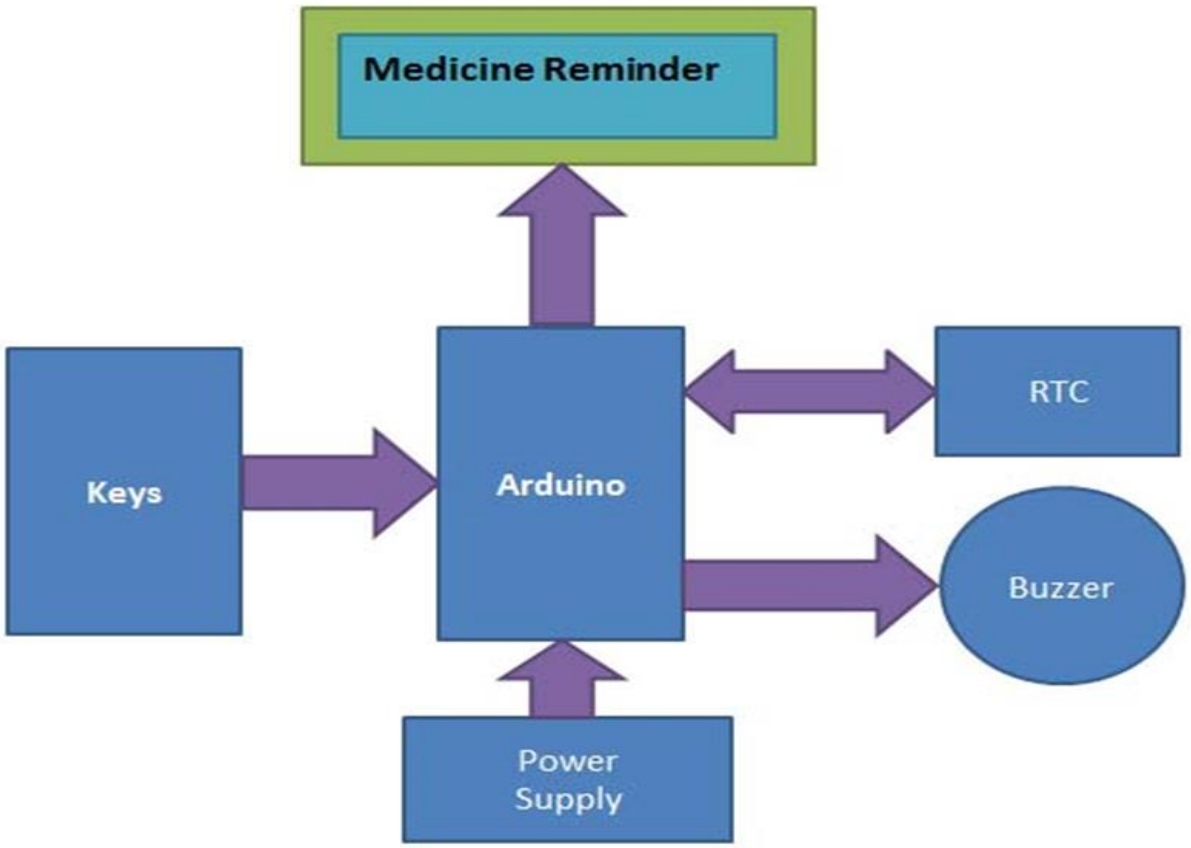
CHAPTER-1
INTRODUCTION

A lot of people have to take certain medicines regularly. A number of diseases like diabetes, blood pressure or heart problem are nowadays very common and the patients need to take medicines without break to keep such health issues in check. Most of the people are not punctual about taking medicines and often forget one or the other dosage. In this project, a medicine reminder system is designed in which the user can feed the schedule of their medicines. The user has to feed the type of medicine.

The morning time is assumed to be 8 AM, Afternoon time is assumed to be 2 PM and Night time is assumed to be 8PM.

The project is built on Arduino Mega and has an RTC interfaced to keep track of time without fail. The medicine information is saved in the internal EEPROM of the Arduino board. A four-switch keypad is provided to feed medicine information and an LCD display is interfaced to the system for providing the user interface. The Arduino sketch manages to input and store medicine information, keep track of time using RTC, compare dosage time with real time. The Arduino code is written on Arduino IDE and burnt to the board.



When we start this system real time clock runs the time on 16×2 LCD. And if we want to set alarm time for medication we have to press set buttons which is connected with pin number 8 of Arduino. After pressing this button LCD shows time. And then we can select the time as we want to set for medication by using INC and Next button which is connected to pin 9 and 10 respectively of Arduino.

Medication alarm time is also feed in Arduino's internal to save from lose data after light failure. And real time is continuously checked with saved Arduino's internal EEPROM time. If any match occurs. LCD shows medication group name and buzzer starts beeping continuously. Buzzer is directly connected with pin number 13 of Arduino for medication time indication.

