|  |  |
| --- | --- |
| Ex. No:15 | **Demonstration of Closed Loop Speed Control of DC Motor** |
| Date: |

**Objective:**

The objective of this Mini project is to Demonstrate closed loop speed control of DC motor in hardware.

**Introduction:**

The Proportional Controller, commonly known as a P controller, is a fundamental component in control systems engineering that plays a crucial role in regulating and maintaining desired outputs in various processes. Control systems are employed in diverse fields, ranging from industrial processes and manufacturing to robotics and environmental systems.

At its core, the P controller operates on the principle of proportionality, adjusting the system's output in direct proportion to the difference between the desired setpoint and the current state or output of the system. This control strategy is based on the premise that the larger the error between the setpoint and the actual output, the greater the corrective action applied by the controller.

The simplicity and effectiveness of the P controller make it a widely used element in control systems. By modulating the system output in proportion to the error signal, the P controller provides a rapid response to deviations from the desired setpoint. However, it has limitations, such as the potential for steady-state error in certain situations

# Required Components:

# Arduino Uno(1N).

# L298N Motor driver(1N).

# Proximity Sensor(1N).

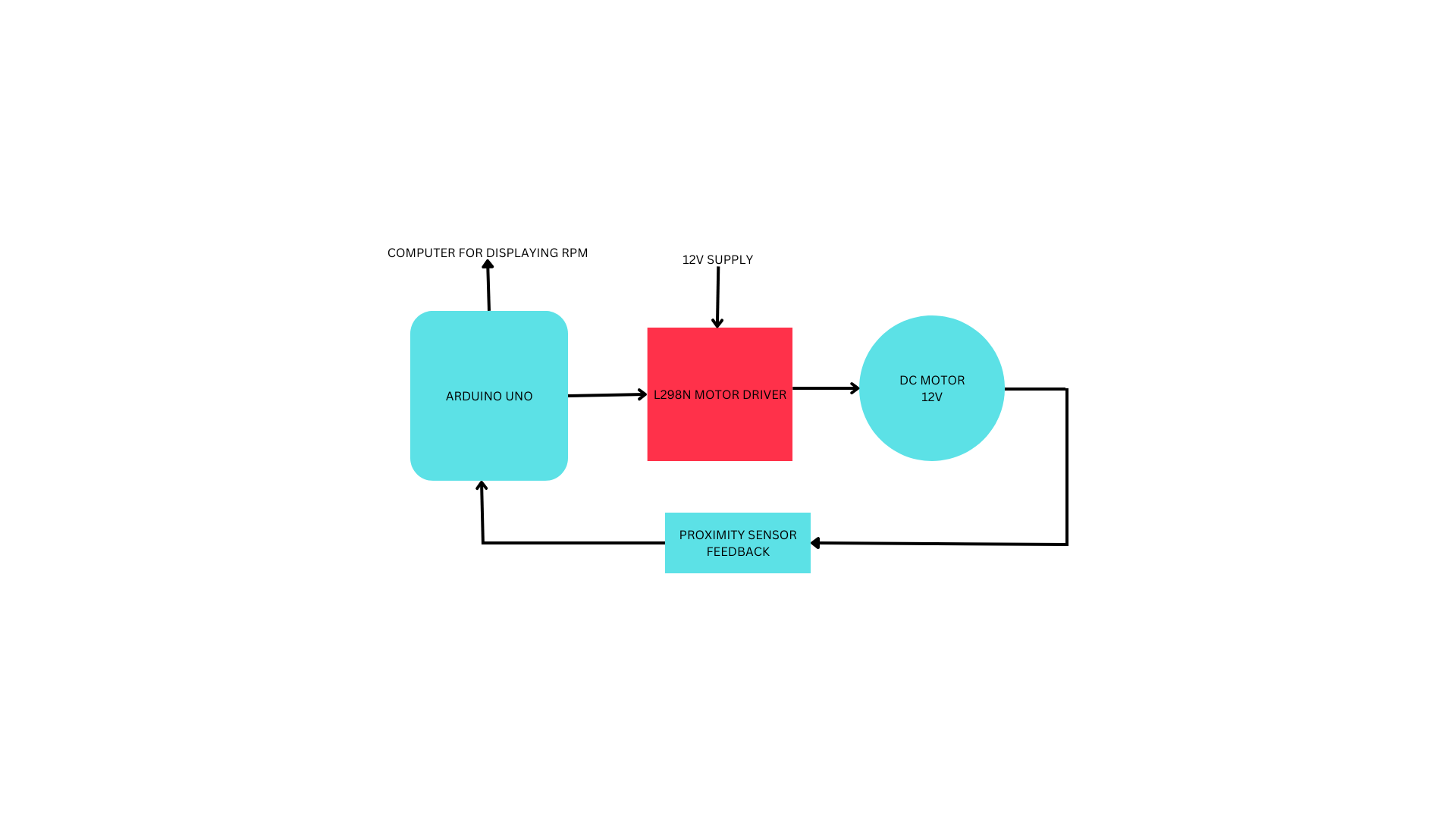
# 12 Dc motor(1N).

