INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Department of Electronic, Telecommunications and Computers Engineering

Home Energy Management System

Paulo Miguel Monteiro Rodrigues | 47118

Carlos Miguel Freire Santos | 45938

Advisor | Professor Rui António Policarpo Duarte

Intermediary Report for the Curricular Unit of Final Course Project of Bachelor's Degree in Electronic, Telecommunications and Computers Engineering

April 2024

Table of Contents

Figure List

Table List

1. Introduction

In the current context of seeking sustainable and efficient solutions in the energy sector, European energy cooperatives promote efficiency by implementing innovative systems.

Therefore, it is necessary to optimise energy consumption at home, using systems capable of controlling home appliances to operate during periods of greater energy availability. Since such systems make use of personal data and closed systems are prone to hacking, serious concerns are adopting such systems.

The project aims to create a residential open-source system incorporating sensors to track energy consumption, offering users insightful feedback, and making it possible for users to make informed decisions regarding energy usage, thereby minimizing waste, as well as cutting down on electricity costs.

1.1. Motivation

The primary motivation of this project is to furnish users with a system capable of delivering insights into their household electricity consumption. The aim is to provide users with information, enabling them to make automated decisions regarding their electrical usage and maximise renewable energy sources integration.

This project is based on a previous study made by REScoop that provides the idea of a controlled environment at home that can see and make decisions about the energy costs of each appliance. This article doesn't provide a final product and doesn't specify which components are better for the working of this system. The idea for this project is to publish an open-source working system that can grow along with the IoT community and that doesn't have obstacles to growing in its implementation.

The project's goal is to assist homeowners in optimising the efficiency of renewable energy sources, such as solar panels.

1.2. Objectives

The project aims to implement an embedded circuit, designed to analyse each section's energy consumption within a household. This initiative seeks to effectively reduce electricity usage.

The data collected by the integrated circuit in each section of the house is transmitted to an application called "Home Assistant", deployed on a Raspberry Pi image, which will be referred to as "cofy-box" throughout the report. This application facilitates the analysis of each monitored section, providing insights into the consumption of individual components. The "cofy-cookie" system provides details on energy consumption thus enabling later analysis and insights on how to save energy.

1.3. System Requirements

The system requirements for the project are as follows:

- Analyse the current and voltage flowing in an electrical line using the *PZEM-004T V3* sensor.

- Use an ESP32 MCU to transmit data obtained from the sensor via Wi-Fi to the "Home Assistant" application.

- In the application, installed on a Raspberry Pi image, it is possible to analyse the data sent from different areas under study.

- From the application, send a message back to the desired microcontroller to turn the respective device on or off.

- Connected to the microcontroller is also a Solid State Relay (SSR) that implements the instruction given to the ESP32 (MCU) by "Home Assistant."

- Setting up a Wi-Fi router, which will serve as the base for the system to work.

In **Figure *1***, it is possible to see the initial project scheme.

