**A logo of a university

Description automatically generated**

**AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH**

**Faculty of Engineering**

**LAB REPORT**

**Experiment # 03**

**Experiment Title:**

Familiarization of assembly language program and Interrupts in a microcontroller.

|  |  |  |  |
| --- | --- | --- | --- |
| **Date of Perform:** | 2 March 2024 | **Date of Submission:** | 20 March 2024 |
| **Course Title:** | MICROPROCESSOR AND EMBEDDED SYSTEMS LAB | | |
| **Course Code:** | COE3104 | **Section:** | M |
| **Semester:** | Spring 2023-24 | **Degree Program:** | BSc in CSE |
| **Course Teacher:** | **Prof. Dr. Engr. Muhibul Haque Bhuyan** | | |

**Declaration and Statement of Authorship:**

1. I/we hold a copy of this Assignment/Case Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case Study is my/our original work and no part of it has been copied from any other student’s work or from any other source except where due acknowledgment is made.
3. No part of this Assignment/Case Study has been written for me/us by any other person except where such collaboration has been authorized.

by the concerned teacher and is acknowledged in the assignment.

1. I/we have not previously submitted or currently submitting this work for any other course/unit.
2. This work may be reproduced, communicated, compared, and archived to detect plagiarism.
3. I/we permit a copy of my/our marked work to be retained by the Faculty Member for review by any internal/external examiners.
4. I/we understand that Plagiarism is the presentation of the work, idea, or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offense that may lead to expulsion from the University. Plagiarized material can be drawn from, and presented in, written, graphic, and visual forms, including electronic data, and oral presentations. Plagiarism occurs when the origin of the source is not appropriately cited.
5. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or copy my/our work.

*\* Student(s) must complete all details except the faculty use part.*

\*\* Please submit all assignments to your course teacher or the office of the concerned teacher.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Group # 01** | | | | | | |
|  |  |  |  | | |  |
| **Sl No** | **Name** | **ID** | **PROGRAM** | | | **SIGNATURE** |
| 1 | Moumet Shakhawat Siddiq | 20-42561-1 | BSc in CSE | | |  |
| 2 | MD. Salauddin Shanto | 19-40421-1 | BSc in CSE | | |  |
| 3 | Alin Das | 22-46379-1 | BSc in CSE | | |  |
| 4 | Asmaul Husna Mayad | 22-46413-1 | BSc in CSE | | |  |
| 5 | Moizuddin Mohammad Mujahid Tashid | 22-46398-1 | BSc in CSE | | |  |
| ***Faculty use only*** | | | | | | | |
| FACULTY COMMENTS | | | | **Marks Obtained** |  | | |
|  | | | |

Table Of Contents:

1. Title: ……………………………………………………………………………………………. 3
2. Objectives: ……………………………………………………………………………………… 3
3. Equipment List: ………………………………………………………………………………… 3
4. Circuit Diagrams: ...…………………………………………………………………………….. 3
5. Simulation Methodology and Setup: …………………………………………………………… 3
6. Assembly Codes and Explanation: ……………………………………………………………... 5
7. Experimental Outcomes: .......…………………………………………………………………... 6
8. Simulation Outcome: …………………………………………………………………………… 8
9. Discussion: .......………………………………………………………………………………… 10
10. Reference: ........…………………………………………………………………………………. 10

Experiment Title: Familiarization of assembly language program in a microcontroller.

# Objectives:

The objectives of this experiment are to-

1. Familiarize with the Assembly language
2. Implement a simple circuit to make LED lights blink using assembly programs
3. Implement a simple traffic control system from switch status using the I/O ports of an Arduino microcontroller
4. Simulate a microcontroller-based system using Proteus

# Equipment List:

# Arduino Uno

# Arduino IDE

# One LED

# One 220 Ω Resistor

# PC having Intel Microprocessor

# Circuit Diagram:

Explain the circuit diagram here for Figs. 1 and 2.

