SRM Institute of Science and Technology

Department of Computer Applications

Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204

Circular – 2020-21

MCA GAI 1st semester

Intelligent Internet of Things (IIoT) (PGI20D04J)

Lab Manual

Lab 1: Arduino Installation and Blink LED using Node MCU ESP8266

Title

Arduino Installation and Blink LED using Node MCU ESP8266

Aim

To understand the process of setting up the Arduino IDE for ESP8266 development and to write a basic program to blink an onboard LED.

Procedure

1. **Install Arduino IDE:** Download and install the latest version of the Arduino IDE from the official Arduino website.
2. **Add ESP8266 Board Manager URL:**
   - Go to `File > Preferences`.
   - In the "Additional Boards Manager URLs" field, add: `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
   - Click "OK".
3. **Install ESP8266 Boards:**
   - Go to `Tools > Board > Boards Manager...`
   - Search for "esp8266" and install the "ESP8266 by ESP8266 Community" package.
4. **Select NodeMCU Board:**
   - Go to `Tools > Board > ESP8266 Boards` and select "NodeMCU 1.0 (ESP-12E Module)".
5. **Connect NodeMCU:** Connect your NodeMCU ESP8266 board to your computer using a micro-USB cable.
6. **Select Port:** Go to `Tools > Port` and select the serial port corresponding to your NodeMCU (it might appear as `COMx` on Windows or `/dev/ttyUSBx` on Linux/macOS).
7. **Write Code:** Copy the provided source code into the Arduino IDE.
8. **Upload Code:** Click the "Upload" button (right arrow icon) to compile and upload the code to your NodeMCU.
9. **Observe:** After successful upload, observe the onboard LED blinking.

Source Code

```
// Lab 1: Blink LED on NodeMCU ESP8266
```

```
void setup() {
  // Initialize the LED pin as an output.
  // The onboard LED on NodeMCU ESP-12E is usually connected to GPIO2 (D4).
  // Some boards might have it on GPIO16 (D0). Check your board's pinout.
  pinMode(D4, OUTPUT); // Using D4 (GPIO2) for the onboard LED
}

void loop() {
  digitalWrite(D4, HIGH); // Turn the LED on (HIGH is usually OFF for onboard
LED, LOW is ON)
  delay(1000);            // Wait for a second
  digitalWrite(D4, LOW);  // Turn the LED off (LOW is usually ON for onboard
LED, HIGH is OFF)
  delay(1000);            // Wait for a second
}
```

Input

- NodeMCU ESP8266 board connected to a power source (via USB).

Expected Output

The onboard LED on the NodeMCU ESP8266 board will blink at an interval of 1 second (1 second ON, 1 second OFF).

Lab 2: Interfacing Light Sensor with ESP8266 NODE MCU WiFi Board

Title

Interfacing Light Sensor (LDR) with ESP8266 NodeMCU WiFi Board

Aim

To connect a Photoresistor (LDR) to the NodeMCU ESP8266 board to measure ambient light intensity and display the readings on the serial monitor.

Procedure

1. **Gather Components:** NodeMCU ESP8266, LDR (Photoresistor), 10k Ohm Resistor, Breadboard, Jumper Wires.
2. **Circuit Connection:**
    - Connect one leg of the LDR to the 3.3V pin of the NodeMCU.
    - Connect the other leg of the LDR to analog input pin A0 of the NodeMCU.
    - Connect one leg of the 10k Ohm resistor to the same analog input pin A0.
    - Connect the other leg of the 10k Ohm resistor to the GND pin of the NodeMCU. This forms a voltage divider circuit.
3. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected as in Lab 1.
4. **Write Code:** Copy the provided source code into the Arduino IDE.
5. **Upload Code:** Upload the code to your NodeMCU.
6. **Open Serial Monitor:** Open the Serial Monitor (`Tools > Serial Monitor`) and set the baud rate to 115200.
7. **Observe:** Vary the light intensity falling on the LDR and observe the changing readings in the Serial Monitor.

Source Code

```
// Lab 2: Interfacing Light Sensor (LDR) with NodeMCU ESP8266

const int ldrPin = A0; // Analog pin A0 connected to the LDR voltage divider

void setup() {
  Serial.begin(115200); // Initialize serial communication at 115200 baud rate
  Serial.println("LDR Sensor Readings:");
}

void loop() {
  int ldrValue = analogRead(ldrPin); // Read the analog value from the LDR pin
  Serial.print("Light Intensity: ");
  Serial.println(ldrValue); // Print the LDR value to the Serial Monitor
  delay(500); // Wait for 500 milliseconds before the next reading
}
```

Input

- Ambient light changes (e.g., covering the LDR, shining a light on it).

Expected Output

The Serial Monitor will display numerical values representing the light intensity. These values will change as the amount of light falling on the LDR varies (e.g., higher values for more light, lower values for less light, or vice-versa depending on the voltage divider configuration).

Lab 3: Interfacing Infrared (IR) Sensor with Node MCU ESP8266

Title

Interfacing Infrared (IR) Sensor with NodeMCU ESP8266

Aim

To connect an IR proximity sensor to the NodeMCU ESP8266 board to detect the presence of objects and display the detection status on the serial monitor.

Procedure

1. **Gather Components:** NodeMCU ESP8266, IR Proximity Sensor (3-pin module: VCC, GND, OUT), Breadboard, Jumper Wires.
2. **Circuit Connection:**
    - Connect the VCC pin of the IR sensor to the 3.3V pin of the NodeMCU.
    - Connect the GND pin of the IR sensor to the GND pin of the NodeMCU.
    - Connect the OUT (digital output) pin of the IR sensor to a digital GPIO pin on the NodeMCU (e.g., D1/GPIO5).
3. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
4. **Write Code:** Copy the provided source code into the Arduino IDE.
5. **Upload Code:** Upload the code to your NodeMCU.
6. **Open Serial Monitor:** Open the Serial Monitor and set the baud rate to 115200.
7. **Observe:** Move an object in front of the IR sensor and observe the detection status in the Serial Monitor.

