Visvesvaraya Technological University

Belgaum, Karnataka-590 018

*A Project Report on*

# "SEQUENTIAL POWER DISTRIBUTION"

*Project Report submitted in partial fulfillment of the requirement for the award of the degree of*

**Bachelor of Engineering**
**In**
**Electrical & Electronics Engineering**

*Submitted by*

**RAKSHITHA AC 1CR17EE055**
**SANDHYA RANI R 1CR17EE063**

*Under the Guidance of*
**Mrs. KEKA MUKHOPADHYAYA**
**Assistant Professor, Department of Electrical & Electronics Engineering**
**CMR Institute of Technology**

**CMR INSTITUTE OF TECHNOLOGY, BENGALURU, 560037**

**Department of Electrical and Electronics Engineering**

**2020-2021**

**CMR INSTITUTE OF TECHNOLOGY**
**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**
**AECS Layout, Bengaluru-560 037**

# *Certificate*

Certified that the project work entitled **“Sequential Power Distribution”** carried out by Ms. Rakshitha AC, USN 1CR17EE055 and Ms. Sandhya Rani R, USN 1CR17EE063; are bonafied students of CMR Institute of Technology, Bengaluru, in partial fulfillment for the award of Bachelor of Engineering in Electrical & Electronics Engineering of the Visvesvaraya Technological University, Belgaum, during the year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

| | | |
|---|---|---|
| *Signature of the Guide* | *Signature of the HOD* | *Signature of the Principal* |
| ----------------------------- | -------------------------------- | ----------------------------- |
| Mrs. Keka Mukhopadhyaya, Assistant Professor EEE Department | Dr. K. Chitra Professor & HOD EEE Department CMRIT, Bengaluru | Dr. Sanjay Jain Principal, CMRIT, Bengaluru |

*External Viva*

Name of the Examiners                    Signature & Date

1.

2.

**CMR INSTITUTE OF TECHNOLOY**
**DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING**
**AECS Layout, Bengaluru-560 037**

# DECLARATION

Ms **Rakshitha AC (1CR17EE055),** Ms **Sandhya Rani R (1CR17EE063),** hereby declare that the report entitled **"Sequential Power Distribution"** has been carried out by us under the guidance of **Mrs. Keka Mukhopadhyaya,** Assistant Professor, Department of Electrical & Electronics Engineering, CMR Institute of Technology, Bengaluru, in partial fulfillment of the requirement for the degree of **BACHELOR OF ENGINEERING in ELECTRICAL & ELECTRONICS ENGINEERING**, of Visveswaraya Technological University, Belagaum during the academic year 2020-21. The work done in this report is original and it has not been submitted for any other degree in any university.

Place: Bengaluru

Date:

Rakshitha AC (1CR17EE055)
Sandhya Rani R(1CR17EE063)

# Abstract

The main aim of this project is to develop the automation system for residential electricity cut off using network based embedded controller.

The purpose of this project is to design an automatic sequential power supply to different areas in electricity department to give the power supply to the particular area in particular time by electricity board. This also helps in effective utilization of the available power supply.

The automation of the distribution feeders in the distribution substation is the latest technology that is developed using the Embedded system technology and microcontrollers. The concept of integrated systems approach is to protect, control and monitoring of given functions.

# Acknowledgement

*The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people, who are responsible for the completion of the project and who made it possible, because success is outcome of hard work and perseverance, but stead-fast of all is encouraging guidance. So with gratitude we acknowledge all those whose guidance and encouragement served us to motivate towards the success of the project work.*

*We take great pleasure in expressing our sincere thanks to* ***Dr. Sanjay Jain**, **Principal, CMR Institute of Technology, Bengaluru*** *for providing an excellent academic environment in the college and for his continuous motivation towards a dynamic career. We would like to profoundly thank* ***Dr. B Narasimha Murthy****, Vice-principal of CMR Institute of Technology and the whole* ***Management*** *for providing such a healthy environment for the successful completion of the project work.*

*We would like to convey our sincere gratitude to* ***Dr. K Chitra**, **Head of Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for her invaluable guidance and encouragement and for providing good facilities to carry out this project work.*

*We would like to express our deep sense of gratitude to* ***Mrs. Keka Mukhopadhyaya, Assistant Professor, Electrical and Electronics Engineering, CMR Institute of Technology, Bengaluru*** *for her exemplary guidance, valuable suggestions, expert advice and encouragement to pursue this project work.*

*We're thankful to all the faculties and laboratory staffs of* ***Electrical and Electronics Engineering Department, CMR Institute of Technology, Bengaluru*** *for helping us in all possible manners during the entire period.*

*Finally, we acknowledge the people who mean a lot to us, our parents, for their inspiration, unconditional love, support, and faith for carrying out this work to the finishing line. We want to give special thanks to all my friends who went through hard times together, cheered us on, helped us a lot, and celebrated each accomplishment.*

*Lastly, to the* ***Almighty****, for showering His Blessings and to many more, whom we didn't mention here.*

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

## 1 INTRODUCTION

Now a day's every system is automated in order to face new challenges. In the present days Automated systems have lesser manual operations, flexibility, reliability and accurate. Due to this demand every field prefers automated control systems. Especially in the field of electronics automated systems are giving good performance.

The project itself indicates that a fixed time is set for the particular areas. That means at that particular time duration the operation will be in either ‘ON’ or ‘OFF’ condition. That time delay will be different for every area. The ON/OFF condition will also be decided in the program. And that delay will be setting through the program. Here we can program the delays for the areas as per our requirement.

In this project we are using Real time clock (RTC) for setting the time delay. The keypad is used for mentioning the delay time. The time and status of that particular device will be displayed on the LCD.

# CHAPTER 2

## 2 BLOCK DIAGRAM

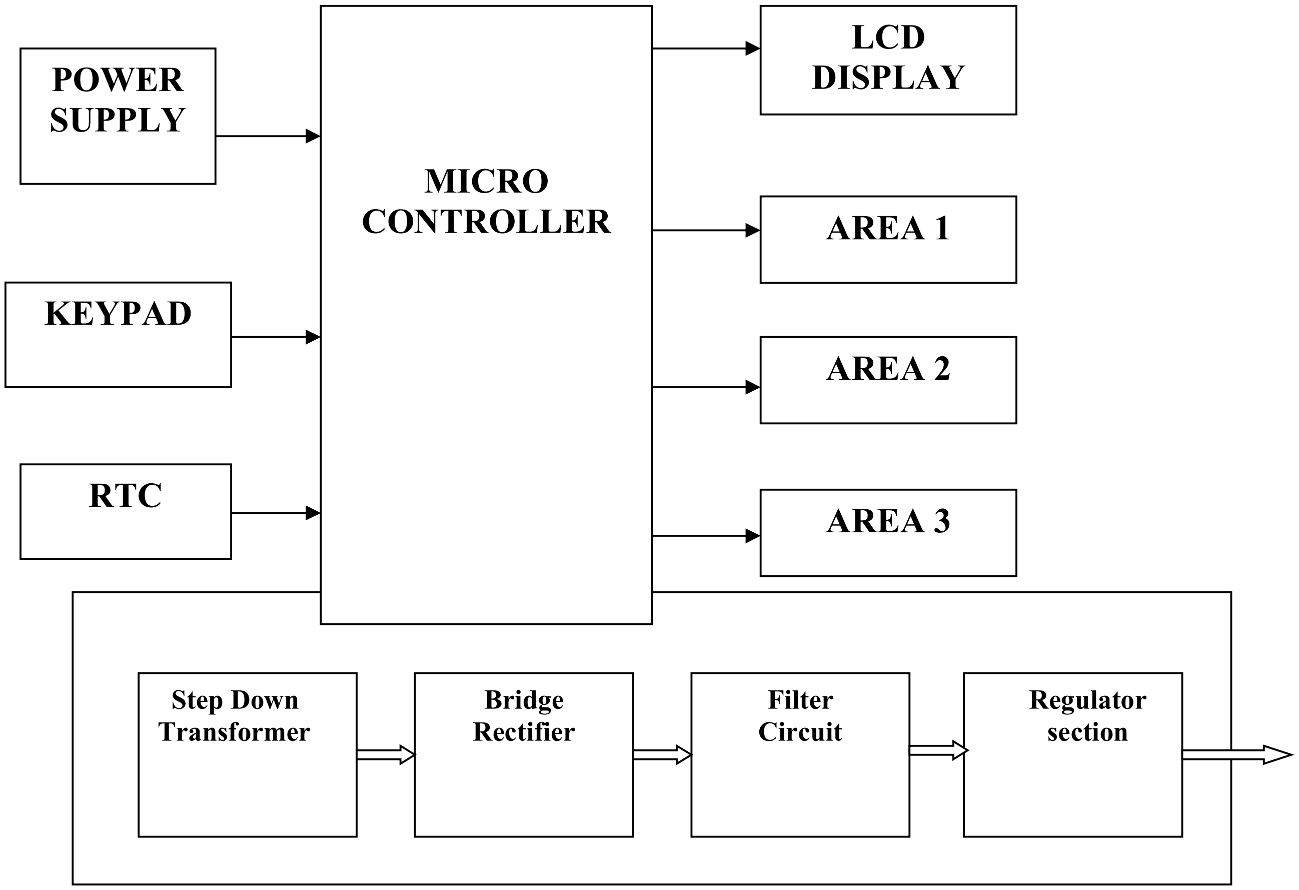


Figure 2.2.1 Block diagram

Figure 2.2.1 shows the block diagram of prototype. The Power supply unit supplies the required dual voltages to work. These dual voltages are supplied by this specially designed power supply. This unit consists of Step-down Transformer, Rectifier stage, Filter stage and Voltage Regulator Section.

