Table of Contents

List of Figures

INTRODUCTION

The increase in demand for remote health monitoring solutions has accelerated the implementation of Internet of Things (IoT) systems in the health sector. This report presents the design and simulation of an IoT-based health monitoring system using Wokwi as the development environment and ThingSpeak as the cloud platform. The major objective is to monitor a patient's vital signs such as temperature; humidity; and pulse and provide real time feedback and alerts when values exceed safe thresholds. This system is aimed at supporting patients in home-based care environments and alert caregivers, when necessary, hence enabling faster interventions and improved health outcomes.

SYSTEM OVERVIEW

The simulated system leverages an ESP32 microcontroller board connected to a DHT22 sensor, a potentiometer (to simulate a pulse sensor), an I2C LCD screen, and a buzzer. Sensor data is continuously collected and displayed on the LCD, while the ESP32 communicates with the ThingSpeak cloud platform using MQTT. When the sensor readings exceed the preconfigured thresholds on ThingSpeak, the ESP32 triggers a buzzer and simultaneously prompts an IFTTT webhook to send an alert email to a caregiver's Gmail account.

This system is particularly beneficial in healthcare monitoring applications where the patient may be unattended to for extended periods of time. Through cloud-based configuration and remote alerts, the design ensures that safety and responsiveness while maintaining simplicity and affordability (Yassine *et al.*, 2019).

RESOURCES USED

Hardware Components (Simulated)

- ESP32 Microcontroller: This is the core controller used with built-in Wi-Fi for cloud communication.
- DHT22 Sensor: This measures temperature and humidity with higher accuracy than DHT11.
- Potentiometer: This is used to simulate analog input from a pulse sensor.
- I2C LCD (16x2): This displays sensor values in real time. The I2C version reduces the number of required pins and simplifies wiring.
- Buzzer: This acts as an output alert mechanism when thresholds are breached.

Software and Cloud Tools

- Wokwi Simulator: This is an online IoT simulation platform.
- ThingSpeak: This is a cloud-based IoT analytics platform for data logging, visualization, and threshold configuration.
- MQTT Protocol: This is used for lightweight publish/subscribe messaging between ESP32 and ThingSpeak.
- IFTTT (If This Then That): This integrates ThingSpeak with Gmail to send email alerts upon threshold violations.

SYSTEM ARCHITECTURE

The system's architecture includes sensor data acquisition, cloud-based threshold validation, alert generation, and real-time feedback via an LCD.

Figure 1 presents the architecture of the system in full.

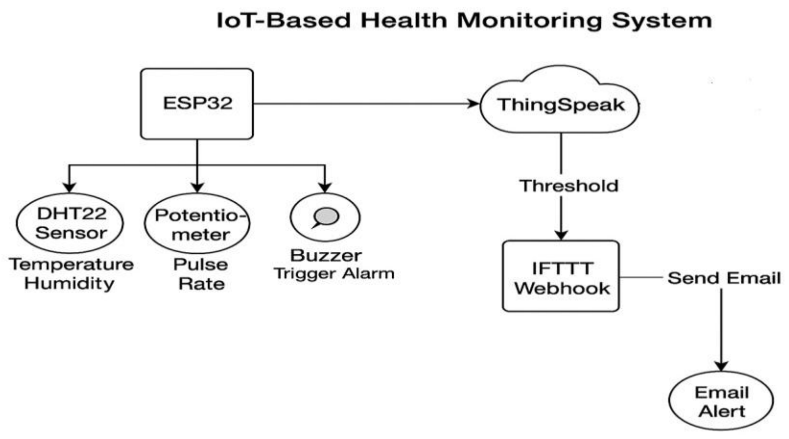


Figure 1 System Architectural Diagram

System Flow

1. The ESP32 reads the data from the sensors (DHT22 and Potentiometer).
2. The readings are displayed on the I2C LCD.
3. Data read from the sensors are published to ThingSpeak via MQTT.
4. Thresholds configured are retrieved from ThingSpeak.
5. If any reading exceeds the threshold:
    - The buzzer is triggered
    - An IFTTT webhook is invoked to send an email alert to the assigned caregiver.

