ACKNOWLEDGEMENT

We express our heartful thanks to **Mr. A. Mahesh Reddy, Assistant Professor**, Department of Electronic and Communication Engineering. Siddharth institute of Engineering and Technology, for her generous help, valuable guidance, and cooperation in completing our project successfully.

We express our deep sense of gratitude and respect to **Dr. P. Ratna Kamala,** Professor & Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work. We are very much thankful to the **Principal and Management,(SIETK), Puttur,** for their encouragement and cooperation to carry out this work. We express our thanks to all teaching and non-teaching staff of Department of ECE, for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully

PROJECT STUDENTS

M GURU PRASAD (21F61A0454)

T GURU PRASAD (21F61A0455)

M GURUNADHAM (21F61A0456)

C HARI BABU (21F61A0457)

Y HARI KUMAR (21F61A0458)

ABSTRACT

In this project, we designed an Arduino based Real Time Clock with alarm. A Real Time Clock or RTC is a battery powered clock that measures time even when there is no external power or the microcontroller is reprogrammed. An RTC displays clock and calendar with all timekeeping functions. The battery, which is connected to the RTC is a separate one and is not related or connected to the main power supply.

When the power is restored, RTC displays the real time irrespective of the duration for which the power is off. Such Real Time Clocks are commonly found in computers and are often referred to as just CMOS . Most microcontrollers and microprocessors have built in timers for keeping time. But they work only when the microcontroller is connected to power supply. When the power is turned on, the internal timers reset to 0. Hence, a separate RTC chip is included in applications like data loggers for example, which doesn't reset to 0 when the power is turned off or reset. Real Time Clocks are often useful in data logging applications, time stamps, alarms, timers, clock builds etc. In this project, a Real Time Clock, which displays accurate time and date along with an alarm feature is designed. In this project an attempt is made to develop and explain the use of Digital alarm clock using Arduino.

Keywords: RTC Module, Arduino, Clock, Power Backup

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A digital clock is a great invention in electronics science. Nowadays, digital clocks are used everywhere. The analog clocks are quite old-fashioned. So the digital clock takes its place day by day. The Arduino digital clock is looking very modern. It has many additional features also like temperature, alarm, timer, etc

In this Electronic project, we are going to build an Arduino digital clock with or without an RTC (Real Time Clock) module. The first circuit represents without RTC module and the second circuit represents with RTC module. The components we used in this project are quite basic and a little expensive. The circuit connection is very easy.

In this project, we are going to make **DIGITAL CLOCK ALARM** using the DS3231 real time clock module. This module is very cheap and works through I2C communication, which makes it easy to use with the microcontrollers. We will get the time using this module and will make the buzzer beep after comparing the current time with the alarm time. This module also tells us the temperature.

CHAPTER 2

METHODOLOGY

The Arduino powers the system with its +5-volt supply, and it controls all of the external components in the system. The Arduino uses the I2C communication protocol to talk with the real-time clock, and it uses the SPI communication to interface with the 7-segment binary coded decimal decoder chip.

A digital timer will display the time by default for24 hours 60 minutes and 60 seconds. This timer can display the actual time in hours and minutes and seconds whereas the traditional clock is not accurate than the digital clock. A timer is a device that shows the time in each aspect like hours, minutes and seconds too.

We build this digital clock with an Arduino, RTC module, and LCD display in this project. Here the clock we made is 24 hours clock. This means the time shows by the clock is 00.00 AM to 23.59. After 23.59 it resets to 0 again.

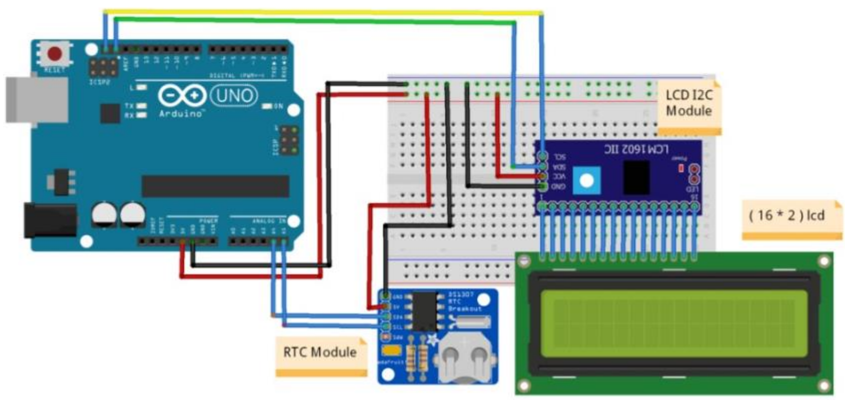


Fig 2.1: Circuit Diagram

CHAPTER 3

SOFTWARE DESCRIPTION

Step 1− First you must have your Arduino board (you can choose your favorite board) and a USBcable.IncaseyouuseArduinoUNO,ArduinoDuemilanove,Nano,ArduinoMega2560, or Diecimal, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

Fig4.1:USB cable

In case you use Arduino Nano, you will need an A to Mini-B cable instead as shown in the following image.

Fig4.2:AtoMini-Bcable

Step2−DownloadArduinoIDESoftware.

You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file downloadis complete, unzip the file.

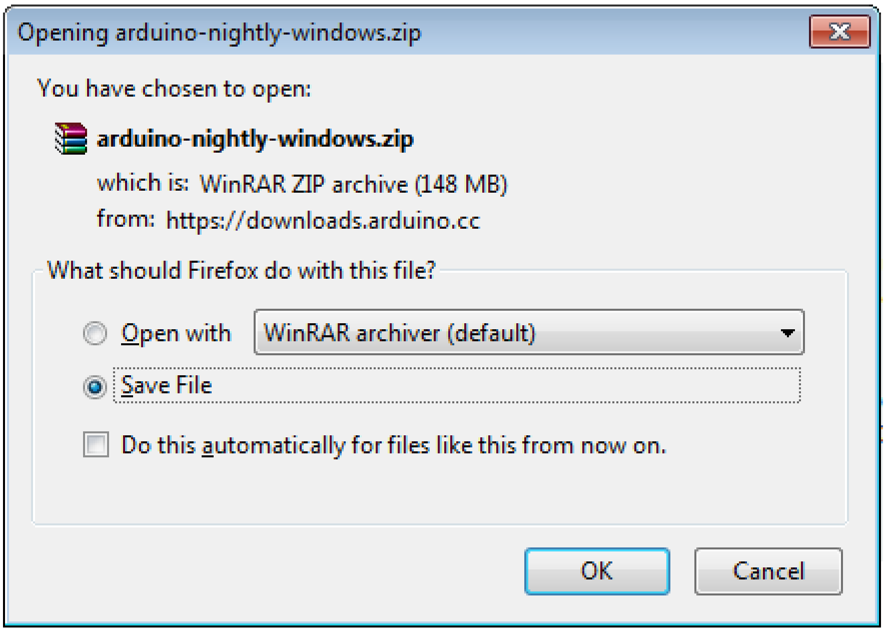


Fig4.3:Un zipping

Step3–Power up your board.

The Arduino Uno, Mega, and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an ArduinoDiecimila,youhavetomakesurethattheboardisconfiguredtodrawpower from the USB connection.The power source isselected with a jumper, a small piece of plastic that fits onto two ofthe three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port.

Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.

Step4–LaunchArduinoIDE.

After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you canfindtheapplicationiconwithaninfinitylabel(application.exe).Double-clicktheicon to start the IDE.

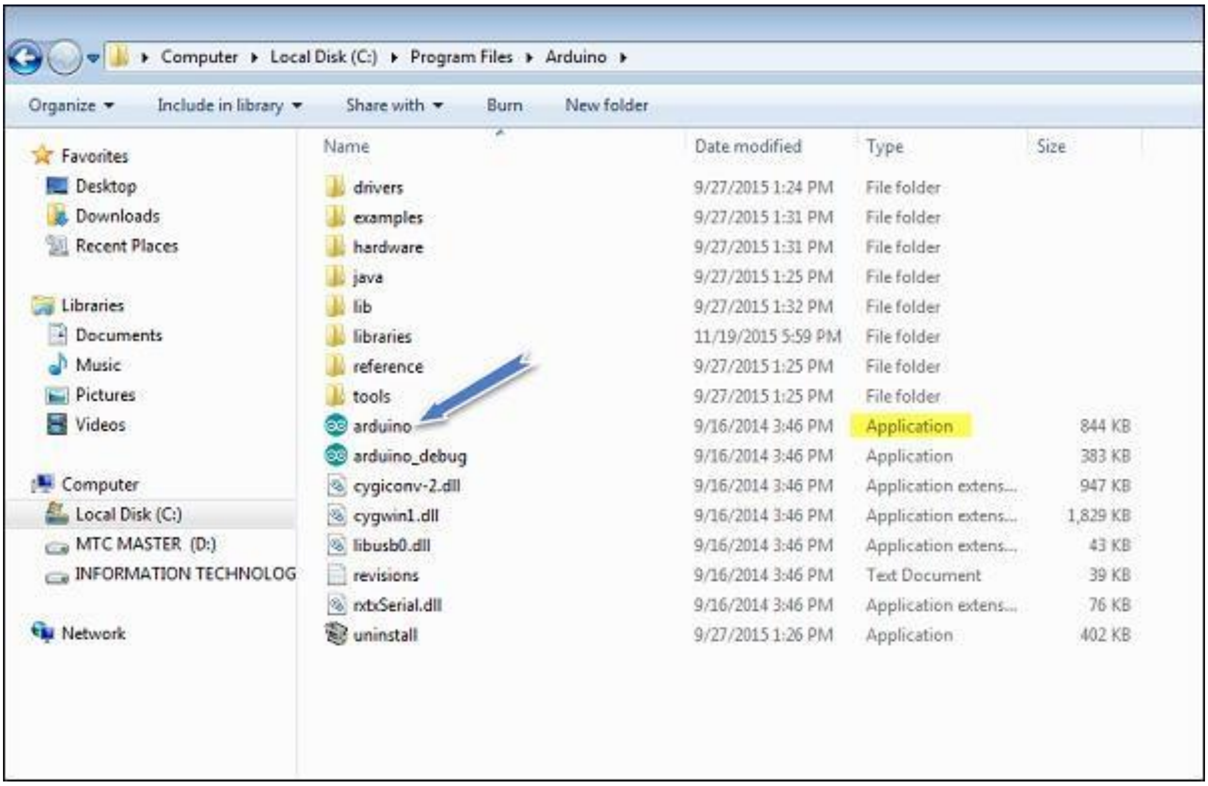


Fig4.4:OpeningArduino Step

5 – Open your first project.

Once the software starts, you have two options−

- Create a new project.
- Open an existing projectexample

To create a new project, select File → **New**.

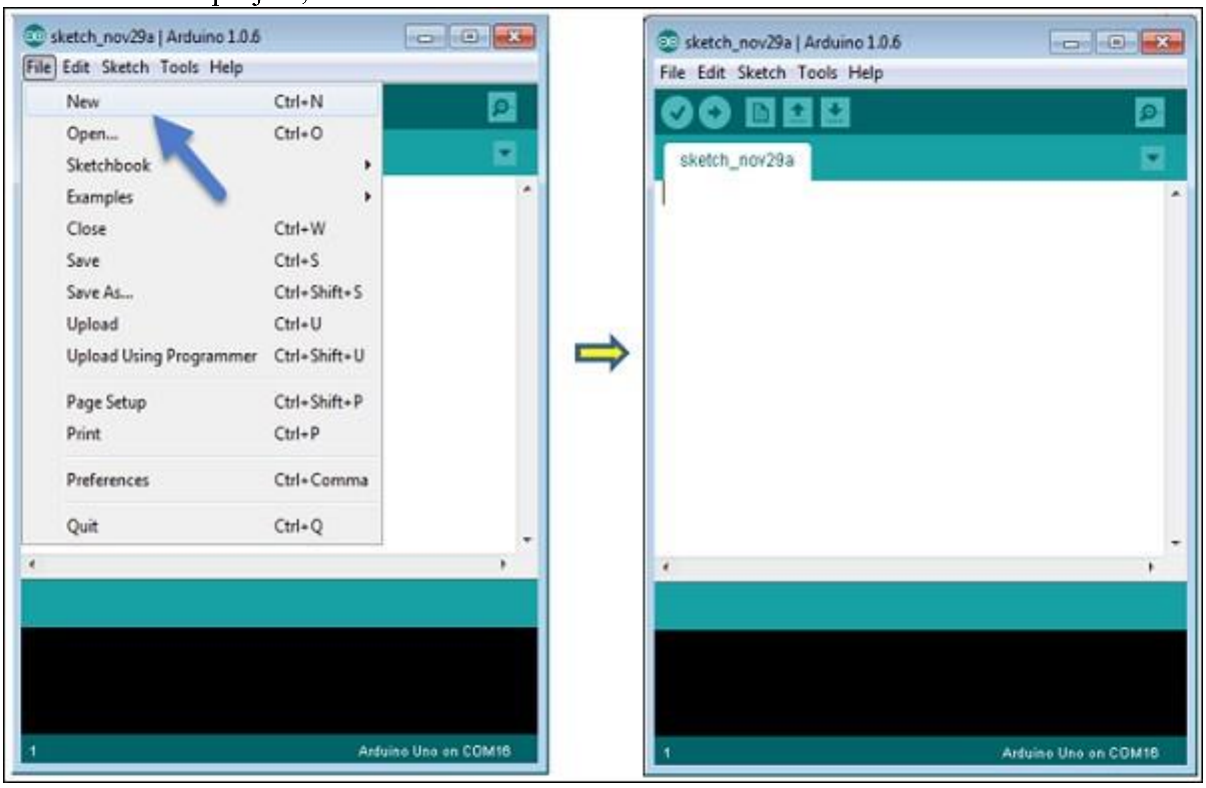


Fig4.5:Opening new file

To open an existing project example, selectFile→Example→Basics→Blink.

