**NAME: REGINA D**

**PROJECT: 4**

# SMART BIN: WOKWI & THINGSPEAK INTERGRATION

**Wokwi:** https://wokwi.com/projects/422468464408486913

**Thingspeak:** https://thingspeak.mathworks.com/channels/2834140/import_export

## OBJECTIVE

- To automatically open the bin lid when motion is detected
- To Monitor the bin's fill level using an ultrasonic sensor.

## RESOURCES

1. Wokwi

2. ThingSpeak

## 1. WOKWI

Wokwi provides a virtual environment for embedded system development, making it ideal for prototyping IoT projects. It allows users to connect sensors, actuators, and displays to microcontrollers and test their functionality without any physical components. The platform supports Arduino libraries, serial monitoring, and cloud-based data visualization, making it a valuable tool for students, hobbyists, and engineers. It also supports custom components and API integrations, allowing for more advanced IoT simulations.

For the Smart Bin IoT Project, Wokwi was used to:

- Simulate the PIR sensor detecting motion to trigger the servo motor.
- Measure the bin's fill level using the ultrasonic sensor.
- Integrate with ThingSpeak to send real-time data about the bin's status.
- Test and debug the code before real-world implementation.

## 2. THINGSPEAK

ThingSpeak is an IoT analytics platform that allows devices to send, store, and visualize sensor data in real time. It is widely used for remote monitoring and data logging applications.

For the Smart Bin IoT Project, ThingSpeak is used to:

- Monitor the bin's fill level in real-time through a graphical chart.
- Display the status of four LEDs, which indicate the fill level of the bin.

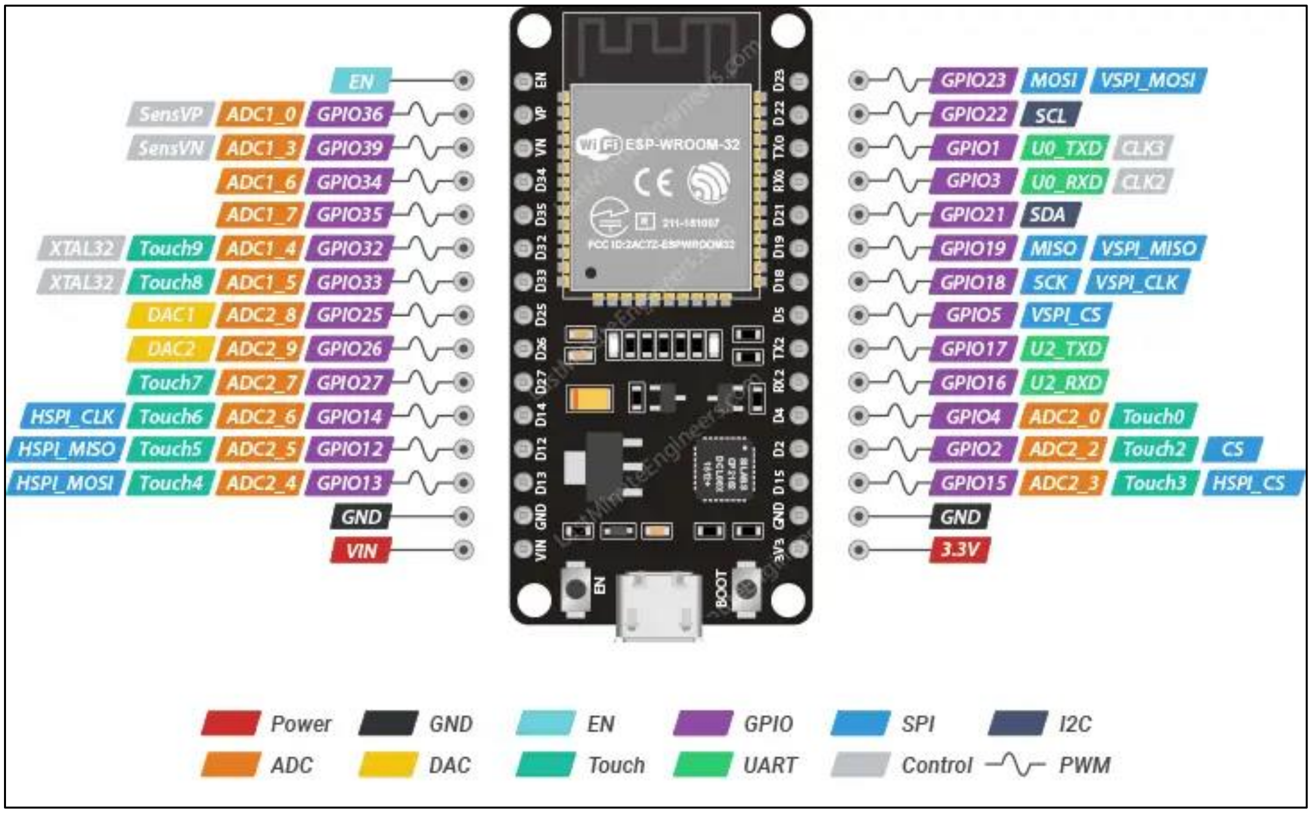
**COMPONENTS**

| S.No | Component | Quantity |
|---|---|---|
| **1.** | ESP32 | 1 |
| **2.** | Ultrasonic Sensor | 1 |
| **3.** | PIR Sensor | 1 |
| **4.** | Servo Motor | 1 |
| **5.** | LEDs | 4 |
| **6.** | Connecting Wires | As required |

## 3.1 ESP32 DEVKIT V1

The ESP32 is a low-cost, low-power system-on-chip (SoC) microcontroller developed by Espressif Systems. It is widely used for IoT and embedded applications due to its integrated Wi-Fi and Bluetooth capabilities, multiple GPIO pins, and support for a variety of peripherals. It is typically operates at 3.3V.

