DC Motor Position Control Using PID

- K M Sudarshan, 170108021, 9182832825
- R Vinay Kumar, 170108031, 7901391330
- Ch Naga Sri Hari, 170108011, 9182908063

Contents

Introduction

Nowadays, DC motors are used in various applications such as robotics, industries, etc. Because of their simplicity, ease of application, reliability and cost effectiveness. Generally, in DC motors, speed control can be achieved by varying the terminal voltage but position control of the shaft cannot be achieved.

The position control of DC motors is crucial in the applications for precise control systems such as Antenna positioning system. In this project we are designing an **Antenna positioning control system** by controlling the position of DC motors using PID.

Project Description

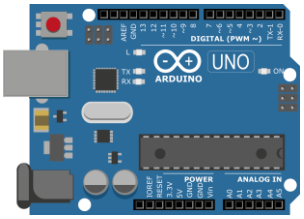
The outdoor antenna is subjected to many external disturbances which cause the deviation of the antenna from the desired position and the process of driving it back precisely are the main challenging issues of the control system designed for such a purpose.

The objective of the project is to design antenna position control system that has the ability to drive the antenna towards its desired position as well as holding it stable on that position by providing a high resistance to external mechanical disturbances.

Hardware devices Used

1. Arduino UNO
2. Motor Driver (L298N)
3. DC gear motor (2 no's)
4. Optical encoder
5. Bluetooth Module (HC-05)
6. Battery (9V)

Arduino UNO



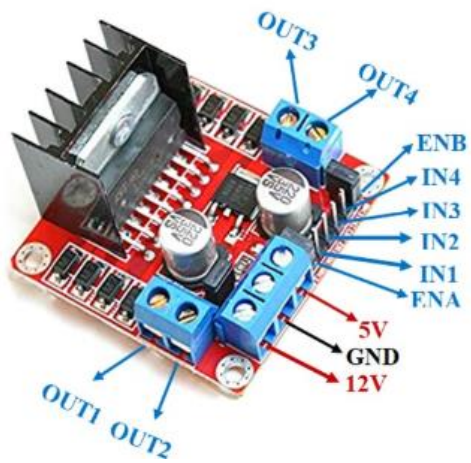
Arduino UNO is a microcontroller board shown in the figure. It has 14 input/output pins of which 6 can be used as PWM outputs, 6 analog pins, a 6MHz quartz crystal, a USB connection, a power jack, an ICSP header and reset button.

Here in our project, the Arduino micro controller is used to drive the PWM signal for DC motor for the improvement of output control of DC motor.

Motor Driver (L298N)

The L298N H-bridge motor driver is shown in the figure below is used to control the speed and direction of two DC motors. This module can be used with motors that have a voltage of between 5V and 35V DC and with a peak current up to 2A. The pin assignments of the motor driver are shown in the table.

The pins IN1 and IN2 are used to control the direction (clockwise and anticlockwise) of the motor A and IN3 and IN4 pins for motor B. The Arduino controller output PWM signal is sent to ENA and ENB to control the position of the motor A and motor B respectively.



Out 1:	Motor A lead out
Out 2:	Motor A lead out
Out 3:	Motor B lead out
Out 4:	Motor B lead out
GND:	Ground
5V :	5V input
ENA:	Enables PWM signal for Motor A
IN1:	Enable Motor A
IN2:	Enable Motor A
IN3:	Enable Motor B
IN4:	Enable Motor B
ENB:	Enables PWM signal for Motor B

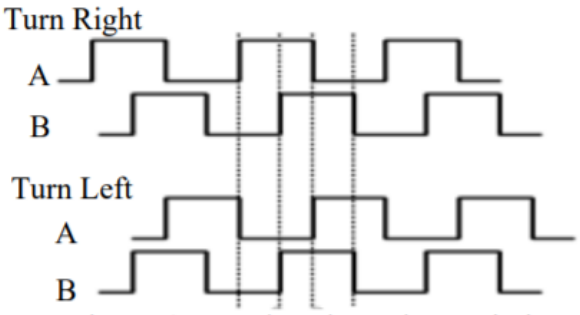
DC Gear Motor

The DC gear motor is shown in the figure below. The two terminals of one DC motor is connected to the Motor A pins of the motor driver and the other DC motor is connected to Motor B pins of the motor driver. The DC motor gets its signal from the motor driver and the IN1 and IN2 pins and the ENA pin of the motor driver decides the direction and speed of motor A respectively.

