

Project-2

IoT Architecture and Protocols

Intrusion Detection System with Arduino Cloud



By:

VU22CSEN0600047-Ganesh Koushik

VU22CSEN0600082-Kintali sai kiran

VU22CSEN0600046- B. Sravani

Table of Contents

- Abstract.....3
- Objectives.....4
- Introduction.....5
- Methodology.....6-10
- Hardware Requirements...10-12
- Results.....12-13
- Future Scope.....14

Abstract

This project developed an IoT-based system to detect motion and vibration using two ESP8266 modules integrated with Arduino Cloud.

A **sensor node**, equipped with a PIR and vibration sensor, monitors environmental changes, sending data via Wi-Fi to Arduino Cloud.

A **gateway node** triggers a buzzer when both sensors detect activity, based on cloud automation rules.

The aim was to create a low-cost, real-time monitoring prototype for applications like home security or industrial safety.

The methodology included hardware interfacing, coding in Arduino IDE, and configuring Arduino Cloud with Things and dashboards. Challenges such as COM port upload errors and sensors reading "0" were overcome by adjusting pin assignments and permissions.

The system successfully updated the cloud dashboard in real-time, with the buzzer activating when both sensors reported HIGH states. This demonstrates the potential of affordable IoT solutions, with scope for future enhancements like additional sensors or mobile alerts.

Objectives

The primary goal of this project was to develop an IoT-based system capable of detecting motion and vibration events and providing audible alerts through a buzzer, leveraging ESP8266 modules and Arduino Cloud. The specific objectives were:

- To interface a PIR sensor and a vibration sensor with an ESP8266 sensor node for accurate event detection.
- To transmit sensor data wirelessly to Arduino Cloud for real-time monitoring and storage.
- To configure a gateway ESP8266 node to activate a buzzer based on cloud-processed sensor states.
- To create a user-friendly dashboard in Arduino Cloud for visualizing sensor and buzzer status.
- To troubleshoot and resolve implementation challenges, such as upload errors and sensor malfunctions, ensuring system reliability.

These objectives aimed to produce a functional prototype that combines low-cost hardware with cloud connectivity, suitable for applications requiring immediate alerts, such as intruder detection or equipment monitoring.

The successful achievement of these goals would validate the system's potential as a scalable IoT solution.

Introduction

The Internet of Things (IoT) has transformed how we interact with our environments, enabling interconnected devices to collect, process, and act on data seamlessly.

The demand for affordable, real-time systems has grown in security and monitoring domains, yet many traditional solutions remain expensive or lack remote accessibility.

This project addresses these gaps by developing an IoT-based motion and vibration detection system using ESP8266 microcontrollers and Arduino Cloud.

The ESP8266, a cost-effective Wi-Fi-enabled microcontroller, serves as the core of both the sensor and gateway nodes.

The **sensor node** integrates a PIR sensor to detect motion and a vibration sensor to identify physical disturbances, while the **gateway node** triggers a buzzer when both conditions are met, as determined by cloud-based automation.

Arduino Cloud provides a platform for **data visualization** and control, making the system accessible from anywhere with internet access.

This work is significant for its simplicity and adaptability, offering applications in home security, industrial settings or environmental studies.

Methodology

System Design

The system comprises two ESP8266 modules: a **sensor node** and a **gateway node**, connected via **Arduino Cloud**.

The sensor node detects motion and vibration, sending data to the cloud, which processes it and controls the gateway's buzzer.

Software Development

The software was developed using Arduino IDE with the ArduinolotCloud and ESP8266WiFi libraries.

Sensor Node: Reads PIR sensor (D2, GPIO 4) and vibration sensor (D6, GPIO 12), converting states to strings (PIR_status, Vibration_status) and sending them to Arduino Cloud every second.

