

TRANSMISSION LINE MULTIPLE FAULT DETECTION AND INDICATION TO ELECTRICITY BOARD

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Engineering Degree in Electronics and Communication
Engineering

BY

**MEDAPATI THRINADH REDDY (38130703)
CHINTA VENKATA REDDY (38130043)**



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING SCHOOL OF ELECTRONICS**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY (DEEMED TO BE
UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119, TAMILNADU

MAY – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **MEDAPATI THRINADH REDDY (38130703), CHINTA VENKATA REDDY (38130043)** who carried out the project entitled **"TRANSMISSION LINE MULTIPLE FAULT DETECTION AND INDICATION TO ELECTRICITY BOARD"** under my supervision from December 2022 to May 2022

Internal Guide

Dr. P. Grace Kanmani., M.E., Ph.D.

Head of the Department

Dr. T. RAVI M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

WE MEDAPATI THRINADH REDDY (Reg no: 38130703) & CHINTA VENKATA REDDY (Reg no: 38130043) hereby declare that the Project Report entitled **“TRANSMISSION LINE MULTIPLE FAULT DETECTION AND INDICATION TO ELECTRICITY BOARD”** done by me under the guidance of **Dr. P. Grace Kanmani., M.E., Ph.D. (Internal)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Electronics and Communication Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

We convey my thanks to **Dr. N. NANDHITA, Ph.D.**, Dean, School of Electronics **Dr. T RAVI M.E., Ph.D.**, Head of the Department of Electronics and communication Engineering for providing me necessary support and details at the right time during the progressive reviews.

We would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. P. Grace Kanmani., M.E., Ph.D.** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

We wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Electronics and communication Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT:

Transmission line cable system is a common practice followed in urban areas. While a fault occurs due to many reasons in the cable, at a time of removing or repairing process, there is difficulty in locating the exact location of the fault.

The system proposed in this system is used to find out the exact location of the fault and display it to the dedicated website over internet using Wi-Fi module, giving prior information to the authorized person at other end.

TABLE OF CONTENTS

CHAPTER NO.	TOPICS	PAGE NO.
	ABSTRACT	v
	LIST OF TABLES	vi
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 LITERATURE SURVEY	1
	1.2 EXISTING SYSTEM	3
	1.3 PROPOSED SYSTEM	3
	1.4 BLOCK DIAGRAM	4
2	EMBEDDED SYSTEMS	5
	2.1 OVERVIEW OF EMBEDDED SYSTEMS	5
	2.2 BLOCK DIAGRAM OF AN EMBEDDED SYSTEM	6
	2.3 CHARACTERISTICS OF EMBEDDED SYSTEMS	7
	2.4 POWER SUPPLY	11
	2.5 TRANSFORMER	12
	2.6 RECTIFIER	12
	2.7 FILTERS	12
	2.8 REGISTERS	13
	2.9 REGULATORS	13
	2.10 FEATURES AND DESCRIPTION OF REGULATORS	13
3	INTERNET OF THINGS(IOT)	14
	3.1 IOT DEFINATION	15
	3.2 APPLICATION ABSTRACTION	17
	3.3 CONCLUSION	18

TABLE OF CONTENTS

CHAPTER NO.	TOPICS	PAGE NO.
4	REVIEW OF IOT-BASED MIDDLEWARE	19
	4.1 NODE MCU	19
	4.1.1. NODE MCU-INTRODUCTION	20
	4.2 RELAY	25
	4.3 ARDUINO IDE	26
	4.4 UBIDOTS EXPLORER	31
5	RESULT	33
	5.1 CONNECTIONS	33
	5.2 CLOUD PLATFORM	33
	5.3 ALERTS SCREENSHOTS	34
6	CONCLUSION	36
7	REFERENCES	37

LIST OF FIGURES

CHAPTER	TOPICS	PAGE NO.
----------------	---------------	-----------------

NO.		
1	BLOCK DIAGRAM	4
2	2.2 BLOCK DIAGRAM OF A TYPICAL EMBEDDED SYSTEMS	7
	2.3 EMBEDDED SYSTEMS HAVING HETEROGENEOUS	9
	2.4 GENERAL BLOCK OF POWER SUPPLY	12
3	3.1 LAYERED ARCHITECTURE OF THE INTERNET OF THINGS	16
4	4.1.1. NODE MCU (version 1)	20
	4.1.2. NODE MCU (version 2)	21
	4.2 RELAY	25
	4.3 ARDUINO UNO-R3 AND A TO B CABLE	27
	4.4 UNIBOT DATA HIERARCHY	31
5	5.1 CONNECTIONS	33
	5.2 CLOUD PLATFORM	33
	5.3 ALERTS SCREENSHOT'S	34

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION:

Power supply networks are growing continuously and their reliability getting more important than ever. The complexity of the whole network comprises numerous components that can fail and interrupt the power supply for end user. For most of the worldwide operated low voltage and medium voltage distribution lines, transmission lines cables have been used for many decades.

Transmission line high voltage cables are used more and more because they are not influenced by weather conditions, heavy rain, storm, snow and pollution.

Even though the Cable manufacturing technology is improving steadily; there are still influences which may cause cable to fail during test and operation. A cable in good condition and installed correctly can last a lifetime of about 30 years. However cables can be easily damaged by incorrect installation or poorly executed jointing, while subsequent third party damage by civil works such as trenching or curb edging.

1.2 LITERATURE SURVEY:

Title: IOT BASED DISPLAY OF UNDERGROUND CABLE FAULT DISTANCE OVER INTERNET SYSTEM.

Author: Mr. N. Sampathraja, Dr. L. Ashok Kumar, Ms. V. Kirubalakshmi and Ms. C. Muthumaniyarasi, Mr. K. Vishnu Murthy

Content: Underground cables are prone to a wide variety of faults due to underground conditions, wear and tear, rodents etc. Diagnosing fault source is difficult and entire cable should be taken out from the ground to check and fix faults. The project work is intended to detect the location of fault in underground cable lines from the base station in km using a PIC16F877A controller. To locate a fault in the cable, the cable must be tested for faults. This prototype uses the simple concept of Ohms law.

Title: IOT BASED UNDERGROUND CABLE FAULT DETECTOR

Author: Kajal Bagade 1, Gita Bhardwaj 2, Shweta Wasnik 3, Dheeraj Kawade 4, Payal Mool5 , Rupali Mohatkar

Content: This paper is to determine the distance of underground cable fault from the base station in kilometers and displayed over the internet. Underground cable system is a common followed in major areas in Metro cities. While a fault occurs for some reason, at that time the fixing process related to that particular cable is difficult due to exact unknown location of the fault in the cable. This IOT Technology is used to find out the exact location of the fault and to send data in graphical format to our website using an IOT module at the same time it display on the LCD screen

Title: UNDER GROUND CABLE FAULT DETECTION OVER IOT

Author: Durgesh Pathak, Mithil Rathod, Sandeep Vishwakarma, Harindar Maurya, Mahalaxmi Palinje

Content: Underground cable system is a common practice followed in urban areas. While a fault occurs due to many reasons in the cable, at a time of removing or repairing process, there is difficulty in locating the exact location of the fault. The system proposed in this paper is used to find out the exact location of the fault and display it to the dedicated website over internet using wi-fi module, giving prior information to the authorized person at other end.

Title: IOT BASED UNDERGROUND CABLE LINE FAULT DETECTION

Author: Ansa Sebastian1, Anu Jacob2, Angel Wilson3, Ann Rija Paul J 4

Content: IOT based underground cable line fault detection system is helpful for find out faults and its location in very easy manner .Underground cables have been widely used with the development of power system grid .Underground cables are prone to a wide variety of faults due to underground conditions, wear and tear, rodents .Detecting fault source is difficult because entire line is to be dug in order to check fault at cableline. The repairmen know exactly which part

has fault and only that area is to be dug to detect the fault source. Thus it saves a lot of time, money and allows to service underground cablelines faster.

1.3 EXISTING SYSTEM:

A variety of technologies and tests are currently available to evaluate underground cables but there is often little relation between the diagnostic results and the actual detracton. The failures of underground power distribution cables represents a serious threat to the reliability of power infrastructure. Replacement must be done selectively since cable replacement is expensive, being estimated at no less than hundred thousand dollars per kilometre of cable in area.

