

IOT BASED FAULT DETECTION OF UNDERGROUND CABLES THROUGH NODEMCU

Submitted in partial fulfilment of the requirements for the award of

Bachelor of Engineering degree in

Electronics and Communication Engineering

By

D HEMANTH REDDY (38130057)



SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAICHENNAI –600119

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 D HEMANTH REDDY (38130057) who carried out the project entitled "**IOT BASED FAULT DETECTION OF UNDERGROUND CABLES THROUGH NODEMCU**" under our supervision from August-2021 to May-2022.

Internal Guide

(Dr. L.MEGALAN LEO, 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

I, D HEMANTH REDDY (38130057) hereby declare that the Project report entitled **“IOT BASED FAULT DETECTION OF UNDERGROUND CABLES THROUGH NODEMCU”** is done under the guidance of **Dr.L. MEGALAN LEO M.E.,Ph.D** is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in **ELECTRONICS AND COMMUNICATION ENGINEERING**.

DATE:

D HEMANTH REDDY

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are 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. We are grateful to them.

We convey our thanks to **Dr .N.M.NANDHITHA, M.E., Ph.D.**, Prof & Dean of school of Electrical and Electronics Engineering and **Dr .T.RAVI, M.E., Ph.D.**, Heads of the Department, Department of Electronics and communication Engineering for providing us necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **DR .L.MEGALAN LEO, M.E., Ph.D.**, for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

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

ABSTRACT

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 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. Blunder recognition has been a significant objective for framework engineers in both transmission and conveyance. Know the beginnings of mix-ups and difficult work; Rapid mistake recognition assists with safeguarding the material from harm before it is harmed. Nearby elements help to dispense with blunders in human assistance and to figure out where regular mistakes happen, consequently lessening mistake rates and shortening feed times. The framework utilizes a microcontroller board to distinguish transmission line blames and sends one to the Centre for observing utilizing an IoT gadget.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY	3
3	EXISTING SYSTEM	5
4	PROPOSED SYSTEM	6
	4.1 BLOCK DIAGRAM	7
	4.2 REQUIRED COMPONENTS	7
	4.3 EMBEDDED SYSTEMS	7
	4.4 POWER MANAGEMENT	13
	4.5 REGISTERS	14
	4.6 REGULATORS	15
	4.7 IOT	15
5	RESULTS AND DISCUSSION	39
6	SUMMARY AND CONCLUSION	50
	REFERENCES	51

LIST OF FIGURES

FIGURE.NO	TITLE	PAGE.NO
4.1	BLOCK DIAGRAM OF A TYPICAL EMBEDDED SYSTEM	9
4.2	EMBEDDED SYSTEMS HAVING HETEROGENEOUS ARCHITECTURE	11
4.4.1	GENERAL BLOCK OF POWER SUPPLY UNIT	13
4.7.1	LAYERED ARCHITECTURE OF THE INTERNET OF THINGS	19
4.7.2	FUNCTIONAL COMPONENTS OF A MIDDLEWARE FOR IOT BASED SYSTEMS	21
4.7.3	NODEMCU DEVELOPMENT BOARD KIT	27
4.7.4	NODEMCU DEVELOPMENT BOARD KIT	28
4.7.5	ARDUINO NANO	30
5.1	ASSEMBLED COMPONENTS AS PER BLOCK DIAGRAM	39
5.2	FAULT DETECTED GRAPH- PHASE C	39
5.3	MEASURED CURRENT RATING VALUES	40
5.4	MONITORING CABLE FAULTS	40

CHAPTER 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, underground cables have been used for many decades.

Underground 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.

The transmission line and framework establishment assume a key part in sharing flawlessly with clients. Research has shown that 80% of client support interferences are because of organization disappointments. It is essential to gather information on the transmission line to work on the dependability of the circulation. Presently, the conveyance framework incorporates frameworks like the utilization and activity of geographic data frameworks, meter substitution, taking care of, and perusing. An element of this thought is that power laborers or watch officials can invert estimations like how much current streaming or any degree of voltage. The separation from power to purchasers is partitioned into groups. Here, the responsibility can be a hub or a group head, contingent upon the power utilized. Electrical cables are important for the power framework and have a high gamble of being gone after by the climate. Line flaws brought about by poisons like lightning, wind, falling vegetables, haze, and salt can't be constrained by people. An assurance framework is introduced in the transmission framework, which

demonstrates an issue that isolates the fixed part from the remainder of the framework, which is harmed and harming the hardware. Circulation and conveyance pipelines are the principle interface that prompts the fundamental power supply line to end clients. Transmission lines associate the station and the freight place. Power plants are found in excess of 100 miles from the middle, so there is a high likelihood that they will show up on the transmission line. They ought to be separated quickly as issues might disturb the power framework. Injury investigation is a significant issue in rapidly wiping out shortcomings in the activity of the power framework, in re-establishing power rapidly, or possibly halting it. The framework consequently sends any data to the dashboard. It screens estimations like voltage and current simultaneously and sends data electronically. The fundamental thought is to furnish customers with a consistent heap of power; Establish fast and solid improvement in a dependable way, subsequently working on the help of power purchasers. It distinguishes flaws in the home of electrical and watch faculty. When a blunder has happened, the mistake is made all the more precisely to keep the freight from being taken.

CHAPTER 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 Bagade¹ , Gita Bhardwaj² , Shweta Wasnik³ , Dheeraj Kawade⁴ , 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.

CHAPTER 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 detractor. The failures of underground power distribution cables represent 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.

Dr. H. Umadevi, Niketh B. Vijaydeep, and Vikas TV [7] have expressed that the world's phones are frequently harmed because of soil conditions, disintegration, mice, and then some. The wellspring of the issue is hard to recognize and all wires should be eliminated starting from the earliest stage check and right the shortcoming. The venture expects to decide the area of deficiencies in the underground link line from the base station in km utilizing the PIC16F877A regulator. The rope ought to be reviewed to decide whether the link is harmed. This model purposes a basic comprehension of Ohm's regulation.

Henna Sam.E.J, Pavithra.U.H, Ramakrishnan.P.V, Riyash Basha. Underground frameworks are frequently utilized in huge urban communities. In the event that a mistake happens under any condition, it is challenging to make rectifications to the circuit on the grounds that the area of the link blunder is obscure. This IoT strategy is utilized to figure out where the mistake is, highlight the LCD screen simultaneously and utilize the IoT module to send data to our site in graphical structure.

Ankit S. Golkar, Karunesh V. Gaurkar, Kazhal V. Dhobl, Prachita K. Jerane, Prof. Nisha Varambhe Online There are various justifications for why it is hard to know the specific area of a blunder while erasing or fixing a street. There are many reasons for wires, however it is challenging to decide precisely where they are harmed during endlessly fixes. The framework gave in this record is utilized to distinguish the area of the harm, to report it to a specific website through the Internet by means of a Wi-Fi module, and to give earlier data.

CHAPTER 4

PROPOSED SYSTEM

This system represents that to compute the distance of the underground and overhead 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.

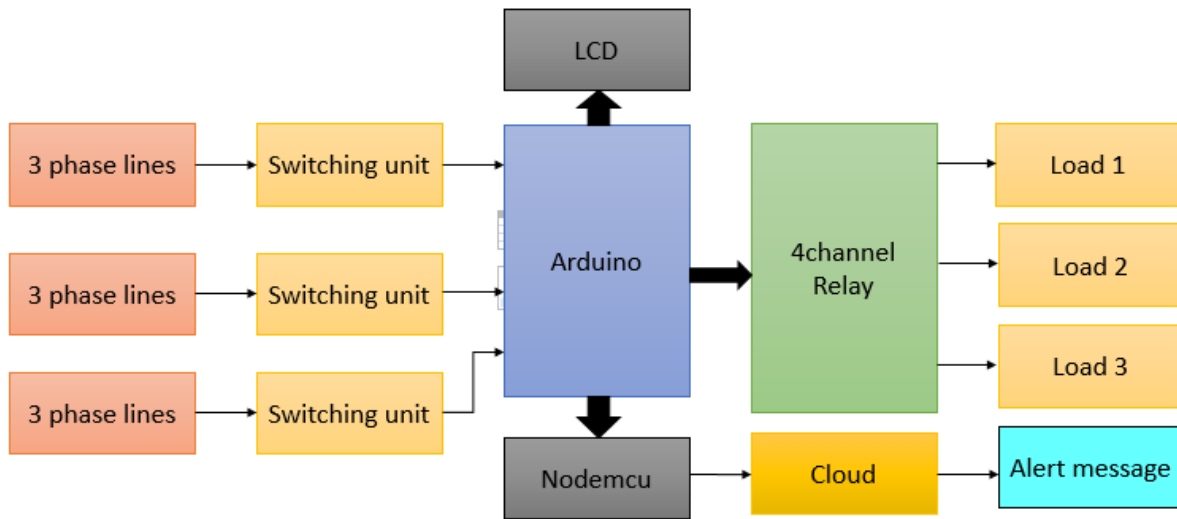
For underground cables (UC) fault detection, we used switching unit at each and every 10km of cable line. If any fault occurs we can easily locate the cable fault.

It's a difficult task to identify the Short Circuit faults in underground 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.

4.1 BLOCK DIAGRAM



4.2 REQUIRED COMPONENTS

Hardware required

- Switching units

- Arduino nano

- Nodemcu

- Relay

Software required

- Arduino IDE

- Embedded C

- Ubidots explorer

4.3 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. Some embedded systems are mass-produced, benefiting from economies of scale.

