CHAPTER-1

INTRODUCTION

INTRODUCTION

This project presents the overall design of Home Automation System (HAS) with low cost and wireless system. It specifically focuses on the development of an IOT based home automation system that is able to control various components via internet or be automatically programmed to operate from ambient conditions. In this project, we design the development of a firmware for smart control which can successfully be automated minimizing human interaction to preserve the integrity within whole electrical devices in the home. We used Node MCU, a popular open source IOT platform, to execute the process of automation. Different components of the system will use different transmission mode that will be implemented to communicate the control of the devices by the user through Node MCU to the actual appliance. The main control system implements wireless technology to provide remote access from smart phone. We are using a cloud server-based communication that would add to the practicality of the project by enabling unrestricted access of the appliances to the user irrespective of the distance factor. We provided a data transmission network to create a stronger automation. The system intended to control electrical appliances and devices in house with relatively low cost design, user-friendly interface and ease of installation. The status of the appliance would be available, along with the control on an android platform. This system is designed to assist and provide support in order to 1fulfil the needs of elderly and disabled in home. Also, the smart home concept in the system improves the standard living at home

BACKGROUND

In this technical era people wants most of the work to be done automatically. To implement this thing Internet of Thing is one of the best way to complete any particular task. High noise levels can contribute to cardiovascular effects in humans and an increased incidence of coronary artery disease. In animals, noise can increase the risk of death by altering predator or prey detection and avoidance, interfere with reproduction and navigation, and contribute to permanent hearing loss. A substantial amount of the noise that humans produce occurs in the ocean. Up until recently, most research on noise impacts has been focused on marine mammals, and to a lesser degree, fish. In the past few years, scientists have shifted to conducting studies on invertebrates and their responses to anthropogenic sounds in the marine environment. This research is essential, especially considering that invertebrates make up 75% of marine species, and thus compose a large percentage of ocean food webs. Of the studies that have been conducted, a sizable variety in families of invertebrates have been represented in the research. A variation in the complexity of their sensory systems exists, which allows scientists to study a range of characteristics and develop a better understanding of anthropogenic noise impacts on living organisms.

OBJECTIVE

Realtime monitoring system that monitors environmental information as it happens, then sends out a warning enabling action to be taken in a timely manner to preserve valuable products. **Fully automated** monitoring, warning and reporting system requiring minimal human input, saving time and money as well as reducing the potential for human error.

SCOPE

Monitor temperature, humidity, CO_2, room pressure differential, power fail, door open, all in real time

Each unit can be individually programmed for parameters, alerts, reports, etc

Regular automatic reports to QA Management in PDF format, with critical data backup

100% reliable – in the event of power failure TAD data loggers continue to log and the data is later automatically extracted to complete the TAD reports, i.e., no data is lost. This has been a huge advantage for our customers with critical products when correlating data has been unavailable.

Portable – The monitoring hardware can be moved easily with cabinets as required.

Each TAD system will support 240 probes. If requiring more than 240 probes then multiple TAD systems can be run, linking with each other via remote access.

TAD Software is "Hands off" automated system which is easy to use, but comprehensive. All channels are easily configurable for individual parameters and alert recipients. Temprecord will pre program with client requirements if provided. Training is provided with new installations. All Temprecord software is free.

Calibrations are built into the TAD temperature and humidity loggers. No requirement for a correction formula to be applied to the downloaded data after recalibration, therefore potential for human error is removed. Data is available with 0.01°C display resolution.

CHAPTER -2

LITERATURE SURVEY

The IoT uses a wide variety of devices, protocols, technologies, networks, middleware, applications, systems, and data, all forming a heterogeneous network. This will increase the degree of interoperability and complexity. Because of this situation, several groups such as ITU,
ETSI, OpenIoT are developing interoperability standards and IoT protocols, among others. However, the high fragmentation and development of vertical IoT systems have increased in a multi-standard context, where features, functions and devices are combined [5]. In [6] an energy and position-based IoT system are controlled in networks, which are also based on a smartphone and cloud computing platform. This provides energy efficiency in buildings and organizations, as a complete system on a large scale. The problem with this proposal is that it is difficult and costly to implement, so it is not recommended to be applied in ordinary smart houses. A conceptual IoT device called an AAL-IoTSys, which includes a Smart Gateway as a key component is proposed in [5]. This enables many heterogeneous devices to be interoperable across different networks, protocols and technologies eg. WiFi, ZigBee, Bluetooth, IEEE 802.15.4 and 6LowPAN. Many research papers and studies in this area do not emphasize microcontroller sensor outputs for data storage and data acquisition. In this article, the resulting data collected from the device can be concurrently processed and charted synchronously with a weather station monitoring system [3]. That is, the details can be shown and seen, directly and indirectly, in two approaches. The word 'direct' implies that the weather can be viewed directly via the NET PI network platform; whereas indirect methodology ensures that weather patterns are recorded and stored in a computer as long as the sensors calculate climatic conditions [3, 7]. The main difficulty of this research is to demonstrate and validate that microcontrollers can be
connected to a data acquisition network with their sensors to build a database system based on
the characteristics of the weather. The current idea helps the microcontroller sensors to predict
possibilities centred on the perceived data rather than strictly tracking the device. Economically,
a single sensor known as DHT sensor is used by the proposed system to provide temperature and humidity readings that were used to create the heretical framework of the climate database

CHAPTER -3

THEORY

IOT (INTERNET OF THINGS)

IOT as a term has evolved long way as a result of convergence of multiple technologies, machine learning, embedded systems and commodity sensors. IOT is a system of interconnected devices assigned a UIDS, enabling data transfer and control of devices over a network. It reduced the necessity of actual interaction in order to control a device. IOT is an advanced automation and analytics system which exploits networking, sensing, big data, and artificial intelligence technology to deliver complete systems for a product or service. These systems allow greater transparency, control, and performance when applied to any industry or system.

Features of IOT

Intelligence

IOT comes with the combination of algorithms and computation, software & hardware that makes it smart. Ambient intelligence in IOT enhances its capabilities which facilitate the things to respond in an intelligent way to a particular situation and supports them in carrying out specific tasks. In spite of all the popularity of smart technologies, intelligence in IOT is only concerned as a means of interaction between devices, while user and device interaction are achieved by standard input methods and graphical user interface.

Connectivity

Connectivity empowers the Internet of Things by bringing together everyday objects. Connectivity of these objects is pivotal because simple object level interactions contribute towards collective intelligence in the IOT network. It enables network accessibility and compatibility in the things. With this connectivity, new market opportunities for the Internet of things can be created by the networking of smart things and applications.

Dynamic Nature

The primary activity of Internet of Things is to collect data from its environment, this is achieved with the dynamic changes that take place around the devices. The state of these devices change dynamically, example sleeping and waking up, connected and/or disconnected as well as the context of devices including temperature, location and speed. In addition to the state of the device, the number of devices also changes dynamically with a person, place and time.

Enormous Scale

The number of devices that need to be managed and that communicate with each other will be much larger than the devices connected to the current Internet. The management of data generated from these devices and their interpretation for application purposes becomes more critical. Gartner (2015) confirms the enormous scale of IOT in the estimated report where it stated that 5.5 million new things will get connected every day and 6.4 billion connected things will be in use worldwide in 2016,

which is up by 30 percent from 2015. The report also forecasts that the number of connected devices will reach 20.8 billion by 2020.

Sensing

IOT wouldn't be possible without sensors that will detect or measure any changes in the environment to generate data that can report on their status or even interact with the environment. Sensing technologies provide the means to create capabilities that reflect a true awareness of the physical world and the people in it. The sensing information is simply the analog input from the physical world, but it can provide a rich understanding of our complex world.

Heterogeneity

Heterogeneity in Internet of Things as one of the key characteristics. Devices in IOT are based on different hardware platforms and networks and can interact with other devices or service platforms through different networks. IOT architecture should support direct network connectivity between heterogeneous networks. The key design requirements for heterogeneous things and their environments in IOT are scalabilities, modularity, extensibility and interoperability.

Security

IOT devices are naturally vulnerable to security threats. As we gain efficiencies, novel experiences, and other benefits from the IOT, it would be a mistake to forget about security concerns associated with it. There is a high level of transparency and privacy issues with IOT. It is important to secure the endpoints, the networks, and the data that is transferred across all of it means creating a security paradigm.

Advantages of IOT

Communication

IOT encourages the communication between devices, also famously known as Machine-to-Machine (M2M) communication. Because of this, the physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and greater quality.

Automation and Control

Due to physical objects getting connected and controlled digitally and centrally with wireless infrastructure, there is a large amount of automation and control in the workings. Without human intervention, the machines are able to communicate with each other leading to faster and timely output.

Information

It is obvious that having more information helps making better decisions. Whether it is mundane decisions as needing to know what to buy at the grocery store or if your company has enough widgets and supplies, knowledge is power and more knowledge is better.

Monitor

The second most obvious advantage of IOT is monitoring. Knowing the exact quantity of supplies or the air quality in your home, can further provide more information that could not have previously been collected easily. For instance, knowing that you are low on milk or printer ink could save you another trip to the store in the near future. Furthermore, monitoring the expiration of products can and will improve safety.

As hinted in the previous examples, the amount of time saved because of IOT could be quite large. And in
today's modern life. We all could use more time.

Money

The biggest advantage of IOT is saving money. If the price of the tagging and monitoring equipment is less than the amount of money saved, then the Internet of Things will be very widely adopted. IOT fundamentally proves to be very helpful to people in their daily routines by making the appliances communicate to each other in an effective manner thereby saving and conserving energy and cost. Allowing the data to be communicated and shared between devices and then translating it into our required way, it makes our systems efficient.

