



# **Namal University**

## **Mianwali**

### **Department of Electrical Engineering**

EE-252L: Introduction to Embedded Systems

#### **Lab Manual: 08**

#### **AVR Programming in C Language I/O Operations and Timer Introduction**

<b>Students Name</b>	<b>Fahim Ur Rehman Shah</b>
<b>Roll Number</b>	<b>NIM-BSEE-2021-24</b>
<b>Submission Date</b>	<b>6-8-2023</b>
<b>Marks Obtained</b>	

**Instructors: Dr. Hamza Zad Gul**

## Objectives

In this lab, the student will learn about logic programming and I/O port interfacing to configure the programmer and Microchip studio to program atmega328p.

## Course Learning Outcomes

CLO1: Practice the correct use of programming constructs of assembly language

CLO2: Construct systems by interfacing AVR peripherals

CLO3: Perform the assigned task individually/as a team effectively

CLO4: Report the outcomes of task performed effectively in oral and written form

## Software

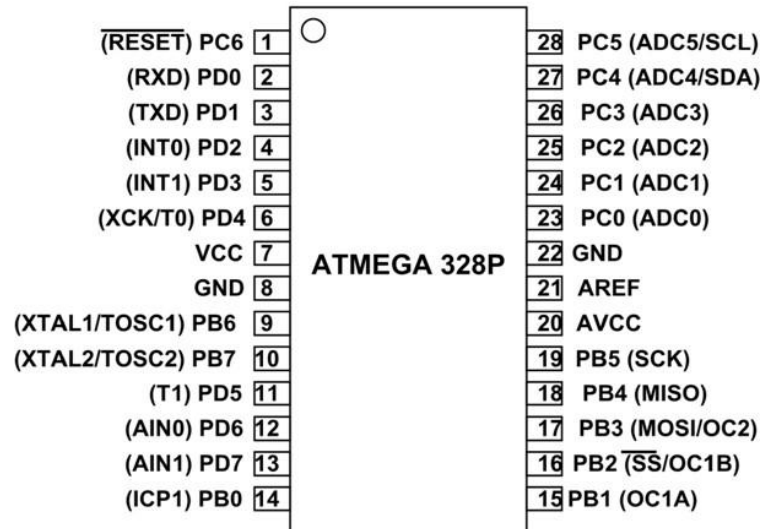
- Microchip studio

## Hardware

- Atmega 328p
- Atmega328p USBasp programmer Board
- Breadboard
- Connecting wires
- LEDs
- Resistors
- Capacitors
- Crystal oscillator
- Push buttons
- Oscilloscope

## Instructions

- You must submit the lab report complete within given deadline.
- Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom.
- Multiple such incidents will result in disciplinary action being taken.



## Introduction:

### Timers/Counters:

Timers/counters modules are used to generate precise time delay. They are separate hardware in a microcontroller which operates independently of microcontrollers. The Atmega328p has three timers timer0, timer1, timer2. These can be configured in different modes to generate precise delays and PWM signal.

### Timer0 (Normal Mode):

The goal is to generate 500ms delay with timer zero. The timer0 is an 8-bit timer which means it can count up to 255. The operation of timer0 is pretty straight forward it increment TCNT0 register on every clock cycle. TCNT0 is an 8-bit register and holds the timer count. When it reaches up to the value of 255 then roll back timer count and sets the timer overflow flag(TOV). To run a timer from a specific value, we can use output compare registers (OCR0).

Timer/Counter 0

TCNT0							
D7	D6	D5	D4	D3	D2	D1	D0

The configuration of timer is controlled by TCCR0b register which have different flag. These flags can be set to achieve the desired behavior. For normal mode we need to set following two flags.

1. The Frequency of the Clock Source with CS02, CS01, CS00 bits.
2. The mode of the timer.

Timer Counter Control Register 0

TCCR0							
D7	D6	D5	D4	D3	D2	D1	D0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

D2	D1	D0	Clock Source
CS02	CS01	CS00	Freq
0	0	0	No Clock (Stopped)
0	0	1	Clk
0	1	0	Clk/8
0	1	1	Clk/64
1	0	0	Clk/256
1	0	1	Clk/1024
1	1	0	Clk/T0-Falling edge
1	1	1	Clk/T0-Rising Edge

D6	D3	PWM
WGM00	WGM01	Mode
0	0	Normal
0	1	CTC (Clear timer on compare match)
1	0	PWM (Phase correct)
1	1	Fast PWM

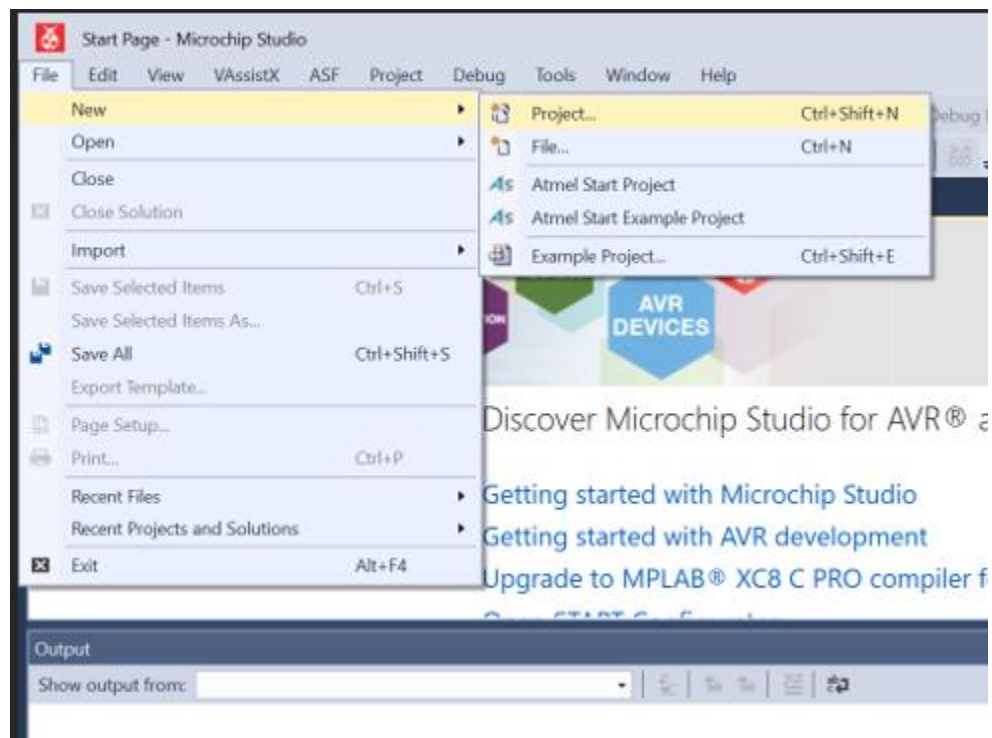
The Timer/counter Interrupt Flag Register (TIFR) holds the two basic flags we need the TOV and OVF.