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

- Monitor the environment; embedded systems read data from input sensors. This data is then processed and the results displayed in some format to a user or users
- Control the environment; embedded systems generate and transmit commands for actuators.
- Transform the information; embedded systems transform the data collected in some meaningful way, such as data compression/decompression
- Although interaction with the external world via sensors and actuators is an important aspect of embedded systems, these systems also provide functionality specific to their applications. Embedded systems typically execute applications such as control laws, finite state machines, and signal processing algorithms. These systems must also detect and react to faults in both the internal computing environment as well as the surrounding electromechanical systems.
- There are many categories of embedded systems, from communication devices to home appliances to control systems. Examples include;

- Communication devices
e.g: modems, cellular phones
- Home Appliances
e.g: CD player, VCR, microwave oven
- Control Systems
e.g: Automobile anti-lock braking systems, robotics, satellite control

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. These systems are embedded into some device for some specific purpose other than to provide general purpose computing . A typical embedded system is shown in Fig 1.1

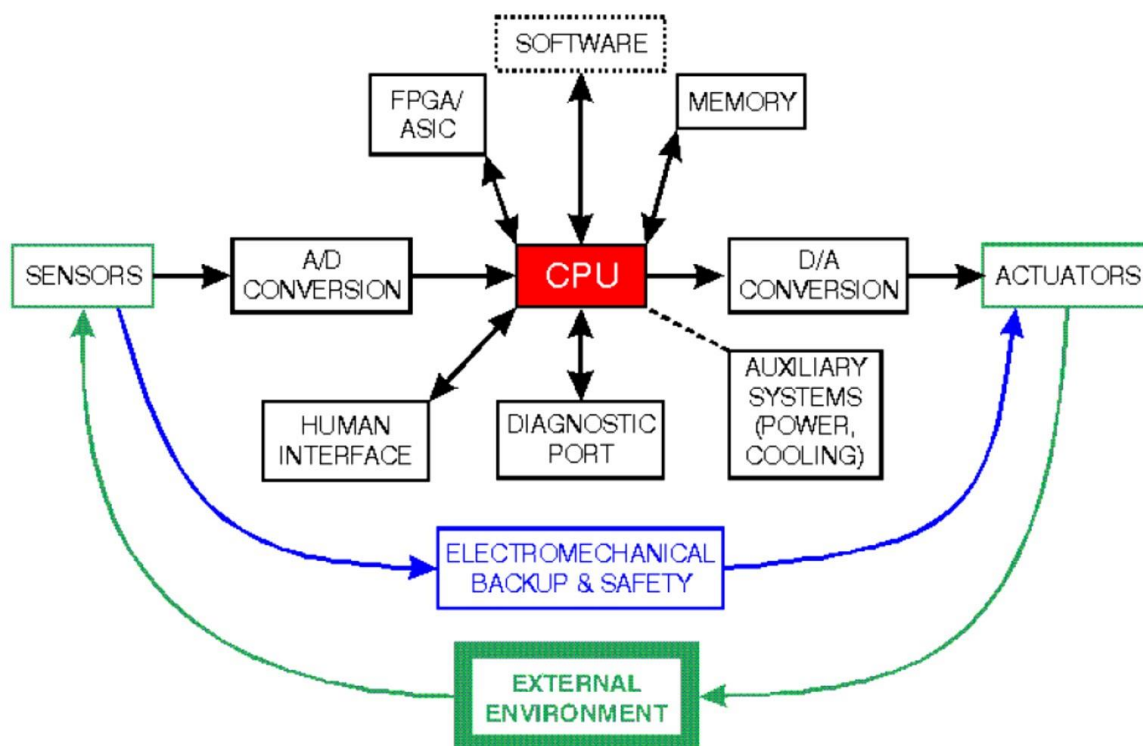


Fig 4.1 Block diagram of a typical embedded system

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 1.2). 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.

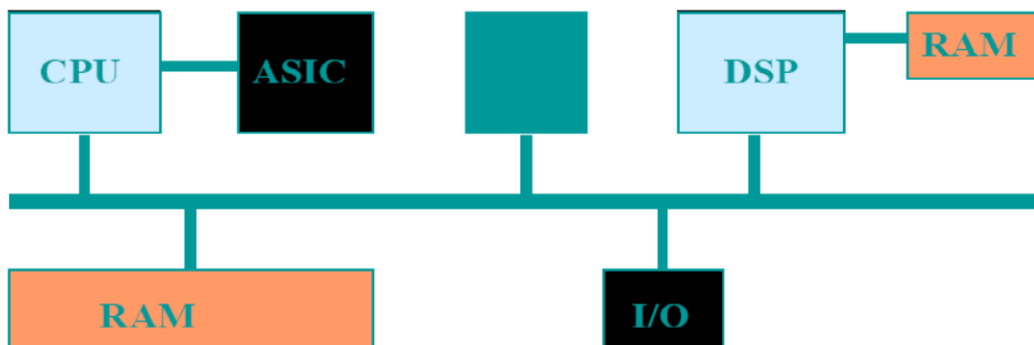


Fig 4.2 Embedded Systems having Heterogeneous Architectures

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.

4.4 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.

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. 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.

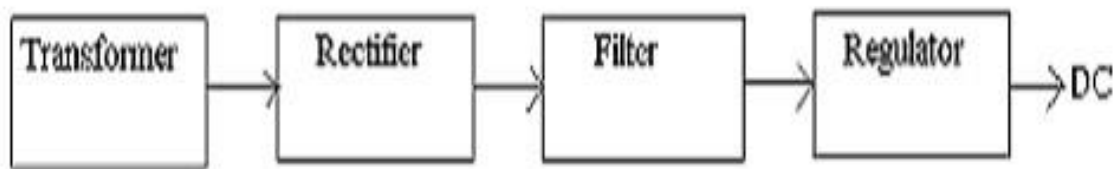


Fig 4.4.1 General Block Of Power Supply Unit

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.

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.

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.

4.5 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.

When an address code is written to IR, the data is automatically transferred from the DD RAM or CG RAM to the DR. data transfer between the MPU is then completed when the MPU reads the DR. likewise, for the next MPU read of the DR, data in DD RAM or CG RAM at the address is sent to the DR automatically. Similarly, for the MPU write of the DR, the next DD RAM or CG RAM address is selected for the write 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

4.6 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.

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
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

4.7 IOT

The structure of this chapter is the following: Section 1.1 defines the motivation for this work. Section 1.2 shows the research objectives and questions. Section 1.3 presents the steps required to achieve our objectives. Section 1.4 shows the structure of the report.

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 [1]. 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 [2]. 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].

IoT-based system is in charge of providing knowledge from an environment to an non-expert user. IoT-based system can be used in different environments, so it needs to be able to address many heterogeneous devices. Thus, a major concern within developing an IoT-based system is how to handle the interaction with the heterogeneous devices for non-expert users. This concern can be addressed by a middleware layer between devices and non-expert users. This layer is responsible to hide the diversity of devices from the user perspective, and provides access transparency to the devices for the end users.

The idea of creating abstractions of devices been addressed in the literature. The middleware we found in the literature can provide satisfaction by facilitating the interaction with devices, but they do not support low-level device configuration [4].

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 [8]. 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. From the aspect of a business user the impacts would be in fields such as automation and industrial manufacturing, logistics, business process management, intelligent transportation of people and goods [9]

IoT integrates physical things into information networks. IoT covers the overall infrastructure, including software, hardware and services, which is used to support these information networks. The integrated physical things can exchange data about the physical properties and information that they sense in their environment. To identify devices, we can use identification technologies like for example RFID, which allow each device be uniquely identified [10].

International Telecommunication Union (ITU)¹ defines IoT as “A global infrastructure for the Information Society, enabling advanced services by interconnecting (physical and virtual) things based on, existing and evolving, interoperable information and communication technologies”[11]

Internet of Things-Architecture² (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.”[12]

IoT has a layered architecture designed to answer the demands of various industries, enterprises and society. Fig. 2.1 shows a generic layered architecture for IoT that consist of five layers, which are discussed, in the following:[13]

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.

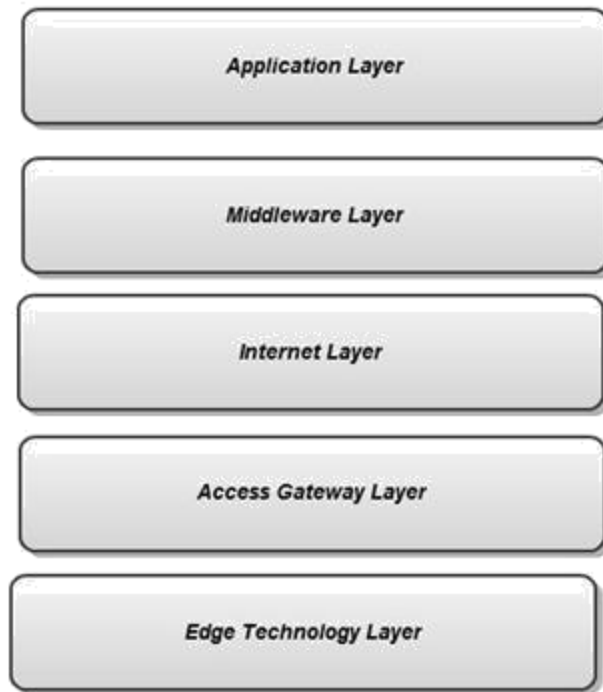


Figure 4.7.1 Layered architecture of the Internet of Things (IoT).

IoT-based systems can be used for different purposes and areas, so that, we have to face different challenges. In this section, we explain some of the challenges that need to be considered in research activities [10]:

Edge technology

At the hardware level, more research efforts are required to develop the technology of embedded devices, sensors, actuators and (passive and active) identification, since an IoT-based system must be able to gather sufficient information about the real world by employing a large variety of devices and environments. Thus, there more work is still required to connect heterogeneous devices and deploy them in IoT applications, and to provide support for new devices.

Networking technologies

In IoT, *things* are connected through different kinds of networks, i.e. mobile, wired and wireless network. These networks support bi-directional communication at different levels among the real world objects, applications and

services that are employed by the IoT applications. This highly distributed structure should provide interconnection with low energy consumption, while distributed data can cause privacy issues.

Middleware system

In IoT, we have heterogeneous devices and networks. Their heterogeneity can potentially increase with new technologies. To facilitate the use of these devices by IoT applications, we should shield their heterogeneity. Therefore, we need to develop a secure, scalable and semantically enriched service-oriented middleware to cope with the heterogeneity of devices.

Platform services

They support a superior management of all involved devices in an integrated way, ensuring scalability, high availability, and the safe and secure execution of the requested functionality from devices.

In continue, we focus on the middleware challenges, because we are looking for the functionalities that a middleware can provide for the application layer in the IoT-based systems.

IoT_based Middleware

Bandyopadhyay, S. et. al. have studied the middleware systems that have been applied in IoT-based systems [1]. They classify the required functionality of middleware to manage interaction with a variety of devices in four functional components, namely (1) interface protocols, (2) device abstraction, (3) central control, context detection & management, and (4) application abstraction (shown in Fig. 2.2). In the following, we explain these components in details.

