Vol. 03, Iss. 1 (2026) 13-24, DOI: 10.61552/JSI.2026.01.002

Journal of Sustainable Development Innovations

www.jsi.aspur.rs

Research article

Sustainable Smart Kitchen: IoT-Enabled Energy-Efficient Automation and Monitoring with ESP8266 and Blynk

Wazeer Ahmad Khan[a], Paresh Sagar[b,*]

[a]*Intelligent Communication Systems, School of Electronics Engineering, Vellore Institute of Technology, Vellore , Tamil Nadu, India,*

[b]*Vellore Institute of Technology, Assistant Professor, Vellore , Tamil Nadu, India.*

Keywords:

IoT Based Smart Kitchen, Remote Monitoring, Blynk Integration, ESP8266 Communication, ThingSpeak Cloud, Smart Home Automation, Real-Time Alerts.

* Corresponding author:

Paresh Sagar
E-mail: pareshkumar.sagar@vit.ac.in

Received: 17 January 2025
Revised: 18 February 2025
Accepted: 14 March 2025

A B S T R A C T

This paper present the Smart Kitchen utilizes cutting-edge IoT technologies to boost productivity, safety, and convenience. By integrating a centralized control network, users may remotely monitor and manage kitchen appliances, assuring seamless operation. In addition to continuously monitoring the atmosphere and identifying potentially dangerous gases, the system also uses motion detection to improve security. An essential part of this system is the Blynk app, which offers an intuitive interface for remote appliance management and real-time access to sensor data through smartphones, increasing convenience by enabling users to make knowledgeable decisions regarding kitchen operations even when they are not there. By spotting possible fire hazards and promptly issuing alerts, the fire detection system is essential to maintaining safety. Furthermore, effective data analysis and storage are made possible by the integration of ThingSpeak cloud, guaranteeing real-time monitoring. This IoT-enabled kitchen solution is a great option for contemporary homes since it maximizes energy use while simultaneously increasing operational efficiency. AI and machine learning may be used in future developments for intelligent decision-making, further automation, and predictive maintenance. This system redefines traditional kitchen management with improved control and security by utilizing smart technology to provide a safer and more effective cooking environment.

1. INTRODUCTION

From healthcare to transportation, the Internet of Things (IoT) has completely changed many facets of our life. The kitchen is one area where notable changes have occurred. The advent of IoT technology in kitchens has resulted in the creation of smart kitchens, which provide improved safety, convenience. In-depth discussion of the

elements, features, and possible advantages of smart IoT-enabled kitchens is provided in this research study [1].

By connecting appliances and systems, smart kitchens make use of IoT devices and sensors to provide remote monitoring, control, and automation. These kitchens can have smart ovens that can cook and preheat food according to precise instructions, voice-activated assistants that can manage several culinary tasks, and clever refrigerators that keep track of inventory and offer recipe suggestions [2]. Smart kitchens use IoT to improve cooking efficiency, cut down on waste, and improve the cooking experience in general.

In-depth descriptions of the main components and technology supporting smart IoT-enabled kitchens will be given in this paper. It will examine the advantages and difficulties of implementing these cutting-edge systems [3].

The study will also go over how smart kitchens might affect food safety, energy efficiency, and consumer behavior. This study intends to support the further development and use of this revolutionary technology by better understanding the subtleties of smart, Internet of Things-enabled kitchens [4].

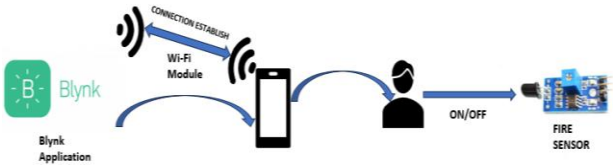


Fig. 1. Working of Fire sensor based on wi-fi module for smart kitchen.

Through the Wi-Fi module, the Blynk server connects to the user's smartphone, allowing for smooth communication between the two. The user may easily control the fire sensor after it is connected to Wi-Fi. For example, the fire sensor is simple to turn on and off, allowing the user to make sure it is active when needed and inactive when not. More control over the kitchen environment and improved user convenience are provided by this functionality. This connection, as shown in Fig. 1, demonstrates how the Blynk application simplifies the administration of fire safety precautions in smart kitchen systems, giving users remote accessibility and real-time monitoring [5].

2. LITERATURE REVIEW

This paper real-time data gathering and remote monitoring capabilities are highlighted as IoT integration in kitchen monitoring systems is discussed. In order to maintain efficiency and safety throughout kitchen activities. The study delves at IoT applications in kitchen settings, emphasizing the utilization of smart devices to automate culinary procedures and keep an eye on kitchen equipment with the goal of minimizing human interaction and improving safety [7].

The study focuses on utilizing the ESP8266 module to integrate voice command features in the context of smart home systems. Voice commands can now be used to operate household appliances, increasing accessibility and convenience for consumers. The architecture and implementation of numerous smart devices connected via a Wi-Fi network are described in depth in another paper titled The significance of effective data transmission and network security is emphasized in this study [6].

The study examines the wider uses of IoT in home automation and touches on a number of topics, including energy management, lighting, and security. The advantages of IoT in developing more responsive and adaptable home environments are highlighted in this study. Likewise, explores how to use IoT in home automation systems, talking about how to combine various sensors and actuators to make a harmonious and intelligent living space [8].

The paper focuses on the Node-MCU platform and describes the hardware and software components needed to construct a scalable and reasonably priced smart home system. An IoT-based smart house automation system's design, implementation, and performance evaluation are covered in detail in another extensive study.

The research paper examines the creation of mobile applications for Wi-Fi module-based smart home device control. This study emphasizes how different smart devices are integrated into a single control system and how user interface design is done [9].

Finally, just like the previous paper, explores the application of IoT technology in kitchen emphasizing the use of sensors, advantages of real-time data monitoring for enhancing kitchen efficiency and safety [10].

3. METHODOLOGY

The system begins by collecting data in real time from various sensors to connect to an Arduino, including motion sensors, temperature and humidity sensors, fire, smoke, and Led [11]. This setup provides focus on environmental changes such temperature fluctuations, the existence of smoke, and motion detection. Once the data has been collected, a Wi-Fi module is used to transmit it from the Arduino to the Thing Speak IoT cloud platform. Thing Speak performs as a tool providing data visualization, providing users access to graphs and alerts based on sensor readings that help users see relationships and learn more. This analysis can be used to turn on or off LEDs remotely and helps identify potential threats like fire or unusual temperature conditions [12].

The same Wi-Fi module has been implemented to connect the system to the Blynk server simultaneously, which allows remote control via the Blynk mobile app. Working as a user interface, the application lets you communicate with the Arduino-connected devices. Users can monitor sensor data through the app while maintaining real-time access to components like the LED. For example, they can reset sensors, turn on or off the LED, or respond to notifications that get set off by sensor data [13].

