Module: Introduction to Microcontrollers

Instructor Module: Temperature Sensor Display with Arduino

Module Overview: This module introduces students to reading analog sensors and displaying data. By the end of this module, students will understand how to interface a temperature sensor with an Arduino and show the results on an LCD screen.

By the end of this module, learners 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 projects (Temperature Sensor Display).
3. Recognize real-world applications in robotics, smart devices, automotive systems, IoT, and

Lesson Flow:

Warm-Up (5-10 minutes)

- **Instructor** Script: "Good morning, everyone! Today, we're diving into the world of microcontrollers—the tiny brains inside almost every electronic device you use."
- **Instructor** Script: "Let me ask you: What do a microwave, a smartwatch, a car, and a WiFi router all have in common?" (Wait for responses)
- **Instructor** Script: "That's right! They all contain a microcontroller. And today, you are going to program a microcontroller to do something pretty amazing: see invisible WiFi signals."
- **Instructor** Script: "We are using the ESP32 microcontroller, which has a built-in WiFi radio. We will program it to scan for nearby WiFi networks. Based on how many networks it finds, it will light up different LEDs. Green means quiet, yellow means busy, red means crowded!"
- **Instructor** Script: "This is the power of microcontrollers: they can read data from the world around them (like WiFi signals), process that information, and control outputs (like LEDs) to communicate with us."

Core Concepts (20 minutes)

- **Instructor** Script: "Before we build, let's review the core components of a microcontroller and see how they apply to our ESP32 and this project."

What is a Microcontroller?

- **Instructor** Script: "A microcontroller is a small computer on a single chip. It contains a processor, memory, and input/output pins—all in one package. It's designed to do specific tasks, unlike the general-purpose computer in your laptop."
- **Instructor** Script: "Our ESP32 is a perfect example of a modern microcontroller."

Central Processing Unit (CPU):

- **Instructor** Script: "The CPU is the 'brain' of the microcontroller. It fetches instructions from memory and executes them."
- **Instructor** Script: "In our project, the CPU will tell the WiFi radio when to scan, process the number of networks found, and decide which LED to turn on."

Memory (RAM & Flash):

- **Instructor** Script: "Microcontrollers have two main types of memory:"
    - Flash Memory: "This is where our program code is stored permanently. Even when we unplug the power, the code remains. In our project, the WiFi detection program is stored here."
    - RAM (Random Access Memory): "This is temporary storage. When our program runs, it stores variable data here—like the current number of WiFi networks found. When power is off, RAM is erased."

Input/Output Peripherals (I/O Pins):

- **Instructor** Script: "I/O pins allow the microcontroller to interact with the outside world."
- **Instructor** Script: "In our project, we are using digital output pins (GPIO 25, 26, 27) to control our three LEDs. We send a HIGH signal (3.3V) to turn an LED on, or LOW (0V) to turn it off."

Communication Peripherals (WiFi):

- **Instructor Script:** "The ESP32 is special because it has a built-in WiFi module. This is a communication peripheral that handles all the complex radio stuff."

Activity: Identify the Component

- **Instructor Script:** "Let's test your understanding. Which component of the microcontroller:"
    - "Stores the code permanently?" (Answer: Flash Memory)
    - "Executes the instructions?" (Answer: CPU)
    - "Turns the LED on and off?" (Answer: I/O Pins / Digital Output)
    - "Listens for WiFi signals?" (Answer: WiFi module / Communication Peripheral)
    - "Temporarily holds the network count?" (Answer: RAM)
- Transition: "Excellent! Now you understand the anatomy of our microcontroller. Let's look at the ESP32 itself."

Microcontroller Families (15 minutes)

- **Instructor Script:** "There are many microcontroller families: Arduino (using ATmega chips), PIC, STM32, and our focus today—the ESP32 by Espressif Systems."
- **Instructor Script:** "The ESP32 is a powerful microcontroller with built-in WiFi and Bluetooth. It's faster, has more memory, and includes wireless capabilities that other microcontrollers would need separate add-on modules for."

- **Instructor Script:** "This is why the ESP32 is so popular in IoT (Internet of Things) projects—it has everything built into one chip."

