University of Puerto Rico
Mayagüez Campus

"Is the A/C on?"

Design Doc

INEL 4206 sec. 030
Prof. Luis B. Roa Pichardo
By: Mariana Pérez, Gabriel Román and Sergio Rivera

Table of Contents:

I. Introduction

The objective of this project is to set up a network of EP32s equipped with temperature and ultrasonic sensors located across campus that continuously transmit data to a central computer in the cloud. The system also offers end-users a web-based platform to monitor current and historical temperature readings, as well as the ability to interact with it using voice assistant Siri on an iPhone. The project aims to demonstrate the capability of the ESP32 and its integration with cloud-based platforms to provide a comprehensive and user-friendly system for temperature and proximity monitoring. In this design document, we discuss the 4+1 architectural views of the project.

II. Logical View

The logical view is concerned with the functionality that the system provides to end-users. In other words, what the system should provide in terms of services to its users. Our system is composed of two primary elements: the ESP32s with temperature and ultrasonic sensors, and the central computer in the cloud. The ESP32s continuously transmit sensor data to the central computer using a wireless communication protocol (MQTT). The central computer receives the sensor data transmitted by the ESP32s. The results are then presented to the end-users through a web-based platform where they can see the temperature and people count in different rooms. The web-based platform provided an easy-to-use interface for end-users to access the sensor data. Users are able to see the data in graphs, gauges, and numeric representations. This allows users to analyze and gain better insights into the temperature and people count in the monitored rooms. The use of voice assistants enhances the overall user experience.

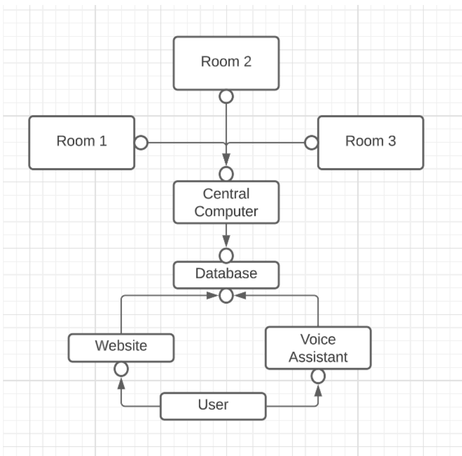


Figure 1. Logical Blueprint

III. Process View

Upon initialization, the ESP32 microcontroller commences operation concurrently with two ultrasound and one temperature sensor. A recurring loop is then executed to measure the temperature and people count. This data is then transmitted to the server. The system allows three modes of data display: actual data, historical data, and voice-activated data. The website exhibits data as per user requirements, in each of the display modes. Voice assistance data is shared with the user upon command. If preferred, data is also available through the serial monitor.

