Experiment No. : 02

Experiment Name:

Brightness control of LED using microcontroller.

Objectives:

In this lab we will learn how we can control the brightness of a LED using microcontroller. The objectives of this experiment include:

- Designing a microcontroller based system to control LED brightness
- Intensifying or dimming LED brightness using push buttons.

Theory:

Microcontroller:

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Microcontrollers are "embedded" inside some other device to control the features or actions of the device. Another name for a microcontroller, therefore, is "embedded controller." Microcontrollers are mostly designed for embedded applications and are heavily used in automatically controlled electronic devices such as cellphones, cameras, microwave ovens, washing machines, etc.

In our experiment we have used Arduino Uno R3. It is a microcontroller board based on a removable, dual-inline-package (DIP) ATmega328 AVR microcontroller. It has 20 digital input/output pins (of which 6 can be used as PWM outputs and 6 can be used as analog inputs).

LED (light-emitting diode):

A light-emitting diode (LED) is a two-lead semiconductor light source. A p–n junction diode emits light when activated. When a suitable voltage is applied to the leads, electrons are able to recombine with electron holes within the device, releasing energy in the form of photons.

LEDs only allow current to flow in one direction, and they are always polarized. A LED has two terminals. The positive side (longer leg) is called the anode, and the negative one (shorter leg) is called the cathode.

Current Limiting Resistor:

We should connect a current-limiting resistor when an LED is used or else the LED can become burned due to excessive current. In this experiment, the operating voltage of the LED is between 1.5V and 2.0V and the operating current is between 10mA and 20mA. The arduino Uno board can supply 5V power, the LED we choose works at 1.7V, 15mA. The current-limiting resistance equals total voltage subtracted by LED voltage, then divided by current. In this case, that would be (5-1.7)/0.015. Thus, the current-limiting resistance equals 220Ω.

Controlling LED Brightness:

We can control the brightness of an LED in two ways.

1. Change the amount of current going through the LED. We can do this by changing the size of the current limiting resistor connected with the LED.

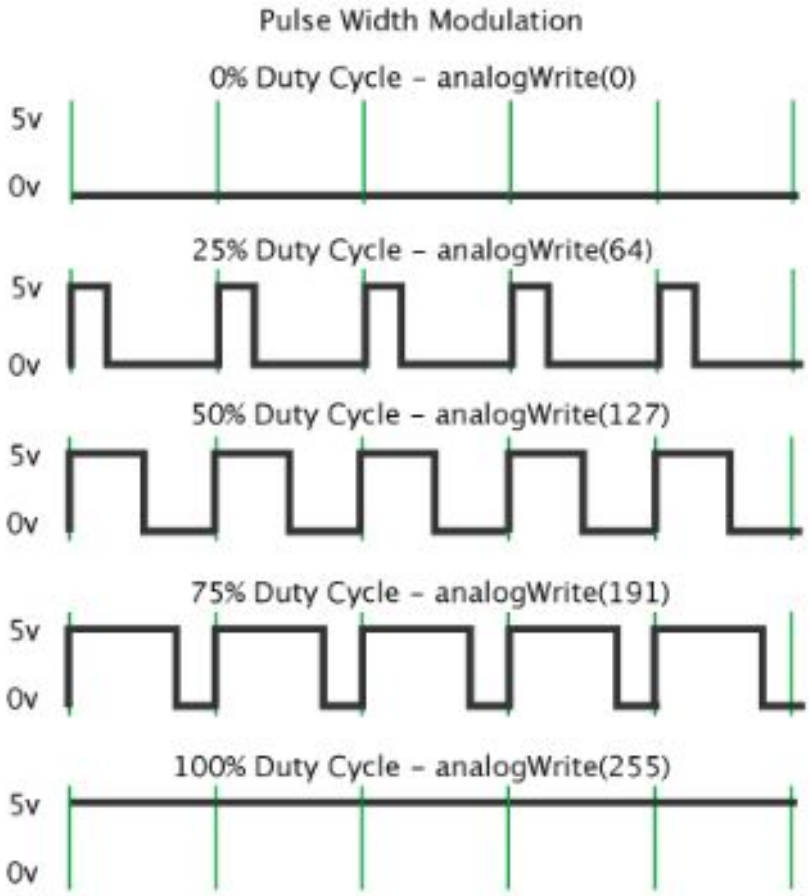
2. We can take advantage of the fact that human eyes can only see things that happen up to a certain speed, and turn the LED on and off faster than we can see. The more time that the LED is on in a given period of time, the "**brighter**" we think it is. The more time it is off, the "**dimmer**" we think it is. (This method supports smoother dimming over a broader range as opposed to changing resistors.)

