

IOT BASED HOME AUTOMATION SYSTEM

**AN INTERNSHIP REPORT SUBMITTED
IN PARTIAL FULFILMENT FOR THE AWARD OF
THE DEGREE OF
BACHELOR OF TECHNOLOGY
IN
ELECTRONICS & INSTRUMENTATION ENGINEERING**

Submitted by

M BHARGAV	19071A1006
M SAI AADARSH	19071A1029
P SAI VIVEK	19071A1034
A RAHUL	19071A1038



DEPARTMENT OF ELECTRONICS & INSTRUMENTATION ENGINEERING
VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY
An Autonomous & ISO 9001:2015 Certified Institution, Accredited by NAAC with 'A++' Grade
Recognized as "Centre for Potential Excellence" by UGC
(Approved by AICTE, Affiliated to JNTUH)
Programme accredited by NBA
Vignana Jyothi Nagar, PragathiNagar, Nizampet (S.O), Hyderabad 500 090, TS, India.

2021-22

DEPARTMENT OF ELECTRONICS & INSTRUMENTATION ENGINEERING
VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING & TECHNOLOGY
An Autonomous & ISO 9001:2015 Certified Institution, Accredited by NAAC with 'A++' Grade
Recognized as "Centre for Potential Excellence" by UGC
(Approved by AICTE, Affiliated to JNTUH)
Programme accredited by NBA
Vignana Jyothi Nagar, PragathiNagar, Nizampet (S.O), Hyderabad 500 090, TS, India.



CERTIFICATE

This is to certify that the report of Internship titled "**IOT BASED HOME AUTOMATION SYSTEM**" is being submitted, by **M BHARGAV (19071A1006)**, **M AADARSH (19071A1029)**, **P SAI VIVEK (19071A1034)** and **A RAHUL (19071A1038)**, in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology in Electronics and Instrumentation Engineering**, to the Department of Electronics & Instrumentation Engineering at the **VNR VignanaJyothi Institute of Engineering and Technology** is a record of *bonafide* work carried out by them under my guidance and supervision. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree.

Mrs. R. Manjula Sri
Prof & Head of the Dept.
Dept. of EIE, VNRVJIET
Hyderabad

External Examiner

ACKNOWLEDGEMENTS

This is an acknowledgement of the intense drive and technical competence of many individuals who contributed to the success of our project.

We express our sincere thanks to **Mrs. R. Manjula Sri**, Professor & Head of the Department of Electronics & Instrumentation Engineering, VNRVJIET, and to other faculty members in the Department for guiding us through our education at the Institute and for encouraging us all through. We particularly thank our mentors, **Mr. M Sreedhar**, Professor, Department of EIE, for helping us through our journey at VNRVJIET. We also extend our gratitude to the internship Coordinators, **Ms. Jyothirmai Joshi**, Assistant Professor and **Ms. P Sampurna Lakshmi**, Assistant Professor with the Department of EIE, for their valuable guidance and for streamlining the review process for our project work. Our thanks are also due to the other members of the Review Panel and All other Faculty members.

We express our thanks to **Dr.C.D. Naidu**, Principal-VNRVJIET, for enabling us to use the Institute facilities and resources for the successful completion of our project work.

M BHARGAV

M AADARSH

P SAI VIVEK

A RAHUL

DECLARATION

We hereby declare that the work done during Internship titled “**IOT BASED HOME AUTOMATION SYSTEM**” submitted, towards partial fulfilment of requirements for the degree of Bachelor of Technology in Electronics and Instrumentation Engineering, to the Department of Electronics & Instrumentation Engineering at the VNR VignanaJyothi Institute of Engineering and Technology, Hyderabad, is an authentic work and had not been submitted to any other University or Institute for any award of degree or diploma.

M BHARGAV
(19071A1006)

M AADARSH
(19071A1029)

P SAI VIVEK
(19071A1034)

A RAHUL
(19071A1038)

ABSTRACT

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 can 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. In this project with the help of Smart home with google assistant and Alexa, we automated home appliances with NodeMCU ESP8266 and Sinric Pro software. We control home appliances via a switch board using google assistant, Alexa, and manual switches. We can control the relays from Google Home and Amazon Alexa app from anywhere in the world. We can control the relay model from the manual switches if there is no internet available. With this home automation project, we can control & monitor the **real-time feedback** of the relays in the **Google Home** and **Alexa App** from anywhere in the world. If the WiFi is available, the NodeMCU will automatically connect with the Wi-Fi.

LITERATURE SURVEY

“Smart Energy Efficient Home Automation System using IOT”, by Satyendra K. Vishwakarma, Prashant Upadhyaya, Babita Kumari, Arun Kumar Mishra.

This paper presents a step-by-step procedure of a smart home automation controller. It uses IOT to convert home appliances to smart and intelligent devices, with the help of design control. An energy efficient system is designed that accesses the smart home remotely using IOT connectivity. The proposed system mainly requires, Node MCU as the microcontroller unit, IFTTT to interpret voice commands, Adafruit a library that supports MQTT acts as an MQTT broker and Arduino IDE to code the microcontroller.

“Enhance Smart Home Automation System based on Internet of Things”, by Tushar Churasia and Prashant Kumar Jain.

This paper proposes a system that develops a model to reduce the computation overhead in existing smart home solutions that uses various encryption technologies like AES, ECHD, hybrid, etc. these solutions use intermediate gateway for connecting various sensor devices. The proposed model provides a method for automation with sensor-based learning. The system uses temperature sensor for development, but other sensors can also be used as per requirement.

“A Low Cost Home Automation System Using Wi-Fi based Wireless Sensor Network Incorporating internet of Things”, by Vikram.N, Harish.K.S, Nihaal.M.S, Raksha Umesh, Shetty Aashik Ashok Kumar.

This paper illustrates a methodology to provide a low-cost Home Automation System (HAS) using Wireless Fidelity (Wi-Fi). This crystallizes the concept of internetworking of smart devices. A Wi-Fi based Wireless Sensor Network (WSN) is designed for the purpose of monitoring and controlling environmental, safety and electrical parameters of a smart interconnected home. The application is developed using Android Studio based on JAVA platform and User Interface of those are exemplified. The primary focus of the paper is to develop a solution cost effective flexible in control of devices and implementing a wide range of sensors to capture various parameters.