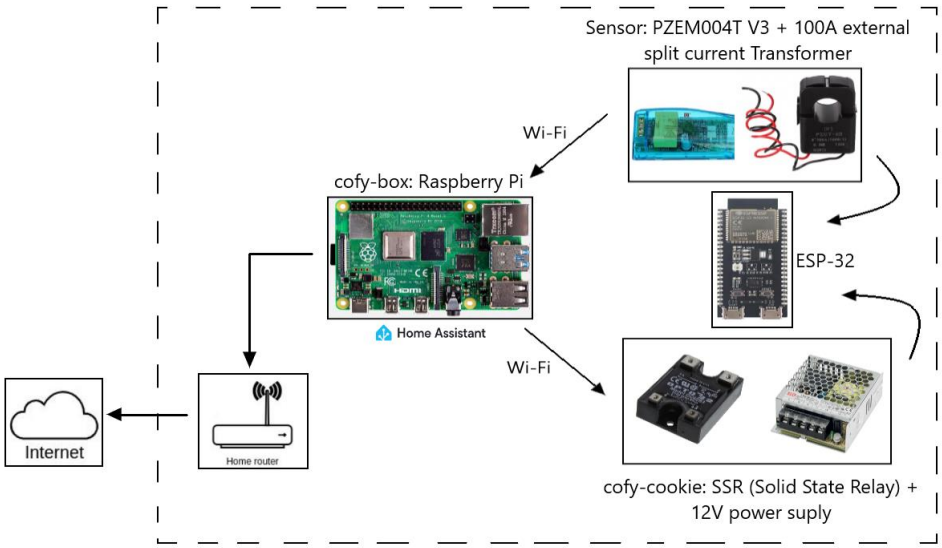

Figure 1. *Initial Project Scheme.*

Besides using the Solid State Relay shown in the previous figure, there is also the opportunity to study the use of a relay.

Subsequently, when implementing the “Home Assistant” application, the usage of a *Raspberry Pi* or an *Orange Pi* will be a task undertaken during the selection process to identify the most optimal solution.

2. Background

This section is dedicated to explaining the foundation of this project, as it took some inspiration from different articles and reports available online and from some companies that already provide closed-source products that have almost the same purpose as this work. There are some comparisons for different materials that could be used in this project and justifications of why each component is being chosen instead of others to achieve the most efficient, safest, and cheapest final product.

2.1. Related Works

After research and analysis of numerous scientific articles and reports available online, several sources were identified that cover the technologies utilised in this project. While many articles and reports were reviewed, the one referred to below provides a comprehensive explanation of the theoretical basis for the current project. There is also mention of some companies that are dedicated to domotic technologies that already sell products that come close to the project at hand, with closed-source software and hardware.

2.1.1. REScoopVPP

The project report "Smart Building Ecosystem for Energy Communities" by REScoop [1], is focused on the description of an open-source energy monitoring and optimization system. It uses a range of existing open-source technologies to integrate and control legacy and new appliances into a smart building ecosystem. Helps in understanding the Hardware behind the final product as well as real-life options to use the device.

This work delivered an idea of how to get consumption data from different elements of a house and how to deliver them to a user, even though they don't specify which components can make part of the cofy-cookie. For the cofy-box, there are some examples of software that make part of their project, one of them the "Home Assistant" as can be seen in ***Figure 2***.

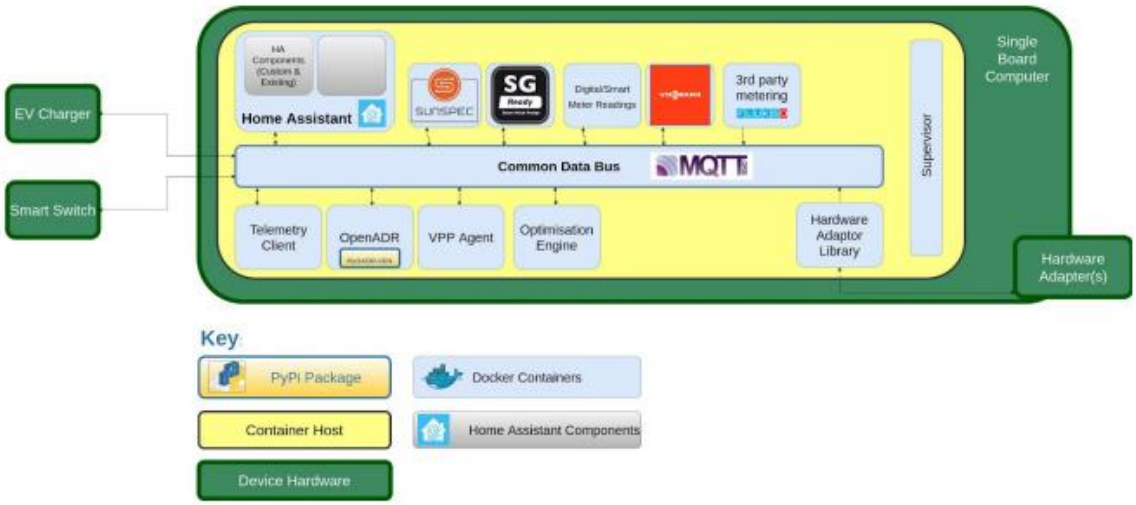

Figure 2. *REScoop's Architecture.*

REScoop made a very simple scheme that allows an easy understanding of how their project is going to work in **Figure *3***, however, they did not specify all the components for its implementation. The challenge for the current project is to find the best hardware components and applications that make this idea the safest and cheapest way to control energy consumption and the working time of household appliances.

