University of Puerto Rico
Mayagüez Campus

"Is the A/C on?"

Design Doc Draft

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 proximity sensors located across campus that will continuously transmit data to a central computer in the cloud. The system will also offer end-users a web-based platform to monitor current and historical temperature readings, as well as the ability to interact with it using voice assistants such as Siri or OK Google. This 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 will 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 proximity sensors, and the central computer in the cloud. The ESP32s will continuously transmit sensor data to the central computer using a **wireless communication protocol (MQTT).** The central computer will receive the sensor data transmitted by the ESP32s and store it in a database. The database will serve as a repository of the sensor data, where it can be accessed and analyzed by end-users. The results shall be presented to the end-users through a web-based platform. The web-based platform provides an easy-to-use interface for end-users to access the sensor data. It would be ideal if our users could visualize the data in graphical formats, such as charts or tables. This would allow our users to analyze and gain better insights into the temperature and proximity trends in the monitored rooms. The use of voice assistants would enhance the overall user experience.

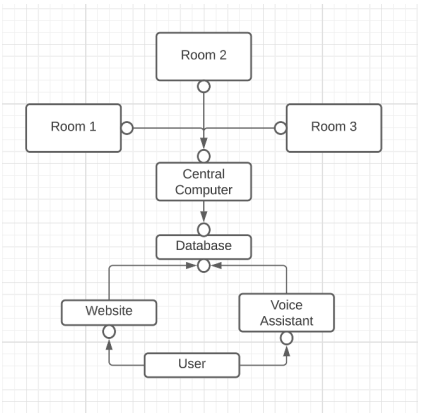


Figure 1. Logical Blueprint

III. Process View

Upon initialization, the ESP32 microcontroller will commence operation concurrently with proximity and temperature sensors. A recurring loop will then be executed to measure the temperature and proximity data. This data will be transmitted to a central computer and saved for future reference. The system will allow for three modes of data display: actual data, historical data, and voice-activated assistance. The website will exhibit data as per user requirements, in each of the display modes.

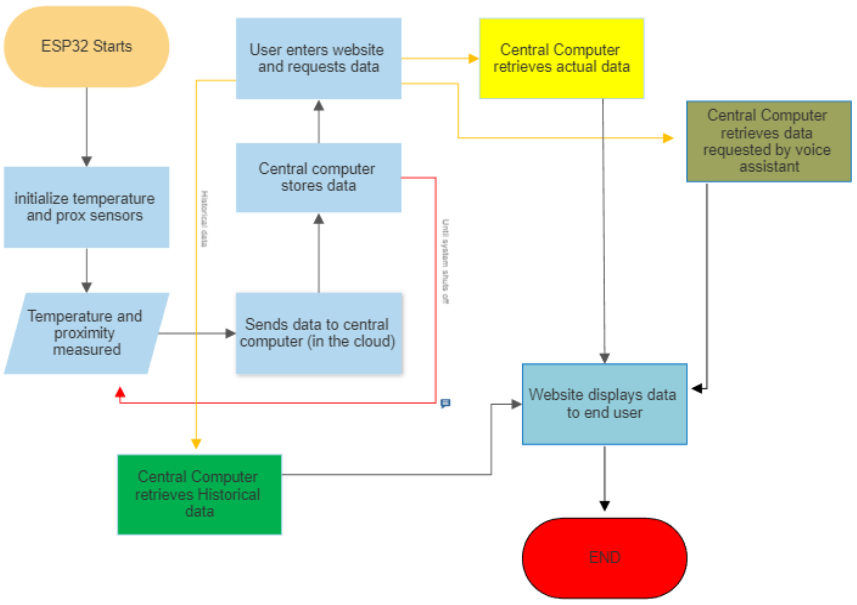


Figure 2. Process Flowchart (mock)

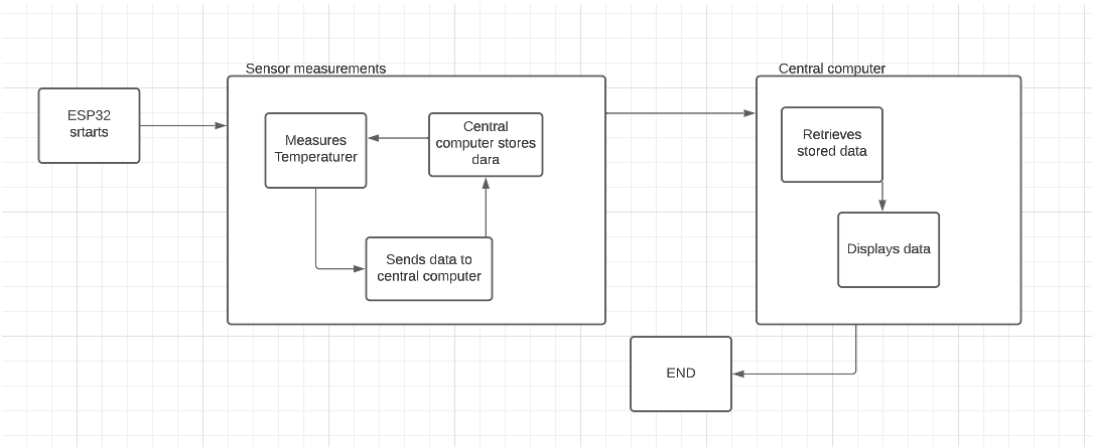


Figure 3. Process Blueprint

IV. Development View

The development view is concerned with software management. We will be using the following software tools to implement our project: Visual Studio Code, GitHub, Amazon Web Services Lightsail, PuTTYgen, Node-RED, and MQTT broker.