TABLE OF CONTENTS

Details of Contents	Page #
Abstract	vi
Literature Survey	vii
Table of Contents	viii
List of Tables	ix
List of Figures	x
Chapter 1: Introduction	1
1.1 Objective	1
1.2 Outline	1
1.3 Motivation	1
1.4 Scope of the Project	2
Chapter 2: Project Overview	3
2.1 IoT	3
2.1.1 IoT	3
2.1.2 Features of IoT	3
2.1.3 Advantages of IoT	4
2.1.4 Disadvantages of IoT	5
2.1.5 Applications of IoT	6
2.2 Node MCU	7
2.2.1 Pin configuration of Node MCU development board	8
2.2.2 Parts of Node MCU Development Board	11
2.3 Block Diagram	14
2.3.1 Block Diagram of the Proposed System	14
2.3.2 Proposed System	15
Chapter 3: Methodology	16
3.1 Circuit Diagram	16

3.2	Components Required	17
3.3	Setting up the System	17
3.3.1	Creating a Sinric Pro account	17
3.3.2	Creating a room in Sinric Pro	18
3.3.3	Adding devices in Sinric Pro	18
3.3.4	Set up Push notifications to mobile	20
3.3.5	Set up Timers	21
3.3.6	Set up energy usage	21
Chapter 4:	Results and Discussion	23
4.1	Result	23
4.2	Discussions	23
Chapter 5:	Conclusions and Future Scope	24
5.1	Conclusions	24
5.2	Future Scope	24
References		25
Appendix A:	Simulation/ Programming	26
A.1	Program Code	26

LIST OF TABLES

Details of Contents	Page #
Table 2.1: Node MCU index ↔ GPIO mapping.	8
Table 3.1: Component Listing	17

LIST OF FIGURES

Details of Contents	Page #
Figure 2.1: Working of IoT enables care devices.	6
Figure 2.2: Node MCU Development Board	7
Figure 2.3: ESP8266 Node MCU pinout	10
Figure 2.4: ESP 12E module in Node MCU Development board	11
Figure 2.5: Power module on a Node MCU Development board	12
Figure 2.6: GPIO pins on Node MCU Development board	13
Figure 2.7: CP2021 on Node MCU development board	13
Figure 2.8: Block diagram of the proposed system.	14
Figure 3.1: Circuit of the Node MCU Home Automation	16
Figure 3.2: Creating a Sinric Pro Account	17
Figure 3.3: Creating room in Sinric Pro	18
Figure 3.4: Adding devices in Sinric Pro	18
Figure 3.5: Adding devices on the Sinric Pro	19
Figure 3.6: Writing the details of the device	19
Figure 3.7: Enabling push notification	20
Figure 3.8: Setting Timers for device	20
Figure 3.9: Setting estimate energy usage	21

CHAPTER 1

INTRODUCTION

1.1 Objective

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 can control various components via internet or be automatically programmed to operate from ambient conditions.

1.2 Introduction

In this project with the help of Smart home with google assistant and Alexa, we automated home appliances with NodeMCU ESP8266 and Sinric Pro software. We control home appliances via a switch board using google assistant, Alexa, and manual switches. We can control the relays from Google Home and Amazon Alexa app from anywhere in the world. It is a low cost and simple home automation technology, and the appliances can be controlled wirelessly with Wi-Fi and by a relay model and switches also.

1.3 Motivation

The concept of “Home Automation” has been in existence for several years. “Smart Home”, “Intelligent Home” are terms that followed and has been used to introduce the concept of networking appliance within the house. Home Automation Systems (HASs) includes centralized control and distance status monitoring of lighting, security system, and other appliances and systems within a house. 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. . This system is designed to assist and provide support in order to fulfil the needs of elderly and disabled in home. Also, the smart home concept in the system improves the standard living at home.

1.4 Scope for the Work

The aim is to design a prototype that establishes wireless remote control over a network of home appliances. The application is designed to run on android device providing features like, switch mode control and voice command control. Considering its wide range of application, following are the scope of this prototype.

- The system can be implemented in homes, small offices and malls as well, being in-charge of control of the electrical appliances.
- For remote access of appliances in internet or intranet. The appliances in the above-mentioned environment can be controlled in intra-network or can be accessed via internet.
- This project is limited to control only 3 devices at a time and this number can be increased by purchasing the Sinric Pro subscription.

CHAPTER 2

PROJECT OVERVIEW

2.1 IOT

This chapter has the Theory that has been acquired to commence the project work. This discussed about IOT, the advantages, disadvantages the network topologies and communication protocols. This chapter also briefs about the main microcontroller unit of the prototype, Node MCU. Its pin configuration, various functional units of the development board. The chapter further give a brief overview of the project, a block diagram of the system and the circuit diagram.

2.1.1 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.

2.1.2 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

2.1.3 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 can 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 can 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.

Time

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.

2.1.4 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.

2.1.5 Applications 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 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.

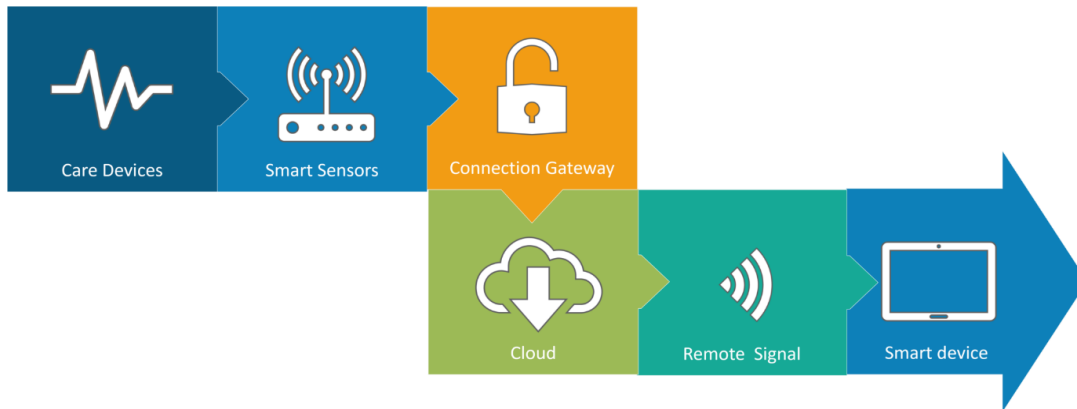


Fig 1 : Working of IoT enables care devices.

2.2 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.

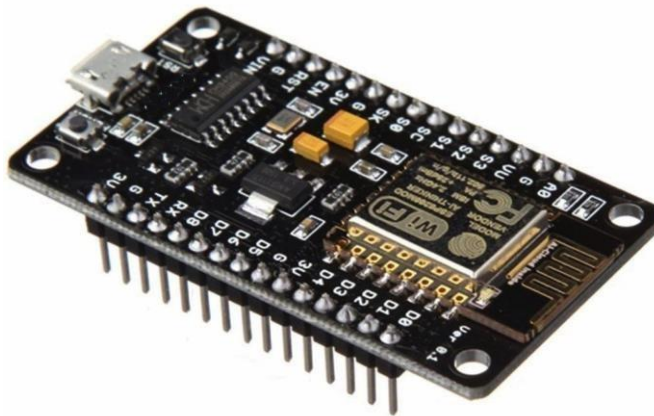


Fig 2: Node MCU Development Board

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 Espressif Non-OS SDK for ESP8266. It uses many open source projects, such as luacjson and SPIFFS. Due to resource constraints, users need to select the modules relevant for their project and build a firmware tailored to their needs. Support for the 32-bit ESP32 has also been implemented.

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

2.2.1 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 mapping table 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		

Table 1: Node MCU index ↔ GPIO mapping.