It turns out that this method of turning things on and off quickly is very common, and a standard method has been designed called Pulse Width Modulation (**PWM** for short).

The Arduino supports **PWM** (on certain pins marked with a tilde(~) on your board - pins 3, 4,5,9,10 and 11) at 500Hz. (500 times a second.) You can give it a value between 0 and 255. 0 means that it is never 5V. 255 means it is always 5V. To do this you make a call to analogWrite() with the value. The ratio of "ON" time to total time is called the "duty cycle". A PWM output that is ON half the time is said to have a duty cycle of 50%.

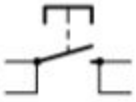
You can think of PWM as being on for x/255 where x is the value you send with analogWrite().

In the graphic below, the green lines represent a regular time period. This duration or period is the inverse of the PWM frequency. In other words, with Arduino's PWM frequency at about 500Hz, the green lines would measure 2 milliseconds each. A call to analogWrite() is on a scale of 0 - 255, such that analogWrite(255) requests a 100% duty cycle (always on), and analogWrite(127) is a 50% duty cycle (on half the time) for example.



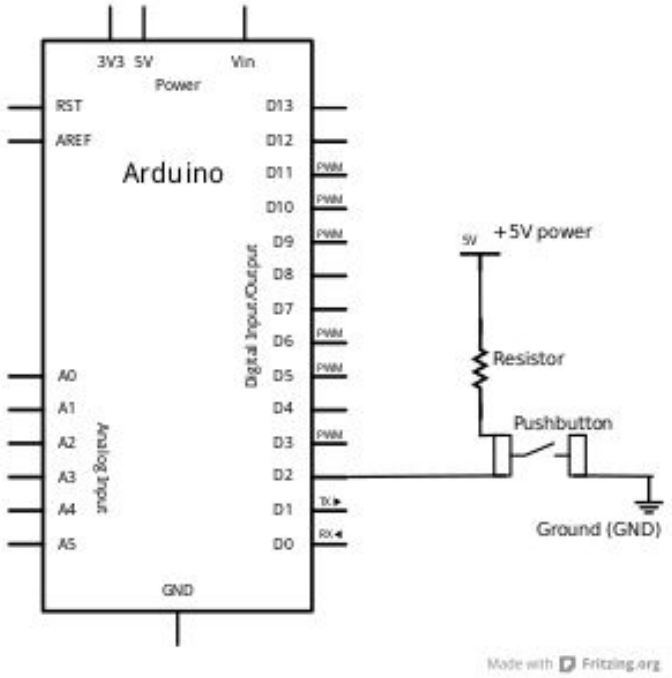
Push Button:

The pushbutton is a component that connects two points in a circuit when you press it. The example turns on an LED when you press the button. Pushing a button causes wires under the button to be connected, allowing current to flow. (called closed) When the button isn't pressed, no current can flow because the wires aren't touching (called open)



We connect three wires to the Arduino board. The first goes from one leg of the pushbutton through a pull-up resistor (here 2.2 KOhms) to the 5 volt supply. The second goes from the corresponding leg of the pushbutton to ground. The third connects to a digital i/o pin (here pin 7) which reads the button's state.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to 5 volts (through the pull-up resistor) and we read a HIGH. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to ground, so that we read a LOW. (The pin is still connected to 5 volts, but the resistor in-between them means that the pin is "closer" to ground.)
For electricity to flow, there has to be a complete circuit from the power to the ground. If a microcontroller pin is not connected to anything, it is said to be “floating” and you can’t know ahead of time what the value will be when you read it. It can also change between times that it is read. When we add a pull-up resistor, we get a circuit like the following:



When a push button is pushed down, the circuit is complete and ground is connected to pin 2. (The +5V goes through the closed switch to ground as well.) When it is not pushed down the circuit is from the +5V through the resistor and the microcontroller sees the +5V. (HIGH)

Code:

In the setup function we need to set pin of the push button to INPUT using the pinMode(pinNo,INPUT) function. Then we can use digitalRead(pinNo) function to check whether the push button is pressed or not. If the button is pressed then digitalRead(pinNo) function will return HIGH otherwise it will return LOW.

