CONNECTED HOME PROJECT

A COMPACT BASED IOT-DRIVEN AUTOMATION SYSTEM

A MINI PROJECT REPORT

Submitted by

RITHISHAASHRI.T.S [211422104401]

REBECCA.M [211422104389]

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

OCTOBER 2024

BONAFIDE CERTIFICATE

Certified that this project report **"CONNECTED HOME PROJECT A COMPACT BASED IOT-DRIVEN AUTOMATION SYSTEM"** is the bonafide work of **RITHISHAASHRI.T.S [211422104401], REBECCA.M [211422104389]** who carried out the project work under my supervision.

SIGNATURE

Dr.L.JEBASHEELA Ph.D.,
HEAD OF THE DEPARTMENT
ASSOCIATE PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARETHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

SIGNATURE

Mrs.S.SOPHANA JENNIFER M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,
NASARETHPETTAI,
POONAMALLEE,
CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI Thiru.C.SAKTHIKUMAR,M.E.,** and **Tmt.SARANYASREE SAKTHIKUMAR, B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr.L.JEBASHEELA Ph.D.,** for thesupport extended throughout the project.

We would like to thank all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

We would like to thank our guide, **Mrs.S.SOPHANA JENNIFER M.E.,**for her support andGuidance throughout our project.

RITHISHAASHRI.T.S [211422104401]

REBECCA.M [211422104389]

ABSTRACT

The main objective of this project is to develop a compact IoT device for smart home system using an NodeMCU board with Wi-Fi being remotely controlled by any Android OS smartphone. As technology is advancing, houses are also getting smarter. Modern houses are gradually shifting from conventional switches to a centralized control system, involving remote-controlled switches. Presently, conventional wall switches located in different parts of the house make it difficult for the user to go near them to operate. Even more, it becomes more difficult for the elderly or physically challenged people to do so. A remote- controlled smart home system provides the most modern solution with smartphones.

To achieve this, a Relay module is interfaced to the NodeMCU board at the receiver end while on the transmitter end, a GUI application on the cell phone sends ON/OFF commands to the receiver where loads are connected. The loads can be turned ON/OFF remotely through this technology by touching the specified location on the GUI. The loads are operated by NodeMCU board.

TABLE OF CONTENTS

ACKNOWLEDGEMENT

ABSTRACT

CHAPTER 1 - INTRODUCTION 1

CHAPTER 2 - LITERATURE SURVEY 2

CHAPTER 3 - THE PROPOSED SYSTEM 3

3.1 BLOCK DIAGRAM 3

3.2 CIRCUIT DIAGRAM 4

3.3 COMPONENTS REQUIRED 6

3.4 ARDUINO SOFTWARE PART - ARDUINO IDE 20

3.5 WEBPAGE 22

3.6 WORKING 23

CHAPTER 4 - SOFTWARE DEVELOPMENT 25

4.1 SOFTWARE INITIALIZATION 25

4.2 ARDUINO CODE 27

CHAPTER 5 - RESULT AND DISCUSSION 36

CHAPTER 6 - CONCLUSION 39

CHAPTER 7 - REFERENCE 41

LIST OF FIGURES

3.1 Block Diagram of Home Automation System 3

3.2 Circuit Diagram of Home Automation System 4

3.3 NodeMCU 7

3.4 ESP 8366 10

3.5 Channel Relay Module 17

3.6 Circuit Diagram for Relay Module-1 18

3.7 Circuit Diagram for Relay Module-2 19

3.8 Screenshots of our webpage 22

4.1.1-Arduino table 26

4.1.2-Arduino table 26

5.1 Real-Time Implementation 36

5.2 Relay 1 in ON State 37

5.3 Relay 2 in ON State 37

5.4 Both Relay 1 and Relay 2 in ON State 38

LIST OF TABLES

3.1 Pin configuration of ESP8266..... 13

3.2 Command and response 23

CHAPTER 1

INTRODUTION

Nowadays, Remote controls are used for television sets and other electronic systems, which have made our lives easy. The currently available smart home systems which would give the facility to control tube lights, fans and other electrical appliances at home using a remote control are not cost-effective. In this project, we have come up with a new system where we use NodeMCU to build a smart home system using Wi-Fi. This system is cost effective and can give the user, the ability to control any electronic device without even spending on the remote control. This project helps the user to control all the electronic devices using his/her smartphone. Time is a very valuable thing. Everybody wants to save time as much as they can. New technologies are being introduced to save time. To save people's time a Smart Home system using Wi-Fi is introduced. With the help of this system, one can control home appliances from mobile phone.

CHAPTER 2

LITERATURE SURVEY

1. Abi,Keerthi.Literature review on Home Automation System.(2012).

This literature review discusses IoT-based home automation systems, highlighting how appliances connect to an Arduino BT board via Bluetooth, allowing remote access and control through a smartphone or portable device.

2. Sharma.A and Gupta.R. GSM Based Home Automation using Arduino.(2015)

In GSM based Home Automation Project, one can control the home appliances, using the simple GSM based phone, just by sending SMS through his phone. In this project, no Smart phone is needed, just the old GSM phone will work to switch ON and OFF any home electronic appliances, from anywhere.

3. Kumar, V.Home Automation using NodeMCU and Relay Modules. Journal of Smart Systems and Technologies.(2018)

This project presents an IoT-based home automation system using NodeMCU and relay modules to control home appliances via Wi-Fi. Users can send ON/OFF commands through a mobile app, providing a cost-effective and efficient solution for remote appliance management, particularly benefiting the elderly and disabled.

Here in this project called “Connected Home”, Wi-Fi Module is used to take commands as input. The webpage is designed and used to fetch the input to the NodeMCU module through Wi-Fi. The commands from the microcontroller to the relay module connected to the loads control its switching.

CHAPTER 3

THE PROPOSED SYSTEM

3.1 BLOCK DIAGRAM

The block diagram shows the flow of the hardware part of the project.

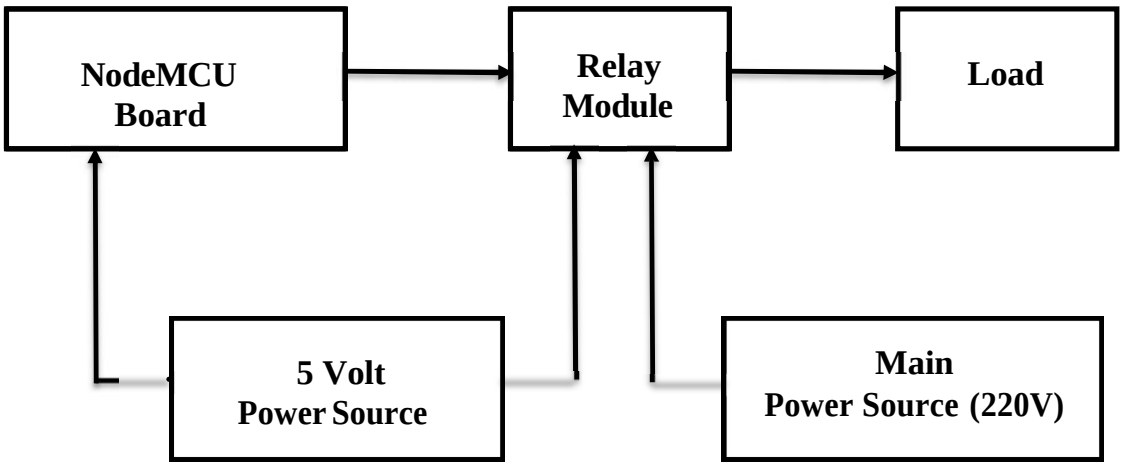

Fig 3.1 Block Diagram of Home Automation System

User Interface: The user interface is a webpage that allows the user to interact with the system. The user can send commands to turn on or off the load, and the webpage can display the status of the load.

