AUTOMATIC CURTAIN

IBRAHIM BIN NASRUM (2116467)
MUHAMMAD HAFIDZUDDIN HANIF
DANIAL BIN NORIZAL (2123651)
CHUI YUAN JIE (2114803)
MUHAMAD AZRI BIN ABDUL AZIZ (2113537)

INTRODUCTION

The automatic curtain system is a smart and efficient solution designed to automate the opening and closing of curtains in homes, offices, or other spaces. It eliminates the need for manual operation, enhancing convenience and comfort while also offering energy-saving and security benefits.

In this project, the curtain's movement is controlled using a DC motor with an encoder, a motor driver, and an ultrasonic sensor. The system uses a PD (Proportional-Derivative) controller to ensure precise and smooth operation of the motor based on real-time feedback from the encoder and sensor.

PROBLEM STATEMENT

In many homes and offices, manually operating curtains can be inconvenient, especially in high or hard-to-reach windows. Traditional curtains lack automation and intelligent features, which limits their ability to adapt to environmental conditions such as sunlight or user preferences.
Additionally, the lack of precision in existing motorized curtain systems often results in inconsistent operation or excessive energy consumption. There is a need for a smart and efficient curtain system that can operate autonomously, offering convenience, energy efficiency, and improved user experience.

METHODOLOGY

OBJECTIVE DEFINITION: DESIGN AN AUTOMATIC CURTAIN SYSTEM THAT ADJUSTS BASED ON THE MEASURED DISTANCE (OR EXTERNAL INPUT) USING A DC MOTOR, AN ENCODER, A MOTOR DRIVER, AND AN ULTRASONIC SENSOR.

COMPONENT SELECTION:

- DC MOTOR WITH ENCODER: FOR PRECISE MOVEMENT CONTROL OF THE CURTAIN.
- MOTOR DRIVER: TO MANAGE THE MOTOR'S DIRECTION AND SPEED.
- ULTRASONIC SENSOR: TO MEASURE THE PROXIMITY OR INPUT DISTANCE FOR AUTOMATION.
- ARDUINO UNO: AS THE MICROCONTROLLER FOR PROCESSING INPUTS AND CONTROLLING OUTPUTS.

CONTROL ALGORITHM:

