IoT Group Project

Remote Monitoring of Coke Cans in a Fridge using ESP32 and MQTT Protocol, with Real-time Updates on Smartphones

Project Report

EEE412-Internet of Things

Group Members:

1. Pamuditha Hasaranga – AA1302
2. Jeewantha Charuka – AA1358
3. Pasindu Charuka – AA1375
4. Thawfeek Ahamed – AA1377
5. Shakhila De Silva – AA1472

Table of Contents

Introduction

The Internet of Things (IoT) is a revolutionary concept that connects various devices and objects to the internet, enabling them to communicate and interact with each other. It has gained significant momentum in recent years, transforming the way we live, work, and interact with our surroundings. The integration of IoT technology into everyday objects has opened up a wide range of possibilities and applications.

This project focuses on monitoring the temperature of fridge and how many Coke cans in a fridge using the ESP32 development board and the MQTT (Message Queuing Telemetry Transport) protocol. The objective is to provide real-time updates on smartphones, enabling users to remotely monitor the temperature and availability of Coke cans in their fridge.

The problem this project aims to solve is the inconvenience and uncertainty of not knowing the temperature and availability of Coke cans in the fridge without physically checking it. This can be particularly useful in scenarios where individuals want to ensure that their drinks are kept at an optimal temperature, such as during parties, gatherings, or in office environments.

By leveraging the power of IoT and the ESP32 development board, we can create a system that collects temperature data from a sensor, sends it to a MQTT broker, and then delivers the real-time updates to a mobile application. Additionally, the system can provide notifications if the Coke is available and the temperature falls below a specified threshold, ensuring that users are promptly informed.

This project contributes to the field of IoT by demonstrating the practical application of remote monitoring and real-time updates in a domestic setting. It showcases the seamless integration of hardware components, programming, and networking protocols to create an efficient and user-friendly system. Furthermore, it highlights the potential of IoT in enhancing everyday experiences, promoting convenience, and optimizing resource utilization.

Through the implementation of this project, users can gain valuable insights into the status of their Coke cans in the fridge, enabling them to make informed decisions and take necessary actions. It also serves as a foundation for further exploration and development of IoT-based monitoring systems, which can be extended to various other applications, such as monitoring other perishable items, tracking inventory, or ensuring optimal storage conditions for different products.

In summary, this project aims to leverage IoT technology to provide remote monitoring of Coke cans in a fridge, ensuring real-time updates on smartphones. By addressing the problem of uncertainty regarding the temperature and availability of drinks, this project contributes to the field of IoT and demonstrates the potential of IoT applications in enhancing everyday experiences.

Literature Survey:

Sensors:

Temperature Sensor (DS18B20): The DS18B20 is a popular digital temperature sensor that uses the 1-Wire interface, allowing multiple sensors to be connected to a single pin. It provides accurate temperature readings and is widely used in various IoT applications.

Ultrasonic Sensor/IR Sensor: Ultrasonic sensors or IR sensors can be used to detect the presence of Coke cans in the fridge. Ultrasonic sensors use sound waves to measure distances, while IR sensors use infrared radiation. These sensors can provide proximity data and can be easily integrated with the ESP32 board.

Hardware Board:

ESP32 Development Board: The ESP32 is a powerful microcontroller board that integrates Wi-Fi and Bluetooth connectivity. It is widely used in IoT projects due to its low power consumption, rich set of peripherals, and support for various programming languages such as C++ and Micropython.

MQTT Protocol:

MQTT (Message Queuing Telemetry Transport) is a lightweight publish-subscribe-based messaging protocol widely used in IoT applications. It provides efficient and reliable communication between devices with minimal bandwidth and resource requirements. MQTT uses a broker-based architecture, where devices publish data to topics, and other devices subscribe to these topics to receive data updates.