Timer/Counter 0 Flag Register

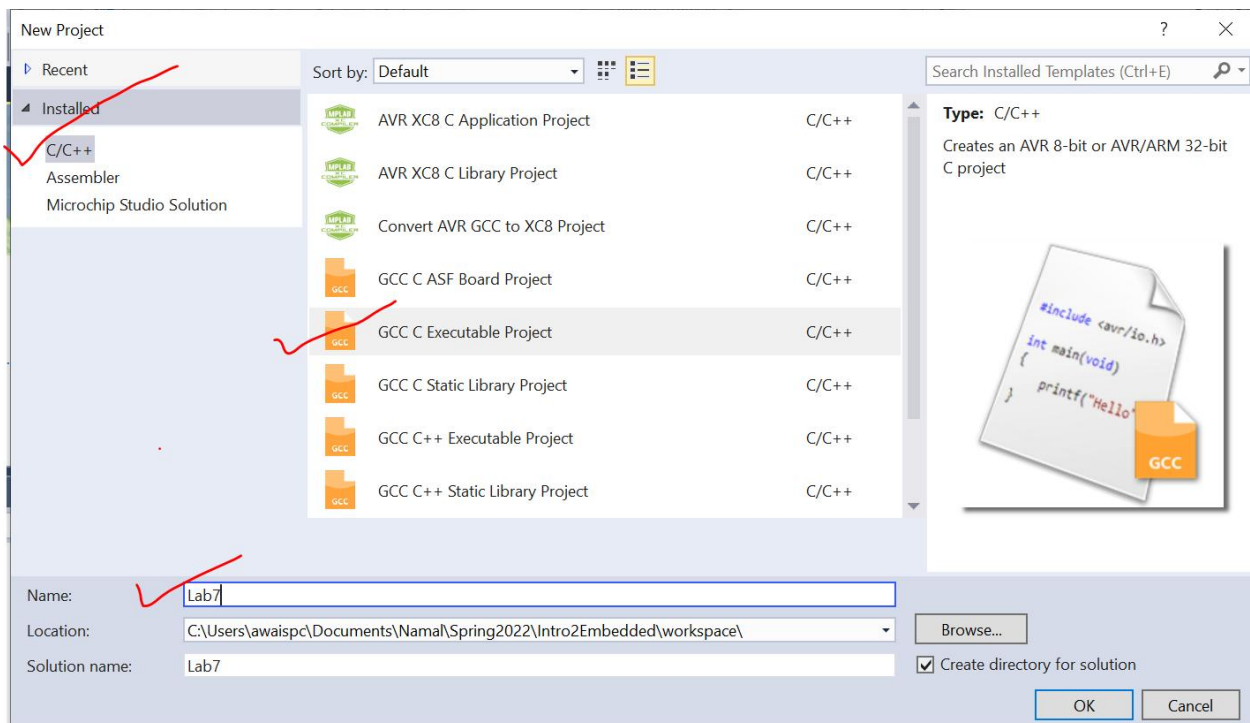
TIFR							
D7	D6	D5	D4	D3	D2	D1	D0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

## Lab Task I

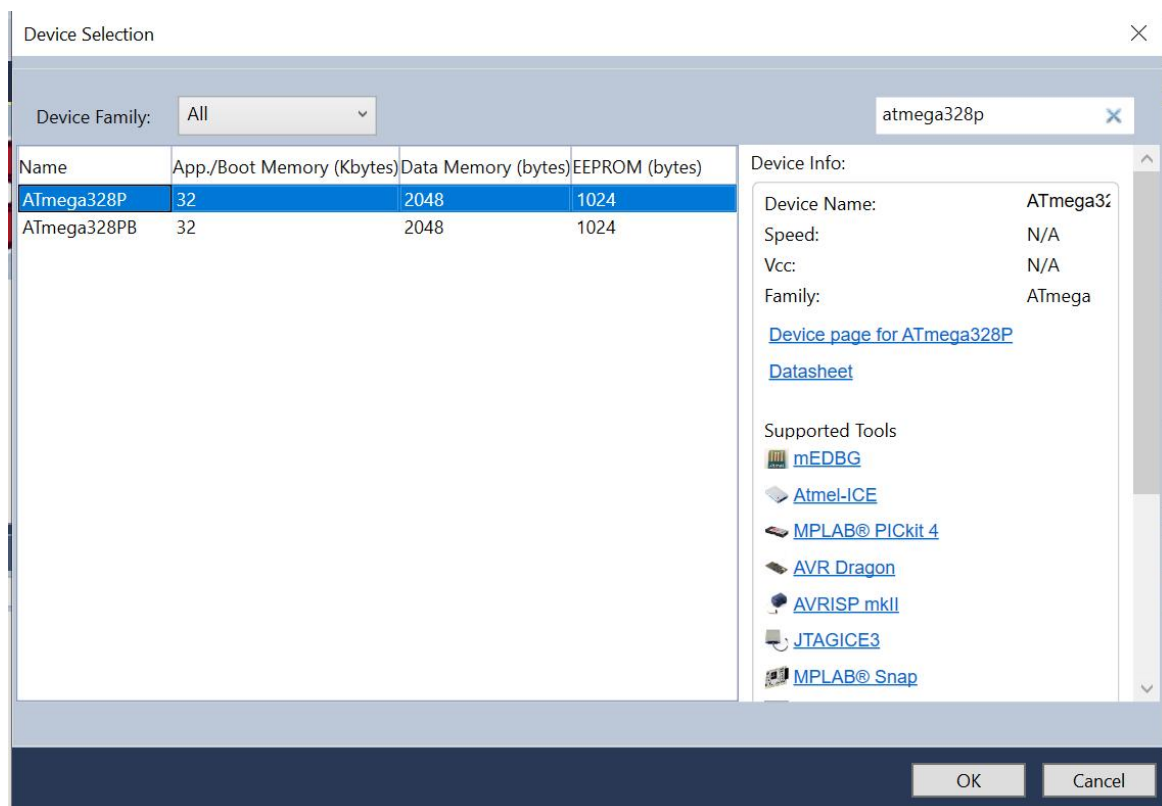
- I. Go to File → New → Project as shown below



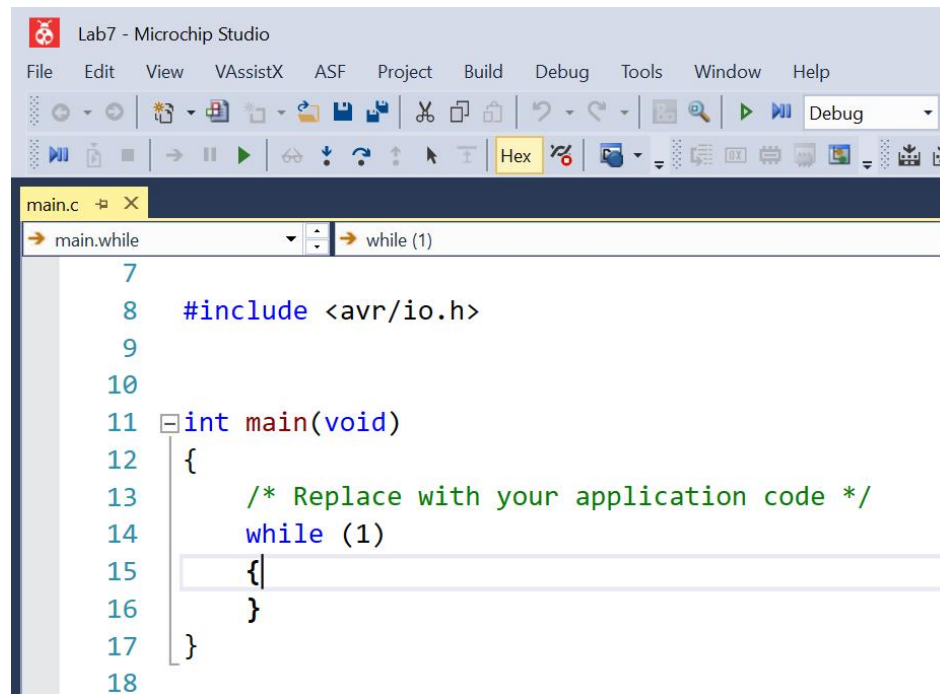
## 2. Select C/C++ and GCC C Executable Project



