Design and Analysis of IoT Systems Project Report
Group 2

Clarissa Branje: 100716458
Tegveer Singh: 100730432
Tolu Elebute: 100724471

Population Tracking Alarm System

Project Description

With the need for public spaces to adhere to covid-19 max capacity guidelines in buildings, the purpose of our system is to track the number of people going in and out of facilities. Through tracking, the system will provide notifications and alerts in case the number of people exceeds a predefined limit. The project will have a user interface that displays the number of people detected entering a particular indoor facility. All in all, our project is helpful in multiple real-world scenarios, particularly during these COVID times, where we need to restrict the number of people in indoor facilities.

The project will provide notifications and alerts in case the number of people exceeds a predefined limit. With the use of an Ultrasonic sound sensor and a clock system, these will gather data on when, where, the trigger was activated, and how many times this occurred. This data would then be sent to an MQTT subscriber and then be published and notified on the user interface.

Note that this application can be reversed when people leave a particular facility as well. This way our IoT system can keep track of the number of people inside a particular place in real-time. Possible expansion on extra hardware could include a buzzer, camera, and lights.

Project Requirements

We based most of our project architecture around on the following requirements:

Functional Requirements:

1. The system will use an ultrasound sensor to detect the number of people in a facility
2. The system will keep track of the people entering and leaving a particular place
3. The application will trigger an alarm or send notifications in case the number of people exceeds a certain threshold
4. The limit can be pre-configured based on users needs
5. The system will manage data transfer through a publisher-subscriber mechanism

Non-functional requirements:

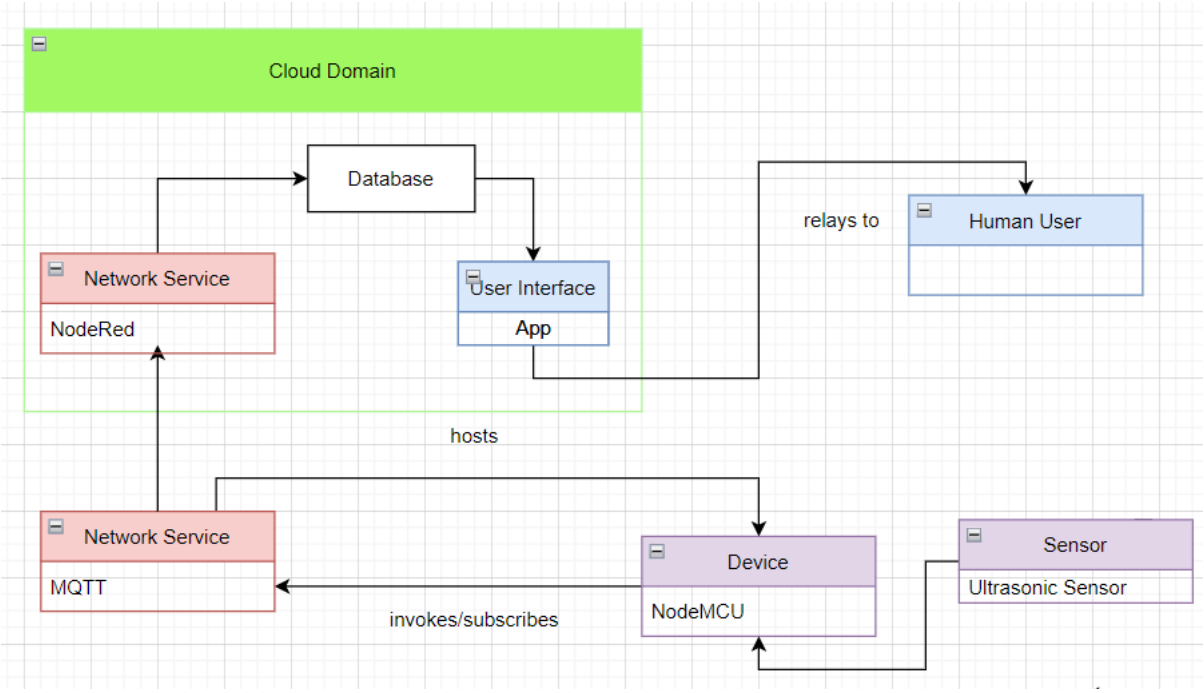
1. The pub-sub mechanism will be handled through MQTT/HTTP
2. The NodeMCU kit with ESP8266 board will be used with the ultrasound sensor to provide data and display it on the serial monitor
3. The system must provide over 90% accuracy of detection
4. The system must be user friendly and generic for multiple similar applications

