 0000	8000	-32768	-14286.8	

Figure 3: Value Conversions for Human Presence Sensor

6.3.2 Fresnel Lens

On the IR sensor we found that the detection range of the was very sporadic and sensitive based on the environment around. On forums online to get a more accurate read, we would need a Fresnel Lens on the focal point of the IR sensor.

We purchased a cheap Fresnel Lens on Amazon that is made of a plastic backing. After

speaking with Professor David Daniszewski, he informed us that we would need to have this lens sit near the focal point of the sensor in order for it to actually refract the IR sensors FOV into a straight beams. The following image was found off of google.com to visually represent the idea behind this.

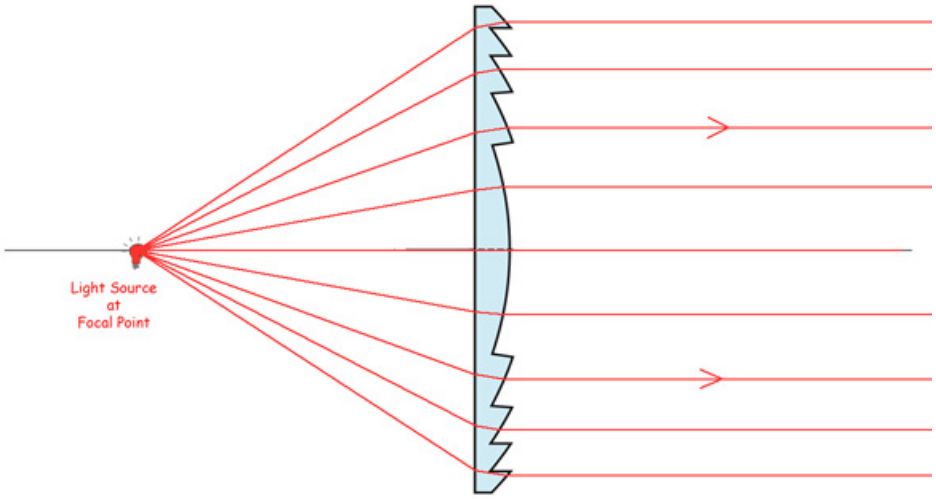

Figure 4: Frensel Lens Refraction Diagram

However we found that the Fresnel Lens will hold heat itself due to it being made of plastic causing an offset in the reading values. While testing the Fresnel lens at a higher reading threshold we have concluded that it does not actually extend the detection range in practice due to the IR sensor being such low power and value.

6.4 Ultrasonic Sensor and Buzzer

As stated before, the Ultrasonic sensor and buzzer were an easy implementation and to design around. For the Ultrasonic, to read its data we want to send the chirp and read the echo time. This value is equal to our pulse width which we can easily convert into a distance.

The diagram below will depict this, the following image was found on Google.com

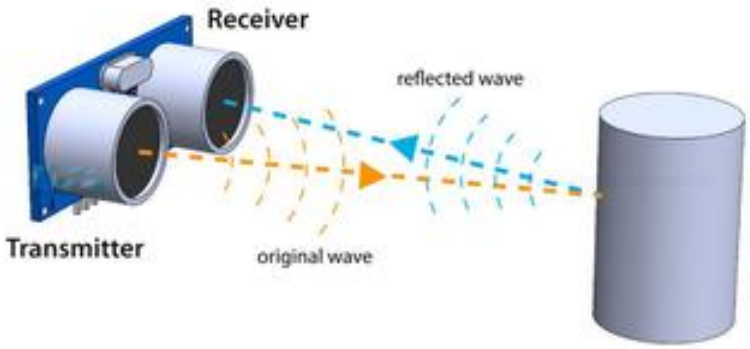

Figure 5: Ultrasonic Diagram

This set up and value can be calculated by the following lines of code that I made into a function:

```
digitalWrite(triggerPin, HIGH); //sends/reads trigger
delayMicroseconds(10);
digitalWrite(triggerPin, LOW); //sends/reads echo
pulse_width = pulseIn(echoPin, HIGH); //finds the time between trigger and
    echo pulses
distance = (pulse_width*.000343)/2; //calculating distance the speed of sound
    output in meters
```

As for the buzzer, it is also as easy as the ultrasonic sensor. Arduino IDE has a function called *Tone*. Where we can set the pin we want a PWM signal sent to and a frequency of the signal. This will allow us to beep at different tones depending on different factors of our choosing. In this project, we have chosen to have different frequencies based on distance.

```
if(distance > 2.01)
{
  tone(buzzerPin, 2000); //tone(x,Y)x=pin on board, Y = frequency tone for
      buzzer output
}
```

6.5 CAD Mounts

6.5.1 Sensor Appuratus

To hold all these sensors and DC motor together, I wanted to make mounts for all of them. The idea behind this was to make sure that these devices were packaged into an easy to integrate package that would be able to be taken on the move. First I will go over how I designed the mount for the sensors. The drawing sheet of the design is shown below with necessary dimensions in millimeters.