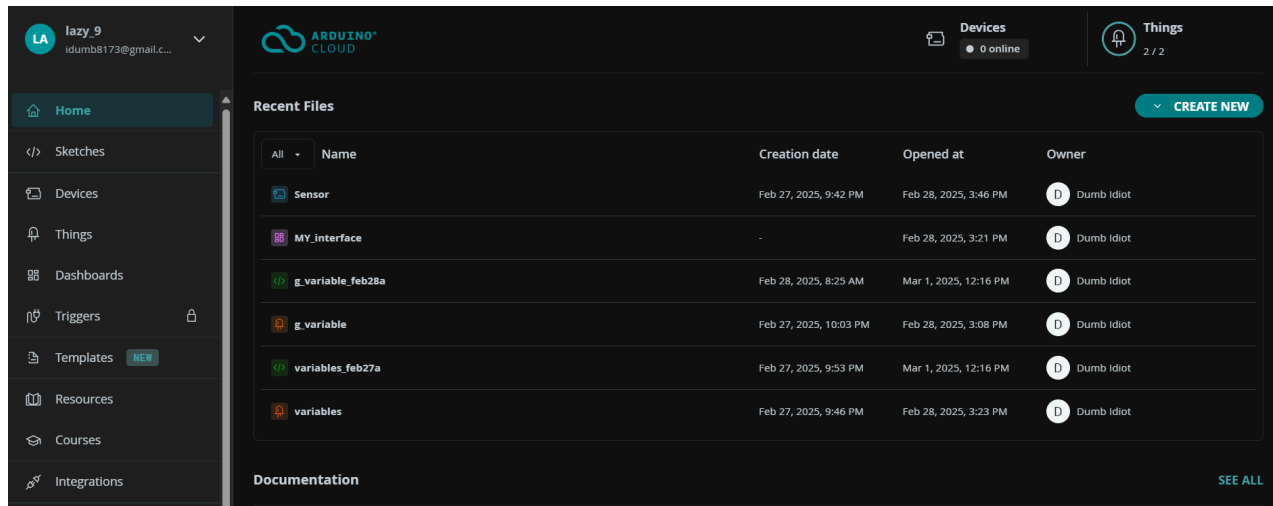
Gateway Node: Monitors a boolean variable (buzzer) from the cloud, activating the buzzer (D5, GPIO 14) when true.

Cloud Configuration:

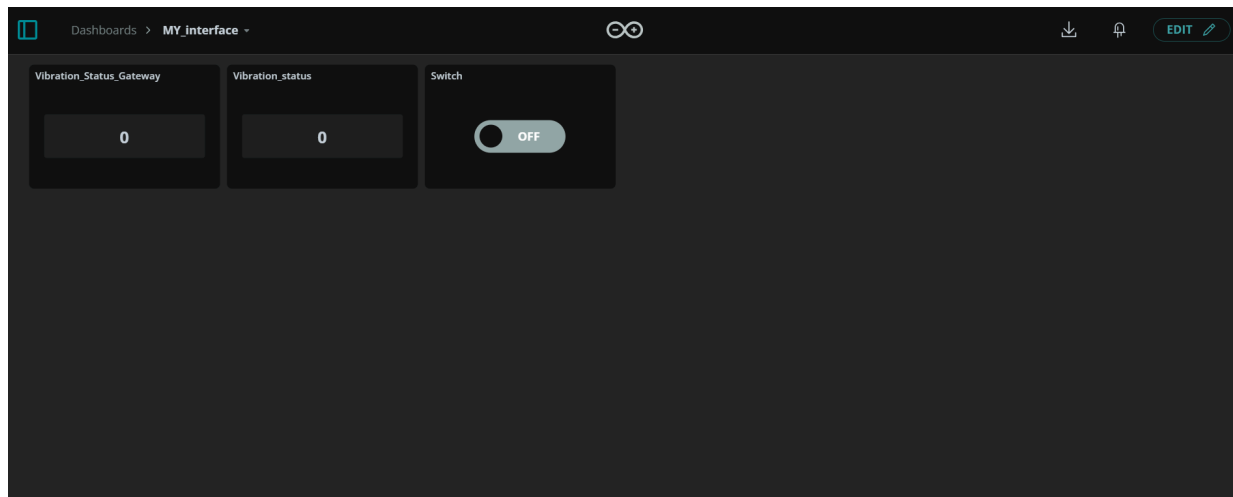
Two Things were created—**SensorNodeThing** (with PIR_status and Vibration_status) and **GatewayNodeThing** (with buzzer).

