SENSORS & MOTORS LAB

Individual Lab Report 1

Jimit Gandhi

Team G- Robographers

Team Members

Rohit Dashrathi

Sida Wang

Gauri Gandhi

Tiffany May

1.0 Introduction

In the sensor and motor lab we planned to design a state machine that could change states to measure parameters like force, temperature, distance and presence of an object to drive the three and control the servo motors, stepper motor and DC motor.

2.0 Individual Progress and Contribution

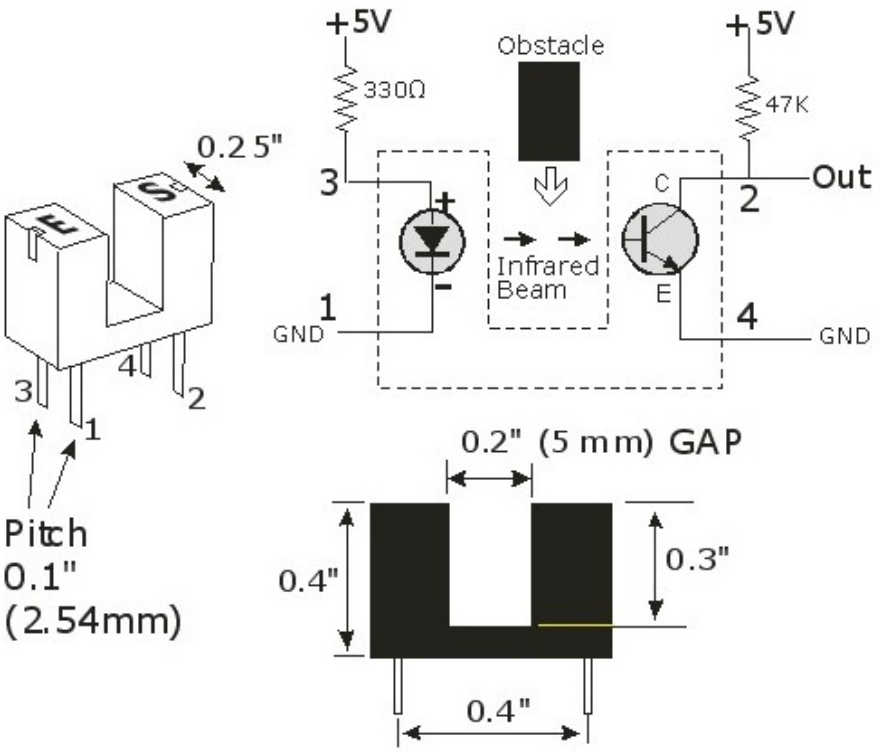
For this assignment, I was responsible for interfacing and integrating DC Motor and Slot sensor.

Slot sensors, sometimes called optical fork sensors [1] because of their "forked" shape, detect objects that pass between the two arms—one with the emitter, the other with the receiver. The fixed slot width provides reliable opposed-mode sensing of objects as small as 0.30 mm.

