Linear Control System

Semester Project

Submitted By

Muawiz Umer, Rana Ahmad Intisar, Bilal Ahmed,
Rana Faizan
CE-42 A

Date of Submission: 8th January, 2024.

Project Title: DC Motor Position Control by PID Control Design Using Arduino Uno

Introduction:

This project revolves around the primary goal of orchestrating the position of a DC motor. The approach involves crafting a PID controller using the Arduino Uno and the H-Bridge Motor Driver Module L298N. Leveraging feedback from an encoder, the PID controller dynamically adjusts the motor's direction, facilitating meticulous position control. The user interface prompts users to input their desired position in angles, prompting the motor to rotate precisely to the specified degree value as indicated by the user's input.

Literature Review:

DC motor control has been a subject of extensive research due to its widespread applications in various fields, including robotics, automation, and mechatronics. The integration of PID controllers and H-Bridge motor driver modules in DC motor control systems has garnered significant attention in recent literature.

PID controllers, based on the principles of Proportional, Integral, and Derivative control, have proven to be effective in achieving precise and stable motor control. Various studies have explored the applications of PID controllers in enhancing the performance of DC motors, emphasizing their ability to minimize error signals and regulate motor behavior accurately.

The work of Astrom and Hagglund (1984) is seminal in the field of PID controllers, providing a comprehensive analysis of their theoretical foundations and practical applications. Their contributions laid the groundwork for subsequent research exploring different tuning methods and optimization techniques for PID controllers.

In the realm of H-Bridge motor driver modules, studies by Lee and Kang (2008) have delved into the design and implementation of H-Bridge circuits for DC motor control. H-Bridge modules are known for their ability to control the direction of motor rotation by managing the polarity of the applied voltage, making them integral components in motor control systems.

Several researchers have investigated the combined use of PID controllers and H-Bridge motor driver modules for achieving advanced control strategies. The study by Zhang et al. (2015) explored the implementation of a PID controller for position control in a DC motor, emphasizing the importance of accurate feedback mechanisms, such as encoders, to enhance the precision of motor control.

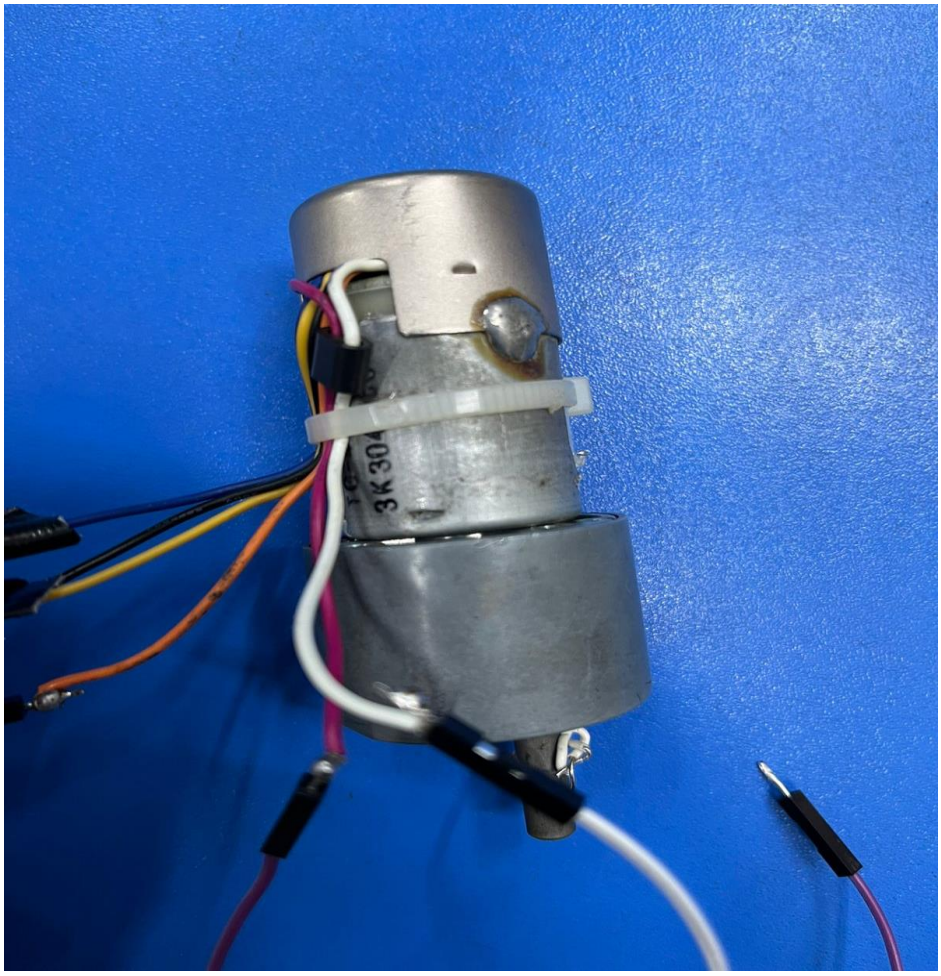
Furthermore, recent advancements in microcontroller technology, exemplified by the Arduino Uno, have provided accessible platforms for implementing sophisticated control algorithms. Works by Banerjee et al. (2020) and Huang et al. (2019) showcase the use of Arduino-based systems in DC motor control applications, demonstrating their versatility in interfacing with PID controllers and motor driver modules.