Controller unit has Arduino Uno. It is a microcontroller board based on ATmega328P. Relay driver circuit used to switch power according to input signals. RTC modules are accustomed for timer-based power switching. Keypad is used as input unit and LCD is used as display unit.

The power is supplied and alarms are set all the three areas i.e. three relay driver modules, according to the requirement. Power switching is done with the help of RTC module and relay driver units.

# CHAPTER 3

# WORKING OF THE PROTOTYPE

In our project we mainly make use of the following:

1. AT-mega 328 micro controller
2. 3 Relay units
3. LCD Display
4. 4X1 Keyboard and
5. Controller unit

All these are connected as per to the block diagram.

The input to the microcontroller At-mega 328 is given using the 4X1 keypad This is stored in the four clocks which are present in the RTC. When the selected time comes the microcontroller will automatically provide the signals to the operating relay which will turn on, which will supply the power to the respective line which has to be turned on/off. This phenomenon is shown in the table below:

| Contactor | Clock 1 | Clock 2 | Clock 3 | Clock 4 | Clock 5 | Clock6 |
|---|---|---|---|---|---|---|
| Line No. | Line 1 | Line 2 | Line 3 | Line 1 | Line 2 | Line 3 |
| NO | OFF | OFF | OFF | ON | ON | ON |
| NC | ON | ON | ON | OFF | OFF | OFF |

Our project can be used for two purposes

1. When the power output connection is connected to the ‘NO’ contacts, then this will work as the “Sequential Power Distributor Project”
2. When the power output connection is connected to the ‘NC’ contacts of the Relay then this prototype can be used “Automatic power cutoff to different areas”

**Entering the data to microcontroller:**

1. By pressing the select switch on the keypad we can select the line number
2. Now by pressing the edit switch we should enter the time for which the relay should TURN ON and TURN OFF.
3. Now click on the save button in the keypad which will be displayed on the keyboard.
4. This will be saved in the E-Prom of the AT-mega 328 microcontroller.
5. When the allotted time comes the microcontroller will send the signal to the relay, this will Turn on (or) Turn off the relay as per the instruction given to the microcontroller.

After the data has been entered into the microcontroller our part of work is done the rest of the work is that is Turn on and Turn Off of the relay will be taken care by the microcontroller itself. This is how the prototype will work. The main application of the prototype is ‘Electrical substation feeder automation’.

# CHAPTER 4

# HARDWARE DESCRIPTION

## 4.1 POWER SUPPLY UNIT

The circuit needs two different voltages, +5V & +12V, to work. These dual voltages are supplied by this specially designed power supply.

This is a simple approach to obtain a 12V and 5V DC power supply using a single circuit. The circuit uses two ICs 7812(IC1) and 7805 (IC2) for obtaining the required voltages. The AC mains voltage will be stepped down by the transformer T1, rectified by bridge B1 and filtered by capacitor C1 to obtain a steady DC level. The IC1 regulates this voltage to obtain a steady 12V DC. The output of the IC1 will be regulated by the IC2 to obtain a steady 5V DC at its output. In this way both 12V and 5V DC are obtained.

Such a circuit is very useful in cases when we need two DC voltages for the operation of a circuit. By varying the type number of the IC1 and IC2, various combinations of output voltages can be obtained. If 7806 is used for IC2, we will get 6V instead of 5V. Same way if 7809 is used for IC1 we get 9V instead of 12V.

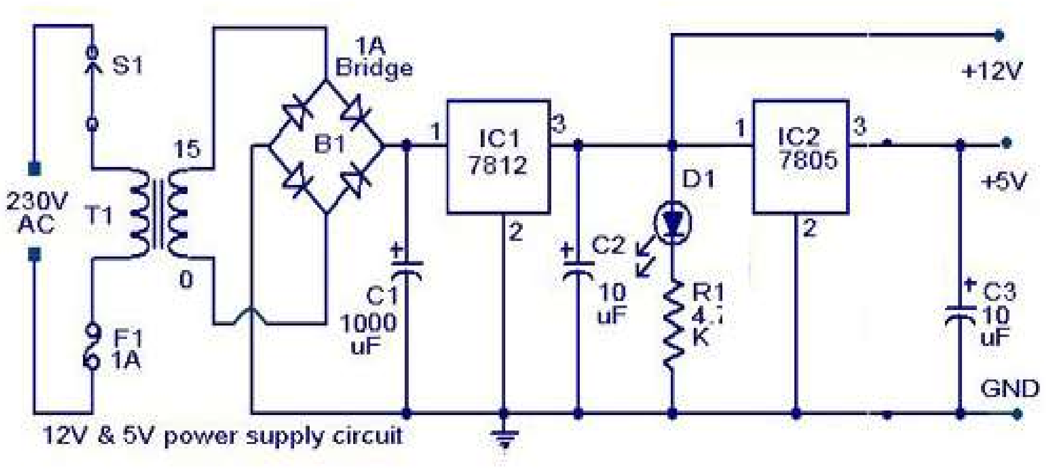


Figure 4.1.1 Power Supply Circuit

***1. Step-down Transformer****:* The conventional supply, which is generally available to the user, is 230V AC. It is necessary to step down the mains supply to the desired level. This is achieved by using suitably rated step-down transformer. While designing the power supply, it is necessary to go for little higher rating transformer than the required one. The reason for this is, for proper working of the regulator IC (say KIA 7812) it needs at least 2.5V more than the expected output voltage.

***2. Rectifier stage:*** Then the step-downed Alternating Current is converted into Direct Current. This rectification is achieved by using passive components such as diodes. If the power supply is designed for low voltage/current drawing loads/circuits (say +12V), it is sufficient to employ full-wave rectifier with center-tap transformer as a power source. While choosing the diodes the PIV rating is taken into consideration.

***3. Filter stage:*** But this rectified output contains some percentage of superimposed AC ripples. To filter this AC component filter stage is built around the rectifier stage. The cheap, reliable, simple and effective filtering for low current drawing loads say up to 50 mA) is done by using shunt capacitors. This electrolytic capacitor has polarities, take care while connecting the circuit.

***4. Voltage Regulation:*** The filtered D.C. output is not stable. It varies in accordance with the fluctuations in mains supply or varying load current. This variation of load current is observed due to voltage drop in transformer windings, rectifier and filter circuit. These variations in DC output voltage may cause inaccurate or erratic operation or even malfunctioning of many electronic circuits. For example, the circuit boards which are implanted by CMOS or TTL ICs.

The stabilization of D.C. output is achieved by using the three terminal voltage regulators IC. This regulator IC comes in two flavors: 78xx for positive voltage output and 79xx for negative voltage output. For example, 7812 gives +12V output and 7912 gives -12V stabilized output. These regulator ICs have in-built short-circuit protection and auto-thermal cutout provisions. If the load current is very high the IC needs ‘heat sink’ to dissipate the internally generated power.

