MakerNova 1.0

WAVESENSE

MENTORS

Abida Polara
Divya Thakkar
Prachi Gupta
Pranshi Lavti

TEAM MEMBERS

Aditya Chandel
Amisha Soni
Chaitanya Singh
Harshal Atre
Mann Maruthi Teja
Mihir Kumar Roy
Ravita Vagaly
Ridhayu Gosai
Rishika Agarwal
Samhita Patil
Shreya Singhal
Uday Khunti

INDEX

PROBLEM STATEMENT 3
INTRODUCTION 3
UTILISED SOFTWARE APPLICATIONS 3
ATMEL STUDIO 3
PROTEUS 3
EXTREME BURNER 3
COMPONENTS 4
ATMEGA32 5
- INTRODUCTION 5
- KEY FEATURE 5
- PIN DIAGRAM AND EXPLANATION 5

LCD 6
- DESCRIPTION 6
- LCD PINOUT AND EXPLANATION 6

USBASP 7
- DESCRIPTION 7
- PIN DIAGRAM 8
- CONNECTION OF ATMEGA32 AND USBASP 8

PORT EXPANDER(PCF8574T) 8
BUCK CONVERTER 9
CONCEPTS 9
DIFFERENCE BETWEEN MICROCONTROLLER AND MICROPROCESSORS 9
SETTING PORTS IN ATMEGA32 USING EMBEDDED C 10
I/O PORT REGISTERS 10
- DATA DIRECTION REGISTERS(DDR) 10
- DATA REGISTERS 10
- INPUT PINS ADDRESS REGISTERS 10

BITWISE OPERATIONS 11
PWM, TIMERS AND DUTY CYCLES 11
- TIMERS IN ATMEGA32 12
- BASIC REGISTERS IN TIMERS 12

COMMUNICATION PROTOCOLS 13
INTER INTEGRATED CIRCUITS (I2C) 14
- REGISTERS IN THE ATMEGA32 I2C MODULE: 14

TIMELINE 16
TASKS 17

ERRORS AND THEIR SOLUTIONS..23
ALGORITHM OF FINAL CODE..25
APPLICATIONS..25
REFERENCES 25

PROBLEM STATEMENT

A Microcontroller is configured to generate a particular waveform of specified frequency and further its capabilities are configured to measure the frequency of any given input signal.

INTRODUCTION

The term "Wavesense" suggests a project or concept related to sensing, manipulating, or working with waves, particularly in the context of signals or frequencies. The Wavesense project represents signal generation driven by the capabilities of advanced microcontroller technology. To empower users to harness precise and customizable signal frequencies, this project combines innovative hardware design and intelligent software programming. By seamlessly integrating user-defined specifications, Wavesense opens the gateway to tailored signal shaping, catering to diverse applications across industries. This project exemplifies the synergy between modern microcontroller advancements and the imperative need for flexible signal generation, positioning itself at the forefront of technological innovation.

SOFTWARE APPLICATIONS FOR SIMULATION AND UPLOADING OF CODE

ATMEL STUDIO

Atmel Studio

Microchip Studio is an Integrated Development Environment (IDE) for developing and debugging AVR® and SAM microcontroller applications. It merges all of the great features and functionality of Atmel Studio into Microchip's well-supported portfolio of development tools to give a seamless and easy-to-use environment for writing, building, and debugging applications written in C/C++ or assembly code.

PROTEUS

Proteus

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.

EXTREME BURNER

eXtreme Burner

The extreme Burner- AVR is a full graphical user interface (GUI) AVR series of MCU that supports several types of clock sources for various applications. It enables to read and write a RC Oscillator or a perfect high-speed crystal oscillator.

COMPONENTS

Sr No.	COMPONENT	DESCRIPTION	QUANTITY
1.	ATmega32 microcontroller	The ATmega32 microcontroller is a high performance, low power Atmel AVR 8-bit microcontroller. It features 40 pins, 32Kb flash memory, 1024 Bytes EEPROM, 2Kb SRAM and features like 3 inbuilt timers and support for I2C, SPI, USART.	1
2.	JHD162A 16x2 LCD Display	A simple 16x2 LCD module with adjustable contrast. Controlled via 8 data ports and 3 configuration ports.	1
3.	USBASP AVR Programmer	USB in-circuit programmer for Atmel AVR controllers.	1
4.	Port Expander (PCF8574T)	The PCF8574T IO Expansion Board is used as remote 8-bit I/O expander for I2C-bus.	1
5.	Resistors	A resistive element with specific resistance used in electrical circuits.	3x1.5kΩ, 3x5.6
6.	Capacitors	A capacitive element with specific capacitance used in electrical circuits.	3x0.1uF, 3x4.7uF
7.	Breadboard	A solderless board with pre-made connection useful for prototyping electronic circuits.	
8.	Jumper wires	Wires terminating in male or female headers used for making connections.	
9.	Digital Storage Oscilloscope	A type of electronic test instrument that graphically displays varying voltages of one or more signals as a function of time.	
10.	Multimeter	A measuring instrument that can measure multiple electrical properties such as voltage, current, resistance.	
11.	Buck converter	It is a type of electronic circuit used to convert a high DC voltage into a low DC voltage	

Table1: Components

ATMEGA32

- **INTRODUCTION**

The ATmega32 is a microcontroller chip from the AVR (Advanced Virtual RISC) family manufactured by Microchip Technology (previously Atmel Corporation). It is widely used in various embedded systems and DIY projects due to its versatility, ease of use, and robust features. The ATmega32 is an 8-bit microcontroller, meaning it processes data in 8-bit chunks, and it is part of the larger AVR microcontroller series known for its low power consumption and high-performance capabilities.

Fig.1: Atmega32

- **KEY FEATURES**

The following are the key features of the ATmega32 microcontroller-

- **Architecture:**
  The ATmega32 is based on the Harvard architecture, which means it has separate program and data memories, allowing it to execute instructions and access data simultaneously.
- **I/O Pins:**
  The ATmega32 features several General-Purpose Input/Output (GPIO) pins, which can be configured as inputs or outputs to interface with external devices and sensors.
- **Timers/Counters:**
  The microcontroller comes with multiple timers/counters, allowing precise timing and event-counting operations. Analog to Digital conversion also has the best features.

- **PIN DIAGRAM AND EXPLANATION**

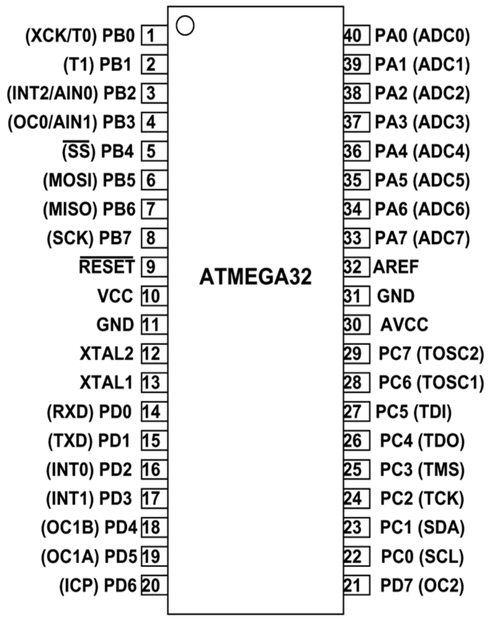


Fig 2: Atmega32 Pin Diagram

- **VCC**: Supply voltage (typically 5V).
- **GND**: Ground (0V reference).
- **RESET**: Reset pin, used to restart the microcontroller.
- **RXD, TXD**: UART communication pins for serial data transmission and reception.
- **XTAL1, XTAL2**: Crystal oscillator pins for the external clock source.
- **AREF**: Analog reference voltage for ADC (Analog-to-Digital Converter).
- **AVCC**: Analog supply voltage for ADC.
- **PC0 to PC7**: Port C, general-purpose digital I/O pins.
- **PD2 to PD7**: Port D, general-purpose digital I/O pins.
- **PB0 to PB7**: Port B, general-purpose digital I/O pins.

LCD

- **DESCRIPTION**

LCD, which stands for Liquid Crystal Display, is an electronic device that is used for data display. LCDs are preferable over seven segments and LEDs, as they can easily represent data in the form of alphabets, characters, numbers, or animations. LCDs are very easy to program and make any work quite attractive and simple. Numerous types of LCDs are available on the market, such as 16X2, 16X4, 20X2, 20X4, graphical LCDs (128X64), etc. The LCD used here is a 16X2 alphanumeric LCD; it displays 32 characters in two rows, so in one row 16 characters can be displayed.

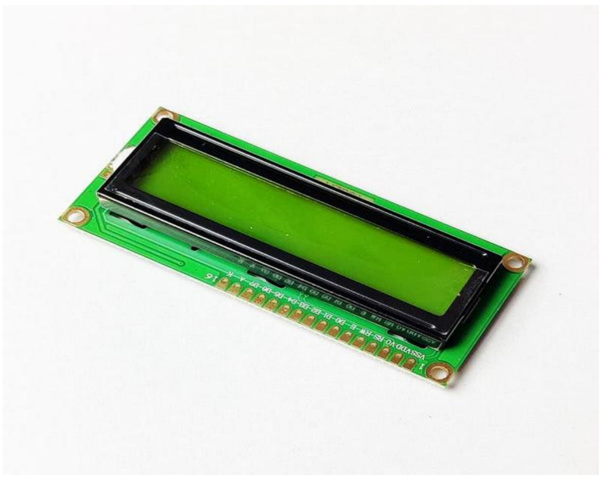


Fig 3: 16x2 LCD

- **LCD PINOUT AND EXPLANATION**

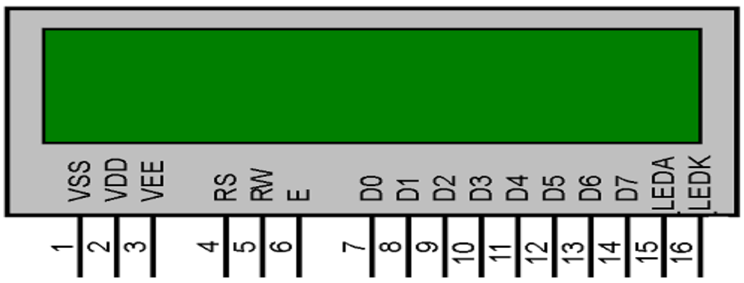


Fig 4: LCD Pinout

16x2 LCDs can interface with AVR microcontrollers in two modes:

1. 4-bit mode
2. 8-bit mode

In 8-bit mode, commands are sent to the LCD by using eight data lines (D0-D7), while in 4-bit mode, four data lines are used to send commands and data these data lines can be connected to any port of the Atmega32.

Pin No.	Pin Name	Description
1.	VCC	Supply pin (+5V DC)
2.	VDD	Ground Pin
3.	VEE	Contrast Pin
4.	RS	Register a selection pin (either data or command). RS = 0; Command Register, RS = 1: Data Register
5.	RW	Selects Read or Write operation RW = 0 for write; RW = 1 for read.
6.	E	Enable Pin
7.	D0-D7	Data pins
8.	LEDA	This pin is connected to +5V
9.	LEDK	This pin is connected to GND.

Table 2: LCD Pinout Description

- **Programming of LCD**

There are two registers: Command and Data. The command register is to be selected while giving the command to the LCD display and while sending data, the data register is to be selected. A command is an instruction given to the display to perform the required function according to the given command. To display textual information, data is sent to the LCD display.

- **Sending Commands on LCD**

For sending commands on LCD write commands on data pins. For this select:

- RS = 0 >> selects command register
- RW = 0 >> selects write operation
- E >> make enable pin from high to low

- **Sending Data On LCD**

For sending data on LCD, write data on data pins. For this select:

- RS = 1 >> selects data register
- RW = 0 >> selects write operation
- E >> make enable pin from high to low

USBASP

- **DESCRIPTION**

This AVR USBASP Programmer module is used to upload the source code from the computer into the ATMEL's microcontroller unit using a standard USB port as input. This module comes with 10 on-board pins connected through the ribbon cable as the output. It can support a wide variety of Atmel (AT) AVR microcontroller chips such as ATMEGA328P, ATmega128P, etc.