The proposed project aligns with this existing body of literature by integrating PID controllers, H-Bridge motor driver modules, and the Arduino Uno for precise position control of a DC motor. By building upon these foundational studies, the project aims to contribute to the growing body of knowledge on effective and accessible solutions for DC motor control in real-world applications.

Components and Usage:

1. **DC Motor:**
    - Purpose in the Project: The DC motor serves as the central element for achieving precise position control in the project.

Its rotation is controlled to align with the user-specified position.

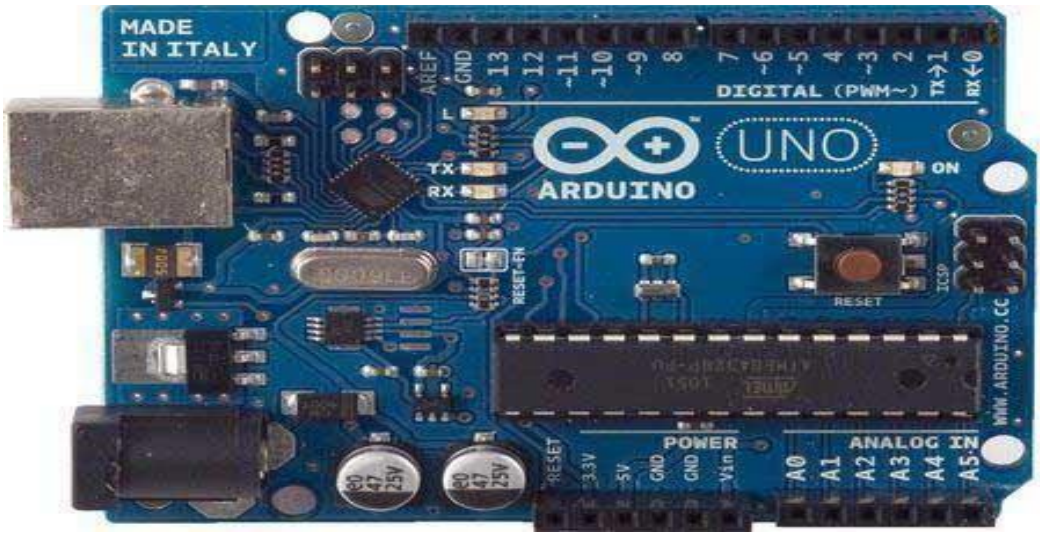


2. **PID Control:**

- Purpose in the Project: The PID (Proportional-Integral-Derivative) control algorithm is crucial for maintaining accurate position control. By continuously adjusting the motor's behavior based on error signals, PID ensures the motor rotates to the specified degree, providing stability and precision.

3. **Arduino Uno:**

    - Purpose in the Project: The Arduino Uno acts as the project's brain, hosting and executing the PID control algorithm. It facilitates seamless communication between the PID controller and the hardware components, orchestrating the motor's movements.