Automation rules were set:

PIR_status == "1" AND Vibration_status == "1" sets buzzer = true; otherwise, buzzer = false.



Arduino IoT Cloud



Dashboard

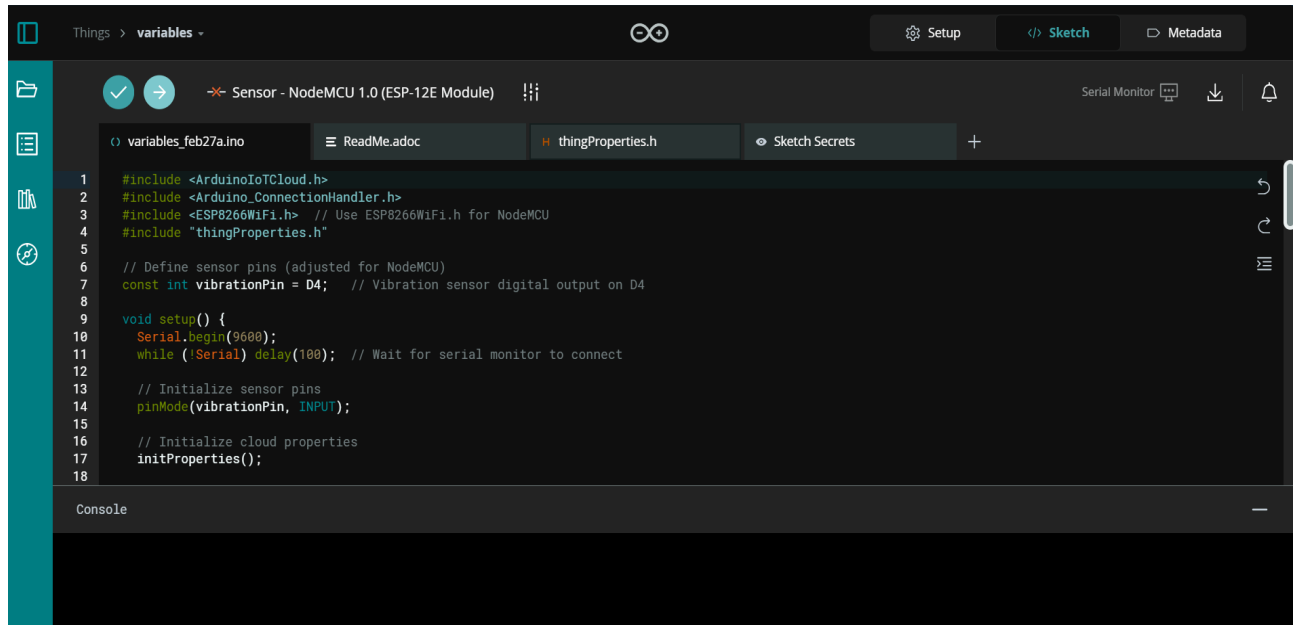
Implementation

The process involved:

- Wiring sensors and buzzer to respective ESP8266 pins.
- Writing and uploading code via Arduino IDE.
- Configuring Arduino Cloud with Things, variables, and dashboard widgets.
- Testing and troubleshooting—initial COM3 upload errors were fixed by running IDE as admin and sensor “0”