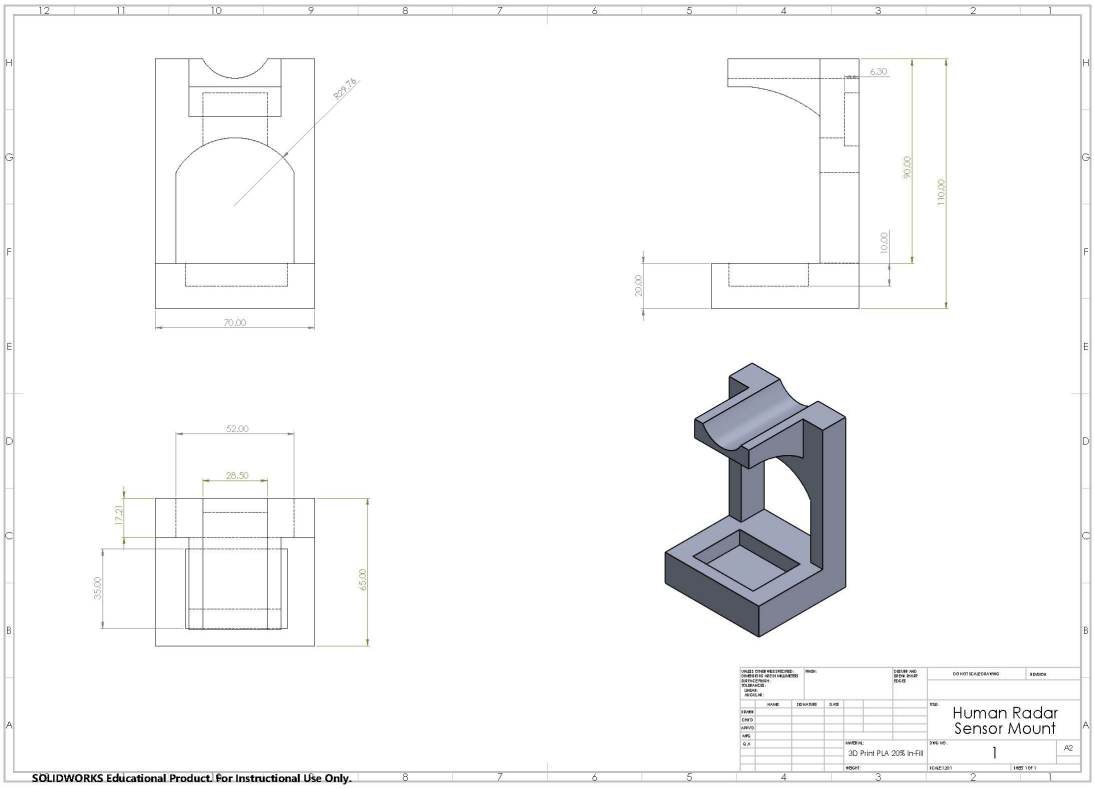

Figure 6: Sensor Mount

The design behind this sensor was the intention of holding a small breadboard to connect the ultrasonic and IR sensor to, before wiring them to the STM board. This allowed us to

have an easier wire harness and make sure that wiring is cleaner than directly connecting from sensors to the STM. This as well will hold the sensors in the correct orientation for our application while it is sweeping/scanning the area back and fourth. The purpose for the channel on the top of the sensor mount is for mounting a laser. The laser will be for the user to tell the absolute direction the IR is pointing to, if this is to be used in night time settings.

6.5.2 DC Motor Holder

For this mount I wanted it to be as a base to support the DC motor while standing with the sensor apparatus on top of it. I made the base of it just as wide as the sensor apparatus to ensure that the motor does not fall over while it is spinning. I as well designed a cap for the holder that will screw into the two threaded holes in the DC motor and then screwed into the base to make sure the motor does not spin within the mount as well.

