Objective:

The objective of this experiment is to introduce students to the PWM signals generated by Arduino and to control the speed of a DC Motor using a microcontroller-based motor control system. The experiment aims to demonstrate how digital output can be converted to analog output using pulse width Modulation (PWM). The methodology involves using an Arduino to generate PWM signals, which are used to control the speed of a DC motor through a transistor. The duty cycle of the PWM signal is varied to change the speed of the motor. To control the direction of rotation, an H-bridge circuit is used, which contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H like configuration. The experement involves programming an Arduino board using the Arduino Integrated Development Environment (IDE) to implement the motor control system. The program uses functions to control ~~system~~ the direction and speed of the motor.

The apparatus and software name:

- Arduino IDE (any version)
- L298N Driver
- 12V High Torque DC Motor
- Arduino Board
- Potentiometer
- A DC power supply
- Breadboard and Jumper Wires

Theory and Programs:

The theory behind this experiment involves using pulse width modulation (PWM) to generate an analog signal using a digital source, specifically an Arduino microcontroller. PWM is a technique by which the width of a pulse is varied while keeping the frequency or time period of the wave constant. The duty cycle is defined as the ratio ~~of~~ ON pulse duration to the time period, and it is commonly expressed as a percentage or a ratio. The duty cycle is a fraction of one time period and can vary while keeping the period fixed. The motor speed varies according to the duty cycle. In this experement, a transistor is used to drive the motor since the micro-Controller and Arduino can process signals and consume almost 20 to 40 mA current and voltage. The transistor is connected in series with the motor, and the ~~trans~~ transistor's base is connected to Arduino's PWM pin through a resistance. The PWM signal is coming from Arduino and the transistor works as a switch that short circuits the Emitter and collector terminals when the PWM signal is in a HIGH state and normally opens when the PWM signal is in a LOW state.

An H-bridge circuit is used to control the the direction of rotation of the motor. the H-bridge circuit contains four switching elements, transistors, or MOSFETS, with the motor at the center forming an H-like configuration.

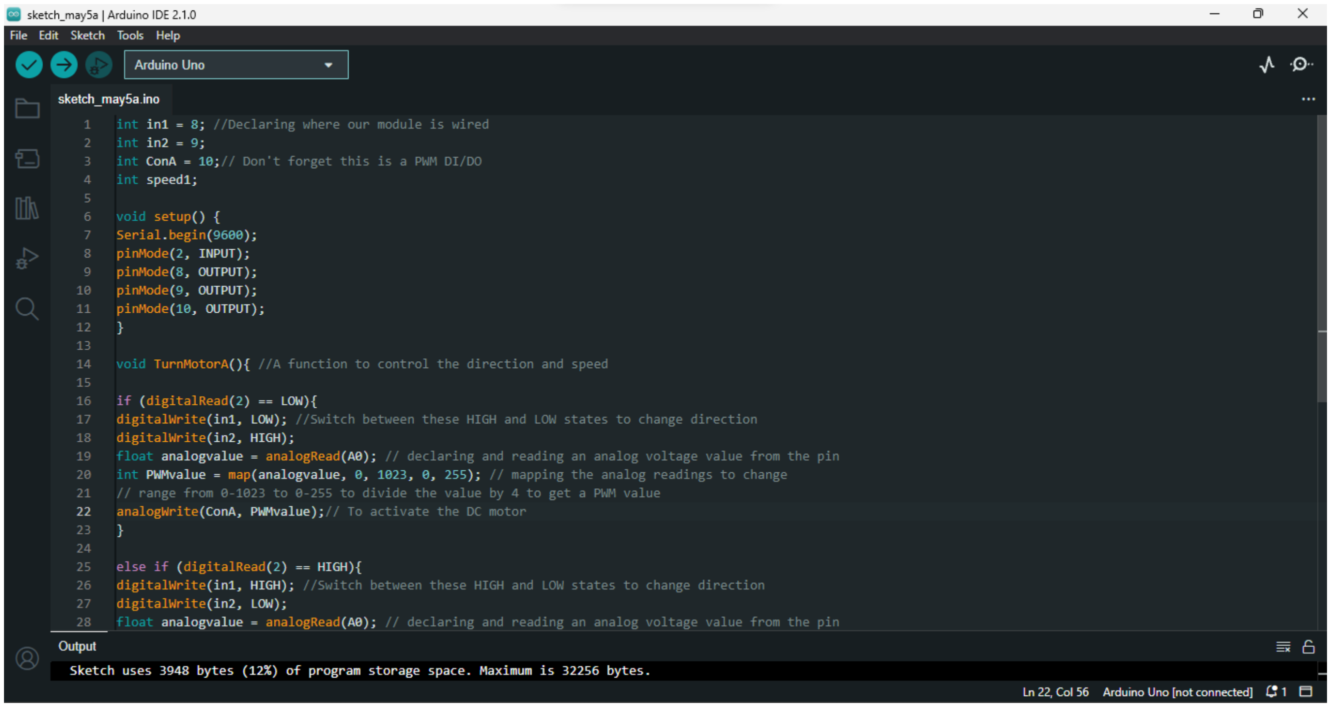
A brief procedure:

- Gather all necessary materials, tools, and equipment required for the procedure.
- set the input/output mode for the necessary pins using the pinMode() function.
- A function called TurnMotorA() is defined, which reads the analog value from pin A0, maps it on a PWM value, and adjusts the motor's direction and speed accordingly.
- In the loop() function, read the analog value from pin A0, convert it to a PWM value, and print the values on the serial monitor.
- Call the TurnMotorA() function to run the motor continously based on the analog input value.
- Additional functions can be added to control the motor in different directions or to stop it.
- Upload the program to the Arduino board using the Arduino IDE and connect the motor driver module to the ~~appor~~ appropriate pins.

• Monitor the progress of the procedure to ensure that it is proceeding as planned, and make any nacessary adjustments if ~~problem~~ problems arise.

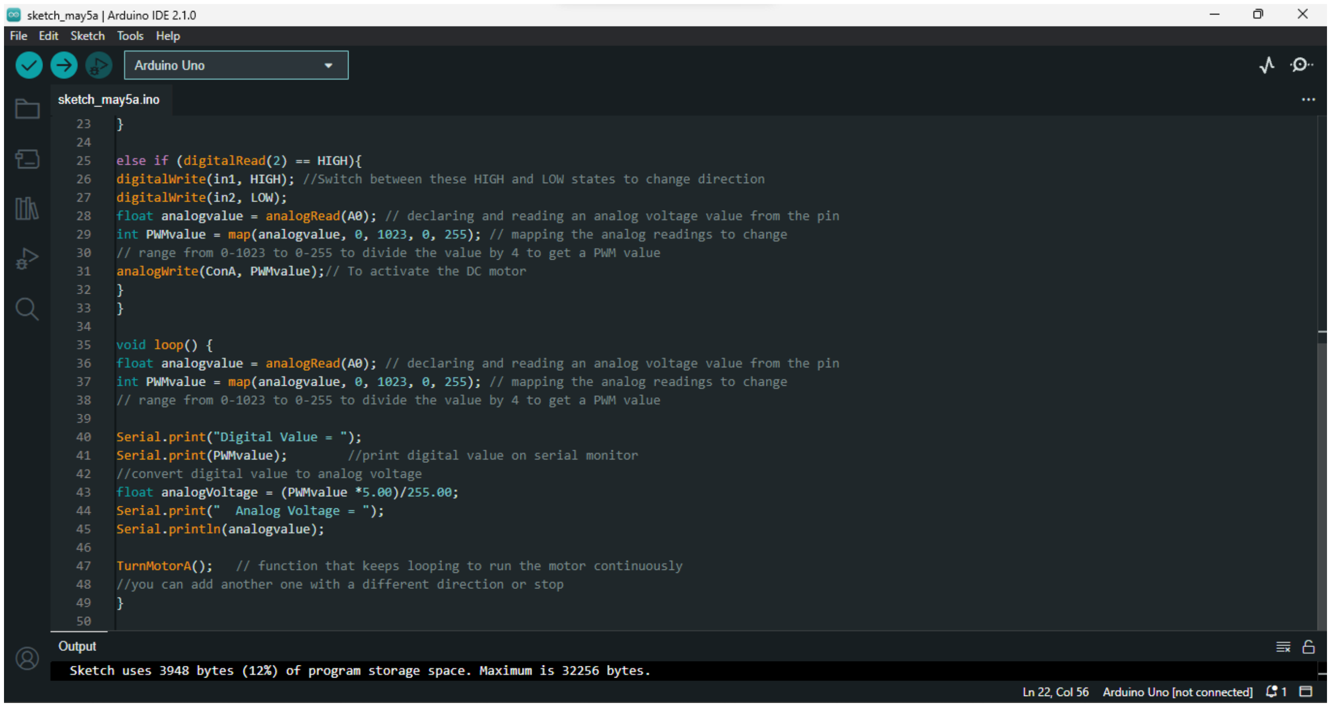
• After the procedure has been completed, evaluate it's effectiveness and identify any areas for improvement. This evaluation can be used to refine and improve the procedure for future use.