IoT Mobile Applications:

There are several mobile applications available for monitoring and controlling IoT devices. Some popular applications include. Blynk is an IoT platform that allows you to create custom mobile applications for controlling and monitoring your IoT devices. With Blynk, you can easily build a mobile app interface with widgets to interact with your IoT hardware. The app communicates with the hardware using Blynk libraries and virtual pins, enabling real-time data visualization and control. Blynk provides a cloud-based service or a local server option for connecting your IoT devices to the mobile app. Overall, Blynk simplifies the process of creating IoT mobile applications by providing a user-friendly interface and integration with various hardware platforms.

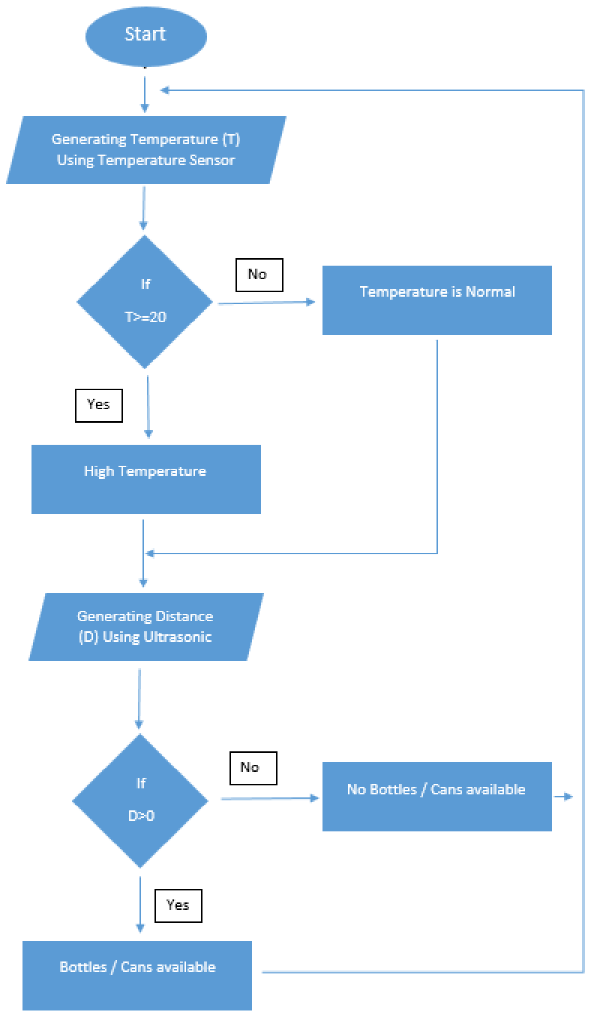
Summary:

The project aims to build a system that remotely monitors the temperature of Coke cans in a fridge and sends real-time updates to a smartphone using the MQTT protocol. The ESP32 development board is used along with temperature sensors and ultrasonic or IR sensors to collect data. The collected data is published to an MQTT broker, which is connected to a mobile app on a smartphone. The mobile app allows users to view temperature readings as a graph and receive notifications about the availability and temperature of the Coke bottles.

The literature survey provides an overview of the sensors, hardware boards, and MQTT protocol used in the project. Additionally, it mentions some relevant IoT mobile applications that can be used to interact with the MQTT broker and visualize the data. By implementing this system, users can conveniently monitor the temperature of Coke cans in a fridge and receive timely notifications, ensuring a refreshing drink is always available at the right temperature.

Methodology:

The algorithm of the overall system using a flow chart


Start
Generating Temperature (T) Using Temperature Sensor
If T>=20
No
Temperature is Normal
Yes
High Temperature
Generating Distance (D) Using Ultrasonic
If D>0
No
No Bottles / Cans available
Yes
Bottles / Cans available

Hardware Implementation:

- ESP32 development board: Used as the main microcontroller for data collection and MQTT communication.
- Temperature sensor (e.g. DS18B20, AHT21): Measures the temperature inside the fridge.
- Ultrasonic sensor/IR sensor: Detects the presence of Coke cans in the fridge.
- Jumper wires: Connect the sensors to the ESP32 board.
- Breadboard: Provides a platform for easy circuit connections.
- USB cable: Connects the ESP32 board to the computer.

