Bulletin of Electrical Engineering and Informatics
Vol. 12, No. 5, October 2023, pp. 2763~2776
ISSN: 2302-9285, DOI: 10.11591/eei.v12i5.5062

An android application and speech recognition-based IoT-enabled deployment using NodeMCU for elderly individuals

Arunav Mallik Avi[1], Md. Sohel Rana[2,3], Mustakim Billah Bedar[4], Md. Asif Talukder[5]
[1]Department of Computer Science and Engineering, National University, Dhaka, Bangladesh
[2]Department of Electronics and Communication Engineering, Khulna University of Engineering and Technology, Khulna, Bangladesh
[3]Department of Electrical and Electronic Engineering, Northern University of Business and Technology Khulna, Khulna, Bangladesh
[4]Department of Computer Science and Engineering, Jahangirnagar University, Dhaka, Bangladesh
[5]Department of Computer Science, Oklahoma City University, Oklahoma, United States

Article Info

Article history:

Received Oct 22, 2022
Revised Jan 29, 2023
Accepted Feb 23, 2023

Keywords:

Android application
Control system
Google STS
Internet of things
NodeMCU

ABSTRACT

The internet of things (IoT) is the trendiest technology in the computing, Embedded and modern control systems industries. An efficient IoT-enabled home automation technique that allows users to remotely control home appliances and observe sensor data from a long distance via internet connectivity. This study identifies a smart approach to assisting any user, particularly elderly individuals, to control home appliances and monitor home weather conditions using an Android application and voice command using speech recognition. Multiple specialized microcontrollers, especially NodeMCU and other components like sensors, networks, relays, fans, lights and android devices are used to implement this proposed system. This IoT platform device is configured to recognize the instructions from the Android application to control appliances as well as compute real-time weather data from sensor and upload the sensor data stream to the IoT web server. The dedicated IoT web server, that is accessible by the Android application for further IoT operations. On the other hand, the Google Speech to Text (STS) service enables a facility to control home appliances through voice command. Mobility issues plague the elderly, complicating even the simplest tasks. From observation of this research, they can control this proposed system without ever touching any physical switches.

Corresponding Author:

Arunav Mallik Avi
Department of Computer Science and Engineering, National University
Gazipur 1704, Dhaka, Bangladesh
Email: armavicse@gmail.com

1. INTRODUCTION

In computing advancements, the modern control/automation system can be combined with computer networks and software applications so that human interaction with electrical objects like fans, lights, speakers, refrigerators, and air conditioners. is minimal. Software engineers and IT professionals are inspired to create new concepts for automation control systems [1]. In the addition of the Android operating system in software technology, it opens a new door of portable multitasking technique to solve several types of problem by using a smartphone. According to online statistics and previous research, more than 70% of people worldwide use the Android operating system [2], [3] so an efficient internet of things (IoT) automation system can be compatible with Android smartphone applications [4]. The aim of this proposed system is to help elderly individuals. The justification behind this is that the world's populations are aging as a result of declining birth rates and rising life expectancy. People over 60 are now present in greater numbers and proportions as a result of this demographic shift. In United States, the ratio of population 65+ years aged people have increased 12.4% in 2000 to 13.3% in 2011 and it will increase 21% of the population by 2040 [5]. Although developed social life style, neoteric medicine and the accessibility of modern life care increase the life expectancy of elderly people [6]. United Nation organization (UNO) reports that the average life expectancy in 1950 was 65 years, 78 years in 2010 and it will increase to 83 years in 2045. Also reported that 35% of people who were 65+ years old in 2011 has some kind of disability [7]. Although many elderly people live independently, in this situation, they require effective assistance to meet their important personal needs. Implementation of a smart home system with remote and voice control facilities will help them as personal assistance and also reduce the cost [8]–[10]. This proposed system includes some special features and user-friendly facilities for the elderly or disabled, which is effective compared to previously built systems. Compared to Arduino, NodeMCU is a very low-cost embedded and IoT platform device [11]. An IoT automation system can be built with Arduino or any other type of microcontroller, but the traditional method requires a separate Wi-Fi module to set up an IoT web server [12]. NodeMCU reduces the complexity of this mechanism in order to integrate it with an ESP8266 or ESP32 Wi-Fi microchip on its system on a chip (SoC). An Android application with a unique algorithm and code that can control or switch home appliances via a user-friendly interface or Google Speech Recognition via an IoT web server while also monitoring real-time weather activity from the DHT-11 sensor. In the traditional home appliance grid system, switches or buttons are far apart from one another; this arrangement is exceedingly inefficient for the person [13], [14], especially those who have mobility issues. So, with the help of an Android application and speech recognition, the elderly or handicapped can easily operate this NodeMCU-based IoT automation system without touching any physical switches.

According to previous research, there are several approaches to home automation. ZigBee, Xbee, Bluetooth, GSM, and ESP-32 based Wi-Fi networking technologies are used for data communications in IoT home automation systems [15]. Arduino and HC-05 are utilized for the Bluetooth based home automation system. The HC-05 Serial Protocol Bluetooth module designed to establish a transparent wireless serial connection. HC-05 Bluetooth module provides switch mode between master mode and slave mode, which means it can be used neither receive nor send data. The data transmission range is limited to approximately 10 meters [16].

Zigbee is used for building low-cost and power-efficient appliance to appliance (ATA) or IoT infrastructure, this types of system is comparatively good for the small network under 10-100 meters [17]. Products of Zigbee have been expanded and modified by suppliers, making it possible for any hobbyist to produces a real voice enabled embedded products in the market [18]. GSM or CDMA modules can be used to build the IoT-enabled smart switching grid. Attention (AT) command is used to control the Modem for transitioning data [19]. Automation was established by sending and receiving SMS between the mobile phone and GSM module. An efficient automation method, but costly in terms of SMS or text messaging charges [20].

ESP-32 is another networking module capable of establishing Wi-Fi communication. A very low-cost and power-efficient module employed with the Arduino or any kind of microcontroller. It holds an IoT web server. Any computing device with Wi-Fi or internet (data) connectivity, such as a smartphone, laptop, desktop, tablet, or personal data assistant (PDA) can communicate with the IoT web server. A simple web application is used to control home appliances through a local IP address [21], but the disadvantage is that it is an independent module and should be connected to the microcontroller separately.

Technically, accessing a web application is a bit complicated; the smart approach would be a Mobile App to handle IoT home automation without any hassle. Many hobbyists and professionals use 3rd party IoT automation softwares to control IoT automation systems, such as ThingSpeak, Blynk, and MQTT, but it has some limitations when it is highlighted in the final product for the user. The unique code-based self-produced automation software is needed for authenticity [22].

2. PROTOTYPE DESIGN PROCESS

Various microcontrollers and network modules, such as Bluetooth, GSM, ZigBee, XP, ESP-32, Arduino, Intel-Galileo as well as other technologies, can be used to build IoT-based systems, but the performance, cost, and efficiency are varied, and our objective was to implement this system at a low cost. The system architecture of the prototype consists of 2 independent units: the home appliance control unit and the weather observation unit. Instrument, software tool selection, and circuit design or assembling are the phases followed by the prototype design process.

2.1. Component and software tool selection

Table 1 depicts the hardware components and software tools applied to create a workable prototype. The NodeMCU, sensors and other components are used for in the hardware section. We applied a CAD tool for designing circuit diagrams. In the software section, an API testing tool is used to manipulate and test HTTP services. Additionally, two types of IDEs are used: one for creating Android applications and another for writing microcontroller instructions. As well as the programming languages and software architectural style.