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

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

Power pins: There are four power pins viz. one VIN pin & three 3.3V pins. The VIN pin can be used to directly 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 as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button programmatically. Each digital enabled GPIO can be configured to internal pull-up or pull-down, or set to high impedance. When configured as 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 voltage of TOUT pin. However, they cannot be implemented at the same time.

UART Pins: ESP8266 Node MCU has 2 UART interfaces, i.e. UART0 and UART1, which provide 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 flow 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 also support the following general-purpose SPI features:

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

SDIO Pins: ESP8266 features Secure Digital Input/output Interface (SDIO) which is used to directly interface 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 output can be implemented programmatically and used for driving digital motors and LEDs. PWM frequency range is adjustable from 1000 μ s to 10000 μ s, i.e., between 100 Hz and 1 kHz.

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 LOW the 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.

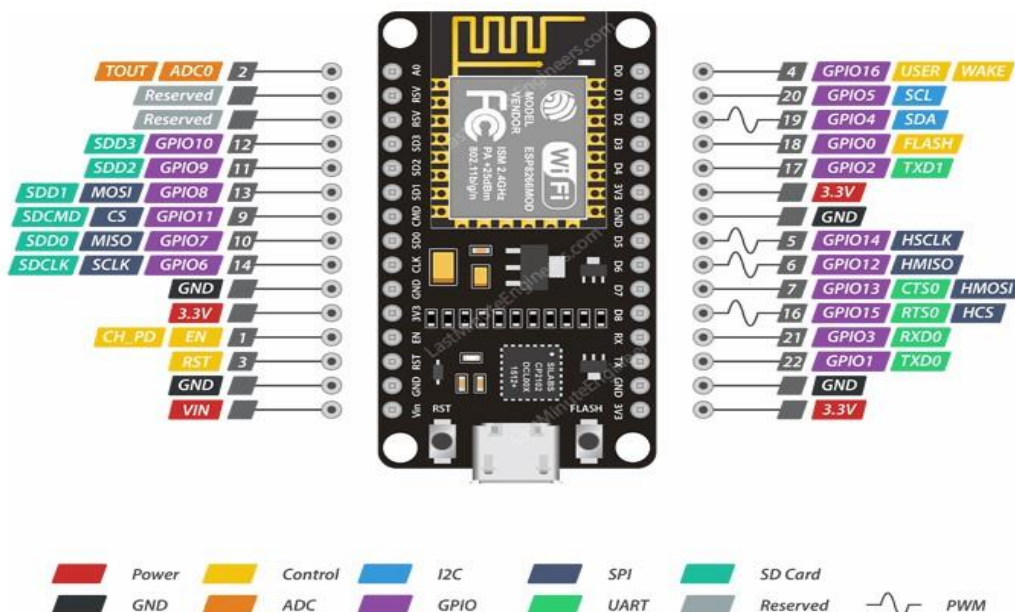


Figure 3: ESP8266 Node MCU pinout.

2.2.2 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 IOT devices 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.

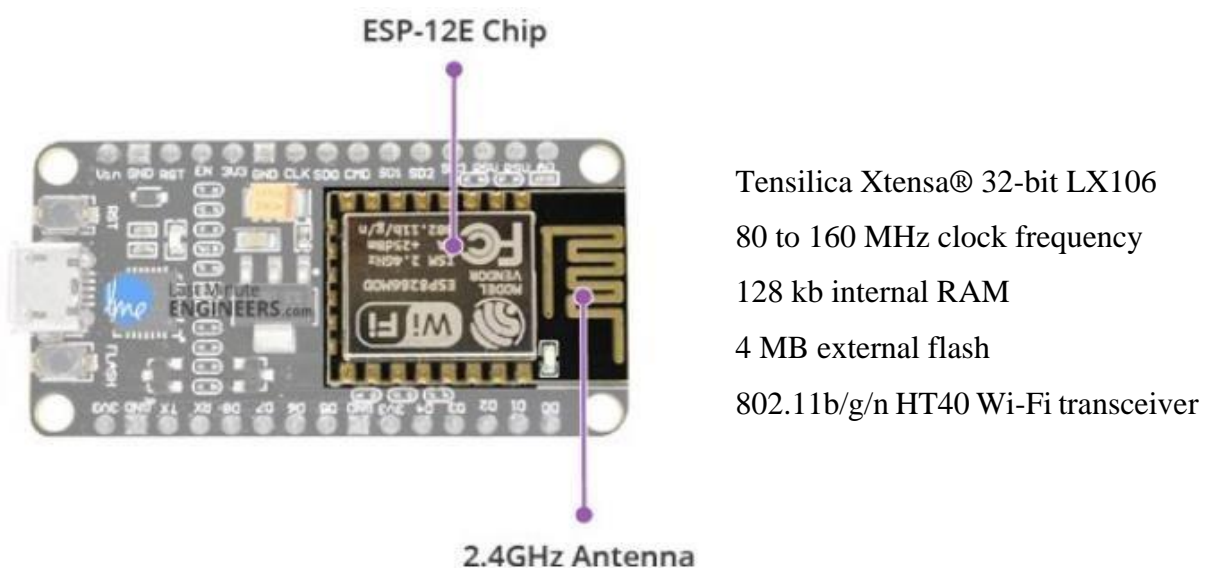


Fig 4 : ESP 12E module in Node MCU Development board

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 the regulator 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 and its peripherals.

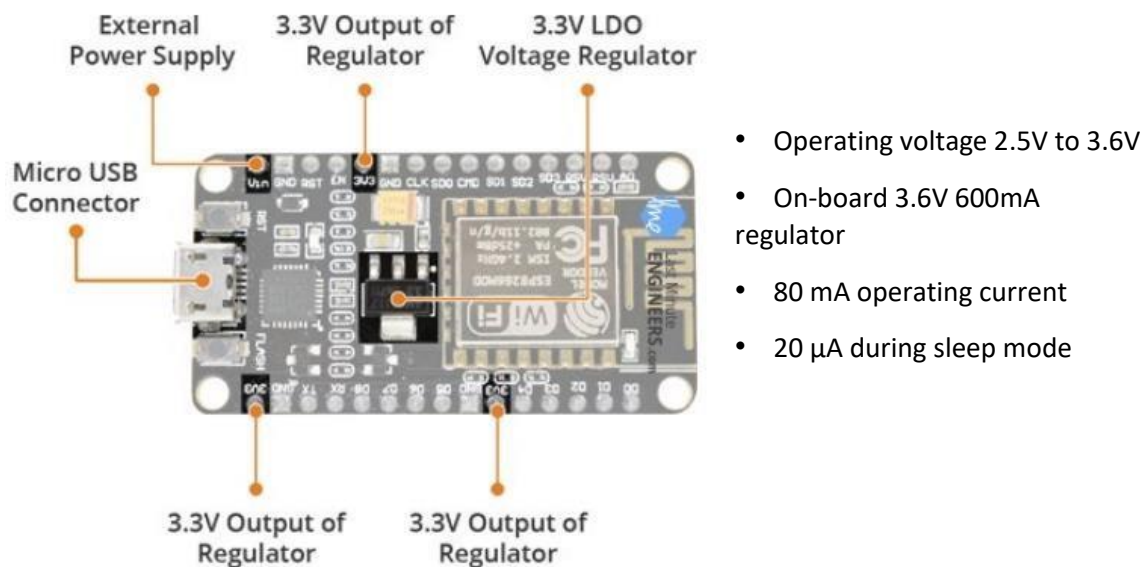


Fig 5: Power module on a Node MCU development board.

