Submitted by

Group G:

Bhavesh, Kasif Raza, Ammar Khan

28.05.2024

Advanced Embedded Systems Report

Bachelor of Engineering - Electronic Engineering

HOCHSCHULE HAMM-LIPPSTADT

Professor: **Prof. Dr. Ali Hayek**

TABLE OF CONTENTS

Chapter 1

THE TEAM

1.1 Members

The team is composed of three members:

- Bhavesh
- Ammar Imran Khan
- Kashif Raza

The members have worked previously in the 4th semester Prototyping project, with Prof. Dr. Henkler, building upon that team relationship, we have recognised our individual strengths and divided the tasks accordingly; The balance of learning everything whilst someone takes the lead in a particular topic is important.

- **Bhavesh**: Primarily responsible for documentation and team management, mainly responsible for presentations and encouraging team members to continue to develop further in the project.
- **Ammar Khan**: Responsible for coding, bringing alot of technical knowledge of networks and micro-controllers to the team, being able to recognise what kind of parts we would require and would be able to implement within the given time-frame.
- **Kasif Raza**: Designing SysML and UML diagrams, 3D modelling and other schematics, assisting in both ideation and coding, a strong support role within the team, essential to getting the project completed with the given time-frame.

Whilst members do have individual duties, everyone is learning from each other about all topics that are involved within the project, and the needs that need to be realised. The team functions by sharing knowledge from our individual tasks, we do not hesitate from teaching each other concepts if the other has asked for it. We collectively increase each others skill-set, whilst taking charge in fields where our individual strengths shine.

1.2 Topic Introduction

The project this semester primarily deals with Internet of Things (IoT). In order to understand how IoT works, we need to first understand what distributed systems are.

- **Distributed Systems**: A distributed system is a collection of independent computer systems that appear to the end-user as a single coherent system.

  The idea is to introduce a form of middle-ware between the end-user and the technical back-end, in order to compensate for the increasing complexity of electronic systems that usually operate with multiple micro-controllers and computing processors, middle-ware provides that level of abstraction that allows the end-user to easily use the system as a collective rather than interfacing with each processor and micro-controller individually.

- **Internet Of Things**: IoT expands the concept of Distributed Systems to the largest distributed system ever created, which is the internet. The internet enables to connect virtually an infinite number of nodes together into a single coherent system, thankfully enabled with the combination of IPV4/IPV6 addresses, which would be very difficult to reach the limit of.

  The internet being a network of networks, is incomprehensibly large, hence it is important to specify unique addresses and services like the Domain Naming Service (DNS), to organise all the information available effectively. Everyday distributed systems are enabled further by the internet through ports. There are special standardised protocols on the Transport Layer of the OSI model, that enable all different sorts of distributed systems to communicate together. An example of such a transport protocol is MQTT, with the port number 1883; by assigning a network this port number, we are able to establish and communicate on a medium-range, low bandwidth disributed system.

  In general, different communication protocols can enable various types of networks. From low range and low bandwidth, to high range and high bandwidth.

- **Wireless Sensor Networks**: Wireless Sensor Networks (WSN) can be seen as a subset of IoT, whereas in IoT, we are sending information to the internet, if we aim to establish a WSN, we do not nessecerily have to connect it to the internet.

A central node, takes the role of organising the data that is published by sensors within this network, and actuators can request to gather this data from this central node.

These systems differ from large data networks in that they are primarily used for applications which require not a large amount of processing power, and transmit a lower bandwidth than a server on the internet would, for example.

Many use cases, such as farming monitor do not require access over the internet and could be restricted to a Local Area Network (LAN), WSN provides the tools to implement this kind of functionality to a Distributed System.

Project Application

Now that basic theory behind IoT systems has been covered, we move onto explaining what our project is.

In order to demonstrate the effectiveness of a distributed system, we have decided to implement a WSN. Many areas of our everyday lives could be significantly improved with the usage of small-scale systems that simplify tedious tasks, even eliminating potential human error.