Source Code

```
// Lab 3: Interfacing Infrared (IR) Sensor with NodeMCU ESP8266

const int irSensorPin = D1; // Digital pin D1 (GPIO5) connected to the IR
sensor's OUT pin

void setup() {
  Serial.begin(115200); // Initialize serial communication
  pinMode(irSensorPin, INPUT); // Set the IR sensor pin as an input
  Serial.println("IR Sensor Object Detection:");
}

void loop() {
  int irValue = digitalRead(irSensorPin); // Read the digital value from the IR
sensor pin

  if (irValue == LOW) { // IR sensor typically outputs LOW when an object is
detected
    Serial.println("Object Detected!");
  } else {
    Serial.println("No Object");
  }
  delay(200); // Small delay to avoid rapid readings
}
```

Input

- Placing an object (e.g., hand) in front of the IR sensor.
- Removing the object from in front of the IR sensor.

Expected Output

The Serial Monitor will display "Object Detected!" when an object is close enough to the IR sensor, and "No Object" when no object is detected within its range.

Lab 4: Interfacing Temperature Humidity Sensor with Node MCU ESP8266

Title

Interfacing Temperature and Humidity Sensor (DHT11/DHT22) with NodeMCU ESP8266

Aim

To connect a DHT11 or DHT22 sensor to the NodeMCU ESP8266 board to measure ambient temperature and humidity and display the readings on the serial monitor.

Procedure

1. **Gather Components:** NodeMCU ESP8266, DHT11 or DHT22 Temperature & Humidity Sensor, Breadboard, Jumper Wires, 10k Ohm Resistor (optional, for DHT11/DHT22 data line pull-up).
2. **Circuit Connection:**
    - Connect the VCC pin of the DHT sensor to the 3.3V pin of the NodeMCU.
    - Connect the GND pin of the DHT sensor to the GND pin of the NodeMCU.
    - Connect the Data pin of the DHT sensor to a digital GPIO pin on the NodeMCU (e.g., D2/GPIO4).
    - (Optional but recommended for stability) Add a 10k Ohm pull-up resistor between the Data pin and VCC.
3. **Install DHT Sensor Library:**
    - Go to `Sketch > Include Library > Manage Libraries...`
    - Search for "DHT sensor library" by Adafruit and install it.
    - Also, search for "Adafruit Unified Sensor" and install it, as it's a dependency.
4. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
5. **Write Code:** Copy the provided source code into the Arduino IDE.
6. **Upload Code:** Upload the code to your NodeMCU.
7. **Open Serial Monitor:** Open the Serial Monitor and set the baud rate to 115200.
8. **Observe:** Observe the temperature and humidity readings displayed in the Serial Monitor.

Source Code

```
// Lab 4: Interfacing Temperature Humidity Sensor (DHT11/DHT22) with NodeMCU
ESP8266

#include <Adafruit_Sensor.h> // Required for DHT library
#include <DHT.h>             // DHT sensor library

#define DHTPIN D2     // Digital pin D2 (GPIO4) connected to the DHT sensor's
Data pin
#define DHTTYPE DHT11 // Or DHT22 if you are using a DHT22 sensor

DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

void setup() {
  Serial.begin(115200); // Initialize serial communication
  Serial.println("DHT11/DHT22 Sensor Readings:");
  dht.begin(); // Start the DHT sensor
}
```

```
void loop() {
  // Wait a few seconds between measurements.
  delay(2000);

  // Read temperature as Celsius (the default)
  float h = dht.readHumidity();
  // Read temperature as Celsius (the default)
  float t = dht.readTemperature();

  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C");
}
```

Input

- Ambient temperature and humidity changes.

Expected Output

The Serial Monitor will display the current humidity in percentage (%) and temperature in degrees Celsius (∘C). If there are reading errors, it will display "Failed to read from DHT sensor!".

Lab 5: Interfacing MQ 4 GAS Sensor with Node MCU ESP8266

Title

Interfacing MQ-4 Gas Sensor with NodeMCU ESP8266

Aim

To connect an MQ-4 gas sensor to the NodeMCU ESP8266 board to detect the presence of methane (natural gas) and display the analog readings on the serial monitor.

Procedure

1. **Gather Components:** NodeMCU ESP8266, MQ-4 Gas Sensor Module (with analog and digital outputs), Breadboard, Jumper Wires.
2. **Circuit Connection:**
    - Connect the VCC pin of the MQ-4 sensor module to the 3.3V pin of the NodeMCU.
    - Connect the GND pin of the MQ-4 sensor module to the GND pin of the NodeMCU.
    - Connect the Analog Output (A0) pin of the MQ-4 sensor module to the analog input pin A0 of the NodeMCU.
    - (Optional) Connect the Digital Output (D0) pin of the MQ-4 sensor module to a digital GPIO pin (e.g., D5/GPIO14) for threshold-based detection.
3. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
4. **Write Code:** Copy the provided source code into the Arduino IDE.
5. **Upload Code:** Upload the code to your NodeMCU.
6. **Open Serial Monitor:** Open the Serial Monitor and set the baud rate to 115200.
7. **Observe:** Expose the MQ-4 sensor to a small amount of natural gas (e.g., from a gas lighter, briefly, ensuring good ventilation) and observe the changing analog readings in the Serial Monitor.

Source Code

```
// Lab 5: Interfacing MQ-4 Gas Sensor with NodeMCU ESP8266

const int mq4AnalogPin = A0; // Analog pin A0 connected to the MQ-4 sensor's A0
pin

void setup() {
  Serial.begin(115200); // Initialize serial communication
  Serial.println("MQ-4 Gas Sensor Readings:");
}

void loop() {
  int sensorValue = analogRead(mq4AnalogPin); // Read the analog value from the
MQ-4 sensor

  Serial.print("Gas Concentration (Analog Value): ");
  Serial.println(sensorValue); // Print the analog reading to the Serial Monitor

  // You can add logic here to trigger an alert if sensorValue exceeds a
threshold
  // For example:
  // if (sensorValue > 500) {
  //   Serial.println("High Gas Concentration Detected!");
```

```
  // }

  delay(500); // Wait for 500 milliseconds before the next reading
}
```

Input

- Presence of methane/natural gas near the sensor.
- Normal ambient air.

