STUDENT MODULE 1: Introduction to Microcontrollers

Learning Objectives

By the end of this module, you will be able to:

1. Explain the role and core components of microcontrollers.
2. Identify popular microcontroller families and apply programming skills to build simple embedded projects (such as temperature display).
3. Recognize real-world applications in robotics, smart devices, automotive systems, IoT, and medical technology.

1. What is a Microcontroller?

You might be wondering what exactly a microcontroller is and why it's important for you to learn about it. Think of a microcontroller as the **"brain"** of many electronic devices you use every day. It's a compact integrated circuit that's specifically designed to control a particular operation in an embedded system.

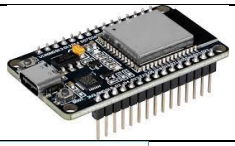
Definition

A microcontroller is a small computer built into a single integrated circuit (chip). It contains:

- **A processor (CPU)** – executes instructions
- **Memory (RAM and Flash)** – stores programs and data
- **Input/Output (I/O) pins** – connects to sensors, LEDs, etc.
- **Communication modules** – WiFi, Bluetooth, UART, etc.

Unlike a desktop computer, a microcontroller is designed to control specific tasks inside electronic systems.

Examples used in this module:

• ESP32 (by Espressif Systems)	
• M5Stack Core (ESP32-based development kit)	

2. Core Components of a Microcontroller

Every microcontroller contains several essential components that work together:

Central Processing Unit (CPU)

This is the actual "brain" that executes your program instructions. It reads your code line by line and performs the operations you've programmed.

Memory Systems

Microcontrollers have two main types of memory:

- **RAM (Random Access Memory)**: Stores temporary data while your program is running. Think of it as your workspace where you keep variables and data that change frequently. **Contents are lost when power is off.**
- **Flash Memory**: Stores your actual program code permanently, even when the power is turned off. **Contents remain when power is off.**

Input/Output Peripherals

These are the connections that allow your microcontroller to interact with the outside world. They include:

- **Digital pins** – can read switches (input) or control LEDs (output)
- **Analog pins** – can measure sensors (like temperature)
- **Communication interfaces** – let different devices talk to each other (I2C, SPI, UART)

3. What Does a Microcontroller Do?

A microcontroller:

1. **Reads inputs** – buttons, sensors, WiFi signals
2. **Processes data** – using programmed instructions
3. **Controls outputs** – LEDs, screens, motors

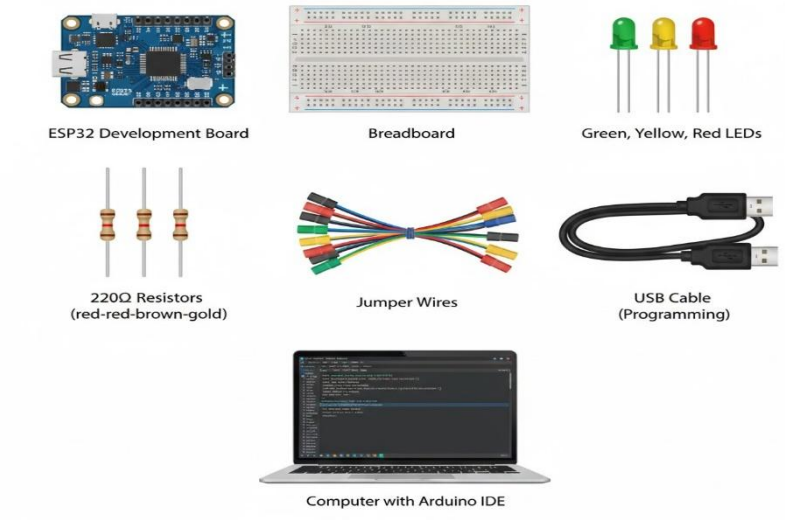
Experiment/Activity: WiFi Detection Project

Step-by-Step Build Procedure

Before You Start

Make sure you have the following materials:

- ESP32 development board
- Breadboard
- 3 LEDs (Green, Yellow, Red)
- 3× 220Ω resistors
- Jumper wires
- USB cable
- Computer with Arduino IDE installed



Step 1 – Insert the ESP32 into the Breadboard

1. Place the ESP32 across the center gap of the breadboard
2. Make sure both rows of pins are accessible
3. Do NOT connect USB power yet

Step 2 – Connect the GREEN LED (GPIO 25)

1. Insert the long leg (anode) of the green LED into the breadboard
2. Insert the short leg (cathode) into a different row
3. Connect:
    - GPIO 25 → one end of a 220Ω resistor
    - Other end of resistor → long leg of green LED
    - Short leg of green LED → GND pin on ESP32

The green LED represents **NORMAL** condition (0-2 networks).

Step 3 – Connect the YELLOW LED (GPIO 26)

1. Insert the yellow LED into the breadboard
2. Connect:
    - GPIO 26 → 220Ω resistor → long leg of yellow LED
    - Short leg → GND

The yellow LED represents **CAUTION** condition (3-5 networks).