NodeMCU: The NodeMCU is a small, low-cost microcontroller withbuilt- in Wi-Fi. It connects to the user interface webpage and receives commands from the user. The NodeMCU then sends these commands to relay. .

Relays: The relays are switches that are controlled by the NodeMCU. They are used to turn the load on or off. The relays can be connected to any kind of load, such as lights, fans, or motors.

Load: The load is the device that is being controlled by the system. It could be any kind of electrical device that can be turned on or off, such as a light bulb, a fan, or a motor.

Overall, the system works by receiving commands from the user interface webpage, sending those commands to the NodeMCU, which then controls the relays, which in turn control the load. The system can also send status updates back to the user interface webpage, allowing the user to see the current state of the load.

3.2 CIRCUIT DIAGRAM

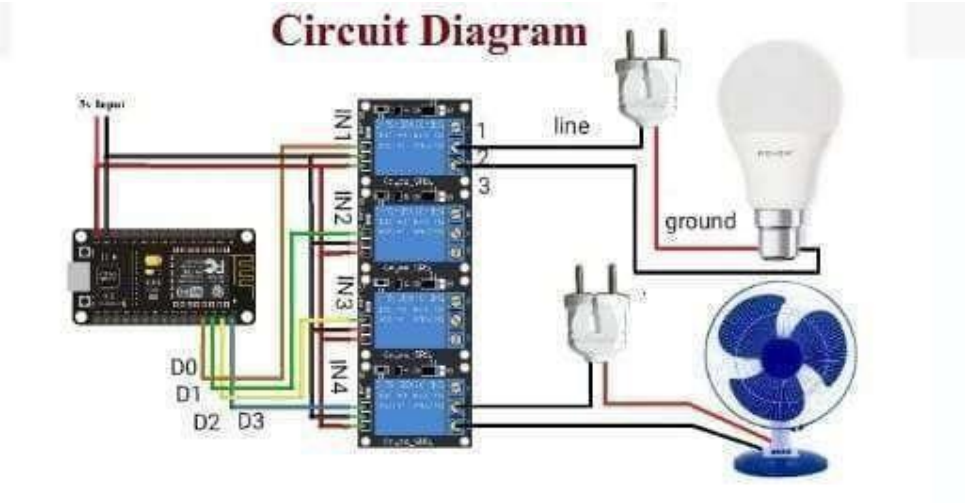

Fig.3.2 Circuit Diagram of Home Automation System

First of all, we will make a connection to the Wi-Fi module. For connecting the Wi-Fi module, we are using Local IP address of the module for connecting the module with our webpage in smartphone. We have to connect the 5V power supply to VCC and GND pins of NodeMCU and Relay module. To connect the relay module with the NodeMCU, use any of the data pins of the NodeMCU to the input pin of the relay module. The line voltage (220 V)is given to the common pin of the relay. The output is taken from Normally open (NC) pin of the relay and given to the load. Make sure the load is grounded.

3.3 COMPONENTS REQUIRED

3.2.1 NODEMCU

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). Strictly speaking, the term "NodeMCU" refers to the firmware rather than the associated development kits.

Both the firmware and prototyping board designs are source. The firmware uses the Lua scripting language.

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.

FEATURES OF THE NODEMCU

Developer	ESP8266 Open-source Community
Type	Single-board microcontroller
Introductory price	300-400 Rs
Operating system	XTOS
CPU	ESP8266

Operating system	XTOS
Memory	128kBytes
Storage	4Mbytes
Power	USB

NodeMCU Pinout and Functions

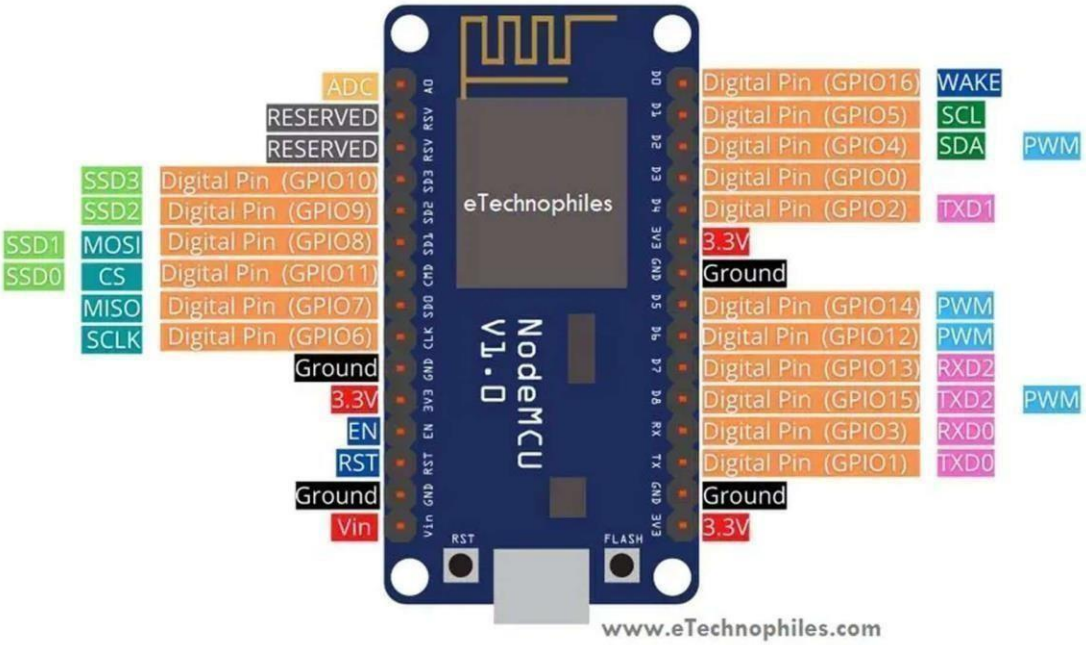

Fig.3.3 NodeMCU

Power Pins There are four power pins. VIN pin and three 3.3V pins.

