Microcontroller Lab Manual

Lab Objectives

By the end of this lab, students will be able to:

- Understand the basic architecture of a microcontroller.
- Set up a development environment (IDE, compiler, programmer).
- Configure GPIO pins as input and output.
- Write, upload, and debug basic microcontroller programs.
- Implement a simple input–output control system (Button → LED).

Required Equipment

- Microcontroller board (Arduino Uno / STM32 Nucleo / ESP32 / PIC board)
- USB cable & programmer/debugger (if needed)
- Breadboard & jumper wires
- LED (1–2 pcs), push button (1 pc)
- Resistors: 220 Ω (LED), 10 kΩ (button pull-down)
- Computer with IDE installed (Arduino IDE / STM32CubeIDE / MPLAB X)

Pre-lab Preparation

- Review the differences between **microcontrollers** and **microprocessors**.
- Study microcontroller architecture: CPU, Flash memory, RAM, I/O ports.
- Install IDE and necessary drivers.

Lab Activities

Part A: Toolchain Setup (30 min)

1. Connect the microcontroller to the computer.
2. Open the IDE, select the correct board and COM port.
3. Upload a sample `Hello World` program (LED blink).

Example Code:

```
void setup() {
  pinMode(13, OUTPUT);    // Set pin 13 as output
}
void loop() {
  digitalWrite(13, HIGH); // LED ON
  delay(1000);            // Wait 1 sec
  digitalWrite(13, LOW);  // LED OFF
  delay(1000);            // Wait 1 sec
}
```

Part B: Digital Input (45 min)

1. Wire a push button to pin 2 with a pull-down resistor.
2. Modify the code to read button state.
3. Print button status to Serial Monitor.

Example Code:

```
int buttonPin = 2;
void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  int state = digitalRead(buttonPin);
  Serial.println(state);
  delay(200);
}
```

Part C: Input → Output Control (1 hr)

1. Connect LED to pin 8.
2. Write a program: LED turns ON only when button is pressed.
3. Extension: Press button once → LED toggles state (like a switch).

Part D: Debugging Practice (30 min)

- Use `Serial.print()` or debugger breakpoints to observe variable values.
- Introduce deliberate error (wrong pin, missing resistor) and troubleshoot.

Theoretical Background

- **Microcontroller vs. Microprocessor:**
  - Microcontroller: CPU + RAM + Flash + I/O ports on a single chip.
  - Microprocessor: Only CPU, requires external memory and peripherals.
- **GPIO (General Purpose Input/Output):** Digital pins that can be configured as input or output.
- **Pull-down Resistor:** Ensures a defined logic LOW when the button is not pressed.
- **Switch Bouncing:** Mechanical switches may produce multiple signals for one press → requires *debouncing*.

Assessment Questions

1. What is the main difference between microcontrollers and microprocessors?
2. Why do we need a pull-down resistor on the button input?
3. If you press the button quickly, why might the LED not always respond? (Hint: bouncing)
4. Modify your program so that the LED stays ON for 5 seconds after a button press.

Example Code for Extended Task

```
int buttonPin = 2;
int ledPin = 8;

void setup() {
  pinMode(buttonPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  if (digitalRead(buttonPin) == HIGH) {
    digitalWrite(ledPin, HIGH);
    delay(5000); // Keep LED ON for 5 seconds
    digitalWrite(ledPin, LOW);
  }
}
```