The seamless transfer of data and control is provided by the integration of Blynk and ThingSpeak. The Blynk app provides an efficient method to control the system from anywhere, while ThingSpeak offers constant monitoring and analysis. The system updates itself with the changes made through the app, maintaining a responsive and dynamic environment [14]. By providing that data is regularly up-to-date this continuous feedback loop allows users to act quickly in response to the conditions that the sensors are observing. All things considered, the method offers a practical way to handle data analysis, device remote control, and real-time monitoring [15].

A smart kitchen system using Arduino are shown in Fig. 2, where sensors including temperature, motion, fire, and smoke are connected for collecting data. A cloud server like ThingSpeak gets this data via a Wi-Fi module and analyzes it. Users can monitor sensor data and control devices like LEDs and sensors from a smartphone through the Blynk app to control the system [16].

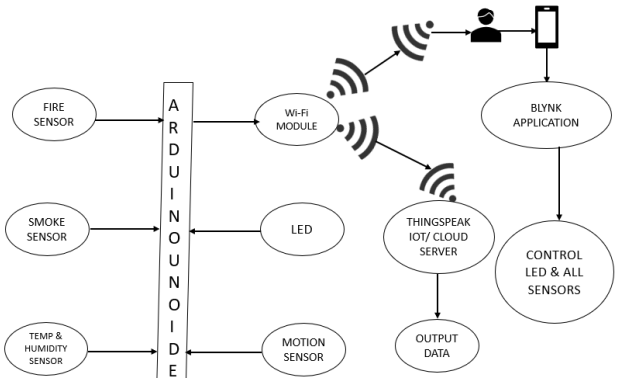


Fig. 2. Wi-fi module and user interface architecture for a smart kitchen.

4. HARDWARE AND SOFTWARE USED

4.1 Arduino Uno Ide

It offers a simple user interface that allows users establish, build, and upload code to various Arduino devices. Using a huge collection of pre-built functions and support for several programming languages (mostly C and C++), the IDE helps up the process of prototyping. The application makes debugging and testing easier with features like syntax highlighting and an included serial monitor. The efficiency of IoT applications is enhanced by the Arduino IDE's smooth interaction with various modules, like the ESP8266 Wi-Fi transceiver, as illustrated in Fig. 3.

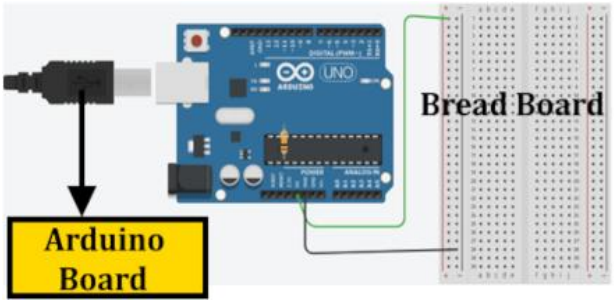


Fig. 3. An illustration of the Arduino Uno's connection to a bread board.

4.2 Fire Sensor

A crucial instrument for maintaining safety and quick response in the case of a fire hazard is a fire sensor, it is designed to detect heat or fire in a certain area. These sensors, which typically employ technologies like photoelectric or ionization detection, may provide notifications quickly to stop major loss and damage. Systems as smart home setups integrate fire sensors for automatic notification and responses. The fire

sensor, as shown in Fig. 4, is linked to an Arduino microcontroller that uses the Internet of Things to give continuous control and monitoring applications, improving safety in both residential and commercial areas.

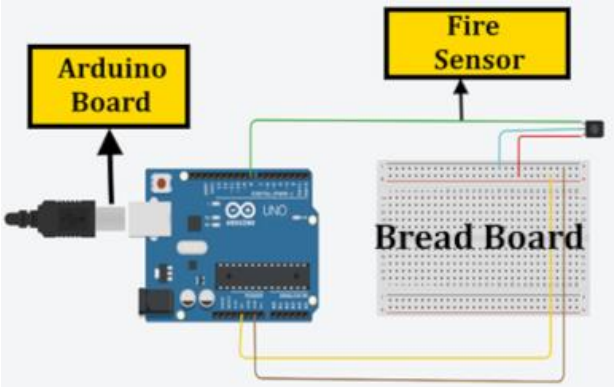


Fig. 4. Circuit diagram of the fire sensor module linked to Arduino Uno through Bread board.

4.3 Smoke Sensor

One of the most crucial tools for security is a smoke sensor, which detects smoke anywhere it is found. It works by sensing smoke particles using a variety of technologies, including photoelectric or ionization techniques. The sensor warns the occupants and enables an instant escape when it senses smoke.

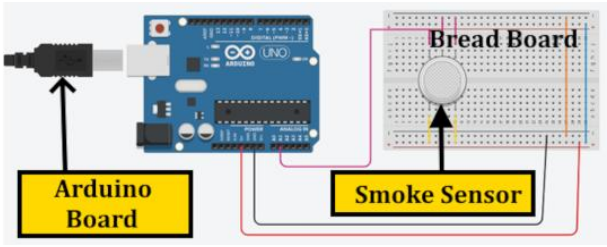


Fig. 5. Circuit diagram of the smoke sensor module linked to Arduino Uno via Bread board.

4.4 Temperature & Humidity Sensor

The ambient temperature and the degree of moisture in the air are determined by DHT11. The component integrates a heating element with a sensitive moisture detector to produce digital output data that is quickly read by Arduino. These sensors are widely used for surrounding condition monitoring in HVAC systems, weather stations, and Internet of Things projects. They are suitable for a range of applications due to their quick response times, calibrated output, and low power consumption. They can be included into smart kitchen systems for in-the-moment climate control, as seen in Fig. 6.

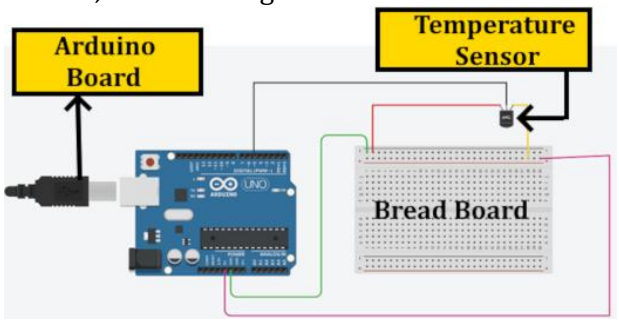


Fig. 6. Circuit diagram of the temperature sensor module connected to Arduino Uno with the help of Bread board.

4.5 Led

It has been used in various fields, including displays, lighting, and indicators, and it is long-lasting and energy-efficient. The colour of an LED might vary depending on the type of semiconductor and doping components employed. They are appropriate for a variety of electrical equipment because to their low heat production and low power consumption. An LED is incorporated into the system in Fig. 7 to give visual feedback, improving user involvement and showing the state of different activities. We may also refer to this LED as a kitchen light as it is attached to digital pin 11 on the Arduino board for output.

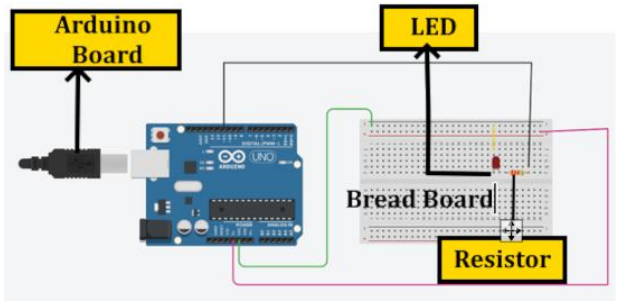


Fig. 7. Circuit diagram of the LED connected to Arduino Uno via Bread board.

