ACKNOWLEDGEMENT

We want to express my sincere gratitude to my guide **Prof. A. Mohan**, Professor in the Department of Electronics and Communication Engineering, for his excellent guidance and invaluable support, which helped me accomplish my Bachelor's degree and prepared me to achieve more life goals in the future. His total support of my dissertation and countless technical and professional development contributions made for a delightful and fruitful experience. Special thanks are dedicated to the discussions we had almost every working day during my project and for reviewing the dissertation.

We are very much grateful to our Project Coordinators, **Mr. G. Santosh Kumar** & **Mr. K. Anil Kumar**, Assistant Professors of ECE, GNITC, Hyderabad, who are not only shown utmost patience but were fertile in suggestions, vigilant in directions of error, and have been infinitely helpful. I am also thankful to our academic Coordinator, **Mr. D. Surendra Rao**, Associate Professor of ECE, GNITC, Hyderabad, for his dedicated support.

We are incredibly grateful to **Dr. B. Kedarnath**, Professor HOD of ECE, GNITC, Hyderabad, for being so thoughtful and helpful with precious insights and guidance during my project

We wish to express my deepest gratitude and thanks to **Dr. K. Venkata Rao**, Director, GNITC, for his constant support and encouragement and for providing me with all the facilities in the college during my project work.

Our sincere thanks to all the faculties, administrative staff, and management of GNITC, without whose support my work would always remain incomplete.

On a more personal note, we thank my **beloved parents and friends** for their moral support during my project.

ABSTRACT

As we know wireless communication systems work on antennas for reception of signals. It is necessary to properly position the antennas in the direction of transmitter for effective wireless communication. The idea is to develop an Internet of Things(IoT) based automated control system that is used to supervise the movement and an angle of a receiving antenna in all the possible directions. Here we use the NodeMCU and Arduino UNO micro-controllers to communicate across the sensors/actuators connected. It is a sensor-based system with a servo motor connected to a receiving antenna to check its direction of facing. Thus, the coordinates are sent to the end user over a secure communication channel.

If the direction of a satellite or transmitting station changes over time, the antenna direction must also be changed accordingly. The receiving antennas may be placed far apart from each other across the globe. So our system allows for antenna positioning over very long distances. The antenna positions are visible over the internet to the controlling operator via an IoT GUI. We here use web interface and/or a mobile TELNET application interface to develop the antenna monitoring GUI system. Our system allows for monitoring the antenna's direction as well as for transmitting the new coordinates to position the antenna and a motor apparently position the antenna accordingly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

The antenna is the means by which an IOT device receives and sends a signal to the outside world and therefore is the fundamental element of an IOT device. So it is very important to have proper antennas. The antenna is crucial it may make or break the communication with devices.

To increase or widen the coverage area, and thus the number of served clients, several sector antennas are installed on the same supporting structure. Once the antenna unit is attached to the supporting structure, it has to be positioned. Positioning means setting a correct direction or the right azimuth to restrict emitted energy to sub-circular arc which offers high-gain and 120-degree wireless beam performance for broad coverage. These Sector Antennas can be combined with other Sector antennas to create 360-degree area coverage for any variety of point to-multipoint scenarios.

To achieve most accurate coverage area results during installations of sector antennas, the technical team must create a coverage map for each of the sectors basing on the provided values. The technical team uses these values to make manual position adjustments of sector antennas. Manual antenna adjustments are sometimes risky in a way that unexpected accidents usually occur that even result into death. This helps the antenna to point straight towards the sending signal device so as to capture the signal. For this the system uses AVR family microcontroller and LCD screen.

The LCD screen is used to display the status of the angle of the antenna. The system makes use of stepper motor to demonstrate as the antenna motor which is used to move the antenna in proper direction. Antenna can be moved by the user commands received through the android application. User commands can then be received by the Bluetooth receiver modem. As the system receives user commands, it moves the antenna on the basis of input parameters provided. The input parameters includes number of steps i.e. the angle in which the antenna is to be moved. The whole system will be powered by the 12V transformer.

1.1 BACKGROUND OF THE STUDY

Antennas have been widely used since the turn of the last century. Ever since, this field has undergone extensive research, resulting in a wide body of experimental and theoretical knowledge alongside numerous designs and applications. The earliest antenna was introduced in the late 19th century by the German physicist Heinrich Hertz. Hertz's work was followed by a great theoretical investigation of the subject during the early to mid-20th century. This investigation proceeded with the development of computer aided design (CAD) tools during the 1970s-2000s, made possible by the development of powerful yet affordable computer technology. Antenna applications are vast and diverse. These include: television and radio broadcasting, RADAR, wireless computer communication, Bluetooth enabled devices, military personal communication, satellite communication, cell phones, RFID tags and much more. This project intends to cover the sector antenna positioning mechanism behind antennas' operation and performance and physical mechanisms governing antenna's operation, and the various parameters comprising antenna specifications.

The rigorous treatment of this subject requires an extensive mathematical background. First, an important figure of merit describing antennas radiation properties, and from which other antenna parameters are derived - the Radiation Intensity. The EM wave radiated by the antenna carries EM power. The radiated power varies in magnitude, depending on both the direction of observation and the distance from the antenna. The EM power's general pattern is maintained in the far-field, regardless of the distance from the antenna. Therefore, a normalized EM power density that will be independent of the distance from the antenna in the far-field is introduced. This is known as the radiation intensity. The radiation intensity is a mathematical description of the angular radiated power distribution in the far field (for a given polarization). Or in simpler terms - how much power is radiated by the antenna in a certain direction in the far field (using proper normalization with respect to the distance from the antenna). In order to mathematically describe the radiation intensity, a way for representing directions is defined.

Two angles with each direction that uniquely define it - an azimuth angle denoted by φ, and an elevation angle denoted by θ are associated. The elevation angle is used to describe 10 the antenna tilt relative to the horizon while the azimuth angle is used to describe the antenna

traverse in a zero tilt state. The azimuth angular separation of sector antennas is 120 degrees from each of the three sector antennas because they are directional antennas meaning that they transmit and receive signals in front of them and not behind or aside. All the three sector antennas are always installed on one mast for the purpose of collecting much more lower signals in return of the less coverage behind and on the sides.

1.2 PROBLEM STATEMENT

Sector antennas have several advantages for mobile communication system. Higher gain antenna improves the performance in the power consumption, coverage and sensitivity. However, for proper maximum coverage of a BTS in a cell, the azimuth angular separation of sector antennas has to be 120 degrees from each of the three sectors because they are directional antennas meaning that they transmit and receive signals in front of them and not behind or aside. All the three sector antennas are always installed on one mast for the purpose of collecting much more lower signals in return of the less coverage behind and on the sides.

To achieve maximum coverage during BTS installation, manual adjustments of these antennas are always made. Here one of the working team members has to stay down the tower with a compass direction to direct the one up while making adjustment in antenna positions so that they suit set angles or achieve the right azimuth. However, several problems have always been reported in the past years of BTS installation in Uganda and these include: Severe accidents that occur due to the falling of technicians from heights trying to make angular adjustments of antennas to suit calculated positions in the link budget. Inaccurate positioning of antennas that normally result into losses. Loss of money to the service providers caused by improper mounting of antennas that leads to network loss in some areas.

1.3 INTRODUCTION OF EMBEDDED SYSTEM

An Embedded System is a combination of computer hardware and software, and perhaps additional mechanical or other parts, designed to perform a specific function. A good example is the microwave oven. Almost every household has one, and tens of millions of them are used every day, but very few people realize that a processor and software are involved in the preparation of their lunch or dinner.

This is in direct contrast to the personal computer in the family room. It too is comprised of computer hardware and software and mechanical components (disk drives, for example). However, a personal computer is not designed to perform a specific function rather; it is able to do many different things. Many people use the term general-purpose computer to make this distinction clear. As shipped, a general-purpose computer is a blank slate; the manufacturer does not know what the customer will do wish it. One customer may use it for a network file server another may use it exclusively for playing games, and a third may use it to write the next great American novel.

Frequently, an embedded system is a component within some larger system. For example, modern cars and trucks contain many embedded systems. One embedded system controls the anti-lock brakes, other monitors and controls the vehicle's emissions, and a third displays information on the dashboard. In some cases, these embedded systems are connected by some sort of a communication network, but that is certainly not a requirement.

At the possible risk of confusing you, it is important to point out that a general-purpose computer is itself made up of numerous embedded systems. For example, my computer consists of a keyboard, mouse, video card, modem, hard drive, floppy drive, and sound card-each of Which is an embedded system? Each of these devices contains a processor and software and is designed to perform a specific function. For example, the modem is designed to send and receive digital data over analog telephone line. That's it and all of the other devices can be summarized in a single sentence as well.

If an embedded system is designed well, the existence of the processor and software could be completely unnoticed by the user of the device. Such is the case for a microwave oven, VCR, or alarm clock. In some cases, it would even be possible to build an equivalent device that does not contain the processor and software. This could be done by replacing the combination with a custom integrated circuit that performs the same functions in hardware. However, a lot of flexibility is lost when a design is hard-cooled in this way. It is mush easier, and cheaper, to change a few lines of software than to redesign a piece of custom hardware.

1.3.1. HISTORY AND FUTURE

Given the definition of embedded systems earlier is this chapter; the first such systems could not possibly have appeared before 1971. That was the year Intel introduced the world's first microprocessor. This chip, the 4004, was designed for use in a line of business calculators produced by the Japanese Company Busicom. In 1969, Busicom asked Intel to design a set of custom integrated circuits-one for each of their new calculator models. The 4004 was Intel's response rather than design custom hardware for each calculator, Intel proposed a general-purpose circuit that could be used throughout the entire line of calculators. Intel's idea was that the software would give each calculator its unique set of features.

The microcontroller was an overnight success, and its use increased steadily over the next decade. Early embedded applications included unmanned space probes, computerized traffic lights, and aircraft flight control systems. In the 1980s, embedded systems quietly rode the waves of the microcomputer age and brought microprocessors into every part of our kitchens (bread machines, food processors, and microwave ovens), living rooms (televisions, stereos, and remote controls), and workplaces (fax machines, pagers, laser printers, cash registers, and credit card readers).

It seems inevitable hat the number of embedded systems will continue to increase rapidly. Already there are promising new embedded devices that have enormous market potential; light switches and thermostats that can be central computer, intelligent air-bag systems that don't inflate when children or small adults are present, pal-sized electronic organizers and personal digital assistants (PDAs), digital cameras, and dashboard navigation systems. Clearly, individuals who possess the skills and desire to design the next generation of embedded systems will be in demand for quite some time.

1.3.2. REAL TIME SYSTEMS

One subclass of embedded is worthy of an introduction at this point. As commonly defined, a real-time system is a computer system that has timing constraints. In other words, a real-time system is partly specified in terms of its ability to make certain calculations or decisions in a timely manner. These important calculations are said to have deadlines for completion. And, for all practical purposes, a missed deadline is just as bad as a wrong answer.

The issue of what if a deadline is missed is a crucial one. For example, if the real-time system is part of an airplane's flight control system, it is possible for the lives of the passengers and crew to be endangered by a single missed deadline. However, if instead the system is involved in satellite communication, the damage could be limited to a single corrupt data packet. The more severe the consequences, the more likely it will be said that the deadline is ”hard” and thus, the system is a hard real-time system. Real-time systems at the other end of this discussion are said to have ”soft” deadlines.

All of the topics and examples presented in this book are applicable to the designers of real-time system who is more delight in his work. He must guarantee reliable operation of the software and hardware under all the possible conditions and to the degree that human lives depend upon three system's proper execution, engineering calculations and descriptive paperwork.