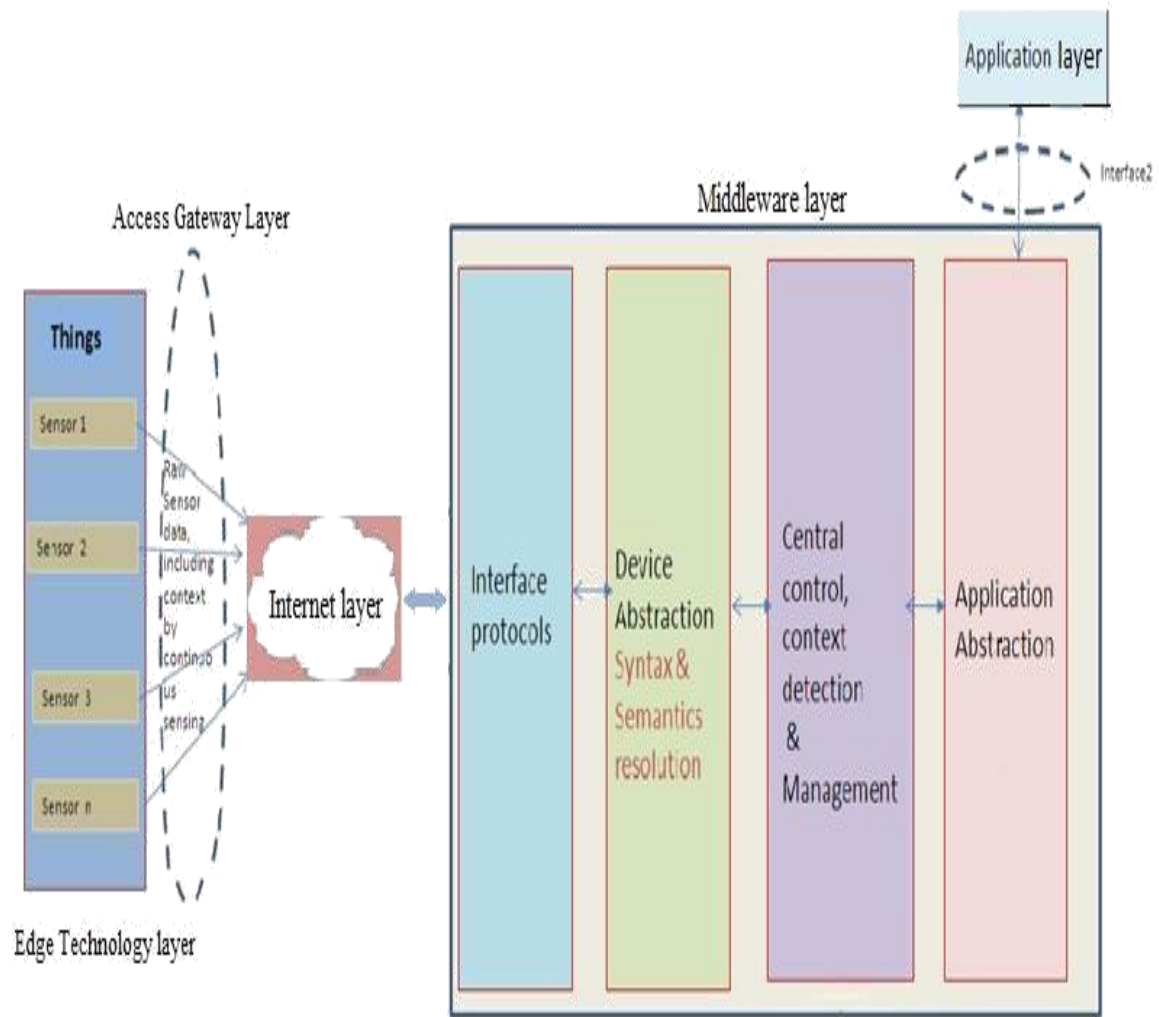


Figure 4.7.2 Functional components of a middleware for IoT-based systems

This component is in charge of providing *technical interoperability*. Interoperability in the context of *Interface protocols* means: *the ability of two systems to interoperate by using the same communication protocols*. According to ETSI (European Telecommunications Standards Institute) [14] *technical interoperability* is defined as the association of hardware or software components, systems and platforms that enable machine-to-machine communication to take place. This kind of interoperability is often centered on (communication) protocols and the infrastructure needed for those protocols to operate [14].

The Interface Protocol component defines protocols for exchanging information among different networks that may work based on different communication

protocols, in order to allow technical interoperability. This component is responsible for handling basic connectivity in the physical and data link, network, transport, and sometimes the application layer of the TCP/IP stack.

To cope with the heterogeneity of devices, we can use a wrapper for each device to translate the protocol supported by the device to a common protocol. This wrapper can be placed either on the device side or the middleware side. If we want to have direct interaction with devices, we should place the wrapper in the middleware side. Devices usually have limited capability of computational process, so this would be a reason to implement wrapper on the middleware side. In contrast, in case of indirect interaction with devices we can develop an intermediary wrapper between the middleware and the devices. The interface protocol component is responsible to allow the middleware to support both direct and indirect interactions.

This component is responsible for providing an abstract format to facilitate the interaction of the application components with devices. This abstraction provides *syntactic* and *semantic* interoperability, which are defined by ETSI [14] as follows:

- *Syntactic interoperability* is associated with data formats. The message transferred by communication protocols must have a well-defined syntax and encoding format, which can be represented by using high-level transfer syntaxes such as, HTML and XML.
- *Semantic interoperability* is usually associated with the meaning of the content of message which is understandable for human. Thus, interoperability on this level means that there is a common understanding among people on the meaning of the content (information) being exchanged among them. However, since DA does not communicate directly with human, semantic interoperability in the context of DA is in charge of providing this common understanding for applications.
- Semantic interoperability relies on semantic models which tends to be domain specific. For example, one way to provide semantic interoperability in Service Oriented (SOA) [15] based middleware is by using Devices Profile for Web

Services (DPWS) [16] . In this context, each device type refers to a distinguished service type

- DA component provide two general functionalities: (1) to ask devices to perform some functionality and (2) to define and configure devices DPWS uses the XML format that is shown in Code 2.1.

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 [1]. 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 [1]. 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 [17]. 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.

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 [18]. 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:

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 (W3C)³ 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-anywhere fashion. Pervasive Computing is all about making the human's life easier by exploiting the available computing technologies* [20]. This motivates scientists to develop new technologies for the everyday people's lives.

Since the vision of IoT is almost similar to Pervasive computing, we reviewed the literatures that address both IoT and Pervasive systems. To select the middleware to discuss we had the following approaches:

1. We looked at typical IoT middleware such as ISMP [21], ASPIRE [22] GSN [23]. We selected HYDRA [24] to review, because it is the most popular and well-documented middleware in comparison to the mentioned middleware. Also, based on the study of Bandyopadhyay, S. et. al [3], those middleware does not support the discussed functional components in Chapter 3, while HYDRA supports them.
2. We looked at typical Pervasive system middleware such as [25], [26], [27]. We selected the following middleware to review:
 - a. AURA [28] because this middleware focus on elaboration and manipulation of the gathered data from devices. we aim to provide ease of configuration and deployment for end user and developer. Since to meet this goal, we require to facilitate the gathered data manipulation, we reviewed AURA that was the only middleware among the reviewed paper that considered the data manipulation.
 - b. TinyDB [19] focus on gathering data from devices. Since in an IoT-based system we need to gather data from environment through different devices, we

reviewed TinyDB that is a popular middleware.

- c. WiseMID [29] is the only middleware among the reviewed middleware that is specific for energy saving purpose. As saving the energy of devices is important issue, we reviewed this middleware.

NodeMCU is an open source IoT platform.^{[4][5]} It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.^{[6][7]} 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^[8] and SPIFFS

NodeMCU was created shortly after the ESP8266 came out. On December 30, 2013, Espressif Systems^[6] began production of the ESP8266.^[10] The ESP8266 is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core,^[citation needed] 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.^[11] 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.^[12] Later that month, Tuan PM ported MQTT client library from Contiki to the ESP8266 SoC platform,^[13] 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^[14] to NodeMCU project,^[15] 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.

As Arduino.cc began developing new MCU boards based on non-AVR processors like the ARM/SAM MCU and used in the Arduino Due, they needed to modify the Arduino IDE so that it would be relatively easy to change the IDE to support alternate toolchains to allow Arduino C/C++ to be compiled for these new processors. They did this with the introduction of the Board Manager and the SAM Core. A "core" is the collection of software components required by

the Board Manager and the Arduino IDE to compile an Arduino C/C++ source file for the target MCU's machine language. Some ESP8266 enthusiasts developed an Arduino core for the ESP8266 WiFi SoC, popularly called the "ESP8266 Core for the Arduino IDE".^[16] This has become a leading software development platform for the various ESP8266-based modules and development boards, including NodeMCUs.

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.

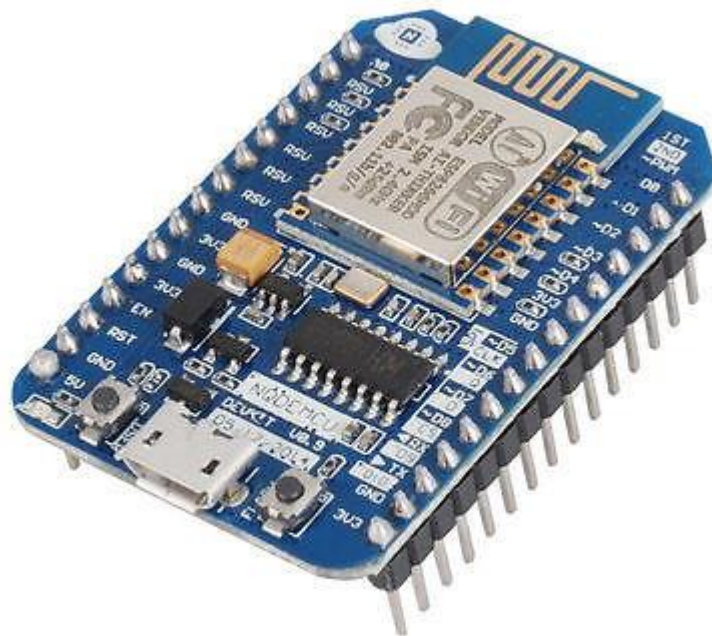
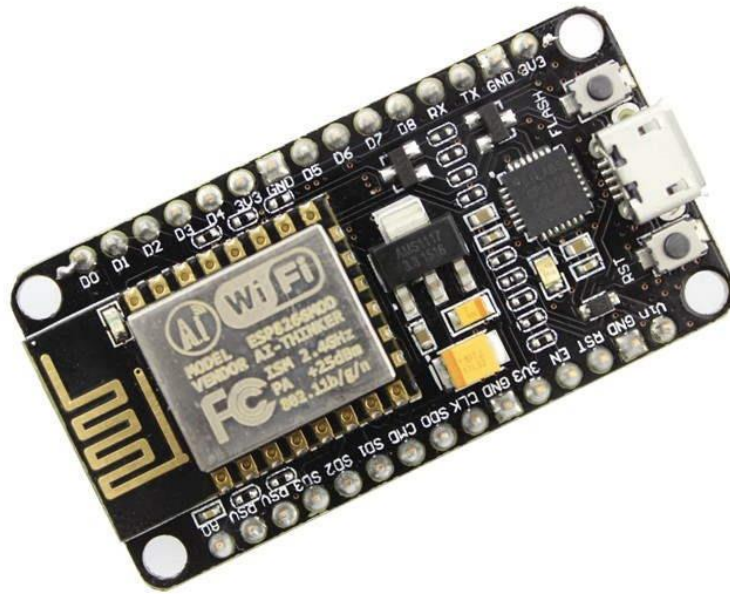


Fig 4.7.3 NodeMCU Development Board/kit v0.9 (Version1)

Since NodeMCU is open source 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.



