A picture containing text, sign

Description automatically generatedText

Description automatically generatedUniversity of Puerto Rico – Mayagüez

Electrical & Computer Engineering Department

Architecture and Design

Final Project: “Is the A/C on?”

INEL 4206 – 020 Microprocessors I

Alanis M. Colón González

Francisco A. Rivera Méndez

**Introduction**

The ESP32 is a microcontroller used to program and better access the information given in electrical circuits and other electrical implementations. During the project, applications of temperature and proximity sensors were used to provide a comprehensive monitoring system for campus facilities. The sensors are then placed so the temperature of different rooms could be measured and detect the presence of people in those rooms using proximity sensors. All the readings are sent to the server where they are stored and processed. The data can be accessed by users through a web interface that displays both the current and historical temperature measurements, as well as the occupancy status of each room. In addition, users can query the system using "Siri" or "Ok Google" voice commands to obtain real-time temperature and occupancy data. This project aims to provide a user-friendly and efficient solution for monitoring campus facilities, enabling facility managers to make informed decisions about resource allocation and energy management.

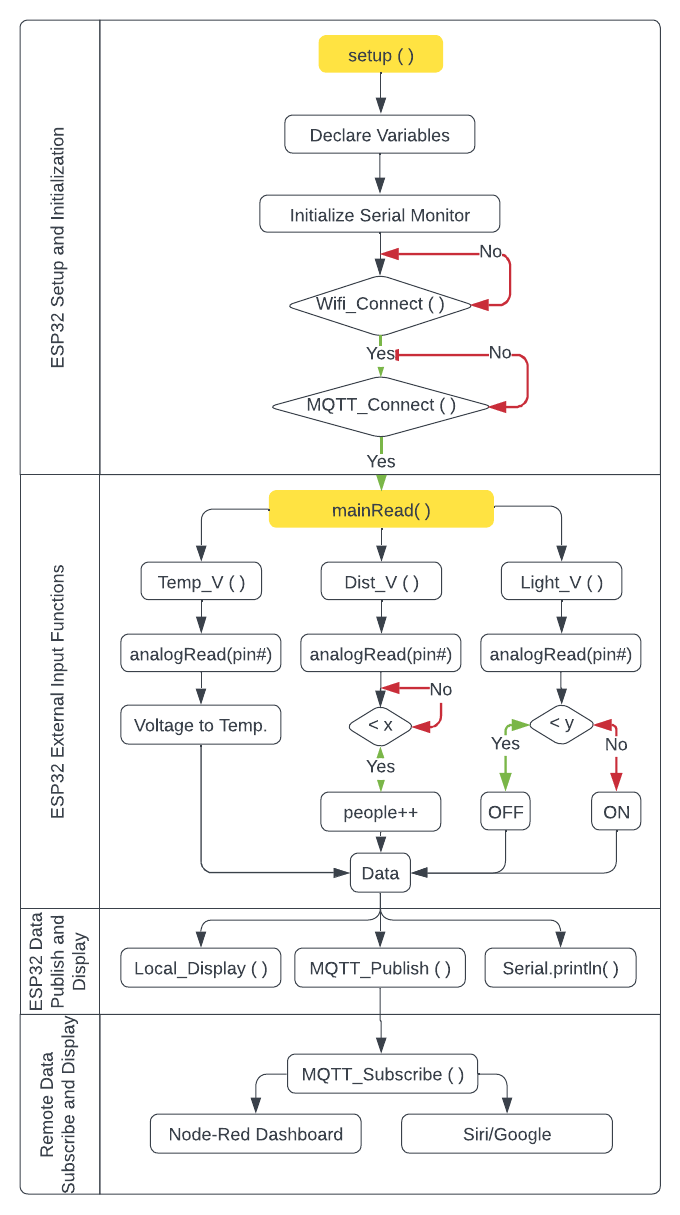
**Logical Diagram

Description automatically generatedView**

The logical architecture primarily supports the functional requirements—what the system should provide in terms of services to its users.[1] Therefore, the logical architecture as displayed is explained in a sequential order with the purpose of executing the product as efficiently as possible. Starting with the “start button” there are three actions that start occurring, these being: read temperature sensor, read distance sensor, and read light sensor. These values of voltage are then converted to functions that can be read and printed in the servers and the local display. With these processes finalizing it can then be shown in the user’s phone efficiently and fast. This process executes the plan in these steps, and it’s made for the user’s accessibility made be easy and fast.

After the use of the MQTT there is a node-RED implementation that can carry out different functions and tasks in one move. Node-Red acts as a way to countify the values being received by the sensor inputs. The functions inside the nodes then separate the values of user inputs and values received by the sensor to make the calibration of the device more accessible and efficient. These then products made by the node functions can be then published in the UI that can be visible by the user or the person that has access to the server's name and port.

