International Journal of Engineering &Technology, 7 (4) (2018) 6358-6366

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i4.13259
Research paper

IoT based design and implementation of WSN smart home automation intelligent system using flask web server

G. Mallikharjuna Rao *

Dept. Of ECE, CBIT, Osmania University, Hyderabad
**Corresponding author E-mail: mallikharjunag@gmail.com*

Abstract

The smart home is changing the way we live today, in this project we introduce to automate the entire home appliances using wireless sensor network to control and monitor the connectivity and communication using Raspberry Pi. The Raspberry Pi is used as central coordinator to connect with four NodeMCU wirelessly operating under star network topology, to handle the home appliances. The Raspberry Pi and NodeMCU modules are connected to the internet using the hotspot. The coordinator is monitored and controlled through HTTP web page protocol. The web page is developed using HTML, CSS, JavaScript, and chart.js. The website hosted on the flask web server, through the webpage automation system control the internet through Wi-Fi module, it makes a complete setup of home automation using the Internet of Things. Finally, the entire core design of hardware and software is developed to manage the home appliances through a Flask web server is done in local storage, can be viewed in a web browser using the client-server model.

Keywords: *Wireless Sensor Networks; Client-Server Model; Web Page; Flask Web Server; Internet of Things.*

1. Introduction

In Smart Home Automation, so many means of technologies are used to monitor and to control the home appliances [1]. Compare the applications of different technologies like Bluetooth, ZigBee [2] and Wi-Fi [3-5] wireless techniques used so far. The technologies, limitations are due to latency, distance, the speed of operation, cost-effectiveness, and internet for interactive devices across the network. Hence, to improve the operation limits, and to implement the smart home automation system for household appliances using WSN [6-7] Star topology is missing in the literature.
The explanation to the part of the problem statement as shown in Figure 1. The router is a part of the wireless sensor network, the connections of the Raspberry Pi, and four Wi-Fi modules are communicated through the hotspot of a router to control home appliances. The Raspberry Pi is portable with open source; the Raspbian operating system is loaded into the memory card to access the board as PC. The putty software is used to access the Raspberry Pi wirelessly to reconfigure it as per updates are considered to have an approach to control the household appliances, which helps to operate the integrated devices connected to the star topology network. The Arduino IDE with third-party platform tools used to develop the code and implement it on the NodeMCU modules. The flask web server is used to deploy the webpage, where the webpage is designed using HTML, CSS, HTTP, JavaScript, RESTful API, for obtaining the web socket connectivity to maintain communication within the local database between client-server protocols make interconnection between the Raspberry Pi and the established node. The design and implementation of the hardware and the software co-design are used to coordinate the complete household networking locally with cost-effective equipment.

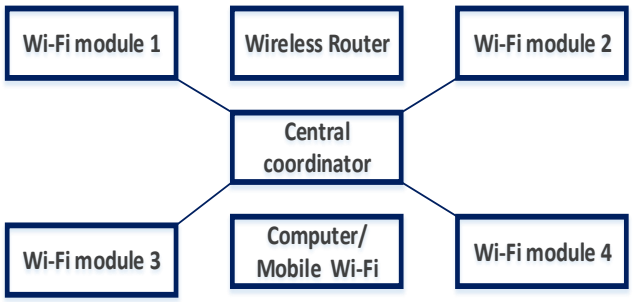

Fig. 1: Block Diagram of WSN Using A Star Topology

Section 2 to discuss the related work. Section 3 to state hardware details. Section 4 describes the software tools required for the project. Section 5 complete hardware implementation in the project. Section 6 software implementation with protocols using different open source software tools. Section 7 to address results and discussions. Section 8 concluding remarks.

2. Related work

Using Bluetooth technology (BT) for home automation system using a cell phone is covered in [1]. Both ZigBee and Bluetooth interface for automation discussed in [2]. To design economical home automation system using Wi-Fi devices [3]. Multiple connections to handle the household appliances discussed in [4]. PIR sensors for automation of surveillance system [5]. Node authentication for household automation system [6]. Wireless sensor networks in urban catchments [7]. The Smart home security automation system [8]. The Validity client-server model approach [9].

3. Hardware

The project co-design of hardware and software implementation to obtain linkage between Raspberry Pi and NodeMCU modules. To configure and communicate the wireless sensor networks in star topology operation, the Raspberry Pi is connected to the internet, to act as a coordinator and remaining four nodes as clients to exchange information through Raspberry Pi. It provides service through the webpage to computing devices from home appliances to the local network.

3.1. Wireless sensor network

A WSN is a Wireless Sensor Network contains spatially distributed independent devices using sensors to monitor physical and environmental circumstances. The system enables a gateway to provide a wireless connection to distributed nodes. The Star topology interface as shown in Figure 2, is to connect the devices in which each node directly related to a coordinator.

Fig. 2: Star Topology.

4. Software configuration

The software used in Raspberry Pi is the Raspbian operating system, Python language, putty software, HTML, CSS, JavaScript, Restful API, flask web server, Web socket and Arduino IDE.

4.1. Arduino IDE

The programming tool is used to develop the code for NodeMCU's to fix the exchange of commands and process from one node Wi-Fi module to other node modules to communicate the data to Raspberry Pi in Star topology.
The steps involved in developing:

1) In Arduino IDE, to install NodeMCU board in board manager is to allow the third-party platform package.
2) Start preferences in the window with JSON in additional board manager Uniform Resource Locator (URL) field.

4.2. Raspberry Pi installation and configuration

In Raspberry Pi is to boot and load the operating system:

1) Raspbian Operating System image is available in Raspberry Pi website download in zip format; we need to unzip the file to get the image, to write it into an SD card.
2) Etcher software is to load the image into a Secure Digital (SD) card by using a card reader.
3) Insert Bootable SD card into Raspberry Pi, connect an HDMI cable to the monitor, a keyboard, and a mouse to the USB port. For example, enter the username as "pi" and the password as "raspberrypi" to open the operating system.
4) Raspberry Pi software configuration tools are used to select the Interfacing options are shown in Figure 3. the

SSH server: The Raspberry Pi access remotely from laptop or PC wirelessly should be on the same router.
VNC server: To connect the screen of Raspberry Pi remotely using VNC viewer app.
Camera: To enable Pi camera Interfaced to the Raspberry Pi board.
Expand file system: The operating system image distribution is of 2 GB, for an unused portion of the SD card enabled expand file system partition. It allows the remaining free space of the SD card, then reboot the OS to configure the options permitted by the user.