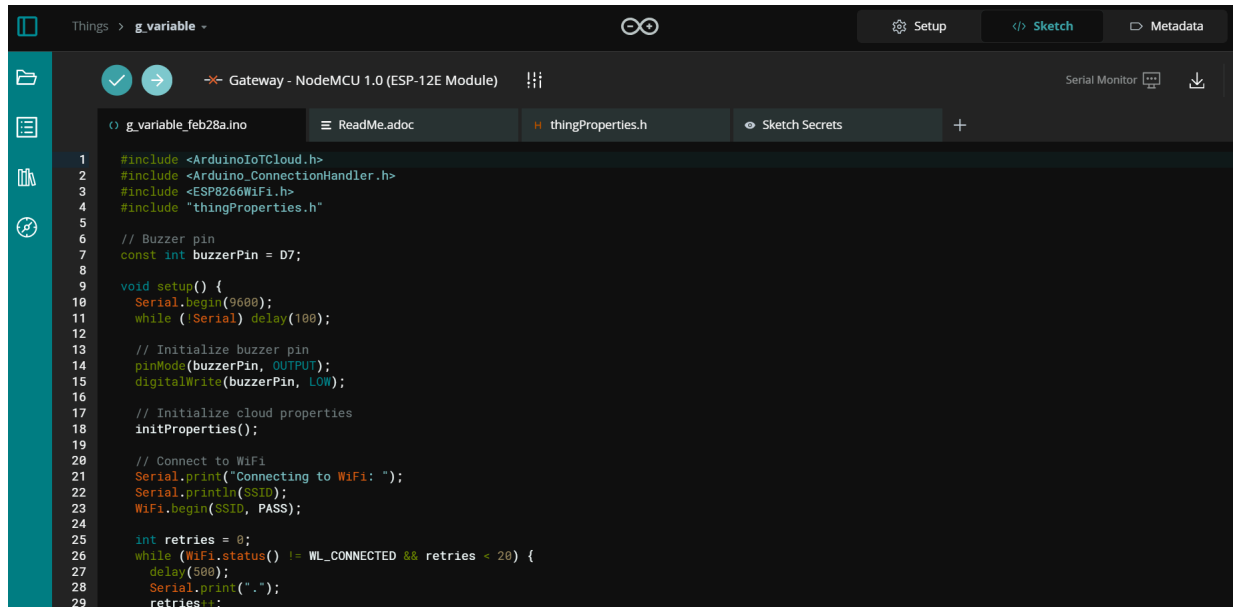
readings were resolved by moving the vibration sensor from D3 to D6 and adjusting sensitivity.

Arduino Cloud Sketches



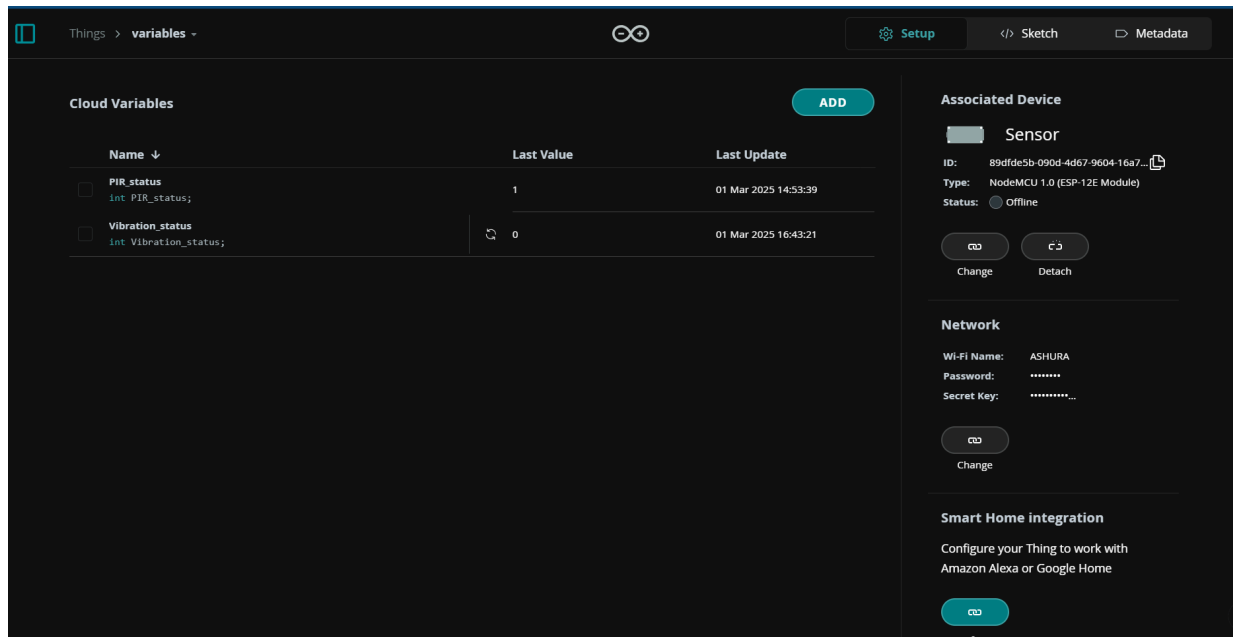
```
1 #include <ArduinoIoTCloud.h>
2 #include <Arduino_ConnectionHandler.h>
3 #include <ESP8266WiFi.h> // Use ESP8266WiFi.h for NodeMCU
4 #include "thingProperties.h"
5
6 // Define sensor pins (adjusted for NodeMCU)
7 const int vibrationPin = D4; // Vibration sensor digital output on D4
8
9 void setup() {
10   Serial.begin(9600);
11   while (!Serial) delay(100); // Wait for serial monitor to connect
12
13   // Initialize sensor pins
14   pinMode(vibrationPin, INPUT);
15
16   // Initialize cloud properties
17   initProperties();
18 }
```

