Arduino Mini Labs & Beginner Hardware Projects

Prepared By: Ipshita.

Lab #1: Blink an LED

Objective: Learn how to wire and control an LED using Arduino

Components Needed:

- 1x LED (any color)
- 1x 220Ω resistor
- 1x Breadboard
- Jumper wires
- Arduino Uno
- USB cable to connect to PC

Steps:

1. **Wiring**:

- **Long leg (anode)** of the LED → connect to **one end of a resistor (220Ω)**
- The **other end of the resistor** → connect to **pin 13** on the Arduino
- **Short leg (cathode)** of the LED → connect to **GND (Ground)** on the Arduino

Why the resistor?: To prevent too much current from flowing through the LED and burning it out.

Tip: If the LED doesn't light up, double-check the leg direction — **long leg = + (goes to pin)**

2. **Upload this code in Arduino IDE**:

```
void setup() {
  pinMode(13, OUTPUT); // Set pin 13 as output
}

void loop() {
  digitalWrite(13, HIGH); // LED ON
  delay(1000);          // Wait 1 second
  digitalWrite(13, LOW);  // LED OFF
  delay(1000);          // Wait 1 second
```

Challenge Tasks:

1. Change the Blink Speed (200ms)

Goal: Make the LED blink faster — every 200 milliseconds.

Hint:

- The `delay(1000)` means wait 1 second (1000 ms).
- What happens if you change both delays to `delay(200)`?
- Try it and observe the LED speed.

```
// Try changing the delay time here ↓
delay(200);  // 200 ms = faster blinking
```

2. Add a Second LED to Pin 12 and Blink It Alternately

Goal: Use a second LED and make it blink when the first one is off.

Hint:

- Copy the first LED's wiring for a second LED (use pin 12 this time).
- Add `pinMode(12, OUTPUT);` in `setup()`
- Inside `loop()`, turn **pin 13 ON** while **pin 12 is OFF**, then swap.

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT); // second LED
}

void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);   // opposite state
  delay(500);

  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);  // now second LED ON
  delay(500);
}
```

3. Try Different Delays (500ms and 2000ms)

Goal: Understand how the delay function changes blink timing.

💡 **Hint:**

- Try changing both delay lines to `delay(500)` and observe
- Then try `delay(2000)` — see how much slower it gets?
- This helps you understand **timing** and **event control**

```
delay(500);   // half a second

delay(2000);  // two full seconds
```

Lab #2: Button-Powered LED

Objective:

Learn to use a **button** as input — press to turn an LED ON, release to turn it OFF.

This will help him understand:

- `pinMode(..., INPUT)`
- `digitalRead()`
- Real-time input → output logic

Components:

- 1x pushbutton
- 1x LED
- 1x 220Ω resistor (for LED)
- 1x 10kΩ resistor (for pull-down on button)
- Breadboard
- Jumper wires
- Arduino Uno

Wiring Hints:

- One side of the **button** → connect to **5V**

- The **other side of the button** → connect to **pin 2** (or any digital input pin)

- Also connect that side of the button to **GND through a 10kΩ resistor** (this is a pull-down resistor)

This setup means:

- When the button is pressed: pin 2 = HIGH
- When the button is released: pin 2 = LOW (thanks to the pull-down)
- **LED wiring:**
  - Long leg → resistor → **pin 13**
  - Short leg → **GND**

Why the pull-down resistor?

To make sure the pin doesn't "float" and read random values when the button isn't pressed.

Tip:

You can test the button logic first by printing to Serial Monitor:

```
Serial.println(digitalRead(2));
```

Basic structure of code:

```
void setup() {
  pinMode(2, INPUT);      // Button pin
  pinMode(13, OUTPUT);    // LED pin
}

void loop() {
  int buttonState = digitalRead(2);  // Read the button

  if (buttonState == HIGH) {
    digitalWrite(13, HIGH);  // Turn LED ON
  } else {
    digitalWrite(13, LOW);   // Turn LED OFF
  }
}
```

Challenge Tasks:

1. **Change the pin numbers** — move LED to pin 12 or button to pin 3.
2. **Reverse the logic** — LED is **ON when button is NOT pressed**
3. **Add a second button** — control 2 LEDs with 2 buttons (great for logic practice)

Lab #3: Read Analog Input from Potentiometer

Objective:
Learn to read analog values using a potentiometer. This lab introduces analog input, which allows you to simulate different levels of control (like simulating brain signal intensity in BCI systems).

Components Needed:
• 1x potentiometer
• 1x LED
• 1x 220Ω resistor
• Breadboard
• Jumper wires
• Arduino Uno

Wiring Hints:
• Connect the **middle pin** of the potentiometer to **A0** on the Arduino
• Connect one side pin of the potentiometer to **5V**
• Connect the other side pin to **GND**
• Connect LED long leg to resistor → pin 13
• LED short leg → GND