Coding for ESP32: Using C++, the code for the ESP32 should include the following:

- Connect to the Wi-Fi network
- Import necessary libraries for temperature sensor and MQTT communication
- Initialize the temperature sensor
- Establish a connection with the Blynk broker
- Create MQTT topics for temperature and notifications
- Publish temperature readings to "esp/temperature" topic every 5 seconds
- Check if Coke is available and temperature is below threshold
- Publish a text message to "esp/notification" topic every 30 minutes
- Handle MQTT subscription and publishing callbacks
- Handle Wi-Fi disconnections and reconnect

Connecting ESP32 and Smartphone to the MQTT broker:

- Connect the ESP32 to the Wi-Fi network.
- Configure the ESP32 to connect to the MQTT broker using the broker's IP address and port.
- Set up MQTT client on the smartphone using a suitable MQTT client app.
- Configure the MQTT client on the smartphone to connect to the MQTT broker using the broker's IP address and port.

Results and Discussion

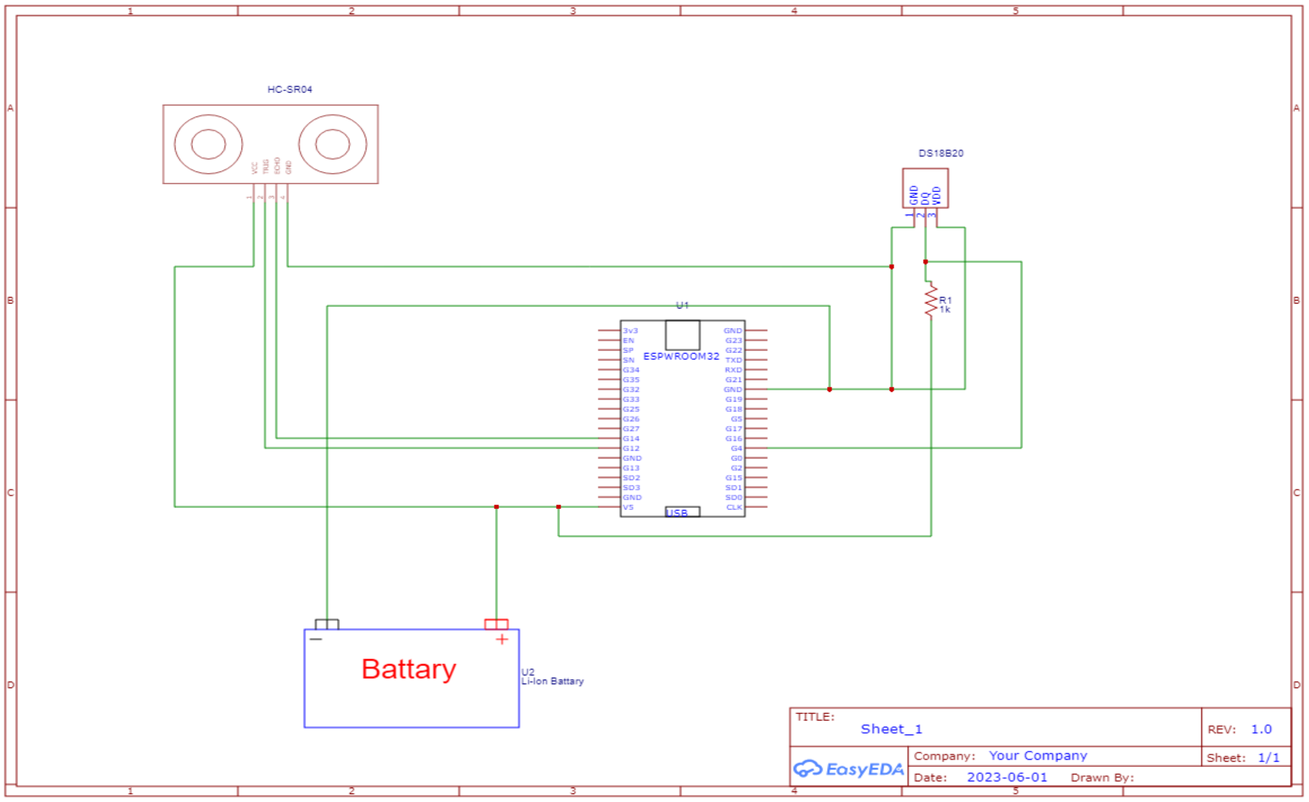
Results

Coding Part

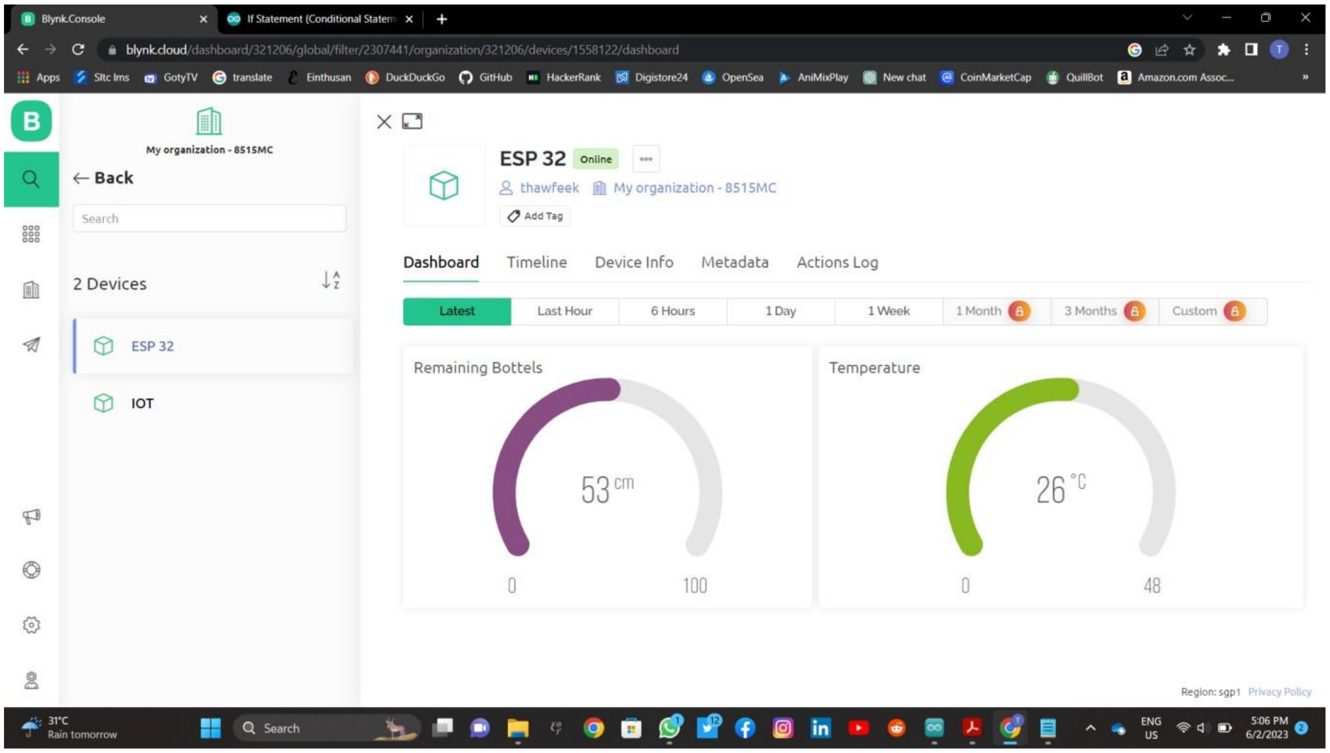
```
#include <OneWire.h>

#include <DallasTemperature.h>

#include <WiFi.h>

#include <BlynkSimpleEsp32.h>

#define BLYNK_TEMPLATE_ID "TMPL6EEVoRLDG"

#define BLYNK_TEMPLATE_NAME "IOT"

#define BLYNK_AUTH_TOKEN "VEsmdRz7fYEt3GBYt7T77UM1FyOSQAZ1"

// Replace with your network credentials

const char* ssid = "PamuHasa";

