**AIM:** Implement IOT based project that motion detection using PIR Sensor, Buzzer, NodeMCU ESP8266 and etc.

* To implement an IoT-based project for motion detection using a PIR (Passive Infrared) sensor, a buzzer, and NodeMCU ESP8266, we need to understand the components involved and their functionality. Let's explore the theory behind each component:

**PIR Sensor:**

* A PIR sensor is a motion detection sensor that detects changes in infrared radiation in its field of view. It consists of a pyroelectric sensor that generates an electrical signal when it detects variations in heat emitted by objects within its range.
* PIR sensors are commonly used for occupancy sensing in security systems, lighting control, and automation applications.

**Buzzer:**

* A buzzer is an electronic component that produces audible sound when activated. It typically consists of a piezoelectric element that vibrates when an electrical signal is applied.
* In our project, we will use a buzzer to generate an alert or alarm when motion is detected by the PIR sensor.

**NodeMCU ESP8266:**

* The NodeMCU ESP8266 is a popular Wi-Fi-enabled microcontroller board that integrates an ESP8266 Wi-Fi module. It provides a cost-effective solution for IoT applications by enabling wireless connectivity and programmability. We will use the NodeMCU ESP8266 to interface with the PIR sensor and buzzer, and transmit data to a remote server or platform.

**Breadboard:**

1. **Breadboard Basics:** A breadboard is a prototyping tool that allows for easy and temporary circuit connections without soldering. It consists of a grid of holes arranged in rows and columns, with metal clips underneath each hole. The rows and columns are typically labelled for easy reference.
2. **Power Supply:** You'll need to provide power to your breadboard circuit. This can be done by connecting a power source, such as a battery or a power supply, to the appropriate rows or columns on the breadboard. Make sure to use the correct voltage and polarity to avoid damaging your components.
3. **Component Placement:** When placing components on the breadboard, it's important to understand the layout and connections. Each hole in a row is electrically connected, but the rows themselves are not connected unless you bridge them with jumper wires. Plan your component placement carefully to ensure proper connections.
4. **Jumper Wires:** Jumper wires are used to create electrical connections between components on the breadboard. They come in different lengths and colors. Shorter wires are ideal for keeping the circuit tidy. You can insert one end of a jumper wire into a hole to establish a connection and connect the other end to another component or part of the circuit.
5. **Pin Configuration:** Some components, such as sensors or microcontrollers, have specific pin configurations. Make sure to consult the datasheets or documentation of your components to identify the correct pin connections on the breadboard. You may need to use jumper wires to connect specific pins to other components or power sources.
6. **Testing and Troubleshooting:** Breadboards allow for easy testing and troubleshooting of circuits. If you encounter issues, double-check the connections, component placement, and power supply. Use a multimeter to verify voltages and continuity in your circuit. Additionally, make sure the components are securely inserted into the breadboard and not loose or improperly seated.

**Project Workflow:**

**Hardware Setup:**

* Connect the VCC and GND pins of the PIR sensor to the respective power supply pins (e.g., 3.3V and GND) on the NodeMCU ESP8266.
* Connect the OUT pin of the PIR sensor to a digital input pin (e.g., D1) on the NodeMCU ESP8266.
* Connect one terminal of the buzzer to a digital output pin (e.g., D2) on the NodeMCU ESP8266 and the other terminal to GND.

**Coding and Programming:**

* Set up the NodeMCU ESP8266 development environment (e.g., Arduino IDE) and install the necessary libraries (e.g., ESP8266WiFi, Blynk).
* Write the code to initialize the Wi-Fi connection, PIR sensor input pin, and buzzer output pin.
* In the main loop, read the input state of the PIR sensor. If motion is detected, activate the buzzer to generate an alarm or alert sound.

**Optional:**

* Configure additional functionalities like sending notifications or data to a remote server or platform (e.g., Blynk).

**Power Supply and Deployment:**

* Power the NodeMCU ESP8266 using a suitable power source (e.g., USB cable, battery).

Deploy the setup in the desired location where motion detection is required.

