

PIR MOTION SENSOR HC-SR501

Introduction:

The **HC-SR501 PIR Motion Sensor** is widely used for motion detection in security systems, home automation, and IoT applications. It detects infrared radiation changes caused by movement and outputs a signal accordingly. This document details its working principle, standalone testing with an oscilloscope, and integration with **ESP8266MOD (NodeMCU)**.



HC-SR501 PIR Motion Sensor Overview:

Specifications:

- **Operating Voltage:** 4.5V - 20V

- **Detection Angle:** Up to 120°
- **Detection Range:** 3m - 7m (Adjustable)
- **Delay Time:** 5s - 300s (Adjustable)
- **Trigger Modes:** Retriggerable and Non-Retriggerable

Working Principle:

The PIR sensor consists of two infrared-sensitive elements. When a moving object (human, animal) enters its detection range, the difference in infrared levels triggers a **HIGH** output signal. When no motion is detected, the output remains **LOW**.

Reference:

<https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf>

PIR Sensor Testing with MSO/DSO (Without ESP8266MOD):

Before integrating the PIR sensor with ESP8266, it was tested using an **MSO (Mixed Signal Oscilloscope)** and **DSO (Digital Storage Oscilloscope)** to analyze its output behavior.

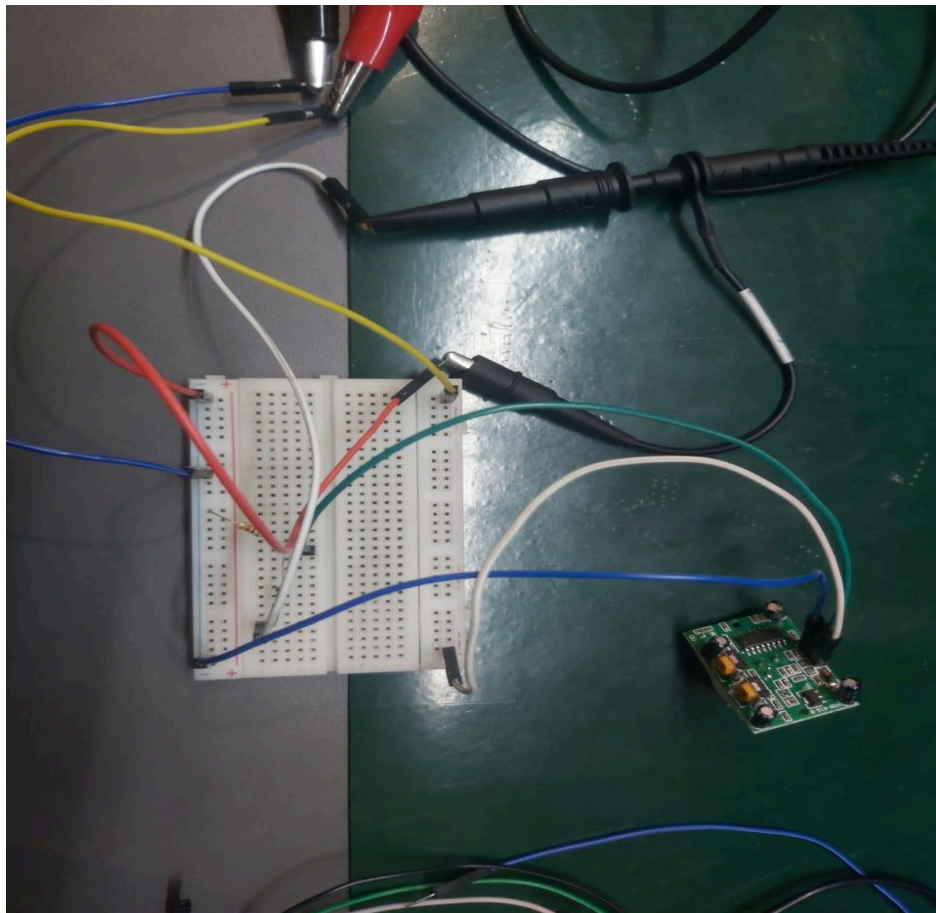
Required Components:

- HC-SR501 PIR Motion Sensor
- Oscilloscope (MSO/DSO)
- Power Supply (5V)
- Jumper Wires
- 1kΩ resistor (added for signal conditioning)