1.4 PROPOSED SYSTEM:

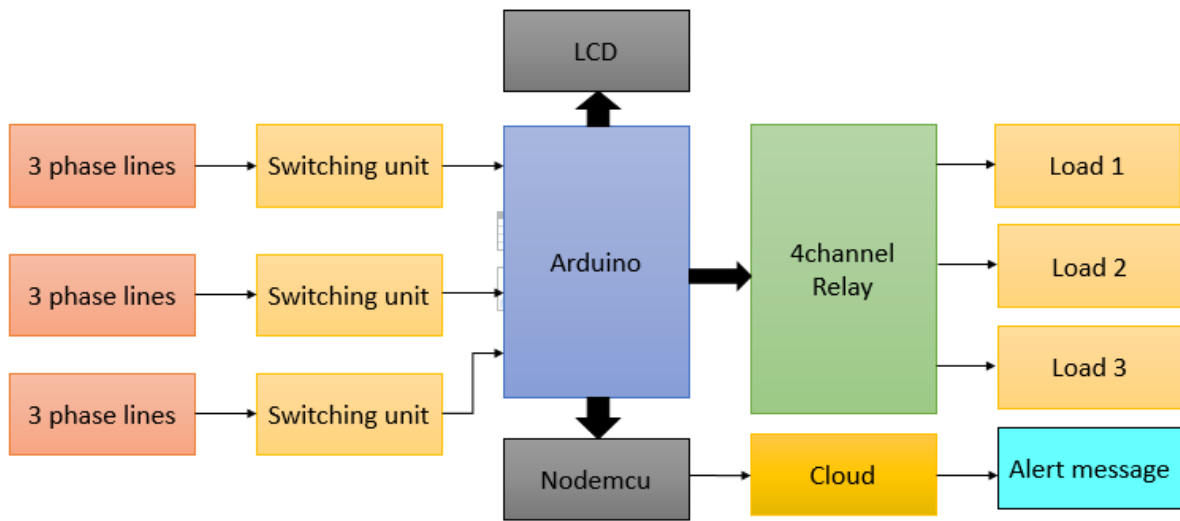
This system represents that to compute the distance of the transmission cable from fault location to the base station using Arduino. In this system, a smart technology based fault detection and location system was used to adequately and accurately indicate and locate the exact spot where fault had occurred.

This will ensure a shorter response time for technical crew to rectify these faults and thus help save transformers and other electrical equipment from damage and disasters. For Transmission line cables fault detection, we used switching unit at each and every 10km of cable line. If any fault occurs we can easily locate the cable fault.

It's a difficult task to identify the Short Circuit faults in Transmission line cables in an appropriate phase. By using this system we can find out exact fault location in particular phase of wire.

Once faults occur in the cable, with the help of micro controller and the display unit displays the exact fault location that occurs in appropriate phase of the cable to a dedicated website with the help of IOT. If the fault occurs then the respective loads will turn off, the reason is for protecting it from damage.

1.5 BLOCK DIAGRAM:



1.6 REQUIRED COMPONENTS:

Hardware required:

- Switching units
- Arduino nano
- Nodemcu
- Relay

Software required:

- Arduino IDE
- Embedded C
- Ubidots explorer

CHAPTER 2

EMBEDDED SYSTEMS

2.1 OVERVIEW OF EMBEDDED SYSTEMS

An embedded system is a special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is usually embedded as part of a complete device including hardware and mechanical parts. In contrast, a general-purpose computer, such as a personal computer, can do many different tasks depending on programming. Embedded systems have become very important today as they control many of the common devices we use.

Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance.

Physically, embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure.

In general, "embedded system" is not an exactly defined term, as many systems have some element of programmability. For example, Handheld computers share some elements with embedded systems — such as the operating systems and microprocessors which power them — but are not truly embedded systems, because they allow different applications to be loaded and peripherals to be connected.

Embedded systems provide several functions

- Control the environment; embedded systems generate and transmit commands for actuators.
- There are many categories of embedded systems, from communication devices to home appliances to control systems. Examples include;
- Communication devices
eg: modems, cellular phones
- Home Appliances
eg: CD player, VCR, microwave oven
- Control Systems
eg: Automobile anti-lock braking systems, robotics, satellite control

2.2 BLOCK DIAGRAM OF AN EMBEDDED SYSTEM:

An embedded system usually contains an embedded processor. Many appliances that have a digital interface -- microwaves, VCRs, cars -- utilize embedded systems. Some embedded systems include an operating system. Others are very specialized resulting in the entire logic being implemented as a single program. A typical embedded system is shown in Fig 2.2

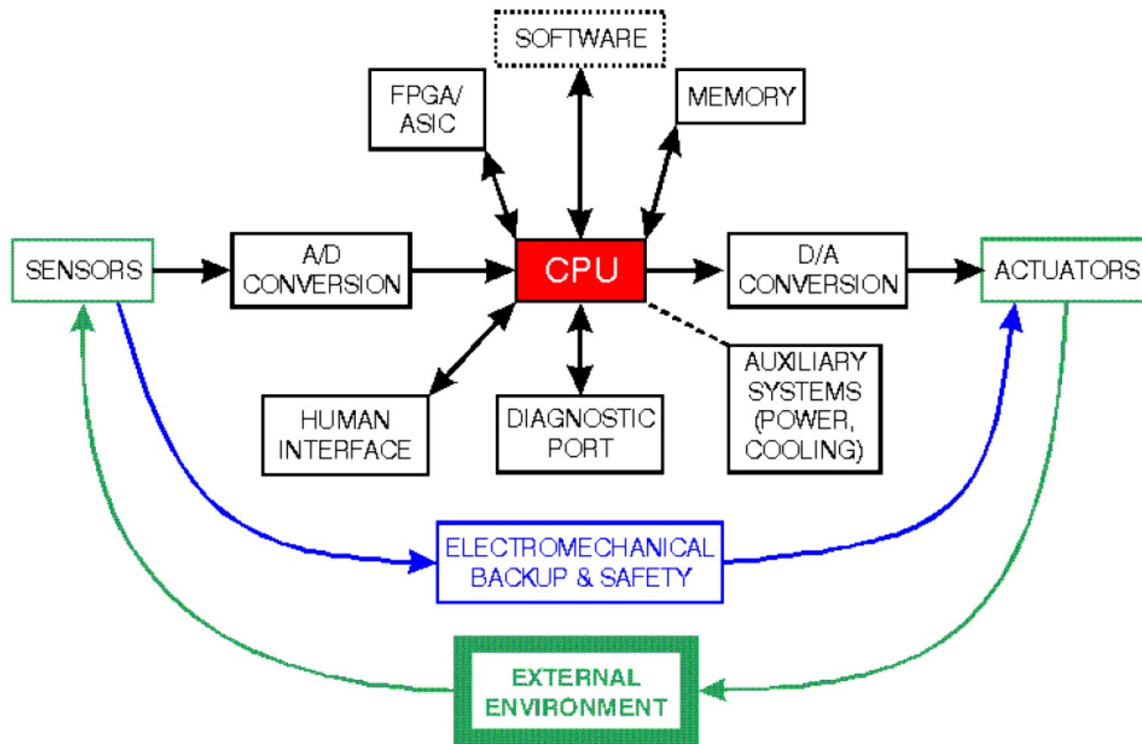


FIG 2.2 BLOCK DIAGRAM OF A TYPICAL EMBEDDED SYSTEM

2.3 CHARACTERISTICS OF EMBEDDED SYSTEMS

Embedded systems are characterized by a unique set of characteristics. Each of these characteristics imposed a specific set of design constraints on embedded systems designers. The challenge to designing embedded systems is to conform to the specific set of constraints for the application.

APPLICATION SPECIFIC SYSTEMS:

Embedded systems are not general-purpose computers. Embedded system designs are optimized for a specific application. Many of the job characteristics are known before the hardware is designed. This allows the designer to focus on the specific design constraints of a well-defined application. As such, there is limited user reprogram ability. Some embedded systems, however, require the flexibility of reprogram ability. Programmable DSPs are common for such applications.