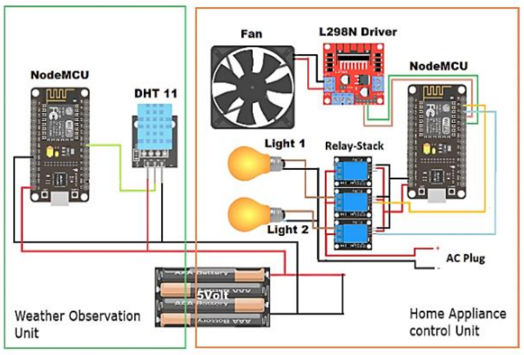
Table 1. Prototype building materials

No.	Name	Application
1	Node MCU	IoT platform device, the processing unit of the proposed prototype.
2	DHT-11	Sensing temperature and relative humidity of weather.
3	L298N	A dual H-bridge circuit controls the speed and direction of the motor.
4	Electrical appliances	Light emitting diodes, 20w (Watt) LED bulb, Atom Berg and ordinary ceiling fan.
5	Relay device	A programmable switch to control ON/OFF for high-voltage electrical devices.
6	Android SDK	Software development kit used to develop and debug Android applications integrated with set of libraries, debugger and emulator.
7	Android studio	Integrated development environment tool, used for developing Android applications.
8	Arduino IDE	A software that used to write and upload code to the Arduino boards.
9	Android smartphone	A smartphone that runs Android operating system (OS).
10	Fritzing	Computer aided design (CAD) tool for designing electronic circuit.
11	RESTful API	Representational state transfer and application programming interface are used in HTTP requests to access and manipulate data between web services and Android applications.
12	Embedded C	A programming language for writing instructions in electronic gadgets or embedded systems.
13	Java	Native compiled language for building Android application for the proposed system.
14	Postman	API platform tool, that allows developers to easily built, share, test, and document APIs.

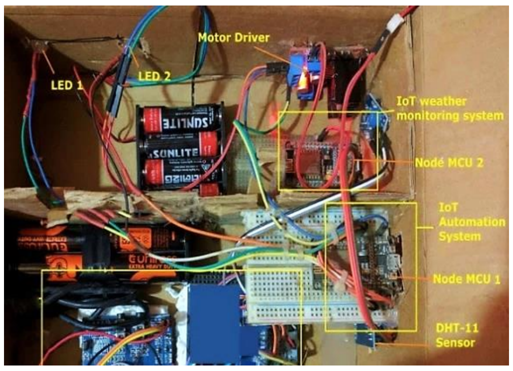
2.2. Circuit assembling

Home appliances like light, fan (for safety measures, we used a DC fan for prototyping). are connected with relay device. Relay device is directly connected with NodeMCU’s GPIO Pin header. Figure 1(a) is the fritzing diagram of the circuit, depicting the blue block as a weather observation and the red block as an automation unit. A DHT-11 sensor is directly connected to the NodeMCU through the D5 (GPIO14) pin. The power source can be provided by an external 3-5V battery through NodeMCU’s 3V pin header.

Two LED lights are connected to the NodeMCU's D1 (GPIO5) and D2 (GPIO4) pins at the home appliance control unit. The L298N motor driver's speed and rotation signal inputs are connected to D3 (GPIO0), D4 (GPIO2) pins, respectively and the fan is connected to the L298N motor driver's output pin. The L298N motor driver requires 5 volts external power source [23]. After the completion of the circuit assembly process, we obtained the actual prototype depicted in Figure 1(b).



(a)



(b)

Figure 1. Depicting the design of; (a) circuit diagram, and (b) actual view of prototype

3. METHOD

This proposed system is consisting of 2 vital unit. The tasks of automation are accomplished by the first unit besides weather monitoring are done by second unit. We applied some methodologies and techniques to make this proposed system more efficient. At the beginning, we design a suitable framework so that we can observe how the data are flown by the entire system secondly the algorithm design. After the algorithm design, there are various methods and techniques that are used such as switching grid technique, sensors data manipulation with microcontroller, IoT automation, software development for prototype and Android application.

3.1. Architecture building methodology for the proposed system

There are several types of facilities that exist in the proposed system. The data communication, infrastructure and the flow of data under the system can be depicted through a framework/architecture, which is shown in Figure 2. The development of architecture can be expressed mathematically from (1), End to End communication is equipped for data layers used for IoT gateway and the cloud instant database can be expressed from (2) and (3).

$$\sum_{m=1}^{n} f(\propto_c) = \sum_{m=1}^{n}\sum_{i=1}^{m}(\beta_{zi}) + \sum_{m=1}^{n}\sum_{i=1}^{m}(\mu_{zi}) + \sum_{k=1}^{n}\sum_{i=1}^{m}(\theta_{zi}) \tag{1}$$

Where, $\sum_{m=1}^{n} f(\propto_c)$ denoting the summations of functionalities used in the facilities of IoT's system architecture; $\sum_{m=1}^{n}\sum_{i=1}^{m}(\beta_{zi})$ denoting the various components like DHT-11 sensors, relay device, light, fan, AC and co-Microcontrollers; $\sum_{m=1}^{n}\sum_{i=1}^{m}(\mu_{zi})$ denoting the various network technologies like Wi-Fi, Bluetooth, radio transmission, GSM/CDMA, long term evaluation (LTE), and 5G; $\sum_{k=1}^{n}\sum_{i=1}^{m}(\theta_{zi})$ denoting the programming paradigm and software architectural design patterns used for data communications between human to machine interactions (such as embedded C++, Java, HTTP, Restful API, Android studio, Arduino IDE's serial monitor, and serial plotter. Figure 2 depicting a framework that represents the data communication and integral mechanism between NodeMCU's IoT cloud platform and Android device.

$$\propto_G = \{A_1, A_2, N_3, \dots, N_z\} \tag{2}$$

$$\begin{aligned} A_1(\sigma) &= \{A_{11}, A_{12}A_{13} \dots, A_{1i}\} \\ A_2(\sigma) &= \{A_{21}, A_{22}, A_{23}, \dots, A_{2i}\} \\ xA &= \{x_1A_1 + x_2A_2 + x_3A_3 \dots, x_nA_{zm}\} \end{aligned} \tag{3}$$

Where,
$\propto_G$=coefficient of IoT gateway
σ=the coefficient of protocols applied in the architecture of proposed system (such as HTTP, MQTT, AMPQ)
A=number of devices or systems, which are connected with web service
x=HTTP requests and responses from protocol (such as GET, POST, PUT)

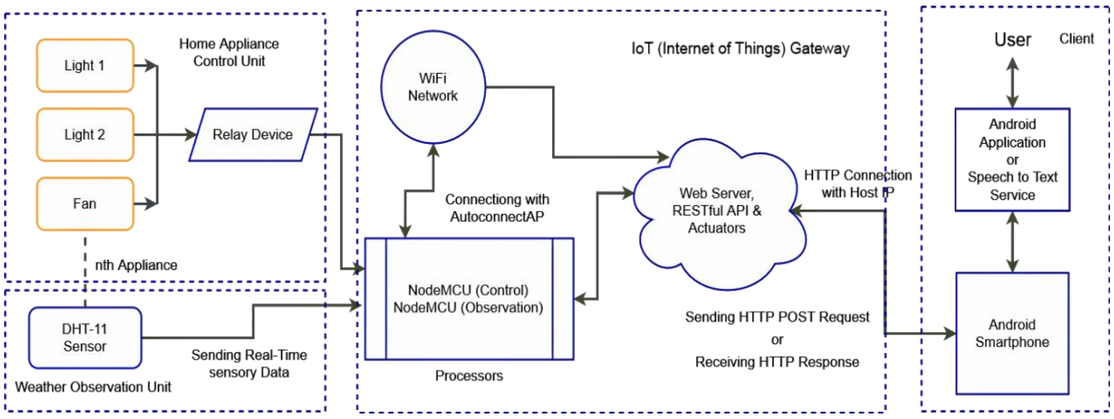
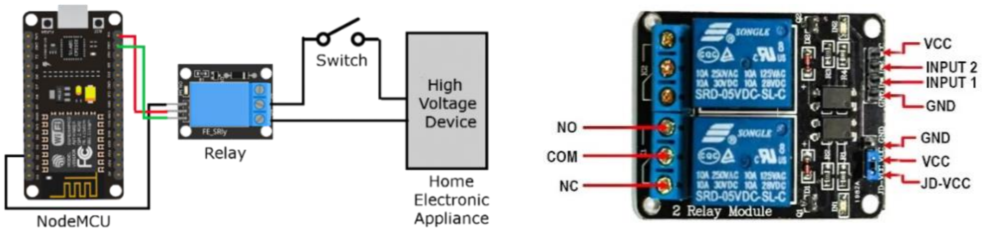

Light 1
Light 2
Fan
Home Appliance Control Unit
Relay Device
nth Appliance
DHT-11 Sensor
Sending Real-Time sensory Data
Weather Observation Unit
WiFi Network
IoT (Internet of Things) Gateway
Connectiong with AutoconnectAP
NodeMCU (Control) NodeMCU (Observation)
Processors
Web Server, RESTful API & Actuators
HTTP Connection with Host IP
Sending HTTP POST Request or Receiving HTTP Response
User
Client
Android Application or Speech to Text Service
Android Smartphone

Figure 2. System framework

3.2. Relay switching technique

A relay device works as a switch for high voltage electrical appliances. The methodology of a relay device is electromagnetism; a switch operated by an electromagnet. Therefore, a low voltage is sufficient to activate an electromagnet. This small voltage is supplied to the relay by the NodeMCU or with an intermediate driver if needed. In 5 V relay device, it has 6 pins out, 3 are used to drive the relay (low voltage side). Figure 3(a) depicts how an appliance can be connected through a relay module and the serial communication between NodeMCU and relay module.