6×2 LCD's data pin D4, D3, D2, D2 are connected with pin5, 4, 3, 2 of Arduino. And command pin RS and EN is directly connected with pin 7, 6 of Arduino. RW pin of LCD is directly connected with ground.

1.1 OBJECTIVES

The aim of this project is to process of overseeing the medications prescribed for a patient to ensure they are taken properly and achieving their planned.

1. To select appropriate mechanism and materials used to design MEDICINE REMINDED.
2. This framework is driven by an LCD display board, Buzzer to remind about taking pill, using ARDUINO UNO and DS3231 which is a real time clock (RTC).
3. Every time the person needed to take the pill, buzzer buzz with sound as preprogrammed earlier.
4. This paper proposes persistent prescription update is a framework which helps in medicine organization and checking.

1.2 SCOPE OF THE PROJECT

The present work is such an attempt to prevent medication management using Arduino uno and RTC DS3232 module. As this helps the patients to take the medicine at the prescribed time suggested by the doctor. The led helps to display the tablet which as to be taken and the buzzer helps to remind the patient at what time the tablet has to be taken. The medication-taking occurrence was improved using this system which is also known as “Medicine reminder”.

CHAPTER-2
LITERATURE SURVEY

1. Jennifer B Zogg, Woods S, Sauceda Ja, Wiebe J, Simoni J: Wiebe& Jane. Journal of Behavioral Medicine ISSN 0160-7715 volume 28 number 1. February 2005.

The role of prospective memory in medication adherence. Most patients cannot remember their entire medication regimen and occasionally forget to take their medication. The objective of the study was to design, develop, and demonstrate the feasibility of a new type of medication self-management system using smartphones with real-time medication monitoring. To designed and developed a alarm-based medication self-management system (SMSS) based on interviews of 116 patients.

2. Mr. Mohammad Mohatram, Mr. Waleed Humaid: Automatic medicine reminder using Arduino at International Research of Engineering and Technology (IRJET) Volume 6 Issue:11. November 2019.

The system offered patients: storage and provision of an accurate, portable medication history and medication-taking records of patients; and provision of a reminder to take medication only when the patient has forgotten to take his/her medication.

3. Terry Eagleton , Walker for, barber n: An investigation into patient compliance with hospital discharge medication in a local population. int j pharm pract 1993; 2: 107- 109

Reading the prescription data of patients and using the medicine reminder for getting the Real-time medication monitoring by novel user-friendly by alarm sound. We believe that the proposed system is feasible and provides an innovative solution to encourage medication self-management

CHAPTER-3

COMPONENT DESCRIPTION

Components Required for Automatic Medicine reminder using Arduino

Arduino Uno (We can use other Arduino boards also, like Pro mini, Nano)

RTC DS3231 module

16x2 LCD Display

Buzzer

Breadboard

Push Buttons

10K Potentiometer

Resistors

Jumper Wires

3.1. Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2010. The microcontroller board is equipped with sets of digital and Analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 Analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery. It has the same microcontroller as the Arduino Nano board, and the same headers as the Leonardo board. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

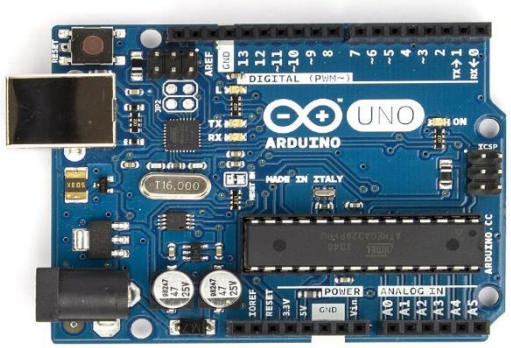


Fig -3.1: Arduino uno

3.2. RTC DS3231 Module -1

RTC means Real Time Clock. RTC modules are simply TIME and DATE remembering systems which have battery setup which in the absence of external power keeps the module running. This keeps the TIME and DATE up to date. So we can have accurate TIME and DATE from RTC module whenever we want.

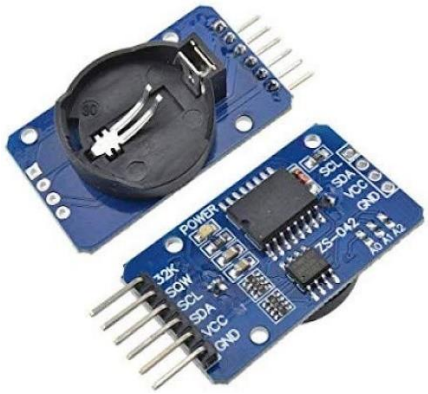


Fig -3.2: DS3231 RTC Module

3.3.LCD (Liquid Crystal Display)

In LCD 16×2, the term LCD stands for Liquid Crystal Display that uses a plane panel display technology, used in screens of computer monitors & TVs, smartphones, tablets, mobile devices, etc. Both the displays like LCD & CRTs look the same but their operation is different. Instead of electrons diffraction at a glass display, a liquid crystal display has a backlight that provides light to each pixel that is arranged in a rectangular network.