The project is the implementation of a Smart Restaurant System. We aim to increase the efficiency of management with restaurants, by having the possibility to control and monitor various functionalities commonly manually done within a restaurant.

The basic needs of our project are:

- Low-Bandwidth data transfer
- Short-Range communication (20 meters)
- Monitoring sensors
- Interfacing with Customers

Based off of these basic needs, the protocol we are going to use is MQTT. As the publisher/subscriber model of this protocol allows us to have a central broker which can manage the data and provide feedback within a short time-span; The system is only concerned with transmitting data no bigger than 2 bytes, which the protocol is more than equipped to handle. MQTT enables us to communicate over distances larger than 20 meters, which would severely reduce any latency issues we might encounter with the deployment of the project.[1]

In order to monitor the environment and interact with the customers, we will use two publishers that will interface with respective sensors, publishing that data to our main broker, the central component controlled by the staff in the restaurant. This data will then be shared to two subscribers, one with the chefs, the other with the database of the restaurant.

Chapter 2

CONCEPT

2.1 Concept

The basic block diagram overview of our project is shown in Fig.2.1:

1. **Customer Publisher:** Sends 4 topics to the broker. 2 topics are for the Customer Subscriber, whilst the other 2 are for the Backend Subscriber. This Publisher is responsible for interacting with the customer, they can place the order through it.

   Button Status: Sent to chef subscriber, checking for request of human assistance.

   Button Response: Sent to customer subscriber, acknowledge response showcased to customer.

   Order Status: Sent to chef subscriber, checking for if order is placed.

   Order Response: Sent to customer subscriber, acknowledging that their order is placed or is invalid.

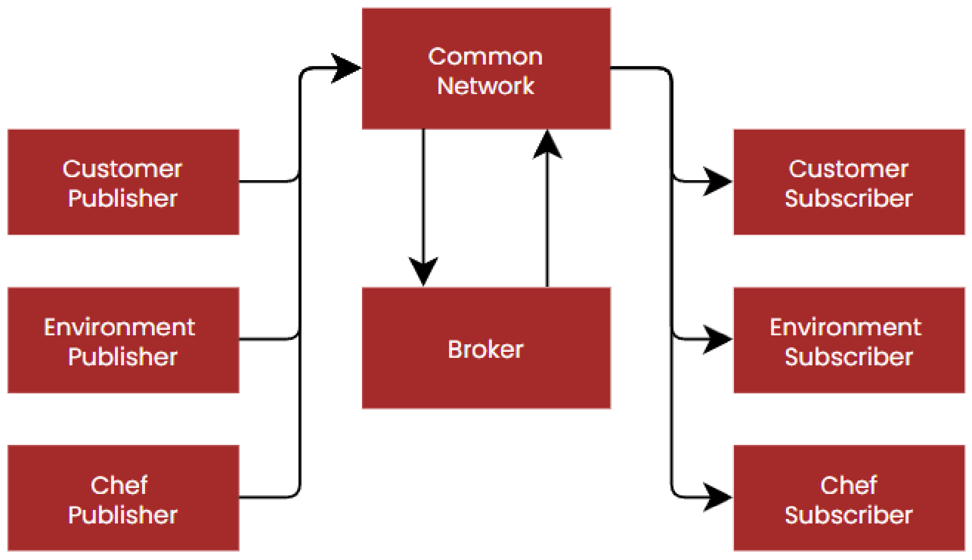

Figure 2.1: Overview

2. **Environment Publisher:** Sends 4 topics to the broker. 2 topics are for the Customer Subscriber, whilst the other 2 are for the Backend Subscriber.

   Sensor/Flame: Publish flame sensor's analog value.

   Sensor/Light: Publish light level from photoresistor.

   Sesnsor/Temperature: Publish temperature reading

   Sensor/TableStatus: Publish status of table, indicating weather a specific table is reserved or available.