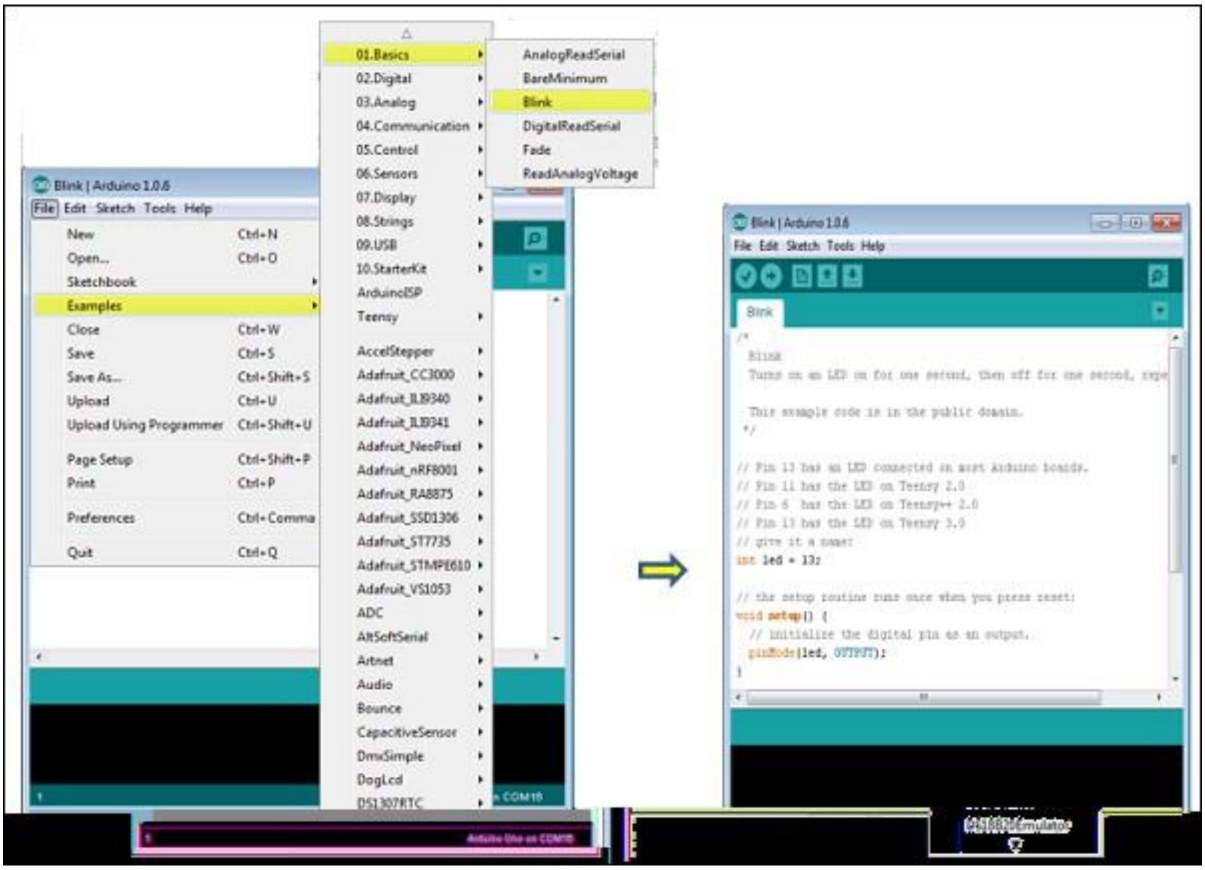


Fig4.6:Examples

Here, we are selecting just one of the examples with the name **Blink**. It turns the LED on and off with some time delay. You can select any other example from the list.

Step6–SelectyourArduinoboard.

To avoid any error while uploading your program to the board ,you must select the correct Arduino board name, which matches with the board connected to your computer.

Goto Tools→ Board and selecty our board.

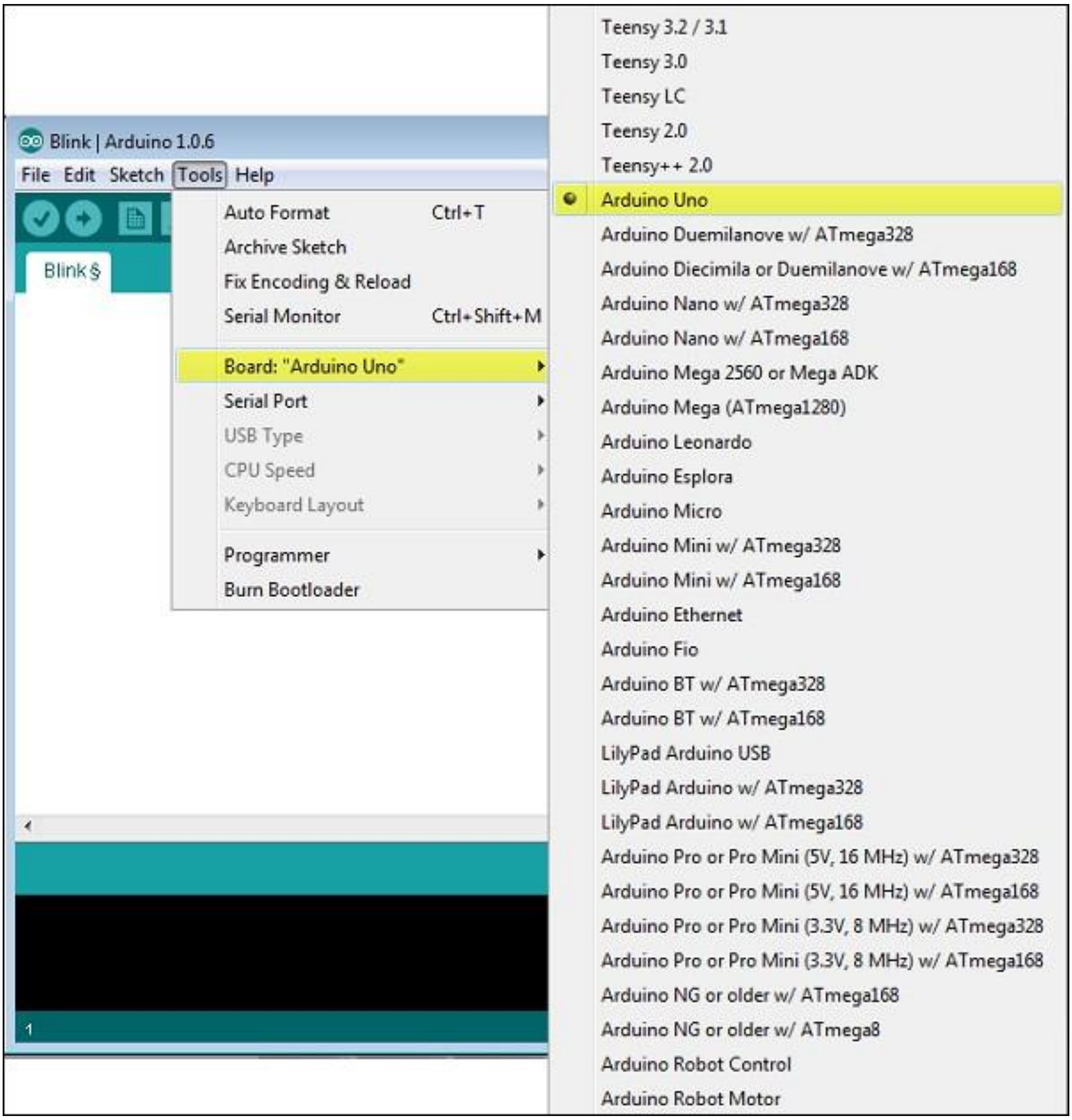


Fig4.7:SelectionofArduinoboard

Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.

