**Lab Exercise 4- Arduino Serial Port for Internal LED**

The **Serial Port** in Arduino allows communication between the Arduino board and a computer or another device. In this lab exercise, you'll learn how to use the Arduino's serial port for debugging, sending, and receiving data. The serial communication is very useful in real-time applications, such as monitoring sensor data or sending commands to the Arduino.

**Objective:**

* Set up serial communication between the Arduino and the computer.
* Send and receive data via the Arduino's serial port.

**Materials:**

* Arduino board (e.g., Arduino Uno)
* USB cable to connect the Arduino to your computer
* Arduino IDE

**Steps to Perform the Lab:**

**Step 1: Setting up the Arduino IDE**

1. Install the **Arduino IDE** if you haven't already.
2. Open the Arduino IDE and connect your Arduino board to your computer via a USB cable.
3. In the **Tools** menu, select the correct **Board** (e.g., Arduino Uno) and **Port**.

**Step 2: Basic Serial Communication (Sending Data)**

The goal here is to print text to the Serial Monitor on your computer.

1. **Code to Send Data from Arduino to Serial Monitor:**

void setup() {

// Start the Serial communication at a baud rate of 9600

Serial.begin(9600);

}

void loop() {

// Send a message to the Serial Monitor

Serial.println("Hello, Arduino!");

// Delay for a second

delay(1000);

}

1. **Explanation:**
   * Serial.begin(9600);: Initializes serial communication at a baud rate of 9600 bits per second. This must be done inside the setup() function.
   * Serial.println("Hello, Arduino!");: Sends the string "Hello, Arduino!" to the Serial Monitor. Each println sends a newline at the end.
   * delay(1000);: Delays the program for 1000 milliseconds (1 second), so the message will be sent every second.
2. **How to View Output:**
   * Upload the sketch to the Arduino.
   * Open the **Serial Monitor** in the Arduino IDE (Tools > Serial Monitor) or press **Ctrl+Shift+M**.
   * You should see the message "Hello, Arduino!" printed every second in the Serial Monitor.

**Step 3: Receiving Data from the Serial Port**

In this step, we'll learn how to send data from the Serial Monitor to the Arduino and how the Arduino can respond to it.

1. **Code to Read Input from the Serial Monitor:**

void setup() {

// Start Serial communication at a baud rate of 9600

Serial.begin(9600);

// Print an initial message to the Serial Monitor

Serial.println("Enter any character:");

}

void loop() {

// Check if data is available to read

if (Serial.available() > 0) {

// Read the incoming byte

char receivedChar = Serial.read();

// Print the received character

Serial.print("You entered: ");

Serial.println(receivedChar);

}

}

1. **Explanation:**
   * Serial.available(): Returns the number of bytes available to read from the serial buffer. If data is available, the Arduino can read it.
   * Serial.read(): Reads a byte from the incoming serial data and stores it in the receivedChar variable.
   * Serial.print(): Prints text without a newline, whereas Serial.println() prints text with a newline.
2. **How to Use:**
   * Upload the code to the Arduino.
   * Open the **Serial Monitor**.
   * In the Serial Monitor's input box, type a character or string and press **Enter**.
   * The Arduino will respond by printing the received character back to the Serial Monitor.

**Step 4: Using the Serial Monitor to Control an LED**

In this step, you’ll control the Arduino’s built-in LED (usually on pin 13) using serial commands sent from the Serial Monitor.

1. **Code to Control an LED via the Serial Port:**

void setup() {

// Initialize Serial communication at 9600 baud

Serial.begin(9600);

// Set the built-in LED pin (13) as output

pinMode(LED\_BUILTIN, OUTPUT);

// Print instructions to Serial Monitor

Serial.println("Type '1' to turn ON the LED, '0' to turn it OFF");

}

void loop() {

// Check if there is data available in the serial buffer

if (Serial.available() > 0) {

// Read the incoming byte

char command = Serial.read();

// Turn ON the LED if the command is '1'

if (command == '1') {

digitalWrite(LED\_BUILTIN, HIGH);

Serial.println("LED is ON");

}

// Turn OFF the LED if the command is '0'

else if (command == '0') {

digitalWrite(LED\_BUILTIN, LOW);

Serial.println("LED is OFF");

}

}

}

1. **Explanation:**
   * pinMode(LED\_BUILTIN, OUTPUT);: Sets the built-in LED pin as an output so it can be controlled.
   * digitalWrite(LED\_BUILTIN, HIGH);: Turns the LED on.
   * digitalWrite(LED\_BUILTIN, LOW);: Turns the LED off.
   * The Arduino listens for '1' or '0' commands from the Serial Monitor to control the LED.
2. **How to Use:**
   * Upload the code to the Arduino.
   * Open the **Serial Monitor**.
   * Type '1' to turn the LED on, and type '0' to turn it off.

**Summary of Key Concepts:**

* **Serial.begin(baudrate):** Starts serial communication at a specified baud rate.
* **Serial.print()** and **Serial.println():** Used to send data to the Serial Monitor.
* **Serial.available():** Checks if there is data to read.
* **Serial.read():** Reads incoming serial data.