`

TAYLOR'S	SCHOOL OF ENGINEERING

ELECTRICAL AND ELECTRONIC ENGINEERING PROGRAMME

PROGRAMMING TECHNIQUES
(EEE 60104)

Group No.	**1**	**Total Marks**	
Lab No.	**1**		
Title of Lab	**Blinking LEDs**		

Lecturer	Dr. Chew Wei Jen	
Date	**Lab Date: 20th September 2023**	**Submitted Date: 3rd October 2023**

Lab Report Item/total marks	**Marks**
Format/10	
Abstract and Introduction/15	
Materials and Method/15	
General Results and Discussions/50	
Conclusion and References/Appendix /10	
Total /100	

No.	**Members ID**	**Name of Group Members**
1	0360231	Muhammad Rafif Edie Wijaya
2	0364144	Law Shing Yi
3	0361508	Wong Kai Tian
4	0349907	Syazani Zikry Zahari
5	0353217	Surryaraj A/L Poobalan

`

TAYLOR'S	SCHOOL OF ENGINEERING

Table of Contents:

`

1. Abstract

In this project, an Arduino Uno microcontroller is linked to five LEDs, and we're essentially building a C program to control those LEDs. The main goal is to show the blinking behaviour of each LED can be controlled to produce sequences of visual effects. To do this, we attach specific Arduino Uno pins to the LEDs and then write code to control those pins, allowing us to generate various flashing patterns. The Arduino Uno is a fantastic choice for those with an interest in electronics or those who are just getting started with microcontroller projects because it is versatile and user-friendly.

2. Introduction

In this experiment, we'll write a C program to manage a set of five LEDs that are connected to an Arduino Uno microcontroller's blinking. The aim of this program is to demonstrate how to control the individual LEDs' blinking patterns to create visual effects. The LEDs will be connected to specific pins on the Arduino Uno, and by manipulating these pins in the code, we can control the LEDs' behavior.

The Arduino Uno is a popular development board that can be programmed to interact with various electronic components, making it an excellent choice for beginners and hobbyists in the world of electronics and microcontrollers. It can be used for a wide range of projects, from simple LED blinking and temperature sensing to more complex robotics and applications. Its versatility and ease of use have made it a popular choice for both beginners and experienced electronics enthusiasts to bring their creative electronic ideas to life.

3. Methodology

a. Materials:

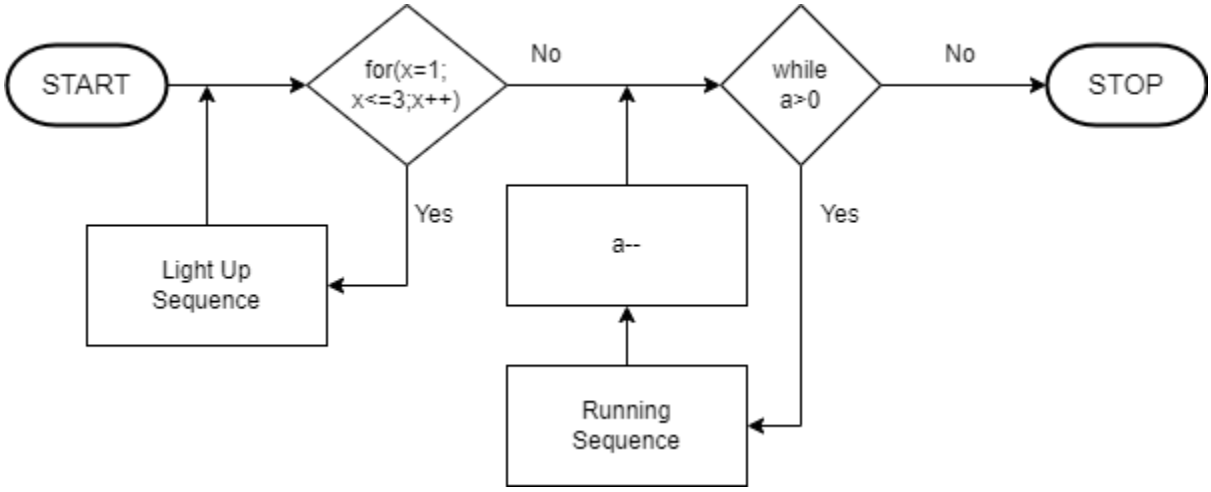
i. Arduino UNO

ii. LEDs

iii. 220Ω Resistors

iv. Multimeter

b. Methods:



c. Procedure:

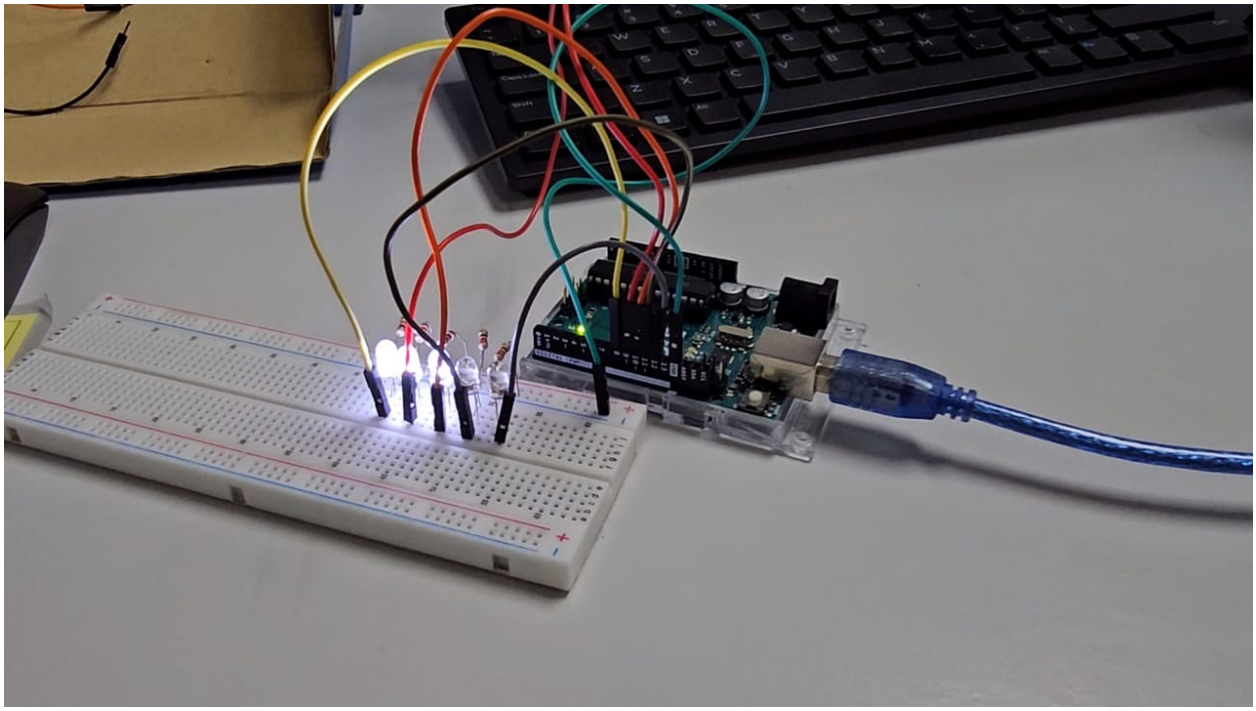
i. Build a circuit consisting of 5 LEDs arranged in a row that are connected to PortB Pin 1 to Pin 5 of an Arduino Uno board. The anode of the LED will be connected to the pins while the cathode of the LED will be connected to the ground through the resistors.

ii. Write a C program that controls the blinking of the LEDs following the pattern below:

1. Turn on each LED one by one, giving a delay of 1 second between each LED.
2. After all LEDs are turned on, turn off each LED one by one, again giving a delay of 1 second between each LED.

3. Repeat this pattern 3 times.
4. Lastly, create a running light sequence and have this pattern run continuously, with the speed of the lights changing and getting faster with each iteration.