REACTIVE SYSTEMS

As mentioned earlier, a typical embedded systems model responds to the environment via sensors and control the environment using actuators. This requires embedded systems to run at the speed of the environment. This characteristic of embedded system is called “reactive”. Reactive computation means that the system (primarily the software component) executes in response to external events. External events can be either periodic or aperiodic. Periodic events make it easier to schedule processing to guarantee performance. Aperiodic events are harder to schedule. The maximum event arrival rate must be estimated in order to accommodate worst case situations. Most embedded systems have a significant reactive component. One of the biggest challenges for embedded system designers is performing an accurate worst case design analysis on systems with statistical performance characteristics (e.g., cache memory on a DSP or other embedded processor). Real time system operation means that the correctness of a computation depends, in part, on the time at which it is delivered. Systems with this requirement must often design to worst case performance. But accurately predicting the worst case may be difficult on complicated architectures. This often leads to overly pessimistic estimates erring on the side of caution. Many embedded systems have a significant requirement for real time operation in order to meet external I/O and control stability requirements. Many real-time systems are also reactive systems.

DISTRIBUTED SYSTEMS

A common characteristic of an embedded system is one that consists of communicating processes executing on several CPUs or ASICs which are connected by communication links. The reason for this is economy. Economical 4 8-bit microcontrollers may be cheaper than a 32-bit processors. Even after adding the cost of the communication links, this approach may be preferable. In this approach, multiple processors are usually required to handle multiple time-critical tasks. Devices under control of embedded systems may also be physically distributed.

HETEROGENEOUS ARCHITECTURES

Embedded systems often are composed of heterogeneous architectures (Fig 2.2). They may contain different processors in the same system solution. They may also be mixed signal systems. The combination of I/O interfaces, local and remote memories, and sensors and actuators makes embedded system design truly unique. Embedded systems also have tight design constraints, and heterogeneity provides better design flexibility.

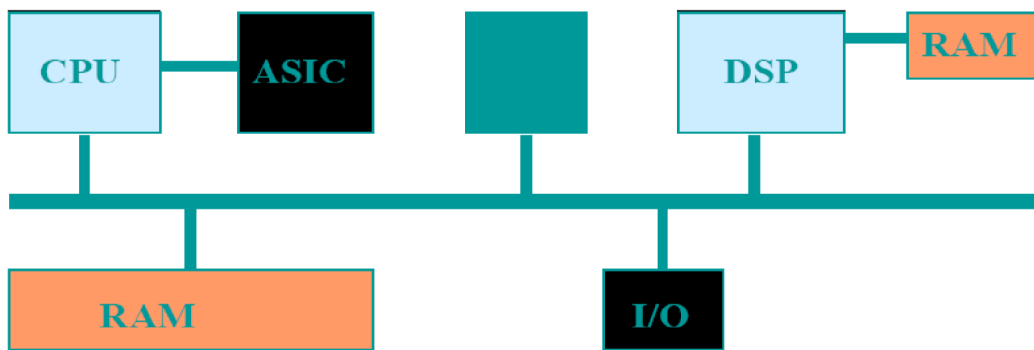


FIG 2.3 EMBEDDED SYSTEMS HAVING HETEROGENEOUS ARCHITECTURES

HARSH ENVIRONMENT

Many embedded systems do not operate in a controlled environment. Excessive heat is often a problem, especially in applications involving combustion (e.g., many transportation applications). Additional problems can be caused for embedded computing by a need for protection from vibration, shock, lightning, power supply fluctuations, water, corrosion, fire, and general physical abuse.

SYSTEM SAFETY AND RELIABILITY

As embedded system complexity and computing power continue to grow, they are starting to control more and more of the safety aspects of the overall system. These safety measures may be in the form of software as well as

hardware control. Mechanical safety backups are normally activated when the computer system loses control in order to safely shut down system operation. Software safety and reliability is a bigger issue. Software doesn't normally "break" in the sense of hardware. However software may be so complex that a set of unexpected circumstances can cause software failures leading to unsafe situations. Discussion of this topic is outside the scope of this book, but the challenges for embedded designers include designing reliable software and building cheap, available systems using unreliable components. The main challenge for embedded system designers is to obtain low-cost reliability with minimal redundancy.

CONTROL OF PHYSICAL SYSTEMS

One of the main reasons for embedding a computer is to interact with the environment. This is often done by monitoring and controlling external machinery. Embedded computers transform the analog signals from sensors into digital form for processing. Outputs must be transformed back to analog signal levels. When controlling physical equipment, large current loads may need to be switched in order to operate motors and other actuators. To meet these needs, embedded systems may need large computer circuit boards with many non-digital components. Embedded system designers must carefully balance system tradeoffs among analog components, power, mechanical, network, and digital hardware with corresponding software.

SMALL AND LOW WEIGHT

Many embedded computers are physically located within some larger system. The form factor for the embedded system may be dictated by aesthetics. For example, the form factor for a missile may have to fit inside the nose of the missile. One of the challenges for embedded systems designers is to develop non-rectangular geometries for certain solutions. Weight can also be a critical constraint. Embedded automobile control systems, for example, must be light weight for fuel economy. Portable CD players must be light weight for portability purposes.

COST SENSITIVITY

Cost is an issue in most systems, but the sensitivity to cost changes can vary dramatically in embedded systems. This is mainly due to the effect of computer costs have on profitability and is more a function of the proportion of cost changes compared to the total system cost.

POWER MANAGEMENT

Embedded systems have strict constraints on power. Given the portability requirements of many embedded systems, the need to conserve power is important to maintain battery life as long as possible. Minimization of heat production is another obvious concern for embedded systems.

2.4 POWER SUPPLY

All electronic circuits works only in low DC voltage, so we need a power supply unit to provide the appropriate voltage supply for their proper functioning .This unit consists of transformer, rectifier, filter & regulator. AC voltage of typically 230volts rms is connected to a transformer voltage down to the level to the desired ac voltage. A diode rectifier that provides the full wave rectified voltage that is initially filtered by a simple capacitor filter to produce a dc voltage(Fig 1.3). This resulting dc voltage usually has some ripple or ac voltage variation. A regulator circuit can use this dc input to provide dc voltage that not only has much less ripple voltage but also remains the same dc value even the dc voltage varies some what, or the load connected to the output dc voltages changes.

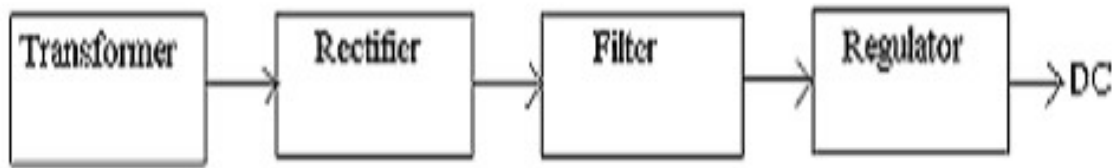


Fig 2.4 General Block Of Power Supply Unit

2.5 TRANSFORMER:

A transformer is a static piece of which electric power in one circuit is transformed into electric power of same frequency in another circuit. It can raise or lower the voltage in the circuit, but with a corresponding decrease or increase in current. It works with the principle of mutual induction. In our project we are using a step down transformer to providing a necessary supply for the electronic circuits. Here we step down a 230volts ac into 12volts ac.

2.6 RECTIFIER:

A dc level obtained from a sinusoidal input can be improved 100% using a process called full wave rectification. Here in our project for full wave rectification we use bridge rectifier. From the basic bridge configuration we see that two diodes(say D2 & D3) are conducting while the other two diodes (D1 & D4) are in off state during the period $t = 0$ to $T/2$. Accordingly for the negative cycle of the input the conducting diodes are D1 & D4 .Thus the polarity across the load is the same.

In the bridge rectifier the diodes may be of variable types like 1N4001, 1N4003, 1N4004, 1N4005, 1N4007 etc.... can be used. But here we use 1N4007, because it can withstand up to 1000v.

2.7 FILTERS:

In order to obtain a dc voltage of 0 Hz, we have to use a low pass filter. So that a capacitive filter circuit is used where a capacitor is connected at the rectifier output & a dc is obtained across it. The filtered waveform is essentially a dc voltage with negligible ripples & it is ultimately fed to the load.

2.8 REGISTERS:

The controller IC has two 8 bit registers, an instruction register (IR) and a data register (DR). The IR stores the instruction codes and address information for display data RAM (DD RAM) and character generator RAM (CG RAM). The IR can be written, but not read by the MPU.