APPLICATION AREAS

Nearly 99 per cent of the processors manufactured end up in embedded systems. The embedded system market is one of the highest growth areas as these systems are used in very market segment- consumer electronics, office automation, industrial automation, biomedical engineering, wireless communication,Data communication, telecommunications, transportation, military and so on.

CONSUMER APPLIANCES

At home we use a number of embedded systems which include digital camera, digital diary, DVD player, electronic toys, microwave oven, remote controls for TV and air-conditioner, VCO player, video game consoles, video recorders etc. Today's high-tech car has about 20 embedded systems for transmission control, engine spark control, air-conditioning, navigation etc. Even wristwatches are now becoming embedded systems. The palmtops are powerful embedded systems using which we can carry out many general-purpose tasks such as playing games and word processing.

OFFICE AUTOMATION

The office automation products using em embedded systems are copying machine, fax ma-

chine, key telephone, modem, printer, scanner etc.

INDUSTRIAL AUTOMATION

Today a lot of industries use embedded systems for process control. These include pharmaceutical, cement, sugar, oil exploration, nuclear energy, electricity generation and transmission. The embedded systems for industrial use are designed to carry out specific tasks such as monitoring the temperature, pressure, humidity, voltage, current etc., and then take appropriate action based on the monitored levels to control other devices or to send information to a centralized monitoring station. In hazardous industrial environment, where human presence has to be avoided, robots are used, which are programmed to do specific jobs.

The robots are now becoming very powerful and carry out many interesting and complicated tasks such as hardware assembly.

MEDICAL ELECTRONICS

Almost every medical equipment in the hospital is an embedded system. These equipments include diagnostic aids such as ECG, EEG, blood pressure measuring devices, X-ray scanners; equipment used in blood analysis, radiation, colonoscopy, endoscopy etc. Developments in medical electronics have paved way for more accurate diagnosis of diseases.

COMPUTER NETWORKING

Computer networking products such as bridges, routers, Integrated Services Digital Networks (ISDN), Asynchronous Transfer Mode (ATM), X.25 and frame relay switches are embedded systems which implement the necessary data communication protocols. For example, a router interconnects two networks. The two networks may be running different protocol stacks. The router's function is to obtain the data packets from incoming pores, analyze the packets and send them towards the destination after doing necessary protocol conversion. Most networking equipments, other than the end systems (desktop computers) we use to access the networks, are embedded systems.

TELECOMMUNICATIONS

In the field of telecommunications, the embedded systems can be categorized as subscriber

terminals and network equipment. The subscriber terminals such as key telephones, ISDN phones, terminal adapters, web cameras are embedded systems. The network equipment includes multiplexers, multiple access systems, Packet Assemblers Dissemblers (PADs), sate11ite modems etc. IP phone, IP gateway, IP gatekeeper etc. are the latest embedded systems that provide very low-cost voice communication over the Internet.

WIRELESS TECHNOLOGIES

Advances in mobile communications are paving way for many interesting applications using embedded systems. The mobile phone is one of the marvels of the last decade of the 20'h century. It is a very powerful embedded system that provides voice communication while we are on the move. The Personal Digital Assistants and the palmtops can now be used to access multimedia services over the Internet. Mobile communication infrastructure such as base station controllers, mobile switching centers are also powerful embedded systems.

INSEMINATION

Testing and measurement are the fundamental requirements in all scientific and engineering activities. The measuring equipment we use in laboratories to measure parameters such as weight, temperature, pressure, humidity, voltage, current etc. are all embedded systems. Test equipment such as oscilloscope, spectrum analyzer, logic analyzer, protocol analyzer, radio communication test set etc. are embedded systems built around powerful processors. Thank to miniaturization, the test and measuring equipment are now becoming portable facilitating easy testing and measurement in the field by field-personnel.

SECURITY

Security of persons and information has always been a major issue. We need to protect our homes and offices; and also the information we transmit and store. Developing embedded systems for security applications is one of the most lucrative businesses nowadays. Security devices at homes, offices, airports etc. for authentication and verification are embedded systems. Encryption devices are nearly 99 per cent of the processors that are manufactured end up in embedded systems. Embedded systems find applications in. every industrial segment-consumer electronics, transportation, avionics, biomedical engineering, manufacturing, process control and industrial automation, data communication, telecommunication, defense,

security etc. Used to encrypt the data/voice being transmitted on communication links such as telephone lines.Biometric systems using fingerprint and face recognition are now being extensively used for user authentication in banking applications as well as for access control in high security buildings.

FINANCE

Financial dealing through cash and cheques are now slowly paving way for transactions using smart cards and ATM (Automatic Teller Machine, also expanded as Any Time Money) machines. Smart card, of the size of a credit card, has a small micro-controller and memory; and it interacts with the smart card reader! ATM machine and acts as an electronic wallet. Smart card technology has the capability of ushering in a cashless society. Well, the list goes on. It is no exaggeration to say that eyes wherever you go, you can see, or at least feel, the work of an embedded system.

1.3.3. OVERVIEW OF EMBEDDED SYSTEM ARCHITECTURE

Every embedded system consists of custom-built hardware built around a Central Processing Unit (CPU). This hardware also contains memory chips onto which the software is loaded. The software residing on the memory chip is also called the ‘firmware’. The embedded system architecture can be represented as a layered architecture as shown in Fig.

The operating system runs above the hardware, and the application software runs above the operating system. The same architecture is applicable to any computer including a desktop computer. However, there are significant differences. It is not compulsory to have an operating system in every embedded system.

For small appliances such as remote control units, air conditioners, toys etc., there is no need for an operating system and you can write only the software specific to that application. For applications involving complex processing, it is advisable to have an operating system. In such a case, you need to integrate the application software with the operating system and then transfer the entire software on to the memory chip. Once the software is transferred to the memory chip, the software will continue to run fora long time you don’t need to reload new software.

Now, let us see the details of the various building blocks of the hardware of an embedded system. As shown in Fig. the building blocks are:

- Central Processing Unit (CPU)
- Memory (Read-only Memory and Random Access Memory)
- Input Devices
- Output devices
- Communication interfaces
- Application-specific circuitry

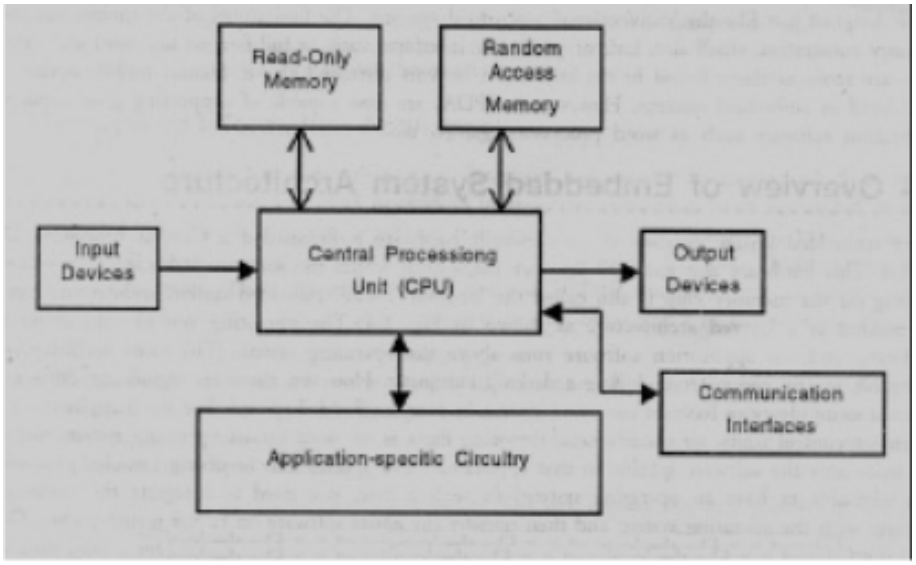

Figure 1.1:: Blocks of hardware embedded system

CENTRAL PROCESSING UNIT (CPU)

The Central Processing Unit (processor, in short) can be any of the following: microcontroller, microprocessor or Digital Signal Processor (DSP). A micro-controller is a low-cost processor. Its main attraction is that on the chip itself, there will be many other components such as memory, serial communication interface, analog-to digital converter etc. So, for small applications, a micro-controller is the best choice as the number of external components required will be very less. On the other hand, microprocessors are more powerful, but you need to use many external components with them. D5P is used mainly for applications

in which signal processing is involved such as audio and video processing.

MEMORY

The memory is categorized as Random Access 11emory (RAM) and Read Only Memory (ROM). The contents of the RAM will be erased if power is switched off to the chip, whereas ROM retains the contents even if the power is switched off. So, the firmware is stored in the ROM. When power is switched on, the processor reads the ROM; the program is program is executed.

INPUT DEVICES

Unlike the desktops, the input devices to an embedded system have very limited capability. There will be no keyboard or a mouse, and hence interacting with the embedded system is no easy task. Many embedded systems will have a small keypad-you press one key to give a specific command. A keypad may be used to input only the digits. Many embedded systems used in process control do not have any input device foruser interaction; they take inputs fromsensors or transducers 1'fnd produce electrical signals that are in turn fed to other systems.

OUTPUT DEVICES

The output devices of the embedded systems also have very limited capability. Some embedded systems will have a few Light Emitting Diodes (LEDs) to indicate the health status of the system modules, or for visual indication of alarms. A small Liquid Crystal Display (LCD) may also be used to display some important parameters.

COMMUNICATION INTERFACES

The embedded systems may need to, interact with other embedded systems at they may have to transmit data to a desktop. To facilitate this, the embedded systems are provided with one or a few communication interfaces such as RS232, RS422, RS485, Universal Serial Bus (USB), IEEE 1394, Ethernet etc.

CHAPTER-2
MATERIALS

2.1 INTRODUCTION TO IOT

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

A thing in the internet of things can be a person with a heart monitor implant, a farm animal with a biochip transponder, an automobile that has built-in sensors to alert the driver when tire pressure is low or any other natural or man-made object that can be assigned an Internet Protocol (IP) address and is able to transfer data over a network. Increasingly, organizations in a variety of industries are using IoT to operate more efficiently, better understand customers to deliver enhanced customer service, improve decision-making and increase the value of the business.

2.1.1. HOW DOES IOT WORK?

An IoT ecosystem consists of web-enabled smart devices that use embedded systems, such as processors, sensors and communication hardware, to collect, send and act on data they acquire from their environments.

IoT devices share the sensor data they collect by connecting to an IoT gateway or other edge device where data is either sent to the cloud to be analyzed or analyzed locally. Sometimes, these devices communicate with other related devices and act on the information they get from one another. The devices do most of the work without human intervention, although people can interact with the devices – for instance, to set them up, give them instructions or access the data.

The connectivity, networking and communication protocols used with these web-enabled devices largely depend on the specific IoT applications deployed.

IoT can also make use of artificial intelligence (AI) and machine learning to aid in making data collecting processes easier and more dynamic.