**Testing and Monitoring:**

* Observe the behavior of the PIR sensor and buzzer when motion is detected. The buzzer should produce sound alerts when motion is detected within the sensor's range.
* Monitor the system's performance and make adjustments as needed to optimize sensitivity, detection range, or other parameters.
* By understanding the theory and following the above steps, we can implement an IoT-based project for motion detection using a PIR sensor, buzzer, and NodeMCU ESP8266. The PIR sensor detects motion, the NodeMCU ESP8266 processes the sensor data, and the buzzer provides audible alerts, creating a simple but effective motion detection system.

**Code:**

int motionSensor = D1;

int buzzer = D6;

int led = D7;

//int LED = D4;  // LED 2

void setup() {

  // put your setup code here, to run once:

  Serial.begin(9600);

  pinMode(LED\_BUILTIN, OUTPUT);

  pinMode(buzzer, OUTPUT);

  pinMode(led, OUTPUT);

  //pinMode(LED, OUTPUT);  //LED ON

  digitalWrite(LED\_BUILTIN, HIGH);

  digitalWrite(buzzer,HIGH);

  digitalWrite(led,HIGH);

  pinMode(motionSensor, INPUT);

  delay(1500);

  Serial.println("Motion sensor test.");

}

unsigned long currentTime;

const unsigned long motionCheckInterval = 150UL;

unsigned long previousMotionCheckTime = 0;

void loop() {

  // put your main code here, to run repeatedly:

  currentTime = millis();

  if (currentTime - previousMotionCheckTime >= motionCheckInterval) {

    int isMotionDetected = digitalRead(motionSensor);

    if (isMotionDetected == 0) {

      Serial.println("Motion ended!");

      digitalWrite(LED\_BUILTIN, HIGH);

      digitalWrite(buzzer,LOW);

      digitalWrite(led, HIGH);

    } else {

      Serial.println("Motion detected!");

      digitalWrite(LED\_BUILTIN, LOW);

      digitalWrite(buzzer,HIGH);

      digitalWrite(led, HIGH);

    }

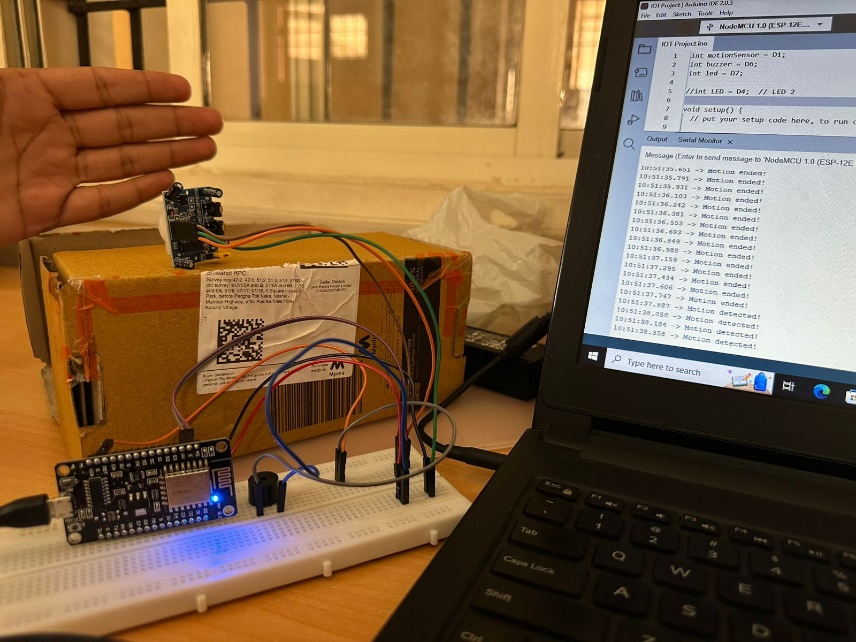
    previousMotionCheckTime = currentTime;

