Final Year Project Report

On

**“POWERLINE PARAMETER MONITORING SYSTEM VIA IOT”**

Submitted in the fulfillment of the requirements

For the Degree of

Bachelor of Technology Semester VII & VIII

In

Electronics & Telecommunication Engineering

By

**Kanak Kharbanda (1714110262)**

**Anchal Gupta (1714110230)**

**Sushant Garg (1714110332)**

Under the guidance of

**Dr. RAJESH B. GHONGADE**



**Department of Electronics & Telecommunication Engineering**

**Bharati Vidyapeeth (Deemed To Be University)**

**College of Engineering, Pune – 4110043**

Academic Year 2020-21

**BHARATI VIDYAPEETH (DEEMED TO BE UNIVERSITY)**

**COLLEGE OF ENGINEERING, PUNE – 4110043**

**DEPARTMENT OF ELECTRONICS & TELECOMMUNICATION ENGINEERING**

**CERTIFICATE**

We hereby certify that the following work which is being presented in the Major project entitled **“TO DESIGN AND FABRICATE POWERLINE PARAMETER MONITORING SYSTEM VIA IOT*”*** in the partial fulfillment of requirement for the award of the degree of Bachelor of Technology (B.Tech) and submitted in **Department of Electronics and Telecommunication Engineering** of Bharati Vidyapeeth Deemed University College Of Engineering-Pune is an authentic record of our own work carried out by us during a period from August 2020 to April 2021 under the supervision of **Dr. R.B. Ghongade**, Professor, Department of Electronics and Telecommunication Engineering, Bharati Vidyapeeth Deemed University College Of Engineering-Pune. The matter presented in this report embodies the result of our work and studies carried out and have not been submitted anywhere else.

Date: May 2021

Dr. R. B. Ghongade Prof. D.K.Ray Dr. S.K.Oza

Guide Project Coordinator HOD

**Table of Contents**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Chapter No.** | **Name of Topic** | | | | **Page No.** | |
|  | **List of Figures** | | | | **i** | |
|  | **List of Tables** | | | | **ii** | |
|  | **List Of Abbreviations** | | | | **iii** | |
|  | **Abstract** | | | | **iv** | |
| **Chapter- 1** | **Introduction** | | | | **6** | |
|  | **1.1** | Purpose | | | **6** | |
|  | **1.2** | Scope | | | **9** | |
|  | **1.3** | Block Diagram | | | **10** | |
| **Chapter-2** | **Literature Survey** | | | | **11** | |
| **Chapter- 3** | **Project Plan** | | | | **13** | |
|  | **3.1** | | | Cost Estimation | **13** | |
|  | **3.2** | | | Plan of Action | **13** | |
| **Chapter- 4** | **System Implementation :** | | | | **14** | |
|  | **4.1** | Arduino Integrated Development Environment ([**IDE**](https://en.wikipedia.org/wiki/Integrated_development_environment)) | | | **14** | |
|  | **4.2** | MQTT Protocol | | | **16** | |
|  | **4.3** | | LCD 20X4 | | **17** | |
|  | **4.4** | | | ESP 32 | | **19** | |
|  | **4.5** | | | PZEM 004T V3 | | **22** | |
|  | **4.6** | | | PCB | | **28** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Chapter 5** | **Interfacing** | | **31** |
| **Chapter-6** | **Results** | | **44** |
| **Chapter- 7** | **CONCLUSION** | | **47** |
|  | **Bibliography** | | **48** |
|  |  | Research Papers | **49** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| **1.1** | Block diagram | **10** |
| **2.1** | Smart Grid | **12** |
| **4.1** | MQTT Working | **16** |
| **4.2** | LCD Pinout | **18** |
| **4.3** | ESP 32 Solo series | **21** |
| **4.4** | ESP 32 WROOM series | **21** |
| **4.5** | PZEM 004T V3 | **22** |
| **4.6** | PZEM 004T Functional Block Diagram | **23** |
| **4.7** | DS3231 RTC Module | **26** |
| **4.8** | DS3231 RTC Module Pinout | **26** |
| **4.9** | LCD i2C | **27** |
| **4.10** | PCB | **28** |
| **4.11** | Schematic Diagram | **29** |
| **4.12** | PCB Layout | **30** |
| **5.1** | ESP &DS3231 Interfacing(1) | **31** |
| **5.2** | ESP &DS3231 Interfacing(2) | **31** |
| **5.3** | ESP32,PZEM004T and DS3231interfacing(1) | **33** |
| **5.4** | Outputs | **33** |
| **5.5** | Load connected with Pzem | **34** |
| **5.6** | Interfacing Of LCD | **35** |
| **6.1** | Arduino Application Output | **44** |
| **6.2** | Output on LCD | **44** |
| **6.3** | MQTT Client Android app Outputs | **45** |
| **6.4** | MQTT Explorer Desktop Application Outputs | **46** |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **NAME OF THE TABLE** | **PAGE NO.** |
| **1** | ARDUINO PRO IDE | **14** |
| **2** | LCD PINOUTS | **19** |
| **3** | RTC Pin Configuration | **25** |

**List Of Abbreviations:**

* Temp: Temperature
* LCD: Liquid Crystal Display
* DHT: Digital Humidity and Temperature
* e.g.: for example
* AREF: Analogue Reference
* Tx: Transmitting
* Rx: Receiving
* GND: Ground
* I/O: Input/Output
* KB: Kilobyte
* mA: Milliampere
* MHz: Megahertz
* MAX: Maximum
* IC: Integrated Circuit

# **ABSTRACT**

IOT has become a part of the modern world; the significance and utilization are increasing with each passing day. This approach is to design an efficient and real-time wireless network to monitor power consumption of electrical appliances. A sensor is set at the heap to ascertain current; a circuit is utilized to figure voltage and with these two, power can be computed. Control qualities are put away in cloud database. A web facilitating and space is made to get the orders from android application and send them to raspberry pi board at load, which triggers an electromagnetic transfer to change the condition of the heap. This project permit to get the power values and control gadgets from anyplace on the planet.

# **Introduction**

The Internet of Things (IoT) is a concept that aims to extend the benefits of continuous internet connection for various things such as data sharing, remote control, and data monitoring. IoT is the idea of researchers who want to optimize equipment’s such as sensor, radio frequency identification (RFID), wireless sensor network, and all equipment’s connected to the Internet network to communicate with humans. IoT’s challenge is to bridge the physical world and the information world such as the processing of data obtained from electronic equipment via an interface between the user and the equipment. Sensors collect physical data such as temperature and humidity, and then send it to the server to be stored in database or be displayed on the application interface.

IoT technology has started being developed in manifestations such as the Smart City, which has

been developed in many metropolitan cities of the world, Smart Factory, which is used for production optimization, Tele Diagnostic which facilitates the process of the patient's health monitoring, weather information system, which is used for weather prediction, and so forth. Two of the processes that are often used in IoT applications are process control and monitoring (acquisition) of data.

Communication protocols for acquisition of data that can be used are HTTP and MQTT protocol. HTTP protocol is used to web data exchange where it can usually be used for handling data acquisition of hardware, but the ability of HTTP is less suitable when being used on IoT-based applications. HTTP protocol uses a request/response model, which is currently the most common message exchange protocol. MQTT uses a publish/subscribe pattern. HTTP protocol is not designed for pervasive network and can cause a decrease in performance, especially in bandwidth usage and battery durability. The protocol for IoT applications requires less bandwidth, real-time response, and low energy use because it is commonly used for small appliances.