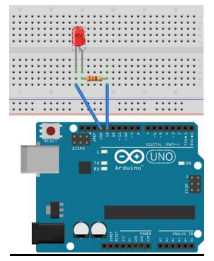


Figure 1: Hardware Circuit for LED Blink Test

**Simulation Methodology and Setup:**

The simulations are done using the Proteus 8.11 professional software.

1. Firstly, the proper components for the LED system were chosen and placed as shown in the circuit diagram.

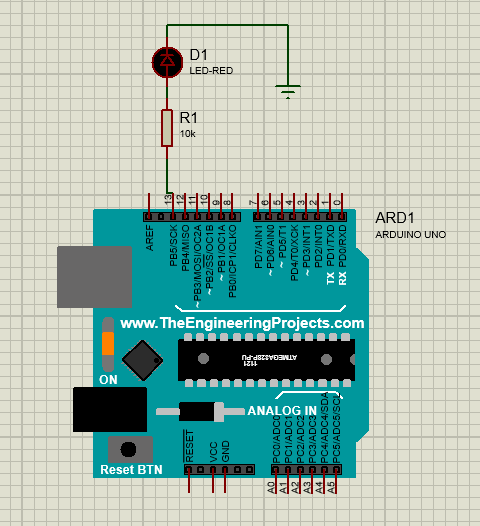


Figure 2: Simulation Set-up for LED System

1. Then for running the code in the Arduino a HEX filetype is created by going to “sketch” and then selecting “Export Compiled Binary” from the Arduino IDE software.
2. Then the “.hex” file is exported to the Arduino in proteus. By double clicking on Arduino a dialog box named “Edit Component” occurred on screen where the file is exported to “Program File”.

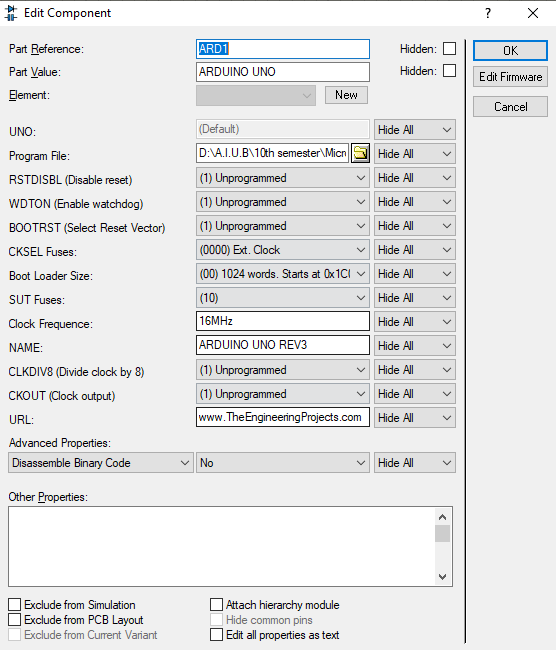


Figure 3: Exporting the code from IDE software to Proteus

1. Finally, the simulation is run on proteus and outcomes are observed.

**Assembly Codes and Explanation:**

**Blink an LED:**

The .ino file:

// C Code for Blinking LED

//-------------------------

extern "C"

{

void start();

void led(byte);

}

//----------------------------------------------------

void setup()

{

start();

}

//----------------------------------------------------

void loop()

{

led(1);

led(0);

}

The .S File:

;---------------

; Assembly Code

;---------------

#define \_\_SFR\_OFFSET 0x00

#include "avr/io.h"

;------------------------

.global start

.global led

;------------------------

start:

SBI DDRB, 5 ; set PB5 (D13) as o/p

RET ; return to setup() function

;---------------------------------------------------------------------------

led:

CPI R24, 0x00 ; value in R24 passed by caller compared with 0

BREQ ledOFF ; jump (branch) if equal to subroutine ledOFF

SBI PORTB, 5 ; set D13 to high

RCALL myDelay ; Calling a delay function

RET ; return to loop() function

;---------------------------------------------------------------------------

ledOFF:

CBI PORTB, 5 ;

RCALL myDelay

RET ; return to loop() function

;---------------------------------------------------------------------------

.equ delayVal, 10000 ; initial count value for the inner loop

;---------------------------------------------------------------------------

myDelay:

LDI R20, 100 ; initial count value for the outer loop

outerLoop:

LDI R30, lo8(delayVal) ; low byte of delayVal in R30

LDI R31, hi8(delayVal) ; high byte of delayVal in R31

innerLoop:

SBIW R30, 1 ; subtract 1 from 16-bit value in R31, R30

BRNE innerLoop ; jump if countVal not equal to 0

;--------------

SUBI R20, 1 ; subtract 1 from R20

BRNE outerLoop ; jump if R20 not equal to 0

RET

;---------------------------------------------------------------------------

FOR PUSH BUTTON LED CONTROL :

**The btnLED.ino file:**

//-----------------------------------

// C Code: RGB LED ON/OFF via Buttons

//-----------------------------------

extern "C"

{

void start();

void btnLED();

}

//-----------------------

void setup()

{

start();

}

//-----------------------

void loop()

{

btnLED();

}

**The btnLED.S file:**

;------------------------------------------

; Assembly Code: RGB LED ON/OFF via Buttons

;------------------------------------------

#define \_\_SFR\_OFFSET 0x00

#include "avr/io.h"

;------------------------

.global start

.global btnLED

;================================================================

start:

SBI DDRB, 4; set PB4 (pin D12 as o/p - red LED)

SBI DDRB, 3; set PB3 (pin D11 as o/p - green LED)

SBI DDRB, 2; set PB2 (pin D10 as o/p - blue LED)

CBI DDRD, 2; clear PD2 (pin D02 as i/p - red button)

CBI DDRD, 3; clear PD3 (pin D03 as i/p - green button)

CBI DDRD, 4; clear PD4 (pin D04 as i/p - blue button)

RET

btnLED:

L2: SBIS PIND, 4 ; Skips below statement if the push button of D04 is not pressed

RJMP L1

SBI PORTB, 2 ; Turn ON LED, PB2(D10), if SW of D04 is not pressed

CBI PORTB, 3 ; Turn OFF LED, PB3(D11), if SW of D04 is not pressed

SBIC PIND, 4 ; Skips below statement if the push button of D04 is pressed

RJMP L2

L1: CBI PORTB, 2 ; Turn OFF LED, PB2(D10), if SW of D04 is pressed

SBI PORTB, 3 ; Turn OFF LED, PB3(D11), if SW of D04 is pressed

RET

PUSH BUTTON LED CONTROL :

bool LED\_State = ‘True’;

void setup() {

pinMode(13, OUTPUT);

cli(); // stop interrupts till we make the settings

TCCR1A = 0; // Reset the entire A and B registers of Timer1 to make sure that

TCCR1B = 0; // we start with everything disabled.

TCCR1B = 0b00000100; // Set CS12 bit of TCCR1B to 1 to get a prescalar value of 256.

TIMSK1 = 0b00000010; // Set OCIE1A bit to 1 to enable compare match mode of A reg.

OCR1A = 31250; // We set the required timer count value in the compare register, A

sei(); // Enable back the interrupts

}

void loop() {

// put your main code here, to run repeatedly.

}

// With the settings above, this ISR will trigger each 500 ms.