3. **Chef Publisher:** Sends 1 topic to the broker. This publisher's job is to act as a form of communication from the Chef to the customer.

   Order Completion: Sent to customer subscriber, to acknowledge if an order has been completed or not.

4. **Broker:** Stores topics within a database, responsible for maintaining connection between Publishers and Subscribers.

5. **Customer Subscriber:** Subscribes to 3 topics. Responsible for outputting relevant data to the customer. Such as acknowledging their request and orders. Information from the chef about if the order is done cooking or not is also relayed here. This subscriber was implemented on a MQTT app for android.

6. **Backend Subscriber:** Subscribes to 6 topics. This Subscriber acts as a form of information gatherer for the restaurant management team. It is ran on a dashboard, connected to the web. It outputs relevant information about the environment, receiving data from the Environment Publisher. And also data regarding if the customer has somehow interacted with the system.

2.2 Application

Whilst we are designing a project specifically for restaurants, the main goal is to showcase the effectiveness of introducing distributed systems, especially WSN within everyday tasks, in order to more effectively control them; Data is one of the easiest ways to isolate problems and inefficiencies within any application.

IoT has enabled the possibility of creating distributed systems on an extremely large-scale, to the scale of cities, however this does not necessarily mean that funding will be diverted to implementing this change, even if we have the technology. Through demonstrations and showcasing of projects, we are able to promote this technology further. The first step would be to tackle smaller scales, before deciding to implement something on the level of cities.

The prototype we are implementing, with a certain level of abstraction, could be applied to many other use-cases, as ultimately, most applications require a form of monitoring, and gathering data, deciding on how to interpret that data is the main form of carrying out a task and service. Through abstractin, we can reasonably arrive to use-cases such as:

- Smart Home Systems
- Smart Farming
- Smart Management

2.3 Components

These are the two main use cases through which customers interact with a restaurant. In order to realise these use cases, the components needed to realise this system are as follows:

Customer Interaction

- **Sensors**:
  1. **KY-004-Button**: This will enable the customer to request assistance of a human if they are ready to place an order. Unique numbers will be assigned to each table node, in order to differentiate which customer requested for assistance. It is better than conventional system where a waiter may be confused as to which table is ready to order.
  2. **3x4 Keypad**: The customer is able to place an order using the Keypad, by pressing one of the buttons they can send a message to the restaurant on what they would like to order, eliminating the need of a middle-man.
- **Actuators**:
  1. **KY-016-RGBLED**: This LED will allow the customer and the restaurant staff to immediately read the status of each table. The functionalities are broken down as such: Red indicates an order has yet to be placed, Yellow indicates that the order is being processed, Green indicates the table has been served.[5]

- **Controller**:

    1. **Arduino-UNO-WiFi-Rev2**: A controller that is readily available whilst also not being expensive, it is easy to program with, as it has a vast number of libraries and interfaces. It has the ability to communicate over a network, which makes it a great controller to pick for this use case.[2]

Environment Monitoring
Sensors:

1. **KY-015-Combi**: This sensor is multi-functional in that it allows us to monitor both Temperature and humidity of the immediate environment, this data could be fed into an airconditioning system that will adjust the ambient temperature as required.
2. **KY-018-Photoresistor**: The photoresistor will allow us to measure the ambient light level within the immediate vicinity of the table, upon request, a customer could request to get it adjusted as-well, we could tune the lighting based on the weather conditions outside.
3. **KY-050-Ultrasonic**: This will allow us to quickly gauge whether a table is currently occupied or not, by feeding information that the sensor detected an individual few meters from itself.

Actuators:

1. **KY-019-Relay**: The relay will be used to interface with the lights in order to adjust their brightness levels.

Controller:

1. **Arduino-UNO-WiFi-Rev2**: A controller that is readily available whilst also not being expensive, it is easy to program with, as it has a vast number of libraries and interfaces. It has the ability to communicate over a network, which makes it a great controller to pick for this use case.