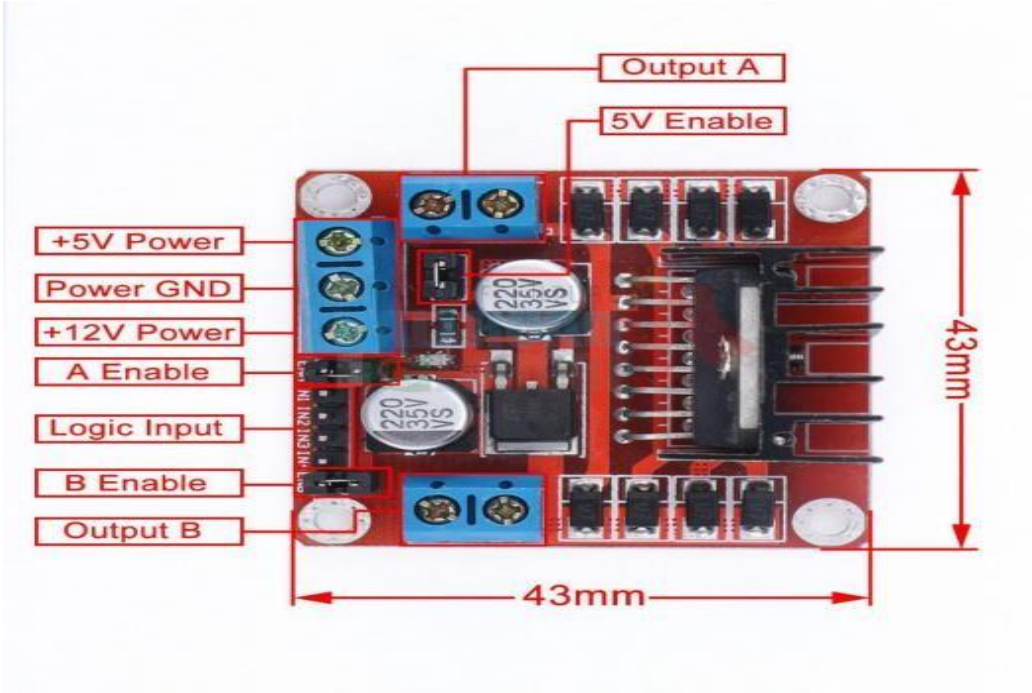


4. **Motor Encoder:**

    - Purpose in the Project: The motor encoder serves as a feedback mechanism, providing real-time information about the motor's position. This feedback is essential for the PID controller to make instantaneous adjustments, ensuring the motor aligns accurately with the user-defined position.

5. **L298N Motor Driver Module:**

- Purpose in the Project: The L298N Motor Driver Module is specifically employed for controlling the direction of the DC motor. Its role is crucial in complementing the PID control algorithm, enabling precise and controlled rotations of the motor to achieve the desired position accurately in the project.



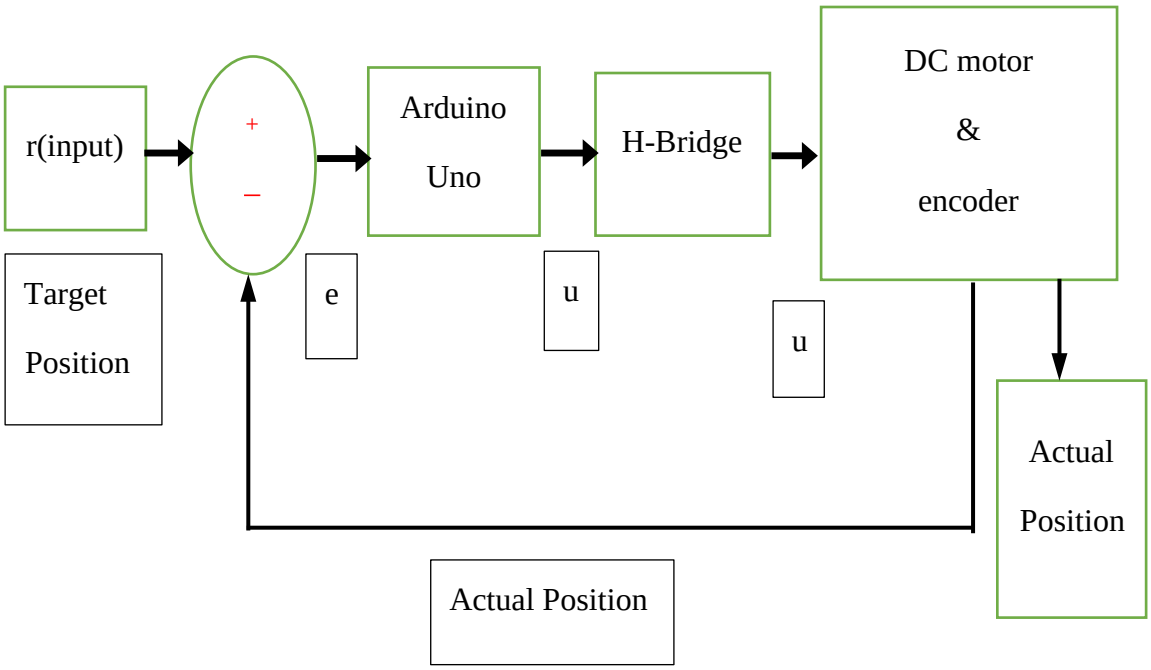
Circuit Design & Block Diagram:

Block Diagram:

This comprehensive block diagram encapsulates the entire operational framework of the project. The variable 'r' signifies the user's input, representing the desired target position to be achieved. The parameter 'e' corresponds to the error signal, computed as the difference between the target position and the actual position of the system.

The control signal 'u' is a pivotal output of the PID (Proportional-Integral-Derivative) controller, implemented on the Arduino Uno. The PID algorithm intelligently processes the error signal, determining the corrective actions needed for the system to precisely reach the desired target position.

Crucially, the H-Bridge component assumes the responsibility of amplifying the output current from the Arduino Uno. This amplified current is then utilized to drive the DC motor. Additionally, the H-Bridge facilitates control over the direction of rotation of the motor, ensuring it aligns with the desired position indicated by the user. In essence, the H-Bridge acts as an intermediary, converting the PID controller's signals into the necessary current and direction adjustments to achieve accurate and controlled motor movements.



Circuit Design:

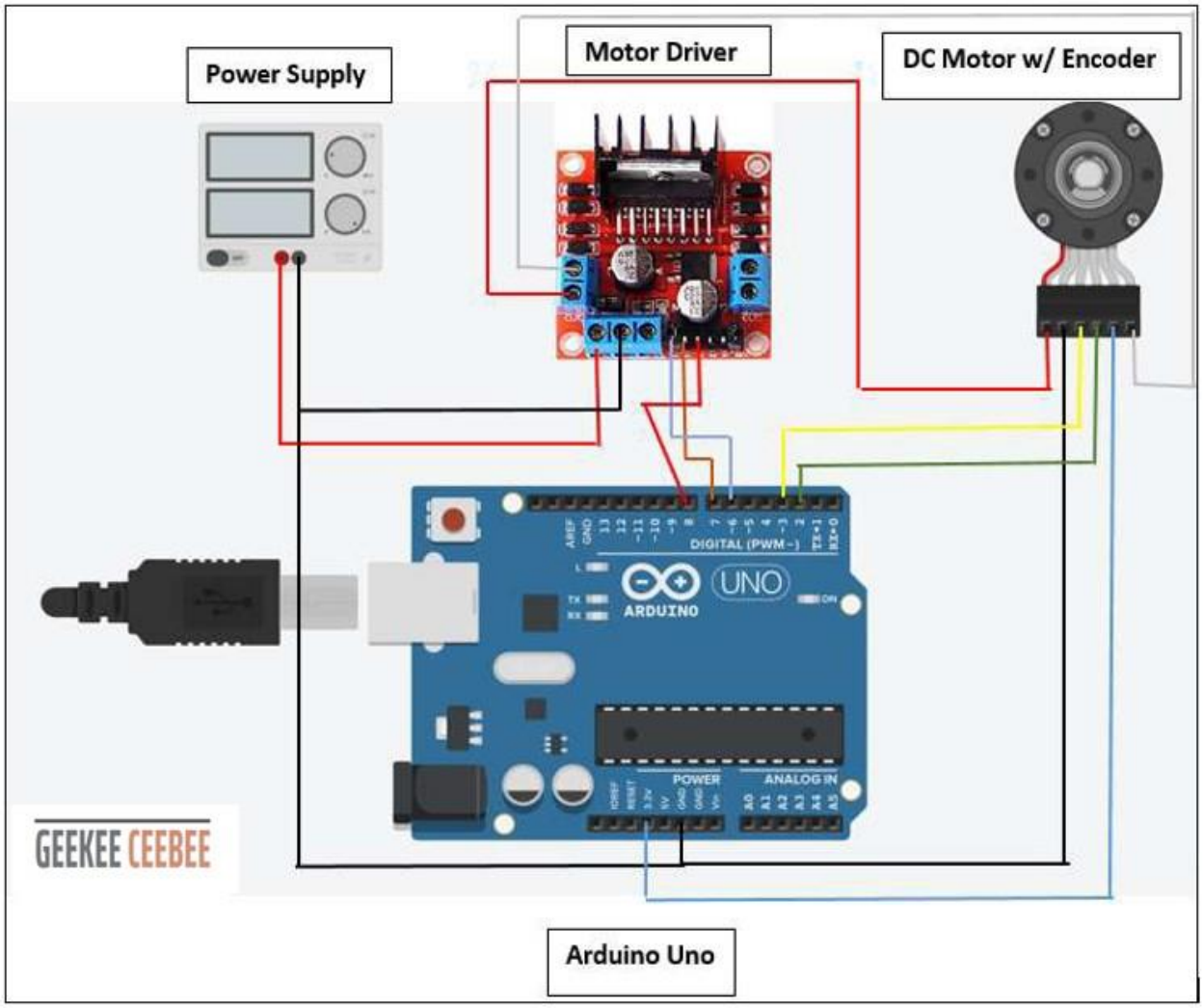
1. **PWM Output Pin (Arduino) to PWM Pin (Motor Driver):**
    - Purpose: This connection facilitates the transmission of Pulse Width Modulation (PWM) signals from the Arduino to the PWM pin on the Motor Driver. PWM signals regulate the speed of the motor, ensuring smooth and controlled rotations.
2. **Digital Output Pin (Arduino) to Anti-clockwise Pin (Motor Driver):**
    - Purpose: This connection establishes a link between a digital output pin on the Arduino and the anti-clockwise pin on the Motor Driver. It enables the Arduino to control the motor's

