University of Puerto Rico Mayagüez Campus
Electrical and Computer Engineering Department

Final Project: Is the A/C On?

Jessy Rodríguez Colón
Harry Márquez Torres
May 4, 2023

Introduction

In order to implement this project. We needed to create a circuit with the ESP-32 and various sensors that would be constantly taking temperature and room capacity readings. Then, the microcontroller needed to establish communication with the cloud to activate the "Hey Siri" function. The ESP32 is connected to a broker that is constantly receiving data from the temperature and distance sensor. This data is then sent to a dashboard that displays both outputs graphically. This way, the user can know if the air conditioner is on or off, and how many people are in a room from any part of the world.

ReadMe File:

Members: Jessy Rodríguez Colón -jessy.rodriguez@upr.edu Harry Márquez Torres -harry.marquez@upr.edu

Description:

The principal idea of this project is to know if the AC is ON in several rooms in the building across campus. To make this project we will be using several components. some of them are: a cloud server, an esp32, a virtual assistant,temperature sensors, wifi connection and Visual Studio Code. The devices send constant measurementes to a central computer in the cloud and we can see this through display and in the website.

Website: harryjessy.site

Software and tools required for the project:

-Visual Studio Code, which we used as our development environment -PlatformIO, an extension of VS Code that helped us with ESP32 programming -An ESP32 starter kit, which provided us with the necessary hardware -LMT-84 temperature sensor -HC-SR04 distance sensor -Breadboard and jumper cables -Red LED, which we used for visual output -AWS, which we used for cloud computing -Puttygen and whois.com, which we used for SSH access to the ESP32 -Windows PowerShell, for executing commands on the ESP32 -Node-RED, which we used for IoT automation -Studio 3T, for managing the database -Siri virtual assistant connected to node-red and mongodb -MongoDB, which we used as our database -An MQTT broker, which we used for message communication -A 9V battery, used as a power supply for the ESP-32

<Final Project Is A/C on?> Copyright (C) <2023> <Harry Márquez and Jessy Rodríguez>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

General State Machine

The general state machine for the temperature measurement project is divided into four states machines: set up, measure temperature, measure occupancy, temperature display via mqtt/node red, and shutdown.

1) Set up Machine State

a. Initialize system

In this state the system is initializing all the components and connecting to the computer via usb.

b. Connect to Wi-Fi

When the esp32 is connecting to wifi it takes a few seconds and once it is completed, we can read the measure of temperature.

c. Connecting error

If the system has an error when it is initializing or connecting wifi it can display an error and reinitialize the system.

2) Measure Temperature Machine State

a. Read voltage Measure.

Once the esp32 is connected to Wi-Fi and everything is turned on correctly, we can take the voltage reference measurement for our temperature.

b. Calibrate temperature.

in this state with our voltage reference and some measurements previously made, we can calibrate the temperature to be able to know if the air conditioning is on and to be able to send it to the user via node red.

c. Send Temperature to Node red.

By having the temperature measured by the sensor ready, we use mqtt and node red to present it to the user on harryjessy.site.

3) Measure Occupancy

a. Read occupancy

In this state, with a proximity sensor connected to the esp32, we can measure how many people have entered the room and be able to present the number of people to the user.

b. Error in measurement
The system goes to this state if an error occurs in the room occupancy measurement, it can occur due to a cable or lack of synchronization.

c. Display to user
Once we have the occupancy measurement ready and the time required to make the display has passed, the esp32 using mqtt and node red presents the occupancy in the room in that period.

4) Display

a. Connecting to node red: harryjessy.site
To make the display we have to connect to our node red and mqtt which is the means to send the message from one side to another

b. Connecting error
The system enters this state if it understands that there is a problem with time or the connection to mqtt and node red. If we enter this state, we need to re-measure the temperature and occupancy to give a more accurate measurement.

