ARDUINO PROJECT

Abstract

Motor speed control based on temperature

Nourhan Mahmoud Hussien

nourhan3993@gmail.com

Project definition:

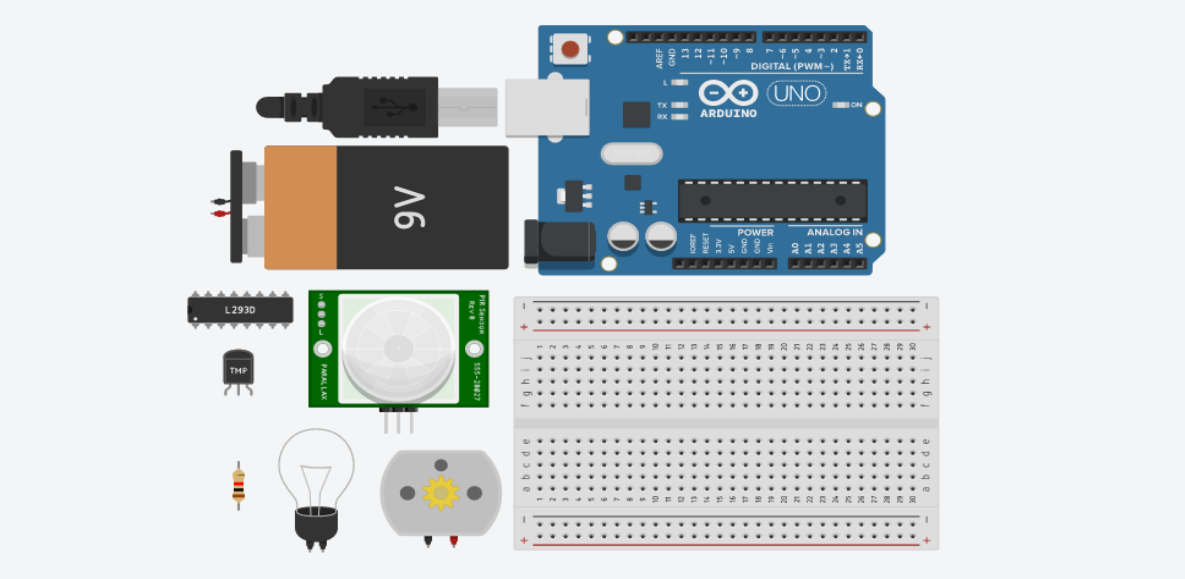
A temperature sensor that measures the temperature and, based on it, controls the speed of the motor so that:

1. temperature is less than 20 >> The motor stops
2. Temperature equal 20 >> The motor rotates at 50%
3. The temperature is between 20 and 40 >> The motor rotates at a speed proportional to the degree
4. The temperature is greater than 40 >> The motor is belted at the highest possible speed

In addition to the PIR sensor, when there is movement, it will turn on the bulb, even if it does not turn it off

Component:

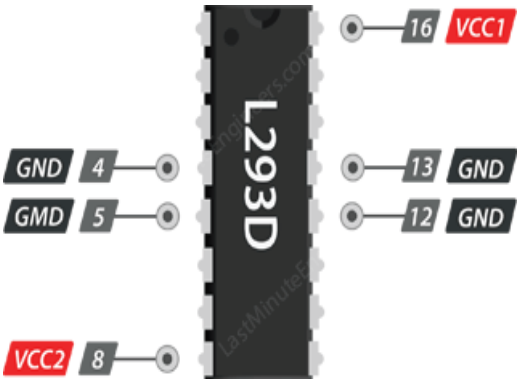
Arduino	-	DC Motor	-	L293D	-	TMP36	
PIR	-	Battery(9V)	-	Bulb	-	Resistance	
Breadboard							



Connecting and coding:

1)H-bridge Motor Driver(L293D)

- **Power Supply**



The L293D motor driver actually has two power input pins viz. 'Vcc1' and 'Vcc2'.