Step7–Selectyourserialport.

Select the serial device ofthe Arduino board. Go to**Tools→ Serial Port**menu. Thisislikely to beCOM3 or higher (COM1 and COM2 areusually reserved for hardware serial ports). To find out, you can disconnect your Arduino board andre-open the menu, the entry that disappears should be ofthe Arduino board. Reconnect the boardand select that serial port.

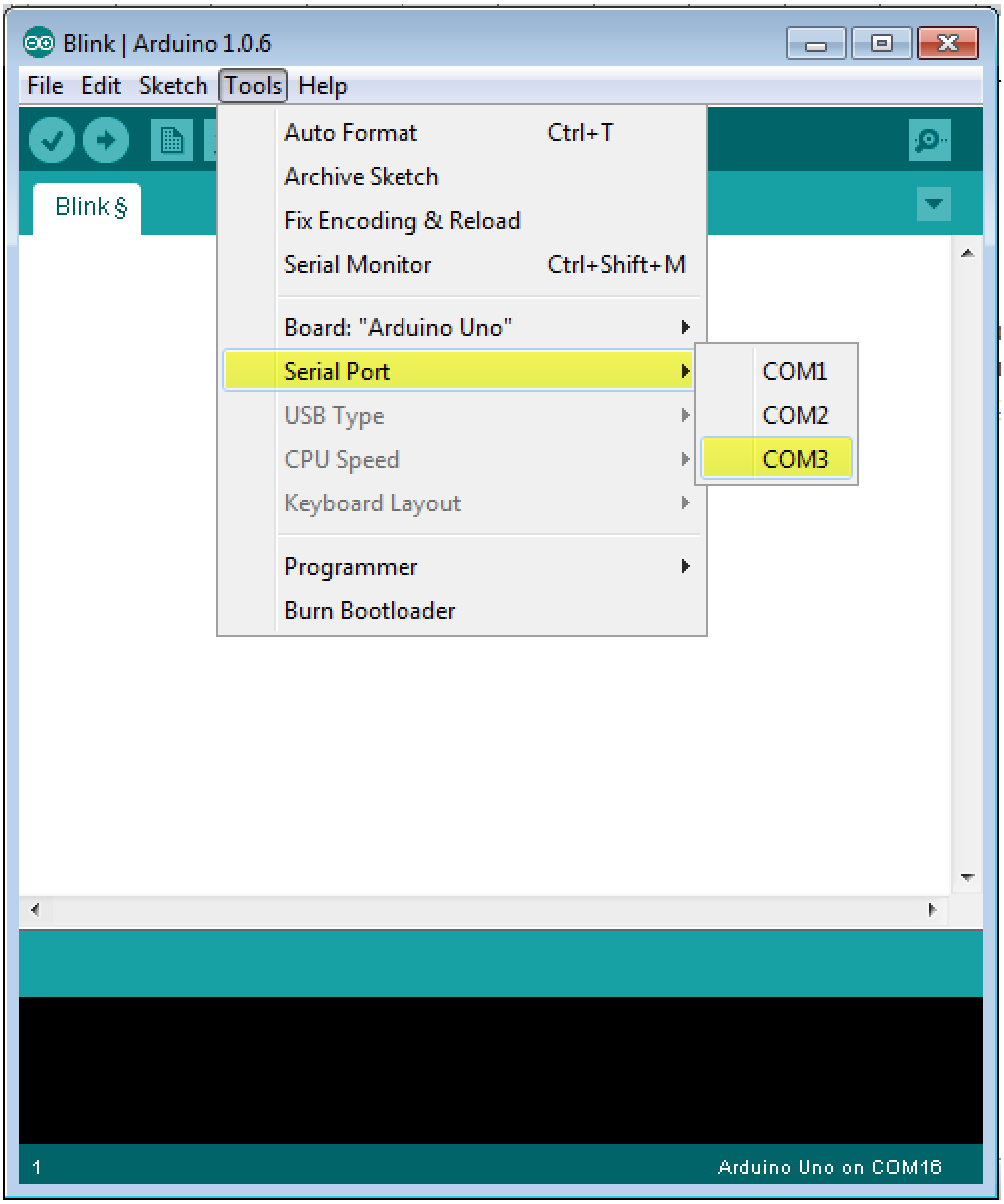
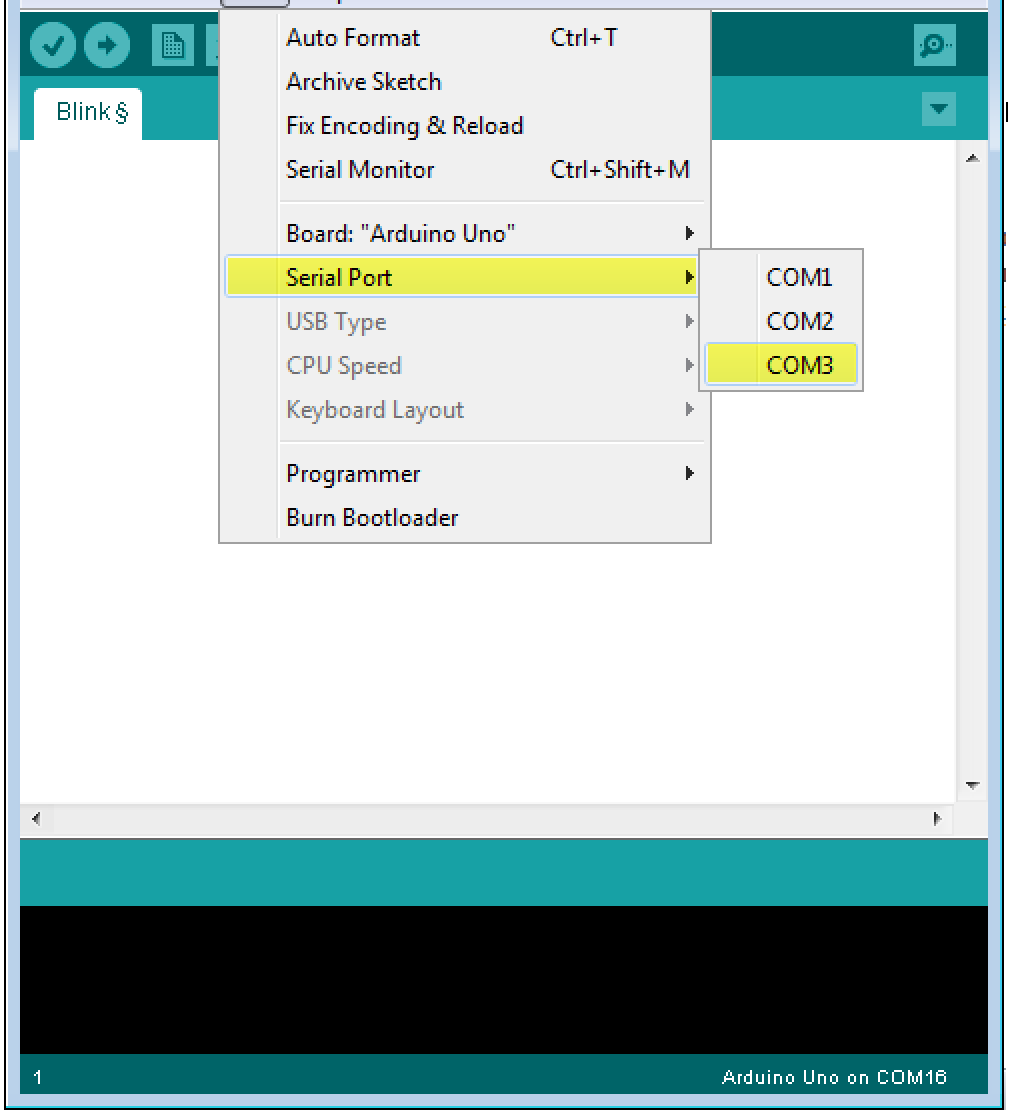


Fig4.8:Portselection Step

8 – Upload the program to your board.