Figure 3(b) shows the demonstration of the relay pin out's scheme. Common (COM) is the common terminal, this terminal is connected to one of the other 2 terminals (NO or NC) depending on the state of the relay. Normally open (NO) is normally an open terminal, e.g. if the relay is not energized (not lit), this pin is open.



(a) (b)

Figure 3. Connection between (a) NodeMCU and relay module; depiction of (b) Rrelay module pinouts

We can say that the switch is OFF by default and when the relay is energized it turns on. Normally closed (NC) it is a NC terminal. When the relay is de-energized (not ON), this pin is closed. The switch is activated by default and when the relay is energized. GND is ground pin. Signal is data pin to apply instructions on the relay. 5 V VCC is operating voltage of relay device [24].

3.3. DHT-11 real-time weather data computation through NodeMCU

In the weather observation unit, a DHT-11 sensor connected to the NodeMCU's GPIO pin header, and this sensor is actually made with ceramic. It holds a capacitive humidity sensor element and a thermistor. For sensing humidity, it uses a capacitor, which has 2 electrodes and a moisturizing substrate [25]. The levels of humidity are changes based on the changes of capacitance value. To measure temperature, the sensor uses a negative temperature coefficient thermistor, which causes its resistance value to decrease with increasing temperature. Higher resistance values are achieved even with small temperature changes. The NodeMCU's one wire communication protocol must be utilized to read data from the DHT-11. Figure 4 depicting the diagram of interfacing of the DHT-11, often available on its datasheet, and also the same diagram is provided below for reference. This diagram explains thoroughly the way to perform the aforementioned task.

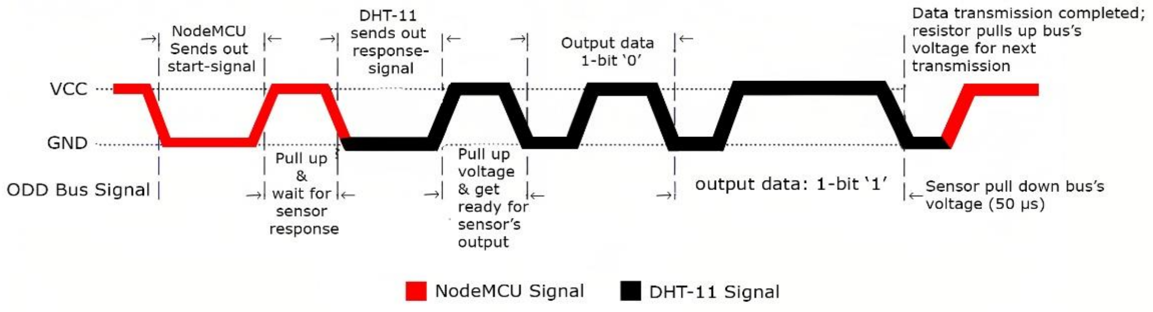

NodeMCU Sends out start-signal
DHT-11 sends out response-signal
Output data 1-bit '0'
Data transmission completed; resistor pulls up bus's voltage for next transmission
VCC
GND
ODD Bus Signal
Pull up & wait for sensor response
Pull up voltage & get ready for sensor's output
output data: 1-bit '1'
Sensor pull down bus's voltage (50 µs)
🟥 NodeMCU Signal
■ DHT-11 Signal

Figure 4. DHT-11 interfacing with NodeMCU

In order for the communication to start, DHT-11 must receive a start signal from the NodeMCU. Therefore, any time the NodeMCU wants to send a start signal to request the DHT-11 sensor to send temperature and humidity values, after the successful completion of the start-signal, the DHT-11 will transmit a response signal that contains information regarding the temperature and humidity. During this case, data communication is performed using an odd-bus data communication protocol. The full length of the data is 40 bits, and also the sensor transmits the upper data bits first. Reason for the pull-up resistor, the data or information line in idle mode always remains at the VCC level. The NodeMCU must reduce this voltage at intervals of at least 18 milliseconds to reduce overvoltage. During this period, the DHT-11 sensor detects a start signal, and the NodeMCU increments the data line by 20-40µs. Waiting time for a response from the DHT-11 is 20-40 µs. After waiting, DHT11 sends data to the microcontroller. The numbers in the data are both decimal and integral numbers. The sensor uses this data format.

"8bit integral RH data +8bit decimal RH data +8bit integral T data +8bit decimal T data +8bit checksum". Verify the checksum value with the received data. After successful data transmission, the checksum should be the sum of "8bit integral RH data +8bit decimal RH data +8bit integral T data+8bit decimal T data" [26].

3.4. Approaches to Android application development

We used several types of software development tools, the design pattern, and the Java programming language for building the "ARM Robolox" Android application. Model view controller (MVC), model view presenter (MVP), model view-view model (MVVM) are the core software architecture methodology or design patterns for Android application development. With these patterns, the project is structured in a way that all the codes can be covered in the unit testing, software code maintainability, addition and removal of features, and tracking of different important logic components. We used the MVP design pattern, because this pattern is broadly accepted and the programmer friendly to apply their own business logic. Our Android application development lifecycle can be illustrated with MVP. Figure 5 depicts the architecture of MVP [27], [28], there are:

a. Model: a layer, that is responsible for handling the real-life business logic as well as communicating with the database and network layers. Here we created model classes for storing the object of DHT-11's sensor data, HTTP Post request objects for automation unit and SQlite database objects for storing defense info.
b. View: it represents the Android application's user interface (UI) in its extensible markup language (XML) blueprint. This layer visualizes the data and keeps track of the user's actions so the presenter is notified.
c. Presenter: this layer is used to display the UI logic based on the data from the model. For example, retrieving or reading DHT-11 sensor data as a JSON object from NodeMCU's Web server, sending data to NodeMCU's web server via Android application's web service and executes the Google RecognizerIntent.

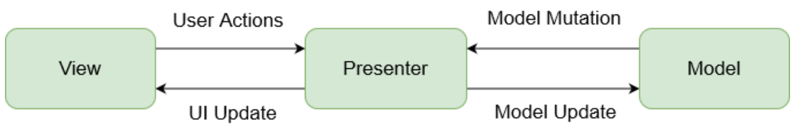


Figure 5. Architecture of MVP design pattern

3.5. Working procedure of the proposed system

NodeMCU has a useful library called WiFiManger, that allows users to connect ESP8266 to different access points (AP) without defining the Wi-Fi SSID and password in code. Also, it can store variables (custom parameters) or static internet protocol (IP) and maintains multiple SSID connections through WiFiManager library. The ESP8266 automatically connects to a known network or sets up an access point that the user can use to configure their network credentials. AutoConnectAP is the function of WiFiManager, that manages to establish a new connection with NodeMCU's Web server. This AutoConnectAP function is accessible from 192.168.4.1 default IP address. After booting NodeMCU, we will access AutoConnectAP from "ARM Robolox" Android application, then input the SSID and password of the Wi-Fi network as well as set a Static IP address.

192.168.1.152 is the static IP we set for test purposes that can be utilized by a URL; this IP address is used for data communications between the Android application and home appliance control unit. The Android application uses shared preference to store the static IP for later use. The proposed system employs two NodeMCUs, one for home appliance control unit and another for the weather observation unit. The procedure of data communication in the home appliance control unit is that when the Android application (client) sends a status via HTTP POST request through static IP/URL then NodeMCU decides to take a decision as to whether appliances should be in an ON or OFF state. For an example, the static IP/URL of NodeMCU's web server is http://192.168.1.152 and the Endpoint is /room_light. When the user switches ON/OFF or says "room light on" or "room light off" by Google RecognizerIntent from Android application, it will POST "1" or "0" as a Status in the NodeMCU's webserver. The NodeMCU will read this status. If the status is "1", then the light (appliance) will be turned on and if it is "0", the light (appliance) will be turned off. The HTTP Response from NodeMCU's web server will indicate the status being sent from the Android application. This procedure can be applied to any appliance which has been connected to NodeMCU. The logic can be described mathematically. The equation of the data transaction between Android application and home appliance control unit is,

$$p + q = \propto \tag{4}$$

where,
p=HTTP POST status; r=HTTP Response; $\propto$=state of appliance
if p=1, r=1 or p=0, r=0 then applying values in (4)

$1 + 1 = \propto OR\ 0 + 0 = \propto$
$\propto = 1\ OR \propto = 0$ here, 1 and 0 are denoted as ON and OFF states of home appliances.

