Lab Worksheet 1

CSE461: Introduction to Robotics

Department of Computer Science and Engineering

Lab 01: Introduction to Arduino microcontroller and control an LED with a switch using the board.

I. Topic Overview

This lab introduces the fundamentals of microcontroller programming and interfacing using the Arduino development board. Students will learn about the Arduino pins, their functions, and how to use them to control simple input/output devices such as LEDs and switches. By the end of the lab, students will gain a foundational understanding of Arduino programming and circuit design.

II. Learning Outcome

After this lab, students will be able to:

1. Identify the various pins on the Arduino board and understand their functions.
2. Build and troubleshoot simple circuits using LEDs, resistors, and switches.
3. Write and upload Arduino code to control the I/O devices.
4. Use the Arduino Serial Monitor to debug and display system status.

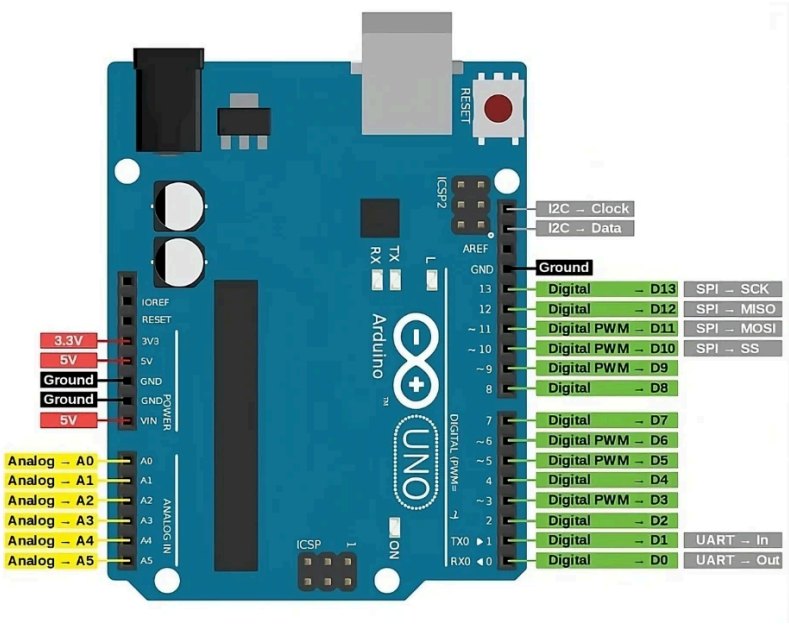
III. Materials

- Arduino Uno R3
- LED (Light Emitting Diode)
- Push-button switch
- Resistor (220Ω for LED)
- Breadboard
- Jumper wires

IV. Arduino Overview

Arduino is a popular open-source electronics platform known for its ease of use, affordability, and versatility, making it a staple in robotics and embedded systems. Powered by the ATmega328P microcontroller, it can be programmed using its user-friendly IDE, which simplifies hardware programming through an extensive collection of open-source libraries, enabling even beginners to control various I/O devices with ease. Its vast community support and compatibility with a wide range of modules make it ideal for rapid prototyping and experimentation in robotics. Additionally, Arduino's real-time control capabilities and support for communication protocols such as I2C, SPI, and UART allow seamless integration of multiple components.

V. Arduino Pins



The Arduino Uno has the following types of pins:

1. **Digital Pins (0–13):** Can be used as input or output pins. They operate at 5V and can provide or receive a maximum of 40mA current.
2. **Analog Pins (A0–A5):** Used to read analog signals (e.g., from sensors). They have a resolution of 10 bits (0–1023).