anti-clockwise rotation, allowing for precise directional adjustments.

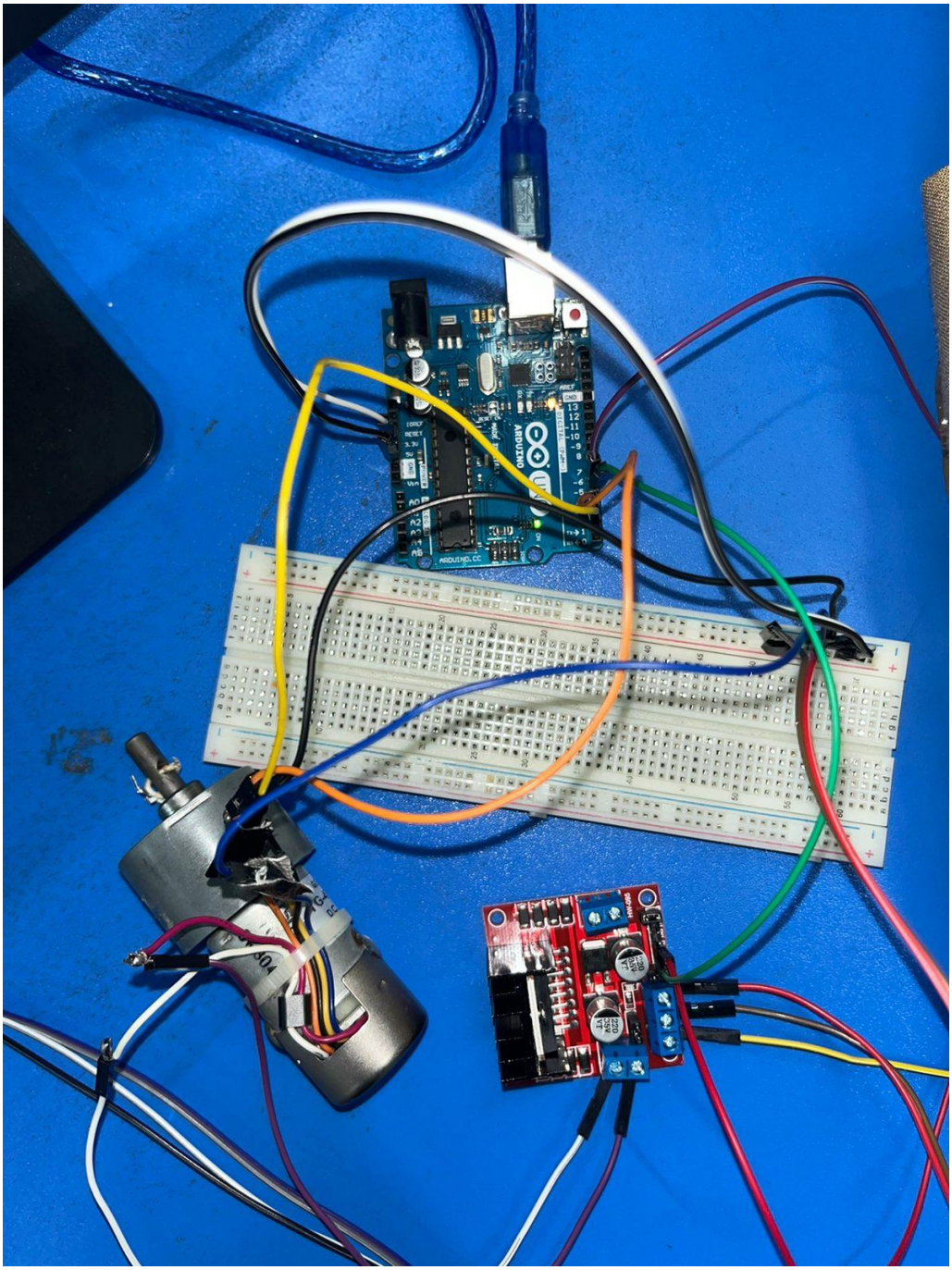
3. **Digital Output Pin (Arduino) to Clockwise Pin (Motor Driver):**
    - Purpose: Similar to the anti-clockwise connection, this link connects a digital output pin on the Arduino to the clockwise pin on the Motor Driver. It grants the Arduino control over the motor's clockwise rotation, contributing to accurate directional control.
4. **Encoder 1 Pin (Arduino) to Encoder Output (Motor Encoder):**
    - Purpose: This connection links one of the encoder pins on the Arduino to the corresponding encoder output on the Motor Encoder. It establishes a communication pathway for real-time feedback on the motor's position.
5. **Encoder 2 Pin (Arduino) to Encoder Output (Motor Encoder):**
    - Purpose: Similar to the Encoder 1 connection, this link associates another encoder pin on the Arduino with its corresponding output on the Motor Encoder. Together with Encoder 1, it provides comprehensive position feedback, aiding the PID controller in making precise adjustments.