## 4.2 CONTROLLER UNIT

**Arduino**

Arduino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with AC-to-DC adapter or battery to get started.

| | |
|---|---|
| Microcontroller | AT- mega 328P |
| Operating Voltage | 5V |
| Digital I/O Pins | 14 |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 20 mA |
| Flash Memory | 32 KB (ATmega328P) |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| UART | 2 |
| SPI | 1 |
| I2C | 1 |

Table 4.2.1 Arduino Pin Description Table

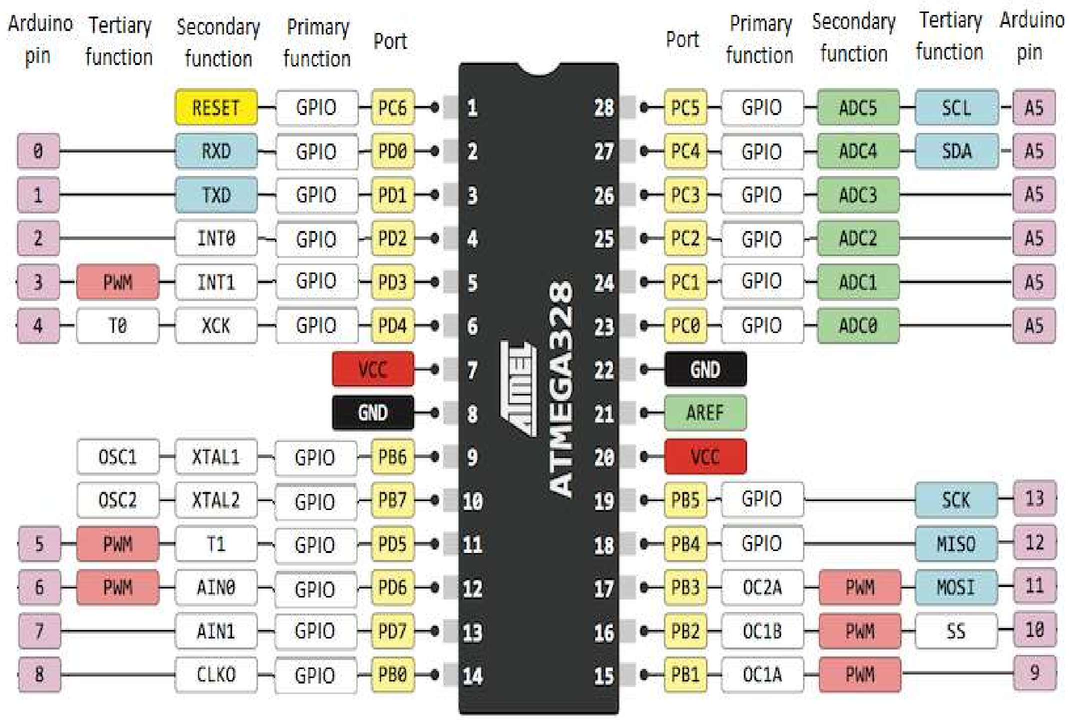

Arduino pin
Tertiary function
Secondary function
Primary function
Port
Port
Primary function
Secondary function
Tertiary function
Arduino pin
RESET GPIO PC6 1 28 PC5 GPIO ADC5 SCL A5
0 RXD GPIO PD0 2 27 PC4 GPIO ADC4 SDA A5
1 TXD GPIO PD1 3 26 PC3 GPIO ADC3 A5
2 INT0 GPIO PD2 4 25 PC2 GPIO ADC2 A5
3 PWM INT1 GPIO PD3 5 24 PC1 GPIO ADC1 A5
4 T0 XCK GPIO PD4 6 23 PC0 GPIO ADC0 A5
VCC 7 22 GND
GND 8 21 AREF
OSC1 XTAL1 GPIO PB6 9 20 VCC
OSC2 XTAL2 GPIO PB7 10 19 PB5 GPIO SCK 13
5 PWM T1 GPIO PD5 11 18 PB4 GPIO MISO 12
6 PWM AIN0 GPIO PD6 12 17 PB3 OC2A PWM MOSI 11
7 AIN1 GPIO PD7 13 16 PB2 OC1B PWM SS 10
8 CLKO GPIO PB0 14 15 PB1 OC1A PWM 9
ATMEGA328

Figure 4.2.1a Arduino pin description

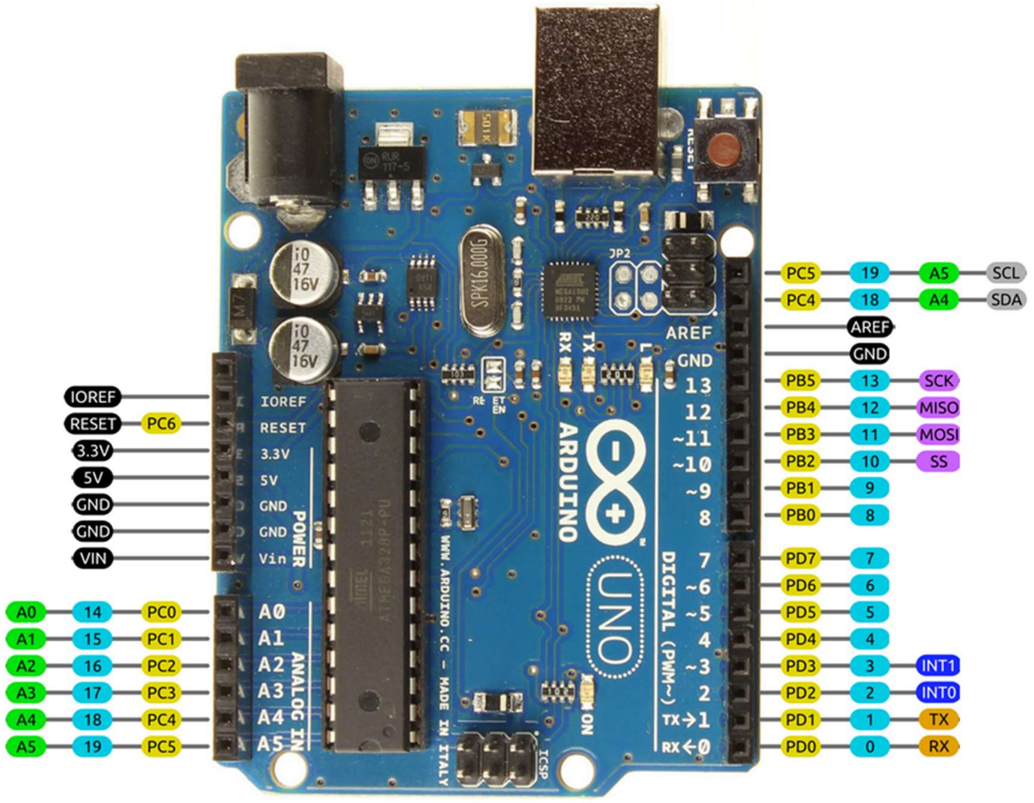


Figure 4.2.1b Arduino Board

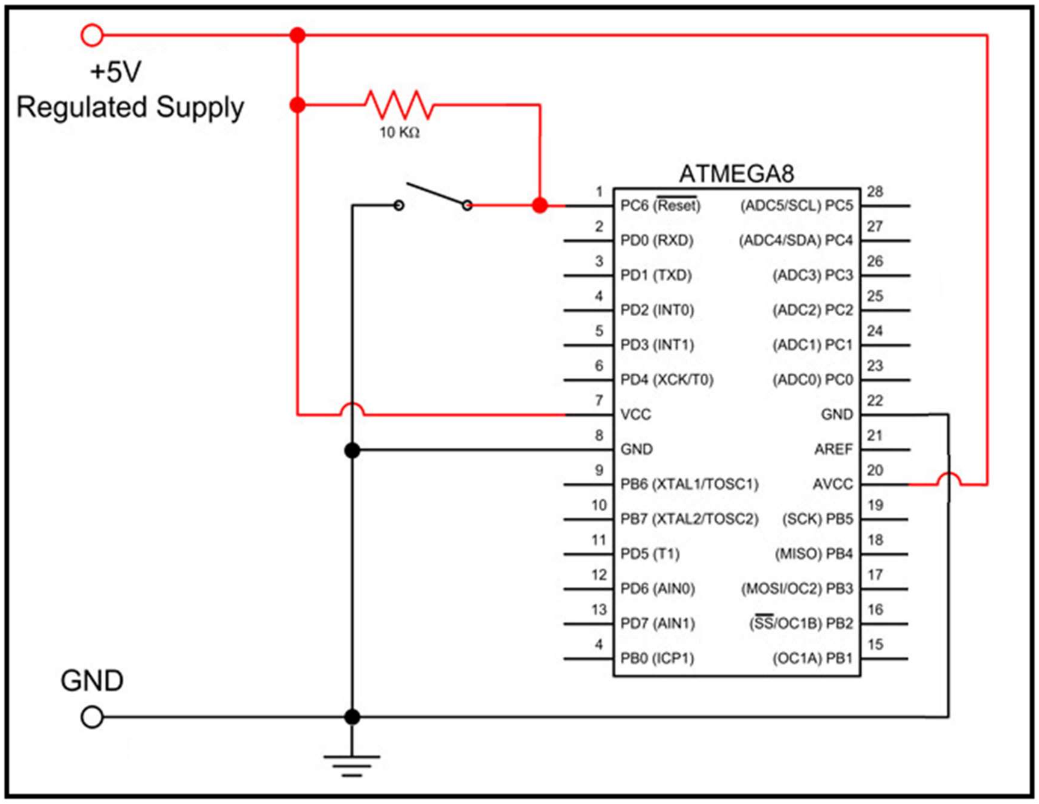


Figure 4.2.2 AT-mega 328 minimum circuit connection

## Atmega328 Memory:

Flash (32K) (15-bit addresses)

Program memory – read only

Non-volatile

Allocate data to Flash using PROGMEM keyword

SRAM (2K)

Temporary values, stack, etc.