4.6 Motion Sensor

Once a motion sensor senses movement in the surrounding environment, it transforms that movement into an electrical signal. It frequently detects changes in motion or body heat using technologies including infrared (PIR), microwave, or ultrasonic. The sensor detects motion and sets off linked devices like cameras, lights, and alarms. By identifying intruders and

controlling lighting according to occupancy, it improves security in smart home applications. Motion sensors can be linked with IoT systems, as seen in Fig. 8, and detected data is sent to cloud servers for additional processing and remote monitoring.

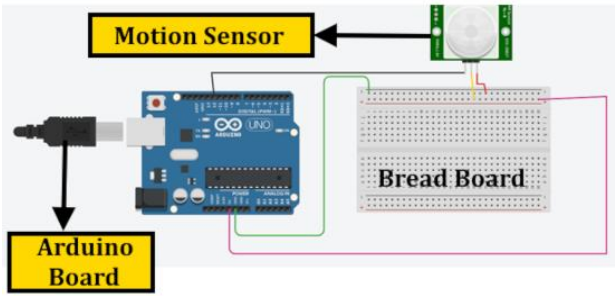


Fig. 8. Circuit diagram of the motion sensor module connected to Arduino Uno through Bread board.

4.7 Wi-Fi Module (Esp8266)

It combines a TCP/IP stack with a microcontroller to enable gadgets for internet browsing and cellular network connections. Flexible network configurations are made possible by the module's support for different modes, which include Station (STA), Access Point (AP), and both combined (STA+AP).

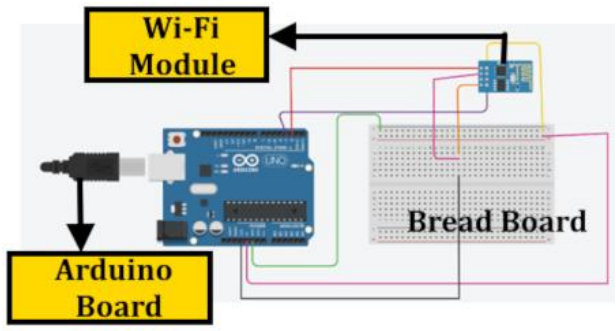


Fig. 9. Circuit diagram of the Wi-Fi module connected to Arduino Uno with the help of Bread board.

It is appropriate for smart kitchens, remote monitoring, and smart devices because it has GPIO pins for attaching sensors and actuators. The Arduino IDE platform offers developers flexibility when programming the ESP8266.

Its popularity is a result of its price, small size, and strong community support, which make it a great option for both novice and seasoned IoT developers. Figure 9 illustrates how the cloud server is receiving the data.

4.8 Hardware Implementation

An integrated smart kitchen monitoring system is created by this project by utilizing an Arduino Uno as the primary controller and connecting it to a variety of sensors.

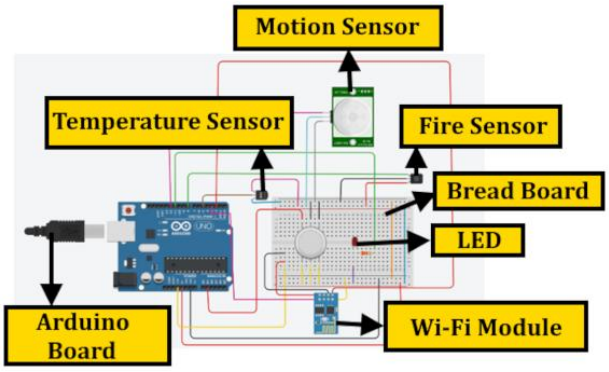


Fig. 10. Image of a bread board is used to connect the Arduino Uno to all of the sensors, LEDs, and wi-fi modules in order to transfer data to the cloud.

4.9 Cloud Server (Thingspeak)

It makes data storage in a cloud-based environment easier and supports a variety of data sources, such as sensors and controllers. Users can use MATLAB analytics for advanced data processing, construct channels for data organization and sharing, and create customisable charts for visualization. In addition, ThingSpeak supports MQTT for instantaneous communication and integrates with services like Blynk for mobile app connectivity. It is perfect for IoT projects and intelligent applications because to its powerful features and easy-to-use API.

4.10 Blynk Application

A versatile Internet of Things (IoT) platform, It enables users to create mobile apps for controlling and keeping an eye on linked devices. It has an intuitive UI that makes creating dashboards with hardware control widgets as simple as dragging and dropping. Blynk is appropriate for a variety of gadgets due to its support for several communication protocols, including Bluetooth, Wi-Fi, and LoRa. Through the app, users may automate chores, view real-time data, and receive notifications. A popular option for IoT projects and home automation systems, Blynk also enables secure device access and cloud storage for data.

5. RESULT AND DISCUSSIONS

5.1 All Sensors Is Not Activated

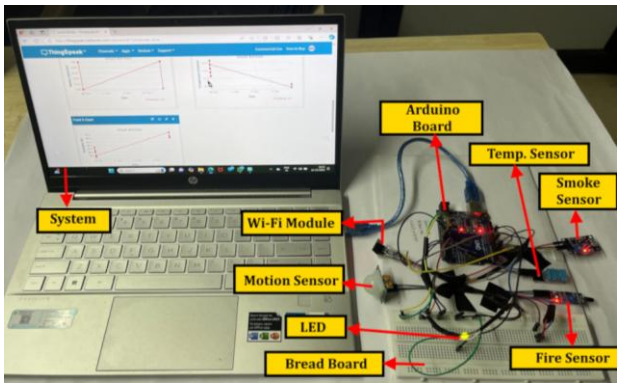


Fig. 11. Wi-Fi module configuration for experimentation with dormant sensors.

Once the code has been uploaded to the devices, they are operational, as illustrated in Fig. 11. USB cable can be used to power the gadgets. Since no detectors are turned on in the initial state, the system stays inactive until certain processes are triggered.

