SRI LANKA TECHNOLOGICAL CAMPUS
ශ්‍රී ලංකා තාක්ෂණික විශ්වවිද්‍යාලය
இலங்கை தொழில்நுட்ப பல்கலைக்கழகம்

School of Engineering

IoT Based Beverage Can Monitoring System

Project Report

Prepared by:

AA1378 - H.L.Adiththa Imasha

AA2129 - J.C. Kehelwatta

AA1424 - Nimesh Mendis

AA1416 - Nawon Perera

AA2122 - Thesara Wickramaarachchi

June 2023

Contents

List of Figures

1. Introduction

1.1 Background

The Internet of Things (IoT) refers to the interconnected network of physical devices, vehicles, appliances, and other objects embedded with sensors, software, and connectivity capabilities. These devices can collect and communicate data, enabling them to interact with each other and with humans, creating a seamless and intelligent ecosystem. In recent years, the popularity of IoT has increased exponentially. This can be attributed to the increasing availability of affordable and reliable connectivity options, advancements in sensor technology, and the growing demand for automation and data-driven decision-making. IoT has found applications in various sectors, including smart homes, industrial automation, healthcare, agriculture, and transportation, transforming the way we live and work. Its potential to enhance efficiency, improve productivity, and create new business opportunities has fueled its rise in popularity and made it a significant driver of technological innovation.

1.2 Problem statement

When selling cold beverage cans, employees have to open the storage refrigerator in order to verify availability and the coolness of the cans. The repeated opening of refrigerators results in more power usage which leads to increased electricity bills. Furthermore, it wastes the time of employees, leading to an inefficient workflow. All of this points to a need for a system to monitor the beverage cans in a refrigerator remotely.

2. Aims and Objectives

2.1 Aim

To design, develop and implement an IoT based system to monitor the status of beverage cans in a refrigerator remotely through a mobile application.

2.2 Objectives

1. Develop a system to gather data about temperature and availability of beverage cans inside a refrigerator.
2. Implement MQTT protocol to view the gathered data through a mobile app.
3. Add option for displaying the exact number of cans remaining in the container.
4. Add option for viewing status changes inside the container in relation to time.

3. Literature Survey

3.1 Hardware board

The ESP32 was selected as the microcontroller for this project. It was selected for its dual-core processor, enabling multitasking and efficient handling of tasks and the the built-in Wi-Fi support, which allows for seamless integration with networks and communication using MQTT protocol between devices and mobile apps. Also, its power-saving features optimize energy consumption for battery-operated devices. The ESP32 offers a rich set of interfaces, including SPI, I2C and GPIOs, for easy connection with the required sensors. It also allows high security with features like secure boot, encryption, and secure connections. Lastly it is compatible with popular IDEs, and cheap in cost making it an ideal choice for IoT projects.

Figure 3.1: ESP32 microcontroller

3.2 Sensors

3.2.1 Temperature sensor

The DHT11 temperature and humidity sensor compatible with the ESP32 microcontroller. The specifications are as follows:

- Operating Voltage: 3.5V to 5.5V.
- Operating current: 0.3mA (measuring) 60uA (standby).
- Output: Serial data.
- Temperature Range: 0°C to 50°C
- Resolution: 16-bit
- Accuracy: ±1°C

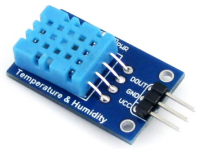

Figure 3.2: DHT11 temperature sensor

3.2.2 Ultrasonic sensor

HC-SRO4 is a ultrastonic sensor capable of measuring distance in non-contact method. It is also capable of an analog measurements, making it suitable for counting the number of coke cans in a container.

- Operating voltage: +5V
- Operating Frequency: 40Hz
- Operating Current: ¡15mA
- Measuring angle covered: ¡15°
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm

- Temperature Range: 0°C to 50°C
- Resolution: 16-bit
- Accuracy: 3mm

Figure 3.3: HC-SRO4 ultrasonic sensor

3.3 MQTT Protocol

MQTT protocol is a standards based messaging protocol for machine to machine communication. It is lightweight and is based on the publish-subscribe model [1]. It is useful for operating in a low network bandwidth environment. A MQTT broker is required for implementation of the MQTT protocol. For this project the Mosquitto by Eclipse was used as the broker. Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol versions 5.0, 3.1.1 and 3.1.