VIN can be used to directly supply the NodeMCU/ESP8266 and its peripherals. Power delivered on VIN is regulated through the onboard regulator on the NodeMCU module – you can also supply 5V regulated to the VIN pin

3.3 V pins are the output of the onboard voltage regulator and can be used to supply power to external components.

GND are the ground pins of NodeMCU/ESP8266

I2C Pins are used to connect I2C sensors and peripherals. Both I2C Master and I2C Slave are supported. I2C interface functionality

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 clockfrequency of the slave device.

GPIO Pins NodeMCU/ESP8266 has 17 GPIO pins which can be assigned to 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 NodeMCU is embedded with a 10-bit precision SAR ADC. The two functions can be implemented using ADC. 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 NodeMCU/ESP8266 has 2 UART interfaces (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. However, UART1 (TXD1 pin) features only data transmit signal so, it is usually used for printing log.

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

- Timing modes of the SPI format transfer
- Up to 80 MHz and the divided clocks of 80MHz

SDIO Pins NodeMCU/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 μs to 10000 μs (100 Hz and 1 kHz).

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

EN: The ESP8266 chip is enabled when EN pin is pulled HIGH. When pulled LOW the chip works at minimum power.

RST: RST pin is used to reset the ESP8266 chip.

WAKE: Wake pin is used to wake the chip from deep-sleep.

3.2.2 ESP-8266 MODULE (HC-05)

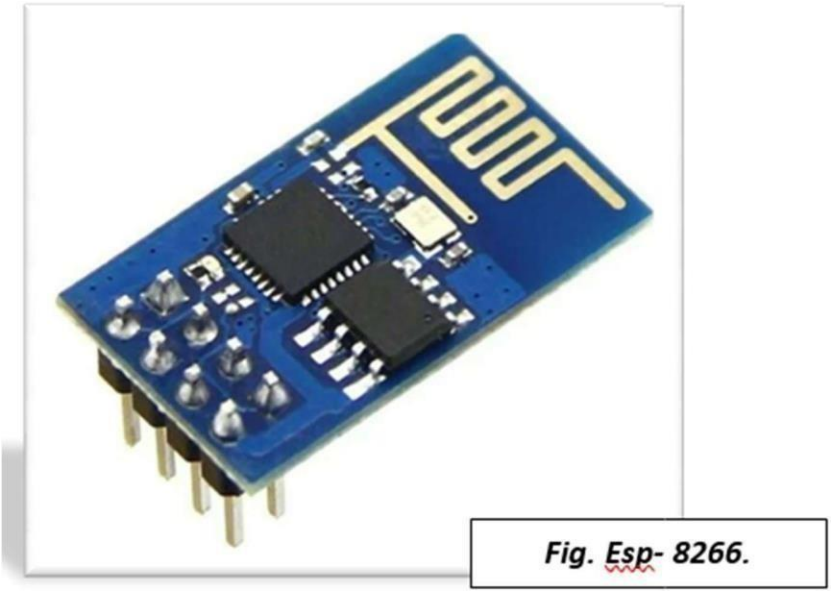

Fig. 3.4 ESP 8366

INTRODUCTION

The chip first came to the attention of Western makers in August 2014 with the ESP- 01 module, made by a third-party manufacturer Ai-Thinker.

This small module allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

The very low price and the fact that there were very few external components on the module, which suggest that it could eventually be very inexpensive in volume, attracted many hackers to explore the module, chip, and the software on it, as well as to translate the Chinese documentation.

The ESP8266 is an ESP8266 with 1 MiB of built-in flash, allowing for single-chip devices capable of connecting to wi-fi.

FEATURES:

- Low cost, compact and powerful Wi-Fi
  Module Power Supply: +3.3V only
- Current Consumption: 100mA
- I/O Voltage: 3.6V (max)
- I/O source current: 12mA (max)
- Built-in low power 32-bit MCU @ 80MHz
- 512kB Flash Memory
- Can be used as Station or Access Point
- Supports Deep sleep (<10uA)
- Supports serial communication hence compatible with many development platforms like Arduino
- Can be programmed using Arduino IDE or AT-commands or Lua Script

SPECIFICATIONS:

- 2.4 GHz Wi-Fi (802.11 b/g/n, supporting
  WPA/WPA2). General-purpose input/output (16
  GPIO).
- Inter-Integrated Circuit (I^2C) serial communication
  protocol.
- Analog-to-digital conversion (10-bit ADC).
- Serial Peripheral Interface (SPI) serial communication protocol.
- I^2S (Inter-IC Sound) interfaces with DMA (Direct Memory Access)(sharing pins).

- UART (on dedicated pins, plus a transmit-only UART can be enabled onGPIO2).
- Pulse-width modulation (PWM).
- It employs a 32-bit RISC CPU (reduced instruction set computer) based onthe Tensilica Xtensa L106 running at 80 MHz (or overclocked to 160 MHz).

Pin Name	Pin No.	Alternate Pin Name	Use of Pin	Alternate Use of Pin (if any)
GND	1	-	Ground	-
GPIO16	2	-	General Purpose Input/Output	Wake up from Deep Sleep
GPIO14	3	HSCLK	General Purpose Input/Output	SPI clock signal
GPIO12	4	HMISO	General Purpose Input/Output	SPI Master Input Slave Output
GPIO13	5	HMOSI	General Purpose Input/Output	SPI Master Output Slave Input
GPIO15	6	HCS	General Purpose Input/Output	SPI chip select signal
VCC	7	-	Power Supply	-
EN	8	-	Enable/Boot mode	-
GPIO2	9	-	General Purpose Input/Output	Alternate boot mode
GPIO0	10	-	General Purpose Input/Output	Boot mode selection
RX	11	-	UART Receive	-
TX	12	-	UART Transmit	-
CH_PD	13	-	Chip Power Down	-

Table 3.1 pin configuration of ESP8266

3.2.3 RELAY

A relay is an electrically operated switch. Many relays use an electromagnet to mechanically operate a switch, but other operating principlesare also used, such as solid-state relays. Relays are used where it is necessary to control a circuit by a separate low-power signal, or where several circuits must be controlled by one signal. The first relays were used in long-distance telegraph circuits as amplifiers: they repeated the signal coming in from one circuit and re- transmitted it on another circuit. Relays were used extensively in telephoneexchanges and early computers to perform logical operations.

A type of relay that can handle the high power required to directly control an electric motor or other loads is called a contactor. Solid-state relays control power circuits with no moving parts, instead of using a semiconductor device to perform switching. Relays with calibrated operating characteristics and sometimesmultiple operating coils are used to protect electrical circuits from overload or faults; in modern electric power systems these functions are performed by digitalinstruments still called "protective relays".

Magnetic latching relays require one pulse of coil power to move their contacts in one direction, and another, redirected pulse to move them back. Repeated pulses from the same input have no effect. Magnetic latching relays are useful in applications where interrupted power should not be able to transition the contacts.

