Experiment No: 4
Experiment Name: Control and Analyze the Operation of DC Motors Using Arduino and L293D Motor Driver IC.

Objective:

- To learn how to interface basic hardware with the Arduino microcontroller.
- To develop simple code for hardware control using Arduino IDE.

Theory:

The implementation of robotic systems often requires precise control of DC motors, which are the primary actuators used to produce movement. However, microcontrollers like the Arduino cannot directly drive these motors due to current limitations. The **L293D Motor Driver IC** serves as a crucial interface between the low-power control circuit and the high-power motors. It acts as a bridge that allows a microcontroller to control the direction and speed of motors safely and efficiently.

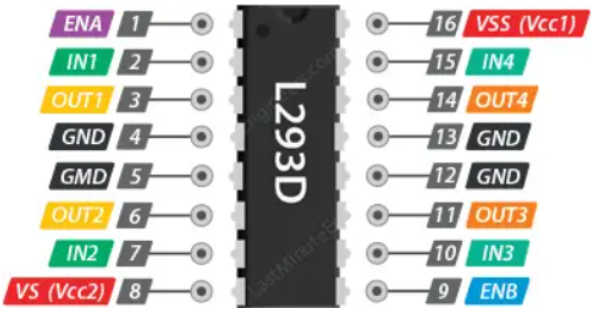

Figure 4.1: L293D Motor Driver IC

Overview of L293D Motor Driver IC:

The L293D is a dual H-Bridge motor driver IC, which means it can drive two DC motors independently in both forward and reverse directions. It operates on a 4.5V to 36V supply range for the motor and has separate logic and motor supply pins. The L293D can supply a maximum current of 600mA per channel, which is suitable for most small DC motors used in educational and hobby-grade robotic systems.
Internally, each H-Bridge is composed of transistor pairs arranged in such a way that changing the input logic levels can reverse the polarity of the motor voltage. This capability is essential for robotic systems that need bidirectional control.

Pin Configuration and Functionality

The IC consists of 16 pins with the following key connections:

1. **Input Pins (IN1, IN2, IN3, IN4):** Receive logic signals from the microcontroller.
2. **Output Pins (OUT1, OUT2, OUT3, OUT4):** Connected to the motor terminals.
3. **Enable Pins (EN1, EN2):** Control whether each motor is enabled. These must be HIGH to allow motor operation.

4. **VCC1 (Logic Voltage):** Typically 5V, powers the internal logic circuit.
5. **VCC2 (Motor Voltage):** Powers the motors (up to 36V).
6. **GND:** Common ground for both logic and motor power.

The basic truth table for motor control using one H-Bridge of the L293D is shown below:

IN1	IN2	Motor Direction
1	0	Forward
0	1	Reverse
1	1	Brake
0	0	Stop

Working Principle

When a HIGH signal is sent to an input pin, the corresponding output pin goes HIGH (assuming the enable pin is also HIGH). By adjusting the combination of HIGH and LOW signals at the input pins, one can control the rotation direction of the motors. This technique is particularly useful for differential drive robotic systems, where varying the direction and speed of two motors allows for forward movement, reverse movement, and turning.

Integration with Arduino for Robotic Control

In a robotic system, the L293D is connected between the Arduino and the DC motors. The Arduino sends digital HIGH/LOW signals to the input pins of the L293D. The enable pins can be tied to HIGH for basic control or connected to PWM outputs of the Arduino to achieve speed control via pulse-width modulation (PWM).
For example, to rotate a robot forward:

1. Left Motor: IN1 = HIGH, IN2 = LOW
2. Right Motor: IN3 = HIGH, IN4 = LOW

To turn left:

1. Left Motor: IN1 = LOW, IN2 = HIGH
2. Right Motor: IN3 = HIGH, IN4 = LOW

The robotic platform can thus navigate through the environment by altering motor signals dynamically based on sensor input or programmed behavior.

Required Apparatus:

Table 1: Components list for the experimental setup

SI No.	**Components**	**Specifications**	**Quantity**
1	Micro-controller Board	Arduino UNO	1
2	Motor Driver	L293D Motor Driver IC	1
3	Power Supply	15V variable DC	1
4	Push Button		1
5	Connecting Wires	Male to Male Connector	1

Circuit Diagram:

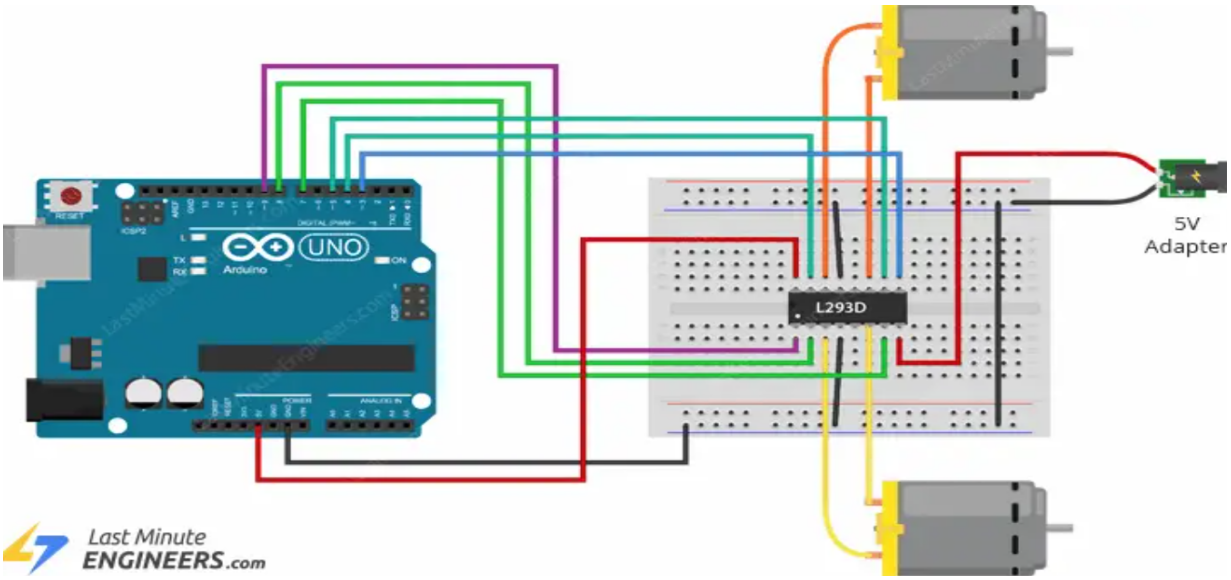

Figure 1: Schematic Diagram

Experimental Setup:

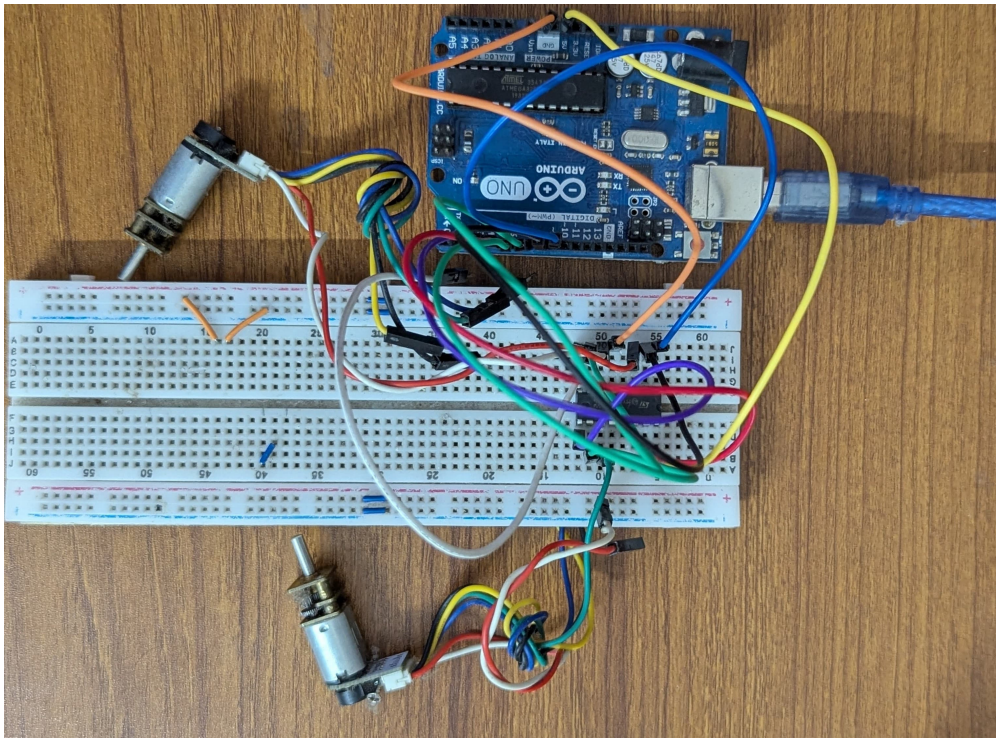

Figure 2: Experimental Setup

Code:

```
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;
// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;
bool a = true;
void setup() {
    // Set all the motor control pins to outputs
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
    pinMode(12, INPUT);
    // Turn off motors - Initial state
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
void loop() {
        directionControl();
        delay(1000);
        speedControl();
        delay(1000);
    }
// This function lets you control spinning direction of motors
void directionControl() {
    // Set motors to maximum speed
    // For PWM maximum possible values are 0 to 255
    analogWrite(enA, 255);
    analogWrite(enB, 255);

    // Turn on motor A & B
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(2000);

    // Now change motor directions
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);
    delay(2000);
    // Turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
```