Show an ESP32 board (physical or image):

- **Instructor Script:** "This is the ESP32 Dev Board. Let's identify the key parts we will use today and connect them to the core components we just learned about."

Point out key features:

Component	Physical Location on ESP32	Function
CPU	Inside the metal shield	Executes our WiFi detection code
Flash Memory	Small black chip near antenna	Stores our program permanently
RAM	Inside the main chip	Holds the current WiFi count temporarily
I/O Pins	The rows of pin headers	Connect to our LEDs (GPIO 25, 26, 27)
WiFi Module	Metal shield and antenna	Scans for networks
USB Port	The connector on the edge	Powers the board and uploads code
GND Pins	Any pin labeled GND	Completes the circuit for LEDs

Discussion:

- **Instructor Script:** "Why do you think the ESP32 is a better choice for this project than a basic Arduino Uno?" (Expected Answer: The Uno doesn't have built-in WiFi. You'd need a separate WiFi shield, which is more expensive and complicated.)

Applications & Careers (15 minutes)

- **Instructor Script**: "The ESP32 microcontroller and WiFi scanning technology are used in real-world applications across many industries."

Highlight uses in each area:

Cybersecurity & Network Analysis:

- **Instructor Script**: "Network security professionals use specialized microcontrollers to scan for rogue WiFi networks and detect security threats. Our project is a simplified version of those professional tools."

Smart Homes (IoT):

- **Instructor Script:** "Smart plugs, light bulbs, and thermostats all contain microcontrollers with WiFi. When you set them up, they scan for nearby networks—exactly like our ESP32 does today."

Retail & Analytics:

- **Instructor Script:** "Retail stores use microcontroller-based WiFi scanners to count foot traffic by detecting smartphones searching for networks."

Industrial IoT:

- **Instructor Script:** "Factories use microcontrollers with WiFi to monitor equipment wirelessly. They scan for network availability and switch between access points automatically."

Group activity: Learners brainstorm devices around them that use microcontrollers

- **Instructor Script:** "Take one minute and tell the person next to you: what is one device you used today that contains a microcontroller AND has wireless capability?" (Examples: Smartphone, laptop, wireless earbuds, smartwatch, WiFi router, gaming console, smart TV, Bluetooth speaker.)

Connect to career paths:

- **Instructor Script:** "Learning about microcontrollers like the ESP32 opens doors to many careers:"
    - Embedded Systems Engineer: Designs the software inside microcontrollers.
    - IoT Developer: Builds connected devices for homes and industry.
    - Network Engineer: Understands wireless communication at a deep level.
    - Cybersecurity Specialist: Analyzes network traffic and vulnerabilities.
    - Robotics Engineer: Uses microcontrollers to control robots wirelessly.

Hands-On Project: WiFi Detection with ESP32 Microcontroller (30 minutes)

Overview:

- **Instructor Script:** "Now we get to build it. We will program our ESP32 microcontroller to:"
    - "READ INPUT: Use its built-in WiFi peripheral to scan for networks"
    - "PROCESS DATA: Count the networks and decide what the number means"
    - "CONTROL OUTPUT: Light up an LED to communicate the result to us"
- **Instructor Script:** "This is the Read → Process → Control cycle at the heart of every microcontroller application."

Demonstration:

- **Instructor Script:** "I will demonstrate the circuit setup step-by-step. Please follow along carefully and keep the USB cable unplugged until I say so

Circuit Setup

Step 1: Place the ESP32 Microcontroller on the Breadboard

- **Instructor Script:** "Take your ESP32 microcontroller board and place it carefully across the center gap of the breadboard. Make sure the pins are inserted firmly and the USB port is hanging off the end for easy access."

Step 2: Connect the GREEN LED (GPIO 25)

- **Instructor Script:** "Insert the green LED into the breadboard. Remember: LEDs have polarity. The long leg is positive (anode), and the short leg is negative (cathode)."
- **Instructor Script:** "Connect a 220Ω resistor from GPIO 25 on the ESP32 to the long leg of the green LED."
- **Instructor Script:** "Connect the short leg of the green LED directly to the GND (Ground) rail on the breadboard."