Expected Output

The Serial Monitor will display numerical analog values. These values will increase when methane gas is detected and decrease when the air is clear. Higher values indicate higher gas concentration.

Lab 6: Interfacing Relay and Control LIGHT with Node MCU ESP8266

Title

Interfacing Relay and Control Light with NodeMCU ESP8266

Aim

To connect a relay module to the NodeMCU ESP8266 board to control an AC light bulb (or any other AC appliance) safely.

Procedure

1. **Gather Components:** NodeMCU ESP8266, 5V Relay Module (single channel), Light bulb with holder and plug, AC power cord, Breadboard, Jumper Wires. **(CAUTION: Working with AC mains voltage can be dangerous. Ensure all connections are secure and take necessary safety precautions. If unsure, seek assistance from a qualified person.)**
2. **Circuit Connection:**
    - Connect the VCC pin of the Relay Module to the 3.3V or 5V pin of the NodeMCU (check your relay module's voltage requirement).
    - Connect the GND pin of the Relay Module to the GND pin of the NodeMCU.
    - Connect the IN (control) pin of the Relay Module to a digital GPIO pin on the NodeMCU (e.g., D3/GPIO0).
    - **AC Wiring (EXTREME CAUTION):**
        - Cut one of the two wires of your AC power cord (e.g., the Live wire).
        - Connect one end of the cut wire to the "Common" (COM) terminal of the relay.
        - Connect the other end of the cut wire to the "Normally Open" (NO) terminal of the relay.
        - Plug the light bulb into the AC power cord.
3. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
4. **Write Code:** Copy the provided source code into the Arduino IDE.
5. **Upload Code:** Upload the code to your NodeMCU.
6. **Observe:** After uploading, the light bulb should toggle on and off at regular intervals.

Source Code

```
// Lab 6: Interfacing Relay and Control Light with NodeMCU ESP8266

const int relayPin = D3; // Digital pin D3 (GPIO0) connected to the Relay
module's IN pin

void setup() {
  Serial.begin(115200); // Initialize serial communication
  pinMode(relayPin, OUTPUT); // Set the relay control pin as an output
  Serial.println("Relay Control for Light:");
  // Ensure the light is off initially (depends on your relay's HIGH/LOW
activation)
  // For most common relays, LOW activates, HIGH deactivates. Adjust as needed.
  digitalWrite(relayPin, HIGH); // Assuming HIGH keeps the relay OFF (light OFF)
}

void loop() {
```

```
  Serial.println("Light ON");
  digitalWrite(relayPin, LOW); // Activate relay (turn light ON)
  delay(5000); // Keep light ON for 5 seconds

  Serial.println("Light OFF");
  digitalWrite(relayPin, HIGH); // Deactivate relay (turn light OFF)
  delay(5000); // Keep light OFF for 5 seconds
}
```

Input

- NodeMCU powered on and connected to the relay and light bulb.

Expected Output

The light bulb connected to the relay will turn ON for 5 seconds, then turn OFF for 5 seconds, and this cycle will repeat continuously.

Lab 7: Create an Access Point and a Webserver using Node MCU ESP8266

Title

Create an Access Point and a Webserver using NodeMCU ESP8266

Aim

To configure the NodeMCU ESP8266 as a Wi-Fi Access Point (AP) and host a simple web server that can be accessed by other devices.

Procedure

1. **Gather Components:** NodeMCU ESP8266, Micro-USB cable.
2. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
3. **Write Code:** Copy the provided source code into the Arduino IDE.
4. **Upload Code:** Upload the code to your NodeMCU.
5. **Open Serial Monitor:** Open the Serial Monitor and set the baud rate to 115200. Note the IP address printed.
6. **Connect to AP:** On your computer or smartphone, search for Wi-Fi networks. You should see a new network named "ESP8266-AP" (or whatever you set in the code). Connect to this network using the password "12345678".
7. **Access Webserver:** Open a web browser on the connected device and navigate to the IP address printed in the Serial Monitor (e.g., `192.168.4.1`).
8. **Observe:** The web browser should display the "Hello from ESP8266 Web Server!" message.

Source Code

```
// Lab 7: Create an Access Point and a Webserver using NodeMCU ESP8266

#include <ESP8266WiFi.h> // Include the ESP8266 WiFi library
#include <ESP8266WebServer.h> // Include the ESP8266 WebServer library

const char *ssid = "ESP8266-AP";     // The name of the Wi-Fi network (SSID)
const char *password = "12345678"; // The password for the Wi-Fi network

ESP8266WebServer server(80); // Create a web server object on port 80

void handleRoot() {
  // Function to handle requests to the root URL "/"
  server.send(200, "text/html", "<h1>Hello from ESP8266 Web Server!</h1><p>This
is a simple web page hosted on NodeMCU.</p>");
}

void setup() {
  Serial.begin(115200); // Initialize serial communication
  Serial.println();
  Serial.print("Setting up AP named: ");
  Serial.println(ssid);

  WiFi.softAP(ssid, password); // Start the ESP8266 in Access Point mode

  IPAddress myIP = WiFi.softAPIP(); // Get the IP address of the Access Point
```

```
  Serial.print("AP IP address: ");
  Serial.println(myIP);

  server.on("/", handleRoot); // Define what to do when a client requests the
root URL
  server.begin();             // Start the web server
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient(); // Handle incoming client requests
}
```

Input

- A client device (computer, smartphone) with Wi-Fi capability.
- Web browser on the client device.

Expected Output

1. The NodeMCU will broadcast a Wi-Fi network named "ESP8266-AP".
2. The client device will successfully connect to "ESP8266-AP".
3. When navigating to the NodeMCU's AP IP address (e.g., `192.168.4.1`) in a web browser, a simple web page displaying "Hello from ESP8266 Web Server!" will be shown.