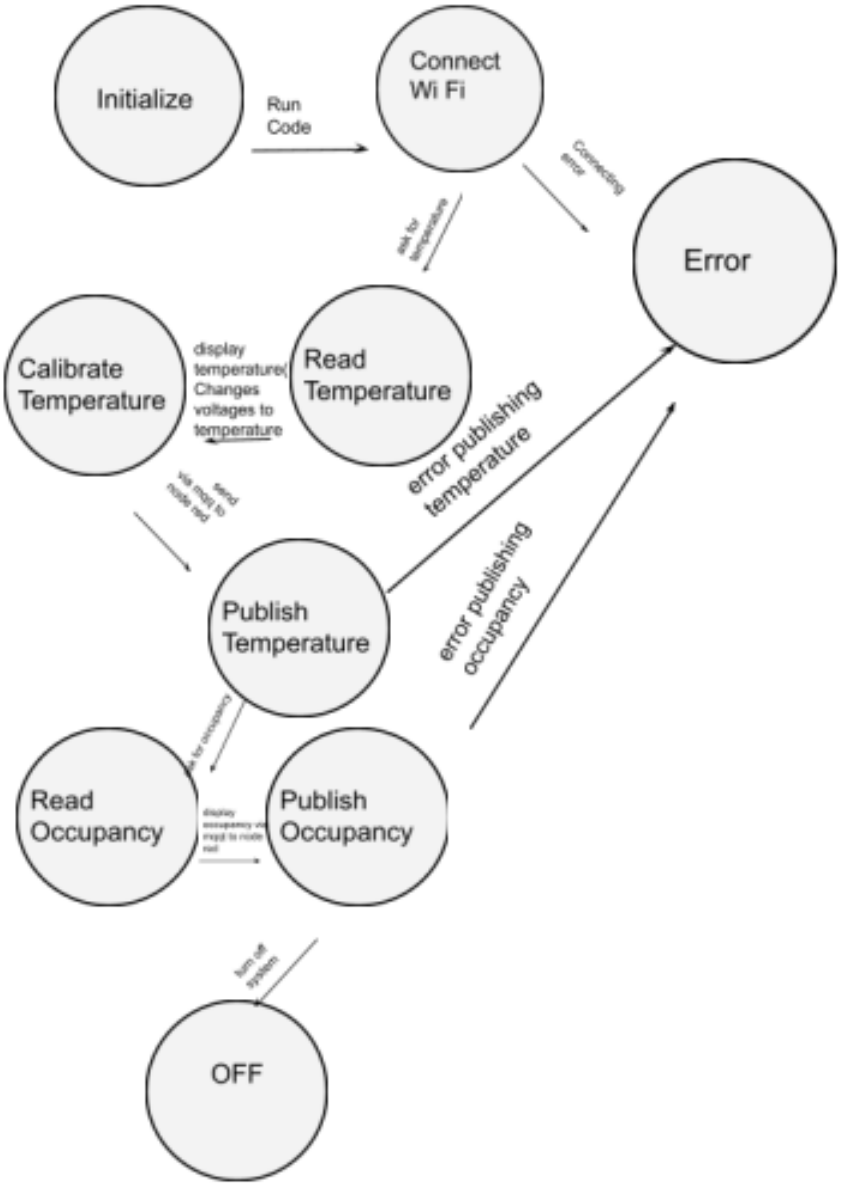
c. Displaying Temperature to User
In this state it is understood that there were no errors in the process and the esp32 can display the temperature and occupancy of the room with the necessary waiting time over and over again.

5) Off

a. Turn off system

General State Machine Diagram

The state machine diagram for the temperature measurement project is presented below:



Software and tools required for the project:

The materials used for the implementation of this project were:

- Visual Studio Code, which we used as our development environment
- PlatformIO, an extension of VS Code that helped us with ESP32 programming
- An ESP32 starter kit, which provided us with the necessary hardware
- LMT-84 temperature sensor
- HC-SR04 distance sensor
- Breadboard and jumper cables
- Red LED, which we used for visual output
- AWS, which we used for cloud computing
- Puttygen and whois.com, which we used for SSH access to the ESP32
- Windows PowerShell, for executing commands on the ESP32
- Node-RED, which we used for IoT automation
- Studio 3T, for managing the database
- Siri virtual assistant connected to node-red and mongodb
- MongoDB, which we used as our database
- An MQTT broker, which we used for message communication
- A 9V battery, used as a power supply for the ESP-32