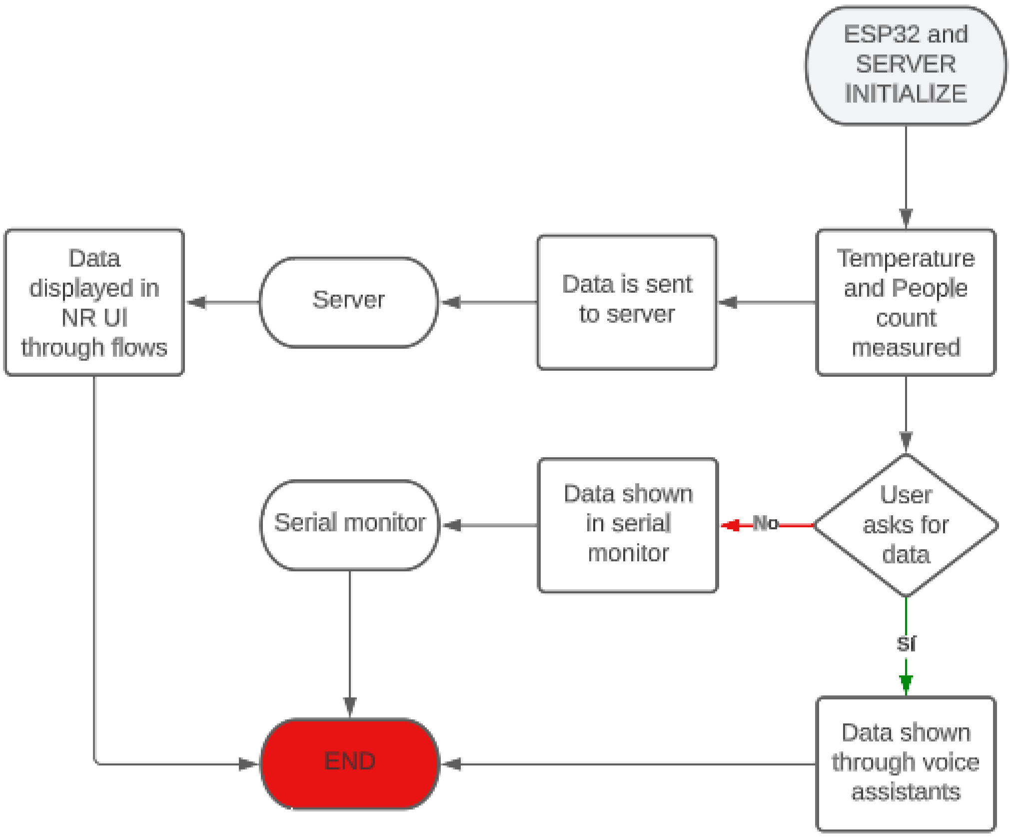


Figure 2. Process Flowchart (mock)

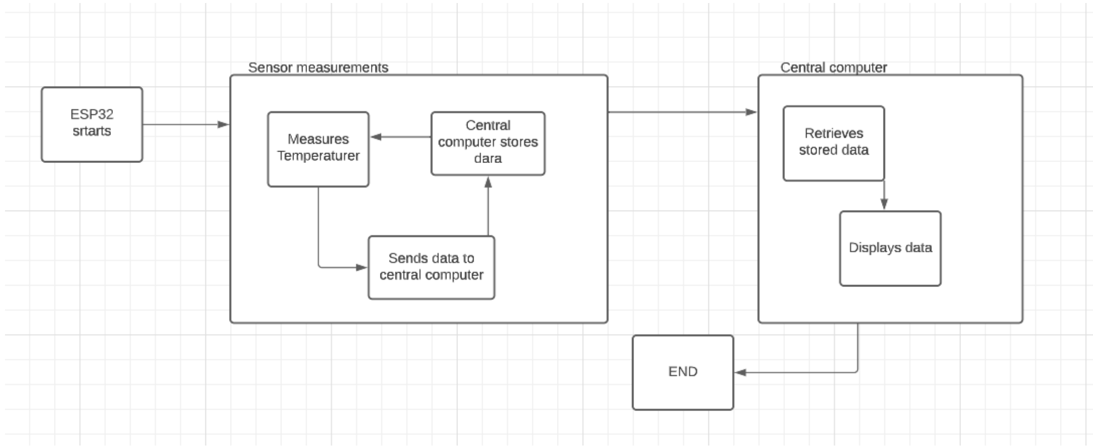


Figure 3. Process Blueprint

IV. Development View

The development view is concerned with software management. The following software tools were used to implement our project: Visual Studio Code, GitHub, Amazon Web Services Lightsail, PuTTYgen, Node-RED, and MQTT broker (communication protocol for ESP32). All of the code was written in Microsoft's source-code editor Visual Studio in the C++ programming language. We used GitHub to ensure version control and facilitate collaboration. We created a repository for our project along with the project plan. Both could be accessed here. Our repository has a "Document" folder where users can find all the important documents including this design document, a "Main" folder that contains the main/final program that can be accessed here, a "Research" folder used to store any helpful information, and a "Draft Code" folder where the draft coding implementations can be found.