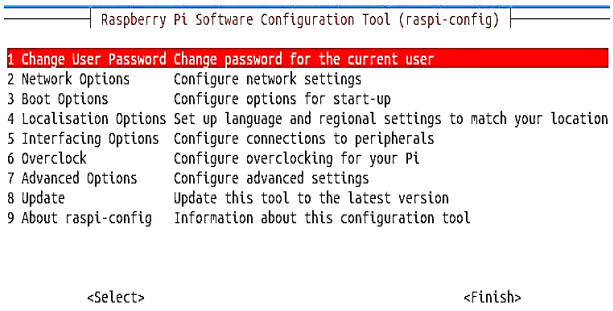

Fig. 3: Interfacing Options Window.

Update: Updating the raspi_config is required to set the additional options to setup Wi-Fi local, memory splitting, and boot options.

4.3. Putty client connection and VNC server

Access the Raspberry Pi board wirelessly on PC, the putty configuration software installed on a Windows PC. Enter the IP address hostname or IP address field as shown in Figure 4.

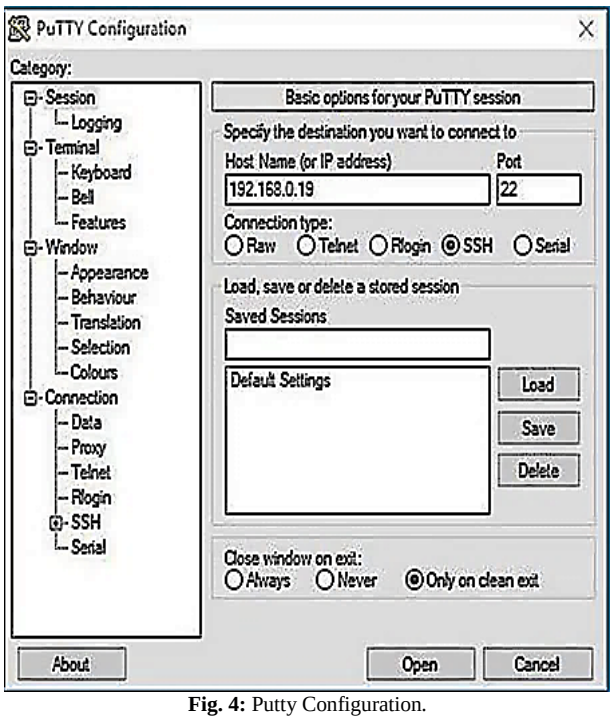

Fig. 4: Putty Configuration.

Click open to get security alert in the terminal window display, enter username and password. The usual command line replaced with my@RaspberryPi:, now you are logged in and working with the command line. To view the connection obtained wirelessly Interface from Raspberry Pi to PC, the command line alternatives (such as Nano or VIM) are used to achieve communication and easy to share files using the secure shell with VNC server. Connect the Raspberry Pi to a local router, use ifconfig to find the IP address, we need IP address to log in. Once Nano editor is open, edit the content of the file, and then exit.
But super user privileges are used to execute commands on different users in a controlled way. Sudo doesn't start a new shell, Advanced package tool (APT) is freely available software for the user interface; it consists of libraries to install or remove the software from the Linux distributions. It simplifies the process of installing software packages either by assembling source code or from precompiled files.

4.4. Python and supporting packages

Python is a high-level language; syntax allows the programmer to fix the concept in fewer lines of code. It has dynamic system features, memory organization, supports multiple programming models, including Object Oriented Programming functions, procedure, and substantial standard library access.Python includes the following packages to make a complete package for home automation:
Flask web server: Python-based web framework.
Requests: To send HTTP request.
Flask socket IO: To implement socket IO abstraction on the server side.
Beautiful soup 4: HTML parser (extract data from HTML or XML files).
The packages can be installed efficiently using pip installation packages. Therefore, pip installer is used to install Flask web framework and packages as shown in Figure 5.
Flask: The web Framework software is to design and develop a web application including Web Services, web resources, and Web API. It provides a standard form to build web applications and deploy it. Due to limited resources on Raspberry Pi, the most suitable framework is Flask web server. Flask is a micro level web Framework, depends on some external libraries like werkzeuy and jinja2. Where werkzeuy easy WSGI (Web Server Gateway Interface) tool, the standard variety of service to develop and deploy web apps. Python interface between web applications and jinja2 renders templates.
Requests: It is a Python HTTP library, HTTP request or easy, more straightforward and more user-friendly. Therefore, manually added query strings to URL or not required.
Socket.io: The real-time event-based bidirectional transport protocol that enables to communicate client and server.

```
(helloflask) blackpearl@blackpearl-Aspire-V5-471PG /home/python/tryflask/helloflask
 $ pip install flask

Collecting flask
  Downloading Flask-0.11.1-py2.py3-none-any.whl (80kB)
    100% |████████████████████████████████| 81kB 132kB/s
Collecting Jinja2>=2.4 (from flask)
  Downloading Jinja2-2.8-py2.py3-none-any.whl (263kB)
    100% |████████████████████████████████| 266kB 419kB/s
Collecting click>=2.0 (from flask)
  Downloading click-6.6.tar.gz (283kB)
    100% |████████████████████████████████| 286kB 190kB/s
Collecting itsdangerous>=0.21 (from flask)
  Downloading itsdangerous-0.24.tar.gz (46kB)
    100% |████████████████████████████████| 51kB 389kB/s
Collecting Werkzeug>=0.7 (from flask)
  Downloading Werkzeug-0.11.11-py2.py3-none-any.whl (306kB)
    100% |████████████████████████████████| 307kB 354kB/s
Collecting MarkupSafe (from Jinja2>=2.4->flask)
  Downloading MarkupSafe-0.23.tar.gz
Building wheels for collected packages: click, itsdangerous, MarkupSafe
  Running setup.py bdist_wheel for click ... done
  Stored in directory: /home/blackpearl/.cache/pip/wheels/b0/6d/8c/cf5ca1146e48bc79
14748bfb1dbf3a40a440b8b4f4f0d952dd
  Running setup.py bdist_wheel for itsdangerous ... done
  Stored in directory: /home/blackpearl/.cache/pip/wheels/fc/a8/66/24d655233c757e17
8d45dea2de22a04c6d92766abfb741129a
  Running setup.py bdist_wheel for MarkupSafe ... done
  Stored in directory: /home/blackpearl/.cache/pip/wheels/a3/fa/dc/0198eed9ad95489b
8a4f45d14dd5d2aee3f8984e46862c5748
Successfully built click itsdangerous MarkupSafe
Installing collected packages: MarkupSafe, Jinja2, click, itsdangerous, Werkzeug, f
lask
Successfully installed Jinja2-2.8 MarkupSafe-0.23 Werkzeug-0.11.11 click-6.6 flask-
0.11.1 itsdangerous-0.24
```