Peripherals I/O

The ESP8266 Node MCU has total 17 GPIO pins broken out to the pin headers on both sides of the development 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 and peripherals.
- 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), a single GPIO pin can act as PWM/UART/SPI.

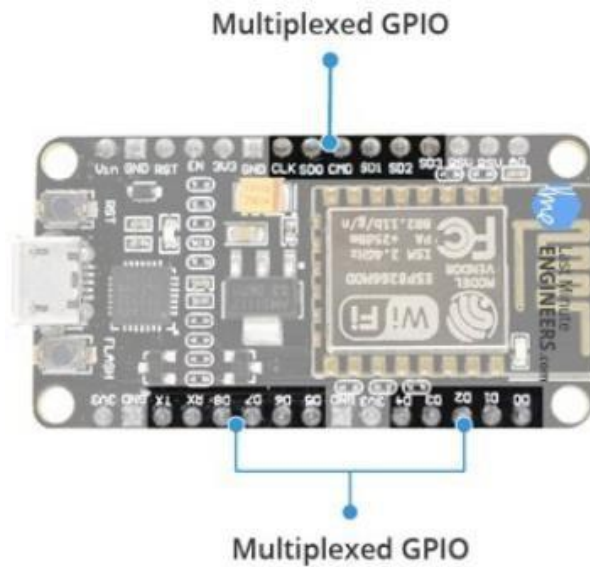
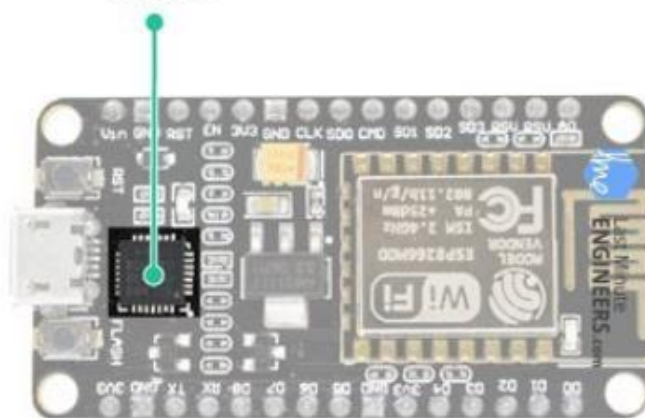


Figure 6: GPIO pins on Node MCU development board.

Serial Communication

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

USB To TTL Converter CP2102



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

Figure 7: CP2120 on Node MCU development board.

2.3 Block Diagram

2.3.1 Block Diagram of the Proposed System

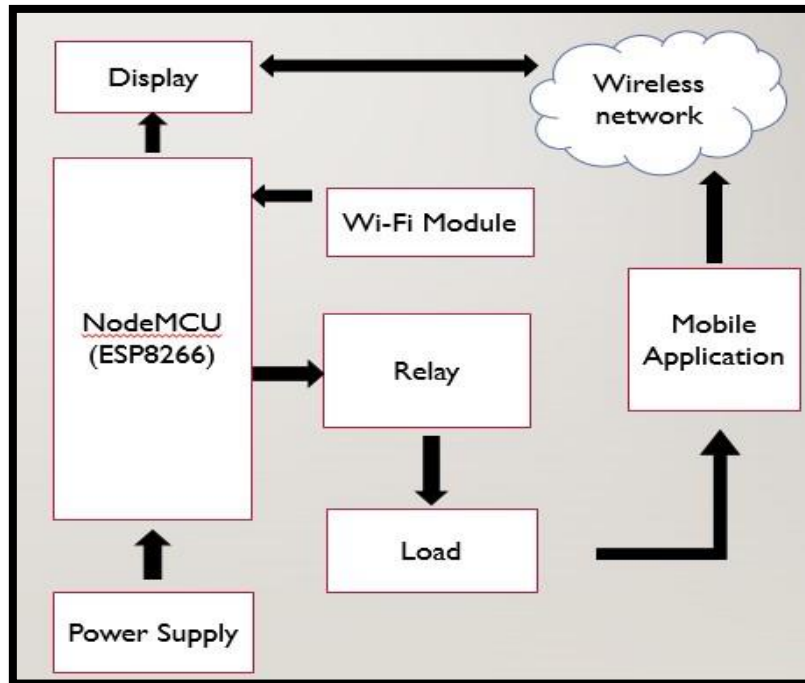


Figure 8: Block Diagram of the Proposed system

The block diagram gives the functionality of the overall project. The Node MCU unit is the microcontroller or the main controlling unit of the system. The user uses the mobile application in setting commands for functioning of the appliances. The mobile application interprets the command form in user in voice or switch mode and sends signal to the Node MCU unit, over a wireless network established by Wi-Fi communication. Hence the Wi-Fi module (actually inbuilt into Node MCU), helps the microcontroller establish Wi-Fi communication with a device and take commands from an application over wireless network. The Node MCU on further receiving the signal then turns on/off the appliance with the help of relay. The Node MCU, relay and the final appliances are physically connected. There is a power supply unit that powers the microcontroller, the relay as well as the final appliances. There is also a display unit that displays the status of the application.

2.3.2 Proposed System

The android OS provides the flexibility of using the open source. The inbuilt sensors can be accessed easily. The application used to control the system has the following features. Android Phone acts as a client and data are sent via sockets programming. The application takes command from user in two different modes.

- **Switch mode:** Switch mode uses the radio buttons that are used to control the home appliances. The radio button sends the status of the switch.
- **Voice mode:** Voice Mode is used to control the home appliances using voice command. Using the inbuilt microphone of Smartphone, the application creates an intent that fetches the speech data to the Google server which responds with a string data. The string data are further analysed and then processed.