Automation of daily tasks leads to better monitoring of devices

The IOT allows you to automate and control the tasks that are done on a daily basis, avoiding human intervention. Machine-to-machine communication helps to maintain transparency in the processes. It also leads to uniformity in the tasks. It can also maintain the quality of service. 3.1.2.8 Efficient and Saves Time

Saves Money

Optimum utilization of energy and resources can be achieved by adopting this technology and keeping the devices under surveillance. We can be alerted in case of possible bottlenecks, breakdowns, and damages to the system. Hence, we can save money by using this technology.

Better Quality of Life

All the applications of this technology culminate in increased comfort, convenience, and better management, thereby improving the quality of life.

Disadvantages of IOT

Compatibility

Currently, there is no international standard of compatibility for the tagging and monitoring equipment. I believe this disadvantage is the most easy to overcome. The manufacturing companies of these equipment just need to agree to a standard, such as Bluetooth, USB, etc. This is nothing new or innovative needed.

Complexity

As with all complex systems, there are more opportunities of failure. With the Internet of Things,

failures could sky rocket. For instance, let's say that both you and your spouse each get a message saying that your milk has expired, and both of you stop at a store on your way home, and you both purchase milk. As a result, you and your spouse have purchased twice the amount that you both need. Or maybe a bug in the software ends up automatically ordering a new ink cartridge for your printer each and every hour for a few days, or at least after each power failure, when you only need a single replacement.

Privacy/Security

With all of this IOT data being transmitted, the risk of losing privacy increases. For instance, how well encrypted will the data be kept and transmitted with? Do you want your neighbours or employers to know what medications that you are taking or your financial situation?

Safety

Imagine if a notorious hacker changes your prescription. Or if a store automatically ships you an equivalent product that you are allergic to, or a flavour that you do not like, or a product that is already expired. As a result, safety is ultimately in the hands of the consumer to verify any and all automation.

As all the household appliances, industrial machinery, public sector services like water supply and transport, and many other devices all are connected to the Internet, a lot of information is available on it. This information is prone to attack by hackers. It would be very disastrous if private and confidential information is accessed by unauthorized intruders.

Lesser Employment of Menial Staff

The unskilled workers and helpers may end up losing their jobs in the effect of automation of daily activities. This can lead to unemployment issues in the society. This is a problem with the advent of any technology and can be overcome with education. With daily activities getting automated, naturally, there will be fewer requirements of human resources, primarily, workers and less educated staff. This may create Unemployment issue in the society.

Technology Takes Control of Life

Our lives will be increasingly controlled by technology, and will be dependent on it. The younger generation is already addicted to technology for every little thing. We have to decide how much of our daily lives are we willing to mechanize and be controlled by technology.

Application Grounds of IOT

Wearables

Wearable technologies is a hallmark of IOT applications and is one of the earliest industries to have deployed IOT at its services. Fit Bits, heart rate monitors, smartwatches, glucose monitoring devices reflect the successful applications of IOT.

Smart homes

This area of application concerned to this particular project, so a detailed application is discussed further. Jarvis, an AI home automation employed by Mark Zuckerberg, is a remarkable example in this field of application.

Health care

IOT applications have turned reactive medical based system into proactive wellness based system. IOT focuses on creating systems rather than equipment. IOT creates a future of medicine and healthcare which exploits a highly integrated network of sophisticated medical devices. The integration of all elements provides more accuracy, more attention to detail, faster reactions to events, and constant improvement while reducing the typical overhead of medical research and organizations

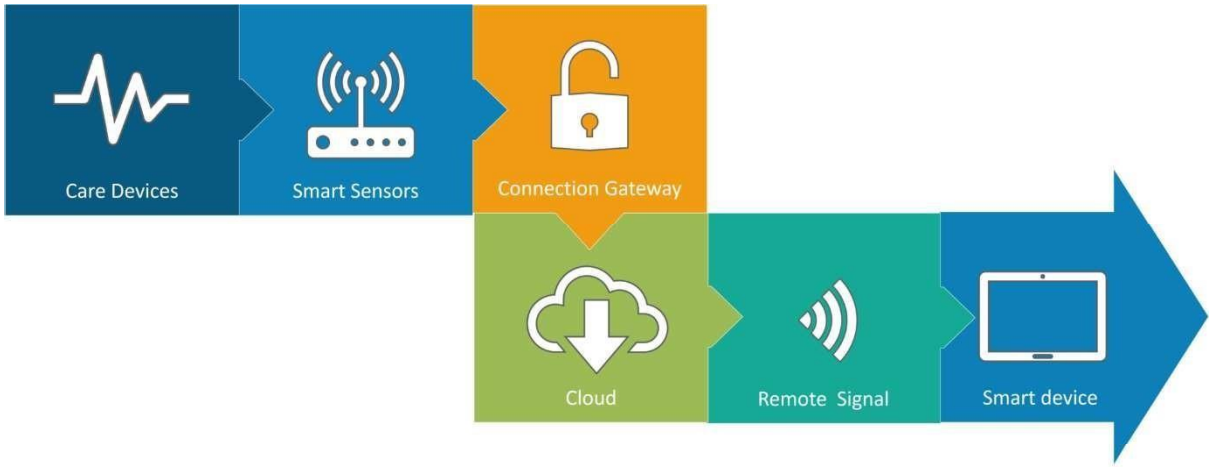


Figure 2. Working of IOT enables care devices.

Agriculture

A greenhouse farming technique enhances the yield of crops by controlling environmental parameters. However, manual handling results in production loss, energy loss, and labour cost, making the process less effective. A greenhouse with embedded devices not only makes it easier to be monitored but also, enables us to control the climate inside it. Sensors measure different parameters according to the plant requirement and send it to the cloud. It, then, processes the data and applies a control action.

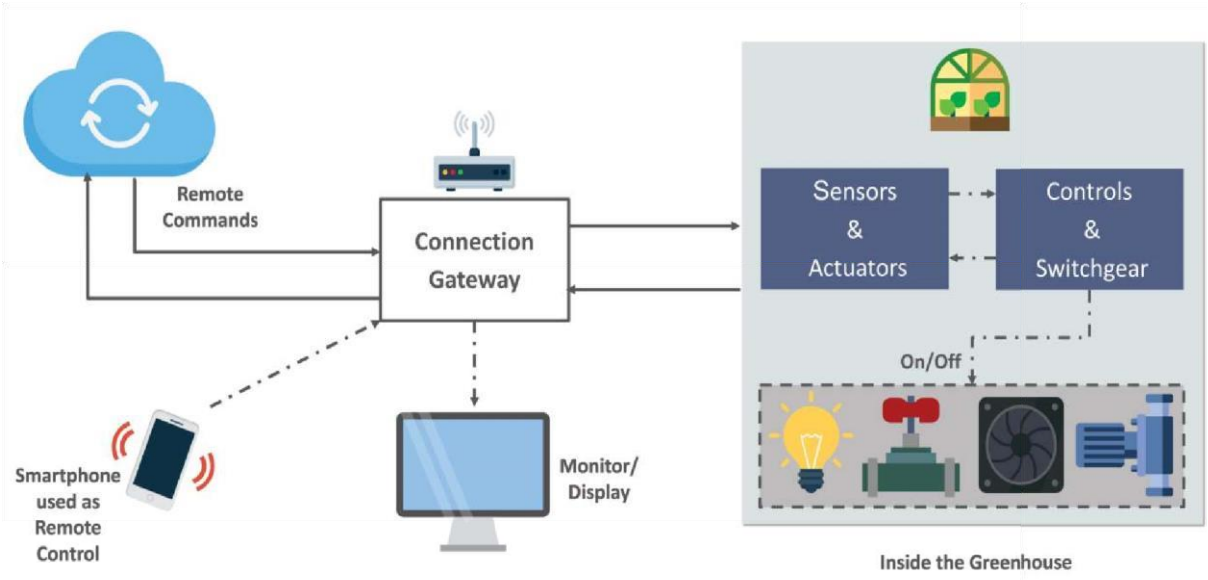


Figure 3. IOT controlled greenhouse environment.

Industrial Automation

For a higher return of investment this field requires both fast developments and quality of products. This vitality thus coined the term IIOT. This whole schematic is re-engineered by IOT applications. Following are the domains of IOT applications in industrial automation

- Factory Digitalization
- Product flow Monitoring
- Inventory Management
- Safety and Security
- Quality Control

Packaging optimization

Logistics and Supply Chain Optimization

Government and Safety

IOT applied to government and safety allows improved law enforcement, defence, city planning, and economic management. The technology fills in the current gaps, corrects many current flaws, and expands the reach of these efforts. For example, IOT can help city planners have a clearer view of the impact of their design, and governments have a better idea of the local economy.

IOT Technologies and Protocols

Several communication protocols and technologies cater to and meet the specific functional requirements of IOT system.

Bluetooth

Bluetooth is a short range IOT communication protocol/technology that is profound in many consumer product markets and computing. It is expected to be key for wearable products in particular, again connecting to the IOT albeit probably via a smartphone in many cases. The new Bluetooth Low-Energy (BLE) or Bluetooth Smart, as it is now branded is a significant protocol for IOT applications. Importantly, while it offers a similar range to Bluetooth it has been designed to offer significantly reduced power consumption.

Zigbee

ZigBee is similar to Bluetooth and is majorly used in industrial settings. It has some significant advantages in complex systems offering low-power operation, high security, robustness and high and is well positioned to take advantage of wireless control and sensor networks in IOT applications. The latest version of ZigBee is the recently launched 3.0, which is essentially the unification of the various ZigBee wireless standards into a single standard.

Z-Wave

Z-Wave is a low-power RF communications IOT technology that primarily design for home automation for products such as lamp controllers and sensors among many other devices. A ZWave uses a simpler protocol than some others, which can enable faster and simpler development, but the only maker of chips is Sigma Designs compared to multiple sources for other wireless technologies such as ZigBee and others.