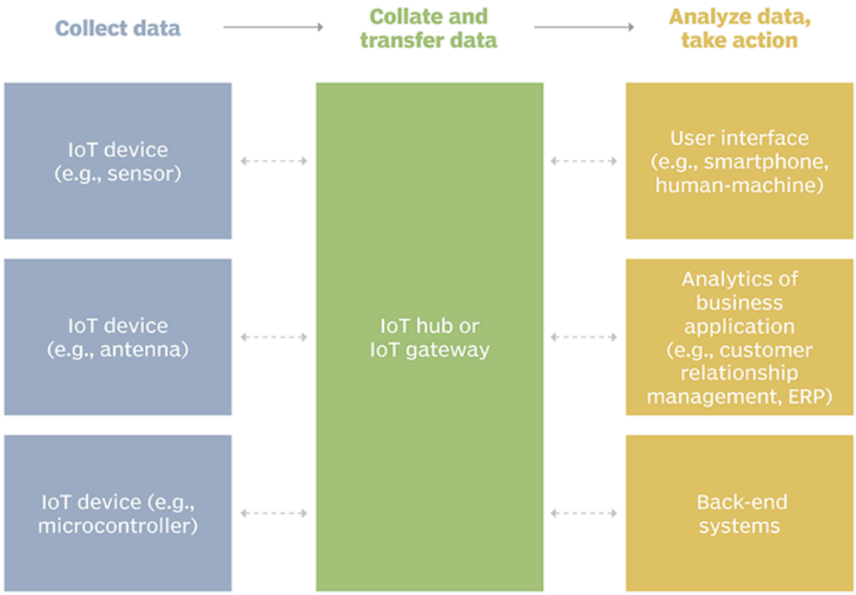

Figure 2.1:: Architecture of an IoT System

Fixed wing drones as the name suggests have two fixed wings along the lateral axis of the aircraft. Fixed wing UAVs consists of a rigid wing that has an aerofoil shape which make flight capable by generating lift caused by the UAV's forward airspeed. This airspeed is generated by forward thrust usually by the means of a propeller (explained in later modules) being turned by an internal combustion engine or electric motor.

2.1.2. IMPORTANCE OF IOT

The internet of things helps people live and work smarter, as well as gain complete control over their lives. In addition to offering smart devices to automate homes, IoT is essential to business. IoT provides businesses with a real-time look into how their systems really work, delivering insights into everything from the performance of machines to supply chain and logistics operations.

IoT enables companies to automate processes and reduce labor costs. It also cuts down on waste and improves service delivery, making it less expensive to manufacture and deliver

goods, as well as offering transparency into customer transactions.

As such, IoT is one of the most important technologies of everyday life, and it will continue to pick up steam as more businesses realize the potential of connected devices to keep them competitive.

2.1.3. BENEFITS OF IOT TO ORGANIZATION

The internet of things offers several benefits to organizations. Some benefits are industry specific, and some are applicable across multiple industries. Some of the common benefits of IoT enable businesses to:

- Monitor their overall business processes;
- Improve the customer experience (CX);
- Save time and money;
- Enhance employee productivity;
- Integrate and adapt business models;
- Make better business decisions; and
- Generate more revenue.

IoT encourages companies to rethink the ways they approach their businesses and gives them the tools to improve their business strategies.

Generally, IoT is most abundant in manufacturing, transportation and utility organizations, making use of sensors and other IoT devices; however, it has also found use cases for organizations within the agriculture, infrastructure and home automation industries, leading some organizations toward digital transformation.

IoT can benefit farmers in agriculture by making their job easier. Sensors can collect data on rainfall, humidity, temperature and soil content, as well as other factors, that would help automate farming techniques.

The ability to monitor operations surrounding infrastructure is also a factor that IoT can help

with. Sensors, for example, could be used to monitor events or changes within structural buildings, bridges and other infrastructure. This brings benefits with it, such as cost saving, saved time, quality-of-life workflow changes and paperless workflow.

A home automation business can utilize IoT to monitor and manipulate mechanical and electrical systems in a building. On a broader scale, smart cities can help citizens reduce waste and energy consumption.

IoT touches every industry, including businesses within healthcare, finance, retail and manufacturing.

Control of the UAV comes from control surfaces built into the wing itself, these traditionally consist of ailerons an elevator and a rudder. They allow the UAV to freely rotate around three axes that are perpendicular to each other and intersect at the UAV's centre of gravity.

2.1.4. PROS & CONS OF IOT

Some of the advantages of IoT include the following:

1. Ability to access information from anywhere at any time on any device;
2. Improved communication between connected electronic devices;
3. Transferring data packets over a connected network saving time and money; and
4. Automating tasks helping to improve the quality of a business's services and
5. Reducing the need for human intervention.

Some disadvantages of IoT include the following:

1. As the number of connected devices increases and more information is shared between devices, the potential that a hacker could steal confidential information also increases.
2. Enterprises may eventually have to deal with massive numbers – maybe even millions – of IoT devices, and collecting and managing the data from all those devices will be challenging.
3. If there's a bug in the system, it's likely that every connected device will become cor-

rupted.

4. Since there's no international standard of compatibility for IoT, it's difficult for devices from different manufacturers to communicate with each other.

2.2 IOT STANDARDS AND FRAMEWORKS

There are several emerging IoT standards, including the following:

- IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) is an open standard defined by the Internet Engineering Task Force (IETF). The 6LoWPAN standard enables any low-power radio to communicate to the internet, including 804.15.4, Bluetooth Low Energy (BLE) and Z-Wave (for home automation).
- ZigBee is a low-power, low-data rate wireless network used mainly in industrial settings. ZigBee is based on the Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard. The ZigBee Alliance created Dotdot, the universal language for IoT that enables smart objects to work securely on any network and understand each other.
- LiteOS is a Unix-like operating system (OS) for wireless sensor networks. LiteOS supports smartphones, wearables, intelligent manufacturing applications, smart homes and the internet of vehicles (IoV). The OS also serves as a smart device development platform.
- OneM2M is a machine-to-machine service layer that can be embedded in software and hardware to connect devices. The global standardization body, OneM2M, was created to develop reusable standards to enable IoT applications across different verticals to communicate.
- Data Distribution Service (DDS) was developed by the Object Management Group (OMG) and is an IoT standard for real-time, scalable and high-performance M2M communication.
- Advanced Message Queuing Protocol (AMQP) is an open source published standard for asynchronous messaging by wire. AMQP enables encrypted and interoperable mes-

saging between organizations and applications. The protocol is used in client-server messaging and in IoT device management.

- Constrained Application Protocol (CoAP) is a protocol designed by the IETF that specifies how low-power, compute-constrained devices can operate in the internet of things.
- Long Range Wide Area Network (LoRaWAN) is a protocol for WANs designed to support huge networks, such as smart cities, with millions of low-power devices.

IoT frameworks include the following:

- Amazon Web Services (AWS) IoT is a cloud computing platform for IoT released by Amazon. This framework is designed to enable smart devices to easily connect and securely interact with the AWS cloud and other connected devices.
- Arm Mbed IoT is a platform to develop apps for IoT based on Arm microcontrollers. The goal of the Arm Mbed IoT platform is to provide a scalable, connected and secure environment for IoT devices by integrating Mbed tools and services.
- Microsoft's Azure IoT Suite is a platform that consists of a set of services that enables users to interact with and receive data from their IoT devices, as well as perform various operations over data, such as multidimensional analysis, transformation and aggregation, and visualize those operations in a way that's suitable for business.
- Google's Brillo/Weave is a platform for the rapid implementation of IoT applications. The platform consists of two main backbones: Brillo, an Android-based OS for the development of embedded low-power devices, and Weave, an IoT-oriented communication protocol that serves as the communication language between the device and the cloud.
- Calvin is an open source IoT platform released by Ericsson designed for building and managing distributed applications that enable devices to talk to each other. Calvin includes a development framework for application developers, as well as a runtime environment for handling the running application.

2.3 NODEMCU

NodeMCU is a low-cost open source IoT platform. It initially included firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which was based on the ESP-12 module. Later, support for the ESP32 32-bit MCU was added.

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name ”NodeMCU” combines ”node” and ”MCU” (micro-controller unit). The term ”NodeMCU” strictly speaking refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are open source.

Figure 2.2:: NodeMCU

The firmware uses the Lua scripting language. The firmware 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. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

The prototyping hardware typically used is a circuit board functioning as a dual in-line package (DIP) which integrates a USB controller with a smaller surface-mounted board containing the MCU and antenna. The choice of the DIP format allows for easy prototyping on breadboards. The design was initially based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated with a Tensilica Xtensa LX106 core, widely used in IoT applications.

2.3.1. TECHNICAL SPECIFICATIONS

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna
- Small Sized module to fit smartly inside your IoT projects

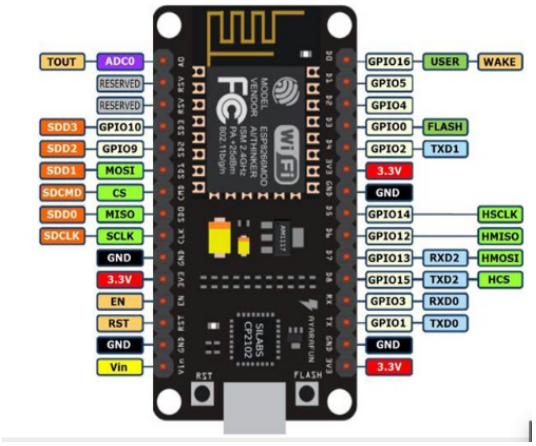

Figure 2.3:: NodeMCU pinout

2.4 ARDUINO UNO

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

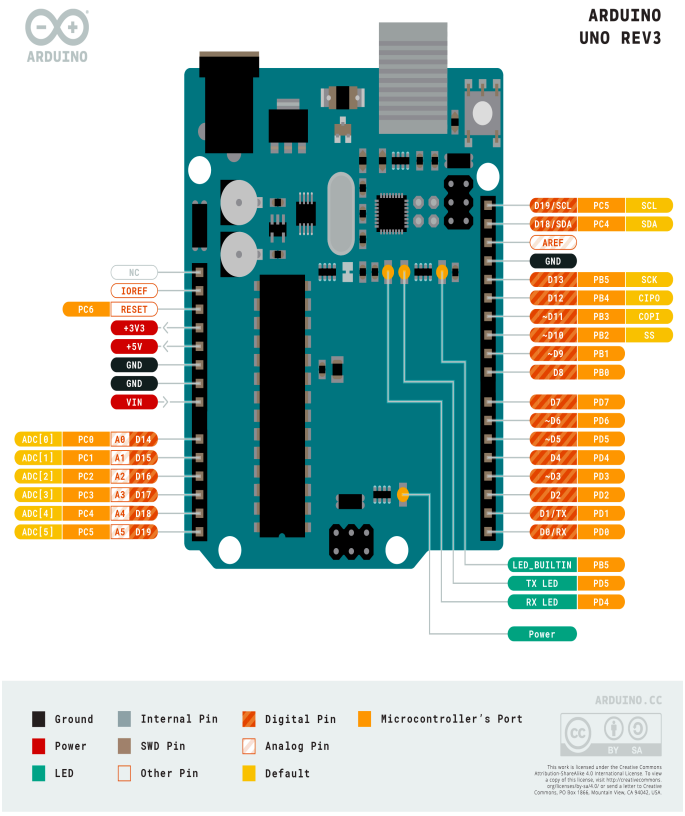

Figure 2.4:: Arduino UNO pinout

The word ”uno” means ”one” in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version

1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

2.4.1. TECHNICAL SPECIFICATIONS

MICROCONTROLLER	ATmega328P
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	14 (of which 6 provide PWM output)
PWM DIGITAL I/O PINS	6
ANALOG INPUT PINS	6
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	68.6 mm
WIDTH	53.4 mm
WEIGHT	25 g

Figure 2.5:: Technical Specs of Arduino ATmega328P

2.5 RF ANTENNA

The 433 MHz RF transmitter and receiver module is a pair of small RF (i.e. radio-frequency) electronic modules used to send and receive radio signals between any two devices. The transmitter module sends the data from the transmitter end and the Receiver module receives that data at the receiver's end.