2.1.1 Slot Sensor

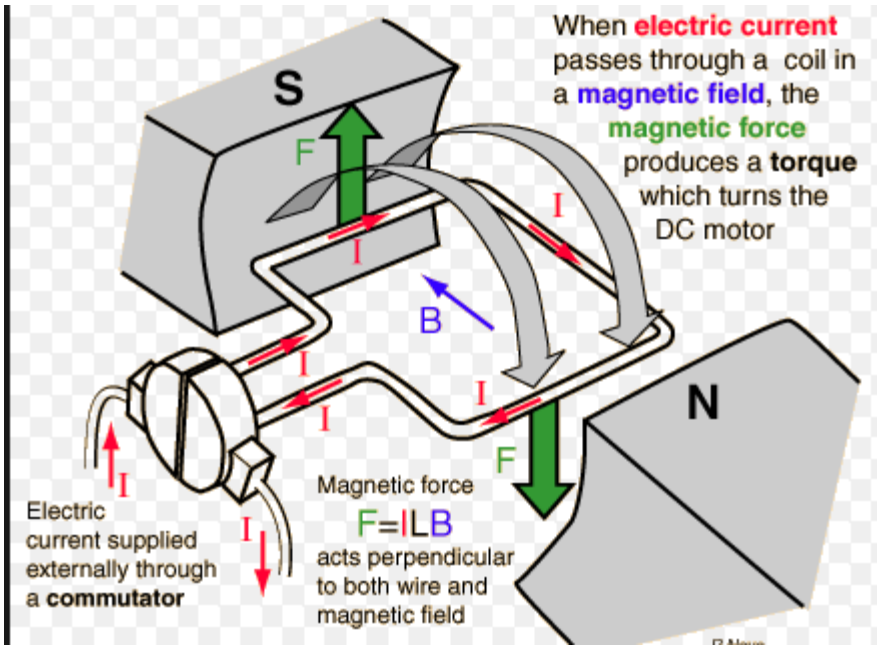
On one arm it has an IR Emitter while on the other arm it has an IR Detector.
When light emitted by the IR LED is blocked because of alternating slots of the encoder disc logic level of the photo diode changes. This change in the logic level can be sensed by the microcontroller. When connected to the microcontroller it gives the reading between 0 (low) to 1024 (high) values.

This sensor was connected as shown in the figure below. [1]



2.1.2 DC motor

I was required to integrate the DC motor functionality to the system. The DC motor works as given in the figure below. My aim was to implement the closed loop controller on DC motor that controlled position and velocity of the DC motor based on position feedback. In this closed loop speed controller, a signal proportional to the motor speed is fed back into the input where it is subtracted from the set point to produce an error signal. This error signal is then used to work out what the magnitude of controller output should be to make the motor run at the required set point speed.

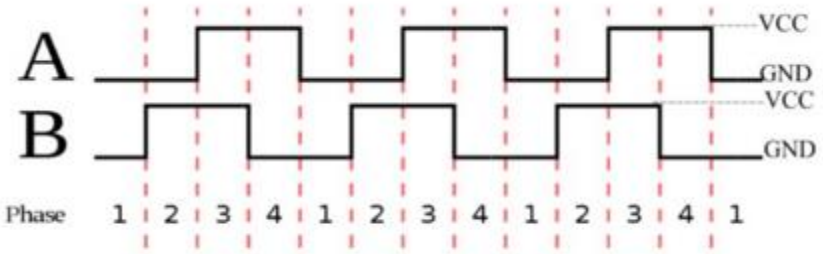


Source: Wikipedia

2.1.2 PROCEDURE

As decided, I worked to control the velocity of the motor on readings of slot sensor. The velocity of the DC motor was to be determined by mapping the reading from the slot sensor.

The SPG30-60K dc motor came a 5V Quadrature Hall Effect Encoder for monitoring the position and direction of rotation. The direction and the angle of the rotation of the motor can be determined by the following principle.



Clockwise Rotation

Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Counter Clockwise Rotation

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

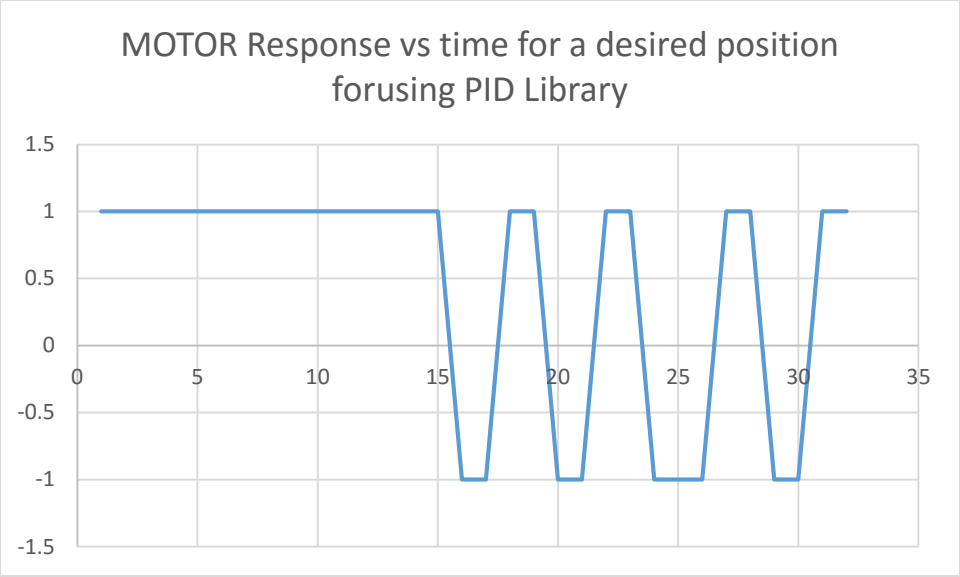
Square quadrature waveform for Channel A and B (Clockwise)