### 3.1.1 DIAGRAM



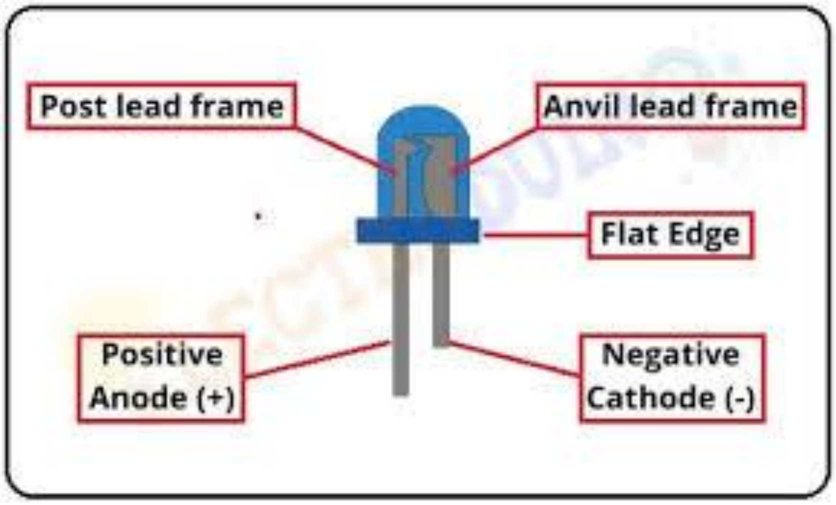
### 3.1.2 ESP32 FEATURES

| S.No | Category | Feature |
|---|---|---|
| 1. | **Processor** | Dual-core or single-core Tensilica Xtensa LX6 processor(s), up to 240 MHz, 32-bit processing |
| 2. | **Memory** | 520 KB SRAM, 4 MB Flash (varies by module) |
| 3. | **Connectivity** | Wi-Fi (802.11 b/g/n), Bluetooth (Classic + BLE) |
| 4. | **Analog Peripherals** | - 15 ADC Channels: 12-bit SAR ADC, ranges: 0-1V, 0-1.4V, 0-2V, 0-4V<br>- 2 DAC Channels: Two 8-bit DACs for analog voltages |
| 5. | **PWM Outputs** | 25 PWM pins for controlling motors, LEDs, etc. |
| 6. | **Capacitive Touch** | 9 GPIO pins with capacitive touch sensing |
| 7. | **UART Interfaces** | 2 UART interfaces with flow control and IrDA support |
| 8. | **SPI, I2C, and I2S** | - 3 SPI Interfaces: For sensors and peripherals<br>- 1 I2C Interface: For connecting I2C devices<br>- 2 I2S Interfaces: For audio |
| 9. | **GPIO Pins** | ~36 configurable GPIO pins, including ADC, DAC, PWM |
| 10. | **Pin Multiplexing** | Enables multiple peripherals to share the same pins |
| 11. | **Operating Temperature** | -40°C to +85°C (varies by module) |
| 12. | **Programming Options** | Compatible with Arduino IDE, ESP-IDF, PlatformIO, and MicroPython |
| 13. | **Security** | Built-in cryptographic hardware acceleration (AES, SHA, RSA, ECC), Secure Boot, Flash Encryption |
| 14. | **Project Applications** | Ideal for IoT, smart devices, home automation, robotics, and low-power systems |

## 3.2 LED

Light emitting diode, abbreviated as LED, is a semiconductor device that emits infrared or visible light when charged with an electric current. The figure below shows a circuit diagram for the LED indicator circuit.

### 3.2.1 DIAGRAM

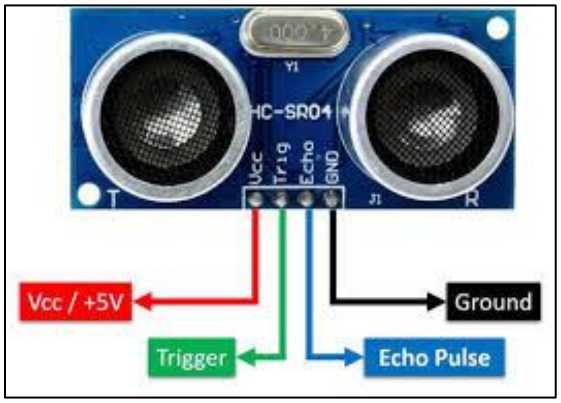


### 3.2.2 FEATURES

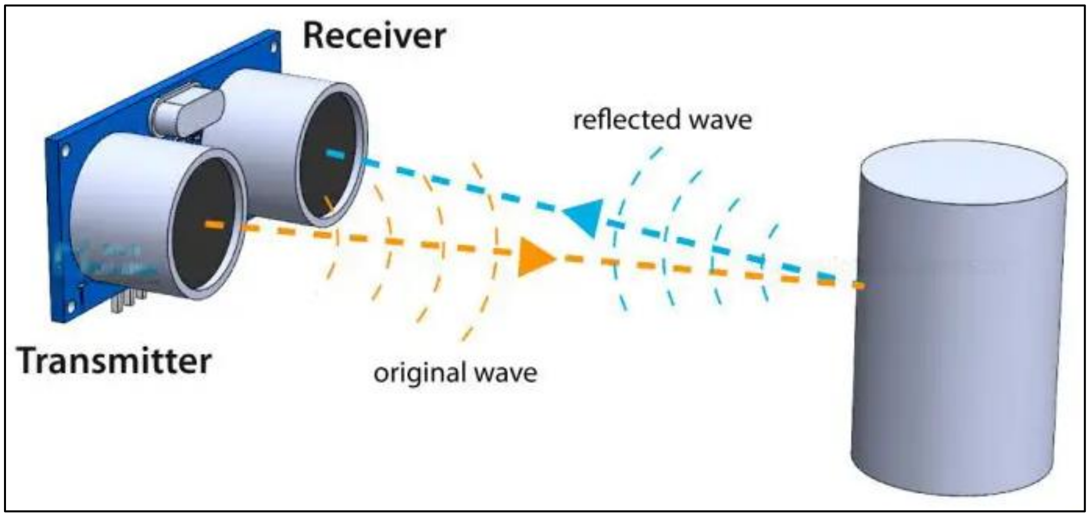
| | | |
|---|---|---|
| **1.** | **Operating Voltage** | Typical forward voltage ranges:<br>- Red/Yellow: ~1.8V-2.2V<br>- Green/Blue/White: ~3.0V-3.6V |
| **2.** | **Polarity** | LEDs are polarized devices with two terminals:<br>- Anode (+): Longer leg (positive)<br>- Cathode (-): Shorter leg (negative) |