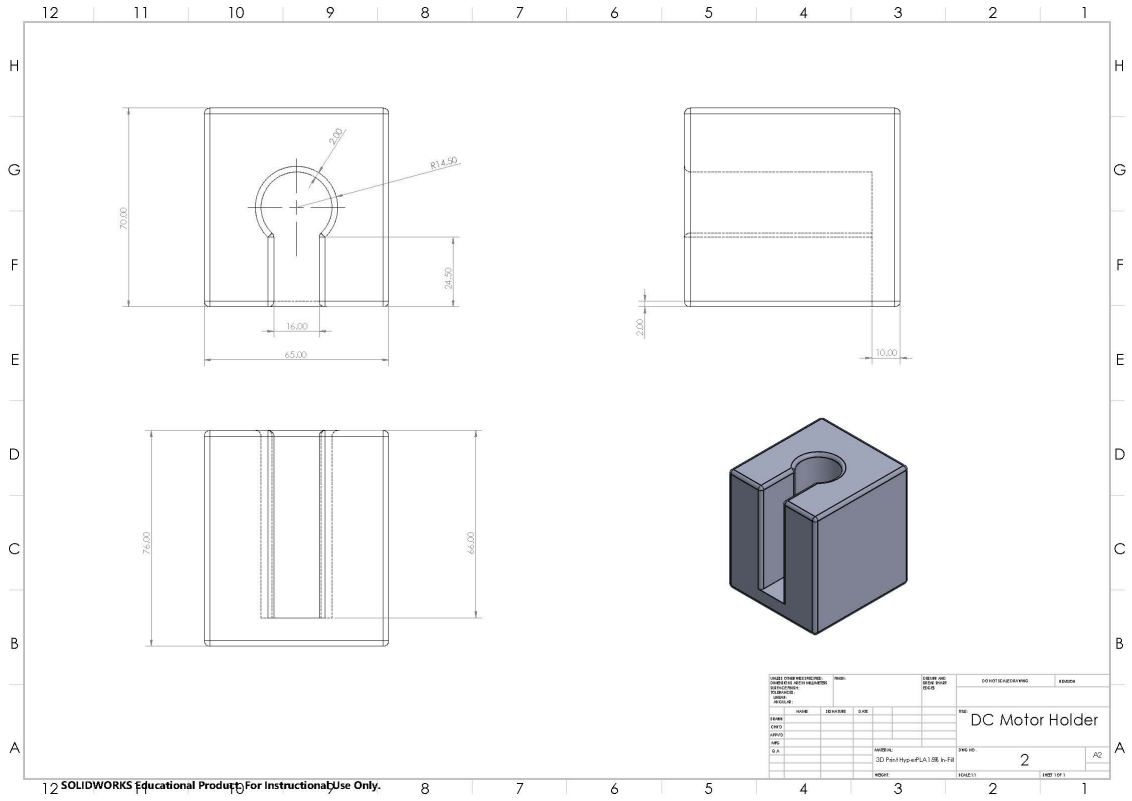

Figure 7: DC Motor Holder

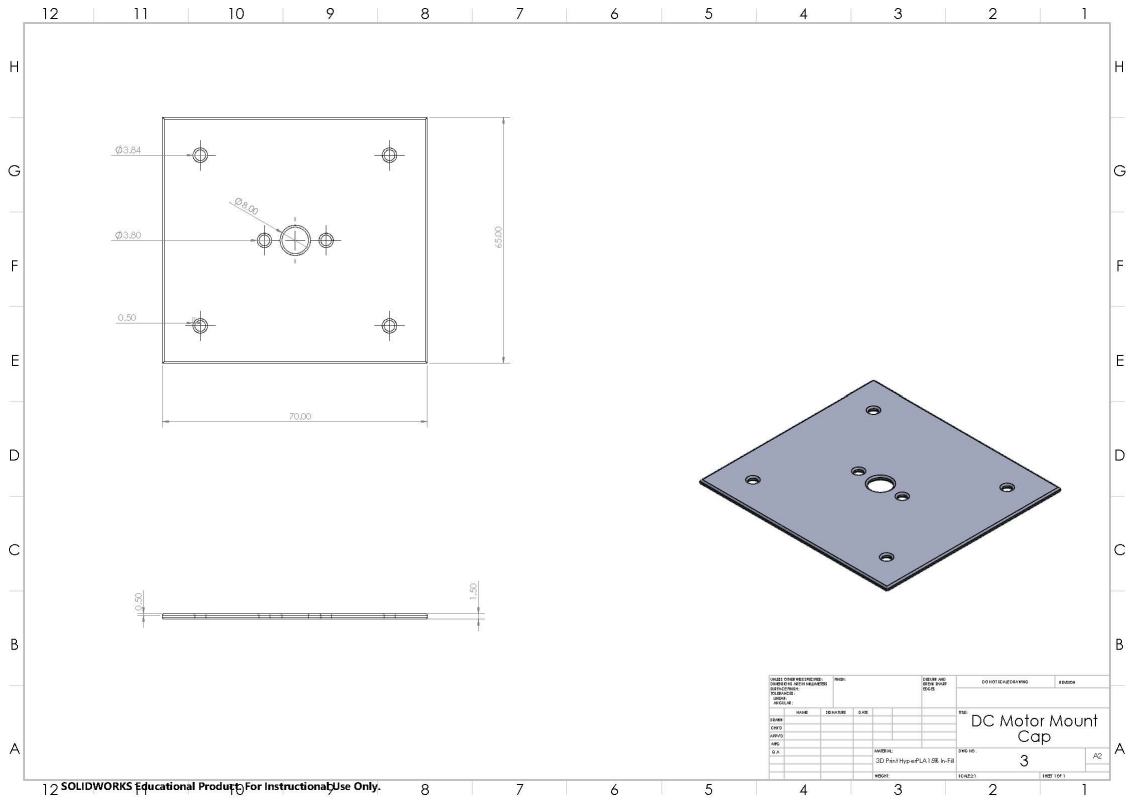

Figure 8: DC Motor Holder Cap

6.5.3 System Integration

Within this section, I will include images of the entire system integration to provide a recap of how all of sensors, mounts, and motors are combined to achieve the correct guidelines for our testing