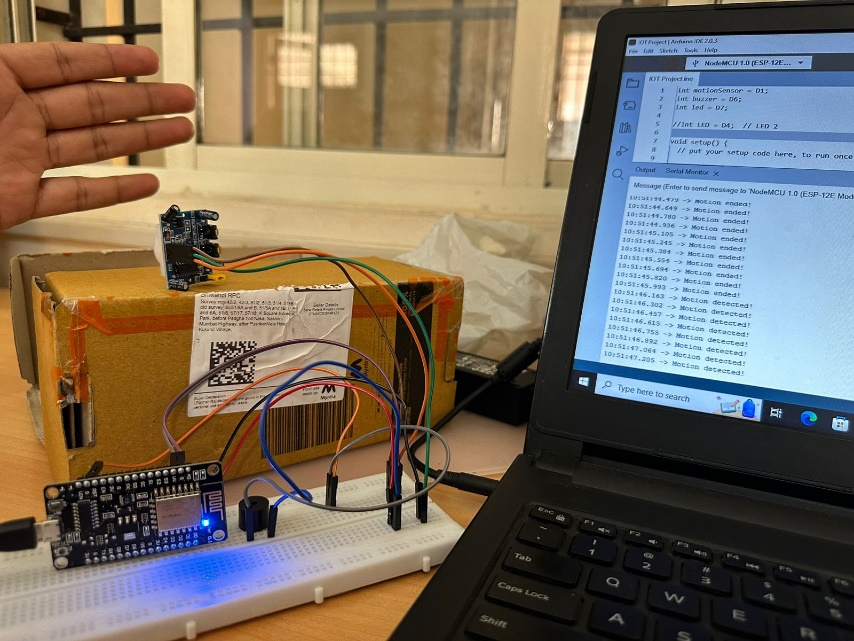
  }

}

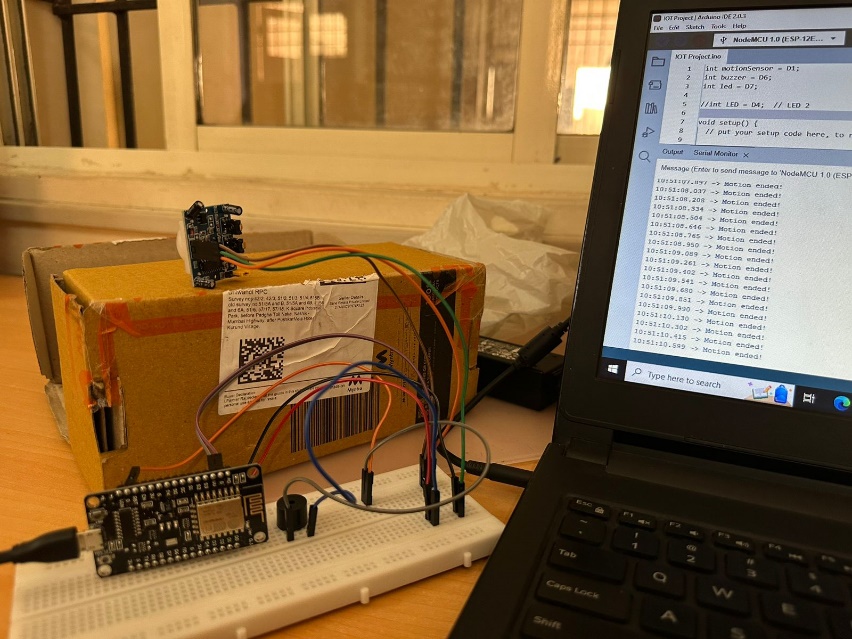
**Output:**

**Motion Detected:**

****

****

**Motion Not Detected:**

****

**Conclusion:**

* In conclusion, the motion detection project in IoT successfully demonstrates the capabilities of integrating Internet of Things (IoT) technology with motion sensors to create a responsive and automated system. The project aimed to detect and respond to human movement within a given space, providing real-time alerts or triggering specific actions based on detected motion.
* The motion detection system offered numerous benefits. It enhanced security by monitoring and alerting users about any unauthorized movement in a designated area. This feature proved valuable in both residential and commercial settings, helping to prevent theft, vandalism, or intrusion.
* The project showcased the potential of IoT technology in enabling smart and connected environments.
* Overall, the motion detection project in IoT successfully demonstrated the capabilities of a responsive and automated system that leverages motion sensors and IoT technology. By detecting motion and responding accordingly, the project showcased the potential for enhanced security, energy efficiency, and the creation of smart environments. Further advancements and refinements in sensor technology, data processing algorithms, and connectivity will continue to drive the evolution of motion detection systems in the IoT landscape.