1. We attach an interrupt to Channel A.
2. Whenever the state of Channel A changes the interrupt function gets activated. The code is below
3. According to the code I implemented , whenever the state of channel of A changes, the interrupt gets activated. The function doEncoder() checks the state of state B whenever state A changes its value.
4. If the state of B is same as state of A when A changes then B leads. Thus the motor rotates anticlockwise.
5. If it is not the same as the changed A then it rotates in the clockwise direction.
6. The following code was implemented by me according to the same logic.

```
//Code for calculating the Encoding value of DC motor.
void doEncoder()
 {
 int aState = digitalRead(Encoder_ChA);
 int bState = digitalRead(Encoder_ChB);
 if (aState == HIGH)                    //Check state of Channel A
 {
  if (bState == LOW)                  //check state of Channel B
  {
   EncoderPos += 1;                    //Clockwise
  }
  else
  {
   EncoderPos -= 1;                //CounterCLockwise
  }
 }
 else
 {
  if (bState == LOW)
  {
   EncoderPos -= 1;                 //CounterClockwise
  }
  else
  {
   EncoderPos += 1;         //Clockwise
  }
  // delay(500);
 }
}
```

2.1.2 Challenges Faced

I got the encoder value of the code using the above value. Now I had to control the velocity as well as the position (angle of rotation) of the DC motor by implementing the PID controller. On searching through various Arduino PID library on the internet (http://playground.arduino.cc/ Code/ PIDLibrary). I implemented the library but the output on my motor was haywire. The motor would not behave in a controlled way as it was supposed to. There were jerky knocks on the motor and hence I decided to go deeper as to why it was behaving this way.



I implemented my own PID control. The PID equation is as follows:

Control= **P**roportional term+ **I**ntegral Term+ **D**ifferential Term

With the assumption that the integral and differential term had **DT** (Derror/Dt) and (integral (e.DT)). The libraries available on the internet have the assumption that DT would be constant and hence the control output would behave erratically in some cases. So I finally decided to implement the entire PID by myself. The following is my code for both position and velocity PID.