Fig -3.3: LCD

3.4. Push-Button

The "push-button" has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial. In industrial and commercial applications, push buttons can be connected together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process has no electrical circuits for control.

Fig -3.4: Push button

3.5. Potentiometer

Potentiometers also known as POT, are nothing but variable resistors. They can provide a variable resistance by simply varying the knob on top of its head. It can be classified based on two main parameters. One is their Resistance (R-ohms) itself and the other is its Power (P-Watts) rating.

Fig – 3.5: Potentiometer

3.6. Resistor

A resistor is a passive two-terminal electrical component that implements electrical resistance as a circuit element. In electronic circuits, resistors are used to reduce current flow, adjust signal levels, to divide voltages, bias active elements, and terminate transmission lines, among other uses. High-power resistors that can dissipate many watts of electrical power as heat may be used as part of motor controls, in power distribution systems, or as test loads for generators. Fixed resistors have resistances that only change slightly with temperature, time or operating voltage. Variable resistors can be used to adjust circuit elements (such as a volume control or a lamp dimmer), or as sensing devices for heat, light, humidity, force, or chemical activity.

Fig – 3.6: Resistor

3.7 Buzzer

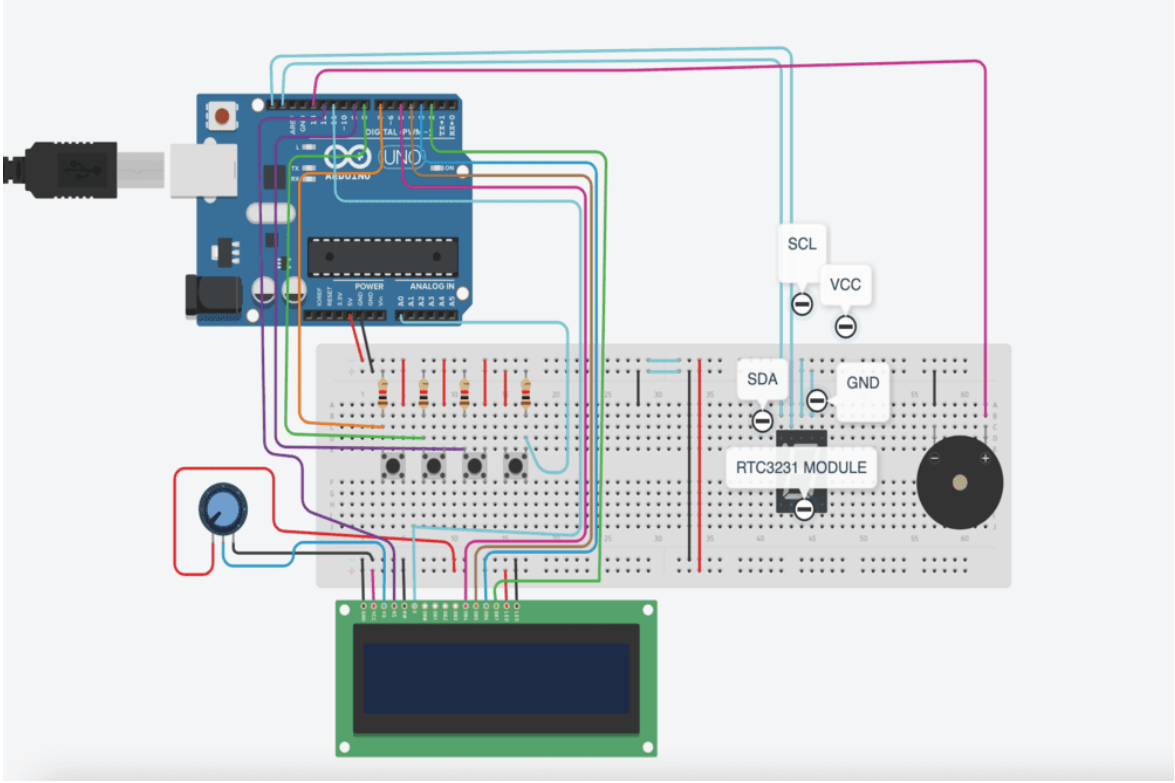
A buzzer or beeper is an audio signaling device, be mechanical, electromechanical, or piezoelectric (*piezo* for short). Typical uses of buzzers and beepers include alarm devices, timers, train and confirmation of user input such as a mouse click or keystroke

Fig – 3.7 Buzzer

CHAPTER-4

4.1 Medicine Reminder Circuit Diagram

The complete circuit diagram to build a **medicine reminder using Arduino** is shown below