Fig. 5: PIP Installer to Install Linux Distribution Packages.

4.5. HTML and CSS

Hyper Text Mark-up Language is to create web apps and web pages with cascading sheets and JavaScript it frames a World Wide Web page. Web browsers receive HTML documents are selected from local storage or a web server and extract them into multimedia web pages. The elements are building blocks of HTML pages, which builds an image, objects and cooperative forms embedded into the rendered page. HTML programs written in a scripting language can impact the behavior and content of the web pages. CSS defines the layout and look used to state the presence of a written mark-up language document. The visual style of the web pages and the user interface written in HTML and XHTML. The language applied to XML document. The HTML, JavaScript, and CSS are used in most websites to create visual web pages, the user interface to web applications and mobile applications. CSS is mainly to separate the presentation content such as colors, fonts, and layout. The visible contents provide flexibility, control, and accessibility the specification of presentation. Multiple HTML pages to share design by denoting the relevant CSS in a separate dot CSS file to reduce the reputation and complexity in a structural contact. The graphical design of a document can be changed easily by editing few lines of code in CSS file, instead of changing markup in the documents. The CSS specifications conserved by World Wide Web Consortium (W3C). The W3C runs a free CSS validation service for CSS documents, where we used the open source CSS library Bootstrap. Bootstrap is a friendly and open source library to design websites and web applications it contains a CSS and HTML based templates for buttons, navigation, design interface components and JavaScript extension frameworks concerned with front-end development only.

4.6. Java script

JS is a high-level programming language; language is the prototype-based multi-paradigm and dynamic node JS is one of the three (HTML, CSS, and JS) core technologies of www content, makes web pages more interactive and includes video. Due to multi-paradigm language, JavaScript separates event-driven functional and object-oriented prototype based. The JavaScript files that included client-side appearance.
Bootstrap: Front-end framework to beautify website JS files.
Socket IO client: To implement the socket IO abstraction on the client side.
Chart.js: The JavaScript library is to generate cooperative, animated graphs for the web pages.

5. Hardware implementation

The hardware implementation of different sensors and actuators where LDR, LM35, smoke detector, and relay interfaced to NodeMCU. The Pi camera, PIR motion sensor interfaced to Raspberry Pi. The entire home automation system considering all rooms are connected wirelessly as shown in Figure 6 is the local internet source to connect wirelessly through the hotspot. The Raspberry Pi is a coordinator connected to the router to serve as a central node in the wireless sensor network. The Raspberry Pi interfaced with a Pi camera, a motor driver, PIR motion sensor, RGB LEDs, and Buzzer.

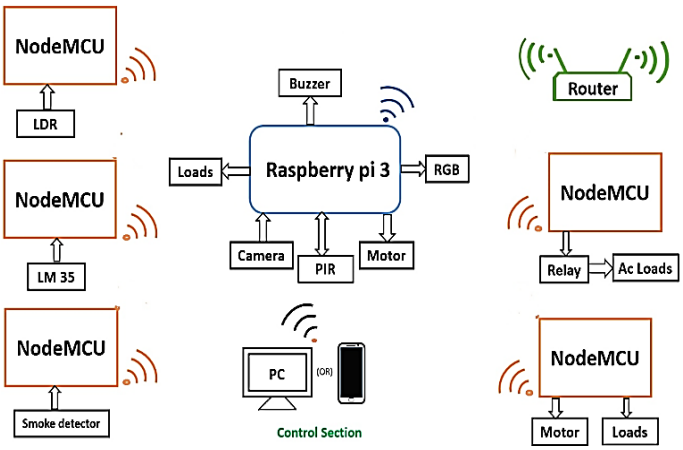

Fig. 6: Complete WSN Home Automation Using a Star Topology.

5.1. Raspberry pi

The Raspberry Pi is linked to a router through hotspot as shown in Figure 6. The interface Raspberry Pi to PIR motion sensor, Pi-camera, the motor driver circuit, and RGB LEDs.

5.2. Node MCU with LDR

The circuit as shown in Figure 7 is used to measure the light intensity and control the proportional action on closing/opening Windows. The components required are NodeMCU, LDR sensor, 10kilo-ohms resistor, Arduino IDE, and USB. The working principle of the LDR shows that as the light intensity falling on the LDR sensor, then the resistance increases for daylight five kilo-ohms and in the dark 2 Mega-ohms. The voltage drop across the LDR sensor will give an analog output to NodeMCU.