Beforeexplaininghowwecanuploadourprogramtotheboard,wemustdemonstratethe function of each symbol appearing in the Arduino IDE toolbar.



l clock alarm

Fig4.8:Portselection Step

8 – Upload the program to your board.

Beforeexplaininghowwecanuploadourprogramtotheboard,wemustdemonstratethe function of each symbol appearing in the Arduino IDE toolbar.

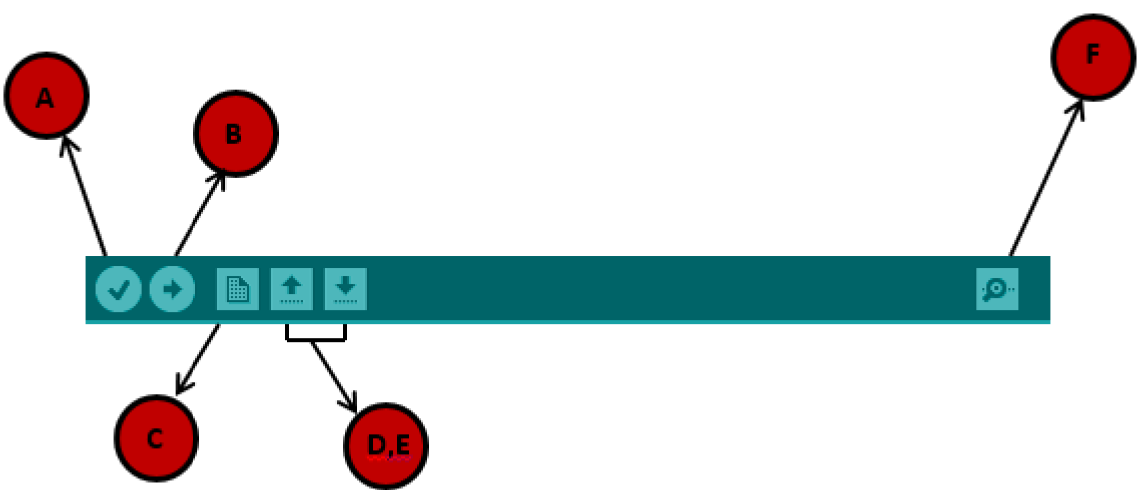


Fig4.9:Toolbar

A–Used to check if there is any compilation error.

B–Used to upload a program to the Arduino board.

C–Shortcut used to create a new sketch.

D–Used to directly open one of the exam ple sketch.

E–Used to save your sketch.

F– Serial monitor used to receive serial data from the board and send the serial data to the board.

Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar.

CHAPTER 4

HARDWARE DESCRIPTION

Arduino

The Arduino micro controller is an easy to use yet powerful single board computer that has gainconsiderable traction in the hobby and professional market. The Arduino is open-source, which meanshardware is reasonably priced and development software is free. This guide is for students in ME 2011,or students anywhere who are confronting the Arduino for the first time. For advanced Arduino users,prowltheweb,there are lotsofresources.Thisiswhatthe Arduinoboardlookslike.

Fig4.1:ArduinoUNOBoard

TheSpecificationFeaturesofArduinoUnoare:

- Microcontroller:Microchip ATmega328P
- OperatingVoltage:5Volts
- Input Voltage:7to20 Volts
- DigitalI/OPins:14 (ofwhich6 canprovidePWM output)
- PWMPins:6(Pin#3,5,6,9,10and11)UART:1
- 12C: 1
- SPL
- AnalogInput Pins:6
- DC Current perI/OPin:20Ma
- DC Current for3.3VPin:50mA
- FlashMemory:32 KBofwhich0.5KBusedbybootloader
- SRAM:2KB
- EEPROM:1KB
- ClockSpeed:16MHz
- PowerSources:DC PowerJack& USBPort

16*2 LCD DISPLAY

The term LCD stands for liquid crystaldisplay. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

Fig 4.2:16X2 LCD

LCD 16×2 Pin Diagram

The 16×2 LCD pinout is shown below.

Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.

Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.

Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.

Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).

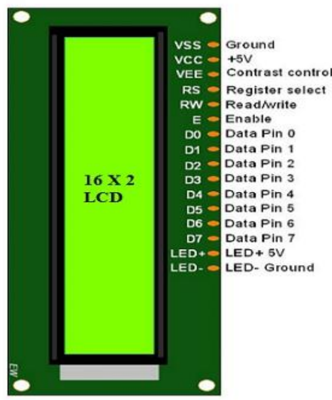
Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).

Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.

Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.

Pin15 (+ve pin of the LED): This pin is connected to +5V

Pin 16 (-ve pin of the LED): This pin is connected to GND.



LCD-pin-diagram

Features of LCD16x2

- The features of this LCD mainly include the following.
- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit
- These are obtainable in Blue & Green Backlight
- It displays a few custom generated characters

I2C MODULE

I2C Module has a inbuilt PCF8574 I2C chip that converts I2C serial data to parallel data for the LCD display.

These modules are currently supplied with a default I2C address of either 0x27 or 0x3F. To determine which version you have check the black I2C adaptor board on the underside of the module. If there a 3 sets of pads labelled A0, A1, & A2 then the default address will be 0x3F. If there are no pads the default address will be 0x27.

The module has a contrast adjustment pot on the underside of the display. This may require adjusting for the screen to display text correctly.

fig 4.3: I2C Module

Features:-

- Operating Voltage: 5V
- Backlight and Contrast is adjusted by potentiometer
- Serial I2C control of LCD display using PCF8574
- Come with 2 IIC interface, which can be connected by Dupont Line or IIC dedicated cable
- Compatible for 16x2 LCD
- This is another great IIC/I2C/TWI/SPI Serial Interface
- With this I2C interface module, you will be able to realize data display via only 2 wires.

DS1307 Real time clock module

- Real Time Clock (RTC) is used to track the current time and date. It is generally used in computers, laptops, mobiles, embedded system applications devices etc.

- In many embedded system, we need to put time stamp while logging data i.e. sensor values, GPS coordinates etc. For getting timestamp, we need to use RTC (Real Time Clock).
- Some microcontrollers like LPC2148, LPC1768 etc., have on-chip RTC. But in other microcontrollers like PIC, ATmega16/32, they do not have on-chip RTC. So, we should use external RTC chip
- There are different types of ICs used for RTC like DS1307, DS12C887 etc. In this section we will see DS1307.