Lab 8: Interfacing Temperature Humidity Sensor and Visualize data in Cloud Server with Node MCU ESP8266

Title

Interfacing Temperature and Humidity Sensor and Visualize Data in Cloud Server with NodeMCU ESP8266

Aim

To read temperature and humidity data from a DHT sensor using NodeMCU ESP8266 and publish this data to a cloud platform (e.g., ThingSpeak, Blynk, or a custom MQTT broker) for visualization.

Procedure

1. **Gather Components:** NodeMCU ESP8266, DHT11/DHT22 Sensor, Breadboard, Jumper Wires.
2. **Circuit Connection:** Connect the DHT sensor to NodeMCU as in Lab 4.
3. **Cloud Platform Setup (Example: ThingSpeak):**
    - Create an account on ThingSpeak (thingspeak.com).
    - Create a new channel. Note down the "Channel ID" and "Write API Key".
    - Add two fields to your channel: "Temperature" and "Humidity".
4. **Install Libraries:**
    - Install "DHT sensor library" and "Adafruit Unified Sensor" (if not already done, as in Lab 4).
    - Install "ThingSpeak" library (or the relevant library for your chosen cloud platform) via `Sketch > Include Library > Manage Libraries...`.
5. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
6. **Write Code:** Copy the provided source code, replacing placeholders with your Wi-Fi credentials and ThingSpeak API key/channel ID.
7. **Upload Code:** Upload the code to your NodeMCU.
8. **Open Serial Monitor:** Open the Serial Monitor to observe connection status and readings.
9. **Observe Cloud Dashboard:** Go to your ThingSpeak channel's "Private View" or "Public View" to see the real-time plots of temperature and humidity data.

Source Code

```
// Lab 8: Interfacing Temperature Humidity Sensor and Visualize data in Cloud
Server with NodeMCU ESP8266
// This example uses ThingSpeak. You can adapt it for other platforms like
Blynk, Adafruit IO, etc.

#include <ESP8266WiFi.h>
#include <ThingSpeak.h> // Include the ThingSpeak library
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Wi-Fi credentials
const char* ssid = "YOUR_WIFI_SSID";       // Your Wi-Fi SSID
const char* password = "YOUR_WIFI_PASSWORD"; // Your Wi-Fi Password

// ThingSpeak Channel details
unsigned long myChannelNumber = YOUR_CHANNEL_NUMBER; // Your ThingSpeak Channel
ID
```

```
const char* myWriteAPIKey = "YOUR_WRITE_API_KEY";   // Your ThingSpeak Write API
Key

// DHT Sensor details
#define DHTPIN D2     // Digital pin D2 (GPIO4) connected to the DHT sensor's
Data pin
#define DHTTYPE DHT11 // Or DHT22

DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password); // Connect to Wi-Fi

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  ThingSpeak.begin(client); // Initialize ThingSpeak
  dht.begin();              // Start DHT sensor
}

void loop() {
  // Read temperature and humidity
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if any reads failed
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Set the fields with temperature and humidity
  ThingSpeak.setField(1, t); // Field 1 for Temperature
  ThingSpeak.setField(2, h); // Field 2 for Humidity

  // Write to ThingSpeak
  int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
  if (x == 200) {
    Serial.println("Channel update successful.");
  } else {
    Serial.println("Problem updating channel. HTTP error code " + String(x));
  }

  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C, Humidity: ");
  Serial.print(h);
  Serial.println(" %");

  delay(20000); // Update ThingSpeak every 20 seconds (ThingSpeak free tier
limit is 15 seconds)
}
```

Input

- NodeMCU ESP8266 with DHT sensor connected.
- Active internet connection for the NodeMCU.
- ThingSpeak (or chosen cloud platform) account and channel configured.

Expected Output

1. The NodeMCU successfully connects to your Wi-Fi network.
2. Temperature and humidity readings are periodically sent to the ThingSpeak cloud.
3. The ThingSpeak channel's private view (or public view if enabled) will display real-time graphs visualizing the temperature and humidity data.
4. Serial Monitor will show "Channel update successful." messages along with temperature and humidity readings.

Lab 9: Create a Smart Switch to Control Light from Internet

Title

Create a Smart Switch to Control Light from Internet

Aim

To build a smart switch using NodeMCU ESP8266 and a relay, allowing remote control of a light bulb (or any AC appliance) over the internet using a cloud service (e.g., Blynk, Adafruit IO, or a custom web interface).

Procedure

1. **Gather Components:** NodeMCU ESP8266, 5V Relay Module, Light bulb with holder and plug, AC power cord, Breadboard, Jumper Wires. **(CAUTION: Working with AC mains voltage can be dangerous. Ensure all connections are secure and take necessary safety precautions. If unsure, seek assistance from a qualified person.)**
2. **Circuit Connection:** Connect the relay to NodeMCU and the light bulb as described in Lab 6.
3. **Cloud Platform Setup (Example: Blynk):**
    - Download the Blynk app on your smartphone.
    - Create a new project. You will receive an "Auth Token" via email.
    - Add a "Button" widget to your project. Configure it to control a Virtual Pin (e.g., V1) and set its mode to "SWITCH".
4. **Install Libraries:**
    - Install the "Blynk" library via `Sketch > Include Library > Manage Libraries...`.
5. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
6. **Write Code:** Copy the provided source code, replacing placeholders with your Wi-Fi credentials and Blynk Auth Token.
7. **Upload Code:** Upload the code to your NodeMCU.
8. **Open Serial Monitor:** Open the Serial Monitor to observe connection status.
9. **Control from Blynk App:** Use the button widget in the Blynk app to toggle the light ON/OFF.