Below are the **pin connections** of Arduino with different peripherals

Arduino Pins	Peripheral Pins
2	----------> D7 of 16x2 LCD Display
3	----------> D6 of 16x2 LCD Display
4	----------> D5 of 16x2 LCD Display
5	----------> D4 of 16x2 LCD Display
7	----------> 3rd push button
8	----------> 2nd push button
9	----------> 1st push button
11	----------> EN pin of 16x2 LCD Display
12	----------> RS pin of 16x2 LCD Display
13	----------> +Ve Pin of Buzzer and Led
A0	----------> Stop Push Button
A4	----------> SDA of DS3231
A5	----------> SCL of DS3231
3.3V	----------> Vcc of DS3231
Gnd	----------> Gnd

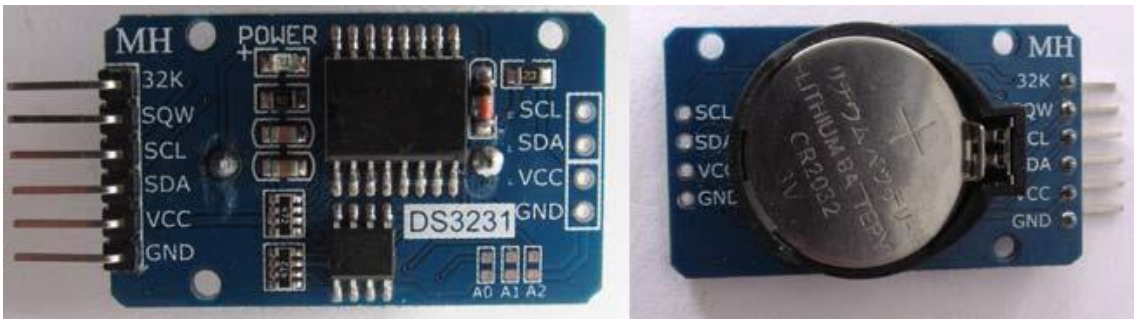


Fig -4.1: DS3231 RTC Module

In this **Medicine Reminder Project,** RTC DS3231 is interfaced through I2C protocol with Arduino Uno. You can also use RTC IC DS1307 for reading the time with Arduino. RTC DS3231 also has inbuilt 32k memory which can be used to store additional data. RTC module is powered through the 3.3V pin of Arduino uno. A 16x2 LCD display is interfaced using SPI. A **buzze**r is used to alert and remind that it's time to take medicine. **Four push buttons** are used where each has distinct select feature. The first push button is used for reminding to take medicine once per day. The second push button is used to remind twice per day and the third push button is used to remind thrice per day. The fourth push button is used to stop the buzzer when user has heard the alert.

4.2 Working of Medicine Reminder

The **Medicine Reminder** is powered using 5V supply. When it first boots up, it shows a welcome massage as "*Welcome to medicine reminder*". The LCD screen is set to cycle in three screens. The 1st screen shows massage as "*Stay Healthy, Get Well Soon*". The second screen is a help screen which tells to press select push button to select any one time-slot to remind (once/twice/thrice in a day). The time slot is changeable in program and can be configured accordingly. Right now we have fixed this into three durations i.e. 8am, 2pm, and 8pm.

We have divided time slots into three modes. Mode 1 selects to take medicine once/day at 8am when user presses 1st push button. Mode 2 selects to take medicine twice/day at 8am and 8pm when user presses 2nd push button. Mode 3 selects to take medicine thrice/day at 8am.

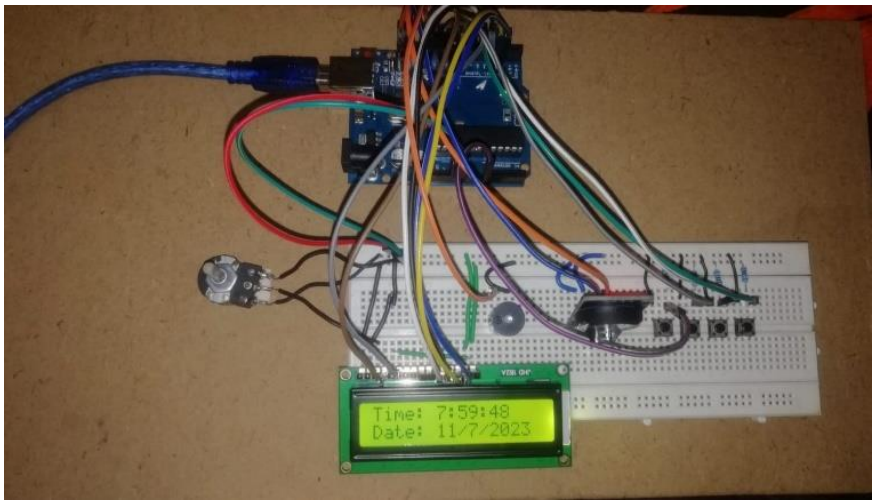


Fig4.2 Working of Medicine reminder

In this system we have used Arduino for controlling the whole system. Working of this project is very simple. In this system ds1307 real time clock chip is used for running the time accurate and to prevent the time after light failure by using 3volt li-on battery connected with this real time clock chip at pin number 3. SDA and SCK pin of real time clock chip is directly connected with SDA and SCK pin of Arduino (A5 and A4) respectively. These two pins should be pull-up using 10K resistor.