3.4 Mobile application

The android application "IoT MQTT" panel was used to subscribe to the topics published by the ESP32. The configuration of the app is discussed in the next chapter.

4. Methodology

4.1 Algorithm of system

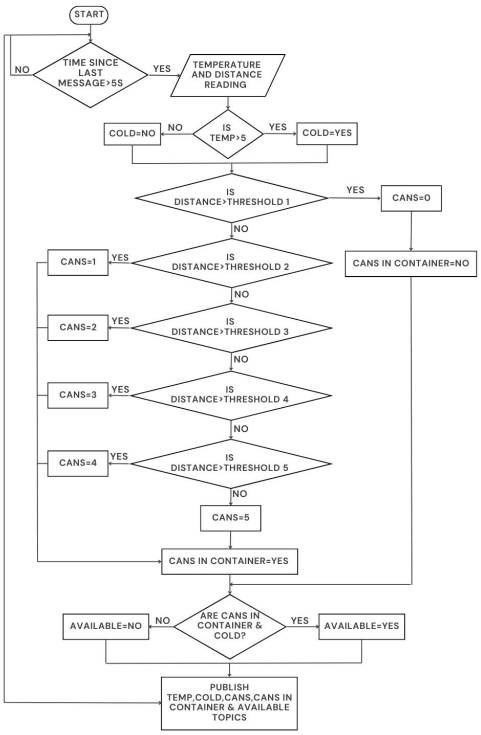

Figure 4.1: Algorithm of System

4.2 Hardware configuration

4.2.1 Selected sensors

This section will give an overview of why these specific sensors were selected. For the full specifications of sensors refer section 3.2 .

DHT11

The DHT11 sensor module can measure temperature and humidity and output in a 16 bit format. It was selected for the high accuracy of ±1°C and the supported temperature range of 0°C to 40°C comfortably accommodating the project requirement. Furthermore, it interfaces easily with ESP32 and has a companion library for easy micropython coding.

HC-SR04

The HC-SR04 ultrasonic sensor can measure distance to the cans up to 80cm. It was selected for the project because it offers an analog reading of the distance, which is essential for the additional objective of counting the exact number of cans in the system. An external library allows for output in the unit of centimetres which allows for easy measurement of the number of cans.

4.2.2 Circuit of the system

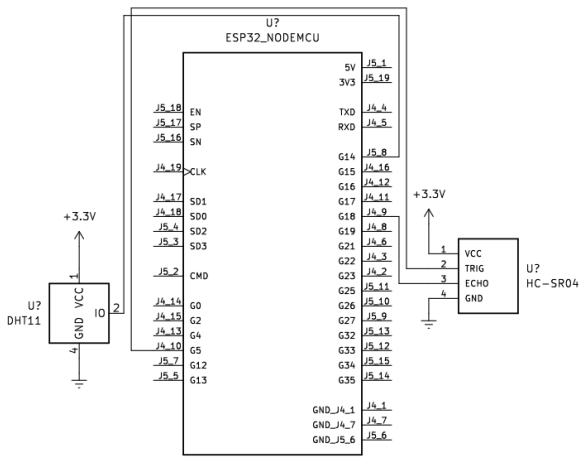

Figure 4.2: Circuit of System

4.2.3 Physical structure

Figure 4.3: Physical structure

4.3 Coding the ESP32

The python scripts were saved to the ESP32 using Thonny python IDE.The following external libraries were installed.