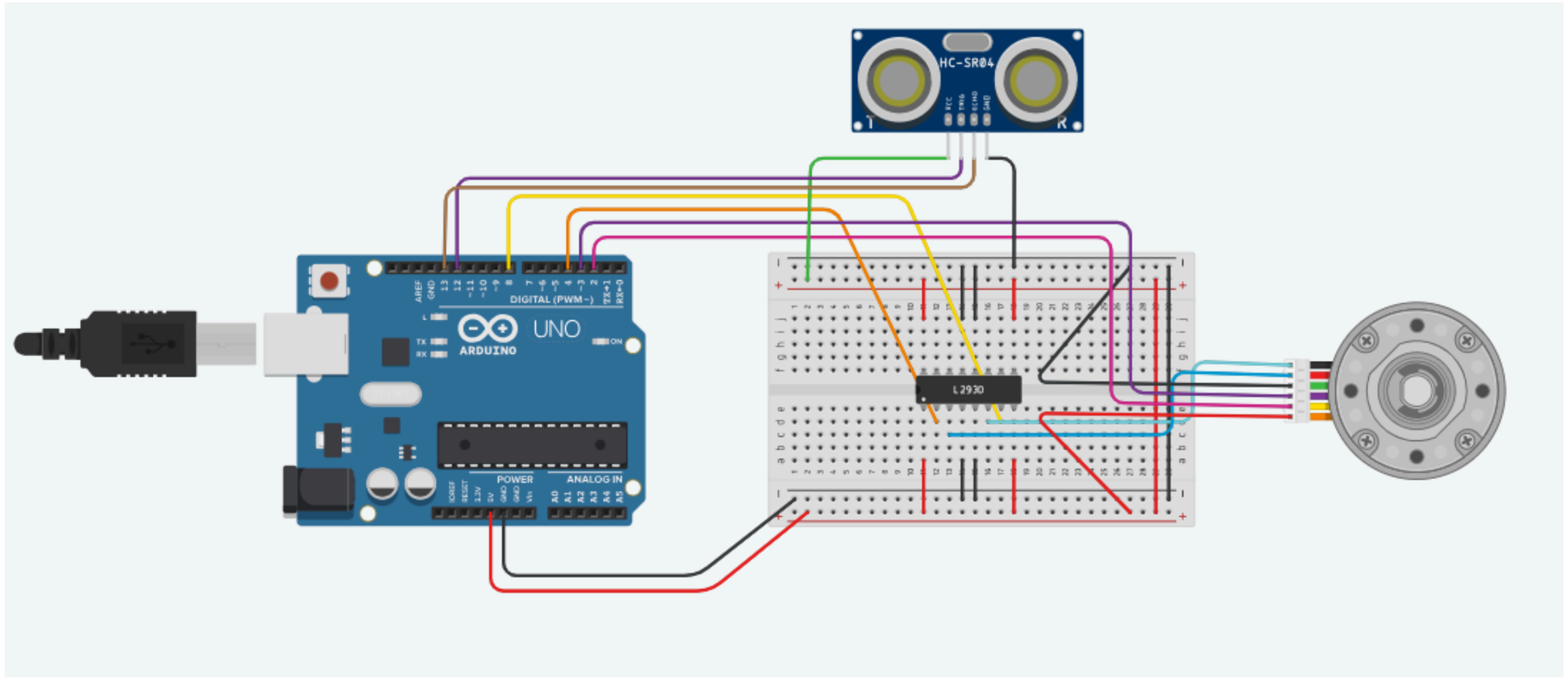
- IMPLEMENT A PID CONTROLLER TO MANAGE THE MOTOR’S POSITION AND ENSURE SMOOTH AND PRECISE MOVEMENT.

MECHANICAL DESIGN:

- DETERMINE HOW THE MOTOR AND ENCODER WILL CONNECT TO THE CURTAIN MECHANISM (E.G., PULLEY SYSTEM).

METHODOLOGY

CIRCUIT DIAGRAM



METHODOLOGY

CODE HIGHLIGHTS

1. INPUT: DISTANCE MEASURED BY THE ULTRASONIC SENSOR

```
float getDistance() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  long duration = pulseIn(ECHO, HIGH);
  float distance = (duration * 0.0343) / 2; // Convert to cm
  return distance;
}
```

- MEASURES THE DISTANCE TO AN OBJECT (E.G., USER'S HAND OR NEARBY OBSTACLE).
- CONVERTS THE MEASURED TIME OF THE ULTRASONIC PULSE INTO A DISTANCE (IN CENTIMETERS).

2. DISTANCE MAPPED TO A TARGET MOTOR POSITION.

```
// Read the distance from the ultrasonic sensor
float distance = getDistance(); // Get distance in cm
int target = map(distance, 0, 10, 0, 1200); // Map distance to position range
target = constrain(target, 0, 1200); // Ensure target is within valid range
```

CONVERTS THE PHYSICAL DISTANCE (0–10 CM) INTO A CORRESPONDING ENCODER POSITION RANGE (0–1200).