When we start this system real time clock runs the time on 16×2 LCD. And if we want to set alarm time for medicine we have to press set mad buttons which is connected with pin number 8 of Arduino. After pressing this button LCD shows Set Time

1**.** And then we can select the time as we want to set for medication by using INC and Next button which is connected to pin 9 and 10 respectively of Arduino. After set time 1, LCD shows set time

2. Now using previous process set the time again. And after second time set, LCD shows again set time

3. And set this time like previous. In this system "Group medicine" indication (take group 1 medicine, take group 2 medicine and take group 3 medicine) is used instead of medicine name.

Medication alarm time is also feed in Arduino's internal to save from lose data after light failure. And real time is continuously checked with saved Arduino's internal time. If any match occurs. LCD shows medication group name and buzzer starts beeping continuously. Buzzer is directly connected with pin number 13 of Arduino for medication time indication.

16×2 LCD's data pin D4, D3, D2, D2 are connected with pin5, 4, 3, 2 of Arduino. And command pin RS and EN is directly connected with pin 7, 6 of Arduino. RW pin of LCD is directly connected with ground.

CHAPTER-5

PROGRAM

5.1 Programming Requirement

In this program some libraries are used given below:

1.**Wire.h:** for I2C interfacing

2.**RTClib.h:** for RTC interfacing

3.**Liquid Crystal**: for 16X2 LCD interfacing

4.**EEPROM.h:** for access the internal Arduino's EEPROM for saving the alarm time.