Source Code

```
// Lab 9: Create a Smart Switch to Control Light from Internet (using Blynk)

#define BLYNK_PRINT Serial // Enables Serial Monitor for Blynk debugging
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h> // For ESP8266

// Your Blynk Auth Token
char auth[] = "YOUR_BLYNK_AUTH_TOKEN";

// Your WiFi credentials
char ssid[] = "YOUR_WIFI_SSID";
char pass[] = "YOUR_WIFI_PASSWORD";

const int relayPin = D3; // Digital pin D3 (GPIO0) connected to the Relay
module's IN pin
```

```
// This function will be called every time the Virtual Pin V1 state changes
BLYNK_WRITE(V1) {
  int pinValue = param.asInt(); // Get value (0 or 1)
  if (pinValue == 1) {
    digitalWrite(relayPin, LOW); // Turn light ON (adjust LOW/HIGH based on your
relay)
    Serial.println("Light ON via Blynk");
  } else {
    digitalWrite(relayPin, HIGH); // Turn light OFF
    Serial.println("Light OFF via Blynk");
  }
}

void setup() {
  Serial.begin(115200); // Initialize serial communication
  pinMode(relayPin, OUTPUT); // Set the relay control pin as an output
  digitalWrite(relayPin, HIGH); // Ensure light is off initially

  Blynk.begin(auth, ssid, pass); // Connect to Blynk server
  Serial.println("Connecting to Blynk...");
}

void loop() {
  Blynk.run(); // Keep Blynk connected and process incoming commands
}
```

Input

- NodeMCU ESP8266 connected to a relay and light bulb.
- Active internet connection for the NodeMCU.
- Blynk app on a smartphone with a configured project and button widget.
- User interaction with the button in the Blynk app.

Expected Output

1. The NodeMCU successfully connects to your Wi-Fi network and the Blynk server.
2. When the button in the Blynk app is toggled, the relay will switch, turning the connected light bulb ON or OFF.
3. Serial Monitor will display messages indicating the light status change.

Lab 10: Create a Voice Controlled Switch Using Arduino IoT Cloud

Title

Create a Voice Controlled Switch Using Arduino IoT Cloud

Aim

To create a voice-controlled switch using the Arduino IoT Cloud, allowing a user to control an LED (or a relay connected to a light) using voice commands through a smart assistant (like Google Assistant or Amazon Alexa) integrated with Arduino IoT Cloud.

Procedure

1. **Gather Components:** NodeMCU ESP8266 (or Arduino Nano 33 IoT), LED, 220 Ohm Resistor, Breadboard, Jumper Wires. (If using NodeMCU, ensure it's compatible with Arduino IoT Cloud or use an alternative like ESP32 with a compatible library).
2. **Arduino IoT Cloud Setup:**
    - Go to `cloud.arduino.cc` and create an account.
    - Create a new "Thing".
    - Configure your ESP8266/Arduino board type.
    - Create a "Property" (e.g., `lightStatus`, type `boolean`, read/write). This property will represent the state of your switch.
    - Link your Thing to your Wi-Fi network.
    - Generate the sketch.
3. **Circuit Connection:**
    - Connect the long leg (anode) of the LED to a digital GPIO pin on the NodeMCU (e.g., D4/GPIO2) via a 220 Ohm resistor.
    - Connect the short leg (cathode) of the LED to GND.
4. **Integrate Voice Assistant:**
    - In Arduino IoT Cloud, go to "Integrations".
    - Connect your Google Assistant or Amazon Alexa account. Follow the on-screen instructions to link your Arduino IoT Cloud devices.
5. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
6. **Write Code:** Copy the generated code from Arduino IoT Cloud and add the LED control logic based on the `lightStatus` property.
7. **Upload Code:** Upload the code to your NodeMCU.
8. **Voice Control:** Use voice commands with your smart assistant, e.g., "Hey Google, turn on light status" or "Alexa, turn off light status".

Source Code

```
// Lab 10: Create a Voice Controlled Switch Using Arduino IoT Cloud
// This is a simplified example. The actual code is largely generated by Arduino
IoT Cloud.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h> // For WiFi connection

// Replace with your Wi-Fi credentials and Arduino IoT Cloud Device ID/Secret
Key
```

```
const char SSID[]     = "YOUR_WIFI_SSID";
const char PASS[]     = "YOUR_WIFI_PASSWORD";

// Define your IoT Cloud properties
// This variable will be synced with the cloud and controlled by voice commands
bool lightStatus;

const int ledPin = D4; // Digital pin D4 (GPIO2) connected to the LED

void onLightStatusChange(); // Forward declaration

void initProperties() {
  ArduinoIoTCloud.set  (lightStatus, ON_CHANGE, onLightStatusChange);
}

// Function to initialize IoT Cloud
void setupCloud() {
  // Initialize Arduino IoT Cloud
  ArduinoIoTCloud.begin(ArduinoIoTCloud.SECRET_DEVICE_ID,
ArduinoIoTCloud.SECRET_KEY, ConnectionHandler);
  setDebugMessageLevel(2); // Set debug level for more verbose logging
  ArduinoIoTCloud.printDebugMessage();
}

void setup() {
  Serial.begin(115200);
  delay(1500); // Allow time for serial monitor to open

  pinMode(ledPin, OUTPUT); // Set LED pin as output
  digitalWrite(ledPin, LOW); // Ensure LED is off initially

  initProperties();
  setupCloud();
}

void loop() {
  ArduinoIoTCloud.update(); // Keep the IoT Cloud connection alive and sync
properties
}

// Callback function when lightStatus property changes from the cloud/voice
assistant
void onLightStatusChange() {
  if (lightStatus) {
    digitalWrite(ledPin, HIGH); // Turn LED ON
    Serial.println("Light ON via Voice Command");
  } else {
    digitalWrite(ledPin, LOW); // Turn LED OFF
    Serial.println("Light OFF via Voice Command");
  }
}
```

Input

- NodeMCU ESP8266 with LED connected.
- Active internet connection for the NodeMCU.
- Arduino IoT Cloud account with a configured Thing and voice assistant integration.
- Voice commands given to a smart assistant (Google Assistant/Alexa).

Expected Output

1. The NodeMCU successfully connects to your Wi-Fi and the Arduino IoT Cloud.

2. When you give a voice command (e.g., "Turn on light status"), the `lightStatus` property in the cloud will change.
3. The `onLightStatusChange()` function on the NodeMCU will be triggered, and the connected LED will turn ON or OFF accordingly.
4. Serial Monitor will display messages confirming the action.