## 3. Select the Device you are using.



4. The project will look like this



5. You can start writing your code.

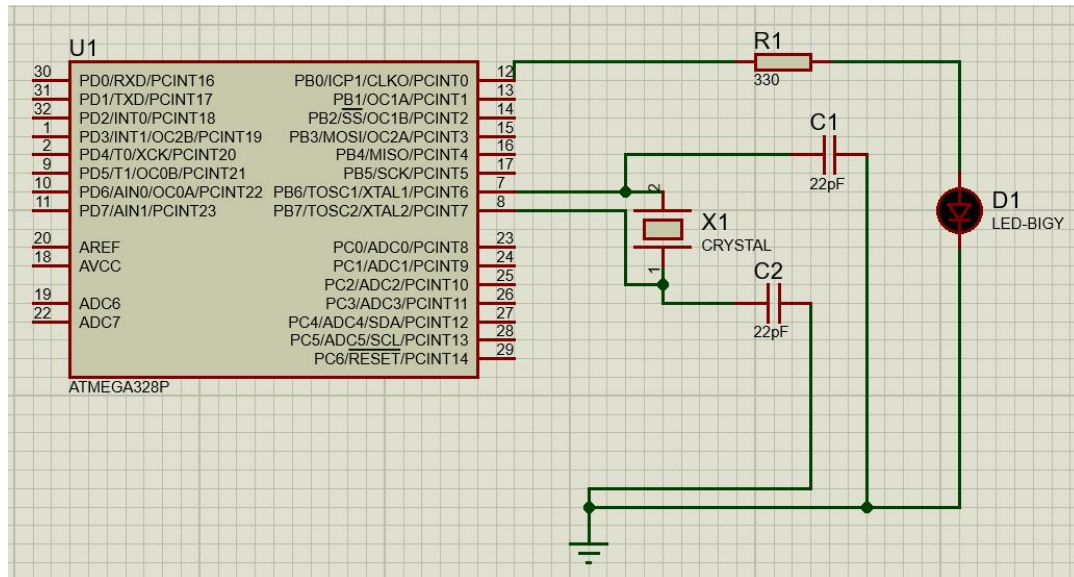
**Lab Task 1: Write the following code in main.c file.**

```
#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>

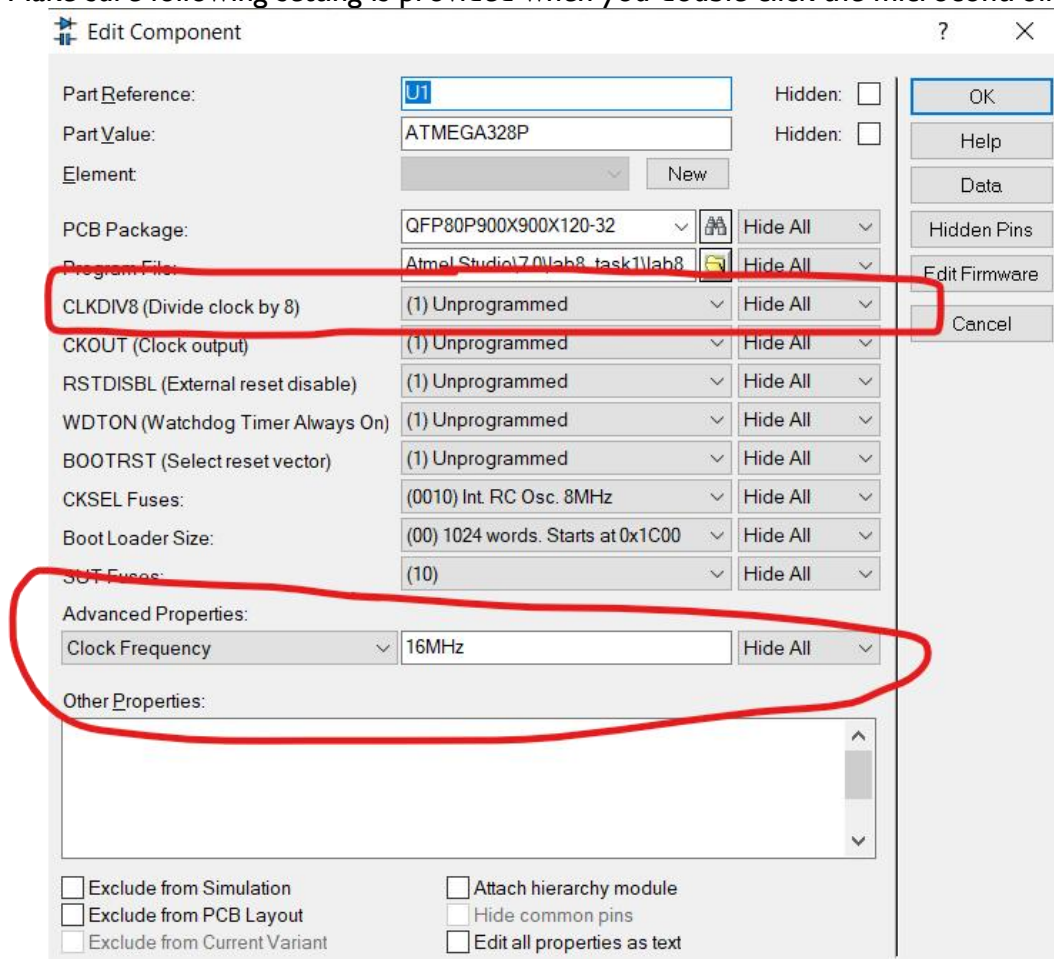
int main(void)
{
    DDRB = 0x01;

    while (1)
    {
        PORTB= 0x01;
        _delay_ms(500);
        PORTB = 0;
        _delay_ms(500);
    }
}
```

Construct the following circuit in proteus and hardware. Run your code and show it to instructor.



Make sure following setting is provided when you double click the microcontroller in proteus.



### Lab Task 3:

Modify the above code so that when a push button at PC0 send a LOW signal the LED at PBI turns ON otherwise the LED is OFF. Write the code in microchip studio and implement the circuit on proteus and show the working to the lab instructor. Also implement this on hardware. Attach your code, proteus circuit and breadboard implemented circuit below

