Lab 1: Introduction to Microcontrollers & GPIO

Instructor: Dr. Shabbir Ahmed
Time: 2:01 PM
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 (for LED), 10 kΩ (for button pull-down)
Computer with IDE installed (Arduino IDE / STM32CubeIDE / MPLAB X depending on MCU)
Pre-lab Preparation
Read about the differences between microcontrollers and microprocessors.
Review microcontroller architecture: CPU, Flash memory, RAM, I/O ports.
Install the required IDE and drivers for your platform.
Lab Activities
Part A: Toolchain Setup (30 min)
Connect the microcontroller board to your computer.
Open the IDE, select the correct board and COM port.
Upload a sample “Hello World” program (LED blink).
Example (Arduino C 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)
Wire a push button to pin 2 with a pull-down resistor.

Modify the code to read the button state.
Print button status to the Serial Monitor.
Example snippet:

```
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)
Connect LED to pin 8.
Write a program where the LED turns ON only when the button is pressed.
Extend: Press → 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) → troubleshoot.
Assessment Questions
What is the main difference between microcontrollers and microprocessors?
Why do we need a pull-down resistor on the button input?
If you press the button quickly, why might the LED not always respond? (hint: bouncing)
Modify your program so that the LED stays ON for 5 seconds after a button press.

I will be doing lab tomorrow, help me to get all these through by explaining all

Here is a detailed guide to help you understand and prepare for each part of your upcoming microcontroller & GPIO lab, including the technical background, code explanations, circuit setup, and answers to the assessment questions.[1] [2] [3] [4]

Microcontroller Basics

A **microcontroller** is an integrated circuit containing a CPU (Central Processing Unit), RAM (for temporary storage of data), Flash/ROM (for storing the program), and programmable I/O (Input/Output) ports for interfacing with external devices. This all-in-one approach distinguishes microcontrollers from microprocessors, which require external memory and peripherals.[2] [3] [1]

Typical architecture:

- **CPU**: Executes instructions, processes data, and coordinates peripherals.
- **Flash/ROM**: Stores your program (firmware) permanently.
- **RAM**: Temporarily stores data, variables, and intermediate results.
- **I/O Ports**: Interface for connecting buttons, LEDs, sensors, etc.[3] [2]
- **Peripherals**: Such as timers, ADCs, communication interfaces (UART, SPI, I2C).[2] [3]

Toolchain Setup

- Connect your board (e.g., Arduino Uno) to your computer using a USB cable.
- Open your IDE (like Arduino IDE) and select your board and COM port.
- Upload a simple program, like the Blink example, to verify everything works.

Sample Blink 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
}
```

This will make the onboard LED blink, verifying your setup.[4]

Digital Input: Button Wiring & Reading

- Wire a push button to digital pin 2.
- Use a **10kΩ pull-down resistor** to ensure the pin is LOW (0V) when the button isn't pressed. This prevents floating state and random readings.[3]
- Modify code to read the button state and print it via Serial Monitor.

Example wiring and code:

```
int buttonPin = 2;
void setup() {
    pinMode(buttonPin, INPUT);
    Serial.begin(9600);
}
void loop() {
    int state = digitalRead(buttonPin);
    Serial.println(state); // prints 0 (not pressed) or 1 (pressed)
    delay(200);
}
```

Check your Serial Monitor to see the change when the button is pressed.