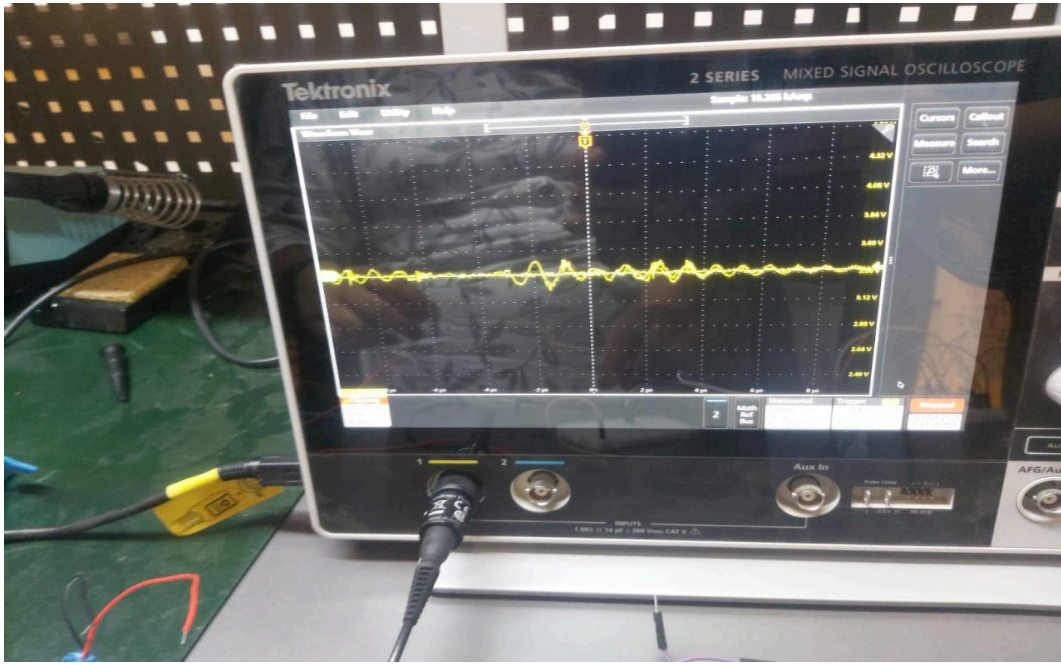
- 10k Ω pull-down resistor (added after issue occurred)

Connections:

- **VCC** \rightarrow 5V power supply
- **GND** \rightarrow Ground
- **OUT** \rightarrow Oscilloscope probe
- **1k Ω resistor** was connected between **OUT** and **GND** initially for signal stabilization.
- **10k Ω pull-down resistor** was added later when an issue occurred, ensuring the output returned to **LOW** correctly.



MSO/DSO Output Analysis & Troubleshooting:



Issues & Solutions:

No Output Signal on MSO/DSO Initially:

- **Fix:** Added a **10k Ω pull-down resistor** between OUT and GND.
- **Reason:** The PIR sensor's output was floating, and the resistor ensured a proper LOW state when no motion was detected.

Arduino IDE Installation & Board Setup:

To program the **ESP8266MOD (NodeMCU)**, we use the **Arduino IDE** with additional board support.

Steps to Install Arduino IDE:

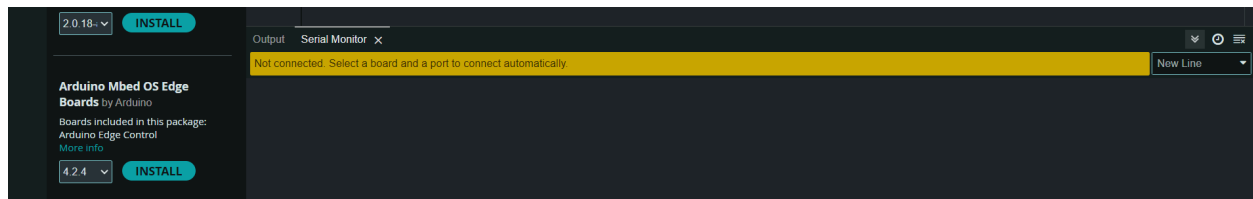
1. Download and install the **Arduino IDE** from the official website:
<https://www.arduino.cc/en/software>.
2. Open Arduino IDE and go to **File** → **Preferences**.
3. In **Additional Board Manager URLs**, add:
http://arduino.esp8266.com/stable/package_esp8266com_index.json
4. Go to **Tools** → **Board** → **Boards Manager** and search for **ESP8266**.
5. Click **Install** and wait for the installation to complete.