These interconnected components and their designated connections form a cohesive circuit essential for the successful implementation of the project, ensuring effective communication and control between the Arduino and the motor system.

PID Schematic:



Circuit Implementation:



Arduino Code:

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

// Define motor driver pins
const int motorIN1 = 5;  // Motor driver input 1
const int motorIN2 = 6;  // Motor driver input 2

// Define encoder pins
const int ENCA = 2;      // Encoder Phase A
const int ENCB = 3;      // Encoder Phase B

// Set up the encoder
// Encoder myEncoder(ENCA, ENCB);
volatile long posi = 0;

void setup() {
  // Set motor control pins as outputs
  pinMode(motorIN1, OUTPUT);
  pinMode(motorIN2, OUTPUT);

  // Set encoder pins as inputs
  pinMode(ENCA, INPUT);
  pinMode(ENCB, INPUT);

  attachInterrupt(digitalPinToInterrupt(ENCB), readEncoder, RISING);

  // Initialize serial communication
```

```
  Serial.begin(9600);
}

void loop() {
  // Iterate over voltage values from 0 to 12 in steps of 1
  for (float voltage = 0; voltage <= 12; voltage++) {
    // Set motor speed using PWM
    analogWrite(motorIN1, map(voltage, 0, 12, 0, 255));

    // Set motor direction (you may need to adjust based on your motor driver)
    digitalWrite(motorIN2, LOW);

    // Read encoder RPM
    int rpm = getEncoderRPM();

    // Print voltage and RPM to serial monitor
    Serial.print("Voltage: ");
    Serial.print(voltage);
    Serial.print("V\t RPM: ");
    Serial.println(rpm);