Step 3: Connect the YELLOW LED (GPIO 26)

- **Instructor Script**: "Repeat the same process for the yellow LED."
- **Instructor Script:** "Connect a 220Ω resistor from GPIO 26 to the long leg of the yellow LED."
- **Instructor Script:** "Connect the short leg to the GND rail."

Step 4: Connect the RED LED (GPIO 27)

- **Instructor Script:** "Finally, do the same for the red LED."
- **Instructor Script:** "Connect a 220Ω resistor from GPIO 27 to the long leg of the red LED."
- **Instructor Script:** "Connect the short leg to the GND rail."

Step 5: Connect the Ground Rail

- **Instructor Script:** "Take a jumper wire and connect the GND pin on the ESP32 to the GND rail on the breadboard. This completes the circuit for all LEDs."

Step 6: The Pre-Power Check (CRITICAL)

- **Instructor Script:** "STOP. Do NOT connect USB power yet. Let's check everything:"
    - "Does every LED have a resistor connected to its long leg?"
    - "Does every short leg go to the GND rail?"
    - "Is the GND rail connected to the ESP32's GND pin?"
    - "Are GPIO pins 25, 26, and 27 connected correctly?"

Step 7: Connect and Power the ESP32 Microcontroller

- **Instructor Script:** "Now, connect the ESP32 to your computer using the USB cable. The green LED on the board should light up, indicating the microcontroller has power."

Walk through Arduino IDE and Code:

- **Instructor Script:** "Open the Arduino IDE. This is where we write the code that will be stored in the ESP32's Flash memory and executed by its CPU."
- **Instructor Script:** "We need to make sure the ESP32 board package is installed. Go to Tools > Board > Board Manager. Search for 'ESP32' and install the package by Espressif Systems."
- **Instructor Script:** "Select your board: Tools > Board > ESP32 Arduino > ESP32 Dev Module."

- **Instructor Script:** "Select the correct port: Tools > Port > (select the COM port with ESP32)

The Code:

```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
}
```

Code Line	Microcontroller Concept
include <WiFi.h>	Includes library for the WiFi peripheral
int G = 25;	Defines which I/O pin is used
pinMode(G, OUTPUT);	Configures I/O pin as output
WiFi.scanNetworks();	Uses the WiFi peripheral to read input
int n = ...	Stores result in RAM temporarily
if (n > 10) {...}	CPU processes the data and makes decision
digitalWrite(R, HIGH);	CPU sends command to I/O pin to control output
delay(5000);	CPU pauses execution for timing

Learners Replicate and Upload Program:

- **Instructor Script:** "Now it's your turn. Type the code exactly as shown, double-check your pin numbers (25, 26, 27), and click the Upload button."
- **Instructor Script:** "When you click Upload, here's what happens:"
    - "The code is compiled into instructions the CPU understands"
    - "Those instructions are stored in the Flash memory of the ESP32"
    - "The ESP32 resets and the CPU starts executing from Flash"
    - "Your program begins running immediately!"
- **Instructor Script:** "If the upload fails: Hold the BOOT button on the ESP32, click Upload, and release when 'Connecting...' appears."
- **Instructor Script:** "Once uploaded, open the Serial Monitor (Tools > Serial Monitor). Set baud rate to 115200. Watch the microcontroller report its findings!"

Wrap-Up (5 minutes)

- **Instructor Script**: "Great job everyone! Today you programmed a real microcontroller to:"
    - "Use its WiFi peripheral to scan for networks (INPUT)"
    - "Store the count in RAM and let the CPU make decisions (PROCESS)"
    - "Control I/O pins to light up LEDs (OUTPUT)"
- **Instructor Script:** "This is exactly how microcontrollers work in the real world. The same principles apply whether you're building a WiFi detector, a smart thermostat, or a robot."
- **Instructor Script:** "Next time you use a device with a 'brain', remember—there's a microcontroller inside, doing exactly what we did today: reading, processing, and controlling."
- **Instructor Script:** "Any questions about microcontrollers, the ESP32, or how this project worked?"