// Implementing PID control on position

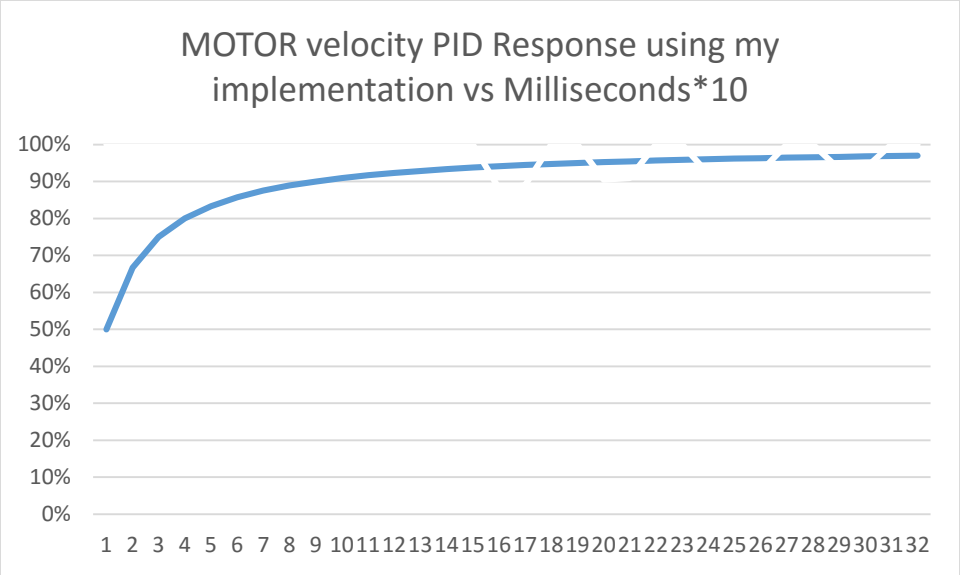
```
int PIDcontrol(int DesiredPos, int CurrentPos)
{
  int Error = DesiredPos - CurrentPos;
  int Timenow = millis();
  int TimeDiff = Timenow - Timelast;
  int ErrorDiff = Error - Errorlast;

  int PControl = Kp * Error;
  ErrorSum += (Error * TimeDiff);
  int IControl = Ki * ErrorSum;
  int DControl = Kd * ErrorDiff / TimeDiff;
  int Control = PControl + IControl + DControl;
  Timelast = Timenow;
  Errorlast = Error;
  return Control;

}
```

```
//Implementing PID control for Motor speed
int PIDVelControl(int DesiredVel, int CurrentVel)
{
  int Errorv = DesiredVel - CurrentVel;
  int Timenowv = millis();
  int TimeDiffv = Timenowv - Timelastv;
  int ErrorDiffv = Errorv - Errorlastv;
  int PControlv = Kpv * Errorv;
  Errorsumv += Errorsumv * TimeDiffv;
  int IControlv = Kiv * Errorsumv;
  int DControlv = Kdv * ErrorDiffv / TimeDiffv;
  int Controlv = PControlv + IControlv + DControlv;
  Timelastv = Timenowv;
  Errorlastv = Errorv;
  return Controlv;
}
```

Tweaking values of Kp , Ki and Kd I found the value at which the system was able to behave properly. This way of approaching the PID controller was advised to me by Rohan who told me that if you use the library function then they would be like a black box to you where in case any problem occurred you would not know why. It was his persuasion and strong valid reasoning that I decided to learn and implement the PID control on my own and I am grateful that I took that decision. My system worked perfectly fine.

The other challenge we faced was during the integration of the parts. Since the system had one interrupt that was controlling the state of the machine and the other interrupt as the encoder of the DC motor, there were concurrency issues. I tried introducing delays in the interrupts but no matter how much delay I introduced there were times when the Push button state interrupt would not respond when the dc motor was running. In this case I came up with the idea of using the concept from Operating Systems course I took in my undergrad which was not only effective but also foolproof. The concept involved using semaphores which is nothing but a variable that is globally initialized and would not let another function run if there is already a function running. I introduced a variable count which was initialized as 0 but when dc motor encoder would take over it would assume the value of 1 and when done, it would assume the value of 0 again. This method was much better and reliable than using the delays of random values.

The value from slot sensor was read and mapped to a scale of 0-255. This was then entered as the desired velocity of the motor in clockwise direction (the direction of the motor was set by me although the direction function was made separately for the GUI.



100% is the desired the velocity of the Motor. The Kp, Ki and Kd values are 2, 0.01 and 0.01 respectively.

The motor gave 360 encoder ticks or counts for one entire revolution.

I got the following readings

Angle Rotation	**Encoder Value**	**Angle Rotated by PID (clockwise) Approximately**	**Angle Rotated by PID (anticlockwise) approx**
180	195	180	170
360	378	360	330
540	590	540	530
720	780	720	690
800	860	800	750
980	1200	980	940

According to the values I got above, I used the value of 360 ticks per revolution as average and in my PID I added a constant term based on my experiments to make up for the extra values. The difference in values in the anticlockwise direction were less than the clockwise rotation so I added another term in the error because the differential term was positive and the proportional and integral term were negative and hence not equal to the clockwise counterpart.

3.0 Team Work

Apart from me, Gauri worked on controlling stepper motor using force sensor. The more force you apply, the more the stepper motor would rotate. Rohit worked on controlling the Servo motor using ultrasonic distance sensor while Tiffany worked on controlling Servo motor using temperature sensor. Sida worked on developing Graphical User Interface for the team. Although we could not showcase the GUI but Sida gave her best. She tried her level best which I guarantee.