8 CHANNEL RELAY FEATURES

Number of Relays: 8 Control signal: TTL level

Rated load: 7A/240VAC 10A/125VAC 10A/28VDC

Contact action time: 10ms/5ms

TYPES:

COAXIAL RELAY

A coaxial relay is often used as a TR (transmit-receive) relay where radio transmitters and receivers share one antenna. It switches the antenna from the receiver to the transmitter. This protects the receiver from the high power of the transmitter. Such relays are often used in transceivers that combine transmitter and receiver in one unit. The relay contacts are designed not to reflect any radio frequency power toward the source, and to provide very high isolation between receiver and transmitter terminals. The characteristic impedance of the relay is matched to the transmission line impedance of the system, for example, 50 ohms.

CONTACTOR

A contactor is a heavy-duty relay with higher current ratings, used for switching electric motors and lighting loads. Continuous current ratings for common contactors range from 10 amps to several hundred amps. High-current contacts are made with alloys containing silver. The unavoidable arcing causes the contacts to oxidize; however, silver oxide is still a good conductor. Contactorswith overload protection devices are often used to start motors.

FORCE-GUIDED CONTACTS RELAY

A 'force-guided contacts relay' has relay contacts that are mechanically linked together, so that when the relay coil is energized or de-energized, all of the linked contacts move together. If one set of contacts in the relay becomes immobilized, no other contact of the same relay will be able to move. The function of force-guided contacts is to enable the safety circuit to check the statusof the relay. Force-guided contacts are also known as "positive-guided contacts", "captive contacts", "locked contacts", "mechanically linked contacts", or "safety relays".

APPLICATIONS OF RELAY

Relays are used wherever it is necessary to control a high power or high voltage circuit with a low power circuit, especially when galvanic isolation is desirable. The first application of relays was in long telegraph lines, where the weak signal received at an intermediate station could control a contact, regenerating the signal for further transmission. High-voltage or high-current devices can be controlled with small, low-voltage wiring and pilot switches. Operators can be isolated from the high voltage circuit. Low power devices such as microprocessors can drive relays to control electrical loads beyond their direct drive capability. In an automobile, a starter relay allows the high current of the cranking motor to be controlled with small wiring and contacts in the ignition key.

RELAY H152S MODULE OVERVIEW

A Relay is a switch that is electrically operated by an electromagnet. The electromagnet is activated with a low voltage, for example, 5 volts from a microcontroller and it pulls a contact to make or break a high voltage circuit.

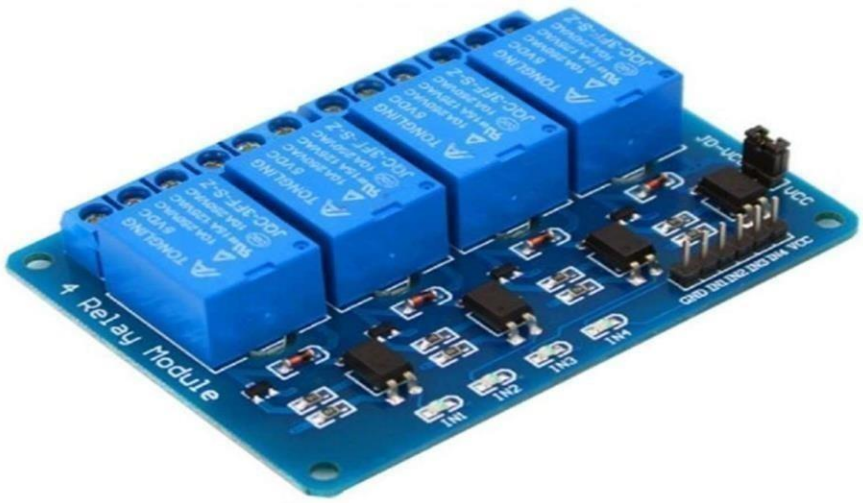

Fig.3.5 Channel Relay Module

As an example, for this Arduino Relay Tutorial, we will use the HL-52S 2 channel relay module, which has 2 relays with a rating of 10A @ 250 and 125 V AC and 10A @ 30 and 28 V DC. The high voltage output connector has 3 pins, the middle one is the common pin, and as we can see from the markings one of the two other pins is for a normally open connection and the other one for a normally closed connection.

On the other side of the module, we have these 2 sets of pins. The first one has 4 pins, a Ground and a VCC pin for powering the module, and 2 input pins In1 and In2. The second set of pins has 3 pins with a jumper between the JDVcc and the Vcc pin. With a configuration like this, the electromagnet of the relay is directly powered by the Arduino Board and if something goes wrong with the relay the microcontroller could get damaged.

CIRCUIT SCHEMATIC

For better understanding let's see the circuit schematics of the relay module in this configuration. So, we can see that the 5 volts from our microcontroller connected to the Vcc pin for activating the relay through the Optocoupler IC are also connected to the JDVcc pin which powers the electromagnet of the relay. So, in this case we got no isolation between the relay and the microcontroller.

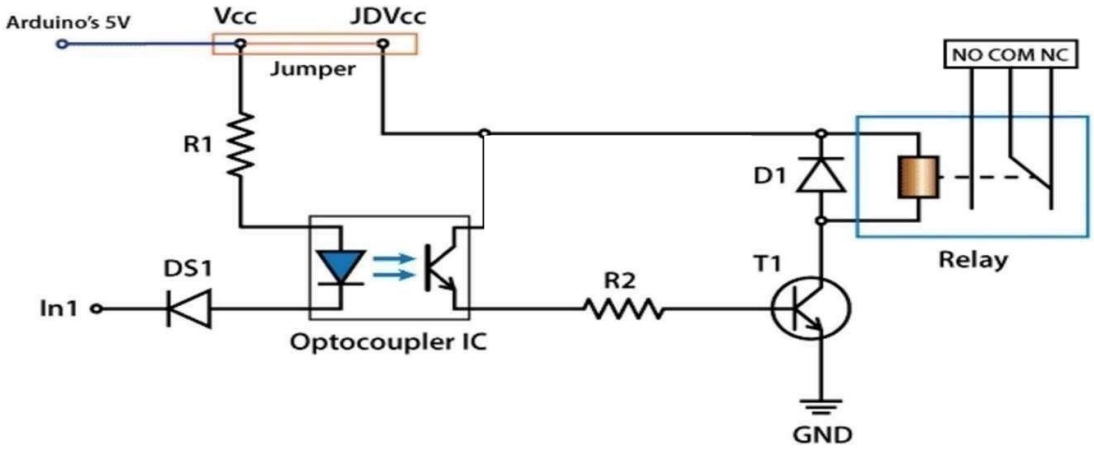

Fig. 3.6 Circuit Diagram for Relay Module-1

To isolate the microcontroller from the relay, we need to remove the jumper and connect a separate power supply for the electromagnet to the JDVcc and the Ground pin. Now with this configuration, the microcontroller doesn't have any physical connection with the relay, it just uses the LED light of the Optocoupler IC to activate the relay.