4.7.4 NodeMCU Development Board/kit v1.0 (Version2)

For more information about NodeMCU Boards available in market refer NodeMCU Development Boards NodeMCU Dev Kit has **Arduino like** Analog (i.e. A0) and Digital (D0-D8) pins on its board. It supports serial communication protocols i.e. UART, SPI, I2C etc. Using such serial protocols we can connect it with serial devices like I2C enabled LCD display, Magnetometer HMC5883, MPU-6050 Gyro meter + Accelerometer, RTC chips, GPS modules, touch screen displays, SD cards etc.

NodeMCU Development board is featured with wifi capability, analog pin, digital pins and serial communication protocols. To get start with using NodeMCU for IoT applications first we need to know about how to write/download NodeMCU firmware in NodeMCU Development Boards. And before that where this NodeMCU firmware will get as per our requirement. There is online NodeMCU custom builds available using which we can easily get our custom NodeMCU firmware as per our requirement.

Difference in using ESPlorer and Arduino IDE

Well, there is a programming language difference we can say while developing application for NodeMCU using ESPlorer IDE and Arduino IDE.

We need to code in C\C++ programming language if we are using Arduino IDE for developing NodeMCU applications and Lua language if we are using ESPlorer IDE.

Basically, NodeMCU is Lua Interpreter, so it can understand Lua script easily. When we write Lua scripts for NodeMCU and send/upload it to NodeMCU, then they will get executes sequentially. It will not build binary firmware file of code for NodeMCU to write. It will send Lua script as it is to NodeMCU to get execute.

In Arduino IDE when we write and compile code, ESP8266 toolchain in background creates binary firmware file of code we wrote. And when we upload it to NodeMCU then it will flash all NodeMCU firmware with newly generated binary firmware code. In fact, it writes the complete firmware.

That's the reason why NodeMCU not accept further Lua scripts/code after it is getting flashed by Arduino IDE. After getting flashed by Arduino sketch/code it will be no more Lua interpreter and we got error if we try to upload Lua scripts. To again start with Lua script, we need to flash it with NodeMCU firmware. Since Arduino IDE compile and upload/writes complete firmware, it takes more time than ESPlorer IDE.

- **Arduino Nano** is a small, compatible, flexible and breadboard friendly Microcontroller board, developed by Arduino.cc in Italy, based on ATmega328p (Arduino Nano V3.x) / Atmega168 (Arduino Nano V3.x).
- It comes with exactly the same functionality as in Arduino UNO but quite in small size.
- It comes with an operating voltage of 5V, however, the input voltage can vary from 7 to 12V.
- **Arduino Nano Pinout** contains 14 digital pins, 8 analog Pins, 2 Reset Pins & 6 Power Pins.
- Each of these Digital & Analog Pins are assigned with multiple functions but their main function is to be configured as input or output.
- They are acted as input pins when they are interfaced with sensors, but if you are driving some load then use them as output.

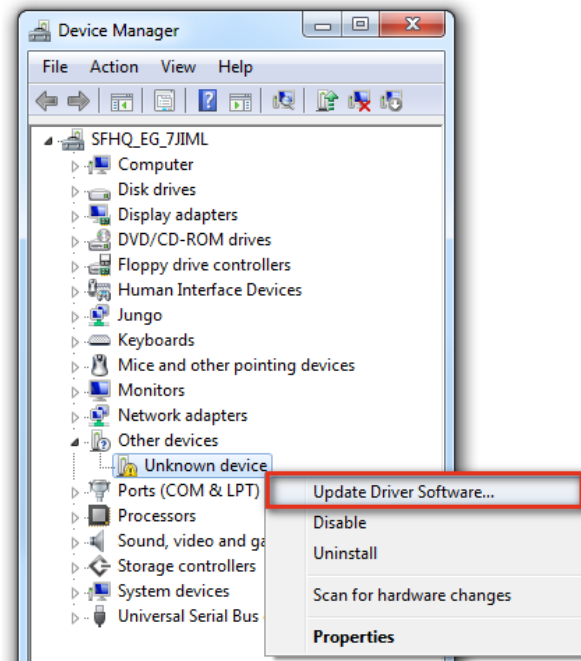
- Functions like `pinMode()` and `digitalWrite()` are used to control the operations of digital pins while `analogRead()` is used to control analog pins.
- The analog pins come with a total resolution of 10bits which measure the value from zero to 5V.
- Arduino Nano comes with a crystal oscillator of frequency 16 MHz. It is used to produce a clock of precise frequency using constant voltage.
- There is one limitation using Arduino Nano i.e. it doesn't come with DC power jack, means you can not supply external power source through a battery.
- This board doesn't use standard USB for connection with a computer, instead, it comes with Mini USB support.
- Tiny size and breadboard friendly nature make this device an ideal choice for most of the applications where a size of the electronic components are of great concern.
- Flash memory is 16KB or 32KB that all depends on the Atmega board i.e. Atmega168 comes with 16KB of flash memory while Atmega328 comes with a flash memory of 32KB. Flash memory is used for storing code. The 2KB of memory out of total flash memory is used for a bootloader.



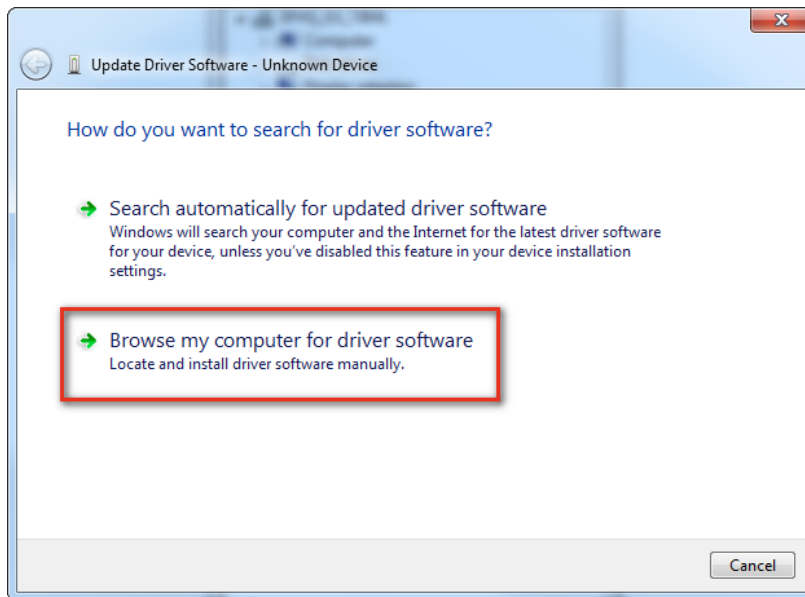
Fig 6.3.1 arduino nano

- The SRAM can vary from 1KB or 2KB and EEPROM is 512 bytes or 1KB for Atmega168 and Atmega328 respectively.
- This board is quite similar to other Arduino boards available in the market, but the small size makes this board stand out from others.
- Following figure shows the specifications of Arduino Nano Board.

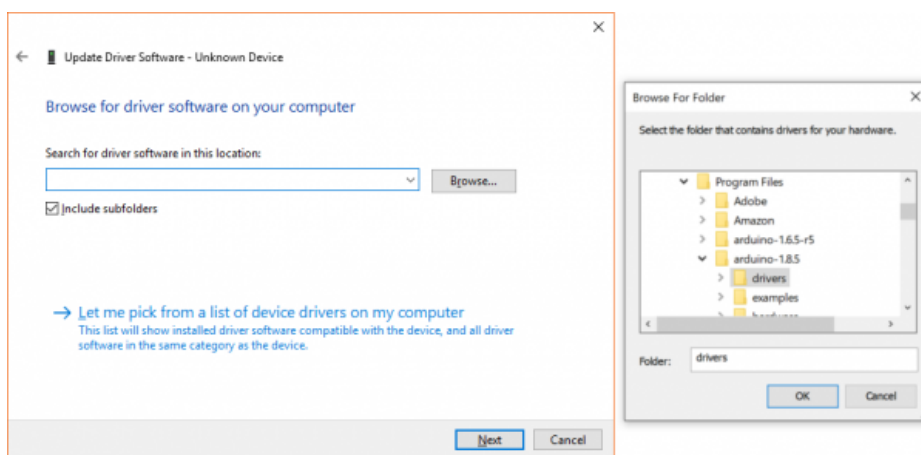
- It is programmed using Arduino IDE which is an Integrated Development Environment that runs both offline and online.
- No prior arrangements are required to run the board. All you need is board, mini USB cable and Arduino IDE software installed on the computer. USB cable is used to transfer the program from computer to the board.
- No separate burner is required to compile and burn the program as this board comes with a built-in boot-loader. Fig 6.3.2 Following figure shows the pinout of Arduino Nano Board.
- Each pin on the Nano board comes with a specific function associated with it.
- We can see the analog pins that can be used as an analog to digital converter where A4 and A5 pins can also be used for I2C communication. Similarly, there are 14 digital pins, out of which 6 pins are used for generating PWM.



