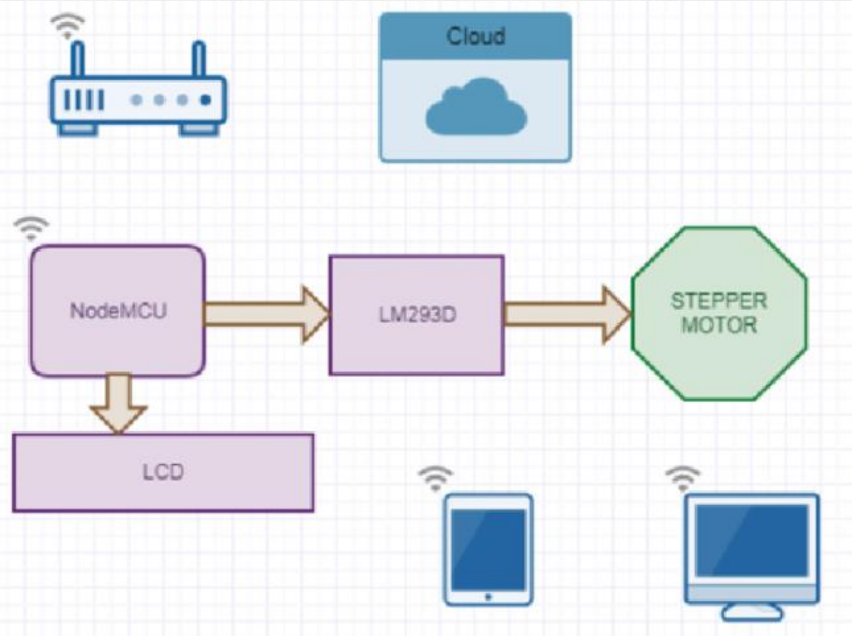
Background and Motivation: The project aims to implement a PID position control system for a 24V DC gear motor, leveraging IoT for remote control. The motivation behind this lies in achieving precise motor positioning for various applications, such as robotics or automation. PID control ensures stable and accurate positioning, while IoT integration allows users to control the motor remotely.

Project Objectives: The primary objectives include designing a robust PID control system, integrating IoT for remote control, and achieving accurate motor positioning based on user input. The project also focuses on providing a scalable solution that can be extended to other systems requiring precise control.

Overview of the System Architecture: The system architecture comprises an Arduino Uno, ESP8266 for IoT connectivity, a 24V DC gear motor, a rotary encoder for feedback, and an L298N motor driver. The Arduino calculates the motor position using PID control logic, and the user can adjust the motor's position remotely through the Blynk IoT platform.



Introduction: This project report focuses on developing a robust motor position control system using IOT. Precise motor position control is crucial for various industrial applications, enabling tasks like object manipulation and system stabilization. The objective is to investigate control algorithms and techniques for achieving high precision, stability, and responsiveness in motor position control.

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

Components:

1. 24 V Geared DC Motor
2. Rotary Encoder 500 ppr
3. Arduino Uno
4. Esp8266
5. Dual H bridge (L298N)
6. Breadboard

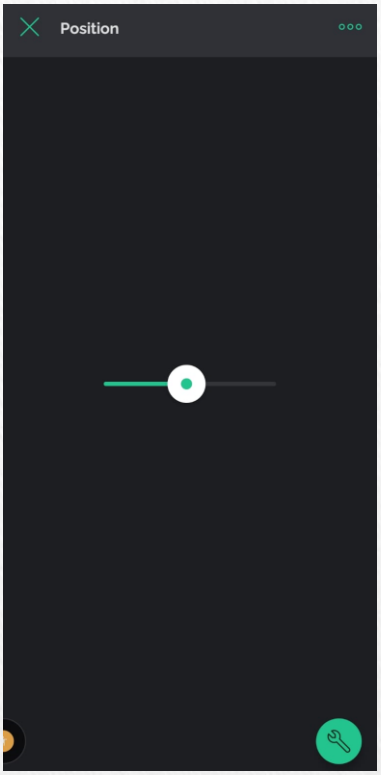
- **Arduino Uno:** The Arduino Uno serves as the main controller, executing the PID control algorithm. It interfaces with the motor driver and reads feedback from the rotary encoder to determine the motor's current position.
- **ESP8266:** The ESP8266 facilitates IoT connectivity, allowing users to control the motor remotely through the Blynk app. It receives user input from the app and forwards it to the Arduino for motor position adjustments.
- **DC Gear Motor (24V):** The 24V DC gear motor is responsible for physical movement. Its torque and speed characteristics make it suitable for applications requiring precise positioning, such as robotics or automation systems.

- **Rotary Encoder (500 Pulses):** The rotary encoder provides feedback on the motor's current position. The Arduino uses this information to calculate any deviation from the desired position and adjusts the motor accordingly.
- **Motor Driver L298N Module:** The L298N motor driver module controls the motor's speed and direction based on signals from the Arduino. It ensures the precise execution of the calculated motor position.
- **DC Gear Motor (24V):** The 24V DC gear motor is responsible for physical movement. Its torque and speed characteristics make it suitable for applications requiring precise positioning, such as robotics or automation systems.

