## Lab Manual

# Embedded system and IOT

## Topic 1: Getting Started with Arduino

- **Objective**: Learn what Arduino is and set up the basics.
- **Theory**: Arduino is a small computer for projects. Basic commands: pinMode(), digitalWrite().
- **Lab**: Install Arduino IDE. Connect an LED to Arduino Uno and write code to turn it on.
- **Hardware**: Arduino Uno, LED, resistor (220Ω), breadboard.
- **Exercise**: Turn the LED on and off manually using digitalWrite().
- **Outcome**: Understand Arduino setup and first circuit.

## Topic 2: Making an LED Blink

- **Objective**: Use delay() to make an LED blink.
- **Theory**: delay() pauses the program to create blinking effects.
- **Lab**: Modify Topic 1 code to make the LED blink (on 1s, off 1s).
- **Hardware**: Arduino Uno, LED, resistor, breadboard.
- **Exercise**: Change blink timing to fast (0.5s on, 0.5s off) and slow (2s on, 2s off).
- **Outcome**: Learn delay() and create a blinking LED.

## Topic 3: Controlling LED with a Button

- **Objective**: Use a button to control an LED.
- **Theory**: digitalRead() reads button state (on/off). if statements make decisions.
- **Lab**: Connect a push button to Arduino. Turn LED on when button is pressed.
- **Hardware**: Arduino Uno, push button, LED, resistor.
- **Exercise**: Make LED blink when button is held.
- **Outcome**: Understand input reading and basic conditions.

## Topic 4: Creating LED Blink Sequences

- **Objective**: Program multiple LEDs with blink patterns.
- **Theory**: Loops (for) and variables (int) create sequences.
- **Lab**: Connect two LEDs to Arduino. Code a sequence (e.g., LED1 blinks twice, LED2 blinks once).
- **Hardware**: Arduino Uno, two LEDs, resistors, breadboard.
- **Exercise**: Create a custom sequence (e.g., alternating fast/slow blinks).
- **Outcome**: Program custom LED patterns using loops.

## Topic 5: Temperature Sensor Basics

- **Objective**: Measure temperature with a sensor.
- **Theory**: DHT11 sensor measures temperature and humidity.
- **Lab**: Connect DHT11 to Arduino. Show temperature on Serial Monitor.
- **Hardware**: Arduino Uno, DHT11 sensor, resistor.
- **Exercise**: Show “Hot!” if temperature is above 25°C.
- **Outcome**: Work with sensors and simple conditions.

## Topic 6: Introduction to ESP8266

- **Objective**: Use ESP8266 for Wi-Fi projects.
- **Theory**: ESP8266 is like Arduino but with Wi-Fi.
- **Lab**: Set up ESP8266 in Arduino IDE. Blink an LED.
- **Hardware**: ESP8266 (e.g., NodeMCU), LED, resistor.
- **Exercise**: Change blink pattern (e.g., three short blinks).
- **Outcome**: Familiar with ESP8266 and its setup.

## Topic 7: Wi-Fi Connection with ESP8266

- **Objective**: Connect to the internet.
- **Theory**: Wi-Fi lets devices talk to the internet.
- **Lab**: Program ESP8266 to connect to Wi-Fi and show status on Serial Monitor.
- **Hardware**: ESP8266.
- **Exercise**: Connect to a different Wi-Fi network.
- **Outcome**: Basic Wi-Fi setup.

## Topic 8: Sending Sensor Data to the Cloud

- **Objective**: Send temperature data to the internet.
- **Theory**: IoT devices send data to apps.
- **Lab**: Use ESP8266 and DHT11 to send temperature to Blynk app.
- **Hardware**: ESP8266, DHT11.
- **Exercise**: Show humidity in Blynk too.
- **Outcome**: First IoT project with cloud app.

## Topic 9: Phone-Controlled LED

- **Objective**: Use a phone to control an LED.
- **Theory**: Apps like Blynk send commands to devices.
- **Lab**: Use Blynk to turn an LED on/off from your phone.
- **Hardware**: ESP8266, LED.
- **Exercise**: Add a Blynk button to control LED brightness.
- **Outcome**: Control devices remotely.