PROGRAM

```
/Medicine Reminder using Arduino Uno
// Reminds to take medicine at 8am, 2pm, 8pm
/*  The circuit:
   LCD RS pin to digital pin 12
   LCD Enable pin to digital pin 11
   LCD D4 pin to digital pin 5
   LCD D5 pin to digital pin 4
   LCD D6 pin to digital pin 3
   LCD D7 pin to digital pin 2
   LCD R/W pin to ground
   LCD VSS pin to ground
   LCD VCC pin to 5V
   10K resistor:
   ends to +5V and ground
   wiper to LCD VO pin (pin 3)*/
#include <LiquidCrystal.h>
#include <Wire.h>
#include <RTClib.h>
#include <EEPROM.h>
int pushVal = 0;
int val;
int val2;
int addr = 0;
RTC_DS3231 rtc;
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;          // lcd pins
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
#define getWellsoon 0
#define HELP_SCREEN 1
#define TIME_SCREEN 2
//bool pushPressed;                     //flag to keep track of push button state
int pushpressed = 0;
const int ledPin =  13;                       // buzzer and led pin
```

```
int ledState = LOW;
nt Signal = 0;
int buzz = 13;
int push1state, push2state, push3state, stopinState = 0;
int push1Flag, push2Flag, Push3Flag = false;              // push button flags
int push1pin = 9;
int push2pin = 8;
int push3pin = 7;
int stopPin = A0;
int screens = 0;              // screen to show
int maxScreen = 2;            // screen count
bool isScreenChanged = true;
long previousMillis = 0;
long interval = 500;                    // buzzing interval
unsigned long currentMillis;
long previousMillisLCD = 0;    // for LCD screen update
long intervalLCD = 2000;          // Screen cycling interval
unsigned long currentMillisLCD;
//   Set Reminder Change Time
int buzz8amHH = 8;          //    HH - hours         ##Set these for reminder time in 24hr
Format
int buzz8amMM = 00;          //    MM - Minute
int buzz8amSS = 00;          //    SS - Seconds
int buzz2pmHH = 14;          //    HH - hours
int buzz2pmMM = 00;          //    MM - Minute
int buzz2pmSS = 00;          //    SS - Seconds
int buzz8pmHH = 20;          //    HH - hours
int buzz8pmMM = 00;          //    MM - Minute
int buzz8pmSS = 00;          //    SS - Seconds
int nowHr, nowMin, nowSec;                     // to show current mm,hh,ss
// All messeges
void gwsMessege(){               // print get well soon messege
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Stay Healthy :)");     // Give some cheers
    lcd.setCursor(0, 1);
    lcd.print("Get Well Soon :)");    // wish
}
void helpScreen() {              // function to display 1st screen in LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Press Buttons");
    lcd.setCursor(0, 1);
    lcd.print("for Reminder...!");
 }
void timeScreen() {              // function to display Date and time in LCD screen
  DateTime now = rtc.now();             // take rtc time and print in display
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Time:");
    lcd.setCursor(6, 0);
```

```
    lcd.print(nowHr = now.hour(), DEC);
    lcd.print(":");
    lcd.print(nowMin = now.minute(), DEC);
    lcd.print(":");
    lcd.print(nowSec = now.second(), DEC);
    lcd.setCursor(0, 1);
    lcd.print("Date: ");
    lcd.print(now.day(), DEC);
    lcd.print("/");
    lcd.print(now.month(), DEC);
    lcd.print("/");
    lcd.print(now.year(), DEC);
}
void setup() {
  Serial.begin(9600);                       // start serial debugging
  if (! rtc.begin()) {                       // check if rtc is connected
    Serial.println("Couldn't find RTC");
    while (1);
  }
  if (rtc.lostPower()) {
    Serial.println("RTC lost power, lets set the time!");
  }
//    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  rtc.adjust(DateTime(2023, 6, 12, 03, 50, 30));                // manual time set
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Welcome");                                        // print a messege at startup
  pinMode(push1pin, INPUT_PULLUP);                                      // define push button pins
type
  pinMode(push2pin, INPUT_PULLUP);
  pinMode(push3pin, INPUT_PULLUP);
  pinMode(stopPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
  delay(1000);
  Serial.println(EEPROM.read(addr));
  val2 = EEPROM.read(addr);                           // read previosuly saved value of push
button to start from where it was left previously
  switch (val2) {
    case 1:
      Serial.println("Set for 1/day");
      push1state = 1;
      push2state = 1;
      push3state = 1;
      pushVal = 1;
      break;
    case 2:
      Serial.println("Set for 2/day");
      push1state = 1;
      push2state = 1;
      push3state = 1;
```

```
      pushVal = 2;
      break;
    case 3:
      Serial.println("Set for 3/day");
      push1state = 1;
      push2state = 1;
      push3state = 1;
      pushVal = 3;
      break;
  }
}
void loop() {
  push1();                                          //call to set once/day
  push2();                                          //call to set twice/day
  push3();                                          //call to set thrice/day
    if (pushVal == 1) {                             // if push button 1 pressed then remind
at 8am
    at8am();                                        //function to start uzzing at 8am
  }
  else if (pushVal == 2) {                          // if push button 2 pressed then remind
at 8am and 8pm
    at8am();
    at8pm();                                        //function to start uzzing at 8mm
  }