### ULTRASONIC SENSOR

An **Ultrasonic Sensor** is a device that measures **distance** by emitting **high-frequency sound waves** and calculating the time it takes for the echo to return. It operates on the **principle of sound wave reflection**, making it useful for object detection, level sensing, and distance measurement in automation and robotics.



**WORKING**



1. The sensor sends out a **sound wave** (ultrasonic pulse).
2. The sound wave **travels through the air** until it **hits an object**.
3. The wave **bounces back** and returns to the sensor.
4. It sends a sound wave and **measures the total time** for it to return.
5. Since the sound **travels to the object and back**, we divide by **2**.
6. We know that the **speed of sound in air** is approximately **340 m/s**.
7. Distance is calculated using

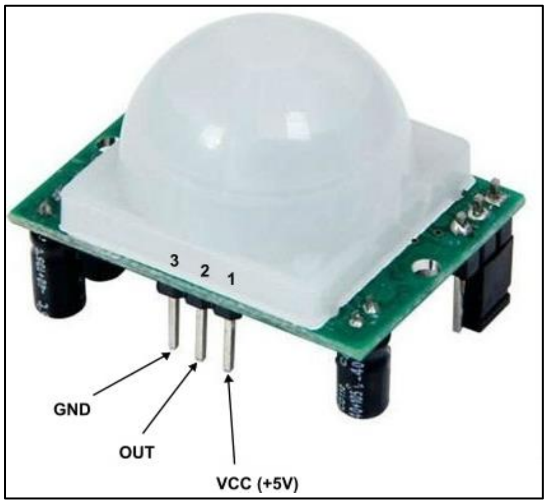
   **Distance = (speed x time)/ 2**

**FEATURES**

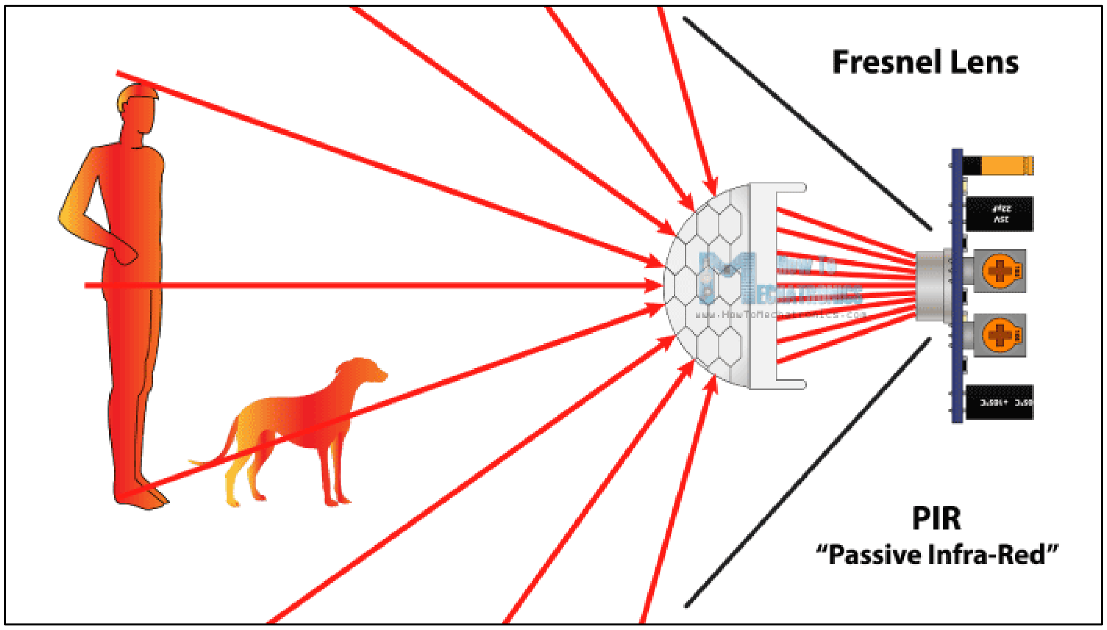
1. Operating voltage: +5V
2. Theoretical  Measuring Distance: 2cm to 450cm
3. Practical Measuring Distance: 2cm to 80cm
4. Accuracy: 3mm
5. Measuring angle covered: <15°
6. Operating Current: <15mA
7. Operating Frequency: 40Hz

## PASSIVE INFRARED (PIR) MOTION SENSOR

A **Passive Infrared (PIR) sensor** is an electronic sensor that detects **infrared (IR) radiation** emitted by objects, particularly living beings like humans and animals. It works by sensing changes in infrared energy levels within its field of view, triggering an output signal when movement is detected.