CHAPTER 3

METHODOLOGY

3.1 Circuit Diagram

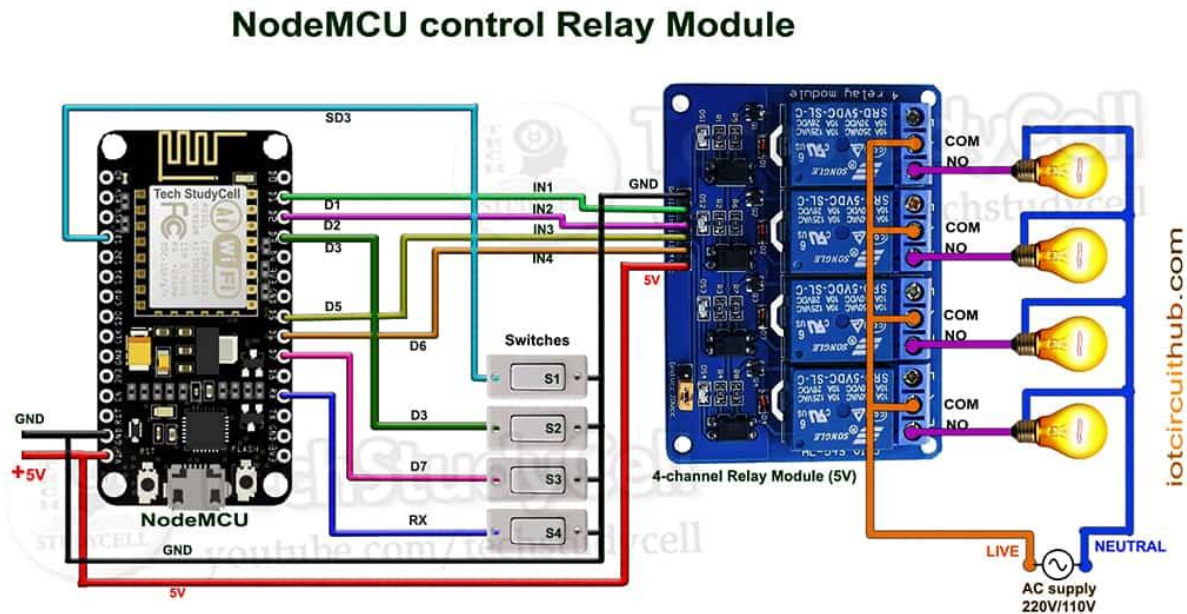


Figure 9: Circuit of the Node MCU Home Automation

The circuit is very simple, we have used **D1, D2, D5 & D6** GPIO to control the 4-channel relay module. And the GPIO **SD3, D3, D7 & RX** are connected with manual switches to control the relay module manually.

We have used the **INPUT_PULLUP** function in Arduino IDE instead of using the pull-up resistors with each switch. As per the source code, when the control pins of the relay module receive the **LOW** signal the respective relay will **turn on** and the relay will **turn off** for the **HIGH** signal in the control pin and have used a 5V 2Amp mobile charger to supply the circuit.

3.2 Components Required

NO	Component SL.	Quantity
1.	Node MCU	1
2.	4 channel 5V SPDT Relay Module	1
3.	Manual Switches or Push button	3
4.	LED	4
5.	2.2K Ω Resistor	4
6.	Male pin header	1
7.	Female pin header	1
8.	Jumper wires	8
9.	USB Cable	1

Table 2: Component Listing

3.3 Setting up the system

3.3.1 Creating a Sinric Pro account

First visit sinric.pro/register

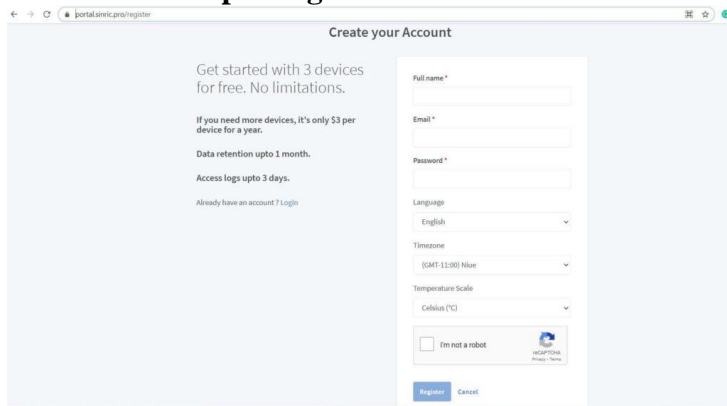


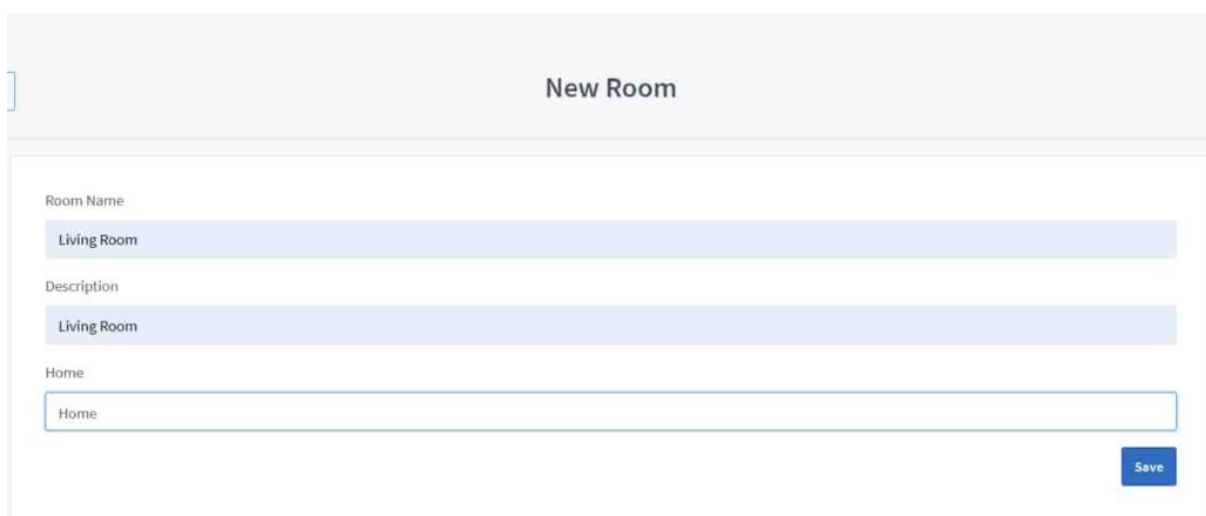
Fig 10: Creating Sinric Pro account.

3.3.2 Create a room in Sinric Pro

Before adding the devices, first we have to create room in the Sinric Pro.

Steps for creating rooms in Sinric Pro:

1. Go to **Rooms** in the left side menu.
2. Click on **Add Room** button.
3. Enter the **Room Name** and **Description**.
4. Click on **Save**.



New Room

Room Name
Living Room

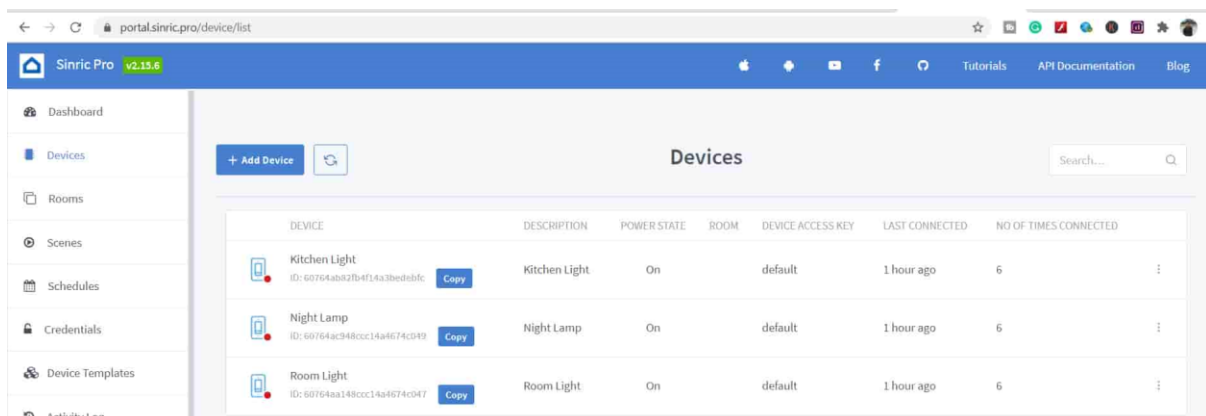
Description
Living Room

Home
Home

Save

Figure 11: Creating room in Sinric Pro

3.3.3 Add devices in Sinric Pro



portal.sinric.pro/device/list

Sinric Pro v2.15.6

Dashboard
Devices
Rooms
Scenes
Schedules
Credentials
Device Templates
Activity Log

+ Add Device

Devices

Search...