Sketch for Sensor node

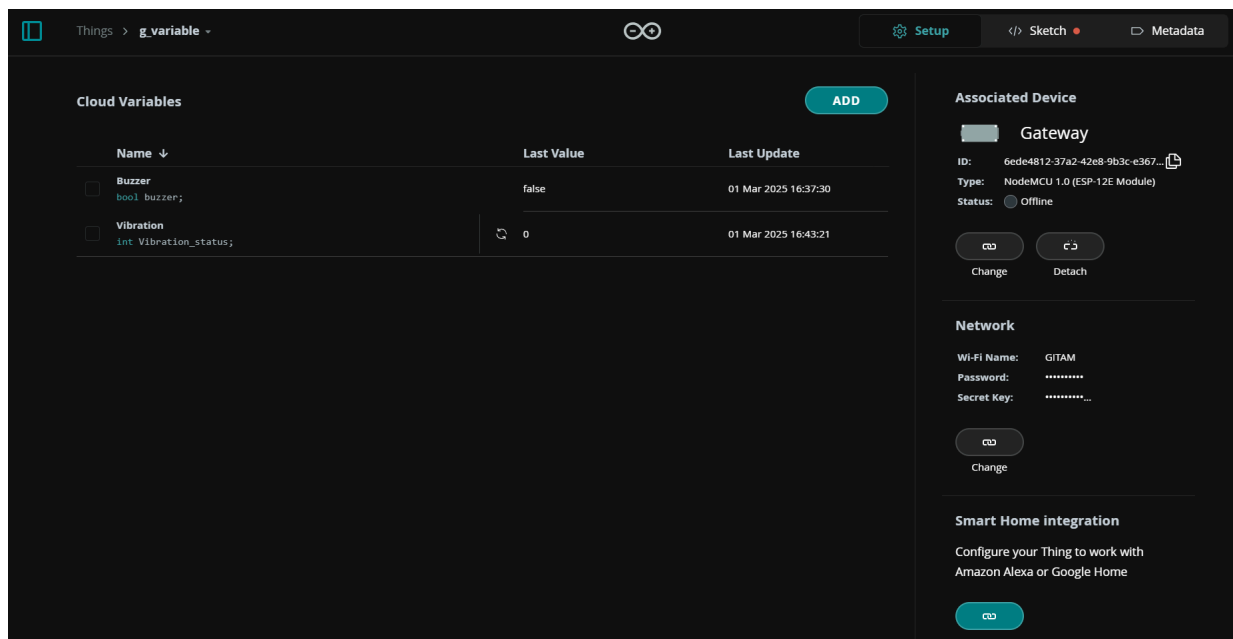


```
1 #include <ArduinoIoTCloud.h>
2 #include <Arduino_ConnectionHandler.h>
3 #include <ESP8266WiFi.h>
4 #include "thingProperties.h"
5
6 // Buzzer pin
7 const int buzzerPin = D7;
8
9 void setup() {
10   Serial.begin(9600);
11   while (!Serial) delay(100);
12
13   // Initialize buzzer pin
14   pinMode(buzzerPin, OUTPUT);
15   digitalWrite(buzzerPin, LOW);
16
17   // Initialize cloud properties
18   initProperties();
19
20   // Connect to WiFi
21   Serial.print("Connecting to WiFi: ");
22   Serial.println(SSID);
23   WiFi.begin(SSID, PASS);
24
25   int retries = 0;
26   while (WiFi.status() != WL_CONNECTED && retries < 20) {
27     delay(500);
28     Serial.print(".");
29     retries++;
30   }
```