Software Components:.

- **Arduino IDE:** The Arduino IDE is used to program the Arduino Uno. The code implements the PID control algorithm, interprets signals from the rotary encoder, and communicates with the motor driver to achieve accurate motor positioning.
- **Blynk IoT Platform:** Blynk is employed for IoT integration, offering a user-friendly platform for remote control. The Blynk app allows users to set the desired motor position through a slider, which is then transmitted to the Arduino via the ESP8266.
- **Mobile App Integration:** The Blynk app is customized to include a slider that corresponds to the desired motor position (0 to 360 degrees). Users can change the position remotely, and the app sends this information to the ESP8266, initiating the adjustment process.

Software Implementation and Technologies:



ARDUINO

D D

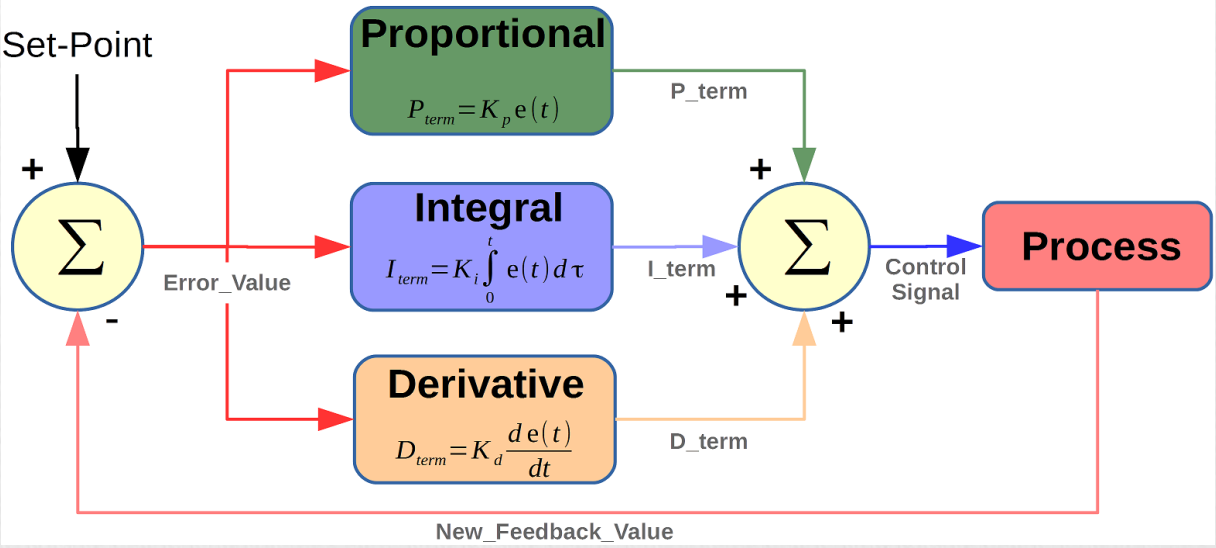
B

Blynk.App

PID Implementation: In this project, a separate controller is utilized to regulate the Position of a DC motor, employing a PID controller. The PID controller combines three terms: one proportional to the error signal, another proportional to the integral of the error signal, and a third proportional to the derivative of the error signal. This configuration ensures stable gain, although it may introduce some steady-state inaccuracy.

To minimize or eliminate steady-state error, an integral controller is employed. On the other hand, the derivative controller reduces the rate of change of the error. PID controllers offer advantages such as lower overshoot, absence of offset, and improved stability.

Tuning a PID control system can be accomplished through various methods, such as the Ziegler-Nichols approach or trial and error, to achieve the desired performance. In this project I used Hit and trial method to tune kp, ki and kd values
To effectively describe system behavior, parameter estimation is crucial, often involving mathematical models such as statistical probability distribution functions and parametric dynamic models. Typical parameter estimation tasks include importing and analyzing input-output data, such as rotor speed (output) and applied voltage (input) for a DC motor. The estimation process involves selecting which initial conditions of the model, such as motor friction and inductance, should be estimated. The unknown parameters are then estimated using the nonlinear least squares approach.



System Architecture

Overview of the Connection Setup: The Arduino, ESP8266, motor, rotary encoder, and motor driver are interconnected. The rotary encoder provides feedback to the Arduino, while the Blynk app communicates with the ESP8266, facilitating remote control. The Arduino processes the data and sends control signals to the motor driver for precise motor positioning.

Communication Flow Diagram: Data flows from the Blynk app to the ESP8266, which forwards it to the Arduino. The Arduino calculates the required motor position, and the motor driver executes the necessary adjustments. Feedback from the rotary encoder ensures the system's accuracy.

