February 5, 2025

Report

Kriti ’25

Rain Rain Go Away

Author ID:
Hostel 78

Contents

1 Circuit Schematics

1.1 Comparator Circuit

The Input from the rain sensor goes through the comparator and gives the output signals. The output signal from here goes to a digital logic to convert into control voltage of switch circuit for enable, input 1 and input 2 circuits. Speedometer also uses the similar circuit from the input of motor driver enable or motor voltage circuit.

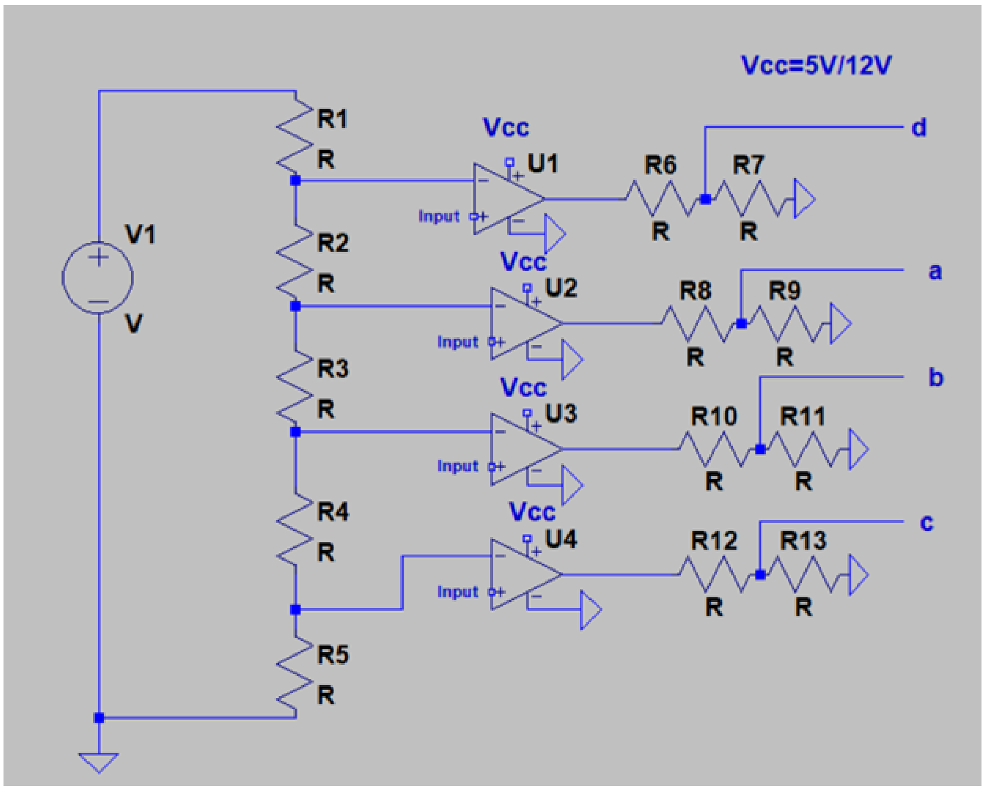

Figure 1: Comparator circuit

1.2 Digital Logic

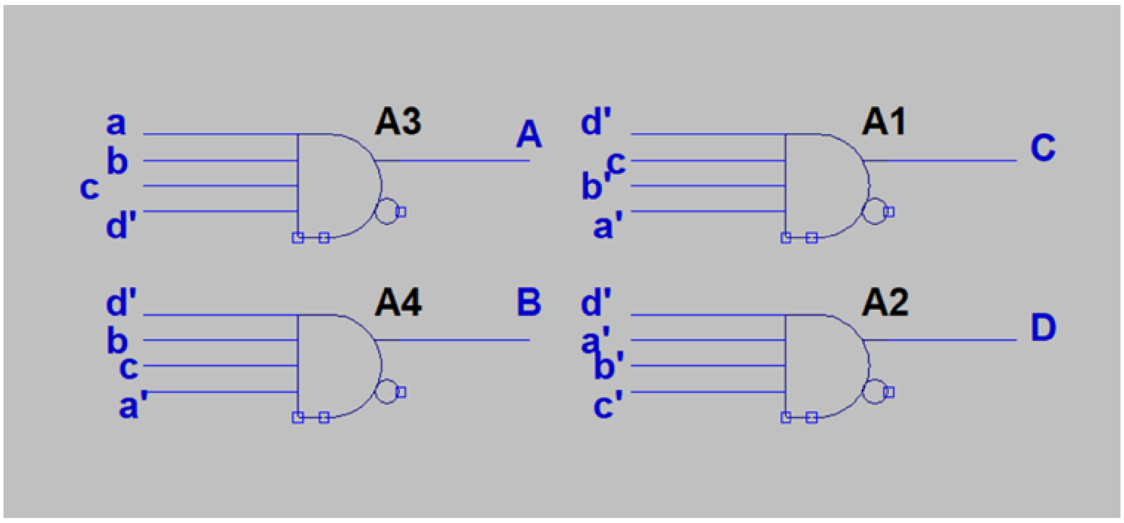

Figure 2: Digital logic for A4, A3, A2 and A1

1.3 Non-Inverting Amplifier

