# PID Control of Motor Sensor

Team 4

December 2, 2023

## 1 Motivation

The motivation behind the PID control of a DC motor experiment is to showcase the fundamental concept of feedback control in engineering and its real-world application.

Feedback control systems are crucial in various industries to regulate processes, maintain stability, and achieve desired performance. DC motors are widely used in automation, robotics, and manufacturing, making them an excellent choice for demonstrating how feedback control can enhance their performance.

## 2 Theory

### 2.1 PID Controller

Implement a PID control algorithm that computes a control signal based on the error between the desired reference speed and the actual speed measured by the RPM sensor.

**Proportional Control (P):** Proportional control adjusts the motor speed in proportion to the error between the desired and actual positions or velocities. It is responsive to the current error, meaning it acts in proportion to the present deviation from the setpoint. The P term contributes to reducing steady-state error but may lead to overshooting.
**Intrgral Control (I):** Integral control eliminates the accumulated error over time by integrating the error signal. It is effective in addressing steady-state errors that P control alone may not eliminate. The I term helps in bringing the system to the setpoint and improving system stability.
**Derivative Control (D):** Derivative control anticipates future behavior by considering the rate of change of the error. It helps dampen oscillations and reduce overshooting by providing a corrective action based on the rate of change of the error signal. D control enhances system stability and responsiveness.

The PID controller output is the sum of the P, I, and D components:
PID output = $K_p\times$ Error + $K_i\times$ Integral of Error + $Kd\times$ Derivative of Error

where $K_p$, $K_i$, and $K_d$ are the proportional, integral, and derivative gains respectively.

### 2.2 Feedback Loop

The RPM sensor measures the actual motor speed and the reading is used to calculate the error between the desired reference speed and the measured speed. The PID controller is then used to compute the control signal. Accordingly, the input voltage is adjusted to the motor based on the control signal. This works in a continuous loop and help detect and cover any errors or deviations.

### 2.3 Ziegler-Nichols Method

For tuning the PID controller we have used the Ziegler-Nichols method. It is performed by setting the I and D terms to zero. The P gain, $K_p$ is then increased (from zero) until it reaches the ultimate gain $K_u$, at which the output of the control loop has stable and consistent oscillations. $K_u$ and the oscillation period $T_u$ are then used to set the P, I, and D gains depending on the type of controller

used and behaviour desired.
The ultimate gain ($K_u$) is defined as $1/M$, where M = the amplitude ratio, $K_i = \frac{K_p}{T_i}$ and $K_d = K_p T_d$. These 3 parameters are used to establish the correction $u(t)$ from the error $e(t)$ via the equation:

$$u(t) = K_p\left(e(t)\right) + K_i\left(\int_0^t e(\tau)d\tau\right) + K_d\left(\frac{de(t)}{dt}\right)$$

which has the following transfer function relationship between error and controller output:

$$u(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)e(s) = K_p\left(\frac{T_d T_i s^2 + T_i s + 1}{T_i s}\right)e(s)$$

# 3 Hardware Components Used

- **DC Motor** : The actuator that generates mechanical motion
- **RPM Sensor** : Measures the actual speed of the motor
- **ESP32 Dev Module** : Controls input voltage to the motor and recieve feedback from the RPM senor
- **Power Supply** : Provides necessry voltage to the motor
- **PC/IOT Device** : Remotely monitors and controls the system
- **Connecting wires and Breadboard** : Make the connections and complete the circuit
- **Wheel and Box** : Hardware Setup
- **Motor Driver (L293D IC)** : Used for PID control

# 4 Working

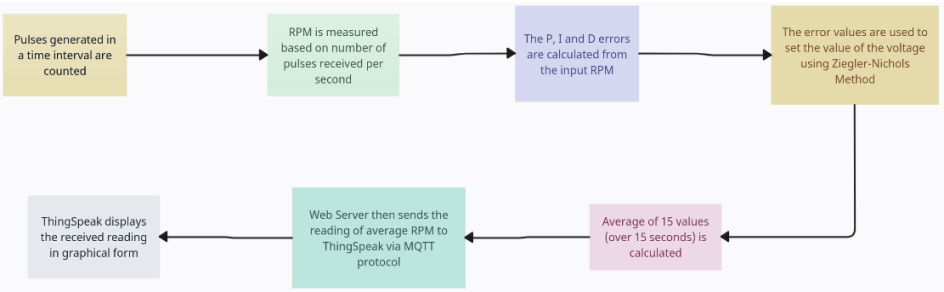


Figure 1: Block Diagram

The RPM sensor beeps every time a pulse is generated (pulse is generated when there is a state change, i.e., Block $\longrightarrow$ Open or Open $\longrightarrow$ Block). The code checks for the total number of beeps and divides with the total time. Using further calculations, it calculates the RPM of the wheel. The calculated RPM is then printed on the Serial Monitor and is sent over to ThingSpeak for plotting and analysis.

From the input RPM, we calculated the proportional(P) error.

$$P = \text{Current RPM} - \text{Previous RPM}$$

We then calculated the integral error and the derivative error

$$I = \int_0^t e(\tau)d\tau$$

$$D = \frac{de(t)}{dt}$$

and set the value of the voltage using the Ziegler-Nichols Method.

# 5 Circuit Diagram

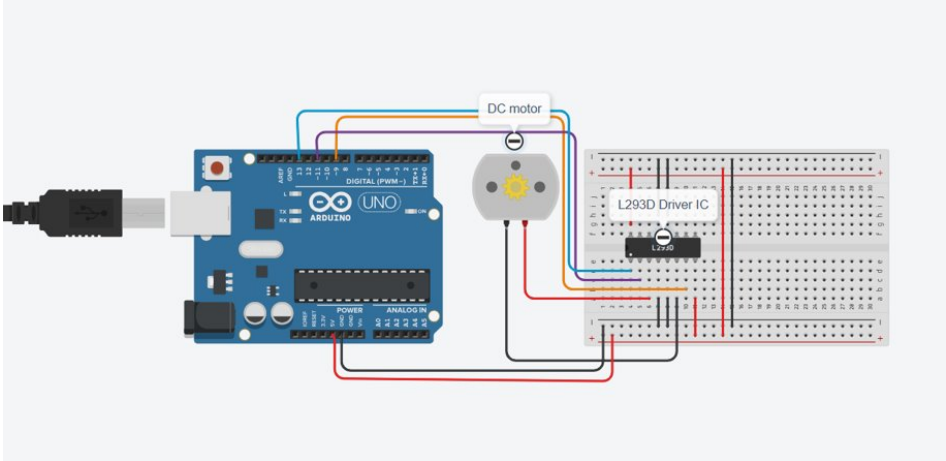


Figure 2: Circuit Diagram

Above is the circuit diagram of our project.

# 6 UI Interface and ThingSpeak

We have developed a UI interface which can be used to give input RPM speed to the motor and also provides PID constants for the same.
We have also incorporated the use of ThingSpeak which can be used to plot values and perform analysis which is crucial for error detection and correction.