**Development View**

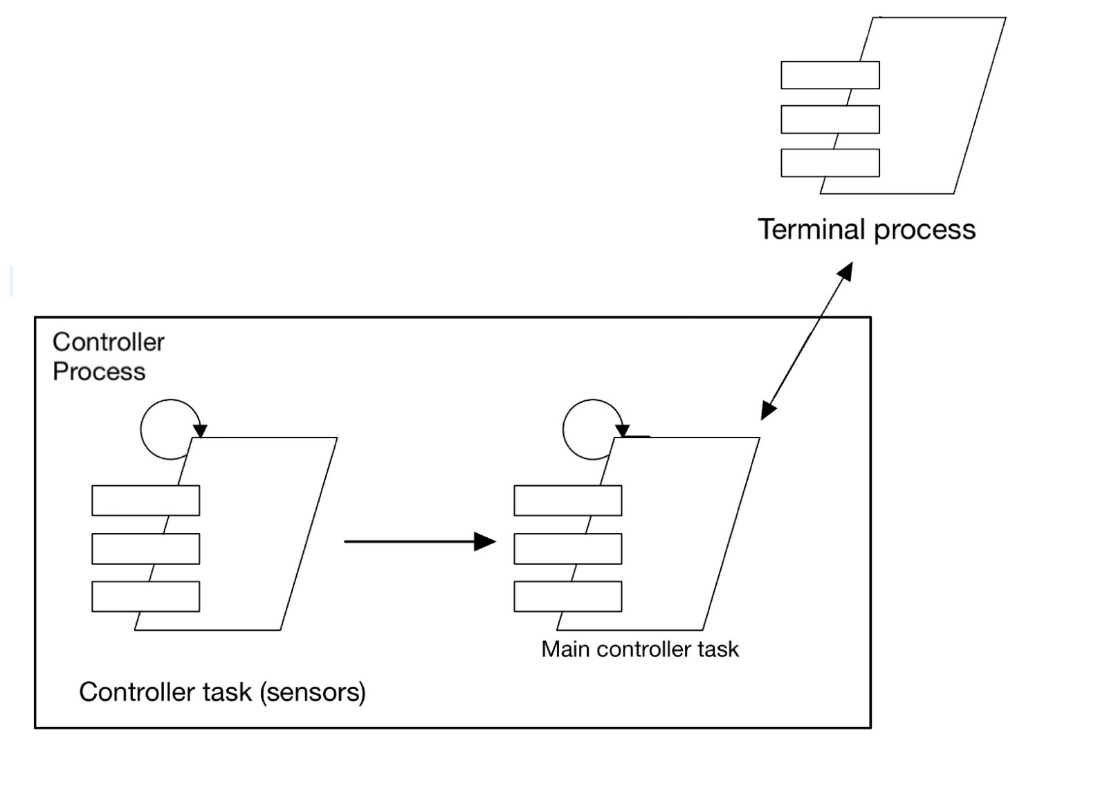


The development architecture focuses on the actual software module organization in the software development environment. [1] To describe this, the previous diagram was created. In this image four main areas can be seen: the ESP32 setup and initialization; ESP32 external input functions; ESP32 data publish and display; and the remote data subscribe and display. First in the ESP32 setup and initialization, the variables needed for the code are initialized and then a setup function is created to execute all the processes needed for the rest of the programming to work such as initializing the serial monitor, establishing the Wi-Fi connection to then establish the MQTT connection. After all that is established, we move to a function where the external inputs are collected. To achieve this, three sub-functions were created to read the voltage values of the sensors and execute different tasks with them. Once this is done the output of these functions are shown on the displays and sent via the MQTT communication protocol. Using this protocol, we can then receive the data in node-red passed through some logical functions to coherently display it on the node-red dashboard. Additionally in node-red function some additional steps must be taken to establish the Siri/Google connection.

**Diagram

Description automatically generatedPhysical View**

This physical architecture draws a more visible approach to the system to be designed, making it easier to read and understand. To the right of the picture there are the sensors that would be used to start taking tests, these then numbers would be connected to the esp32. Then an MQTT is used to pass the information generated to the servers that are going to be up in the net. The MQTT function although simple is a complicated set of sequences that is done within the server to better polish the numbers received and send them accordingly. This information would be now discoverable and able to pass to anything with internet access and access to the server’s information, such as apple/google services. With this system users could be able to understand the process easily with the steps shown above. Users would also be able to input values so that these values could be compared to those of the sensors within the circuit and do a calibration of sorts.

**Process View**

For this view, a simple but direct approach was taken to accommodate the sensor's technicalities. The controller task and main controller share the similarity of having a process and these also share that they loop continuously. For the controller task (sensors) is going to be the designed circuitry that will detect the temperature levels and the number of people entering the room. This process is one that will loop as seen in the top left corner of the process. The information given will then be uploaded to the users' phone, this being inside the controller process. The terminal process would be the representation of the servers and it shows direct interaction with the main controller.