Code that uses sensors to detect motion, temperature, humidity, flame, and smoke and transmits the data to a cloud server via a wi-fi module:

```
#include <SoftwareSerial.h>
#include "DHT.h"
#define RX 2
#define TX 3
const int fireSensorPin = 8;
const int smokeA0 = A0;
#define DHTPIN 5
#define DHTTYPE DHT11
const int pirPin = 12;
const int ledPin = 11;
String API = "1FOG8FE9SJZC3CQI";
String HOST = "api.thingspeak.com";
String PORT = "80";
int countTrueCommand;
SoftwareSerial esp8266(RX, TX);
unsigned char check_connection = 0;
unsigned char times_check = 0;
DHT dht(DHTPIN, DHTTYPE);
void setup() {
  Serial.begin(115200);
  esp8266.begin(115200);
  pinMode(fireSensorPin, INPUT);
  pinMode(smokeA0, INPUT);
  pinMode(pirPin, INPUT);
  dht.begin();
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);
  Serial.println("SMART KITCHEN IMPLEMENTED BY WAZEER KHAN!!!!");
  esp8266.print("***VER:");
  delay(2000);
  esp8266.println("AT+RST");
  delay(1000);
  esp8266.println("AT+GMR");
  delay(1000);
  esp8266.println("AT+CWMODE=3");
  delay(1000);
  esp8266.println("AT+CWLAP");
  delay(1000);
  Serial.println("Connecting to Wifi");
  while (check_connection == 0) {
    esp8266.print("AT+CWJAP=\"wazir349\",\"6396029692\"\r\n");
    delay(5000);
    if (esp8266.find("WIFI CONNECTED")) {
      Serial.println("WIFI CONNECTED");
      check_connection = 1;
      break;
    }
    times_check++;
    if (times_check > 3) {
      times_check = 0;
      Serial.println("Trying to Reconnect..");
    }
  }
  sendCommand("AT+CIPMUX=0", 5);
}
void loop() {
  int fireValue = digitalRead(fireSensorPin);
  String fireDetected = (fireValue == LOW) ? "1" : "0";
  Serial.println(fireDetected == "1" ? "Flame detected" : "Flame not detected");
  float smokeValue = 0;
  for (int i = 0; i < 10; i++) {
    smokeValue += analogRead(smokeA0);
    delay(10);
  }
  smokeValue /= 10;
  Serial.print("Averaged smoke sensor value: ");
  Serial.println(smokeValue);
  String smokeDetected = (smokeValue > 175) ? "1" : "0";
  Serial.println(smokeDetected == "1" ? "Smoke detected" : "Smoke not detected");
  String motionDetected = (digitalRead(pirPin) == HIGH) ? "1" : "0";
  Serial.println(motionDetected == "1" ? "Motion detected" : "Motion not detected");
  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();
  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print(" %\tTemperature: ");
  Serial.print(temperature);
  Serial.println(" *C");
  String getData = "GET /update?api_key=" + API + "&field1=" + fireDetected
  + "&field2=" + smokeDetected
                 + "&field3=" + motionDetected + "&field4=" + String(temperature)
                 + "&field5=" + String(humidity);
  sendCommand("AT+CIPSTART=\"TCP\",\"" + HOST + "\"," + PORT, 15);
  delay(2000);
  sendCommand("AT+CIPSEND=" + String(getData.length() + 4), 4);
  esp8266.println(getData);
  delay(2000);
  Serial.println("Data sent to ThingSpeak: " + getData);
  sendCommand("AT+CIPCLOSE", 5);
  delay(2000);
}
void sendCommand(String command, int maxTime) {
  Serial.print(countTrueCommand);
  Serial.print(". at command => ");
  Serial.print(command);
  Serial.print(" ");
  esp8266.println(command);
  delay(maxTime * 1000);
  Serial.println("Command sent");
  countTrueCommand++;
}
```

Output

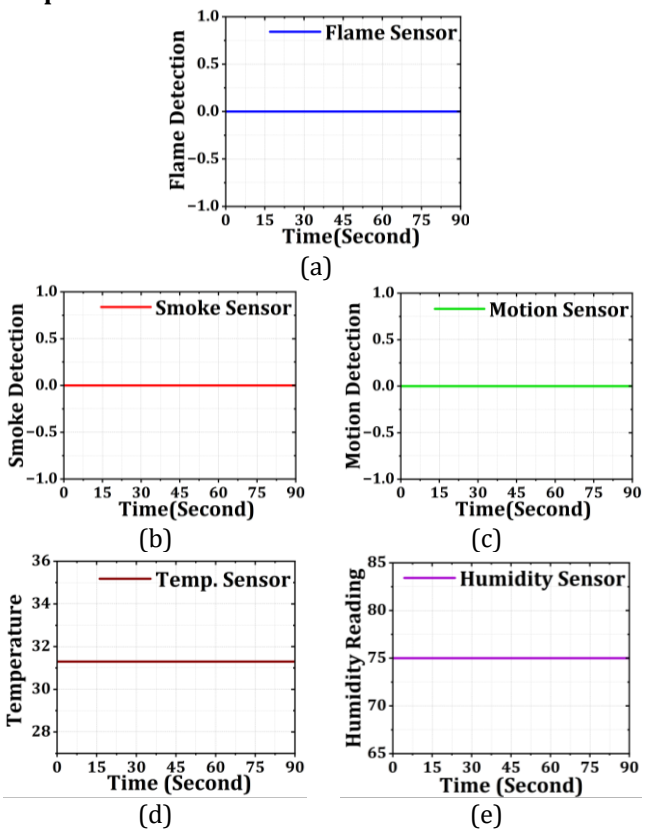


Fig. 12. Experimental result of proposed setup when all sensor are in off state (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
15. at command => AT+CIPCLOSE Command sent
Flame not detected
Averaged smoke sensor value: 83.90
Smoke not detected
Motion not detected
Humidity: 84.00 %       Temperature: 31.30 *C
16. at command => AT+CIPSTART="TCP","api.thingspeak.com",80
```

5.2 Only Flame Sensor Is Active Other Are Remain Unchanged

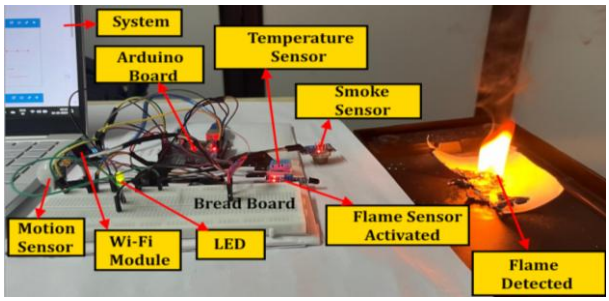


Fig. 13. Experimental setup demonstrating the alarm system and fire sensor.an instant escape The Smart Kitchen configuration is shown in Figure 13. The system's sole active part, the flame sensor, keeps an eye out for any fire threats. It records variations in the readings when it detects a flame, signifying the existence of fire. To save energy and concentrate just on detecting fires, all other sensors stay dormant.

Output

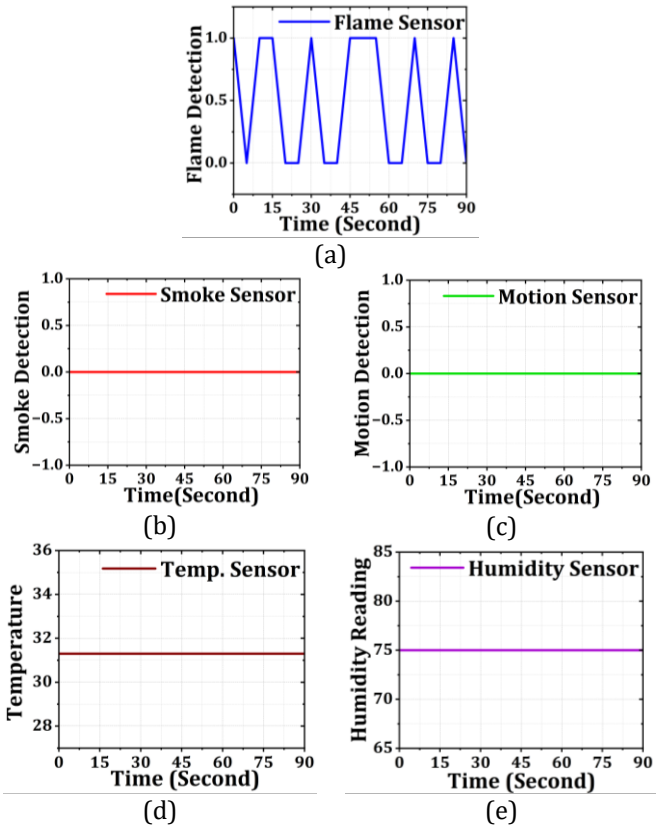


Fig. 14. Experimental result of proposed setup when only flame is detected (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
33. at command => AT+CIPCLOSE Command sent
Flame detected
Averaged smoke sensor value: 119.50
Smoke not detected
Motion not detected
Humidity: 83.00 %       Temperature: 31.80 *C
34. at command => AT+CIPSTART="TCP","api.thingspeak.com",80
```