Volatile

Limited space!

EEPROM (1K)

Long-term data

## Atmega328 Ports

Three 8-bit Ports (B, C, D)

Each port controlled by three 8-bit registers. Each bit controls one I/O pin

**DDRx**– Direction register

Defines whether a pin is an input (0) or and output (1)

**PINx**– Pin input value

Reading this "register" returns value of pin

**PORTx** – Pin output value

Writing this register sets value of pin

## 4.3 RELAY DRIVER

Relays are components which allow a low-power circuit to switch a relatively high current on and off, or to control signals that must be electrically isolated from the controlling circuit itself.

To make a relay operate, you have to pass a suitable ‘pull-in and ‘holding’ current (DC) through its energizing coil. And the relay coils are designed to operate from a particular supply voltage often 12V or 5V, in the case of many of the small relays used for electronics work. In each case the coil has a resistance which will draw the right pull-in and holding currents when it’s connected to that supply voltage. So, the basic idea is to choose a relay with a coil designed to operate from the supply voltage you’re using for your control circuit (and with contacts capable of switching the currents you want to control), and then provide a suitable ‘relay driver’ circuit so that your low-power circuitry can control the current through the relay’s coil. This will be somewhere between 25mA and 70mA.

Often your relay driver can be very simple, using little more than an NPN or PNP transistor to control the coil current. All your low-power circuitry has to do is provide enough base current to turn the transistor on and off.

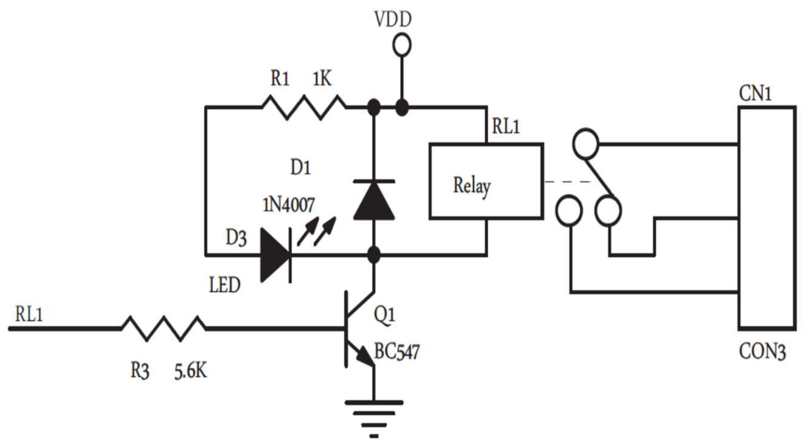


Figure 4.3.1 Relay Driver Unit

NPN transistor Q1 (say a BC547) is being used to control a relay (RLY1) with a 12V coil, operating from a +12V supply. Series base resistor R1 is used to set the base current for Q1, so that the transistor is driven into saturation (fully turned on) when the relay is to be energized. That way, the transistor will have minimal voltage drop, and hence dissipate very little power as well as delivering most of the 6V to the relay coil.

How do you work out the value of R1? It’s not hard. Let’s say RLY1 needs 50mA of coil current to pull in and hold reliably, and has a resistance of 240W so it draws this current from 6V. Our BC547 transistor will need enough base current to make sure it remains saturated at this collector current level.

To work this out, we simply make sure that the base current is greater than this collector current divided by the transistors minimum DC current gain hFE. So as the BC547 has a minimum hFE of 100 (at 100mA), we'd need to provide it with at least 50mA/100 = 0.5mA of base current.

As you can see a power diode D1 (1N4001 or similar) is connected across the relay coil, to protect the transistor from damage due to the back-EMF pulse generated in the relay coils inductance when Q1 turns off.

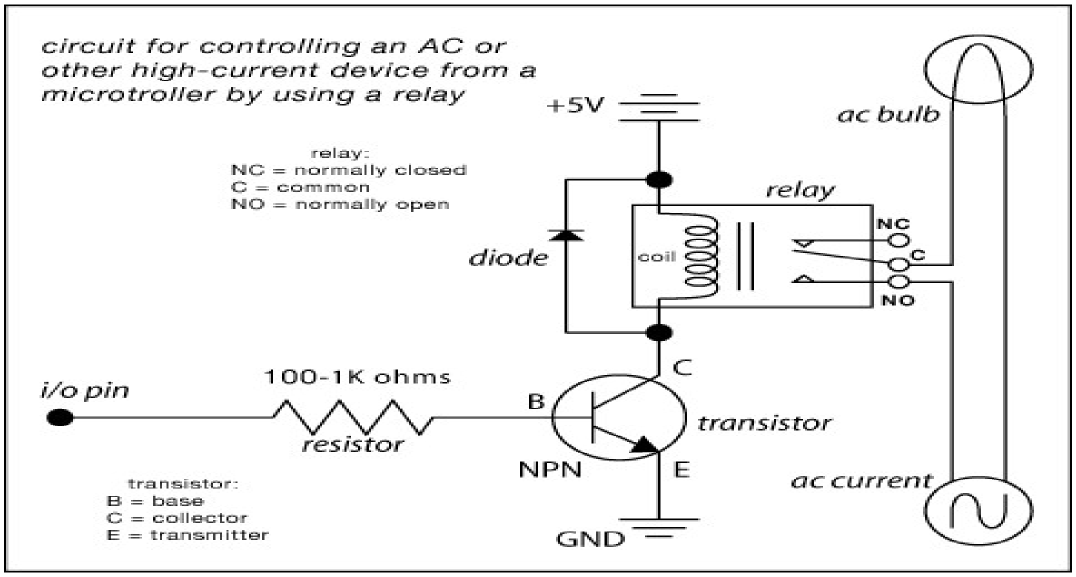
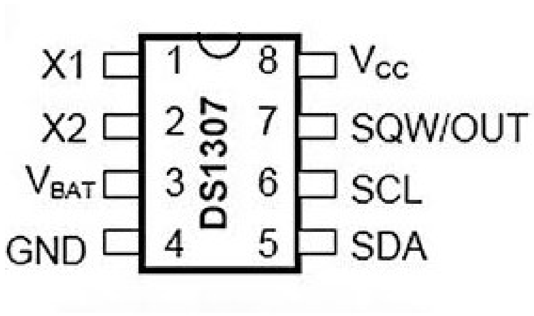


Figure 4.3.2 Relay Driver circuit

## 4.4 RTC INTERFACING

Real Time Clocks, as the name suggests are clock modules. They are available as integrated circuits (ICs) and manages timing like a clock. Some RTC ICs also manages date like a calendar. The main advantage is that they have a system of battery backup which keeps the clock/ca lender running even in case of power failure. A very small current is required for keeping the RTC alive. This in most case is provided by a miniature 3v lithium coin cell. So even if the embedded system with RTC is powered off the RTC module is up and running by the backup cell. This same technique is used in PC timing also. If you have opened your computer case you will notice a small coin cell in the mother board.



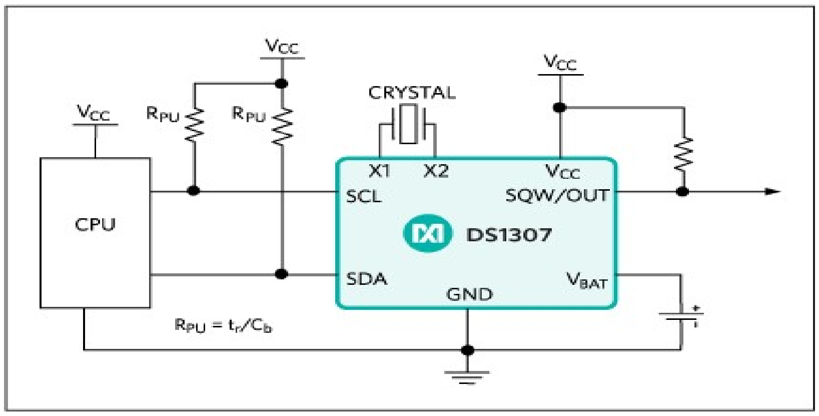


*Figure 4.4.1 RTC Interfacing*

**VCC (5V) Pin:** When this pin is high then the ds1307 sends the data and when it is low it runs on the backup button cell.

**GND:** This is the ground pin for the module. Both the ground of the battery and the power supply are tied together.

**SCL:** It is the i2c clock pin - Which communicates with the RTC.

**SDA:** It is the i2c data pin - Which communicates with the RTC.

The DS1307 is a low-power clock/calendar with 56 bytes of battery-backed SRAM. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The date at the end of the month is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The DS1307 operates as a slave device on the I2C bus.