3. **Power Pins:**
    a. **5V and 3.3V:** Provide regulated voltage output.
    b. **GND:** Ground pins.
    c. **Vin:** Input voltage for external power supply.
4. **Special Pins:**
    a. **PWM (~) Pins:** Digital pins capable of Pulse Width Modulation (e.g., pins 3, 5, 6, 9, 10, 11).
    b. **UART Pins:** Used for UART communication (TX: pin 1, RX: pin 0).
    c. **SPI Pins:** Used for SPI communication (MOSI: pin 11, MISO: pin 12, SCK: pin 13, SS: pin 10).
    d. **I2C Pins:** Used for I2C communication (SDA: A4, SCL: A5)

Note: *The Arduino Uno has two sets of I2C pins:* ***SDA (A4) and SCL (A5)****, as well as dedicated* ***SDA and SCL pins near the RESET pin****. Early Arduino boards did not have dedicated I2C pins, so* ***A4 (SDA) and A5 (SCL)*** *were used for I2C communication. Both sets are internally connected and function the same way, allowing flexibility in wiring and shield compatibility.*

VI. Setting Up the Arduino IDE

Before starting the experiments, ensure the Arduino IDE is properly set up on your computer. Follow these steps:

- **Install the Arduino IDE:** Download and install the latest version of the Arduino IDE from the official website: https://www.arduino.cc/en/software
- **Connect the Arduino Board:** Use a USB cable to connect the Arduino board to the computer.
- **Select the Board and Port**:
    - Go to `Tools > Board` and select the appropriate Arduino board (e.g., Arduino Uno).
    - Go to `Tools > Port` and select the COM port to which your Arduino is connected.

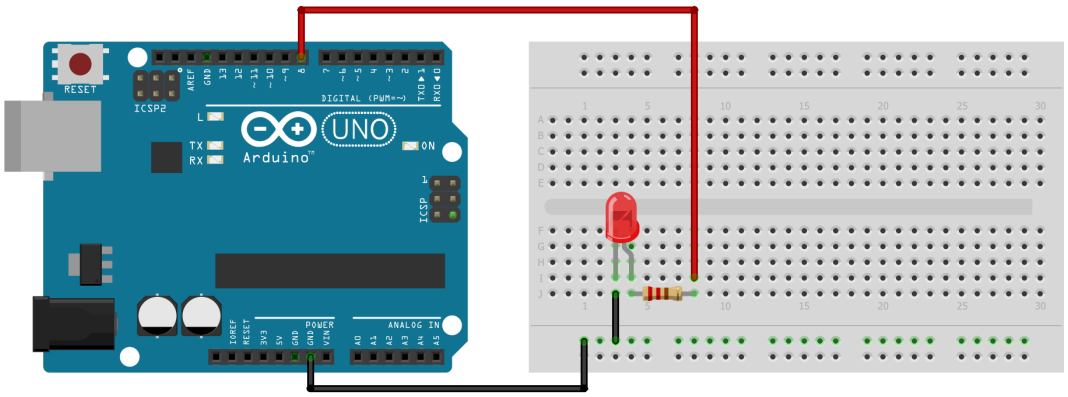
This setup ensures that the connected Arduino board is ready for programming and interfacing with external components.o

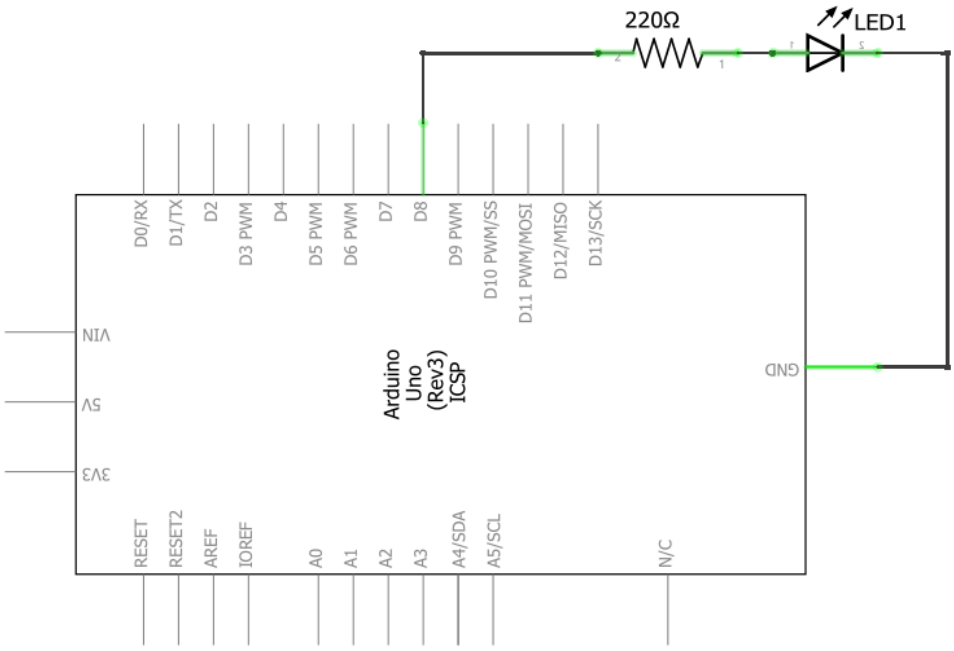
VII. Experiment 1: LED blinking.