METHODOLOGY

CODE HIGHLIGHTS

3. FEEDBACK CONTROL WITH PID

```
// Error
int e = pos - target;

// PID constants
float kp = 0.5;
float kd = 0.017;
float ki = 0.0;

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

- CALCULATES THE DIFFERENCE BETWEEN THE CURRENT MOTOR POSITION (POS) AND THE DESIRED TARGET POSITION (TARGET)

- PROPORTIONAL (KP * E): RESPONDS TO THE MAGNITUDE OF THE ERROR TO ADJUST MOTOR MOVEMENT.
- DERIVATIVE (KD * DEDT): REACTS TO THE RATE OF CHANGE OF THE ERROR TO DAMPEN OSCILLATIONS.

METHODOLOGY

CODE HIGHLIGHTS

4. MOTOR CONTROL

```
// 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);
```

- THE CONTROL SIGNAL (U) IS TRANSLATED INTO MOTOR POWER (PWR) AND DIRECTION (DIR
- MOTOR POWER IS CONSTRAINED TO THE VALISENDS APPROPRIATE SIGNALS TO THE MOTOR DRIVER TO CONTROL THE MOTOR'S SPEED AND DIRECTIOND PWM RANGE (0–255).
- SENDS APPROPRIATE SIGNALS TO THE MOTOR DRIVER TO CONTROL THE MOTOR'S SPEED AND DIRECTION:

METHODOLOGY

CODE HIGHLIGHTS

5. POSITION TRACKING WITH ENCODER

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

- TRACKS MOTOR POSITION BY INCREMENTING OR DECREMENTING THE POSI VARIABLE BASED ON THE ENCODER SIGNALS
- USES INTERRUPTS FOR REAL-TIME POSITION UPDATES.

6. REAL-TIME PROCESSING

```
// Time difference
long currT = micros();
float deltaT = ((float)(currT - prevT)) / (1.0e6);
prevT = currT;

// Read the position in an atomic block to avoid misreads
int pos = 0;
ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
  pos = posi;
}
```

- THE TIME DIFFERENCE (DELTAT) IS CALCULATED USING MICROS() TO ENSURE PRECISE CONTROL\
- ENSURES THE POSITION VARIABLE (POSI) IS READ SAFELY IN CRITICAL SECTIONS USING ATOMIC_BLOCK TO PREVENT CORRUPTION

METHODOLOGY

CODE HIGHLIGHTS

7. SERIAL PLOTTER OUTPUT

```
// Print data for Serial Plotter
Serial.print("Target:");
Serial.print(target);
Serial.print("\t");

Serial.print("Position:");
Serial.print(pos);
Serial.print("\t");

Serial.print("Distance:");
Serial.println(distance);
```

OUTPUTS DATA SUCH AS TARGET POSITION, CURRENT POSITION, AND DISTANCE FOR REAL-TIME VISUALIZATION

METHODOLOGY

CODING

```
#include <util/atomic.h> // For the ATOMIC_BLOCK macro

#define ENCA 2 // YELLOW
#define ENCB 3 // WHITE
#define PWM 9
#define IN2 8
#define IN1 4
#define TRIG 12
#define ECHO 13

volatile int posi = 0;
long prevT = 0;
float eprev = 0;
float eintegral = 0;

// Function to read distance from ultrasonic sensor
float getDistance() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  long duration = pulseIn(ECHO, HIGH);
  float distance = (duration * 0.0343) / 2; // Convert to cm
  return distance;
}
```

```
}

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

  // Initialize encoder pins
  pinMode(ENCA, INPUT);
  pinMode(ENCB, INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCA), readEncoder, RISING);

  // Initialize motor driver pins
  pinMode(PWM, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  // Initialize ultrasonic sensor pins
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);

  Serial.println("target pos");
}

void loop() {
  // Read the distance from the ultrasonic sensor
  float distance = getDistance(); // Get distance in cm
  int target = map(distance, 0, 10, 0, 1200); // Map distance to position range
  target = constrain(target, 0, 1200); // Ensure target is within valid range

  // PID constants
  float kp = 0.5;
  float kd = 0.017;
  float ki = 0.0;

```

CONTINUE .. NEXT SLIDE

METHODOLOGY

CODING