Figure 2.6:: Radio Frequency Transmitter & Receiver

2.5.1. RF TRANSMITTER MODULE PINOUT

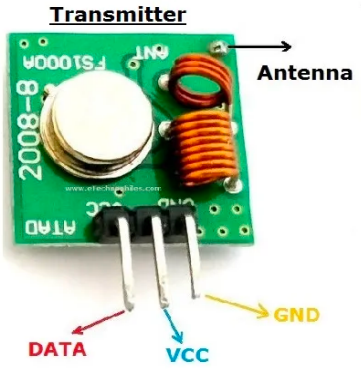

Figure 2.7:: RF transmitter pin configuration.

PIN name	Description
VCC	Used to power the RF transmitter module.
GND	Ground pin of the module. Connect it with the controllers and encoder/decoders GND pin.
DATA	This is the data pin of the transmitter. It takes the data from the microcontroller or encoder and broadcast it via the antenna.
ANT	The antenna pin is not necessary to use but it is recommended. The module can only operate max 3 meters without an antenna but its range can be extendable up to 100 meters by using a small hookup wire as an Antenna.

Table 2.1:: Pin description of RF transmitter

2.5.2. FEATURES OF RF TRANSMITTER

- The Transmitter offers only one-way communication through 433.92MHz frequency at 1Kb data rate.
- It operates at a range of 3-12V which is also the power operating volts of most of the microcontrollers and boards.
- The module uses the ASK (Amplitude Shift Key) modulation method to transmits the data.
- It is one of the very low-cost power effective modules for both commercial, hobbyist, and developers.
- 433MHz Transmitter is one of the cheapest RF transmitters and it has a lot of applications and can be used interface with almost every microcontroller.

2.5.3. RF RECEIVER MODULE PINOUT

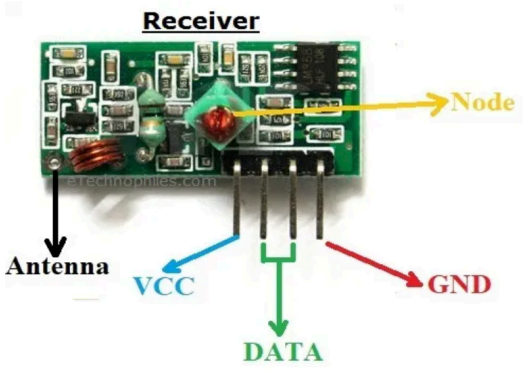

Figure 2.8:: RF receiver pin configuration.

PIN name	Description
VCC	Used to power the RF receiver module.
GND	Ground pin of the module. Connect it with the controllers and encoder/decoders GND pin.
DATA	These pins output the digital data received. The two center pins are internally connected, so we can use either one of them for data output.
ANT	The antenna pin is not necessary to use but it is recommended. The module can only operate max 3 meters without an antenna but its range can be extendable up to 100 meters by using a small hookup wire as an Antenna.

Table 2.2:: Pin description of RF receiver

2.5.4. FEATURES OF RF RECEIVER

- The RF receiver delivers the output to the data pin in an encoded form.
- The operational voltage range of the module is 5V maximum.
- The frequency of the receiver can be changed using a node present on it.

- It is one of the popular and cheapest receivers and has low power consumption.
- 433MHz RF receiver module uses the ASK signal as an input.

2.6 SERVO MOTOR

A servo motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a servo mechanism.

Figure 2.9:: Servo Motor.

If motor is powered by a DC power supply then it is called DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the DC servo motor working. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy car, RC helicopters and planes, Robotics, etc.

Servo motors are rated in kg/cm (kilogram per centimeter) most hobby servo motors are rated at 3kg/cm or 6kg/cm or 12kg/cm. This kg/cm tells you how much weight your servo motor can lift at a particular distance. For example: A 6kg/cm Servo motor should be able to lift 6kg if the load is suspended 1cm away from the motors shaft, the greater the distance the

lesser the weight carrying capacity. The position of a servo motor is decided by electrical pulse and its circuitry is placed beside the motor.

2.6.1. MECHANISM

It consists of three parts:

1. Controlled device
2. Output sensor
3. Feedback system

It is a closed-loop system where it uses a positive feedback system to control motion and the final position of the shaft. Here the device is controlled by a feedback signal generated by comparing output signal and reference input signal.

Here reference input signal is compared to the reference output signal and the third signal is produced by the feedback system. And this third signal acts as an input signal to the control the device. This signal is present as long as the feedback signal is generated or there is a difference between the reference input signal and reference output signal. So the main task of servomechanism is to maintain the output of a system at the desired value at presence of noises.

2.6.2. WORKING PRINCIPLE

A servo consists of a Motor (DC or AC), a potentiometer, gear assembly, and a controlling circuit. First of all, we use gear assembly to reduce RPM and to increase torque of the motor. Say at initial position of servo motor shaft, the position of the potentiometer knob is such that there is no electrical signal generated at the output port of the potentiometer.

Now an electrical signal is given to another input terminal of the error detector amplifier. Now the difference between these two signals, one comes from the potentiometer and another comes from other sources, will be processed in a feedback mechanism and output will be provided in terms of error signal. This error signal acts as the input for motor and motor starts rotating.

Now motor shaft is connected with the potentiometer and as the motor rotates so the potentiometer and it will generate a signal. So as the potentiometer's angular position changes, its output feedback signal changes. After sometime the position of potentiometer reaches at a position that the output of potentiometer is same as external signal provided.

At this condition, there will be no output signal from the amplifier to the motor input as there is no difference between external applied signal and the signal generated at potentiometer, and in this situation motor stops rotating.

2.7 LCD DISPLAY

LCD (Liquid Crystal Display) is a type of flat panel display which uses liquid crystals in its primary form of operation. LEDs have a large and varying set of use cases for consumers and businesses, as they can be commonly found in smartphones, televisions, computer monitors and instrument panels.

Figure 2.10:: Liquid Crystal Display.

LCDs were a big leap in terms of the technology they replaced, which include light-emitting diode (LED) and gas-plasma displays. LCDs allowed displays to be much thinner than cathode ray tube (CRT) technology. LCDs consume much less power than LED and gas-display displays because they work on the principle of blocking light rather than emitting it. Where an LED emits light, the liquid crystals in an LCD produces an image using a backlight.

As LCDs have replaced older display technologies, LCDs have begun being replaced by new display technologies such as OLEDs.

2.7.1. TYPES OF LCDS

Types of LCDs include:

- Twisted Nematic (TN)- which are inexpensive while having high response times. However, TN displays have low contrast ratios, viewing angles and color contrasts.
- In Panel Switching displays (IPS Panels)- which boast much better contrast ratios, viewing angles and color contrast when compared to TN LCDs.
- Vertical Alignment Panels (VA Panels)- which are seen as a medium quality between TN and IPS displays.
- Advanced Fringe Field Switching (AFFS)- which is a top performer compared IPS displays in color reproduction range.

2.7.2. WORKING PRINCIPLE

A display is made up of millions of pixels. The quality of a display commonly refers to the number of pixels; for example, a 4K display is made up of 3840 x2160 or 4096x2160 pixels. A pixel is made up of three subpixels; a red, blue and green—commonly called RGB. When the subpixels in a pixel change color combinations, a different color can be produced. With all the pixels on a display working together, the display can make millions of different colors. When the pixels are rapidly switched on and off, a picture is created.

The way a pixel is controlled is different in each type of display; CRT, LED, LCD and newer types of displays all control pixels differently. In short, LCDs are lit by a backlight, and pixels are switched on and off electronically while using liquid crystals to rotate polarized light. A polarizing glass filter is placed in front and behind all the pixels, the front filter is placed at 90 degrees. In between both filters are the liquid crystals, which can be electronically switched on and off.

LCDs are made with either a passive matrix or an active matrix display grid. The active matrix LCD is also known as a thin film transistor (TFT) display. The passive matrix LCD has a grid of conductors with pixels located at each intersection in the grid. A current is sent across two conductors on the grid to control the light for any pixel. An active matrix has a

transistor located at each pixel intersection, requiring less current to control the luminance of a pixel. For this reason, the current in an active matrix display can be switched on and off more frequently, improving the screen refresh time.

Some passive matrix LCD's have dual scanning, meaning that they scan the grid twice with current in the same time that it took for one scan in the original technology. However, active matrix is still a superior technology out of the two.

2.7.3. LCD VS OLED VS QLED

LCDs are now being outpaced by other display technologies, but are not completely left in the past. Steadily, LCDs have been being replaced by OLEDs, or organic light-emitting diodes.

OLEDs use a single glass or plastic panels, compared to LCDs which use two. Because an OLED does not need a backlight like an LCD, OLED devices such as televisions are typically much thinner, and have much deeper blacks, as each pixel in an OLED display is individually lit. If the display is mostly black in an LCD screen, but only a small portion needs to be lit, the whole back panel is still lit, leading to light leakage on the front of the display. An OLED screen avoids this, along with having better contrast and viewing angles and less power consumption. With a plastic panel, an OLED display can be bent and folded over itself and still operate. This can be seen in smartphones, such as the controversial Galaxy Fold; or in the iPhone X, which will bend the bottom of the display over itself so the display's ribbon cable can reach in towards the phone, eliminating the need for a bottom bezel.

However, OLED displays tend to be more expensive and can suffer from burn-in, as plasma-based displays do.

QLED stands for quantum light-emitting diode and quantum dot LED. QLED displays were developed by Samsung and can be found in newer televisions. QLEDs work most similarly to LCDs, and can still be considered as a type of LCD. QLEDs add a layer of quantum dot film to an LCD, which increases the color and brightness dramatically compared to other LCDs. The quantum dot film is made up of small crystal semi-conductor particles. The crystal semi-conductor particles can be controlled for their color output.

When deciding between a QLED and an OLED display, QLEDs have much more brightness

and aren't affected by burn-in. However, OLED displays still have a better contrast ratio and deeper blacks than QLEDs.

2.8 WIFI MODULE

The ESP 01 ESP8266 Serial WIFI Wireless Transceiver Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any micro-controller access to your WiFi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor.

Each ESP8266 module comes pre-programmed with an AT command set firmware, meaning, you can simply hook this up to your Arduino device and get about as much WiFi-ability as a WiFi Shield offers (and that's just out of the box)! The ESP8266 module is an extremely cost-effective board with a huge, and ever-growing, community.

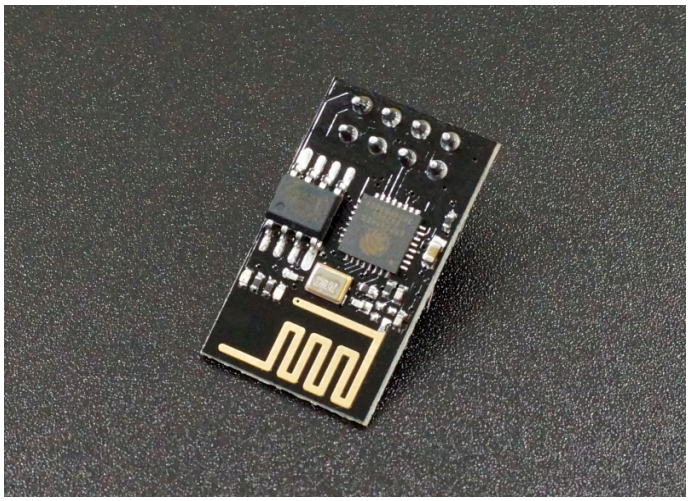

Figure 2.11:: WiFi Module (ESP8266)

Its high degree of on-chip integration allows for minimal external circuitry, including the front-end module, which is designed to occupy minimal PCB area. The ESP8266 supports APSD for VoIP applications and Bluetooth co-existence interfaces, it contains a self-calibrated RF allowing it to work under all operating conditions and requires no external RF parts.