1. **Description the components:**
   - LED (Light Emitting Diode)
   - Resistor (220Ω for LED)

2. **Setting up the circuit:**
   a. Connect the 220Ω resistor to the anode (longer leg) of the LED.
   b. Connect the other end of the resistor to digital pin 13 on the Arduino.
   c. Connect the cathode (shorter leg) of the LED to the GND pin on the Arduino.

3. **Circuit diagram:**





4. **Code:**

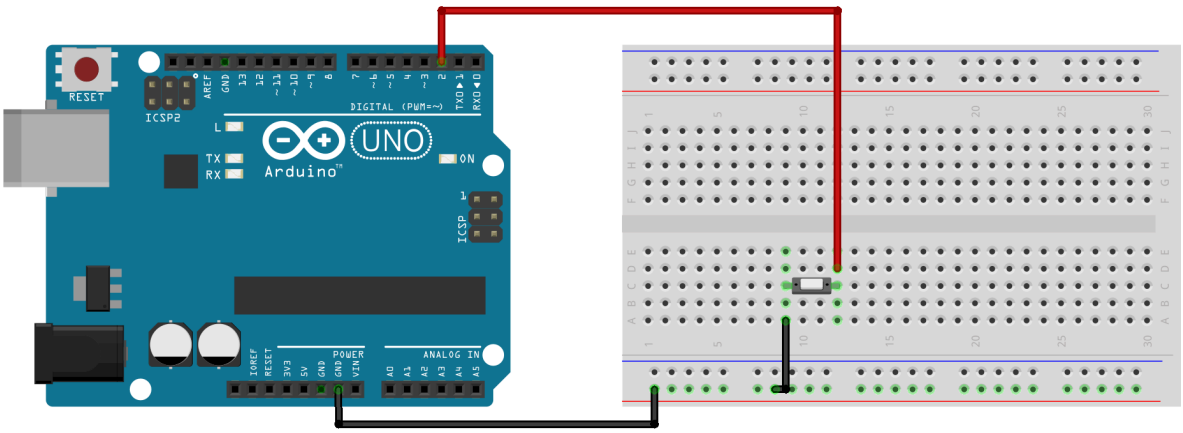
```
#define LED_PIN 8 // Define the LED pin

void setup() {
  pinMode(LED_PIN, OUTPUT); // Set the LED pin as an output
  Serial.begin(9600);       // Initialize serial communication
}

void loop() {
  digitalWrite(LED_PIN, HIGH); // Turn the LED on
  Serial.println("LED ON");    // Print LED status
  delay(1000);                 // Wait for 1 second
  digitalWrite(LED_PIN, LOW);  // Turn the LED off
  Serial.println("LED OFF");   // Print LED status
  delay(1000);                 // Wait for 1 second
}
```