## Topic 10: Using an OLED Display

- **Objective**: Show data on a small screen.
- **Theory**: OLED displays show text or numbers.
- **Lab**: Connect an OLED to Arduino. Show temperature from DHT11.
- **Hardware**: Arduino Uno, OLED (128x64), DHT11.
- **Exercise**: Display your name or a smiley face.
- **Outcome**: Work with displays for cool outputs.

## Topic 11: Building a Simple IoT Dashboard

- **Objective**: Make a web page to show sensor data.
- **Theory**: ESP8266 can host a simple website.
- **Lab**: Program ESP8266 to show DHT11 data on a web page.
- **Hardware**: ESP8266, DHT11.
- **Exercise**: Add a title to the web page.
- **Outcome**: Create a basic IoT dashboard.

## Topic 12: Controlling Devices with a Relay

- **Objective**: Use a relay to control a device.
- **Theory**: Relays act like switches for bigger devices.
- **Lab**: Connect a relay to Arduino. Turn a small light on/off.
- **Hardware**: Arduino Uno, relay module, LED strip or small bulb.
- **Exercise**: Use a button to control the relay.
- **Outcome**: Control real-world devices.

## Topic 13: Motion Detection

- **Objective**: Detect motion with a sensor.
- **Theory**: PIR sensors detect movement (e.g., for alarms).
- **Lab**: Connect a PIR sensor to Arduino. Light an LED when motion is detected.
- **Hardware**: Arduino Uno, PIR sensor, LED.
- **Exercise**: Print “Motion!” to Serial Monitor.
- **Outcome**: Understand motion sensing.

## Topic 14: Power-Saving Techniques

- **Objective**: Make devices use less power.
- **Theory**: Sleep mode saves battery life.
- **Lab**: Program Arduino to sleep and wake with a button.
- **Hardware**: Arduino Uno, push button.
- **Exercise**: Make LED blink only when awake.
- **Outcome**: Basic power-saving skills.

## Topic 15: Device-to-Device Communication

- **Objective**: Make two ESP8266s share data.
- **Theory**: Devices can send messages over Wi-Fi.
- **Lab**: Use two ESP8266s—one sends temperature, the other shows it.
- **Hardware**: Two ESP8266s, DHT11.
- **Exercise**: Send a “Hi” message between them.
- **Outcome**: Simple device communication.

## Topic 16: Smart Light System Project

- **Objective**: Build a smart light system.
- **Lab**: Use ESP8266, relay, and Blynk to control a light from your phone.
- **Hardware**: ESP8266, relay, LED strip.
- **Exercise**: Add a local button to control the light.
- **Outcome**: Fun IoT project with phone control.

## Topic 17: Mini Weather Station Project

- **Objective**: Create a mini weather station.
- **Lab**: Use Arduino, DHT11, and OLED to show temperature and humidity.
- **Hardware**: Arduino Uno, DHT11, OLED.
- **Exercise**: Show “Humid!” if humidity is high.
- **Outcome**: Working weather station.

## Topic 18: Motion Alarm Project

- **Objective**: Build a motion-detecting alarm.
- **Lab**: Use Arduino, PIR sensor, and buzzer to sound when motion is detected.
- **Hardware**: Arduino Uno, PIR sensor, buzzer.
- **Exercise**: Send motion alert to Blynk via ESP8266.
- **Outcome**: Functional alarm system.

## Topic 19: Debugging IoT Projects

- **Objective**: Fix common project issues.
- **Theory**: Debugging means finding errors (e.g., LED not lighting).
- **Lab**: Fix a broken project (e.g., wrong wiring, bad code).
- **Hardware**: Arduino Uno, ESP8266, any sensor.
- **Exercise**: Use Serial Monitor to check errors.
- **Outcome**: Basic troubleshooting skills.