- **PIN DIAGRAM**

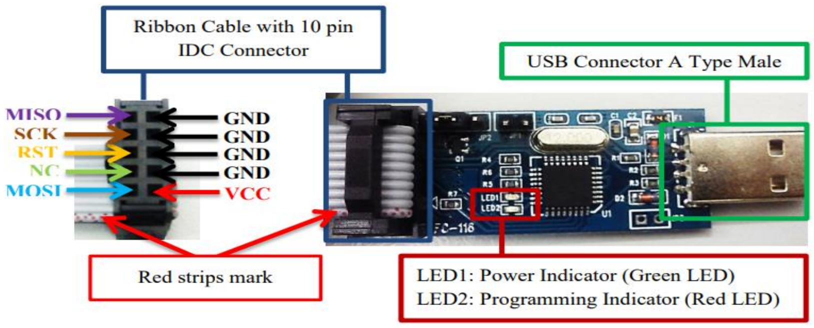

Ribbon Cable with 10 pin IDC Connector
USB Connector A Type Male
MISO
SCK
RST
NC
MOSI
GND
GND
GND
GND
VCC
Red strips mark
LED1: Power Indicator (Green LED)
LED2: Programming Indicator (Red LED)

Fig.5: USBASP Pin diagram

- **CONNECTION OF ATMEGA32 AND USBASP**

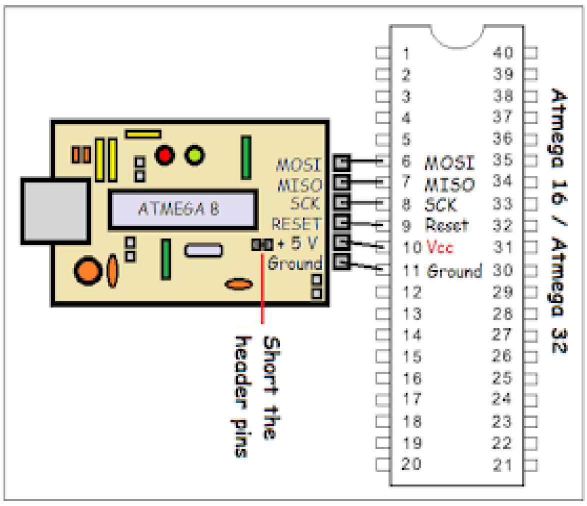


Fig. 6: Connection of USBASP and Atmega32

PORT EXPANDER(PCF8574T)

This 8-bit input/output (I/O) expander for the two-line bidirectional bus (I2C) is designed for 2.5-V to 6-V VCC operation. The PCF8574T device general-purpose remote I/O expansion for most microcontroller families by way of the I2C interface [serial clock (SCL), serial data (SDA)]. The device features an 8-bit quasi-bidirectional I/O port (P0–P7), including latched outputs with high current drive capability for directly driving LCDs. Each quasi-bidirectional I/O can be used as an input or output without the use of a data-direction control signal. At power on, the I/Os are high, in this mode, only a current source to VCC is active.

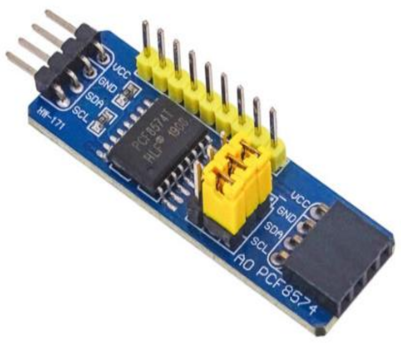


Fig. 7: Port Expander (PCF8574T)

BUCK CONVERTER

A buck converter or step-down converter is a DC-TO-DC convertor which steps down voltage (while stepping up current) from its input (supply) to its output (load). Switching converters (such as buck converters) provide much greater power efficiency as DC-to-DC converters than linear regulators, which are simpler circuits that lower voltages by dissipating power as heat, but do not step-up output current. The efficiency of buck converters can be very high, often over 90%, making them useful for tasks such as converting a computer's main supply voltage, which is usually 12 V, down to lower voltages needed by USB, DRAM and the CPU, which are usually 5, 3.3 or 1.8 V.

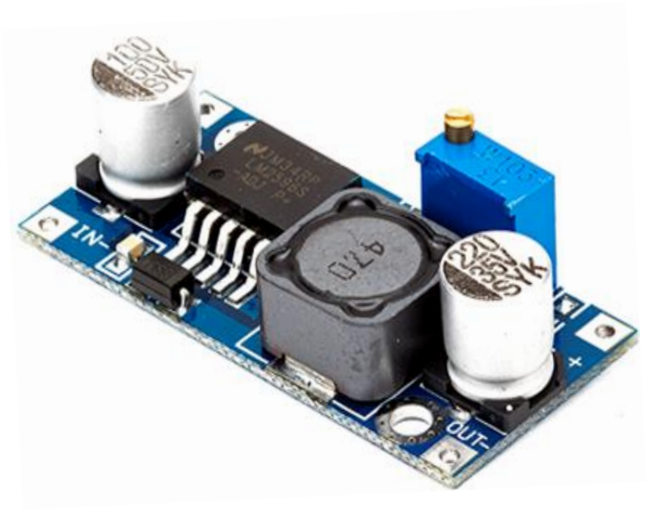


Fig. 8: Buck Converter

CONCEPTS

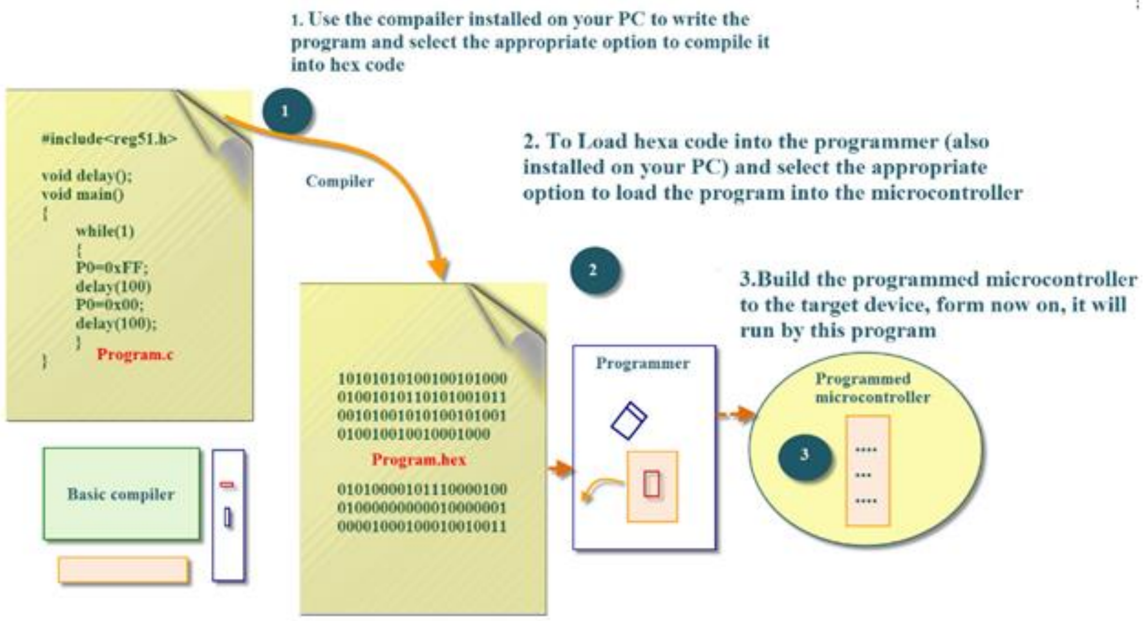
DIFFERENCE BETWEEN MICROCONTROLLER AND MICROPROCESSORS

MICROCONTROLLER	MICROPROCESSOR
A microcontroller is a compact integrated circuit designed to govern specific operation in an embedded system. A typical microcontroller includes a processor memory and input/output (I/O) peripherals on a single chip.	A microprocessor is a component that performs the instructions and tasks involved in computer processing. In a computer system, the microprocessor is the central unit that executes and manages the logical instructions passed to it.
Since components are internal, most of the operations are internal instruction, hence speed is fast.	Since memory and I/O components are all external, each instruction will need an external operation hence it is relatively slow.
Commonly available in both DIP and SMD form	Not commonly available in DIP form and hence difficult to integrate into projects for beginners
Comes with prepackaged components that cannot be changed	Gives more flexibility in terms of hardware as we can decide on what RAM, Flash and I/O drivers are available to the microprocessor
Microcontroller doesn’t have a zero status flag.	Microprocessors have a zero status flag.

Table 3: Difference Between Microcontrollers and Microprocessors

SETTING PORTS IN ATMEGA32 USING EMBEDDED C

Embedded C is a popular programming language in the software field for developing electronic gadgets. Each processor used in electronic systems is associated with embedded software. Embedded C programming plays a key role in performing specific functions by the processor.



```
#include<reg51.h>

void delay();
void main()
{
    while(1)
    {
    P0=0xFF;
    delay(100)
    P0=0x00;
    delay(100);
    }
}
```

1. Use the compailer installed on your PC to write the program and select the appropriate option to compile it into hex code
1
Compiler
Program.c
Basic compiler
2. To Load hexa code into the programmer (also installed on your PC) and select the appropriate option to load the program into the microcontroller
2
3.Build the programmed microcontroller to the target device, form now on, it will run by this program
Program.hex
Programmer
Programmed microcontroller
3

```
10101010100100101000
01001010110101001011
00101001010100101001
010010010010001000
```

```
01010000101110000100
01000000000010000001
00001000100010010011
```

Fig. 9: Block diagram for Blinking LED

The Embedded C code written in the above block diagram is used for blinking the LED connected with Port0 of the microcontroller.

I/O PORT REGISTERS

I/O port contains three I/O registers for each port:

- **DATA DIRECTION REGISTERS(DDR)**
    - ➢ DDRx registers are used to configure the direction of the pins of the port, determining whether they will act as input pins or output pins.
    - ➢ Writing a '1' to a bit in this register sets the corresponding pin as an output pin.
    - ➢ Writing a '0' to a bit in this register sets the corresponding pin as an input pin.
    - ➢ All bits in this register can be read and written too.
    - ➢ The default or initial value of all bits in this register is '0'.
    - ➢ Example: To set all 8 pins of Port D as output pins, we write 0xFF to DDRD (0xFF = 0b11111111).
- **DATA REGISTERS**
    - ➢ PORTx registers are used to set the logic level (HIGH or LOW) on the pins of the port.
    - ➢ Writing a '1' to a bit in this register puts a logic HIGH (usually 5V) on the corresponding pin.
    - ➢ Writing a '0' to a bit in this register puts a logic LOW (usually 0V) on the corresponding pin.
    - ➢ All bits in this register can be read and written to.
    - ➢ The default or initial value of all bits in this register is '0'.
    - ➢ Example: To write the value 0x55 to Port D, we write 0x55 to PORTD.
- **INPUT PINS ADDRESS REGISTERS**
    - ➢ PINx registers are used to read the logic levels on the specific pins of the port.
    - ➢ These registers are read-only, meaning you cannot write to them.
    - ➢ Reading a bit from this register allows you to determine the logic level on the corresponding pin.
    - ➢ Example: To read the value on Port D and store it in an 8-bit variable named 'port_value', we use the statement: `port_value = PIND`.

BITWISE OPERATIONS

In C, the following 6 operators are bitwise operators:

- **& - Bitwise AND:**

  The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of the corresponding bit is evaluated to 0.
- **| - Bitwise OR:**

  The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1.
- **^ - Bitwise XOR:**

  The result of the bitwise XOR operator is 1 if the corresponding bits of two operands are opposite.
- **~ - Bitwise NOT:**

  The output of the bitwise NOT operator is that it takes one number and inverts all bits of it.
- **<< - Left Shift:**

  It left shifts the bits of the first operand, the second operand decides the number of places to shift. (e.g.: 0b00011010<<3=0b11010000)
- **>> - Right Shift:**

  It right shifts the bits of the first operand, the second operand decides the number of places to shift. (e.g:0b00011010>>3=0b01000011)

Symbols	Meaning
&	AND
|	OR
^	XOR
~	Invert, Complement
>>	Shift Right
<<	Shift Left

Table 4: Bitwise Operators

PWM, TIMERS AND DUTY CYCLES

PWM stands for Pulse Width Modulation. Pulse refers to the short period of time over which the signal will be asserted. Width refers to the duration of time over which the signal is asserted. Modulation refers to the changes in the pulse width over time.