## WORKING PRINCIPLE



1. **PIR sensors** detect movement from **humans** and **animals** within a specific range.
2. They consist of a **pyroelectric sensor** that detects varying levels of **infrared (IR) radiation**.
3. The sensor does not **emit any energy** but passively **receives infrared radiation** from the environment.

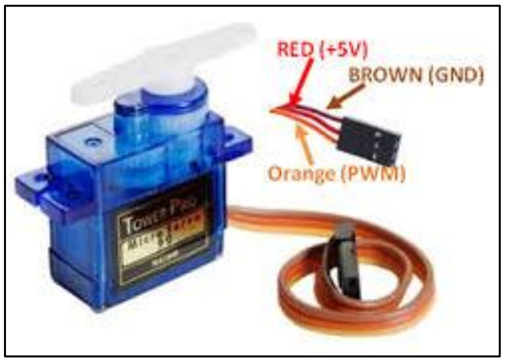
4. When a **human** or **animal** with **body temperature** enters the detection range, it emits **infrared radiation**, which is **focused** by the **optical system** onto the **pyroelectric sensor**, generating a **sudden electrical signal**.
5. When an **object enters** the detection area, it **intercepts** the **first slot** of the **PIR sensor**, causing a **positive differential change** between the two **bisects**.
6. When the **object leaves** the sensing area, the sensor generates a **negative differential change** between the two **bisects**.

## FEATURES

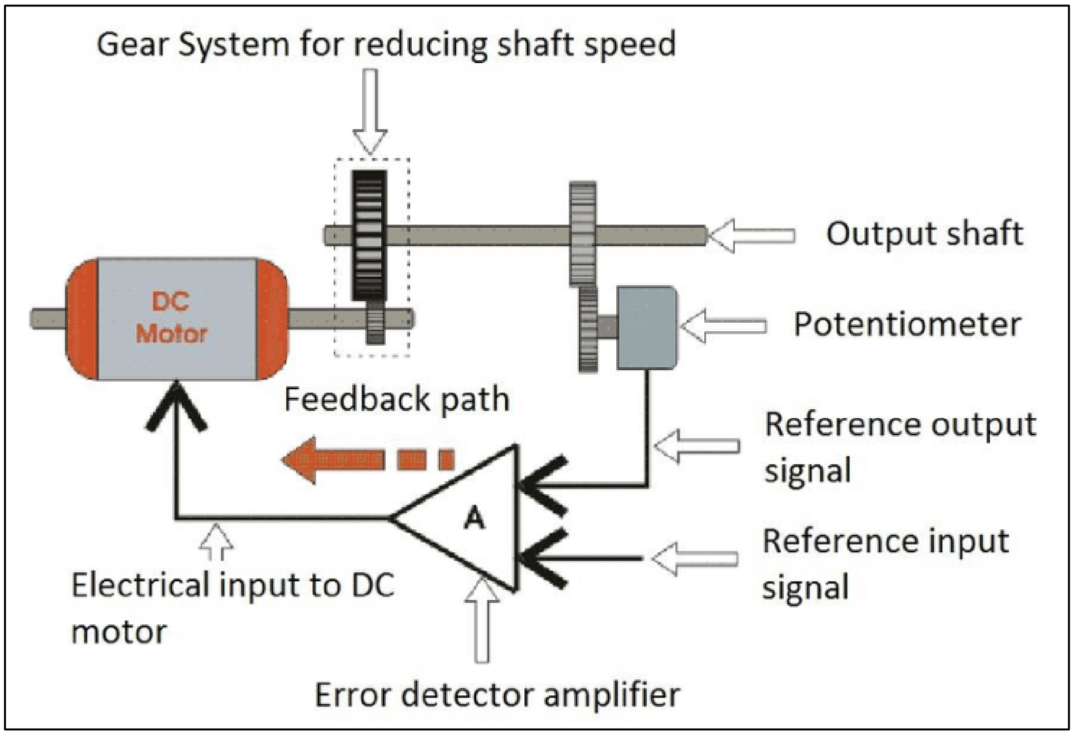
1. **Operating Voltage**: **3V to 5V** (some models support up to **12V**).
2. **Current Consumption**: Low power consumption, usually **< 1mA to 2mA**.
3. **Detection Range**: Commonly **3 to 7 meters** (adjustable in some models).
4. **Detection Angle**: Usually **100° to 120°** (depends on lens design).
5. **Output Signal**: Digital **High (1) or Low (0)** based on motion detection.
6. **Trigger Modes**:
    a. **Non-repeatable (single trigger)** – Output stays high for a fixed time.
    b. **Repeatable (multiple triggers)** – Resets timer when motion continues.
7. **Warm-up Time**: Takes **30 to 60 seconds** after power-up for stabilization.
8. **Sensitivity Adjustment**: Some sensors have a **potentiometer** for range tuning.
9. **Lens Type**: Uses a **Fresnel lens** to focus infrared radiation onto the sensor.
10. **Operating Temperature**: **-20°C to 50°C**.