Optical Encoder

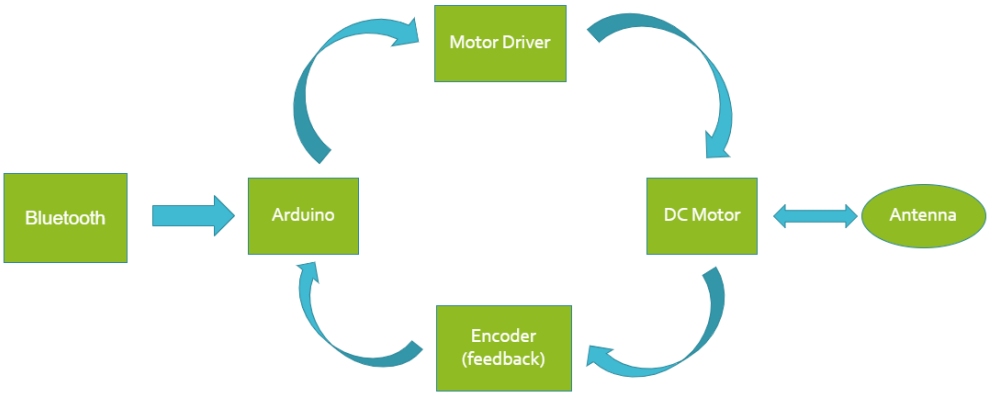
The optical encoder is shown in the figure below. The optical encoder has 4pins i.e., Encoder power, Encoder ground, Channel A and Channel B. The DC motor shaft and the optical encoder are coupled using a coupler so when the DC motor rotates the optical encoder also rotates and generates a series of pulses from Channel A and Channel B of the encoder. These pulses determine the direction of rotation of the motor and position of the motor. The series of pulses are shown below.

The significance of the optical encoder is it acts as a sensor which senses the position of the DC motor and the Channel A and Channel B are given as inputs to the Arduino controller.