The DR temporally stores data to be written to /read from the DD RAM or CG RAM. The data written to DR by the MPU, is automatically written to the DD RAM or CG RAM as an internal operation.

The dot-matrix liquid crystal display controller and driver LSI displays alphanumeric, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver

2.9 REGULATORS:

The output voltage from the capacitor is more filtered & finally regulated. The voltage regulator is a device, which maintains the output voltage constant irrespective of the change in supply variations, load variations & temperature changes. Here we use fixed voltage regulator namely LM7805. The IC LM7805 is a +5v regulator which is used for microcontroller.

2.10 FEATURES AND DESCRIPTION OF REGULATORS

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection

CHAPTER 3

INTERNET OF THINGS(IOT)

BACKGROUND

The Future Internet goal is to provide an infrastructure to have an immediate access to information about the physical world and its objects. Physical objects can be applicable to different application domains, such as e-health, warehouse management, etc. Each application domain may have different types of physical devices. Each physical device can have its own specifications, which is required to use in order to interact with it. To achieve the future Internet goal, a layered vision is required that can facilitate data access. Internet of Things (IoT) is a vision that aims to integrate the virtual world of information to the real world of devices through a layered architecture.

The term „Internet of Things“ consists of two words, namely Internet and Things. Internet refers to the global network infrastructure with scalable, configurable capabilities based on interoperable and standard communication protocols. Things are physical objects or devices, or virtual objects, devices or information, which have identities, physical attributes and virtual personalities, and use intelligent interfaces . For instance, a virtual object can represent an abstract unit of sensor nodes that contains metadata to identify and discover its corresponding sensor nodes. Therefore, IoT refers to the things that can provide information from the physical environment through the Internet.

Middleware is as an interface between the hardware layer and the application layer, which is responsible for interacting with devices and information management . The role of a middleware is to present a unified programming model to interact with devices. A middleware is in charge of masking the heterogeneity and distribution problems that we face when interacting with devices .

3.1. IOT DEFINITION

In this section, we explain some of the IoT definitions. Also, we explain the layered architecture for IoT.

Internet of Things (IoT) has increasingly gained attention in industry to interact with different types of devices. IoT can have influence on industry and society by integrating physical devices into information networks. IoT impacts can be on different perspectives, namely for private and business users. From the perspective of a private user, IoT has effect on both working and personal fields, such as smart homes and offices, e-health and assisted living.

Internet of Things-Architecture (IoT-A) defines it as “The idea of a globally interconnected continuum of devices, objects and things in general emerged with the RFID technology, and this concept has considerably been extended to the current vision that envisages a plethora of heterogeneous objects interacting with the physical environment.”

IoT has a layered architecture designed to answer the demands of various industries, enterprises and society. Fig. 3.1 shows a generic layered architecture for IoT that consist of five layers, which are discussed, in the following:

EDGE TECHNOLOGY LAYER

This is a hardware layer that consists of embedded systems, RFID tags, sensor networks and all of the other sensors in different forms. This hardware layer can perform several functions, such as collecting information from a system or an environment, processing information and supporting communication.

ACCESS GATEWAY LAYER

This layer is concerned with data handling, and is responsible for

publishing and subscribing the services that are provided by the *Things*, message routing, and hovelling the communication between platforms.

MIDDLEWARE LAYER

This layer has some critical functionalities, such as aggregating and filtering the received data from the hardware devices, performing information discovery and providing access control to the devices for applications.

APPLICATION LAYER

This layer is responsible for delivering various application services. These services are provided through the middleware layer to different applications and users in IoT-based systems. The application services can be used in different industries such as, logistics, retail, healthcare, etc.

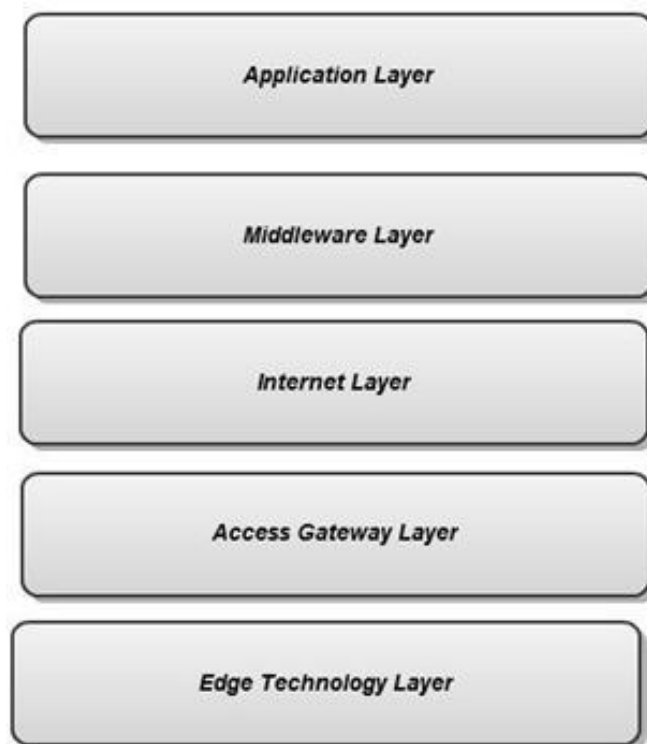


Figure 3.1 Layered architecture of the Internet of Things (IoT).

Context characterizes the situation of an entity, which can be a place, a person or an object that is relevant to the user, applications and their interactions .The CCM functional component is responsible to support context-aware computation that is a computational style that take to account the context of the entities that interact with the system. A middleware for IoT-based systems must be context-aware to work in smart environments . Smart environments refer to a *physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network* . Context-awareness includes two functionalities:

- 1) Context detection, which consists of collecting data from resources, and selecting the information that can have an impact on the computation.
- 2) Context processing, to use the gathered information to perform a task or make a decision.

3.2 APPLICATION ABSTRACTION

This functional component provides an interface for both high-level applications and end-users to interact with devices. For instance, this interface can be a RESTful interface or can be implemented with some query-based language.

A. Malatras. et. al. propose a SOA-based middleware to perform data management and data monitoring services . This middleware uses a RESTful interface to facilitate the interaction of high-level applications with sensors, which can communicate with Wireless Sensor Network (WSN) through the HTTP operations:

- (1) GET to issue a query on an existing resource,
- (2) DELETE to remove an existing resource,
- (3) POST to create a new resource, and
- (4) PUT to update an existing resource. For instance, a client application

gets the result of a domain task by sending a GET request through the URL: „http://{hostname}/REST/ {version}/DomainTaskResult/id’. In this URI that reference to an „id” leads to a unique result.

In the TinyDB middleware [19], an end-user can use a query language to interact with devices. For instance, a user can send the following query to get the id of a sensor(*nodeid*) nearby and temperatures that are sensed by this sensor during the past 10 seconds before executing the query:

```
SELECT nodeid, temp  
FROM sensors
```

```
SAMPLE PERIOD 1s FOR 10s
```

3.3 CONCLUSION

In this chapter, we defined IoT and identified common IoT layers. Furthermore, we discussed a reference middleware architecture for IoT-based systems. This architecture has been proposed by Bandyopadhyay, S. et. al based on a study on the existing middleware frameworks for IoT-based systems .

CHAPTER 4

REVIEW OF IOT-BASED MIDDLEWARE

The term Internet of Things (IoT) refers to a wide set of applications and research areas such as, distributed computing and knowledge management. Much of the IoT research we found in the literature is related to the field of pervasive computing. World Wide Web Consortium defines pervasive computing as a vision about our future computing life style. In this vision, computer systems will be seamlessly integrated into our everyday life, anticipating our needs and providing relevant services and information for us in an anytime-and-any-where fashion. Pervasive Computing is all about making the human's life easier by exploiting the available computing technologies. This motivates scientists to develop new technologies for the everyday people's lives.

4.1 NODE MCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The term "NodeMCU" by default refers to the firmware rather than the development kits. The firmware uses the Lua scripting language. It is based on the eLua project, and built on the Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as lua-cjson and SPIFFS

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems began production of the ESP8266. The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications (see related projects). NodeMCU started on 13 Oct 2014, when Hong committed the first file of nodemcu-firmware to GitHub. Two months later, the project expanded to include an open-hardware platform when developer Huang R committed the gerber file of an ESP8266 board, named devkit v0.9. Later that month, Tuan PM ported MQTT client library from Contiki to the

ESP8266 SoC platform, and committed to NodeMCU project, then NodeMCU was able to support the MQTT IoT protocol, using Lua to access the MQTT broker. Another important update was made on 30 Jan 2015, when Devsaurus ported the u8glib to NodeMCU project, enabling NodeMCU to easily drive LCD, Screen, OLED, even VGA displays.