Board & Port Selection:

1. Connect **NodeMCU** via USB.
2. In **Arduino IDE**, go to **Tools** → **Board** → **ESP8266** → **NodeMCU 1.0 (ESP-12E Module)**.
3. Select the correct **COM Port** under **Tools** → **Port**

Common Issues & Fixes:

1.Board Not Found in Boards Manager:



- Ensure the **Board Manager URL** is correct and internet connection is stable.

2. Error: "esptool.FatalError: Failed to connect to ESP8266"

- Hold the **Flash button** on NodeMCU while uploading code or go to device manager and and ports select ports and and uninstall the already existing library and reinstall it and restart the system.

PIR Sensor Integration with ESP8266MOD:

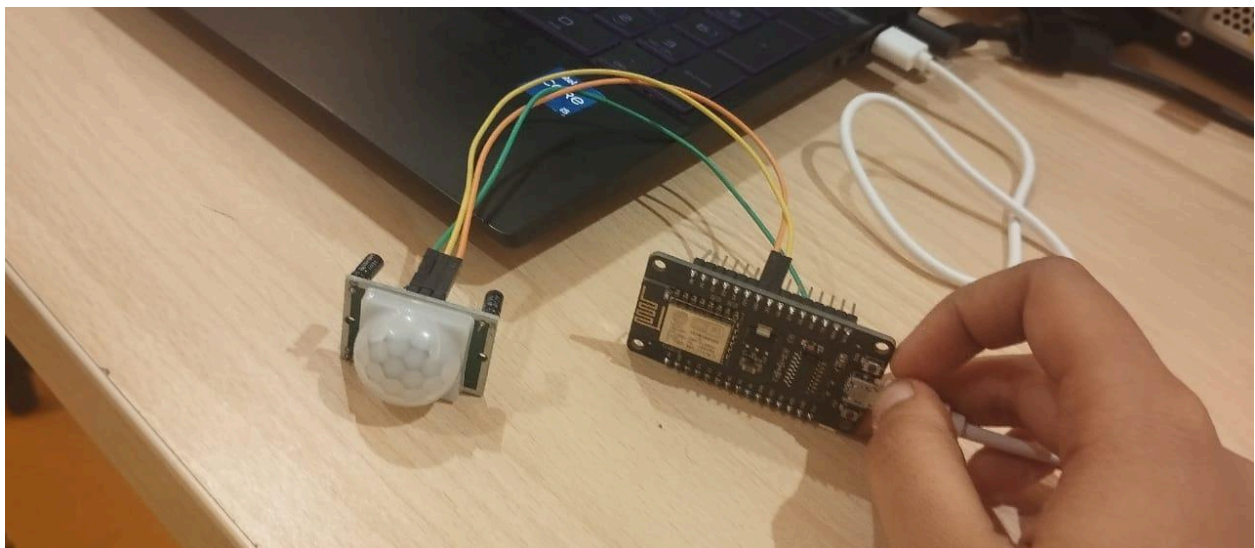
After confirming the PIR sensor's standalone functionality, it was interfaced with **ESP8266MOD (NodeMCU)** for IoT-based applications.

Required Components:

- ESP8266MOD (NodeMCU)
- HC-SR501 PIR Motion Sensor
- Jumper Wires

Connections:

- **VCC** → VIN (5V) on ESP8266
- **GND** → GND on ESP8266
- **OUT** → GPIO **D5** on ESP8266



Code Implementation:

Code:

```
#define PIR_SENSOR_PIN D1
#define LED_PIN D2
void setup() {
  Serial.begin(115200); // Start Serial Monitor
```



```

pinMode(PIR_SENSOR_PIN, INPUT);
pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  int motionState = digitalRead(PIR_SENSOR_PIN);
  if (motionState == HIGH)
  {
    Serial.println("Motion detected!");
    digitalWrite(LED_PIN, HIGH);
  }
else
  {
    Serial.println("No motion detected");
    digitalWrite(LED_PIN, LOW);
  }
  delay(1000);
}

