Internship Project Report

# Arduino-Based DC Motor Speed Control with RPM Display

Reported By: **CEPCO GROUP**

Group Members:

1. Ahmed Abdulrahman Abdullah Bin Alfaqeeh: BU21EECE0200026
2. S Siva Kumar: BU21EECE0200038
3. Challa Santhosh: BU21EECE0100394

GITAM University

EECE Department

## Introduction:

This project aims to develop an Arduino-controlled system that regulates the speed of a DC motor and displays its revolutions per minute (RPM) on an LCD. This application is particularly useful in robotics, manufacturing, and other areas where precise motor control is crucial. The project utilizes Pulse Width Modulation (PWM) for speed control and employs a feedback mechanism to display the motor's RPM on a liquid crystal display (LCD).

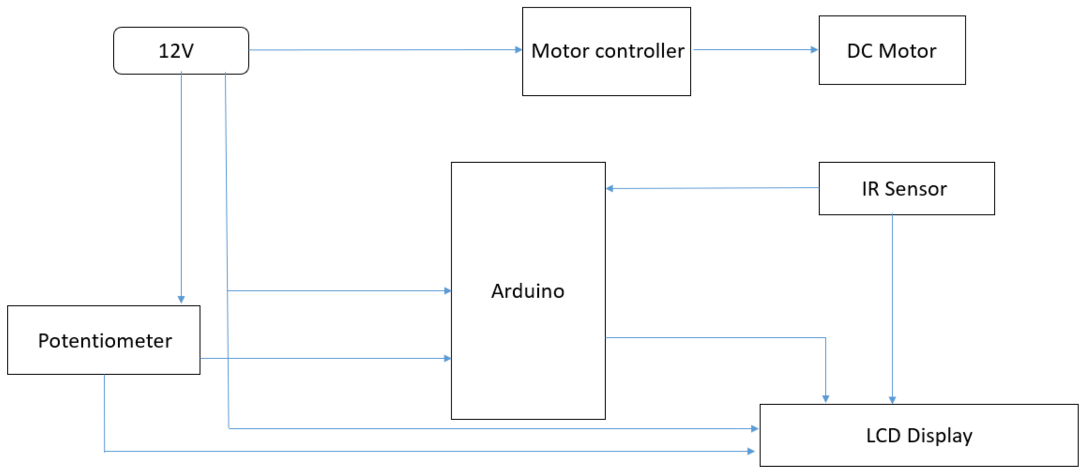
## Components Used

1. **Arduino Uno R3**: A microcontroller board based on the ATmega328P. It's used here for overall control, to process input from potentiometers and to drive the motor and LCD.
2. **LCD (LM016L)**: A 16x2 character LCD display that shows the motor's RPM and other status messages.
3. **IRFZ44N MOSFET**: Acts as a switch to control the power to the DC motor, allowing for speed control via PWM signals from the Arduino.
4. **1N4007 Diode**: Used to prevent back EMF from the motor from damaging the circuit when the motor is turned off or changes direction.
5. **Potentiometer (RV1)**: Allows the user to adjust the desired speed of the motor, which is read by the Arduino.
6. **Resistors (R1, R2)**: Used for pulling up/down signals and ensuring correct voltage levels are applied, especially for the gate of the MOSFET.
7. **DC Motor**: The actuator whose speed is being controlled.
8. **Connecting wires and Breadboard**: For assembling the circuit without soldering.