```
void setup() {
  Serial.begin(9600); // To monitor values from the potentiometer
  pinMode(A0, INPUT);
}

void loop() {
  int potValue = analogRead(A0);  // Read value between 0 and 1023
  Serial.println(potValue);      // Print to Serial Monitor
  delay(200);                    // Slow down reading updates
}
```

Challenge Tasks:

1. Observe how the value changes as you turn the knob
2. Replace A0 with A1 or A2 and test other analog pins
3. Use `Serial.print("Value: ");` to label the output

Lab #4: LED Brightness Control with Potentiometer

Objective:
Use a potentiometer to control the brightness of an LED using PWM (analog output simulation).

Components Needed:
• 1x potentiometer
• 1x LED
• 1x 220Ω resistor
• Breadboard
• Jumper wires
• Arduino Uno

Wiring Hints:
Same as Lab #3 for the potentiometer:
• Middle pin → A0
• One side → 5V
• Other side → GND

LED:
• Long leg → resistor → pin 9 (PWM pin)
• Short leg → GND

```
void setup() {
  pinMode(A0, INPUT);
  pinMode(9, OUTPUT);
}

void loop() {
  int potValue = analogRead(A0);          // 0 to 1023
  int brightness = map(potValue, 0, 1023, 0, 255); // Map to PWM range
  analogWrite(9, brightness);             // LED brightness
  delay(10);
```

Challenge Tasks:

1. Try using a different PWM pin (e.g., pin 5 or 6)
2. Reverse the brightness logic (high pot = low brightness)
3. Add `Serial.println(brightness);` to see mapped values

Lab #5: Servo Motor Control with Potentiometer

Objective:
Control the position of a servo motor using a potentiometer. This simulates physical movement based on input level, similar to BCI-controlled robotics.

Components Needed:
- 1x potentiometer
- 1x servo motor
- Breadboard
- Jumper wires
- Arduino Uno

Wiring Hints:
- Potentiometer wiring is the same as in previous labs
- Servo:

  - Red wire → 5V
  - Brown/black wire → GND
  - Orange/yellow wire (signal) → pin 9

```
#include <Servo.h>


Servo myServo;


void setup() {
  myServo.attach(9);    // Servo signal pin
  pinMode(A0, INPUT);
}


void loop() {
  int potValue = analogRead(A0);            // 0 to 1023
  int angle = map(potValue, 0, 1023, 0, 180);  // Map to angle range
  myServo.write(angle);                  // Rotate servo
  delay(10);
}
```

Challenge Tasks:

1. Change the mapping to limit rotation (e.g., 30 to 150 degrees only)
2. Add `Serial.println(angle);` to watch servo values
3. Try different potentiometer pins (A1, A2) or servo pins (10, 11)

Lab #6: Reaction Game with Button and Buzzer

Objective:
Create a simple reaction game. A buzzer turns on when the LED blinks, and you have to press the button as fast as possible. This introduces timing, input handling, and output feedback — good practice for BCI-inspired reaction models.

Components Needed:
• 1x pushbutton
• 1x LED
• 1x buzzer module or piezo buzzer
• 1x 10kΩ resistor (for pull-down)
• 1x 220Ω resistor (for LED)
• Breadboard
• Jumper wires
• Arduino Uno

Wiring Hints:

Button:
• One side of the button → 5V
• Other side → pin 2 and also to GND via 10kΩ resistor (pull-down)

LED:
• Long leg → resistor → pin 13
• Short leg → GND

Buzzer:
• Positive (usually marked) → pin 9
• Negative → GND