**Features:**

- Two wire I2C interface
- Hour: Minutes: Seconds AM/PM
- Day Month, Date - Year
- Leap year compensation
- Accurate calendar up to year 2100
- Battery backup included
- 1Hz output pin
- 56 Bytes of Non-volatile memory available to user
- 0x68 I2C Address

**DS1307 Internal Registers**

From software point of view the DS1307 is just a collection of some 8bit registers. You can read this register to obtain the current time and date. You can also modify them to hold the correct time. After that the DS1307 keeps then updated with current date and time. The following registers are there.

| **ADD** | **BIT7** | **BIT6** | **BIT5** | **BIT4** | **BIT3** | **BIT2** | **BIT1** | **BIT0** | **FUNCTION** | **RANGE** |
|---|---|---|---|---|---|---|---|---|---|---|
| **00H** | CH | 10 SECONDS | | | SECOND | | | | **SECOND** | 00-59 |
| **01H** | 0 | 10 MINUTES | | | MINUTES | | | | **MINUTES** | 00-59 |
| **02H** | 0 | 12<br>24 | 10HR<br>am/pm | 10HR | HOUR | | | | **HOUR** | AM/PM<br>OR<br>00-23 |
| **03H** | 0 | 0 | 0 | 0 | 0 | DAY | | | **DAY** | 01-07 |
| **04H** | 0 | 0 | 10 DATE | | DATE | | | | **DATE** | 01-31 |
| **05H** | 0 | 0 | 0 | 10MONTH | MONTH | | | | **MONTH** | 01-12 |
| **06H** | 10 YEAR | | | | YEAR | | | | **YEAR** | 00-99 |
| **07H** | THIS IS USED FOR SQUARE WAVE GENERATION ON PIN7 | | | | | | | | | |
| **08-3FH** | | | | | | | | | 56 BYTE RAM | 00h-FFh |

Table 4.4.1 RTC Internal resistors

## 4.5 LCD

Liquid Crystal Display also called as LCD is very helpful in providing user interface as well as for debugging purpose. The most common type of LCD controller is HITACHI 44780 which provides a simple interface between the controller & an LCD.

These LCD's are very simple to interface with the controller as well as are cost effective. The most commonly used ALPHANUMERIC displays are 1x16 (Single Line & 16 characters), 2x16 (Double Line & 16 character per line), 4x20 (four lines & Twenty characters per line).

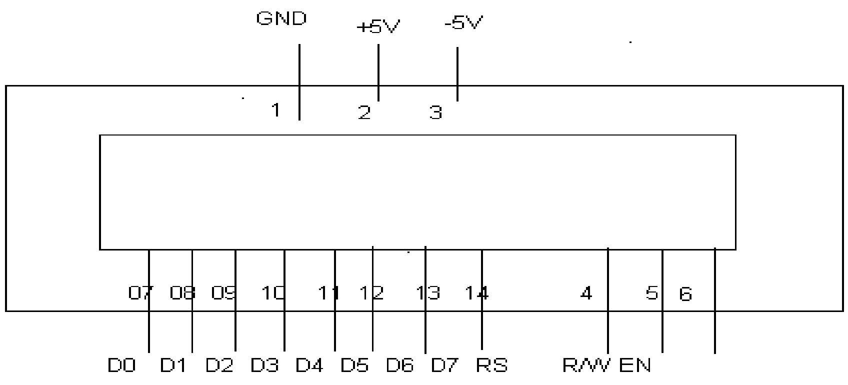


*Figure4.5.1 LCD*

**PIN DESCRIPTIONS: -**

**$V_{cc}$, $V_{ss}$ and $V_{ee}$: -**

While $V_{cc}$ and $V_{ss}$ provide +5V and ground respectively, Vee is used for controlling LCD contrast.

**RS Register Select: -**

There are two very important registers inside the LCD. The RS pin is used for their selection as follows.

If RS=0, the instruction command code register is selected, allowing the user to send a command such as clear display, cursor at home, etc.

If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.

**R/W, read/write: -**

R/W input allows the user to write information to the LCD or read information from it.

R/W = 1 for reading.

R/W= 0 for writing.

**EN, enable: -**

The LCD to latch information presented to its data pins uses the enable pin. When data is supplied to data pins, a high–to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450 ns wide.

**D0 – D7: -**

The 8–bit data pins, DO – D7, are used to send information to the LCD or read the contents of the LCD's internal registers.

To display letters and numbers, we send ASCII codes for the letters A–Z, a-z numbers 0-9 to these pins while making RS=1.

There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to home position or blink the instruction command codes.

We also use RS = 0 to check the busy flag bit to see if the LCD is ready to receive information. The busy flag is D7 and can be read when R/W=1 and RS=0, as follows: if R/W = 1, RS = 0. When D7= 1 (busy flag = 1), the LCD is busy taking care of internal operations and will not accept any information.

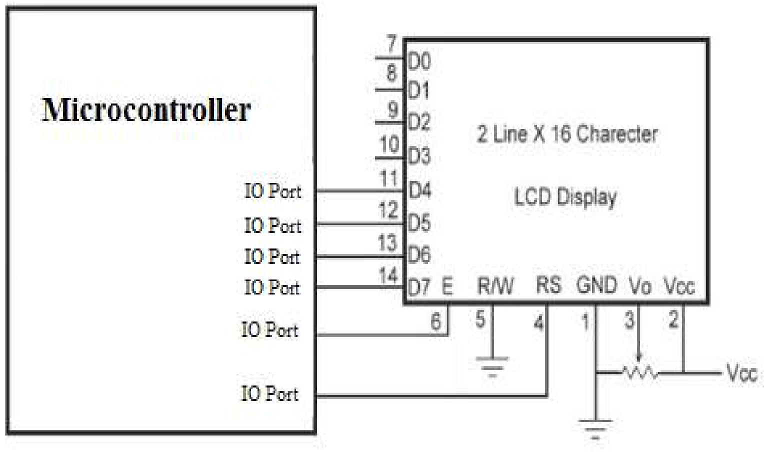


Figure 4.5.2 LCD Interfacing

| **Pin** | **Symbol** | **Function** |
|---|---|---|
| 1 | Vss | Ground |
| 2 | Vdd | Supply Voltage |
| 3 | Vo | Contrast Setting |
| 4 | RS | Register Select |
| 5 | R/W | Read/Write Select |
| 6 | En | Chip Enable Signal |
| 7-14 | DB0-DB7 | Data Lines |
| 15 | A/Vee | Gnd for the backlight |
| 16 | K | Vcc for backlight |

Table 4.5.1 LCD pin description

When RS is low (0), the data is to be treated as a command. When RS is high (1), the data being sent is considered as text data which should be displayed on the screen.

When R/W is low (0), the information on the data bus is being written to the LCD. When RW is high (1), the program is effectively reading from the LCD. Most of the times there is no need to read from the LCD so this line can directly be connected to Gnd thus saving one controller line.

The ENABLE pin is used to latch the data present on the data pins. A HIGH - LOW signal is required to latch the data. The LCD interprets and executes our command at the instant the EN line is brought low. If you never bring EN low, your instruction will never be executed.

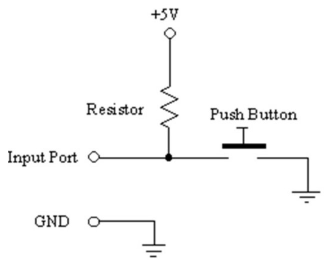
For Contrast setting a 10K pot should be used as shown in the figure.

Display Data Ram (DDRAM) stores the display data. When we have to display a character on LCD we basically write it into DDRAM. For a 2x16 LCD the DDRAM address for first line is from 80h to 8fh & for second line is 0c0h to 0cfh. If we want to display 'H' on the 7$^{th}$ position of the first line then we will write it at location 87h.

## 4.6 INTERFACING 4X1 KEYPAD

Keypad is a widely used input device with lots of application in our everyday life. From a simple telephone to keyboard of a computer, ATM, electronic lock, etc., keypad is used to take input from the user for further processing.

To interface a push button or switch to the Controller, all we need is just the push button or the switch itself and an accompanying pull-up resistor as shown below.



In this configuration, the bit that will enter to the Controller's port is HIGH (1) if the push button is not pressed.

If the push button is pressed, the bit that goes inside the computer is LOW (0). This is so since the input port will be connected directly to ground if the push button is pressed.