## Topic 20: Project Presentation

- **Objective**: Present your project and learn from others.
- **Lab**: Build a simple IoT project (e.g., smart fan, door sensor) with instructor help. Present it to the class.
- **Hardware**: Arduino Uno, ESP8266, any sensors/actuators.
- **Exercise**: Write a short project description.
- **Outcome**: Complete project and presentation experience.

# IoT Student Handout

Welcome! This guide will walk you through the exciting world of electronics and the Internet of Things (IoT). Each topic builds on the last, taking you from blinking a single light to building your own smart devices. Follow the labs, try the exercises, and have fun creating!

### Topic 1: Getting Started with Arduino

- **Objective:** Learn what Arduino is and set up the basics.
- **Theory:** The Arduino is a small, programmable computer (called a microcontroller) that can be used to build and control electronic projects. The basic commands `pinMode()` and `digitalWrite()` are used to configure and control the electronic pins on the board.
- **Lab:** Install the Arduino IDE. Connect an LED to your Arduino Uno and write the code to turn it on.
- **Hardware:**
    1. Arduino Uno
    2. USB Cable
    3. Breadboard
    4. 1 x LED (any color)
    5. 1 x 220Ω Resistor
    6. Jumper Wires
- **Setup & Code:**
    1. Connect the **long leg** of the LED (anode) to a jumper wire.
    2. Connect the **short leg** of the LED (cathode) to one end of the resistor.
    3. Connect the other end of the resistor to a jumper wire that goes to the Arduino's **GND** pin.
    4. Connect the wire from the long leg of the LED to **Pin 13** on the Arduino.

```
// This code runs once when the Arduino starts up

void setup() {

  // Set Pin 13 as an output pin

  pinMode(13, OUTPUT);

}

// This code runs over and over again
```

```
void loop() {

  // Turn the LED on by sending 5V to Pin 13

  digitalWrite(13, HIGH);

}
```

- **Exercise:** Modify the code to turn the LED off. Hint: Use `digitalWrite(13, LOW);`.
- **Outcome:** You will understand the basic Arduino setup and have built your first circuit.

## Topic 2: Making an LED Blink

- **Objective:** Use the `delay()` function to make an LED blink automatically.
- **Theory:** The `delay()` function pauses the program for a specified number of milliseconds. By turning an LED on, delaying, turning it off, and delaying again, you create a blinking effect.
- **Lab:** Modify the code from Topic 1 to make the LED blink (1 second on, 1 second off).
- **Hardware:** Same as Topic 1.

**Code:**

```
void setup() {

  pinMode(13, OUTPUT);

}



void loop() {

  digitalWrite(13, HIGH);   // Turn the LED on

  delay(1000);              // Wait for 1000 milliseconds (1 second)

  digitalWrite(13, LOW);    // Turn the LED off

  delay(1000);              // Wait for 1 second

}
```

- 
- **Exercise:** Change the blink timing to be fast (e.g., 200ms on, 200ms off) and then slow (e.g., 3s on, 3s off).
- **Outcome:** You will learn how to control timing in your code to create simple animations.

## Topic 3: Controlling an LED with a Button

- **Objective:** Use a push button to control an LED.
- **Theory:** A push button is a type of switch. We can read its state (pressed or not pressed) using the `digitalRead()` function. An `if` statement allows the program to make decisions based on the button's state.
- **Lab:** Connect a push button to the Arduino. Write code to turn the LED on only when the button is pressed.
- **Hardware:**
    1. Arduino Uno & Breadboard
    2. 1 x LED & 1 x 220Ω Resistor
    3. 1 x Push Button
    4. 1 x 10kΩ Resistor (for pulldown)
    5. Jumper Wires
- **Setup & Code:**
    1. Connect the LED to Pin 13 and GND as before.
    2. Connect one leg of the push button to Arduino **Pin 2**.
    3. Connect the other leg of the same side to the **5V** pin on the Arduino.
    4. Connect a 10kΩ resistor from Pin 2 to GND. This is a "pulldown" resistor that keeps the pin `LOW` when the button isn't pressed.

```
const int buttonPin = 2;

const int ledPin = 13;

int buttonState = 0;



void setup() {

  pinMode(ledPin, OUTPUT);

  pinMode(buttonPin, INPUT); // Set the button pin as an input

}



void loop() {

  // Read the state of the button

  buttonState = digitalRead(buttonPin);



  // Check if the button is pressed

  if (buttonState == HIGH) {
```

```
    digitalWrite(ledPin, HIGH); // Turn LED on
  } else {
    digitalWrite(ledPin, LOW);  // Turn LED off
  }
}
```

- 
- **Exercise:** Modify the code to make the LED blink only when the button is held down.
- **Outcome:** You will understand how to read inputs and use conditional logic.