## SERVO MOTOR

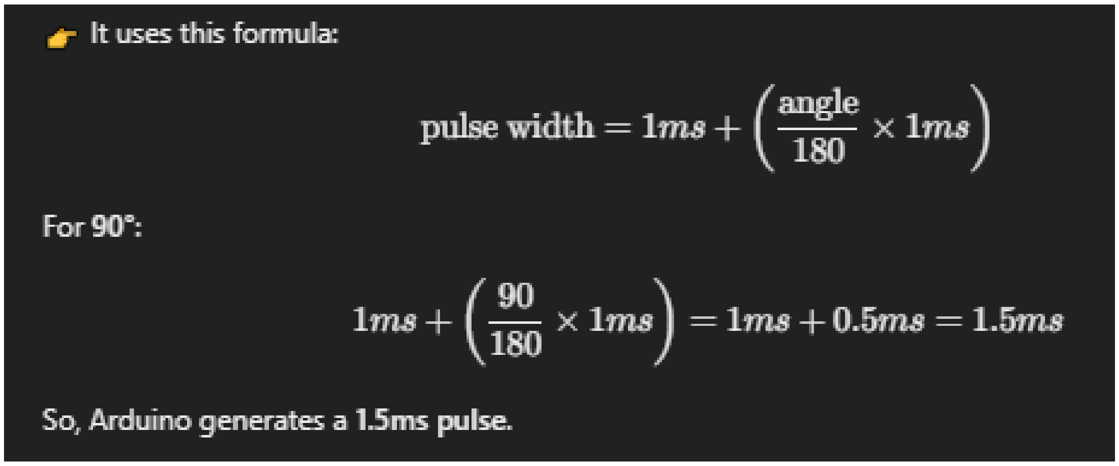
A **positional servo motor** is a type of servo motor that controls angular position with high accuracy. It consists of a **DC motor, position sensor, gear assembly, and a control circuit**. The motor maintains its position based on the input control signal, which is typically a **PWM (Pulse Width Modulation) signal**.



**WORKING**



a. A **positional servo motor** is controlled using **Pulse Width Modulation (PWM)**.
b. The **width of the pulse** determines the exact **angle** of the motor's shaft.
c. A **1.5ms pulse** positions the servo at **90° (middle position)**.
d. A **shorter pulse (<1.5ms)** moves it **counterclockwise toward 0°**.
e. A **longer pulse (>1.5ms)** moves it **clockwise toward 180°**.
f. Converting angle to PWM



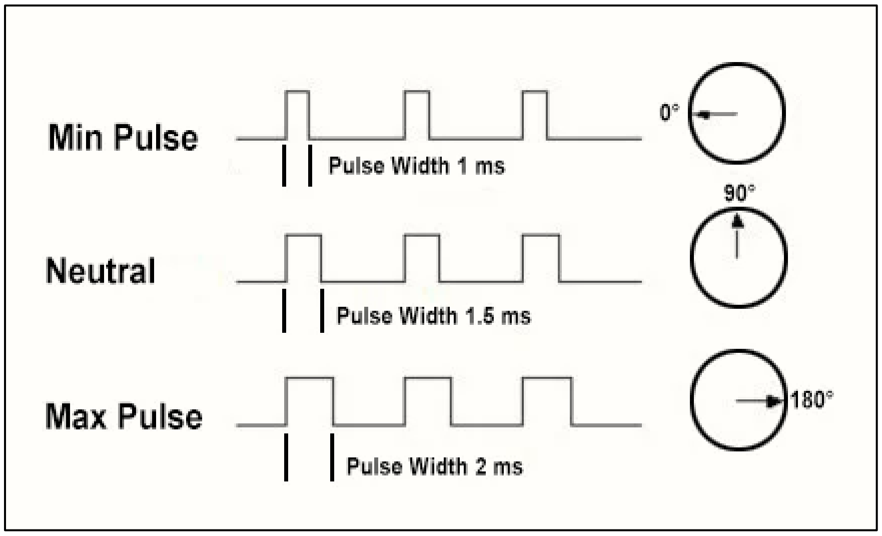
👉 It uses this formula:

$$\text{pulse width} = 1ms + \left(\frac{\text{angle}}{180} \times 1ms\right)$$

For 90°:

$$1ms + \left(\frac{90}{180} \times 1ms\right) = 1ms + 0.5ms = 1.5ms$$

So, Arduino generates a 1.5ms pulse.



g. Inside the servo, there is a **position sensor (potentiometer)** that continuously **monitors** the actual position of the motor.
h. The **control circuit** compares the **actual position** with the **desired position**.
i. If there is a difference (error), the control circuit **sends power to the motor** to move in the correct direction.
j. The motor **stops moving** once the error becomes **zero** (i.e., the desired position is reached).

**FEATURES**

1. **Operating Voltage**: 4.8V - 6V
2. **Rotation Angle**: 0° to 180°
3. **Control Signal**: PWM (1ms → 0°, 1.5ms → 90°, 2ms → 180°)
4. **Torque Rating**: 3-25 kg.cm (varies by model)
5. **Speed**: ~0.12s/60° (depends on model)
6. **Gear Type**: Plastic (standard) / Metal (high torque)