Chapter 3

PROJECT MANAGEMENT

3.1 Project Methods

The methodology that most suited our team's working environment was an agile-form of development. As we had to continuously deal with new tasks within the university as they were being assigned to us weekly, one individual may decide to focus on one module more than the other, so tasks were divided based off of availability rather than hard strict-coded deadlines. Weekly meetings and discussions were held to push for more progress on the project.

The main challenges we encountered were having to build motivation at certain parts of the week where workload was heavy, but through collaborative encouragement and talking to individuals 1-on-1, we were able to overcome some of the motivation issues and move forward with our project within the confined deadlines we had originally planned.

3.2 Breakdown

The tasks were broken down as follows:

- **Phase 1: Documentation and Conceptualisation**

  The initial phase consisted of team members educating themselves on the concepts of MQTT and studying how IOT systems interact. In the initial phase, Bhawish was responsible for creating the concept that could be implemented, Kashif was responsible for assisting in conceptualisation and thinking of the design of the final design. Ammar was responsible for setting up the hardware components and giving feedback on what would be possible to do within the defined time constraints.

- **Phase 2: Node Implementation**

  Phase 2 was after we had studied enough about MQTT to get familiar with the theory, being able to implement the publisher nodes and the broker. Bhawish was responsible for the Customer Node. Kashif was responsible for the Environment Node. Ammar was responsible for the Broker on the Raspberry Pi. We had our individual responsibilities like creating header files and explaining concepts to each other from what we had learned during our implementations.[4]

- **Phase 3: Compilation of Final System** After the broker had been established and information could be published to it, we moved onto programming our individual subscribers and setting up the the final design of the system. Bhawish was responsible for designing a box for his customer publisher and programming the subscriber for his node. Kashif was responsible for helping to program the backend subscriber, alongside designing a box for his node. Ammar was responsible for creating the dashboard that is used by the backend subscriber.

It is important to emphasise, that we were all learning off of each other and nobody was left to do a task alone if they were struggling with it, we assisted each other in order to reach a final product in the end.

3.3 Team Roles and Tasks

Bhawish:

Mainly for management role, keeping the team up with the workload for weekly tasks and getting everyone to work together, covering responsibilities in programming and designing wherever needed. Responsible for documentation and presenting.

Kashif:

Designer/Hardware support role, provided with insight regarding alot of the availability of components, contributed to making the project alot more practical in application by suggesting the smart restaurant idea. Contributed to programming of environment publisher and designing of boxes to hold said publishers.

Ammar:

Development role, provided the team with realistic implementation details regarding the programming of publishers and brokers, was mainly guiding for which components could be easily secured and how difficult it would be to program them. Provided alot of insight into the working of Raspberry Pi and MQTT Brokers[3]. Designed the dashboard which acts as our backend subscriber.

Regardless, everyone did everything the other team member couldn’t, we did not define strict roles, team members were encouraged to help each other often and cover content in the weekly lab classes.

Chapter 4

TECHNOLOGIES

4.1 Sensors

The sensors and actuators we used are described in Section 2.3. These sensors are mainly single purpose and are only for the purposes of being used within student labs. Meaning they do not have a high lifespan and are often prone to errors due to their physical construction. We did encounter some errors with the 3x4 Matrix Keypad in that it tends to have alot of button bouncing, however we were able to counteract this with a form of delay.

4.2 Communication Protocol

MQTT was used as the main communication protocol for our project. We did have the possibility of using Bluetooth or WiFi Low Power, however the MQTT was selected as the best protocol due to it's ease of access within alot of programming environments, Bluetooth was also as convenient to use, however transferring strings over Bluetooth proved to be challenging, and the Broker/Publisher model of MQTT is far superior for the data transferring we had to do.