Source Code:


sketch_may5a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Arduino Uno
sketch_may5a.ino
Output
Ln 22, Col 56 Arduino Uno [not connected]

```
int in1 = 8; //Declaring where our module is wired
int in2 = 9;
int ConA = 10;// Don't forget this is a PWM DI/DO
int speed1;

void setup() {
Serial.begin(9600);
pinMode(2, INPUT);
pinMode(8, OUTPUT);
pinMode(9, OUTPUT);
pinMode(10, OUTPUT);
}

void TurnMotorA(){ //A function to control the direction and speed

if (digitalRead(2) == LOW){
digitalWrite(in1, LOW); //Switch between these HIGH and LOW states to change direction
digitalWrite(in2, HIGH);
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
analogWrite(ConA, PWMvalue);// To activate the DC motor
}

else if (digitalRead(2) == HIGH){
digitalWrite(in1, HIGH); //Switch between these HIGH and LOW states to change direction
digitalWrite(in2, LOW);
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
```

```
Sketch uses 3948 bytes (12%) of program storage space. Maximum is 32256 bytes.
```


sketch_may5a | Arduino IDE 2.1.0

File Edit Sketch Tools Help

Arduino Uno

sketch_may5a.ino

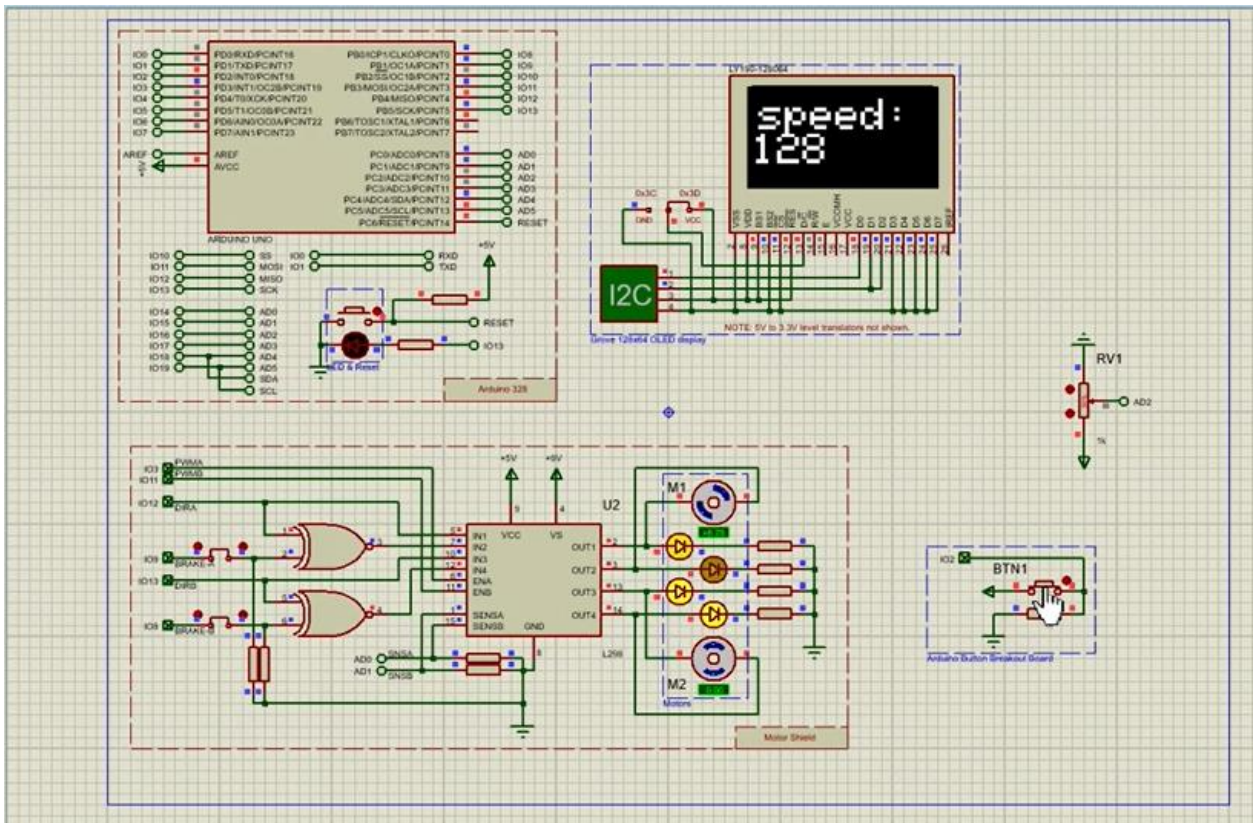
```
}

else if (digitalRead(2) == HIGH){
digitalWrite(in1, HIGH); //Switch between these HIGH and LOW states to change direction
digitalWrite(in2, LOW);
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value
analogWrite(ConA, PWMvalue);// To activate the DC motor
}
}

void loop() {
float analogvalue = analogRead(A0); // declaring and reading an analog voltage value from the pin
int PWMvalue = map(analogvalue, 0, 1023, 0, 255); // mapping the analog readings to change
// range from 0-1023 to 0-255 to divide the value by 4 to get a PWM value

Serial.print("Digital Value = ");
Serial.print(PWMvalue);        //print digital value on serial monitor
//convert digital value to analog voltage
float analogVoltage = (PWMvalue *5.00)/255.00;
Serial.print("  Analog Voltage = ");
Serial.println(analogvalue);

TurnMotorA();   // function that keeps looping to run the motor continuously
//you can add another one with a different direction or stop
}