**PROGRAM**

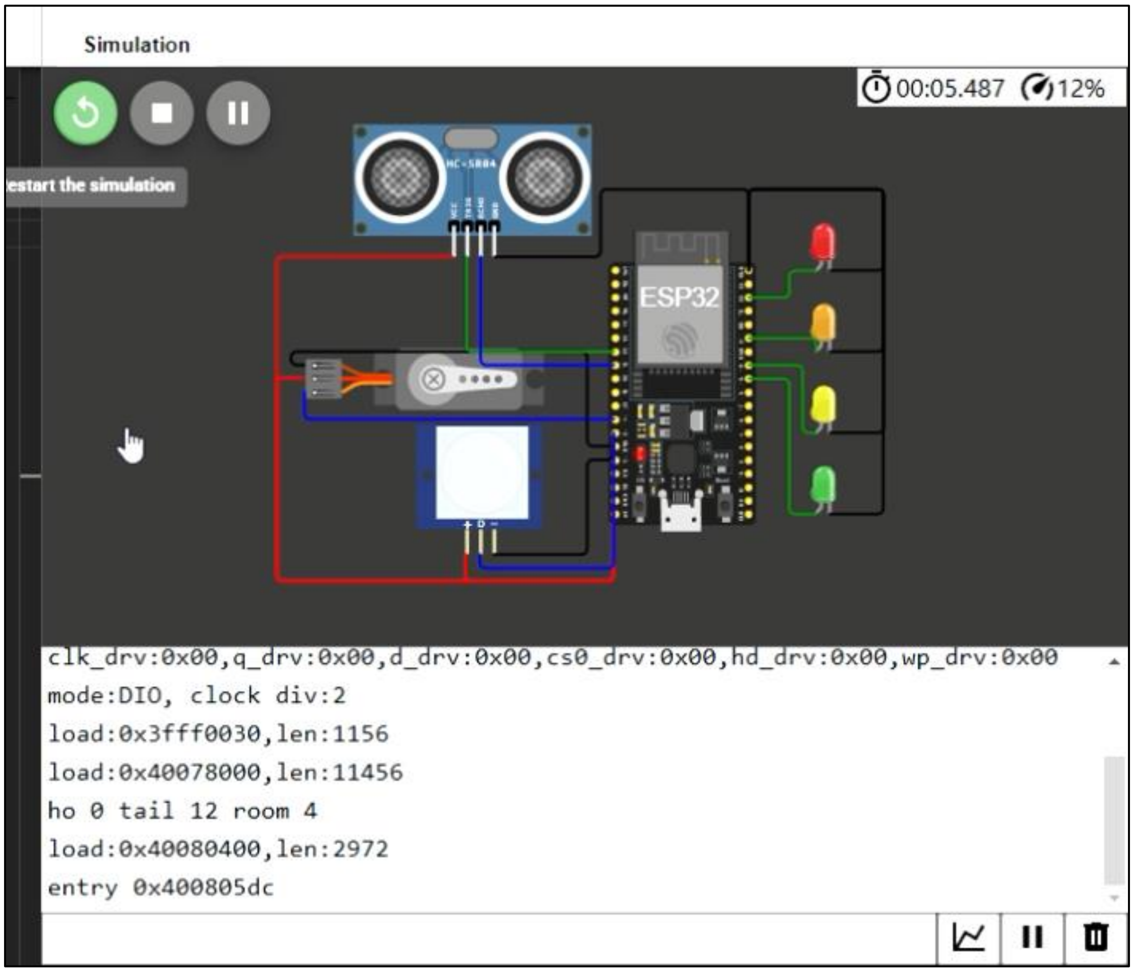
```
#include <WiFi.h>
#include <ESP32Servo.h>
#include <ThingSpeak.h>

const char* SSID= "Wokwi-GUEST";
const char* Password= "";

WiFiClient client; //Creates a WiFiClient object
const int channel_number= 2834140;
const char* api_key= "MDPRLBCN9BQWJSRR"; //write api key
const char* server= "api.thingspeak.com";
```

```
#define trig_pin 32
#define echo_pin 33
#define pir_pin 12
#define servo_pin 14
#define led_green 18
#define led_yellow 19
#define led_orange 21
#define led_red 22

Servo lid_control;

void setup(){
  Serial.begin(115200);

  WiFi.begin(SSID, Password); //for internet access
  WiFi.mode(WIFI_STA); //to connect esp32 to existin wifi
  ThingSpeak.begin(client); // ThingSpeak communication

  pinMode (trig_pin, OUTPUT);
  pinMode (echo_pin, INPUT);
  pinMode (pir_pin, INPUT);
  pinMode (servo_pin, OUTPUT);
  pinMode (led_green, OUTPUT);
  pinMode (led_yellow, OUTPUT);
  pinMode (led_orange, OUTPUT);
  pinMode (led_red, OUTPUT);

  pinMode(pir_pin, INPUT_PULLDOWN);

  lid_control.attach(servo_pin);
}

int bin_level_(){
  digitalWrite(led_green, LOW);
  digitalWrite(led_yellow, LOW);
  digitalWrite(led_orange, LOW);
  digitalWrite(led_red, LOW);

  digitalWrite(trig_pin, LOW);
  delay(2);
  digitalWrite(trig_pin, HIGH);
  delay(10);
  digitalWrite(trig_pin, LOW);
  int duration= pulseIn(echo_pin, HIGH);
  int distance= duration* 0.034/2;
  int bin_height= 100; //in cm
  int bin_level= 100- distance;

  if(bin_level>=30 && bin_level<50){
    digitalWrite(led_green, HIGH);  //if level>=30 and level<50
  }
  else if(bin_level>=50 && bin_level<85){
    digitalWrite(led_green, HIGH);  //if level>=50 and level<85
    digitalWrite(led_yellow, HIGH);
  }
  else if(bin_level>=85 && bin_level<90){
    digitalWrite(led_green, HIGH);  //if level>85 and level<90
    digitalWrite(led_yellow, HIGH);
    digitalWrite(led_orange, HIGH);
  }
  else if(bin_level>90){
    digitalWrite(led_green, HIGH);  //if level>90
    digitalWrite(led_yellow, HIGH);
    digitalWrite(led_orange, HIGH);
    digitalWrite(led_red, HIGH);
  }
  return bin_level;
}

void lid(){
  int motion_detected= digitalRead(pir_pin);
  Serial.println(motion_detected);
  if(motion_detected==HIGH){
    Serial.println("Lid Opened");
    lid_control.write(90);
  }
  else{
  Serial.println("Lid Closed");
    lid_control.write(0);
  }delay(500);
}

void to_thingspeak(int bin_level){
  ThingSpeak.setField(1, bin_level);
  int status = ThingSpeak.writeFields(channel_number, api_key);

if(status == 200){ //200 is an HTTP status code that means "OK"
    Serial.println("Data sent to ThingSpeak successfully.");
}
```

```
else{
    Serial.println("Failed to send data.");
}

}

void loop(){
  lid();
  int bin_level = bin_level_();

  to_thingspeak(bin_level);
  Serial.println("BIN LEVEL: ");
  Serial.println(bin_level);
  delay(1000);
}
```