Another really common term in PWM is duty cycle. Duty cycle is the amount of time a digital signal is in the "active" state relative to the period of the signal. Duty cycle is usually given as a percentage.

For example, a perfect square wave with equal high time and low time has a duty cycle of 50%. Here is a diagram showing duty cycle in a general way.

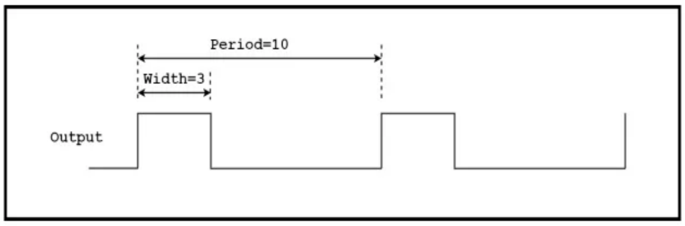


Fig.10: General Diagram for Duty Cycle

- **Duty Cycle Microcontroller Timer Calculation**

  Based on the example above, steps to calculate duty cycle:
  If active high, the duty cycle is

(Width ÷ Period) × 100 = (3 ÷ 10) × 100 = 30%
If the signal is defined as active low, the duty cycle is 70%

- **TIMERS IN ATMEGA32**

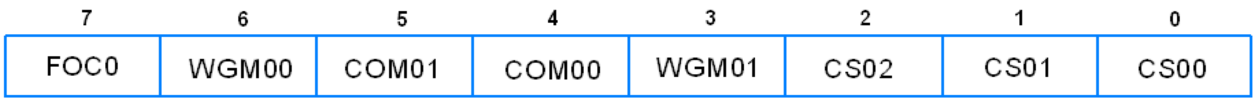
Timers are used to generate time delays, waveforms, or to count events. Also, the timer is used for PWM generation, capturing events, etc.
In AVR ATmega16 / ATmega32, there are three timers:

1. **Timer0**: 8-bit timer
2. **Timer1**: 16-bit timer
3. **Timer2**: 8-bit timer

- **BASIC REGISTERS IN TIMERS**

  - **TCNTn: Timer / Counter Register:**
    Every timer has a timer/counter register. It is zero upon reset. This register is used to access and write a value to. It counts up with each clock pulse.
  - **TOVn: Timer Overflow Flag:**
    Each timer has a Timer Overflow flag. When the timer overflows, this flag will get set.
  - **TCCRn: Timer Counter Control Register:**
    This register is used for setting the modes of timer/counter.
  - **OCRn: Output Compare Register:**
    The value in this register is compared with the content of the TCNTn register. When they are equal, the OCFn flag will get set.

- **Timer 0**



7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

1. **TCNT0:** Timer / Counter Register 0

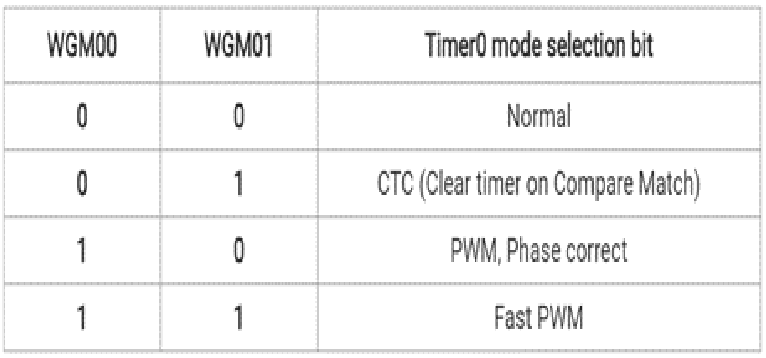
It is an 8-bit register. It counts up with each pulse.

2. **TCCR0:** Timer / Counter Control register 0

This is an 8-bit register used for the operation mode and the clock source selection.

- Bit 7 - FOC0**:** Force compare match
  Writing 1 to this bit causes the wave generator to act as if a compare match has occurred.

- Bit 6, 3 - WGM00, WGM01: Waveform Generation Mode

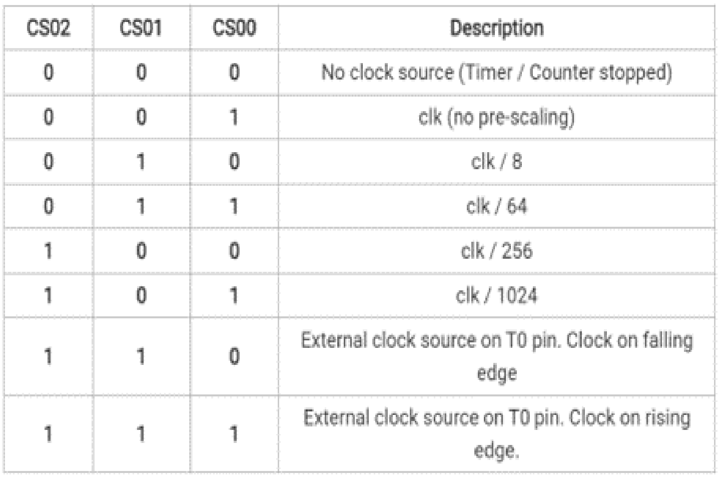


WGM00	WGM01	Timer0 mode selection bit
0	0	Normal
0	1	CTC (Clear timer on Compare Match)
1	0	PWM, Phase correct
1	1	Fast PWM

Table 5

- Bit 5:4 - COM01:00: Compare Output Mode
  These bits control the waveform generator. Used in the compare mode of the timer.

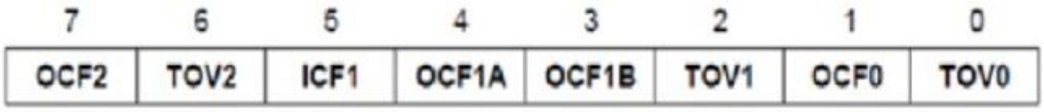
➢ Bit 2:0 - CS02:CS00: Clock Source Select
These bits are used to select a clock source. When CS02: CS00 = 000, then timer is stopped. As it gets a value between 001 to 101, it gets a clock source and starts as the timer.



CS02	CS01	CS00	Description
0	0	0	No clock source (Timer / Counter stopped)
0	0	1	clk (no pre-scaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge.

Table 6

3. **TIFR:** Timer Counter Interrupt Flag register



7	6	5	4	3	2	1	0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

➢ Bit 0 - TOV0**:** Timer0 Overflow flag
➢ Bit 1 - OCF0: Timer0 Output Compare flag
➢ Bit 2 – TOV1: Timer1 Overflow flag
➢ Bit 3 - OCF1B: Timer1 Output Compare B match flag
➢ Bit 4 - OCF1A: Timer1 Output Compare A match flag
➢ Bit 5 - ICF1: Input Capture flag
➢ Bit 6 - TOV2: Timer2 Overflow flag
➢ Bit 7 - OCF2: Timer2 Output Compare match flag

Similarly refer to Datasheet for Timer1 and Timer2.

COMMUNICATION PROTOCOLS

There are three protocols in Atmega32:

- **UART (Universal Asynchronous Receiver/Transmitter):**
  UART is a popular asynchronous serial communication protocol used for transmitting and receiving data between two devices. The ATmega32 has built-in hardware UART modules that make it easy to implement this protocol for communication with other devices, such as sensors, displays, or other microcontrollers.
- **SPI (Serial Peripheral Interface):**
  SPI is a synchronous serial communication protocol commonly used to connect multiple peripheral devices to a microcontroller.
- **I2C (Inter-Integrated Circuit):**
  I2C is a multi-master, multi-slave, serial communication protocol used to connect multiple devices on the same bus. The ATmega32 can act as both an I2C master and slave, making it suitable for applications requiring communication with I2C-compatible sensors, EEPROMs, or other microcontrollers.

INTER INTEGRATED CIRCUITS (I2C)

The I2C bus is a very popular and powerful bus used for communication between a master (or multiple masters) and a single or multiple slave device.

The general procedure for a master to access a slave device is the following:

- Suppose a master wants to send data to a slave:
  - Master-transmitter sends a START condition and addresses the slave-receiver
  - Master-transmitter sends data to slave-receiver
  - Master-transmitter terminates the transfer with a STOP condition
- If a master wants to receive/read data from a slave:
  - Master-receiver sends a START condition and addresses the slave-transmitter
  - Master-receiver sends the requested register to read to slave-transmitter
  - Master-receiver receives data from the slave-transmitter
  - Master-receiver terminates the transfer with a STOP condition

START and STOP Conditions of I2C communication with this device is initiated by the master sending a START condition and terminated by the master sending a STOP condition. A high-to-low transition on the SDA line while the SCL is high defines a START condition. A low-to-high transition on the SDA line while the SCL is high defines a STOP condition.

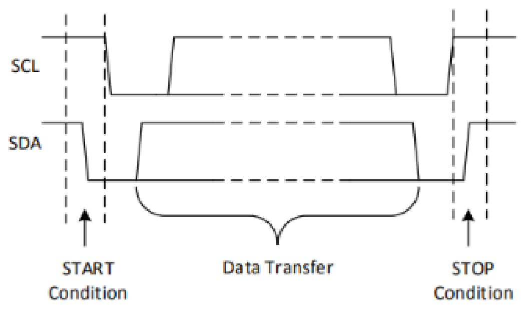


Fig. 11

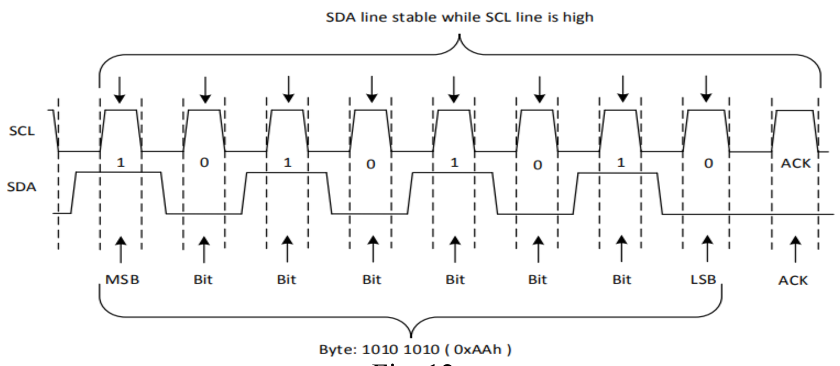

SDA line stable while SCL line is high
SCL
SDA
1
0
1
0
1
0
1
0
ACK
MSB
Bit
Bit
Bit
Bit
Bit
Bit
LSB
ACK
Byte: 1010 1010 (0xAAh)

Fig. 12

- **REGISTERS IN THE ATMEGA32 I2C MODULE:**
  1. TWBR: TWI Bit Rate Register
  2. TWCR: TWI Control Register
  3. TWSR: TWI Status Register
  4. TWDR: TWI Data Register
  5. TWAR: TWI Address Register

- TWBR

7	6	5	4	3	2	1	0	
TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0	TWBR Register

TWI bit rate register used in generating SCL frequency while operating in master mode.

- TWCR
  TWI control resistor used to control events of all I2C communication.

7	6	5	4	3	2	1	0	
TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	TWCR Register

  - Bit 7 – TWINT: TWI interrupt
    This bit gets set whenever TWI completes its current event (like start, stop, transmit, receive, etc). While I-bit in SREG and TWIE bit in TWCR is enabled then the TWI interrupt vector is called whenever TWI interrupt occurs. TWI interrupt flag must be cleared by software by writing a logical one to it. This bit is not automatically cleared by hardware.
  - Bit 6 – TWEA: TWI enable acknowledgment bit
    This is TWI acknowledgment enable bit, it is set in receiver mode to generate acknowledgment and cleared in transmit mode.
  - Bit 5 – TWSTA: TWI START condition bit
    The master device set this bit to generate START condition by monitoring free bus status to take control over the TWI bus.
  - Bit 4 – TWSTO: TWI STOP condition bit
    The master device set this bit to generate STOP condition to leave control over the TWI bus.
  - Bit 3 – TWWC: TWI write collision
    This bit gets set when writing to the TWDR register before the current transmission not complete i.e. TWINT is low.
  - Bit 2 – TWEN: TWI enable bit
    This bit is set to enable the TWI interface in the device and takes control over the I/O pins.
  - Bit 1 – Reserved
  - Bit 0 – TWIE: TWI interrupt enable
    This bit is used to enable TWI to interrupt routine while the I-bit of SREG is set as long as the TWINT flag is high.