LCD Display:

LCD stands for Liquid Crystal Display. We will refer to it as either an LCD or simply a display.

In this experiment we used a LCD display to show when the LED brightness is intensifying or dimming as we press the push button.

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

A Read/Write (R/W) pin that selects reading mode or writing mode

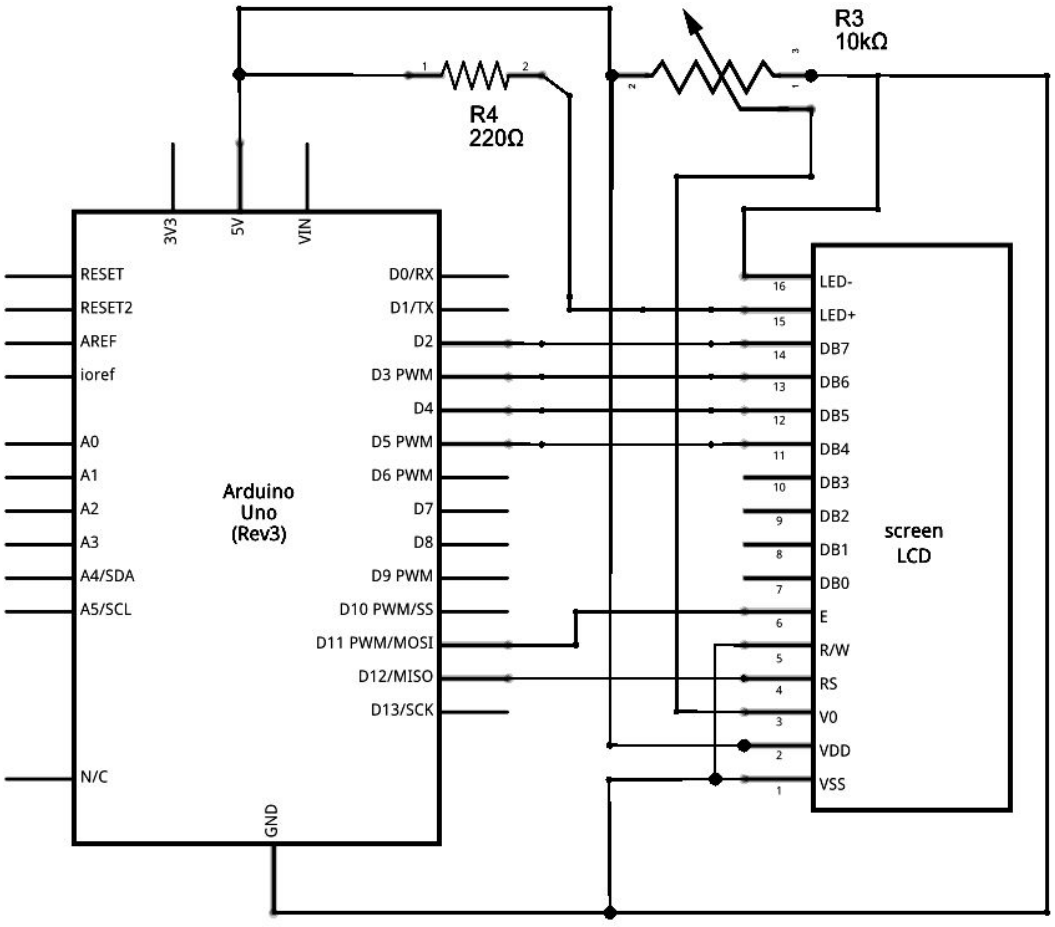
An Enable pin that enables writing to the registers

8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display constrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The LiquidCrystal Library simplifies this for you so you don't need to know the low-level instructions.