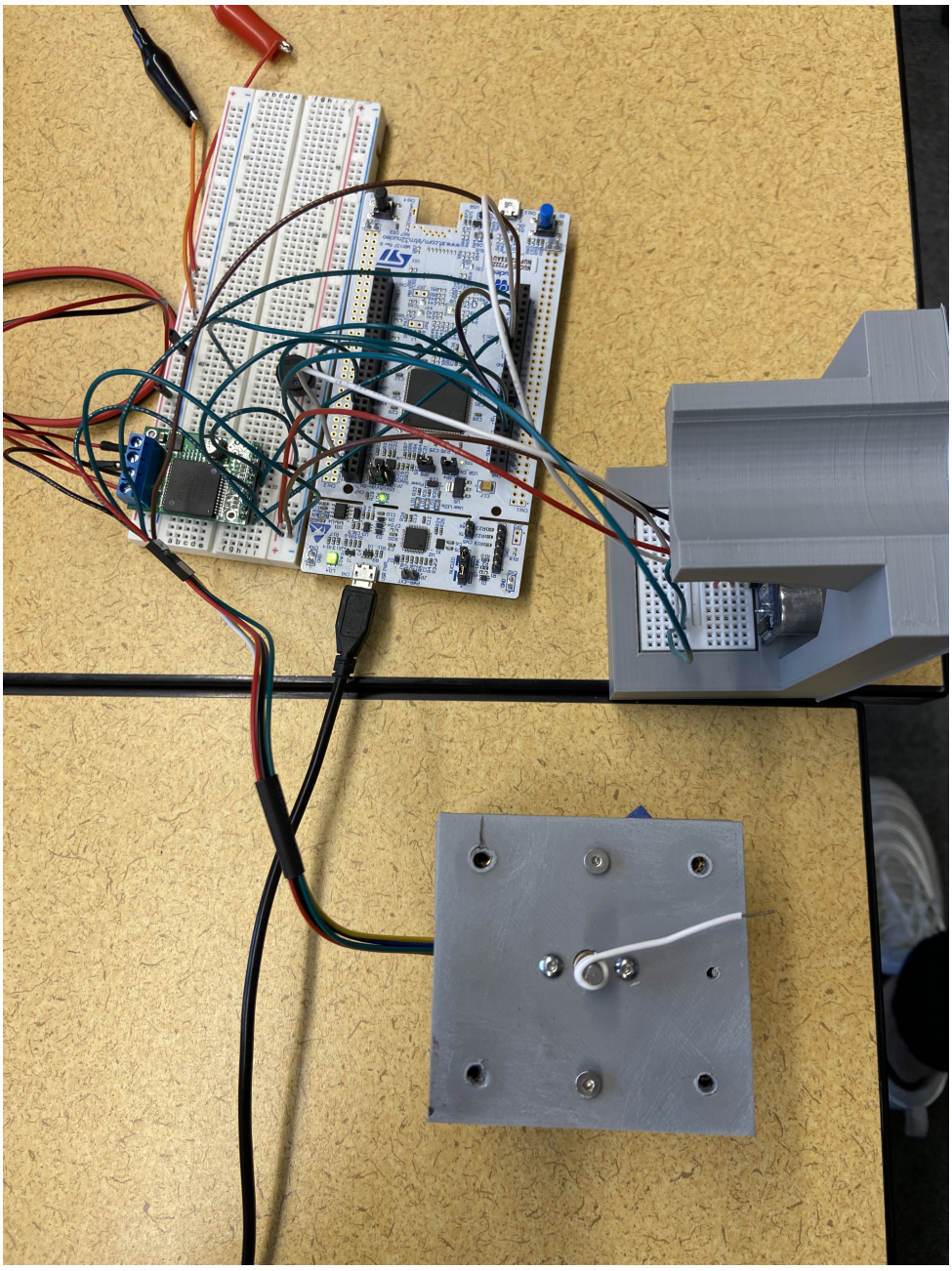

Figure 9: System Overview

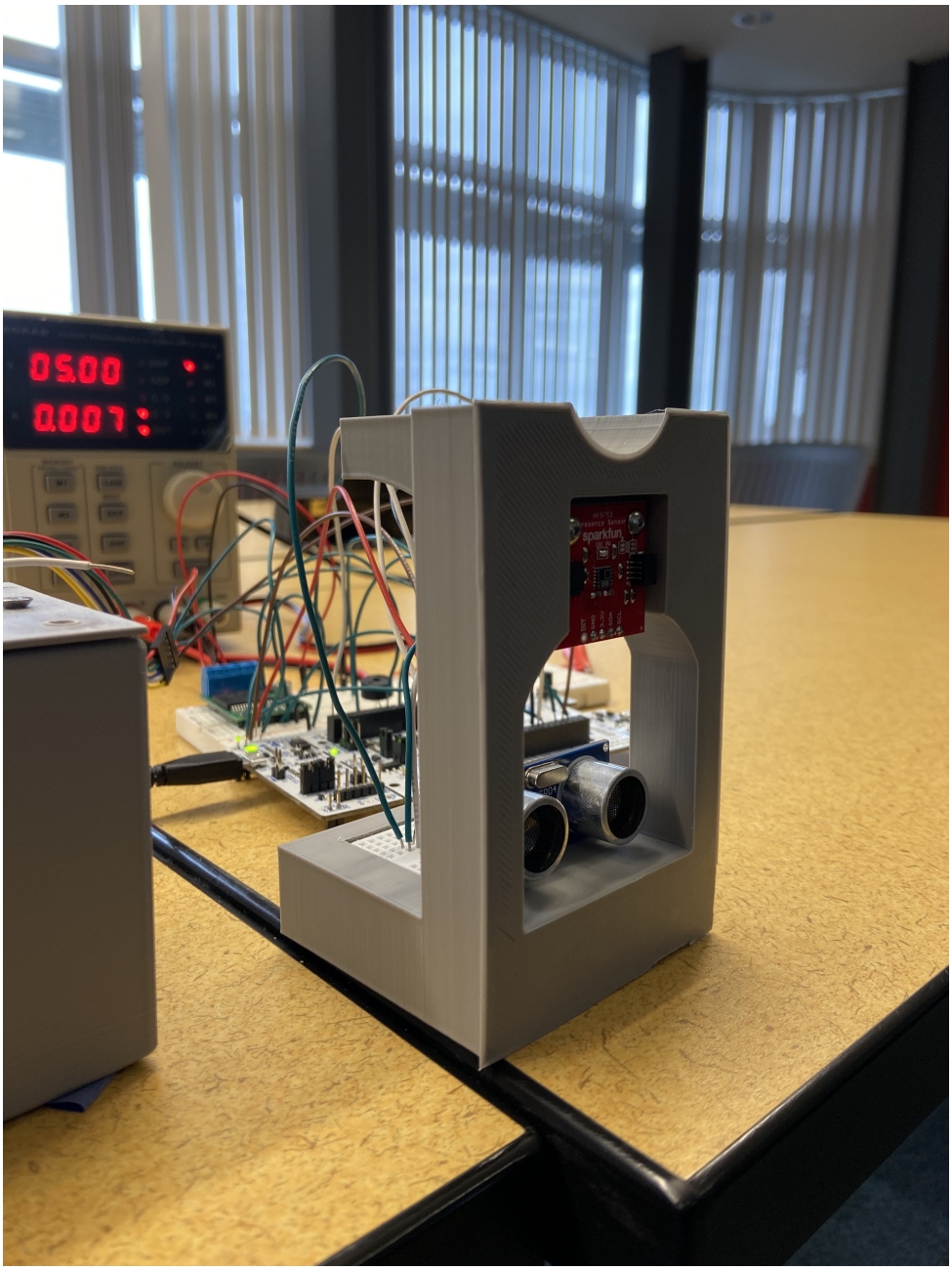

Figure 10: Sensor Apparatus Sensor Integration

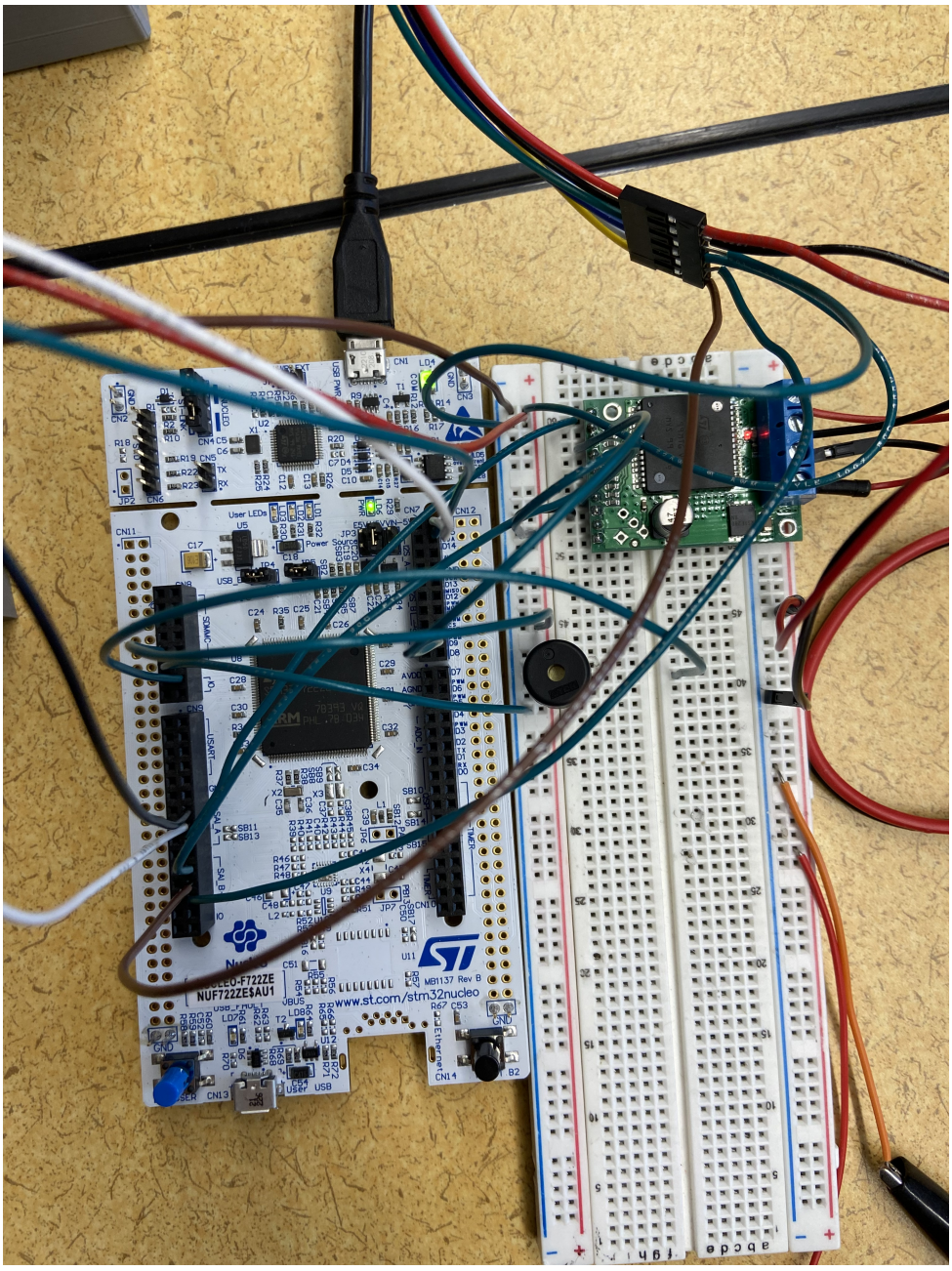

Figure 11: Micro controller and Motor Controller Wiring

6.6 Interrupt Logic

In this program, there is one interrupt which is an if case. The logic flows as if the all 4 channels of the IR sensor are higher than a set threshold, we can assume a human is in front

of it. Turn the motor off, delay for a set time, send the chirp from distance and beep the buzzer. Else, turn the buzzer off.

This logic is shown below:

```
  long elapsedtime;
if(ir1 && ir2 && ir3 && ir4 >= 1500)
{
  elapsedtime = micros();
  setMotor(0, 0, PWM, IN1, IN2);
  delay(2000);
  Ultrasonic();
  Buzzer(true);
  currT = currT - elapsedtime;
}
else
{
  Buzzer(false);
}
```

7 Test Results

7.1 Ultrasonic Test

To test the ultrasonic sensor, the intention of this test was to determine how accurate the readings will be up to a certain distance. The set up is that I took a tape measure and stretched it to the absolute maximum that the sensor can read up to. I took a block of wood and put it at different distances on the tape measure to see how far it can accurately read upto.