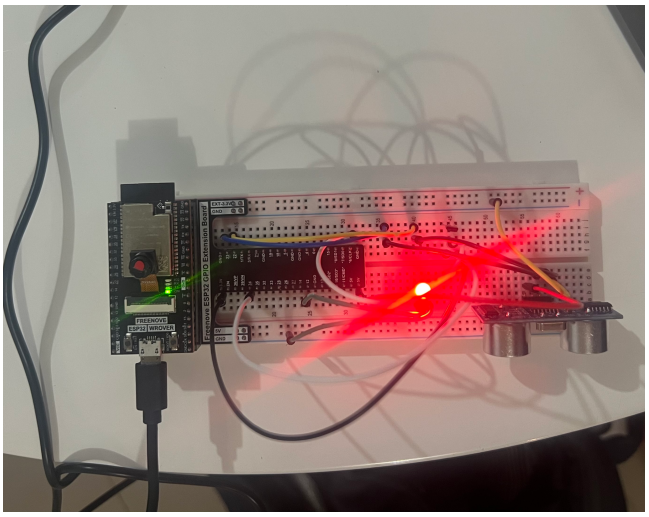


Figure 1: Circuit of the project

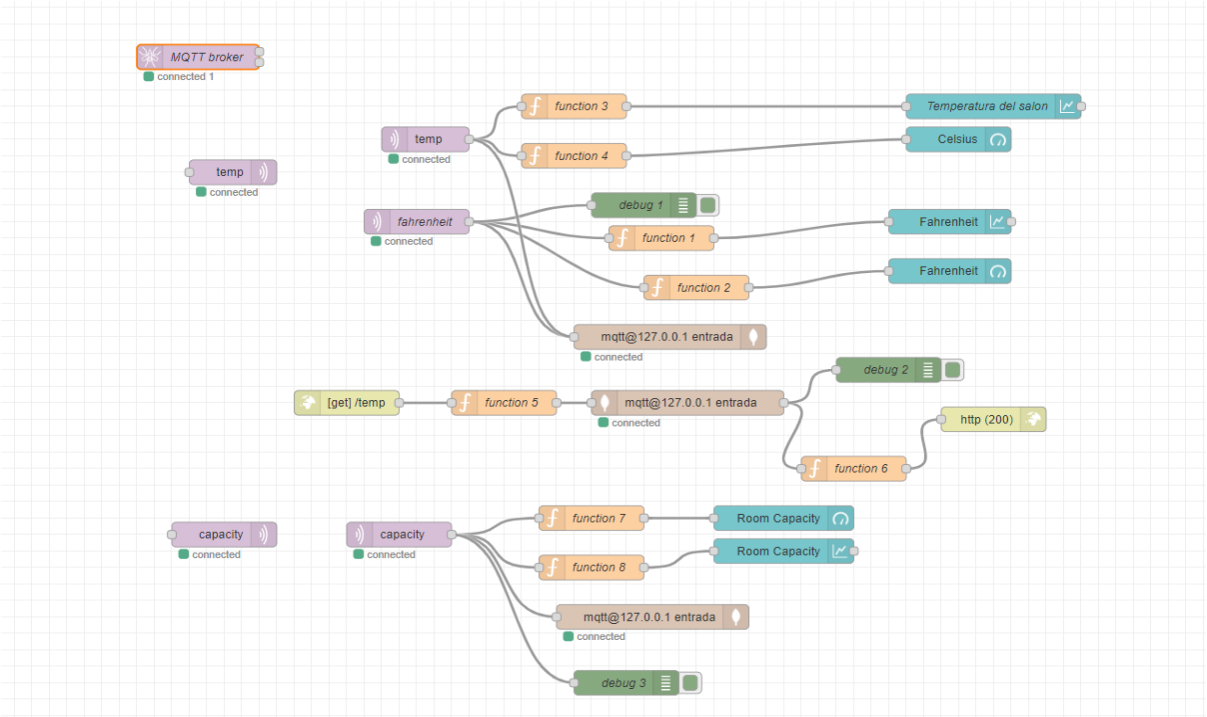


Figure 2: Node Red Flow Chart

Figure 4: Display of Node Red

Code:

```
#include <stdio.h>
#include <Arduino.h>
#include "WiFi.h"
#include <PubSubClient.h>
#include "driver/gpio.h"
#include "driver/adc.h"

// WIFI connection information
#define WIFI "Harry iPhone" // wifi name
#define password "harry123" // wifi password
#define timeout 20000 // connection waiting time

// server connection information
#define server "harryjessy.site" // website name

// temperature sensor information
#define SENSOR_PIN  34 // output pin for temperature sensor
#define CONVERSION  1.8 + 32 // raw value conversion to voltage

// distance sensor information
#define TRIG 23 // trigger pin
#define ECHO 22 // echo pin

#define led 12 // led pin

const char *Temp = "temp"; // for publishing temperature to website
const char *Capacity = "capacity"; // for publishing room capacity to website

void wificonnection () // connection to wifi
{
  Serial.print("Connecting to WiFi\n");
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI, password);

  unsigned long startAttemptTime = millis ();

  while(WiFi.status() != WL_CONNECTED && millis() - startAttemptTime < timeout){
    Serial.print(".");
    delay(100);
  }

  if(WiFi.status() != WL_CONNECTED){
    Serial.println("Connection to WiFi failed\n");
```

```
  else{
    Serial.print("Connection to WiFi succesful\n");
    Serial.println(WiFi.localIP());
    digitalWrite(led, HIGH);
  }
}

WiFiClient espClient;
PubSubClient client(espClient);

void serverconnection() // connection to mqtt broker
{
  client.setServer(server,1883);

  while (!client.connected())
  {
    Serial.println("Connecting to server");

    if (client.connect("ESP32Client"))
    {
      Serial.println("Connection to server succesful\n");
    }
    else
    {