ISR(TIMER1\_COMPA\_vect) {

TCNT1 = 0; // First, set the timer back to 0 so that it resets for the next interrupt

LED\_State = !LED\_State; // Invert the LED State

digitalWrite(13, LED\_State); // Write this new state to the LED connected to pin D5

}

**Experimental Outcome:**

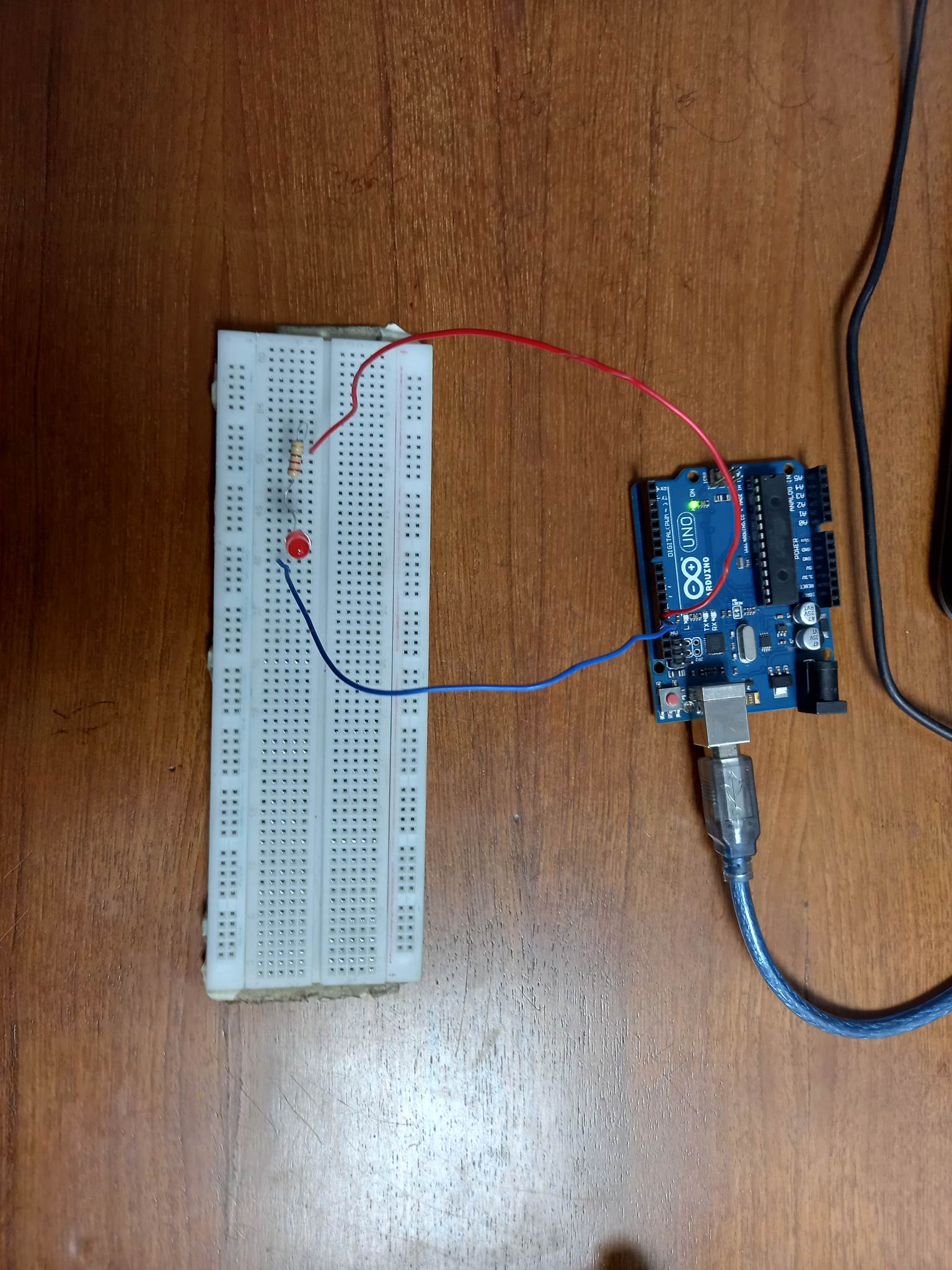
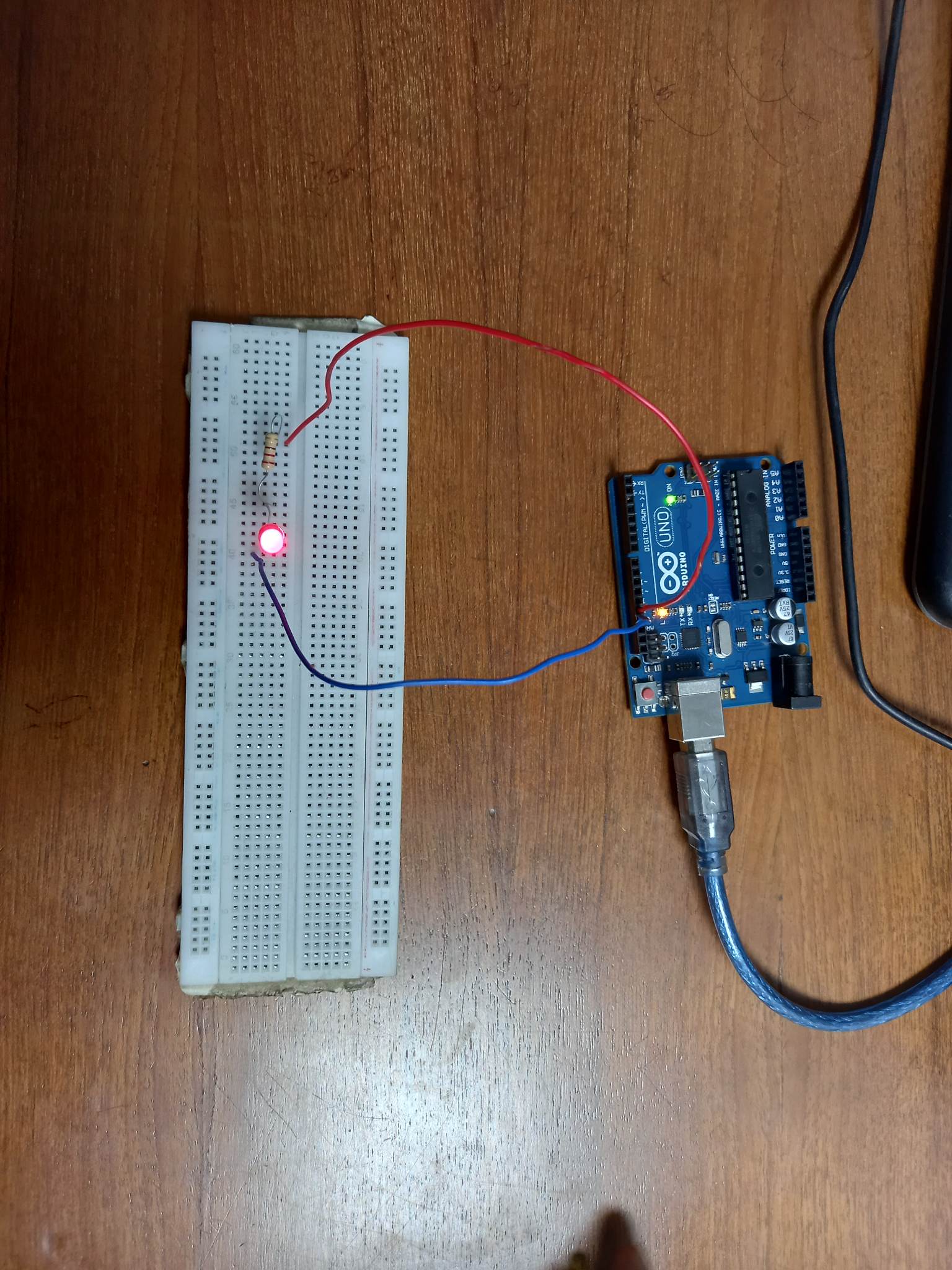
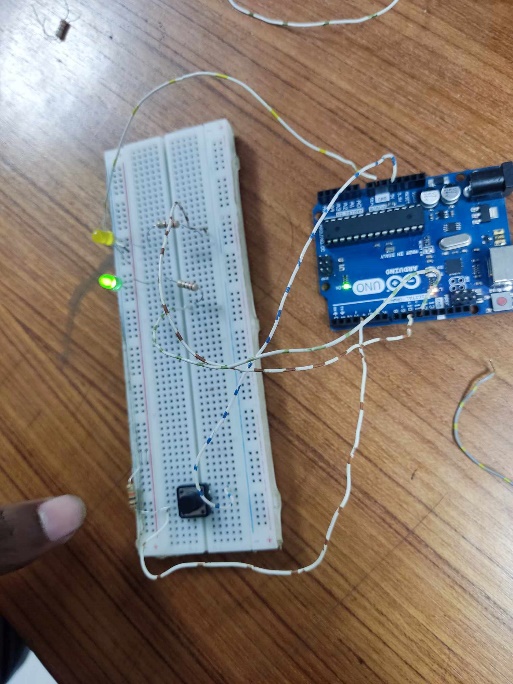
 