### Code:

```
/*
 * Lab_08_Task_03.cpp
 *
 * Created: 6/6/2023 1:46:23 PM
 * Author : fahim
 */

#include <avr/io.h> // Include the AVR IO library for accessing registers and I/O operations
#define F_CPU 16000000UL // Define the CPU frequency

int main(void)
{
    DDRB = 0b00000010; // Set PORTB pin 1 as output
    DDRC = 0x00; // Set PORTC as input

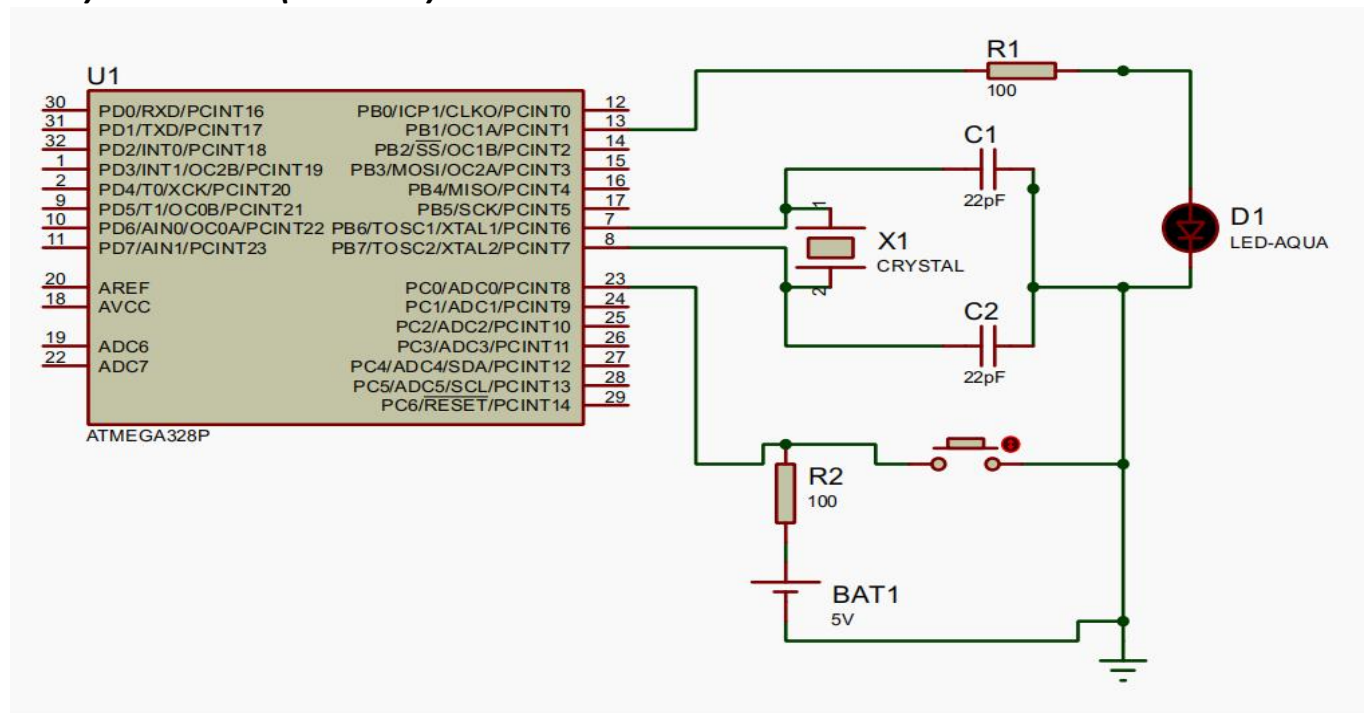
    while(1) // Infinite loop to continuously monitor the input pin
    {
        if(PINC & 0b00000001) // If the least significant bit of PINC is high
        {
            PORTB = 0x00; // Set PORTB pin 1 to low (turn off the LED)
        }
        else
        {
            PORTB = 0b00000010; // Set PORTB pin 1 to high (turn on the LED)
        }
    }

    return 0;
}
```

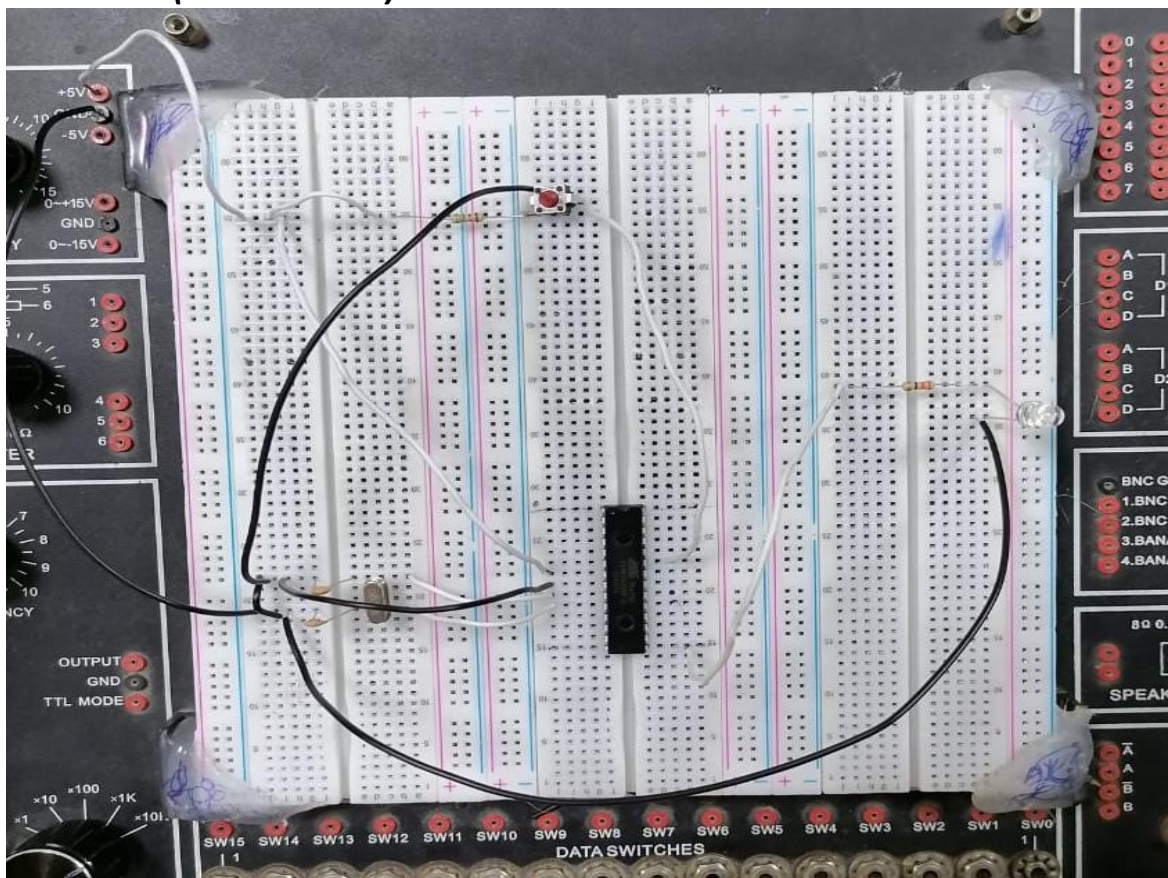
### Circuit :