Input→Output Control: Button to LED

- Connect an LED (with a 220Ω resistor) to pin 8. Longer leg (anode) to pin 8, shorter leg (cathode) to ground.
- Write code so that pressing the button turns ON the LED:

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

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

void loop() {
    int state = digitalRead(buttonPin);
    if(state == HIGH){
        digitalWrite(ledPin, HIGH); // turn LED ON
    } else {
        digitalWrite(ledPin, LOW);  // turn LED OFF
    }
}
```

- To toggle the LED with each button press, you'll need to remember the last button state and change the LED state only on a LOW-to-HIGH transition ("edge detection").

Debugging Practice

- Use `Serial.print()` to output variable values and understand program flow.
- If things don't work, check for common errors: wrong pin numbers, missing/incorrect resistors, faulty connections.
- Deliberately make mistakes (e.g., remove the pull-down resistor) and observe unpredictable behavior.

Assessment Questions & Answers

1. Main difference: Microcontroller vs Microprocessor
A microcontroller has CPU, memory (RAM + program storage), and I/O ports on a single chip—it is a "computer on a chip" for control tasks. A microprocessor is just a CPU and needs external RAM, ROM, and peripherals; it is designed for general-purpose computing.[5] [1] [2]

2. Why a pull-down resistor on the button input?
A pull-down resistor ensures the input pin reads LOW (0V) when the button is not pressed, preventing floating readings and random values when the circuit is open.[3]

3. Why might the LED not always respond to quick button presses?
Physical buttons can "bounce"—they make and break contact very rapidly when pressed or released, causing multiple fast changes in the digital signal. If not handled in software (debouncing), these could cause missed or extra triggers.[3]

4. LED stays ON for 5 seconds after button press: sample code

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

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

void loop() {
    static unsigned long lastPressTime = 0;
    static bool ledOn = false;

    if (digitalRead(buttonPin) == HIGH) {
        lastPressTime = millis();
        ledOn = true;
        digitalWrite(ledPin, HIGH);
    }

    if (ledOn && (millis() - lastPressTime >= 5000)) {
        ledOn = false;
        digitalWrite(ledPin, LOW);
    }
}
```

This code keeps the LED ON for 5 seconds after a button press, even if the button is quickly released.

Tips for the Lab

- Double-check wiring. Use the right resistors.
- Read Serial Monitor output to debug.
- Save your code versions after each step.
- If using a different board (STM32, ESP32, PIC), setup steps are similar, but pin numbers and uploading methods will differ.[4]

You'll be well prepared for your lab after reviewing and understanding these steps and explanations.[2] [4] [3]

⁂

1. https://www.geeksforgeeks.org/digital-logic/microcontroller-and-its-types/
2. https://datacapturecontrol.com/articles/io-devices/microcontrollers/overview
3. https://hyphendeux.com/microcontrollers-the-heart-of-embedded-systems/
4. https://www.rknec.edu/wp-content/uploads/2024/02/Microcontroller-Lab_-Manual.pdf
5. http://gpnilokheri.ac.in/GPContent/MCES.pdf
6. https://www.pdit.ac.in/images/programs/ece/lab-manuals/MICROCONTROLLER LAB MANUAL 2024.pdf
7. https://arxiv.org/html/2503.05741v1
8. https://www.ect.ac.uk/resources/types-and-applications-of-microcontrollers/
9. https://www.scribd.com/document/849373489/Microcontrollers-Lab-Manual-East-Point

10. https://site.nafc.gov.ng/fill-and-sign-pdf-form/browse/_pdfs/8_Lab_Details_8_1_Microprocessor_And_Microcontrollers_Lab.pdf
11. https://gnindia.dronacharya.info/ECE/Downloads/Labmanuals/Microprocessor-Microcontroller-20012025.pdf
12. https://www.scribd.com/document/462963739/Microcontroller-Lab-Manual-pdf
13. https://www.calameo.com/books/000571032faccbb4bfe47
14. https://www.scribd.com/document/748384995/Micro-Controller-Lab-Manual-2022-Scheme
15. https://www.geeksforgeeks.org/electronics-engineering/8051-microcontroller-architecture/
16. https://en.wikipedia.org/wiki/Microcontroller
17. https://ate.is/downloads/40649/Introduction-to-Microcontrollers-1735332388.pdf
18. https://uomustansiriyah.edu.iq/media/lectures/5/5_2021_12_29!01_08_37_AM.pdf
19. https://www.techtarget.com/iotagenda/definition/microcontroller