All of the code will be written in Microsoft's source-code editor Visual Studio in C programming language. We will be using GitHub as a code hosting platform to ensure version control and facilitate collaboration. We created a repository for our project along with the project plan. Both can be accessed here. As of right now, our repository has a "Document" folder to store all important documents including this design document, a "Main" folder to store our main/final program, a "Research" folder to store any helpful information, and a "Draft Code" folder to upload our draft coding implementations. We already started implementing a draft code, which can be found here.

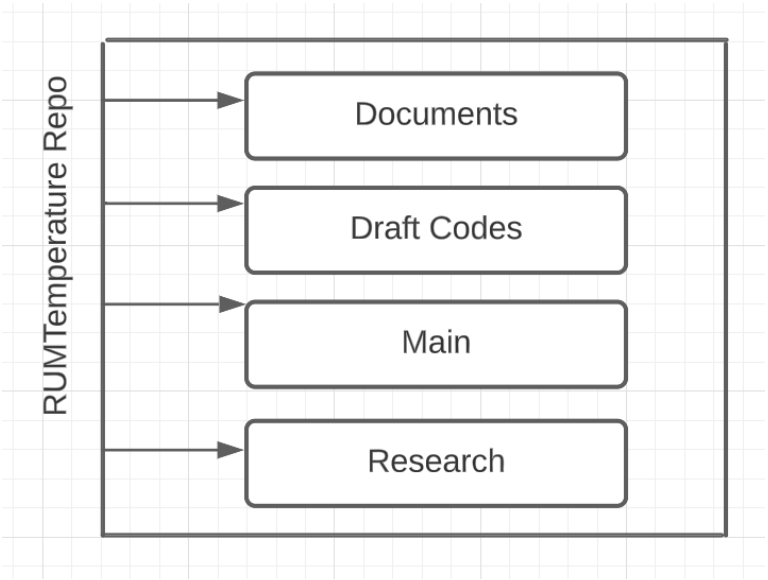


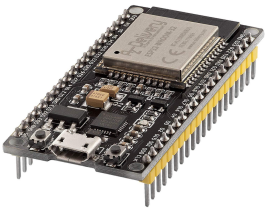
Figure 4. GitHub repository structure

We will be using AWS Lightsail as our virtual private server provider, which offers developers compute, storage, and networking capabilities to manage websites and web applications in the cloud. PuTTYgen will be used to create an ssh 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 will be achieved via the MQTT lightweight messaging protocol. The AWS Lightsail cloud infrastructure will host the MQTT broker and a flow-based programming tool named Node-RED. will be used to create a web-based dashboard that displays temperature and proximity data. The voice assistant integration will also be done using Node-RED. This tool will be used to create web-based visualizations displaying the temperature and proximity data. The voice assistant integration will also be done using Node-RED.

V. Physical View

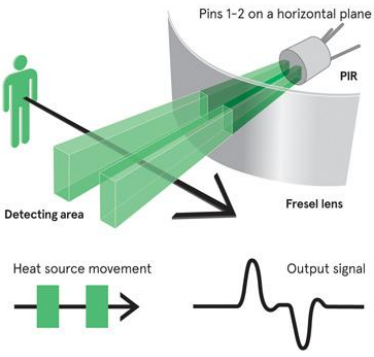
The physical view of the system describes the hardware components and deployment configuration required to run the system. In this project, the physical view includes the following components:

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



- Temperature Sensors: The temperature sensors are connected to the ESP32 via the MQTT protocol and will be responsible for measuring the temperature of the environment. This will enable us to determine the specific temperature in each room when the command is made.

- PIR Sensors: The PIR Sensors are connected to the ESP32 via the digital input pins and will be responsible for detecting motion in the environment. This will enable us to determine the exact number of people that enter and exit the specific room. By using two PIR sensors, each facing away from the other, we will be able to determine who is entering the room and who is exiting the room.



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 readings through the web-based dashboard.
- The user should be able to interact with the system using voice assistants, such as Siri, to turn on/off AC, to know the number of people currently in the room, to change the location name, to know the temperature in Fahrenheit and Celcius, etc.
- The system should also be scalable to accommodate additional ESP32s, temperature sensors and PIR sensors if the system were to expand to more rooms.

VII. Conclusión

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 and development process. As demonstrated, we intend to complete this project by implementing the ideas we sought in our research. However, our research is not definitive, some aspects are susceptible to change. Therefore, we will carry on with our research as well as with the development of the project.