A picture containing text, sign

Description automatically generatedText

Description automatically generatedUniversity of Puerto Rico – Mayagüez

Electrical & Computer Engineering Department

Architecture and Design Doc

Final Project: “Is the A/C on?”

INEL 4206 – 030 Microprocessors I

Fall Semester 2022

Alanis M. Colón González (802-19-1971)

Raul O. Ortiz (802-##-###)

Table of Content

* Introduction
* Hardware
* Software for a ESP32
* Server/ Node-Red Implementation

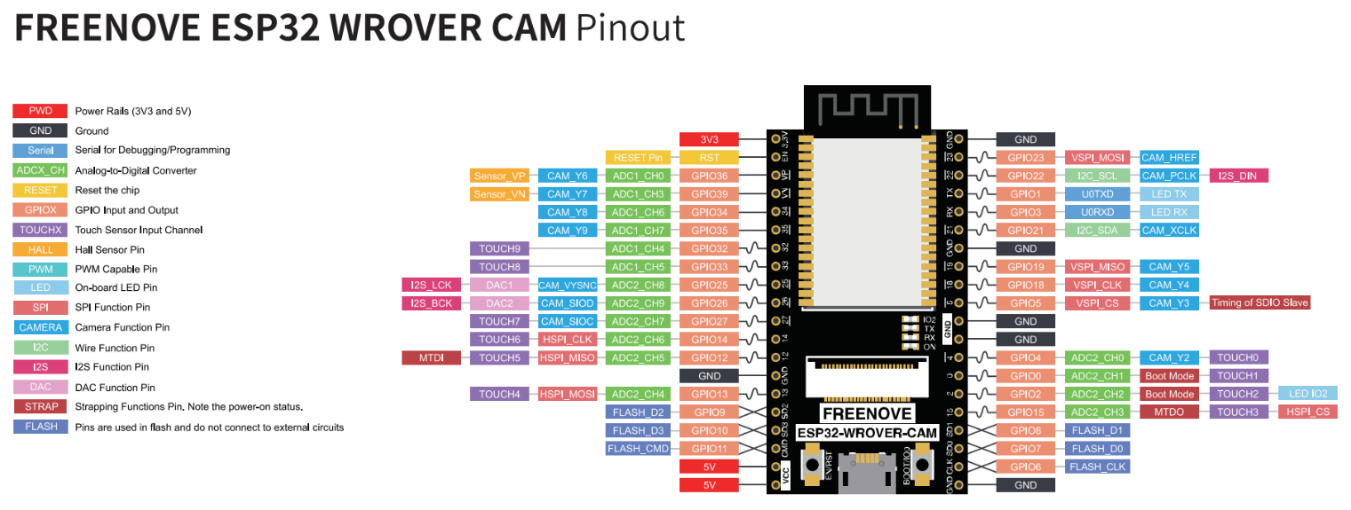
**Introduction**

The goal for this project was to develop a circuit and a program in C language, for which the temperature could be observed on a screen or through a web page. For the development of the circuit, the ESP32-WROVER-DEV microprocessor was used, it has a great variety of characteristics such as; it has the ability to connect via Wi-Fi or Bluetooth, comes equipped with a 32-bit dual core processor, among many other features. With the goal of discovering what the temperature is, an analog temperature sensor was used. Which we can find as KY-013, has an operating voltage of 3.3V-5V and a measurement range of -55 Celsius to 125 Celsius. With these simple components, a code was created which supports sensor calibration, configuration, sending data to the server and to the screen in the circuit.

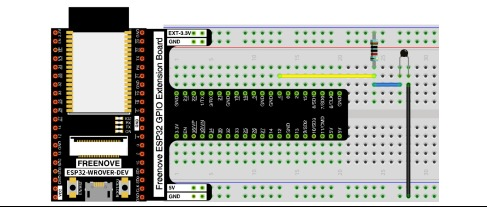
**Hardware**

To create the circuit for the project, the following components were needed; 7-pin OLED display, analog sensor (thermistor), 10 k Ω resistor, and ESP32. Important to note that no matter which way you connect the temperature sensor it will work the same way, you just must make sure to connect the 10 k Ω resistor. In table 1 is a list of the specific pins used and the pinout diagram with the protocols of each pin. For the hardware part of this project the ADC2\_CHX channels were used to perform the analog-to-digital conversion. It is important to know that the ADC1\_CHX pins could not be used as a result that the ESP32 uses those pins for WIFI. Also, in figure 3 the connections used for the thermistor are shown.

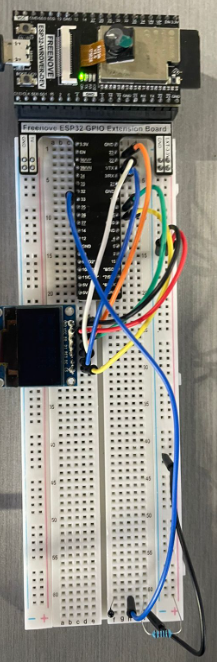
|  |  |
| --- | --- |
| GPIO | PINOUT |
| GND | GND |
| 3.3V | VCC |
| 18 | DO |
| 23 | D1 |
| 1/TX | RES |
| 3/RX | DC |
| 5 | CS |
| 33 | Thermistor pin |
| GND | Thermistor pin |