DEVICE	DESCRIPTION	POWER STATE	ROOM	DEVICE ACCESS KEY	LAST CONNECTED	NO OF TIMES CONNECTED
 Kitchen Light ID: 60764aba2fb4f14a3bedebfc	Kitchen Light	On	default	default	1 hour ago	6
 Night Lamp ID: 60764ac348ccc14a4674c949	Night Lamp	On	default	default	1 hour ago	6
 Room Light ID: 60764aa148ccc14a4674c947	Room Light	On	default	default	1 hour ago	6

Figure 11: Adding devices in sinric pro

Go to devices from the left side menu

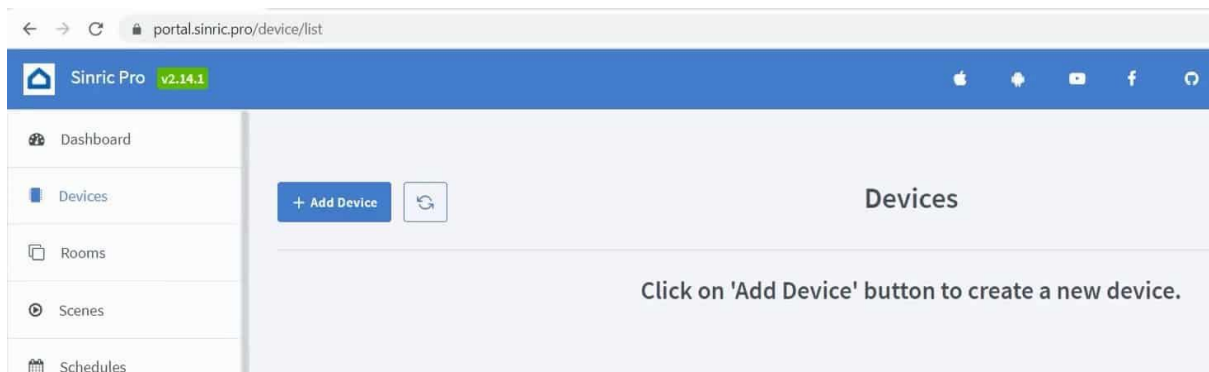


Figure 12: Adding devices on the sinric pro

Enter the device details

The 'New Device' form is displayed with four steps: 1. Device Information, 2. Notifications, 3. Timers, and 4. Other. The first step is active. The form contains the following fields:

- Device Name: Room Light
- Description: Room Light
- Device Type: Switch
- Device Access Key: default
- Room: Living Room

A note at the bottom states: '* We recommend using different Device Access Key for different hardware modules'. A 'Next' button is located at the bottom left.

Figure 13: Writing the details of the device

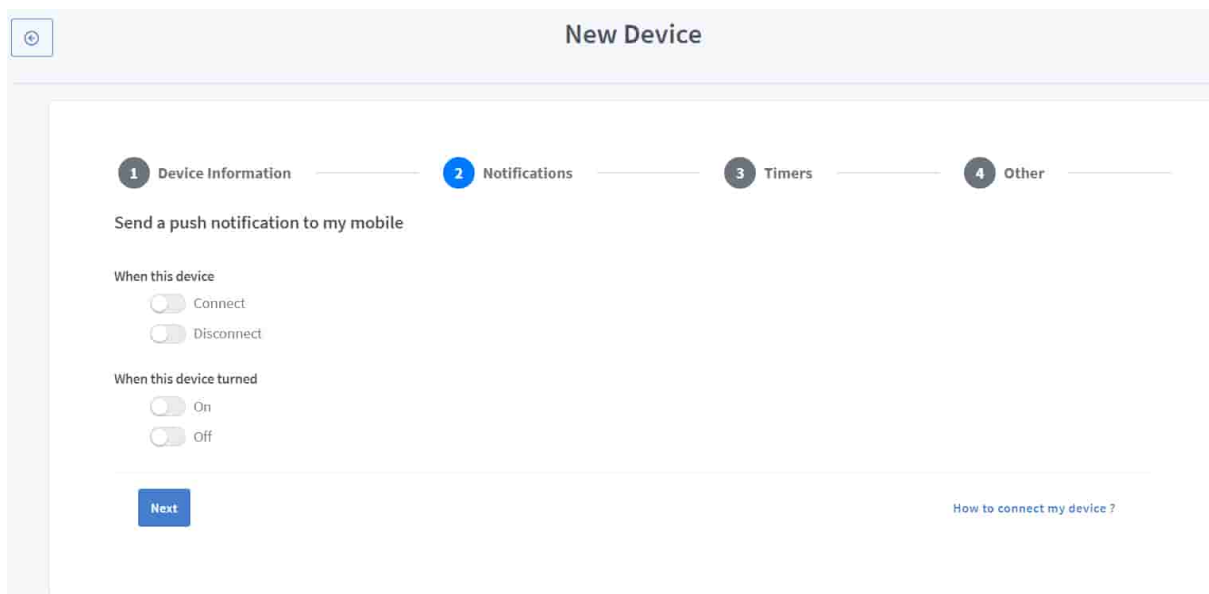
Enter the Device Name and Description.

Then select the Device Type as per the requirement. Here we have selected the device type as switch as we will control the SPDT relay.

Then select the room for the device.

After that, click on next.