The procedure for the Wi-Fi network configuration in weather observation unit will be the same as in the home appliance control unit. Although this system is using a separate NodeMCU and a separate static IP. The static IP will be different from the previous unit. 192.168.55.78 is the accessible static IP address configured by AutoConnectAP. The DHT-11 sensor measures the temperature and humidity at the home and sends real-time sensory data to NodeMCU. The DHT-11 sensory data is generated in JSON format and uploaded to the webserver. The JSON skeleton is looks like {"temperature": "29", "kelvin": "302", "fahranheit": "84", "heatindex": "37", "humidity": "90"}. The pattern of the URL looks like http://192.168.55.78 and the Endpoint is "/data". The Android application will read those JSON objects from the http://192.168.55.78/data URL and displays them in its UI (user interface), the thread (background process) of the "ARM Robolox" Android application will update the user interface (real-time temperature, humidity, heat-index data) every 1 second. A user can set a threshold value in the Android application. If the threshold value is lower than the current temperature, it will give a potential alarm to the user.

3.6. Develop algorithms for system programming

The algorithm of proposed system consists of two units connected to the same network and configured by AutoConfigAP, as shown in Figure 6 flowchart. The instructions of NodeMCU are written in embedded C language. Algorithms followed the functional programming paradigm that contains "preparation block" and "execution block". The "preparation block" describes the type of NodeMCU-connected hardwires or components and their pinouts. On the other hand, the "execution block" contains the logic or instructions of NodeMCU. Arduino IDE compiles and uploads instructions in NodeMCU. Source code is available in the GitHub repository attached in "source code availability" section. In Algorithm 1, step 1 and 2 describes the preparation block and step 3 to step 12 describe execution block of home appliance control unit. Algorithm 2, step 2 describes the preparation block and step 3 to step 9 describe the execution block.

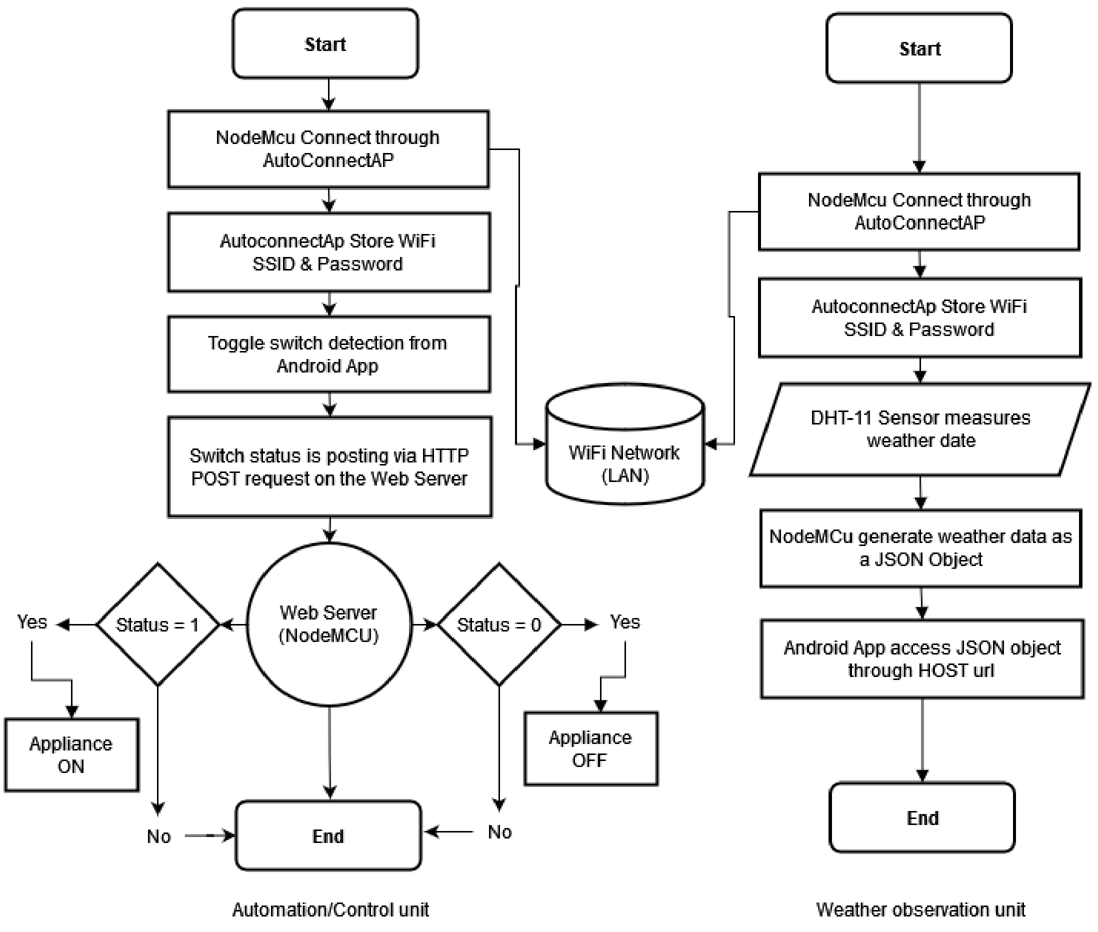

Start
NodeMcu Connect through AutoConnectAP
AutoconnectAp Store WiFi SSID & Password
Toggle switch detection from Android App
Switch status is posting via HTTP POST request on the Web Server
Web Server (NodeMCU)
Status = 1
Status = 0
Yes
No
Appliance ON
Appliance OFF
End
WiFi Network (LAN)
Start
NodeMcu Connect through AutoConnectAP
AutoconnectAp Store WiFi SSID & Password
DHT-11 Sensor measures weather date
NodeMCu generate weather data as a JSON Object
Android App access JSON object through HOST url
End
Automation/Control unit
Weather observation unit

Figure 6. System's flowchart

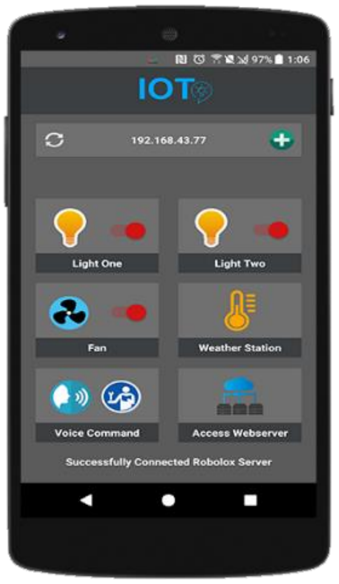
```
Algorithm 1: Home appliance control unit
Input: HTTP Post request from Android application's Web Service.
Output: Appliance will be turned On and Off based on HTTP Responses
1.  Initialize Relay_Pin = D2 //Relay module pin header assignment
2.  Initialize RELAY_APPLIANCE_State = 0 //Relay STATE count
3.  wifiManeger.autoConnectAP() //Wi-Fi network configuration
4.  If Server.available() //NodeMCU's server successfully established
5.  Else repeat Step 3
6.  If (HTTP_POST.status() == 0) return HTTP_Response = 0 //http response
7.  RELAY_APPLIANCE_State = 0
8.  analogWrite(Relay_Pin, RELAY_APPLIANCE_State) //Appliance is turned Off
9.  If (HTTP_POST.status() == 1) return HTTP_Response = 1 //http response
10. RELAY_APPLIANCE_State = 1
11. analogWrite(Relay_Pin, RELAY_APPLIANCE_State) //Appliance is turned On
12. Else jump to Step 3
13. End
```

```
Algorithm 2: Weather observation unit
Input: NodeMCU read DHT-11 real-time weather data output
Output: HTTP Response display DHT-11 weather data output as a JSON format
1. Initialize DHT11_Pin = D5 //DHT-11 pin header assignment
2. wifiManeger.autoConnectAP() //Wi-Fi network configuration
3. If Server.available() //NodeMCU's server successfully established
4. Else repeat Step 2
5. digitalWrite(DHT11_Pin, LOW) //NodeMCU read data from DHT-11
6. DHT.readTemperature() & DHT.readHumidity() //NodeMCU will read DHT-11 input
7. wifiClient.print(temperature) & wifiClient.print(humidity) //outputting sensor data in JSON stream
8. End
```