# Jumper wire(As required).

# Regulated power supply(0-36V).

# Optical Tachometer.

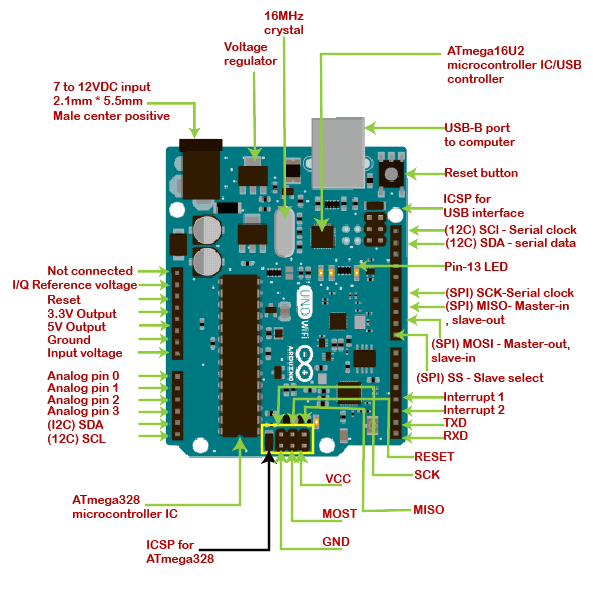
**Block diagram:**



**Pin Diagram:**



L298N motor driver



Arduino Uno

**Procedure:**

Step 1: Connections:

* Connect the Vcc of the sensor to 5V of Arduino.
* Connect the GND of the sensor to GND of Arduino.
* Connect the OUT pin of the sensor to digital pin 2 of the Arduino.
* Connect the GND of Arduino and the GND of L298N motor driver.
* Connect the +V of the RPS to the 12V of the motor driver.
* Common Ground the GND of sensor,Arduino and RPS.
* Connect the ENA pin of the L298N motor driver to the ~3 pin of the Arduino.
* Connect the OUT1 of the motor driver to the 12V Dc Motor.
* Connect Arduino pin 4 to the IN1 pin in L298N motor driver.
* Connect Arduino pin 5 to the IN2 pin in L298N motor driver.

Step 2:Write the Arduino code for implementing P control using Arduino Uno using Arduino IDE.

Step 3: Verify the speed of the motor using a optical tachometer.

**Arduino Code:**

const int sensorPin = 2; // The digital input pin to which the proximity sensor is connected

unsigned long prevTime = 0;

unsigned long pulseCount = 0;

unsigned long rpm = 0;

int currentRPM = 0;

const int motorSpeedPin = 3; // PWM pin for motor speed control

const int motorDirectionPin1 = 4; // Motor driver input 1

const int motorDirectionPin2 = 5;

const int targetRPM=1000;// Target RPM (adjust as needed)

const float Kp = 0.71;

void setup() {

  pinMode(motorSpeedPin, OUTPUT);

  pinMode(motorDirectionPin1, OUTPUT);

  pinMode(motorDirectionPin2, OUTPUT);

  Serial.begin(9600); // Initialize serial communication

  attachInterrupt(digitalPinToInterrupt(sensorPin), countPulse, RISING); // Attach interrupt on rising edge

}

void loop() {

  unsigned long currentTime = millis();

  // Calculate RPM every 1 second

  if (currentTime - prevTime >= 1000) {

    rpm = (pulseCount \* 60) / 12; // RPM calculation

    pulseCount = 0; // Reset pulse count

    prevTime = currentTime; // Update previous time

    currentRPM = rpm;

    int error = targetRPM - currentRPM;

    int motorSpeed = Kp \* error;

    motorSpeed = constrain(motorSpeed, -225, 225);

    if (motorSpeed > 0) {

      digitalWrite(motorDirectionPin1, HIGH);

      digitalWrite(motorDirectionPin2, LOW);