```

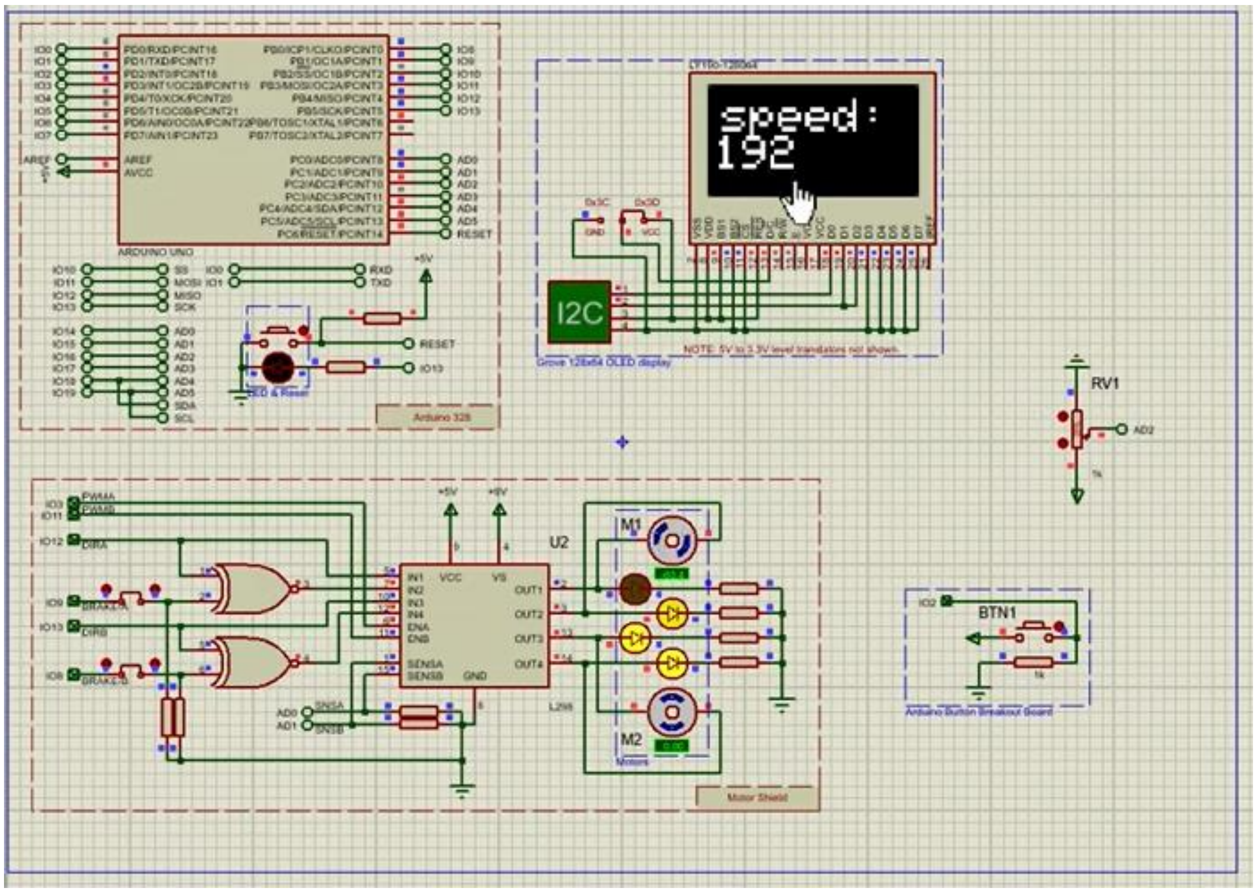
Output

```
Sketch uses 3948 bytes (12%) of program storage space. Maximum is 32256 bytes.
```