- TWSR

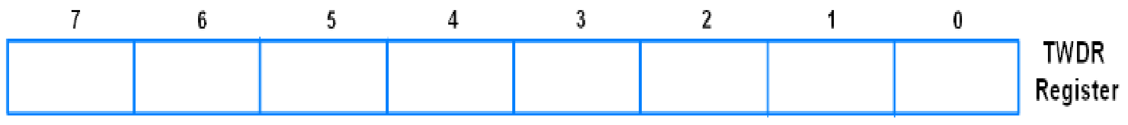
7	6	5	4	3	2	1	0	
TWS7	TWS6	TWS5	TWS4	TWS3	—	TWPS1	TWPS0	TWSR Register

  - Bit 7: Bit 3 - TWS7: TWS3: TWI status bits
    TWI status bits show the status of TWI control and bus
  - Bit 1:0 - TWPS1:TWPS0: TWI pre-scaler bits
    TWI pre-scaler bits used in bit rate formula to calculate SCL frequency

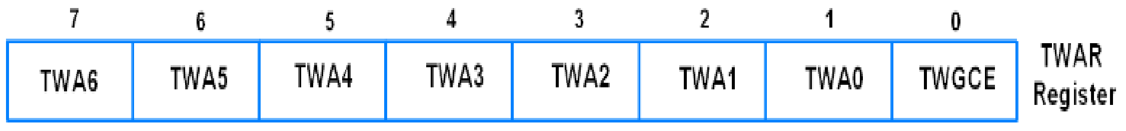
TWPS1	TWPS0	Exponent	Pre-scaler value
0	0	0	1
0	1	1	4
1	0	2	16
1	1	3	64

Table 7

- TWDR
  TWDR contains data to be transmitted or received. It's not writable while TWI is in process of shifting a byte. The data remains stable as long as TWINT is set.



- TWAR
  TWAR register contains the address of the TWI unit in slave mode. It is mostly used in the multi-master system.



  - Bit 7:1 - TWA6: TWA0: TWI address bits
    TWI address bits contain TWI 7-bit address with which it can called by other masters in slave mode.
  - Bit 0 – TWGCE: TWI general call enable bit
    TWI general call enable bit when set it enables recognition of general call over the TWI bus voltages needed by USB, DRAM and the CPU, which are usually 5, 3.3 or 1.8 V.

TIMELINE

- **First week (10-07-23 to 14-07-23)**
  - Discussion regarding the oscilloscope, microcontroller, microprocessor and its uses.
  - Discussion on the architecture of the microcontroller i.e.; von-Neumann and Harvard.
  - Discussion on AVR and RISC.
  - Discussion on choosing Atmega32 over Atmega328p.
  - Installation of Proteus and Atmel.

- **Second week (15-07-23 to 22-07-23)**
  - Learned how to write code on Atmel.
  - Learned how to upload Atmel code in Proteus.
  - Learned about bit wise operators.
  - Study about the PWM, Timer, Timer register and generation of waves through it.
  - Fixed the issues coming in the circuit of generation of waves by different methods.

- **Third Week (25-07-23 to 02-08-23)**
  - Discussion on why amplitude of wave decreases.
  - Study about communication protocol (I2C) .
  - Discussion about implementing software simulation on hardware.
  - Listed different hardware components to be used in the project.

- **Fourth Week (03-08-23 to 07-08-23)**
  - Discussion on finding the value of R and C for square wave conversion, made changes in the circuit and code to make the circuit work.
  - Researched on pin configuration of USB ASP that is used to upload code in hardware circuits.
  - Installation of Extreme Burner software which is used to write code from Atmel studio to Atmega32 board.
  - Demonstration of the hardware of previous tasks.

- **Fifth Week (08-08-23 to 13-08-23)**
  - Getting familiar with Lcd interference in 4-bit and 8-bit mode.
  - Tried to do Lcd interfacing with port expander, understood the code, Reviewed the code.
  - Connection done for LCD interfacing 4 bit with port expander on Proteus.
  - Merged oscilloscope and Lcd code together.
  - Searching about pcf8574T due to it not being an active component of proteus.
  - Searched for the address of port expander (pcf8574T), soldering practice.

- **Sixth Week (14-08-23 to 21-08-23)**
  - Lcd interfacing in 4-bit mode with port expander (pcf8574T)- hardware connections.
  - Software Team: simulated merged circuit of frequency generator and frequency metre using timers.
  - Hardware team: listed currently known components that will be required for hardware assembly. Troubleshooting the software team's code and Proteus circuit.
  - Successfully implemented integration of function generator and frequency metre i.e. the final circuit.

TASKS

- **Task 1:**

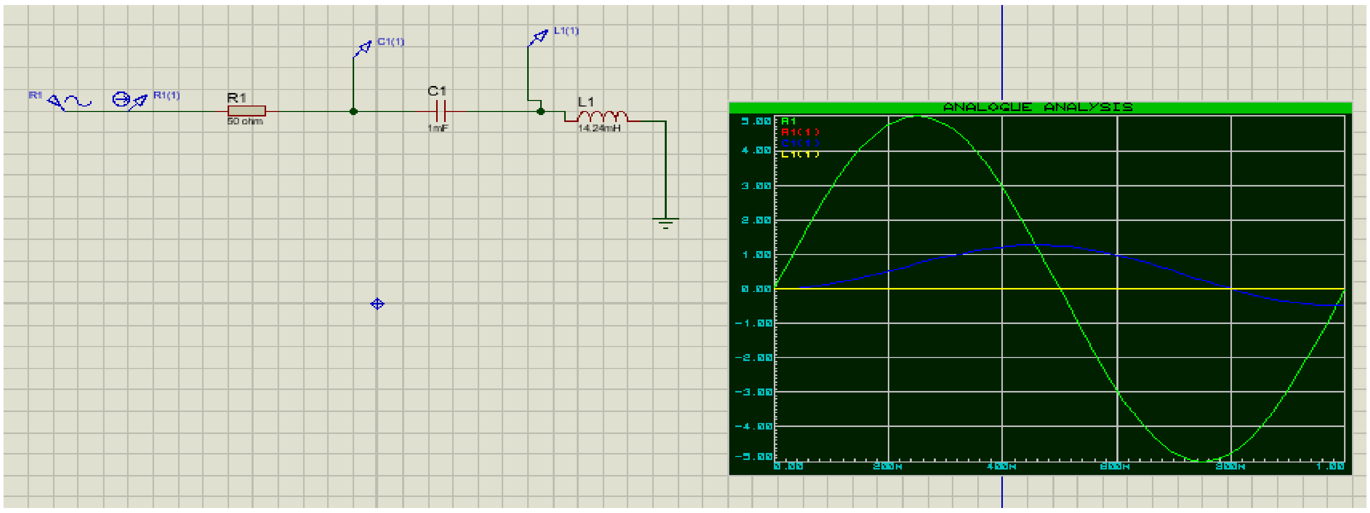


Fig. 13: Series RLC in Proteus and plot V vs I

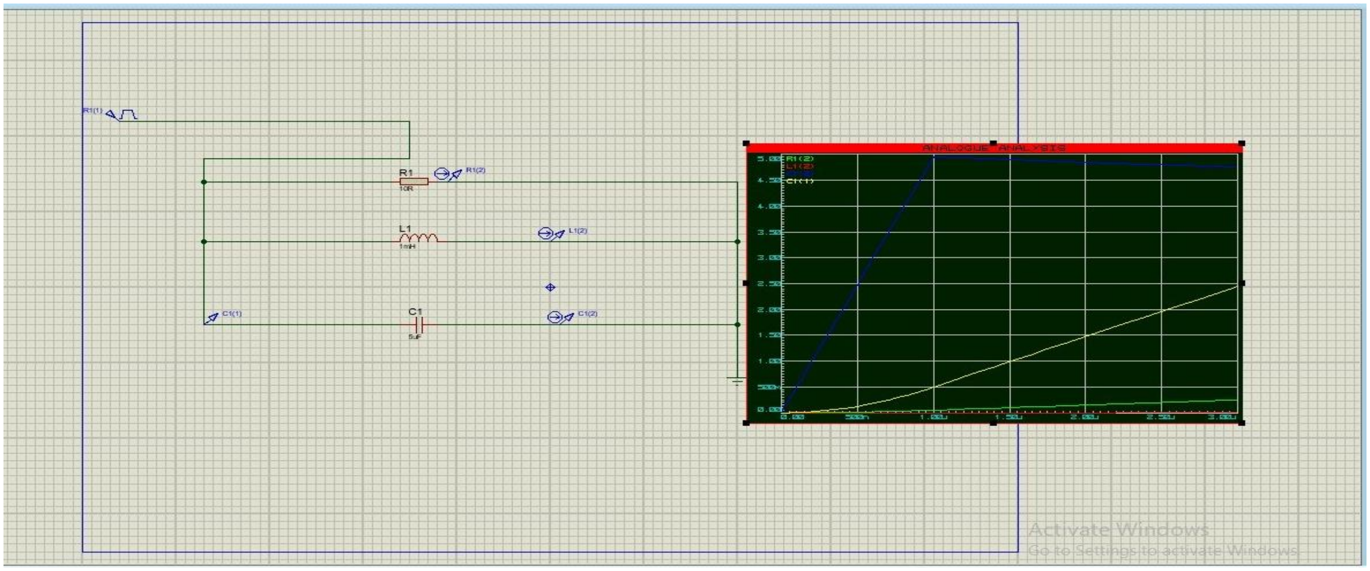
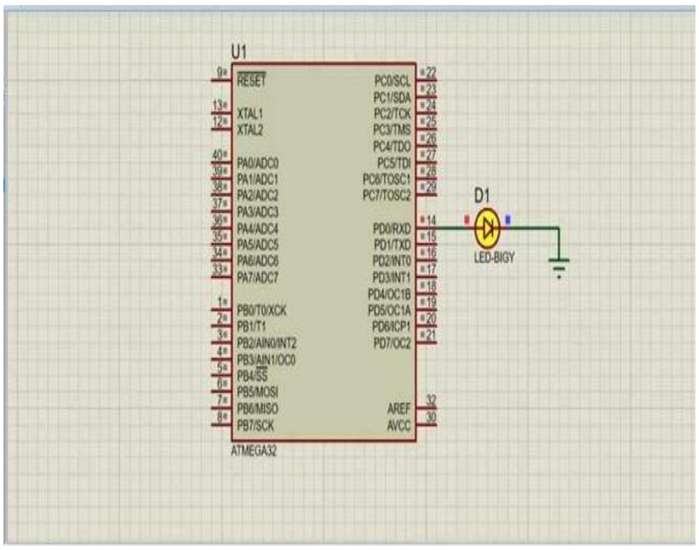


Fig. 14: Parallel RLC in Proteus and plot V vs I

- **Task 2: Blink LED**



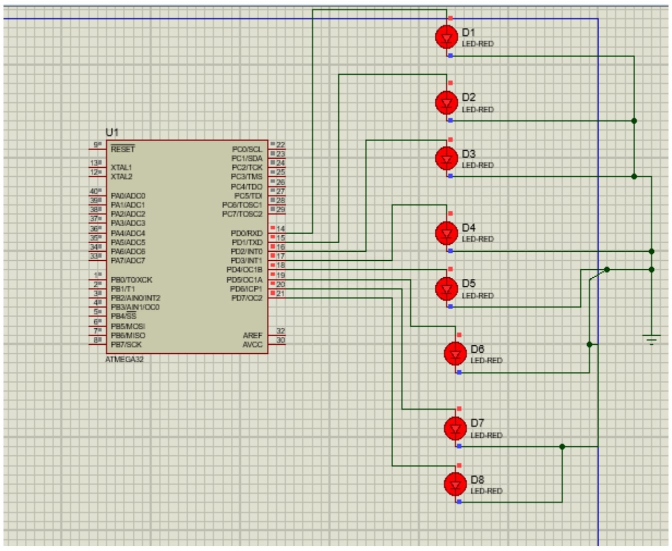


Fig. 15: Blink 1 LED

Fig. 16: Blink 8 LED

Blink 8 LEDs in different pattern using switches:

- Switch 1: All LEDs will be turned on one by one.
- Switch 2: All LEDs will be turned on one by one while keeping the previous LED on.
- Switch 3: First all even LEDs will glow and then the odd ones.