A study conducted by Vergara et.al showed the superiority of MQTT implemented on Android devices, which can decrease energy consumption compared to HTTP protocol. MQTT and HTTP are both running over TCP. However, MQTT provides some advantages such as low energy, among others. Compared to IoT protocols such as COAP, MQTT also provides advantage to be used on client type of smartphone. This study aims to implement the usability of MQTT protocol for monitoring power parameters interfaced on mobile-based and web-based applications.

**1.1 PURPOSE**

Now a days, energy eﬃciency is a major concern for every researcher in the ﬁeld of power system. In order to maximize the energy eﬃciency of power system such infrastructure is needed that can tolerate maximum disturbances like transients, harmonics, voltage sags or swells, voltage surges and voltage imbalances. Development in energy eﬃciency is generally accomplished by developing more energy eﬃcient technologies, making system reliable and making energy eﬃcient. Currently, the transmission line infrastructure is highly vulnerable due to several reasons such as natural disaster and manmade mishaps, which can badly aﬀect the stability and overall performance of the system.

Additionally, redundancy in the consumption of power leads to increase the losses of transmission lines. So, there is necessity to modify the transmission lines with eﬀective communication system in order to monitor its diﬀerent parameters and to support several reviews like real time monitoring, faster fault identiﬁcation and exact fault diagnosis. By introducing information and communication technologies into a Traditional Grid (TG), it can transform into Smart Grid (SG).SG refers as integration of two-way communication technologies, distributed energy resources and generation, smart technologies, and advanced electricity storage system into a TG. All above features make SG more reliable, less stringent, self-healing and less vulnerable.SG is comprised of various components like Control Center (CC), substations, transmission lines, towers and with advanced information and communication technologies. Distance between two substations is too high due to this reason towers are used to support long distance transmission lines. Substation transmits information to CC in every few seconds. The speed of communication link between the substation and CC is quite slow, thus there is need to improve the communication links. Due to development of technologies, it is anticipated that more power companies will use higher bandwidth and low latency communication links, e.g., optical ﬁber. Now a day’s, optical communication is better one than other communication links due to its better capacity, low latency, and high reliability. On the other side, one factor that overshadow the performance of optical ﬁber is its high installation and maintains cost. However, it is impossible to deploy optical ﬁber communication links along the transmission lines, thus a wireless communication idea is proposed in. A wireless communication technology provides cheaper, less complicated, and highly ﬂexible solution than that of optical ﬁber. However, wireless technology is widely used in smart grid. Wireless Sensor Networks (WSNs) play a signiﬁcant role in the monitoring of transmission lines. For real time status monitoring, diﬀerent type of sensors are placed on various location of transmission lines. Due to short range of communication between sensors and relay nodes, sensors are installed near to transmission tower while relay nodes on top of the tower.

Theses sensors are responsible for collecting ﬁne measurement of information and employ short range communication for data transmission to relay node. All relay nodes communicate with each other and transmit accumulative data to substation. Aftermost, data will be transmit from substation to CC through optical ﬁber. At the CC, data will be compare with existing data and correct decision will be taken after comparing. Delivering the ﬁne measurements of information to CC with cost eﬃcient and timely manner is a critical challenge for an intelligent smart grid

**1.2 SCOPE**

The scope is to provide grid observability, controllability of assets, enhance power system performance and security, reduction in operating cost, maintenance and system planning. To accommodate a wide variety of generation centralized and distributed, intermittent and dispatchable. To communicate with energy management system in smart buildings to enable customers to manage their energy use and reduce their energy cost. To provide improved power quality to the users. To provide real time information, lower operation cost and electricity available to everyone. To use information technology for monitoring and control to optimize its capital and operational cost. To predict and instantly respond to system problems in order to avoid power outages and power quality problems. To make the nation energy independent. To provide employment. Smart Grids is not felt to be a necessity only for the integration of distributed generation, renewable energy sources and plug-in (hybrid) cars into the electricity grid but also for active participation of consumers for improvements in overall system efficiency, meet the peak demand without investment in generation and variable pricing system. Failure in protection Mechanism If there is any failure in protection mechanism network then it is required to restore network from failure to normal operation in the fastest possible time. This protection mechanism can be divided into two topics as identification, diagnosis and recovery of failure & predication and prevention of failure.

1. Identification, diagnosis and recovery of failure - If failure occurred , it must be identified quickly in the shortest possible time to avoid future damaging or cascading of event.

2. predication and prevention of failure. In this subsection, the protection mechanism briefly reviewed. **Failure to Predict and Prevent:** In smart grid to locate weak points is also one approach to predict the failure location. Chertkov have developed an approach to efficiently predict power grid weak points, and identify probable failure mode in static load distribution In preventing network failure instead of accurate predicting the weak point accurate forecasting of short circuit fault and its magnitude in smart grid are also essential. In power distribution systems to perform short-circuit current forecast, Chen implemented the artificial neural network (ANN). The algorithm was verified on hardware system based on TMS320F2812-DSP and the formulated model was demonstrated through computer simulation In the shortest possible time the algorithm was verified to be efficient in predicting the magnitude of short circuit Failure to Identify, Diagnose and Recover: If there is any failure in the system it is necessary to identify in the shortest possible time, so that further damaging or cascading is to be avoided. Again when fault is cleared then the system must be resynchronized and restored quickly to normal operation, based on the design of Petri Net (PN) theory Calderaro et al., [10] presented a method to identify and localize failure in smart grid. By using matrix operation, from the captured modeling data in distribution network we can detects the failure in data transmission and fault in distribution network. they have verified the method. In their research, with two case studies while avoiding occurrence of cascading failures in power system protection, described its effectiveness and identify the method is useful to erase more complexity associated in data analysis and permit quick assessment and evaluation of information.