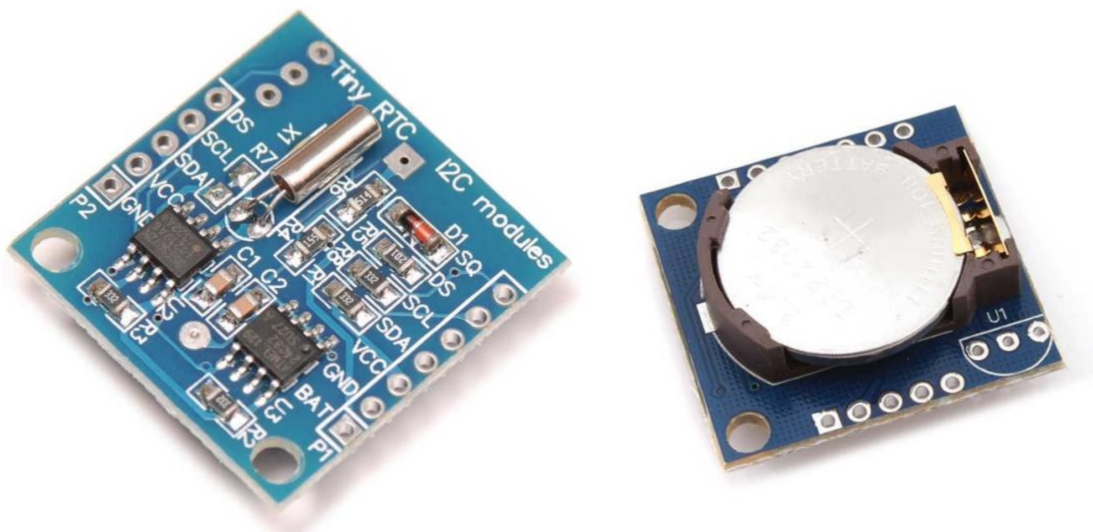


fig 4.4:DS1307 RTC Module

Push buttons

A Push Button switch is **a type of switch which consists of a simple electric mechanism or air switch mechanism to turn something on or off**. Depending on model they could operate with momentary or latching action function. The button itself is usually constructed of a strong durable material such as metal or plastic

Push Button Switches come in a range of shapes and sizes. We have a selection of push button switches here at Harga.

Push button switches are used throughout industrial and medical applications and are also recognisable in everyday life.

For uses within the Industrial sector, push buttons are often part of a bigger system and are connected through a mechanical linkage. This means that when a button is pressed it can cause another button to release.

Fig 4.5: Push button

Buzzer

An audio signalling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

Buzzer Pin Configuration

Fig 4.6: Buzzer

The pin configuration of the buzzer is shown below. It includes two pins namely positive and negative. The positive terminal of this is represented with the '+' symbol or a longer terminal. This terminal is powered through 6Volts whereas the negative terminal is represented with the '-'symbol or short terminal and it is connected to the GND terminal.

9V Battery

This is General purpose 9V Original HW marked Non-Rechargeable Battery for all your project and application needs. As we experienced the use of this battery in our testing lab for various purposes, we can assure you the best quality, long life and genuineness of this battery among all the options available in the market at this cost. With its Universal 9V battery size and connecting points, it can be used in many DIY projects as well as household applications and they can easily be replaced and installed, the same as you would an AA battery or a AAA battery.

Whether you need a new battery for your applications like a Flashlight, Portable Phone Charger, Wireless doorbell, Wireless audio transmitter-receiver systems or your kid's toys, etc. or even if you are looking for a long-lasting, reliable option for your sensor devices like a smoke detector, everyone needs a good 9-volt battery every once in a while. It's also a great idea to keep extra 9 volt batteries around in case of an emergency. That's why we've found one of the best 9-volt batteries available.

Specifications

Battery type: Zinc Carbon battery

Dimension: 26.5mm x 48.5mm x 17.5mm

Nominal voltage: 9V

Cut-off voltage: 5.4V

Discharge Resistance(Ω): 620

Capacity : 600 mAh

fig 4.7:9V Battery=

Fig 4.8 jumper wires

JumpWires

A jump wire (also known as jumper, jumper wire, DuPont wire) is an electrical wire, or group of themin a cable, with a connector or pin at each end (or sometimes without them – simply "tinned"), which isnormallyusedtointerconnectthecomponentsofa breadboard orotherprototypeortestcircuit,internallyorwithotherequipmentorcomponents,withoutsoldering.

Individualjumpwiresarefittedbyinsertingtheir"endconnectors"intotheslotsprovidedinabreadboard,theheaderconnector ofacircuitboard,orapiece oftestequipment.

Resistor

The resistor is a passive electrical component that creates resistance in the flow of electric current. In almost all electrical networks and electronic circuits they can be found. The resistance is measured in ohms (Ω). An ohm is the resistance that occurs when a current of one ampere (A) passes through a resistor with a one volt (V) drop across its terminals. The current is proportional to the voltage across the terminal ends. This ratio is represented by Ohm's law:

$$R=V/I$$

Resistors are used for many purposes. A few examples include limiting electric current, voltage division, heat generation, matching and loading circuits, gain control, and setting time constants. They are commercially available with resistance values over a range of more than nine orders of magnitude. They can be used as electric brakes to dissipate kinetic energy from trains, or be smaller than a square millimeter for electronics.

Fig 4.9: Resistors

Bread board

Breadboards are one of the most fundamental pieces when learning how to build circuits. In this tutorial, you will learn a little bit about what breadboards are, why they are called breadboards, and how to use one. Once you are done you should have a basic understanding of how breadboards work and be able to build a basic circuit on a breadboard.

1. Fig 4.10:Bread

WORKINGPRINCIPLE

Arduino-based clocks use the current time as a timer for reminders or to execute a scheduled command via the Arduino's I/O pins. [10:48 pm, 26/01/2024] Gurugoud: With a real-time clock module, this circuit is working in automatic mode. Although we can manually set the time as our requirements through Arduino

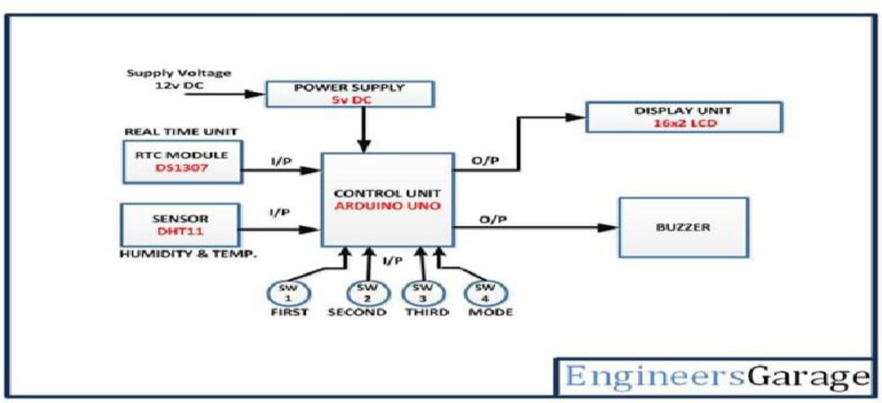
In more detail, the oscillator inside a digital clock or watch generates a precise electrical signal with a constant frequency, usually using a quartz crystal. This oscillator signal is then sent to a counter circuit that counts the number of oscillations and generates output signals that correspond to the time units.