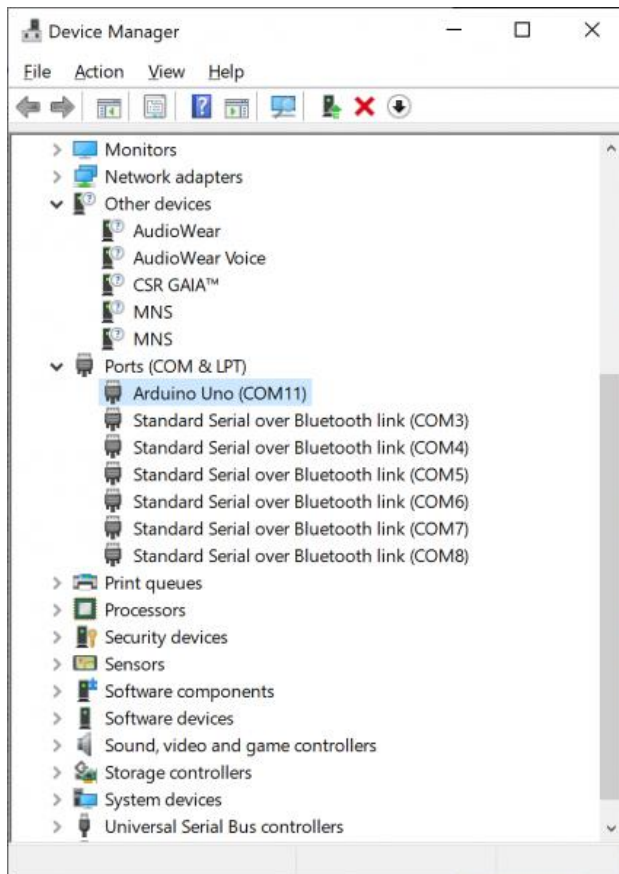
Next, choose the **"Browse my computer for Driver software"** option



Finally, navigate to the Arduino IDE folder. This should be where you unzipped the Arduino IDE (e.g. it should be similar to the following path with a different version number: `C:\Program Files\arduino-1.8.5\drivers`). Depending on what version of Windows you have, you may be able to select the Uno's driver file, named "**Arduino.inf**", located in the "**Drivers**" folder (not the "FTDI USB Drivers" sub-directory). If you cannot see the *.inf file, it is probably just hidden. You can select the "**drivers**" folder with the "**include sub-folders**" option selected instead.



Windows will finish up the driver installation from there! Your computer will enumerate with a COM port. You may see a COM port number depending on what is currently saved in your computer. Try to remember what the number is when uploading. If not, you can always navigate back to the device manager and power cycle the Arduino to determine what number your Arduino enumerated on.



For earlier versions of the Arduino boards (e.g. Arduino Duemilanove, Nano, or Diecimila) check out [this page](#) for specific directions.

ARDUINO.CC - GUIDE: WINDOWS

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 RedBoardQwiic, 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.

How to Install FTDI Drivers

JUNE 4, 2013

How to install drivers for the FTDI Basic on Windows, Mac OS X, and Linux.

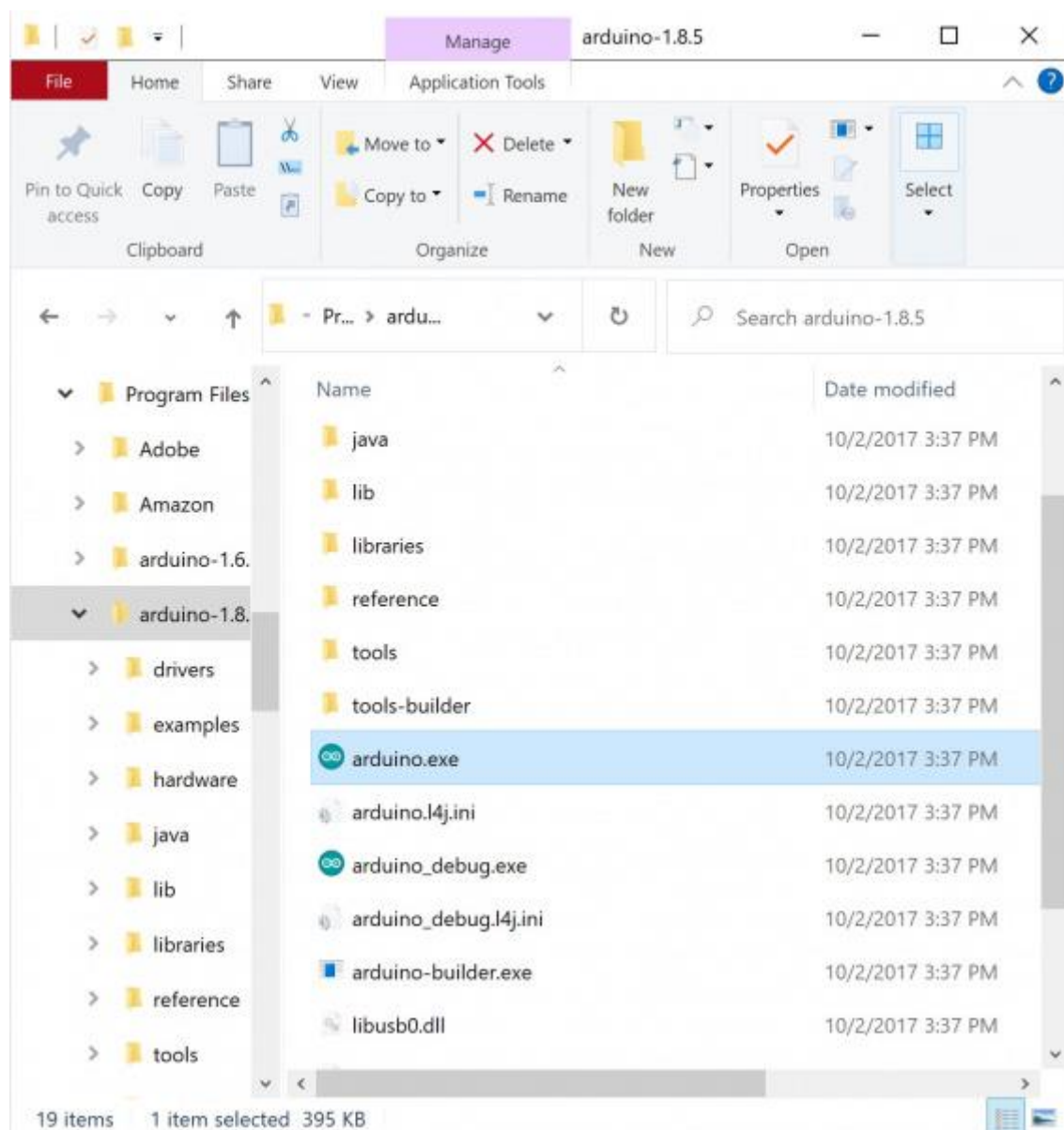
How to Install CH340 Drivers

AUGUST 6, 2019

How to install CH340 drivers (if you need them) on Windows, Mac OS X, and Linux.

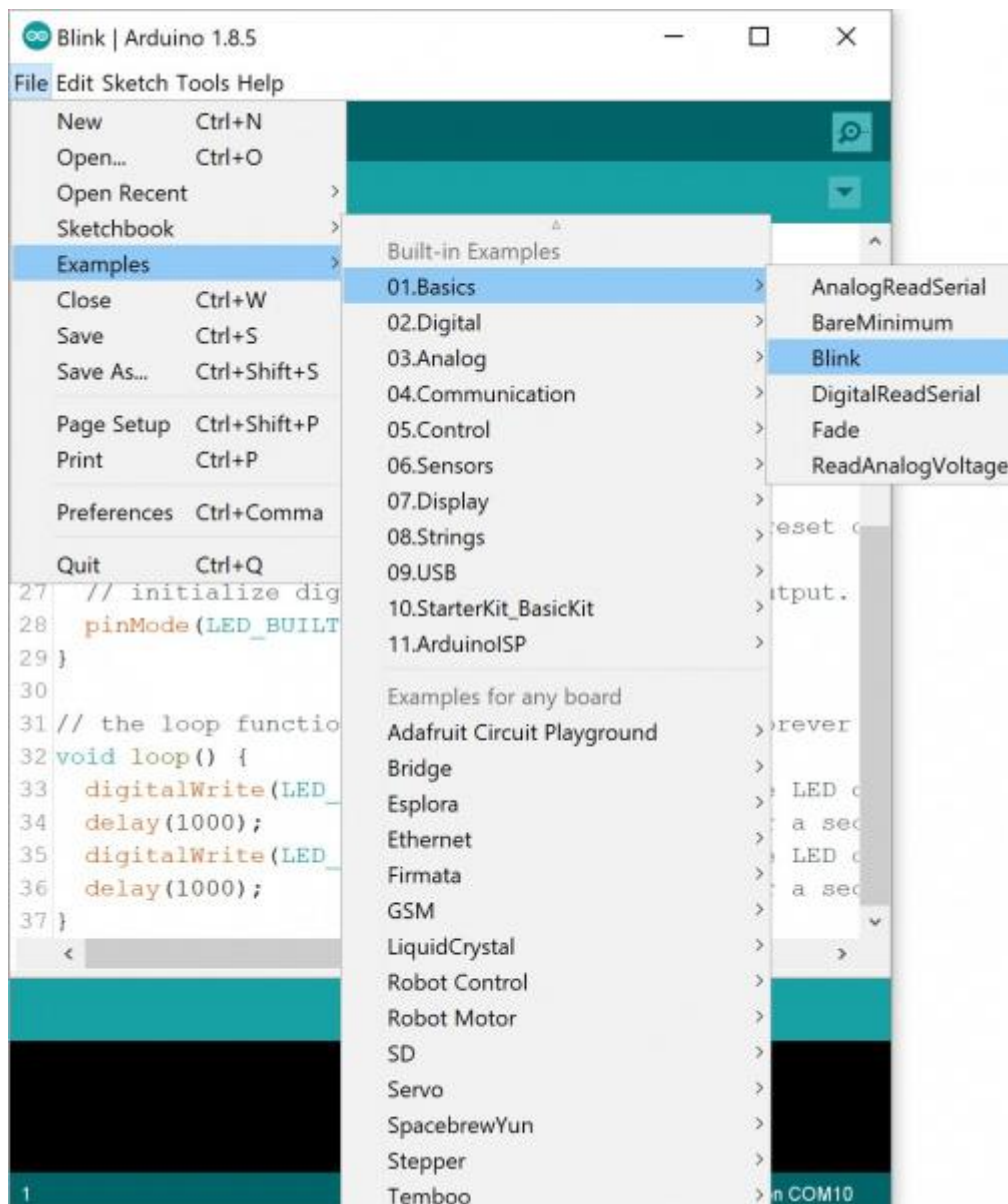
Blink!

After following the appropriate steps for your software install, we are now ready to test your first program with your Arduino board! Launch the Arduino application. If you disconnected your board, plug it back in.