In summer 2015 the creators abandoned the firmware project and a group of independent contributors took over. By summer 2016 the NodeMCU included more than 40 different modules. Due to resource constraints users need to select the modules relevant for their project and build a firmware tailored to their needs.

4.1.1 INTRODUCTION

NodeMCU is an open source LUA based firmware developed for ESP8266 wifi chip. By exploring functionality with ESP8266 chip, NodeMCU firmware comes with ESP8266 Development board/kit i.e. NodeMCU Development board.

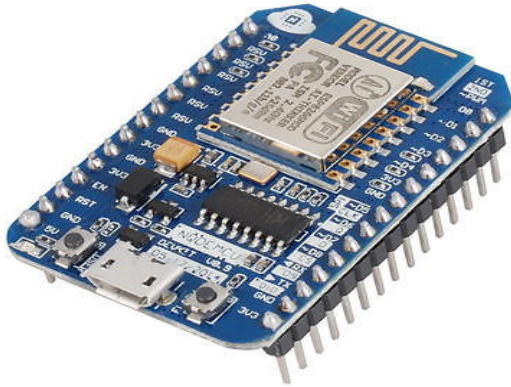


FIG 4.1.1 NodeMCU Development Board/kit v0.9 (Version1)

Since NodeMCU is opensource platform, their hardware design is open for edit/modify/build.

NodeMCU Dev Kit/board consist of ESP8266 wifi enabled chip. The **ESP8266** is a low-cost Wi-Fi chip developed by Espressif Systems with TCP/IP protocol. For more information about ESP8266, you can refer ESP8266 WiFi Module.

There is Version2 (V2) available for NodeMCU Dev Kit i.e. **NodeMCU Development Board v1.0 (Version2)**, which usually comes in black colored PCB.

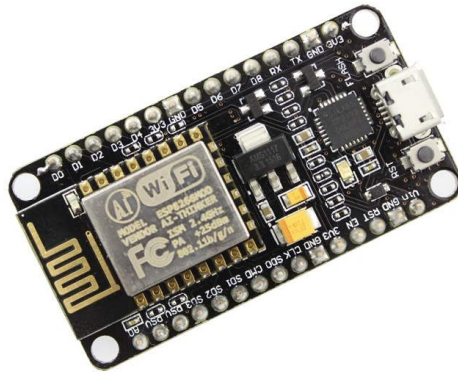


FIG 4.1.2 NodeMCU Development Board/kit v1.0 (Version2)

How to reset ATMEGA BOARD?

There are two ways to reset the board i.e. electronically or programmatically.

In order to reset the board electronically, you need to connect the reset pin of the board with any of digital pins on the controller. Don't forget to add 1K or 2K Ohm resistor while setting up this connection. Now, use the digital pin as an output and keep it HIGH before the reset. Once the reset is required, set this digital pin to LOW. This method is very useful because using it sends a hardware reset signal to the controller once the digital pin is set to LOW. You can use the following program to reset the controller electronically. `int Reset = 4;`

```
void setup () {  
  digitalWrite (Reset, HIGH);  
  delay (200);  
  pinMode (Reset, OUTPUT);  
  Serial.begin (9600);  
  Serial.println("How to Reset Arduino Programmatically");  
  Serial.println("www.TheEngineeringProjects.com");  
}
```

```
    Delay (200);  
}  
void loop ()  
{  
    Serial.println("A");  
    Delay (1000);  
    Serial.println("B");  
    Delay (1000);  
    Serial.println("Now we are Resetting Arduino Programmatically");  
    Serial.println();  
    Delay (1000);  
    digitalWrite (Reset, LOW);  
    Serial.println("Arduino will never reach there.");  
  
}
```

Once you upload the program, open your Arduino Serial Monitor that shows output as follows.



Another method we can use to reset the board is by software only without using any hardware pin. Nano board comes with a built-in function known as `resetFunc()`. The board will reset automatically as we define this function and then call it. Without using any hardware pin you can upload the following program to reset the board programmatically.

```
void(* resetFunc) (void) = 0;
```

```
void setup() {  
  Serial.begin(9600);  
  Serial.println("How to Reset Arduino Programmatically");  
  Serial.println("www.TheEngineeringProjects.com");  
  delay(200);  
}
```

```
void loop()
```

```
{  
  Serial.println("A");  
  delay(1000);  
  Serial.println("B");  
  delay(1000);  
  Serial.println("Now we are Resetting Arduino Programmatically");  
  Serial.println();  
  delay(1000);  
  resetFunc();  
  Serial.println("Arrduino will never reach there.");  
  
}
```

APPLICATIONS

Arduino Nano is a very useful device that comes with a wide range of applications and covers less space as compared to other Arduino board. Breadboard friendly nature makes it stand out from other board. Following are the main applications of the board.

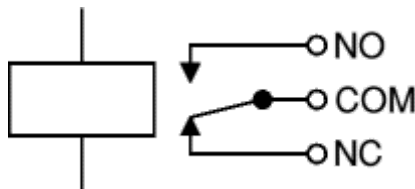
- Arduino Metal Detector
- Real-Time Face Detection
- Medical Instruments
- Industrial Automation
- Android Applications
- GSM Based Projects
- Embedded Systems
- Automation and Robotics
- Home Automation and Defense Systems
- Virtual Reality Applications

4.2 RELAY:

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches. Relays allow one circuit to switch a second circuit which can be completely separate from the first. For example a low voltage battery circuit can use a relay to switch a 230V AC mains circuit.



Relays are usually SPDT or DPDT but they can have many more sets of switch contacts, for example relays with 4 sets of changeover contacts are readily available.



The relay's switch connections are usually labeled COM, NC and NO:

- **COM** = Common, always connect to this, it is the moving part of the switch.
- **NC** = Normally Closed, COM is connected to this when the relay coil is **off**.
- **NO** = Normally Open, COM is connected to this when the relay coil is **on**.

4.3 ARDUINO IDE(APPLICATION):

INTRODUCTION

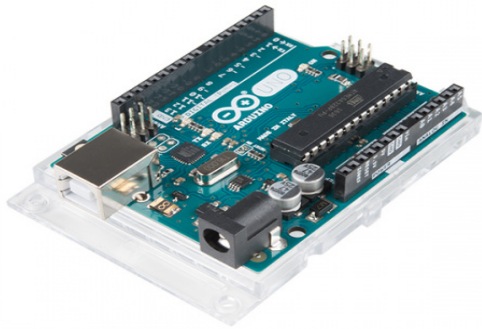
This tutorial will walk you through downloading, installing, and testing the Arduino software (also known as the Arduino IDE - short for Integrated Development Environment). Before you jump to the page for your operating system, make sure you've got all the right equipment.



REQUIRED MATERIALS

To follow along with this tutorial, you will need the following materials. You may not need everything though depending on what you have. Add it to your cart, read through the guide, and adjust the cart as necessary.

- A computer (Windows, Mac, or Linux)
- An Arduino-compatible microcontroller (anything from this guide should work)
- A USB A-to-B cable, or another appropriate way to connect your Arduino-compatible microcontroller to your computer (check out this USB buying guide if you're not sure which cable to get).



Arduino Uno - R3

USB Cable A to B - 6 Foot

Downloading the Arduino IDE

You can download the Arduino IDE from their website. They have installation instructions, but we will also go over the installation process as well. The installation procedure is fairly straightforward, but it does vary by OS. Here are some tips to help you along.

WINDOWS

This page will show you how to install and test the Arduino software with a Windows operating system (Windows 10, Windows 7, Vista, and XP).

Go to the [Arduino download page](#) and download the latest version of the Arduino software for Windows if you have not already.

INSTALLER

The Windows version of Arduino is offered in two options: an installer or a zip file. The **installer** is the easier of the two options, just download that, and run the executable file to begin the installation.

When you're prompted to install a driver during installation, select "Install". This will install drivers for Arduino specific boards (like the Uno, Nano, etc.) that you may use in the future.