## 1) Simulation ( Proteus ) :



## 2) Hardware ( Breadboard ) :



#### Lab Task 4: Write the following code in main.c file

```
#include <avr/io.h>
#define F_CPU 16000000L

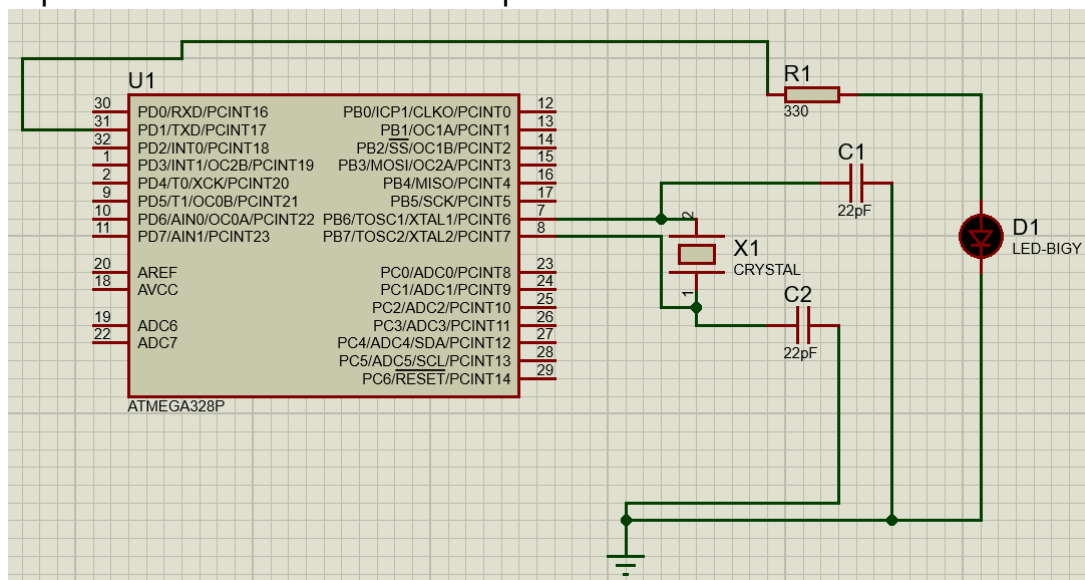
void T0Delay();

int main()
{
    DDRD = 0x02;

    while (1)
    {
        T0Delay ();
        PORTD = PORTD ^ 0x02;
    }
}

void T0Delay()
{
    TCNT0 = 0;
    TCCR0B = 0x01;
    while ((TIFR0 & (1 << TOV0)) == 0);
    TCCR0A = 0;
    TIFR0 = 0x01;
}
```

Implement the circuit for this code in proteus as shown below



#### Lab Task 5:

Generate the square waves of 25%, 50% and 75% duty cycle using timer function and display the waveform across oscilloscope. Use PortB pin 0, 1 and 2 for output and complete this task in a single program. Implement the circuit on the breadboard and show the results on oscilloscope to lab instructor. Attach your code, proteus simulation, hardware and oscilloscope results below.

### Code :

```
/*
 * Lab_08_Task_05.cpp
 *
 * Created: 6/6/2023 2:54:25 PM
 * Author : fahim
 */

#include <avr/io.h> // Include the AVR IO library for accessing registers and I/O operations
#define F_CPU 16000000UL // Define the frequency

void T0Delay(); // Function prototype for delay using Timer0

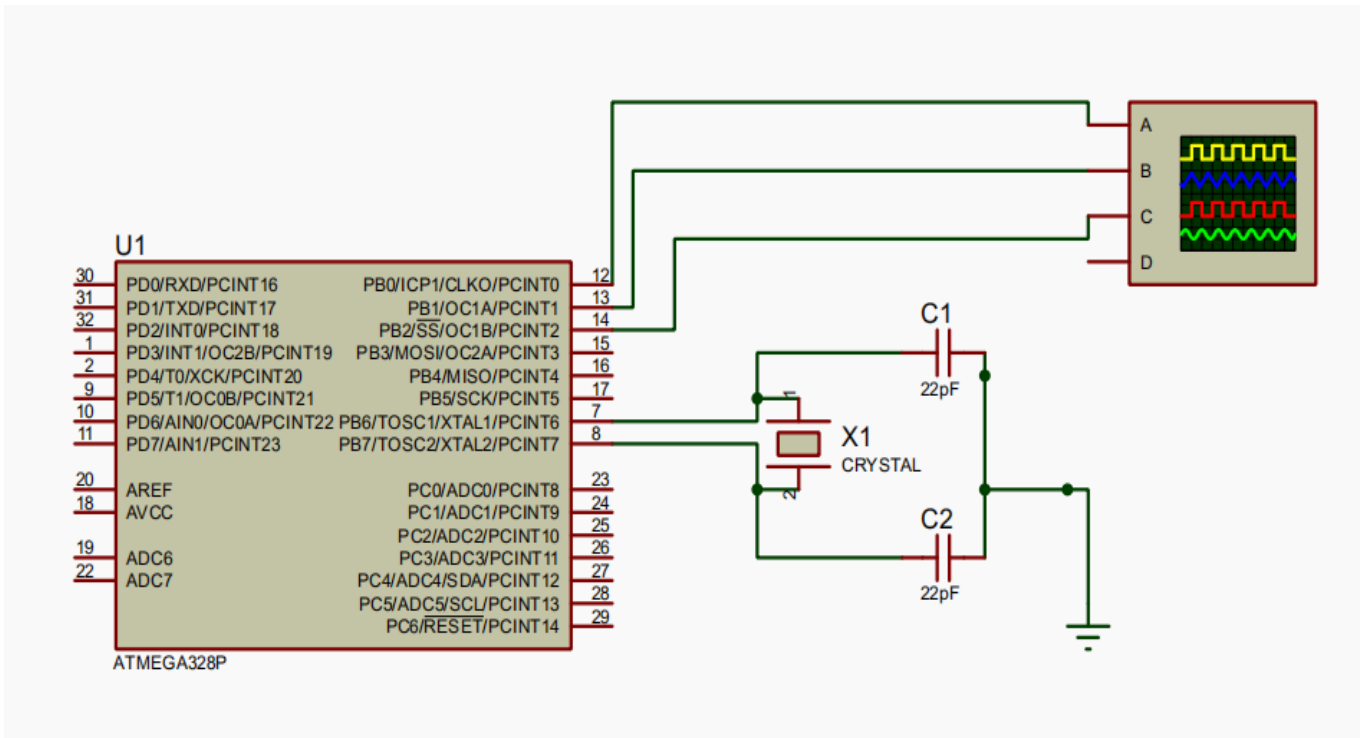
int main(void)
{
    DDRB = 0b00000111; // Set PORTB pins 0, 1, and 2 as output

    while (1) // Infinite loop for Square Wave
    {
        PORTB = 0b00000111; // Set PORTB pins 0, 1, and 2 to high
        T0Delay(); // Delay of ( 25% of the square wave ) using Timer0
        // Set PORTB pins 1 and 2 to high, pin 0 to low ( Pin 0 will give 25% duty cycle square wave )
        PORTB = 0b00000110;
        T0Delay(); // Delay of ( 25% of the square wave ) using Timer0
        // Set PORTB pin 2 to high, pin 0 and pin 1 to low ( Pin 1 will give 50% duty cycle square wave )
        PORTB = 0b00000100;
        T0Delay(); // Delay of ( 25% of the square wave ) using Timer0
        // Set PORTB pins 1 , 2 and 0 to low ( Pin 2 will give 75% duty cycle square wave )
        PORTB = 0b00000000;
        T0Delay(); // Delay of ( 25% of the square wave ) using Timer0
        // Now one Square wave is completed
    }
}

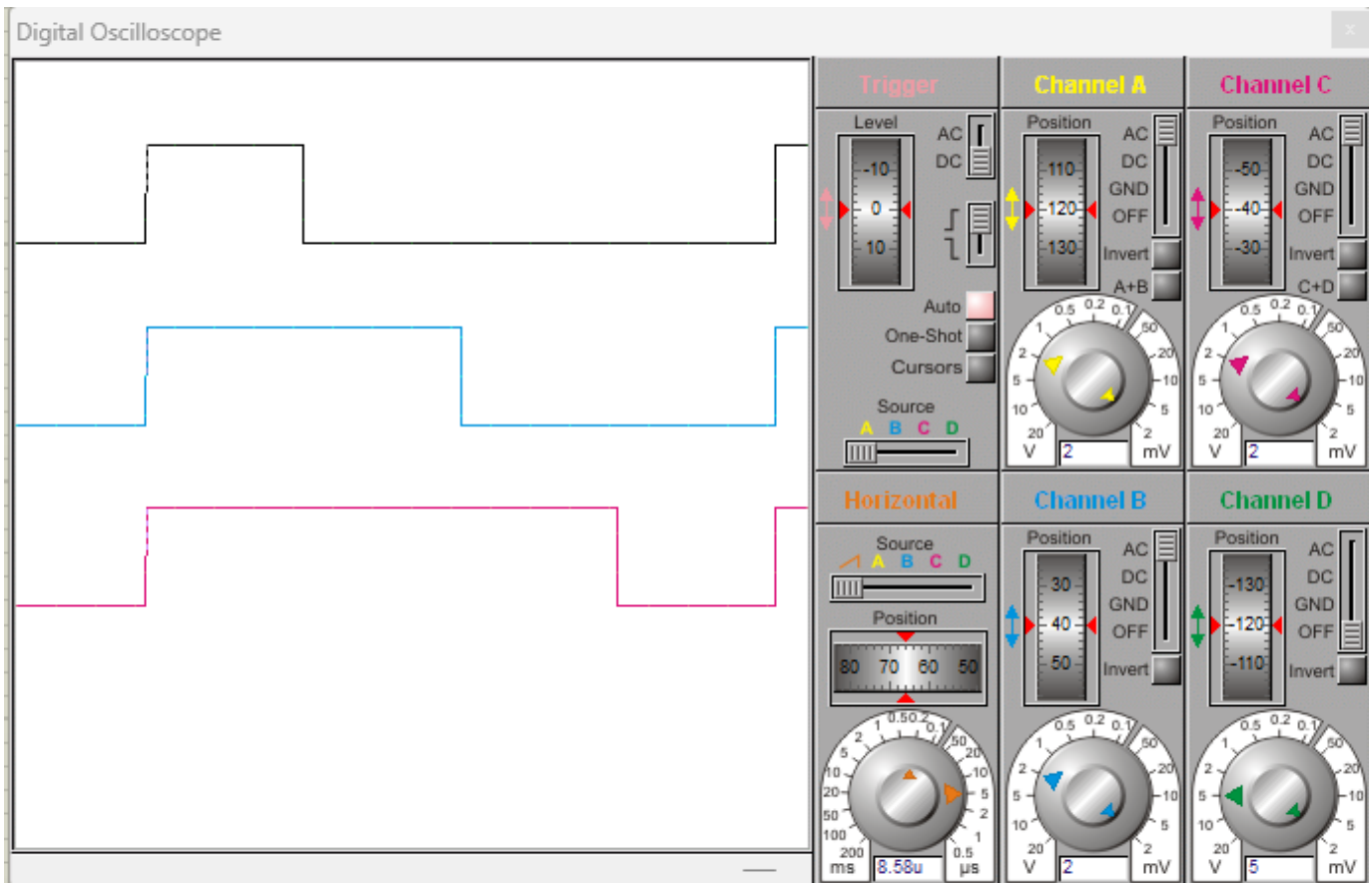
void T0Delay()
{
    TCNT0 = 0; // Reset Timer0 counter
    TCCR0B = 0x01; // Start Timer0 normal mode
    while ((TIFR0 & (1 << TOV0)) == 0); // Wait for Timer0 overflow flag to be set
    TCCR0A = 0; // Reset Timer0 control register A
    TIFR0 = 0x01; // Clear Timer0 overflow flag
}
```