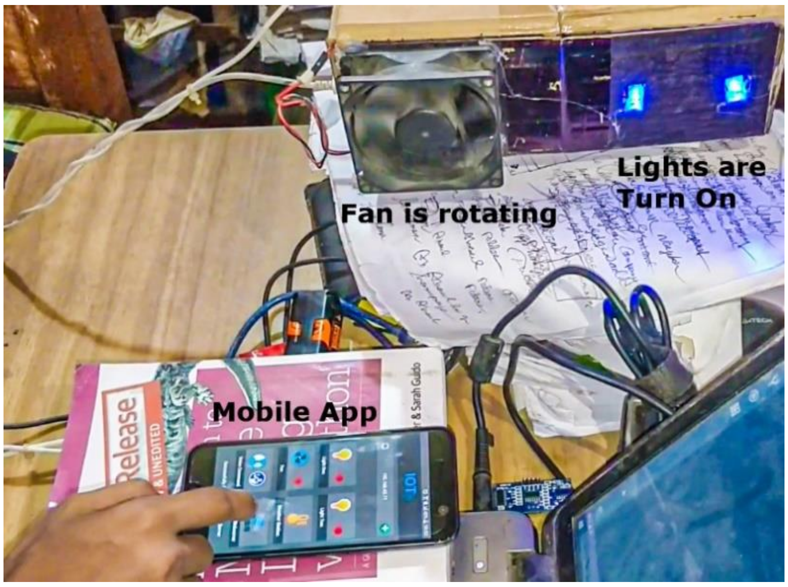
3.7. Prototype testing

3.7.1. Controlling home appliances through the "ARM Robolox" Android application

Before prototype testing it's necessary to configure network connection. We can access AutoConnectAp from browser through 192.168.4.1 URL, then input Wi-Fi SSID and password. From Figure 7(a), open home appliance control activity from "ARM Robolox" Android app then press (+) button to input static IP. Press the "Start" button located in the left corner of the (+) button to activate the Android web service. Turn on all switches in the app's user interface. Figure 7(b) shows the lights and fan turned on.



(a)

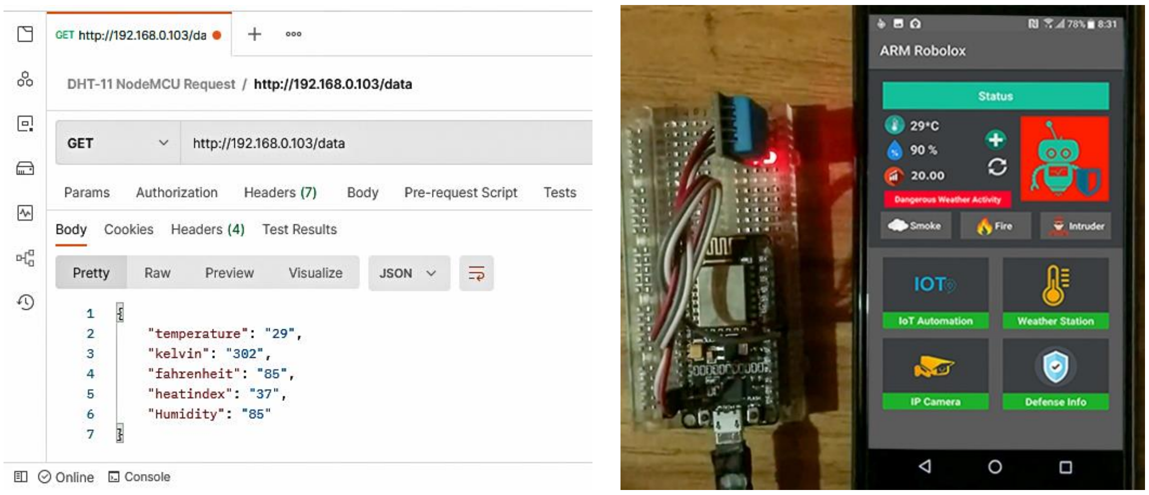


(b)

Figure 7. IoT automation UI in: (a) ARM Robolox Android application and (b) the automation process of home appliances through app

3.7.2. Displaying DHT-11 weather data through the "ARM Robolox" Android application

Check whether the DHT-11 sensor is properly connected to the NodeMCU, then open the “ARM Robolox” Android app and the postman API platform tool. In Figure 8(a), postman makes HTTP POST request in 192.16.0.103/data URL. Verify that the HTTP response is a valid JSON array that contains the DHT-11 weather information like temperature, kelvin, Fahrenheit, humidity, and heat-index. From Figure 8(b), press the "+" button to input static IP and threshold value to start the Android Web service. Check threshold value is less than the current temperature. The temperature and humidity values are displayed in the user interface of the “ARM Robolox” Android application, as well as the alarming activity if the threshold value is less than the current temperature.



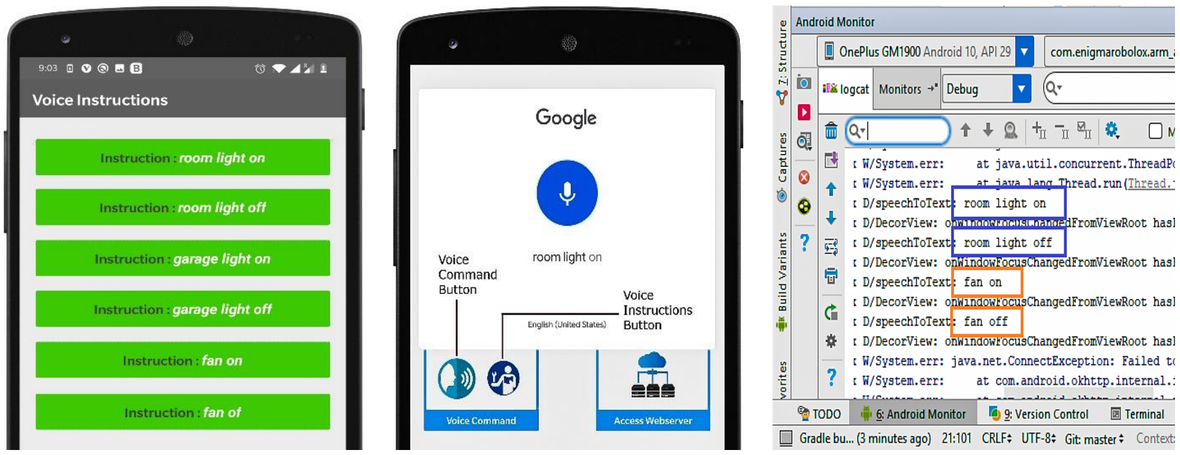
```
{
    "temperature": "29",
    "kelvin": "302",
    "fahrenheit": "85",
    "heatindex": "37",
    "Humidity": "85"
}
```

(a) (b)

Figure 8. Weather observation testing (a) post API platform tool and (b) weather monitoring UI of “ARM Robolox” APP

3.7.3. Testing speech recognition by RecoconizerIntent service

Check whether the Android device has a supported microphone or not. In Figure 9(a), the "voice instructions" UI displays the predefined commands for the appliance control. Figure 9(b) depicts the appearance of a prompt dialog bar on the RecognizerIntent activity, instructing the user to speak predefined command. Figure 9(c) depicting the Logcat from Android studio. The text that was found to be converted from the user’s speech through RecognizerIntent is seen in the Logcat Console [29].



```
W/System.err:     at java.util.concurrent.ThreadP
W/System.err:     at java.lang.Thread.run(Thread.
D/speechToText: room light on
D/DecorView: onWindowFocusChangedFromViewRoot hasl
D/speechToText: room light off
D/DecorView: onWindowFocusChangedFromViewRoot hasl
D/speechToText: fan on
D/DecorView: onWindowFocusChangedFromViewRoot hasl
D/speechToText: fan off
D/DecorView: onWindowFocusChangedFromViewRoot hasl
W/System.err: java.net.ConnectException: Failed to
W/System.err:     at com.android.okhttp.internal.
```