2.8.1. AT COMMANDS

Command	Feature
AT	Test AT startup.
AT+RST	Restart a module.
AT+GMR	Check version information
AT+CMD	List all AT commands and types supported in current firmware.
AT+GSLP	Enter Deep-sleep mode.
ATE	Configure AT commands echoing.
AT+RESTORE	Restore factory default settings of the module.
AT+UART_CUR	Current UART configuration, not saved in flash.
AT+UART_DEF	Default UART configuration, saved in flash.
AT+FS	Filesystem Operations.

Table 2.3:: Commands used to interact with ESP8266

2.8.2. TICKER

Library for calling functions repeatedly with a certain period.

It is currently not recommended to do blocking IO operations (network, serial, file) from Ticker callback functions. Instead, set a flag inside the ticker callback and check for that flag inside the loop function.

Here is library to simplificate Ticker usage and avoid WDT reset: **TickerScheduler**

2.8.3. EEPROM

EEPROM.write does not write to flash immediately, instead you must call **EEPROM.commit()** whenever you wish to save changes to flash. **EEPROM.end()** will also commit, and will release the RAM copy of EEPROM contents.

EEPROM library uses one sector of flash located just after the embedded filesystem.

2.8.4. DNS SERVER

Implements a simple DNS server that can be used in both STA and AP modes. The DNS server currently supports only one domain (for all other domains it will reply with NXDOMAIN or custom status code). With it, clients can open a web server running on ESP8266 using a domain name, not an IP address.

CHAPTER-3

IMPLEMENTATION

2.1 POWER SUPPLY

All digital circuits require regulated power supply. In this article we are going to learn how to get a regulated positive supply from the mains supply.

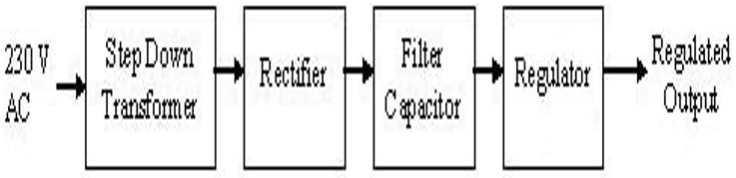

Figure 3.1:: Fixed Regulated Power supply.

The abover shows the basic block diagram of a fixed regulated power supply. Let us go through each block.

2.1.1. TRANSFORMER

A transformer consists of two coils also called as “WINDINGS” namely PRIMARY SECONDARY. They are linked together through inductively coupled electrical conductors also called as CORE. A changing current in the primary causes a change in the Magnetic Field in the core & this in turn induces an alternating voltage in the secondary coil. If load is applied to the secondary then an alternating current will flow through the load. If we consider an ideal condition then all the energy from the primary circuit will be transferred to the secondary circuit through the magnetic field.

$$\begin{aligned} P_{primary} &= P_{secondary} \\ I_p\, V_p &= I_s\, V_s \end{aligned} \tag{1}$$

The secondary voltage of the transformer depends on the number of turns in the Primary as well as in the secondary.

$$\frac{V_s}{V_p} = \frac{N_s}{N_p}$$

2.1.2. RECTIFIER

A rectifier is a device that converts an AC signal into DC signal. For rectification purpose we use a diode, a diode is a device that allows current to pass only in one direction i.e. when the anode of the diode is positive with respect to the cathode also called as forward biased condition blocks current in the reversed biased condition.

Rectifier can be classified as follows:

1. Full wave rectifier
2. Half wave rectifier
3. Bridge rectifier

2.1.3. VOLTAGE REGULATOR

A Voltage regulator is a device which converts varying input voltage into a constant regulated output voltage. Voltage regulator can be of two types.

1. **Linear Voltage Regulator**
   Also called as Resistive Voltage regulator because they dissipate the excessive voltage resistively as heat.
2. **Switching Regulators**
   They regulate the output voltage by switching the Current ON/OFF very rapidly. Since their output is either ON or OFF it dissipates very low power thus achieving higher efficiency as compared to linear voltage regulators. But they are more complex generate high noise due to their switching action. For low level of output power switching regulators tend to be costly but for higher output wattage they are much cheaper than linear regulators.

The most commonly available Linear Positive Voltage Regulators are the 78XX series where the XX indicates the output voltage. And 79XX series is for Negative Voltage Regulators.

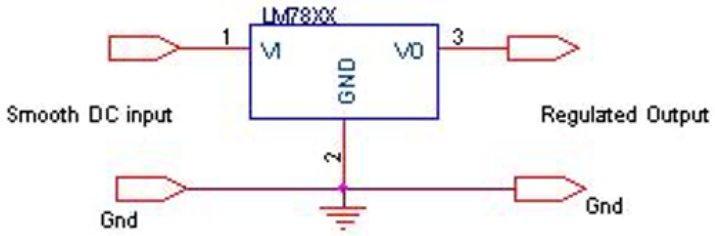

Figure 3.2:: Power Regulating Filter.

After filtering the rectifier output the signal is given to a voltage regulator. The maximum input voltage that can be applied at the input is 35V.Normally there is a 2-3 Volts drop across the regulator so the input voltage should be at least 2-3 Volts higher than the output voltage. If the input voltage gets below the Vmin of the regulator due to the ripple voltage or due to any other reason the voltage regulator will not be able to produce the correct regulated voltage.

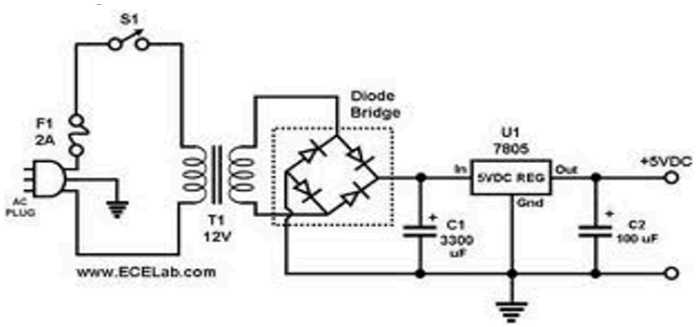

Figure 3.3:: Circuit diagram of power supply.

2.2 INTERCONNECTING SENSORS

2.2.1. WIFI CONNECTIVITY

ESP8266WIFI LIBRARY

ESP8266 is all about Wi-Fi. If you are eager to connect your new ESP8266 module to a Wi-Fi network to start sending and receiving data, this is a good place to start. If you are looking for more in depth details of how to program specific Wi-Fi networking functionality, you are also in the right place.

The Wi-Fi library for ESP8266 has been developed based on ESP8266 SDK, using the nam-

ing conventions and overall functionality philosophy of the Arduino WiFi library. Over time, the wealth of Wi-Fi features ported from ESP8266 SDK to esp8266 / Arduino outgrew Arduino WiFi library and it became apparent that we would need to provide separate documentation on what is new and extra.

To hook up the ESP module to Wi-Fi (like hooking up a mobile phone to a hot spot), you need only a couple of lines of code;

```
#include <ESP8266WiFi.h>
void setup()
{
  Serial.begin(115200);
  Serial.println();

  WiFi.begin("network-name", "pass-to-network");

  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println();

  Serial.print("Connected, IP address: ");
  Serial.println(WiFi.localIP());
}
void loop() {}
```

In the line `WiFi.begin("network-name", "pass-to-network")` replace network-name and pass-to-network with the name and password of the Wi-Fi network you would like to connect to.

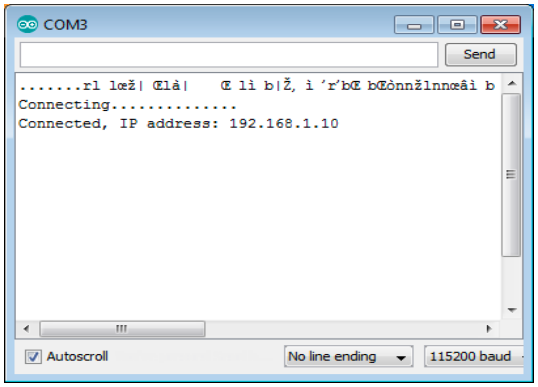

Figure 3.4:: Wifi status on Serial Monitor

2.2.2. RF ANTENNA MODULE

The project can be implemented with or without the help of a special library called "VirtualWire.h". The project implemented here uses the library. If we want to implement the project without the library, then we need to change the receiver part of the circuit.

VirtualWire.h is a special library for Arduino created by Mike McCauley. It is a communication library that allows two Arduino's to communicate with each other using RF Module i.e. transmitter – receiver pair. This library consists of several functions that are used for configuring the modules, transmission of data by the transmitter module and data reception by the receiver module.

In this project, the transmitter simply sends two characters i.e. it sends the character "1" and with a delay of few seconds, it sends the character "0". Whenever the "1" is sent, the LED on the transmitting side of the project will be turned ON. As this "1" is transmitted via RF communication, the receiver will receive the data "1".

When the receiver receives "1", the Arduino on the receiver side of the project will turn ON the LED on its side.

Similarly, when the data "0" is transmitted by the RF transmitter, the LED on the transmitter side is turned OFF. As a result, the receiver now receives "0" and the LED on the receiver side is also turned OFF.

Hence, the receiver is imitating the actions of the transmitter.

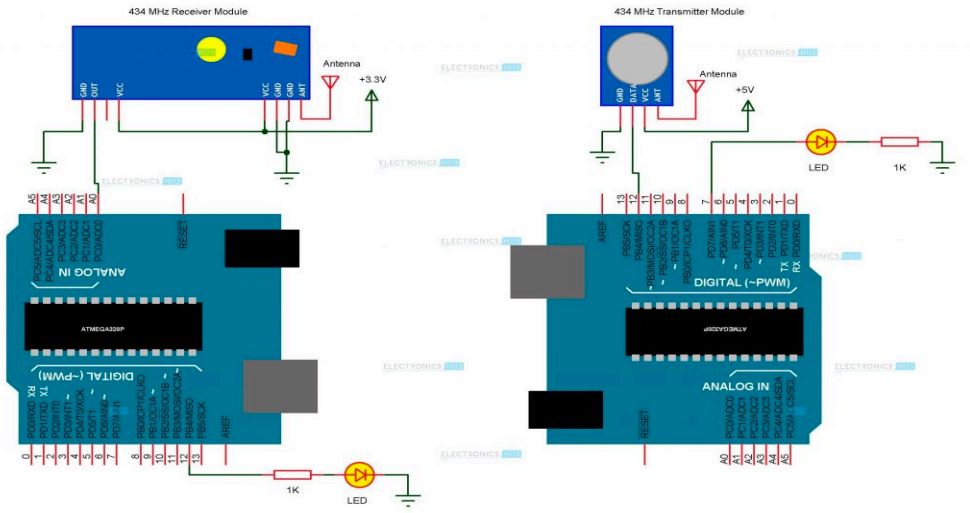

Figure 3.5:: Connecting RF module to Arduino UNO

2.2.3. SERVO MOTOR

Servo motors are great devices that can turn to a specified position.

Usually, they have a servo arm that can turn 180 degrees. Using the Arduino, we can tell a servo to go to a specified position and it will go there. As simple as that!

Servo motors were first used in the Remote Control (RC) world, usually to control the steering of RC cars or the flaps on a RC plane. With time, they found their uses in robotics, automation, and of course, the Arduino world.

Here we will see how to connect a servo motor and then how to turn it to different positions.

The first motor I ever connected to an Arduino, seven years ago, was a Servo motor. Nostalgic moment over, back to work!

We will need the following things:

1. An Arduino board connected to a computer via USB
2. A servo motor.
3. Jumper wires

There are few big names in the servo motor world. Hitec and Futaba are the leading RC servo

manufacturers. Good places to buy them are Servocity, Sparkfun, and Hobbyking.