For more ease to compare voltages we use the non inverting amplifier to amplify the input signal from rain sensor to comparator circuit. And we also use the amplifier for amplifying the output signal from comparator circuit to Motor Voltage circuit.

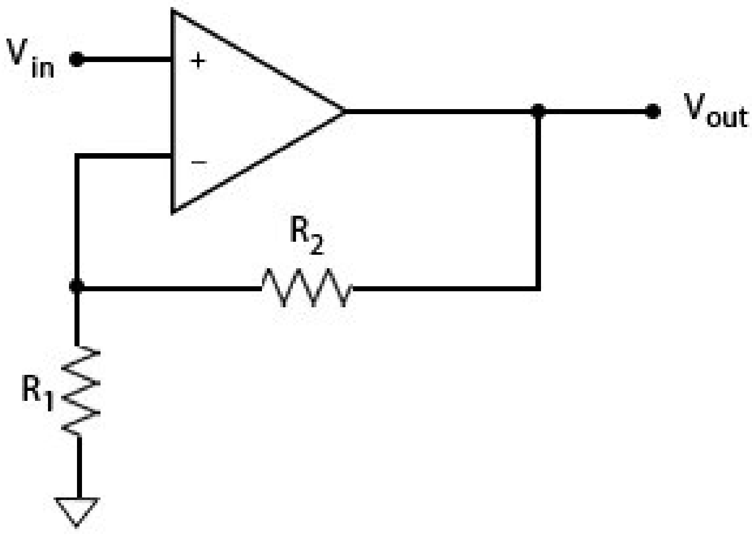

Figure 3: Non-Inverting Amplifier

1.4 Input 1,2 Circuit

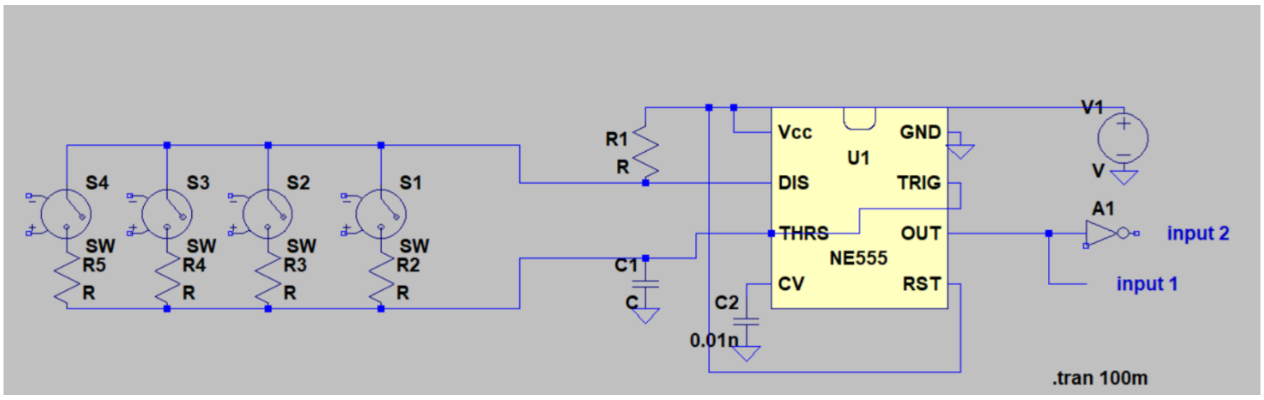

Figure 4: Above is Input circuit made using NE555 and combination of input sinals from comparator circuit to set the frequency of rotation of motor.

2 Speed Control Mechanisms

Two primary methods for motor speed control:

2.1 Buck Converter Method

Provides steady 5V to the enable pin and varies motor supply voltage (12V, 9V, 7V, 5V) to control motor speed.