Lab 11: ESP8266 NodeMCU Data Logging to Firebase Realtime Database

Title

ESP8266 NodeMCU Data Logging to Firebase Realtime Database

Aim

To send sensor data (e.g., temperature and humidity) from a NodeMCU ESP8266 to a Firebase Realtime Database for persistent storage and remote access.

Procedure

1. **Gather Components:** NodeMCU ESP8266, DHT11/DHT22 Sensor, Breadboard, Jumper Wires.
2. **Circuit Connection:** Connect the DHT sensor to NodeMCU as in Lab 4.
3. **Firebase Project Setup:**
    - Go to `console.firebase.google.com` and create a new project.
    - In your project, navigate to "Build > Realtime Database".
    - Create a new Realtime Database instance.
    - Go to "Rules" and set them to allow read/write access for testing (e.g., `".read": "true", ".write": "true"`). **(NOTE: For production, implement proper authentication rules.)**
    - Go to "Project settings" (gear icon) > "Service accounts" > "Database secrets". Copy your "Database Secret" (Legacy API key).
4. **Install Libraries:**
    - Install "DHT sensor library" and "Adafruit Unified Sensor" (if not already done).
    - Install "Firebase ESP8266 Client" library by Mobizt via `Sketch > Include Library > Manage Libraries....`
5. **Open Arduino IDE:** Ensure the NodeMCU board and port are selected.
6. **Write Code:** Copy the provided source code, replacing placeholders with your Wi-Fi credentials, Firebase Host URL, and Firebase Auth (Database Secret).
7. **Upload Code:** Upload the code to your NodeMCU.
8. **Open Serial Monitor:** Open the Serial Monitor to observe connection status and data logging.
9. **Observe Firebase Database:** Go to your Firebase Realtime Database console to see the incoming temperature and humidity data.

Source Code

```
// Lab 11: ESP8266 NodeMCU Data Logging to Firebase Realtime Database

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h> // Firebase library for ESP8266
#include <Adafruit_Sensor.h>
#include <DHT.h>

// Wi-Fi credentials
#define WIFI_SSID "YOUR_WIFI_SSID"
#define WIFI_PASSWORD "YOUR_WIFI_PASSWORD"

// Firebase project credentials
```

```
#define FIREBASE_HOST "YOUR_FIREBASE_PROJECT_ID.firebaseio.com" // e.g., my-
project-12345.firebaseio.com
#define FIREBASE_AUTH "YOUR_FIREBASE_DATABASE_SECRET"          // Your Database
Secret (Legacy API key)

// DHT Sensor details
#define DHTPIN D2     // Digital pin D2 (GPIO4) connected to the DHT sensor's
Data pin
#define DHTTYPE DHT11 // Or DHT22

DHT dht(DHTPIN, DHTTYPE); // Initialize DHT sensor

FirebaseData firebaseData; // Firebase data object

void setup() {
  Serial.begin(115200);
  delay(100);

  Serial.print("Connecting to Wi-Fi");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); // Initialize Firebase
  Firebase.reconnectWiFi(true); // Reconnect Wi-Fi if connection is lost

  dht.begin(); // Start DHT sensor
}

void loop() {
  // Read temperature and humidity
  float h = dht.readHumidity();
  float t = dht.readTemperature();

  // Check if any reads failed
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Create JSON data to send
  String path = "/sensor_data"; // Path in your Firebase Realtime Database
  String timestamp = String(millis()); // Simple timestamp for unique entries

  if (Firebase.setFloat(firebaseData, path + "/temperature", t) &&
      Firebase.setFloat(firebaseData, path + "/humidity", h) &&
      Firebase.setString(firebaseData, path + "/last_update", timestamp)) {
    Serial.println("Data sent to Firebase successfully!");
    Serial.print("Temperature: ");
    Serial.print(t);
    Serial.print(" *C, Humidity: ");
    Serial.print(h);
    Serial.println(" %");
  } else {
    Serial.print("Failed to send data to Firebase: ");
    Serial.println(firebaseData.errorReason());
  }

  delay(10000); // Send data every 10 seconds
}
```

Input

- NodeMCU ESP8266 with DHT sensor connected.
- Active internet connection for the NodeMCU.
- Firebase project configured with Realtime Database and appropriate rules.

Expected Output

1. The NodeMCU successfully connects to your Wi-Fi network and Firebase.
2. Temperature and humidity readings will be periodically logged as new entries (or updated at a specific path) in your Firebase Realtime Database.
3. The Firebase console will show the incoming data.
4. Serial Monitor will display "Data sent to Firebase successfully!" messages along with temperature and humidity readings.

Lab 12: Install OS and Configure Raspberry Pi

Title

Install OS and Configure Raspberry Pi

Aim

To install a suitable operating system (e.g., Raspberry Pi OS Lite or Desktop) on a Raspberry Pi and perform initial configuration steps for headless access or desktop environment setup.