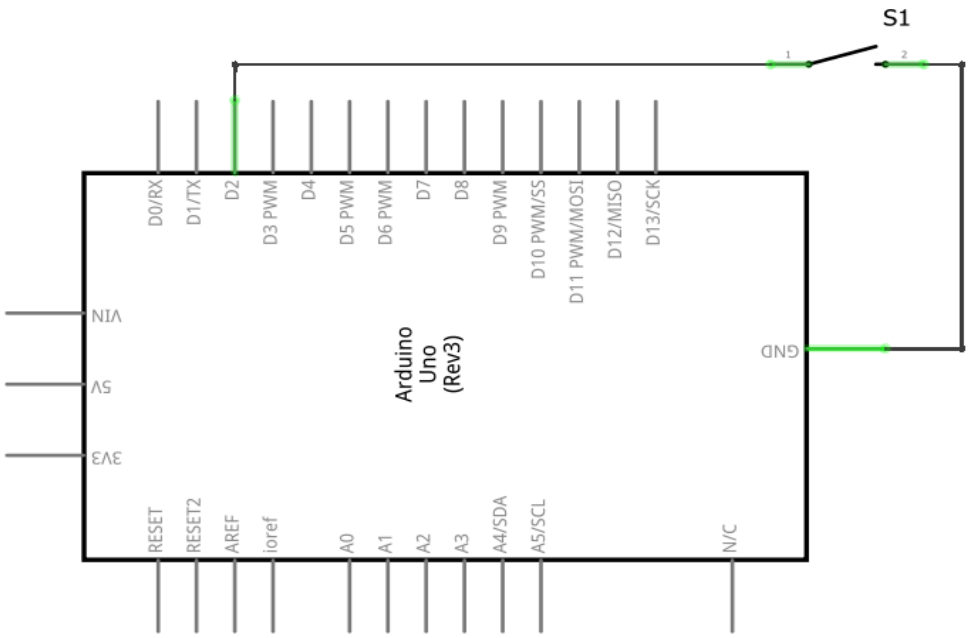
How this works: *The Arduino controls the LED by setting the specified digital pin (e.g., pin 13) as an output. Internally, the microcontroller toggles the pin's voltage between HIGH (5V) and LOW (0V) using the digitalWrite() function. When the pin is set to HIGH, current flows through the LED, turning it on. When set to LOW, the current stops, turning it off. The delay() function pauses the program for a specified time, creating the blinking effect. The Arduino's internal circuitry ensures precise timing and voltage control, making it easy to manage simple output devices like LEDs.*

VIII. Experiment 2: Printing “Switch pressed.” on the serial monitor using a switch.

1. **Description the components:**
   - Push-button switch
2. **Setting up the circuit:**
   a. Connect one terminal of the switch to the GND pin.
   b. Connect the other terminal of the switch to digital pin 2

3. **Circuit diagram:**





4. **Code:**

```
#define BUTTON_PIN 2 // Define the button pin

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP); // Set the button pin as input
  Serial.begin(9600);              // Initialize serial communication
}

void loop() {
  if (digitalRead(BUTTON_PIN) == LOW) { // Button is pressed
    Serial.println("Button pressed."); // Print message to S. Monitor
  }else{
      Serial.println("Button not pressed.");
  }
  delay(100); // Small delay to debounce the button
}
```

How this works: *The Arduino reads the state of the switch using a digital input pin configured with an internal pull-up resistor. When the switch is not pressed, the pull-up resistor keeps the pin at HIGH (5V). When the switch is pressed, it connects the pin to GND, pulling the voltage to LOW (0V). The digitalRead() function detects this change, and the Arduino prints "Switch pressed." to the Serial Monitor. This interaction demonstrates how the microcontroller processes input signals to trigger specific actions.*

VIII. Lab Task: Control LED using a switch.

Task Description:

Modify the circuit and code from Experiments 1 and 2 to turn ON an LED only when the switch is pressed. The LED should turn OFF when the switch is released.

Deliverables:

- Circuit diagram
- Arduino code
- Demonstration of the working circuit

IX. References:

1. Arduino Official Documentation: https://www.arduino.cc/en/Guide
2. Arduino IDE Download: https://www.arduino.cc/en/software