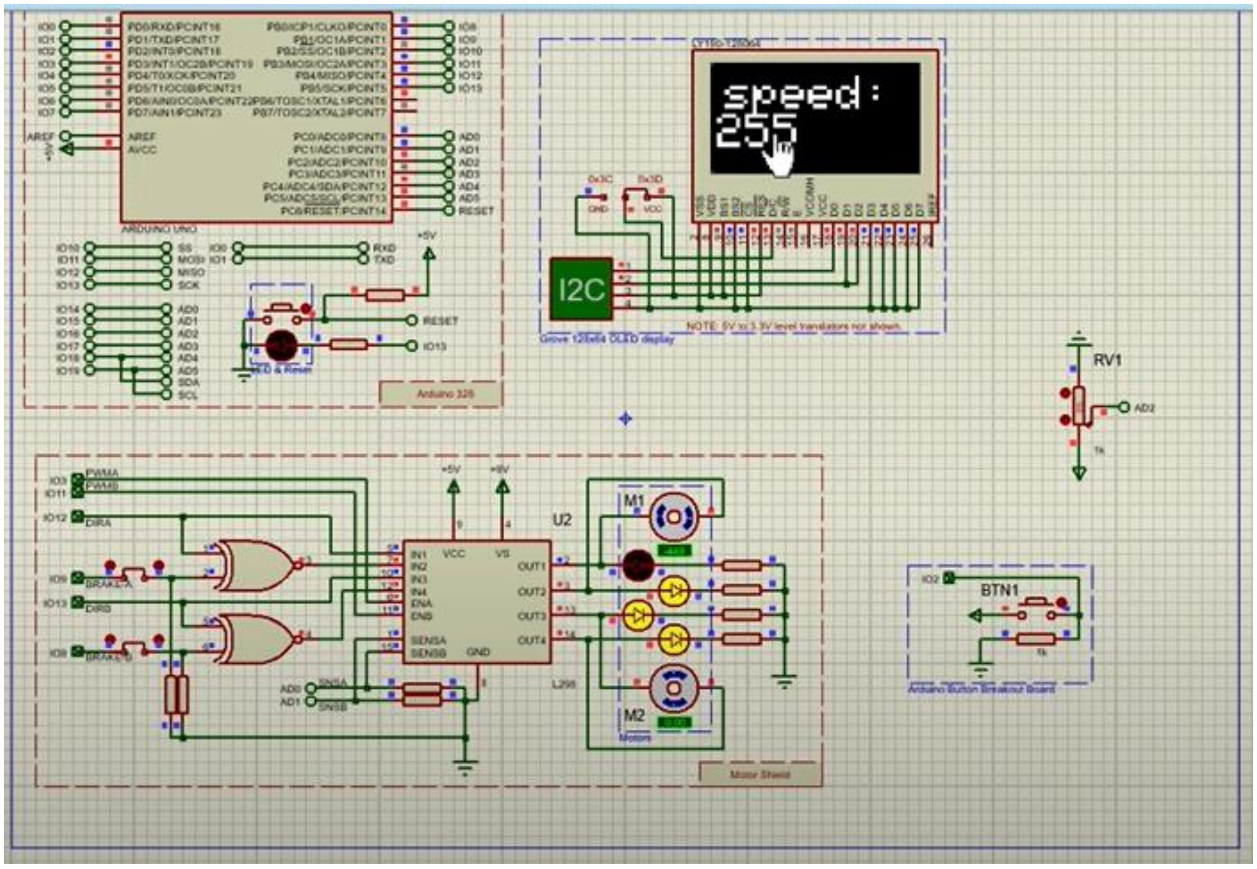
Ln 22, Col 56 Arduino Uno [not connected] 1