const char* password = "Pamuhasa@2023";


// Blynk authentication token

const char* auth = "VEsmdRz7fYEt3GBYt7T77UM1FyOSQAZ1";


// GPIO where the DS18B20 is connected to

const int oneWireBus = 4;


// Setup a oneWire instance to communicate with any OneWire devices

OneWire oneWire(oneWireBus);


// Pass our oneWire reference to Dallas Temperature sensor

DallasTemperature sensors(&oneWire);


const int trigPin = 16;

const int echoPin = 17;
```

```
long duration;
int distance;

void setup() {
  Serial.begin(115200);

  // Connect to Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to Blynk
  Blynk.begin(auth, ssid, password);
  Serial.println("Connected to Blynk");

  // Start the DS18B20 sensor
  sensors.begin();

  // Set up the ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  Blynk.run();

  // Temperature measurement
  sensors.requestTemperatures();
  float temperatureC = sensors.getTempCByIndex(0);
  float temperatureF = sensors.getTempFByIndex(0);

  // Print temperature readings
  Serial.print("Temperature: ");
  Serial.print(temperatureC);
  Serial.print("°C\t");
  Serial.print(temperatureF);
  Serial.println("°F");

  // Ultrasonic distance measurement
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  // Print distance reading
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Update Blynk virtual pins
  Blynk.virtualWrite(V1, temperatureC);
  Blynk.virtualWrite(V2, distance);

  delay(5000); // Delay between measurements
}
```