Steps to make this project:

- **Gather components like Arduino Nano, Breadboard, LCD display, 10K potentiometer, Push buttons etc.**
- **Place Arduino Nano, LCD display and Push Buttons on Breadboard.**
- **Do connections according to Circuit Diagram.**

Upload the code.

- **Set time according to you: An accurate clock with date displayed on a 16x2 LCD using just the Arduino, the display and few buttons. No RTC module required. This is the script for Processing that will read the milliseconds sent from the Arduino and compare it to the elapsed milliseconds in processing.**



CODE :

```
// Real time clock and calendar with set buttons using DS3231 and Arduino

#include <Wire.h>        // include Wire library code (needed for I2C protocol devices)

#include <LiquidCrystal_I2C.h>
// include Wire library code (needed for I2C protocol devices)
LiquidCrystal_I2C lcd(0x27,16,2);

void setup()
{
Serial.begin(9600);
Wire.begin();          // Join i2c bus
lcd.init();
lcd.backlight();
pinMode(8, INPUT);
pinMode(9,OUTPUT);
}
char Time[] = "Time:        ";
char ti[] ="Time: HH:MM:SS";
char Calendar[]="Date:        ";
char ca[]="Date: DD/MM/YY";
```

```
byte i,pre, second, minute, hour,day, date, month, year;
char bs = "";
//String d,h;
int x=0,y=0,k=0,l=0,al=0;
byte h[2], m[2], d[2];

void DS3231_display()
{

// Convert BCD to decimal
second = (second >> 4) * 10 + (second & 0x0F);
minute = (minute >> 4) * 10 + (minute & 0x0F);
hour = (hour >> 4) * 10 + (hour & 0x0F);
date = (date >> 4) * 10 + (date & 0x0F);
month = (month >> 4) * 10 + (month & 0x0F);
year = (year >> 4) * 10 + (year & 0x0F);
// End conversion

Time[12+i] = second % 10 + 48; //Converting to ASCII
Time[11+i] = second / 10 + 48;
Time[10+i] = ':';
Time[9+i] = minute % 10 + 48;
Time[8+i] = minute / 10 + 48;
Time[7+i] = ':';
Time[6+i] = hour % 10 + 48;
```

```
Time[5+i] = hour / 10 + 48;
Calendar[12+i] = year % 10 + 48;
Calendar[11+i] = year / 10 + 48;
Calendar[10+i] = '/';
Calendar[9+i] = month % 10 + 48;
Calendar[8+i] = month / 10 + 48;
Calendar[7+i] = '/';
Calendar[6+i] = date % 10 + 48;
Calendar[5+i] = date / 10 + 48;



lcd.setCursor(0,0);
lcd.print(Time);
lcd.setCursor(0,1);
lcd.print(Calendar);
}

int editbutton()
{
int alarm,hp,mp,dp,tp,yp;
alarm = 0;
hp = hour;
mp = minute;
dp = date;
tp = month;
```

```
yp = year;
int j,k=1;

char ins1[] = "Leave button";
char ins2[] = "Press button";


for(j=0;j<k;j++)
{
lcd.clear();

if(digitalRead(8)==0)
{
hour = 0;
while(true)
{
hour = hour + 1;
if(digitalRead(8)==1)
break;
if(hour > 23)
{
hour = 0;
lcd.clear();
}
lcd.setCursor(0,0);
```

```
lcd.print(ins1);
lcd.setCursor(0,1);
lcd.print("hour ");
lcd.print(hour);
delay(800);

}
}

lcd.clear();
if(digitalRead(8)==1)
{
minute = 0;
while(true)
{
minute = minute + 1;
if(digitalRead(8)==0)
break;
if(minute > 59)
{
minute = 0;
lcd.clear();
}
lcd.setCursor(0,0);
lcd.print(ins2);
```

```
lcd.setCursor(0,1);
lcd.print("minute ");
lcd.print(minute);
delay(800);
}
}

lcd.clear();
if(digitalRead(8)==0)
{
date = 0;
while(true)
{
date = date + 1;
if(digitalRead(8)==1)
break;
if(date > 30)
{
date = 0;
lcd.clear();
}
lcd.setCursor(0,0);
lcd.print(ins1);
lcd.setCursor(0,1);
lcd.print("date ");
```

```
lcd.print(date);
delay(800);
}
}

lcd.clear();
if(digitalRead(8)==1)
{
month = 0;
while(true)
{
month = month + 1;
if(digitalRead(8)==0)
break;
if(month > 12)
{
month = 0;
lcd.clear();
}
lcd.setCursor(0,0);
lcd.print(ins2);
lcd.setCursor(0,1);
lcd.print("month ");
lcd.print(month);
```

```
delay(800);
}
}

lcd.clear();
if(digitalRead(8)==0)
{
year = 0;
while(true)
{
year = year + 1;
if(digitalRead(8)==1)
break;
if(year > 50)
{
year = 0;
lcd.clear();
}

lcd.setCursor(0,0);
lcd.print(ins1);
lcd.setCursor(0,1);
lcd.print("year ");
lcd.print(year);
delay(800);
```

```
}
}

lcd.clear();

for (int tc=0; tc<5 ; tc++)
{

if(alarm == 0)
{
lcd.setCursor(0,0);
lcd.print("To Set Alarm 1");
lcd.setCursor(0,1);
lcd.print("Press Button ");
lcd.print(tc);
delay(2000);
}
if ((digitalRead(8) == 0 && alarm < 2) || k == 2)
{
lcd.clear();
h[alarm] = hour - 1;
m[alarm] = minute - 1;
d[alarm] = date - 1;
alarm = alarm + 1;
tc = 0;
```

```
lcd.setCursor(0,0);

lcd.print("Alarm ");
lcd.print(alarm);
lcd.print(" Set");
delay(5000);
}

lcd.clear();
if(alarm == 1)
{
lcd.setCursor(0,0);
lcd.print("To Set Alarm 2");
lcd.setCursor(0,1);
lcd.print("Hold Button Now");
delay(2000);
lcd.clear();
}

if(digitalRead(8) == 0 && k<=1)
{
lcd.setCursor(0,0);
lcd.print("Hold Button");
lcd.setCursor(0,1);
lcd.print("Set Alarm Time");
```

```
delay(1000);
k=k+1;
}

if(k==2)
{
break;
}

delay(2000);
//lcd.clear();
}
}

if(alarm != 0)
{
hour = hp;
minute = mp;
date = dp;
month = tp;
year = yp;
}

return alarm;
```

```
}

void loop()
{
while(digitalRead(8)==0)
{
al = editbutton();
Serial.println(al);

if(al==0)
{
hour = hour - 1;
minute = minute - 1;
date = date - 1;
month = month - 1;
year = year - 1;
}

// Convert decimal to BCD
minute = ((minute / 10) << 4) + (minute % 10);
hour = ((hour / 10) << 4) + (hour % 10);
date = ((date / 10) << 4) + (date % 10);
month = ((month / 10) << 4) + (month % 10);
year = ((year / 10) << 4) + (year % 10);
// End conversion
```

```
// Write data to DS3231 RTC

Wire.beginTransmission(0x68); // Start I2C protocol with DS3231 address

Wire.write(0); // Send register address

Wire.write(0); // Reset sesonds and start oscillator

Wire.write(minute); // Write minute

Wire.write(hour); // Write hour

Wire.write(day); // NOT USED

Wire.write(date); // Write date

Wire.write(month); // Write month

Wire.write(year); // Write year

Wire.endTransmission(); // Stop transmission and release the I2C bus

delay(300); // Wait 300ms

}


i=1;

y = y + 1;


if (al > 0)
/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////

{

for (int x =0;x<al;x++)

{

if(d[x] == date && h[x] == hour && m[x] == minute)

{

lcd.clear();
```

```
lcd.setCursor(0,0);
lcd.print("Alarm SET ON");
delay(500);
digitalWrite(9, HIGH);
delay(500);
digitalWrite(9, LOW);
}
else
{
digitalWrite(9,LOW);
}
}
}

if(Serial.available()>0)
{
bs = Serial.read();
if(bs == 'e')
{
Serial.println("Enter Hour");
while(x==0)
{
x = Serial.parseInt();
}
Serial.println(x);
```

```
pre = hour;
hour = x;
x=0;
}
if(hour > 23)
{
Serial.println("Invalid Entry For Hour");
hour = pre;
}
if(bs == 'e')
{
Serial.println("Enter Minute");
while(x==0)
{
x = Serial.parseInt();
}
Serial.println(x);
pre = minute;
minute = x;
x=0;
}
if(minute > 59)
{
Serial.println("Invalid Entry For Minute");
minute = pre;
```

```
}
if(bs == 'e')
{
Serial.println("Enter date");
while(x==0)
{
x = Serial.parseInt();
}
Serial.println(x);
pre = date;
date = x;
x=0;
}
if(date > 31)
{
Serial.println("Invalid Entry For date");
date = pre;
}
if(bs == 'e')
{
Serial.println("Enter Month");
while(x==0)
{
x = Serial.parseInt();
}
```

```
Serial.println(x);
pre = month;
month = x;
x=0;
}
if(month > 12)
{
Serial.println("Invalid Entry For Month");
month= pre;
}
if(bs == 'e')
{
Serial.println("Enter Year");
while(x==0)
{
x = Serial.parseInt();
}
Serial.println(x);
pre = year;
year = x;
x=0;
}
if(year > 25)
{
Serial.println("Invalid Entry For Year");
```

```
year = pre;
}



// Convert decimal to BCD
minute = ((minute / 10) << 4) + (minute % 10);
hour = ((hour / 10) << 4) + (hour % 10);
date = ((date / 10) << 4) + (date % 10);
month = ((month / 10) << 4) + (month % 10);
year = ((year / 10) << 4) + (year % 10);
// End conversion

if(bs=='v')
{
Serial.println(al);

Serial.println(h[0]);
Serial.println(hour);

Serial.println(m[0]);
Serial.println(minute);

Serial.println(d[0]);
Serial.println(date);
```