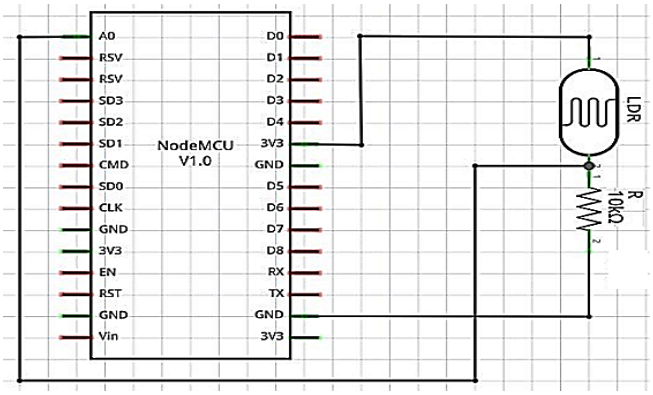

Fig. 7: Interfacing LDR to Node MCU.

The client and server side operates a program to communicate Raspberry Pi and NodeMCU.

5.3. Node MCU with LM35

As shown in Figure 8, to measure room temperature and display through the website the components in the circuit are NodeMCU and LM35 temperature sensor.

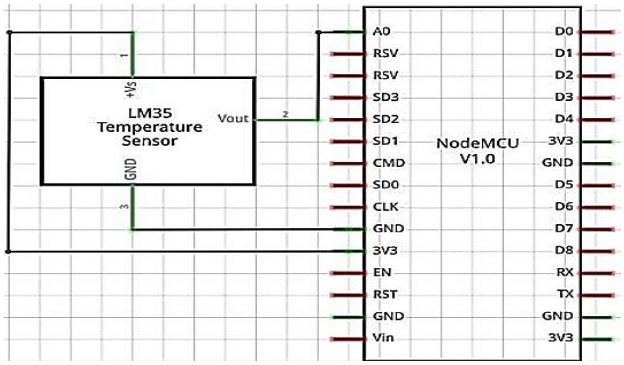

Fig. 8: Interfacing LM35 to Node MCU.

The temperature sensor is to monitor and control the AC loads rely on the circuit.

5.4. Node MCU with L293D

Change the DC voltage to control DC motor speed, the motor rotation changed by reversing the direction of the current passes through the motor driver circuit. The interfacing components are NodeMCU, motor driver L293D and DC motor as shown in Figure 9. The motor driver circuit is used to control the doors to open, the windows curtains to open and close.

5.5. Node MCU with relay

A simple circuit drawing consists of NodeMCU and relays, AC loads are controlled using the circuitry. The on and off Switching control by computing the program in NodeMCU and the AC loads are controlled to manage home appliances. The relay

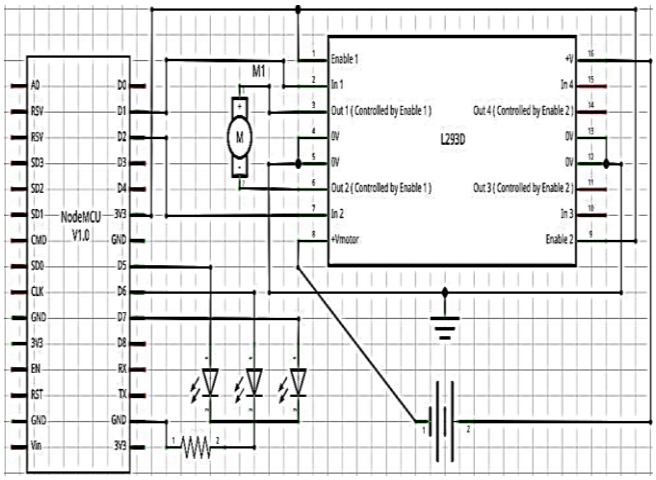

Fig. 9: Interfacing L293d to Node MCU.

to replace the manual switch, to switch on and off fans, air conditioners, and lights, etc.

6. Software implementation

6.1. Communicating data from raspberry pi to node MCU

Raspberry Pi access a client and it sends HTTP request and also have NodeMCU behaves as a server which handles application using Raspberry Pi as a coordinator, we can send control request on the website to the NodeMCU, and also we configure the Raspberry Pi automatically control NodeMCU to switch on actuators as shown in Figure 10.

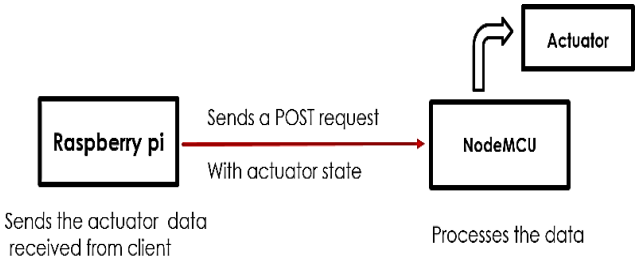

Fig. 10: Communicating Data from Raspberry Pi to Node MCU Module.

6.1.1. Raspberry pi as a server

Here flask a Web Server Gateway Interface (WSGI) debugging server is used for the site, even we can use services like apache2 by editing configuration files, the NodeMCU acts as a client and Raspberry Pi as a server, and here we can have a small snippet of flask server.

The following steps discussed to state the code as shown in Figure 11.

Step 1: From Figure 8 Line 1 to import the modules like Flask socket.io and render the template.

Step 2: Line 2 includes socket IO from flask_socketio for implementing server-side socket abstraction.

Step 3: From line 3, it is considered to declare the flask style in the web app, and here to create an instance of flask class and _name_ is a particular variable that generates a value the string "_main_".

```
CODE:
    1. from flask import Flask, render_template
    2. from flask_socketio import SocketIO
    3. app = Flask(_name_)
    4. socketio = SocketIO(app)
    5. @app.route('/')
    6. def index():
    7.  return render_template('index.html')
    8. if __name__ == '__main__':
    9.  socketio.run(app,host='0.0.0.0', debug=True
```

Fig. 11: Snippet Code for Raspberry Pi as A Server.