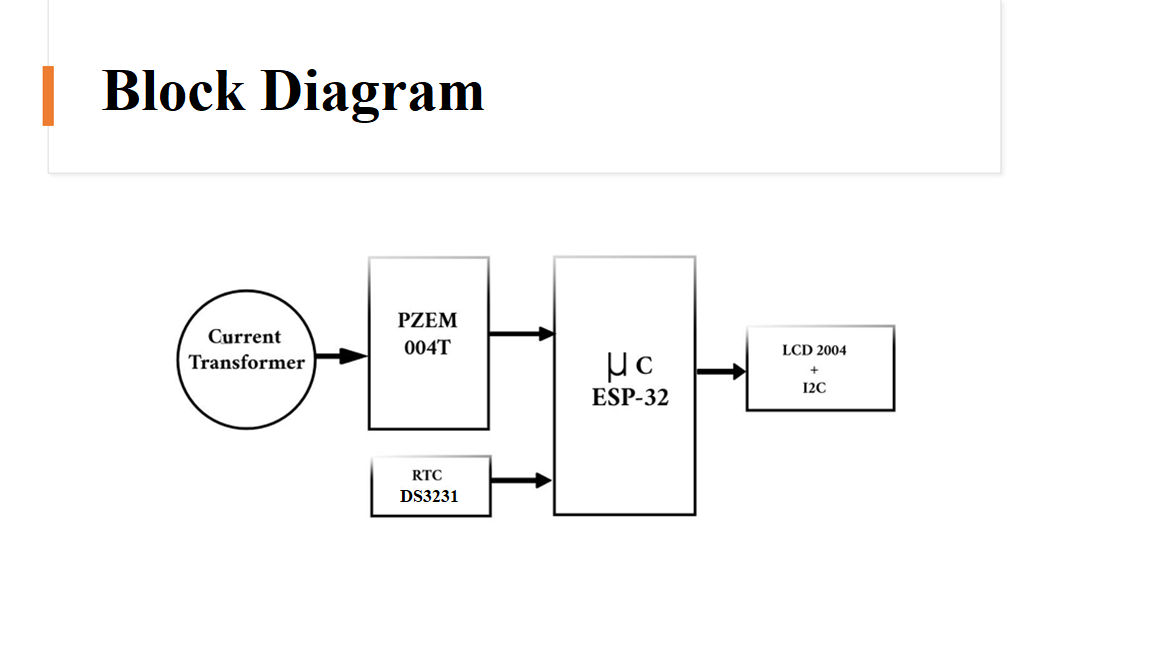


Fig 1.1: **Block Diagram**

1. **LITERATURE SURVEY**

By carrying out a study of references and scientific proposals on SGs, we can retrieve thousands of sources, most of them related to Smart Power Grids, Power Generation Systems, Electric Power Systems, Economy and Management of the Electric Network, and Control and Optimization of Electric Sustainability 2020, 12, 8662 4 of 26 Systems . However, a list of SGs with sensors applied to the home limits the list of sources quite a bit. We will focus our review on indoor environments to narrow down the sources that have developed similar research. By reviewing the related literature, we find sources that focus their research on the fields of the Internet of Things (IoT), SGs, Energy Engineering Computing, Wireless Sensor Networks, and Home Automation, among others. We shall now highlight some of the studies found within these fields for their level of citations, their presence in high-impact journals or special relevance to the subject studied in this work. As described trends in IoT, cloud computing, autonomous control, and artificial intelligence. This work discusses the need for synchronization of the Internet, wireless sensors, and actuators along with distributed computing to enable successful technologies for IoT. The regulatory aspect referred to here is interesting. Other works propose research on IoT applied to mobility, taking into account that this aspect is a human need and that there are many technological resources that can improve it, such as the use of sensors and intelligent devices in vehicles. In this work, we also include these sources, which are relevant to the study of SGs, since electric mobility is a basic factor in energy management, environmental sustainability, and, of course, to improve people’s quality of life, especially vulnerable groups such as the disabled, the elderly, and others. In an overview is given of the wide variety of sensors applied to intelligent environments. It catalogues and classifies these according to uses and applications. Sensors applied to health, agriculture, environment, energy sources, among others, are analyzed. The references to manufacturers, technologies, and future trends studied in this work are interesting. This work offers valuable information to advance in the study of sensors and their application in intelligent environments, taking into account climate change mitigation policies and the energy transition. In relation to the hierarchical architecture of the intelligent network to evaluate the advantages of distributed data processing, one paper offers a specific analysis of the sensors present in the home. In addition, local data processing is analyzed through automatic learning algorithms integrated in the distributed

system. There are studies that integrate the design of sensors and HAR in applications whose purpose is energy or electricity demand control . It is essential to study the response of stochastic demand in the smart grid considering the use patterns of random devices. An advance in this type of research will undoubtedly improve the quality of life of users . User interactions, household appliances, and human activity recognition data and their relationship with household equipment are analyzed. An automated method is proposed to determine when an electrical device is activated by household residents only from its energy trail. The approach is interesting, since the interaction of people with the equipment is analyzed through energy and consumption measurement. Data and consumption management are taken into account for the analysis of the efficiency and sustainability of the actions we study, an important vector for our analysis, since in our study we will consider the sustainable development goals and the challenge of the energy transition.

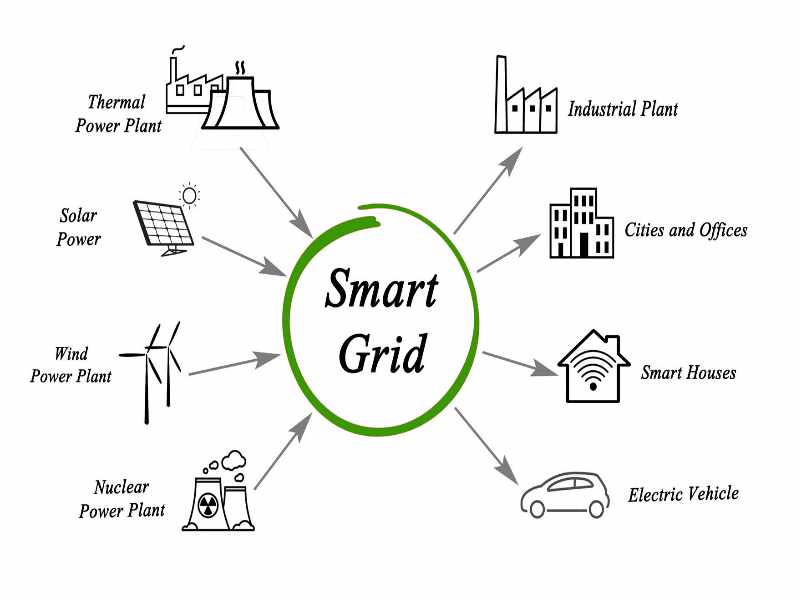


Fig. 2.1 Smart Grid

# **PROJECT PLAN**

## **COST ESTIMATION**

The overall project cost has been estimated taking into account the various other costs that had been encountered en-routed to development of the project.

Overall Cost of the project is around Rs.5000 which includes the cost of all the components required.

These costs include:

* + - Study Material: This includes the expenses incurred in purchasing books and the expense incurred for web browsing for collection of information.
    - Cost of components: This includes the expenses incurred in purchasing the modules and other material for circuitry of the monitoring system.

## **3.2 Plan of Action**

* + 1. Select appropriate sensors.
    2. Select microcontroller and power supply.
    3. Select a display.
    4. write code for all the interfacing and upload on micro controller
    5. check the output on display.

# **System Implementation**

**4.1. Software Used:**

## **Arduino Integrated Development Environment (**[**IDE**](https://en.wikipedia.org/wiki/Integrated_development_environment)**):-**

This is a [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) application (for [Windows,](https://en.wikipedia.org/wiki/Windows) [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux)) that is written in functions from [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B_(programming_language))[.](https://en.wikipedia.org/wiki/Arduino_IDE#cite_note-3) It is used to write and upload programs to [Arduino](https://en.wikipedia.org/wiki/Arduino) compatible boards, but also, with the help of third-party cores, other vendor development boards.