The Broker/Publisher model helps in that it allows us to seperate our project in nodes, that individually either send, store or receive data. These nodes can then be interconnected through MQTT with very intuitive and easy to understand libraries for existing hardware within our lab.

4.3 Programming Language

Mainly C and C++ due to the team member's familiarity with the programming language, it also provided the option to separate alot of code functionality into header files, allowing for sleeker code design.

Python was necessary in order to program the broker, Unix is also used within Raspberry Pi in order to control it, so team members gained experience in using Unix alongside Python, however most of the implementation was done within C/C++.

Chapter 5

IMPLEMENTATION

5.1 Overview

Within our implementation phase, we decided to add a 3rd Publisher in the form of a Chef Publisher, that ultimately wants to send information via the broker to the customer subscriber that their order is ready or not.

We are using two main publishers, one for interaction with the customer, the other for monitoring the environment, our prototype will restrict itself to using the most readily available sensors so the most amount of users are able to realise the system and test it for themselves. A Raspberry Pi Broker will be used, with two subscribers, one using a custom dashboard via HTTP, one using the MQTT application on mobile devices.

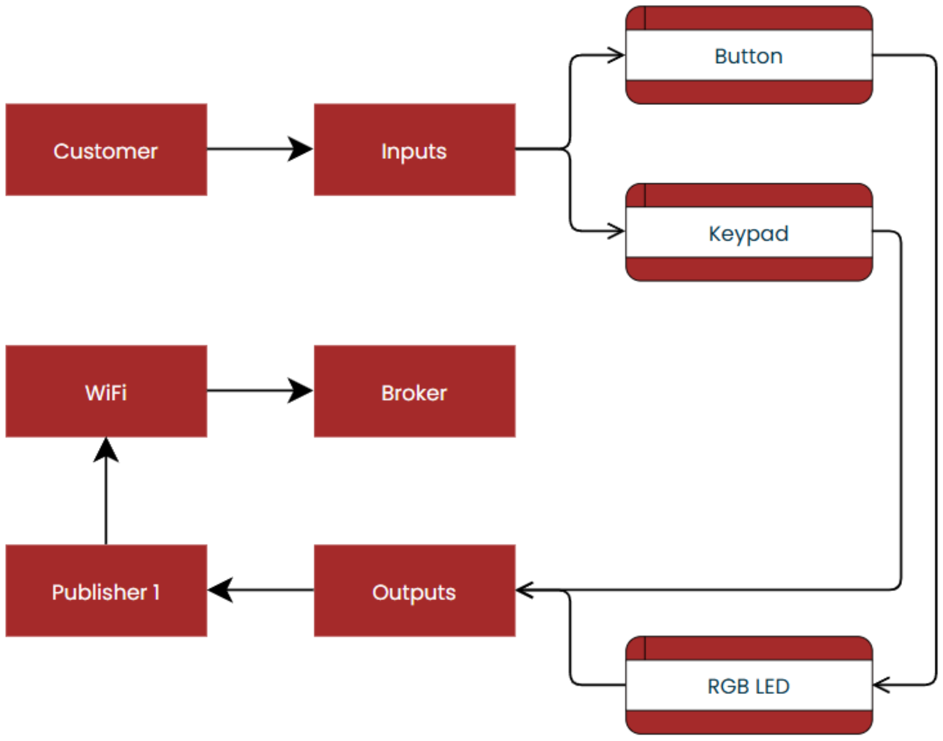

Figure 5.1: Customer Node Block Diagram

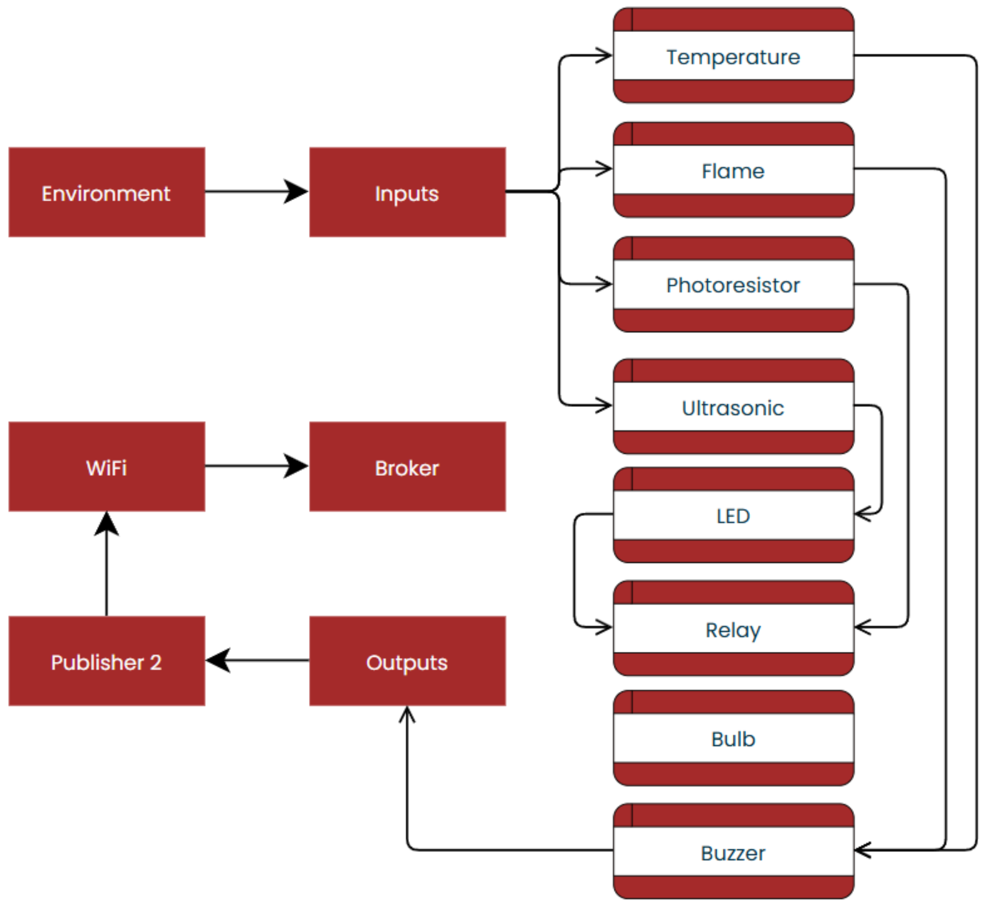

Figure 5.2: Environment Node Block Diagram

Customer Interaction

This subsection covers what the publisher that is responsible for interacting with the customers, does. Fig 5.1 showcases the basic block diagram for this node, it will be subject to changes depending on the testing and development iteration phase.

The main tasks that the publisher needs to handle is, being able to interface with the customer, we will model the use cases for our customer as the following:

- Placing the order
- Check order status

Environment Monitoring

This subsection covers the publisher is constantly monitoring the environment around a section of restaurant, in order to create a stable, pleasant environment for the customers to enjoy sitting in. Fig 5.2 showcases the basic block diagram for this node, it will be subject to changes depending on the testing and development iteration phase.

The main tasks that the publisher needs to handle is, being able to interface with the customer, we will model the data we need to extract from the environment as the following:

- Temperature and Humidity Sensing
- Lighting Adjustment
- Reserving Tables

These three use cases allow us to gather crucial information that could impact the experience of the customers and keep the restaurant informed of how much seating is available, this data in turn could be useful to determine the peak hours of the restaurant.

Broker

The broker is a Raspberry Pi, used primarily to store the payload, it facilities communication with the subscribers, whilst also forwarding data to the HTTP dashboard that we are using.

Environment Subscriber

This subscriber will display the information from the environment publisher, in the form of graphs and statistical data using a laptop. A custom dashboard template called Graphana will be used in order to realise this functionality, the subscriber will be used by the restaurant management team, in order to monitor any potential harmful changes to the environment and counteract them.