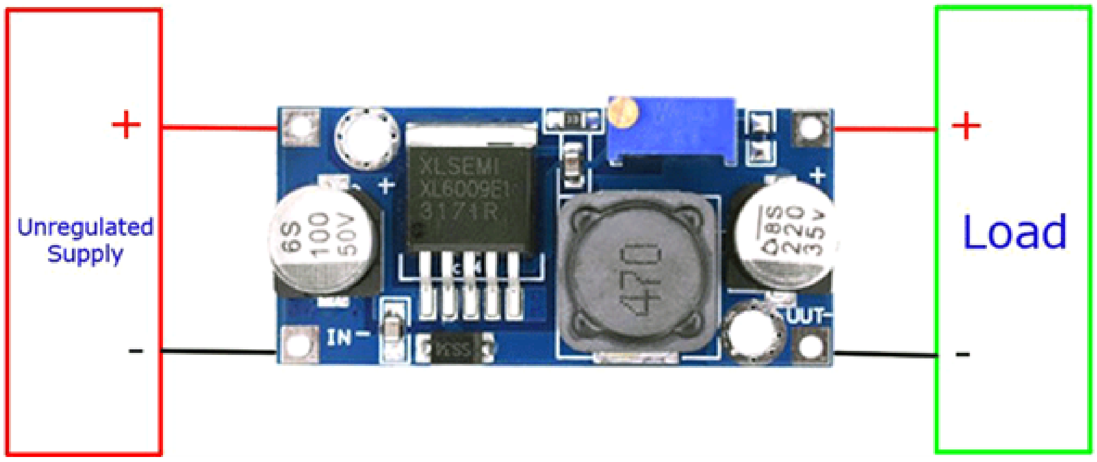

Figure 5: Motor Voltage circuit using Buck Converter.In this circuit we supply a steady voltage of 5V for enable pin. We change the motor supply voltage using buck converter for voltages of 12V,9V,7V,5V which changes the speed of motor.

2.2 NE555 Enable Circuit

Utilizes NE555 to change duty cycle and modify voltage magnitude supplied to L293D for different motor speeds.

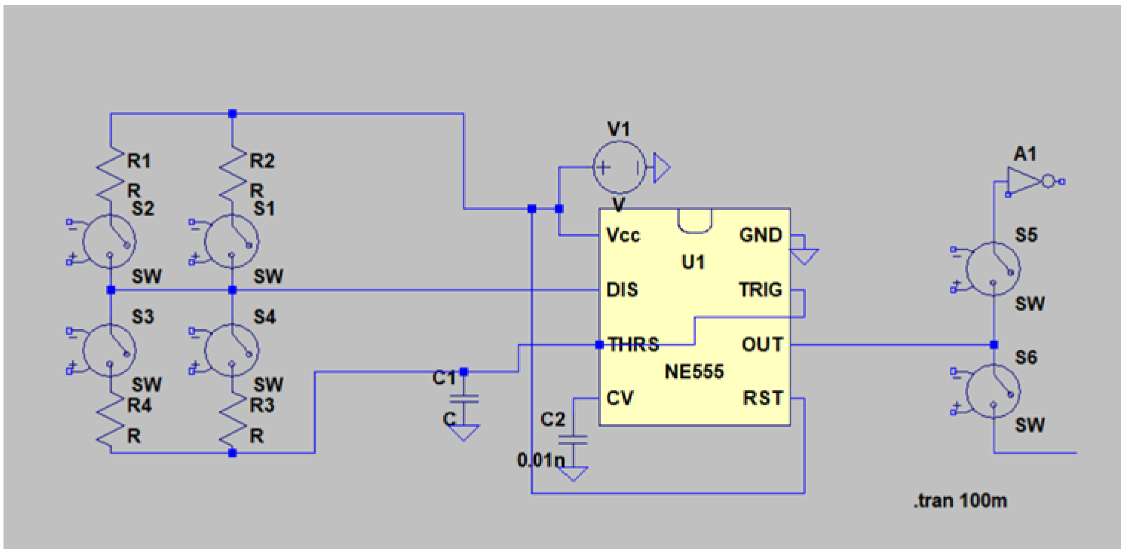

Figure 6: Above is the enable circuit made using NE555. The combination of input signals set the resistors and not gate for output. Here, it works on the principal of changing duty cycle for changing magnitude of voltage give to L293D for different speeds. Here we supply Motor Voltage a constant supply of 12V.