PROTOCOLS USED

- Wi-Fi: This is used to connect the ESP32 to the internet, allowing seamless data transfer to the cloud. Wi-Fi is a preferred protocol due to its wide availability, ease of setup, and support for real time data transmission.
- MQTT (Message Queuing Telemetry Transport): MQTT is a publish/subscribe protocol best for low-power, bandwidth-sensitive IoT systems. In this project, the ESP32 publishes sensor data to ThingSpeak and subscribes to threshold fields for validation.
- I2C (Inter-Integrated Circuit): I2C is used to connect the LCD to the ESP32. Unlike parallel communication, I2C requires just one SDA (Data) wire and one SCL (Clock) wire, which reduces the complexity of circuit wiring and frees up GPIO pins for other sensors.

IFTTT- GMAIL INTEGRATION

The system includes a cloud-based alert mechanism through IFTTT (If This Then That). IFTTT webhooks are configured to respond to specific events from ThingSpeak. When thresholds are breached or get exceeded, ThingSpeak triggers a webhook to IFTTT, which sends an email alert to a preconfigured Gmail address. This ensures caregivers are notified instantly without the need for constant monitoring.

RESULTS

The system was successfully simulated using the Wokwi platform, with all components functioning as intended. The ESP32 consistently reads temperature, humidity, and pulse values, which were displayed in real time on the I2C LCD screen. Data was successfully published to ThingSpeak using the MQTT protocol, and thresholds configured in ThingSpeak were accurately retrieved and evaluated by ESP32.

Figure 2 shows the real-time reading of the sensors data on the I2C LCD.

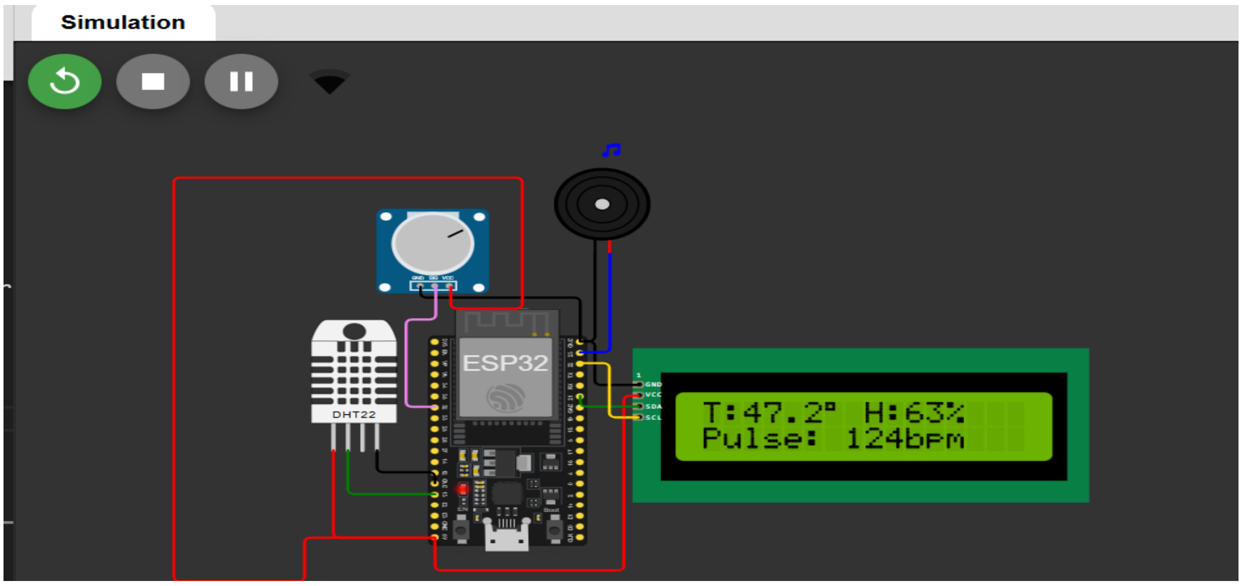


Figure 2 Real-time Readings from Sensors

Figure 3 shows the readings from the sensors and the preconfigured threshold fields on the ThingSpeak cloud platform.