```
59
60  // Time difference
61  long currT = micros();
62  float deltaT = ((float)(currT - prevT)) / (1.0e6);
63  prevT = currT;
64
65  // Read the position in an atomic block to avoid misreads
66  int pos = 0;
67  ATOMIC_BLOCK(ATOMIC_RESTORESTATE) {
68    pos = posi;
69  }
70
71  // Error
72  int e = pos - target;
73
74  // Derivative
75  float dedt = (e - eprev) / (deltaT);
76
77  // Integral
78  eintegral = eintegral + e * deltaT;
79
80  // Control signal
81  float u = kp * e + kd * dedt + ki * eintegral;
82
83  // Motor power
84  float pwr = fabs(u);
85  if (pwr > 255) {
86    pwr = 255;
87  }
```

```
88
89  // Motor direction
90  int dir = 1;
91  if (u < 0) {
92    dir = -1;
93  }
94
95  // Signal the motor
96  setMotor(dir, pwr, PWM, IN1, IN2);
97
98  // Store previous error
99  eprev = e;
100
101 // Print data for Serial Plotter
102 Serial.print("Target:");
103 Serial.print(target);
104 Serial.print("\t");
105
106 Serial.print("Position:");
107 Serial.print(pos);
108 Serial.print("\t");
109
110 Serial.print("Distance:");
111 Serial.println(distance);
112
113 delay(100); // Small delay to make the plot smoother (optional)
114 }
```

CONTINUE .. NEXT SLIDE

METHODOLOGY

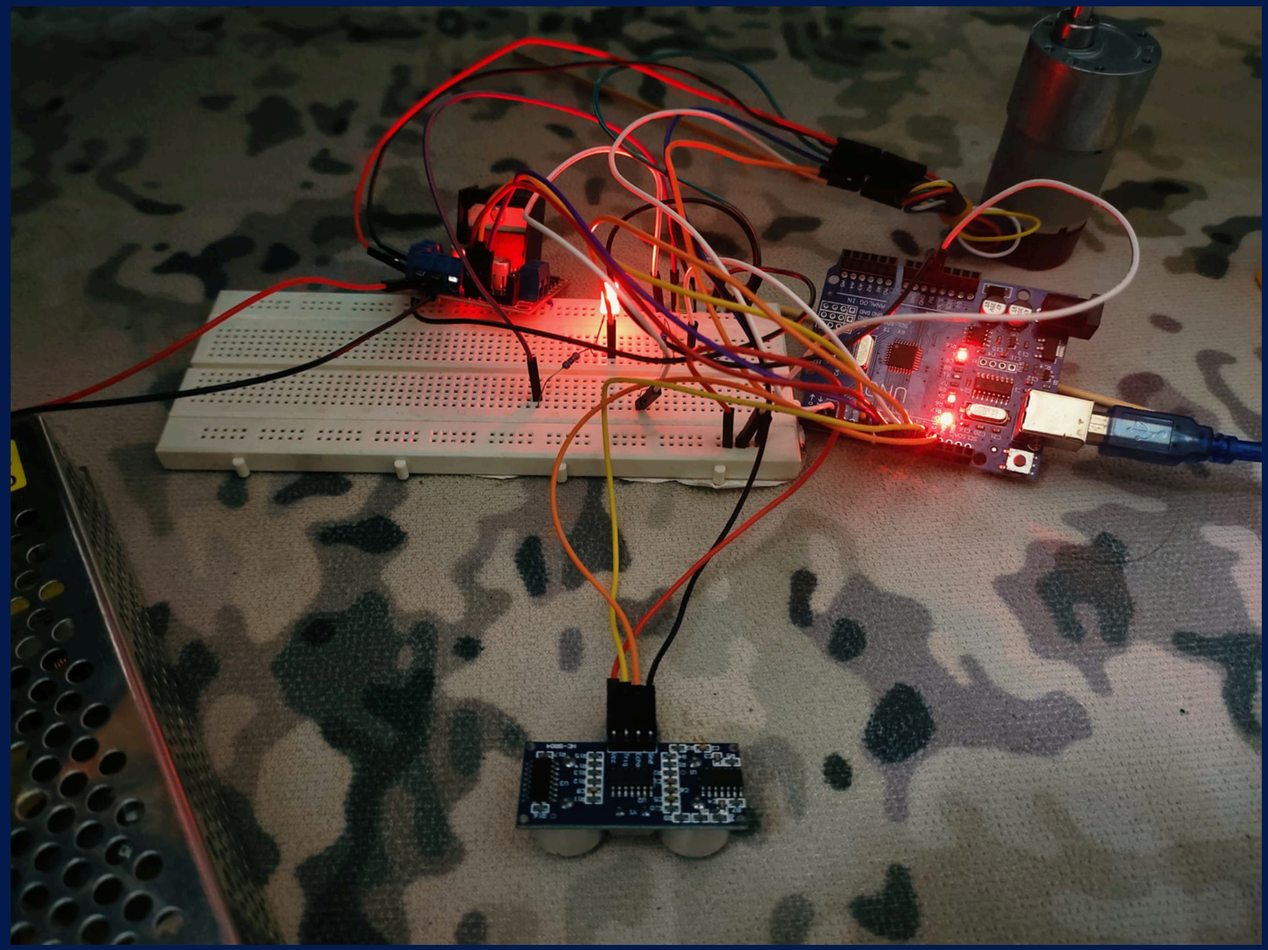
CODING