2.3 Motor Driver L293D

Implements H-bridge principles for motor direction and speed control.

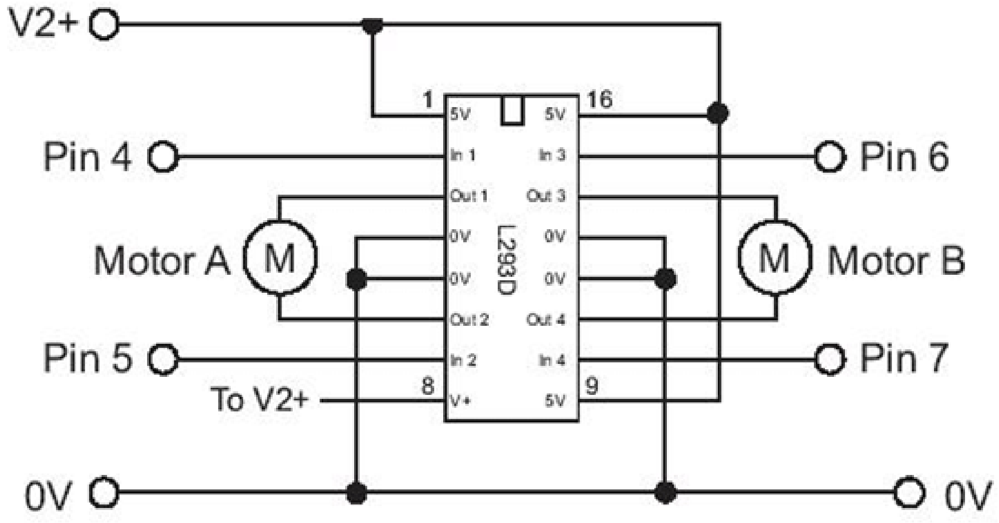

Figure 7: Here in pin 1 we give the enable circuit, input 1 in pin 4 and input 2 in pin 5,Motor voltage in pin 8. This motor driver uses H-bridge principles for changing the direction of motor using input 1 and 2 signals.

2.4 Wireless Braking System

Utilizes a 4-channel Remote Control Transmitter Receiver, connected to the rain sensor comparator circuit via an OR gate.

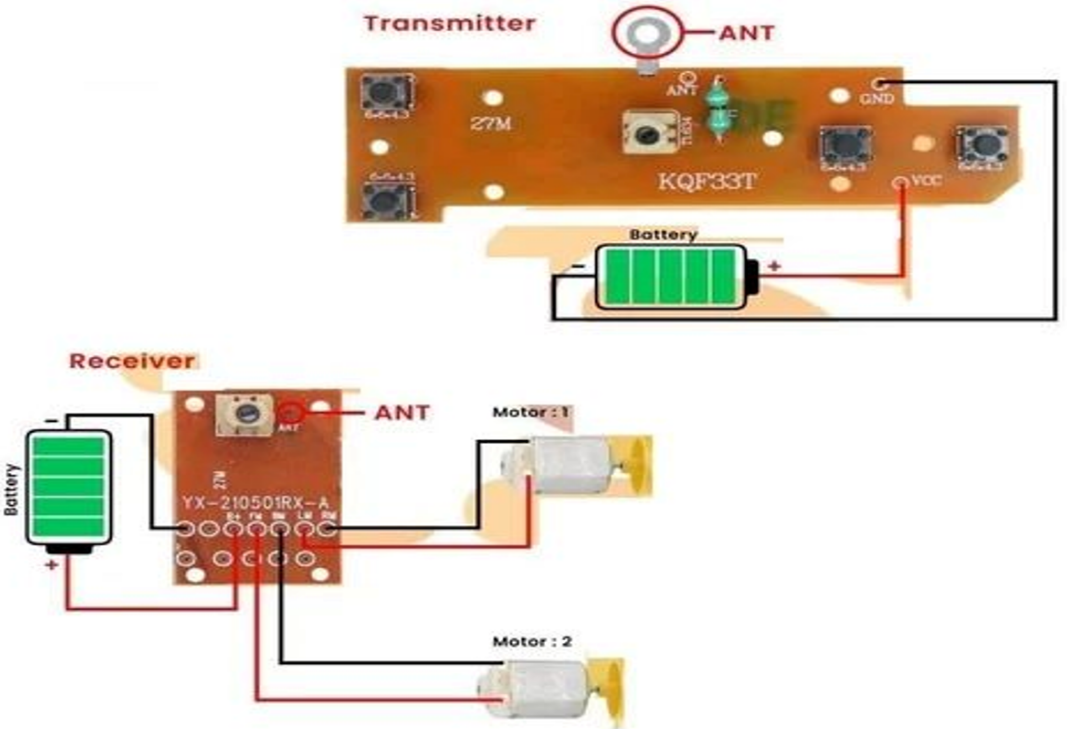

Figure 8: We are using 4ch Remote control Transmitter Receiver for the wireless braking system by connecting one of the receiver wires to the rain sensor comparator circuit's component by an or gate.