## Circuit:

### 1) Simulation ( Proteus ) :

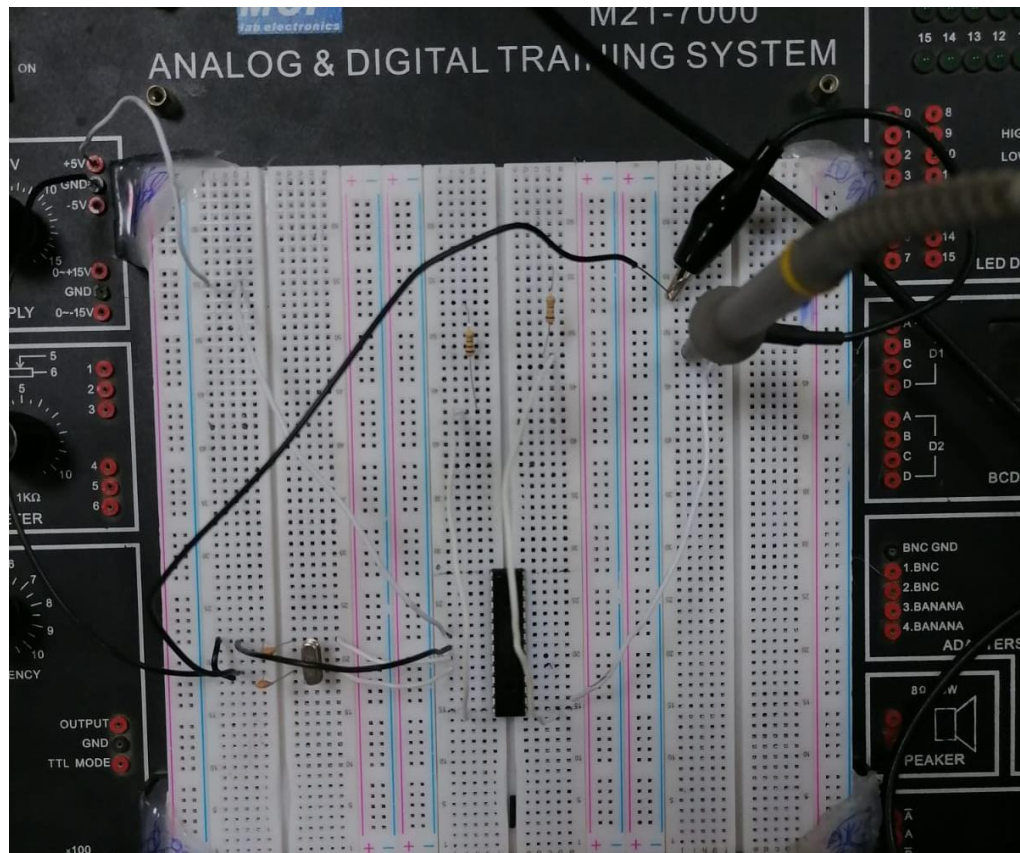


### Output :