  else if (pushVal == 3) {                          // if push button 3 pressed then remind
at 8am and 8pm
    at8am();
    at2pm();                                         //function to start uzzing at 8mm
    at8pm();
  }
  currentMillisLCD = millis();                      // start millis for LCD screen switching
at defined interval of time
  push1state = digitalRead(push1pin);               // start reading all push button pins
  push2state = digitalRead(push2pin);
  push3state = digitalRead(push3pin);
  stopinState = digitalRead(stopPin);
  stopPins();                                         // call to stop buzzing
  changeScreen();                                     // screen cycle function
}

// push buttons
void push1() {                      //function to set reminder twice/day
  if (push1state == 0) {
    push2state = 1;
    push1state = 1;
    push3state = 1;
//    pushPressed = true;
    EEPROM.write(addr, 1);
    Serial.print("Push1 Written : "); Serial.println(EEPROM.read(addr));
    pushVal = 1;
    lcd.clear();
```

```
    lcd.setCursor(0, 0);
    lcd.print("Reminder set ");
    lcd.setCursor(0, 1);
    lcd.print("for Once/day !");
    delay(1200);
    lcd.clear();
  }
}
void push2() {                      //function to set reminder twice/day
  if (push2state == 0) {
    push2state = 1;
    push1state = 1;
    push3state = 1;
//    pushPressed = true;
    EEPROM.write(addr, 2);
    Serial.print("Push2 Written : "); Serial.println(EEPROM.read(addr));
    pushVal = 2;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Reminder set ");
    lcd.setCursor(0, 1);
    lcd.print("for Twice/day !");
    delay(1200);
    lcd.clear();
  }
}
void push3() {                    //function to set reminder thrice/day
  if (push3state == 0) {
    push3state = 1;
    push1state = 1;
    push2state = 1;
//    pushPressed = true;
    EEPROM.write(addr, 3);
    Serial.print("Push3 Written : "); Serial.println(EEPROM.read(addr));
    pushVal = 3;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Reminder set ");
    lcd.setCursor(0, 1);
    lcd.print("for Thrice/day !");
    delay(1200);
    lcd.clear();
  }
}
void stopPins() {                   //function to stop buzzing when user pushes stop push button
  if (stopinState == 0) {
//    stopinState = 0;
//    pushPressed = true;
    pushpressed = 1;
    lcd.clear();
    lcd.setCursor(0, 0);
```

```
    lcd.print("Take Medicine  ");
    lcd.setCursor(0, 1);
    lcd.print("with Warm Water");
    delay(1200);
    lcd.clear();
  }
}
void startBuzz() {                    // function to start buzzing when time reaches to defined
interval
//  if (pushPressed == false) {
 if (pushpressed == 0) {
    Serial.println("pushpressed is false in blink");
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;         // save the last time you blinked the LED
      Serial.println("Start Buzzing");
      if (ledState == LOW) {                  // if the LED is off turn it on and vice-versa:
        ledState = HIGH;
      }  else {
        ledState = LOW;
      }
      digitalWrite(ledPin, ledState);
   }
  }
  else if (pushpressed == 1) {
    Serial.println("pushpressed is true");
    ledState = LOW;
    digitalWrite(ledPin, ledState);
  }
}
void at8am() {                        // function to start buzzing at 8am
  DateTime now = rtc.now();
  if (int(now.hour()) >= buzz8amHH) {
    if (int(now.minute()) >= buzz8amMM) {
      if (int(now.second()) > buzz8amSS) {
        ////////////////////////////////////////////////////
        startBuzz();
        ////////////////////////////////////////////////////
      }
    }
  }
}
void at2pm() {                          // function to start buzzing at 2pm
  DateTime now = rtc.now();
  if (int(now.hour()) >= buzz2pmHH) {
   if (int(now.minute()) >= buzz2pmMM) {
      if (int(now.second()) > buzz2pmSS) {
        ////////////////////////////////////////////////////
        startBuzz();
        ////////////////////////////////////////////////////
      }
```

```
    }
  }
}
void at8pm() {                          // function to start buzzing at 8pm
 DateTime now = rtc.now();
  if (int(now.hour()) >= buzz8pmHH) {
    if (int(now.minute()) >= buzz8pmMM) {
      if (int(now.second()) > buzz8pmSS) {
        ////////////////////////////////////////////////
        startBuzz();
        ////////////////////////////////////////////////
      }
    }
  }
}
//Screen Cycling
void changeScreen() {                  //function for Screen Cycling
  // Start switching screen every defined intervalLCD
  if (currentMillisLCD - previousMillisLCD > intervalLCD)             // save the last time you
changed the display
  {
    previousMillisLCD = currentMillisLCD;
    screens++;

    if (screens > maxScreen) {
      screens = 0;  // all screens over -> start from 1st
    }
    isScreenChanged = true;
  }
  // Start displaying current screen
  if (isScreenChanged)   // only update the screen if the screen is changed.
  {
    isScreenChanged = false; // reset for next iteration
    switch (screens)
    {
      case getWellsoon:
        gwsMessege();                // get well soon message
        break;
      case HELP_SCREEN:
        helpScreen();               // instruction screen
        break;
      case TIME_SCREEN:
        timeScreen();                  // to print date and time
        break;
      default:
        //NOT SET.
        break;
    }
  }
}
```