Figure 4: LED is OFF Figure 5: LED is ON

**A white circuit board with wires on a wooden surface

Description automatically generatedA hand holding a wire to a circuit board

Description automatically generated**

Figure 4: LED is OFF Figure 5: LED is ON Figure 6: LED ON and OFF

**Simulation Outcome:**

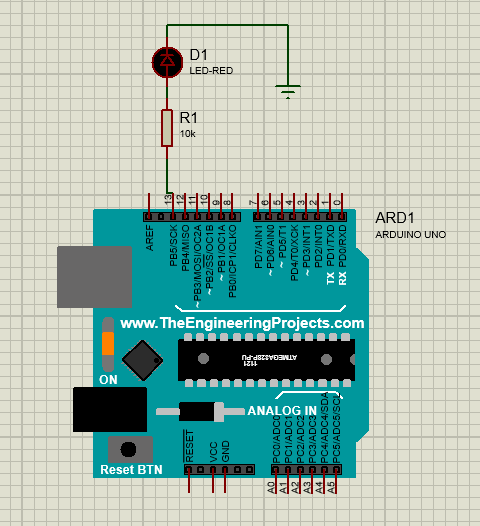
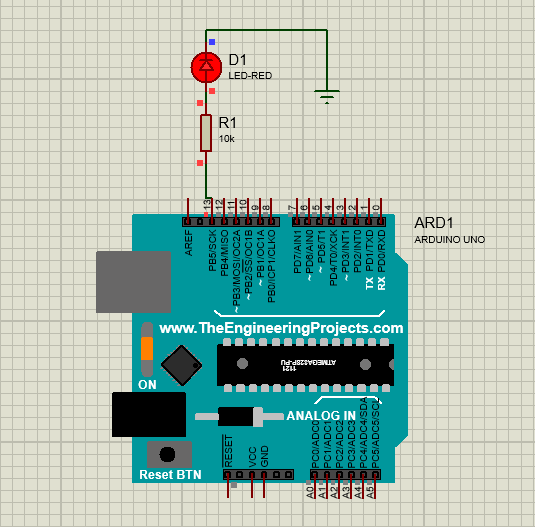
** **

Figure 6: LED is OFF Figure 7: LED is ON

# Discussion:

# Familiarizing oneself with assembly language programming in a microcontroller is invaluable for gaining direct access to hardware, optimizing performance, and ensuring real-time responsiveness. This low-level approach enables precise control over resource usage and facilitates debugging and troubleshooting by providing insights into machine-level execution. Understanding assembly language fosters proficiency in fundamental computer architecture concepts, such as memory management and interrupt handling. Additionally, the skills acquired in assembly language programming are transferable across different microcontroller architectures, promoting versatility and adaptability in embedded systems development. Ultimately, mastering assembly language empowers developers to create efficient and reliable solutions for a wide range of embedded applications.

# References:

1. List the references here.
2. Mainly books, conference papers, and journals papers are good.