InfluxDB and Grafana Integration

We chose Grafana for our project because it offers a robust and flexible solution for visualizing time series data stored in InfluxDB. Grafana's powerful query editor allows us to create dynamic and interactive dashboards that can display real-time sensor data, such as temperature and humidity, crucial for monitoring the restaurant environment. Its wide range of visualization options and alerting capabilities make it ideal for detecting and responding to potential environmental hazards quickly. This integration ensures that the restaurant management team can maintain a safe and comfortable dining experience through intuitive and actionable insights provided by Grafana.

Customer Subscriber

The customer subscriber is a mobile device that is present with the customer on their table, it will be acting as a form of feedback to the customer about their inputs and how long their order will take to process.

Chef Subscriber

This subscriber is only responsible for receiving information about the order the customer has placed. It is in the form of a mobile device.

5.2 Use Case

There are 2 important users here. Both will be individually discussed.

- **Customer in Restaurant:**

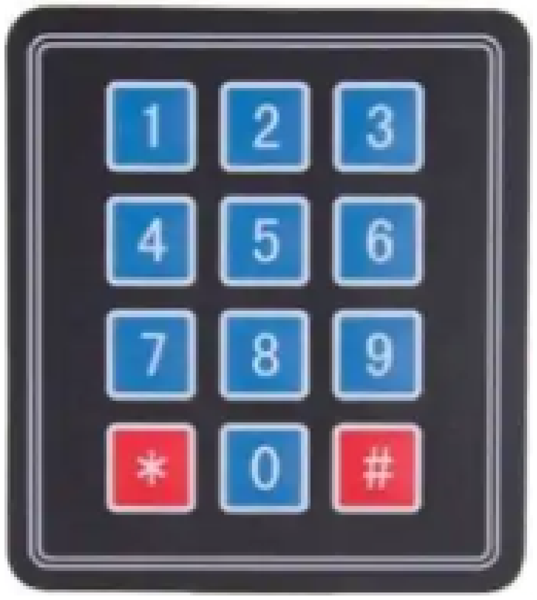

Figure 5.3: 3x4 Keypad

They can place an order using the keypad shown in Fig.5.3. They have to press any button from 1 till 9. Each number corresponds to a dish as the following:

1. Pizza
2. Spagetti

3. Spargel
4. Curry
5. Soup
6. Schnitzel
7. Tequila
8. Biryani
9. Burger

More importantly, if they would for some reason like to request human assistance, they are able to press a button. If the request was processed correctly and sent to the restaurant team, the LED will light green for 3 seconds as acknowledgement.

- **Restaurant Management Team:**

  The management team primarily wants to monitor the environment to make sure everything is as they have planned, the temperature, ambient lighting, any emergency sensors like the flame sensors are also displayed on the Backend Subscriber. A dashboard readily displays graphical information surrounding the system and any information that the customer wants to send, is received on the Backend Subscriber.

BIBLIOGRAPHY

[1] OASIS, “MQTT Version 5.0,” 2024. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. [Accessed: June 23, 2024].

[2] Arduino, “Arduino UNO WiFi Rev 2,” 2024. [Online]. Available: https://docs.arduino.cc/hardware/uno-wifi-rev2/. [Accessed: June 23, 2024].

[3] Raspberry Pi, “Raspberry Pi 3 Documentation,” 2024. [Online]. Available: https://www.raspberrypi.com/documentation/. [Accessed: June 23, 2024].

[4] Arduino, “MQTT Library,” 2024. [Online]. Available: https://www.arduino.cc/reference/en/libraries/mqtt/. [Accessed: June 23, 2024].

[5] Joy-IT, “Sensor Code,” 2024. [Online]. Available: https://sensorkit.joy-it.net/en/. [Accessed: June 23, 2024].