Procedure

1. **Gather Components:** Raspberry Pi board (e.g., Pi 3B+, Pi 4), MicroSD card (16GB or larger, Class 10 recommended), SD card reader, Power supply for Raspberry Pi, Ethernet cable (optional for headless setup), HDMI cable and monitor (for desktop setup), USB keyboard and mouse (for desktop setup).
2. **Download Raspberry Pi Imager:** Download the official Raspberry Pi Imager tool from `raspberrypi.com/software`.
3. **Choose OS:** Open Raspberry Pi Imager. Click "CHOOSE OS" and select "Raspberry Pi OS (32-bit)" or "Raspberry Pi OS Lite (32-bit)" based on your preference (Lite for headless, Desktop for GUI).
4. **Choose Storage:** Click "CHOOSE STORAGE" and select your MicroSD card.
5. **Configure Advanced Options (Optional but Recommended for Headless):**
    - Click the gear icon (advanced options).
    - Enable SSH.
    - Set username and password.
    - Configure Wi-Fi (SSID and password).
    - Set locale settings.
    - Click "SAVE".
6. **Write Image:** Click "WRITE" to flash the OS image to the MicroSD card. This process will erase all existing data on the SD card.
7. **Insert SD Card and Boot:** Once writing is complete, safely eject the MicroSD card, insert it into your Raspberry Pi, and connect the power supply.
8. **Initial Configuration (Headless):**
    - Wait a few minutes for the Pi to boot up.
    - Use a network scanner (e.g., `nmap` or a mobile app) to find the IP address of your Raspberry Pi on your network.
    - Connect via SSH from your computer: `ssh <your_username>@<raspberry_pi_ip_address>` (e.g., `ssh pi@192.168.1.100`).
    - Enter the password you set.
    - Run `sudo raspi-config` to perform further configurations (e.g., expand filesystem, change hostname, enable interfaces).
9. **Initial Configuration (Desktop):**
    - Connect HDMI cable to a monitor, USB keyboard, and mouse.
    - Power on the Raspberry Pi.
    - Follow the on-screen prompts for initial setup (locale, Wi-Fi, software updates).

Source Code

N/A (This lab involves OS installation and configuration, not programming code.)

Input

- Raspberry Pi hardware.
- MicroSD card.
- Computer with Raspberry Pi Imager.
- Network connection for downloading OS and for SSH access (if headless).

Expected Output

- The Raspberry Pi successfully boots up with the chosen operating system.
- For desktop versions, you will see the graphical desktop environment.
- For Lite versions, you will be able to log in via SSH to the command line interface.
- The Raspberry Pi will be configured with your chosen username, password, and network settings.

Lab 13: Integrating IR sensor with Raspberry Pi and control an LED using Python

Title

Integrating IR Sensor with Raspberry Pi and Control an LED using Python

Aim

To connect an IR proximity sensor and an LED to the Raspberry Pi's GPIO pins and write a Python script to detect objects using the IR sensor and control the LED based on the detection.

Procedure

1. **Gather Components:** Raspberry Pi, IR Proximity Sensor (3-pin module), LED, 220 Ohm Resistor, Breadboard, Jumper Wires.
2. **Circuit Connection:**
    - **IR Sensor:**
        - Connect the VCC pin of the IR sensor to a 3.3V pin on the Raspberry Pi (e.g., Pin 1).
        - Connect the GND pin of the IR sensor to a GND pin on the Raspberry Pi (e.g., Pin 6).
        - Connect the OUT (digital output) pin of the IR sensor to a GPIO pin on the Raspberry Pi (e.g., GPIO17 / Pin 11).
    - **LED:**
        - Connect the long leg (anode) of the LED to a GPIO pin on the Raspberry Pi (e.g., GPIO27 / Pin 13) via a 220 Ohm resistor.
        - Connect the short leg (cathode) of the LED to a GND pin on the Raspberry Pi (e.g., Pin 9).
3. **Access Raspberry Pi:** Connect to your Raspberry Pi via SSH or open a terminal on its desktop.
4. **Install RPi.GPIO (if not already present):**

```
5. sudo apt-get update
6. sudo apt-get install python3-rpi.gpio
```

7. **Write Python Code:** Create a new Python file (e.g., `ir_led_control.py`) using a text editor like `nano` or `vim`. Copy the provided source code into the file.
8. **Run Python Script:** Execute the script from the terminal:

```
9. python3 ir_led_control.py
```

10. **Observe:** Move an object in front of the IR sensor and observe the LED turning ON/OFF. Press `Ctrl+C` in the terminal to stop the script.

Source Code

```
# Lab 13: Integrating IR sensor with Raspberry Pi and control an LED using
python

import RPi.GPIO as GPIO # Import the RPi.GPIO library
import time             # Import the time library for delays
```

```
# Pin Definitions
IR_SENSOR_PIN = 17  # GPIO17 (Physical Pin 11) for IR sensor OUT
LED_PIN = 27        # GPIO27 (Physical Pin 13) for LED Anode

def setup():
    # Set the GPIO mode to BCM (Broadcom SOC channel numbers)
    GPIO.setmode(GPIO.BCM)
    # Suppress warnings for GPIO setup
    GPIO.setwarnings(False)

    # Set up IR sensor pin as input with a pull-up resistor
    # IR sensors typically output LOW when object detected, so we look for
falling edge
    GPIO.setup(IR_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    # Set up LED pin as output
    GPIO.setup(LED_PIN, GPIO.OUT)
    # Ensure LED is off initially
    GPIO.output(LED_PIN, GPIO.LOW)

    print("IR Sensor and LED control script started.")
    print("Move an object in front of the IR sensor to toggle the LED.")
    print("Press Ctrl+C to exit.")

def loop():
    try:
        while True:
            # Read the state of the IR sensor
            # Assuming IR sensor outputs LOW when object is detected
            if GPIO.input(IR_SENSOR_PIN) == GPIO.LOW:
                print("Object Detected! Turning LED ON.")
                GPIO.output(LED_PIN, GPIO.HIGH) # Turn LED ON
            else:
                print("No Object. Turning LED OFF.")
                GPIO.output(LED_PIN, GPIO.LOW)  # Turn LED OFF
            time.sleep(0.2) # Small delay to prevent rapid readings

    except KeyboardInterrupt:
        # Clean up GPIO settings when Ctrl+C is pressed
        print("\nExiting script. Cleaning up GPIO.")
        GPIO.cleanup()

if __name__ == "__main__":
    setup()
    loop()
```

Input

- Raspberry Pi powered on with IR sensor and LED connected.
- Placing an object (e.g., hand) in front of the IR sensor.
- Removing the object from in front of the IR sensor.

Expected Output

- The Python script will run in the terminal, printing "Object Detected! Turning LED ON." or "No Object. Turning LED OFF."
- The LED connected to the Raspberry Pi will turn ON when an object is detected by the IR sensor and turn OFF when no object is detected.