There is one more thing to be noticed from this circuit schematics. The input pins of the module work inversely. As we can see the relay will be activatedwhen the input pin will be LOW because in that way the current will be able to flow from the VCC to the input pin which is low or ground, and the LED will light up and activate the relay.When the input pin will be HIGH there will be no current flow, so the LED will not light up and the relay not be activated.

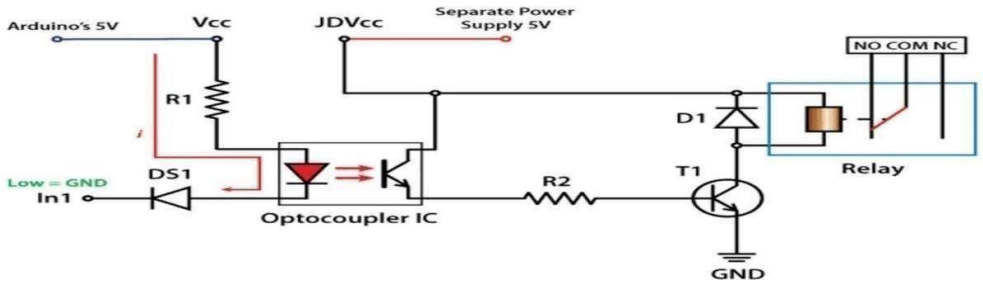

Fig. 3.7 Circuit Diagram for Relay Module-2

First, let's take a look at the circuit diagram. As previously described, we will use a 5V Adapter as a separate power supply for the electromagnet connected to the JDVcc and the Ground pin. The Arduino's 5V pin will be connected to the Vcc pin of the module and PIN 7 to the In1 input pin for controlling the relay. Now for the HIGH Voltage part, we need a power plug, a socket, and a cable withtwo wires. One of the two wires will be cut and connected to the common and the normally open pin of the module output connector. So, with this configuration when we will activate the relay, we will get the high voltage circuit closed and working.

Here's how made the cable. So, I bought a plug, a socket, and a cable. ThenI carefully cut the cable and cut one of the wires as shown in the picture below andconnect them to the normally open connection pins of the relay module. Also connected the ends of the cable to the plug and the socket.

3.4 ARDUINO SOFTWARE PART - ARDUINO IDE

The Arduino integrated development environment (IDE) is a cross- platform application (for Windows, macOS, and Linux) that is written in the programming language Java. It originated from the IDE for the languages Processing and Wiring. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple one- click mechanisms to compile and upload programs to an Arduino board. It also contains a message area, a text console, a toolbar with buttons for common functions, and a hierarchy of operation menus. The source code for the IDE is released under the gnu general public license, version2.

The Arduino IDE supports the languages C and C++ using special rules of code structuring. The Arduino IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, which is compiled and linked with a program stub main() into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The Arduino IDE employs the program avrdude to convert the executable code into a text file in hexadecimal encoding that is loaded into the Arduino board by a loader program in the board's firmware.

SKETCH

A program written with the Arduino IDE is called a sketch. Sketches are saved on the development computer as text files with the file extension .ino. Arduino Software (IDE) pre-1.0 saved sketches with the extension .pde.

A minimal Arduino C/C++ program consists of only two functions:

setup(): This function is called once when a sketch starts after power-up or reset. It is used to initialize variables, input and output pin modes, and other libraries needed in the sketch.

loop(): After setup() has been called, function loop() is executed repeatedly in the main program.

It controls the board until the board is powered off or reset.

Blink example

Most Arduino boards contain a light-emitting diode (LED) and a load resistor connected between pin 13 and the ground, which is a convenient feature for many tests and program functions. A typical program for a beginning Arduino programmer blinks a LED repeatedly. This program uses the functions pinMode(), digitalWrite(), and delay(), which are provided by the internal libraries included in the IDE environment. This program is usually loaded into a new Arduino board by the manufacturer.

3.5 WEBPAGE:

HTML is the standard markup language for creating websites. CSS is the language that describes the style of an HTML document. When it comes to building a responsive design, it is a challenge to choose between toggle switches, checkboxes or radio buttons. However, using toggle switches is a better choice as it allows us to select between opposite modes, such as on/off. CSS toggle switches are an excellent technique to increase website responsiveness and hence we have combined HTML and CSS to create our web page.

Here we have inserted few screenshots of our webpage.

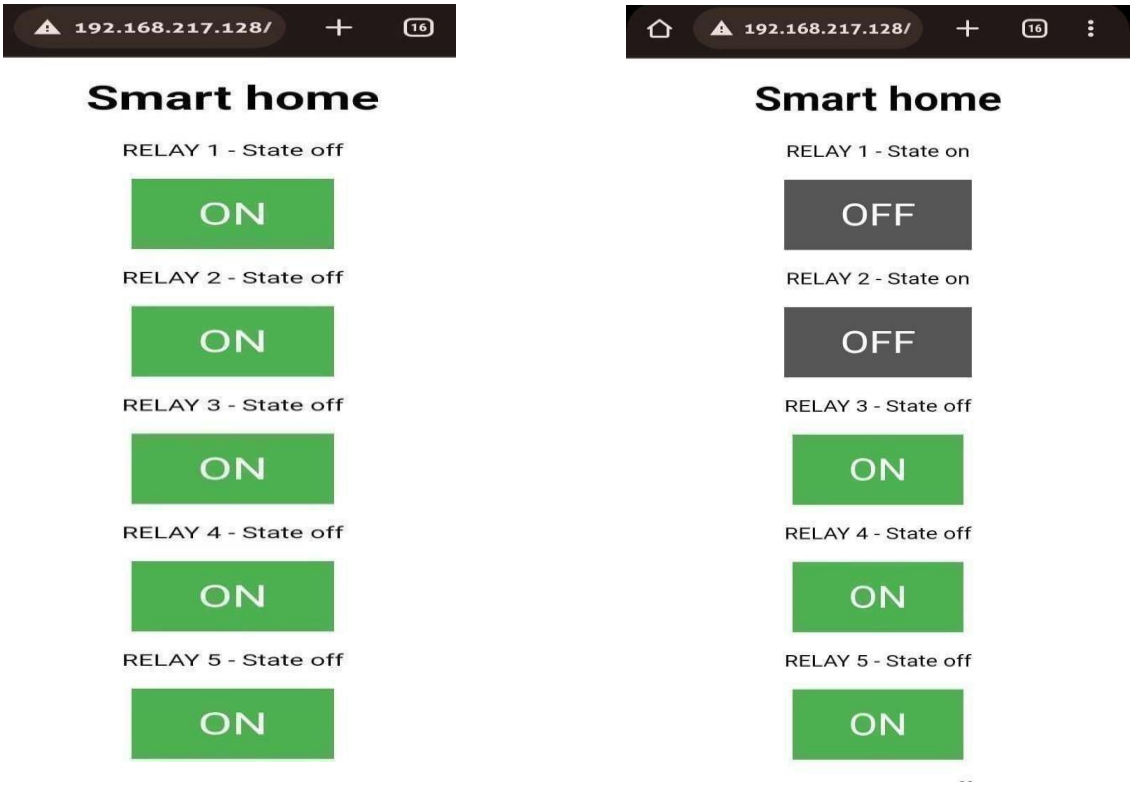

Fig. 3.8 Screenshots of our webpage

3.6 WORKING :

First, we need to choose Commands to control the devices. Here, the “ON” command in the webpage is used to Turn the Load ON, and the “ OFF” command is used to Turn the Load OFF. The table below shows the commands from the webpage and its response by first two relays (we have used 8 relays) which will give a clear understanding of the basic working.