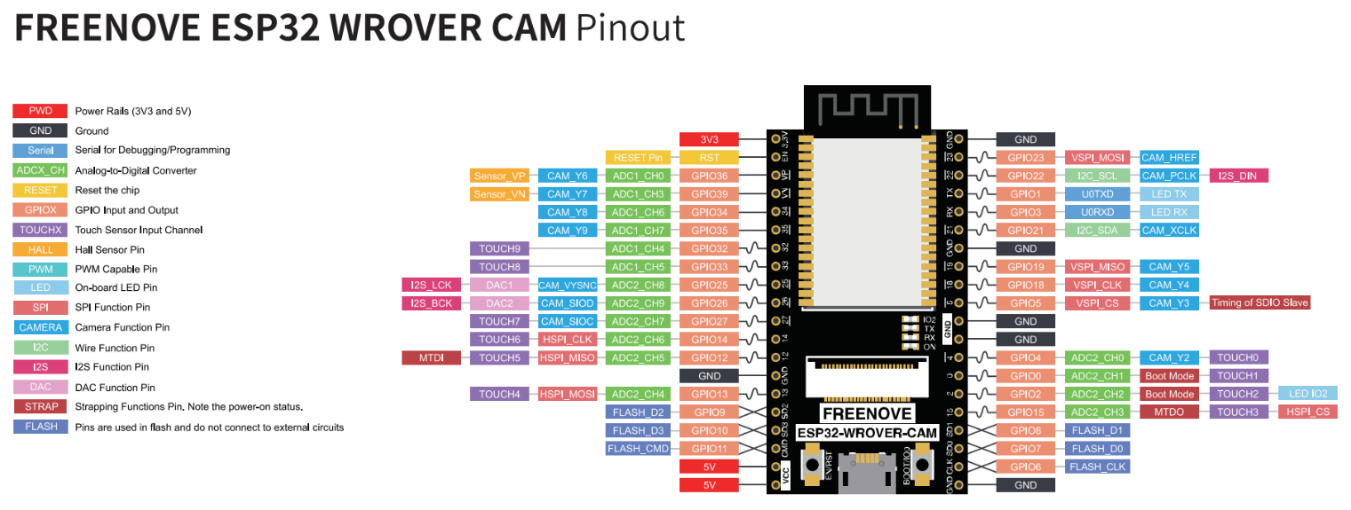
**Scenarios**

Here are some scenarios and applications where an ESP32 equipped with temperature and proximity sensors located in several rooms in buildings can be used:

1. Office Temperature Monitoring: In an office building, the ESP32 devices can be used to monitor the temperature of each room to ensure that employees are comfortable while working. The central computer can use this data to adjust the HVAC system to save energy while still keeping the office at a comfortable temperature.
2. Hospital Monitoring: In a hospital setting, the ESP32 devices can be used to monitor the temperature and the number of patients in their rooms. This data can be used to ensure that patients are comfortable and to alert healthcare providers if a patient's condition is deteriorating.
3. Data Center Monitoring: The ESP32 devices can be placed in a data center to monitor the temperature and proximity of servers and other equipment. This data can be used to optimize the cooling system and prevent equipment from overheating.
4. Greenhouse Monitoring: In a greenhouse, the ESP32 devices can be used to monitor the temperature and proximity of plants. The central computer can use this data to adjust the temperature and humidity levels to ensure that the plants are growing optimally.
5. Energy Management: The ESP32 devices can be used to monitor the temperature and if there are personnel in a commercial building. This data can be used to optimize the energy usage of the building by adjusting the HVAC system to only cool occupied rooms.
6. Food Storage: The ESP32 devices can be used to monitor the temperature of food stored in the cafeteria and food storage rooms. This data can be used to ensure that food is stored at the correct temperature to prevent spoilage.

Hardware

To create the circuit for the project, the following components were needed; LED display, analog temperature sensor (thermistor), resistors, ultrasonic sensors, photoresistors, and ESP32. In fig. 1 the pinout diagram of the microcontroller can be seen with the specification of the communication protocols used in those specific pins. In fig.2 the circuit diagram to connect the thermistor can be seen.



*Figure 1: ESP32 Pinout Diagram*

Connections:

|  |  |
| --- | --- |
| Temp sensor output | GPIO 0 (pin 0) |
| Ultrasound 1 Trigger | GPIO 5 |
| Ultrasound 1 Echo | GPIO 18 |
| Ultrasound 2 Trigger | GPIO 25 |
| Ultrasound 2 Echo | GPIO 26 |
| LCD Screen SDA | GPIO 34 |
| LCD Screen SCL | GPIO 35 |
|  |  |

State Machine

The state machine of the project consists of several logical systems to allow the most simplified interactions between users and machine. Starting the system in an off state, it is then the ESP32 turned on for it to connect to Wi-Fi thus allowing connection to the dedicated server. If any of these stages where to fail to connect it would simply turn back and try again until it succeeds, this redundancy is repeated in all failed sequences. After this, the temperature sensor and distance temperature record data and have it sent to database. The data is also displayed through the GUI for the user to see. It is then this information accesible to the user through their phone ai assistant so the data collected can be shown in present time.

A picture containing circle, screenshot, diagram, text

Description automatically generated