3.3.4 Set up Push notifications to the mobile

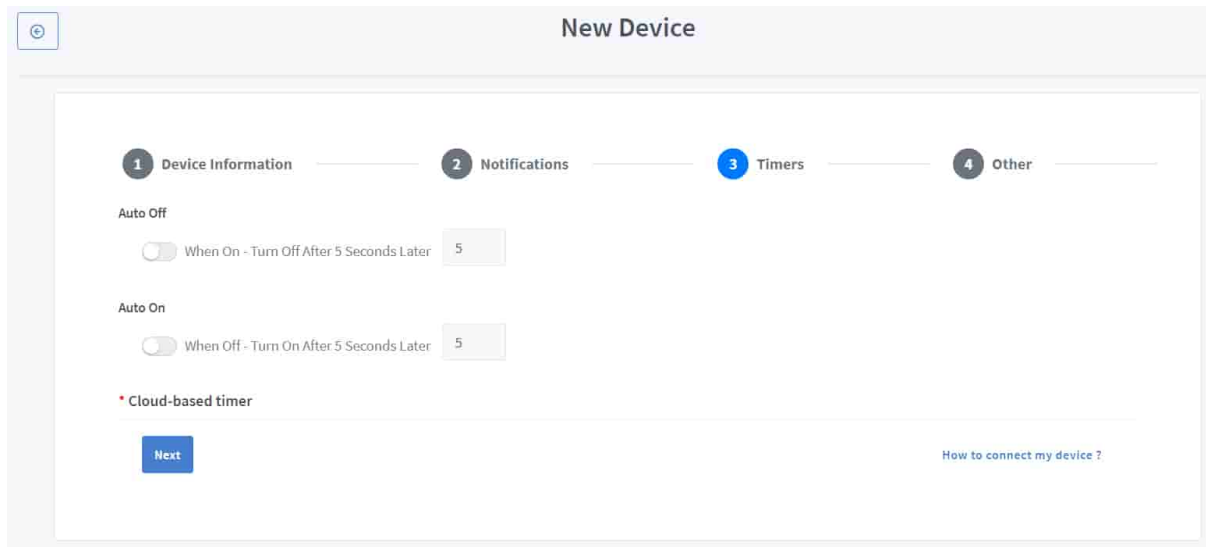


The screenshot shows a web interface for adding a new device. At the top, there's a header bar with a back arrow icon and the title "New Device". Below the header, a progress bar indicates four steps: 1. Device Information, 2. Notifications (currently active), 3. Timers, and 4. Other. The main content area is titled "Send a push notification to my mobile". It contains two sections of toggle switches. The first section, "When this device", has two toggles: "Connect" and "Disconnect", both of which are currently turned off. The second section, "When this device turned", has two toggles: "On" and "Off", both of which are also currently turned off. At the bottom left of the form is a blue "Next" button. At the bottom right is a link that says "How to connect my device?".

Figure 14: Enabling push notifications

If we want any push notifications, we can enable it here(this feature is optional).
Then click on Next.

3.3.5 Set up Timers



The screenshot shows the 'New Device' setup interface. At the top, there's a header 'New Device' and a back button. Below it, a progress bar indicates four steps: 1 Device Information, 2 Notifications, 3 Timers (highlighted), and 4 Other. The 'Timers' section contains two options: 'Auto Off' and 'Auto On'. Each option has a toggle switch and a text input field. The 'Auto Off' toggle is currently off, and its input field contains the number '5'. The 'Auto On' toggle is also off, and its input field contains the number '5'. Below these, there's a section for 'Cloud-based timer' marked with a red asterisk. At the bottom left, there is a blue 'Next' button, and at the bottom right, there is a link 'How to connect my device?'.

Figure 15 : Setting Timers for device

If you want to set any timer to **Auto On** or **Auto Off** the device, after the predefined time, then you the setup here. Again this field is optional.

Click on Next.

3.3.6 Set up Energy usage

The screenshot shows the 'New Device' setup interface. At the top, there's a header 'New Device' and a progress bar with four steps: 1 Device Information, 2 Notifications, 3 Timers, and 4 Other (which is currently selected). Below the progress bar, the 'Energy Usage (Estimation)' section is visible, featuring a toggle switch that is currently turned on. This section contains two input fields: 'Stand by Wattage (W)' with the description 'The energy the device consumes while turned off, or in standby mode.', and 'On Wattage (W)' with the description 'The typical energy the device consumes while turned on'. Below these fields, there are two footnotes: '* Alexa en-US language only. It takes few hours to reflect the usage details in the Alexa app.' and '* Coming soon to Sinric Pro app'. At the bottom left of the form is a blue 'Save' button, and at the bottom right is a link that says 'How to connect my device?'.

Figure 16: Setting estimate energy usage

This is another optional field. You can define the wattage rating of the connected appliance to get the energy consumption estimation.

Now Click on **Save**. the device will be created.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Results

The experimental model was made according to the circuit diagram and the results were as expected. The home appliances could be remotely switched over Wi-Fi network. Both the switch mode and the voice mode control methodologies were successfully achieved. The Sinric Pro software was also successful in displaying the status of every application.

4.2 Discussions

Android devices having lower API version than 16 requires internet access to convert the speech data to string data. Currently, the application is made for Android Smart Phones; other OS platform doesn't support our application. During voice mode, external noises (voice) may affect our result. The speech instruction that we command in our voice mode may not give exact result as expected. There hence lies an ambiguity in result.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

It is evident from this project work that an individual control home automation system can be cheaply made from low-cost locally available components and can be used to control multifarious home appliances ranging from the security lamps, the television to the air conditioning system and even the entire house lighting system. And better still, the components required are so small and few that they can be packaged into a small inconspicuous container. The designed home automation system was tested a number of times and certified to control different home appliances used in the lighting system, air conditioning system, home entertainment system and many more . Hence, this system is scalable and flexible.

5.2 Future Scope

Looking at the current situation we can build cross platform system that can be deployed on various platforms like iOS, Windows. Limitation to control only several devices can be removed by extending automation of all other home appliances. The prototype can include sensors to implement automatic control of the home appliances like; an LDR that can sense daylight and switch lamp accordingly, a PIR to detect motion and be used for security purposes making an alarm buzz, or a DHT11 sensor that's senses ambient temperature and humidity of atmosphere and switch fan/air conditioner accordingly. Scope of this project can be expanded to many areas by not restricting to only home, but to small offices.

References:

1. M. Stella Mercy, **“Performance Analysis of Epileptic Seizure Detection Using DWT & ICA with Neural Networks”**, International Journal Of Computational Engineering Research (ijceronline.com) Vol. 2 Issue. 4,pg 1109-1113, August’2012.
2. *“Smart Energy Efficient Home Automation System using IOT”*, by Satyendra K. Vishwakarma, Prashant Upadhyaya, Babita Kumari, Arun Kumar Mishra.
3. *“Enhance Smart Home Automation System based on Internet of Things”*, by Tushar Churasia and Prashant Kumar Jain; in Proceedings of the Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC 2019) IEEE Xplore Part Number:CFP19OSVART; ISBN:978-1-7281-4365-1 .
4. *“A Low Cost Home Automation System Using Wi-Fi based Wireless Sensor Network Incorporating internet of Things”*, by Vikram.N, Harish.K.S, Nihaal.M.S, Raksha Umesh, Shetty Aashik Ashok Kumar; in 2017 IEEE 7th International Advance Computing Conference.