Listing 1: Basic Motor operation Code

```
// This function lets you control speed of the motors
void speedControl() {
    // Turn on motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);

    // Accelerate from zero to maximum speed
    for (int i = 0; i < 256; i++) {
        analogWrite(enA, i);
        analogWrite(enB, i);
        delay(20);
    }

    // Decelerate from maximum speed to zero
    for (int i = 255; i >= 0; --i) {
        analogWrite(enA, i);
        analogWrite(enB, i);
        delay(20);
    }
    // Now turn off motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}
```

Listing 2: Basic Motor operation Code

Experimental Procedure:

For the basic (main) experiment, the following steps were executed:

1. Connected the L293D motor driver IC to the Arduino UNO as per the circuit diagram.
2. Connected Motor A to pins IN1 (8) and IN2 (7), and Motor B to pins IN3 (5) and IN4 (4) of the Arduino.
3. Connected the enable pins ENA (9) and ENB (3) to the PWM-capable pins of Arduino for speed control.
4. Uploaded the Arduino code to the board using the Arduino IDE.
5. In the `setup()` function, set all motor-related pins as `OUTPUT` and turned motors off initially.
6. In the `loop()` function, first executed the `directionControl()` function to rotate both motors in one direction and then reverse.
7. Used the `speedControl()` function to gradually increase and decrease motor speed using PWM signals.
8. Observed motor behavior during directional and speed control phases.

Hands-on Learning Projects for Mastery of DC Motor Control with Arduino and L293D

Task No.: 01
Task Name: Designing Timed Sequential Control of Dual Motors in a Repetitive Cycle

Objective:

- To implement a time-based motor control system where Motor 1 runs for a specified duration, followed by Motor 2, and then both motors run together, forming a repeatable operational cycle

Code:

```
// Motor A connections
int enA = 9;
int in1 = 8;
int in2 = 7;

// Motor B connections
int enB = 3;
int in3 = 5;
int in4 = 4;

void setup() {
    // Set all motor control pins to outputs
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);

    // Turn off motors initially
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
}

void loop() {
    // Run Motor A for 5 seconds
    analogWrite(enA, 255);
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    delay(5000);
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);

    // Run Motor B for 5 seconds
    analogWrite(enB, 255);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(5000);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
    delay(1000);
```

Listing 3: Task 1 Arduino code

```
    // Run both motors for 10 seconds
    analogWrite(enA, 255);
    analogWrite(enB, 255);
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);
    delay(10000);

    // Stop all motors
    digitalWrite(in1, LOW);
    digitalWrite(in2, LOW);
    digitalWrite(in3, LOW);
    digitalWrite(in4, LOW);
    delay(1000);
}
```

Listing 4: Task 1 Arduino code

Task No.: 02
Task Name: Implementing Conditional Activation in Timed Dual Motor Control

Objective:

- To design a control system where Motor 2 activates only upon receiving a user input or signal after Motor 1 stops, followed by simultaneous motor operation for a fixed duration, in a continuous cycle.

Code:

```
    // Motor A connections
    int enA = 9;
    int in1 = 8;
    int in2 = 7;
    // Motor B connections
    int enB = 3;
    int in3 = 5;
    int in4 = 4;
    bool a = true;
    void setup() {
        // Set all the motor control pins to outputs
        pinMode(enA, OUTPUT);
        pinMode(enB, OUTPUT);
        pinMode(in1, OUTPUT);
        pinMode(in2, OUTPUT);
        pinMode(in3, OUTPUT);
        pinMode(in4, OUTPUT);
        pinMode(12, INPUT);
        // Turn off motors - Initial state
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
    }
```

Listing 5: Task 2 Arduino code

```
    void loop() {
        if (a == true) {
            analogWrite(enA, 255);
            digitalWrite(in1, HIGH);
            digitalWrite(in2, LOW);
            delay(5000);
            a = false;
            digitalWrite(in1, LOW);
            digitalWrite(in2, LOW);
        }
        if (digitalRead(12) == HIGH) {
            analogWrite(enB, 255);
            // Motor 2 run 5 sec
            digitalWrite(in3, HIGH);
            digitalWrite(in4, LOW);
            delay(5000);
            digitalWrite(in3, LOW);
            digitalWrite(in4, LOW);
            delay(1000);

            // Now run both motors for 10 seconds
            analogWrite(enA, 255);
            analogWrite(enB, 255);
            digitalWrite(in1, HIGH);
            digitalWrite(in2, LOW);
            digitalWrite(in3, HIGH);
            digitalWrite(in4, LOW);
            delay(10000);

            digitalWrite(in1, LOW);
            digitalWrite(in2, LOW);
            digitalWrite(in3, LOW);
            digitalWrite(in4, LOW);
            delay(1000);
            a = true;
        }
    }
```

Listing 6: Task 2 Arduino code

Task No.: 03
Task Name: Developing an Interlock-Based Safety Control in Dual Motor Sequencing

Objective:

- To integrate a safety mechanism into the motor control sequence where the entire process halts upon detecting a danger signal, while preserving conditional and timed motor operation.