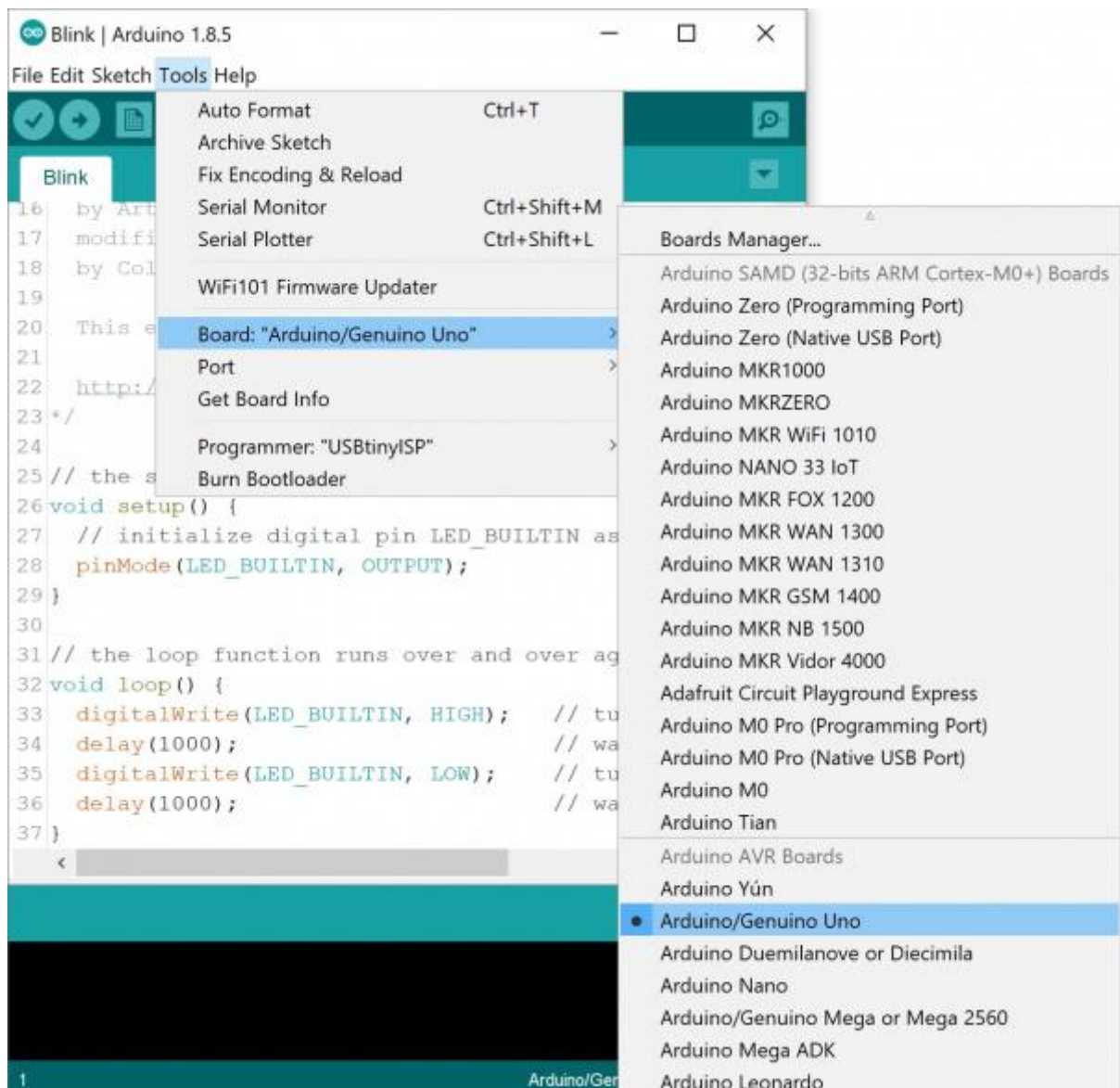


Note: Depending on your method of installing the Arduino IDE, the application may be on your desktop or the program folder.

Open the Blink example sketch by going to: File > Examples > 01.Basics > Blink.

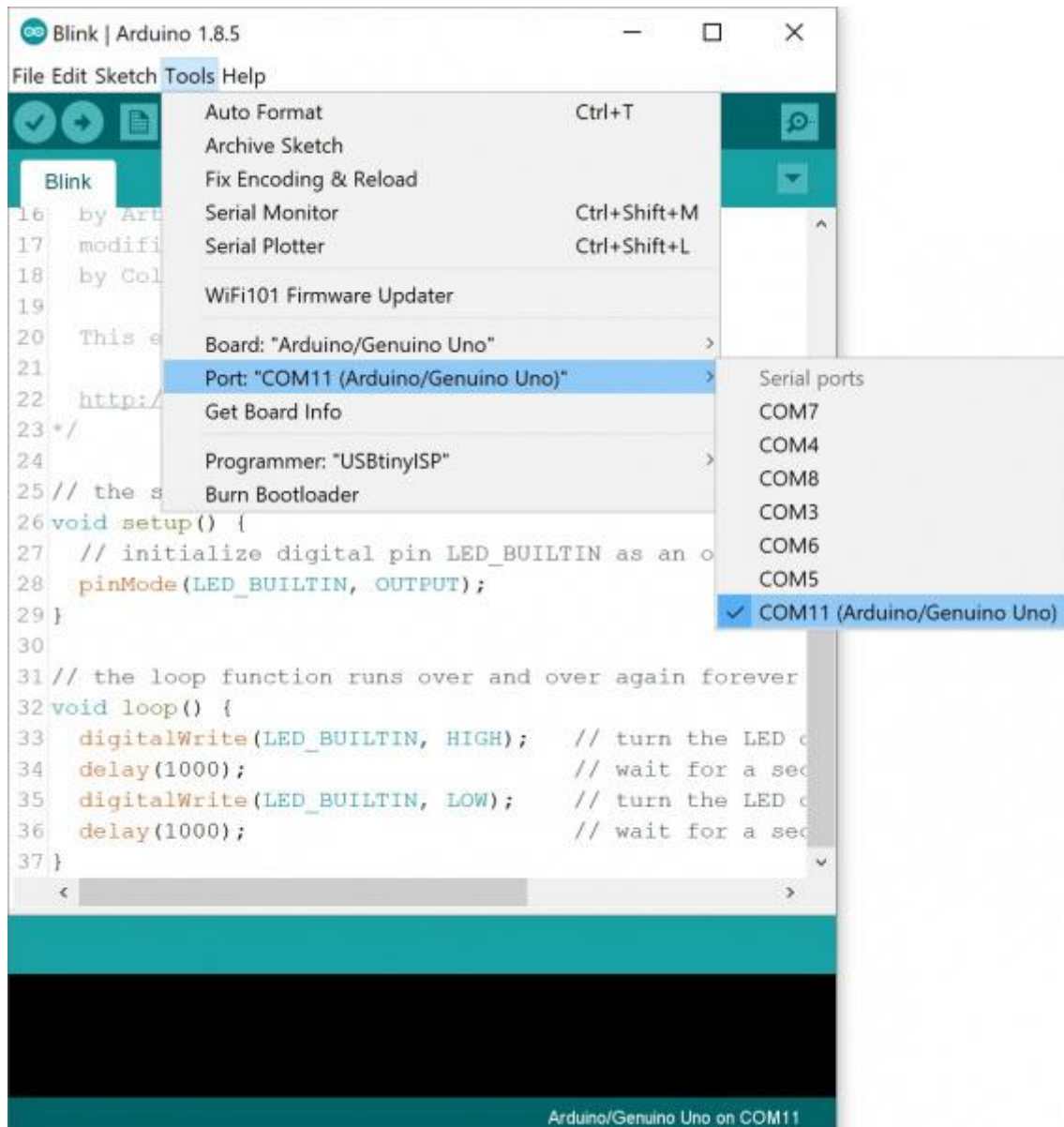


Select the type of Arduino board you're using: Tools > Board > Arduino Uno.



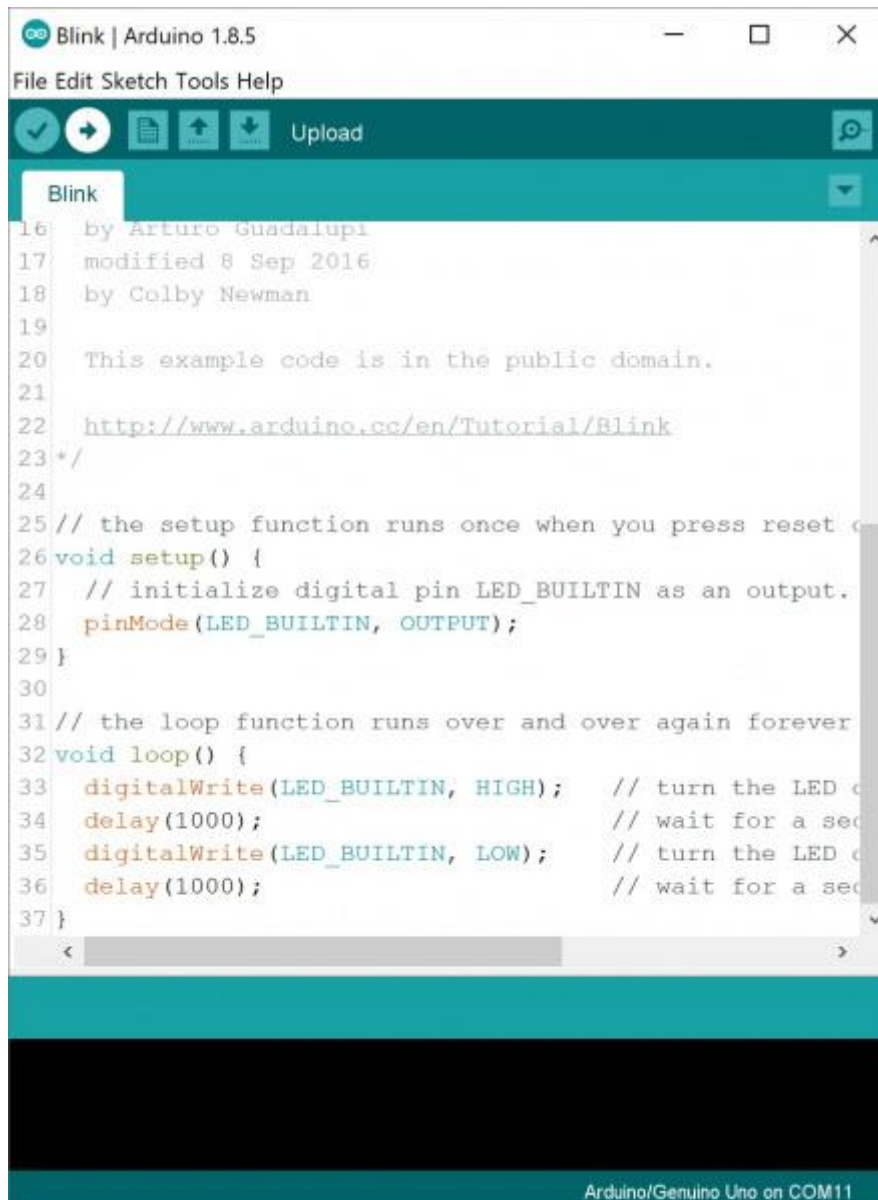
Note: As you move to other architectures, you may need to select a different board definition depending on your development board. For the Arduino Uno R3 and RedBoard development boards with ATmega328P, you can simply select **Arduino Uno**. Certain Arduino IDE versions may have you select **Arduino/Genuino Uno**.