## Topic 4: Creating LED Blink Sequences

- **Objective:** Program multiple LEDs to blink in a specific pattern.
- **Theory:** Loops (like the `for` loop) allow you to repeat a block of code a set number of times. This is perfect for creating sequences and patterns without writing the same code over and over.
- **Lab:** Connect two LEDs to the Arduino. Program a sequence where the first LED blinks twice, then the second LED blinks once.
- **Hardware:**
    1. Arduino Uno & Breadboard
    2. 2 x LEDs & 2 x 220Ω Resistors
    3. Jumper Wires
- **Setup & Code:**
    1. Connect the first LED to **Pin 13**.
    2. Connect the second LED to **Pin 12**.

```
const int led1 = 13;
const int led2 = 12;


void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}


void loop() {
  // Blink LED 1 twice
```

```
  for (int i = 0; i < 2; i++) {
    digitalWrite(led1, HIGH);
    delay(250);
    digitalWrite(led1, LOW);
    delay(250);
  }

  // Blink LED 2 once
  digitalWrite(led2, HIGH);
  delay(500);
  digitalWrite(led2, LOW);
  delay(500);
}
```

- 
- **Exercise:** Create your own custom sequence. For example, make the LEDs alternate blinking quickly.
- **Outcome:** You will be able to use loops to create more complex and interesting outputs.

## Topic 5: Temperature Sensor Basics

- **Objective:** Measure temperature using a sensor.
- **Theory:** The DHT11 is a digital sensor that can measure both temperature and humidity. To use it, you need to install a special "library" in the Arduino IDE, which contains the code needed to communicate with the sensor.
- **Lab:** Connect a DHT11 sensor to the Arduino. Display the temperature reading in the Serial Monitor.
- **Hardware:**
    1. Arduino Uno & Breadboard
    2. DHT11 Temperature and Humidity Sensor
    3. 1 x 10kΩ Resistor
    4. Jumper Wires
- **Setup & Code:**
    1. In the Arduino IDE, go to `Tools > Manage Libraries...` and install the "DHT sensor library" by Adafruit.
    2. Connect the DHT11's **VCC** pin to **5V**, **GND** to **GND**, and the **Data** pin to **Pin 2**.
    3. Place a 10kΩ resistor between the **Data** pin and the **5V** pin.

```
#include <dht11.h>
#define DHT11 PIN 4
dht11 DHT11;
void  setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.println();
  int chk = DHT11.read(DHT11PIN);

  Serial.print("Humidity (%): ");
  Serial.println((float)DHT11.humidity, 2);

  Serial.print("Temperature  (C): ");
  Serial.println((float)DHT11.temperature, 2);

  delay(2000);

}
```

- 

- **Exercise:** Modify the code to print "It's hot!" to the Serial Monitor if the temperature is above 25°C.
- **Outcome:** You will know how to work with sensors and display their data.