Code:

```
// Motor A connections & Motor B connections
int enA = 9;
int in1 = 8;
int in2 = 7;
int enB = 3;
int in3 = 5;
int in4 = 4;
const int dangerPin = 12; // Danger button pin
bool stageA_done = false;
bool paused = false;
unsigned long lastTime = 0;
int state = 0;

void setup() {
    // Set motor pins as outputs
    pinMode(enA, OUTPUT);
    pinMode(enB, OUTPUT);
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    pinMode(in3, OUTPUT);
    pinMode(in4, OUTPUT);
    pinMode(dangerPin, INPUT);
    stopAllMotors();
}
void loop() {
    // Check danger signal
    if (digitalRead(dangerPin) == HIGH) {
        stopAllMotors();
        paused = true;
        return;  // Exit loop early if paused
    }
    if (paused) {
        // Wait until dangerPin is LOW to resume
        if (digitalRead(dangerPin) == LOW) {
            paused = false;
            delay(200);  // Debounce
        } else {
            return;
        }
    }
    switch (state) {
        case 0:  // Run Motor A for 5 sec (only once per full cycle)
        analogWrite(enA, 255);
        digitalWrite(in1, HIGH);
        digitalWrite(in2, LOW);
        lastTime = millis();
        state = 1;
        break;

        case 1:  // Wait while Motor A runs
        if (millis() - lastTime >= 5000) {
            digitalWrite(in1, LOW);
            digitalWrite(in2, LOW);
            state = 2;
        }
        break;
```

Listing 7: Task 3 Arduino code

```
            case 2:  // Wait for danger button press to begin next phase
            if (digitalRead(dangerPin) == HIGH) {
                paused = true;
                return;
            }

            // Now run Motor B for 5 sec
            analogWrite(enB, 255);
            digitalWrite(in3, HIGH);
            digitalWrite(in4, LOW);
            lastTime = millis();
            state = 3;
            break;

            case 3:  // Wait while Motor B runs
            if (millis() - lastTime >= 5000) {
                digitalWrite(in3, LOW);
                digitalWrite(in4, LOW);
                delay(1000);
                state = 4;
            }
            break;

            case 4:  // Run both motors for 10 sec
            analogWrite(enA, 255);
            analogWrite(enB, 255);
            digitalWrite(in1, HIGH);
            digitalWrite(in2, LOW);
            digitalWrite(in3, HIGH);
            digitalWrite(in4, LOW);
            lastTime = millis();
            state = 5;
            break;

            case 5:  // Wait while both motors run
            if (millis() - lastTime >= 10000) {
                stopAllMotors();
                delay(1000);
                state = 0;  // Reset for next cycle
            }
            break;
        }
    }
    void stopAllMotors() {
        digitalWrite(in1, LOW);
        digitalWrite(in2, LOW);
        digitalWrite(in3, LOW);
        digitalWrite(in4, LOW);
    }
```

Listing 8: Task 3 Arduino code

Results:

Task 1 :

1. Motor A: 5 seconds

2. Motor B: 5 seconds
3. Both Motors: 10 seconds

Task 2 :

1. After Motor A finishes:
2. Wait for input to activate Motor B.
3. Then proceed to dual motor run for 10 seconds.

Task 3 :

1. State 0: Run Motor A (5s)
2. State 1: Wait for completion
3. State 2: Wait for signal to run Motor B
4. State 3: Run Motor B (5s)
5. State 4: Run both motors (10s)
6. State 5: Complete cycle and reset

Discussions:

In the shop practice , from task 1 it was clearly seen that the system was fully automatic as there was no user input to perform specific operation.
From the task 2 it was seen that one of the motor run for 5 seconds and wait for user input button to be pressed. Here we introduced a Boolean flag 'a' to control first-run condition. After pressing the button the conditional and timed operations were performed and this process was continued by changing the flag states.
From task 3 it was seen that due to additional safety pin, here we introduced millis() instead of delay() for non-blocking timing operations. As there was two user input to perform, we used switch conditions where five states were used to check external input and perform the conditional operations.

Conclusion:

Task 1 was fully automatic, timed sequence operations and there was no user input to perform. It was simple and cyclic. Task 2 was conditional and timed operations which was interactive due to user input.But due to delay() functions it didn't detect user input during delay operation. Task 3 was safety interlock operation which stops on danger signal. It was responsive. Due to millis() functions, it was non-blocking operation. To preserve the previous operations , some flag was introduced. It was safe and complex logical operations.

Reference:

1. https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/