Table 3.2 Command and response

Commands	**Device Status**	**Control Relay (Relay Pin)**
ON	Relay 1-State on	Relay-1 (IN1)
OFF	Relay 1-State off	
ON	Relay 2-State on	Relay-2 (IN2)
OFF	Relay 2-State off	

When we send a command through the webpage, the Wi-Fi modulereceives that command and passes it to the NodeMCU. Now NodeMCU compares this command with the predefine Commands (which are defined in Arduino code)and sends a command to operate the relay module. We can see the Device status (on or off) in the webpage.

For example, when we send the “ON” command for relay one through the page, then NodeMCU gets this command through the Wi-Fi module. Then the NodeMCU sends a low input voltage to the Input-1 (IN1) pin of the relay module(active high relay). Now the relay is in ON mode. So, the device which isconnected to the relay-1 of the relay module will also turn on. At the same time,

“Relay 1- State ON” status is shown in webpage.

When we send the **“OFF”** command through the webpage, Again the NodeMCU gets this command through the Wi-Fi module. This time the NodeMCU sends a high input voltage to the **Input-1(IN1)** pin of the relay module(active high relay). Now the relay is in **Off mode**. So, the device, which is connected to the relay-1 of the relay module will also turn off. At the same time, the **“Relay 1- State OFF”** status is displayed in the webpage.

In this way, we can control all relays of the 8 channel Relay module through the commands from webpage.

CHAPTER 4

SOFTWARE DEVELOPMENT MODULES

4.1 SOFTWARE INITIALIZATION

In this project,We use the C language as a programming language for Arduino to run the Arduino microcontroller. Relay as a switch to move the solenoid is set in pin 6 in the Arduino microcontroller. Output pin 13 in the relay will be in high condition and 1s delay after the user gives an order. The USB connection with the PC is necessary to program the board and not just to power it up. The Uno automatically draws power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power **LED (labeled PWR)** should go on.

Arduino software (IDE): It is open-source software (IDE) that makes it easy for us to write code and upload it to the Arduino boards. It runs on Windows, Linux, etc.

SELECT YOUR BOARD TYPE AND PORT

Select the entry in the Tools > Board menu that corresponds to your Arduino board. Select the serial device of the board from the Tools | Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out this, disconnect the board and reopen the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

Fig. 4.1.1-Arduino table

UPLOAD THE PROGRAM

Now, simply click the "Upload" button in the environment. Wait a few seconds - then the RX and TX led on the board flashing. If the upload issuccessful, the message "Done uploading." will appear in the status bar.

A few seconds after the upload finishes, then the pin 13 (L) LED on the board start to blink (in orange). If it does, Arduino is up-and-running**.**

Fig 4.1.2-Arduino table

4.2 ARDUINO CODE

```
#include <ESP8266WiFi.h>

// Replace with your network credentials
// to create its own wifi network
const char* ssid     = "Niranjan";
const char* password = "12092002";
// Assign output variables to GPIO pins
const int output12 = D0;
const int output13 = D1;
const int output14 = D2;
const int output15 = D3;
const int output16 = D4;
const int output17 = D5;
const int output18 = D6;
const int output19 = D7;
const int output20 = D8;


WiFiServer server(80);
String header;
String output12State = "off";
String output13State = "off";
String output14State = "off";
String output15State = "off";
String output16State = "off";
```

```
String output17State = "off";
String output18State = "off";
String output19State = "off";
String output20State = "off";

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(output12, OUTPUT);
  pinMode(output13, OUTPUT);
  pinMode(output14, OUTPUT);
  pinMode(output15, OUTPUT);
  pinMode(output16, OUTPUT);
  pinMode(output17, OUTPUT);
  pinMode(output18, OUTPUT);
  pinMode(output19, OUTPUT);
  pinMode(output20, OUTPUT);

  // Set outputs to high because we are using active low type relay module
  digitalWrite(output12, HIGH);
  digitalWrite(output13,  HIGH);
  digitalWrite(output14,  HIGH);
  digitalWrite(output15,  HIGH);
  digitalWrite(output16,  HIGH);
  digitalWrite(output17,  HIGH);
  digitalWrite(output18,  HIGH);
  digitalWrite(output19, HIGH);
```

```
  digitalWrite(output20, HIGH);

   connectWiFi();
    while ((!(WiFi.status() == WL_CONNECTED)))
    {
      connectWiFi();
    }

   server.begin();
}

void loop()
{
  while ((!(WiFi.status() == WL_CONNECTED)))
    {
      connectWiFi();
    }

  WiFiClient client = server.available(); // Listen for incoming clients

  if (client) {                          // If a new client connects,
    Serial.println("New Client.");          // print a message out in the serial port
    String currentLine = "";                // make a String to hold incoming data from
the client
    while (client.connected()) {            // loop while the client's connected
      if (client.available()) {             // if there's bytes to read from the client,
      char c = client.read();               // read a byte, then
```

```
        Serial.write(c);                    // print it out the serial monitor
        header += c;
        if (c == '\n') {                    // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200
OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection:     close");
            client.println();

            // turns the GPIOs on and off
            //foe GPIO12
            if (header.indexOf("GET /12/on") >= 0)
            {
              Serial.println("RELAY 1 on");
              output12State = "on";
              digitalWrite(output12, LOW);
            }
            else if (header.indexOf("GET /12/off") >= 0)
            {
              Serial.println("RELAY 1 off");
              output12State = "off";
              digitalWrite(output12, HIGH);
```

```
}

//for GPIO13
else if (header.indexOf("GET /13/on") >= 0)
{
  Serial.println("RELAY 2 on");
  output13State = "on";
  digitalWrite(output13,LOW);
}
else if (header.indexOf("GET /13/off") >= 0)
{
  Serial.println("RELAY 2 off");
  output13State = "off";
  digitalWrite(output13, HIGH);
}


 //for GPIO14
 else if (header.indexOf("GET /14/on") >= 0)
{
  Serial.println("RELAY 3 on");
  output14State = "on";
  digitalWrite(output14, LOW);
}
else if (header.indexOf("GET /14/off") >= 0)
{
  Serial.println("RELAY 3 off");
```

```
  output14State = "off";
  digitalWrite(output14,HIGH);
}

//for GPIO15
else if (header.indexOf("GET /15/on") >= 0)
{
  Serial.println("RELAY 4 on");
  output15State = "on";
  digitalWrite(output15, LOW);
}
else if (header.indexOf("GET /15/off") >= 0)
{
  Serial.println("RELAY 4 off");
  output15State = "off";
  digitalWrite(output15,HIGH);
}

//for GPIO16
else if (header.indexOf("GET /16/on") >= 0)
{
  Serial.println("RELAY 5 on");
  output16State = "on";
  digitalWrite(output16, LOW);
}
else if (header.indexOf("GET /16/off") >= 0)
{
```

```
  Serial.println("RELAY 5 off");
  output16State = "off";
  digitalWrite(output16, HIGH);
}

//for GPIO17
else if (header.indexOf("GET /17/on") >= 0)
{
  Serial.println("RELAY 6 on");
  output17State = "on";
  digitalWrite(output17, LOW);
}
else if (header.indexOf("GET /17/off") >= 0)
{
  Serial.println("RELAY 6 off");
  output17State = "off";
  digitalWrite(output17, HIGH);
}

//for GPIO18
else if (header.indexOf("GET /18/on") >= 0)
{
  Serial.println("RELAY 7 on");
  output18State = "on";
  digitalWrite(output18, LOW);
}
else if (header.indexOf("GET /18/off") >= 0)
```

```
{
  Serial.println("RELAY 7 off");
  output18State = "off";
  digitalWrite(output18,HIGH);
}

//for GPIO19
else if (header.indexOf("GET /19/on") >= 0)
{
  Serial.println("RELAY 8 on");
  output19State = "on";
  digitalWrite(output19, LOW);
}
else if (header.indexOf("GET /19/off") >= 0)
{
  Serial.println("RELAY 8 off");
  output19State = "off";
  digitalWrite(output19, HIGH);
}

//for GPIO20
else if (header.indexOf("GET /20/on") >= 0)
{
  Serial.println("RELAY 9 on");
  output20State = "on";
  digitalWrite(output20, LOW);
}
```

```
else if (header.indexOf("GET /20/off") >= 0)
{
  Serial.println("RELAY 9 off");
  output20State = "off";
  digitalWrite(output20, HIGH);
}
```