ZIP

If you choose to download the **zip file** version of Arduino, you'll need to extract the files yourself. Don't forget which folder you extract the files into! You will need to run the executable Arduino file in the folder to start the Arduino IDE.

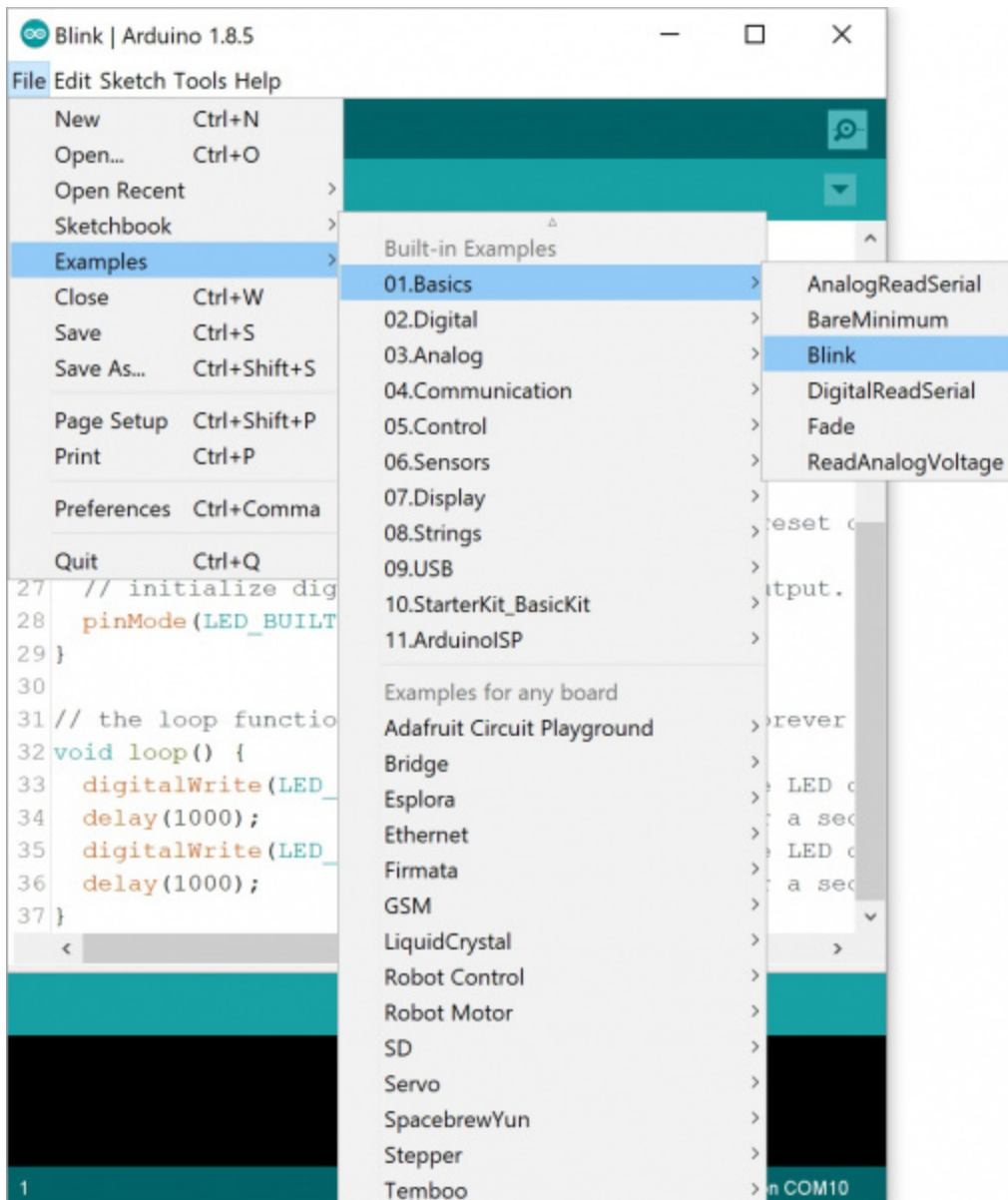
When the download is finished, un-zip it and open up the Arduino folder to confirm that yes, there are indeed some files and sub-folders inside. The file structure is important so don't be moving any files around unless you really know what you're doing.

CONNECTING YOUR ARDUINO

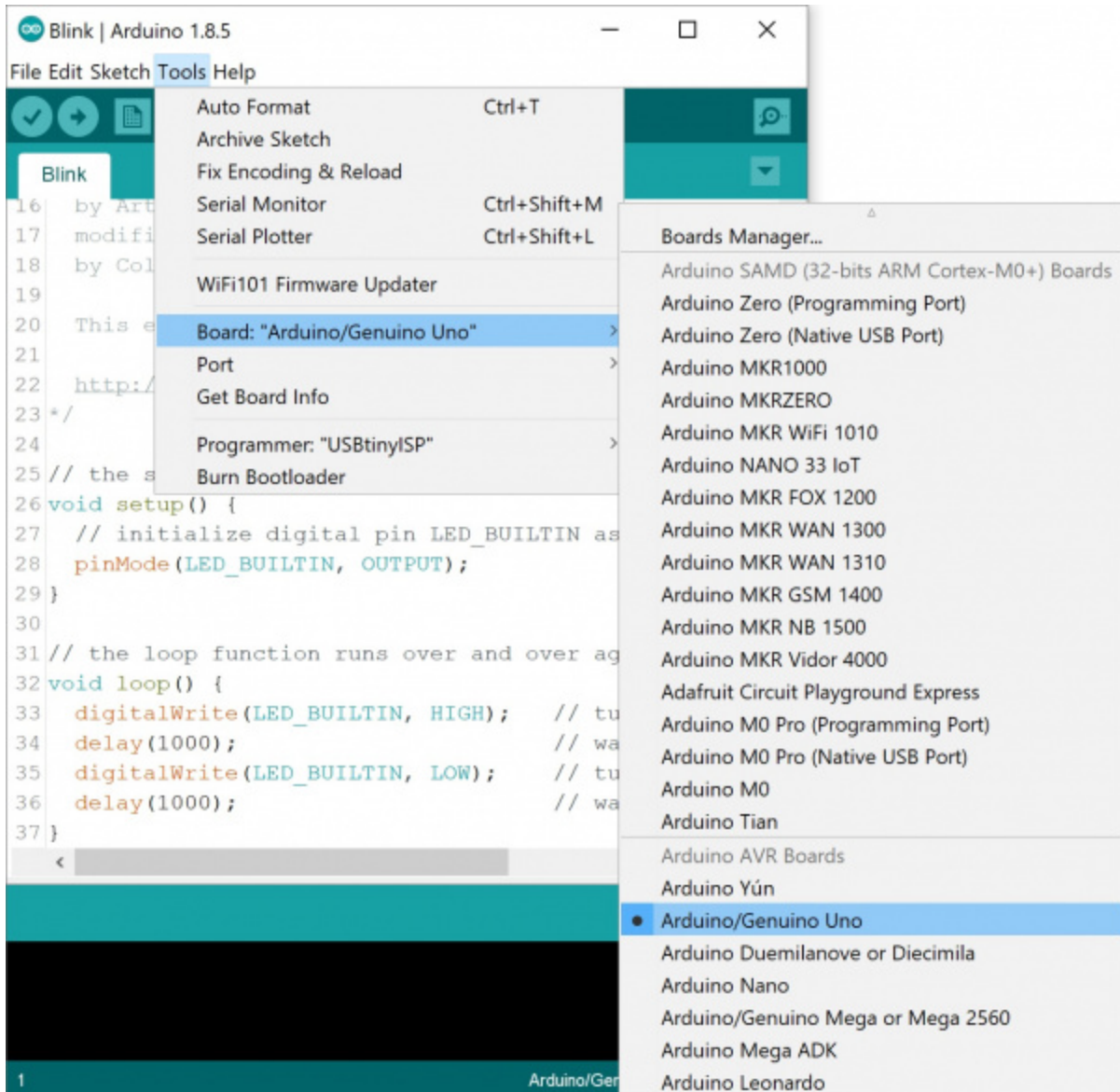
Power up your Arduino by connecting your Arduino board to your computer with a USB cable (or FTDI cable if you're using an Arduino Pro). You should see the an LED labeled 'ON' light up. (This diagram shows the placement of the power LED on the UNO).