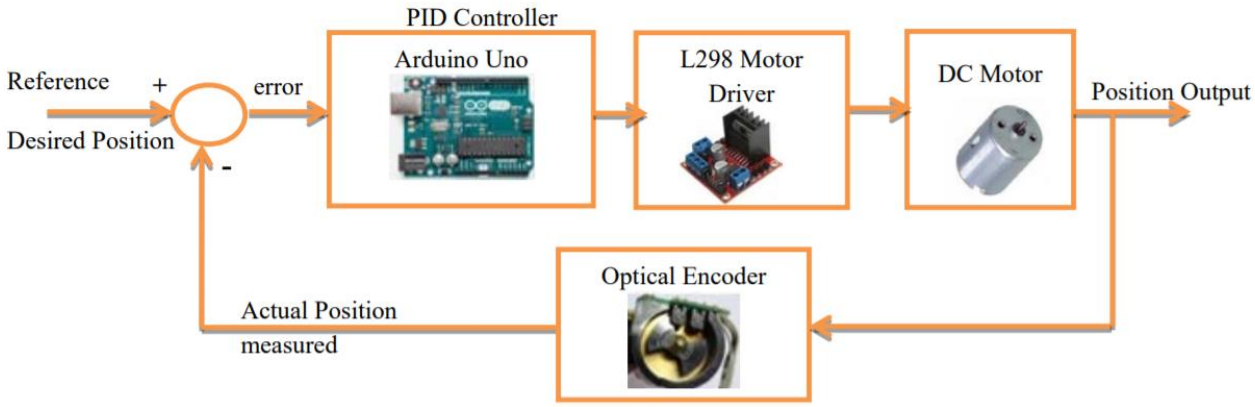


Circuit Diagram and Block Diagram

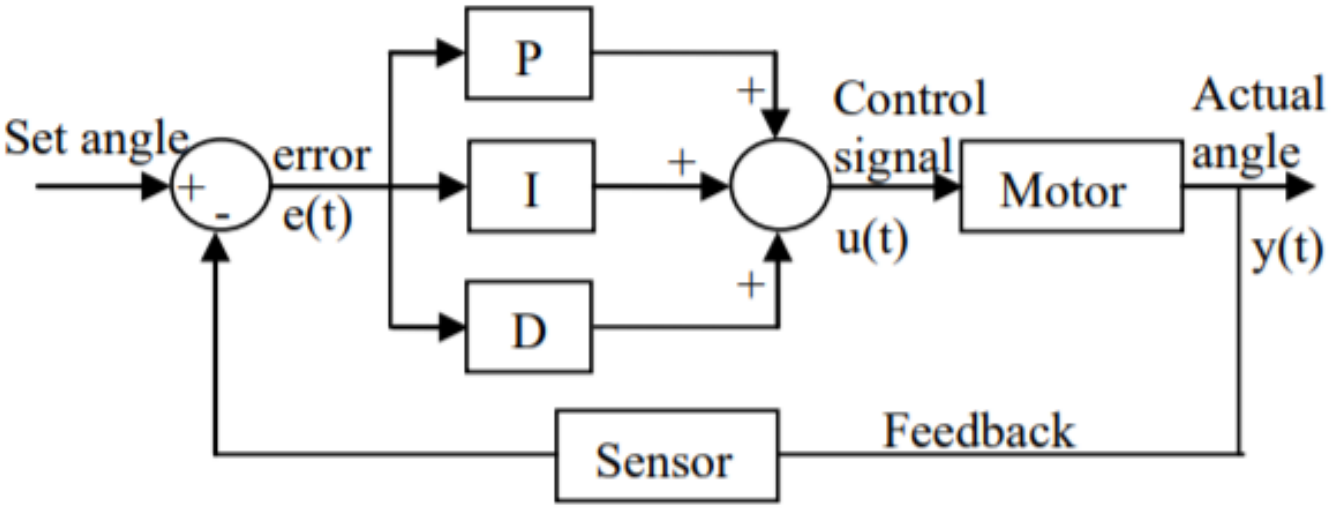
The overview of the model is shown below