5.3 Only Smoke Sensor Is Active And Other Are Remain Unchanged

The Smart Kitchen configuration is shown in Figure 15. The system's single active part, the smoke sensor, keeps an eye out for potential smoke threats. It captures changes in the measurements when it detects smoke, proving

that smoke is there. To save energy and concentrate just on smoke detection, all other sensors stay dormant.

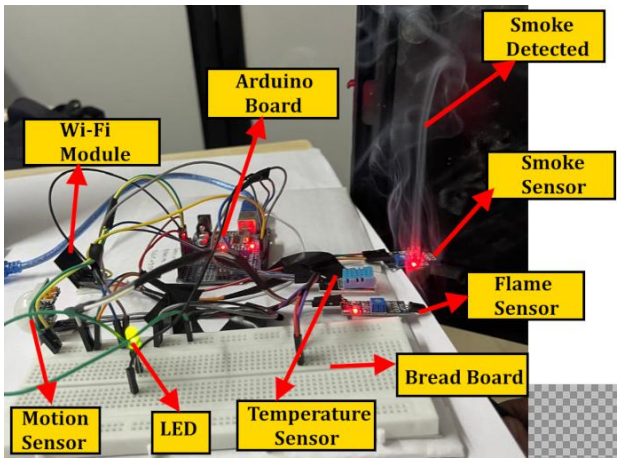


Fig. 15. The working smoke sensor is used in an experimental setup to demonstrate smoke detection.

Output

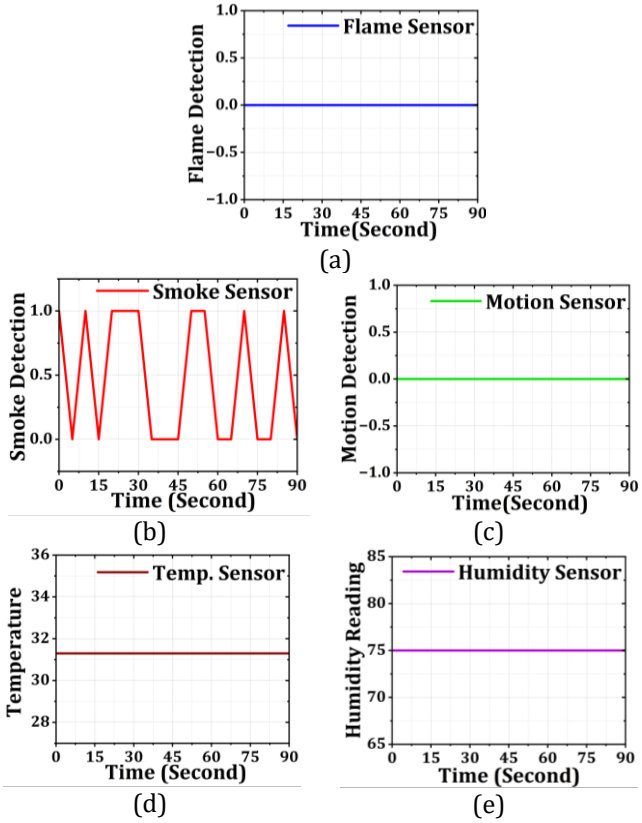


Fig. 16. Experimental result of proposed setup when only smoke is detected (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
75. at command => AT+CIPCLOSE Command sent
Flame not detected
Averaged smoke sensor value: 323.80
Smoke detected
Motion not detected
Humidity: 87.00 %       Temperature: 31.30 *C
76. at command => AT+CIPSTART="TCP","api.thingspeak.com",80 Command sent
```

5.4 Only Motion Sensor Is Active And Other Sensors Unchanged

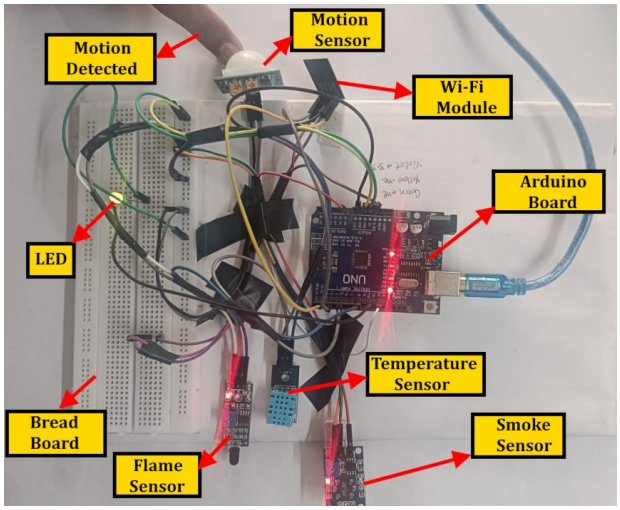
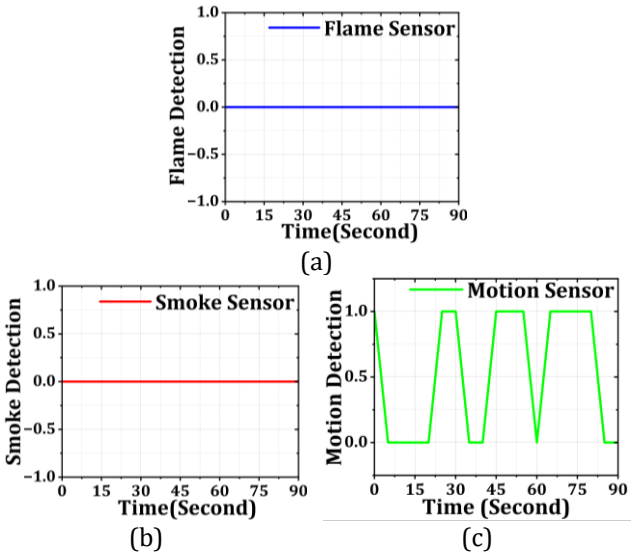
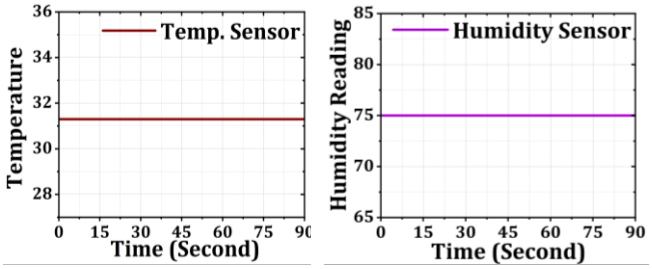


Fig. 17. Finger movement is detected via an experimental setup using a PIR motion sensor.

The Smart Kitchen configuration is shown in Figure 17. The system's sole active part, the motion sensor, keeps an eye out for motion risks at all times. It captures variations in the readings when it detects motion, showing that there is motion of any kind. For example, in this image, we can see that the finger is moving. To save electricity and concentrate just on motion detection, all other sensors stay idle.

Output





(d) (e)

Fig. 18. Experimental result of proposed setup when only motion is detected (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
SMART KITCHEN IMPLEMENTED BY WAZEER KHAN!!!!
Connecting to Wifi
WIFI CONNECTED
0. at command => AT+CIPMUX=0 Command sent
Flame not detected
Averaged smoke sensor value: 88.40
Smoke not detected
Motion detected
Humidity: 87.00 %       Temperature: 31.30 *C
```