Wi-Fi

Wi-Fi connectivity is one of the most popular IOT communication protocol, often an obvious choice for many developers, especially given the availability of Wi-Fi within the home environment within LANs. There is a wide existing infrastructure as well as offering fast data transfer and the ability to handle high quantities of data. Currently, the most common Wi-Fi standard used in homes and many businesses is 802.11n, which offers range of hundreds of megabit per second, which is fine for file transfers but may be too power-consuming for many IOT applications.

Cellular

Any IOT application that requires operation over longer distances can take advantage of GSM/3G/4G cellular communication capabilities. While cellular is clearly capable of sending high quantities of data, especially for 4G, the cost and also power consumption will be too high for many applications. But it can be ideal for sensorbased low-bandwidth-data projects that will send very low amounts of data over the Internet.

NFC

NFC (Near Field Communication) is an IOT technology. It enables simple and safe communications between electronic devices, and specifically for smartphones, allowing consumers to perform transactions in which one does not have to be physically present. It helps the user to access digital content and connect electronic devices. Essentially it extends the capability of contactless card technology and enables devices to share information at a distance that is less than 4cm.

LoRaWAN

LoRaWAN is one of popular IOT Technology, targets wide-area network (WAN) applications. The

LoRaWAN design to provide low-power WANs with features specifically needed to support low-cost mobile secure communication in IOT, smart city, and industrial applications. Specifically meets requirements for lowpower consumption and supports large networks with millions and millions of devices, data rates range from 0.3 kbps to 50 kbps.

IOT software

IOT software addresses its key areas of networking and action through platforms, embedded systems, partner systems, and middleware. These individual and master applications are responsible for data collection, device integration, real-time analytics, and application and process extension within the IOT network. They exploit integration with critical business systems (e.g., ordering systems, robotics, scheduling, and more) in the execution of related tasks.

Data Collection

This software manages sensing, measurements, light data filtering, light data security, and aggregation of data. It uses certain protocols to aid sensors in connecting with real-time, machineto-machine networks. Then it collects data from multiple devices and distributes it in accordance with settings. It also works in reverse by distributing data over devices. The system eventually transmits all collected data to a central server.

Device Integration

Software supporting integration binds (dependent relationships) all system devices to create the body of the IOT system. It ensures the necessary cooperation and stable networking between devices. These applications are the defining software technology of the IOT network because without them, it is not an IOT system. They manage the various applications, protocols, and limitations of each device to allow communication.

Real-Time Analytics

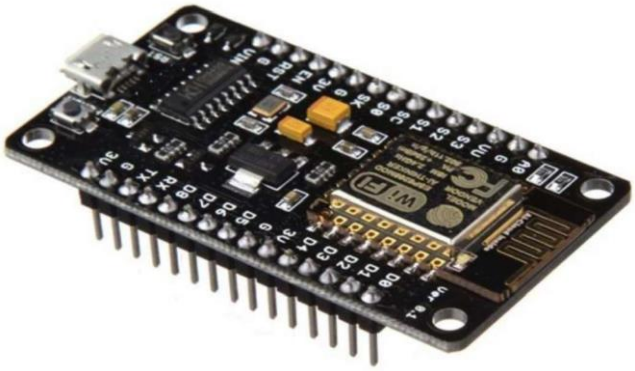
These applications take data or input from various devices and convert it into feasible actions or clear patterns for human analysis. They analyse information based on various settings and designs in order to perform automation-related tasks or provide the data required by industry.

Application and Process Extension

These applications extend the reach of existing systems and software to allow a wider, more effective system. They integrate predefined devices for specific purposes such as allowing certain mobile devices or engineering instruments access. It supports improved productivity and more accurate data collection.

NODE MCU:

NodeMCU (Node Microcontroller Unit) 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 (Node Microcontroller Unit) 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.

The firmware uses the Lua scripting language. The firmware is based on the eLua project, and built on the Express if 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 was based on the ESP-12 module of the ESP8266, which is a Wi-Fi SoC integrated witha Tensilica Xtensa LX106 core, widely used in IOT applications.

Pin Configuration of Node MCU Development Board

This module provides an access to the GPIO subsystem. All the access is based on I/O index number of Node MCU kits, not the internal GPIO pins. For example, the D0 pin on the development kit is mapped to GPIO pin 16. Node MCU provides access to the GPIO pins and the following pin mappingtable is a part of the API documentation.

PIN NAME ON NODE MCU DEVELOPMENT KIT	ESP8266 INTERNAL GPIO PIN NUMBER	PIN NAME ON NODE MCU DEVELOPMENT KIT	ESP8266 INTERNAL GPIO PIN NUMBER
0 [*]	GPIO16	**7**	GPIO13
1	GPIO5	**8**	GPIO15
2	GPIO4	**9**	GPIO3
3	GPIO0	**10**	GPIO1
4	GPIO2	**11**	GPIO9
5	GPIO14	**12**	GPIO10
6	GPIO12		

Figure 4. Node MCU Development Board

[*] D0 (GPIO16) can only be used for GPIO read/write. It does not support open-drain/interrupt/PWM/I²C or 1-Wire.

The ESP8266 Node MCU has total 30 pins that interface it to the outside world. The pins are groupedby their functionality as:

Power pins: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used todirectly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

a ground pin of ESP8266 Node MCU development board.

12 IC Pins: are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins: ESP8266 Node MCU has 17 GPIO pins which can be assigned to various functions such asI2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabledGPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configuredas an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts

ADC Channel: The Node MCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltageof TOUT pin. However, they cannot be implemented at the same time.

Table 1. Node MCU index ↔ GPIO mapping.

[*] D0 (GPIO16) can only be used for GPIO read/write. It does not support open-drain/interrupt/PWM/I²C or 1-Wire.

The ESP8266 Node MCU has total 30 pins that interface it to the outside world. The pins are groupedby their functionality as:

Power pins: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used todirectly supply the ESP8266 and its peripherals, if you have a regulated 5V voltage source. The 3.3V pins are the output of an on-board voltage regulator. These pins can be used to supply power to external components.

GND: is a ground pin of ESP8266 Node MCU development board

12 IC Pins: are used to hook up all sorts of I2C sensors and peripherals in your project. Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized

programmatically, and the clock frequency is 100 kHz at a maximum. It should be noted that I2C clock frequency should be higher than the slowest clock frequency of the slave device.

GPIO Pins: ESP8266 Node MCU has 17 GPIO pins which can be assigned to various functions such asI2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabledGPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configuredas an input, it can also be set to edge-trigger or level-trigger to generate CPU interrupts

ADC Channel: The Node MCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC viz. Testing power supply voltage of VDD3P3 pin and testing input voltageof TOUT pin. However, they cannot be implemented at the same time.

asynchronous communication (RS232 and RS485), and can communicate at up to 4.5 Mbps. UART0 (TXD0, RXD0, RST0 & CTS0 pins) can be used for communication. It supports fluid control. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

SPI Pins: ESP8266 features two SPIs (SPI and HSPI) in slave and master modes. These SPIs alsosupport the following general-purpose SPI features:

- 4 timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks
- of 80 MHzUp to 64-Byte FIFO

SDIO Pins: ESP8266 features Secure Digital Input/output Interface (SDIO) which is used to directlyinterface SD cards. 4-bit 25 MHz SDIO v1.1 and 4-bit 50 MHz SDIO v2.0 are supported.

PWM Pins: The board has 4 channels of Pulse Width Modulation (PWM). The PWM is adjustable from 1000 µs to 10000 µs, i.e., between 100 Hz and 1 kHz.
output can beimplemented programmatically and used for driving digital motors and LEDs. PWM frequency range

Control Pins: are used to control ESP8266. These pins include Chip Enable pin (EN), Reset pin (RST)and WAKE pin.

- EN pin The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOWthe chip works at minimum power.
- RST pin RST pin is used to reset the ESP8266 chip.
- WAKE pin Wake pin is used to wake the chip from deep-sleep.

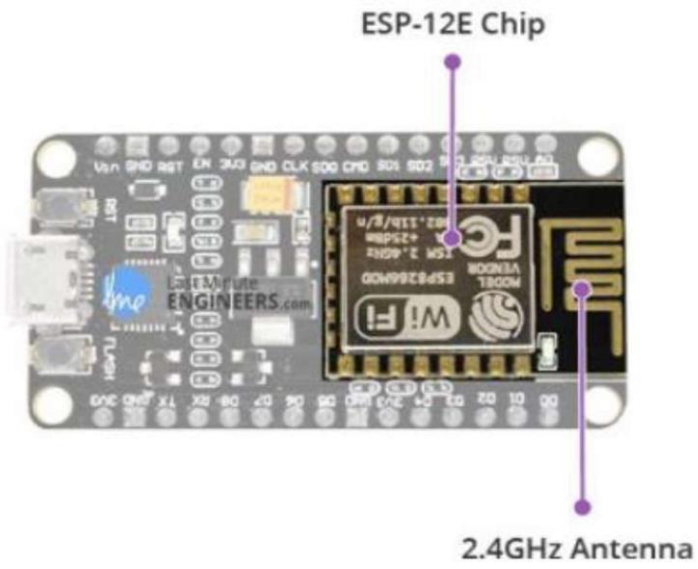
Parts of Node MCU Development Board

ESP 12-E Module

The development board equips the ESP-12E module containing ESP8266 chip having Tensilica Xtensa® 32-bit LX106 RISC microprocessor which operates at 80 to 160 MHz adjustable clock frequency and supports RTOS.

There's also 128 KB RAM and 4MB of Flash memory (for program and data storage) just enough to cope with the large strings that make up web pages, JSON/XML data, and everything we throw at IOTdevices nowadays.

The ESP8266 Integrates 802.11b/g/n HT40 Wi-Fi transceiver, so it can not only connect to a Wi-Fi network and interact with the Internet, but it can also set up a network of its own, allowing other devices to connect directly to it. This makes the ESP8266 Node MCU even more versatile.