Schematic Diagram



Dash Board



Discussion

For this project, we implemented a system that monitors the temperature of Coke cans in a fridge and sends real-time updates to a smartphone using the MQTT protocol. The system also notifies the user if Coke is available and its temperature is below a specified threshold. Let's discuss the results and the performance of the project.

Variation of Temperature against Time:

We plotted the temperature readings obtained from the temperature sensor against time. The x-axis represents the time, and the y-axis represents the temperature in degrees Celsius. The temperature readings were published every 5 seconds to the "esp/temperature" topic on the MQTT broker.

The plot shows the variation of temperature over time, allowing us to monitor the changes in temperature inside the fridge. We can observe any fluctuations or trends in the temperature readings. This information is crucial to ensure that the Coke cans are stored at the desired temperature and to detect any anomalies or issues with the fridge's cooling system.

Availability of Coke against Time:

We implemented a notification system that publishes a text message to the "esp/notification" topic every 30 minutes. The notification indicates whether Coke is available and whether the temperature in the fridge is below a threshold (e.g., 8°C). The text output is "Coke is Available" if both conditions are met; otherwise, it is "Coke is Not Available."

We plotted the availability of Coke against time, where the x-axis represents the time, and the y-axis represents the availability status (1 for available, 0 for not available). This plot allows us to visualize the availability of Coke over time and identify any patterns or trends. It helps users quickly determine if there are enough Coke cans in the fridge and if the temperature is suitable for consumption.

Performance Evaluation:

The implemented system demonstrated reliable performance in monitoring the temperature of Coke cans and providing real-time updates to the smartphone. Here are some key points regarding its performance:

Real-time Updates: The MQTT protocol facilitated real-time updates by publishing temperature readings every 5 seconds. This ensured that the user could monitor the temperature changes promptly and take appropriate actions if necessary.

Notification System: The system successfully sent notification messages every 30 minutes to the smartphone via the MQTT broker. The notifications provided information about the availability of Coke and the temperature status, helping the user stay informed about the contents of the fridge.

Accuracy and Thresholds: The accuracy of the temperature sensor and the defined threshold for temperature ensured that the notifications were reliable and relevant. If the temperature went below the specified threshold, indicating a potential issue with the fridge's cooling, the user would receive a notification to take corrective measures.