Project Design and Architecture

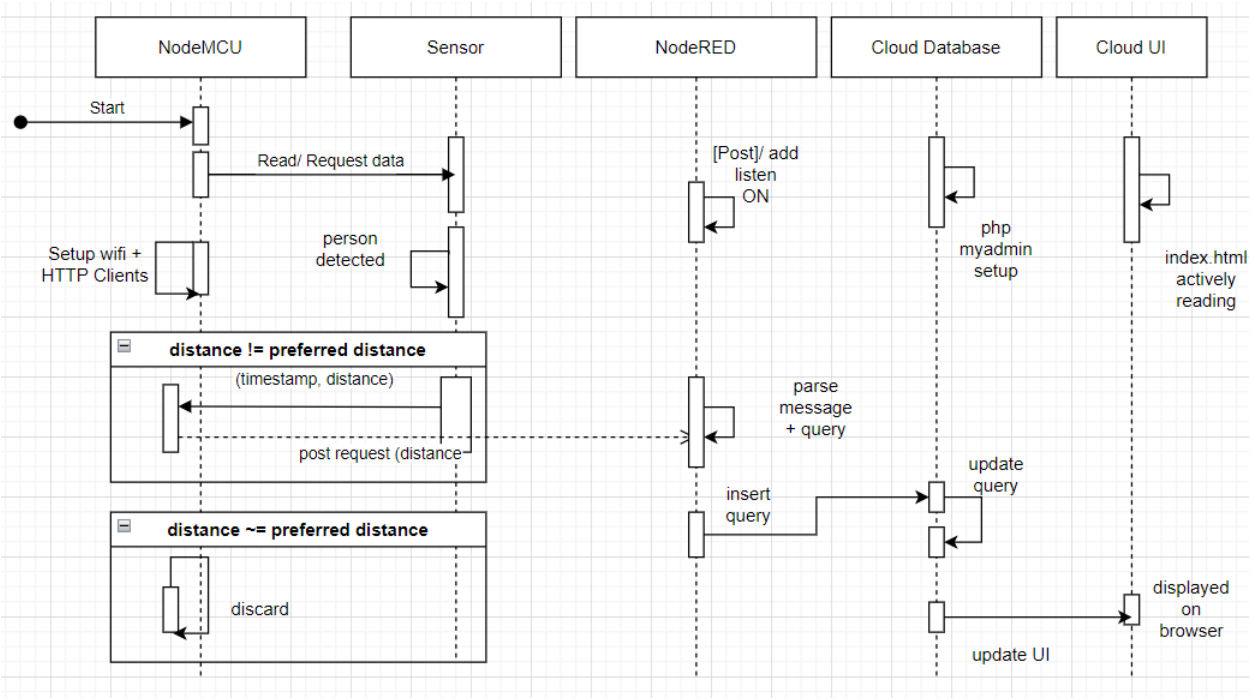
Our project uses a cloud domain setup using Digital Ocean. The Domain Name Server (DNS) we are using can be found at the following link:
Link: www.tegveersingh.xyz

The cloud domain contains the database and the Node-Red service. Node-Red is a flow-based visualization tool that can help the programmer handle HTTP endpoints and MQTT pub-sub mechanisms. The main webpage can be found at the following link:
Link: www.tegveersingh.xyz/index.php

This page contains the number of people currently inside an indoor facility along with their IDs and distance values displayed in table format. The sensor circuit contains the NodeMCU and ultrasound sensor that detects the distance to the objects and then publishes that data to the Node-Red endpoint. This message is then parsed and converted into a query for the database. The database is updated and the front end shows the updated values. Refer to the following diagram for a general overview of the system architecture



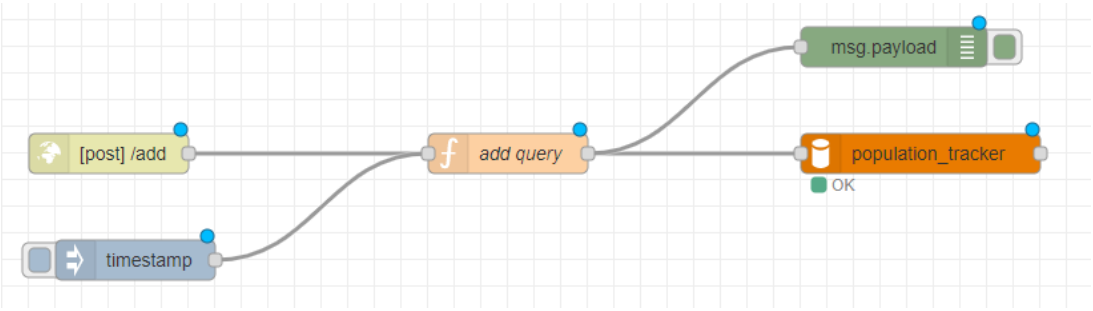
The team proposed to use HTTP requests to handle the data transfer in the application. Refer to the following sequence diagram the understand the current working of the application