```
116 void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
117   analogWrite(pwm, pwmVal);
118   if (dir == 1) {
119     digitalWrite(in1, HIGH);
120     digitalWrite(in2, LOW);
121   } else if (dir == -1) {
122     digitalWrite(in1, LOW);
123     digitalWrite(in2, HIGH);
124   } else {
125     digitalWrite(in1, LOW);
126     digitalWrite(in2, LOW);
127   }
128 }
129
130 void readEncoder() {
131   int b = digitalRead(ENCB);
132   if (b > 0) {
133     posi++;
134   } else {
135     posi--;
136   }
137 }
```

END OF CODING

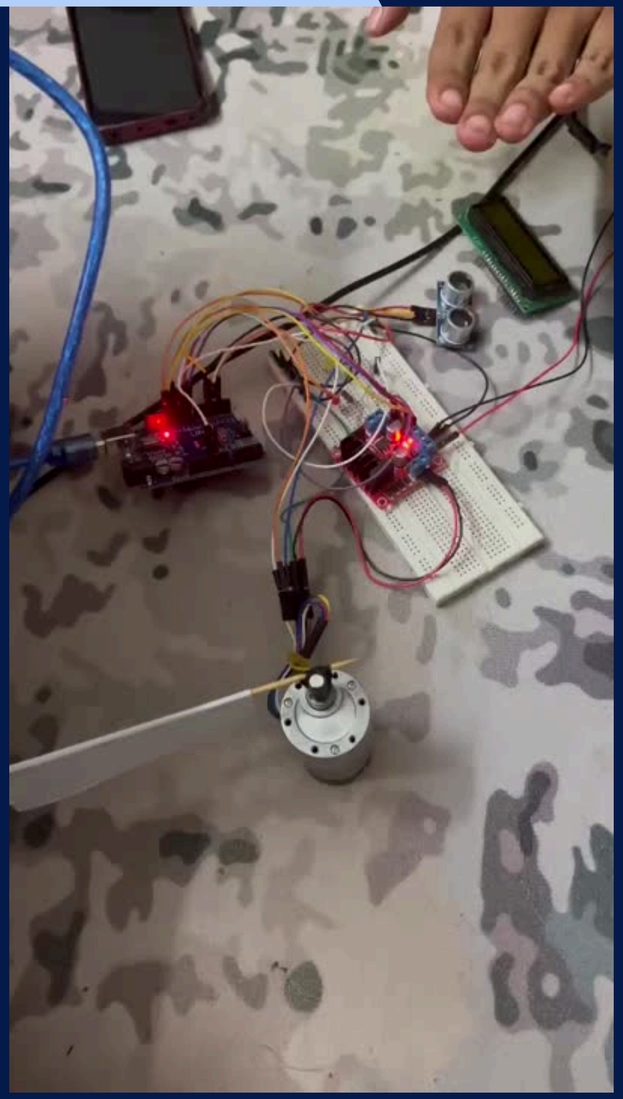
RESULTS

PICTURE



RESULTS

VIDEO



RESULTS

ORIGINAL KP AND KD
- KP = 0.8
- KD = 0.15

VERY LARGE JERK MOTION WHEN OBJECT IS WITHIN THE RANGE OF ULTRASONIC SENSOR

Target:1200	Position:1247	Distance:79.64
Target:1200	Position:1247	Distance:79.75
Target:1200	Position:1247	Distance:79.76
Target:1200	Position:1247	Distance:80.06
Target:1200	Position:1247	Distance:80.47
Target:1200	Position:1247	Distance:80.98
Target:1200	Position:1247	Distance:80.12
Target:1200	Position:1247	Distance:80.88
Target:1200	Position:1247	Distance:80.88
Target:1200	Position:1247	Distance:80.00
Target:1200	Position:1247	Distance:80.00
Target:1200	Position:1247	Distance:80.11

Target:360	Position:367	Distance:3.46
Target:360	Position:367	Distance:3.46
Target:360	Position:367	Distance:3.46
Target:360	Position:367	Distance:3.86
Target:360	Position:367	Distance:3.41
Target:360	Position:367	Distance:3.86
Target:360	Position:367	Distance:3.76
Target:360	Position:367	Distance:3.86
Target:360	Position:367	Distance:3.45
Target:360	Position:367	Distance:3.45

RESULTS

LOWER KP AND HIGHER KD

- KP = 0.5
- KD = 0.17

SMALLER JERK MOTION AND SLOWER ROTATION

DISCUSSION

COMPARISON TABLE

Feature	Servo Motor	DC Motor + Encoder + PID
