Title of Experiment:

Controlling DC Motor Using Transistor Driver Circuit

Aim:

To control the speed of a DC motor using PWM signals from Arduino through a transistor-based driver circuit.

Apparatus:

- Arduino Uno board
- NPN transistor (e.g., TIP120 or 2N2222)
- DC motor (5V or 9V rated)
- Flyback diode (e.g., 1N4007)
- External power supply (battery or adapter)
- Potentiometer (10kΩ) – for variable control (optional)
- 220Ω resistor (for base of transistor)
- Breadboard and jumper wires
- Arduino IDE installed on PC

Precautions:

1. Use a proper flyback diode across the motor to protect the transistor from voltage spikes.
2. Do not power the motor directly from the Arduino's 5V pin.
3. Ensure correct orientation of transistor terminals (base, collector, emitter).
4. Use an appropriate transistor rated for your motor current.
5. Avoid short circuits; double-check connections before powering the circuit.

Theory:

DC motors convert electrical energy into mechanical motion. The **speed of a DC motor** can be controlled by varying the **average voltage** applied to it, which is achieved using **PWM (Pulse Width Modulation)**.

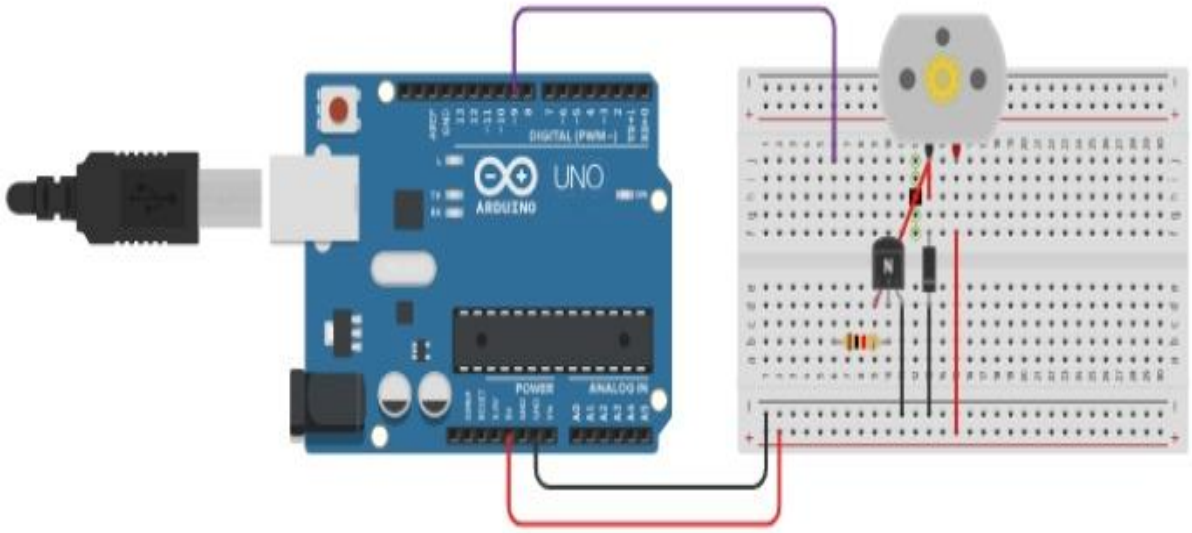
Arduino generates PWM signals using the `analogWrite()` function. However, Arduino's output pins can't supply sufficient current to drive a motor directly. Hence, a **transistor driver circuit** (e.g., NPN transistor) is used to **amplify current**, acting as a switch or amplifier.

A **flyback diode** is connected across the motor terminals to protect the transistor from high voltage spikes generated due to the inductive load.

Circuit Diagram:

Connections:

- Motor (+) → External Power (+)
- Motor (–) → Collector of Transistor
- Emitter of Transistor → GND
- Base of Transistor → D9 (PWM) via 220Ω resistor
- Flyback Diode**: Anode to transistor collector, cathode to motor +**
- Arduino GND and external power GND should be common



Procedure:

1. Connect the DC motor and transistor driver circuit as shown in the diagram.
2. Use a 220Ω resistor between Arduino D9 and the transistor base.
3. Place a flyback diode across the motor terminals.
4. If using a potentiometer, connect it to analog pin A0.
5. Upload the Arduino program to generate PWM using `analogWrite()` to control motor speed.
6. Observe motor speed variation in response to either time or potentiometer input.

Arduino Programming

(A) Using PWM Ramp-Up (automatic speed increase):

```
const int motorPin = 9;  // Motor connected via transistor to PWM pin D9

void setup() {

  pinMode(motorPin, OUTPUT);   // Set motorPin as output

}

void loop() {

  // Gradually increase motor speed from 0 to 255

  for (int speed = 0; speed <= 255; speed++) {

    analogWrite(motorPin, speed);  // Write PWM signal

    delay(20);                  // Small delay for gradual speed up

  }

  delay(1000);  // Keep motor at full speed for a while


  // Gradually decrease motor speed from 255 to 0

  for (int speed = 255; speed >= 0; speed--) {

    analogWrite(motorPin, speed);

    delay(20);

  }


  delay(1000);  // Wait before restarting loop

}
```

Conclusion:

The experiment demonstrated how a DC motor's speed can be controlled using a transistor driver circuit and PWM signal from Arduino. By using either a timed ramp-up or a potentiometer input, real-time speed control of the motor was achieved efficiently and safely.

Questions and Answers:

Q1: Why is a transistor used to drive the DC motor instead of connecting it directly to Arduino?
Answer: Because Arduino cannot supply enough current to drive a motor directly, a transistor acts as a switch or amplifier to control higher current.

Q2: What is the role of the flyback diode in the circuit?
Answer: The flyback diode protects the transistor from high-voltage spikes generated when the motor (an inductive load) is turned off.

Q3: What is PWM, and how does it control motor speed?
Answer: PWM (Pulse Width Modulation) controls motor speed by adjusting the average voltage delivered to the motor through rapid ON-OFF pulses.

Q4: What function is used in Arduino to generate PWM signals?
Answer: The `analogWrite(pin, value)` function is used, where `value` ranges from 0 (0% duty cycle) to 255 (100% duty cycle).