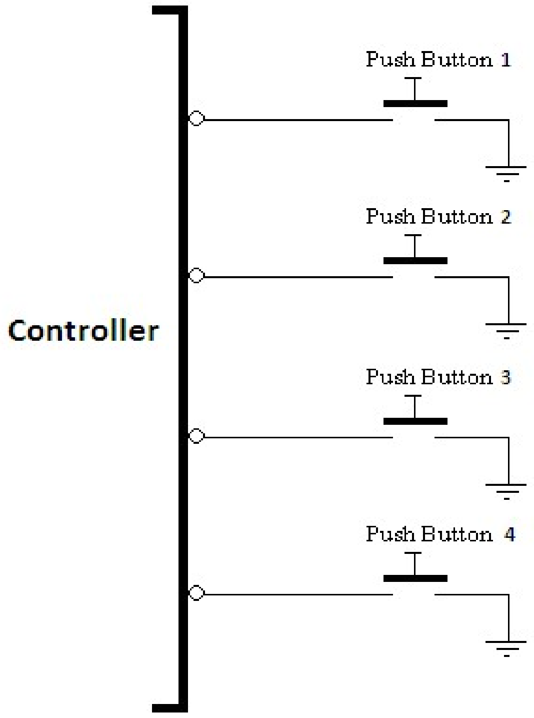


Figure 4.6.1 Keypad interfacing

**Working:**

If controller's first port is HIGH (1) then push button_1 is not pressed, else push button_1 is pressed.

Similarly, if controller's second port is HIGH (1) then push button_2 is not pressed, else push button_2 is pressed.

If controller's third port is HIGH (1) then push button_3 is not pressed, else push button_3 is pressed.

If controller's fourth port is HIGH (1) then push button_4 is not pressed, else push button_4 is pressed.

## 4.7 PC Interfacing

Debugging of 8051 Application can be very easy if we able to send debug information to serial port of PC. And its output can be seen on HyperTerminal (in Windows XP) or **Teraterm of Windows7/8/8.1** or Minicom (in Linux). We can display content of any variable, memory location etc. We can also print other useful information on serial terminal which could replicate the flow of the code.

**Circuit description**

Circuit is quite simple in any project we can add this facility to do serial communication with computer. For that UART section of 8051 Microcontroller is used. Rx and Tx line of microcontroller are connected to the TX and RX of USB to UART bridge.

This USB to UART bridge converts UART protocol (of 8051) to USB protocol (of PC) and vice versa.

8051 is programmed in such a way that serial communication will take place at the speed of 9600 bps.

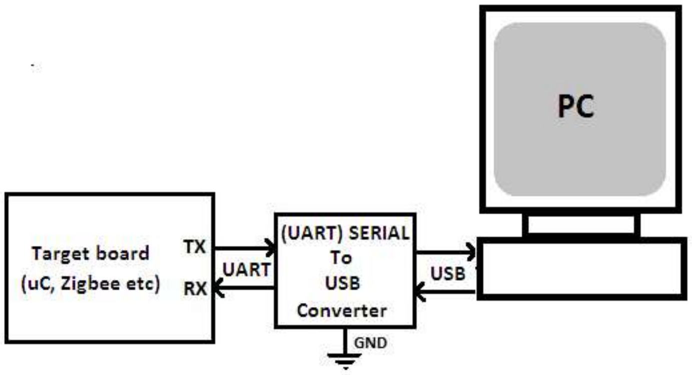


Figure 4.7.1 PC Interfacing

# CHAPTER 5

# SOFTWARE AND TOOLS

A program for Arduino may be written in any programming language for a compiler that produces binary machine code for the target processor. Atmel provides a development environment for their microcontrollers, AVR Studio and the newer Atmel Studio. The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one-click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus.

A program written with the IDE for Arduino is called a sketch. Sketches are saved on the development computer as text files with the file extension.ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub main() into an executable cyclic executive program with the GNU tool chain, also included with the IDE distribution. The Arduino IDE employs the program AVR code to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Arduino is a cross-platform IDE that works in conjunction with an Arduino controller in order to write, compile and upload code to the board.

The software provides support for a wide array of Arduino boards, including Arduino Uno, Nano, Mega, Esplora, Ethernet, Fio, Pro or Pro Mini, as well as Lily Pad Arduino. The universal languages for Arduino are C and C++, thus the software is fit for professionals who are familiar with these two. Features such as syntax highlighting, automatic indentation and brace matching makes it a modern alternative to other IDEs.

**Getting Started in the Arduino IDE**

1. **Verify:** Compiles and approves your code. It will catch errors in syntax (like missing semi-colons or parenthesis). The shortcut key for verify is Control + R.
2. **Upload:** Sends your code to the Arduino Board. When you click it, you should see the lights on your board blink rapidly. The shortcut key for upload is Control + U.
3. **New:** These buttons open up a new code window tab.
4. **Open:** This button will let you open up an existing sketch.
5. **Save:** This saves the currently active sketch.
6. **Serial Monitor:** This will open a window that displays any serial information your Arduino board is transmitting. It is very useful for debugging.
7. **Sketch Name:** This shows the name of the sketch you are currently working on.
8. **Code Area:** This is the area where you compose the code for your sketch.
9. **Message Area:** This is where the IDE tells you if there were any errors in your code.
10. **Text Console:** The text console shows complete error messages. When debugging, the text console is very useful.
11. **Board and Serial Port:** Shows you what board and the serial port selections.

## TERA TERM

**Tera Term** (rarely **Tera Term**) is an open-source, free, software implemented, terminal emulator (communications) program. It is a telnet program for communicating with serial ported devices such as modems, routers and programmable hobby kits etc.

It emulates different types of computer terminals, it supports telnet, SSH 1 & 2 and **serial port connections.**

In our project we use **Tera Term** program to access **serial port** of computer. Using Tera Term we can read the data sent by modem to serial port.