3.1 Future Work

We plan to utilize the system of servo motor in our pan tilt unit that controls the camera. We will follow the face pose or head pose of the person using this system. This Lab taught us more than what we taught. The lessons are enumerated as follows.

1. Start as early as possible because integration poses problems one cannot foresee.
2. Keep people on the same page on your progress.
3. Make your program as modular as possible so it is easy to modify.
4. Keep detailed documentation for future purposes.
5. Use good programming habits of commenting for future purposes.

The next week we plan to do as follows for progress review 1

1. Order the parts list for the project.
2. Make a CAD drawing for the Pan Tilt unit to be mounted on robot.
3. Familiarize with the INTRAFACE software for facial expression recognition using at least one camera.

4.0 References

1. https://en.wikipedia.org/wiki/DC_motor

2. http://www.cytron.com.my/p-spg30e-60k?keyword=spg30%2060%20k

3. http://www.nskelectronics.com/slot_sensor.html

CODE (DC Motor and Slot Sensor- Individual Code)

```
//initializing PIN constants
const int IN1 = 11; //two inputs for motor Driver L298N
const int IN2 = 12;
const int Encoder_ChA = 2;
const int Encoder_ChB = 3;   // Encoder Pins
const int IN3 = 10; //PWM for motorspeed
int EncoderPos = 0; // encoder count

// Initializing constants for Position Feedback PID
int DesiredPos = 720;
int CurrentPos = 0;
int ErrorSum = 0;
int Timelast = 0;
int Errorlast = 0;
int Kp = 2.0;
int Ki = 0.01;
int Kd = 0.01;
//Initializing Constants for Velocity FeedBack PID
int Timelastv = 0;
int DesiredVel = 100*300/255;
int Errorsumv = 0;
int Errorlastv = 0;
int Kpv = 2;
int Kiv = 0.01;
int Kdv = 0.01;

void stopmotor();
void moveclockwise();
void moveanticlockwise();
void doEncoder();



void setup() {
 // put your setup code here, to run once:
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(Encoder_ChA, INPUT);
 digitalWrite(Encoder_ChA, HIGH);      // turn on pullup resistor
 pinMode(Encoder_ChB, INPUT);
 digitalWrite(Encoder_ChB, HIGH);      // turn on pullup resistor
 digitalWrite(IN3, 80);
 attachInterrupt(0, doEncoder, CHANGE);  // encoder pin on interrupt 0 - pin 2
 Serial.begin (9600);
}

void loop()
{
    moveclockwise();
   //CurrentPos = EncoderPos;
  int PrevTime=0;
  int PrevPos=0;
  delay(3000);
  int Timenow=millis();
  int Vel= EncoderPos-PrevPos/(Timenow-PrevTime);
 while (DesiredVel> 0)
 {
```

```
   Serial.println(Vel);
   int Output = PIDVelControl(DesiredVel, Vel);
    moveclockwise;
    PrevTime= Timenow;
   PrevPos= EncoderPos;
   SetVelocity(Vel+Output);
   delay(1000);
  int Vel= EncoderPos-PrevPos/(millis()-PrevTime);
 }
 while (DesiredPos <0)
 {
   Serial.println(Vel);
   int Output = PIDVelControl(DesiredVel,Vel);
   moveanticlockwise;
   PrevTime= Timenow;
   PrevPos= EncoderPos;
   SetVelocity(Vel+Output);
   delay(1000);