CHAPTER 5

RESULT AND DISCUSSION

REAL-TIME IMPLEMENTATION

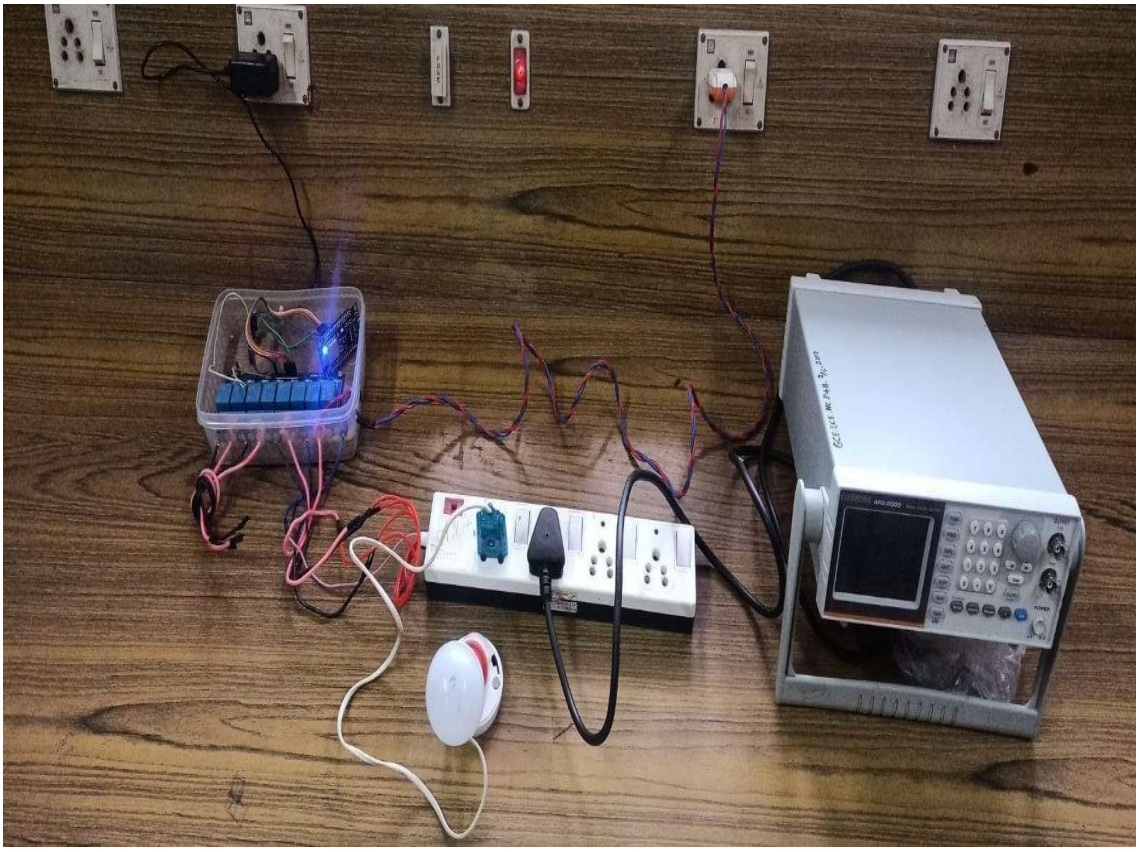

Fig. 5.1 Real-Time Implementation

The above figure is the real-time implementation of this mini-project. Here, only loads to two relays out of 8 relays in the module are connected but the code is capable of handling 8 loads.

RELAY -1 “ON” STATE

Fig. 5.2 Relay 1 in ON State

By giving ON command for relay 1 through the webpage, the load connected to relay one is turned on.

RELAY 2 “ON” STATE

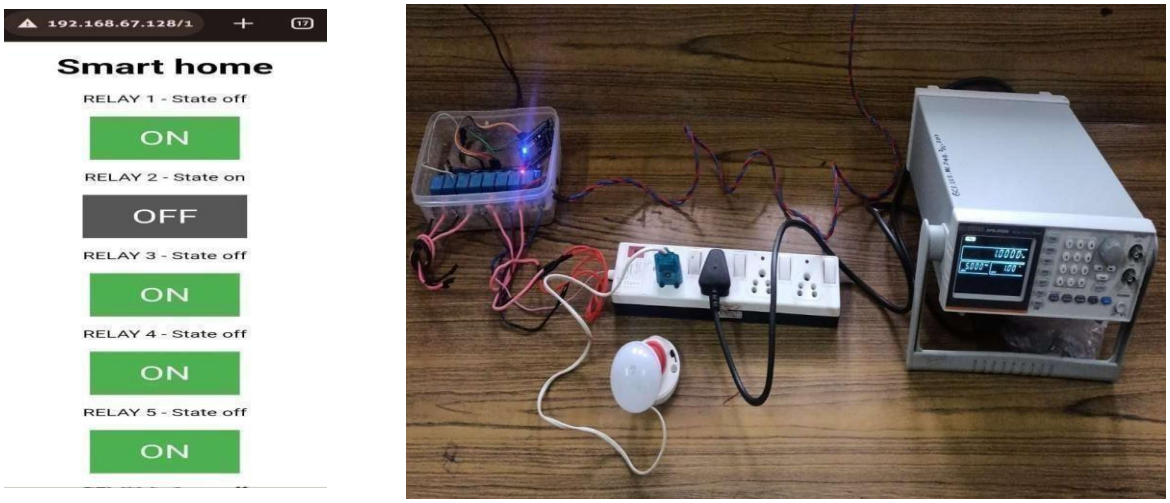

Fig. 5.3 Relay 2 in ON State

By giving ON command for relay 2 through the webpage, the load connected to relay 2 is turned on.