Step 4 – Connect the RED LED (GPIO 27)

1. Insert the red LED into the breadboard
2. Connect:
    - GPIO 27 → 220Ω resistor → long leg of red LED
    - Short leg → GND

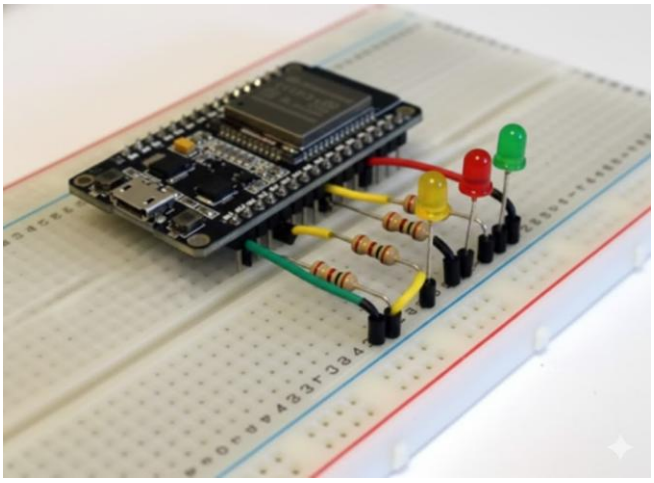
The red LED represents **ALERT** condition (6+ networks).

Step 5 – Check All Connections

Before powering the board:

✅ Each LED must have its own 220Ω resistor
✅ All short LED legs must go to GND
✅ No wires should touch incorrectly
✅ Verify GPIO numbers are correct (25, 26, 2

Step 6 – Connect and Power the ESP32



final should look like this 1

1. Connect the ESP32 to your computer using USB
2. Wait for the computer to detect the device

Step 7 – Configure Arduino IDE

1. Open Arduino IDE
2. Go to **Tools → Board → Select "ESP32 Dev Module"**
3. Go to **Tools → Port → Select the correct COM port**
4. Paste the provided code into the editor

```cpp
#include <WiFi.h>
int G = 25;  // Green LED pin
int Y = 26;  // Yellow LED pin
int R = 27;  // Red LED pin
void setup() {
  Serial.begin(115200);
  pinMode(G, OUTPUT);
  pinMode(Y, OUTPUT);
  pinMode(R, OUTPUT);
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
}
void loop() {
  Serial.println("Scanning...");
  int n = WiFi.scanNetworks();
  Serial.print("Networks detected: ");
  Serial.println(n);
  if (n > 10) {
    digitalWrite(R, HIGH);
    digitalWrite(Y, LOW);
    digitalWrite(G, LOW);
    Serial.println("ALERT - Red LED ON");
  }
  else if (n > 5) {
    digitalWrite(Y, HIGH);
    digitalWrite(R, LOW);
    digitalWrite(G, LOW);
    Serial.println("CAUTION - Yellow LED ON");
  }
  else {
    digitalWrite(G, HIGH);
    digitalWrite(R, LOW);
    digitalWrite(Y, LOW);
    Serial.println("NORMAL - Green LED ON"); }
  delay(5000);  // Wait 5 seconds before next scan
}
```

Step 9 – Test the System

1. Open Serial Monitor (Tools → Serial Monitor)
2. Set baud rate to **115200**
3. Observe the number of WiFi networks detected
4. Observe which LED turns on

Note

- **Understanding the Code (WiFi Detection Project) Include WiFi Library: #include <WiFi.h> includes the library that allows the ESP32 to use WiFi functions.**
- **Define LED Pins: Defines which GPIO pins on the ESP32 are connected to the Green, Yellow, and Red LEDs.**
- **Setup Function: Runs once when the ESP32 starts. Initializes the serial monitor, sets LED pins as OUTPUT, and prepares the WiFi module for scanning.**
- **Loop Function: Runs continuously. Scans for nearby WiFi networks, counts them, and makes decisions based on the number found.**

Uploading and Testing

1. **Connect your ESP32 to your computer using the USB cable.**
2. **Open the Arduino IDE.**
3. **Select your board: Tools → Board → "ESP32 Dev Module"**
4. **Select the correct port: Tools → Port → (select the COM port with ESP32)**
5. **Copy and paste the code into the editor.**
6. **Click the Upload button (right arrow icon).**
7. **Wait for "Done Uploading" to appear in the status bar.**

If upload fails:

- **Hold the BOOT button on the ESP32**
- **Click Upload in Arduino IDE**
- **Release the BOOT button when uploading begins**

Troubleshooting

Problem	Possible Solution
No LED lights up	Check wiring, verify GND connections, check resistor values
LED always stays on	Check if pins are correctly defined in code
Serial Monitor shows nothing	Check baud rate (must be 115200), check USB connection
"WiFi.h" not found error	ESP32 board package not installed (install via Boards Manager)
Upload fails	Hold BOOT button during upload, check port selection
Networks count = 0	Move to area with more WiFi signals, check WiFi module