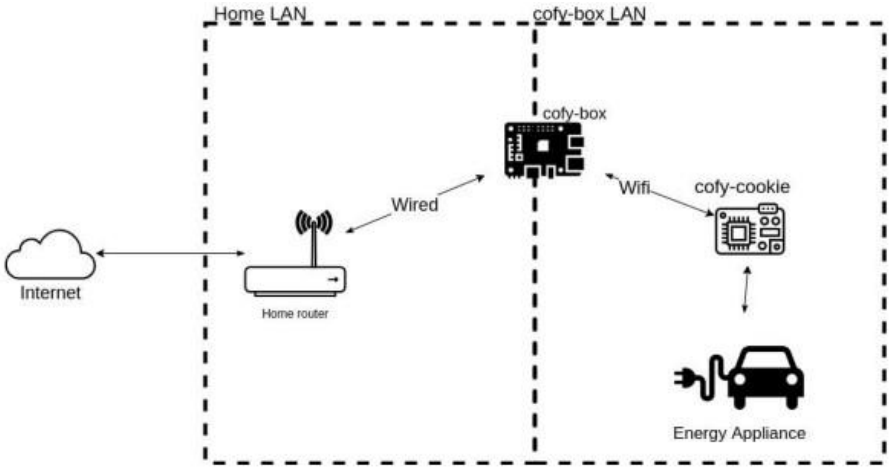

Figure 3. *REScoop's Model*.

2.1.2. Shelly

The advisor of this project provided equipment from Shelly, a company that dedicates their work to many areas, one of them being domotic devices. The devices provided were Shelly Plus 1 and Shelly Plus ADD-ON, shown in **Figure *4*** and ***Figure 5*** respectively. The first device is an interrupter with a relay controlled by Wi-Fi and it can be installed in almost any appliance at home. The other device is used to make measurements of temperature and humidity, light intensity, and movement, for example.

Figure 4. *Shelly Plus 1.*

Figure 5. *Shelly Plus ADD-ON.*

These devices are closed-source, which can present adversity. This project as an open-source aims to become a solution to this issue.

2.1.3. Tuya

Tuya is another company dedicated to providing IoT solutions for many areas, one being house domotic. Different from Shelly, Tuya doesn't provide measurement devices to install in home appliances, but they provide software and equipment to have in a smart and efficient home. As a closed-source product, it doesn't provide the reliability and flexibility that is provided by the proposed project. **Figure *6*** shows one of the most used smart switches by this brand.

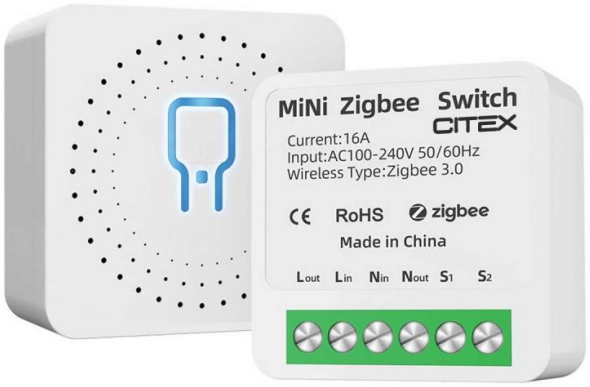

Figure 6. *Tuya Switch.*

2.2. Relevant Technologies

In this section, there is a brief explanation of each component and technologies involved in the project and a comparison between components to show which ones are the best possible for this project.