Vcc1 is used for driving the internal logic circuitry which should be 5V.
From Vcc2 pin the H-Bridge gets its power for driving the motors which can be 4.5V to 36V.
And they both sink to a common ground named GND

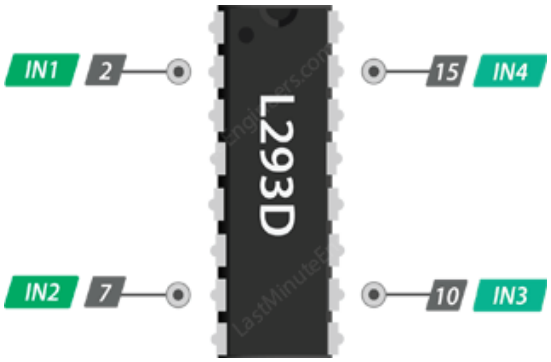
- **Output Terminals**

The L293D motor driver's output channels for the motor A and B are brought out to pins OUT1,OUT2 and OUT3,OUT4 respectively

- **Control Pins**

For each of the L293D's channels, there are two types of control pins which allow us to control speed and spinning direction of the DC motors at the same time viz. Direction control pins & Speed control pins

a. Direction Control Pins



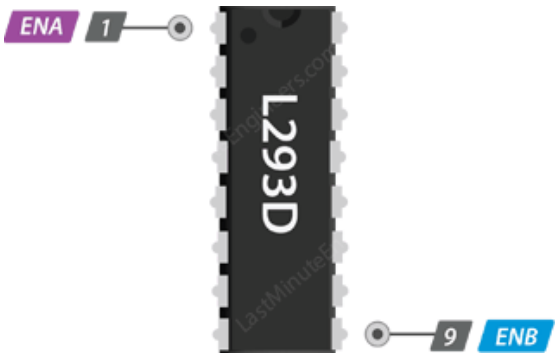
Using the direction control pins, we can control whether the motor spins forward or backward. These pins actually control the switches of the H-Bridge circuit inside L293D .

It has two direction control pins for each channel. The IN1,IN2 pins control the spinning direction of the motor A while IN3,IN4 control motor B.

The spinning direction of a motor can be controlled by applying either a logic HIGH (5 Volts) or logic LOW(Ground) to these pins. The below chart illustrates how this is done.

IN1	IN2	Spinning Direction
Low(0)	Low(0)	Motor OFF
High(1)	Low(0)	Forward
Low(0)	High(1)	Backward
High(1)	High(1)	Motor OFF

b. Speed Control Pins



The speed control pins viz. ENA and ENB are used to turn ON, OFF and control speed of motor A and motor B respectively.

Pulling these pins HIGH will make the motors spin, pulling it LOW will make them stop. But, with Pulse Width Modulation (PWM), we can actually control the speed of the motors.

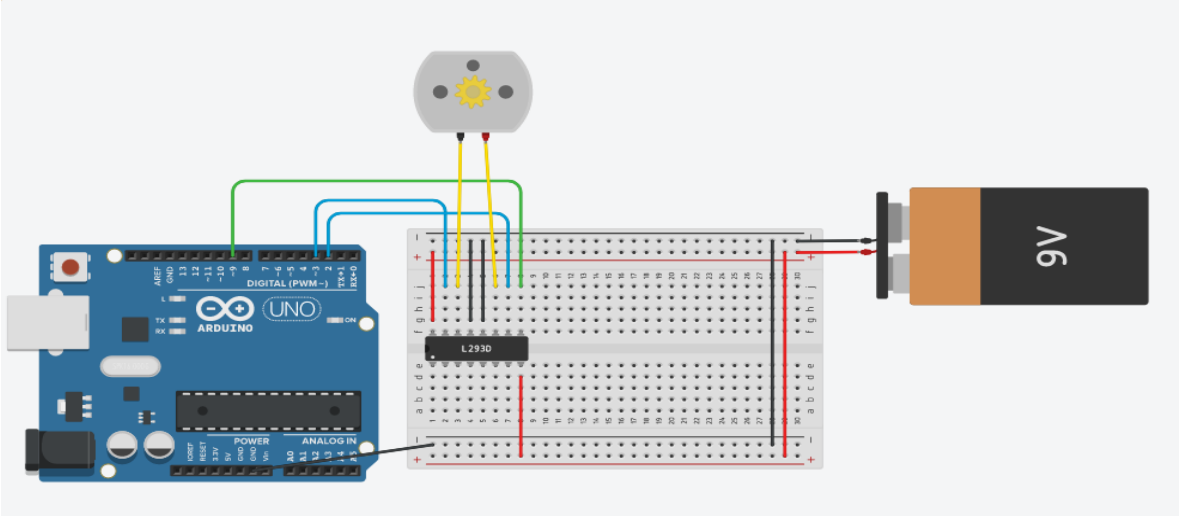
In this project use L293D to control speed for one DC motor