    } else {

      digitalWrite(motorDirectionPin1, LOW);

      digitalWrite(motorDirectionPin2, HIGH);

      motorSpeed = -motorSpeed;

    }

    analogWrite(motorSpeedPin, motorSpeed);

    Serial.print("Current RPM: ");

    Serial.print(currentRPM);

    Serial.print(", Target RPM: ");

    Serial.print(targetRPM);

    Serial.print(", Error: ");

    Serial.print(error);

    Serial.println();

  }

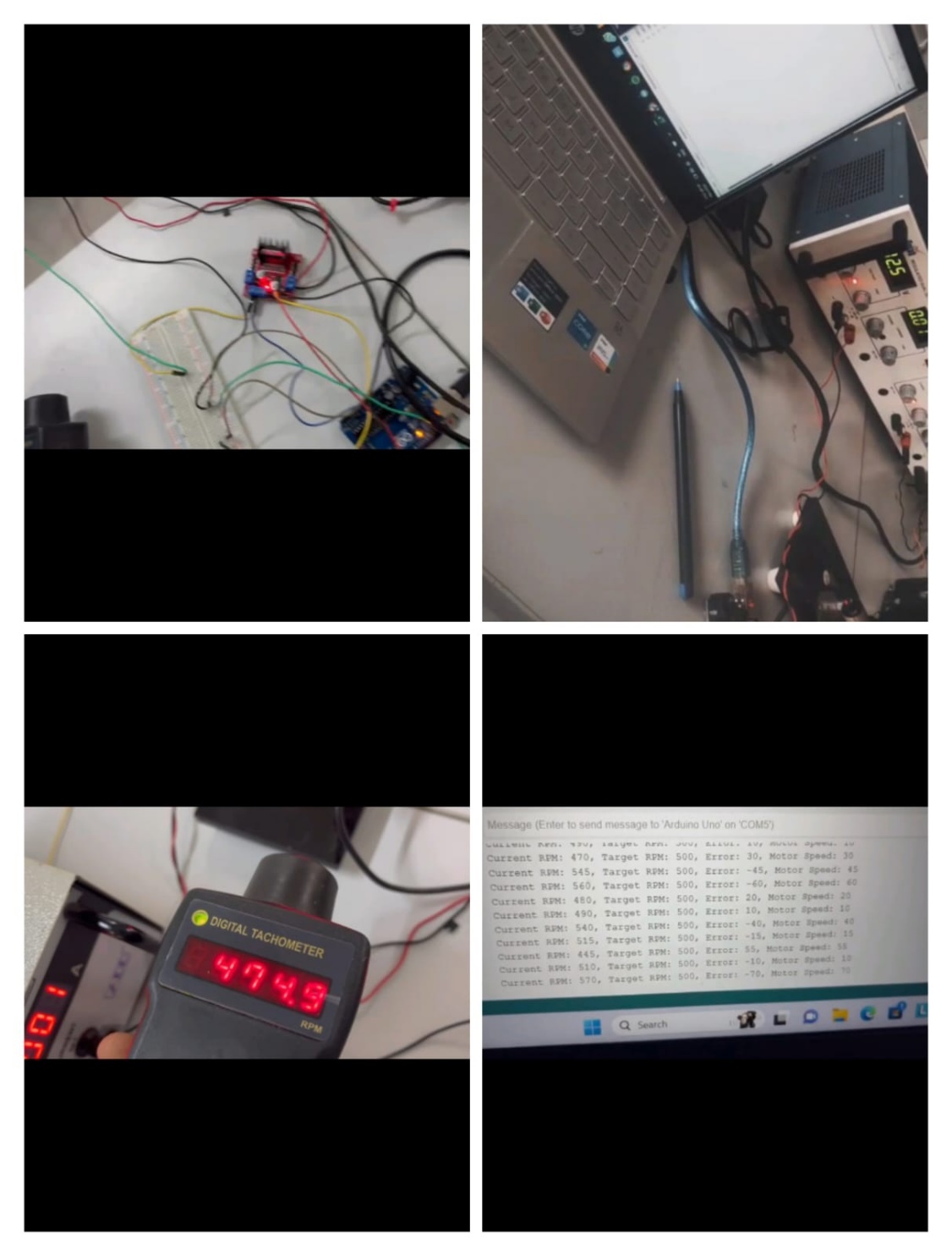
}

void countPulse() {

  // This function is called when the proximity sensor detects a pulse

  pulseCount++;

}



**Result**

Hence the demonstration of closed loop system in hardware is successfully implemented.