  int Vel= EncoderPos-PrevPos/(millis()-PrevTime);
  }
 delay(200);
 Serial.println(EncoderPos);


}



void moveanticlockwise()
{
 digitalWrite(IN1, HIGH);
 digitalWrite(IN2, LOW);
}


void moveclockwise()
{
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
}
void stopmotor()
{
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
}



//Function that activates on interrupts to check and calculate the encoder value.. Positive for clockwise and negative for anticlockwise

void doEncoder()
 {
 int aState = digitalRead(Encoder_ChA);
 int bState = digitalRead(Encoder_ChB);
 if (aState == HIGH)      //Check state of Channel A
 {
   if (bState == LOW)    //check state of Channel B
   {
     EncoderPos += 1; //Clockwise
   }
```

```
   else
   {
    EncoderPos -= 1;  //CounterCLockwise
   }
  }
  else
  {
   if (bState == LOW)
   {
    EncoderPos -= 1; //CounterClockwise
   }
   else
   {
    EncoderPos += 1;  //Clockwise

   }
   // delay(500);
  }
}


//Function to set MotorSpeed
void SetVelocity (int vel)
{
 analogWrite (IN3, vel);
}


void SetDirection (int dir)
{
 if (dir == 0)
 {
  moveclockwise();
 }
 else
 {
  moveanticlockwise();
 }
}

int MotorControl(int vel, int dir)
{
 SetVelocity(vel);
 SetDirection(dir);
}
// Implementing PID control on position
int PIDcontrol(int DesiredPos, int CurrentPos)
{

 int Error = DesiredPos - CurrentPos;
 int Timenow = millis();
 int TimeDiff = Timenow - Timelast;
 int ErrorDiff = Error - Errorlast;

 int PControl = Kp * Error;
 ErrorSum += (Error * TimeDiff);
 int IControl = Ki * ErrorSum;
 int DControl = Kd * ErrorDiff / TimeDiff;
 int Control = PControl + IControl + DControl;
 Timelast = Timenow;
 Errorlast = Error;
```

```
 return Control;

}

//Implementing PID control for Motor speed
int PIDVelControl(int DesiredVel, int CurrentVel)
{

 int Errorv = DesiredVel - CurrentVel;
 int Timenowv = millis();
 int TimeDiffv = Timenowv - Timelastv;
 int ErrorDiffv = Errorv - Errorlastv;
 int PControlv = Kpv * Errorv;
 Errorsumv += Errorsumv * TimeDiffv;
 int IControlv = Kiv * Errorsumv;
 int DControlv = Kdv * ErrorDiffv / TimeDiffv;
 int Controlv = PControlv + IControlv + DControlv;
 Timelastv = Timenowv;
 Errorlastv = Errorv;
 return Controlv;
}



void PIDposition(int DesiredPos)
{
 //CurrentPos = EncoderPos;
 stopmotor();
 while (DesiredPos - EncoderPos > 0)
 {
   int Output = PIDcontrol(DesiredPos, EncoderPos);

   Serial.println(DesiredPos - EncoderPos);
   if (Output > 0)
   {
     if (Output > 255)
     {
       SetVelocity(255);
       moveclockwise();
     }
     else
     {
       SetVelocity(Output);
     }
   }
 }
 while (EncoderPos - DesiredPos > 0)
 {
   int Output = PIDcontrol(DesiredPos-110, EncoderPos);
   if (Output < -255)
   {
     SetVelocity(255);
     moveanticlockwise();
   }
   else
   {
     SetVelocity(Output);
     moveanticlockwise();
   }
 }
```

```
 delay(200);
 stopmotor();
 Serial.println(EncoderPos);
}
```