## Topic 6: Introduction to ESP8266

- **Objective:** Learn to use the ESP8266 microcontroller for Wi-Fi projects.
- **Theory:** The ESP8266 is a low-cost microcontroller with built-in Wi-Fi capabilities, making it ideal for IoT projects. It can be programmed using the same Arduino IDE.
- **Lab:** Set up the ESP8266 board in the Arduino IDE and make its built-in LED blink.
- **Hardware:**
    1. ESP8266 Board (e.g., NodeMCU or Wemos D1 Mini)
    2. Micro-USB Cable
- **Setup & Code:**
    1. In the Arduino IDE, go to `File > Preferences`.
    2. In "Additional Boards Manager URLs," add: `http://arduino.esp8266.com/stable/package_esp8266com_index.json`
    3. Go to `Tools > Board > Boards Manager...`, search for "esp8266" and install it.
    4. Select your ESP8266 board from the `Tools > Board` menu (e.g., "NodeMCU 1.0").

```
// The built-in LED on most ESP8266 boards is on Pin 2.

// LED_BUILTIN is a constant that knows the correct pin.

void setup() {

  pinMode(LED_BUILTIN, OUTPUT);

}



void loop() {

  digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note: LOW is ON for the built-in LED)

  delay(1000);

  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off

  delay(1000);

}
```

-

- **Exercise:** Change the blink pattern to three short blinks followed by a pause.
- **Outcome:** You will be familiar with the ESP8266 and how to program it.

## Topic 7: Wi-Fi Connection with ESP8266

- **Objective:** Connect your ESP8266 to the internet.
- **Theory:** The ESP8266's main feature is Wi-Fi. We can use the `ESP8266WiFi` library to scan for and connect to wireless networks.
- **Lab:** Program the ESP8266 to connect to your Wi-Fi network and print its connection status to the Serial Monitor.
- **Hardware:**
    - ESP8266 Board
    - Micro-USB Cable

**Code:**

```
#include <ESP8266WiFi.h>


const char* ssid = "YOUR_WIFI_NAME";

const char* password = "YOUR_WIFI_PASSWORD";


void setup() {

  Serial.begin(115200);

  delay(10);

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);


  WiFi.begin(ssid, password);


  while (WiFi.status() != WL_CONNECTED) {

    delay(500);

    Serial.print(".");

  }
```

```
  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

}


void loop() {

  // Nothing to do here

}
```

- 
- **Exercise:** Try connecting to a different Wi-Fi network (like a phone hotspot) by changing the `ssid` and `password`.
- **Outcome:** You will have the basic skill needed for any IoT project: connecting to the internet.

## Topic 8: Sending Sensor Data to the Cloud

- **Objective:** Send temperature data from a sensor to an online app.
- **Theory:** IoT devices send sensor data to cloud platforms (like Blynk) where it can be stored, viewed, and analyzed. This allows you to monitor your devices from anywhere in the world.
- **Lab:** Use an ESP8266 and a DHT11 sensor to send temperature data to the Blynk app.
- **Hardware:**
    1. ESP8266 Board
    2. DHT11 Sensor
    3. Jumper Wires
- **Setup & Code:**
    1. Download the **Blynk IoT** app on your phone and create an account.
    2. Create a new project and get your **Blynk Auth Token**.
    3. In the Arduino IDE, install the "Blynk" library.
    4. Connect the DHT11 to the ESP8266 (e.g., Data pin to **D4**).

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>
```

```
#include <DHT.h>

char auth[] = "YOUR_BLYNK_AUTH_TOKEN";
char ssid[] = "YOUR_WIFI_NAME";
char pass[] = "YOUR_WIFI_PASSWORD";

#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
BlynkTimer timer;

void sendSensor() {
  float t = dht.readTemperature();
  if (isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Blynk.virtualWrite(V5, t); // Send temperature to Virtual Pin 5
}

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);
  dht.begin();
  // Setup a function to be called every second
  timer.setInterval(1000L, sendSensor);
```

```
}


void loop() {

  Blynk.run();

  timer.run();