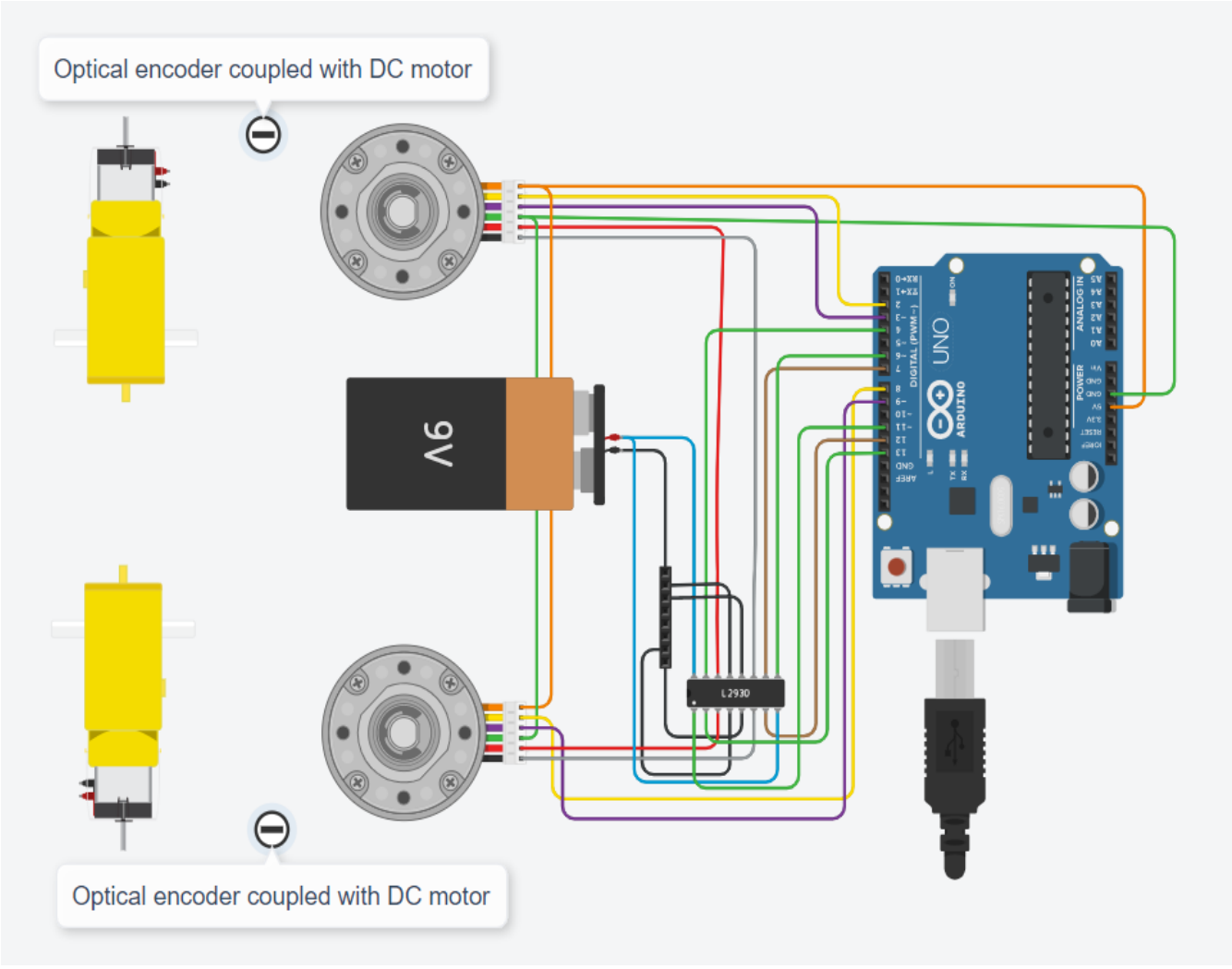
The control system block diagram is as shown in the figure below. Here the desired position of the DC motor is given as the input/reference. Arduino UNO acts as PID controller which sends the PWM signal to the L298N motor driver and this rotates the DC motor when the DC motor rotates the position of the DC motor is sensed by Optical encoder (Optical encoder acts as sensor) and returns the actual position to the Arduino controller, the Arduino controller generates an error signal(difference of desired position given from input and actual position sensed by optical encoder) and Arduino computes the PID and sends the computed PID output to the motor driver and this turns the motor to the desired position and this process repeats in every cycle and when the motor position is as close to the desired position then the error signal becomes equal to zero and now the Arduino stops feeding signal to the motor driver



This figure represents the closed loop position control of the DC motor using PID controller



This is the circuit we have designed using Tinker cad. Here the DC motor and optical encoder are coupled together so there is no requirement of any external coupler.