- Tensilica Xtensa® 32-bit LX106 80 to 160 MHz clock frequency 128 kb internal RAM
- 4 MB external flash
- 802.11b/g/n HT40 Wi-Fi transceiver

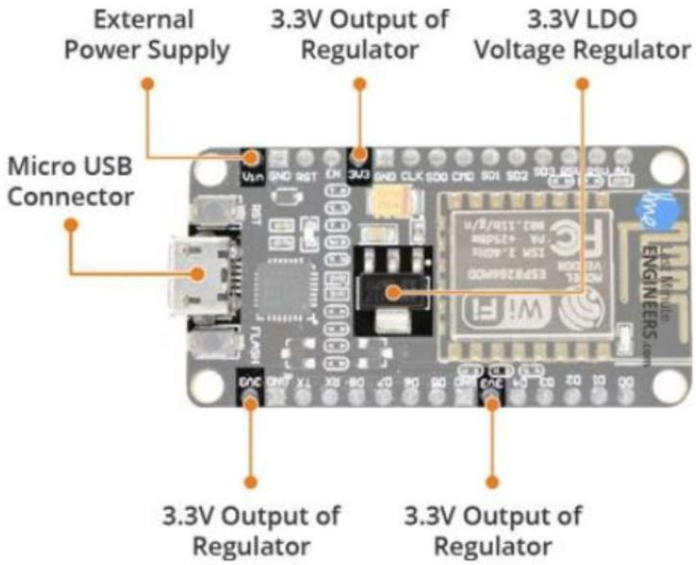
Figure 6. ESP 12E module in Node MCU Development board

.

Power Requirements

As the operating voltage range of ESP8266 is 3V to 3.6V, the board comes with a LDO voltage regulator to keep the voltage steady at 3.3V. It can reliably supply up to 600mA, which should be more than enough when ESP8266 pulls as much as 80mA during RF transmissions. The output of theregulator is also broken out to one of the sides of the board and labelled as 3V3. This pin can be used to supply power to external components.

Power to the ESP8266 Node MCU is supplied via the on-board Micro B USB connector. Alternatively,if you have a regulated 5V voltage source, the VIN pin can be used to directly supply the ESP8266 andits peripherals.



erating voltage
V to 3.6V On-
ırd 3.6V 600mA
ulator80 mA
erating current

- 20 µA during sleep mode

Figure 7. Power module on a Node MCU development board.

Peripheral I/O

The ESP8266 Node MCU has total 17 GPIO pins broken out to the pin headers on both sides of thedevelopment board. These pins can be assigned to all sorts of peripheral duties, including:

- ADC channel A 10-bit ADC channel.
- UART interface UART interface is used to load code serially.
- PWM outputs PWM pins for dimming LEDs or controlling motors.
- SPI, I2C & I2S interface SPI and I2C interface to hook up all sorts of sensors andperipherals.
- I2S interface I2S interface if you want to add sound to your project.

As a result of the pin multiplexing feature (Multiple peripherals multiplexed on a single GPIO pin), asingle GPIO pin can act as PWM/UART/SPI.

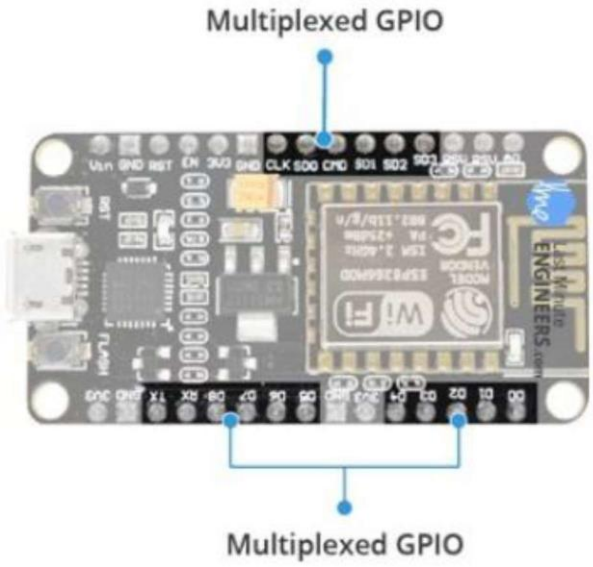
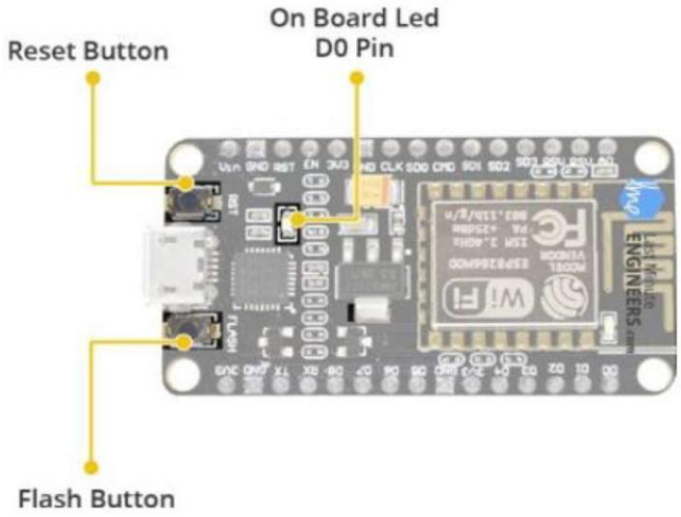


Figure 8. GPIO pins on Node MCU development board.

On Board Switches and LED Indicators

The ESP8266 Node MCU features two buttons. One marked as RST located on the top left corner is the Reset button, used of course to reset the ESP8266 chip. The other FLASH button on the bottom left corner is the download button used while upgrading firmware. The board also has a LED indicator which is user programmable and is connected to the D0 pin of the board.

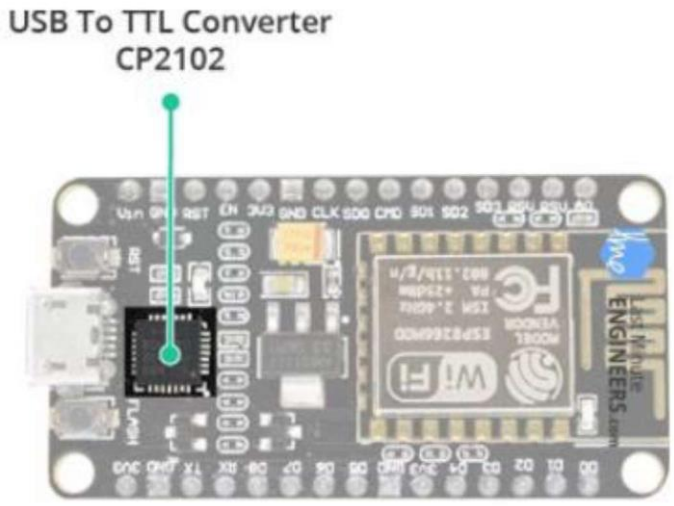


Switches and indicators

- RST: Reset the ESP8266 chip
- FLASH: Download new
- programsBlue LED: User
- programmable

Serial Communication

The board includes CP2102 USB-to-UART Bridge Controller from Silicon Labs, which converts USBsignal to serial and allows your computer to program and communicate with the ESP8266 chip.



- CP2120 USB-to-UART converter
- 4.5 Mbps
- communication speed
- Flow control support

Figure 10. CP2120 on Node MCU development board.

UART Pins: ESP8266 Node MCU has 2 UART interfaces, i.e. UART0 and UART1, which provide

Installation of Node MCU

Mostly these days devices download and install drivers on their own, automatically. Windows doesn't know how to talk to the USB driver on the Node MCU so it can't figure out that the board is a Node

MCU and proceed normally. Node MCU Amica is an ESP8266 Wi-Fi module based development board. It has got Micro USB slot that can directly be connected to the computer or other USB host devices. Ti has got 15X2 header pins and a Micro USB slot, the headers can be mounted on a breadboard and Micro USB slot is to establish connection to USB host device. It has CP2120 USB to serial converter. In order to install CP2120 (USB to serial converter), user is needed to download thedriver for the same. Once user downloads drivers as per its respective operating system, the systemestablishes connection to Node MCU. The user needs to node down the COM post allotted to newlyconnected USB device (Node MCU) from device manager of the system. This com port number will be required while using Node MCU Amica. As the CP2120 driver is been installed, the Node MCU canbe programmed using Arduino IDE software by coding in embedded C. this requires ESP8266 board installation in Arduino IDE from board manager, and assigning communication

CHAPTER -4

SOFTWARE REQUIREMENTS

Hardware Requirements

• Operating System Version: Microsoft® Windows® 7/8/10 (32- or 64-bit) The Android Emulator only supports 64-bit Windows.
• Random Access Memory (RAM): 4 GB RAM minimum; 8 GB RAM recommended
• Free digital storage: 2 GB of available digital storage minimum, 4 GB Recommended (500 MB for IDE + 1.5 GB for Android SDK and emulator system image).

Software Requirement

i. Android Studio
ii. Programming languages: Java
iii. Minimum required JDK version: Java Development Kit 8 Minimum screen resolution: 1280 x 800
iV. Firebase: Acting as a Database.

CHAPTER-5

SYSTEM DESIGN

System designing is a modelling process. It is a solution, how to approach to create a new system. It can be defined as a transition from user's view to programmer's or database person's view. The design phase mainly depends on the detailed specification in the feasibility study. The system design phase acts as a bridge between the requirement specification and the implementation phase.

From a project management point of view software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data and software architecture. Detailed design focuses on refinement to the architectural representation that leads to detailed data structure and algorithmic representation for software.