4. Results



5. Discussion

a. The code for this lab consists of two main objectives. The first objective is to turn the set of LEDs one by one, with a delay of 1 second, and then turn them off one by one with a delay of 1 second. The second objective of the lab is to create a running light sequence where the delay time between each LED turning on gets faster with every iteration.

b. A similar function and two header files were used for both objectives of the lab. The first header file used was <*avr/io.h*>. This specific header file was used to

including the input and output pins of the Arduino. The other header file used was the *<util/delay.h>* which can be used to run the delay time between processes. The next part was defining the delay time as 0.1 s. This specific delay time was used due to the limitation of the lab setup. This delay time will be put into a function called *delayTime(int a)*, which will loop the 0.1 seconds and use the input value as the number of repetitions.

c. To turn on the LEDs, we used port manipulation [1], which can control each LED directly. The first thing that we did was to set the pins on port B of the Arduino. The register *DDRB = 0b111111* set the pins in port B to act as outputs for the Arduino. The pins 8 to 13 are controlled by the bits starting from the right, while the 2 most left bits control crystal pins, which are unusable [2].

d. To achieve the first objective of the lab, we placed the entire sequence inside a for loop. In this loop, we used x as the counter which starts from 1 and increases by one after each cycle. The loop ends after 3 repetitions of the sequence. Since port manipulation was used for this lab, the LEDs' ability to turn on or off was controlled by the PORT register, which in this case was PORTB. The control register PORTB, controls the LED by setting the intended pin to 1 for on and 0 for off. For this part of the lab, we used the function *delayTime*, with the input *a=10*, which results in a total time of 1 second between each LED turning on. So, PORTB starts with *0b00000001*, which turns on the LED on pin 8, and continues turning on all the other LEDs until PORTB = *0b00011111*, which indicates all LEDs are on. Then, it starts turning off each LED one by one until it reaches PORTB = *0b00000000*, which means all LEDs are off.

e. The next section of the lab was to create a running lab sequence. The objective was to turn one LED at a time and decrease the time between LEDs every round. To achieve this, we utilized a while loop. The parameter for this loop was set to *a>0*, where a is the same counter used for the *delayTime* function. In this loop,

after 1 sequence is complete, the parameter of this loop will be decreased by one, which will also decrease the time between each LED. Finally, when the parameter gets to *a=0*, it will end the *while* loop and at the same time, end the program.

6. Conclusion

For improvements, the program can be written in a more modular way for the light-up sequence and running sequence by using functions. This can allow a lesser line of codes when the program gets longer and longer. Overall, the program written can recreate the light-up sequence and running sequence given in the tasks. This indicates the program worked as expected, the experiment is a success.

7. Reference

[1] Arduino. "Port Manipulation." [Online]. Available: https://docs.arduino.cc/hacking/software/PortManipulation. Accessed: September 19, 2023.

[2] Arduino. "Pin Mapping for Arduino Uno." [Online]. Available: https://docs.arduino.cc/hacking/hardware/PinMapping168. Accessed: September 19, 2023.

8. Appendix

```
#include <avr/io.h> // header file for input output pins
#include <util/delay.h> // header file for delay

#define BLINK_DELAY_MS 100
int delayTime(int a);
int main (void)
{
   /* set pin PORTB for output*/
   DDRB =0b11111111;
   int x ;
   int a = 10;
    for (x = 1; x <= 3;x++ )
    {
      /* set pin high to turn LED on */
      PORTB=0b00000001;
      delayTime(a);
```

`


```
        /* set pin low to turn LED off */
        PORTB=0b00000011;
        delayTime(a);
        PORTB=0b00000111;
        delayTime(a);
        PORTB=0b00001111;
        delayTime(a);
        PORTB=0b00011111;
        delayTime(a);

        PORTB=0b00001111;
        delayTime(a);
        PORTB=0b00000111;
        delayTime(a);
        PORTB=0b00000011;
        delayTime(a);
        PORTB=0b00000001;
        delayTime(a);
        PORTB=0b0000000;
        delayTime(a);
    }

    while (a>0)
    {

     PORTB=0b00000001;
     delayTime(a);
     PORTB=0b00000010;
     delayTime(a);
     PORTB=0b00000100;
     delayTime(a);
     PORTB=0b00001000;
     delayTime(a);
     PORTB=0b00010000;
     delayTime(a);
     a--;
    }
}
int delayTime(int a)
```

`


```
{
 int b=1;

 for(b=1;b<=a;b++)
 {
   _delay_ms(BLINK_DELAY_MS);
 }
}
```