(a) (b) (c)

Figure 9. User interface of (a) voice instructions list, (b) RecognizerIntent activity and (c) Android studio’s LogCat console

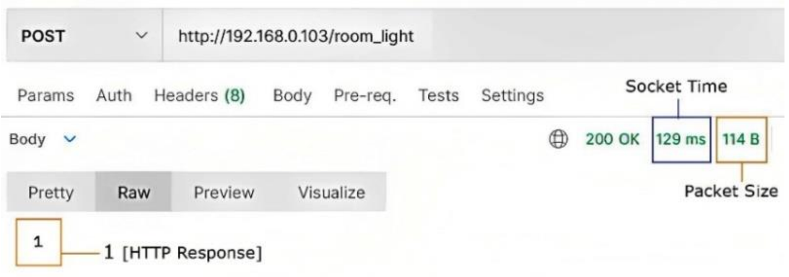
4. RESULTS AND DISCUSSION

Results can be obtained by analyzing two factors. The first one is the data transaction analysis secondly the DHT-11 real-time weather data observation. In the data transaction analysis, the web service of the "ARM Robolox" Android application will post data to NodeMCU's web server. It is our goal to measure the size, type, and value of the data; furthermore, how much time each piece of data needs to complete the transaction. In contrast, we will simulate, compare, and create the datasets based on the real-time data from the DHT-11 sensor.

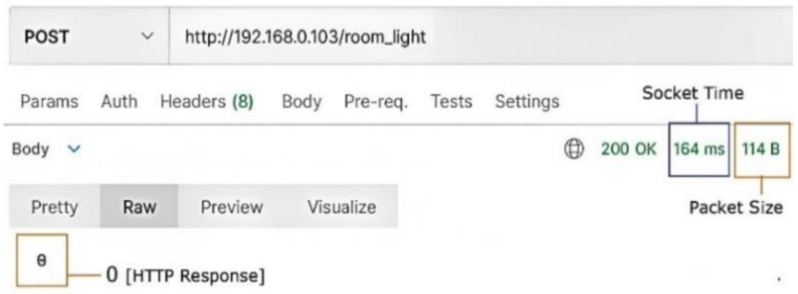
4.1. Data transaction analysis between the home appliance control unit and the Android application

The data transactions of the home appliance control unit are analyzed by the postman API platform tool. If a user wants to control an appliance, the web service of "ARM Robolox" Android application will send a HTTP POST request to the web server, and it will return "1" or "0" as a HTTP response. Each value is used to decide the action that will be applied to the appliance.

In Figure 10(a), HTTP Response 1 indicates that the appliance is turned on, whereas Figure 10(b) depicts HTTP response 0, which implies the appliance is turned off. The HTTP socket time and packet size can be analyzed from PostMan API platform tool. The socket time represents how long it takes an HTTP response to complete the socket as well as the size of the data packet received by the "ARM Robolox" Android application from an HTTP response. The packet size and socket size can be measured during data transactions, those are already depicting in Figures 10(a) and (b) respectively. Table 2 represents the data transactions between the home appliance control unit and the "ARM Robolox" Android application.



(a)



(b)

Figure 10. Data transaction, time and size measurement (a) "1" denoted appliance is ON mode whereas and (b) "0" denoted OFF mode

Table 2. Data transaction results

URL endpoint	Packet size (bytes)	Socket time (milliseconds)	Voice command	HTTP POST P	HTTP response r	Decision $\propto = p + r$	State of appliance
room_light/	114	129	room light on	1	1	1	ON
room_light/	114	164	room light off	0	0	0	OFF
garage_light/	114	65	garage light on	1	1	1	ON
garage_light/	115	181	garage light off	0	0	0	OFF
fan/	114	133	fan on	1	1	1	ON
fan/	114	157	fan off	0	0	0	OFF

4.2. DHT-11 real-time sensory data observation

The DHT-11 real-time sensory data observation testbench or result has three criteria. The first criterion is to visualize the DHT-11 data from the Arduino IDE's serial plotter. The second one is to collect the dataset from the Arduino IDE's serial monitor to check accuracy and the third one is to compare the real-time temperature value of the DHT-11 sensor with the user-defined threshold value of the Android application. Serial plotter generates plots based on the DHT-11 real-time sensory data.

Figure 11 depicts the differences in the rising and falling actions of humidity and temperature values based on the weather conditions at home; those are collected from several dates. The "blue plot" and the "red plot" visualize humidity and temperature, respectively. The range of temperature in DHT-11 is, 0 to 50 degrees Celsius, and the accuracy is + -2 degrees; the range of humidity is 20% to 80% with a 5% accuracy [30]. In Figure 12, serial monitor displays the different datasets of temperature and humidity from the DHT-11 sensor at several timestamps, and the red blocks depict the data we extracted for accuracy level analysis. At the beginning, we input 31.00 as a threshold value in the "ARM Robolox" Android application to compare it with the current value of temperature of the DHT-11 sensor.

If the threshold value of the Android application is<(less) or==(logical equal) to the current temperature, the "ARM Robolox" Android application will give an alarm to the user. Table 3 depicts the comparison rules between the current temperature and the threshold value. This test is based on the dataset of DHT-11 sensor during the summer weather in Bangladesh. The maximum temperature in Bangladesh during summer is approximately 32 to 34 degrees Celsius, and the falling temperature is approximately 27 to 28 degrees Celsius [31] so for the prototype testing purpose, we set the threshold value at 31.00 based on the peak temperature in the datasets. This threshold value can be varied depending on the usability of different users.

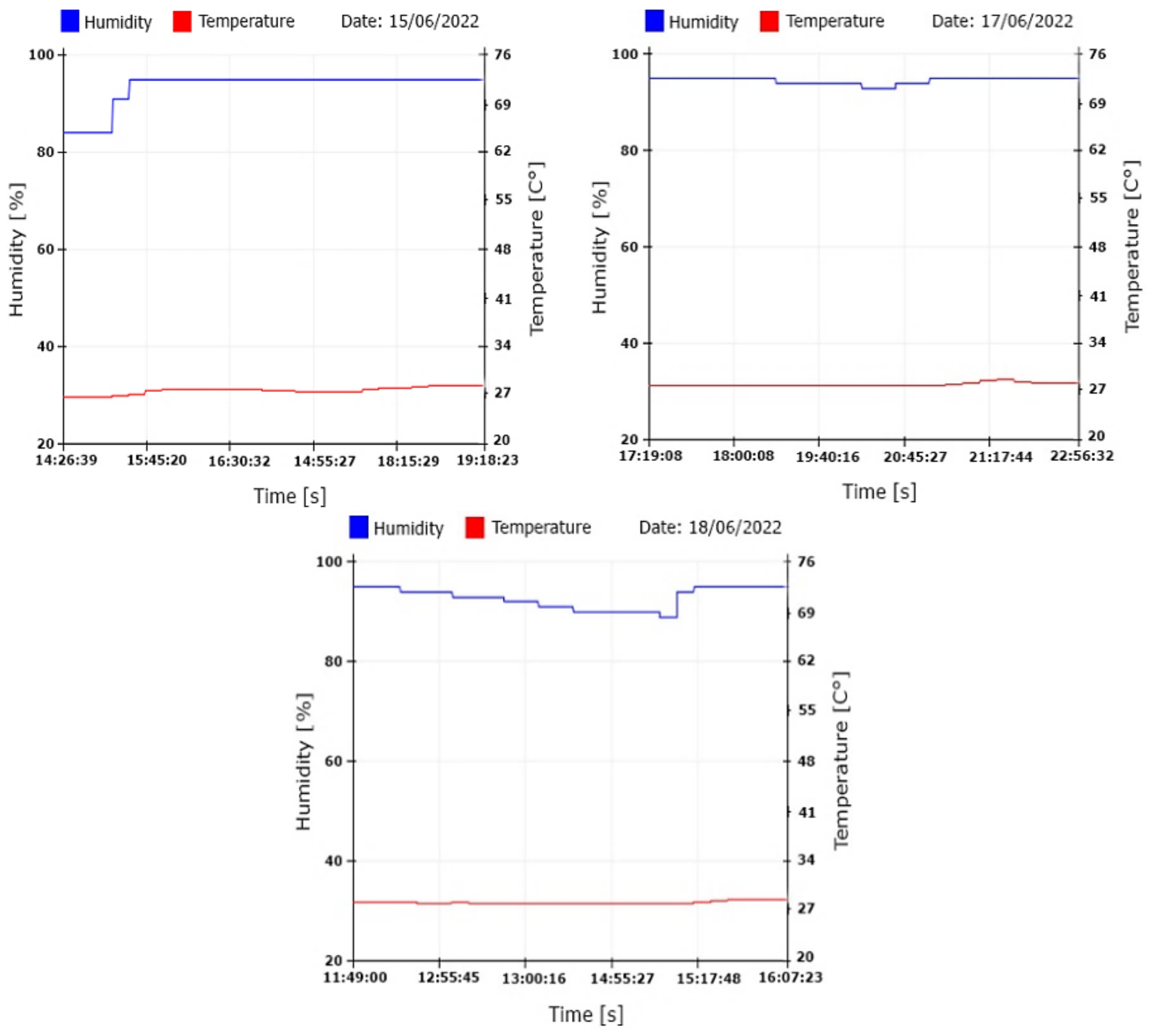
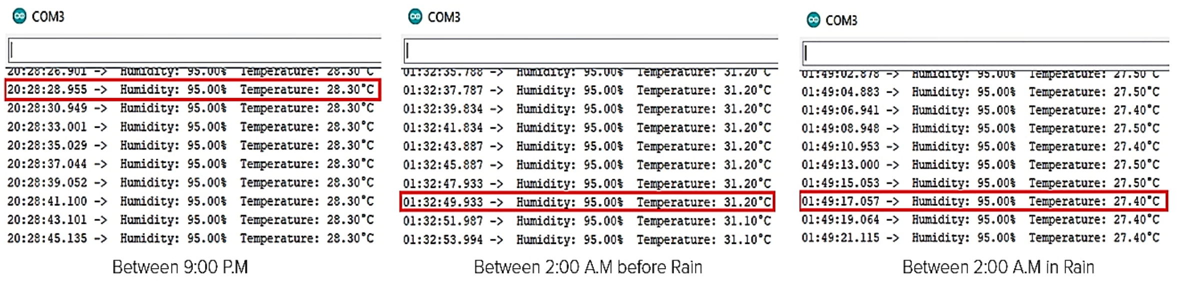