The source code for the IDE is released under the [GNU General Public License,](https://en.wikipedia.org/wiki/GNU_General_Public_License) version 2.The Arduino IDE supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B) using special rules of code structuring. The Arduino IDE supplies a [software library](https://en.wikipedia.org/wiki/Software_library) from the [Wiring](https://en.wikipedia.org/wiki/Wiring_(development_platform)) 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](https://en.wikipedia.org/wiki/Cyclic_executive) program with the [GNU toolchain,](https://en.wikipedia.org/wiki/GNU_toolchain) also included with the IDE distribution.The Arduino IDE employs the program *avrdude* 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. By default, avrdude is used as the uploading tool to flash the user code onto official Arduino boards.[[9]](https://en.wikipedia.org/wiki/Arduino_IDE#cite_note-9)

Arduino IDE is a derivative of the [Processing IDE,](https://en.wikipedia.org/wiki/Processing_(programming_language))[[10]](https://en.wikipedia.org/wiki/Arduino_IDE#cite_note-10) however as of version 2.0, the Processing IDE will be replaced with the [Visual Studio Code](https://en.wikipedia.org/wiki/Visual_Studio_Code)-based [Eclipse Theia](https://en.wikipedia.org/wiki/Eclipse_Theia) IDE framework.[[2]](https://en.wikipedia.org/wiki/Arduino_IDE#cite_note-2.0-beta-2)

**Table 1 :Arduino Pro IDE**

|  |  |
| --- | --- |
| [**Developer(s)**](https://en.wikipedia.org/wiki/Software_developer) | Arduino Software |
| [**Preview release**](https://en.wikipedia.org/wiki/Software_release_life_cycle#BETA) | v0.1.2 / 14 September 2020; 8 months ago |
| [**Repository**](https://en.wikipedia.org/wiki/Repository_(version_control)) | [github.com/Arduino/Arduino](https://github.com/arduino/Arduino) |
| **Written in** | [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B) |
| [**Operating system**](https://en.wikipedia.org/wiki/Operating_system) | [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Linux](https://en.wikipedia.org/wiki/Linux) |
| [**Platform**](https://en.wikipedia.org/wiki/Computing_platform) | [IA-32](https://en.wikipedia.org/wiki/IA-32), [x86-64](https://en.wikipedia.org/wiki/X86-64), [ARM](https://en.wikipedia.org/wiki/ARM_architecture) |
| [**Type**](https://en.wikipedia.org/wiki/Software_categories#Categorization_approaches) | [Integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) |
| [**License**](https://en.wikipedia.org/wiki/Software_license) | [LGPL](https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License) or [GPL](https://en.wikipedia.org/wiki/GNU_General_Public_License) license |
| **Website** | [blog.arduino.cc/2020/08/24/cli-and-ide-get-](https://blog.arduino.cc/2020/08/24/cli-and-ide-get-better-together/)  [better-together/](https://blog.arduino.cc/2020/08/24/cli-and-ide-get-better-together/) |

With the rising popularity of Arduino as a software platform, other vendors started to implement custom open source compilers and tools (cores) that can build and upload sketches to other [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller) that are not supported by Arduino's official line of microcontrollers

## **4.1.2. Mosquitto MQTT Broker**

Mosquitto is a lightweight open source message broker that implements MQTT versions 3.1.0, 3.1.1 and version 5.0 It is written in C by Roger Light, and is available as a free download for Windows and Linux and is an Eclipse project.

**Installing The Broker**

Starting and Stopping The Broker

Depending on the install it will probably be started automatically on **system startup.**

Although this is very desirable in production environments it is less so in test environments. My preferred approach is to stop the mosquitto service and start it manually from the command prompt.

This gives you access to the console which is invaluable for testing.

On **Windows** you can stop the service if it is running by using the **control panel>admin>services.**

You can also the net command:

net stop mosquitto On **Linux** use:

sudo service mosquitto stopsudo systemctl stop mosquitto.service

Starting from command line is the best option when testing and to do that use.

**Windows and Linux**

mosquitto -v #start in verbose mode To see other start options use:

mosquitto -h

By default the broker will start listening on **port 1883**.

You can change that by editing the configuration file-mosquitto.conf. See [Quick Guide to The](http://www.steves-internet-guide.com/mossquitto-conf-file/) [Mosquitto.conf File With Examples](http://www.steves-internet-guide.com/mossquitto-conf-file/)

Alternatively you can use a command line switch to specify the port e.g. mosquitto -p 1884

You can run Multiple brokers on the same machine by starting them on different ports See this video- on [running multiple brokers.](https://youtu.be/dHH1kSQLhsU)

Mosquitto Client Programs

The mosquitto install includes the client testing programs.

There is a simple subscriber client

**mosquitto\_sub**

and a publisher client

**mosquitto\_pub**

They are useful for some quick tests.

Diagram

Description automatically generated

Fig4.1 MQTT Working

## **MQTT PROTOCOL:-**

MQTT (Message Queuing Telemetry Transport) protocol is protocol specifically designed for "machine to machine" communication. MQTT protocol runs over TCP / IP and has a data packet size with low overhead minimum (> 2 bytes) so that consumption of the power supply is also small enough. This protocol is a data-agnostic protocol that can transmit data in various forms such as binary

data, text, XML, or JSON and this protocol uses a publish/subscribe model rather than a client-server model.

Stack TCP/IP is now widely supported by microcontroller likeSTM32Fx7 series, as well as common market device board such as Wemos and Raspberry Pi. There are so many options for implementing MQTT protocol on devices. A common system of MQTT requires two main software components.

The advantage of publish/subscribe system is that the data sender (publisher) and the data receiver (client) do not know each other because there is a broker between both. In addition, there is time decoupling which makes publisher and client unable to be connected simultaneously so that client will stay to receive delayed data previously.

MQTT has 14 types of control signal, namely:

1. CONNECT — Client request to connect to Server

2. CONNACK — Connection Acknowledgement

3. PUBLISH — A message which represents a new/separate publish

4. PUBACK — QoS 1 Response to a PUBLISH message

5. PUBREC — First part of QoS 2 message flow

6. SUBSCRIBE — A message used by clients to subscribe to specific topics

7. SUBACK — Acknowledgement of a SUBSCRIBE message

8.UNSUBSCRIBE — A message used by clients to unsubscribe from specific topics

9. UNSUBACK — Acknowledgement of an UNSUBSCRIBE message

10. PINGREQ — Heartbeat message

11. PINGRESP — Heartbeat message acknowledgement

12.DISCONNECT — Graceful disconnect message sent by clients before disconnecting

13.PUBREL — Second part of QoS 2 message flow

14.PUBCOMP — Last part of the QoS 2 message flow

* 1. **Hardware Used:**

**4.2.1 LCD 20X4 :-**

The [LCD](https://www.theengineeringprojects.com/2019/12/introduction-to-20-x-4-lcd-module.html) stands for liquid crystal display, that works on the light modulation features of liquid crystals. It is available in electronic visible display, video display and flat panel display. There are numerous categories and features are exits in markets of LCD and you can see it on your mobile, laptop, computer and television screen. The invention of LCD gives new life to electronic industries and replaces lED and gas plasma techniques. It also replaces the CTR (cathode ray) tube that used for visual display. The input power consumed by the liquid crystal display is less then light-emitting [diode](https://www.theengineeringprojects.com/2018/05/introduction-to-diode.html) and plasma display. In today's post, we will have look at 20 x 4 LCD, its features, working, applications, and practical implementation in different electronic devices.

* In a **20x4 LCD** module, there are four rows in display and in one row twenty character can be displayed and in one display eighty characters can be shown.
* This liquid crystal module uses HDD44780 (It is a controller used to display monochrome text displays) parallel interfacing.
* The liquid crystal display interfacing code is easily accessible. We just required eleven input and output pinouts for the interfacing of the LCD screen.
* The input supply for this module is three volts or five volts, with that module other components like [PIC](https://www.theengineeringprojects.com/2018/11/introduction-to-pic16f877.html), [Raspberry PI](https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html), Arduino.
* Thie electronic device can be used in different embedded systems, industries, medical devices, and portable devices like mobile, watches, laptops.
* Liquid crystal display works on two types of the signal first one is data and the second one is for control.
* The existence of these signals can be identified through the on and off condition of RS pinout. Data can be read by pushing the Read/write pinout.