Control	Simple, angle control	Complex, full speed/position control
Feedback	Built-in	External encoder provides feedback
Range of Motion	Limited (0–180° or 360°)	Continuous
Precision	High for small angles	High for position and speed control
Complexity	Low	High
Flexibility	Limited to fixed tasks	Suitable for dynamic tasks

// Dc motor + Encoder + PID is better than servo motor because it offers greater flexibility,precision, and continuous rotation capability, making it better suited for dynamic and advanced motion control applications.

DISCUSSION

CALCULATION :

1. ULTROSONIC SENSOR DISTANCE MEASUREMENT

1. Ultrasonic sensor distance measurement

$$\text{Distance (cm)} = \frac{\text{Duration } (\mu s) \times 0.0343}{2}$$

speed of sound air approximately = 343 m/s or 0.0343 cm/μs

assume duration = 600 μs // time/duration for the ultrasonic sensor emit frequency (pulse) and reflect back the pulse.

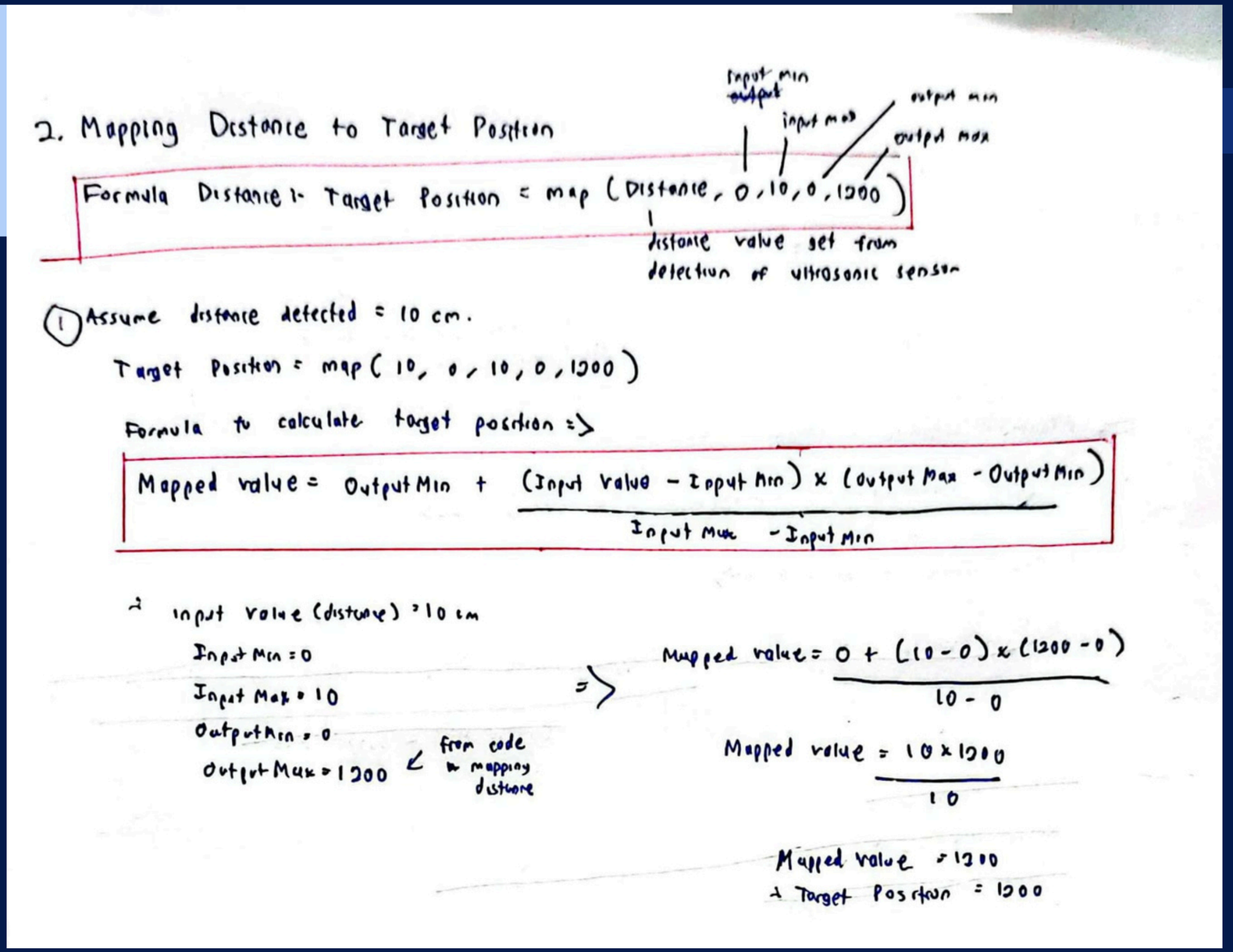
$$\therefore \text{Distance} = \frac{600 \times 0.0343}{2} = 10.29 \text{ cm}$$

∴ The object is approximately 10.29 cm away from sensor.

This value of distance is function to give accurate distance measurement for mapping position.

DISCUSSION

CALCULATION :
2. MAPPING DISTANCE TO TARGET POSITION



DISCUSSION

CALCULATION :
2. MAPPING DISTANCE TO TARGET POSITION

(2) Assume second distance detected = 8 cm

Target Position = map (8, 0, 10, 0, 1200)

∴ input value (distance) = 8 cm

Input Min = 0

Input Max = 10

Output Min = 0

Output Max = 1200

=>

$$\text{Mapped value} = \frac{0 + (8-0) \times (1200-0)}{10-0}$$

$$\text{Mapped value} = \frac{8 \times 1200}{10}$$

$$= \frac{9600}{10}$$

Mapped value = 960

∴ Target position = 960

DISCUSSION

CALCULATION :

3. PID CONTROLLER