The major steps in the design phase are input design, output design, and dealing with the coding issue. The very first step is design of input and output screen to the client requirements. Next comes the various issues that should be dealt coding and the code should be such that it should be compatible with the real time environment and should be generic in nature.

System design is a process through which requirements are translated into a representation of software. Initially the representation depicts a holistic view of software. Subsequent refinement leads to a design presentation that is very close to source code. Design is a place where quality fostered in software development. Design provides us with representation of software that can be assessed for quality, this is the only way that can accurately translate the customer requirements into finished software product or system.

Conceptual Design

Conceptual Design is the process of acquiring and evaluating, documenting and then validating what the user envisions to be business relation. It identifies the user and business requirements of the application and leads to a business solution as seen by the user.

All applications are built to solve business problems and it is important to pay close attention to principle that the business need drives application development. At any point the design process, the current state of the design should be directly traceable to a business problem and requirements.

To achieve this conceptual design is driven by developing usage scenarios. These scenarios are a direct representation of the user's view of the solution to a specific business problem. A

conceptual view places the emphasis on solving a business problem and deriving a solution that corresponds to the needs and requirement of the users. It is based on deriving the behaviour of the solution with a primary emphasizes on the user. Beginning with an emphasis on the activities of the business rather than aspects of software development, underscores the fact that systems exist to serve the business. A strong focus on the user in the beginning of the project will help in maintaining a proper perspective throughput the development lifecycle. The conceptual design results in the first description of what the system does to solve the business problem articulated in the vision /scope document.

Logical Design

Logical Design derives business objects and their related services directly from these usage scenarios. The logical view of the solution provides a basis for evaluating different physical options. It also formalizes the solution for the project team.

The idea of the application is that the system first emerges in logical design. Its boundaries and business objects and it contain the system definition. Logical design specifies the interfaces between the system and external entities, such as users and other systems. Within a system there may be a number of sub-systems, and these boundaries are also specified.

ER Diagram

ER-modelling is a data modelling technique used in software engineering to produce a conceptual data model of a information system. Diagrams created using this ER-modelling technique are called Entity-Relationship Diagrams or ER diagrams or ERDs

As entity relationship diagrams (ERD) shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.

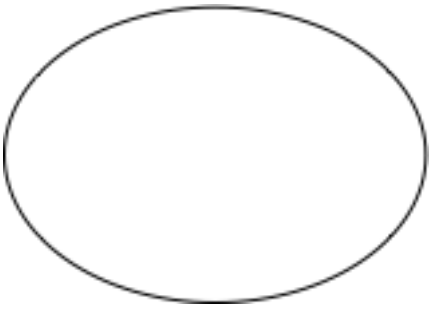
At first glance an entity relationship diagram looks very much like a flowchart. It is the specialized symbol, and the meanings of those symbols, that make it unique. Entity-relationship analysis uses three major abstractions to describe data.

- **Entity**

  Entity is represented by rectangles. An entity is an object or concept about which you want to store information

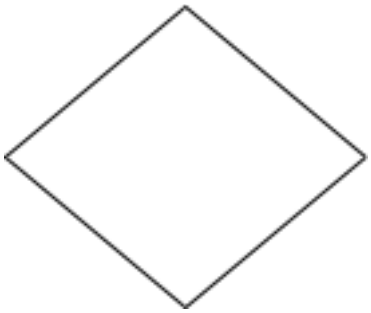
- **Attribute**

  Attributes are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the employee's key attribute



- **Relationship**

Relationship is represented by diamond shapes, show how two entities share information in the database.



Some cases, entity can be self-linked. For example, employees can supervise other employees

Multi-valued attribute:

A multivalued attribute can have more than one value. For example, an employee entity can be multi skill value

Data Flow Digram

- A data flow diagram shows the logical flow of data through a transaction processing system of an organization.

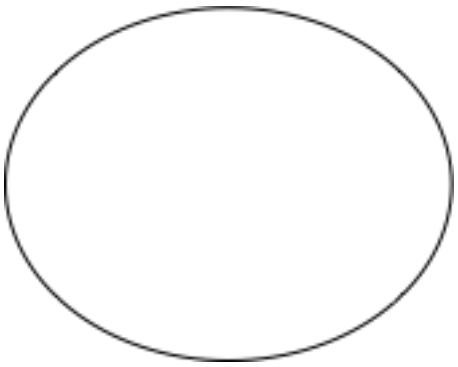
- They are primarily used in the system development process as a tool for analyzing an existing system.

Data Flow:

- Data move in specific direction from an origin to a destination in the form of a document.

Process:

- Procedure or devices that use or transform data



Source or Destination of Data:

- Source or Destination of data, which may be people, organization or other entities, interact with the system but are outside its boundary.

Feasibility Study

A feasibility study is an important phase in the software development process. It enables the developer to have an assessment of the product being developed. It refers to the feasibility study of the product in terms of outcome of the product, operational use and technical support

required for implementing it. Feasibility study should be performed on the basis of various criteria and parameters. The various feasibility studies are:

a) Economic Feasibility
b) Operational Feasibility
c) Technical Feasibility
d) Time-schedule Feasibility
e) Implementation Feasibility

Technical Feasibility

Technical Feasibility corresponds to determination of whether it is technically feasible to develop the software.

1. Necessary technology exists to do what is suggested and required by the organization.
2. The proposed equipment's have the technical capacity to hold the data required to use the new system
3. The proposed system will provide adequate response to inquiries regardless of the location if users.
4. The hardware needed to develop and implement the system is adequate.
5. The software guarantees accuracy, reliability and ease of access and data security.

Economic Feasibility

A system that can be developed and that will be used if installed must still be a good investment for the organization. Financial benefits must equal or exceed the costs. The financial and economic issues are raised are as under:

- No extra is incurred for developing the system. As required software are already used by the department

- No extra cost for the modification or addition of software and hardware will require in case of future expansion of the current system.

- As the project is to be developed by developed by trainees the cost incurred by the company is in the form of resource allocation rather than monetary. The cost on the company is indirect in the form of resources utilization.

- The company will be at profit if they implement this system because of the cost of implementation is nominal as compared to the profit they will be earning in terms of efficiency.
  Considering above factors project is economically feasible.

Operational Feasibility

Operational feasibility focuses on whether the system will work when it is developed and installed.
Operationally the system is feasible because:

- There is sufficient support for the project from management and user. The system is well liked and used to the extent that persons will not be able to see reasons for change.
- The current business methods are not acceptable because the manual system is time consuming.
- The users though initially repressive worked along with the development team once the initial doubts were cleared.
- The users have been involved in the planning and development of the project. This reduces the chances of resistance to the system.
- The proposed system will not cost any harm to the existing system and its users.
- No special training required for the user as it has a self-explanatory interface. Validation of data input is taken care of by the system and not by the user.
- Since the most trivial of issues assumes a major problematic state later in the development cycle, every possible aspect of operational feasibility was checked. The proposed project passed all the feasibility tests and hence was declared feasible to organization and its functioning.