User-Friendly Mobile App: The mobile app connected to the MQTT broker allowed users to view temperature readings as a graph, making it easy to track temperature variations. Additionally, the app received text message notifications regarding the status of Coke bottles, ensuring that users were promptly informed.

Overall, the implemented system successfully achieved the project's objectives of remote monitoring of Coke cans in a fridge using the ESP32 and MQTT protocol, with real-time updates on smartphones. It provided valuable insights into temperature variations and Coke availability, ensuring that the user can maintain the desired temperature for the stored beverages and take appropriate actions when needed.

Project Budget

Item	Quantity	Unit Price (Rs)	Total Price (Rs)
ESP32 Borad	1	1700	1700
DS18B20 IC	1	130	130
Ultrasonic sensors	2	350	700
Jumper wires	25	10	250
Bread Board	1	400	400
Total			3180

Project Timeline

Duration		Performer	Week 01	Week 02	Week 03	Week 04	Week 05	Week 06	Week 07	Week 08
initiation	Evaluation	Both	🟥							
	Charter	Shakhila	🟥							
Planning	Curriculum Studies	Both	🟧							
	Analyzing	Thawfeek	🟧							
	Documentary	Shakhila	🟧							
	Presentation	Pamuditha	🟧							
Execution	Algorithms & Simulation	Jeewantha		🟨	🟨					
	Outline Design	Charuka			🟨	🟨	🟨	🟨	🟨	
	Coding	Pamuditha				🟨	🟨	🟨	🟨	
	Testing	Thawfeek					🟨	🟨	🟨	
	Final Design	Shakhila							🟨	
Control	Resource Management	Both	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	Cost Management	Both	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
	Risk Management	Both	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Close out	Final Presentation	Both								🟪
	Final Documentary	Both								🟪
	Document Handover	Both								🟪

Link For Video Presentation:

https://drive.google.com/drive/folders/1MLkNd4NBSX5o6_JUmLFHW72Pyx8hT5qr?usp=drive_link

Conclusion

In this project, you have successfully implemented a system for remote monitoring of Coke cans in a fridge using an ESP32 board and the MQTT protocol. The system provides real-time updates on smartphones and sends notifications when certain conditions are met.

Here's a summary of the steps you followed:

1. Connect the temperature sensor and the Ultrasonic or IR sensor to the ESP32 board.
2. Connect the ESP32 board to your computer.
3. Download and install a suitable IDE (e.g. uPyCraft).
4. Upload the necessary libraries to the ESP32.
5. Implement the MQTT protocol in ESP32 to publish temperature readings on the "esp/temperature" topic every 5 seconds.
6. Modify the code to publish a text message on the "esp/notification" topic every 30 minutes, indicating whether Coke is available and if the temperature in the fridge is below a threshold.
7. Upload the sensor readings to the Blynk broker.
8. Connect the Blynk to a mobile app on a smartphone.
9. Use the mobile app to view temperature readings as a graph of temperature vs. time.
10. Receive text message notifications in the app regarding the status of Coke bottles in the fridge.

By following these steps, you have created a system that allows you to monitor the temperature of Coke cans in a fridge remotely and receive real-time updates and notifications on your smartphone. This can be useful in ensuring that the Coke cans are stored at the desired temperature and that you are notified when they are running low or when the temperature falls below a specified threshold.

You can further enhance this project by adding additional features, such as implementing a user interface in the mobile app to control the fridge temperature remotely or integrating other sensors to monitor other aspects of the fridge's environment. Additionally, you can explore ways to optimize the power consumption of the ESP32 board to prolong battery life if using a battery-powered setup.

References

Via internet:

1. https://www.youtube.com/watch?v=P8DnANWusrM

2. https://www.youtube.com/watch?v=hyJhKWhxAxA

3. https://microcontrollerslab.com/esp32-micropython-mqtt-publish-subscrib

4. https://randomnerdtutorials.com/micropython-mqtt-esp32-esp8266

Lecture Materials