DRIVERS FOR REDBOARD ON WINDOWS

Depending on your board, there may be different USB-to-serial converters on your board. If you are using a RedBoard you will need to go to the [How to Install FTDI Drivers](#) tutorial. If you are using the or RedBoard Qwiic, you will need to go to the [How to Install CH340 Drivers](#) tutorial. Make sure to look closely at your board and its respective hookup guide to determine USB-to-serial converter that is on board.



Select the type of Arduino board you're using: **Tools > Board > Arduino Uno.**



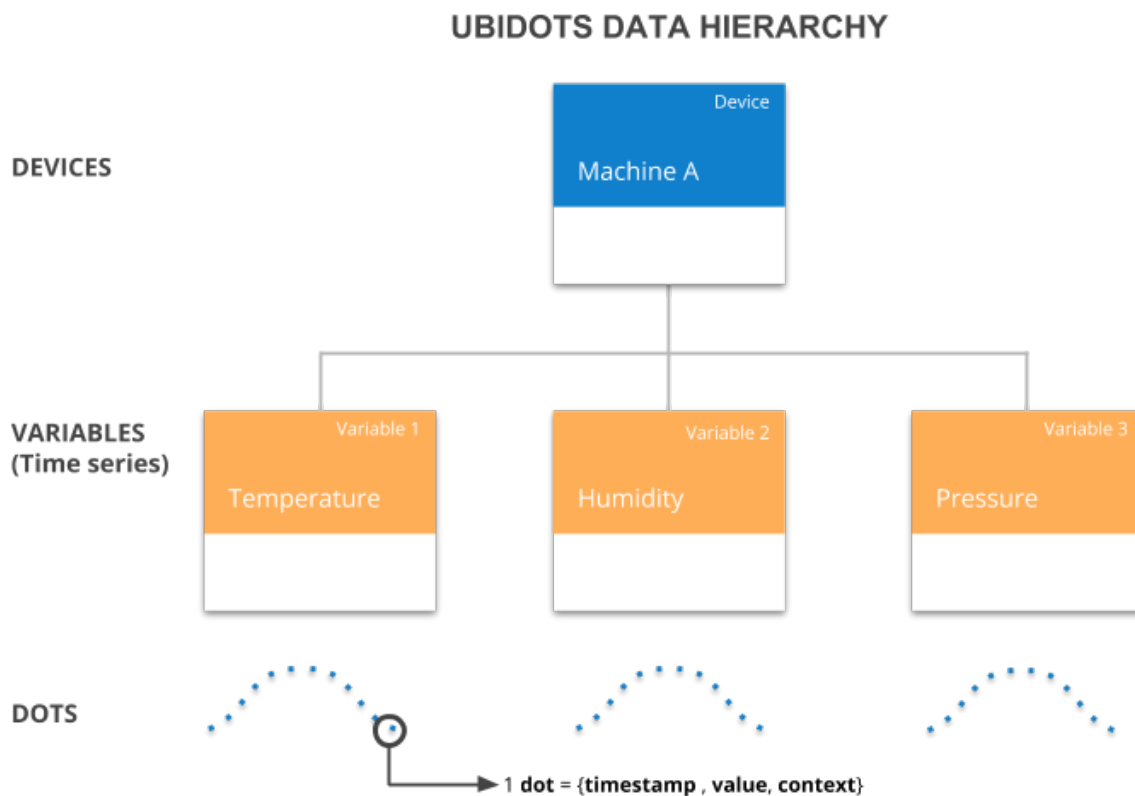
With your Arduino board connected, and the Blink sketch open, press the **"Upload"** button.

After a second, you should see some LEDs flashing on your Arduino, followed by the message **"Done Uploading"** in the status bar of the Blink sketch.

If everything worked, the onboard LED on your Arduino should now be blinking!
You just programmed your first Arduino!

4.4 UBIDOTS EXPLORER: INTRODUCTION

Ubidots is an Internet of Things (IoT) Development Platform that empowers engineers and System Integrators to build IoT applications that turn sensor data into actionable insights. Our technology stack puts together backend and frontend ingredients to accelerate IoT innovation. Equipped with device friendly APIs (accessed over HTTP, MQTT, TCP or UDP protocols), Ubidots provides a simple and secure connection for sending and retrieving data to and from our cloud service in real-time. For this reason, we avoid the use of examples using libraries. However, if you'd like to use one of our libraries (Python, C, and more), check out our compatible devices and API clients sections.



The easiest way to **send values to Ubidots** is by specifying your Device label in the request *path* and making a POST request to it.

Replace the {DEVICE_LABEL} with the unique label of the desired device to send data to. If the device does not exist in your Ubidots account, it will be

automatically created. Device labels can contain any alphanumerical character, without blank spaces, i.e weather-station, my-home, esp8266.

Unibot create chatbots for your brand communication. We are a team of developers, digital marketers, creative writers, and strategists. And we create chatbots for instant messaging between brands and people. Messengers are now the primary communication tool. They are simple, instant, and convenient. And the advent of chatbot platforms has made them also a powerful business tool. We bring communication to a new level by designing smart bots for popular platforms.

CHAPTER 5

RESULT

5.1 CONNECTIONS:

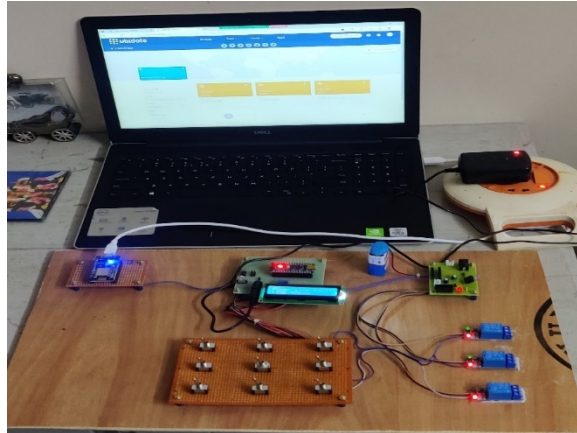


FIG 5.1 CONNECTIONS

Fig 5.1 Is the Connections of the following components NodeMCU(ESP 8266), Microcontroller (ATmega 8A), Power Unit , Switching Unit, and finally the relay Unit. Where they are connected with the normal connecting wires as shown in the figure