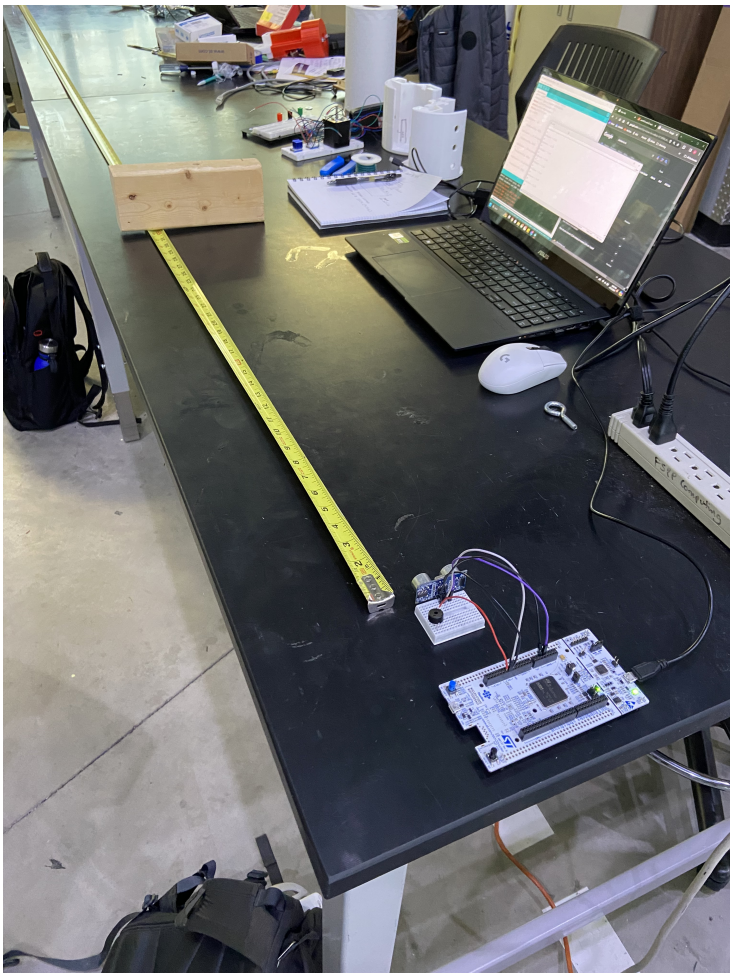

Figure 12: Set Up of Ultrasonic Test

I found that the Ultrasonic sensor can detect and read a entities distance of 3.17 meters. Passed that the sensor will misread distances as 31 meters and be bouncing around this. Because of this I have determined the ultrasonic sensor can not be pushed passed this limit.

7.2 IR Sensor Test

For our IR sensor test, I did the same thing as I did in the Ultrasonic test. Determining distance at which the IR sensor can detect a human accurately while having the human directly in front of the sensor. Because of the low price, quality and overall build of the IR sensor, its accuracy is only limited to about a 1.5 feet from the sensor. Even with the Fresnel

Lens on its focal point its detecting distance is not extended out to the 3 meters it states in the data sheet. We found that using a value of 900 from the registers is a distance of 1.5 feet from it and will be enough to trigger our interrupt of stopping the motor and grabbing distance and beeping the buzzer.

7.3 PID Stop/Resume Test

This test is to see how quickly the PID will stop when the IR sensor goes high on a reading and how it reacts when it resumes the sweep after the sensor has a low reading.

I found that setting a constant target for the motor when the IR goes high will cause the PID to have a lot of back and fourth oscillations to get to its target position. The best and fastest way I found was to turn the motor off. This will allow for it to detect, stop sweep, then resume. Upon the resume, I found that it would jump to a position very quickly rather than continuing on the smooth path it was on, before it got interrupted. This is because the PID works off the time it is running. What I had to do was to record the time the motor stopped moving and subtract that from the total time the program has been running. This will allow for the motor to continue on the same path it was before we interrupted it. This solution can be shown below in code:

```
elapsedtime = micros();
.
.
.
currT = currT - elapsedtime;
```

8 Learning Outcomes

The software architecture used in this project is a round robin architecture, this is the native format for an Arduino project, the only reason we had to use Arduino IDE is to use the AK975x library for the Human Presence Sensor. Timing deadlines are met by calling

functions at the correct time within the interrupt and keeping functions as short as possible. This is so we do not spend too much time processing and running through functions.

The engineering design process allowed for us to breakdown our project into non changeable requirements and testing requirements for us to breakdown how to produce the working system for our project. It allowed us to see that every requirement must have a test and a way to test it.

9 Conclusion

In conclusion, the development of the Human Radar project has been a journey of exploration, experimentation, and learning. Despite encountering challenges along the way, such as integrating various sensors, adapting code to a new microcontroller platform, and refining the control algorithm. The iterative process of designing, implementing, and testing allowed us to simplify our approach and optimize the performance of the Human Radar. While some aspects of the project, such as implementing real-time tracking of human targets, proved to be more complex than initially anticipated and had to be scaled back, we embraced the opportunity to adapt and prioritize the core functionalities. This decision enabled us to focus on delivering a robust solution that effectively detects human presence, stops the motor sweep, and gathers distance data with precision.