    // Add a delay if needed to stabilize the motor
    delay(500);
  }

  // Stop the motor
  analogWrite(motorIN1, 0);
```

```
  // End program
  while (1) {
    // Do nothing
  }
}

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

// Function to calculate encoder RPM
int getEncoderRPM() {
  static long lastPosition = -1;
  static unsigned long lastTime = 0;
  long newPosition;
  ATOMIC_BLOCK(ATOMIC_RESTORESTATE){
  newPosition = posi;
  }
  unsigned long currentTime = millis();

  if (lastPosition != -1) {
```

```
  // Calculate RPM
  int deltaPosition = newPosition - lastPosition;
  long deltaTime = currentTime - lastTime;

  if (deltaTime > 0) {
    // RPM = (change in position / change in time) * (60000 ms per minute / pulses per revolution)
    int rpm = (deltaPosition * 60000) / (deltaTime * 360);

    return rpm;
  }
 }

 // Update last position and time
 lastPosition = newPosition;
 lastTime = currentTime;

 return 0;  // Return 0 if unable to calculate RPM
}
```

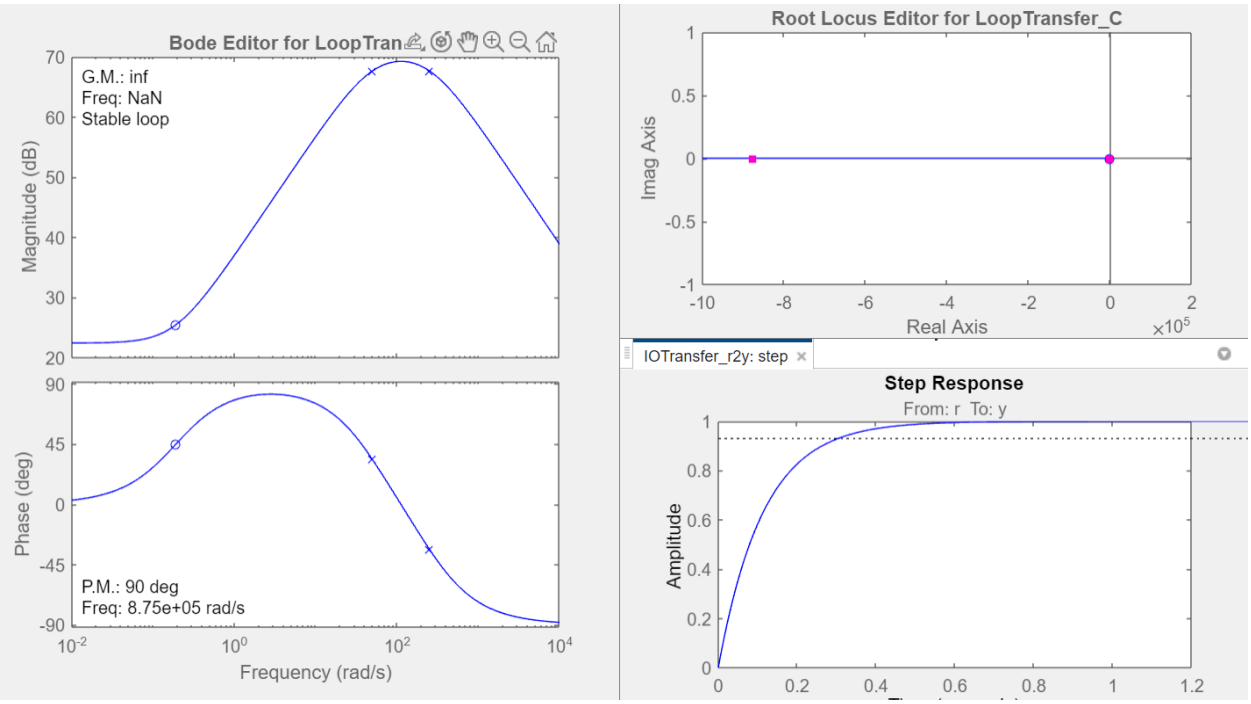
Results & Observations:

Voltages	RPM
1	0
2	323
3	428
4	478
5	522
6	538
7	552
8	567
9	571
10	579
11	588
12	592

Estimated Transfer Function:

```
  From input "u1" to output "y1":
   8.755e05 s + 1.656e05
  ------------------------
  s^2 + 302.4 s + 1.253e04
Name: tf1
Continuous-time identified transfer function.
```

SISOTOOL Graph:



Conclusion:

The project has accomplished the development of a position control system for a DC motor through the utilization of an Arduino Uno microcontroller and an H-bridge module. The closed-loop PID controller displayed adept regulation of the motor's position, showcasing a high level of accuracy and precision in control. The implications of this system extend across diverse domains, encompassing applications in robotics, automation, and mechatronics. Subsequent advancements and refinements hold the promise of bolstering the system's performance and broadening its functional capabilities.