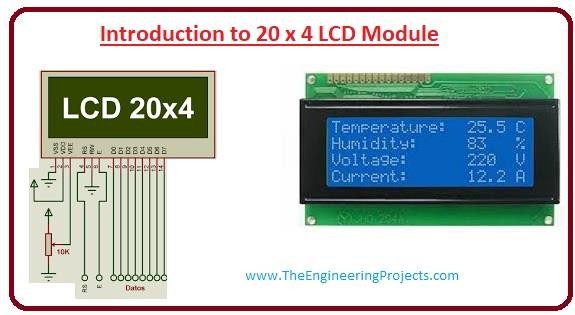


Fig4.2: 20x4 LCD Pinout

These are some pinouts of 20x4 LCD modules that are described here with the detailed.

|  |  |  |
| --- | --- | --- |
| **Pin No:** | **Pin Name:** | **Parameters** |
| Pin#1 | It denoted as Vss | It is ground pinout potential at this pinout is zero. |
| Pin#2 | It denoted as Vdd | At this pinout, five volts are provided. |
| Pin#3 | This pinout denoted as Vo | This pinout is used to set the contrast of the screen. |
| Pin#4 | This pin denoted as RS | It used to H/L register select signal. |
| Pin#5 | It denoted as R/W | It used for H/L read/write signal. |
| Pin#6 | This pinout denoted as E | It used for H/L enable signal. |
| Pin#7-14 | The pinouts from seven to fourteen denoted as DB0 – DB7. | It used for H/L data bus for 4 bit or 8-bit mode. |
| Pin#15 | It identified as A (LED+) | It used to set backlight anode. |
| Pin#16 | It recognized as K (LED-). | It used to set backlight cathode. |

Table 2 LCD 20X 4 Pinout

### **Features of 20 x 4 LCD**

* These are some features of 20 x 4 LCD modules that are described here with the detailed.
* The most important feature of this module is that it can display 80 characters at a time.
* The cursor of this module has 5x8 (40) dots.
* On this module already assembled the controller of RW1063.
* This module operates on the plus five volts input supply and can also work on the plus three volts.
* The plus three volts pinout can also be used for the negative supply.
* The duty cycle of this module is one by sixteen (1/16).
* The light-emitting [diode](https://www.theengineeringprojects.com/2018/05/introduction-to-diode.html) of this module can get supply from the pinout one, pinout two, pinout fifteen, pinout sixteen, or pinout A and K.

## **4.2.2 ESP 32:-**

### **Robust Design**

Icon

Description automatically generatedESP32 is capable of functioning reliably in industrial environments, with an operating temperature ranging from –40°C to +125°C. Powered by advanced calibration circuitries, ESP32 can dynamically remove external circuit imperfections and adapt to changes in external conditions.

### **Ultra-Low Power Consumption**

Engineered for mobile devices, wearable electronics and IoT applications, ESP32 achieves ultra-low power consumption with a combination of several types of proprietary software. ESP32 also includes state-of-the-art features, such as fine-grained clock gating, various power modes and dynamic power scaling.

A picture containing light

Description automatically generated

### **High Level of Integration**

ESP32 is highly-integrated with in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. ESP32 adds priceless functionality and versatility to your applications with minimal Printed Circuit Board (PCB) requirements.

* **Hybrid Wi-Fi & Bluetooth Chip**

ESP32 can perform as a complete standalone system or as a slave device to a host MCU, reducing communication stack overhead on the main application processor. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI / SDIO or I2C / UART interfaces.

* **Wi-Fi & Bluetooth Dual Mode**

The integration of Wi-Fi, Bluetooth and Bluetooth LE ensures that a wide range of applications can be targeted, and that our modules are truly versatile. Using Wi-Fi ensures connectivity within a large radius, while using Bluetooth allows the user to easily detect (with low-energy beacons) a module and connect it to a smartphone.

**High Integration**

With in-built antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules, our chips add priceless functionality and versatility to your applications with minimal PCB requirements.

**Configurability and Customization**

ESP32 modules can be ordered with different antenna configurations (e.g. PCB antenna, antenna connector) and flash sizes, so that they correspond to the needs of different applications. ESP32 modules also offer manufacturing customizations with the pre-programming of application firmware, custom data and pre-provisioning with cloud certificate.

#### **ESP32-SOLO Series**

The ESP32-SOLO-1 module is based on ESP32-S0WD and has integrated flash memory, thus providing a cost-effective solution for simple Wi-Fi and Bluetooth/Bluetooth LE-based connectivity applications.

Qr code

Description automatically generated with medium confidence

#### Fig 4.3 - ESP32-SOLO Series

#### **ESP32-WROOM Series**

These are ESP32-D0WD-based modules with integrated flash. These modules are well suited for Wi- Fi and Bluetooth/Bluetooth LE-based connectivity applications and provide a solid dual-core performance.

Text

Description automatically generated

#### Fig 4.4: ESP32-WROOM Series

**4.2.3. PZEM-004T v3.0:**

## 

Fig 4.5: PZEM-004T V3

PZEM-004T is the best for the purpose of the DIY project, where we need to measure the voltage, current and power using Arduino/ESP8266/Raspberry Pi like opensource platform. In many electrical projects, engineer directly deals with measurements with few basic requirements like

* High galvanic isolation
* Parameter display
* Direct communication with computer
* Data acquisition and storage with subsequent viewing or copying to the computer.

This module serve all these basic requirements of measurement PZEM-004T as a separate board. The physical dimensions of the PZEM-004T board is 3.1×7.4 cm, The pzem-004t module is bundled with 33mm diameter current transformer coil.

The main part of the PZEM-004T module is the SD3004 chip from the SDIC Microelectronics Co., Ltd. In addition, the board having the EEPROM from Atmel(now microchip) 24C02C which is a 2K bit Serial Electrically Erasable PROM with a voltage range of 4.5V to 5.5V. with More than 1 Million Erase/Write Cycles and 200+ Years Data Retention. Two optocouplers PC817, providing galvanic isolation of the serial interface.

## **Module Features**

* Measuring consumed electricity
* Serial interface UART with a speed of 9600
* Supply voltage 5V
* The possibility of connecting LCD or LED displays

| **Specifications** | **Descriptions** |
| --- | --- |
| Working voltage | 80 ~ 260VAC |
| Current measurement | 0 – 100 A |
| Rated power | 22kW |
| Operating frequency | 45-65Hz |
| Measurement accuracy | 1.0 grade |

## Problem PZEM-004T Version 3.0 - Projects made with Blynk - Blynk Community

Fig 4.6 PZEM-004T-10A Functional block diagram

## **Function**

1. Electrical parameter measurement function such as voltage, current, connected load and total consumption.  
2. The power button clear function.  
3. Power-down data storage function (cumulative power down before saving).  
4. The serial communication function (comes with TTL serial interface, via various terminals communicate with the adapter plate, read, and set the parameters).

## **Display Format**

**1. Power: measurement range 0 ~ 22kW**

* 0 ~ 10kW within the display format of 0.000 to 9.999;
* Within 10 ~ 22kW display format 10.00 ~ 22.00.

**2. Power: measurement range 0 ~ 9999kWh**

* 0 ~ 10kWh within the display format of 0.000 to 9.999;
* 10 ~ 100kWh within the display format of 10.00 to 99.99;
* 100 ~ 1000kWh within the display format of 100.0 to 999.9;
* 1000 ~ 9999kWh and above the display format from 1000 to 9999.

