**Lab Report No**: 02

**Lab Title:**

Controlling LED Blinking at Specified Intervals Using Digital and Analog Modes on an Arduino

**Introduction:**

This lab demonstrates how to control the blinking behavior of an LED on an Arduino at specified time intervals using both digital and analog modes. In digital mode, the LED will blink on and off with precise timing. In analog mode, the LED will fade in and out gradually, creating a smoother visual effect. This experiment covers the basics of digital and analog signal control, including timing and Pulse Width Modulation (PWM) on the Arduino.

**Objectives:**

1. To control the blinking of an LED using digital and analog outputs on the Arduino.
2. To set specific time intervals for blinking in digital mode.
3. To achieve a gradual fading effect in analog mode using PWM.

**Apparatus:**

1. Arduino board (e.g., Arduino Uno or Mega)
2. LED
3. 220Ω resistor
4. Jumper wires
5. Breadboard
6. USB cable for Arduino connection

**Working Procedure:**

**Part 1: Digital Mode Blinking**

1. **Circuit Setup**: Connect the LED to a digital pin on the Arduino (e.g., pin 13) through a 220Ω resistor. Connect the other end of the LED to ground.
2. **Programming**: In the Arduino IDE, write a code that turns the LED on and off at a 1-second interval using digitalWrite() for digital control.

**Code**:

void setup() {

pinMode(13, OUTPUT); // Set pin 13 as an output

}

void loop() {

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

delay(1000); // Wait for 1 second

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

delay(1000); // Wait for 1 second

}

1. **Execution**: Upload the code to the Arduino and observe the LED blinking on and off every second.

**Part 2: Analog Mode Fading**

1. **Circuit Adjustment:**
   * Connect the LED to pin 13 on the Arduino with a 220Ω resistor between the LED’s longer leg (anode) and the pin.
   * Connect the LED’s shorter leg (cathode) to the ground (GND) on the Arduino.
2. **Programming:**
   * Use the following code to create a smooth fading effect by gradually increasing and then decreasing the LED brightness.
   * This code uses analogWrite() to adjust brightness levels from 0 to 255 in steps, with a delay of 10 milliseconds to achieve a smooth transition.

**Code**:

const int ledPin = 13; // Pin connected to the LED

void setup() {

pinMode(ledPin, OUTPUT); // Set the LED pin as an output

}

void loop() {

// Increase brightness

for (int brightness = 0; brightness <= 255; brightness++) {

analogWrite(ledPin, brightness); // Set the brightness

delay(10); // Wait for 10 milliseconds

}

// Decrease brightness

for (int brightness = 255; brightness >= 0; brightness--) {

analogWrite(ledPin, brightness); // Set the brightness

delay(10); // Wait for 10 milliseconds

}

}

**Execution:** Upload the code to the Arduino and observe the LED fading in and out smoothly as brightness levels gradually increase and decrease.

**Results:**

* **Digital Mode**: The LED blinked on and off at a 1-second interval, demonstrating basic control over digital outputs.
* **Analog Mode**: The LED smoothly faded in and out using PWM, showing how analogWrite() can control brightness levels to create an analog-like fade effect. This demonstrates how PWM enables varying output intensities on digital pins to simulate an analog output.

**Conclusion:**

This lab successfully showed how to control an LED using both digital and analog (PWM) modes on an Arduino. Digital output allowed precise on/off timing, while PWM enabled a gradual fade effect in analog mode. These techniques form a basis for controlling LEDs, motors, and other components that require variable output in Arduino-based projects.