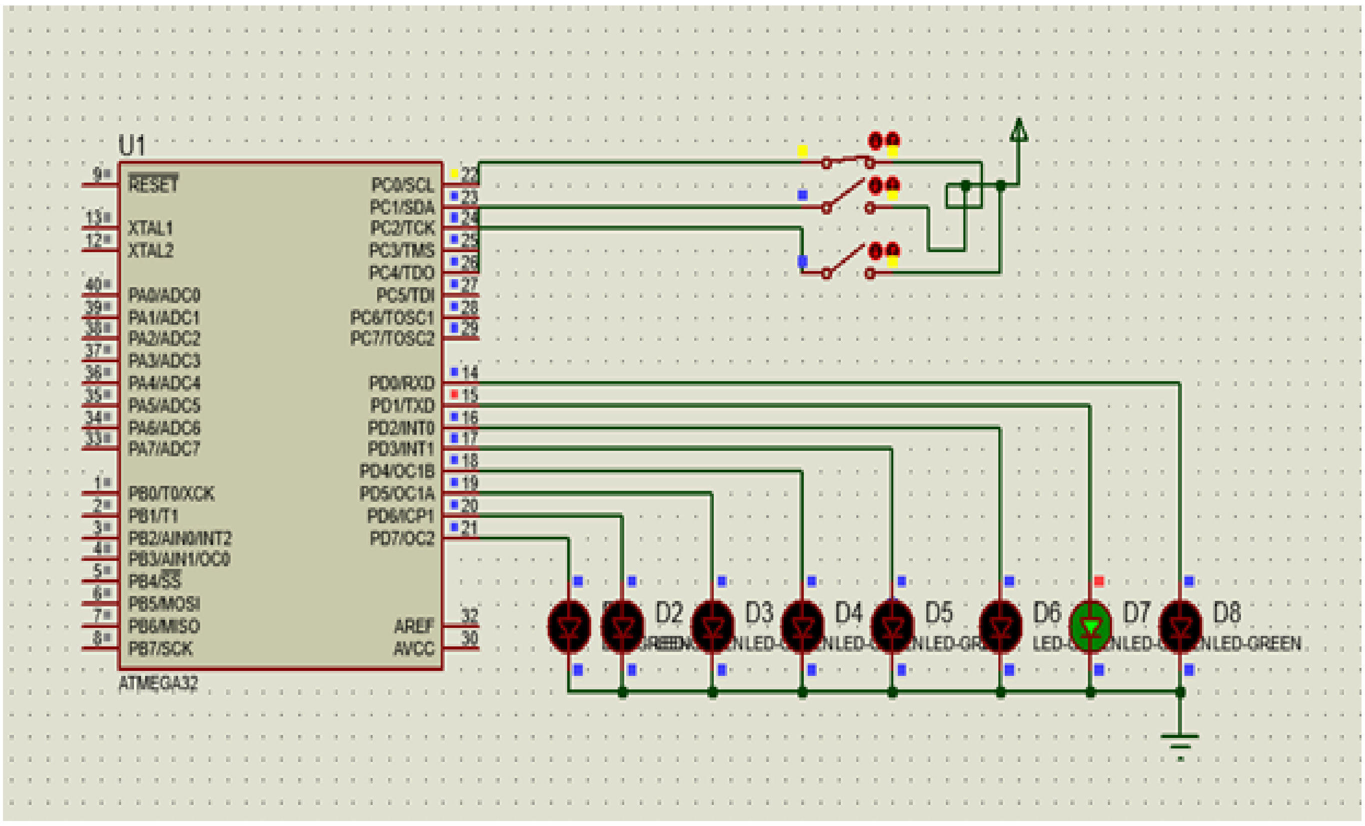


Fig. 16: Blinking 8 LEDs by switches

- **Task 3: Waveform Generation using timers**
  To generate waveform through oscilloscope with the frequency of wave generated 31250 Hz, non-inverted and the frequency of Atmega32 is 8 MHz with timer0:

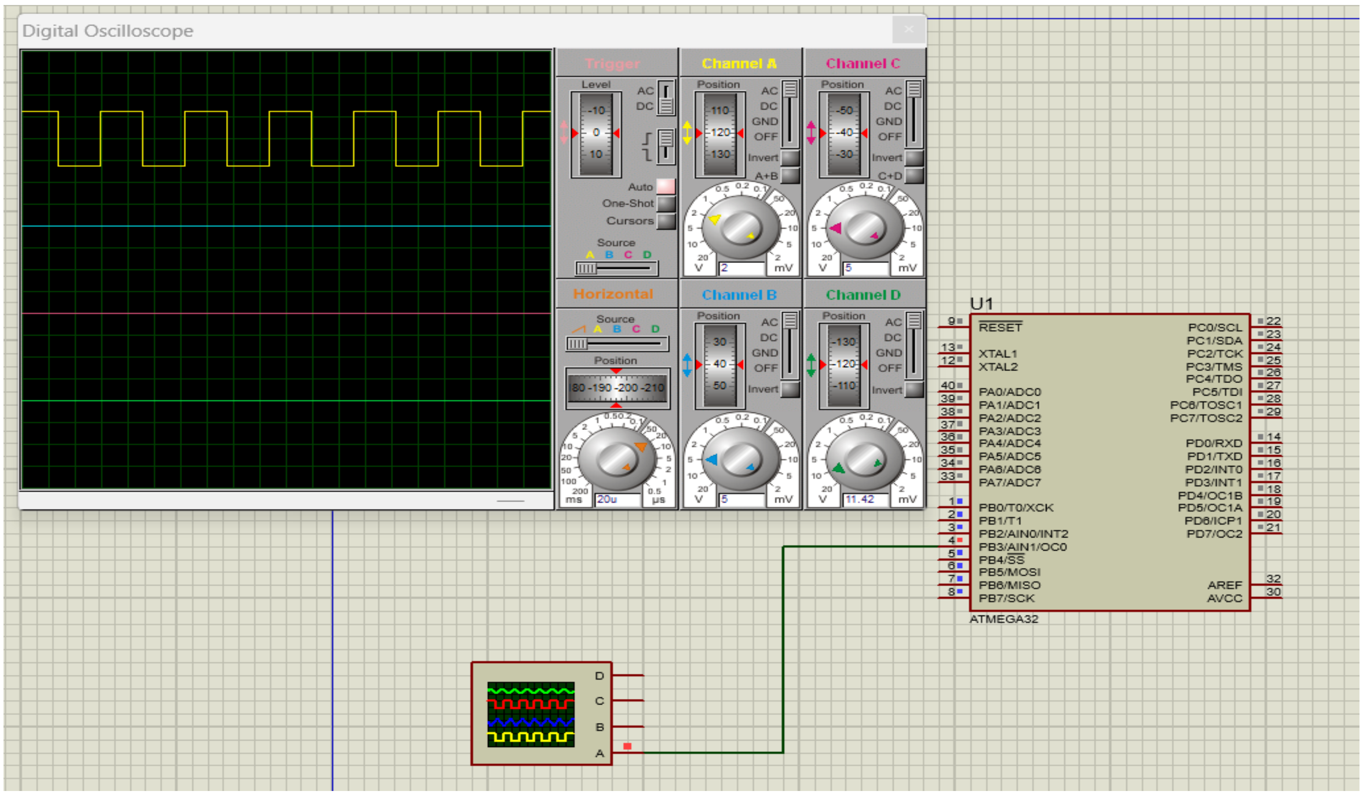


Fig. 17: Duty Cycle 50%

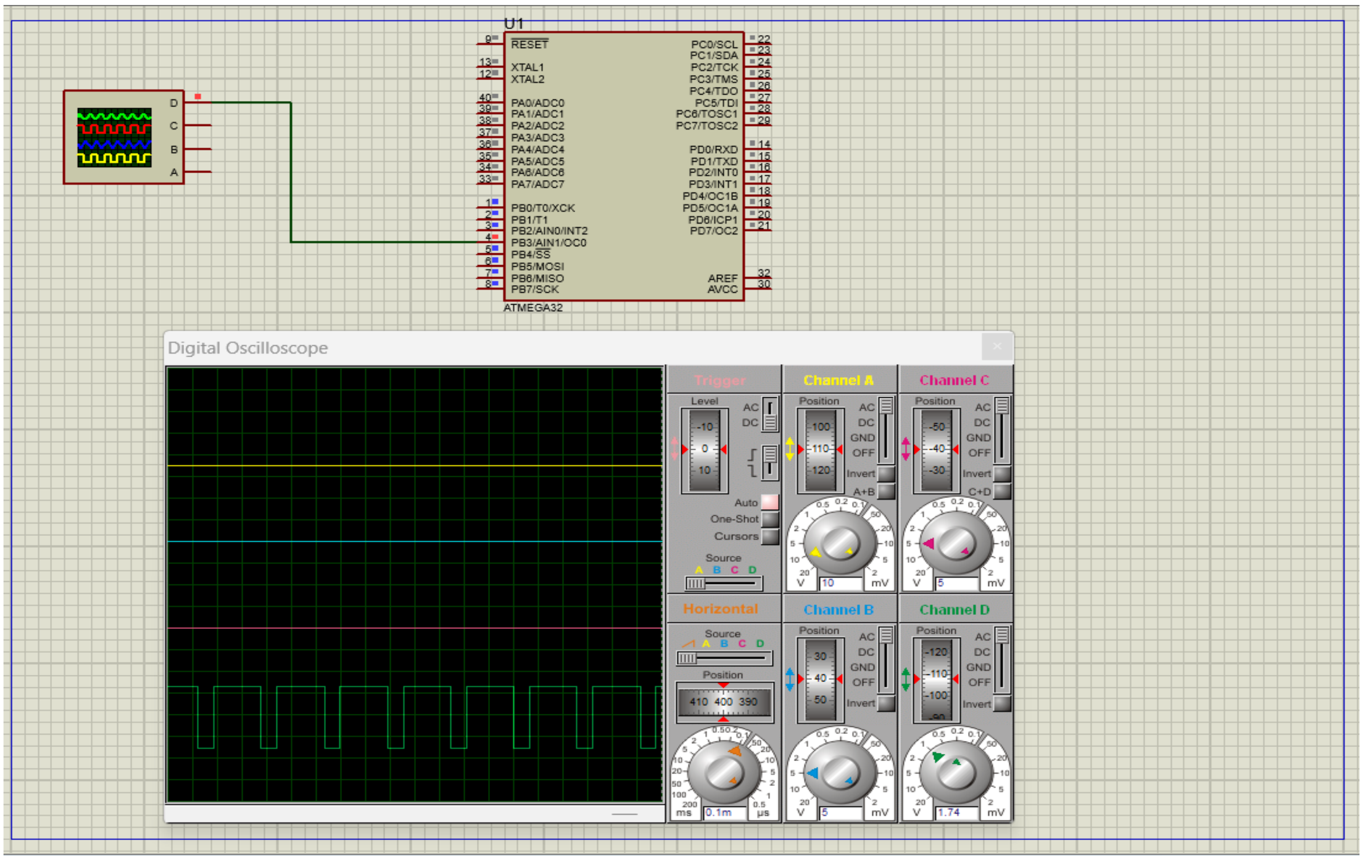


Fig. 18: Duty Cycle 75%

To generate waveform on oscilloscope with the frequency of wave generated 31250 Hz, duty cycle 50 percent, non-inverted and the frequency of Atmega is 8 MHz with timer0, timer1, timer2:

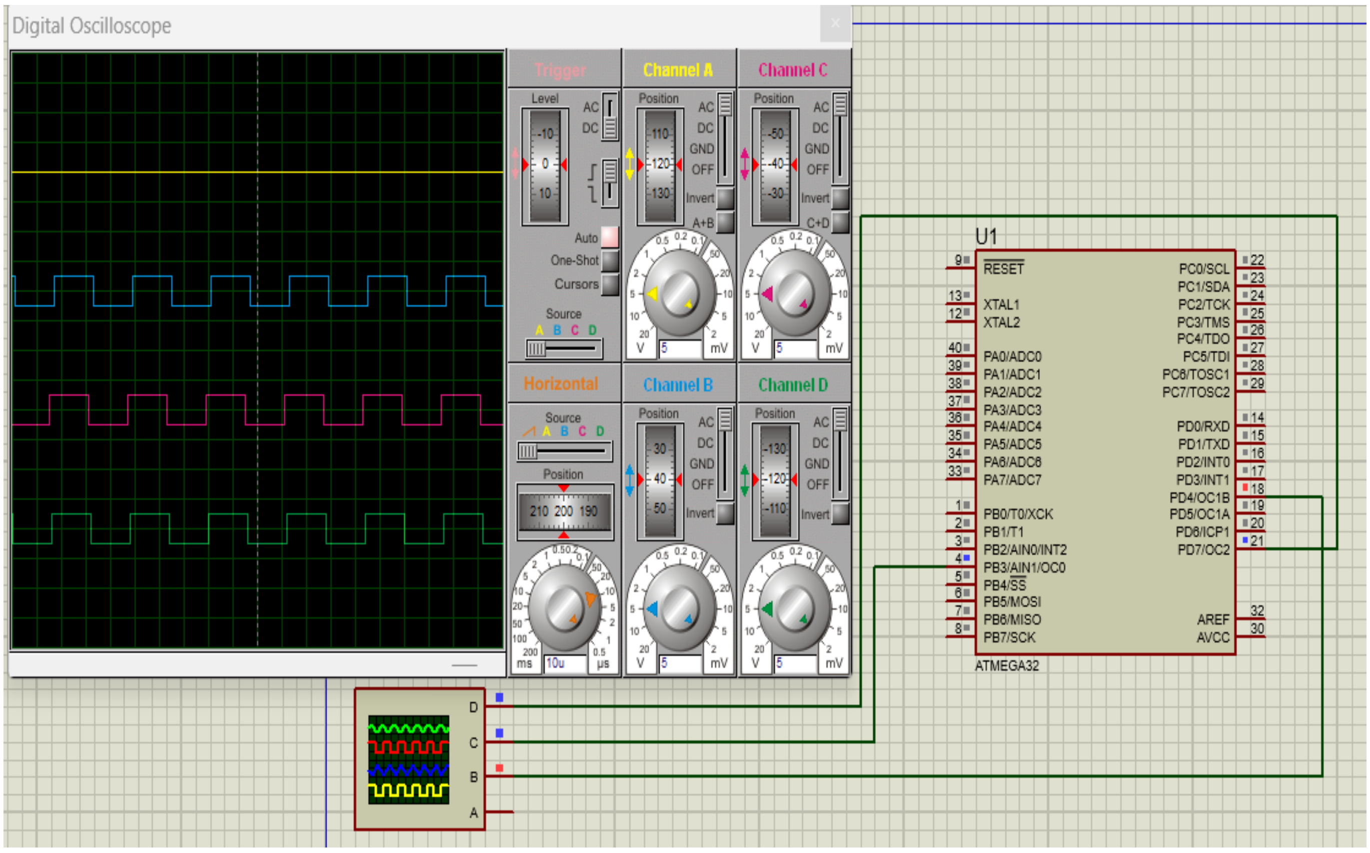


Fig. 19

- **Task 4: By using timer 0 timer 1 timer 2, change the shape of waveform using push button to select between timer**

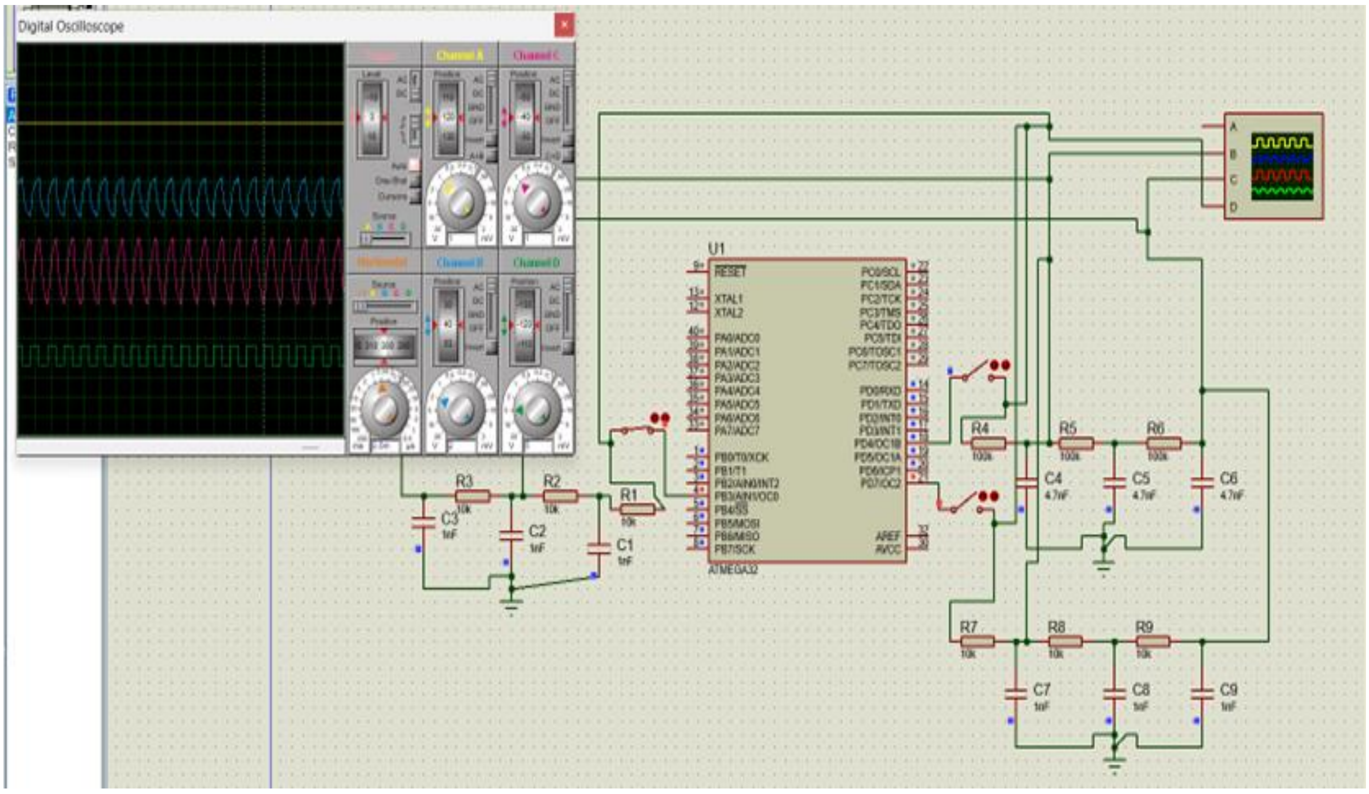


Fig. 20

- **Task 5: Amplify the provided signal by designing an amplifier through RC circuit:**