2.5 Block Diagram

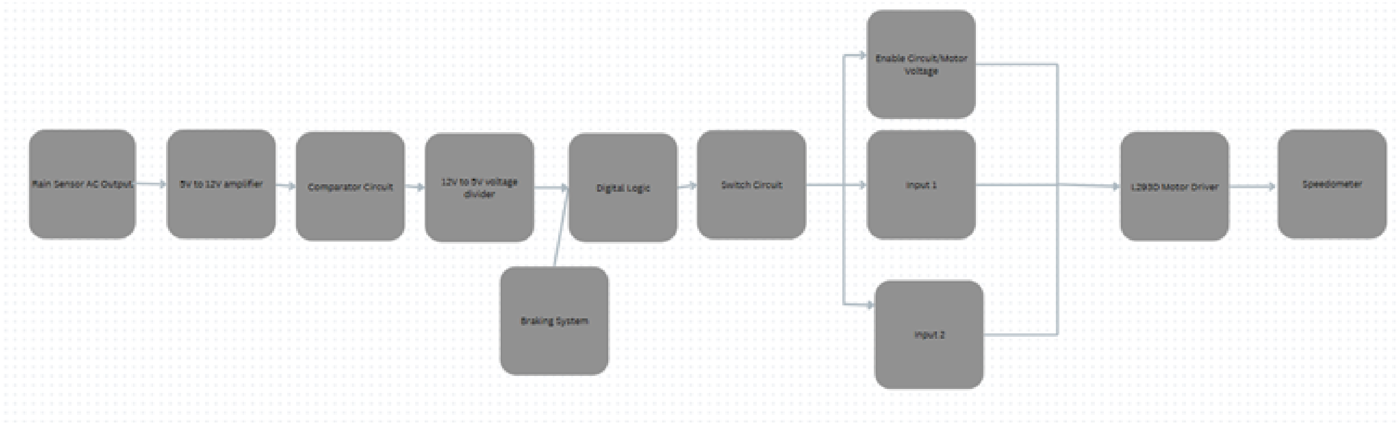

Figure 9: We are using 4ch Remote control Transmitter Receiver for the wireless braking system by connecting one of the receiver wires to the rain sensor comparator circuit’s component by an or gate.

3 Analog Reading Version

```
#include <Servo.h>

% Define LED pins using analog inputs
const int led1 = A0; // LED 1 connected to analog pin A0
const int led2 = A1; // LED 2 connected to analog pin A1
const int led3 = A2; // LED 3 connected to analog pin A2
const int led4 = A3; // LED 4 connected to analog pin A3

% Define servo control pin
const int servoPin = 9; // Servo signal pin

% Create servo object for angle control
Servo myServo;

% Define predefined angles for different LED intensities
const int angle1 = 30;  // Low-intensity LED angle
const int angle2 = 60;  // Medium-low intensity LED angle
const int angle3 = 120; // Medium-high intensity LED angle
const int angle4 = 150; // High-intensity LED angle

% Initialize pins and communication
void initializePins() {
    % Attach servo to specific pin
    myServo.attach(servoPin);

    % Set initial servo position to 0 degrees
    myServo.write(0);

    % Start serial communication for debugging
    Serial.begin(9600);
}

% Calculate average analog reading to reduce noise
int getAverageAnalogRead(int pin) {
    long sum = 0;
    % Take 10 readings with small delay between
    for (int i = 0; i < 10; i++) {
        sum += analogRead(pin);
        delay(10); % Small delay to stabilize readings
    }
    % Return average of 10 readings
    return sum / 10;
}

% Determine servo angle based on LED intensity
int getGlowingLedAngle() {
    % Debug: Print analog readings of specific LEDs
    Serial.print("LED4: ");
    Serial.println(getAverageAnalogRead(led4));
    Serial.print("LED3: ");
    Serial.println(getAverageAnalogRead(led3));

    % Check LEDs in priority order (LED4 -> LED1)
    if (getAverageAnalogRead(led4) >= 200) {
        return angle4; % Highest intensity LED
    } else if (getAverageAnalogRead(led3) >= 200) {
        return angle3; % High intensity LED
    } else if (getAverageAnalogRead(led2) >= 200) {
        return angle2; % Medium intensity LED
    } else if (getAverageAnalogRead(led1) >= 200) {
        return angle1; % Low intensity LED
    } else {
        return 0; % No LED glowing, default position
    }
}

% Move servo to specified angle
void moveServoToAngle(int angle) {
    % Debug: Print target angle
    Serial.print("Moving servo to angle: ");
    Serial.println(angle);

    % Write angle to servo
    myServo.write(angle);
    delay(50); % Stabilize servo movement
}

% Setup function runs once
void setup() {
    initializePins();
}

% Main loop runs continuously
void loop() {
    % Get target angle based on LED intensity
    int targetAngle = getGlowingLedAngle();

    % Move servo to calculated angle
    moveServoToAngle(targetAngle);
}
```