The following code will turn a servo motor to 0 degrees, wait 1 second, then turn it to 90, wait one more second, turn it to 180, and then go back.

```
// Include the Servo library
#include <Servo.h>
// Declare the Servo pin
int servoPin = 3;
// Create a servo object
Servo Servo1;
void setup() {
   // We need to attach the servo to the used pin number
   Servo1.attach(servoPin);
}
void loop(){
   // Make servo go to 0 degrees
   Servo1.write(0);
   delay(1000);
   // Make servo go to 90 degrees
   Servo1.write(90);
   delay(1000);
   // Make servo go to 180 degrees
   Servo1.write(180);
   delay(1000);
}
```

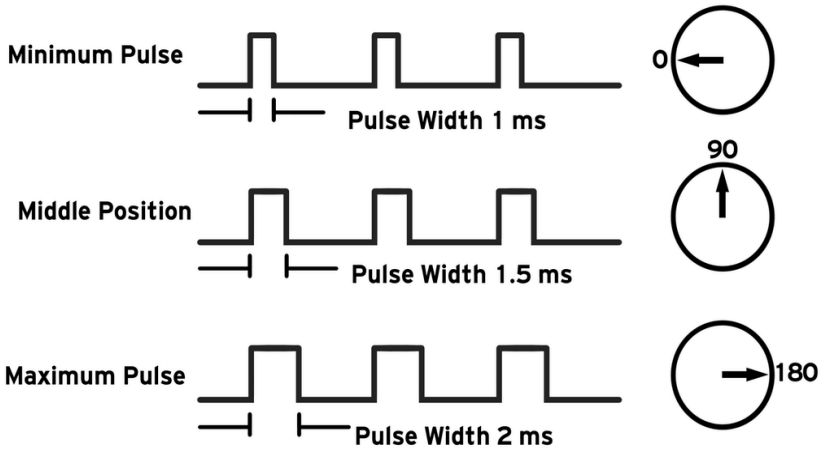

Figure 3.6:: Generated pulse of 180 degree servo motor.

Servos are clever devices. Using just one input pin, they receive the position from the Arduino and they go there. Internally, they have a motor driver and a feedback circuit that makes sure that the servo arm reaches the desired position. But what kind of signal do they receive on the input pin?

It is a square wave similar to PWM. Each cycle in the signal lasts for 20 milliseconds and for most of the time, the value is LOW. At the beginning of each cycle, the signal is HIGH for a time between 1 and 2 milliseconds. At 1 millisecond it represents 0 degrees and at 2 milliseconds it represents 180 degrees. In between, it represents the value from 0–180. This is a very good and reliable method. The graphic makes it a little easier to understand.

Remember that using the Servo library automatically disables PWM functionality on PWM pins 9 and 10 on the Arduino UNO and similar boards.

2.2.4. LCD

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.

- A Read/Write (R/W) pin that selects reading mode or writing mode.
- An Enable pin that enables writing to the registers.
- 8 data pins (D0 -D7). The states of these pins (high or low) are the bits that you're writing to a register when you write, or the values you're reading when you read.

There's also a display contrast pin (Vo), power supply pins (+5V and GND) and LED Backlight (Bklt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on and off the LED backlight, respectively.

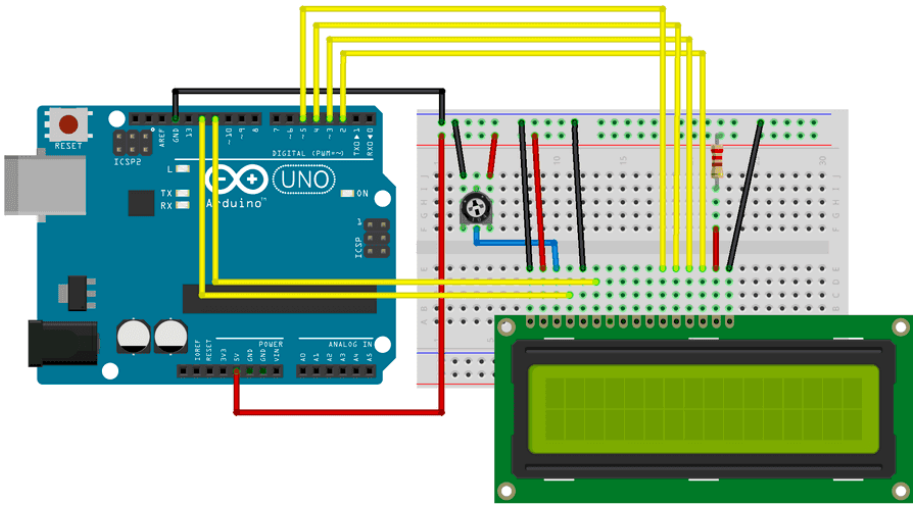

Figure 3.7:: Connecting LCD to Arduino UNO.

TESTING THE INTERFACE OF LCD

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library by associating any
// needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
```

```
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row,
  // since counting begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

2.3 FLASK IMPLEMENTATION

2.3.1. INITIALIZING THE APPLICATION

1. First we imported the Flask class. An instance of this class will be our WSGI application.
2. Next we create an instance of this class. The first argument is the name of the application's module or package. _ _name_ _ is a convenient shortcut for this that is appropriate for most cases. This is needed so that Flask knows where to look for resources such as templates and static files.
3. We then use the route() decorator to tell Flask what URL should trigger our function.
4. The function returns the message we want to display in the user's browser. The default content type is HTML, so HTML in the string will be rendered by the browser.

Save it as hello.py or something similar. Make sure to not call your application flask.py because this would conflict with Flask itself.

```
from flask import Flask
```

```
app = Flask(__name__)
@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

To run the application, use the flask command or python -m flask. Before you can do that you need to tell your terminal the application to work with by exporting the FLASK_APP environment variable:

```
    #Execute command (Bash profile)
$ export FLASK_APP=hello
$ flask run
 * Running on http://127.0.0.1:5000/
```

2.3.2. HTML ESCAPING

When returning HTML (the default response type in Flask), any user-provided values rendered in the output must be escaped to protect from injection attacks. HTML templates rendered with Jinja, introduced later, will do this automatically.

escape(), shown here, can be used manually. It is omitted in most examples for brevity, but you should always be aware of how you're using untrusted data.

```
from markupsafe import escape


@app.route("/<name>")
def hello(name):
    return f"Hello, {escape(name)}!"
```

2.3.3. ROUTING

Modern web applications use meaningful URLs to help users. Users are more likely to like a page and come back if the page uses a meaningful URL they can remember and use to directly visit a page.

Use the **route()** decorator to bind a function to a URL.

```
@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'Hello, World'
```

2.3.4. URL BUILDING

To build a URL to a specific function, use the **url_for()** function. It accepts the name of the function as its first argument and any number of keyword arguments, each corresponding to a variable part of the URL rule. Unknown variable parts are appended to the URL as query parameters.

Use of url_for():

1. Reversing is often more descriptive than hard-coding the URLs.
2. You can change your URLs in one go instead of needing to remember to manually change hard-coded URLs.
3. URL building handles escaping of special characters transparently.
4. The generated paths are always absolute, avoiding unexpected behavior of relative paths in browsers.
5. If your application is placed outside the URL root, for example, in /myapplication instead of /, url_for() properly handles that for you.

For example;

```
from flask import url_for

@app.route('/')
def index():
    return 'index'
```

```
@app.route('/login')
def login():
    return 'login'

@app.route('/user/<username>')
def profile(username):
    return f'{username}\'s profile'

with app.test_request_context():
    print(url_for('index'))
    print(url_for('login'))
    print(url_for('login', next='/'))
    print(url_for('profile', username='John Doe')
```

2.4 HTTP METHODS

Web applications use different HTTP methods when accessing URLs. You should familiarize yourself with the HTTP methods as you work with Flask. By default, a route only answers to GET requests. You can use the methods argument of the **route()** decorator to handle different HTTP methods.

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

If GET is present, Flask automatically adds support for the HEAD method and handles HEAD requests according to the HTTP RFC. Likewise, OPTIONS is automatically implemented for

you.

2.5 TERMINAL NETWORK (TELNET)

TELNET stands for TErminaL NETwork. It is a type of protocol that enables one computer to connect to local computer. It is a used as a standard TCP/IP protocol for virtual terminal service which is given by ISO. Computer which starts connection known as the local computer. Computer which is being connected to i.e. which accepts the connection known as remote computer. When the connection is established between local and remote computer. During telnet operation whatever that is being performed on the remote computer will be displayed by local computer. Telnet operates on client/server principle. Local computer uses telnet client program and the remote computers uses telnet server program.

2.5.1. TELNET COMMANDS

Commands of the telnet are identified by a prefix character, Interpret As Command (IAC) which is having code 255. IAC is followed by command and option codes. Basic format of the command is as shown in the following figure;

Figure 3.8:: Instruction set of TELNET

Following are some of the important TELNET commands.

Character	Decimal	Binary	Meaning
WILL	251	11111011	1. Offering to enable. 2. Accepting a request to enable.
WON'T	252	11111100	1. Rejecting a request to enable. 2. Offering to disable. 3. Accepting a request to disable.
DO	253	11111101‘	1. Approving a request to enable. 2. Requesting to enable.
DON'T	254	11111110	1. Disapproving a request to enable. 2. Approving an offer to disable. 3. Requesting to disable.

Table 3.4:: TELNET Characters

2.6 ACCESSING REQUEST DATA

For web applications it's crucial to react to the data a client sends to the server. In Flask this information is provided by the global **request** object. If you have some experience with Python you might be wondering how that object can be global and how Flask manages to still be threadsafe. The answer is context locals:

Certain objects in Flask are global objects, but not of the usual kind. These objects are actually proxies to objects that are local to a specific context. What a mouthful. But that is actually quite easy to understand.

Imagine the context being the handling thread. A request comes in and the web server decides to spawn a new thread (or something else, the underlying object is capable of dealing with concurrency systems other than threads). When Flask starts its internal request handling it figures out that the current thread is the active context and binds the current application and the WSGI environments to that context (thread). It does that in an intelligent way so that one application can invoke another application without breaking.

So what does this mean to you? Basically you can completely ignore that this is the case

unless you are doing something like unit testing. You will notice that code which depends on a request object will suddenly break because there is no request object. The solution is creating a request object yourself and binding it to the context. The easiest solution for unit testing is to use the **test_request_context()** context manager. In combination with the with statement it will bind a test request so that you can interact with it. Here is an example:

```
from flask import request


with app.test_request_context('/hello', method='POST'):
    # now you can do something with the request until the
    # end of the with block, such as basic assertions:
    assert request.path == '/hello'
    assert request.method == 'POST'
```

The other possibility is passing a whole WSGI environment to the **request_context()** method:

```
    with app.request_context(environ):
    assert request.method == 'POST'
```

2.7 MQTT PROTOCOL

MQTT stands for Message Queuing Telemetry Transport. MQTT is a machine to machine internet of things connectivity protocol. It is an extremely lightweight and publish-subscribe messaging transport protocol. This protocol is useful for the connection with the remote location where the bandwidth is a premium. These characteristics make it useful in various situations, including constant environment such as for communication machine to machine and internet of things contexts. It is a publish and subscribe system where we can publish and receive the messages as a client. It makes it easy for communication between multiple devices. It is a simple messaging protocol designed for the constrained devices and with low bandwidth, so it's a perfect solution for the internet of things applications.

2.7.1. ARCHITECTURE OF MQTT

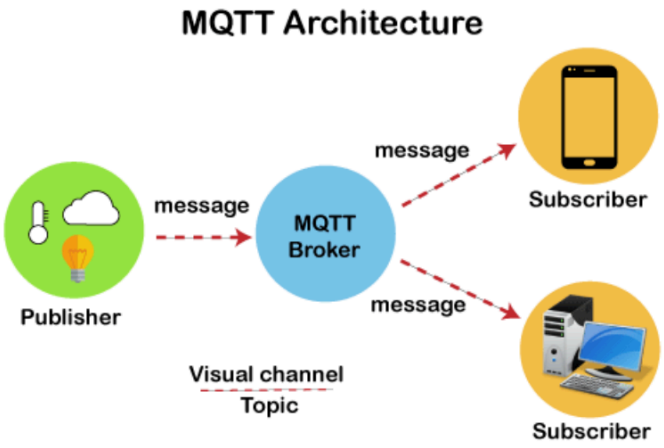

Figure 3.9:: MQTT Architecture

To understand it more clearly, we will look at the example. Suppose a device has a temperature sensor and wants to send the rating to the server or the broker. If the phone or desktop application wishes to receive this temperature value on the other side, then there will be two things that happened. The publisher first defines the topic; for example, the temperature then publishes the message, i.e., the temperature's value. After publishing the message, the phone or the desktop application on the other side will subscribe to the topic, i.e., temperature and then receive the published message, i.e., the value of the temperature. The server or the broker's role is to deliver the published message to the phone or the desktop application.

CHAPTER-4

SOURCE CODE & DATA VALIDATION

4.1 CONNECTING TO SERVER & METHOD CALLS

To implement the server side application, we are using a NodeMCU. The main prospect of this project is to turn the position of a receiving antenna and to do so we are using a servo motor to have an angular propulsion. The receiving antenna is placed on top of a servo motor.

To rotate the servo motor, we have implemented two kinds of mechanisms

1. Send the commands from a TELNET application. This is used to interface an Android mobile device to the NodeMCU. The servo motor is rotated as in turn of the commands we passed.
2. The other way is by implementing a webpage. The webapge is designed with necessary buttons and a position of a receiving antenna. We are designed a basic Application Interface to have a method calls in between the client and a server.

The below code ensures the proper configuration of a servo motor. Upload the code in a NodeMCU and connect the servo sensor to GPIO-2.

```
1   /*************************************************
2  * Written by: Bhavesh Asanabada
3  * ************************************************/
4
5  #include <ESP8266WiFi.h>
6  #include <Servo.h>
7  Servo servo;
8
9  //const char* ssid = "Enter your WiFi Name";
10 //const char* password = "Enter WiFi password";
11
```

```
12 WiFiServer server(80);
13
14 void setup() {
15   Serial.begin(115200);
16   delay(10);
17   servo.attach(2); //Gpio-2 of nodemcu with pwm pin of servo motor
18   // Connect to WiFi network
19   Serial.println();
20   Serial.println();
21   Serial.print("Connecting to ");
22   Serial.println(ssid);
23
24   WiFi.begin(ssid, password);
25
26   while (WiFi.status() != WL_CONNECTED) {
27     delay(500);
28     Serial.print(".");
29   }
30   Serial.println("");
31   Serial.println("WiFi connected");
32
33   // Start the server
34   server.begin();
35   Serial.println("Server started");
36
37   // Print the IP address on serial monitor
38   Serial.print("Use this URL to connect: ");
39    //URL IP to be typed in mobile/desktop browser
40   Serial.print("http://");
41   Serial.print(WiFi.localIP());
42   Serial.println("/");
```

```
43
44  }
45
46  void loop() {
47    // Check if a client has connected
48    WiFiClient client = server.available();
49    if (!client) {
50      return;
51    }
52
53    // Wait until the client sends some data
54    Serial.println("new client");
55    while(!client.available()){
56      delay(1);
57    }
58
59    // Read the first line of the request
60    String request = client.readStringUntil('\r');
61    Serial.println(request);
62    client.flush();
63
64   int value = 0;
65   // Match the request
66
67  if (request.indexOf("/Req=0") != -1)  {
68    servo.write(0); //Moving servo to 0 degree
69    value=0;
70  }
71  if (request.indexOf("/Req=20") != -1)  {
72    servo.write(20); //Moving servo to 20 degree
73    value=20;
```

```
74  }
75  if (request.indexOf("/Req=40") != -1)  {
76    servo.write(40); //Moving servo to 40 degree
77    value=40;
78  }
79  if (request.indexOf("/Req=60") != -1)  {
80    servo.write(60); //Moving servo to 60 degree
81    value=60;
82  }
83  if (request.indexOf("/Req=80") != -1)  {
84    servo.write(80); //Moving servo to 80 degree
85    value=80;
86  }
87  if (request.indexOf("/Req=90") != -1)  {
88    servo.write(90); //Moving servo to 90 degree
89    value=90;
90  }
91  if (request.indexOf("/Req=120") != -1)  {
92    servo.write(120); //Moving servo to 1200 degree
93    value=120;
94  }
95  if (request.indexOf("/Req=140") != -1)  {
96    servo.write(140); //Moving servo to 140 degree
97    value=140;
98  }
99  if (request.indexOf("/Req=160") != -1)  {
100   servo.write(160); //Moving servo to 160 degree
101   value=160;
102 }
103 if (request.indexOf("/Req=180") != -1)  {
104   servo.write(180); //Moving servo to 180 degree
```

```
105     value=180;
106   }
107
108     // Return the response
109     client.println("HTTP/1.1 200 OK");
110     client.println("Content-Type: text/html");
111     client.println(""); //  do not forget this one
112     client.println("<!DOCTYPE HTML>");
113     client.println("<html>");
114     client.println("<h1 align=center>
115                 Positioning the receiver module over WiFi
116                 </h1><br><br>");
117     client.print("<align=center>Current position =");
118     client.print(value);
119     client.println("<br><br>");
120     client.println("<a href=\"/Req=0\"\"><button>
121                 Move to = 0 degree</button></a>");
122     client.println("<a href=\"/Req=20\"\"><button>
123                 Move to = 20 degree</button></a>");
124     client.println("<a href=\"/Req=40\"\"><button>
125                 Move to = 40 degree</button></a>");
126     client.println("<a href=\"/Req=60\"\"><button>
127                 Move to = 60 degree</button></a>");
128     client.println("<a href=\"/Req=80\"\"><button>
129                 Move to = 80 degree</button></a>");
130     client.println("<a href=\"/Req=120\"\"><button>
131                 Move to = 120 degree</button></a>");
132     client.println("<a href=\"/Req=140\"\"><button>
133                 Move to = 140 degree</button></a>");
134     client.println("<a href=\"/Req=160\"\"><button>
135                 Move to = 160 degree</button></a>");
```

```
136     client.println("<a href=\"/Req=180\"\"><button>
137                 Move to = 180 degree</button></a><br/>");
138
139     client.println("</html>");
140     delay(1);
141     Serial.println("Client disonnected");
142     Serial.println("");
143
144 }
```

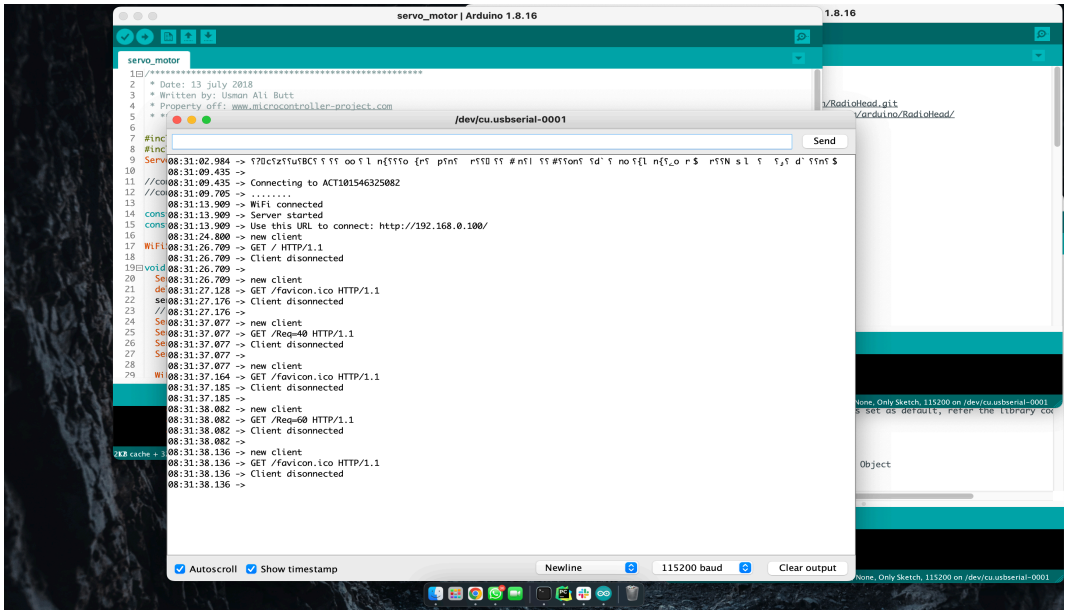

Figure 4.1:: Server datalog

Figure 4.2:: A desktop webpage (client side)

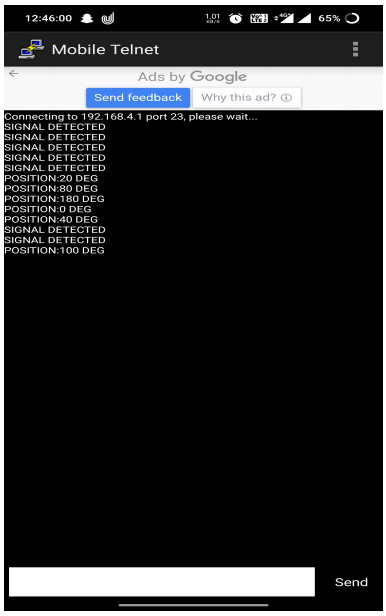

Figure 4.3:: TELNET Interface

4.2 INTERFACING RF TRANSMITTING MODULE

```
1 // Radio Frequency Transmitter Module Code
2 // Micro-controller: Arduino UNO
3 // Author: Bhavesh Asanabada
```

```
4  // To download libraries visit:
5  // https://github.com/PaulStoffregen/RadioHead.git
6  // For more info about RadioHead visit:
7  // https://www.airspayce.com/mikem/arduino/RadioHead/
8
9  // <-------- Initialize required libraries (Begin) -------->
10 // Include RadioHead Amplitude Shift Keying Library
11 #include <RH_ASK.h>
12
13 // Include dependant SPI Library
14 #include <SPI.h>
15 // <-------- Initialize required libraries (End) -------->
16
17 // The data pin of transmitter is set as default,
18 // refer the library code.
19
20 // Set LED pin
21 #define ledPin 7
22
23 // Create Amplitude Shift Keying Object
24 RH_ASK rf_driver;
25
26 void setup()
27   {
28     // Initialize ASK Object
29     rf_driver.init();
30
31     // Begin the serial monitor at 9600 Baud rate.
32     Serial.begin(9600);
33
34     pinMode(ledPin, OUTPUT);
```