Code of Other Members + My code Combined:

```
#include <Servo.h>
Servo myservo;
// For Sensor 1(sonar)
int pos=0;
int pos1=0;
//int ledPin=13;
float Reading=0;
float Reading_ft=0;


//initializing PIN constants for dc encoder
const int IN1 = 12; //two inputs for motor Driver L298N
const int IN2 = 13;
const int Encoder_ChA = 3;
const int Encoder_ChB = 9;   // Encoder Pins
const int IN3 = 11; //PWM for motorspeed
int EncoderPos = 0; // encoder count

// Initializing constants for Position Feedback PID
//int DesiredPos = 720;
int CurrentPos = 0;
int ErrorSum = 0;
int Timelast = 0;
int Errorlast = 0;
int Kp = 2.0;
int Ki = 0.01;
int Kd = 0.01;

//Initializing Constants for Velocity FeedBack PID
int Timelastv = 0;
int DesiredVel = 100*300/255;
int Errorsumv = 0;
int Errorlastv = 0;
int Kpv = 2;
int Kiv = 0.01;
```

```
int Kdv = 0.01;

void stopmotor();
void moveclockwise();
void moveanticlockwise();
void doEncoder();

const int  buttonPin = 2;    // the pin that the pushbutton is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

// For Debouncing
long lastDebounceTime = 0;  // the last time the output pin was toggled
long debounceDelay = 150;

//For Sensor 2(temp)
int thermalSensorpin;
float thermalReading;

// For Sensor 3(force)
int fsrAnalogPin = A1; // FSR is connected to analog 1
int fsrReading;// the analog reading from the FSR resistor divider
int steps;

// For Sensor 4(slot)
int slotReading;
int angle_encode;

void setup() {
  myservo.attach(10);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A4, INPUT);
  pinMode(A5, INPUT);
  //pinMode(ledPin,OUTPUT);
//  digitalWrite(ledPin,HIGH);
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
```

```
  thermalSensorpin=A4;
  thermalReading= analogRead(thermalSensorpin);

  //dc encoder
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(Encoder_ChA, INPUT);
  digitalWrite(Encoder_ChA, HIGH);      // turn on pullup resistor
  pinMode(Encoder_ChB, INPUT);
  digitalWrite(Encoder_ChB, HIGH);      // turn on pullup resistor
  digitalWrite(IN3, HIGH);
  attachInterrupt(0, doEncoder, CHANGE);  // encoder pin on interrupt 0 - pin 2

  //Stepper
pinMode(6,OUTPUT); // Enable
  pinMode(5,OUTPUT); // Step
  pinMode(4,OUTPUT); // Dir
  digitalWrite(6,LOW); // Set Enable low

  Serial.begin(9600);
}


void loop()
{ // For Sensor 1
  Reading=analogRead(A0);
  Reading_ft=Reading*0.046;

  // For Sensor 2
  thermalSensorpin=A4;
  thermalReading= analogRead(thermalSensorpin);

//For Sensor 3
fsrReading = analogRead(fsrAnalogPin)/4;
steps=map(fsrReading,0,255,0,360)/1.8;

 //For Sensor 4
 slotReading=analogRead(A5);
 angle_encode=map(slotReading,0,1023,0,720);
```

```
 if (Reading_ft<=0.51)
 {
  Reading_ft=0;
 }

 // read the pushbutton input pin:
 buttonState = digitalRead(buttonPin);
 // compare the buttonState to its previous state
 if (buttonState != lastButtonState)
 {
    if (buttonState == HIGH)
  {
    buttonPushCounter++;
    Serial.println("on");
    Serial.print("number of button pushes:  ");
    Serial.println(buttonPushCounter);
  }
  else
  {
       Serial.println("off");
  }

 }
debounce(buttonPin);
lastButtonState = buttonState;

// Sensor Calculations
pos=19.5*Reading_ft+0.5;
pos1 = (thermalReading-15)*9;
    if (pos>180)
    {
     pos=180;
     }

 if (buttonPushCounter % 2 == 0)
 {  //digitalWrite(ledPin,HIGH);
  myservo.write(pos);
   Serial.println(pos);

      // Stepper
      for(int x = 0; x < steps; x++)
```

```
  {
  digitalWrite(4,HIGH);
  digitalWrite(5,HIGH); // Output high
  delayMicroseconds(500); // Wait 1/2 a ms
  digitalWrite(5,LOW); // Output low
  delayMicroseconds(500); // Wait 1/2 a ms
  }

     delay(150);
for(int x = 0; x < steps; x++)
  {
  digitalWrite(4,LOW);
  digitalWrite(5,HIGH); // Output high
  delayMicroseconds(500); // Wait 1/2 a ms
  digitalWrite(5,LOW); // Output low