      Serial.print("Connection to server failed\n");
      Serial.print(client.state());
      delay(2000);
    }
  }
}

void setup()
{
  Serial.begin(9600);
  wificonnection();
  serverconnection();
  adc1_config_width(ADC_WIDTH_BIT_12);
  adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_DB_11);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(led, OUTPUT);
}
```

```
void loop()
{
  wificonnection();
  serverconnection();
  delay(2000);

  long duration, distance;
  int capacity = 0;

  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  duration = pulseIn(ECHO, HIGH); // measures pulse duration from signal changes
  distance = duration * 0.034 / 2; // changes duration to distance in cm

  if (distance > 250){  // if distance is less than 250cm
    Serial.println("Person detected"); // prints if someone was detected
    capacity++;
    delay(500);
  }
    printf("Room Capacity: %d\n", capacity); // prints room capacity


  uint32_t adc_raw_value = adc1_get_raw(ADC1_CHANNEL_6); // converts input analog voltage to digital value
  float voltage = (float)adc_raw_value / 4095 * 3.3; // converts digital value to voltage

  float tempcel = (voltage - 0.40) * 100; // voltage conversion to celsius
  float temp = tempcel * CONVERSION; // celsius to fahrenheit

  printf("Temperature: %.1fC\n", tempcel); // prints temperature in celcius
  printf("Temperature: %.1fF\n", temp); // prints temperature in fahrenheit

  if (temp < 80) {
    printf("AC is on\n"); // AC is on if temperature is less than 80F
  } else {
    printf("AC is off\n"); // AC is off if temperature is greater than 80F
  }
```

```
    client.publish(Temp, String(temp).c_str()); // publishes temperature in fahrenheit to mqtt broker
    client.publish(Capacity, String(capacity).c_str()); // publishes room capacity to mqtt broker
    client.loop();
    delay(1000);

  }
```

Process Creating Server:

- Create an account in AWS and go to Lightsail to create an Ubuntu instance with a static IP address.
- Open the remote server and check the SSH key created by the server.
- Use Puttygen to generate a public key that will be stored both on the cloud server and on your local computer.
- Generate a private key that remains on the local computer.
- Save the two private keys and name the cloud server's IP address.
- Search the Internet for Whois.com and buy a domain name.
- Add a direction in the domain, the static IP address created for the cloud server.
- Once the process is complete, open Windows PowerShell and enter the SSH key using the server name oriented with the domain name

Implementing Node Red:

- In order for MQTT to connect to the server, the user needs to enable ports 443 and 1880-1885 on the server.
- Through the commands provided by the professor and the web, Node-RED can be successfully installed on the server.
- Additionally, the npm and admin packages need to be installed for Node-RED to work properly.
- In order to run Node-RED, a command must be run on the server.
- Once installed, users can open Node-RED in their browser by typing harryjessy.site:1880 into Google.
- Once Node-RED is turned on, it receives messages and feeds the graph with nodes.
- Use this program to create a functional dashboard that displays temperature data in a graphical format. Access this dashboard from any device by simply searching for harryjessy.site:1880/ui in your browser.

Connecting MQTT:

- Install the required packages for MongoDB on the server.
- Start MongoDB and make sure it is running properly.
- Download Studio 3T and create an SSH tunnel between the server and the MQTT broker.
- Add PubSubClient library to establish connection between ESP32 and MQTT broker.

Ios Shortcut:

- Open the Shortcuts app on your iPhone.
- Click the "+" button to create a new shortcut.
- Click Actions and look for "Get Contents of URL".
- In the Configuration section, paste the URL of the website (harryjessy.site:1880/temp) whose content you want to read.
- Click Actions again and look for Read text.
- Configure the Read Text action to speak content retrieved from a URL.

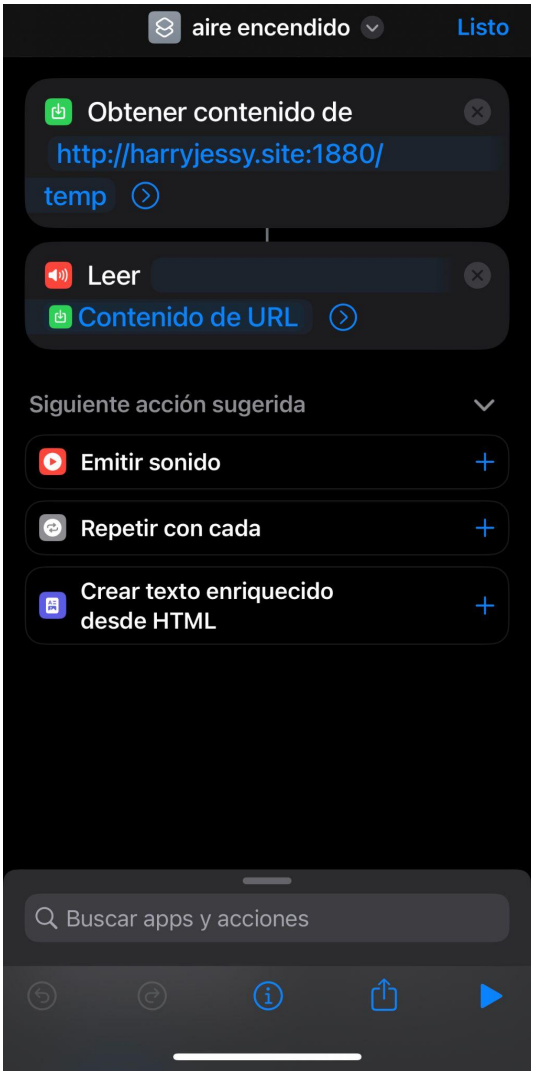


Figure 5: Ios Shortcut

Conclusion

All goals and necessary results were achieved in the project. The circuit and code were designed and implemented successfully for the reading of temperature and room capacity. Both of these measures were successfully sent to the server where mqtt broker received the data and shared it with node-red. Throughout the nodes, node red published the data in a display available for the user to access and read. Finally, if the user uses a virtual assistant like Siri and asks “is the A/C on?”, Siri will respond with the room temperature and will answer the question if the air conditioner is on or off.