BOTH RELAY 1 AND RELAY 2 IN “ON” STATE

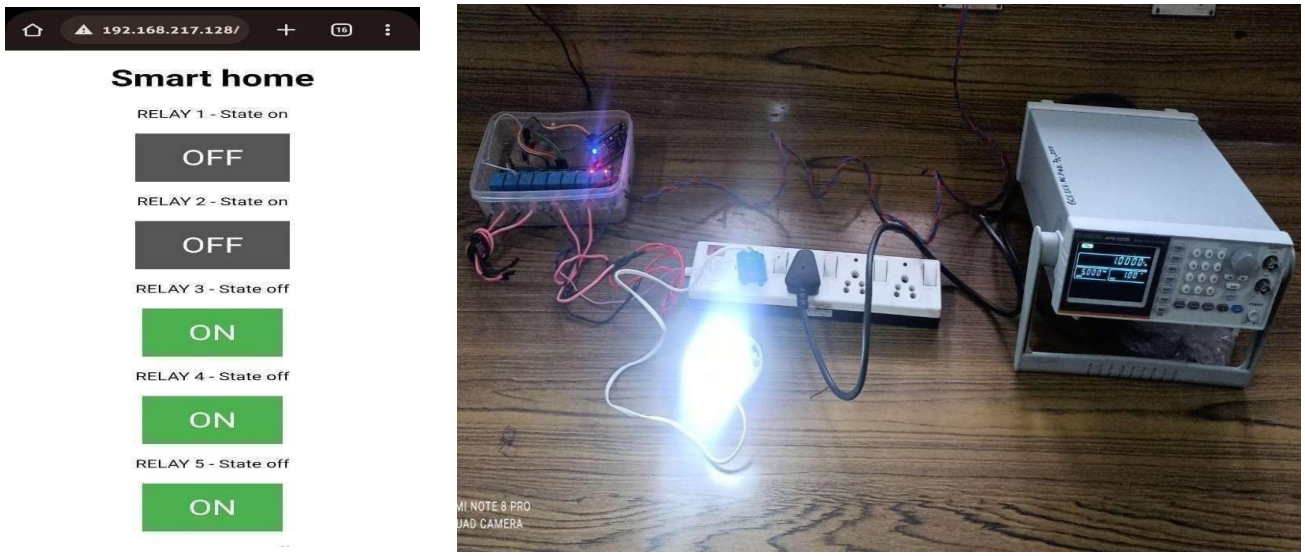

Fig. 5.4 Both Relay 1 and Relay 2 in ON State

By giving ON command for both the relays through the webpage, the load connected to both the relays are turned on.

CHAPTER 6
CONCLUSION

As the name indicates, ‘Connected home’ makes the system more flexible and provides an attractive user interface compared to other home automation systems. In this system, mobile phones are integrated into home automation systems. A novel architecture for a home automation system is proposed using relatively new communication technologies. The system consists of mainly two components a NodeMCU (Wi-Fi module, ESP8266 microcontroller) and a relay module. Wi-Fi is used as the communication channel between the android phone and the microcontroller. The complexity of the notions involved in the home automation system are hidden by including them in a simple, but comprehensive set of related concepts. This simplification is needed to fit as much of the functionality on the limited space offered by a mobile phone’s display. This paper proposes a low-cost, secure, ubiquitously accessible, auto- configurable, remotely controlled solution. The approach discussed in the paper is novel and has achieved the target to control home appliances remotely using Wi-Fi technology to connect system parts, satisfying user needs and requirements. Wi-Fi technology- capable solution has proved to be controlled remotely, provide homesecurity, and is cost- effective as compared to the previously existing systems. Hence, it is concluded that the required goals and objectives of the home automation system have been achieved. The system design and architecture were discussed, and the prototype presents the basic level of home appliance control and remote monitoring has been implemented. Finally, the suggested system outperforms the commercially available home automation systems as it much simpler and convenient to use.

FUTURE ENHACEMENT

The future enhancement for IoT-based smart homes is enormous and has the potential to revolutionize the way we live. With the increasing adoption of smart home devices and the proliferation of IoT technology, it is expected to see more advanced and sophisticated smart home systems in the future.

Integration of smart home systems with other IoT devices and services, such as wearables, smart appliances, and home security systems will enable more seamless and interconnected smart home ecosystems that can provide a range of benefits, such as enhanced safety and security, improved energy efficiency, and more convenient and comfortable living.

Overall, the future scope for IoT-based smart homes is vast and promising. More advanced, wiser, and interconnected smart home solutions that benefit homeowners and enhance their quality of life are excepted.

CHAPTER 7
REFERENCE

1.K. A. H. Ahmed El Shafee, “Design and Implementation of a Wi-Fi Based Home Automation System,” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. Vol: 6, no. No: 8, 2012

2.S. A. A. W. C. a. n. M. V. H. Rozita Teymourzadeh, “Smart GSM Based Home Automation System,” in IEEE Conference on Systems, Process & Control, Kuala Lumpur, Malaysia, 2013

3.M. G. S. M. R. K. A. K. Kim Baraka, “Low-cost Arduino/Android-based Energy- Efficient Home Automation System with Smart Task Scheduling,” in Fifth International Conference on Computational Intelligence, Communication Systems and Networks., London, 2013.

4.M. G. S. M. R. K. A. K. Kim Baraka, “Smart Power Management System For Home Appliances And Wellness Based On Wireless Sensors Network And Mobile Technology,” in XVIII AISEM Annual Conference, 2015.

5. Abhijit Shejal, Amit Pethkar, Akash Zende, Pratyusha Awate, Prof.Sudhir.G.Mane, “DESIGNING OF SMART SWITCH FOR HOME AUTOMATION.”

6. Sudha Kousalya, G Reddi, Priya Vasanthi, B Venkatesh, IOT Based Smart Security and Smart Home Automation presented at International Journal of Engineering Research & Technology 04, April-2018.

7. Home Automation & Wiring (1 ed.). New York: McGraw-Hill/TAB Electronics. 1999- 03-31. ISBN 9780070246744.

8.B.-R. L. J.-L. P. a. C. J. L. Shih-Pang Tseng, “An Application of Internet of Things with Motion Sensing on Smart House,” in IEEE, 2014.