*Table 1: List of connections*

*Figure 1: ESP32 Pinout Diagram*



*Figure 2: Thermistor Connections*

****

*Figure 3: Complete Circuit Connections*

**Software- ESP-32**

For the software part of ESP32 visual studio code was used with the Platform IO IDE to connect to the ESP32. Therefore, the coding language primarily used was C language. In this program some of the libraries and their respective purposes were the following:

|  |  |
| --- | --- |
| Library | Purpose |
| #include <Arduino.h> | Development environment to communicate with the ESP32 |
| #include <WiFi.h> | Library to connect with the Wi-Fi |
| #include <stdio.h> | Standard Input and Output library. Contains variables, functions, and macros are related to the input and output operations in C. |
| #include "freertos/FreeRTOS.h" | Libraries to use FreeRTOS real-time operating system kernel for embedded devices |
| #include "freertos/task.h" |
| #include <PubSubClient.h> | This library provides a client for doing simple publish/subscribe messaging with a server that supports MQTT. |
| #include <driver/adch> | This driver supports sampling and converting raw values into microvolts. |
| #include <SPI.h> | Libraries used for the functionality of the display. |
| #include <Wire.h> |
| #include <Adafruit\_SSD1306.h> |

Diagram

Description automatically generated

*Figure 4: ESP32 Code Flowchart*

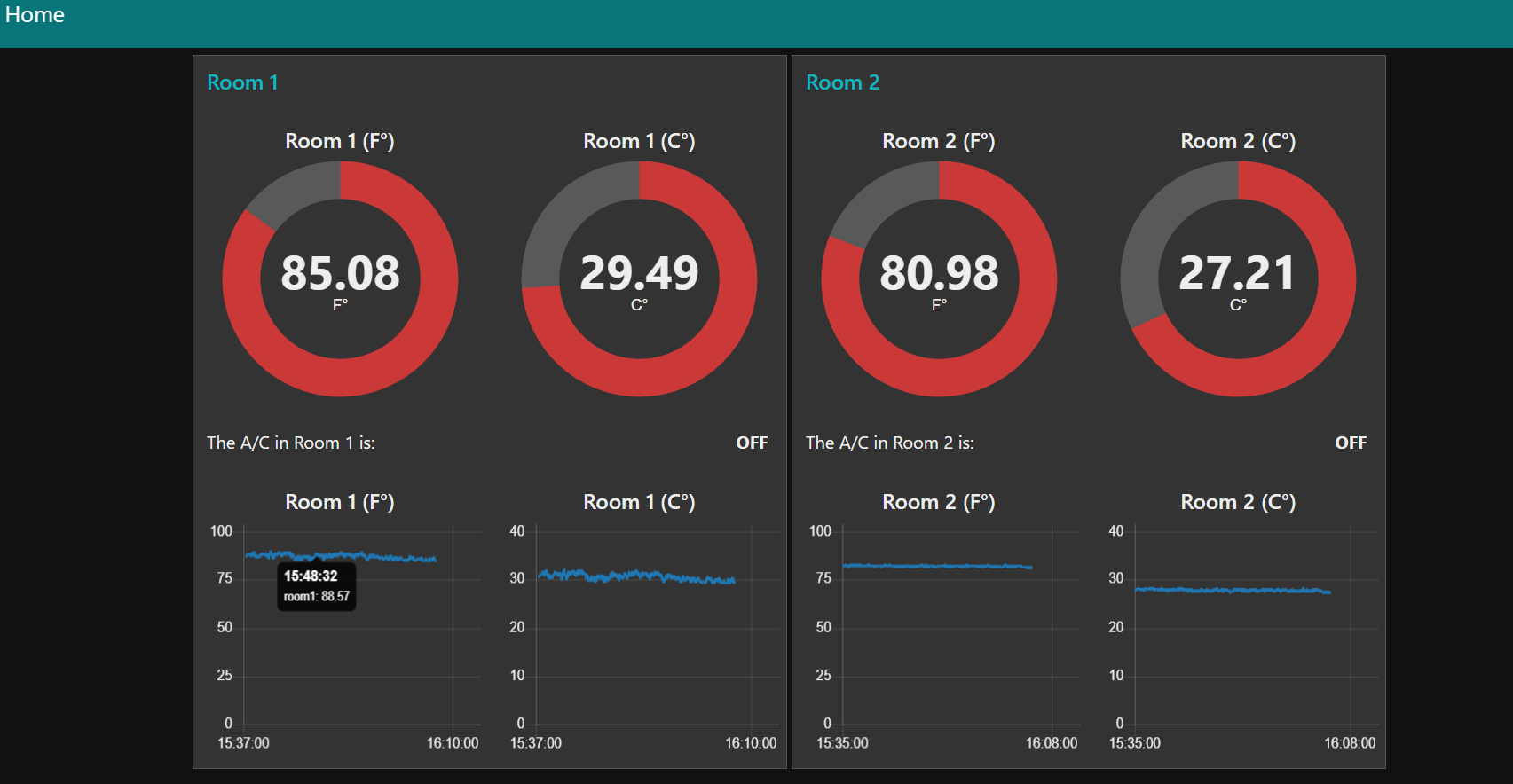
**Server/ Node-Red Implementation**

In order for us to be able to receive the data sent by the ESP32 a server was created in AWS. In this server node-red was installed with several packages such as the aedes broker used for the MQTT connections and the dashboard packages to set up the user front end. The flow of the data once it arrives the MQTT in node red can be seen in the following figure.

Diagram

Description automatically generated

*Figure 5: Node-Red Code*



*Figure 6: Node-Red Dashboard*

**Global Flowchart**

Diagram

Description automatically generated