NOTE: This sequence diagram above only shows how one sensor reading is transferred to the cloud and displayed on the User Interface. The above described process continues looping as long as the system stays active.

The above picture contains the Node-Red flows defined for our project. These flows have been added to perform CRUD (create, read, update, delete) operations on the database. There are 3 flows in total

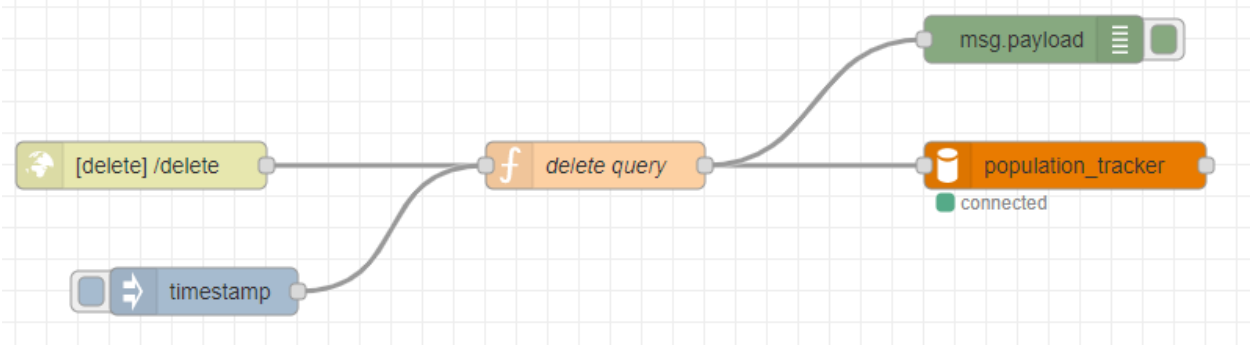
1. **Add** - This flow is illustrated in the following screenshot



Following are the main features of this flow:

- Handles post type requests through its HTTP request node
- Parses posted messages using a function node called add_query
- Adds id, distance, and timestamp values to the sensor_data table population tracker database after parsing the posted message

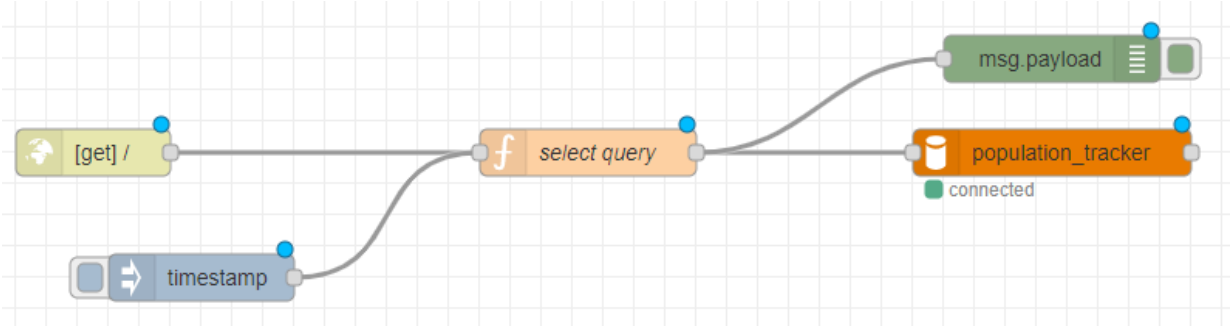
2. **Delete** - This flow is shown as the middle flow in the following screenshot.