CHAPTER -6

CODING

```
package thundersharp.aigs.spekteriot;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.view.LayoutInflater;
import android.view.View;

import androidx.appcompat.widget.AppCompatButton;

import com.google.firebase.auth.FirebaseAuth;


public class Progressbars {

    private static Progressbars progressbars = null;
    private Activity activity;

    public Progressbars(){}

    public static Progressbars getInstance(){
        if (progressbars == null){
            progressbars = new Progressbars();
        }
        return progressbars;
    }

    public AlertDialog createDefaultProgressBar(Activity activity){
        AlertDialog.Builder b = new AlertDialog.Builder(activity);

b.setView(LayoutInflater.from(activity).inflate(R.layout.progress_ba
r,null));
        b.setCancelable(false);
        AlertDialog alertDialog = b.create();

        alertDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));

        return alertDialog;
    }

    public void displayExitDialog(Activity activity){
        AlertDialog.Builder b = new AlertDialog.Builder(activity);
        View bottomSheetDialog
=LayoutInflater.from(activity).inflate(R.layout.exit_bottom_sheet,nu
ll);
```

```
        AppCompatButton appCompatButton =
bottomSheetDialog.findViewById(R.id.exit);
        AppCompatButton appCompatButtonCancel =
bottomSheetDialog.findViewById(R.id.no);

        b.setView(bottomSheetDialog);
        AlertDialog alertDialog = b.create();
        alertDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));



        if (appCompatButtonCancel != null && appCompatButton !=
null) {
            appCompatButtonCancel.setOnClickListener(e-
>alertDialog.dismiss());
            appCompatButton.setOnClickListener(f->
activity.finish());
        }

       alertDialog.show();

    }


    public void displayLogoutDialog(Activity activity){
        AlertDialog.Builder b = new AlertDialog.Builder(activity);
        View bottomSheetDialog
=LayoutInflater.from(activity).inflate(R.layout.logout_bottom_sheet,
null);
        AppCompatButton appCompatButton =
bottomSheetDialog.findViewById(R.id.exit);
        AppCompatButton appCompatButtonCancel =
bottomSheetDialog.findViewById(R.id.no);

        b.setView(bottomSheetDialog);
        AlertDialog alertDialog = b.create();
        alertDialog.getWindow().setBackgroundDrawable(new
ColorDrawable(Color.TRANSPARENT));

        if (appCompatButtonCancel != null && appCompatButton !=
null) {
            appCompatButtonCancel.setOnClickListener(e-
>alertDialog.dismiss());
            appCompatButton.setOnClickListener(f-> {
                FirebaseAuth.getInstance().signOut();

//ProfileDataSync.getInstance(activity).initializeLocalStorage().cle
arAllData();
                activity.finish();
                activity.startActivity(new Intent(activity,
MainActivity.class));
            });
        }

        alertDialog.show();
```

```
    }

}

package thundersharp.aigs.spekteriot;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                if (FirebaseAuth.getInstance().getCurrentUser() !=
null){
                    startActivity(new
Intent(MainActivity.this,HomeActivity.class));
                }else
                startActivity(new
Intent(MainActivity.this,Login.class));
                finish();
            }
        },2000);
    }
}
package thundersharp.aigs.spekteriot;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class Login extends AppCompatActivity {

    TextView ID,password;
    private ImageView password_toggle;
    private boolean passwordVis = false;
    private AlertDialog alertDialog;
    private String memberId = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        alertDialog =
Progressbars.getInstance().createDefaultProgressBar(this);

        findViewById(R.id.qrScanner).setOnClickListener(n-
>startActivityForResult(new
Intent(this,BarCodeScanner.class),1001));

        ID = findViewById(R.id.ID);
        password = findViewById(R.id.editText_password);
        password_toggle = findViewById(R.id.password_toggle);


((ImageView)findViewById(R.id.password_toggle)).setOnClickListener(g
->{
            if (passwordVis){

password.setTransformationMethod(PasswordTransformationMethod.getIns
tance());

password_toggle.setImageDrawable(getDrawable(R.drawable.ic_outline_v
isibility_off_24));
                passwordVis = false;
            }else {

password.setTransformationMethod(HideReturnsTransformationMethod.get
Instance());

password_toggle.setImageDrawable(getDrawable(R.drawable.ic_outline_r
emove_red_eye_24));
                passwordVis = true;
            }
        });

        ID.addTextChangedListener(new TextWatcher() {
            @Override
```

```
            public void beforeTextChanged(CharSequence charSequence,
int i, int i1, int i2) {

            }

            @Override
            public void onTextChanged(CharSequence charSequence, int
i, int i1, int i2) {
                if (charSequence.length() == 6){
                    alertDialog.show();
                    FirebaseDatabase
                            .getInstance()
                            .getReference("MEMBERS")
                            .child(charSequence.toString())
                            .addListenerForSingleValueEvent(new
ValueEventListener() {
                                @Override
                                public void onDataChange(@NonNull
DataSnapshot snapshot) {
                                    if (snapshot.exists()){
                                        memberId =
snapshot.child("EMAIL").getValue(String.class);
                                        new
AlertDialog.Builder(Login.this)
                                                .setMessage("Welcome
"+snapshot.child("NAME").getValue(String.class)+" \nEnter your
passcode to continue.")

.setPositiveButton("OK", new DialogInterface.OnClickListener() {
                                                    @Override
                                                    public void
onClick(DialogInterface dialogInterface, int i) {

dialogInterface.dismiss();
                                                    }

}).setCancelable(false).show();
                                        alertDialog.dismiss();
                                    }else {

Toast.makeText(Login.this,"MEMBER NOT FOUND RETRY SCANNING YOUR ID
CARD",Toast.LENGTH_SHORT).show();
                                    }
                                }

                                @Override
                                public void onCancelled(@NonNull
DatabaseError error) {

                                }
                            });
                }
            }

            @Override
            public void afterTextChanged(Editable editable) {
```

```
            }
        });

        findViewById(R.id.signIN).setOnClickListener(n->{
            if (!ID.getText().toString().isEmpty() &&
!password.getText().toString().isEmpty() && memberId!=null){
                FirebaseAuth
                        .getInstance()

.signInWithEmailAndPassword(memberId,password.getText().toString())
                        .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull
Task<AuthResult> task) {
                                if (task.isSuccessful()){
                                    startActivity(new
Intent(Login.this,HomeActivity.class));
                                    finish();
                                }else {
                                    memberId = null;
                                    Toast.makeText(Login.this,
""+task.getException().getMessage(), Toast.LENGTH_SHORT).show();
                                    alertDialog.dismiss();
                                }
                            }
                        });

            }else Toast.makeText(this, "Fields required !",
Toast.LENGTH_SHORT).show();
        });
    }


    @Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == 1001 && data != null)
            ID.setText(data.getAction());
    }

    @Override
    protected void onStart() {
        super.onStart();
    }
}
package thundersharp.aigs.spekteriot;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.AppCompatButton;

import android.app.AlertDialog;
import android.os.Bundle;
```

```
import android.os.Handler;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

public class HomeActivity extends AppCompatActivity {

    private AppCompatButton button1,button2,button3,button4;
    private String device1,device2,device3,device4;
    private AlertDialog alertDialog;

    boolean init = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);
        alertDialog =
Progressbars.getInstance().createDefaultProgressBar(this);

        button1 = findViewById(R.id.device1);
        button2 = findViewById(R.id.device2);
        button3 = findViewById(R.id.device3);
        button4 = findViewById(R.id.device4);

        button1.setOnClickListener(b->{
            alertDialog.show();
            if (device1.equalsIgnoreCase("ON")){
                FirebaseDatabase
                        .getInstance()
                        .getReference("cmd")
                        .child("Device1")
                        .setValue("OFF")
                        .addOnCompleteListener(task -> {
                            if (task.isSuccessful()){
                                button1.setText("Device 1 OFF");
                                new Handler().postDelayed(new
Runnable() {
                                    @Override
                                    public void run() {
                                        alertDialog.dismiss();
                                    }
                                },7000);
                            }
                        });

            }else {
                alertDialog.show();

FirebaseDatabase.getInstance().getReference("cmd").child("Device1").
```

```
setValue("ON").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button1.setText("Device 1 ON");
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);
                        }
                    }
                });;

            }
        });

        button2.setOnClickListener(b->{
            alertDialog.show();
            if (device2.equalsIgnoreCase("ON")){

FirebaseDatabase.getInstance().getReference("cmd").child("Device2").
setValue("OFF").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button2.setText("Device 2 OFF");
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);
                        }
                    }
                });


            }else {

FirebaseDatabase.getInstance().getReference("cmd").child("Device2").
setValue("ON").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button2.setText("Device 2 ON");
```

```
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);
                        }
                    }
                });
            }
        });

        button3.setOnClickListener(b->{
            if (device3.equalsIgnoreCase("ON")){

FirebaseDatabase.getInstance().getReference("cmd").child("Device3").
setValue("OFF").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button2.setText("Device 3 OFF");
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);
                        }
                    }
                });

            }else {

FirebaseDatabase.getInstance().getReference("cmd").child("Device3").
setValue("ON").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button2.setText("Device 3 ON");
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);
                        }
                    }
                });
                ;
```

```
            }
        });

        button4.setOnClickListener(b->{
            if (device4.equalsIgnoreCase("ON")){

FirebaseDatabase.getInstance().getReference("cmd").child("Device4").
setValue("OFF").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button4.setText("Device 4 OFF");

                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);

                        }
                    }
                });

            }else {

FirebaseDatabase.getInstance().getReference("cmd").child("Device4").
setValue("ON").addOnCompleteListener(new OnCompleteListener<Void>()
{
                    @Override
                    public void onComplete(@NonNull Task<Void> task)
{
                        if (task.isSuccessful()){
                            button4.setText("Device 4 ON");
                            new Handler().postDelayed(new Runnable()
{
                                @Override
                                public void run() {
                                    alertDialog.dismiss();
                                }
                            },7000);

                        }
                    }
                });
                ;

            }
        });

        alertDialog.show();
        FirebaseDatabase
                .getInstance()
```

```
                .getReference("cmd")
                .addValueEventListener(new ValueEventListener() {
                    @Override
                    public void onDataChange(@NonNull DataSnapshot
snapshot) {
                        if (snapshot.exists()){
                            device1 =
snapshot.child("Device1").getValue(String.class);
                            device2 =
snapshot.child("Device2").getValue(String.class);
                            device3 =
snapshot.child("Device3").getValue(String.class);
                            device4 =
snapshot.child("Device4").getValue(String.class);
                            button1.setText("Device 1 "+device1);
                            button2.setText("Device 2 "+device2);
                            button3.setText("Device 3 "+device3);
                            button4.setText("Device 4 "+device4);


((TextView)findViewById(R.id.ts1)).setText("The Device is
"+device1);

((TextView)findViewById(R.id.ts2)).setText("The Device is
"+device2);

((TextView)findViewById(R.id.ts3)).setText("The Device is
"+device3);

((TextView)findViewById(R.id.ts4)).setText("The Device is
"+device4);

                            alertDialog.dismiss();
                        }
                    }

                    @Override
                    public void onCancelled(@NonNull DatabaseError
error) {

Toast.makeText(HomeActivity.this,error.getMessage(),Toast.LENGTH_LON
G).show();
                    }
                });
    }
}
package thundersharp.aigs.spekteriot;

import android.Manifest;
import android.app.AlertDialog;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraManager;
import android.os.Build;
import android.os.Bundle;
```

```
import android.os.Handler;
import android.os.Looper;
import android.util.Log;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.ImageView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;
import com.google.android.gms.vision.barcode.BarcodeDetector;
import com.google.android.material.bottomsheet.BottomSheetDialog;

import java.io.IOException;

public class BarCodeScanner extends AppCompatActivity {

    private SurfaceView surfaceView;
    private BarcodeDetector barcodeDetector;
    private CameraSource cameraSource;
    private boolean isFlashAvailable = false;
    private boolean flashStatus = false;
    private static final int REQUEST_CAMERA_PERMISSION = 201;
    private boolean dialog = true;

    private String scanValue = null;

    private CameraManager mCameraManager;
    private String mCameraId;

    private BottomSheetDialog bottomSheetDialog;
    private AlertDialog alertDialog;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bar_code_scanner);
        isFlashAvailable =
getApplicationContext().getPackageManager()

.hasSystemFeature(PackageManager.FEATURE_CAMERA_FRONT);
        alertDialog  =
Progressbars.getInstance().createDefaultProgressBar(BarCodeScanner.t
his);

        surfaceView = findViewById(R.id.surfaceView);

        bottomSheetDialog = new BottomSheetDialog(this);


        //initialiseDetectorsAndSources();
```

```
    }

    public void toggleFlash(boolean status) {
        try {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                mCameraManager.setTorchMode(mCameraId, status);
            }
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    private void initialiseDetectorsAndSources() {
        //Toast.makeText(getApplicationContext(), "Scanner started",
Toast.LENGTH_SHORT).show();
        barcodeDetector = new BarcodeDetector.Builder(this)
                .setBarcodeFormats(Barcode.ALL_FORMATS)
                .build();

        cameraSource = new CameraSource.Builder(this,
barcodeDetector)
                .setRequestedPreviewSize(300, 300)
                .setAutoFocusEnabled(true) //you should add this
feature
                .build();


        surfaceView.getHolder().addCallback(new
SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                try {
                    if
(ActivityCompat.checkSelfPermission(BarCodeScanner.this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED) {
                        cameraSource.start(surfaceView.getHolder());
                    } else {

ActivityCompat.requestPermissions(BarCodeScanner.this, new

String[]{Manifest.permission.CAMERA}, REQUEST_CAMERA_PERMISSION);
                    }

                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void surfaceChanged(SurfaceHolder holder, int
format, int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                cameraSource.stop();
```

```
            }
        });


        barcodeDetector.setProcessor(new
Detector.Processor<Barcode>() {
            @Override
            public void release() {
                //Toast.makeText(getApplicationContext(), "To
prevent memory leaks barcode scanner has been stopped",
Toast.LENGTH_SHORT).show();
            }

            @Override
            public void
receiveDetections(Detector.Detections<Barcode> detections) {
                final SparseArray<Barcode> barcodes =
detections.getDetectedItems();
                if (barcodes.size() != 0) {
                    new Handler(Looper.getMainLooper()).post(new
Runnable() {
                        @Override
                        public void run() {
                            if (dialog) {
                                dialog = false;
                                alertDialog.show();
                                scanValue =
barcodes.valueAt(0).displayValue;
                                if (scanValue.startsWith("SA")){
                                    //ID FOUND
                                    setResult(1001,new
Intent(scanValue));
                                    finish();

                                }else {
                                    new
AlertDialog.Builder(BarCodeScanner.this)
                                            .setMessage("Not a valid
Spekter ID!!")
                                            .setPositiveButton("OK",
((dialogInterface, i) -> {

alertDialog.dismiss();
                                                dialog = true;
                                            })).setCancelable(false)
                                            .show();
                                }

                            }
                        }
                    });

                    Log.d("TAG", barcodes.valueAt(0).rawValue);
                }
            }
        });
```

```
    }


    @Override
    protected void onPause() {
        super.onPause();
        //toggleFlash(false);
        flashStatus = false;
        cameraSource.release();
    }

    @Override
    protected void onResume() {
        super.onResume();
        initialiseDetectorsAndSources();
    }
}
```