}
```

- 
- **Exercise:** Modify the code to also send the humidity reading to a different virtual pin in the Blynk app.
- **Outcome:** You will have built your first end-to-end IoT project.

## Topic 9: Phone-Controlled LED

- **Objective:** Use your phone to control an LED connected to your ESP8266.
- **Theory:** Cloud apps like Blynk can also send commands *to* your device. You can add a button in the app that, when pressed, tells your ESP8266 to turn an LED on or off.
- **Lab:** Use a Blynk button widget to turn an LED on and off from your phone.
- **Hardware:**
    1. ESP8266 Board
    2. 1 x LED & 1 x 220Ω Resistor
    3. Breadboard & Jumper Wires
- **Setup & Code:**
    1. In your Blynk app, add a "Button" widget and set its output to a Virtual Pin (e.g., **V1**).
    2. Connect an LED to a digital pin on your ESP8266 (e.g., **D5**).

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>


char auth[] = "YOUR_BLYNK_AUTH_TOKEN";

char ssid[] = "YOUR_WIFI_NAME";

char pass[] = "YOUR_WIFI_PASSWORD";

const int ledPin = D5;
```

```
// This function is called every time the button widget in Blynk changes value
BLYNK_WRITE(V1) {
  int pinValue = param.asInt(); // Get value from the app (0 or 1)
  digitalWrite(ledPin, pinValue);
}

void setup() {
  Serial.begin(115200);
  pinMode(ledPin, OUTPUT);
  Blynk.begin(auth, ssid, pass);
}

void loop() {
  Blynk.run();
}
```

- 
- **Exercise:** Add a "Slider" widget in Blynk to control the brightness of the LED (requires using `analogWrite()` and a PWM-capable pin).
- **Outcome:** You will be able to control physical devices remotely from your phone.

## Topic 10: Using an OLED Display

- **Objective:** Show data on a small screen instead of the Serial Monitor.
- **Theory:** OLED (Organic Light Emitting Diode) displays are small, bright screens that are great for showing text, numbers, and simple graphics. They communicate with the Arduino using a protocol called I2C.
- **Lab:** Connect an OLED display to an Arduino and show the temperature from a DHT11 sensor.
- **Hardware:**
    1. Arduino Uno
    2. OLED Display (128x64, I2C)
    3. DHT11 Sensor
    4. Breadboard & Jumper Wires
- **Setup & Code:**
    1. Install the "Adafruit SSD1306" and "Adafruit GFX" libraries.

2. Connect the OLED: **VCC** to **5V**, **GND** to **GND**, **SDA** to Arduino pin **A4**, and **SCL** to Arduino pin **A5**.
3. Connect the DHT11 to Pin 2 as before.

```
#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include "DHT.h"


#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);


#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);


void setup() {

  dht.begin();

  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x64

    // Don't proceed, display not found

    for(;;);

  }

  display.display();

  delay(2000);

}


void loop() {
```

```
  float t = dht.readTemperature();

  display.clearDisplay();

  display.setTextSize(1);

  display.setTextColor(SSD1306_WHITE);

  display.setCursor(0,0);

  display.print("Temperature: ");

  display.setTextSize(2);

  display.setCursor(0,10);

  display.print(t);

  display.print(" C");

  display.display();

  delay(2000);

}
```

- 
- **Exercise:** Modify the code to display your name or draw a simple smiley face on the screen.
- **Outcome:** You will be able to use displays to create a user interface for your projects.

## Topic 11: Building a Simple IoT Dashboard

- **Objective:** Create a web page to show sensor data.
- **Theory:** An ESP8266 can act as a small web server, hosting a simple web page that you can access from any device on the same Wi-Fi network.
- **Lab:** Program an ESP8266 to host a web page that shows the current temperature from a DHT11 sensor.
- **Hardware:**
    - ESP8266 Board
    - DHT11 Sensor
    - Jumper Wires

**Setup & Code:**

```
#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include "DHT.h"
```

```
const char* ssid = "YOUR_WIFI_NAME";
const char* password = "YOUR_WIFI_PASSWORD";
ESP8266WebServer server(80);

#define DHTPIN D4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void handleRoot() {
  float t = dht.readTemperature();
  String html = "<!DOCTYPE html><html><head><title>ESP8266 Temp</title>";
  html += "<meta http-equiv='refresh' content='5'></head>";
  html += "<body><h1>Temperature Sensor</h1>";
  html += "<h2>Temperature: " + String(t) + " C</h2></body></html>";
  server.send(200, "text/html", html);
}

void setup() {
  dht.begin();
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) { delay(500); }
  server.on("/", handleRoot);
  server.begin();
}
```

```
void loop() {

  server.handleClient();

}
```

- 
- **Exercise:** Add a title to the web page using the `<title>` HTML tag.
- **Outcome:** You will be able to create a basic web-based dashboard for your IoT devices.