Figure 11. DHT-11 plots (serial plotter)



```
20:28:26.901 -> Humidity: 95.00% Temperature: 28.30°C
20:28:28.955 -> Humidity: 95.00% Temperature: 28.30°C
20:28:30.949 -> Humidity: 95.00% Temperature: 28.30°C
20:28:33.001 -> Humidity: 95.00% Temperature: 28.30°C
20:28:35.029 -> Humidity: 95.00% Temperature: 28.30°C
20:28:37.044 -> Humidity: 95.00% Temperature: 28.30°C
20:28:39.052 -> Humidity: 95.00% Temperature: 28.30°C
20:28:41.100 -> Humidity: 95.00% Temperature: 28.30°C
20:28:43.101 -> Humidity: 95.00% Temperature: 28.30°C
20:28:45.135 -> Humidity: 95.00% Temperature: 28.30°C
```

```
01:32:35.788 -> Humidity: 95.00% Temperature: 31.20°C
01:32:37.787 -> Humidity: 95.00% Temperature: 31.20°C
01:32:39.834 -> Humidity: 95.00% Temperature: 31.20°C
01:32:41.834 -> Humidity: 95.00% Temperature: 31.20°C
01:32:43.887 -> Humidity: 95.00% Temperature: 31.20°C
01:32:45.887 -> Humidity: 95.00% Temperature: 31.20°C
01:32:47.933 -> Humidity: 95.00% Temperature: 31.20°C
01:32:49.933 -> Humidity: 95.00% Temperature: 31.20°C
01:32:51.987 -> Humidity: 95.00% Temperature: 31.10°C
01:32:53.994 -> Humidity: 95.00% Temperature: 31.10°C
```

```
01:49:02.878 -> Humidity: 95.00% Temperature: 27.50°C
01:49:04.883 -> Humidity: 95.00% Temperature: 27.50°C
01:49:06.941 -> Humidity: 95.00% Temperature: 27.40°C
01:49:08.948 -> Humidity: 95.00% Temperature: 27.50°C
01:49:10.953 -> Humidity: 95.00% Temperature: 27.40°C
01:49:13.000 -> Humidity: 95.00% Temperature: 27.50°C
01:49:15.053 -> Humidity: 95.00% Temperature: 27.50°C
01:49:17.057 -> Humidity: 95.00% Temperature: 27.40°C
01:49:19.064 -> Humidity: 95.00% Temperature: 27.40°C
01:49:21.115 -> Humidity: 95.00% Temperature: 27.40°C
```

Figure 12. Serial monitor displaying DHT-11 output

Table 3. Current temperature and threshold value comparison rules

+-Current temperature (°C)	Threshold value	Rule	Weather status	Android app alarm	Timestamp range (from-to PM)
29.70	31.00	29.70<31.00	Moderate	No Alarm	3:40-7:00 PM
28.30	31.00	28.30<31.00	Moderate	No Alarm	8:28-9:00 PM
30.60	31.00	30.60<31.00	Moderate	No Alarm	10:05-11:14 PM
29.50	31.00	29.50<31.00	Moderate	No Alarm	11:14-12:00 AM
31.20	31.00	31.20>31.00	Unstable	Alarm	12:00-01:33 AM
27.40	31.00	27.40<31.00	Moderate	No Alarm	01:33-01:49 AM
32.00	31.00	32.00>31.00 (expected)	Unstable	Alarm	expected

5. CONCLUSION

Based on our research findings, we say that this proposed system is a low-cost IoT and Wi-Fi communication-based project built with locally available prototyping components. Although this research is still in the prototype stage, it has some limitations. The first finding is that the network of this system is only configured for the first time through AutoConnectAP. If we want to connect this system with another new network, it's necessary to re-flash memory and re-program the NodeMCU. In the future, we can use the Raspberry Pi to remove this limitation for network configuration. The DHT-11 sensor can measure temperatures up to 50 degrees Celsius, it is only suitable for moderate weather. In the future, we can use the DHT-22 instead of DHT-11 sensor for better weather data measurement. Limited predefined values are used for voice command. We can enhance it by using machine learning and natural language processing (NLP) methods to store predictive values for voice command operations. This system provides an effective automation mechanism, where anyone especially elderly person can control or monitor their home electronic stuff from any distance. It also provides a safety measure to detecting the unstable home weather activity. Everything is maintained by a small Android application with efficient voice control feature. A portable, lightweight system, install anywhere in the room. The network connectivity of this system is a typical Wi-Fi network or Internet connection. The aim of this project is to help people make their lives a little bit easier with the touch of smart automation.

REFERENCES

[1] W. ElMaraghy, H. ElMaraghy, T. Tomiyama, and L. Monostori, "Complexity in engineering design and manufacturing," *CIRP Annals*, vol. 61, no. 2, pp. 793–814, 2012, doi: 10.1016/j.cirp.2012.05.001.

[2] S. B. S. Bhati, "Review on Google Android a Mobile Platform," *IOSR Journal of Computer Engineering*, vol. 10, no. 5, pp. 21–25, 2013, doi: 10.9790/0661-1052125.

[3] P. Kaur and S. Sharma, "Google Android a mobile platform: A review," in *2014 Recent Advances in Engineering and Computational Sciences (RAECS)*, Mar. 2014, doi: 10.1109/raecs.2014.6799598.

[4] G. Aloi *et al.*, "Enabling IoT interoperability through opportunistic smartphone-based mobile gateways," *Journal of Network and Computer Applications*, vol. 81, pp. 74–84, Mar. 2017, doi: 10.1016/j.jnca.2016.10.013.

[5] F. Hobbs and N. Stoops, "Demographic Trends in the 20th Century. Census 2000 Special Reports," Superintendent of Documents, U, Washington, DC, Nov. 2002. Accessed: Mar. 13, 2023. [Online]. Available: https://eric.ed.gov/?id=ED477270

[6] J. G. EVANS, "Hypothesis: Healthy Active Life Expectancy (HALE) as an Index of Effectiveness of Health and Social Services for Elderly People," *Age and Ageing*, vol. 22, no. 4, pp. 297–301, 1993, doi: 10.1093/ageing/22.4.297.

[7] U. Nations, *World population ageing 2013*. United Nations, 2014, doi: 10.18356/30d0966c-en.