**3. Voltage: Test Range 80 ~ 260VAC**

* Display Format 110.0 ~ 220.0.

**4. Current: measurement range 0 ~ 100A**

* Display Format 00.00 to 99.99.

## **Button**

A built-in button on the panel, used to make an electricity clear function.  
Electricity clear method: Press ZERO key five seconds, then release the button!  
Press again to clear the key, the data is cleared and exit charge cleared, cleared so far completed.

## **Serial Communication**

The module is equipped with a TTL serial data communication interface via the serial port can be read and set the relevant parameters; but if you want a device with a USB or RS232 (such as computers) to communicate, then you need to be equipped with different TTL adapter hardware board (USB communication needs with TTL to USB adapter plate; RS232 communication needs with TTL to RS232 adapter plate), the adapter plate of the specific connection details can be found in FIG.

## **Considerations**

1. This module is suitable for indoor, not outdoor use.  
2. The applied load should not exceed the rated power.  
3. The wiring can not be wrong.

**4.2.4 RTC DS3231:**

**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.

### **DS3231 RTC Pin Configuration**

**DS3231** is a six terminal device, out of them two pins are not compulsory to use. So we have mainly four pins. These four pins are given out on other side of module sharing the same name.

|  |  |
| --- | --- |
| **Pin Name** | **Description** |
| VCC | Connected to positive of power source. |
| GND | Connected to ground. |
| SDA | Serial Data pin (I2C interface) |
| SCL | Serial Clock pin (I2C interface) |
| SQW | Square Wave output pin |
| 32K | 32K oscillator output |

### **Table 3. RTC Pin Configuration**

### **DS3231 RTC MODULE Features**

* RTC counts seconds, minutes, hours and year
* Accuracy: +2ppm to -2ppm for 0ºC to +40ºC , +3.5ppm to -3.5ppm for -40ºC to +85ºC
* Digital temperature sensor with ±3ºC accuracy
* Two Time-of-day alarms
* Programmable square wave output
* Register for Aging trim
* 400Khz I2C interface
* Low power consumption
* Automatic power failure battery switch circuitry
* CR2032 battery backup with two to three year life
* Potable size

### **DS3231 RTC MODULE Specifications**

* Operating  voltage of  DS3231 MODULE: 2.3V – 5.5V
* Can operate on LOW voltages
* Consumes 500nA on battery backup
* Maximum voltage at SDA , SCL : VCC + 0.3V
* Operating temperature: -45ºC to +80ºC

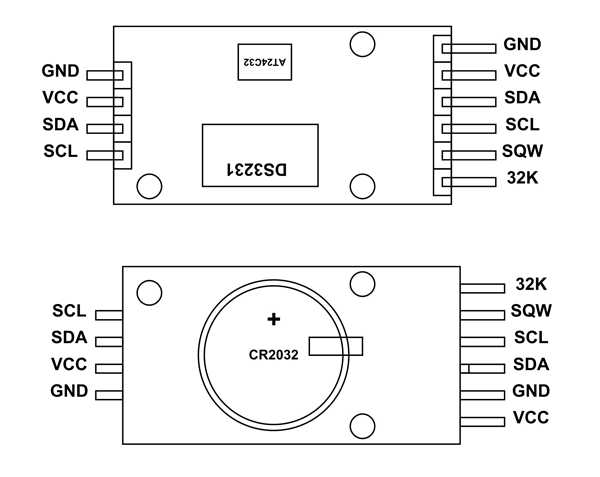
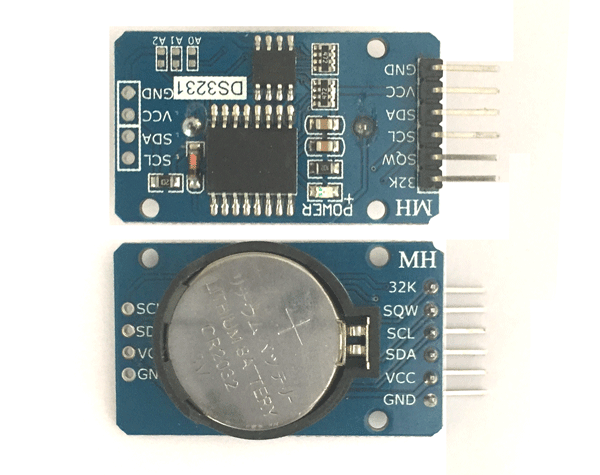


Fig.4.8 DS3231 RTC Module Pinout

Fig4.7. RTC 3231

### **Applications**

* Robotics
* Gaming
* Servers
* Computer Peripherals
* GPS
* Utility power meters

**4.2.5: LCD I2C:**

The I2C 1602 LCD module is a 2 line by 16 character display interfaced to an I2C daughter board. The I2C interface only requires 2 data connections, +5 VDC and GND to operate.

**Specification**

* Compatible with 16x2 and 20x4 LCD's
* Default I2C Address = 0X27
* Address selectable - Range 0x20 to 0x27



Fig 4.9 LCD i2c

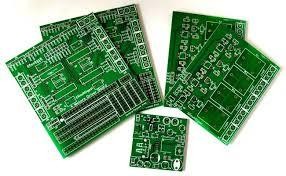
## **Printed Circuit Board (PCB)**

A PCB mechanically supports and electrically connects electronic components or electrical components using conductive tracks, pads and other features etched from one or more sheet layers of copper laminated onto and/or between sheet layers of a non-conductive substrate.

* + - **Types of Materials Used for PCB Fabrication**

There are three major types of materials used for fabrication of PCBs:

1. **FR-4:** This is the most commonly used material in PCBs. It is a glass reinforced epoxy laminate sheet. The epoxy used is flame retardant and water resistant. It provides good strength to weight ratios. The tensile strength offered by this material is very high.
2. **PTFE (Teflon):** PTFE is a kind of plastic material that does not provide any resistance, thus is used for high speed, high frequency applications. PTFE is extremely flexible, making it invaluable in applications with tight tolerances. It is also extremely lightweight, allowing it to be used across various industries. It is also flame resistant, exhibits high physical strength, provides temperature stability, and is versatile in application.
3. **Metal :** The traditional materials of copper, aluminum, iron, etc. are still used in PCBs. These materials allow the use of Surface Mount Technology (SMT) for the integration of components. They also provide mechanical durability. Thus, the product life of metal base PCBs is much longer.



##### Figure 4.10: PCB

* **SCHEMATIC**

PCBs are printed as per the requirements of the place it is going to be used, for this project we required a customized PCB specially designed fulfilling our requirements and for that we need to provide what the final PCB should look like, for this we have to prepare schematic diagram for our circuit first.

Beginning with the most important requirements can help streamline the design process by determining the constraints on the design. Sizing and placement of critical elements, including minimum and maximum tolerances, required components, electrical demands including impedance factors and power needs all combine to generate an initial set of constraints for the PCB design

It can help us to streamline the design process by determining the constraints on the design. Sizing and placement of critical elements, including minimum and maximum tolerances, required components, electrical demands including impedance factors and power needs all combine to generate an initial set of constraints for the PCB design and better working.