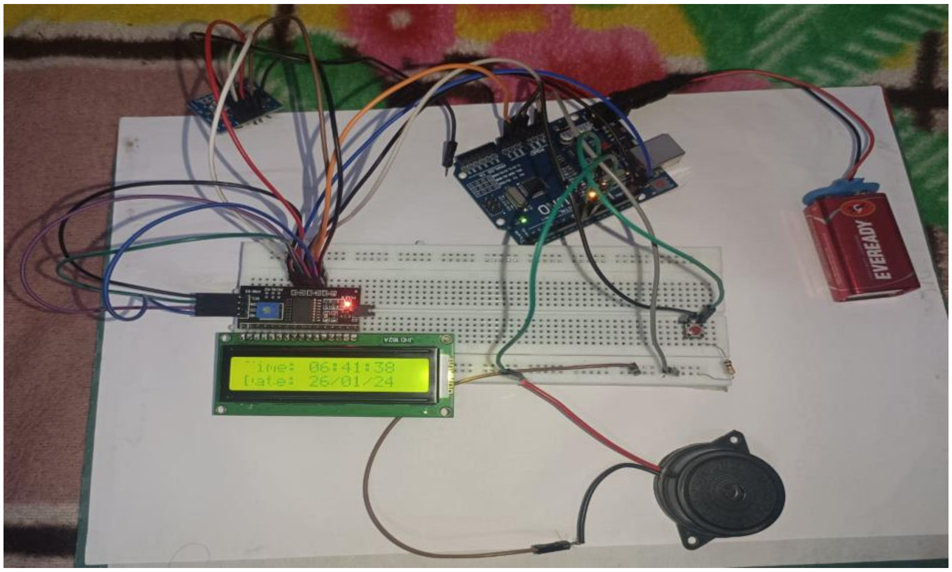
```
}

// Write data to DS3231 RTC
Wire.beginTransmission(0x68); // Start I2C protocol with DS3231 address
Wire.write(0); // Send register address
Wire.write(0); // Reset sesonds and start oscillator
Wire.write(minute); // Write minute
Wire.write(hour); // Write hour
Wire.write(day); // NOT USED
Wire.write(date); // Write date
Wire.write(month); // Write month
Wire.write(year); // Write year
Wire.endTransmission(); // Stop transmission and release the I2C bus
delay(300); // Wait 300ms
}

Wire.beginTransmission(0x68); // Start I2C protocol with DS3231 address
Wire.write(0); // Send register address
Wire.endTransmission(false); // I2C restart
Wire.requestFrom(0x68, 7); // Request 7 bytes from DS3231 and release I2C bus at end of
reading
second = Wire.read(); // Read seconds from register 0
minute = Wire.read(); // Read minuts from register 1
hour = Wire.read(); // Read hour from register 2
day = Wire.read(); //NOT USED
```

```
date = Wire.read(); // Read date from register 4
month = Wire.read(); // Read month from register 5
year = Wire.read(); // Read year from register 6


DS3231_display(); // Display time & calendar

if(y==5)
{
Serial.println("To edit date and time enter e");
y=0;
}

Serial.println(ti);
Serial.println(Time);
Serial.println(ca);
Serial.println(Calendar);
Serial.println("\n");

delay(2000); // Wait 50ms
```

APPLICATIONS:

- It is used for cars
- It is applicable for radios
- It is used for televisions
- It is applicable for microwave ovens
- It is used for standard ovens
- It is applicable for computers and cell phones

RESULT:



Hence we have successfully completed and executed the project of "Digital alarm clock" using Adenine Uno.

BUDGET PAGE :

S.NO	COMPONENTS	MONEY
1	Arduino-Uno	400
2	16*2 lcd display	350
3	12c module	60
4	mini bread board	100
5	DS1307 real time clock module	140
6	buzzer	20
7	jumper wires	80
8	9v battery	40
9	resistor	10
10	push button	5
	TOTAL	=1205

CONCLUSION

This system provides a fast and cost-effective solution to avert the gas leak effect by reducing the risk to human life. The statistics of the application of gas clam on to the application can be useful to own the faulty valves and regulators prior and do the necessary replacement. Apart from detecting the leakage, a two-level prevention apparatus makes the system more valid.