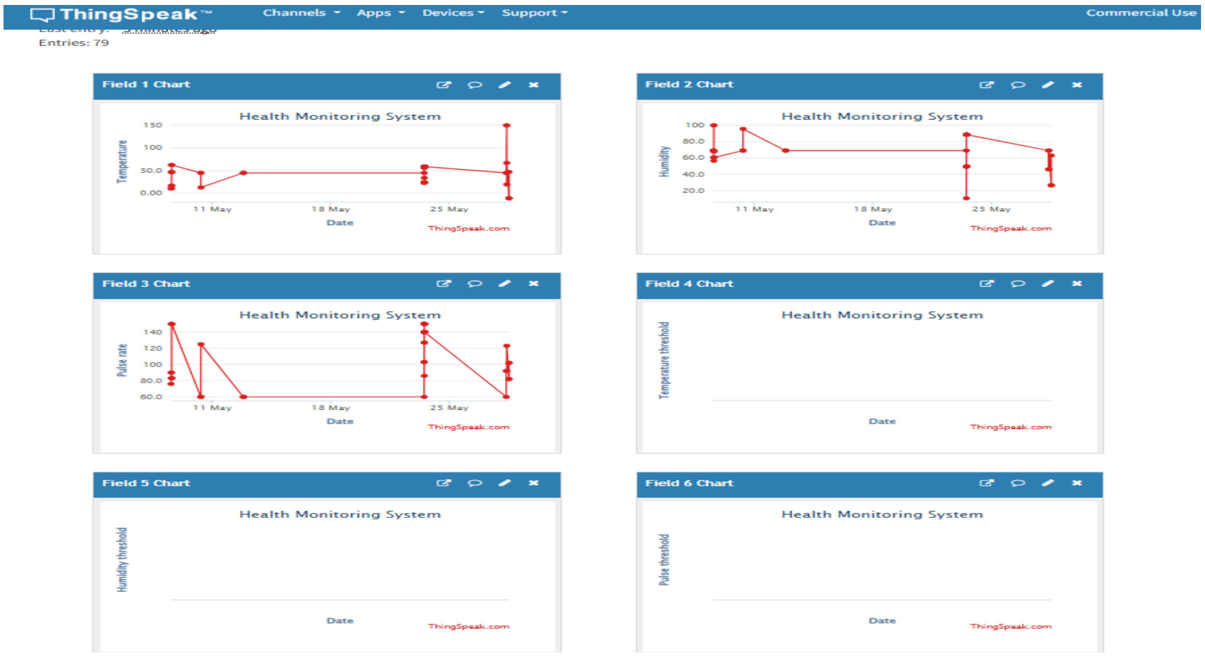
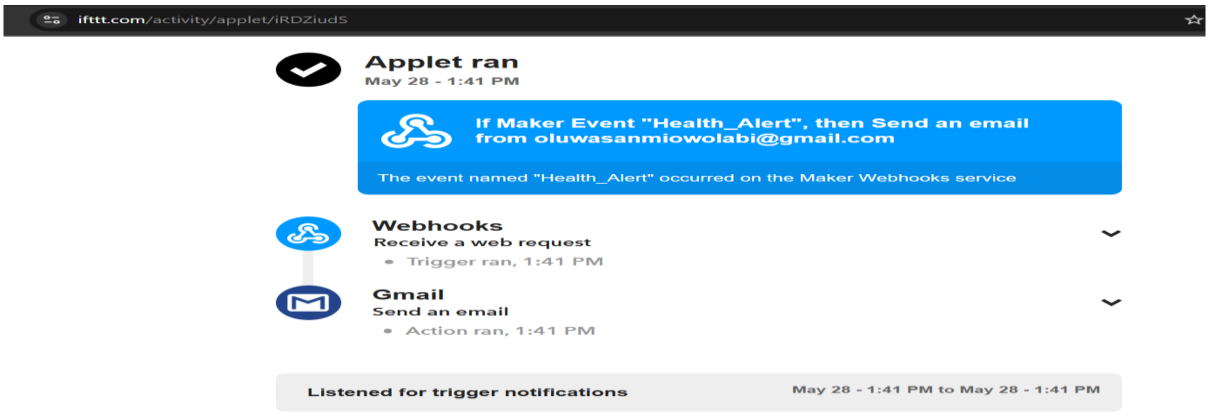


Figure 3 Temperature, Humidity, and Pulse reading and threshold field on ThingSpeak

During testing, when sensor reading exceeded the preset limits, the buzzer was activated immediately, and an IFTTT-triggered email alert was sent to the caregiver's Gmail account. The response time from threshold breach to alert delivery was under 5 seconds, confirming the effectiveness of real-time monitoring and alerting. These results demonstrate the system's ability to provide timely notifications and support responsive healthcare intervention.