<u>Simulation:</u>



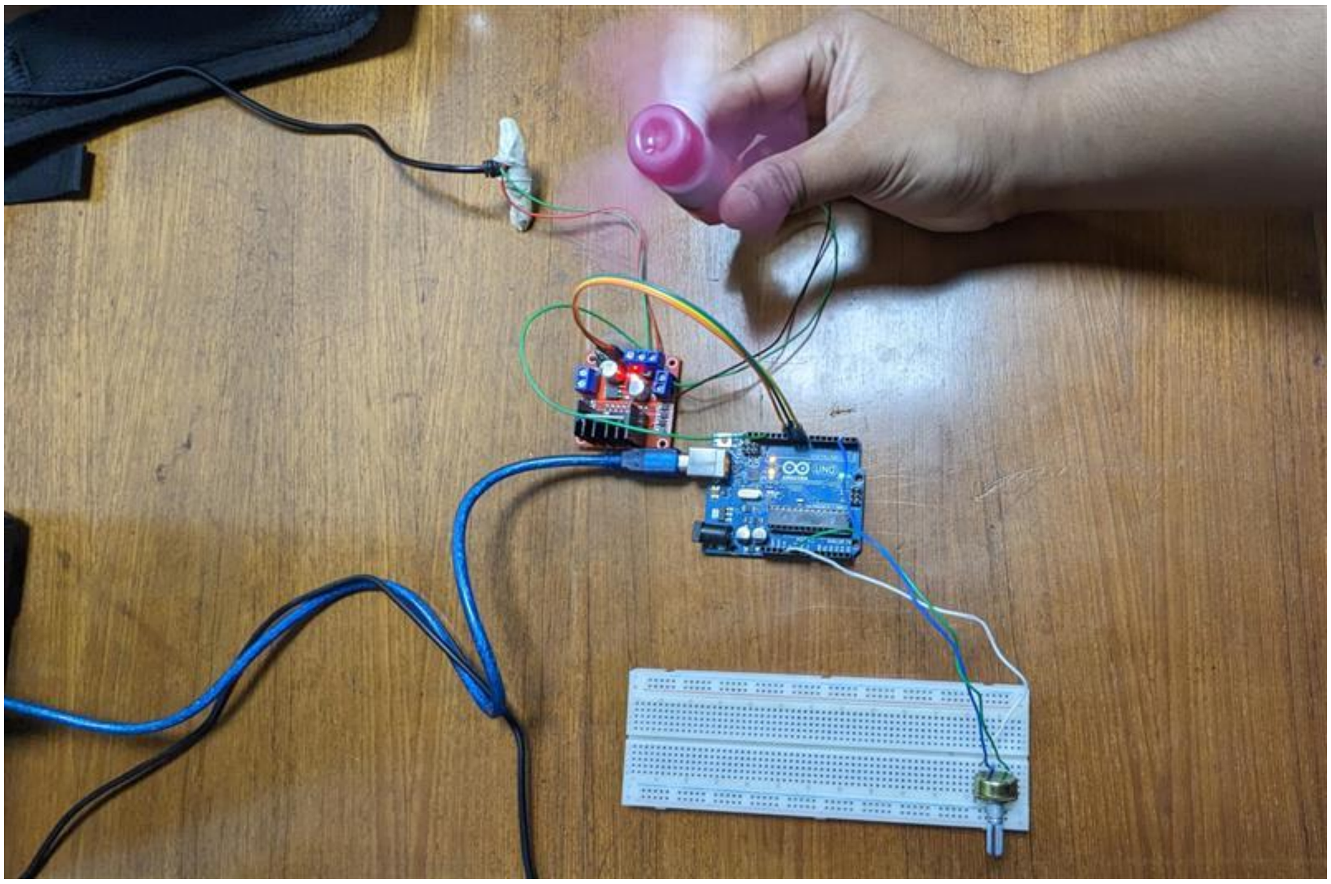


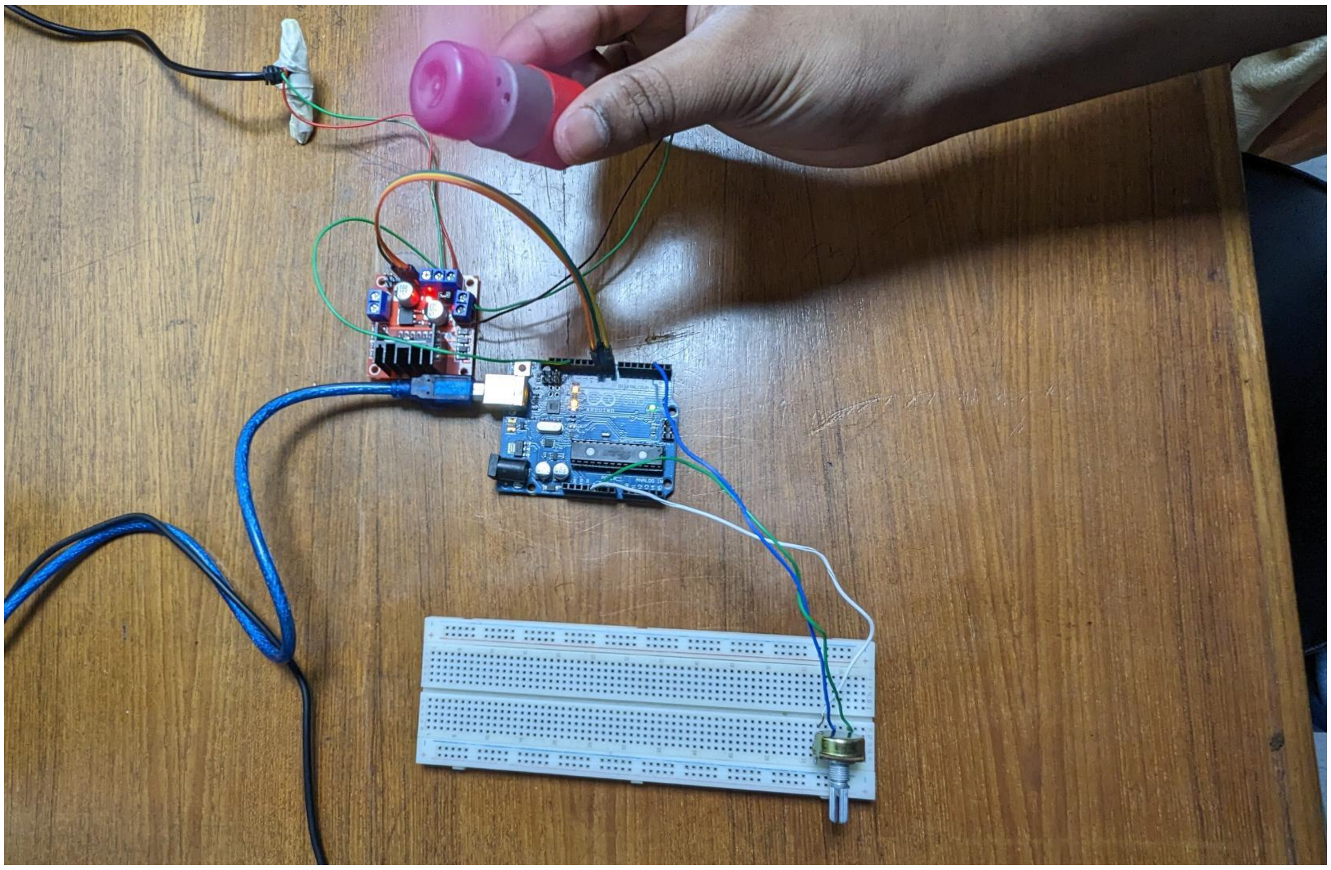
```
speed:
192
```



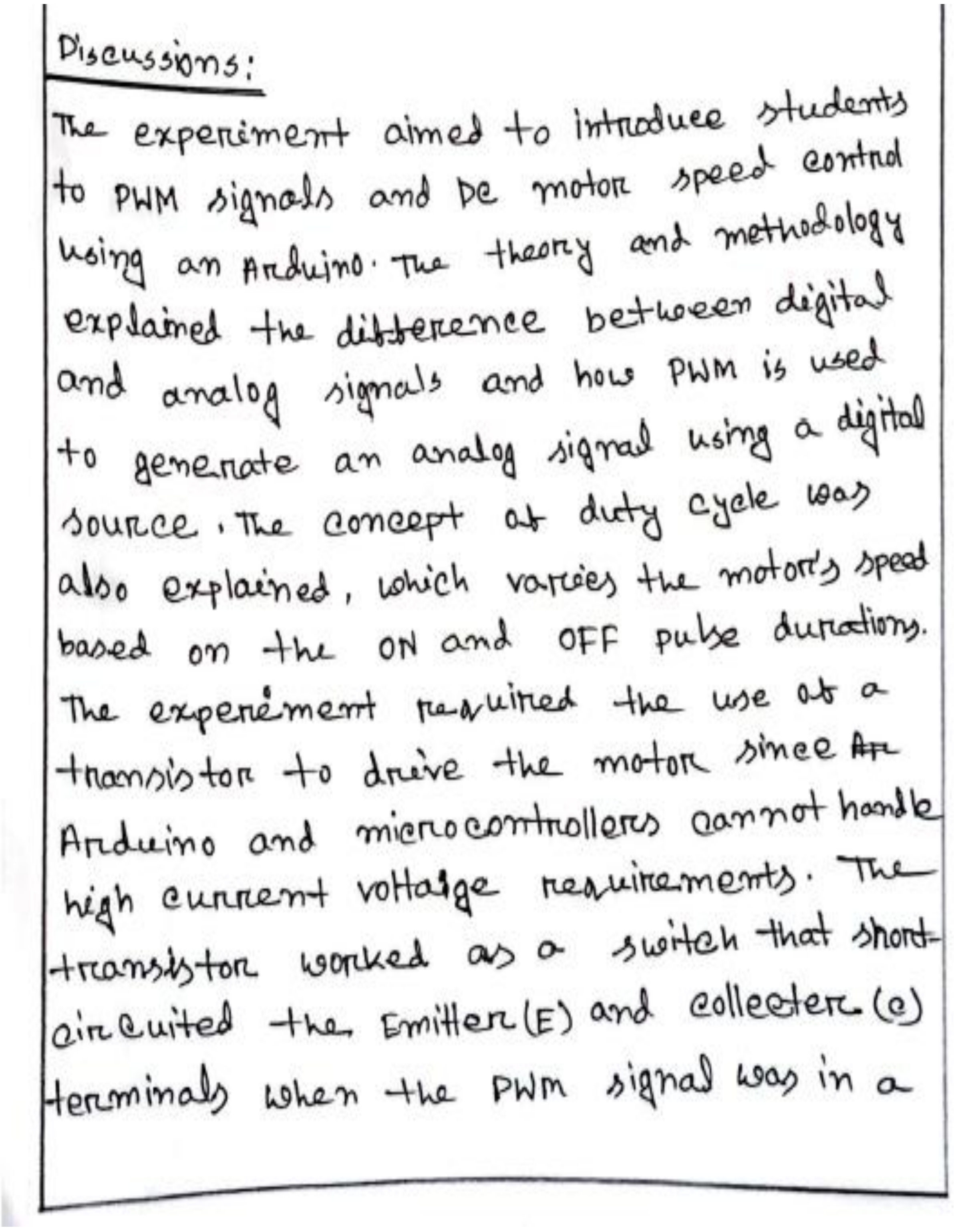
```
speed:
255
```

Lab Simulation:





Report:
All codes, scripts and proteus simulation of the blink program and traffic light system is attached above.



Discussions:

The experiment aimed to introduce students to PWM signals and DC motor speed control using an Arduino. The theory and methodology explained the difference between digital and analog signals and how PWM is used to generate an analog signal using a digital source. The concept of duty cycle was also explained, which varies the motor's speed based on the ON and OFF pulse durations. The experiment required the use of a transistor to