Start by connecting power supply to the motor. In our experiment we are using DC Motor, we will connect external 9V power supply to the Vcc1 and Vcc2 pins and make sure you common all the grounds in the circuit.

Now, the input and enable pins (ENA, IN1 and IN2) of the L293D are connected to three Arduino digital output pins (9, 3 and 2). Note that the Arduino output pin 9 is PWM-enabled.

Finally, connect one motor to across OUT3 & OUT4.

When you're done you should have something that looks similar to the illustration shown below.



Arduino Code – Controlling a DC Motor

The following sketch will give you complete understanding on how to control speed of a DC motor with L293D motor driver

```
// Motor A connections
int ENA = 9;
int IN1 = 2;
int IN2 = 3;

void setup()
{
  // Set the motor control pins to outputs
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);


  // Turn off motors - Initial state
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

void loop()
{
  speedControl();
  delay(1000);

}
```

```
// This function for controling speed of the motor
void speedControl() {

  // Turn on the motor
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

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


  // Now turn off motors
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

}
```

2)TMP36 Temperature Sensor

The TMP36 is a low voltage, precision centigrade temperature sensor manufactured by Analog Devices. It is a chip that provides a voltage output that is linearly proportional to the temperature in °C and is, therefore, very easy to use with an Arduino.

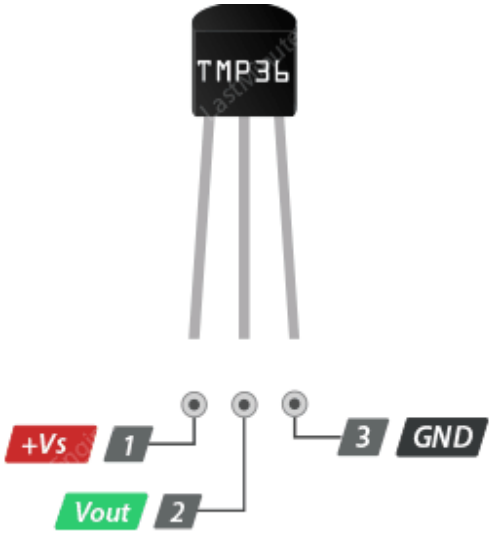
The TMP36 temperature sensor is fairly precise, never wears out, works under many environmental conditions and requires no external components to work. In addition, the TMP36 sensor does not require calibration and provides a typical accuracy of ±1°C at +25°C and ±2°C over the −40°C to +125°C temperature range.

The sensor can be powered with a 2.7V to 5.5V power supply and consumes only 50µA during active temperature conversions, providing very low self-heating (less than 0.1°C in still air). In addition, a shutdown function is provided to reduce the supply current to less than 0.5µA.

To convert the voltage to temperature, simply use the basic formula:

$$\text{Temp (°C)} = (\text{Vout} - 0.5) * 100$$

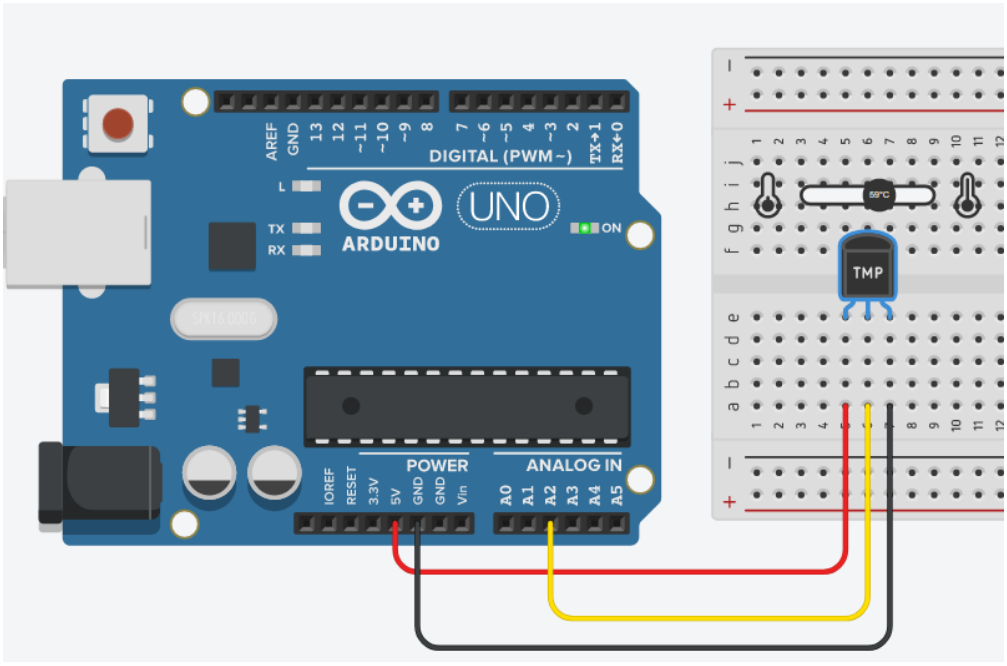
+Vs is the power supply for the sensor which can be anywhere between 2.7V to 5.5V.
Vout pin produces an analog voltage that is directly proportional (linear) to the temperature. It should be connected to an Analog (ADC) input.
GND is a ground

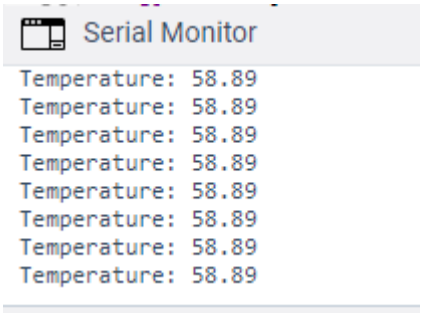


Connecting the TMP36 Temperature to an Arduino

Hooking up the TMP36 to an Arduino is super simple. You only need to connect three pins: two for power and one for reading the sensor value.

The sensor can be powered from 3.3 or 5V output. The positive voltage connects to '+Vs' and ground connects to 'GND'. The middle pin 'Vout' is the analog signal output from the sensor and connects to the A0 analog input of an Arduino.





Arduino Code

The following sketch shows a quick way to read a TMP36 temperature sensor and can serve as the basis for more practical experiments and projects. It simply reads the value from the TMP36 using analog port A2 and prints the current temperature (in °C) on the serial monitor. Go ahead and upload it to your Arduino.