3. PID Controller Calculation

Assume : 960 => Previous position

1200 => target position

1. Previous error = Target Position - Previous Position

= 1200 - 960 = 240

2. Current error (if we assume current position measured by encoder = 1000)

∴ Current error (e) = Target Position - Current Position

e = 1200 - 100 = 200

3. Derivative term (dedt) => how fast error change

$$dedt = \frac{e - \text{Previous error}}{\Delta T}$$

Given ΔT = 0.01 second (time difference)

$$\therefore dedt = \frac{200 - 240}{0.01} = -4100$$

DISCUSSION

CALCULATION :

3. PID CONTROLLER

4. Integral term (I) : accumulated past error

$$I = \text{Previous integral} + e \times \Delta T$$

$$I = 0 + 200 \times 0.01 = 2$$

5. Control signal

$$u = k_p \cdot e + k_d \cdot \frac{de}{dT} + k_I \cdot \int e\,dt$$

(de/dt and I)

From the code we type the

$k_p = 0.5$

$k_d = 0.017$

$k_I = 0.0$

$$\therefore u = (0.5 \cdot 200) + (0.017 \times (-4000)) + (0.0 \times 2)$$

$$u = 100 - 68 + 0 = 32$$

6. Motor pwm (pwr) is the motor power is the absolute value of control signal.

$$pwr = |u| = 32$$

u = positive => motor will rotate positive direction

DISCUSSION

PID CALCULATION SUMMARY:

The PID calculation helps control the motor's movement towards a target position, adjusting its speed and direction based on the following components:

- Proportional (P): Corrects the motor's position based on the current error (difference between target and actual position). A larger error results in a larger correction.
- Integral (I): Addresses persistent, small errors over time, eliminating steady-state errors that might not be corrected by the proportional term alone.
- Derivative (D): Prevents overshooting by adjusting the motor's response based on the rate of change of the error.