[8] L. Liu, Y. Liu, L. Wang, A. Zomaya, and S. Hu, "Economical and Balanced Energy Usage in the Smart Home Infrastructure: A Tutorial and New Results," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 556–570, Dec. 2015, doi: 10.1109/tetc.2015.2484839.

[9] K. M. S. Azad, N. Hossain, N. A. Samia, M. J. Islam, A. Hossain, and S. Kabir, "A Cost-Effective Internet of Things Based Smart Home System for Upcoming Technologies," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Jul. 2021, doi: 10.1109/acmi53878.2021.9528153.

[10] X. Guo, Z. Shen, Y. Zhang, and T. Wu, “Review on the Application of Artificial Intelligence in Smart Homes,” *Smart Cities*, vol. 2, no. 3, pp. 402–420, Aug. 2019, doi: 10.3390/smartcities2030025.
[11] K. S. S. Javvaji, U. R. Nelakuditi, and B. P. Dadi, “IoT Based Cost Effective Home Automation and Security System,” in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2020, doi: 10.1109/icccnt49239.2020.9225557.
[12] R. Susany and R. Rotar, “Remote Control Android-based Applications for a Home Automation implemented with Arduino Mega 2560 and ESP 32,” *Technium: Romanian Journal of Applied Sciences and Technology*, vol. 2, no. 2, pp. 1–8, Mar. 2020, doi: 10.47577/technium.v2i2.231.
[13] B. K. Sovacool and D. D. F. D. Rio, “Smart home technologies in Europe: A critical review of concepts, benefits, risks and policies,” *Renewable and Sustainable Energy Reviews*, vol. 120, p. 109663, Mar. 2020, doi: 10.1016/j.rser.2019.109663.
[14] H. Kim, H. Choi, H. Kang, J. An, S. Yeom, and T. Hong, “A systematic review of the smart energy conservation system: From smart homes to sustainable smart cities,” *Renewable and Sustainable Energy Reviews*, vol. 140, p. 110755, Apr. 2021, doi: 10.1016/j.rser.2021.110755.
[15] A. H. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and M. Z. Sheng, “IoT Middleware: A Survey on Issues and Enabling technologies,” *IEEE Internet of Things Journal*, pp. 1–1, 2016, doi: 10.1109/jiot.2016.2615180.
[16] K. Akshay and M. Sumanth, “Bluetooth based home automation by using arduino,” *Graduate Research in Engineering and Technology*, pp. 37–42, Dec. 2021, doi: 10.47893/gret.2021.1116.
[17] Y. Zhang and Q. Li, “Exploiting ZigBee in Reducing WiFi Power Consumption for Mobile Devices,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 12, pp. 2806–2819, Dec. 2014, doi: 10.1109/tmc.2014.2315788.
[18] D. S. T. D. S. Thakur, “Voice Recognition Wireless Home Automation System Based On Zigbee,” *IOSR Journal of Electronics and Communication Engineering*, vol. 6, no. 1, pp. 65–75, 2013, doi: 10.9790/2834-616575.
[19] Y. C. Ma and Y. H. Huang, “General Application Research on GSM Module,” *Applied Mechanics and Materials*, vol. 151, pp. 96–100, Jan. 2012, doi: 10.4028/www.scientific.net/amm.151.96.
[20] R. Teymourzadeh, S. A. Ahmed, K. W. Chan, and M. V. Hoong, “Smart GSM based Home Automation System,” in *2013 IEEE Conference on Systems, Process &amp$\mathsemicolon$ Control (ICSPC)*, Dec. 2013, doi: 10.1109/spc.2013.6735152.
[21] A. Francis, Sundar, and Gowtham, “Simulation of iot based home automation using arduino,” May 2021, pp. 93–97.
[22] M. Ali *et al.*, “An IoT based Approach for Efficient Home Automation with ThingSpeak,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, 2020, doi: 10.14569/ijacsa.2020.0110615.
[23] A. Maier, A. Sharp, and Y. Vagapov, “Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things,” in *2017 Internet Technologies and Applications (ITA)*, Sep. 2017, doi: 10.1109/itecha.2017.8101926.
[24] Y. Misra, *Programming and Interfacing with Arduino*. CRC Press, 2021. Accessed: Mar. 13, 2023. [Online]. Available: https://www.routledge.com/Programming-and-Interfacing-with-Arduino/Misra/p/book/9781032063164
[25] I. Toyoda, “Capacitive Humidity Sensor,” US20030010119A1, Jan. 16, 2003 Accessed: Mar. 13, 2023. [Online]. Available: https://patents.google.com/patent/US20030010119/en
[26] D. M. Faris and D. M. F. M. B. Mahmood, “Data Acquisition of Greenhouse Using Arduino,” *Journal of University of Babylon*, vol. 22, no. 7, pp. 1908–1916, 2014.
[27] R. Esbai and M. Erramdani, “Model-to-model transformation in approach by modeling: From UML model to Model-View-Presenter and Dependency Injection patterns,” in *2015 5th World Congress on Information and Communication Technologies (WICT)*, Dec. 2015, doi: 10.1109/wict.2015.7489648.
[28] Y. Zhang and Y. Luo, “An architecture and implement model for Model-View-Presenter pattern,” in *2010 3rd International Conference on Computer Science and Information Technology*, Jul. 2010, doi: 10.1109/iccsit.2010.5565090.
[29] I. Darwin, *Android Cookbook: Problems and Solutions for Android Developers*, 2nd edition. Beijing ; Boston: O’Reilly Media, 2017.
[30] R. Shrestha, “Study and Control of DHT11 Using Atmega328P Microcontroller,” *International Journal of Scientific and Engineering Research*, vol. 10, pp. 518–521, Apr. 2019.
[31] R. Ahmed and I.-K. Kim, “Patterns of Daily Rainfall in Bangladesh during the Summer Monsoon Season: Case Studies at Three Stations,” *Physical Geography*, vol. 24, no. 4, pp. 295–318, Jan. 2003, doi: 10.2747/0272-3646.24.4.295.

BIOGRAPHIES OF AUTHORS

Arunav Mallik Avi received his Bachelor degree in Computer Science & Engineering from the National University (NU) of Bangladesh. He has 2+ years of experience in software development. He worked as a Software Developer in BSD more than 1 year then he joined Dewan ICT as a Mobile Application Developer. Currently he is working as a Software Engineer at CMED Health Ltd. He has done several types of projects based on the software development, IoT, & computer vision, both industrially and for research purposes. His areas of interest are the software development, Internet of Things (IoT), embedded systems, artificial intelligence and computer vision. He can be contacted at email: armavicse@gmail.com.

Md. Sohel Rana graduated from Prime University (PU), Bangladesh, with a Bachelor of Science in Electrical and Electronic Engineering degree in 2015. In 2018, he graduated with a Master of Science in Telecommunication Engineering from the University of Information Technology and Sciences (UITS) in Bangladesh. In the year 2022, a second MSc Engineering degree was awarded by the Bangladesh University of Professionals (BUP) in Bangladesh. Currently, he is participating in the Department Electronics and Communication Engineering of the Ph.D. program offered by the Khulna University of Engineering and Technology (KUET), located in Bangladesh. Some of his research interests are microstrip patch antennas, wireless communication, image processing, renewable energy, biomedical engineering, control systems, and power electronics. Mr. Sohel Rana authored and co-authored in 26 papers regarding to his research works. At present he is working as a Lecturer at Northern University of Businesss and Technology, Khulna, Bangladesh. He can be contacted at email: sohel.rana@uits.edu.bd.

Mustakim Billah Bedar received his MSc degree in Computer Science & Engineering from Jahangirnagar University (JU), Bangladesh. Currently he is working as a lecturer at the Shaikh Burhanuddin Post Graduate College (SBPGC) in the department of Computer Science & Engineering. His areas of interest are theoretical computing, operating system, computer linguistic, computer network and embedded systems. He can be contacted at email: mustakimbilla007@gmail.com.

Md. Asif Talukder received his BSc degree from National University (NU), Bangladesh, in the Department of Computer Science & Engineering. He has 2 years of experience in Software Quality Assurance & Testing. He worked as a SQA Engineer in Addie Soft Ltd. Currently he is pursuing his Master’s degree at the Oklahoma City University (OKCU) in the department of Computer Science. His areas of interest are software quality assurance & testing, web engineering, software development & business analytics. He can be contacted at email: matalukder@my.okcu.edu.