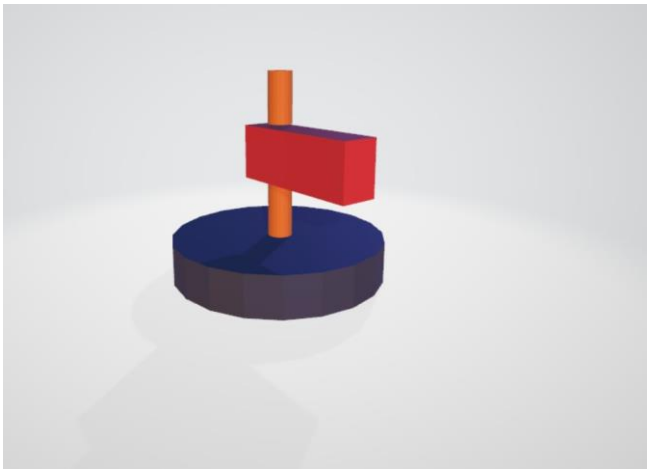


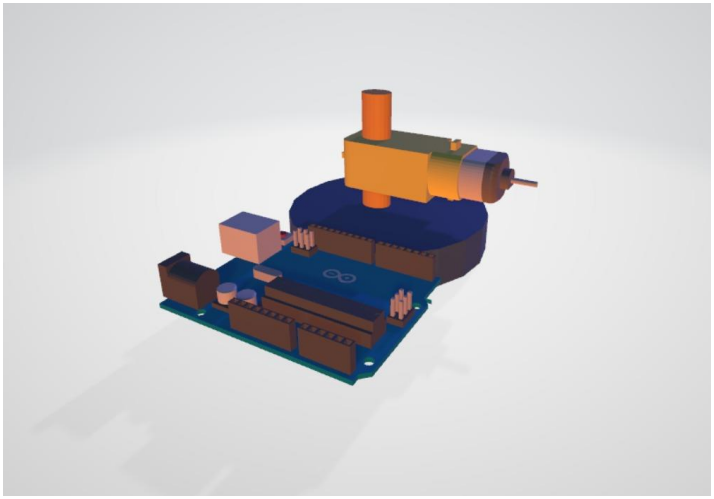
Setup and Design

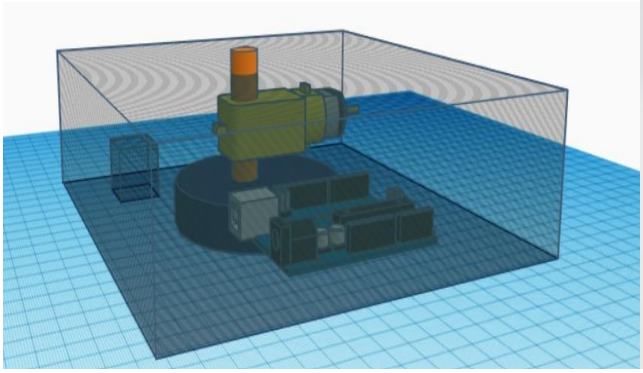
The design includes the base on which azimuthal rotation axis and all the system will be carried. The base is a box on the top of it the axes of the azimuthal motor passes, while inside it the azimuthal motor coupled with Optical encoder and the controller are placed.

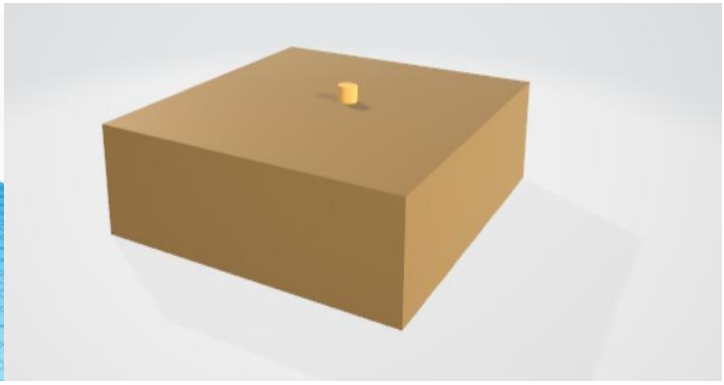
After the base and on the axis of azimuthal motor comes azimuth rotator, which is another circular base that carries the elevation motor and the antenna with its arm. This base rotates with the azimuth motor and with it the antenna rotates within a range of +/- 180 degrees. When the elevator motor rotates then antenna will move in the vertical plane. From this model for every position in the space we can have a corresponding position the antenna points out.