## Topic 12: Controlling Devices with a Relay

- **Objective:** Use a relay to control a high-power device.
- **Theory:** A relay is an electronically operated switch. It allows a low-power signal from an Arduino to control a much higher power circuit, like a lamp or a fan.
- **Lab:** Connect a relay module to an Arduino and use it to turn a small light (like an LED strip) on and off.
- **Hardware:**
    1. Arduino Uno
    2. Relay Module (5V)
    3. LED Strip or small 5V/12V bulb with power supply
    4. Jumper Wires
- **Setup & Code:**
    1. **CAUTION:** Be careful when working with voltages higher than 5V.
    2. Connect the relay module's **VCC** to Arduino **5V**, **GND** to **GND**, and **IN** to **Pin 7**.
    3. Connect the external light's power supply to the relay's **COM** and **NO** (Normally Open) terminals.

```
const int relayPin = 7;



void setup() {

  pinMode(relayPin, OUTPUT);

}



void loop() {

  digitalWrite(relayPin, HIGH); // Turn the relay ON

  delay(2000);

  digitalWrite(relayPin, LOW);  // Turn the relay OFF
```

```
  delay(2000);
}
```

- 
- **Exercise:** Connect a push button to the Arduino and use it to manually control the relay.
- **Outcome:** You will learn how to safely control real-world AC/DC devices.

## Topic 13: Motion Detection

- **Objective:** Detect motion with a PIR sensor.
- **Theory:** A PIR (Passive Infrared) sensor detects the infrared energy emitted by living things. It's commonly used in automatic lights and security alarms.
- **Lab:** Connect a PIR sensor to an Arduino. When motion is detected, light up an LED.
- **Hardware:**
    1. Arduino Uno
    2. PIR Motion Sensor
    3. 1 x LED
    4. Jumper Wires
- **Setup & Code:**
    1. Connect the PIR sensor's **VCC** to **5V**, **GND** to **GND**, and **OUT** to **Pin 2**.
    2. Connect an LED to **Pin 13**.

```
const int pirPin = 2;

const int ledPin = 13;

int pirState = LOW;



void setup() {

  pinMode(ledPin, OUTPUT);

  pinMode(pirPin, INPUT);

  Serial.begin(9600);

}



void loop() {

  pirState = digitalRead(pirPin);

  if (pirState == HIGH) {
```

```
    digitalWrite(ledPin, HIGH);

    Serial.println("Motion detected!");

  } else {

    digitalWrite(ledPin, LOW);

  }

  delay(100);

}
```

- 
- **Exercise:** Instead of just turning on an LED, print "Motion!" to the Serial Monitor when movement is detected.
- **Outcome:** You will understand how motion sensing works and how to use it in projects.

## Topic 14: Power-Saving Techniques

- **Objective:** Learn how to make battery-powered devices last longer.
- **Theory:** "Sleep mode" allows a microcontroller to turn off most of its components, consuming very little power. It can be "woken up" by an external event, like a button press or a timer.
- **Lab:** Program an ESP8266 to go into a "deep sleep" and wake up when a button is pressed.
- **Hardware:**
    1. ESP8266 Board
    2. Push Button
    3. Jumper Wires
- **Setup & Code:**
    1. To enable waking from deep sleep, you must connect the ESP8266's **RST** pin to its **D0** (GPIO16) pin.

```
// ESP8266 Deep Sleep Example

void setup() {

  Serial.begin(115200);

  Serial.println("I'm awake! Going to sleep for 10 seconds.");

  // Deep sleep for 10 seconds, then wake up automatically

  ESP.deepSleep(10e6); // 10e6 microseconds = 10 seconds

}
```

```
void loop() {

  // This part of the code will not be reached

}
```

- **Exercise:** Make an LED blink five times when the device wakes up, before it goes back to sleep.
- **Outcome:** You will learn a fundamental skill for creating battery-powered IoT devices.