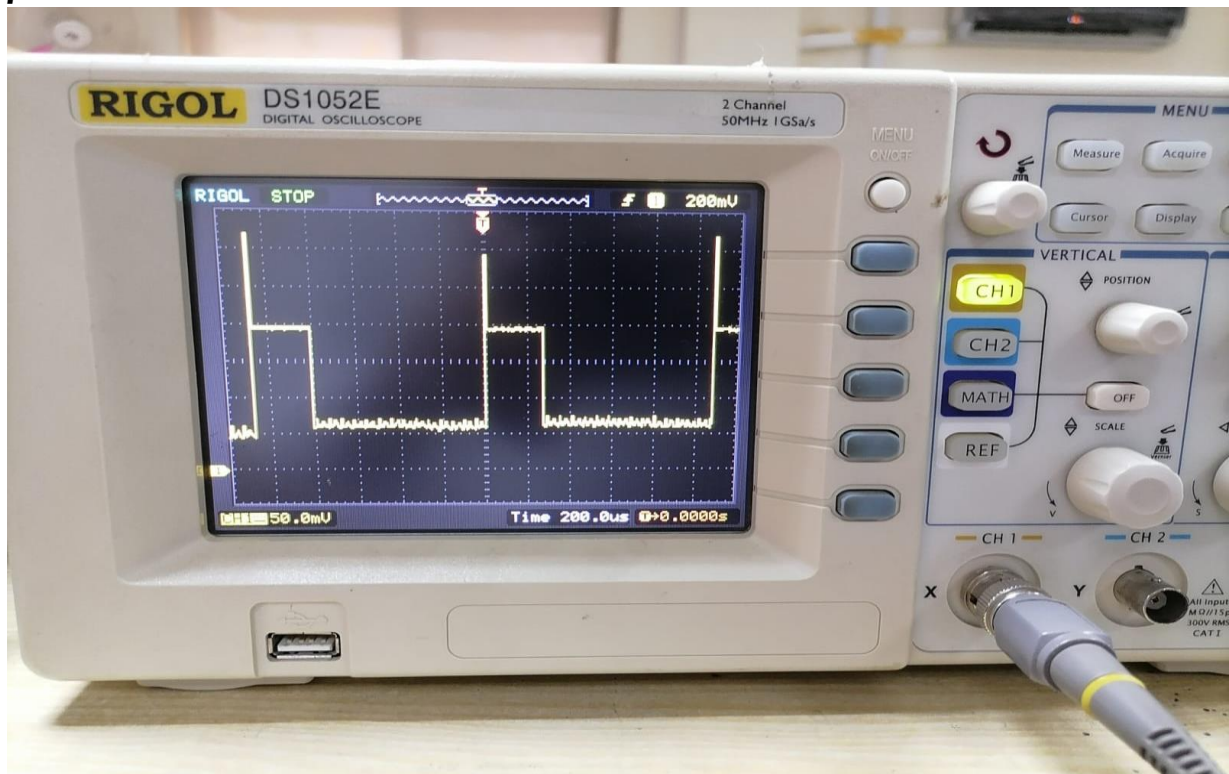


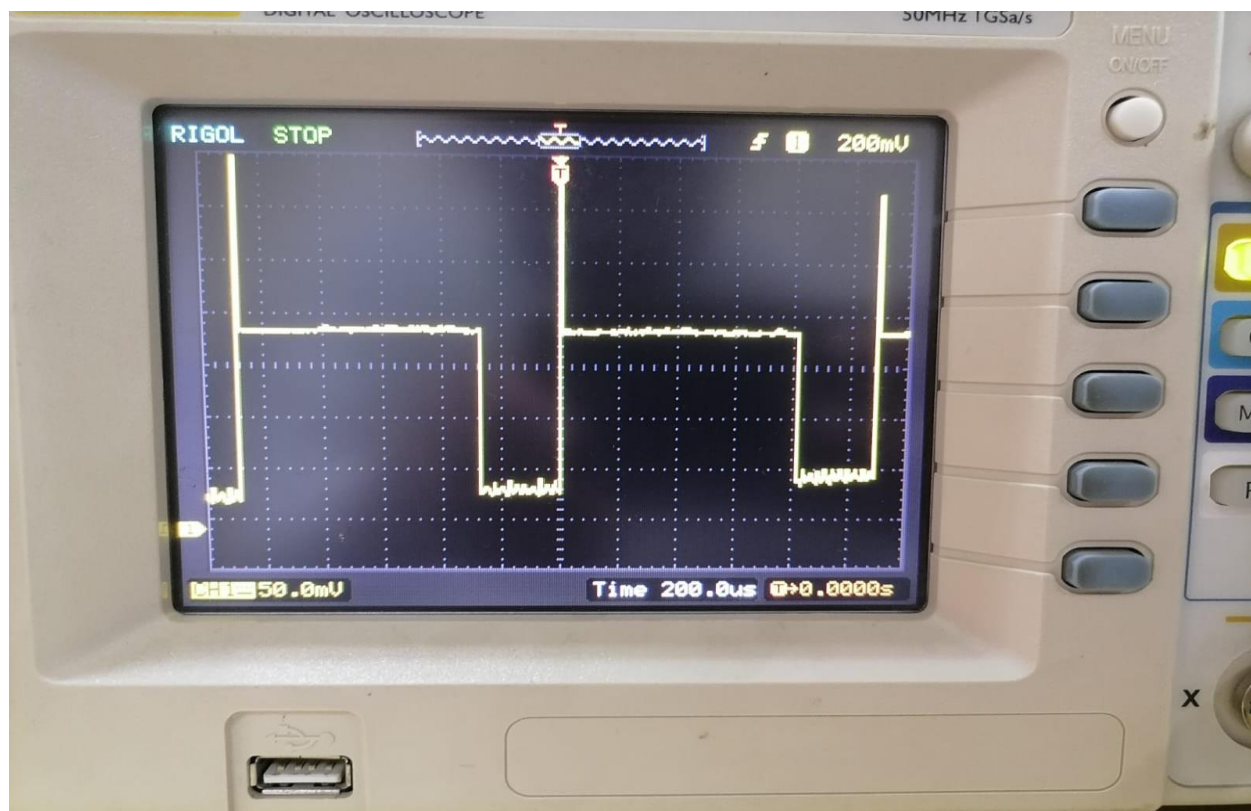
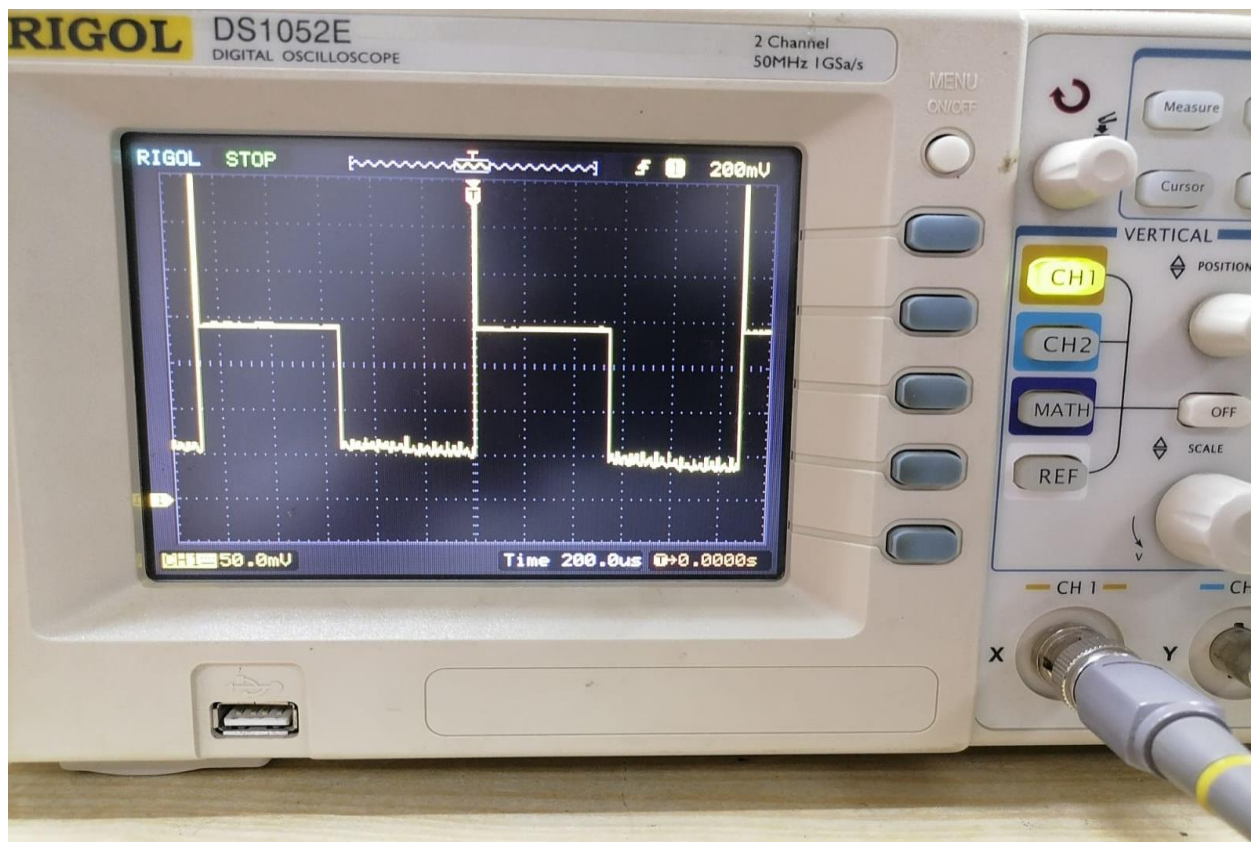