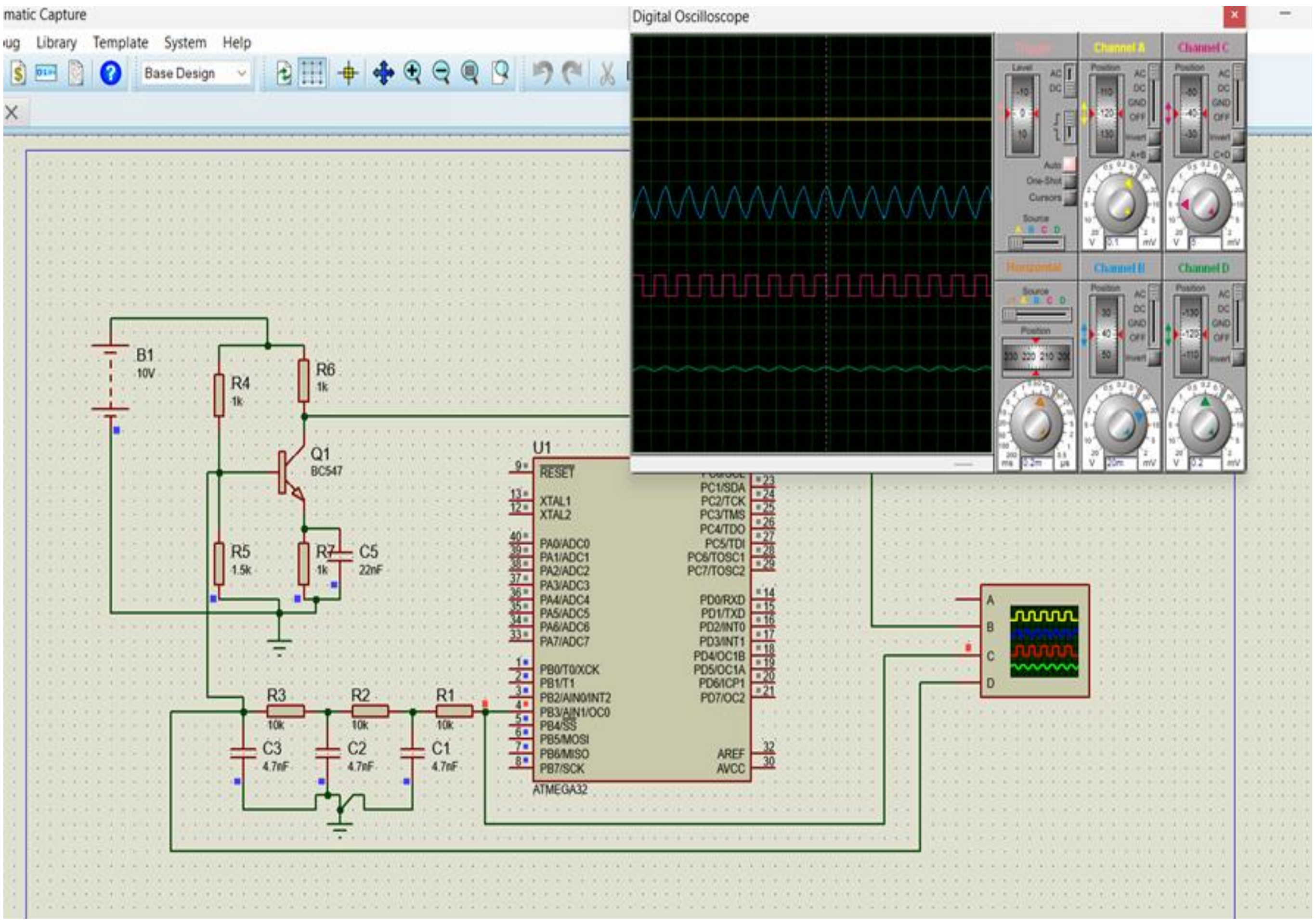


Fig. 21

- **Task 6: LCD Interfacing with 8-bit without port expander**

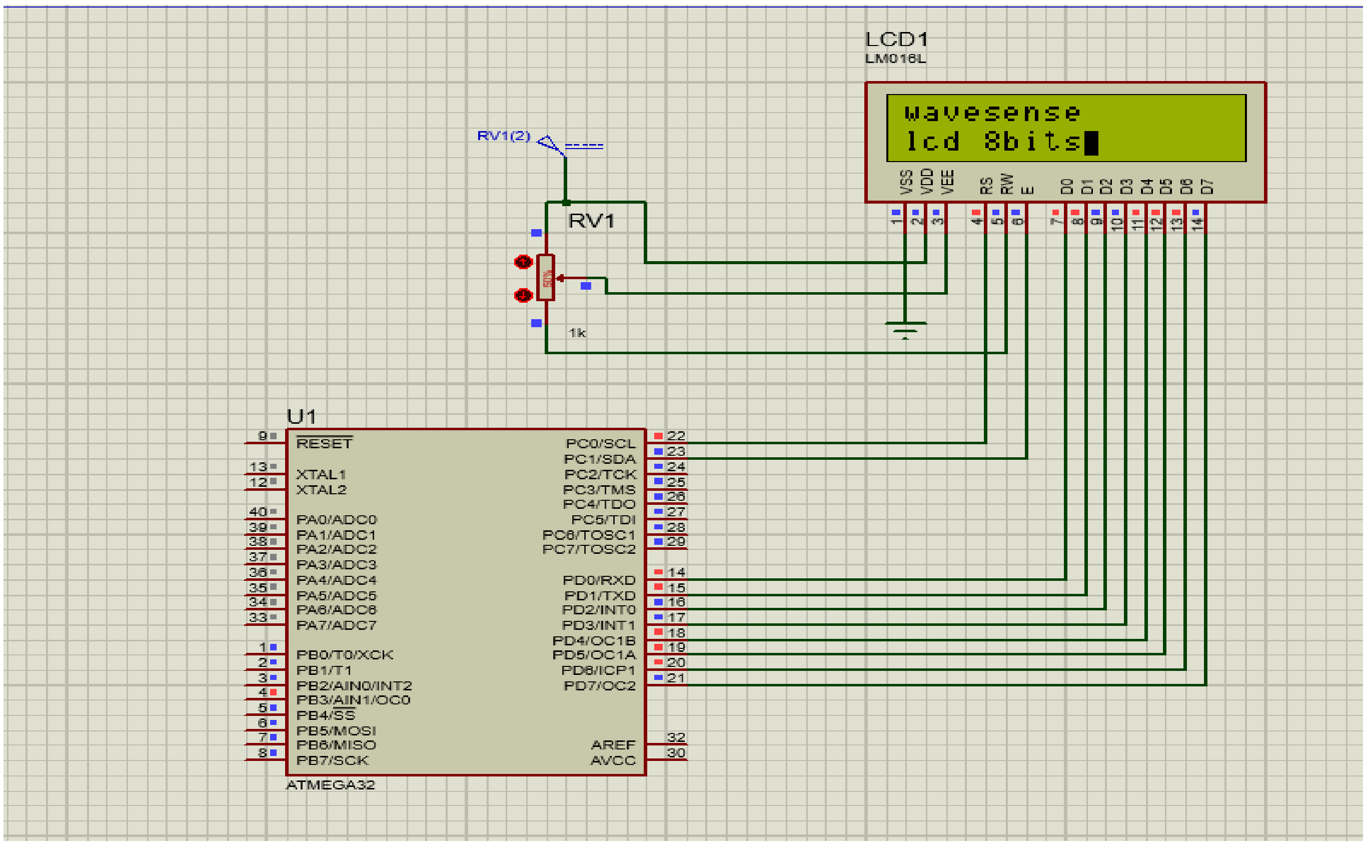


Fig. 22

- **Task 7: LCD Interfacing 4 bit without port expander**

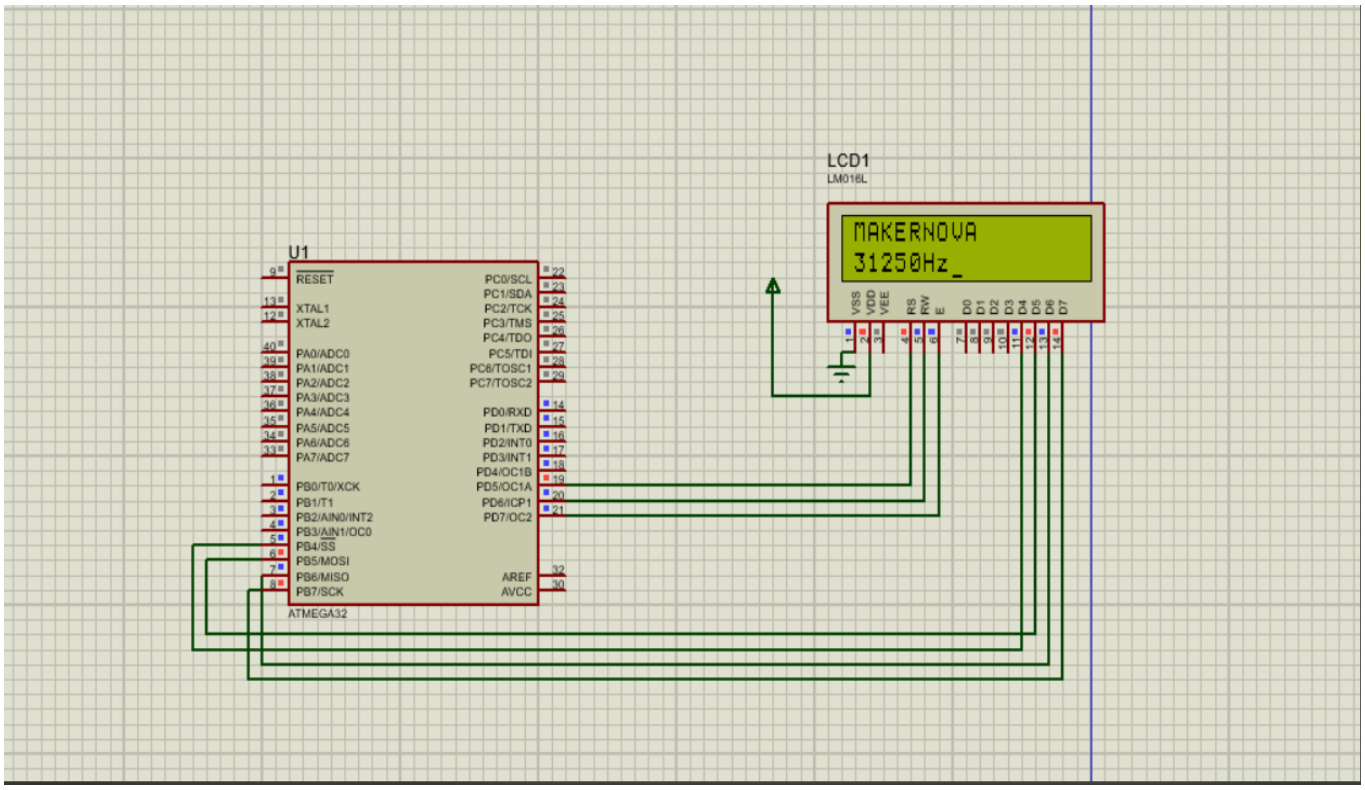


Fig. 23

- **Task 8: LCD interfacing 4 bit with port expander**

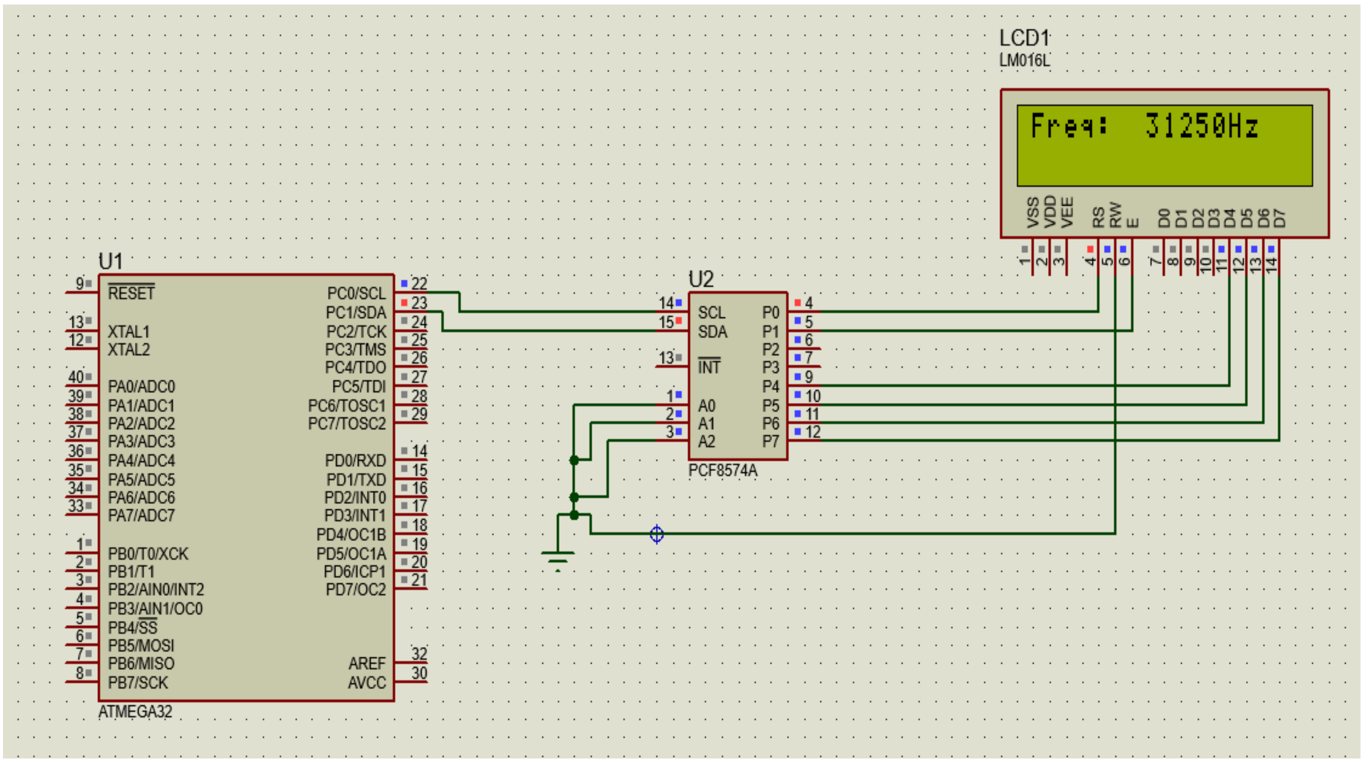


Fig. 24

- **Task 9: Generation of different waveforms using timer 0 and displaying the wave frequency on LCD**