Figure 4 shows the activity log of the IFTTT webhook applet that keeps record of all activity logs.



ifttt.com/activity/applet/iRDZiudS

Applet ran
May 28 - 1:41 PM

If Maker Event "Health_Alert", then Send an email from oluwasanmiowolabi@gmail.com

The event named "Health_Alert" occurred on the Maker Webhooks service

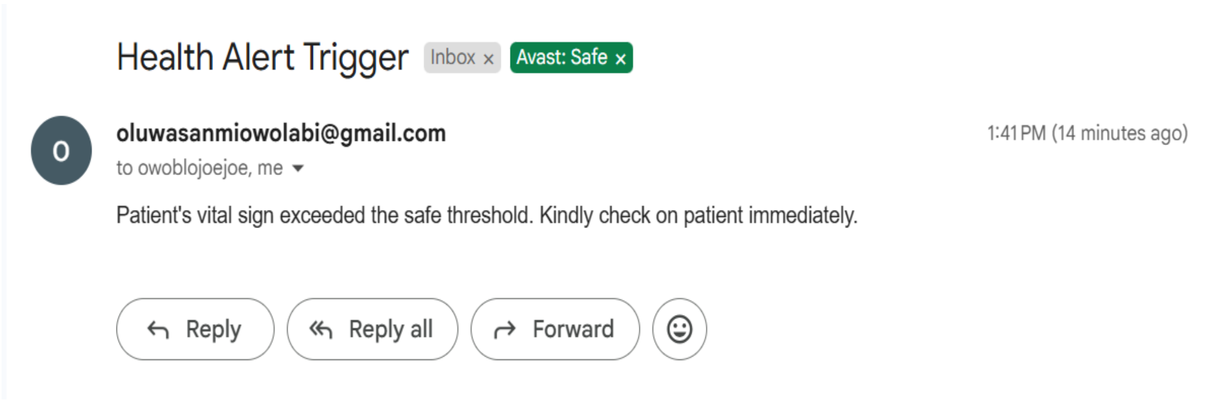
Webhooks
Receive a web request
• Trigger ran, 1:41 PM

Gmail
Send an email
• Action ran, 1:41 PM

Listened for trigger notifications — May 28 - 1:41 PM to May 28 - 1:41 PM

Figure 4 Activity Log of IFTTT Applet runs

Figure 5 shows the email trigger alert sent from the IFTTT webhook to the caregiver's Gmail address to inform of breach on the thresholds set for vitals.



Health Alert Trigger   Inbox ×   Avast: Safe ×

oluwasanmiowolabi@gmail.com   1:41 PM (14 minutes ago)
to owoblojoejoe, me

Patient's vital sign exceeded the safe threshold. Kindly check on patient immediately.

Figure 5 Email alert sent to configured Caregivers Gmail via IFTTT

DISCUSSION

System Benefits: The proposed design is lightweight, low-cost, and easily scalable. It enables real-time monitoring and remote notification, which are essential in both home and clinical care settings. The use of MQTT and ThingSpeak ensures efficient data handling and cloud control.

Challenges: While Wokwi provides an effective simulation environment, actual deployment may face the bellow challenges:

- Unstable internet connections
- Power supply interruptions
- Cloud service latency

Security and Privacy: Security is crucial in healthcare systems, although ThingSpeak and IFTTT offers basic protection, full deployment should consider the following:

- Data encryption during transmission.
- Authentication mechanisms for cloud APIs.
- Compliance with GDPR and HIPAA regulations for storing patient health data.