# CHAPTER 6
# PROGRAM SOURCE CODE

```
// Date and time functions using a DS1307 RTC connected via I2C and Wire lib
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>
#include "RTClib.h"
RTC_DS1307 rtc;

LiquidCrystal_I2C lcd(0x27, 16, 2);//int RW = A1;

#define key_1    1
#define key_2    2
#define key_3    3
#define key_4    4

#define relay_1   8
#define relay_2   9
#define relay_3   10

#define siren     A0

// Primary_menu ------------------
#define select_alarm   1
#define cancel_alarm   2
#define set_time       4

// Secondary_menu ----- select_alarm
#define select        1
#define up            2
#define down          3
#define back          4

// Secondary_menu ----- cancel_alarm
#define ok            1
#define up            2
#define down          3
```

```
#define back        4

// Secondary_menu ----- edit_time
#define save        1
#define HH          2
#define MM          3
#define back        4

//---------------------------------
#define on          1
#define off         0

#define YES         1
#define NO          0

#define playing     1
#define not_playing    2


int i, flag, modified_flag, selected_alarm, modified_hh, modified_mm;
int timer;

unsigned int current_hh, current_mm;

unsigned int alarm_1_hh, alarm_1_mm, alaram_1_status;
unsigned int alarm_2_hh, alarm_2_mm, alaram_2_status;
unsigned int alarm_3_hh, alarm_3_mm, alaram_3_status;
unsigned int alarm_4_hh, alarm_4_mm, alaram_4_status;
unsigned int alarm_5_hh, alarm_5_mm, alaram_5_status;
unsigned int alarm_6_hh, alarm_6_mm, alaram_6_status;

//EEPROM Address
#define al_1_h     0
#define al_1_m     1
#define al_1_s     2

#define al_2_h     4
#define al_2_m     5
#define al_2_s     6
```

```
#define al_3_h      8
#define al_3_m      9
#define al_3_s      10

#define al_4_h      12
#define al_4_m      13
#define al_4_s      14

#define al_5_h      16
#define al_5_m      17
#define al_5_s      18

#define al_6_h      20
#define al_6_m      21
#define al_6_s      22

void save_EEPROM(int alarm) //saves only perticular alarm info
{
  if (selected_alarm == 1)
  {
    alarm_1_hh = modified_hh;
    alarm_1_mm = modified_mm;
    alaram_1_status = on;

    EEPROM.write(al_1_h, modified_hh);
    EEPROM.write(al_1_m, modified_mm);
    EEPROM.write(al_1_s, on);
    LCD_clear();
    lcd.setCursor(0, 0);
    lcd.print("Alarm 1 Saved..     ");
    Serial.print("Alarm 1 Saved..     ");
    delay(2000);
  }

  if (selected_alarm == 2)
  {
    alarm_2_hh = modified_hh;
    alarm_2_mm = modified_mm;
    alaram_2_status = on;
    EEPROM.write(al_2_h, modified_hh);
    EEPROM.write(al_2_m, modified_mm);
    EEPROM.write(al_2_s, on);
  }

  if (selected_alarm == 3)
  {
    alarm_3_hh = modified_hh;
    alarm_3_mm = modified_mm;
    alaram_3_status = on;
    EEPROM.write(al_3_h, modified_hh);
    EEPROM.write(al_3_m, modified_mm);
    EEPROM.write(al_3_s, on);
  }

  if (selected_alarm == 4)
  {
    alarm_4_hh = modified_hh;
    alarm_4_mm = modified_mm;
    alaram_4_status = on;
    EEPROM.write(al_4_h, modified_hh);
    EEPROM.write(al_4_m, modified_mm);
    EEPROM.write(al_4_s, on);
  }

  if (selected_alarm == 5)
  {
    alarm_5_hh = modified_hh;
    alarm_5_mm = modified_mm;
    alaram_5_status = on;
    EEPROM.write(al_5_h, modified_hh);
    EEPROM.write(al_5_m, modified_mm);
    EEPROM.write(al_5_s, on);
  }

  if (selected_alarm == 6)
  {
    alarm_6_hh = modified_hh;
```

```
    alarm_6_mm = modified_mm;
    alaram_6_status = on;
    EEPROM.write(al_6_h, modified_hh);
    EEPROM.write(al_6_m, modified_mm);
    EEPROM.write(al_6_s, on);
  }
}

void read_EEPROM()  //read all alarm info
{
  alarm_1_hh      = EEPROM.read(al_1_h);
  alarm_1_mm      = EEPROM.read(al_1_m);
  alaram_1_status  = EEPROM.read(al_1_s);

  alarm_2_hh      = EEPROM.read(al_2_h);
  alarm_2_mm      = EEPROM.read(al_2_m);
  alaram_2_status  = EEPROM.read(al_2_s);

  alarm_3_hh      = EEPROM.read(al_3_h);
  alarm_3_mm      = EEPROM.read(al_3_m);
  alaram_3_status  = EEPROM.read(al_3_s);

  alarm_4_hh      = EEPROM.read(al_4_h);
  alarm_4_mm      = EEPROM.read(al_4_m);
  alaram_4_status  = EEPROM.read(al_4_s);

  alarm_5_hh      = EEPROM.read(al_5_h);
  alarm_5_mm      = EEPROM.read(al_5_m);
  alaram_5_status  = EEPROM.read(al_5_s);

  alarm_6_hh      = EEPROM.read(al_6_h);
  alarm_6_mm      = EEPROM.read(al_6_m);
  alaram_6_status  = EEPROM.read(al_6_s);
}

void check_alarm()
{
  if ((current_hh == alarm_1_hh) && (current_mm == alarm_1_mm))
  {
```

```
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Line 1 ON..      ");
  Serial.print("Line 1 ON..      ");
  digitalWrite(relay_1, LOW);
}

else if ((current_hh == alarm_2_hh) && (current_mm == alarm_2_mm))
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Line 2 ON..      ");
  Serial.println("Line 2 ON..      ");
  digitalWrite(relay_2, LOW);
}

else if ((current_hh == alarm_3_hh) && (current_mm == alarm_3_mm))
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Line 3 ON..      ");
  Serial.println("Line 3 ON..      ");
  digitalWrite(relay_3, LOW);
}

else if ((current_hh == alarm_4_hh) && (current_mm == alarm_4_mm))
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Line 1 OFF..      ");
  Serial.println("Line 1 OFF..      ");
  digitalWrite(relay_1, HIGH);
}

else if ((current_hh == alarm_5_hh) && (current_mm == alarm_5_mm))
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.println("Line 2 OFF..      ");
```

```
    Serial.println("Line 2 OFF..      ");
    digitalWrite(relay_2, HIGH);
  }

  else if ((current_hh == alarm_6_hh) && (current_mm == alarm_6_mm))
  {
    LCD_clear();
    lcd.setCursor(0, 0);
    lcd.print("Line 3 OFF..      ");
    Serial.println("Line 3 OFF..      ");
    digitalWrite(relay_3, HIGH);
  }

}



void setup()
{
  Serial.begin(9600);
  pinMode(key_1, INPUT_PULLUP);
  pinMode(key_2, INPUT_PULLUP);
  pinMode(key_3, INPUT_PULLUP);
  pinMode(key_4, INPUT_PULLUP);

  pinMode(relay_1, OUTPUT);
  pinMode(relay_2, OUTPUT);
  pinMode(relay_3, OUTPUT);

  digitalWrite(relay_1, LOW);
  digitalWrite(relay_2, HIGH);
  digitalWrite(relay_3, HIGH);
  delay(1000);
  digitalWrite(relay_1, HIGH);

  lcd.begin();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("   Welcome to     ");
  Serial.println("   Welcome to      ");
  lcd.setCursor(0, 1);
  lcd.print("CMRIT, Bangalore   ");

  delay(2000);
  LCD_clear();

  if (! rtc.begin())
  {
    lcd.setCursor(0, 0);
    lcd.print("RTC Error..       ");
    Serial.println("RTC Error..       ");
    while (1);
  }
  else{
    Serial.println("No Error");
  }

//  if (! rtc.isrunning())
//  {
//    Serial.println("RTC is NOT running!");
//    lcd.setCursor(0, 0);
//    lcd.print("RTC is NOT set     ");
//    Serial.println("RTC is NOT set     ");
//    rtc.adjust(DateTime(2017, M, D, HH, MM, SS));
//    rtc.adjust(DateTime(2021, 7, 12, 13, 16, 15));
//  }
//  else{
//    Serial.println("RTC Set");
//  }
  read_EEPROM();
}

void loop()
{
  DateTime now = rtc.now();
  current_hh = now.hour();
  current_mm = now.minute();
  check_alarm();
```

```
LCD_clear();
lcd.setCursor(0, 0);
lcd.print(" (HH:MM)  ");
Serial.print(" (HH:MM)  ");
if (now.hour() <= 9)
{
  lcd.print("0");
  Serial.print("0");
}
lcd.print(now.hour());
Serial.print(now.hour());

lcd.print(":");
Serial.print(":");
if (now.minute() <= 9)
{
  lcd.print("0");
  Serial.print("0");
}
lcd.print(now.minute());
Serial.println(now.minute());

lcd.setCursor(0, 1);
lcd.print("Edt Cnl ---- Tim");
Serial.println("Edt Cnl ---- Tim");

delay(250);

//Primary menu
if (digitalRead(select_alarm) == LOW)
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Edit timer Menu  ");
  Serial.println("Edit timer Menu  ");
  delay(2000);
  while (digitalRead(select_alarm) == LOW);
  modified_flag = NO;
  selected_alarm = 0;
  select_alarm_function();
  if (selected_alarm > 0)
  {
    edit_time_function();
    if (modified_flag == YES)
    {
      save_EEPROM(selected_alarm);
    }

    else
    {
      selected_alarm = 0;
    }
  }
}

else if (digitalRead(cancel_alarm) == LOW)
{
  LCD_clear();
  lcd.setCursor(0, 0);
  lcd.print("Cancl timer Menu  ");
  Serial.println("Cancl timer Menu  ");
  delay(2000);
  while (digitalRead(cancel_alarm) == LOW);
  selected_alarm = 0;
  select_alarm_function();
  if (selected_alarm > 0)
  {
    modified_hh = 88;
    modified_mm = 88;
    lcd.setCursor(0, 1);
    lcd.print("Timer ");
    Serial.println("Timer ");
    lcd.print(selected_alarm);
    lcd.print(" Canceled ");
    Serial.println(" Canceled ");
    save_EEPROM(selected_alarm);
    }
  }

  else if (digitalRead(set_time) == LOW)
  {
    delay(100);
    while (digitalRead(set_time) == LOW);
    modified_flag = NO;
    edit_time_function();
    if (modified_flag == YES)
    {
      updateTime();
    }

    else
    {
      //
    }
  }
}

void select_alarm_function()
{
  selected_alarm = 0;
  flag = 1;
  lcd.setCursor(0, 0);
  lcd.print("Select line: -    ");
  Serial.println("Select line: -    ");
  lcd.setCursor(0, 1);
  lcd.print("OK  UP  DN  Back    ");
  Serial.println("OK  UP  DN  Back    ");
  delay(2000);
  while (flag == 1)
  {
    if (digitalRead(select) == LOW)
    {
      lcd.setCursor(0, 1);
      lcd.print("Done...!            ");
      Serial.println("Done...!            ");
      delay(1000);
      while (digitalRead(select) == LOW);
      flag = 0;
    }

    if (digitalRead(up) == LOW)
    {
      if (selected_alarm < 6)
      {
        selected_alarm ++;
      }
      lcd.setCursor(14, 0);
      lcd.print(selected_alarm);
      Serial.println(selected_alarm);

      delay(100);
      while (digitalRead(up) == LOW);
    }

    if (digitalRead(down) == LOW)
    {
      if (selected_alarm > 1)
      {
        selected_alarm --;
      }
      lcd.setCursor(14, 0);
      lcd.print(selected_alarm);
      Serial.println(selected_alarm);

      delay(100);
      while (digitalRead(down) == LOW);
    }

    if (digitalRead(back) == LOW)
    {
      lcd.setCursor(14, 0);
      lcd.println("0");
      lcd.setCursor(0, 1);
      lcd.print("Menu canceled..      ");
```

```
      Serial.println("Menu canceled..       ");
      delay(1000);
      while (digitalRead(back) == LOW);
      selected_alarm = 0;
      flag = 0;
    }
  }
}

void edit_time_function()
{
  lcd.setCursor(0, 0);
  lcd.print("Edit time HH:MM ");
  Serial.println("Edit time HH:MM ");
  lcd.setCursor(0, 1);
  lcd.print("Save HH  MM Back");
  Serial.println("Save HH  MM Back");
  delay(2000);
  modified_flag = NO;
  modified_hh = 0;
  modified_mm = 0;
  flag        = 1;

  while (flag == 1)
  {
    if (digitalRead(save) == LOW)
    {
      lcd.setCursor(0, 1);
      lcd.print("Saving...        ");
      Serial.println("Saving...        ");
      while (digitalRead(save) == LOW);
      delay(1000);
      flag = 0;
      modified_flag = YES;
    }

    if (digitalRead(HH) == LOW)
    {
      modified_hh ++;
      if (modified_hh == 24)
      {
        modified_hh = 0;
      }
      lcd.setCursor(10, 0);
      if (modified_hh <= 9)
      {
        lcd.print('0');
      }
      lcd.print(modified_hh);
      Serial.println(modified_hh);
      delay(200);
    }

    if (digitalRead(MM) == LOW)
    {
      modified_mm ++;
      if (modified_mm == 60)
      {
        modified_mm = 0;
      }
      lcd.setCursor(13, 0);
      if (modified_mm <= 9)
      {
        lcd.print('0');
        Serial.println('0');
      }
      lcd.print(modified_mm);
      Serial.println(modified_mm);
      delay(200);
    }

    if (digitalRead(back) == LOW)
    {
      lcd.setCursor(0, 0);
      lcd.print("Edit time HH:MM ");
      Serial.println("Edit time HH:MM ");
      lcd.setCursor(0, 1);
      lcd.print("Cancelling....  ");
```

```
      Serial.println("Cancelling....  ");
      delay(1000);
      while (digitalRead(back) == LOW);
      flag = 0;
      modified_flag = NO;
      modified_hh = 0;
      modified_mm = 0;
    }
  }
}

void updateTime()
{
  rtc.adjust(DateTime(2017, 1, 1, modified_hh,  modified_mm,  0));
}

void LCD_clear()
{
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print("                ");
}
```