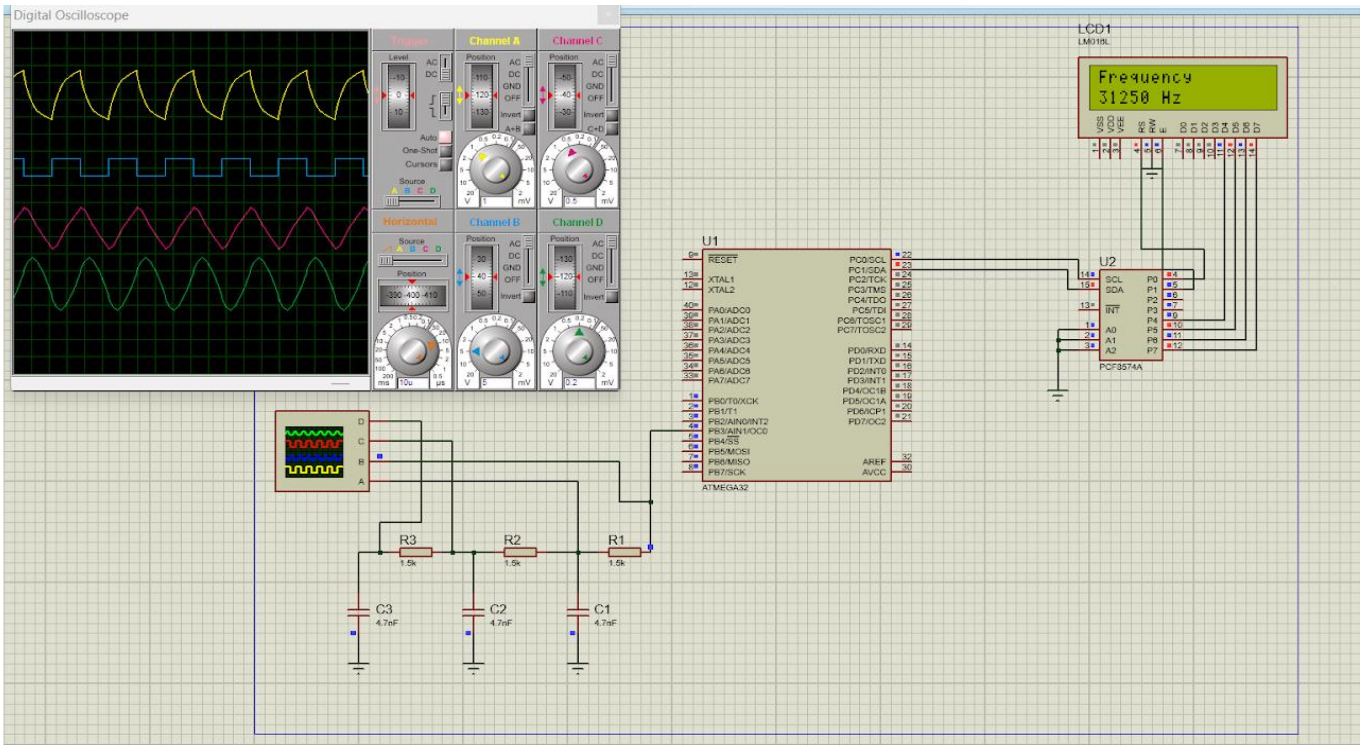
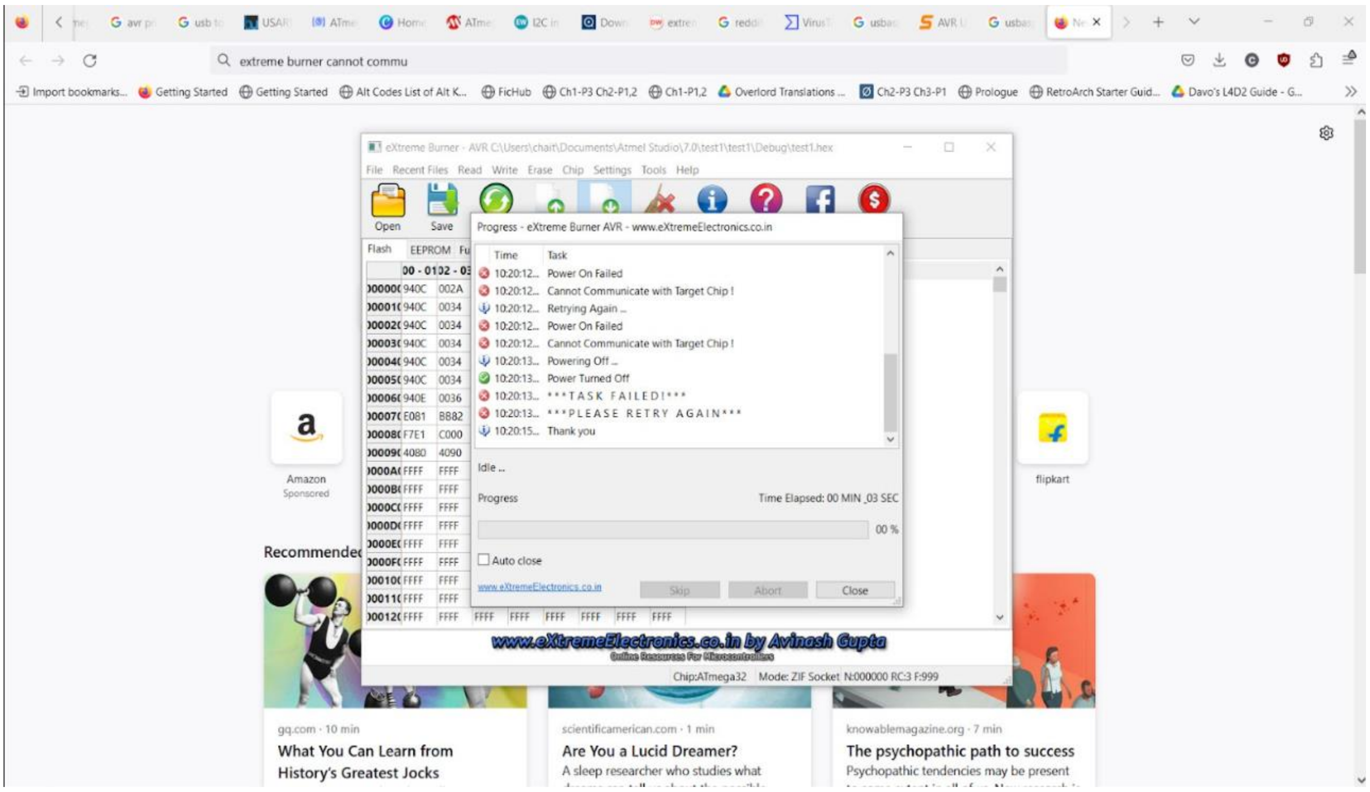


Fig. 25

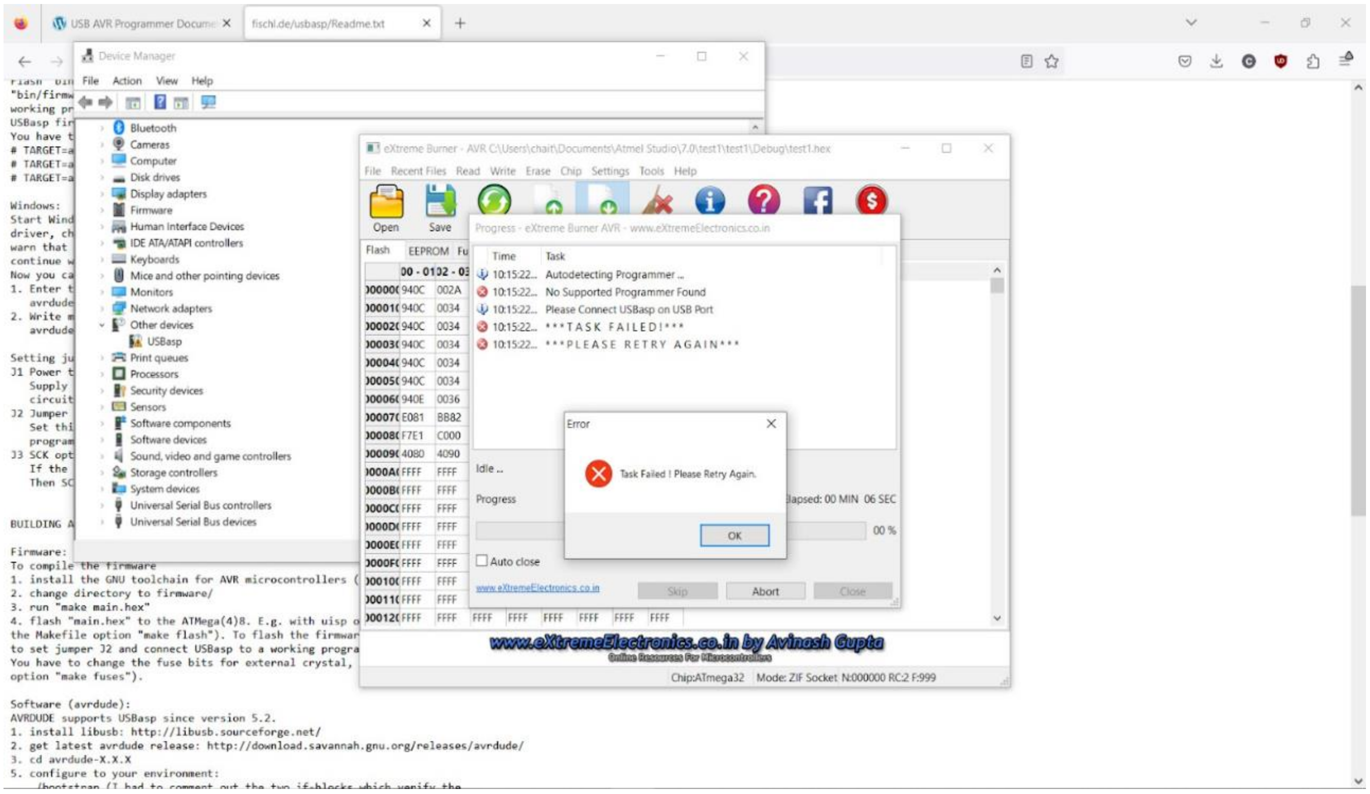
ERRORS AND THEIR SOLUTIONS

- ERROR1:
  No libraries found error at the time proteus software setup.
  Solution: In properties section of proteus software check the security and compatibility section.
- ERROR2:
  In LED blinking task, middle LEDS were not glowing.
  Solution: The reason was all LEDS were not grounded.
- ERROR3:
  In insertion three buttons task, due to error in iteration in for loop it was behaving like infinite loop due to which next switch was not working.
- ERROR4:
  In generation, conversion and amplification of wave with all the three timers we were getting distorted wave due to current flow in complete circuit that's why we introduced new switches to control the flow of current in other two circuit. In amplification we were not getting desired amplitude because of short bandwidth.
  Solution: Through calculations we set proper value of R and C so that we get proper bandwidth.
- ERROR5:
  In generation of different waveforms using timer 0 and displaying the wave frequency on LCD we were not getting proper sine wave due to incorrect values of R and C.
- ERROR6:
  In LCD interfacing with ATmega32, following were the errors:
  While turning on the USBASP, we got Power On failed error due to wrong connection of MOSI – MISO Pin. USBASP cannot communicate with target chip.



```
Progress - eXtreme Burner AVR - www.eXtremeElectronics.co.in
Time        Task
10:20:12... Power On Failed
10:20:12... Cannot Communicate with Target Chip !
10:20:12... Retrying Again ...
10:20:12... Power On Failed
10:20:12... Cannot Communicate with Target Chip !
10:20:13... Powering Off ...
10:20:13... Power Turned Off
10:20:13... ***TASK FAILED!***
10:20:13... ***PLEASE RETRY AGAIN***
10:20:15... Thank you
Idle ...
Progress                      Time Elapsed: 00 MIN ,03 SEC
00 %
Chip:ATmega32  Mode: ZIF Socket N:000000 RC:3 F:999
```

Fig. 26



```
Firmware:
To compile the firmware
1. install the GNU toolchain for AVR microcontrollers (
2. change directory to firmware/
3. run "make main.hex"
4. flash "main.hex" to the ATMega(4)8. E.g. with uisp o
the Makefile option "make flash"). To flash the firmwar
to set jumper J2 and connect USBasp to a working progra
You have to change the fuse bits for external crystal,
option "make fuses").

Software (avrdude):
AVRDUDE supports USBasp since version 5.2.
1. install libusb: http://libusb.sourceforge.net/
2. get latest avrdude release: http://download.savannah.gnu.org/releases/avrdude/
3. cd avrdude-X.X.X
5. configure to your environment:
```

```
Progress - eXtreme Burner AVR - www.eXtremeElectronics.co.in
Time        Task
10:15:22... Autodetecting Programmer ...
10:15:22... No Supported Programmer Found
10:15:22... Please Connect USBasp on USB Port
10:15:22... ***TASK FAILED!***
10:15:22... ***PLEASE RETRY AGAIN***

Error
Task Failed ! Please Retry Again.
OK

Chip:ATmega32  Mode: ZIF Socket N:000000 RC:2 F:999
```

Fig. 27

- ➢ Mismatch location with target chip due to wrong address of Port expander in code and sometimes due to loosen connection.
- ➢ Not getting any text on LCD due to wrong address of I2C (Port Expander).
- ➢ Cannot see the output properly so we attached a potentiometer to change the LCD contrast.

  Solution: Downloading USBASP driver with the help of Zadig software.

ALGORITHM OF FINAL CODE

1. Set PORTB as output.
2. Enter an infinite loop.
3. Set PORTB bit 4 to high.
4. Call the T1delay() function to generate a delay.
5. Set PORTB bit 4 to low.
6. Call the T1delay() function to generate a delay.
7. Start an infinite loop.
8. Configure Timer0 in CTC mode and with a pre-scalar of 256.
9. Wait for the Timer0 compare match flag (OCF0) to be set.
10. Stop Timer0 and clear the compare match flag.
11. Repeat from step 8.
12. Initialize variables and perform necessary initializations.
13. Enter an infinite loop.
14. Configure Timer0 and Timer1 for frequency measurement.
15. Measure the period of the input signal using Timer1's Input Capture functionality.
16. Calculate the frequency based on the measured period.
17. I2C addresses the port expander for sending the frequency value data to the LCD.
18. Display the frequency on an LCD screen.
19. Clear the LCD screen.
20. Repeat from step 15.

APPLICATIONS

- It can be used to generate square waves which are used to control the timing of operations in digital systems, such as clock generators for microprocessors.
- It can be used to predict the nature of a wave and its frequency band based on the frequency measured by the microcontroller.
- Our project also deals with the amplification of sine and triangular waves to increase the amplitude and energy of those waves.

REFERENCES

- Datasheet Atmega32
- AVR Microcontrollers tutorials `
- Understanding I2C Bus
- Basic operations of Atmega32
- Input Capture Mode