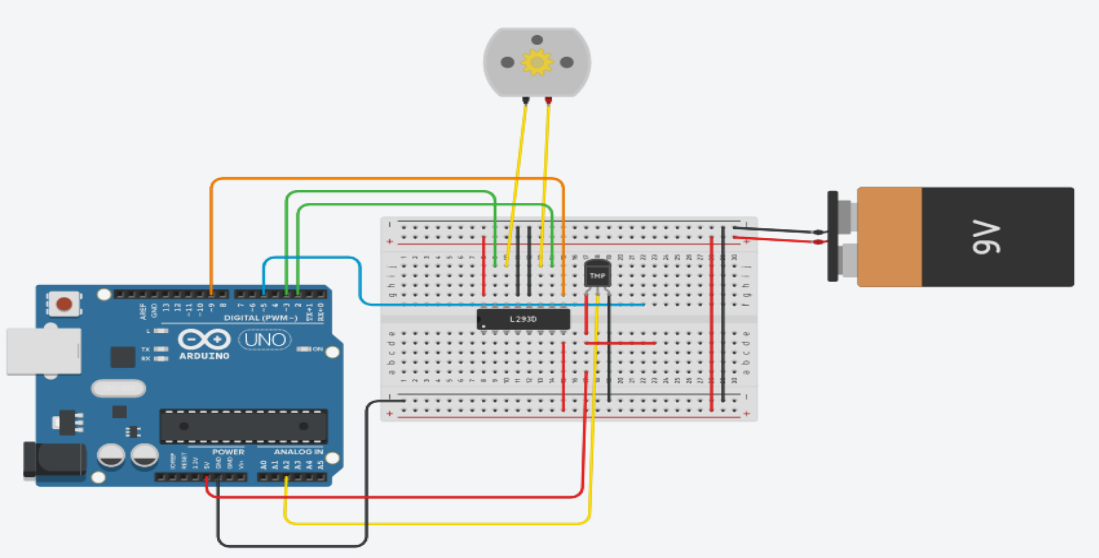
```
// Define the analog pin, the TMP36's Vout pin is connected to
#define sensorPin A2

void setup()
{
  pinMode(A2,INPUT);

  // Begin serial communication at 9600 baud rate
  Serial.begin(9600);

}

void loop()
{
    // Get the voltage reading from the TMP36
  int reading = analogRead(sensorPin);

  // Convert that reading into voltage
  // Replace 5.0 with 3.3, if you are using a 3.3V Arduino
  float voltage = reading * (5.0 / 1024.0);

  // Convert the voltage into the temperature in Celsius
  float temperatureC = (voltage - 0.5) * 100;

  // Print the temperature in Celsius
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.println("°C");


  delay(1000); // wait a second between readings
}
```

3)DC Motor with Temperature Sensor

A temperature sensor that measures the temperature and, based on it, controls the speed of the motor so that:

1. temperature is less than 20 >> The motor stops
2. Temperature equal 20 >> The motor rotates at 50%
3. The temperature is between 20 and 40 >> The motor rotates at a speed proportional to the degree
4. The temperature is greater than 40 >> The motor is belted at the highest possible speed

Connecting to an Arduino



Arduino Code

```
// Define the analog pin, the TMP36's Vout pin is connected to
#define sensorPin A2

// Motor A connections
int ENA = 9;
int IN1 = 2;
int IN2 = 3;

void setup() {

  // Set the motor control pins to outputs
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);

  // Turn off motors - Initial state
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

void loop() {

   speedControl(temp());
    delay(10);
}
```

```
// This function for controling speed of the motor based on
void speedControl(int tem) {

//maximum speed
  if(tem>40)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA,255);
  }

  //50%
  else if(tem==20)
  {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA,128);

    pinMode(A2,INPUT);//pin tem
  }

  //zero
  else if(tem<20)
  {
      digitalWrite(IN1, LOW);
      digitalWrite(IN2, LOW);
  }

  else
  {
    double val = temp();
    val = map(val,21 , 40, 128, 254);

    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    analogWrite(ENA,val);

  }
```

```
int temp(void)
{

  // Get the voltage reading from the TMP36
  int reading = analogRead(sensorPin);

  // Convert that reading into voltage
  // Replace 5.0 with 3.3, if you are using a 3.3V Arduino
  float voltage = reading * (5.0 / 1024.0);

  // Convert the voltage into the temperature in Celsius
  float temperatureC = (voltage - 0.5) * 100;

  // Print the temperature in Celsius

  return temperatureC;

}
```

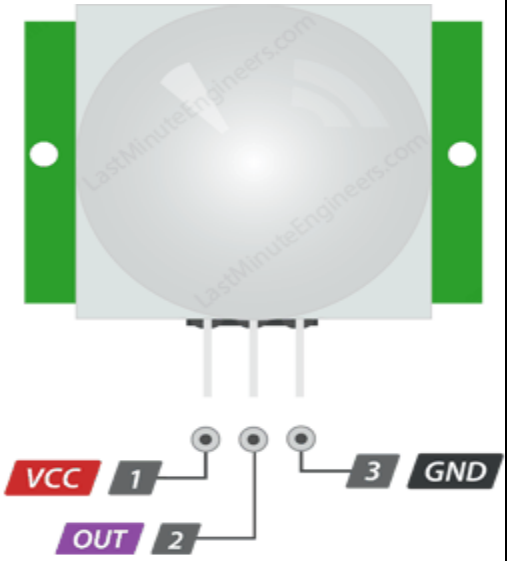
4)PIR Motion Sensor

PIR sensors allow you to sense motion, almost always used to detect whether a human has moved in or out of the sensors range. They are small, inexpensive, low-power, easy to use and don't wear out. For that reason they are commonly found in appliances and gadgets used in homes or businesses. They are often referred to as PIR, "Passive Infrared", "Pyroelectric", or "IR motion" sensors.