```
void setup() {
  pinMode(2, INPUT);      // Button
  pinMode(13, OUTPUT);    // LED
  pinMode(9, OUTPUT);     // Buzzer
}

void loop() {
  digitalWrite(13, HIGH);  // LED on
  delay(1000);             // Wait
  digitalWrite(13, LOW);   // LED off

  int buttonState = digitalRead(2);

  if (buttonState == HIGH) {
    digitalWrite(9, HIGH);  // Buzzer on if button is pressed
    delay(500);
    digitalWrite(9, LOW);   // Buzzer off
  }
```

Challenge Tasks:

1. Add a delay between LED turning on and allowing button press
2. Make the buzzer sound only if the button is pressed within 1 second
3. Use `millis()` instead of `delay()` to improve responsiveness (advanced)

Lab #7: Display Sensor Values on LCD

Objective:
Learn how to display messages and sensor readings on an LCD screen using the LiquidCrystal library. This helps simulate live feedback, similar to what is seen in real BCI systems.

Components Needed:
• 1x 16x2 LCD display (with or without I2C module)
• 1x potentiometer (for contrast)
• Breadboard
• Jumper wires
• Arduino Uno

Wiring Hints (for regular LCD without I2C):

• Connect LCD pins as follows:

- RS → pin 12
- E → pin 11
- D4 → pin 5
- D5 → pin 4
- D6 → pin 3
- D7 → pin 2
- VSS → GND
- VDD → 5V
- V0 (contrast) → middle pin of potentiometer
- RW → GND
- A (backlight +) → 5V
- K (backlight -) → GND

```
#include <LiquidCrystal.h>


// Create LCD object: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);


void setup() {
  lcd.begin(16, 2);            // Set LCD size
  lcd.print("Hello, Alan!");    // Print message
  delay(2000);
  lcd.clear();
}


void loop() {
  int potValue = analogRead(A0);
  lcd.setCursor(0, 0);
  lcd.print("Pot Value: ");
  lcd.setCursor(11, 0);
  lcd.print(potValue);  // Print changing value
  delay(200);
}
```

Challenge Tasks:

1. Display a second line with a message like “Focus: HIGH” based on pot value
2. Use `map()` to show angles or brightness instead of raw values
3. Replace potentiometer with a button and show "Pressed"/"Released" on LCD

Lab #8: IR Remote Controlled Output

Objective:
Use an IR remote and sensor to control outputs like LEDs or a buzzer. This simulates sending "commands" from a distance, similar to brain-triggered actions in real BCI systems.

Components Needed:
- 1x IR receiver module
- 1x IR remote control
- 1x LED
- 1x 220Ω resistor
- Breadboard
- Jumper wires
- Arduino Uno

Wiring Hints:
- IR receiver has 3 pins:

  - GND → GND
  - VCC → 5V
  - OUT → pin 11 (digital input)

- LED wiring:

  - Long leg → resistor → pin 13
  - Short leg → GND

Setup Notes:
- You must install the **IRremote** library

In Arduino IDE: Go to **Sketch > Include Library > Manage Libraries**, search for **IRremote** and install it

```
#include <IRremote.h>


const int recv_pin = 11;

IRrecv irrecv(recv_pin);

decode_results results;


void setup() {

 Serial.begin(9600);

 irrecv.enableIRIn();  // Start the receiver

 pinMode(13, OUTPUT);

}


void loop() {

 if (irrecv.decode(&results)) {

  Serial.println(results.value, HEX);  // Print the button code

  if (results.value == 0xFFA25D) {    // Replace with your remote's button code

    digitalWrite(13, HIGH);          // Turn LED on

  } else {

    digitalWrite(13, LOW);           // Turn LED off

  }

  irrecv.resume(); // Receive next signal

 }

}
```

Challenge Tasks:

1. Use different buttons to turn on different outputs (e.g. pin 12, pin 9)
2. Add a buzzer that sounds with a specific button
3. Create a mode switch: One button enables LED blinking, another disables it

Lab #9: Distance Sensing with Ultrasonic Sensor

Objective:
Use an ultrasonic sensor to measure distance and control outputs based on proximity. This simulates environmental awareness, useful in adaptive BCI scenarios.

Components Needed:
• 1x Ultrasonic sensor (HC-SR04)
• 1x LED
• 1x 220Ω resistor
• Breadboard
• Jumper wires
• Arduino Uno

Wiring Hints:
• VCC → 5V
• GND → GND
• Trig → pin 9
• Echo → pin 10

• LED wiring:

- Long leg → resistor → pin 13
- Short leg → GND

Code:

```
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
```

```
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance < 10) {
    digitalWrite(13, HIGH);  // LED ON if object is near
  } else {
    digitalWrite(13, LOW);   // LED OFF
  }

  delay(500);
}
```

Challenge Tasks:

1. Change the trigger distance (e.g. light LED if object < 20 cm)
2. Use distance to control brightness of an LED (map to PWM)
3. Show distance on LCD instead of Serial Monitor