CHAPTER -7

TESTING

Testing is the process of running a system with the intention of finding errors. Testing aims at detecting error -prone areas. This helps in the in the prevention of errors in a system. Testing also adds value to the product by confirming to the user requirements. The main purpose of testing is to detect error and error -prone areas in a system testing must be through and well planned. A partially tested system is as bad as an untested system and the price of an untested and under tested system is high. System testing involves unit testing, integration testing, white-box testing, black-box testing. Strategies for integration software components into a product include the bottom-up strategy, top-down strategy. Careful planning and scheduling are required to ensure that modules that will be available for integration into evolving software product when needed a serious of testing are performed for the proposed system before the system is ready for user acceptance testing.

Unit Testing

Instead of testing the system as a whole, Unit testing focuses on the modules that make up the system. Each module is taken up individually and tested for correctness in coding and logic**.**

The advantages of unit testing are:

- Size of the module is quite small and that errors can easily are located.
- Confusing interactions of multiple errors in wide different parts of the software is eliminated.
- Modules level testing can be exhaustive.

Integration Testing

It tests for the errors resulting from integration of modules. One specification of integration testing is whether parameters match on both sides of type, permissible ranges and meaning. Integration testing is functional black box test method. It includes testing each module as an impenetrable mechanism for information. The only concern during integration testing is that the modules work together properly.

White Box Testing (Code Testing)

The code-testing strategy examines the logic of the program. To follow this testing method, the analyst develops test cases that result in executing every instruction in the program or module so that every path through the program is tested. A path is a specific combination of

conditions that is handled by the program. Code testing does not check the range of data that the program will accept.

- Exercises all logical decisions on their true or false sides**.**
- Executes all loops at their boundaries and within these operational bounds.

Black Box Testing (Specification Testing)

To perform specification testing, the analyst examines the specification, starting from what the program should do and how it should perform under various conditions. Then test cases are developed for each condition or combinations of conditions and submitted for processing. By examining the results, the analyst can determine whether the programs perform according to its specified requirements. This testing strategy sounds exhaustive. If every statement in the program is checked for its validity, there doesn't seem to be much scope for errors.

Functional Testing

In this type of testing, the software is tested for the functional requirements. The tests are written in order to check if the application behaves as expected. Although functional testing is often done toward the end of the development cycle, it can and should, be started much earlier. Individual components and processes can be tested early on, even before it's possible to do functional testing on the entire system. Functional testing covers how well the system executes the functions it is supposed to execute including user commands, data manipulation, searches and business processes, user screens, and integrations. Functional testing covers the obvious surface type of functions, as well as the back-end operations (such as security and how upgrades affect the system).

Performance Testing

In software engineering, performance testing is testing that is performed, from one perspective, to determine how fast some aspect of a system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability, reliability and resource usage. Performance testing is a subset of Performance engineering, an emerging computer science practice which strives to build performance into the design and architecture of a system, prior to the onset of actual coding effort.

Performance testing can serve different purposes. It can demonstrate that the system meets performance criteria. It compares two systems to find which performs better. Or it can measure what parts of the system or workload cause the system to perform badly. In the diagnostic case, software engineers use tools such as profilers to measure what parts of a device or software contribute most to the poor performance or to establish throughput levels (and thresholds) for maintained acceptable response time. It is critical to the cost performance a new system; the performance test efforts begin at the inception of the development project

and extend through to deployment. The later a performance defect is detected, the higher the cost of remediation.

This is true in the case of functional testing, but even more so with performance testing, due to the end-to-end nature of its scope.

In performance testing, it is often crucial (and often difficult to arrange) for the test conditions to be similar to the expected actual use. This is, however, not entirely possible in actual practice. The reason is that production systems have a random nature of the workload and while the test workloads do their best to mimic what may happen in the production environment, it is impossible to exactly replicate this workload variability - except in the simplest system.

Stress Testing

The application is tested against heavy load such as complex numerical values, large number of inputs, large number of queries etc. which checks for the stress/load the applications can withstand. Stress testing deals with the quality of the application in the environment. The idea is to create an environment more demanding of the application than the application would experience under normal workloads. A test environment is established with many testing stations. At each station, a script is exercising the system. These scripts are usually based on the regression suite. More and more stations are added, all simultaneous hammering on the system, until the system breaks. The system is repaired and the stress test is repeated until a level of stress is reached that is higher than expected to be present at a customer site. Race conditions and memory leaks are often found under stress testing. A race condition is a conflict between at least two tests. Each test works correctly when done in isolation. When the two tests are run in parallel, one or both of the tests fail. This is usually due to an incorrectly managed lock. A memory leak happens when a test leaves allocated memory behind and does not correctly return the memory to the memory allocation scheme. The test seems to run correctly, but after being exercised several times, available memory is reduced until the system fails.

CHAPTER-8

CONCLUSION

This paper presents an innovative and dependable concept of a low-cost simple weather monitoring and controlling system. The system operates under IoT technology supervision which effectively optimizes remote areas. The creativity of this revolutionary weather station allows monitoring and controlling of the web server-based climate conditions using the ESP8266 node MCU microcontroller. The outputs of the measurements employed are meant to be shown via the NET PI web server as adjustable gauges. In terms of network connectivity, the devices can be turned ON or OFF at any moment and anywhere. The complete dependence on the webserver control system and the applicability of the local IP given by the ESP8266 means that the design's cost is inexpensive. The system contributes to being applicable in two fields.

CHAPER -9
REFERENCES

1. https://www.w3schools.in/dbms/intro/ Introduction to DBMS

2. https://beginnersbook.com/2015/04/dbms-introduction/ Introduction to DBMS

3. www.oracle.com/technetwork/database/enterprise-edition/documentation/database 093888.html

Oracle database documentation

4. www.oracle.com/technetwork/developer-tools/sql-developer/documentation/index.html

SQL Developer documentation

5. https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm

Procedures documentation

6. https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

JDBC documentation

CHAPTER -10

USER MANUAL

Arduino UNO

The Arduino UNO is a standard board of Arduino. Here UNO means 'one' in Italian. It is considered as the powerful board used in various projects. Arduino.cc developed the Arduino UNO board.