# CHAPTER 7

## ADVANTAGES AND DISADVANTAGES

| SLNO | ADVANTAGES | DISADVANTAGES |
|---|---|---|
| 1 | Reduced line losses | Limitations of technology. |
| 2 | Power quality can be maintained | Un-predictable development costs. |
| 3 | Deferred capital expenses | Initial costs are high. |
| 4 | Energy cost can be reduced | A skilled labour is always required. |
| 5 | Helps in optimal energy utilization | |
| 6 | Economic benefits | |
| 7 | Improved reliability | |
| 8 | Compatibility | |

# CHAPTER 8

# APPLICATIONS

Some of the applications of our project are:

1. Real-time Distribution Operation Model and Analysis (DOMA)
2. Fault location, Isolation and service restoration (FLIR)
3. Voltage/var control (VVC)
4. Distribution contingency analysis (DCA)
5. Multi-Level Feeder Reconfiguration (MFR)
6. Pre-arming of Remedial Action Schemes (PRAS)
7. Coordination of emergency actions (CEMA)
8. Relay Protection Re-coordination (RPRC)
9. Coordination of Restorative actions s(CRA)
10. Intelligent Alarm Processing (IAP)

# CHAPTER 9

## FUTURE SCOPE

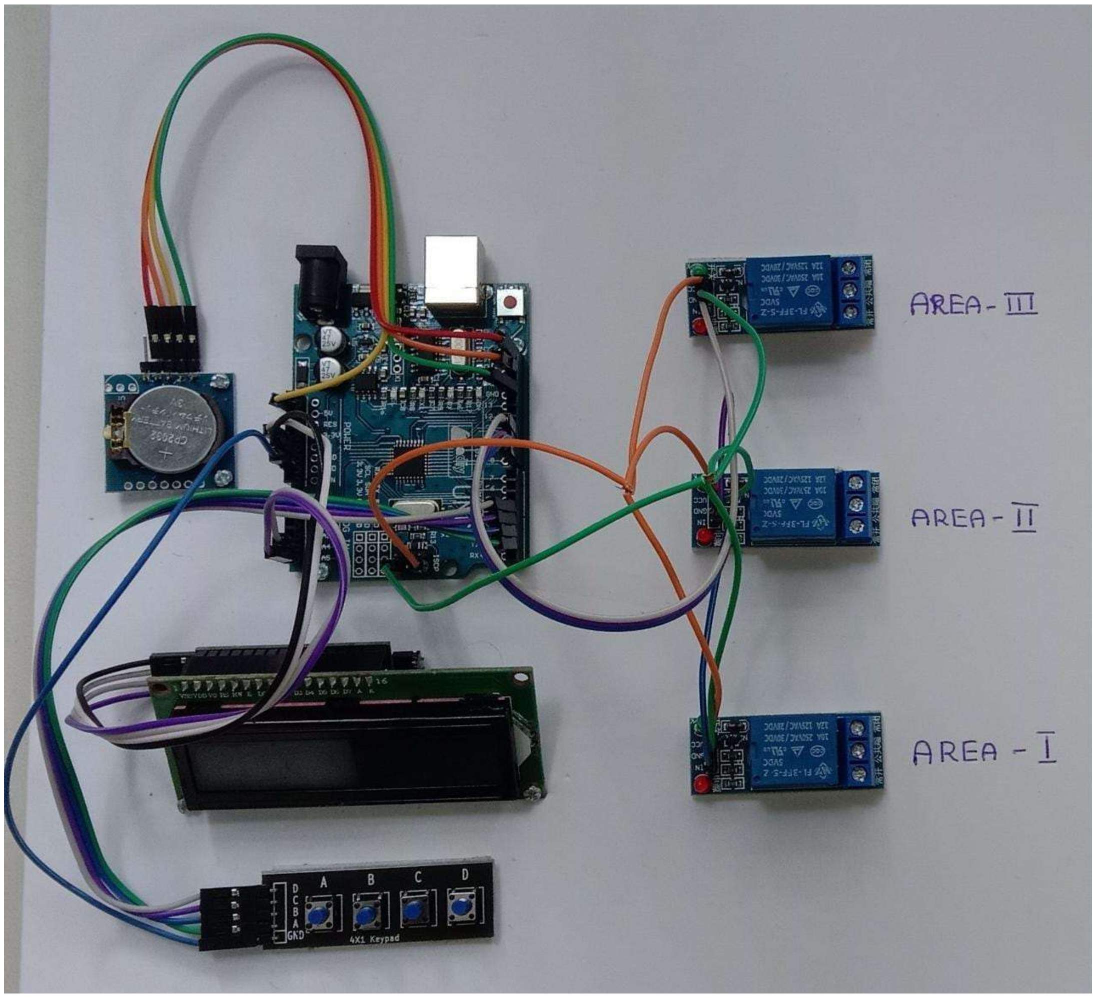
- ☐ Further the project can be enhanced by using appropriate sensors to know the which type of fault.
- ☐ Automatically distributing the existing power supply to the different areas.
- ☐ Automatic load shedding depending upon the consumer.

# CHAPTER 10
# CONCLUSION

The prototype of ‘Sequential power distribution’ to the different areas is designed. This prototype has the following features:

1) Automatic power supply to different areas
2) Automated device control
3) Automatic switching between two consecutive areas
4) Automatic time controller

At the end we conclude that by using our project we will be able to automatically distribute the power to the given three areas, and also if necessary this prototype can be extended to the number of areas as required.



# CHAPTER 11

# REFERENCES

## BOOK REFERENCES

1. Mohammad Ali Mazidi, “Microcontroller and embedded systems”,2nd edition, Pearson education.
2. GayakwadRamakant A, “Op-amps and Linear integrated circuits”, 4th edition.
3. Robert Horning, at al. “Serial programming”, wiki books, 2005.
4. K.J.Ayala, “The AT89s52 microcontroller”, 2nd edition, angage learning Englewood Cliffs, New Jersey, 1987.

## WEB REFERENCES

1. https://learn.adafruit.com/ds1307-real-time-clock-breakout-board-kit/wiring-it-up
2. www.circuitstoday.com/interfacing-lcd-to-arduino
3. www.instructables.com/id/interface-keypad-with-arduino
4. **https://ieeexplore.ieee.org/abstract/document/736248**
5. **https://ieeexplore.ieee.org/document/5057147**
6. **https://ieeexplore.ieee.org/document/7094902**