Sketch for Gateway node



Sensor node setup



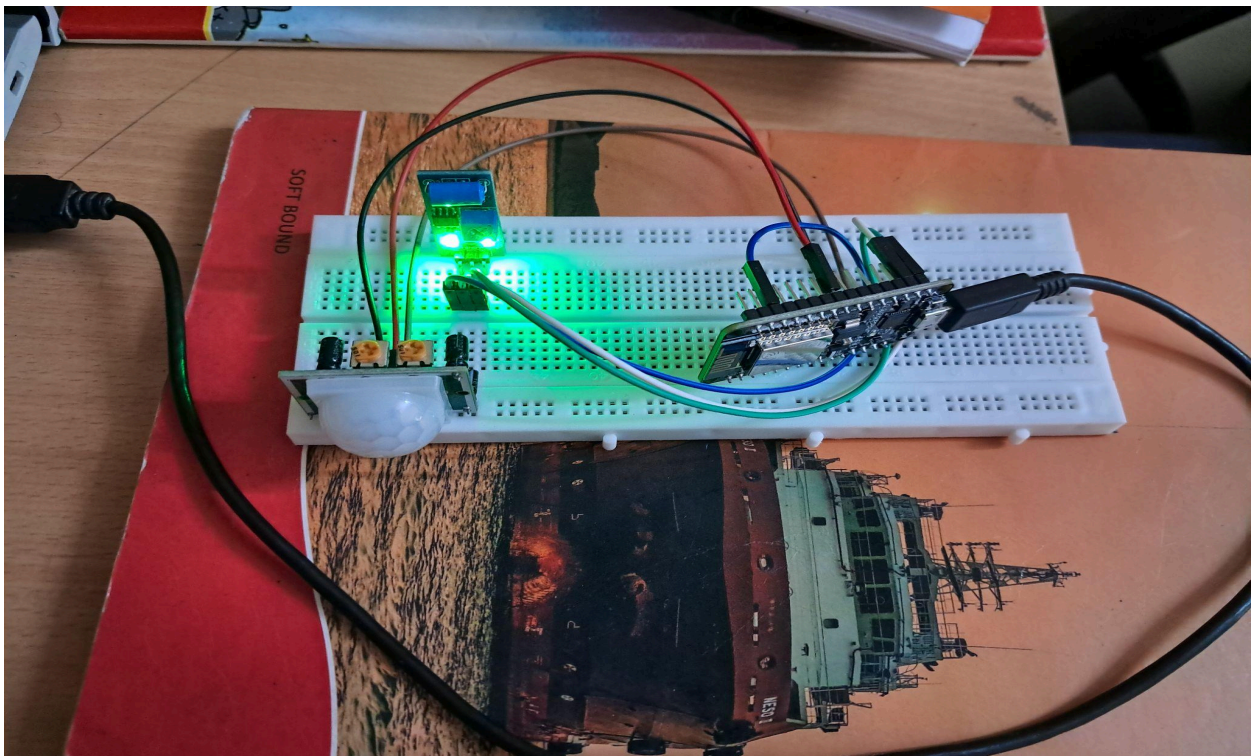
Gateway node setup

Hardware Requirements

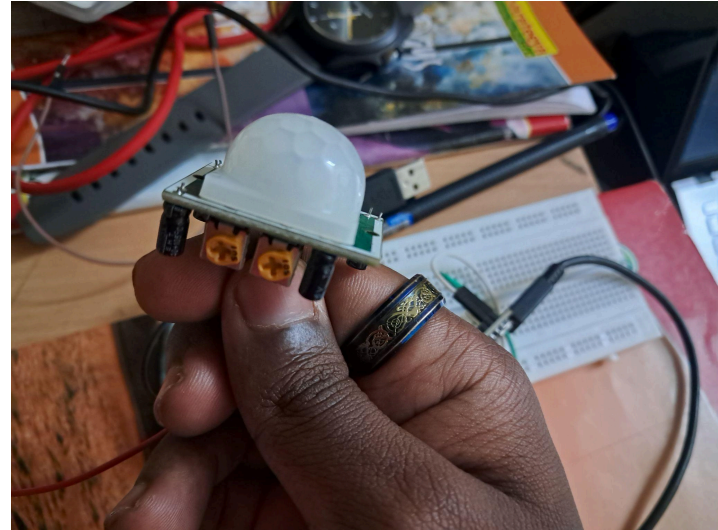
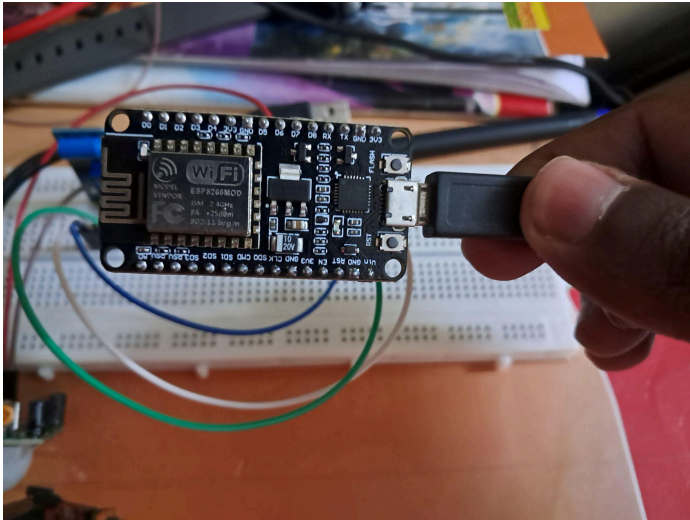
This section details the hardware used to build the system. The setup leverages affordable, widely available components to ensure accessibility and scalability.

- **ESP8266 (NodeMCU):** Two units—one as the **sensor node**, one as the **gateway node** features Wi-Fi connectivity and multiple GPIO pins.

Components:



Sensor Node setup



Node MCU and PIR sensor

Connections Table:

Component	Pin	Sensor node	Gateway Node	Note
ESP8266	3V3	Power Supply	Power Supply	Connected to 3.3V USB
	GND	Ground	Ground	Common Ground
PIR Sensor	VCC	3V3	-	3.3V power
	GND	GND	-	Ground connection
	OUT	D2	-	Digital output for motion detection
Vibration Sensor	VCC	3V3	-	3.3V power

Component	Pin	Sensor node	Gateway Node	Note
	GND	GND	-	Ground connection
	DO	D6	-	Digital output for Vibration detection
Buzzer	Positive	-	D5	Power supply
	Negative	-	GND	Ground
USB Cable	-	USB Port	USB Port	For power and programming

Results

The system was tested after resolving initial challenges.

- **Upload Success:** Code was successfully uploaded to both ESP8266 modules after fixing COM3 access issues by running Arduino IDE as administrator and verifying port settings.