- dht - For the temperature sensor
- hcsr04 - To get readings from ultrasonic sensor in centimetre format. [2]
- umqttsimple - For connecting to MQTT broker and publishing topics

4.4 Implementing MQTT protocol

The MQTT protocol was used to enable communication between the ESP32 and the mobile application. All the topics are published by the ESP32 while the application subscribes to them. For the MQTT broker, Mosquitto by Eclipse was implemented. Eclipse Mosquitto is an open source (EPL/EDL licensed) message broker that implements the MQTT protocol 5.0, 3.1.1 and 3.1. For the purpose of this project, the free test server provided by Eclipse was utilized. Authentication was done using a Username and Password. When releasing this product in a

commercial environment the broker can be easily hosted in in-premise or in the cloud.

4.5 Configuring mobile app

The android app Iot MQTT panel was used to subscribe to the topics that are published by the ESP32. First the connection to the MQTT broker is configured defining the broker IP, port and authentication. The application offers a dashboard with a variety of options for displaying information. For the purpose of this project, gauges, multi state indicators, text logs, progress bars and graphs are used.

The main displays of the app are:

- Temperature gauge
- Can availability display
- Can count display with capacity bar
- Amount of cans history graph
- Temperature history graph
- Time display showing last update time

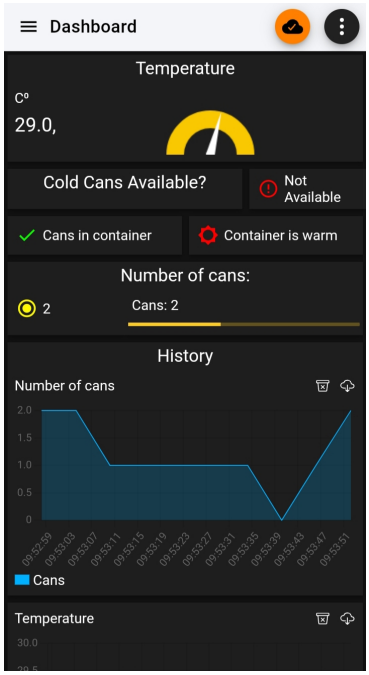

Figure 4.4: Mobile App

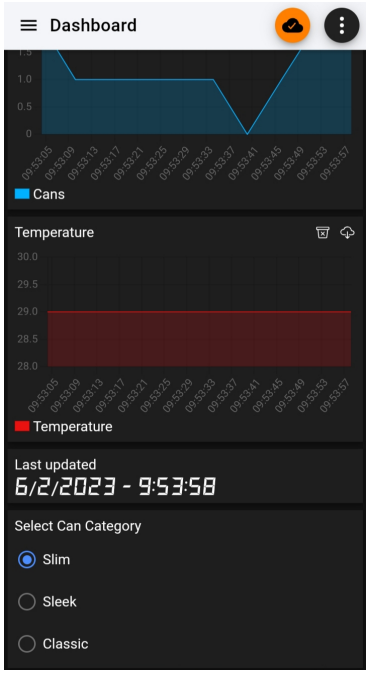

Figure 4.5: Mobile App (Scrolled down)

5. Results and performance

5.1 Results

All the required project objectives were successfully reached and the following additional outcomes were added as well.

- Counting the exact number of cans in container.
- View history of the number of cans.
- Give the user option to switch between can sizes of slim, sleek and classic

5.2 Performance

- Accuracy of temperature readings: ±1°C
- Accuracy of can number counting: 100%
- Quality of service: QoS level 2 by default (Can change through app)