```

The screenshot shows the Arduino IDE interface. On the left, there is a sidebar with board selection options: "Boards by Arduino" (with a "REMOVE" button), "32 Boards by Arduino" (with an "INSTALL" button), "ed OS Edge Boards by" (with an "INSTALL" button), "ed OS Giga Boards by" (with an "INSTALL" button), and "ed OS Nano Boards by". The main area is the code editor, which contains the following C++ code:

```

1 // Define the LED and PIR sensor pins
2 #define LED_PIN 13
3 #define PIR_SENSOR_PIN 2
4
5 // Connect an LED to GND (GND)
6
7 void setup() {
8   Serial.begin(115200); // Start Serial Monitor
9   pinMode(PIR_SENSOR_PIN, INPUT);
10  pinMode(LED_PIN, OUTPUT);
11 }
12
13 void loop() {
14   int motionState = digitalRead(PIR_SENSOR_PIN); // Read PIR sensor state
15
16   if (motionState == HIGH) { // Motion detected
17     Serial.println("Motion detected!");
18     digitalWrite(LED_PIN, HIGH); // Turn LED on
19   } else {
20     Serial.println("No motion");
21     digitalWrite(LED_PIN, LOW); // Turn LED off
22   }
23   delay(1000); // Wait for a second before reading again
24 }

```

At the bottom, the "Serial Monitor" window is open, showing the output of the program. The output consists of alternating "No motion" and "Motion detected!" messages, indicating that the PIR sensor is correctly detecting motion and controlling the LED.

Output:

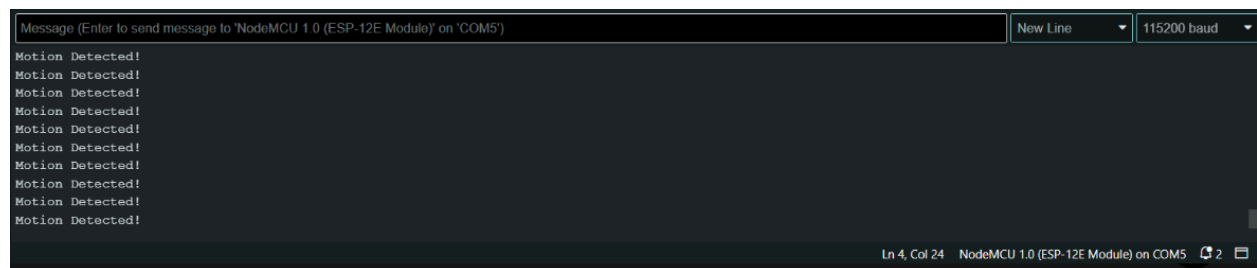


The screenshot shows a Serial Monitor window with a title bar 'Output Serial Monitor x'. Below the title bar is a text input field with the placeholder 'Message (Enter to send message to "NodeMCU 1.0 (ESP-12E Module)" on "COM4")'. The main area of the window displays a series of text messages: 'No motion' followed by 'Motion detected!' in a repeating pattern. The messages are: 'No motion', 'No motion', 'Motion detected!', 'No motion', 'No motion', 'No motion', 'No motion', 'No motion', 'Motion detected!', 'No motion', 'No motion', 'Motion detected!', 'No motion', 'No motion', 'No motion'.

Output Analysis & Troubleshooting:

Common Issues & Fixes:

1.PIR Sensor Always Shows "Motion Detected!"



The screenshot shows a Serial Monitor window with a title bar 'Message (Enter to send message to "NodeMCU 1.0 (ESP-12E Module)" on "COM5")'. The window has two buttons: 'New Line' and '115200 baud'. The main area of the window displays a series of text messages: 'Motion Detected!' repeated multiple times. The messages are: 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!', 'Motion Detected!'. At the bottom right, there is a status bar that reads 'Ln 4, Col 24 NodeMCU 1.0 (ESP-12E Module) on COM5'.

- Reduce PIR sensitivity.
- Add a **10kΩ pull-up resistor** between OUT and 3.3V for signal stabilization.

Conclusion:

This project successfully demonstrated motion detection using a PIR sensor, first by analyzing its behavior using an oscilloscope, and then integrating it with an ESP8266MOD for IoT applications. By following proper wiring, using stabilizing resistors, and troubleshooting common issues, the system reliably detects movement and can be used in security systems, automation, and smart home projects.