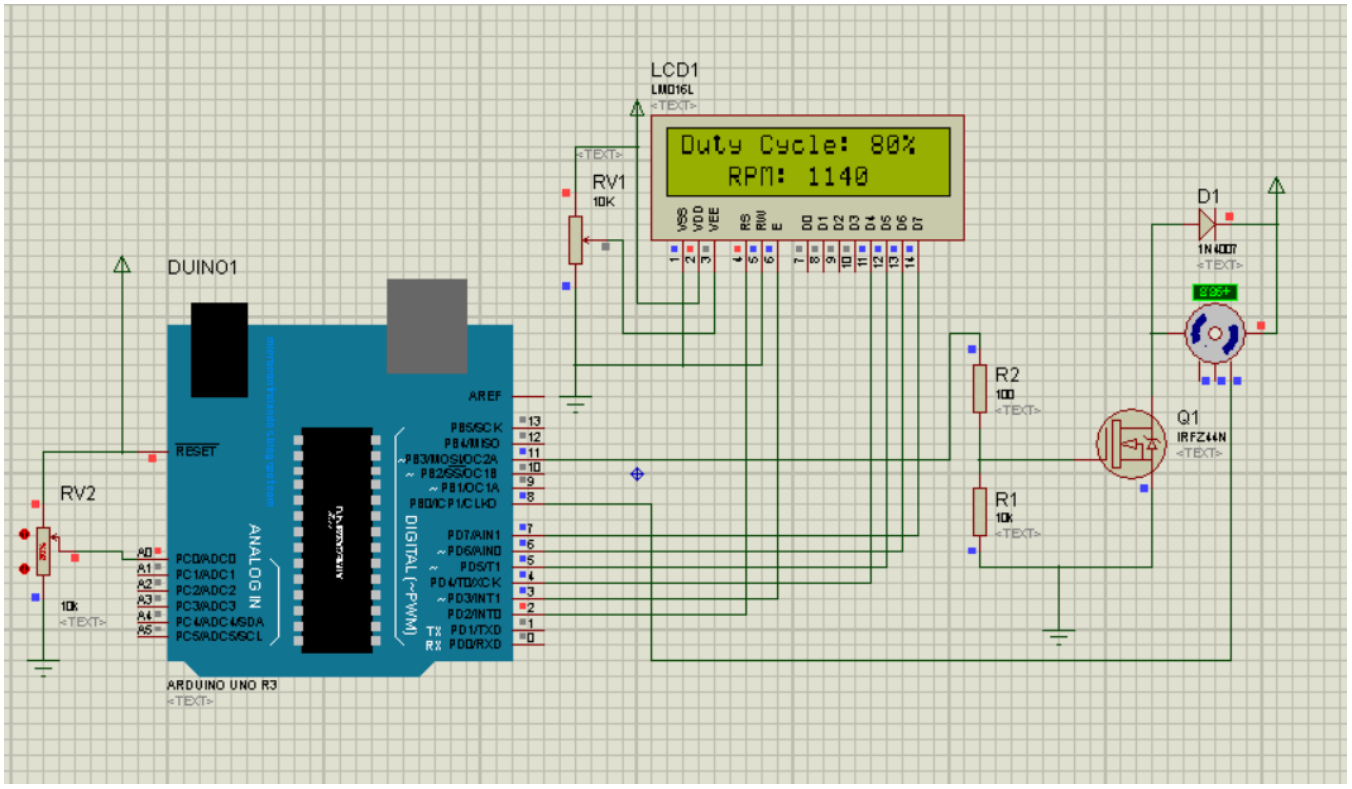
## Software Used

- **Arduino IDE**: Used to write, compile, and upload the program to the Arduino board. It supports C/C++ language with special functions and libraries designed for Arduino.
- **Libraries**:
- LiquidCrystal.h: For interfacing with the LCD.
- Wire.h: If any I2C communication is needed, though not shown here.

## Block Diagram:



## Proteus Simulation:



## Code:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

const int hallPin = 8;

const unsigned long sampleTime = 1000;

const int maxRPM = 10000;

int volume_pin = A0;

int read_ADC;

int pwm_pin = 11;

int duty_cycle;

int duty_cycle_lcd;

void setup() {

  pinMode(hallPin, INPUT);

  pinMode(volume_pin, INPUT);
  pinMode(pwm_pin, OUTPUT);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("WELCOME To GITAM");
  lcd.setCursor(0, 1);
  delay(2000);
  lcd.clear();
}
void loop() {
  int rpm = getRPM();
  read_ADC = analogRead(volume_pin);
  duty_cycle = map(read_ADC, 0, 1023, 50, 255);
  duty_cycle_lcd = map(read_ADC, 0, 1023, 0, 100);
  if (duty_cycle_lcd > 0) {
    analogWrite(pwm_pin, duty_cycle);
  } else {
    digitalWrite(pwm_pin, LOW);
  }
  lcd.setCursor(0, 0);
  lcd.print("Duty Cycle: ");
  lcd.print(duty_cycle_lcd);
  lcd.print("%  ");
  lcd.setCursor(0, 1);
  lcd.print("   RPM: ");
  lcd.print(rpm);
  lcd.print("   ");
  delay(10);
}
int getRPM() {
  // sample for sampleTime in millisecs
  int kount = 0;
  boolean kflag = LOW;
  unsigned long currentTime = 0;
  unsigned long startTime = millis();
  while (currentTime <= sampleTime) {
    if (digitalRead(hallPin) == HIGH) { kflag = HIGH; }
    if (digitalRead(hallPin) == LOW && kflag == HIGH) {
      kount++;
      kflag = LOW;
    }
    currentTime = millis() - startTime;
  }
  int kount2rpm = int(30000. / float(sampleTime)) * kount;
  return kount2rpm;
}
```

## Summary

The project involves setting up an Arduino circuit to control the speed of a DC motor through PWM and reading the motor's RPM via sensors not detailed in the schematic. The desired speed is set using a potentiometer. Feedback on the actual speed is displayed on an LCD, allowing real-time monitoring and adjustment. This setup can be further expanded to include features like motor direction control, and safety features like overload protection. The schematic provided forms the basic electronic foundation for these functions, emphasizing practical application of microcontroller-based control systems in automated environments.