2.2.1. Single Board Computer

The communication between a household appliance and the user is facilitated through an application installed on a Raspberry Pi. Specifically, the application utilized is Home Assistant, with the scientific report [13] detailing the ease of its implementation on *Raspberry Pi* and the advantages it offers over alternative applications. This report shares many similarities with the project being discussed, providing ample evidence that the Home Assistant is a good method of exploring IoT within a domestic setting. **Table 1** shows the comparison.

Single Board Computer Comparison		
Model	Raspberry Pi 4	Orange Pi zero
RAM Memory	4GB	0.5GB
CPU Speed	4 x 1.8 GHz	4 x 1.2 GHz
Wi-Fi Version	Wi-Fi 4 (802.11n); Wi-Fi 5 (802.11ac)	Wi-Fi 4 (802.11n)
Bluetooth	yes	no
HDMI Output	yes	no
USB Type-C	yes	no
Operating Voltage	5V	5V

Table 1. *Single Board Computers considered in the Initial Study.*

Raspberry Pi 4 was the chosen computer to use in this project, there was the option to use an *orange Pi Zero*, but after studying the hardware, it was decided that it wasn´t recommended due to the lack of computer power. *Raspberry PI 4* also has better and newer features, for example, support for newer Wi-Fi protocols, Bluetooth, HDMI output and USB type-C. In **Figure** *7*, you can see the comparison made between the 2 computers and better understand the reasoning for this choice. This comparison was based on [7].

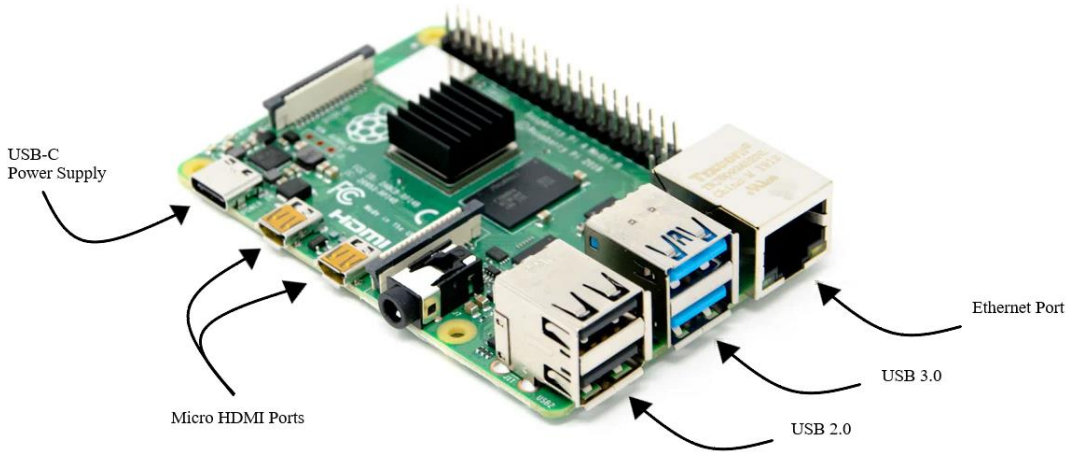

Figure 7. *Raspberry Pi4 used in the project.*

2.2.2. Microcontroller with Wi-Fi

The project needs a microcontroller to implement the cofy-cookie. It will gather information from the sensors and communicate them with the Home Assistant serving on the SBC. The microcontroller needs to be able to talk to the server, so it needs Wi-Fi.

The article specified in [4] explains in detail the ESP-32 and its use, which will be an important device in this project. This microcontroller already has a Wi-Fi module and is connected to BLE (Bluetooth Low Energy) via a chip, so it is very powerful and can be a good choice for creating an IoT application system. In [5], there is also the comparison between different versions of the ESP32, so it was also used for this project.

This last one along with [8], helped extract a comparison table between the *ESP32-S2*, *ESP32-S3* and *ESP32-C3*, which were the models frequently used for this kind of project.