Arduino UNO is based on an ATmega328P microcontroller

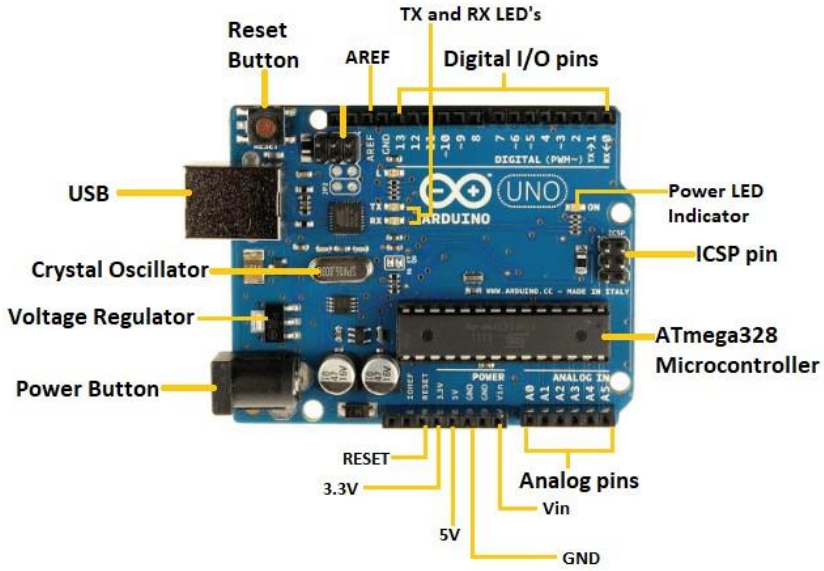
It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB

connector, a power jack, and an ICSP (In-Circuit Serial Programming) header. It is programmed based on IDE, which stands for Integrated Development Environment. It can run on both online and offline platforms. The IDE is common to all available boards of Arduino.

The Arduino board is shown below:

The components of Arduino UNO board are shown below:



Let's discuss each component in detail.

- **ATmega328 Microcontroller**- It is a single chip Microcontroller of the ATmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- **ICSP pin** - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.
- **Power LED Indicator**- The ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- **Digital I/O pins**- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- **TX and RX LED's-** The successful flow of data is represented by the lighting of these LED's. ○ **AREF-** The Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- **Reset button**- It is used to add a Reset button to the connection.
- **USB**- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- **Crystal Oscillator**- The Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.

- **Voltage Regulator**- The voltage regulator converts the input voltage to 5V. ○ **GND**- Ground pins. The ground pin acts as a pin with zero voltage. ○ **Vin**- It is the input voltage.
- **Analog Pins**- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

Technical Specifications of Arduino UNO

The technical specifications of the Arduino UNO are listed below:

- There are 20 Input/output pins present on the Arduino UNO board. These 20 pins include 6 PWM pins, 6 analog pins, and 8 digital I/O pins.
- The PWM pins are Pulse Width Modulation capable pins.
- The crystal oscillator present in Arduino UNO comes with a frequency of 16MHz. ○ It also has a Arduino integrated WiFi module. Such Arduino UNO board is based on the Integrated WiFi ESP8266 Module and ATmega328P microcontroller.
- The input voltage of the UNO board varies from 7V to 20V. ○ Arduino UNO automatically draws power from the external power supply. It can also draw power from the USB.

HOW TO DOWNLOAD ARDUINO IDE AND HOW TO INSTALL THE NODE MCU IN ARDUINO IDE

1. FOR ARDUINO INSTALLATION

https://youtu.be/Z7mrg4dd6ig

2. FOR INSTALLING NODE MCU

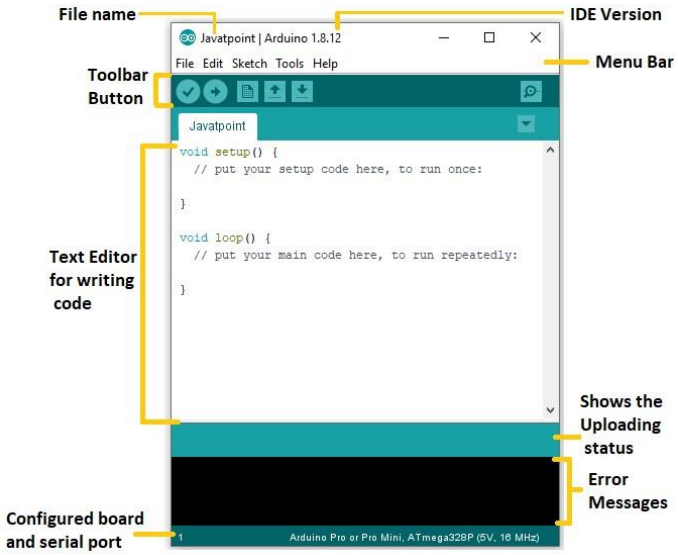
https://youtu.be/YN522_npNqs

Arduino IDE

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**.

The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Arduino IDE will appear as:



 IMPORTANT TERMS

Serial Monitor

The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor. It is shown below:

When we connect the serial monitor, the board will reset on the operating system Windows, Linux, and Mac OS X. If we want to process the control characters in our sketch, we need to use an external terminal program. The terminal program should be connected to the COM port, which will be assigned when we connect the board to the computer.

Sketchbook

It stores the current sketches created in the Arduino IDE software. It opens the selected sketch or code in a new editor at an instance.

Preferences

It allows the customization settings of the Arduino IDE.

Include Library

Include Library includes various Arduino libraries. The libraries are inserted into our code at the beginning of the code starting with the #. We can also import the libraries from .zip file.

- Tools

When we click on the Tools button on the Menu bar, a drop-down list appears. It is shown below:

Let's discuss each option in detail.

Auto Format

The Auto Format button is used to format the written code. For example, lining the open and closed curly brackets in the code.

Archive Sketch

The copy of the current sketch or code is archived in the .zip format. The directory of the archived is same as the sketch.

Fix Encoding and Reload

This button is used to fix the inconsistency between the operating system char maps and editor char map encoding.

Manage Libraries...

It shows the updated list of all the installed libraries. We can also use this option to install a new library into the Arduino IDE.

Serial Monitor

It allows the exchange of data with the connected board on the port.

Serial Plotter

The Serial Plotter button is used to display the serial data in a plot. It comes preinstalled in the Arduino IDE.

WiFi101/WiFiNINA Firmware Updater

It is used to check and update the Wi-Fi Firmware of the connected board.

Board

We are required to select the board from the list of boards. The selected board must be similar to the board connected to the computer.

Processor

It displays the processor according to the selected board. It refreshes every time during the selection of the board.

Port

It consists of the virtual and real serial devices present on our machine.

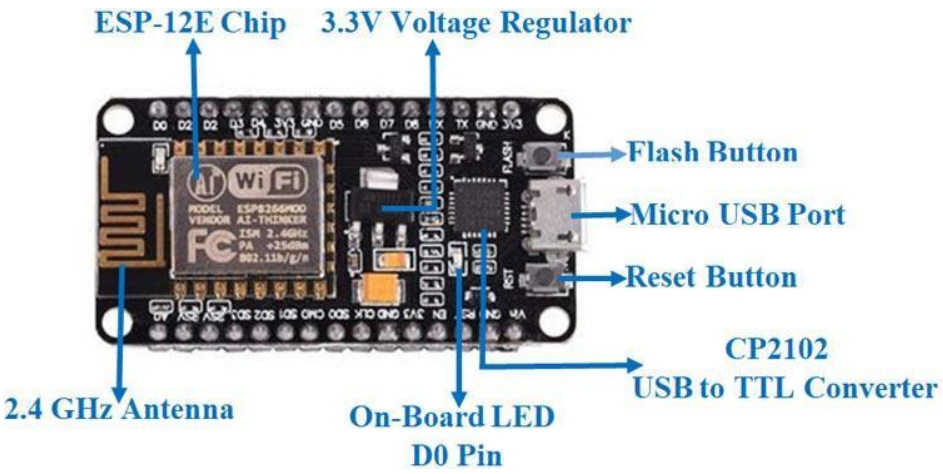
Get Board Info

It gives the information about the selected board. We need to select the appropriate port before getting information about the board.

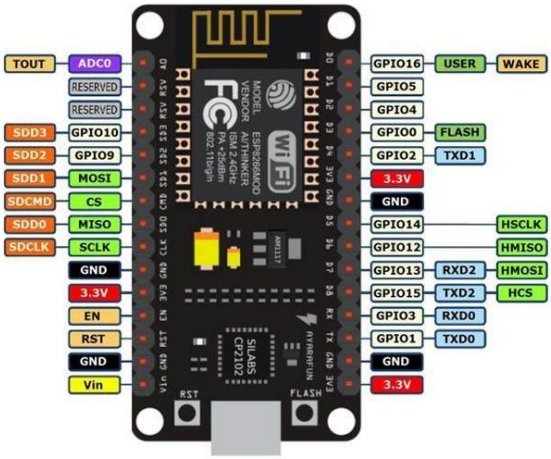
NODE MCU

The **NodeMCU ESP8266 development board** comes with the ESP-12E module containing the ESP8266 chip, This microprocessor supports RTOS and operates at 80MHz to 160 MHz adjustable clock frequency. NodeMCU has 128 KB RAM and 4MB of Flash memory to store data and programs. Its high processing power with in-built Wi-Fi / Bluetooth and Deep Sleep Operating features make it ideal for IoT projects.

NodeMCU can be powered using a Micro USB jack and VIN pin (External Supply Pin). It supports UART, SPI, and I2C interface.



NodeMCU is an open-source Lua based firmware and **development board** specially targeted for IoT based Applications. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.



NodeMCU Development Board Pinout Configuration

Pin Category	Name	Description
Power	Micro-USB, 3.3V, GND, Vin	**Micro-USB:** NodeMCU can be powered through the USB port **3.3V:** Regulated 3.3V can be supplied to this pin to power the board **GND:** Ground pins **Vin:** External Power Supply
Control Pins	**EN, RST**	The pin and the button resets the microcontroller
Analog Pin	A0	Used to measure analog voltage in the range of 0-3.3V
GPIO Pins	GPIO1 to GPIO16	NodeMCU has 16 general purpose input-output pins on its board
SPI Pins	SD1, CMD, SD0, CLK	NodeMCU has four pins available for SPI communication.
UART Pins	TXD0, RXD0, TXD2, RXD2	NodeMCU has two UART interfaces, UART0 (RXD0 & TXD0) and UART1 (RXD1 & TXD1). UART1 is used to upload the firmware/program.
I2C Pins		NodeMCU has I2C functionality support but due to the internal functionality of these pins, you have to find which pin is I2C.