  delayMicroseconds(500); // Wait 1/2 a ms
  }

     delay(150);


  }

  else if(buttonPushCounter % 2 != 0) {
    //servo-temp
//    digitalWrite(ledPin,LOW);
    myservo.write(pos1);
    delay(150);
    //dc-slot
    Serial.print("slot reading ");
    Serial.println(slotReading);
    PIDposition(360);
  }

}

void debounce(int button) {
  // read the state of the switch into a local variable:
  int reading = digitalRead(button);

  // If the switch changed, due to noise or pressing:
```

```
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;
    }
  }

  lastButtonState = reading;
}

void moveanticlockwise()
{
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
}


void moveclockwise()
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
}
void stopmotor()
{
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}



//Function that activates on interrupts to check and calculate the encoder value.. Positive for clockwise and negative for anticlockwise

void doEncoder()
  {
```

```
 int aState = digitalRead(Encoder_ChA);
 int bState = digitalRead(Encoder_ChB);
 if (aState == HIGH)       //Check state of Channel A
 {
  if (bState == LOW)     //check state of Channel B
  {
   EncoderPos += 1; //Clockwise
  }
  else
  {
   EncoderPos -= 1;  //CounterCLockwise
  }
 }
 else
 {
  if (bState == LOW)
  {
   EncoderPos -= 1; //CounterClockwise
  }
  else
  {
   EncoderPos += 1;  //Clockwise

  }
  // delay(500);
 }
}


//Function to set MotorSpeed
void SetVelocity (int vel)
{
 analogWrite (IN3, vel);
}


void SetDirection (int dir)
{
 if (dir == 0)
 {
  moveclockwise();
```

```
  }
  else
  {
    moveanticlockwise();
  }
}



int MotorControl(int vel, int dir)
{
  SetVelocity(vel);
  SetDirection(dir);
}
// Implementing PID control on position
int PIDcontrol(int DesiredPos, int CurrentPos)
{

  int Error = DesiredPos - CurrentPos;
  int Timenow = millis();
  int TimeDiff = Timenow - Timelast;
  int ErrorDiff = Error - Errorlast;

  int PControl = Kp * Error;
  ErrorSum += (Error * TimeDiff);
  int IControl = Ki * ErrorSum;
  int DControl = Kd * ErrorDiff / TimeDiff;
  int Control = PControl + IControl + DControl;
  Timelast = Timenow;
  Errorlast = Error;
  return Control;

}

//Implementing PID control for Motor speed
int PIDVelControl(int DesiredVel, int CurrentVel)
{

  int Errorv = DesiredVel - CurrentVel;
  int Timenowv = millis();
  int TimeDiffv = Timenowv - Timelastv;
```

```
 int ErrorDiffv = Errorv - Errorlastv;
 int PControlv = Kpv * Errorv;
 Errorsumv += Errorsumv * TimeDiffv;
 int IControlv = Kiv * Errorsumv;
 int DControlv = Kdv * ErrorDiffv / TimeDiffv;
 int Controlv = PControlv + IControlv + DControlv;
 Timelastv = Timenowv;
 Errorlastv = Errorv;
 return Controlv;
}




void PIDposition(int DesiredPos)
{
 if(DesiredPos<0)
 {
  while( DesiredPos-EncoderPos>10)
  {
    int Output = PIDcontrol(DesiredPos,EncoderPos);

     SetVelocity(Output);
     moveclockwise();
     delay(DesiredPos/0.36);
      if (DesiredPos-EncoderPos>50)
       {
        DesiredPos=EncoderPos;
        stopmotor();
 }}

 else
 {
  while(EncoderPos- DesiredPos>10)
  {
  int Output = PIDcontrol(DesiredPos,EncoderPos);

     SetVelocity(Output);
     moveclockwise();
     delay(DesiredPos/0.36);
      Serial.println(DesiredPos);
```

```
      Serial.println(EncoderPos);
        CurrentPos=EncoderPos;
        if (DesiredPos-EncoderPos<50)
        {
         DesiredPos=EncoderPos;


         stopmotor();
          //



        delay(50);
        }
  }}

  Serial.print(DesiredPos);
  while (EncoderPos - DesiredPos > 10)
  {
    int Output = PIDcontrol(DesiredPos-110, EncoderPos);
    if (Output < -255)
    {
      SetVelocity(255);
      moveanticlockwise();
    }
    else
    {
      SetVelocity(Output);
      moveanticlockwise();

    }
  }
  //delay(200);

  stopmotor();


}
```