10 Code

```
    #include <SparkFun_AK975X_Arduino_Library.h>
#include <Wire.h>


//GLOABAL VARIABLES FOR ALL PROJECT WILL BE SEPRARTED BY SENSOR OF USE CASE
//PID controller and interface pins
#define ENCA PF8 //2 // YELLOW
#define ENCB PF7 //3 // WHITE
```

```
#define PWM PD15 //5 //PWM
#define IN2 PE5//6 //INB
#define IN1 PE3 //7 //INA

volatile int posi = 0;
long prevT = 0;
float eprev = 0;
float eintegral = 0;
float alpha = 1.005; // Smoothing factor, adjust as needed
float filteredPos = 0;
float target_g;
float kp;
float ki;
float kd;
long currT;

//Human Presesnce Sensor
AK975X movementSensor;
int ir1, ir2, ir3, ir4;


//Ultrasonic Sensor
const int triggerPin = PE2; //Pin definitions on STM32F722ZE;
const int echoPin = PE4;
float pulse_width, distance;

//Buzzer
int buzzerPin = PG2;

void setup()
{
  Serial.begin(9600);


  //PID Setup
  pinMode(ENCA,INPUT);
  pinMode(ENCB,INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCA),readEncoder,RISING);
  pinMode(PWM,OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);

  //HPS Setup
  Wire.begin();
  void setMode(uint8_t mode = AK975X_MODE_0);
  if (movementSensor.begin() == false)
  {
```

```
    Serial.println("Device not found. Check wiring.");
    while (1);
  }

  //Ultrasonic Setup
  pinMode(triggerPin, OUTPUT);
  pinMode(echoPin, INPUT);
  digitalWrite(triggerPin, LOW);

}




void loop()
{
  if (movementSensor.available())
  {
    ir1 = movementSensor.getIR1();
    ir2 = movementSensor.getIR2();
    ir3 = movementSensor.getIR3();
    ir4 = movementSensor.getIR4();
  }

  movementSensor.refresh(); //Read dummy register after new data is read

  target_g = -25*sin(prevT/1e6 + 90);

  kp = 5;
  kd = .05;
  ki = 0;

//PID HERE
//====================================
  PID();
//========================================

  long elapsedtime;
  if(ir1 && ir2 && ir3 && ir4 >= 1500)
  {
    elapsedtime = micros();
    setMotor(0, 0, PWM, IN1, IN2);
    delay(2000);
    Ultrasonic();
    Buzzer(true);
    currT = currT - elapsedtime;
  }
```

```
  else
  {
    Buzzer(false);
  }
//PRINTING THE VALUE OF THE IR
//    Serial.print("]\t2:LFT[");
//    Serial.print(ir2);
//    Serial.print("]\t4:RGH[");
//    Serial.print(ir4);
//    Serial.print("]\t3:UP[");
//    Serial.print(ir3);
//    Serial.print(" :DWN[");
//    Serial.print(ir1);
//    Serial.println();

//PRINTING THE VALUE OF THE DISTANCE
//  Serial.print("Distance = ");
//  Serial.println(distance);
//  Serial.println(" m");

//PRINTING THE PERFORMANCE OF THE PID
//  Serial.print(target_g);
//  Serial.print(" ");
//  Serial.print(filteredPos); // Output the filtered position instead of the raw
    position
//  Serial.println();
}

//FUNCTIONS
//==================================
void setMotor(int dir, float pwmVal, float pwm, float in1, float in2)
{
  analogWrite(pwm, pwmVal);
  if (dir == 1) {
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
  }
  else if (dir == -1) {
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
  }
  else {
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
  }
}
```

```
void readEncoder() {
  float b = digitalRead(ENCB);
  if (b > 0) {
    posi++;
  }
  else {
    posi--;
  }
}

void Buzzer(bool state)
{
  if (state) {
    digitalWrite(buzzerPin, HIGH);
    if(distance > 2.01)
    {
      tone(buzzerPin, 2000); //tone(x,Y)x=pin on board, Y = frequency tone for
          buzzer output
    }
    else if(distance < 2 && distance > 1.01)
    {
      tone(buzzerPin, 1500); //tone(x,Y)x=pin on board, Y = frequency tone for
          buzzer output
    }
    else if(distance < 1 && distance > .76)
    {
      tone(buzzerPin, 1100); //tone(x,Y)x=pin on board, Y = frequency tone for
          buzzer output
    }
    else if(distance < .75 && distance > .51)
    {
      tone(buzzerPin, 800);
    }
    else if(distance < .50 && distance >.26)
    {
      tone(buzzerPin, 600);
    }
    else if(distance < .25 && distance > .3)
    {
      tone(buzzerPin, 100);
    }
  }
  else
  {
    digitalWrite(buzzerPin, LOW);
    noTone(buzzerPin); // Stop any ongoing tone
  }
```

```
}
void PID()
{
  kp = 5;
  kd = .05;
  ki = 0;

  // time difference
  currT = micros();
  float deltaT = ((float) (currT - prevT))/(1.0e6); //calculats: seconds =
     microseconds / 1e6
  prevT = currT;

  // Read the position
  float pos = 0;
  noInterrupts(); // disable interrupts temporarily while reading
  pos = posi;
  interrupts(); // turn interrupts back on

  // Apply low-pass filter (EMA)
  filteredPos = (alpha * pos) + ((1 - alpha) * filteredPos);

  // error
  float e = filteredPos - target_g;

  // derivative
  float dedt = (e - eprev) / deltaT;

  // integral
  eintegral = eintegral + e * deltaT;

  // control signal
  float u = kp * e + kd * dedt + ki * eintegral;

  // motor power
  float pwr = fabs(u);
  if (pwr > 255) {
    pwr = 255;
  }

  // motor direction
  int dir = 1;
  if (u < 0) {
    dir = -1;
  }

  // signal the motor
```

```
  setMotor(dir, pwr, PWM, IN1, IN2);

  // store previous error
  eprev = e;
}

void Ultrasonic(){
    digitalWrite(triggerPin, HIGH); //sends/reads trigger
    delayMicroseconds(10);
    digitalWrite(triggerPin, LOW); //sends/reads echo
    pulse_width = pulseIn(echoPin, HIGH); //finds the time between trigger and
        echo pulses
    distance = (pulse_width*.000343)/2; //calculating distance the speed of sound
        output in meters
}
```