5.5 Only Humidity & Temperature Sensor Is Active And Other Sensors Unchanged

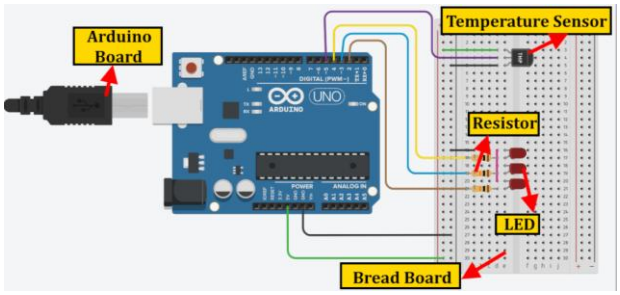


Fig. 19. Temperature and humidity are detected by a working DHT11 in an experimental configuration.

The Smart Kitchen configuration is shown in Figure 19. The system's sole active part, the temperature and humidity sensor, continuously checks the kitchen's temperature and humidity levels. It records variations in the measurements when it detects high or low temperature and humidity, signifying the presence of anything hot or cold. To save energy and concentrate just on detecting temperature and humidity, all other sensors stay dormant.

Output

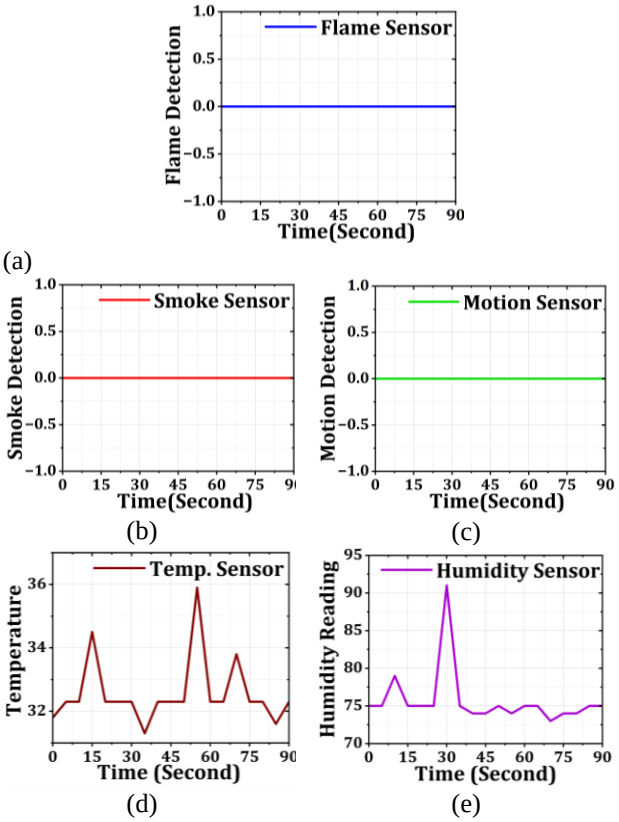


Fig. 20. Experimental result of proposed setup when only Temperature & humidity is detected (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
48. at command => AT+CIPCLOSE Command sent
Flame not detected
Averaged smoke sensor value: 87.50
Smoke not detected
Motion not detected
Humidity: 98.00 %       Temperature: 36.30 *C
49. at command => AT+CIPSTART="TCP","api.thingspeak.com",80 Command sent
```

5.6 All Sensors Is Activated

Figure 21. shows the Smart Kitchen configuration. Every sensor, including those for motion, temperature, humidity, smoke, and flame, is active. By identifying flames and documenting variations in the data, the flame sensor continuously checks for fire threats. When smoke is detected, it signals a possible fire or other dangerous situation. In order to identify any unlawful activity or access, the motion sensor tracks movement in the kitchen. By measuring environmental conditions, the temperature and humidity sensors provide useful information for preserving a secure and regulated environment.

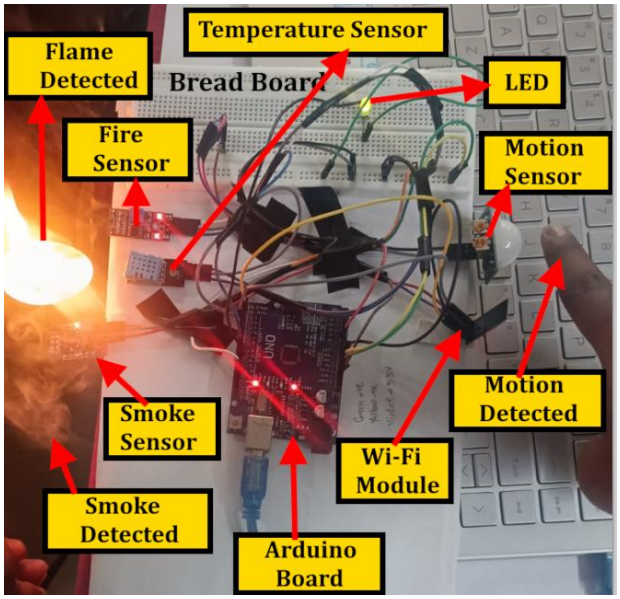


Fig. 21. The experimental configuration displays all of the active sensors, including motion, smoke, flame, and DHT11.

Output

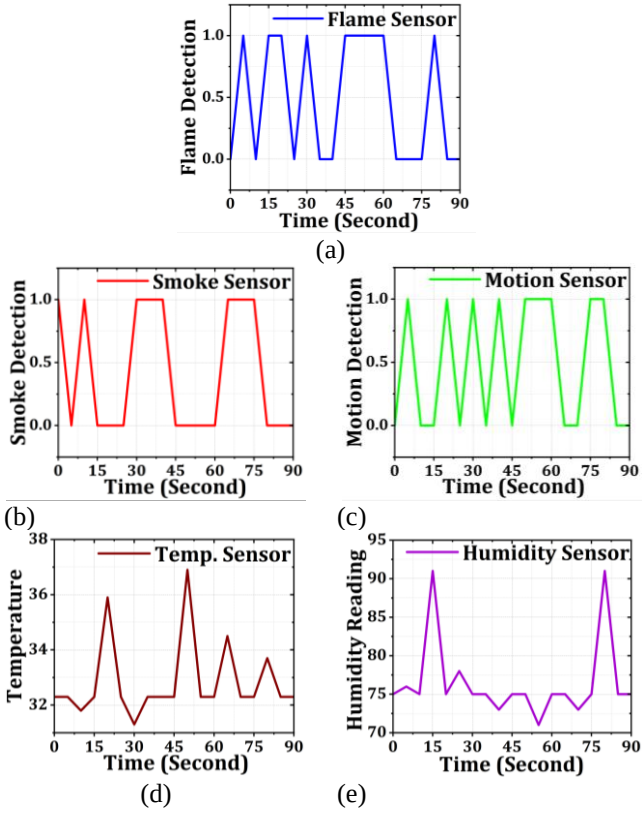


Fig. 22. Experimental result of proposed setup when all sensor are in on state (a) Plot of time Vs flame detection for flame sensor (b) Plot of time Vs smoke detection for smoke sensor (c) Plot of time Vs motion detection for motion sensor (d) Diagrams showing time versus temperature detection for temperature sensor (e) Diagrams showing time versus humidity detection for humidity sensor.