HC-SR501 PIR Sensor Pinout

The HC-SR501 has a 3-pin connector that interfaces it to the outside world. The connections are as follows:

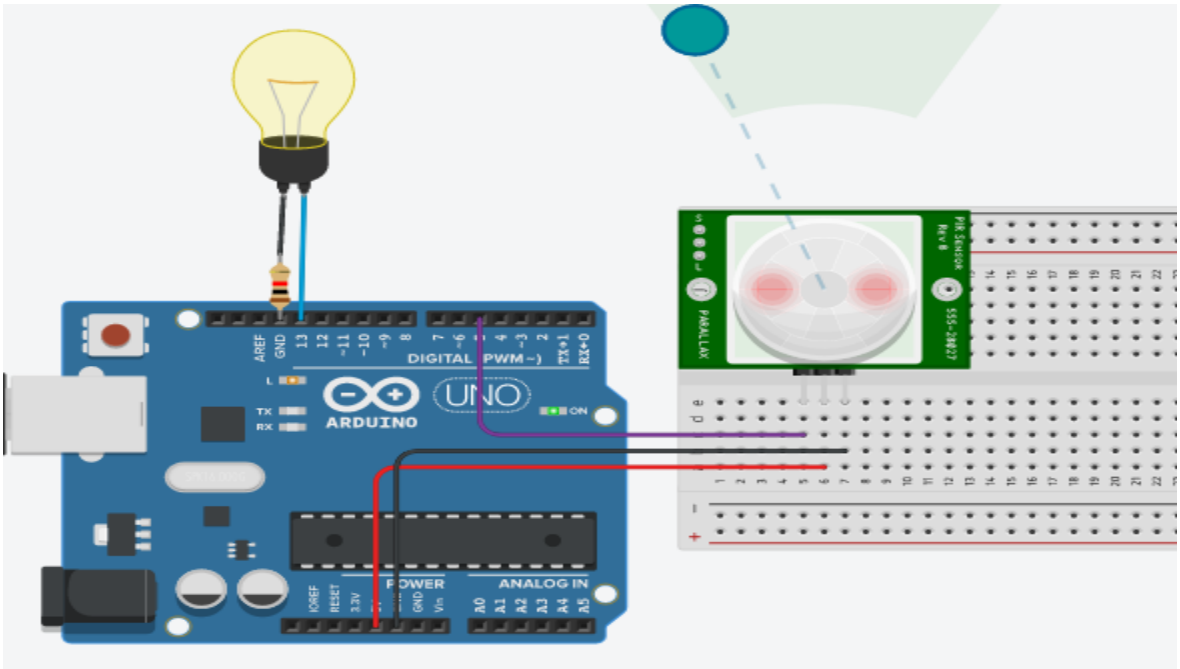
VCC is the power supply for HC-SR501 PIR sensor which we connect the 5V pin on the Arduino.
Output pin is a 3.3V TTL logic output. LOW indicates no motion is detected, HIGH means some motion has been detected.
GND should be connected to the ground of Arduino.



5)PIR Motion Sensor with Lamp

when there is movement, PIR sensor will turn on the bulb, even if it does not turn it off

Connecting to an Arduino



Arduino Code

```
/lumb
int lumb = 13;

//PIR
int PIR = 5;

int State = 0;

void setup() {

  //set lumb pin to output
  pinMode(lumb, OUTPUT);

  //set PIR pin to output
  pinMode(PIR, INPUT);


}

void loop() {

   State = digitalRead(PIR);

   if (State == HIGH)
     digitalWrite(lumb, HIGH);

   else if(State == LOW)
     digitalWrite(lumb, LOW);

    delay(10);

}
```

Project link in tinkercad