Select the serial/COM port that your Arduino is attached to: **Tools > Port > COMxx**. In this case it was **COM11**.

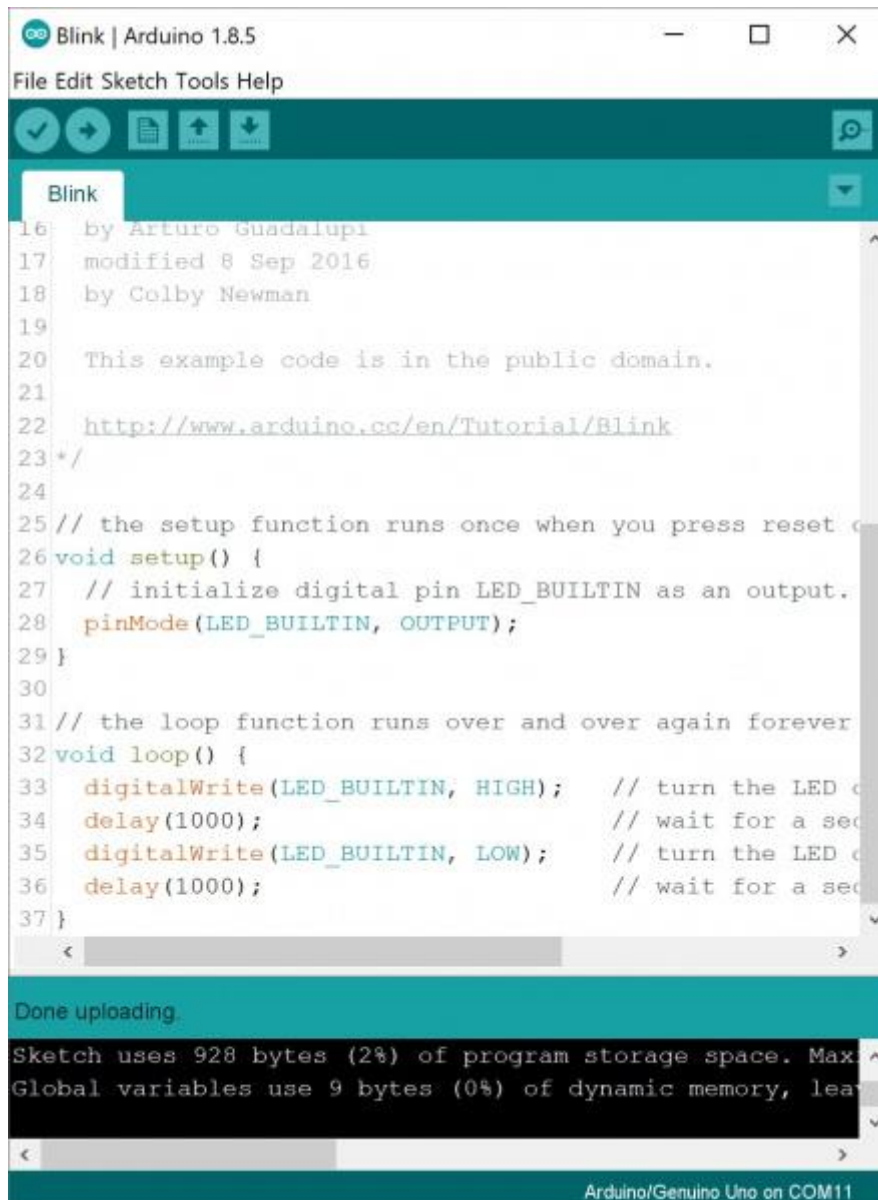


Note: If you're not sure which serial device is your Arduino, take a look at the available ports, then unplug your Arduino and look again. The one that disappeared is your Arduino.

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.



```
16  by Arturo Guadalupi
17  modified 8 Sep 2016
18  by Colby Newman
19
20  This example code is in the public domain.
21
22  http://www.arduino.cc/en/Tutorial/Blink
23  */
24
25  // the setup function runs once when you press reset
26  void setup() {
27    // initialize digital pin LED_BUILTIN as an output.
28    pinMode(LED_BUILTIN, OUTPUT);
29  }
30
31  // the loop function runs over and over again forever
32  void loop() {
33    digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the positive voltage)
34    delay(1000);                       // wait for a second
35    digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the pin LOW
36    delay(1000);                       // wait for a second
37  }
```

Done uploading.

Sketch uses 928 bytes (2%) of program storage space. Maximum is 32256 bytes.
Global variables use 9 bytes (0%) of dynamic memory, leaving 2047 bytes for local variables.
Maximum stack size is 8192 bytes (16% of 51200 bytes total).
Maximum heap size is 1024 bytes (2% of 51200 bytes total).

Arduino/Genuino Uno on COM11

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

Note: Depending on the architecture and development board, the built-in LED may be defined on a different pin. You may need to adjust `LED_BUILTIN` or pin 13 to a different value before uploading.

CHAPTER 5

RESULTS AND DISCUSSION

The hardware setup of fault detection system is shown in figure 5.1, to measure the actual distance of fault and individual switch is connected in zone to give fault in this proposed system which is sensed by relay that transfer the fault to Arduino nano and hence distance of fault is displayed by 16x2 LCD display, the fault is transfer to node mcu from Arduino nano which send data to cloud to get alerts and data to be monitored, Ubidots is the SaaS model that monitored and give alerts as shown in then figure 5.3.

Following results has been obtained by doing different case study.

Case1: if there is no fault in underground cables the display unit will display line normal.

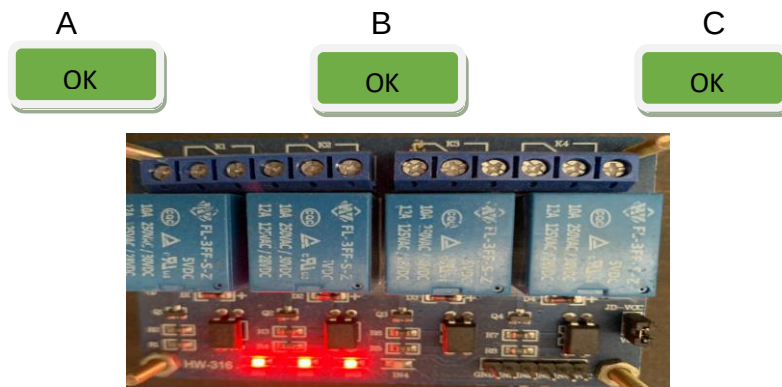


Figure 5.1.1 phase-relay indication

Case2: if there is any short circuit fault in any of 3 lines within the range of 150km then the system will alert the user and display the fault, and respective phase relay goes switch off which is indicated by LED.

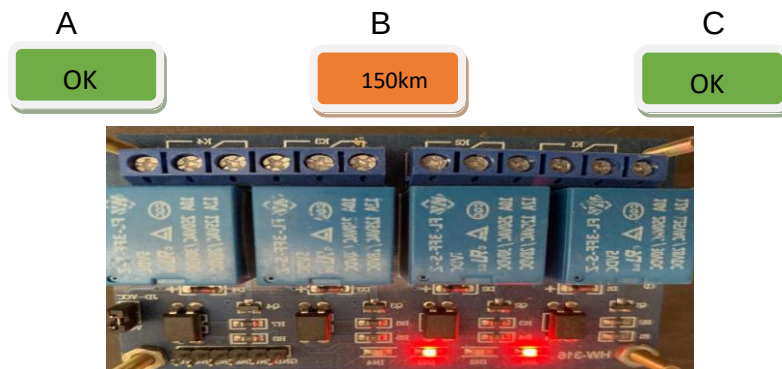


Figure 5.1.2 phase-relay indication

Here the fault is detected at B line at 150 km so user will get alert by showing kms

and line.

Case3: if there is all three-line short circuit fault with in different kms range then the system will alert the user and display all faults detected, all three phase relay goes switch off which is indicated by LED.



Figure 5.1.3 phase-relay indication

Here the fault is detected in all three lines A line at 100km, B line at 50km and C line at 150km, so displaying unit will display all 3 faults accordingly and user will get all three alerts.

Figure 5.2 shows the fault detected graph, figure 5.3 shows the measured current rating values, figure 5.4 shows monitoring of cable fault detections. The work” IOT based fault detection of Underground Cables through Node MCU” is a efficient system as it reduces the time to detect the exact location of fault.