```
105. at command => AT+CIPCLOSE Command sent
Flame detected
Averaged smoke sensor value: 193.30
Smoke detected
Motion detected
Humidity: 74.00 %       Temperature: 34.70 *C
106. at command => AT+CIPSTART="TCP","api.thingspeak.com",80 Command sent
```

5.7 The Blynk Application's Code For Managing Every Sensor And Led

This flow chart shows how all of the sensors and LEDs are controlled by the Blynk program. I have to connect my phone to the internet before start the simulation. Next, we need to use a wi-fi module to see the Blynk server is linked to my physical circuit or not. It's easy to control our sensors and LED after everything is connected.

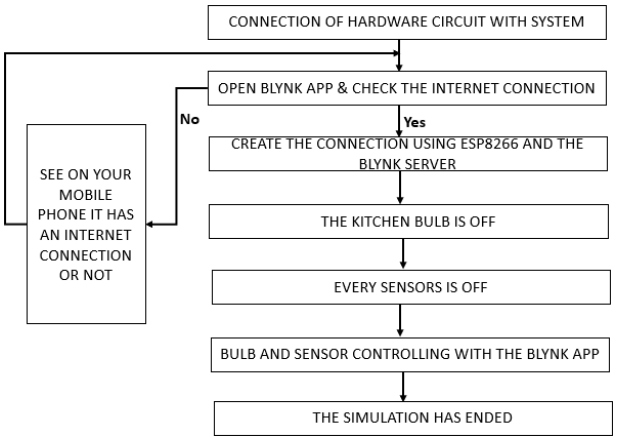


Fig. 23. Blynk Architecture for smart kitchen by using wi-fi module.

```
#define BLYNK_TEMPLATE_ID "TMPL3DwTH6X2v"
#define BLYNK_TEMPLATE_NAME "Cloud Kitchen"
#define BLYNK_AUTH_TOKEN "_Ayy4k2m0mlC-Bhj_Dk6LL5pIbuKBX5o"
#define BLYNK_PRINT Serial
#include <SoftwareSerial.h>
#include <ESP8266_Lib.h>
#include <BlynkSimpleShieldEsp8266.h>
char ssid[] = "wazir349";
char pass[] = "abcdefgh";
SoftwareSerial EspSerial(2, 3);
#define ESP8266_BAUD 38400
ESP8266 wifi(&EspSerial);
void setup()
{
  Serial.begin(115200);
  EspSerial.begin(115200);
  delay(1000);
  Serial.println("Testing ESP8266 at 115200 baud...");
  EspSerial.println("AT");
  delay(1000);
  if (EspSerial.available()) {
    String response = EspSerial.readString();
    Serial.println("Response: " + response);
  } else {
    Serial.println("No response from ESP8266 at 115200.");
  }
```