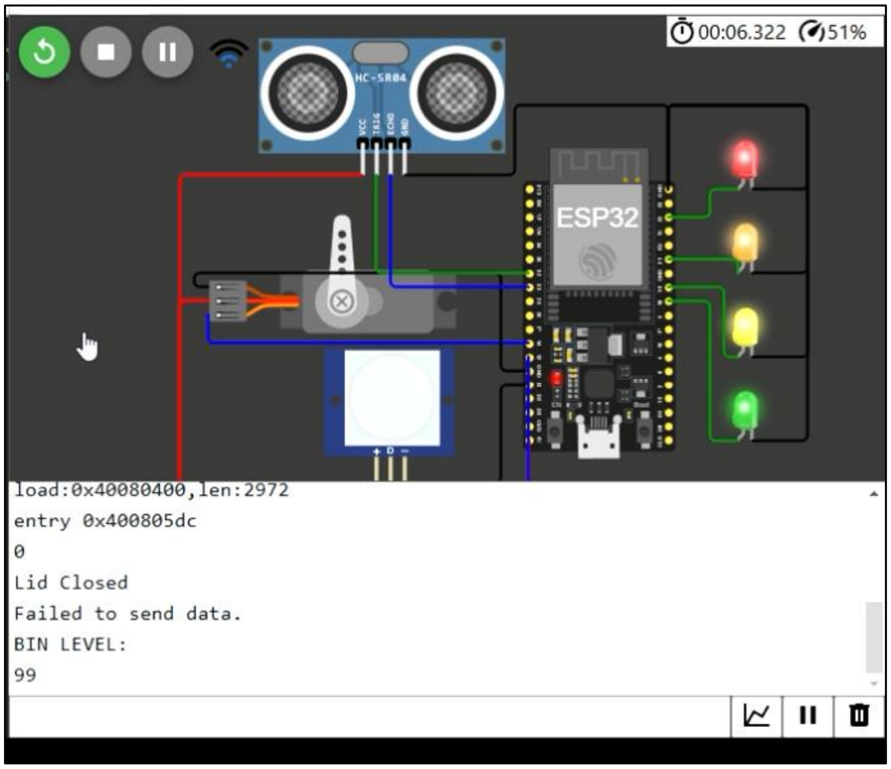
**OUTPUT**

**1. Before Simulation (Servo resting at 90 degree)**



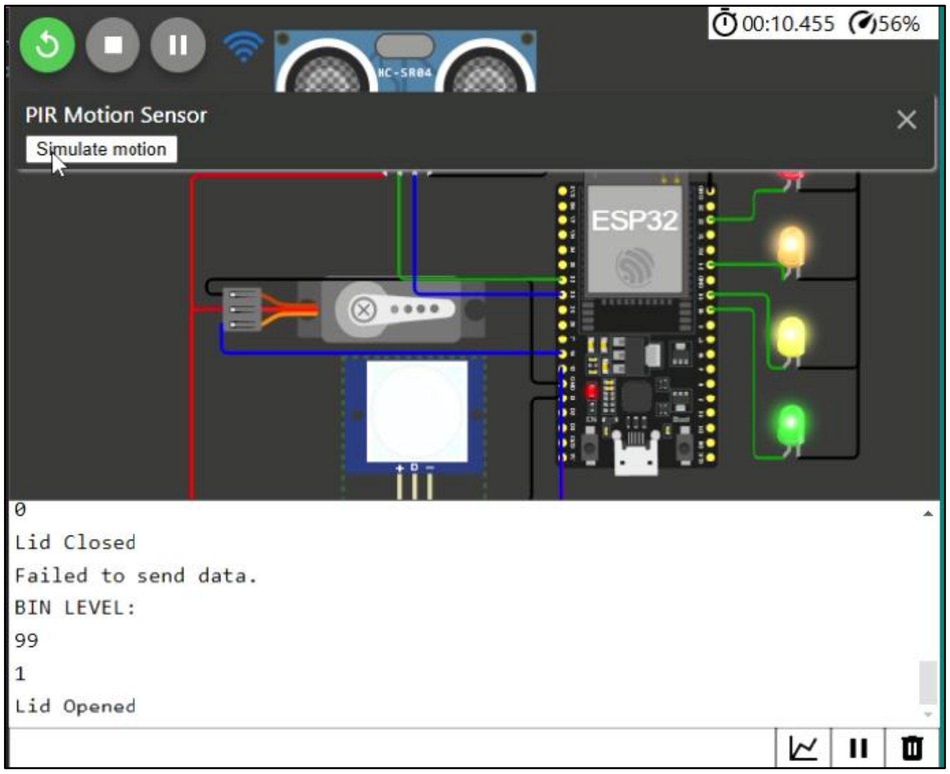
```
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
```