## 2) Hardware (Hardware) :



**Output :**







### Answer the following questions:

1. Explain the code written in lab task 1. What does the code do? Did you learn anything new in lab task 1?

ANS : This was an introductory code of C language for AVR Atmega 328P in which portb pin 0 was monitored for led .AVR input output header file was included at the first line ,Then the CPU frequency was defined which is 16MHz and also predefined delay function of AVR is included. After that the main syntax of C language , where firstly portb is set for out and then while loop which is always true for portb pin 0 ,at first it was high for 500ms then it is low for 500ms and repeat infinitely.

This code toggle LED on pin 0 of portb of Atmega328p for 500ms.

Yes we learn the basic of C language ,also learned that for AVR frequency we define it in the code ,we also learned that C language is too much easy for AVR.

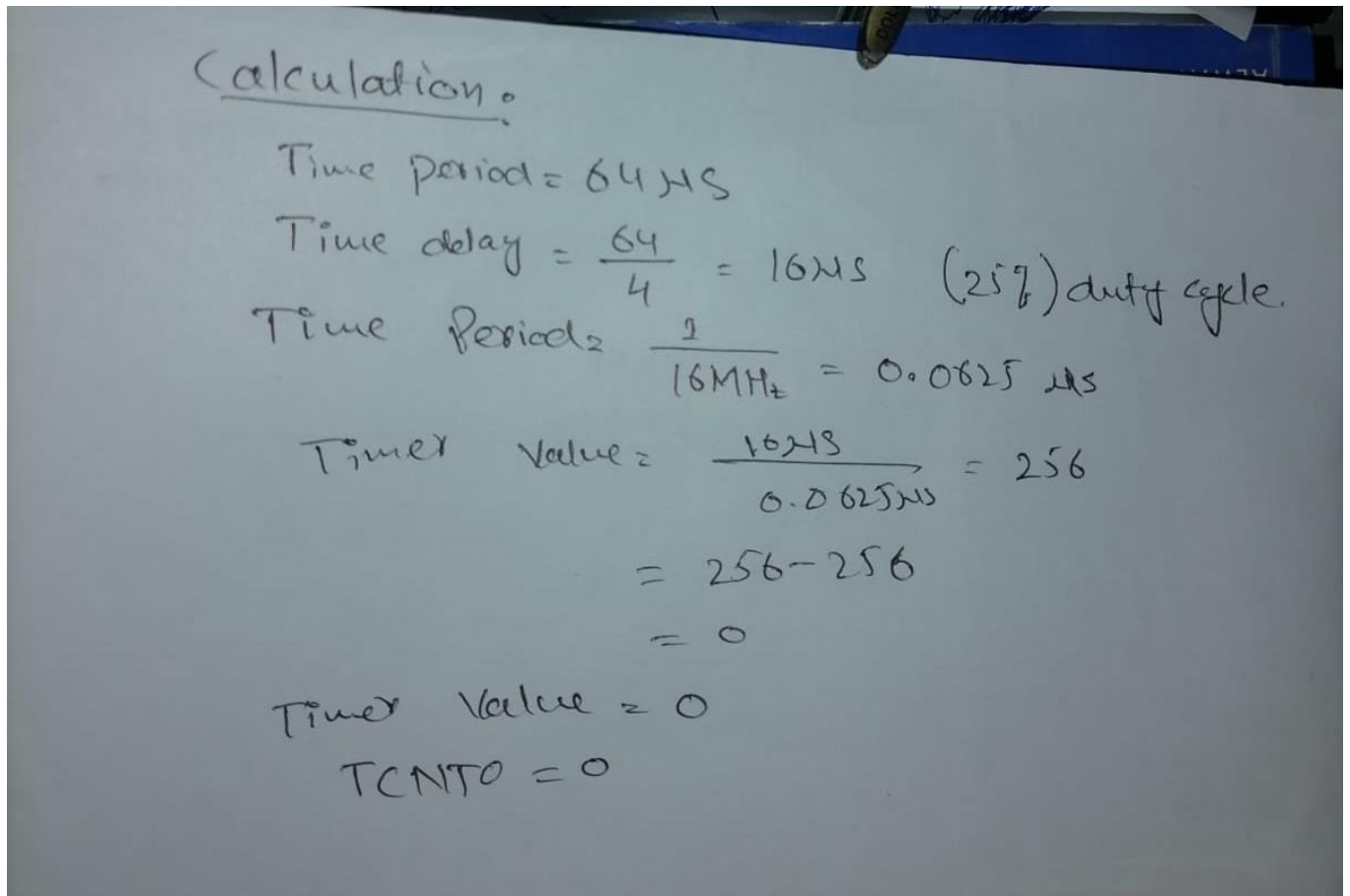
2. Explain the code given in lab task 4. What does the code do? What difference did you observe between what we studied in class and what you executed in lab?

ANS : This C code is for toggling the LED connected at pin 1. Here for delay the Timer0 is used , in which TCNT is 0 and no prescaler is defined , timer0 is normal mode .

This Code toggling the portd pin 1 with Timer0 delay.

The difference we observed that in class we use the timer counter as TCNT while here we used it as TCNT0 . Also timer control as TCCR in class while here it was TCCR0B.

3. In lab task 5, what is the time period and frequency of square wave you have generated on PC2, PC3 and PC4? Show the calculation below



Calculation:

$$\begin{aligned}\text{Time period} &= 64 \mu\text{s} \\ \text{Time delay} &= \frac{64}{4} = 16 \mu\text{s} \quad (25\%) \text{ duty cycle.} \\ \text{Time Period} &= \frac{1}{16 \text{ MHz}} = 0.0625 \mu\text{s} \\ \text{Timer Value} &= \frac{16 \mu\text{s}}{0.0625 \mu\text{s}} = 256 \\ &= 256 - 256 \\ &= 0 \\ \text{Timer Value} &= 0 \\ \text{TCNT0} &= 0\end{aligned}$$