Microcontroller Comparison			
Model	ESP32-S2	ESP32-S3	ESP-C3
Frequency (MHz)	240	240	160
SRAM (KB)	320	512	400
ROM (KB)	128	384	384
Flash (MB)	Up to 4	Up to 8	Up to 4
SPI (MB)	4	4	3
USB	Yes	Yes	Yes
Wi-Fi	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz	802.11 b/g/n, 2.4 GHz
Bluetooth	No	Yes	Yes
GPIO	43	45	22
CPU	single-core	dual-core	single-core
Price (€)	7.52	14.1	8.46

Table 2. *Microcontroller Comparison.*

According to **Table 2**, the most efficient choice for the project will be the *ESP32-C3*. The C3 contains sufficient ROM memory and SRAM capacity, offering the necessary resources for interfacing with various peripherals. The C3 also has Bluetooth and the S2 does not have this feature, even though this project is very unlikely to use a Bluetooth connection since the use of Wi-Fi is optimal for further distances. Moreover, the *ESP32-C3*, as seen in **Figure *8***, provides an efficient performance with its single-core CPU, when compared to a dual-core CPU. Therefore, considering these factors, the *ESP32-C3* emerges as the most suitable option for our project.
The project also incorporated a solution for using the ESP32-S2 microcontroller, as seen in **Figure *9***, which was solicited to be also considered by this project's advisor.

Figure 8. *Esp32-C3 used in the project.*

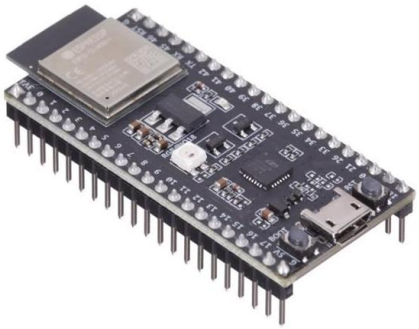

Figure 9. *Esp32-S2 used in the project.*

2.2.3. Temperature and Humidity Sensor

A component integrated into the "cofy-cookie" is the temperature and humidity sensor. This sensor interfaces with an MCU ESP32, with specific parameters outlined in this paper [2]. Although this work primarily evaluates the connection between the sensor and an Arduino, understanding the necessary response times for accurate data is important to understand the use of this component.

The research made, and based on [9], showed two sensors with the potential to be used. **Table *3*** shows the different specifications of each.

Humidity and Temperature Sensors		
Model	DHT11	DHT22
Measures	Temperature and Humidity	Temperature and Humidity
Interface	One-wire	One-wire
Supply Voltage	3 to 5.5V	3 to 5.5V
Current Supply	0.5 – 2.5 mA	1 - 1.5 mA
Temperature Range	0º to 50ºC	-40 to 80ºC
Resolution	Humidity: 1% ; Temperature: 1ºC	Humidity: 0.1%; Temperature: 0.1ºC
Price (€)	6,99	10,99

Table 3. *Humidity and Temperature Sensor Comparison.*

The following figures, **Figure *10*** and **Figure *11*** represent the temperature and humidity sensors compared in the previous table.

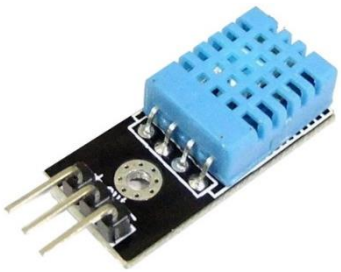

Figure 10. *DHT11.*

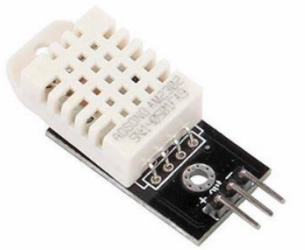

Figure 11. *DHT22.*

For communication, both DHTs use one wire which makes it easier to use, than SPI. Even though, in DHT11, the temperature range does not include negative temperatures and the resolution is less accurate due to having a bigger interval, this sensor is sufficient for the case study as the ambient in which the work will be performed does not require more specification.