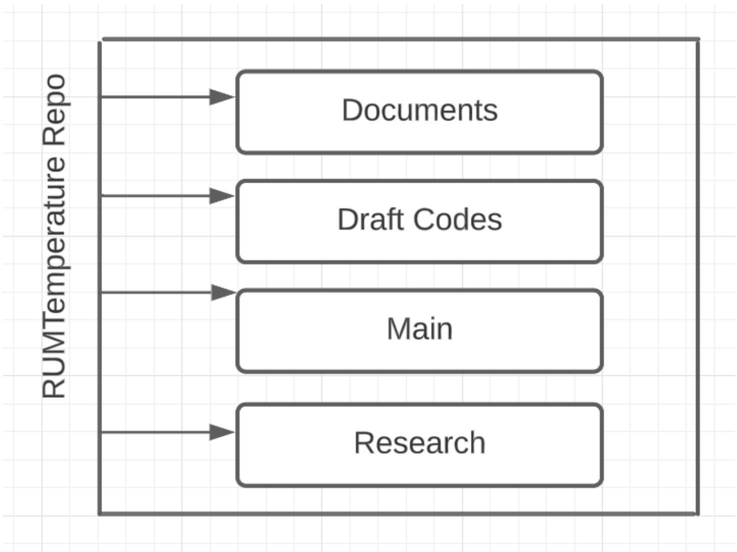


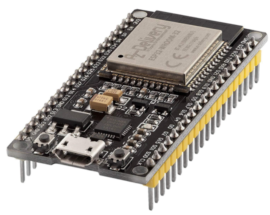
Figure 4. GitHub repository structure

The server is hosted on an AWS Lightsail Cloud Server, which offers developers compute, storage, and networking capabilities to manage websites and web applications in the cloud. PuTTYgen was used to create an ssh (secure socket shell) key pair to help set up our server. As mentioned before, the ESP32s are responsible for reading the temperature measurements in the different locations and transmitting the data obtained to the cloud. This was achieved via the MQTT lightweight messaging protocol. The AWS Lightsail cloud infrastructure hosts the MQTT broker and a flow-based programming tool named Node-RED. This tool was used to create a web-based dashboard displaying the temperature and people count data. The voice assistant integration was also done using Node-RED and with the iPhone app, Shortcuts.

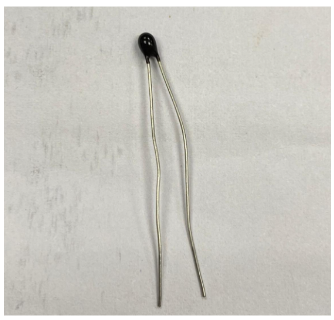
V. Physical View

The physical view of the system describes the hardware components and deployment configuration required to run the system. The following components were used to implement our project:

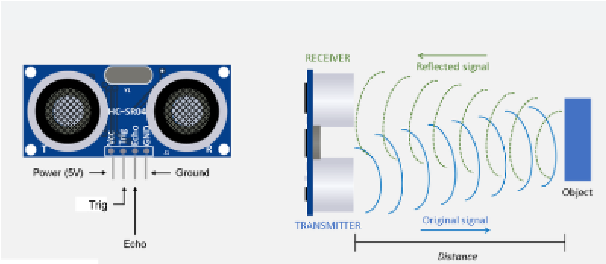
- ESP32 microcontroller: The ESP32 microcontroller is the main component of the system and served as the central hub for processing the sensor data (temperature and PIR sensors) and communicating with external devices (AWS Lightsail server). It was mounted on a breadboard.