Step 4: line 4 is to implement a socket.io for passing the declared flask instance app to the socket.
Step 5: From fifth, sixth and seventh lines of code, we can understand the declaration of a local route server to pass the requested data.
Step 6: Sixth and seventh lines of code are defining the function and returning the data.
Step 7: Line 7 renders the template function by default is the index.html file in the project folder.
Step 8: From Line 9 code, to know the Python assigned the name "_main_" when the script got executed. To execute the script, "_name_" will be equal to the "_main_." If the conditional statement is satisfied, then socketio.run() method will withstand, the technique allows control over the behavior of the script.

6.1.2. Raspberry pi as a client

From Raspberry Pi, the HTTP sends the request to a server using a python package.
The request package sends a single statement by remembering to set all the headers. The sample snippet as shown in Figure 12.

```
@socket.on('change_relay2')
def relay2(relay2_state):
    if(relay2_state=='off':
    print('relay 2 off')
    try:
        jsonfan2_off =
requests.posr('192.168.1.10/relay2=off', data =
{'relay2':'OFF'}, headers={'connection':'close'})
    except requests.exceptions.RequestEception as e
        print(e)
    elif relqy2_state == 'on':
        print('relay 2 on')
        try:
            jsonfan2_on =
requests.posr('192.168.1.10/relay2=ON', data =
{'relay2':'ON'}, headers={'connection':'close'})
    except requests.exceptions.RequestEception as e
        print(e)
```

Fig. 12: Snippet Code for Raspberry Pi as A Client.

Step 1: Listening to a socket event change to execute some code.
Step 2: Changing the event will receive the data with a variable.
Step 3: It is to verify the data using if condition and try to act accordingly.
Step 4: By using requests.post () method, here it is posting the data to the URL with some address.
Step 5: Try method to extract exceptions.

6.2. Communicating data from node MCU to raspberry pi

To send data from the Node MCU using HTTP post request with a content type of application from encoded data and on the server side will handle this request using request.form.get. The data extracted can be used to save data to a text file or a database.

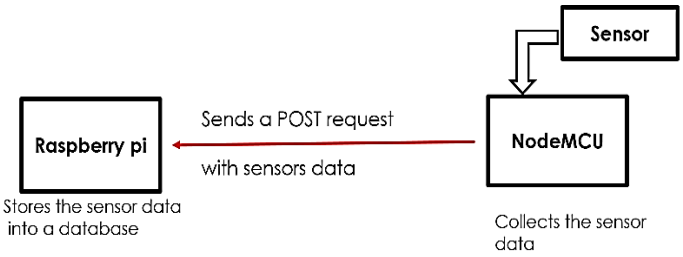

Fig. 13: Communicating Data from Node MCU to Raspberry Pi.

The following steps of a small snippet code as shown in Figure 14 are:

```
CODE:
@app.route("/temperature/postemp/", methods=['GET','POST'])
@requires_auth
def postemp():
    if reuquest.method =='GET':
        return render_template('erro.html',error="Method
            not Allowed !!!")
    elif request.method == 'POST' :
    try:
        temp_value = request.form.get('temp')
        timevalue =str(time.time())
        socketio.emit{'temp_change',{'temperataure':
            temp_value,'time':timevalue}}
            if float(temp_value) > 40:
                gpio_setup.set_rgb(2)
            elif float(temp_value)< 18:
                gpio_setup.set_rgb(0)
            else:
                gpio_setup.set_rgb91)
            line_prepender("temp.txt",timevalue+' '+temp_value)
                return(' ',204)
            except:
                return(' ',400)
```

Fig. 14: Communicating Data from the Node MCU to Raspberry Pi.

Step 1: Declare the route to accept the data and restrict the HTTP request such that setting the accepted variable methods to post the value.
Step 2: Set path for different types of request in if conditional statement.
Step 3: Get requests for sending the response as a method not allowed.
Step 4: The post request is for sending the data with a timestamp into the temporary text file.
Step 5: Sending the data to the website using socketio.emit dynamically update the site. The method takes two variables. One is an event name and a dictionary data, it is used to broadcast all connected clients.
Step 6: Finally, we are setting some GPIO pins.
Step 7: Try to handle all exceptions using Python handler.

6.2.1. Node MCU as a client

A small snippet of code, how the NodeMCU save and access the client that connects to a server is shown in Figure 15, the stepwise explanation of the quote as follows:
Step 1: From code on line one, we have declared an object to the class HTTP client, used as a header file accessed from the library.
Step 2: On Line 2 is to create a TCP connection to the IP address with port address 5000 to the route.

```
CODE:
    1. HTTPClient http
    2. http.begin(http://192.168.1.9:5000/ldr/postldr/)
    3. http.addHeader("Content_type","application/x-www-form-urlencoded
    4. http.addHeader("Authorizaton","BasiccGk6a2FseWFua3Jpc2huYQ==");
    5. int httpcode = http.POST("ldr="+ string9sensor))
    6. string payload = http.getString()
    7. Serial.println(httpCode)
    8. Serial.println(payload)
    9. http.end()

OUTPUT:
    POST /ldr/postldr/HTTP/1.1
    Host: 192.168.43.97:5000
    User-Agent: ESP8266HTTPClient
    Connection: close
    Accept-Encoding: identity; q=1, chunked; q=0.1 ; q=0
    Content-Type: application/x-www-form-urlencoded
    Authorization: BasiccGk6a2FseWFua3Jpc2huYQ==
    Content-Length: 0
    Server: Werkzeug/0.14.1 Python/3.6.3
    Date: Fri, 23 Mar 2018 07:25:47 GMT
```

Fig. 15: Snippet Code for Node MCU as A Client.

Step 3: From 3rd and 4th lines to add headers to the HTTP request and send authorization.
Step 4: From Third line "Content-Type" is significant because here we can send various types of contents like text/Plain, application/JSON. So we can name the form of the content sending to the server can decode the data.
Step 5: In Fourth line HTTP necessary authorization with username and password has to base64 encoded. Here the base64 encoded form, where username and password and both are separated using a colon.
Step 6: The Fifth line is to send HTTP post request with the data of string type, we are typecasting the float value of the sensor to string format and also collecting the return response code in HTTP code.
Step 7: The Sixth line is to collect any extra data that being sent by the server with the response code.
Step 8: In 7th and 8th lines are to print the results on the serial monitor.
Step 9: Finally we are trying to close the TCP connection using http.end () function.