Listing 1: Speedometer with Analog LED Reading

4 Digital Reading Version

```
#include <Servo.h>

% Define LED pins as digital inputs
const int led1 = 2; // LED 1 connected to digital pin 2
const int led2 = 3; // LED 2 connected to digital pin 3
const int led3 = 4; // LED 3 connected to digital pin 4
const int led4 = 5; // LED 4 connected to digital pin 5

% Define servo control pin
const int servoPin = 9; // Servo signal pin

% Create servo object for angle control
Servo myServo;

% Define predefined angles for different LED states
const int angle1 = 30;  // Low-intensity LED angle
const int angle2 = 60;  // Medium-low intensity LED angle
const int angle3 = 120; // Medium-high intensity LED angle
const int angle4 = 150; // High-intensity LED angle
const int defaultAngle = 0; % Default angle if no LED is active

% Initialize pins and communication
void initializePins() {
    % Set LED pins as input
    pinMode(led1, INPUT);
    pinMode(led2, INPUT);
    pinMode(led3, INPUT);
    pinMode(led4, INPUT);

    % Attach servo to specific pin
    myServo.attach(servoPin);

    % Set initial servo position
    myServo.write(defaultAngle);

    % Start serial communication for debugging
    Serial.begin(9600);
}

% Determine servo angle based on LED digital state
int getGlowingLedAngle() {
    % Check LEDs in priority order (LED4 -> LED1)
    if (digitalRead(led4) == HIGH) {
        Serial.println("LED4 is HIGH, setting angle to 150");
        return angle4; % Highest priority LED
    } else if (digitalRead(led3) == HIGH) {
        Serial.println("LED3 is HIGH, setting angle to 120");
        return angle3; % High priority LED
    } else if (digitalRead(led2) == HIGH) {
        Serial.println("LED2 is HIGH, setting angle to 60");
        return angle2; % Medium priority LED
    } else if (digitalRead(led1) == HIGH) {
        Serial.println("LED1 is HIGH, setting angle to 30");
        return angle1; % Low priority LED
    } else {
        Serial.println("No LED is HIGH, setting angle to 0");
        return defaultAngle; % No LED active
    }
}

% Move servo to specified angle
void moveServoToAngle(int angle) {
    % Debug: Print target angle
    Serial.print("Moving servo to angle: ");
    Serial.println(angle);

    % Write angle to servo
    myServo.write(angle);
    delay(50); % Stabilize servo movement
}

% Setup function runs once
void setup() {
    initializePins();
}

% Main loop runs continuously
void loop() {
    % Get target angle based on LED digital state
    int targetAngle = getGlowingLedAngle();

    % Move servo to calculated angle
    moveServoToAngle(targetAngle);
}
```

Listing 2: Speedometer with Digital LED Reading

5 Components List

- LM339 Comparator
- LM358 Operational Amplifier
- UA741 Operational Amplifier
- Logic Gates: AND, NOT, OR
- LED Indicators
- NE555 Timer IC
- L293D Motor Driver
- DC Motor
- IC4066 Analog Switch
- Buck Converter
- Passive Components: Resistors, Capacitors
- Breadboard
- Arduino Microcontroller
- Connecting Wires
- Transistors (NPN & PNP)
- 4-Channel Remote Control Transmitter Receiver