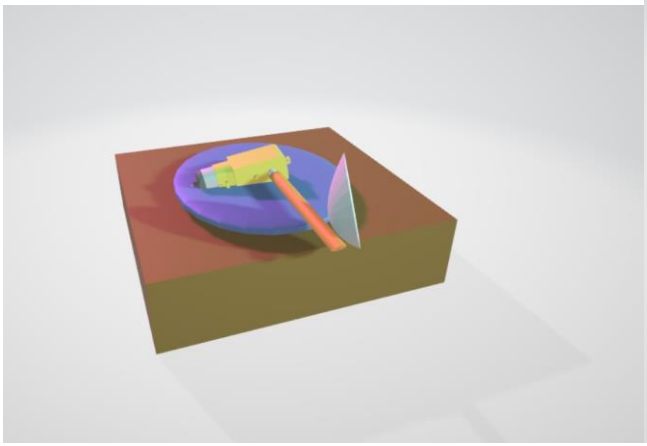
All the designs are shown in the figure below

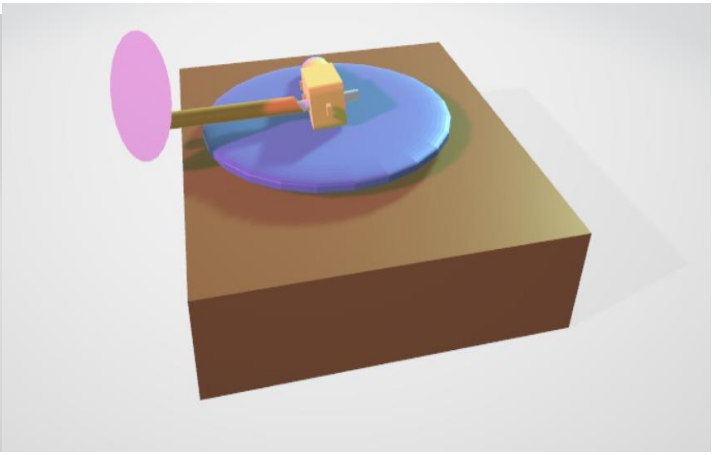












Working

The input angle is given from the android device using Bluetooth module (HC-05) which communicates between android device and Arduino. Arduino receives the input and generates PWM signal which drives the motor. When motor starts rotating Optical encoder senses and sends real time position feedback to Arduino which uses PID controlling algorithm and sends the output to the motor driver. When the motor reaches its desired position, Arduino stops sending the signal to the motor driver. Here we use PID to for smooth and clean motion and precise output.

Arduino Code

```
#include <PID_v1.h>       //Include PID library

#define MotorEnable 6     //Motor Enable pin Runs on PWM signal
#define MotorForward  4   // Motor Forward pin
#define MotorReverse  7   // Motor Reverse pin

String readString;        //stores the user input data
int User_Input = 0;       // converts input string into integer
int encoderPin1 = 2;      //Encoder Output 'A' must connected with interrupt pin of Arduino.
int encoderPin2 = 3;      //Encoder Otput 'B' must connected with interrupt pin of arduino.
volatile int lastEncoded = 0;       // updated value of encoder is stored.
volatile long encoderValue = 0;     // Raw encoder value
int PPR = 1600;           // Encoder Pulse per revolution.
int angle = 360;          // Maximum degree of motion.
int REV = 0;              // Set point 'REQUIRED ENCODER VALUE'
int lastMSB = 0;          // Input from encoderpin1
int lastLSB = 0;          // Input from encoderpin2
double kp = 5 , ki = 1 , kd = 0.01;             // modify for optimal performance
double input = 0, output = 0, setpoint = 0;

PID myPID(&input, &output, &setpoint, kp, ki, kd, DIRECT);

void setup() {
    pinMode(MotorEnable, OUTPUT);
    pinMode(MotorForward, OUTPUT);
    pinMode(MotorReverse, OUTPUT);
    Serial.begin(9600);     //initialize serial comunication

    pinMode(encoderPin1, INPUT_PULLUP);
    pinMode(encoderPin2, INPUT_PULLUP);

    digitalWrite(encoderPin1, HIGH);    //turn pullup resistor on
    digitalWrite(encoderPin2, HIGH);    //turn pullup resistor on

    //call updateEncoder() function when any high/low changed
    //seen on interrupt pins i.e., pin 2 and pin 3
    attachInterrupt(digitalPinToInterrupt(encoderPin1), updateEncoder, CHANGE);
    attachInterrupt(digitalPinToInterrupt(encoderPin2), updateEncoder, CHANGE);

    TCCR1B = TCCR1B & 0b11111000 | 1;     // set 31KHz PWM to prevent motor noise

    myPID.SetMode(AUTOMATIC);           //set PID in Auto mode
    myPID.SetSampleTime(1);             // refresh rate of PID controller
    myPID.SetOutputLimits(-125, 125); // this is the MAX PWM value to move motor,here
                                      //change in value reflect change in speed of motor.
}
```