6.2.2. Node MCU as a server

A small snippet of the Node MCU access as a server, which accepts the request from clients as shown in Figure 16 for the following the stepwise execution of the code explained as follows:
Step 1: Line 1 to create a server on port 80 of a Node MCU.
Step 2: From line 2 to begin the server, we're trying to create an object client to class Wi-Fi client.
Step 3: Third line the client object is not declared where starting again from the beginning.
Step 4: Fourth line, it is waiting in the while loop to get a client connection.
Step 5: Here it is trying to read the first line of HTTP request sent by the client and storing its variable.
Step 6: It is flushing all the remaining data from the HTTP request.
Step 7: From 7th and 8th lines waiting for the local pass string by using the index of the first line of HTTP request and writing the digital I/O accordingly.
Step 8: The last three lines are sending a return acknowledgment response to the client telling that sent HTTP request successfully.

```
CODE:
    1. WiFiServer server(80)
    2. WiFiClient client = serer.available();
    3. if (!client)
        return
    4. while(!client.avaiable())
        delay(1)
    5. String request = client.readStingUntil('\r')
    6. client.flush()
    7. if(request.indexof("/light = on") ! = -1)
        digitalWrite(light,LOW)
    8. if(request.indexof("/light=off")!=-1)
        digitalWrite(light,LOW)
    9. client.println("HTTP/1.0 204 NO CONTENT")
    10. client.println("Content-Type: text/html")
    11. client.println(" ")

OUTPUT:

    POST /light=on HTTP/1.1
    Host: 192.168.43.47
    User-Agent: curl/7.55.1
    Accept: '/'
    Content-Type: application/x-www-form-unlencoded

    HTTP/1.0 204 NO CONTENT
    Content-Type: text/html
```

Fig. 16: Snippet Code for Node MCU as A Server.

7. Results and discussions

7.1. Webpage

The highlighted homepage in the browser as shown in Figure 17, modules connected to a Smart home with a highlighted background colors for Door Lock & Camera, temperature sensor, LDR and window control, motion sensor relays.

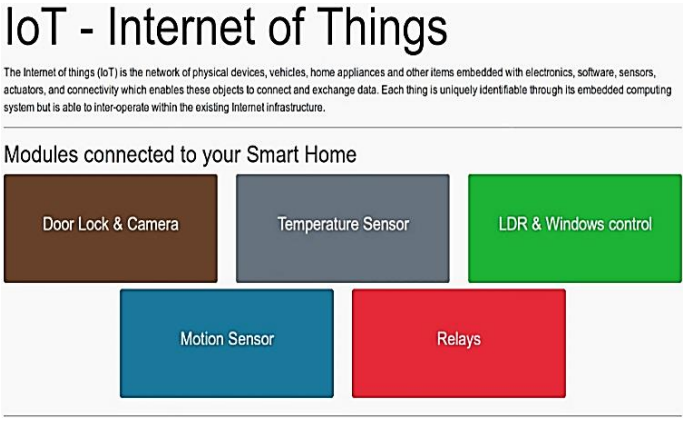

Fig. 17: Smart Home Webpage for Home Appliances.

Example the door lock and a camera with a brown background color we can state the condition by clicking on the icon. Similarly, we can have a web page access to the individual color style. The complete setup of the hardware connections made to the Raspberry Pi and node MCU modules as shown in Figure 18. The power supply is connected to the individual NodeMCU circuits separately. Connect each sensor and actuator to the respective Node MCU.

Fig. 18: The Complete Hardware Setup.

7.2. The raspberry pi

Door lock and camera, when the user tries to access this highlighted webpage blocks, it asks for the authentication as shown in Figure 19, obtained by IP address or website with the server address. Here http://localhost:5000 is sample server, which we have posted, the user has to enter the username and the password.

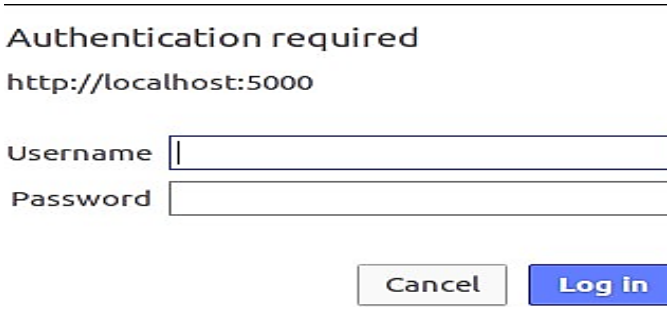

Fig. 19: User Authentication Required to Access Server.

When the user enters the correct information, then it is directed to open the web page. On this page, we can control door lock by pressing the lock and unlock options. The PIR motion sensor is connected to Raspberry Pi if it detects any changes it sends an alert message to the authorized person as shown in Figure 20 via Way2SMS protocol.

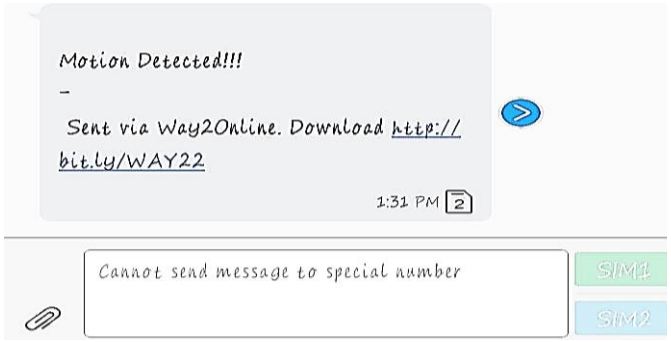

Fig. 20: SMS Received from Raspberry Pi to Mobile.

In parallel to above-said process, the camera connected to the Raspberry Pi acts and captures the pictures, the pictures are viewed on the webpage as shown in Figure 21. If the user wants to allow the person he can press unlock so that the motor connected to the Raspberry Pi will open the door.