APPENDIX A

SIMULATION AND PROGRAMMING

Code for Node MCU Home automation system

A.1 : Program code

```
// Uncomment the following line to enable serial debug output
// #define ENABLE_DEBUG

#ifdef ENABLE_DEBUG
    #define DEBUG_ESP_PORT Serial
    #define NODEBUG_WEBSOCKETS
    #define NDEBUG
#endif

#include <Arduino.h>
#include <ESP8266WiFi.h>
#include "SinricPro.h"
#include "SinricProSwitch.h"

#include <map>

#define WIFI_SSID    "YOUR-WIFI-NAME"
#define WIFI_PASS    "YOUR-WIFI-PASSWORD"
#define APP_KEY      "YOUR-APP-KEY"    // Should look like "de0bxxxx-1x3x-4x3x-ax2x-5dabxxxxxxxx"
#define APP_SECRET   "YOUR-APP-SECRET" // Should look like "5f36xxxx-x3x7-4x3x-xexe-e86724a9xxxx-4c4axxxx-3x3x-x5xe-x9x3-333d65xxxxx"

// Enter the device IDs here
#define device_ID_1  "SWITCH_ID_NO_1_HERE"
#define device_ID_2  "SWITCH_ID_NO_2_HERE"
#define device_ID_3  "SWITCH_ID_NO_3_HERE"
#define device_ID_4  "SWITCH_ID_NO_4_HERE"

// define the GPIO connected with Relays and switches
#define RelayPin1 5 //D1
#define RelayPin2 4 //D2
#define RelayPin3 14 //D5
#define RelayPin4 12 //D6

#define SwitchPin1 10 //SD3
#define SwitchPin2 0 //D3
#define SwitchPin3 13 //D7
#define SwitchPin4 3 //RX

#define wifiLed 16 //D0

// comment the following line if you use a toggle switches instead of tactile buttons
// #define TACTILE_BUTTON 1
```

```

#define BAUD_RATE 9600

#define DEBOUNCE_TIME 250

typedef struct { // struct for the std::map below
    int relayPIN;
    int flipSwitchPIN;
} deviceConfig_t;

// this is the main configuration
// please put in your deviceId, the PIN for Relay and PIN for flipSwitch
// this can be up to N devices...depending on how much pin's available on your device ;)
// right now we have 4 deviceIds going to 4 relays and 4 flip switches to switch the relay manually
std::map<String, deviceConfig_t> devices = {
    //{deviceId, {relayPIN, flipSwitchPIN}}
    {device_ID_1, { RelayPin1, SwitchPin1 }},
    {device_ID_2, { RelayPin2, SwitchPin2 }},
    {device_ID_3, { RelayPin3, SwitchPin3 }},
    {device_ID_4, { RelayPin4, SwitchPin4 }}
};

typedef struct { // struct for the std::map below
    String deviceId;
    bool lastFlipSwitchState;
    unsigned long lastFlipSwitchChange;
} flipSwitchConfig_t;

std::map<int, flipSwitchConfig_t> flipSwitches; // this map is used to map flipSwitch PINs to deviceId
and handling debounce and last flipSwitch state checks
// it will be setup in "setupFlipSwitches" function, using informations from
devices map

void setupRelays() {
    for (auto &device : devices) { // for each device (relay, flipSwitch combination)
        int relayPIN = device.second.relayPIN; // get the relay pin
        pinMode(relayPIN, OUTPUT); // set relay pin to OUTPUT
        digitalWrite(relayPIN, HIGH);
    }
}

void setupFlipSwitches() {
    for (auto &device : devices) { // for each device (relay / flipSwitch combination)
        flipSwitchConfig_t flipSwitchConfig; // create a new flipSwitch configuration

        flipSwitchConfig.deviceId = device.first; // set the deviceId
        flipSwitchConfig.lastFlipSwitchChange = 0; // set debounce time
        flipSwitchConfig.lastFlipSwitchState = true; // set lastFlipSwitchState to false (LOW)--

        int flipSwitchPIN = device.second.flipSwitchPIN; // get the flipSwitchPIN

        flipSwitches[flipSwitchPIN] = flipSwitchConfig; // save the flipSwitch config to flipSwitches map
        pinMode(flipSwitchPIN, INPUT_PULLUP); // set the flipSwitch pin to INPUT
    }
}

```

```

}

bool onPowerState(String deviceId, bool &state)
{
    Serial.printf("%s: %s\r\n", deviceId.c_str(), state ? "on" : "off");
    int relayPIN = devices[deviceId].relayPIN; // get the relay pin for corresponding device
    digitalWrite(relayPIN, !state);           // set the new relay state
    return true;
}

void handleFlipSwitches() {
    unsigned long actualMillis = millis(); // get actual millis
    for (auto &flipSwitch : flipSwitches) { // for each flipSwitch in flipSwitches map
        unsigned long lastFlipSwitchChange = flipSwitch.second.lastFlipSwitchChange; // get the timestamp
        // when flipSwitch was pressed last time (used to debounce / limit events)

        if (actualMillis - lastFlipSwitchChange > DEBOUNCE_TIME) { // if time is > debounce time...

            int flipSwitchPIN = flipSwitch.first; // get the flipSwitch pin from configuration
            bool lastFlipSwitchState = flipSwitch.second.lastFlipSwitchState; // get the lastFlipSwitchState
            bool flipSwitchState = digitalRead(flipSwitchPIN); // read the current flipSwitch state
            if (flipSwitchState != lastFlipSwitchState) { // if the flipSwitchState has changed...
#ifdef TACTILE_BUTTON
                if (flipSwitchState) { // if the tactile button is pressed
#endif
                    flipSwitch.second.lastFlipSwitchChange = actualMillis; // update lastFlipSwitchChange
time
                    String deviceId = flipSwitch.second.deviceId; // get the deviceId from config
                    int relayPIN = devices[deviceId].relayPIN; // get the relayPIN from config
                    bool newRelayState = !digitalRead(relayPIN); // set the new relay State
                    digitalWrite(relayPIN, newRelayState); // set the trelay to the new state

                    SinricProSwitch &mySwitch = SinricPro[deviceId]; // get Switch device from
SinricPro
                    mySwitch.sendPowerStateEvent(!newRelayState); // send the event
#ifdef TACTILE_BUTTON
                }
#endif
                flipSwitch.second.lastFlipSwitchState = flipSwitchState; // update lastFlipSwitchState
            }
        }
    }
}

void setupWiFi()
{
    Serial.printf("\r\n[Wifi]: Connecting");
    WiFi.begin(WIFI_SSID, WIFI_PASS);

    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.printf(".");
        delay(250);
    }
    digitalWrite(wifiLed, LOW);
}

```

```

    Serial.printf("connected!\r\n[WiFi]: IP-Address is %s\r\n", WiFi.localIP().toString().c_str());
}

void setupSinricPro()
{
    for (auto &device : devices)
    {
        const char *deviceId = device.first.c_str();
        SinricProSwitch &mySwitch = SinricPro[deviceId];
        mySwitch.onPowerState(onPowerState);
    }

    SinricPro.begin(APP_KEY, APP_SECRET);
    SinricPro.restoreDeviceStates(true);
}

void setup()
{
    Serial.begin(BAUD_RATE);

    pinMode(wifiLed, OUTPUT);
    digitalWrite(wifiLed, HIGH);

    setupRelays();
    setupFlipSwitches();
    setupWiFi();
    setupSinricPro();
}

void loop()
{
    SinricPro.handle();
    handleFlipSwitches();
}

```