Diagram, schematic

Description automatically generated

##### Fig 4.11: Schematic Diagram

* **PCB LAYOUT:-**

Board layout techniques can include such strategies as devices embedded on inner layers of PCBs to reduce board size. This must be evaluated with manufacturers to verify that these capabilities can be met in actual fabrication. Designers are therefore incentivized to produce a design that will not only produce the desired functionality consistently, but will generate a layout that allows efficient manufacturing at the desired cost point. Engineers need to understand the manufacturing process to some level in order to comprehend how fabrication methodology will react to their designs. Multi-layer boards and double-sided laminates or double-sided component placement designs can make the layout all the more critical in designing for manufacturing.

An effective tool available to PCB designers prior to prototype creation is DFM

software. Such tools analyze the designer’s files and evaluate them for any issues or omissions related to fabrication. Combining the use of PCB design tools and DFM applications is the best solution to designing PCBs of the highest quality that will be functional and cost-effective to manufacture.

A picture containing text, computer, screenshot, monitor

Description automatically generated

##### **Fig 4.12: PCB Layout**

**5.Interfacing**

**5.1 ESP32 and DS3231 interfacing**

Graphical user interface

Description automatically generated

Fig 5.1 ESP32 and DS3231 interfacing(1)

A picture containing text, electronics, computer

Description automatically generated

Fig 5.2 ESp32 and DS3231 interfacing(2)

#include <Wire.h>

#include <DS3231.h>

DS3231 c;

RTCDateTime dt;

void setup()

{

Serial.begin(9600);

// Initialize DS3231

Serial.println("Initialize DS3231");

c.begin();

// Set sketch compiling time

c.setDateTime(\_DATE, \_\_TIME\_);

}

void loop()

{

dt = c.getDateTime();

// For leading zero look to DS3231\_dateformat example

Serial.print("Date: ");

Serial.print(dt.day); Serial.print("-");

Serial.print(dt.month); Serial.print("-");

Serial.print(dt.year); Serial.println(" ");

Serial.print("Time: ");

Serial.print(dt.hour); Serial.print(":");

Serial.print(dt.minute); Serial.print(":");

Serial.print(dt.second); Serial.println("");

delay(1000);

}

**5.2 . ESP32, PZEM-004T v3.0 and DS3231 interfacing:**

Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

Fig 5.4 Outputs

Fig 5.3 ESP32, PZEM-004T v3.0 and DS3231 interfacing

A picture containing light

Description automatically generated

Fig 5.5 Load Connected with PZEM

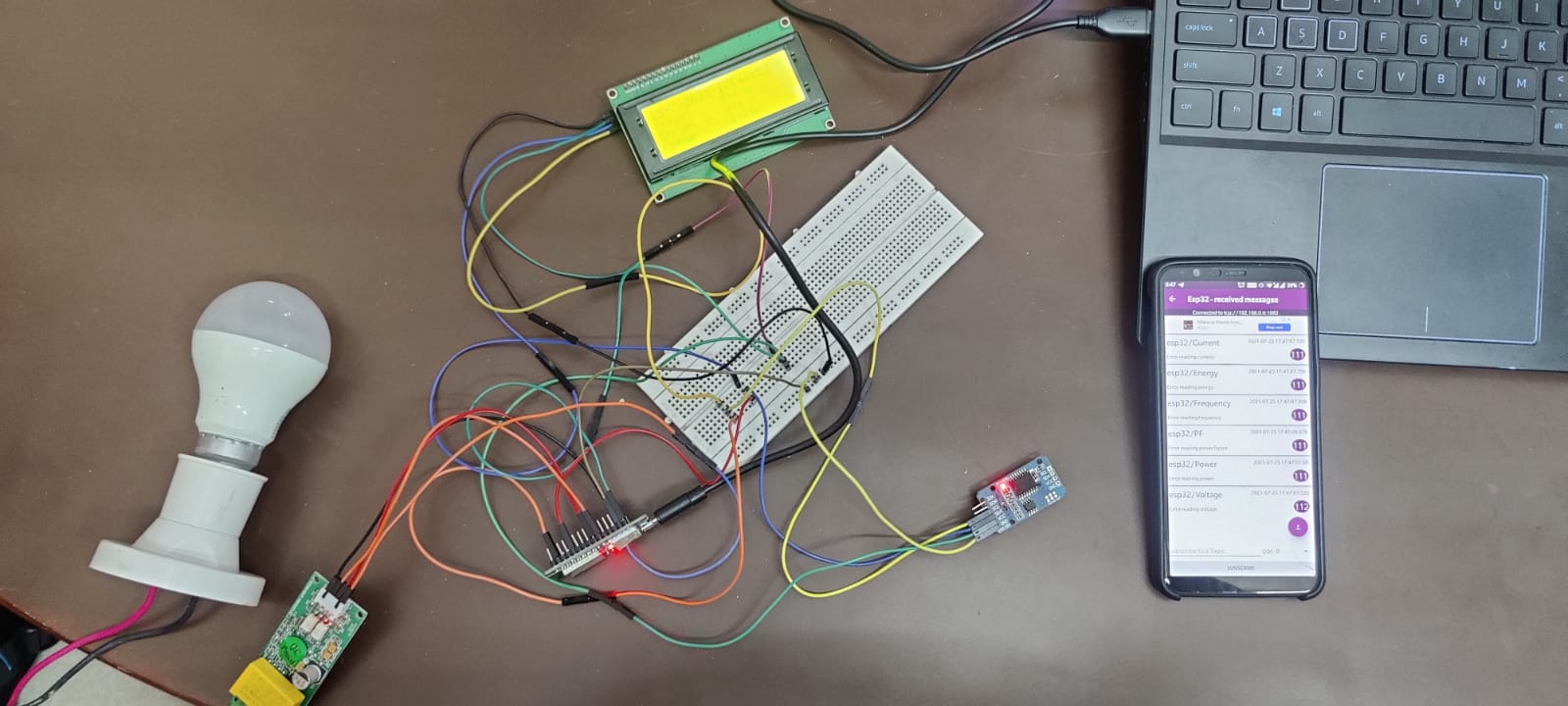


Fig 5.6 Interfacing Of LCD

#include <WiFi.h>

#include <PubSubClient.h>//callback //reconnect

#include <DS3231.h>

#include <PZEM004Tv30.h>

#include <LiquidCrystal\_I2C.h>

/\* change it with your ssid-password \*/

const char\* ssid = "Kharbanda net 2";

const char\* password = "sunilsushil@";

/\* this is the IP of PC/raspberry where you installed MQTT Server

on Wins use "ipconfig"

on Linux use "ifconfig" to get its IP address \*/

const char\* mqtt\_server = "192.168.0.6";

/\* topics \*/

#define TOPIC "esp32"

/\* create an instance of PubSubClient client \*/

DS3231 c;

RTCDateTime dt;

PZEM004Tv30 pzem(&Serial2);

LiquidCrystal\_I2C lcd(0x27, 20, 4);

WiFiClient espClient;

PubSubClient client(espClient);

long lastMsg = 0;

char msg[50];

int value = 0;

void receivedCallback(char\* topic, byte\* payload, unsigned int length) {

Serial.print("Message received: ");

Serial.println(topic);

Serial.print("payload: ");

for (int i = 0; i < length; i++) {

Serial.print((char)payload[i]);

}

Serial.println();

}

void mqttconnect() {

/\* Loop until reconnected \*/

while (!client.connected()) {

Serial.print("MQTT connecting ...");

/\* client ID \*/

String clientId = "ESP32Client";