5.3 Plots of the data

5.3.1 Temperature

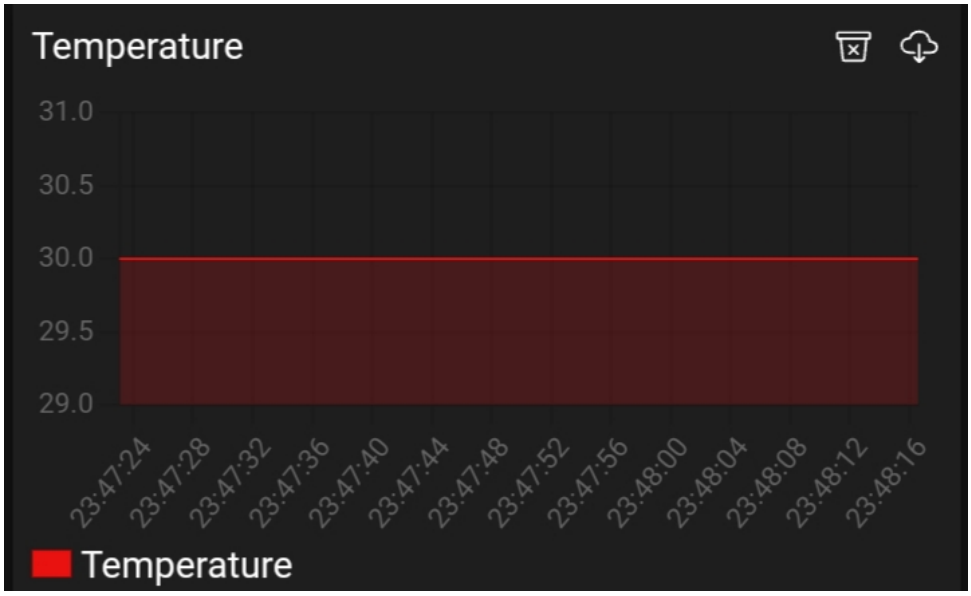

Figure 5.1: temperature graph at room temperature

5.3.2 Number of cans

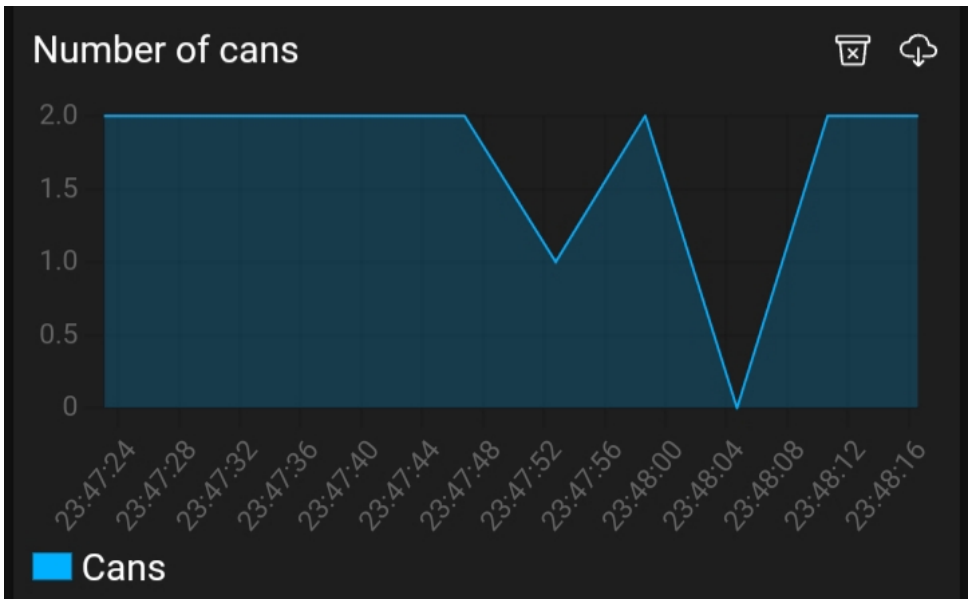

Figure 5.2: Number of cans over time

References

[1] Guilherme MB Oliveira, Danielly CM Costa, Ricardo JBVM Cavalcanti, Josiel PP Oliveira, Diego RC Silva, Marcelo B Nogueira, and Marconi C Rodrigues. Comparison between mqtt and websocket protocols for iot applications using esp8266. In *2018 Workshop on Metrology for Industry 4.0 and IoT*, pages 236–241. IEEE, 2018.

[2] GitHub - rsc1975/micropython-hcsr04: Micropython driver for ultrasonic sensor HC-SR04 — github.com. `https://github.com/rsc1975/micropython-hcsr04`. [Accessed 02-Jun-2023].