CODE BASE FOR SYSTEM DESIGN

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <ThingSpeak.h>

// WiFi credentials
const char* ssid = "Wokwi-GUEST";
const char* password = "";

// ThingSpeak settings
const char* writeAPIKey = "NGINM26X18PLIGA0";
const char* readAPIKey = "CU9T2S0XN1P5LFCA";
const char* channelID = "2953005";

// Sensor setup
#define DHTPIN 13
#define DHTTYPE DHT22
#define POTPIN 32
#define BUZZER_PIN 23

DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(115200);
  dht.begin();
  lcd.init();
  lcd.backlight();
  pinMode(BUZZER_PIN, OUTPUT);

  WiFi.begin(ssid, password);
  lcd.setCursor(0, 0); lcd.print("Connecting WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  lcd.clear();
  lcd.print("WiFi Connected");
  delay(1000);
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  int potVal = analogRead(POTPIN);
  int pulse = map(potVal, 0, 4095, 60, 150);

  if (isnan(temp) || isnan(hum)) {
    Serial.println("Sensor error");
    return;
  }

  // Display on LCD
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("T:"); lcd.print(temp, 1); lcd.print((char)223); lcd.print(" H:"); lcd.print(hum, 0); lcd.print("%");
  lcd.setCursor(0, 1);
  lcd.print("Pulse: "); lcd.print(pulse); lcd.print("bpm");

  // Send to ThingSpeak
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    String url = "http://api.thingspeak.com/update?api_key=" + String(writeAPIKey)
                 + "&field1=" + String(temp, 1)
                 + "&field2=" + String(hum, 1)
                 + "&field3=" + String(pulse);
    http.begin(url);
    int httpCode = http.GET();
     if (httpCode > 0) {
      Serial.println("Data uploaded to Cloud.");
    } else {
      Serial.println("HTTP error.");
    }

    http.end();
  }

  delay(15000); // Minimum update interval

  // Read thresholds from ThingSpeak Fields 4–6
  float tempThreshold = getThreshold(4);
  float humThreshold = getThreshold(5);
  float pulseThreshold = getThreshold(6);

  Serial.print("Thresholds → T: ");
  Serial.print(tempThreshold);
  Serial.print(", H: ");
  Serial.print(humThreshold);
  Serial.print(", P: ");
  Serial.println(pulseThreshold);

bool alert = false;
  if (temp > tempThreshold || hum > humThreshold || pulse > pulseThreshold) {
    digitalWrite(BUZZER_PIN, HIGH);
    tone(BUZZER_PIN, 100);
delay(1500);

  } else {
    digitalWrite(BUZZER_PIN, LOW);
    noTone(BUZZER_PIN);
  }

  delay(1500);
}

float getThreshold(int fieldNum) {
  HTTPClient http;
  String url = "http://api.thingspeak.com/channels/" + String(channelID) +
               "/fields/" + String(fieldNum) + "/last.txt?api_key=" + String(readAPIKey);
  http.begin(url);
  int httpCode = http.GET();
  float value = 1000; // default high value to avoid false buzzer
  if (httpCode > 0) {
    String payload = http.getString();
    value = payload.toFloat();
  }
  http.end();
  return value;
}
```

LINKS

Wokwi Simulator Link: https://wokwi.com/projects/430318952868273153

ThingSpeak Cloud Platform Link: https://thingspeak.mathworks.com/channels/2953005

CONCLUSION

The IoT-based health monitoring system presented in this report demonstrates the feasibility and effectiveness of using low-cost components and cloud services for real-time patient care. Through integration with ThingSpeak and IFTTT, the system provides remote alerting and continuous health monitoring, supporting caregivers in delivering timely medical assistance. The simulation in Wokwi verifies the functional logic of sensor acquisition, MQTT communication, cloud threshold comparison, local alerting, and email notification. Future work could expand the system to include GPS location tracking, mobile app integration, or voice-based alerts for visually impaired users.

REFERENCE

Yassine, A. *et al.* (2019) ‘IoT big data analytics for smart homes with fog and cloud computing’, *Future Generation Computer Systems*, 91, pp. 563–573. Available at: https://doi.org/10.1016/j.future.2018.08.040.