drive the motor since ~~Ar~~ Arduino and microcontrollers cannot handle high current voltaige requirements. The transistor worked as a switch that short-circuited the Emitter (E) and collecter (c) terminals when the PWM signal was in a

HIGH state and normally opens when the PWM
signal was in a LOW state. The H-bridge ~~cu~~
~~curcuit~~ circuit was used to control the direction
of the motor rotation, and by activating
two switches at the same time, the
direction of the current flow could be
changed, thus changing the rotation ~~di~~
direction of the motor.

Conclusions:

The experiment provided a basic understan-
ding of PWM signals and DC motor speed
control using an Arduino. The students
learned the difference between digital
and analog signals, the concept of duty
cycle, and how to use PWM to generate
an analog signals, ~~the concept~~ using a
digital source. The experiment also demonstrated
the use of a transistor to drive the motor
and the H-bridge circuit to control the
direction of the motor rotation. The program
written in the ArduinoIDE showed how to
control the direction and speed of the
motor using the analogRead and map functions.
The students learned how to read the analog
voltage value and map the analog readings to
change the range from 0-1023 to 0-255
to get a PWM value. The experiment could

be extended by adding more functionality to the program, such as controlling multiple motors, ~~addign~~ adding sensors to detect the motor's speed and direction, and implementing PID control to maintain a constant speed.