- **Sensor Readings:** Initial tests showed PIR_status and Vibration_status stuck at “0” on the dashboard. Serial Monitor debugging revealed sensors weren’t triggering, moving the vibration sensor to D6 (GPIO 12) and adjusting sensitivity corrected this, yielding “1” when triggered.
- **Cloud Integration:** Arduino Cloud dashboard updated in real-time, showing PIR_status and Vibration_status as “0” or “1” based on sensor states. Automation rules successfully toggled the buzzer to true when both were “1”.
- **Buzzer Response:** The gateway node’s buzzer activated reliably when both sensors detected events, stopping when either returned to “0”.
- **Challenges:** Serial port closing and garbled output were fixed by ensuring 115200 baud consistency and stabilizing hardware connections.

Future Scope

This project lays a foundation for further development:

1. **Multi-Sensor Expansion:** Add temperature, sound, or gas sensors for broader monitoring.
2. **Mobile Alerts:** Integrate notifications via email or the Arduino Cloud app for remote alerts.
3. **Data Logging:** Store historical data in the cloud for analysis.
4. **Power Efficiency:** Use deep sleep modes on ESP8266 to reduce power consumption.
5. **Scalability:** Deploy multiple sensor-gateway pairs for large-scale applications (e.g., warehouse security).

These enhancements could elevate the system into a robust, commercial-grade IoT solution for diverse use cases.