## Topic 15: Device-to-Device Communication

- **Objective:** Make two ESP8266 boards share data directly.
- **Theory:** Devices on the same Wi-Fi network can communicate without a central server using protocols like ESP-NOW. This is fast and efficient for local communication.
- **Lab:** Use two ESP8266s. One (the "sender") will read a DHT11 sensor. The other (the "receiver") will receive the temperature and print it to its Serial Monitor.
- **Hardware:**
    1. 2 x ESP8266 Boards
    2. 1 x DHT11 Sensor
- **Setup & Code:**
    1. Find the MAC Address of the receiver ESP8266.
    2. Upload the Sender code to one ESP8266 and the Receiver code to the other. *This lab is advanced. Code will be provided by the instructor.*
- **Exercise:** Modify the code to send a simple "Hi" message from one device to the other.
- **Outcome:** You will understand the basics of peer-to-peer device communication.

## Topic 16: Project - Smart Light System

- **Objective:** Combine your skills to build a complete smart light system.
- **Lab:** Use an ESP8266, a relay module, and the Blynk app to control a real light (e.g., an LED strip) from your phone.
- **Hardware:**
    - ESP8266 Board
    - Relay Module
    - LED Strip (or other small light) with its own power supply
- **Exercise:** Add a physical push button as a local override to control the light, in addition to the phone app.
- **Outcome:** You will have built a practical and fun IoT project.

## Topic 17: Project - Mini Weather Station

- **Objective:** Create a compact, standalone weather station.
- **Lab:** Use an Arduino, a DHT11 sensor, and an OLED display to show the current temperature and humidity on screen.

- **Hardware:**
    - Arduino Uno (or an ESP8266)
    - DHT11 Sensor
    - OLED Display
- **Exercise:** Add a warning message like "Very Humid!" to the display if the humidity goes above 70%.
- **Outcome:** You will have a functional, portable weather station.

## Topic 18: Project - Motion Alarm

- **Objective:** Build a motion-detecting security alarm.
- **Lab:** Use an Arduino, a PIR sensor, and a buzzer. When the PIR sensor detects motion, the buzzer should make a noise.
- **Hardware:**
    - Arduino Uno
    - PIR Sensor
    - Piezo Buzzer
- **Exercise:** Upgrade the project by using an ESP8266 instead of an Arduino. When motion is detected, send a notification to your phone using Blynk.
- **Outcome:** You will have built a functional security alarm system.

## Topic 19: Debugging IoT Projects

- **Objective:** Learn how to find and fix common problems in your projects.
- **Theory:** Debugging is the process of troubleshooting. Common issues include incorrect wiring, code errors (bugs), or problems with power or connections. The `Serial.print()` function is your most powerful debugging tool.
- **Lab:** You will be given a project with a deliberate error (e.g., a wire in the wrong place, a typo in the code). Your job is to find and fix it.
- **Debugging Checklist:**
    1. **Check the Wires:** Is everything connected to the right pin? Is VCC connected to 5V/3.3V and GND to GND?
    2. **Check the Code:** Did you select the correct Board and COM Port in the IDE? Are there any typos?
    3. **Use the Serial Monitor:** Add `Serial.println()` statements in your code to check the values of variables and see which parts of your code are running.
    4. **Check Power:** Is your board getting enough power?
- **Exercise:** Use the Serial Monitor to check the values coming from a sensor to see if they make sense.
- **Outcome:** You will develop basic troubleshooting skills essential for any maker.

## Topic 20: Final Project Presentation

- **Objective:** Present a project you've built and learn from your classmates.

- **Lab:** With instructor help, you will design and build your own simple IoT project. Ideas include a smart fan, an automated pet feeder, a door open/close sensor, or a plant watering system. You will then present your project to the class.
- **Exercise:** Write a short (1-2 paragraph) description of your project, explaining what it does and what components you used.
- **Outcome:** You will gain experience in project design, implementation, and presentation.