5.2 CLOUD PLATFORM:

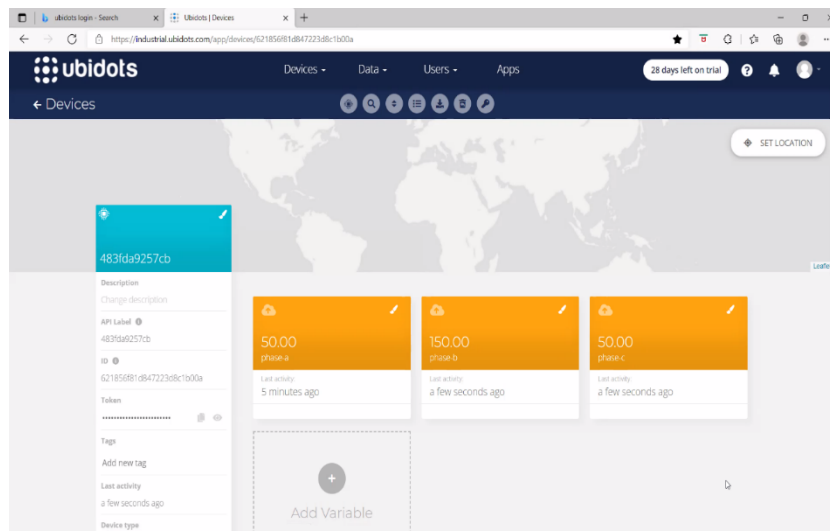
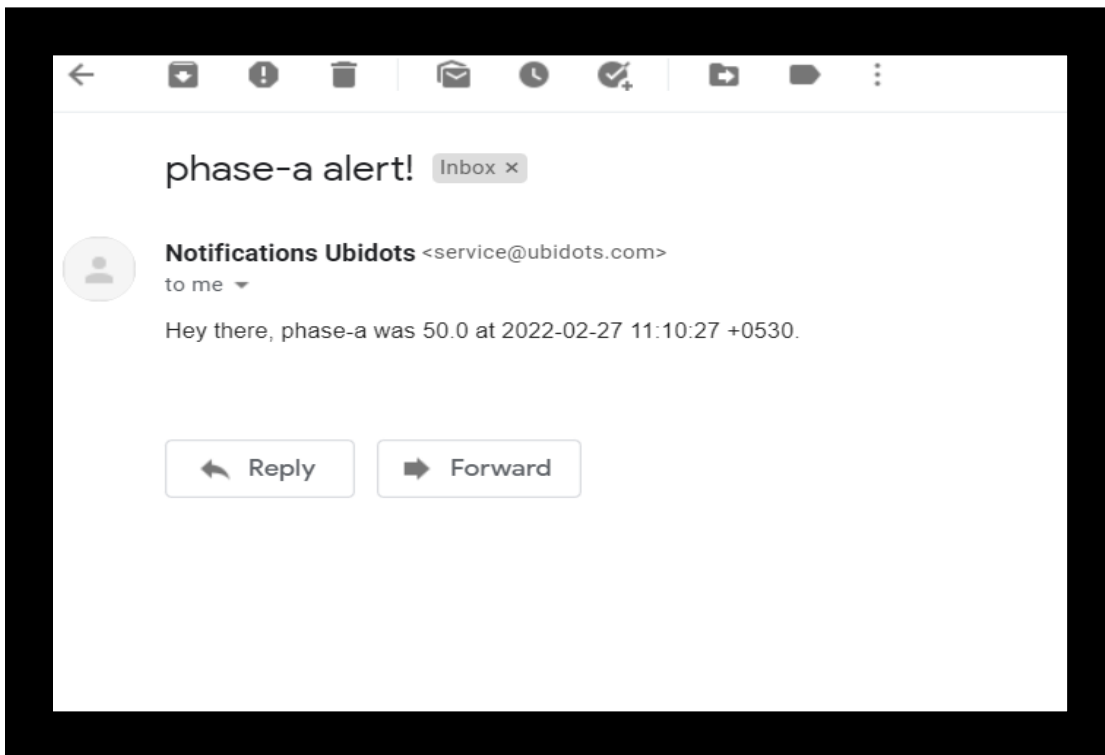


FIG 5.2 CLOUD PLATFORM

Fig 5.2 The Cloud Platform which we are using is Ubidots. Here we can observe the each and everything in this website quite accurately. Where as here we can observe that, at which particular place the fault occurred, in which particular phase the fault occurred, in which time as shown in the figure

5.3 ALERTS SCREENSHOTS:



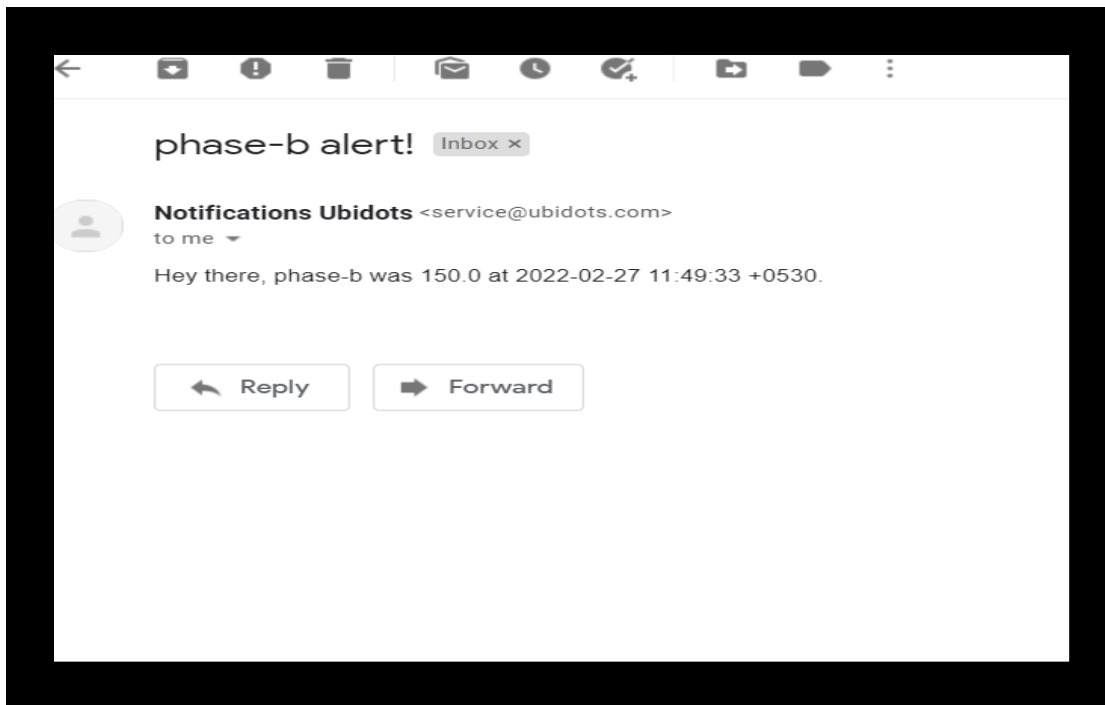


FIG 5.3 ALERTS SCREENSHOTS

Fig 5.3: These are the Alerts means the kind of warning msg where we can send according to our option like mail, text, etc..., Hear in the fig we are using mail as a default.

CHAPTER 6

SUMMARY:

The basic idea about is to suppress the maximum damage occurred caused by the transmission lines when the fault occurred, with help of the IOT equipment.

CONCLUSION:

Finally, we have designed in such a way that, whenever the fault occurred the relay will sense it and with help of the relay we can tell the exact location too. So that the nearer power plant will get alert msg, and the following line man comes under that power plant will get alert msg too so that he can reach the location accurately.

CHAPTER 7

REFERENCES

1. Dr. H. Umadevi, Niketh B. Vijaydeep, Vikas TV, 'Underground Cable Fault Monitoring & Detection System using IoT & Arduino', Journal for research, (2018).
2. Henna Sam. E. J, Pavithra. U. H, Ramakrishnan. P. V, Riyash Basha. S and Mr. T. M. Sathish kumar, 'Underground Cable Fault Detection Using IoT', Asian Journal of Applied Science and Technology (AJAST), (2018).
3. Ankit S. Gaulkar, Karunesh V. Gaurkar, Kajal V. Dhoble, Prachita K. Yerane, Prof. Nisha Warambhe, 'Result Paper of Automatic Underground Cable Fault Detector Using IOT', MANTECH Publications (2019).
4. Emmanuel Gbenga Dada, Abdulkadir Hamidu Alkali, Stephen Bassi Joseph, Umar Abba Sanda, 'Design and Implementation of Underground Cable Fault Detector', International Journal of Science and Engineering Investigations (IJSEI), (2019).
5. Harshit Srivastava; Naveen Mishra; Prajjwal Srivastava, 'Underground Cable Fault Detection', International Journal of Computing and Technology (IJCAT), (2019).
6. 6. Lakshmi Goswami and Piyush Agrawal, 'IoT based Diagnosing of Fault Detection in Power Line Transmission through GOOGLE Firebase database', Institute of Electrical and Electronics Engineers (IEEE) (2020).
7. 7. Prakash T; Arun G; Arun Kumar B; Deepan S; Kalyana Venkata Ramanan. "IoT Based Smart Irrigation for Multi Cropping System". International Research Journal on Advanced Science Hub, 2, 4, 2020, 7-12. doi: 10.47392/irjash.2020.20

8. 8. Neelam Yadav; Sunil K. Singh. "Implementation of Smart Helmet for Bikers". International Research Journal on Advanced Science Hub, 3, Special Issue ICARD-2021 3S, 2021, 135-139. doi: 10.47392/irjash.2021.081
9. 9. Soundharya B M; Yogeshwaran R; Tharanish Krishna A D; Vinitha V. "Automatic Check Post and E-Toll Payment System". International Research Journal on Advanced Science Hub, 3, Special Issue ICARD-2021 3S, 2021, 57-62. doi: 10.47392/irjash.2021.064
10. 10. Jijusasukumar S; Kaviya K; Logida R; Chinmaya S; Sangeetha K. "Gas Leakage Monitoring System Using IOT". International Research Journal on Advanced Science Hub, 3, Special Issue ICARD-2021 3S, 2021, 108-111. doi: 10.47392/irjash.2021