CHAPTER-6

RESULT AND DISCUSSION

a) The DS3231 RTC module is particularly useful in medication management systems where precise timing is used model will significantly improve safety of line man crucial. By integrating the DS3231, we can schedule medication reminders, track dosage times, and ensure accurate record-keeping. The module's alarm functionality can be utilized to trigger alerts when it's time for medication administration.

b) Although the DS3231 is highly accurate, occasional calibration may be required to ensure long term precision. Some libraries or code examples provide functionality for calibration adjustments if necessary. The below figure depicts the smart medicine reminder what time the medicine has to be taken.

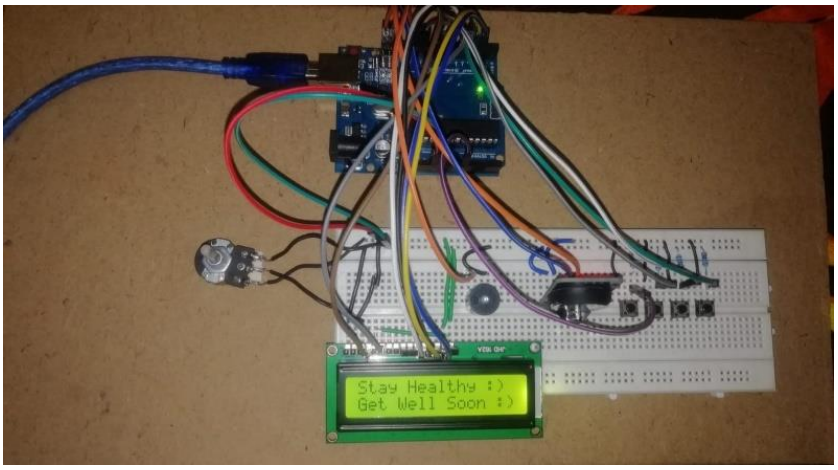


Fig6.1 display of message

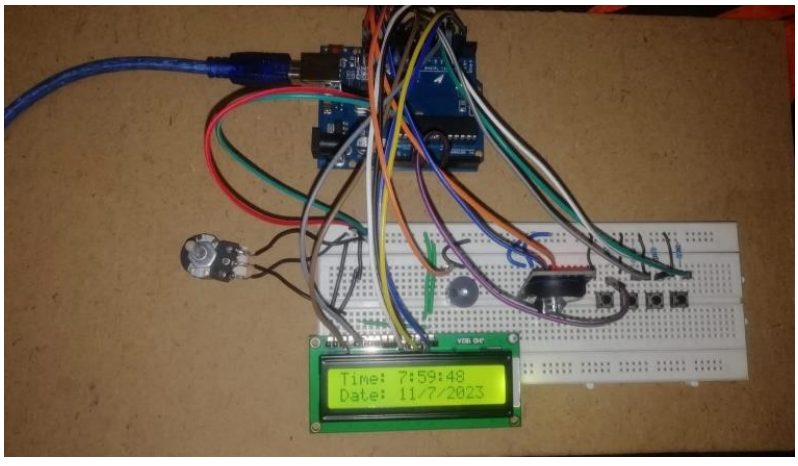


Fig6.2 display of time and date

CHAPTER-7
CONCLUSION

Numerous Medication Reminder Systems have been created on various stages. A considerable lot of these frameworks require unique equipment gadgets to remind the patients about the drug in-take timing. Acquiring new equipment gadgets turns out to be expensive and additional time and cash devouring so in the given work an endeavor has been made to actualize a framework which will be prudent, effectively available and improves drug adherence. Persistent Medication update framework will decrease the viability of a treatment and forces money related trouble on medicinal services frameworks. The patients will get the calendar of medication in-require some investment with medication portrayal, beginning and completion date of medication, warning through fluid precious stone presentation (LCD), programmed alert ringing framework. The booked update will propose the sort of medication the patient will take at the specific time of caution.

7.1 FUTURE SCOPE

During the system testing it was observed that can use for three or four medication give that the user for one patient. For instance, when the use microcontroller became limit because capacity of microcontroller is for three or four medication such cases could be avoided by memory insert and change code od microcontroller. This would involve creating a model of the code and memory which contains the possible to get from one Patient to multi patient

REFERENCES

1. Mr. Mohammad Mohatram, Mr. Waleed Humaid. Automatic medicine reminder using Arduino at International Research of Engineering and Technology (IRJET) Volume 6 Issue:11. November 2019.

2. Simpson SH, Eurich DT, Majumdar SR, Padwal RS, Tsuyuki RT, Varney J, et al. A meta-analysis of the association between adherence to drug therapy and mortality. BMJ 2006; 333(7557): 15-21 [PMID:16790458] [PMC free article] [PubMed] [Google Scholar].

3. Ni Ni San Hlaing, San San Naing. Alarm System for medicine reminder at International Journal of Trend Scientific Research and Development (IJTSRD). Volume 3 Issue 5, August 2019.

4. Jennifer B. Zogg, Woods SP, Steven Paul Woods, John A. Sauceda. John S. Simoni J. Wiebe& Jane. Journal of Behavioral Medicine ISSN 0160-7715 volume 28 number 1. February 2005.

5.www.researchgate.net/publication/301234345. Medicine Reminder and Monitoring System for Secure Health

6. www.ncbi.nlm.nih.gov/books/NBK260118/

APPENDIX

1. Arduino board (such as Arduino Uno or Arduino Nano): This serves as the main microcontroller for your project.
2. Real-Time Clock (RTC) module: This module allows your Arduino to keep track of time, even when it's powered off. It ensures accurate timing for medication reminders.
3. LCD Display: An LCD (Liquid Crystal Display) is used to provide visual output for the user. It can display information like the current time, medication reminders, and any additional messages.
4. Buzzer: A small piezo buzzer can be used to produce audible alerts or reminders when medication needs to be taken.
5. Push buttons: These are used as input devices to set the timing and interact with the system. They can be used for tasks such as setting the reminder time or acknowledging the medication intake.
6. LED indicators: LEDs can be used to indicate the status of the system, such as power on/off or medication reminder active.
7. Power supply: Depending on the specific Arduino board and components used, you might need a power supply to provide the required voltage and current. This can be a battery pack or a regulated power supply.

Once you have these components, you can proceed with the following steps to create the medicine reminder:

1. Connect the RTC module to the Arduino board using the appropriate pins (usually I2C communication).
2. Connect the LCD display to the Arduino, ensuring the correct wiring for data and control signals.
3. Connect the buzzer to an available digital pin on the Arduino.
4. Connect the push buttons to the Arduino, assigning them to specific input pins.
5. Connect the LED indicators to the Arduino, choosing appropriate pins to control them.
6. Write the Arduino code to control the RTC module, read the current time, and compare it with the programmed medication schedule.
7. If it's time for medication, activate the buzzer to provide an audible alert and display the reminder on the LCD.
8. Implement button functionalities to set the medication schedule and acknowledge when the medication has been taken.
9. Test the system thoroughly, making sure the reminders and alerts function as expected.
10. Enclose the Arduino and circuitry in a suitable enclosure to protect it and make it user-friendly.

Remember to pay attention to safety precautions and consider using a reliable power source for uninterrupted operation.