Following are the main features of this flow:

- Handles Delete type requests at the URL /delete
- Deletes value with the specified ID from the database using its delete query function node

3. **Select** - Illustrated in the following screenshot:



Following are the features of this flow:

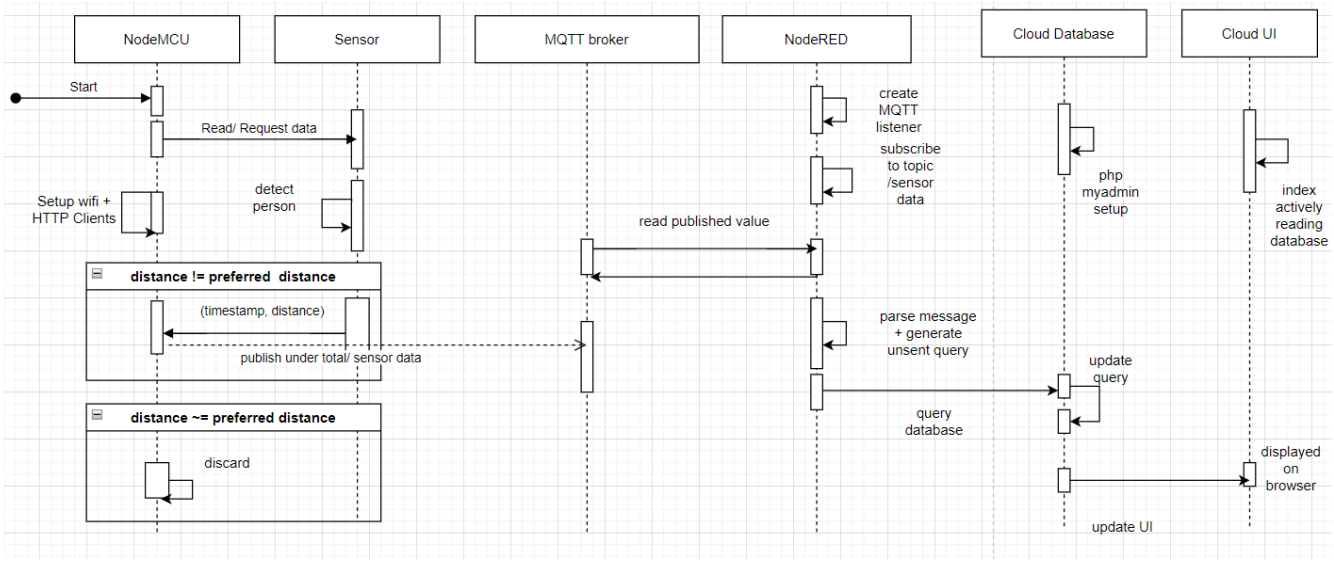
- Handles GET type requests
- Gets the list of tuples in the sensor_data table in our database
- Mainly used for testing when checking the current stored values on the cloud database

Each of these flows are accessed through the Arduino code to create the POST requests to the node-red endpoints. From here the node-red reads and parses those requests and then updates the database. For troubleshooting purposes, the following 2 nodes were added to each flow:

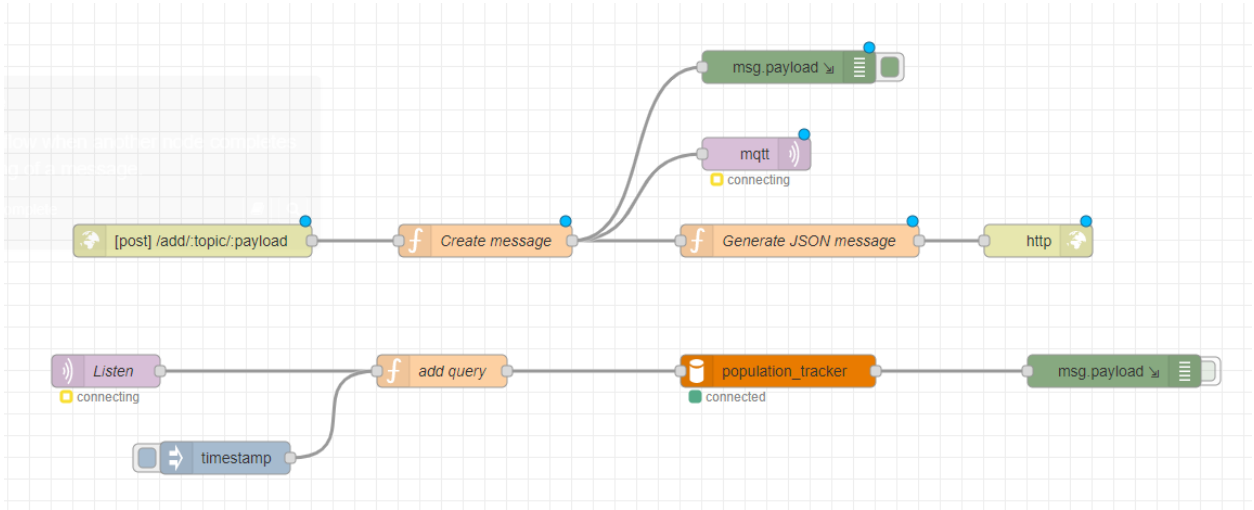
1. **Debug:** This is used for troubleshooting, the error/success messages are displayed on the debug console
2. **Inject:** This node was mainly used for testing. This node triggers the working of a flow. For instance, the deleted flow will be triggered if a user clicks on the inject timestamp node. It runs and prints any error messages on the debug console.