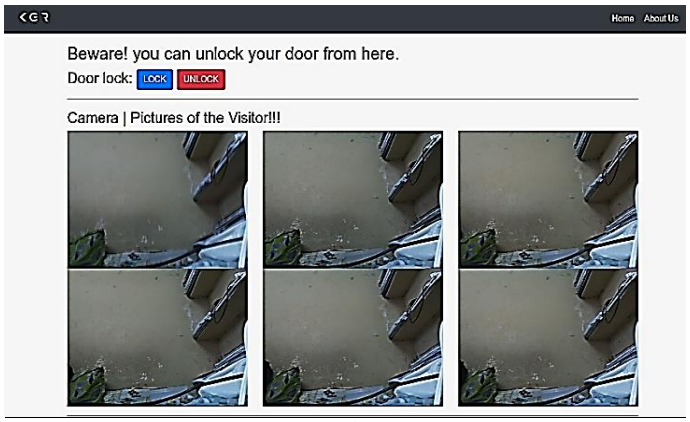

Fig. 21: Images Captured from the Front Door.

7.3. Temperature sensor

From the homepage of the website, if the user clicks on temperature sensor icon, we can have access the current temperature in thing speak web browser livelily, where it was a chart for the temperature concerning time and also current time as shown in Figure 22.

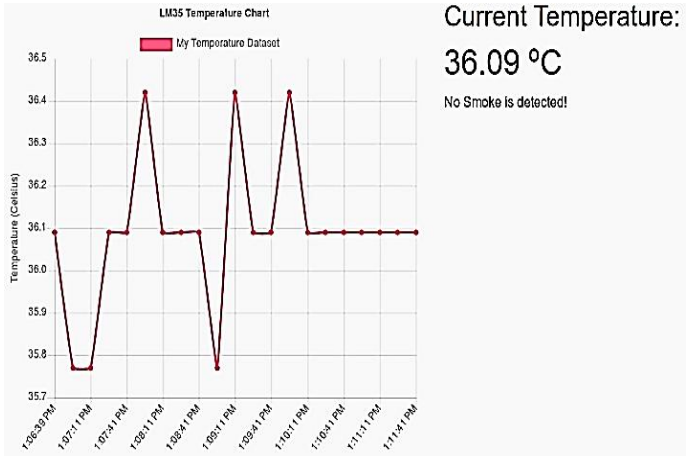

Fig. 22: Temperature vs. Time Plot in Real-Time.

The plot gives the complete status of the temperature sensor concerning date and time the hardware connections as shown in Figure 23.

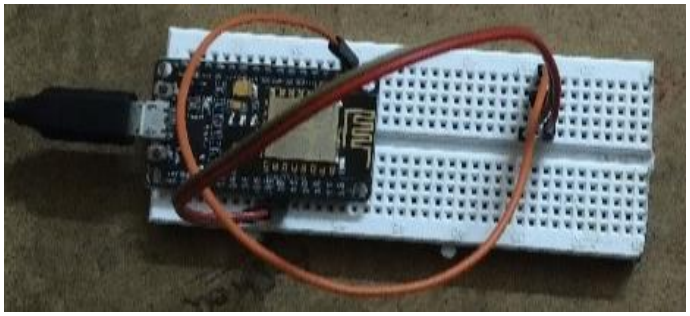

Fig. 23: Interfacing LM35 to Node MCU.

7.4. LDR and window control

From the home page of the website, if the user clicks on LDR and window control, he can view the result as shown in Figure 24. The plot shows the current luminosity concerning the date and time periodically. The hardware connections of LDR and motor driver control as shown in Figure 25.

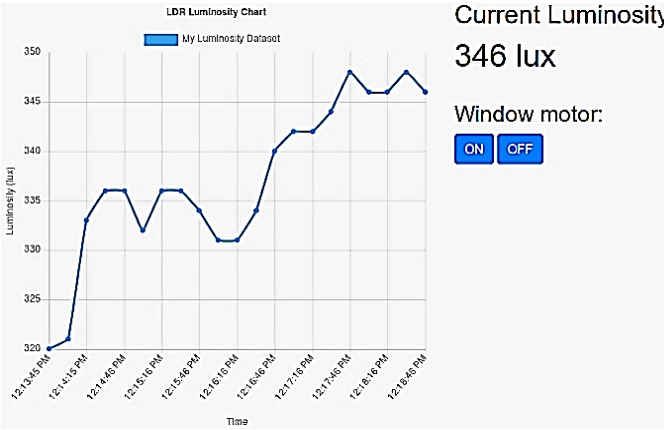

Fig. 24: Luminosity vs. Time Plot in Real Time.

Here we can control the windows using L293D motor driver board to a DC motor and by making the motor rotate in a clockwise and anticlockwise direction for a specified period to open and close the windows.

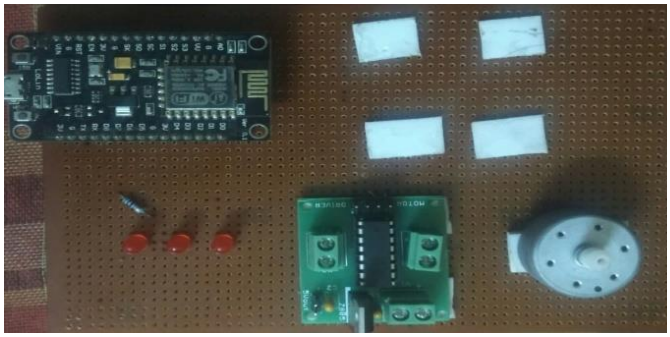

Fig. 25: Interfacing the Motor to Node MCU.

7.5. Relays

The webpage as shown in Figure 26 is to control the switches through the website. For example Light-1, Light-2, Light-3, Fan-1, Fan-2, Relay-1, and Relay-2. The change in colour on the webpage will indicate whether the switch is on or off the home appliances. We use the different buttons with the titles upon them, and these click buttons help in controlling different loads present in the home. The hardware connections for the relays to 230 volts bulb as shown in Figure 27 on one side and the other side connected to NodeMCU module.