DISCUSSION

USING TARGET AND CURRENT POSITION

- Given a target position (e.g., 1200) and current position (e.g., 960), the PID calculation produces a control signal (u) that determines the motor's power and direction.
- The error is calculated as the difference between the target and the current position.
- Proportional term adjusts based on the error, integral accounts for cumulative errors over time, and derivative reduces overshoot.

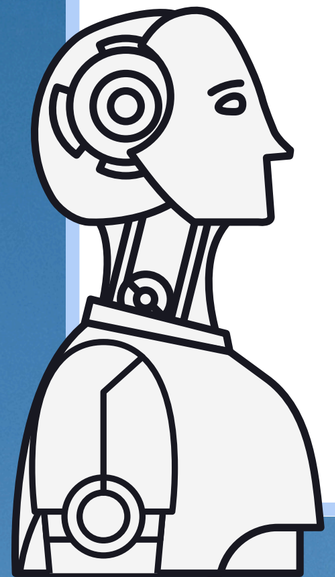
DISCUSSION

SUMMARY FOR THE CALCULATION

In summary, to help the motor reach the desired position smoothly and precisely, the PID controller produces a control signal (uu) that modifies the motor's power and direction. The error, or the difference between the goal position (for example, 1200) and the current position (for example, 960), is first calculated. For quick corrections, the proportional term (Kp) responds to the error's magnitude. While the derivative term (Kd) anticipates and avoids overshooting by taking into account how quickly the mistake is changing, the integral term (Ki) fixes any long-term deviations from the objective by examining previous errors. Together, these three components guarantee steady and accurate motion in the direction of the objective.

CONCLUSION

Based on this results, this project is a success as it highlighted the impact of PD controller on the control system. When Kp is lower and Kd is higher, it lowers the overshoot and stabilize the system as demonstrated. A PID controller is useful in ensuring precise and smooth control especially in application that requires precise control like . This valueable technology can be used to better the condition of humankind thus improving robotics, medical industry and automotive.




THANK YOU.