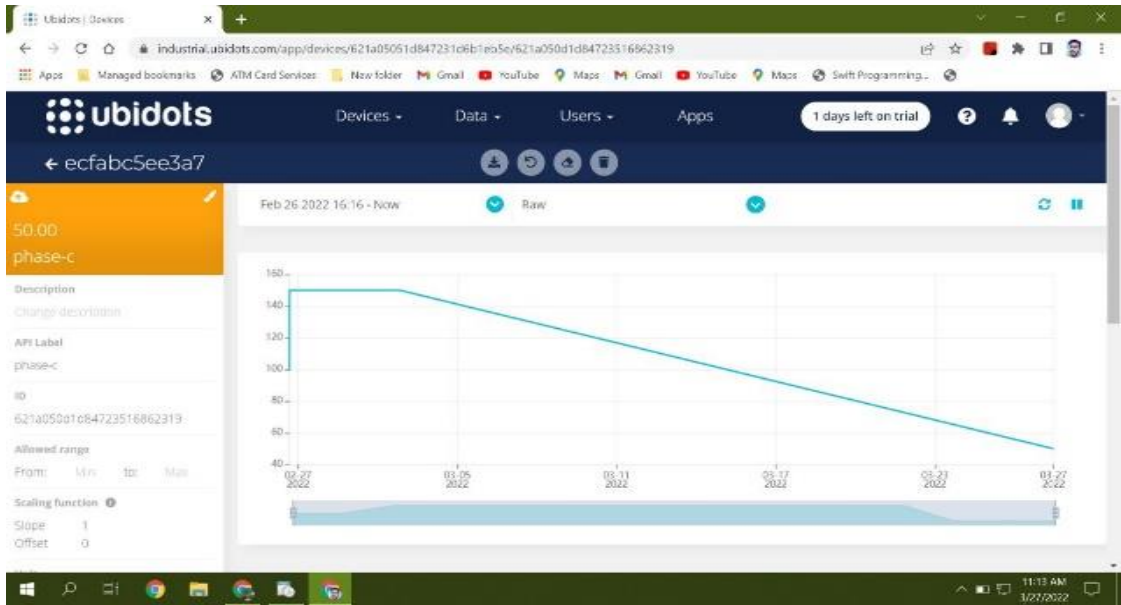


Figure 5.2 Fault detected Graph- Phase C

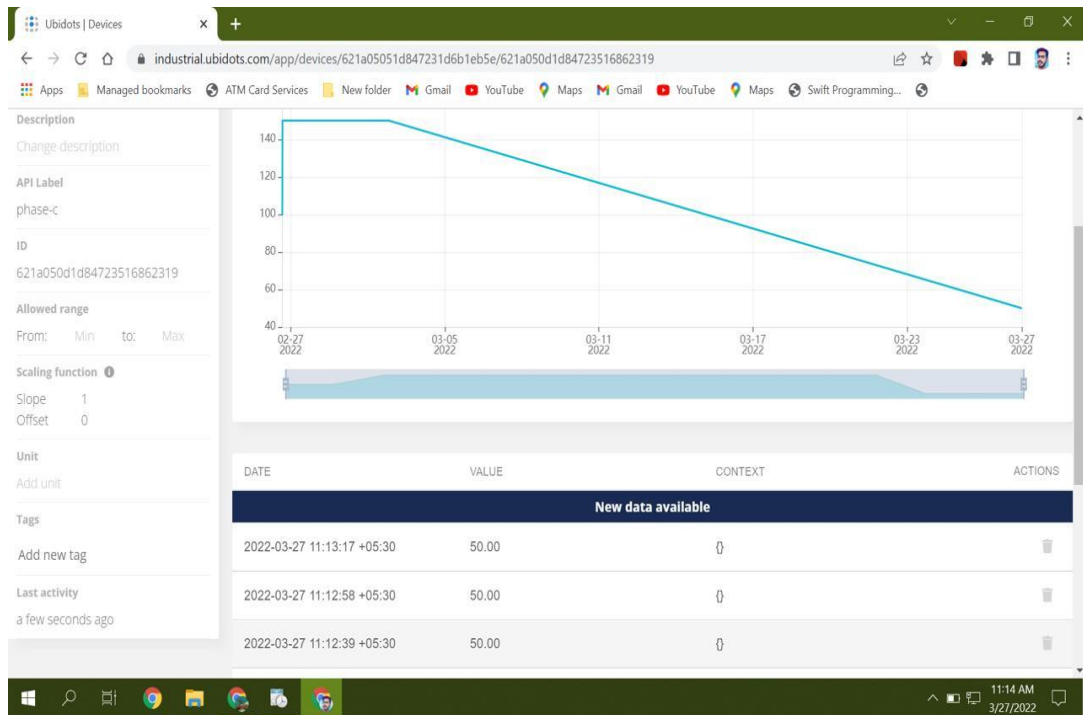


Figure 5.3 Measured Current rating values

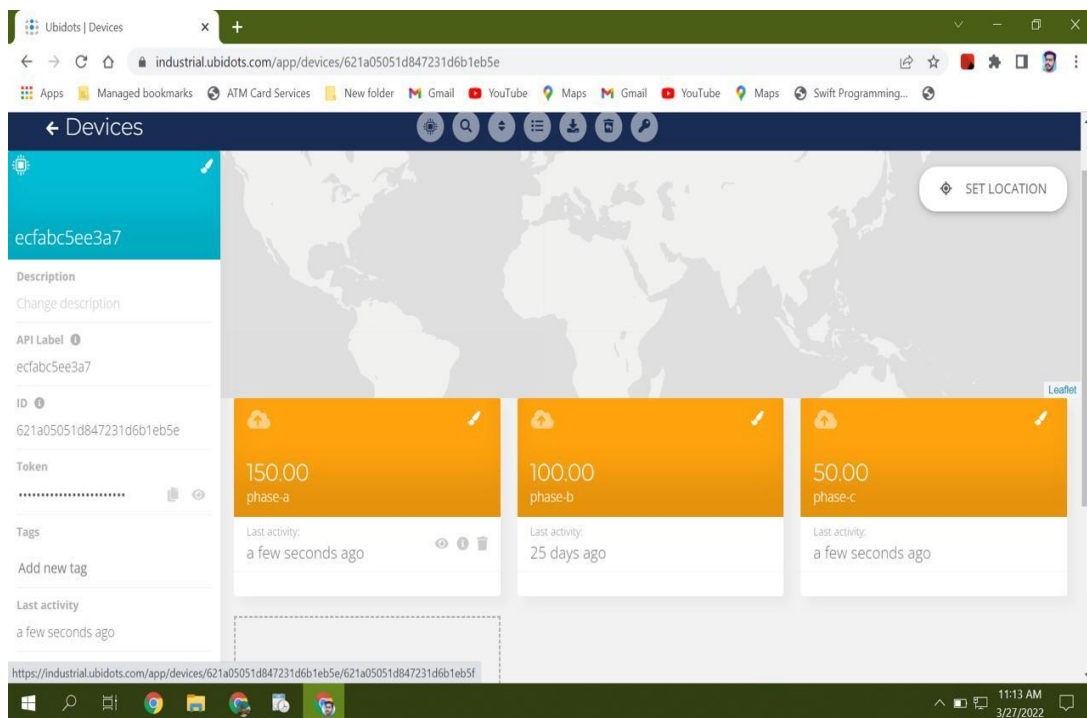


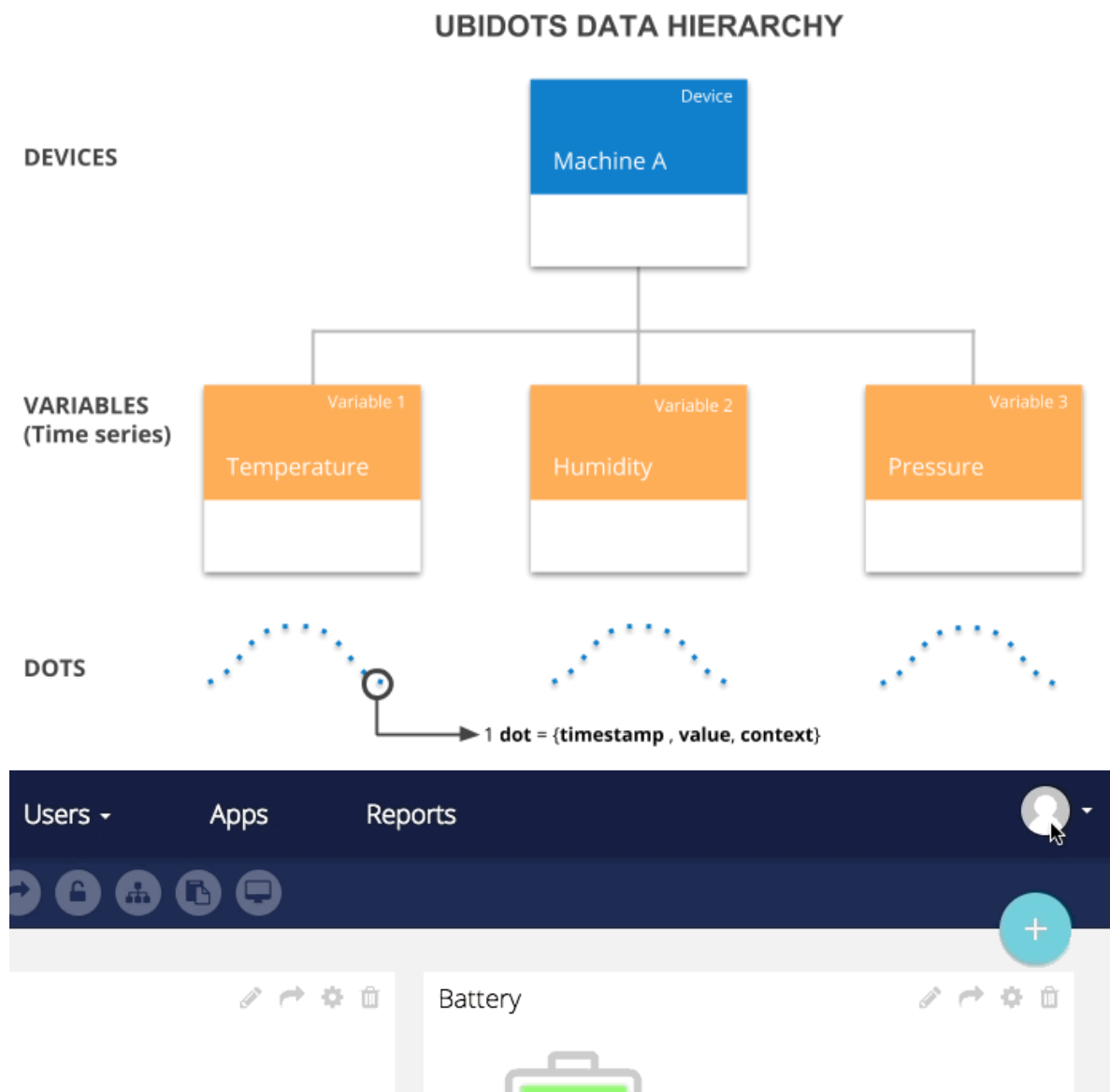
Figure 5.4 Monitoring cable faults

Welcome to the Ubidots REST API hardware docs. Here you would reference how to send or retrieve data from your hardware devices using our supported communication protocols: **HTTP**, **MQTT** and **TCP/UDP**.

The purpose of this section is to help you understand what happens in the backstage when communicating with Ubidots, so you can replicate this in your firmware. 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.

How Ubidots Works?

Every time a device updates a sensor value in a variable, a data-point or "**dot**" is created. Ubidots stores dots that come from your devices inside variables, and these stored dots have corresponding **timestamps**



CHAPTER 6

SUMMARY AND 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. To create and carry out appropriation the executives in different areas of the conveyance network by gathering sound data in the versatile and modern businesses. The motivation behind the thought is to give a steady progression of power from buyers to shoppers; Obviously and promptly foster dependability and distinguish the harm to the power, so don't foster the help for the power clients. Our progressions might mirror the insufficiency of the dam and the watch force. The shortcoming spot can be plainly and precisely recognized after the issue happens, forestalling line robbery in the heap line. This article presents a better approach for recognizing blunders and showing how they work. This article frames the most ideal ways to work on web-based execution. The proposed program was widely tried utilizing autonomous reproduction models and accomplished promising outcomes. The impacts of various frameworks and necessities are examined. Broad examination has shown that pipelines can separate between various kinds of shortcomings productively and rapidly.

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. Sathishkumar, '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.2021.075.