```
35    }
36
37  void loop()
38    {
39      // Set a message to transmit
40      const char *msg = "Hello World";
41
42      if(rf_driver.send((uint8_t *)msg, strlen(msg)))
43        {
44          Serial.println("Message Transmitted");
45
46          // To indicate the message is transmitted
47          digitalWrite(ledPin, HIGH);
48        }
49
50      rf_driver.waitPacketSent();
51      delay(2000);
52
53      digitalWrite(ledPin, LOW);
54      delay(2000);
55
56    }
```

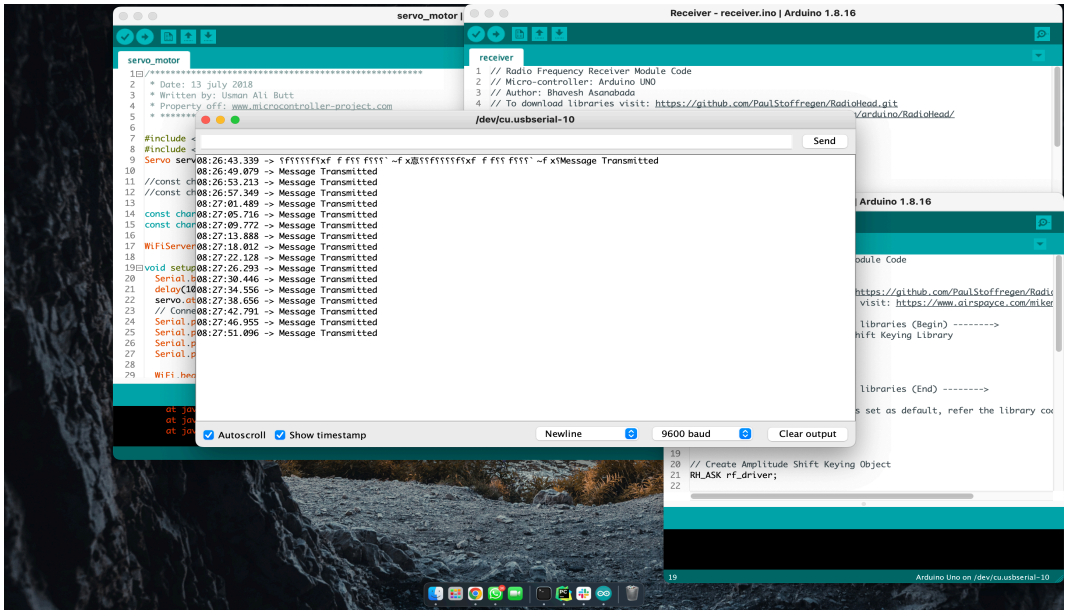

Figure 4.4:: Transmitter Datalog

4.3 INTERFACING RF RECEIVING MODULE

The receiving module is connected to an Arduino UNO. Upload the following code check for the feedback from a serial monitor.

```
// Radio Frequency Receiver Module Code
// Micro-controller: Arduino UNO
// Author: Bhavesh Asanabada
// To download libraries visit:
// https://github.com/PaulStoffregen/RadioHead.git
// For more info about RadioHead visit:
// https://www.airspayce.com/mikem/arduino/RadioHead/


// <-------- Initialize required libraries (Begin) -------->
#include <LiquidCrystal.h>
#include <RHGenericDriver.h>
// Include RadioHead Amplitude Shift Keying Library
#include <RH_ASK.h>
```

```
// Include dependant SPI Library
#include <SPI.h>
// <-------- Initialize required libraries (End) -------->

// Re-configure the LCD pins associated as below
const int rs = 13, en = 12, d4 = 11, d5 = 10, d6 = 9, d7 = 8;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Receiver datapin config
const int datain = 12;

#define ledPin 7

// Create Amplitude Shift Keying Object
RH_ASK rf_driver;

void setup()
{
    // Initialize ASK Object
    rf_driver.init();

    // Setup Serial Monitor
    Serial.begin(9600);

    pinMode(ledPin, OUTPUT);
}

void loop()
{
    // Set buffer to size of expected message
    uint8_t buf[11];
```

```
46      uint8_t buflen = sizeof(buf);
47
48      // Check if received packet is correct size
49      if (rf_driver.recv(buf, &buflen)){
50          // Message received with valid checksum
51          Serial.print("Message Received: ");
52          digitalWrite(ledPin, HIGH);
53
54          // set up the LCD's number of columns and rows:
55          lcd.begin(16, 2);
56          // Print a message to the LCD.
57          lcd.print("Receiving");
58          Serial.println((char*)buf);
59
60          uint8_t lastRssi = rf_driver.lastRssi();
61          // Serial.println(rf_driver.lastRssi(), DEC);
62          int16_t iRssi = (rf_driver.lastRssi(), DEC);
63          // String sRssi = String((rf_driver.lastRssi(), DEC));
64
65          Serial.println(iRssi);    // 10
66          // Serial.println(sRssi);
67          delay(1000);
68        }
69
70      else{
71        Serial.println("Message not received");
72        Serial.println(rf_driver.lastRssi(), DEC);
73        lcd.begin(0, 1);
74        lcd.print("Transmission Error");
75      }
76
```

```
77      digitalWrite(ledPin, LOW);
78      delay(1000);
79  }
```

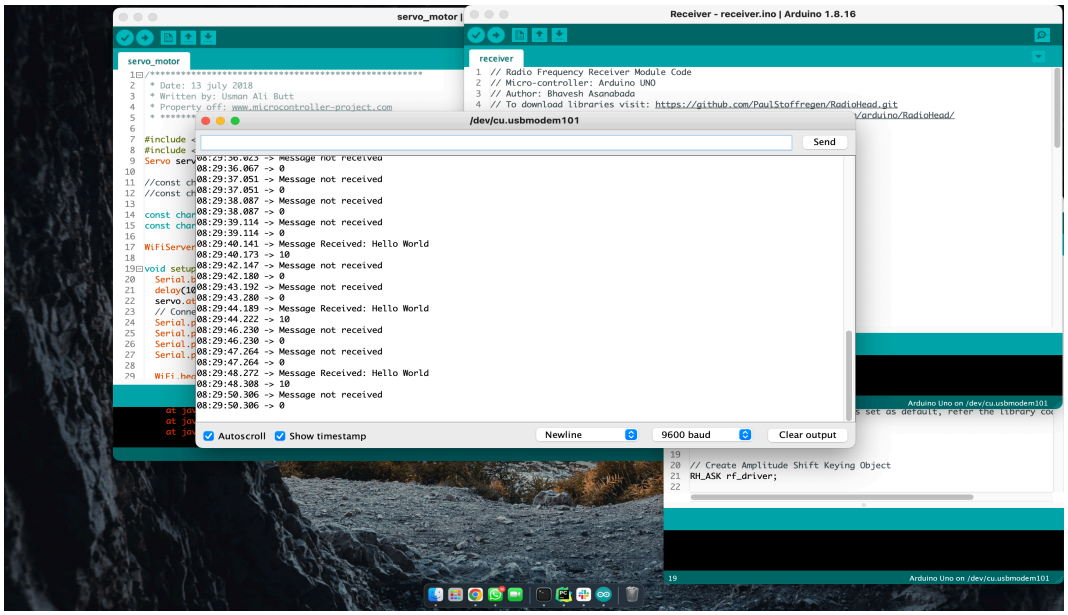

Figure 4.5:: Receiver Datalog

CHAPTER-5

CONCLUSION

The Antenna Positioning System primarily functions to spot the presence of any signal. The thought is to develop a system which can manage the movement of the saucer altogether direction. The most application of employing a dish is to receive signal from satellite and alternative broad casting sources. So as to position the dish to the precise angle to receive the utmost signal of a specific frequency it has to be adjusted manually. So as to beat the problem of adjusting manually, this planned system helps in adjusting the position of the antenna.

A hyperbolic positioning method with antenna arrays consisting of approximately-located antennas and a multi-channel pseudo-lite is proposed in order to overcome the problems of indoor positioning with conventional pseudolites (ground-based GPS transmitters. The experimental result shows that the positioning accuracy varies centimeter to meter level according to the geometric relation between the pseudolite antennas and the receiver. It also shows that the bias error of the carrier-phase difference observable is more serious than their random error. Based on the size of the bias error of carrier-phase difference that is inverse-calculated from the experimental result, three-dimensional positioning performance is evaluated by computer simulation. The simulated values and evaluation methods introduced in this work can be applied to various antenna setups; therefore, by using them, positioning performance can be predicted in advance of installing an actual system.

REFERENCES

1. “Microcontroller Based Wireless 3D Position Control ” By Amritha Mary A. S.1, Divyasree M V2, Jesna Prem2, Kavyasree S M2, Keerthana Vasu

2. “Microcontroller Based Wireless Automatic Antenna Positioning System” By Surya Deo Choudhary, Pankajrai, Aravind Kumar, Irshad Alam, International Journal Of Electrical And Computational Systems, IJEECS ISSN 2348-117X Vol 3, Issue 6.

3. “Remote Alignment of Dish Positioning By Android Application” by Prajwal Basnet, Pranjal Grover Preeti Pannu, International Journal of Engineering Research Management Technology, ISSN: 2348-4039, March- 2015 Volume 2, Issue-2

4. “Automated Antenna Positioning for Wireless Networks” by Amit Dvir, Yehuda Ben-Shimol, Yoav Ben-Yehezkel, Michael Segal, Department of Communication Systems Engineering, Ben Gurion University, Israel, Boaz Ben-Moshe, School of Computing Science, Simon Fraser University, Canada.

5. Dish Positioning By Using IR Remote by Prof. S. A Maske, Mr. Shelake Aniket V, Mr.Shinde Anup S, Mr. Mugade Nitin K., Sanjeevan Engineering Institute Technology, Panhala,Department of ETC, International Research Journal of Engineering and Technology (IRJET), ISSN: 2395 -0056, Volume: 04 Issue: 03 | Mar - 2017.

6. “Background Positioning for Mobile Devices” by Denis Huber School of Electrical and Computer Engineering Technical University Berlin.

7. “Satellite Communication” By Anirban Sengupta, Electronics Communication Engineering Department, Asansol Engineering College Kanyapur, Senraleigh Road, Asansol 713304, Burdwan, India.

APPENDIX

1.1 RADIO HEAD LIBRARY

In this project we are using a NodeMCU and an Arduino UNO microcontrollers to connect and communicate among the devices. The Node micro-controller unit has an inbuilt WiFi support and a RAM, through which we can implement the method calls to return the data from client to the server.

We are using RadioHead Packet Radio library to transmit the messages in embedded microprocessors. It provides a complete object-oriented library for sending and receiving packetized messages via a variety of common data radios and other transports on a range of embedded microprocessors.

RadioHead consists of 2 main sets of classes:

- **Drivers**
  Drivers provide low level access to a range of different packet radios and other packetized message transports.
- **Managers**
  Managers provide high level message sending and receiving facilities for a range of different requirements.

Every RadioHead program will have an instance of a Driver to provide access to the data radio or transport, and usually a Manager that uses that driver to send and receive messages for the application. The programmer is required to instantiate a Driver and a Manager, and to initialise the Manager. Thereafter the facilities of the Manager can be used to send and receive messages.

It is also possible to use a Driver on its own, without a Manager, although this only allows unaddressed, unreliable transport via the Driver's facilities.

In some specialised use cases, it is possible to instantiate more than one Driver and more than

one Manager.

A range of different common embedded microprocessor platforms are supported, allowing your project to run on your choice of processor.

1.2 RADIO SIGNAL STRENGTH INDICATOR

Returns the most recent RSSI (Receiver Signal Strength Indicator). Usually it is the RSSI of the last received message, which is measured when the preamble is received. If you called readRssi() more recently, it will return that more recent value.

Returns

1. The most recent RSSI measurement in dBm.
2. Re-implemented from RHGenericDriver.

References RHGenericDriver::lastRssi().

1.2.1. RSSI CHECK AT RECEIVING MODULE

```
uint8_t lastRssi = rf_driver.lastRssi();
// Serial.println(rf_driver.lastRssi(), DEC);
int16_t iRssi = (rf_driver.lastRssi(), DEC);
// String sRssi = String((rf_driver.lastRssi(), DEC));
```