/\* connect now \*/

if (client.connect(clientId.c\_str())) {

Serial.println("connected");

/\* subscribe topic with default QoS 0\*/

client.subscribe(TOPIC);

} else {

Serial.print("failed, status code =");

Serial.print(client.state());

Serial.println("try again in 5 seconds");

/\* Wait 5 seconds before retrying \*/

delay(5000);

}

}

}

void setup() {

Serial.begin(115200);

Serial2.end();

Serial2.begin(9600, SERIAL\_8N1, 16, 17);

// We start by connecting to a WiFi network

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

/\* configure the MQTT server with IPaddress and port \*/

client.setServer(mqtt\_server, 1883);

/\* this receivedCallback function will be invoked

when client received subscribed topic \*/

client.setCallback(receivedCallback);

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

/\* configure the MQTT server with IPaddress and port \*/

client.setServer(mqtt\_server, 1883);

/\* this receivedCallback function will be invoked

when client received subscribed topic \*/

client.setCallback(receivedCallback);

c.begin();

// Set sketch compiling time

c.setDateTime(\_DATE, \_\_TIME\_);

// initialize the LCD,

lcd.begin();

// Turn on the blacklight and print a message.

lcd.backlight();

lcd.clear();

lcd.setCursor (0,0); //

lcd.print("LCD2004 Test");

lcd.setCursor (0,1); //

lcd.print("Please Wait - 3");

lcd.setCursor (0,1); //

delay(1000);

lcd.print("Please Wait - 2");

delay(1000);

lcd.setCursor (0,1); //

lcd.print("Please Wait - 1");

delay(1000);

}

char\* toCharArray(String str){

return &str[0];

}

void loop() {

/\* if client was disconnected then try to reconnect again \*/

if (!client.connected()) {

mqttconnect();

}

/\* this function will listen for incomming

subscribed topic-process-invoke receivedCallback \*/

dt = c.getDateTime();

// For leading zero look to DS3231\_dateformat example

lcd.clear();// clearn previous values from screen

lcd.setCursor (0,0); //character zero, line 1

Serial.print("Date: ");

Serial.print(dt.day); Serial.print("-");

Serial.print(dt.month); Serial.print("-");

Serial.print(dt.year); Serial.println(" ");

lcd.print(dt.day); lcd.print("-");

lcd.print(dt.month); lcd.print("-");

lcd.print(dt.year); lcd.print(" ");

Serial.print("Time: ");

Serial.print(dt.hour); Serial.print(":");

Serial.print(dt.minute); Serial.print(":");

Serial.print(dt.second); Serial.println("");

lcd.print(dt.hour); lcd.print(":");

lcd.print(dt.minute); lcd.print(":");

lcd.print(dt.second); lcd.print("");

lcd.setCursor (0,1); //character 0, line 2

String svoltage;

float voltage = pzem.voltage();

svoltage=String(voltage);

if( !isnan(voltage) ){

Serial.print("Voltage: "); Serial.print(voltage); Serial.println("V");

client.publish("esp32/Voltage", toCharArray(svoltage));

lcd.print("V: "); lcd.print(voltage); lcd.print("V");

} else {

Serial.println("Error reading voltage");

client.publish("esp32/Voltage","Error reading voltage","false");

lcd.print("Error");

}

lcd.setCursor (10,1); //character 11, line 2

String scurrent;

float current = pzem.current();

scurrent=(String)current;

if( !isnan(current) ){

Serial.print("Current: "); Serial.print(current); Serial.println("A");

client.publish("esp32/Current", toCharArray(scurrent));

lcd.print("C: "); lcd.print(current); lcd.print("A");

} else {

Serial.println("Error reading current");

client.publish("esp32/Current","Error reading current");

lcd.print("Error");

}

lcd.setCursor (0,2); //character 0, line 3

String spower;

float power = pzem.power();

spower=(String)power;

if( !isnan(power) ){

Serial.print("Power: "); Serial.print(power); Serial.println("W");

client.publish("esp32/Power", toCharArray(spower));

lcd.print("P: "); lcd.print(power); lcd.print("W");

} else {

Serial.println("Error reading power");

client.publish("esp32/Power","Error reading power");

lcd.print("Error");

}

lcd.setCursor (10,2); //character 11, line 3

String senergy;

float energy = pzem.energy();

senergy=(String)energy;

if( !isnan(energy) ){

Serial.print("Energy: "); Serial.print(energy,3); Serial.println("kWh");

client.publish("esp32/Energy", toCharArray(senergy));

lcd.print("E: "); lcd.print(energy,3); lcd.print("kWh");

} else {

Serial.println("Error reading energy");

client.publish("esp32/Energy","Error reading energy");;

lcd.print("Error");

}

lcd.setCursor (0,3); //character 0, line 4

String sfrequency;

float frequency = pzem.frequency();

sfrequency=(String)frequency;

if( !isnan(frequency) ){

Serial.print("Frequency: "); Serial.print(frequency, 1); Serial.println("Hz");

client.publish("esp32/Frequency", toCharArray(sfrequency));

lcd.print("F: "); lcd.print(frequency, 1); lcd.print("Hz");

} else {

Serial.println("Error reading frequency");

client.publish("esp32/Frequency","Error reading frequency");

lcd.print("Error");

}

lcd.setCursor (10,3); //character 11, line 4

String spf;

float pf = pzem.pf();

spf=(String)pf;

if( !isnan(pf) ){

Serial.print("PF: "); Serial.println(pf);

client.publish("esp32/PF", toCharArray(spf));

lcd.print("PF: "); lcd.print(pf);

} else {

Serial.println("Error reading power factor");

client.publish("esp32/PF","Error reading power factor");

lcd.print("Error");

}

Serial.println();

delay(2000);

client.loop();

}

**6. Result**

****Graphical user interface, text, application

Description automatically generated

Fig 6.2 Output On LCD

Fig 6.1 Arduino Application Output

Graphical user interface, application

Description automatically generated

Fig 6.3 MQTT Client Android app Outputs

**Graphical user interface, text, application

Description automatically generated**

Fig 6.4 MQTT Explorer Desktop Application Outputs

**7. CONCLUSION**

A smart power monitoring and control system has been designed and developed towards the implementation of an intelligent building. This system monitors and controls the power consumption of home appliances remotely by using wireless network. And also protect the load from High voltages. The entire system is designed on an embedded platform which is easy to design and consume less power and provides at low cost with portable size. Thus, the continuous monitoring of the electrical appliances can be observed through a website as well as android app. Further, this work can be extended for power consumption of whole building and electricity bill can be determined. This project can be installed at the transformer to determine the illegal connections for households and by verifying the power in each transmission line, the load at the end of line from transformer can be regulated.

1. **BIBLOGRAPHY**

**WEB SEARCHES-**

* [www.google.com/Google](http://www.google.com/Google) search basics.html
* [www.sourceforge.net](http://www.sourceforge.net/)
* [www.developer.android.com](http://www.developer.android.com/)
* [www.tutorialspoint.com](http://www.tutorialspoint.com/)
* https://[www.embedded-computing.com](http://www.embedded-computing.com/)
* https://[www.engineersgarage.com](http://www.engineersgarage.com/)
* https://[www.theengineeringprojects.com](http://www.theengineeringprojects.com/)
* https://components101.com
* https://datasheets.maximintegrated.com
* https://[www.teachmemicro.com](http://www.teachmemicro.com/)
* https://en.wikipedia.org