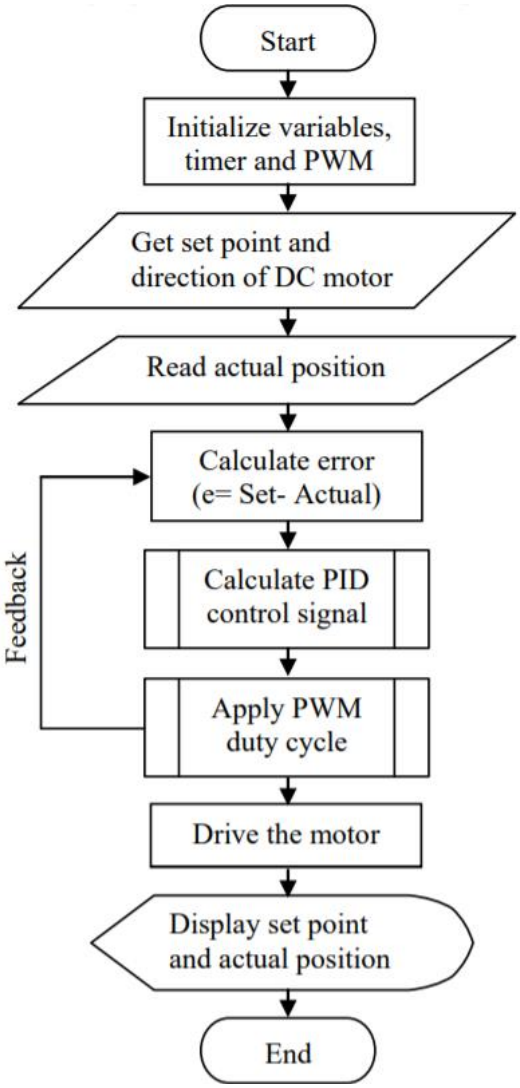
```
void loop() {
    while (Serial.available()) {    // Check if the serial data is available.
      delay(3);                     // a small delay
      char c = Serial.read();       // storing input data
      readString += c;              // accumulate each of the characters in readString
    }

    if (readString.length() >0) {   // Verify that the variable contains information
      Serial.println(readString.toInt());     // Printing the input data in integer form
      User_Input = readString.toInt();        // Here input data is store in integer form
    }

    REV = map (User_Input, 0, 360, 0, 1600);   // mapping user input degree into pulse
    Serial.print("this is REV - ");
    Serial.println(REV);                        // printing Required Encoder value

    setpoint = REV;                             // Setting the desired value
    input = encoderValue ;                // data from encoder consider as a Process value
    Serial.print("encoderValue - ");
    Serial.println(encoderValue);

    myPID.Compute();     // calculate new output using the input
    pwmOut(output);     //from the encoder,setpoint,kp,ki,kd
}




void pwmOut(int out) {
    if (out > 0) {
      analogWrite(MotorEnable, out);
      forward();                            // calling motor to move forward
    }
    else {
      analogWrite(MotorEnable, abs(out)); .
      reverse();                             // calling motor to move reverse
    }
    readString="";      // Cleaning User input, ready for new Input
}

void updateEncoder(){
    int MSB = digitalRead(encoderPin1);     //MSB = most significant bit
    int LSB = digitalRead(encoderPin2);     //LSB = least significant bit

    int encoded = (MSB << 1) |LSB;   //converting the single bits to 2 bit binary
    int sum  = (lastEncoded << 2) | encoded;  //adding it to the previous encoded value

    // If Motor moves forward then increment the encodervalue
    if(sum == 0b1101 || sum == 0b0100 || sum == 0b0010 || sum == 0b1011) encoderValue ++;
    // If Motor moves backward then decrement the encodervalue
    if(sum == 0b1110 || sum == 0b0111 || sum == 0b0001 || sum == 0b1000) encoderValue --;

    lastEncoded = encoded;      //store this value for next time
}

void forward () {
```

```
   digitalWrite(MotorForward, HIGH);
  digitalWrite(MotorReverse, LOW);
}

void reverse () {
   digitalWrite(MotorForward, LOW);
  digitalWrite(MotorReverse, HIGH);
}

void finish () {
   digitalWrite(MotorForward, LOW);
  digitalWrite(MotorReverse, LOW);
}
```