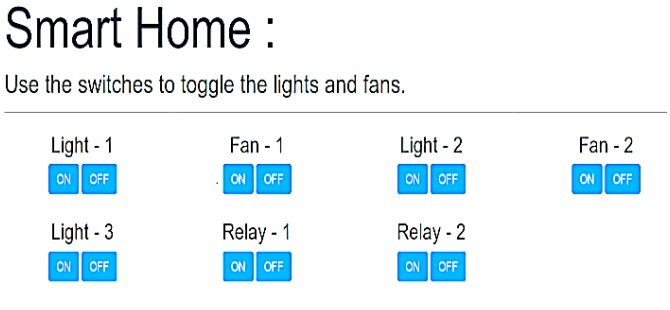

Fig. 26: Smart Home Webpage for Home Appliances.

The appliances like fan, lights and some other loads connected to the NodeMCU modules. Using NodeMCU modules for portability, because complete wiring throughout the home cannot be done just for automation. Nodes are distributed in the rooms to manage the appliances available in each room in parallel. The control of relays is done through the website so that the user can control the devices as shown in Figure 27, to mount the lamp go on and off.

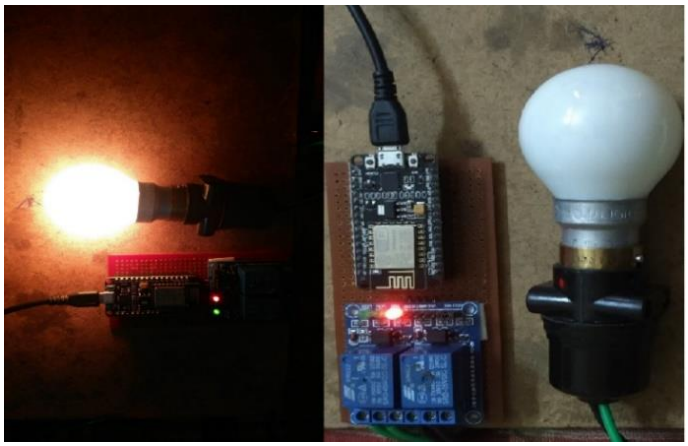

Fig. 27: Interfacing Lamp Through the Relay to Node MCU.

8. Conclusions

In this project, we proposed the design and implementation of home automation system using Wi-Fi modules and Raspberry Pi module. When the motion sensor sense and alerts the entry of a person near the door through Way2SMS, the notification sent to the user mobile. Then the user can have access to the home remotely through a web page, where he can view the entry of the trespasser, by capturing an image using the fixed camera in front of the door to the Raspberry Pi. Further, the Restful API, WebSocket, and flask server are used to integrate all the nodes and to communicate the data to the web page through the server. The webpage is used to monitor and control the appliances as all the devices are connected wirelessly to the Raspberry Pi, they are on the same network locally. Hence the security can be improved, and the type of information exchanged on the internet. The advantage of this project based on the open source using Raspberry Pi, which is so cheaper and Node MCU very cheap, to implement the star topology based Wireless Sensor Network. The project cost is very minimal so that every common man can make use of it to control the entire home appliances. The number of modules can be added or removed in the star topology without disturbing other modules, makes very easy to equip and maintain the whole project.

References

[1] R. Piyare, M. Tazil, “Bluetooth based home automation system using cell phones”, Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on 14-17 June 2011. [2]. https://doi.org/10.1109/ISCE.2011.5973811.

[2] Eurico Leite, Luis Várela, V. Fernão Pires, Filipe D. Cardoso, A. J. Pires, João F. Martins, "A ZigBee wireless domotic system with Bluetooth interface", Industrial Electronics Society IECON 2014 - 40th Annual Conference of the IEEE, pp. 2506-2511, 2014. https://doi.org/10.1109/IECON.2014.7048858.

[3] Chan H. See, Kirill V. Horoshenkov, Raed A. Abd-Alhameed, Yim Fun Hu, Simon J. Tait, "A Low Power Wireless Sensor Network for Gully Pot Monitoring in Urban Catchments", Sensors Journal IEEE, vol. 12, pp. 1545-1553, 2012, ISSN 1530-437X.

[4] Ashutosh Bhatt, Jignesh Patoliya, "Cost-effective digitization of home appliances for home automation with low-power WiFi devices", Advances in Electrical Electronics Information Communication and Bioinformatics (AEEICB) 2016 2nd International Conference on, pp. 643-648, 2016. https://doi.org/10.1109/AEEICB.2016.7538368.

[5] Libor Majer, Jozef Mihalov, Viera Stopjakova, Juraj Brenkus, Gabor Gyepes, Matej Uram, "Multi-communication wireless system for smart households", Telecommunications Forum (TELFOR) 2013 21st, pp. 272-275, 2013. https://doi.org/10.1109/TELFOR.2013.6716224.

[6] Khirod Chandra Sahoo, Umesh Chandra Pati, "IoT based intrusion detection system using PIR sensor", Recent Trends in Electronics Information & Communication Technology (RTEICT) 2017 2nd IEEE International Conference on, pp. 1641-1645, 2017.

[7] Thomas Fischer, Christian Lesjak, Andrea Hoeller, Christian Steger, "Security for building automation with hardware-based node authentication "Emerging Technologies and Factory Automation (ETFA)", 22nd IEEE International Conference, pp. 1-6, 2017. https://doi.org/10.1109/ETFA.2017.8247567.

[8] Ravi Kishore Kodali, Vishal Jain, Suvadeep Bose and Lakshmi Boppana, "IoT Based Smart Security and Home Automation System", pp. 1286 – 1289, 2016. https://doi.org/10.1109/CCAA.2016.7813916.

[9] Brundha S.M., Lakshmi P., and Santhanalakshmi S., "Home Automation in Client-Server Approach with User Notification along with Efficient Security Alerting system", pp. 596-601, 2017. https://doi.org/10.1109/SmartTechCon.2017.8358441.