```
Serial.println("Setting ESP8266 baud rate to 38400 with 8N1 and flow control...")
EspSerial.println("AT+UART_DEF=38400,8,1,0,1");
delay(1000);
if (EspSerial.available()) {
  String response = EspSerial.readString();
  Serial.println("UART settings response: " + response);
} else {
  Serial.println("No response after setting UART.");
}
EspSerial.begin(ESP8266_BAUD);
delay(1000);
Serial.println("Testing ESP8266 at 38400 baud...");
EspSerial.println("AT");
delay(1000);
if (EspSerial.available()) {
  String response = EspSerial.readString();
  Serial.println("Response at 38400: " + response);
} else {
  Serial.println("No response from ESP8266 at 38400.");
  return;
}
Serial.println("Connecting to Blynk...");
Blynk.begin(BLYNK_AUTH_TOKEN, wifi, ssid, pass, "blynk.cloud", 80);
}
void loop()
{
  Blynk.run();
}
```

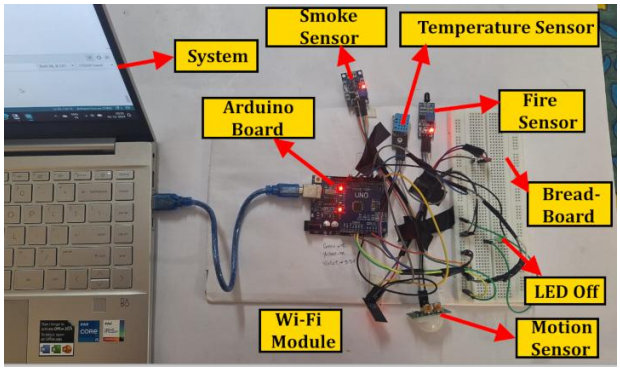


Fig. 24. Before connecting to the Blynk server, the experimental setup had all sensors and LEDs turned off.

This image indicates that every sensor is in the off state, and the screenshots demonstrate that it has been successfully connected to the Blynk app. I want to operate every sensor using a wi-fi module with the assistance of Blynk.

In this instance, the LED is also off, and all of the sensors are in an inactive state.

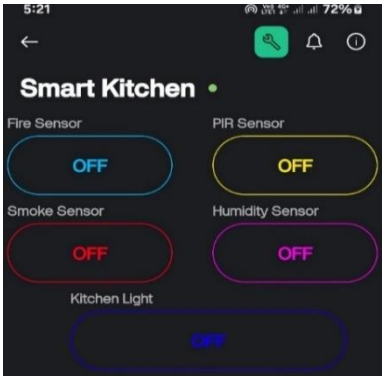


Fig. 25. This screenshot demonstrates how the hardware was successfully connected to the Wi-Fi network; the sensors and LEDs were initially turned off.

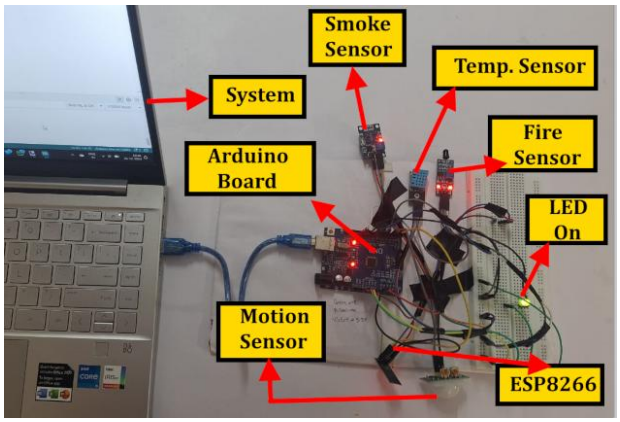


Fig. 26. The experimental setup is completely controlled, LED-lit, and all sensors are operational after successfully connecting to Blynk.

With the aid of the Blynk app, I will be able to operate every sensor via a wi-fi module. The screenshots and figure demonstrate that the sensors are all operational and that the connection to the Blynk app has been established.

In this instance, every sensor is operational and in an active state, and the LED is also turned on.

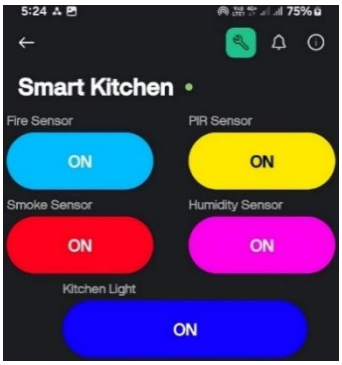


Fig. 27. This screen capture demonstrates a successful hardware connection over Wi-Fi, with sensors and LEDs controlled manually.

6. CONCLUSION

By improving convenience, safety, and energy efficiency, the Smart Kitchen project transforms kitchen management through the use of Internet of Things (IoT) technology. To facilitate real-time monitoring and control of kitchen appliances, it integrates Wi-Fi modules, gas leak sensors, and flame detectors. Users can monitor and control their kitchen appliances from any location thanks to the Blynk app and the integration of cloud-based platforms like ThingSpeak, which guarantee smooth data storage and analysis.

The Smart Kitchen project optimizes resource consumption and enhances user control, which not only improves daily operations but also contributes to a more sustainable and intelligent living environment. One of the main advantages of this system is its ability to prevent potential hazards, such as gas leaks, overheating, and fire incidents, by sending instant alerts and triggering necessary actions. It also helps to reduce manual effort and energy wastage by automating various kitchen functions.

REFERENCES

[1] G. R. Babu, P. V. Ch, P. K. Sree, A. V. S. Asha, G. Sujatha, K. S. Sharmila, and V. V. Sai, "Design and implementation of a dynamic IoT cloud based processing platform," *Proc. Eng.*, vol. 6, no. 4, pp. 1813–1820, 2024.

[2] S. Gupta, "IoT based smart home automation using NodeMCU," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. 9, pp. 1179–1188, 2021.

[3] S. Kousalya, G. R. Priya, R. Vasanthi, and B. Venkatesh, "IoT based smart security and smart home automation," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 7, no. 4, pp. 43–46, 2018.

[4] A. Doshi, D. Vakharia, and Y. Rai, "IoT based home automation," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 9, no. 8, 2021.

[5] B. M. Amine, C. F. Zohra, H. Ilyes, A. Lahcen, and A. Tayeb, "Smart home automation system based on Arduino," *IAES Int. J. Robot. Autom.*, vol. 7, no. 4, p. 215, 2018.

[6] S. A. Devi, M. S. Ram, K. Ranganarayana, D. B. Rao, and V. Rachapudi, "Smart home system using voice command with integration of ESP8266," in *Proc. 2022 Int. Conf. Appl. Artif. Intell. Comput. (ICAAIC)*, May 2022, pp. 1535–1539, IEEE.

[7] T. Chava, A. T. Srinivas, A. L. Sai, and V. Rachapudi, "IoT based smart shoe for the blind," in *Proc. 2021 6th Int. Conf. Invent. Comput. Technol. (ICICT)*, Jan. 2021, pp. 220–223, IEEE.

[8] S. Alani, S. N. Mahmood, S. Z. Attaallah, H. S. Mhmood, Z. A. Khudhur, and A. A. Dhannoon, "IoT based implemented comparison analysis of two well-known network platforms for smart home automation," *Int. J. Electr. Comput. Eng.*, vol. 11, no. 1, pp. 442–450, 2021.

[9] T. Reshamwala, C. Shah, and S. Naik, "Computerization in home: Change in way of life," in *Proc. 2021 2nd Int. Conf. Smart Technol. Comput. Electr. Electron. (ICSTCEE)*, Dec. 2021, pp. 1–6, IEEE.

[10] K. T. Swami and A. A. Moghe, "A review of LiFi technology," in *Proc. 2020 5th IEEE Int. Conf. Recent Adv. Innov. Eng. (ICRAIE)*, Dec. 2020, pp. 1–5, IEEE.

[11] A. Gupta, N. Zaidi, H. Kaushik, and P. Kumar, "Practical concepts and future implications of IoT: In perspective of India," in *Proc. 1st Int. Conf. Smart Comput. Informatics (SCI) 2016*, vol. 2, pp. 475–486, Springer, Singapore, 2018.

[12] D. Haripriya, K. Kumar, A. Shrivastava, H. M. R. Al-Khafaji, V. Moyal, and S. K. Singh, "Energy-efficient UART design on FPGA using dynamic voltage scaling for green communication in industrial sector," *Wireless Commun. Mobile Comput.*, vol. 2022, Art. ID 4336647, 2022.

[13] M. S. Soliman, A. A. Alahmadi, A. A. Maash, and M. O. Elhabib, "Design and implementation of a real-time smart home automation system based on Arduino microcontroller kit and LabVIEW platform," *Int. J. Appl. Eng. Res.*, vol. 12, no. 18, pp. 7259–7264, 2017.

[14] C. A. U. Hassan, J. Iqbal, M. S. Khan, S. Hussain, A. Akhunzada, M. Ali, et al., "Design and implementation of real-time kitchen monitoring and automation system based on Internet of Things," *Energies*, vol. 15, no. 18, p. 6778, 2022.

[15] S. More, S. Shelar, V. Randhave, and A. Bagde, "IoT based smart kitchen system," *Int. J. Sci. Res. Sci. Eng. Technol.*, Article 479485, 2021.

[16] G. Selvakumar, R. Kamesh, K. K. Deepa, and K. G. Mayurpranav, "Smart kitchen provisions monitoring system using Internet of Things," in *Proc. 2022 2nd Int. Conf. Adv. Technol. Intell. Control, Environ., Comput. Commun. Eng. (ICATIECE)*, Dec. 2022, pp. 1–4, IEEE.