Explanation of Code:

- We include the <PID v1.h> library for our PID computation
- We define some motor pins
- Declare some variables
- We assume some values for kp, ki, kd and we observe the outputs for different values of kp, ki, kd
- Define an instance of PID with the arguments and name it as myPID
- In the setup part we set motor enable, motor forward and motor reverse pins as output
- Turn on input pull up resister at pins 2 and 3
- Attach interrupts for pins 2 and 3 and when these values change the ISR should execute updateEncoder()
- Set the frequency of PWM signal to 31KHz
- Set the PID to automode and give the sample time and set output limits
- In the loop part run a while loop for reading the input from user
- If the input is valid then convert the input stiring to integer value
- Map the user input degree to pulse
- Set this input value as the setpoint which is our desired ouput angle
- Take the actual angle nput from the encoder value and exectue the PID by using PID.Compute()
- Get the output generated from PID.Compute() and run the pwmOut(output) function
- Here when the output is > 0 turn the motor in forward direction
- When the output is < 0 turn the motor in reverse direction
- In the Update encoder function we get the inputs from channel A and channel B of the encoder and convert it to a 2 bit binary and add this to the previous input from the encoder and make a 4 bit binary. This 4 bit binary decides the motor movement. When it is moving forward increment the encoder value and when it is moving backward decrement the encoder value
- Forward and Reverse functions makes the motor forward and motor reverse pins HIGH respectively

The arduino code is visualised as a flow chart shown below



Status of Completion of project.

Progress of the project

- We completed the sketch of mechanical design of project and even verified the working of code without introducing PID controller to it in one dimension
- We assembled the circuit for one motor made it run without using PID
- We planned the design of mechanical prototype of antenna positioning system
- All the prototypes and design of the circuit were made in tinker cad
- We designed the Arduino code for one motor and we could not verify the code because of unavailability of parts as they were in the college itself. Major part of the project is unverified cause of this Pandemic.

To be done

- Assemble all the components and construct the mechanical prototype for antenna positioning system
- Check the validity if the above Arduino code and improve the code further to two DC motors
- Get to know the difficulties of designing while simultaneously operating with two motors at a time.

Applications

The concept of DC motor position control is used in many real-life applications some of them are

- Antenna positioning system
- Design of smart switches that can be used in residential buildings
- Used in scanners, printers, etc.

Links

- Design of the circuit in Tinker Cad can be found here
- Design and prototype of Antenna position system made in Tinker Cad can be found here
- Arduino code can be found here
- PID library used in Arduino code can be found here

THANK YOU

Submitted by:

K M SUDARSHAN, 170108021

R VINAY KUMAR, 170108031

CHINKA NAGA SRIHARI, 170108011

– Under the guidance of

Dr. Chayan Bhawal, Asst. Professor, Dept of EEE, IIT GUWAHATI