Control System Design

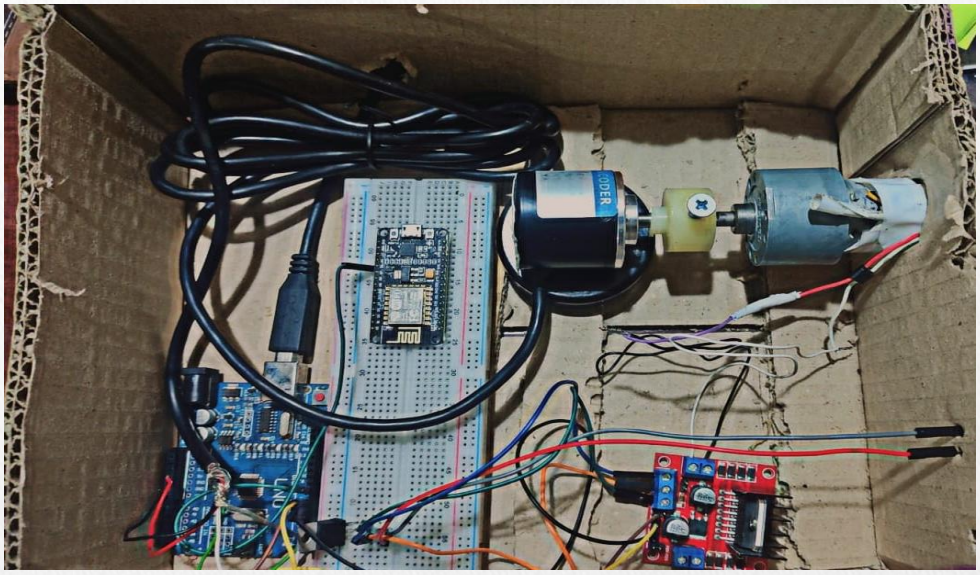
PID Controller Explanation: The PID controller combines proportional, integral, and derivative terms to minimize the error between the desired and actual motor positions. This ensures a smooth and stable transition to the target position, reducing overshooting and oscillations.

Implementation on Arduino: The Arduino code incorporates the PID control algorithm. It reads the current position from the rotary encoder, receives user input from the Blynk app, calculates the deviation, and adjusts the motor's position accordingly.

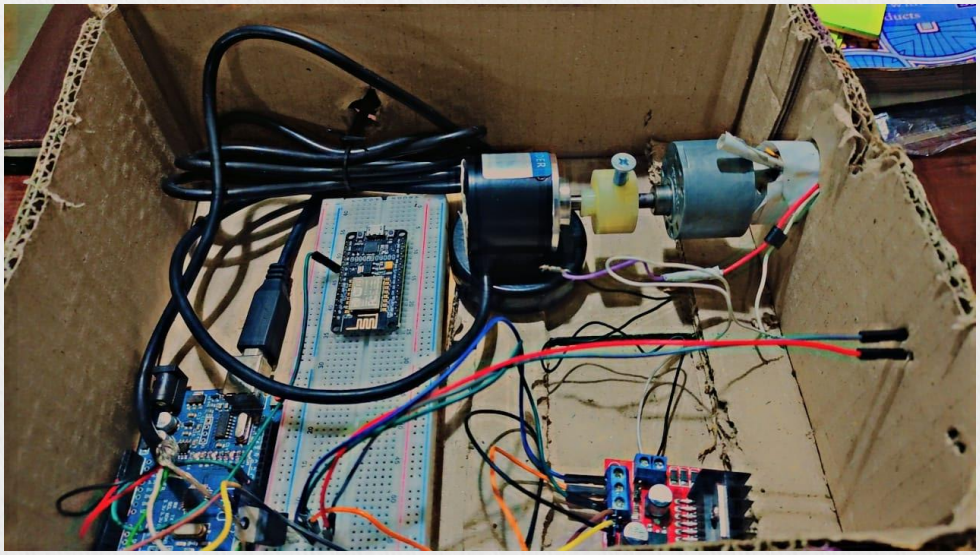
PID Controller Explanation: PID parameters are fine-tuned to optimize the system's performance. This involves adjusting the proportional, integral, and derivative gains to achieve a balance between responsiveness and stability, ensuring accurate and efficient motor positioning.

Hardware Design

Top View:



Side View:



Conclusion: In summary, this project successfully implemented a PID controller for precise position control of a DC motor using an Arduino Uno, 24V Geared DC Motor with Encoder, Dual H bridge (L298N), Breadboard, and Voltage Regulator. The PID algorithm, with proportional, integral, and derivative terms, ensures high precision in motor positioning for industrial applications. Tuning parameters through the PID tuner toolbox allows customization based on user-defined response time and transient behavior. The integral and derivative control components play a crucial role in minimizing steady-state error and enhancing system stability. Overall, this work contributes to advancing the understanding and practical application of PID-controlled DC motor position systems in industrial automation.