**2. Start Simulation. Servo motor goes to starting position (0 degree) and esp32 connects to wifi.**



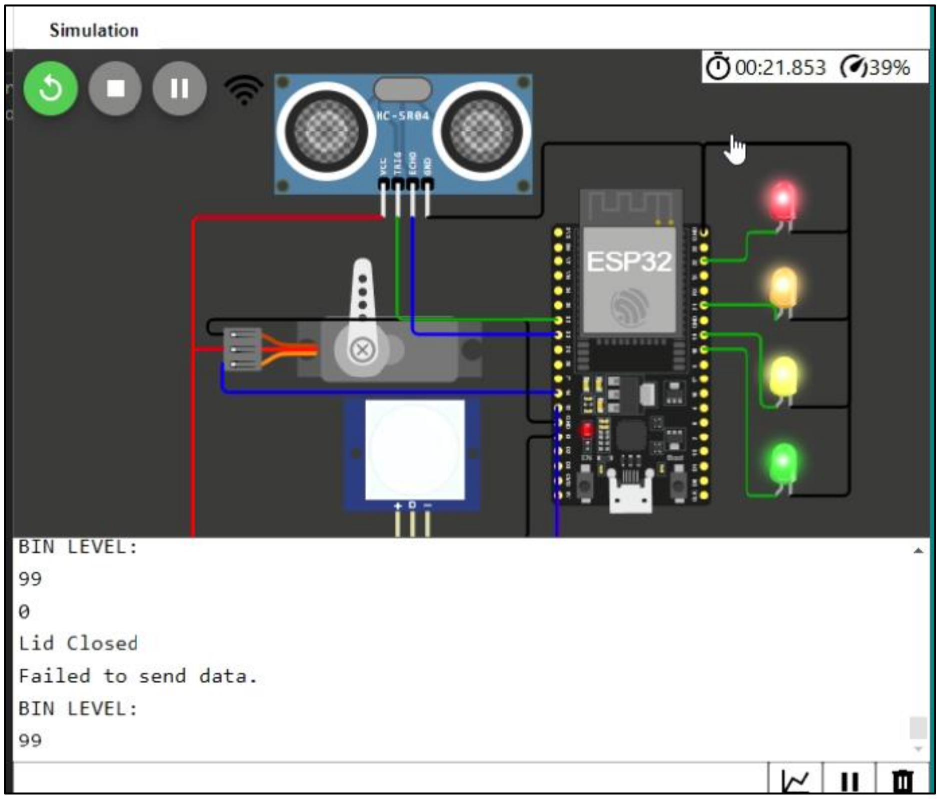
```
load:0x40080400,len:2972
entry 0x400805dc
0
Lid Closed
Failed to send data.
BIN LEVEL:
99
```

**3. Click motion simulation by clicking pir sensor. Now the sensor reads 1 and the servo motor rotates 90 degree from 0 degree.**



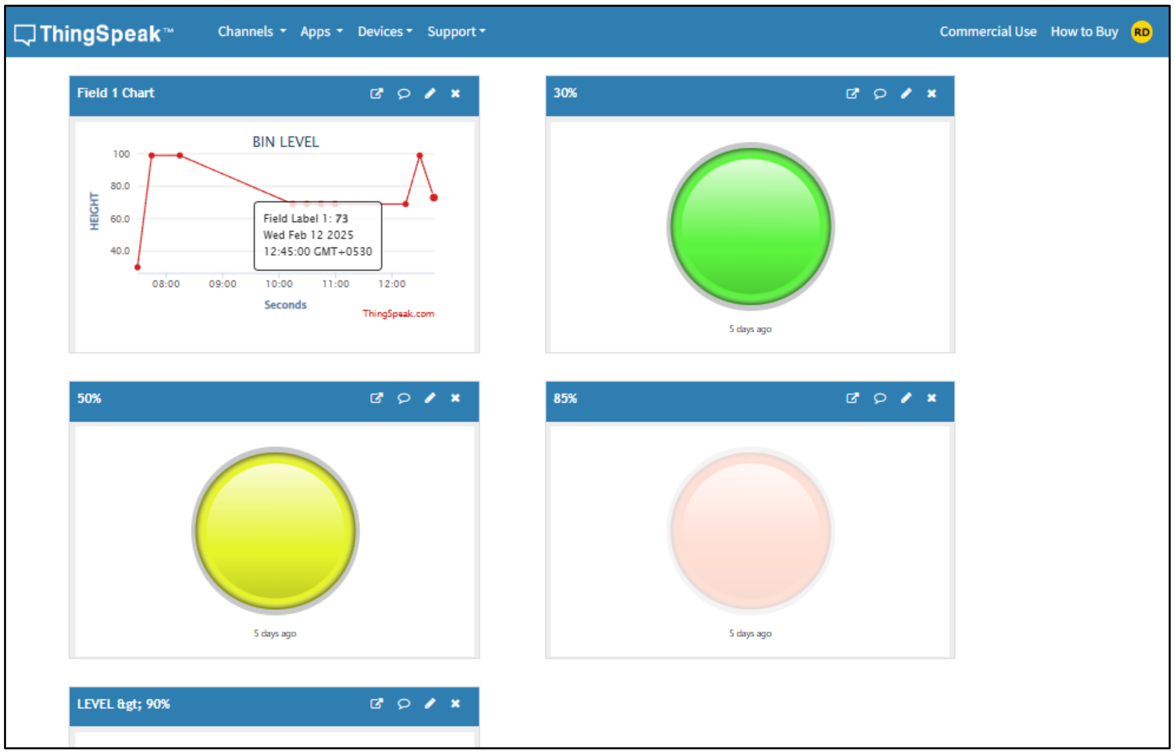
```
0
Lid Closed
Failed to send data.
BIN LEVEL:
99
1
Lid Opened
```

**4. By varying the distance by clincking ultrasonic sensor, the bin level changes.**



```
BIN LEVEL:
99
0
Lid Closed
Failed to send data.
BIN LEVEL:
99
```

**5. The real time sensor data can be observed in thingspeak. Here the green and yellow led's are ON, since bin level is 73 (50<73<85).**



## FUTURE SCOPE

1. RECYCLABLE/ NON RECYCLABLE
2. ODOR MANAGEMENT
3. SOLAR PANEL
4. ROUTE OF ALL THE TRASHCANS CAN BE TRACED USING GOOGLE MAPS