Lab 14: Raspberry Pi: Send an Email using Python

Title

Raspberry Pi: Send an Email using Python

Aim

To write a Python script on the Raspberry Pi that can send emails using the `smtplib` and `email` modules, typically for notifications or alerts.

Procedure

1. **Access Raspberry Pi:** Connect to your Raspberry Pi via SSH or open a terminal on its desktop.
2. **Enable Less Secure Apps (for Gmail):** If you are using a Gmail account, you might need to enable "Less secure app access" in your Google Account settings (or use App Passwords if 2-Factor Authentication is enabled). **(NOTE: Using App Passwords is more secure than enabling "Less secure app access".)**
3. **Write Python Code:** Create a new Python file (e.g., `send_email.py`) using a text editor. Copy the provided source code into the file.
4. **Modify Script:** Replace the placeholder values for `sender_email`, `sender_password`, `receiver_email`, `subject`, and `body` with your actual email details.
5. **Run Python Script:** Execute the script from the terminal:
6. `python3 send_email.py`

7. **Observe:** Check the inbox of the `receiver_email` to verify that the email was sent successfully.

Source Code

```
# Lab 14: Raspberry Pi: Send an Email using Python

import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

def send_email(sender_email, sender_password, receiver_email, subject, body):
    """
    Sends an email using the specified SMTP server and credentials.
    """
    try:
        # Create a multipart message and set headers
        message = MIMEMultipart()
        message["From"] = sender_email
        message["To"] = receiver_email
        message["Subject"] = subject
        message["Bcc"] = receiver_email  # Add Bcc for self-copy or multiple
recipients

        # Add body to email
        message.attach(MIMEText(body, "plain"))

        # Use Gmail's SMTP server (you can change this for other providers)
        smtp_server = "smtp.gmail.com"
```

```
        smtp_port = 587  # For TLS/STARTTLS

        # Create a secure SSL context
        # context = ssl.create_default_context() # Not needed for STARTTLS

        # Connect to the SMTP server
        print(f"Connecting to SMTP server: {smtp_server}:{smtp_port}...")
        with smtplib.SMTP(smtp_server, smtp_port) as server:
            server.ehlo()  # Can be omitted
            server.starttls()  # Upgrade connection to secure TLS
            server.ehlo()  # Can be omitted

            # Log in to the server
            print("Logging in to email account...")
            server.login(sender_email, sender_password)
            print("Login successful.")

            # Send email
            text = message.as_string()
            server.sendmail(sender_email, receiver_email, text)
            print("Email sent successfully!")

    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    # --- Configuration ---
    # Replace with your actual email details
    SENDER_EMAIL = "your_email@gmail.com"        # Your email address
    SENDER_PASSWORD = "your_email_password_or_app_password" # Your email
password or App Password
    RECEIVER_EMAIL = "recipient_email@example.com" # Recipient's email address

    EMAIL_SUBJECT = "Raspberry Pi Notification"
    EMAIL_BODY = "Hello from your Raspberry Pi! This is an automated email."

    # --- Call the function to send email ---
    send_email(SENDER_EMAIL, SENDER_PASSWORD, RECEIVER_EMAIL, EMAIL_SUBJECT,
EMAIL_BODY)
```

Input

- Raspberry Pi with internet access.
- Valid sender and receiver email addresses.
- Correct email password or app-specific password for the sender's account.

Expected Output

- The Python script will execute without errors.
- The recipient will receive an email with the specified subject and body.
- The terminal will print "Email sent successfully!"

Lab 15: Install Mosquitto MQTT Broker on Raspberry Pi

Title

Install Mosquitto MQTT Broker on Raspberry Pi

Aim

To install and configure the Mosquitto MQTT broker on a Raspberry Pi, turning it into a central message hub for IoT devices.

Procedure

1. **Access Raspberry Pi:** Connect to your Raspberry Pi via SSH or open a terminal on its desktop.
2. **Update Package List:** Ensure your Raspberry Pi's package list is up to date:

```
sudo apt update
```

4. **Install Mosquitto Broker:** Install the Mosquitto MQTT broker and client utilities:

```
sudo apt install mosquitto mosquitto-clients
```

6. **Enable and Start Service:** Ensure Mosquitto starts automatically on boot and is currently running:

```
sudo systemctl enable mosquitto
sudo systemctl start mosquitto
```

9. **Check Service Status:** Verify that the Mosquitto service is active and running:

```
systemctl status mosquitto
```

   (Press `q` to exit the status view.)

11. **Test MQTT Broker (Publish):** Open a new terminal window or SSH session on your Raspberry Pi. Use the `mosquitto_pub` client to publish a message:

```
mosquitto_pub -h localhost -t "test/topic" -m "Hello MQTT from Raspberry Pi!"
```

13. **Test MQTT Broker (Subscribe):** In another new terminal window or SSH session (or the original one if you stopped the previous command), use the `mosquitto_sub` client to subscribe to the topic:

```
mosquitto_sub -h localhost -t "test/topic"
```

    You should see the "Hello MQTT from Raspberry Pi!" message appear in the subscriber's terminal.

15. **Configure External Access (Optional):** By default, Mosquitto might only accept connections from `localhost`. To allow other devices on your network to connect, you might need to edit the Mosquitto configuration file:

```
sudo nano /etc/mosquitto/mosquitto.conf
```

Add or uncomment the line `listener 1883` and `allow_anonymous true` (for testing, disable for production) at the end of the file. Save and exit (`Ctrl+X`, `Y`, `Enter`). Then restart Mosquitto:

```
sudo systemctl restart mosquitto
```

Source Code

N/A (This lab involves command-line installation and configuration, not programming code.)

Input

- Raspberry Pi with internet access.
- Terminal access to the Raspberry Pi.

Expected Output

- Mosquitto MQTT broker and client utilities are successfully installed on the Raspberry Pi.
- The Mosquitto service is running and enabled to start on boot.
- You can successfully publish messages to and subscribe from topics on the local broker using `mosquitto_pub` and `mosquitto_sub`.
- (Optional) Other devices on the network can connect to the Raspberry Pi's IP address on port 1883 and exchange MQTT messages.