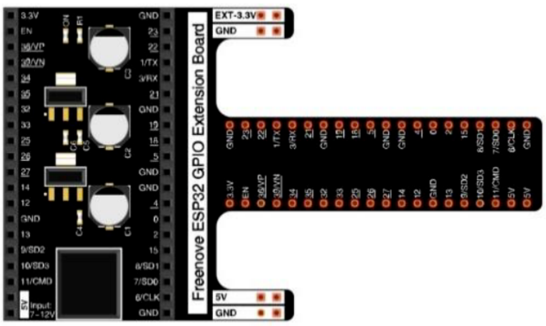
- Thermistor Temperature Sensors: The thermistor temperature sensors were connected to the ESP32 via the MQTT protocol and are responsible for measuring the temperature of the environment. This enabled us to determine the specific temperature in each room when the command was made. These work by changing resistance with changes in the surrounding temperature.



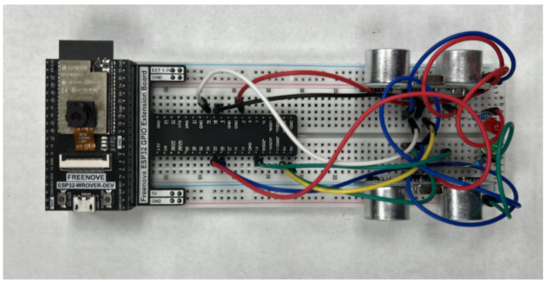
- Ultrasonic sensors: The Ultrasonic Sensors were connected to the ESP32 via the digital input pins and are responsible for detecting interference at the entrance of the room. This enabled us to determine the exact number of people that entered and exited the specific room. By using two ultrasonic sensors, each facing away from the other, we were able to determine how many people were entering and exiting the room.



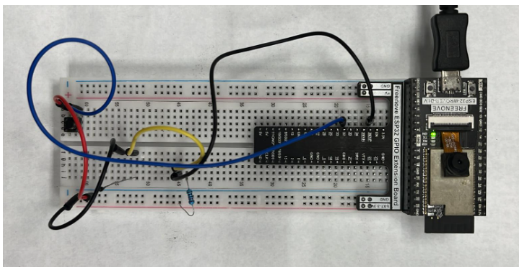
- Freenove ESP32 GPIO Extension Board: The ESP32 was attached to an extension to make the circuit more comfortable and easier to use. It had a total of 32 additional GPIO pins, which could be used for a variety of purposes such as controlling LEDs, motors, sensors, etc. One benefit was its expandability which allowed it to easily expand the number of GPIO pins available by adding more boards..



- ESP32 Ultrasonic Distance Sensors: One of the ESP32 microcontrollers in the setup was equipped with two ultrasonic distance sensors facing opposite directions. This ESP32 counts the number of people entering and exiting the room. One side of the sensor counts the people that go into the room, and the other side counts the people that leave the room. A limit could be set up for the number of people in a room.



- ESP32 Temperature Sensor: One of the ESP32 microcontrollers in the setup was equipped with an analog thermistor. Thermistors worked by measuring the changes in resistance that occurred as the temperature changed. A benefit of a thermistor temperature sensor was that it was small and did not require much space. It also had a fast response time which meant that it could quickly detect changes in temperature. Last but not least, thermistor temperature sensors had a high accuracy when compared to other types of sensors like thermocouples.



VI. Scenarios

The scenarios view of the system includes the use cases and scenarios that the system should support.

- A user should be able to view current and historical temperature and occupation readings through the web-based dashboard.
- The user should be able to interact with the system using Siri voice assistant to indicate the current temperature in a specific room and the number of people that are currently in that room.
- The system should also be scalable to accommodate additional ESP32s, temperature sensors and ultrasonic sensors if the system were to expand to more rooms.

VII. Conclusion

The 4+1 architectural views provide a comprehensive understanding of our project's system architecture. It ensures that all aspects of the system are considered and properly addressed in the design, development and testing process. We implemented the ideas we sought in our research and were able to achieve our objective. We can assure that this solution provides an efficient way to monitor the temperature and occupancy of several rooms in the University of Puerto Rico - Mayaguez Campus, and that the sensor data is easily accessible via Siri and the web interface.