Architecture Design Decisions:

Initially, the group was considered using MQTT as the main transfer protocol. This method involved publishing the data under an MQTT topic (/sensor_data). This topic was subscribed to by MQTT input nodes setup on Node-red. These nodes read the values published under the topic and parsed them to generate an INSERT/DELETE query. This later updated the database followed, by updating the Front end. Refer to the following sequence diagram to get a better understanding of this implementation:



The following screenshot shows the Node-Red flows used to implement the application with MQTT as a primary message protocol



The above diagram shows two major flows that are mainly read for POST-type requests. Following are the main features of the first flow:

1. Data is formatted to JSON using the function nodes called Create a message and Generate JSON message.
2. HTTP response node added to the first flow to display the JSON data on the webpage at the url: /add/topic_name/payload.
3. The MQTT output node has been connected to the Create message function. This node is connected to the universal broker at test.mosquitto.org

The second flow, on the other hand, contains the following features:

1. MQTT subscriber nodes receive the sensor data that is published by the MQTT output node.
2. The parsed messages from the first flow are received by the MQTT listener node
3. Once received, the add query function node queries the database

The implementation was discarded because of 2 main reasons:

1. Inaccuracy of the MQTT listener node. This missed 1 out of 4 values on an average. This reduced the accuracy of the system by 25%. This was not in line with the specified functional requirements.
2. To successfully implement the add query flow, the listener node had to be configured to listen to all topics. This creates security concerns that are not acceptable as per the set nonfunctional requirements

More challenges were encountered while implementing the sensor functionality. The sensor generated extremely high values while no objects were obstructing its laser path. Moreover, if an object stayed in the path for too long, its input was recorded multiple times. Another issue was initialization of the distance. If not configured correctly, multiple junk values could be generated. The following development decisions were made to mitigate these challenges:

1. The first 10 values of the sensor are defined so as to initialize the distance values. This provides an initial measure to compare the new values with. If the deviation from initial values is higher than +- 5%, a new person is recorded.
2. Junk values over 2m are discarded
3. The previous value is stored and recorded with each loop. The tolerance value is considered to be 2% for 2 consecutive values. If it’s under 2%, It’s not recorded as another value and otherwise, it’s added as another person.

Initially the group was using Ubuntu 18.04 because the provided textbook was using this version. It was realized later on that the newer versions of Node Red are more compatible with Ubuntu 20.04. This required restarting the whole process of setting up the Phpmyadmin server, MySQL and Node installations. Fortunately, the DNS could directly be binded to the newly created droplet.

Deployment Design Decisions
Following were the main deployment design decisions made for our project:

1. For the deployment of our prototype, we chose to organize the files into different directories like Front-end, Django-REST-API, etc. These directories can be found on the following **GitHub link:** https://github.com/TegSingh/IoT-Project-Population-Tracker
2. The SSL certificates were acquired using the apache server certbot. This report will be included in the submitted repository.
3. An Adapter script was created to test if the speed of the process could be made faster. Following were the features of this script:
    - Acts as a local MQTT subscriber for the sensor data topic
    - Uses Python requests library to send requests to the cloud(node-red)

   This script has been included in the scripts directory of the repository
4. A well-formatted README file has been included in the submitted repository along with specific READMEs for subdirectories in the repositories

Link to Demonstration Video
https://drive.google.com/file/d/1VBXTdBiyvbmCGMPCLKkyYwZ73hJcRwCL/view?usp=sharing

Link to Presentation Slides
https://docs.google.com/presentation/d/13aDqTG44b_ZDbJYcGDCTH6TZ6meUKdQAU45xpNVR3zY/edit?usp=sharing

Conclusion
In conclusion, taking from all that was stated and elaborated on in our report, it can be said that we were able to apply all topics, ideas, and things discussed and learned from class, into creating a project that builds upon it. We successfully established a relevant project while adhering to the functional and nonfunctional requirements that were set for the project. We successfully implemented a cloud platform and a working front end for our application.