Circuit:



To wire your LCD screen to your board, connect the following pins:

LCD RS pin to digital pin 12
LCD Enable pin to digital pin 11
LCD D4 pin to digital pin 5
LCD D5 pin to digital pin 4
LCD D6 pin to digital pin 3
LCD D7 pin to digital pin 2

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

Equipment:

1.A computer (Windows, Mac, or Linux).
2.An Arduino compatible microcontroller.
2. Arduino Software
3. A breadboard
4. A USB A-to-B cable
5. Male/Male Jumper Wires
6. LED
7. Resistors(220 Ω)
8. Push Button
9. LCD Display
10. Potentiometer

Experiment:

Circuit Diagram:

Program Source Code:

Controlling the brightness of LED using microcontroller.

```
#include<LiquidCrystal.h>
const int pinHigh = 8;
const int pinLow = 9;
const int led = 6;
int brightness = 0;
int fadeAmount = 10;
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
int msg = 0;
void setup() {
  pinMode(pinHigh, INPUT);
  pinMode(pinLow, INPUT);
  pinMode(led, OUTPUT);
}

void loop() {


  if (digitalRead(pinHigh) == HIGH) {
    if(msg!=1)
    {
      lcd.clear();
      lcd.write("Intensifying LED");
      msg = 1;
    }
    else if(brightness==255)
    {
      lcd.clear();
      delay(1000);
    }
```

```
    brightness = brightness + fadeAmount;
  }
  else if (digitalRead(pinLow) == HIGH) {
    if(msg!=2)
    {
      lcd.clear();
      lcd.write("   Dimming LED");
      msg = 2;
    }



    else if(brightness==0)
    {
      lcd.clear();
      delay(1000);
    }
    brightness = brightness - fadeAmount;

  }
  else
  {
    lcd.clear();
    msg = 0;
  }
  brightness = constrain(brightness, 0, 255);
  analogWrite(led, brightness);
  delay(250);
}
```

Procedure:

Part1: Designing the circuit:

We designed the circuit by connecting the LED, push buttons, LCD display, potentiometer and resistors as defined above. We used two push button, one for intensifying the brightness of LED and another for dimming the brightness of LED. We used LCD display so that we can visualize when the LED is intensifying and when the LED is dimming.

Part 2: Connecting Arduino to the computer

We launched the Arduino IDE from the computer then created a new sketch. Then, we connected the Arduino UNO board with the computer through a USB A-to-B cable.

Part 3: Compiling and Uploading the sketch to the arduino

Now we need to write the source code for the experiment.

We included the LiquidCrystal.h library so that we can use the LCD display. Then we set up the pins of the LCD display as follows

RS to pin 12
EN to pin 11
D4 to pin 5
D5 to pin 4
D6 to pin 3
D7 to pin 2

We set pin 8 and pin 9 for the push button and pin 6 for the LED. We setup the pins in the setup() function. We also declared two variables brightness and fadeAmount to control the LED brightness

In the loop function, we checked which push button is pushed.

If the push button connected with pin 8 is pushed then we increase the brightness variable's value by fadeAmount and displayed in the LCD display "Intensifying LED". When the brightness variable's value reaches 255 we didn't show anything in the LCD display.

If the push button connected with pin 9 is pushed then we decrease the brightness variable's value by fadeAmount and displayed in the LCD display "Dimming LED". When the brightness variable's value reaches 0 we didn't show anything in the LCD display.

If no button is pushed then we cleared the LCD display.

Finally we use the analogWrite(led,brightness) function to lit the LED to our desired brightness level. We used a built in function constrain(brightness,0,255) to control the range of the brightness variable. The function is defined as follows.

```
int constrain(int value, int min, int max)
{
	if(value > max){
		value = max;
	}
	if(value < min){
		value = min;
	}
	return value;
}
```

Conclusion:

From this experiment, we learned about PWM(Pulse Width Modulation) and how we can control the brightness of a LED using PWM ports of the arduino.

We also used two push buttons to increase and decrease the brightness of LED and a LCD display to show whether the brightness of the LED is intensifying or dimming. We also used a potentiometer to control the contrast of the LCD display.

We observed how the brightness of the LED was changing as we pushed different push buttons and the LCD display showing whether LED is intensifying or dimming.

Finally, we were able to control the brightness of LED and completed the experiment successfully.