The MCU utilized in the cofy-cookie is an ESP32, operating at a voltage of 3.3V. Similarly, the temperature and humidity sensor DHT11 employed in this project operates at the same voltage level as the MCU.

2.2.4. Power Consumption Sensor

To keep track of the power consumption on an electric wire, or line, it is necessary to use a sensor, which was decided would be the *PZEM-004T V3*.

The *PZEM-004T V3*, represented in **Figure 13**, is a multifunctional sensor module that functions to measure power, voltage, current, and energy contained in an electric current. The paper specified in [3], explains a lot about the use of this device and will be important to understand how to implement it in the project at hand.

PZEM-004T V3 has a serial UART (Universal Asynchronous Receiver/Transmitter) that works with 5V, to allow the communication to work at 3.3V with a board such as *Raspberry Pi* or *ESP32* a resistance of 1kΩ must be used, so with this modification, UART can work at 5V or 3.3V [6].

Another option could be the power consumption sensor represented in **Figure 12**. The downsides of this sensor are that the wire has to pass through the printed circuit board (PCB), making its usability more complex, and the fact that it does not have a box that isolates the 220V, which implies a more careful experiment.

Therefore, the PZEM-004T V3 will be the chosen sensor for the current project. It was provided by the advisor of this project and after exhaustive research it was concluded that this module is the most utilized in this type of work, for its accuracy and efficiency.

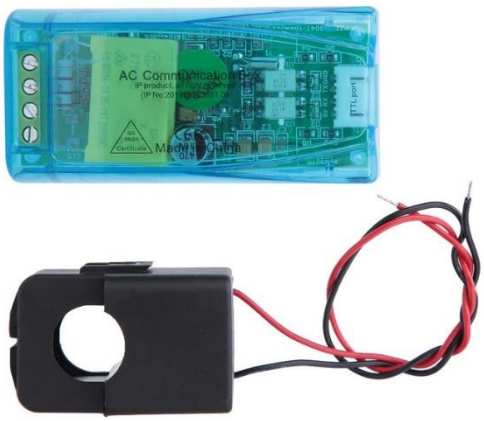

Figure 12. *PZEM-004T V3.*

Figure 13. *JSY-MK-109.*

2.2.5. Light Sensor

It was suggested to add a light sensor to monitor the ambient light as this may be convenient in the future to optimize for energy savings during daytime, which after investigation was a simple component of a Light Dependent Resistor (LDR).

The LDR, as can be seen in **Figure 14**, is a component form that changes resistance depending on the amount of light. It has a high value when no light is present. The value of resistance of the LDR depends on the type as the light level increases the resistance drops, which makes the current increase (by Ohm's Law).

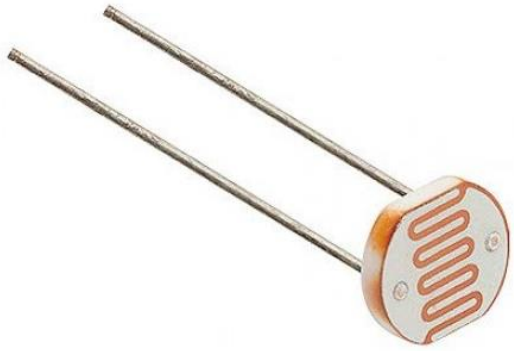

Figure 14. *LDR used in the project.*

2.2.6. Electronic Switching Device

To make the user's instructions happen, there is a need for a switching device to be able to control the appliances through the application.

By researching on the internet, there were a few options to implement. The most common were the Solid State Relay and the Mechanical Relay.

Solid State Relays use semiconductor devices for switching, which means that they consume very little power. They can operate on as little as a few milliwatts. Mechanical Relays, on the other hand, use electromechanical contacts to switch the current. This process requires more power [10]. Therefore, the project will include the Solid State Relay, represented in **Figure 15**.

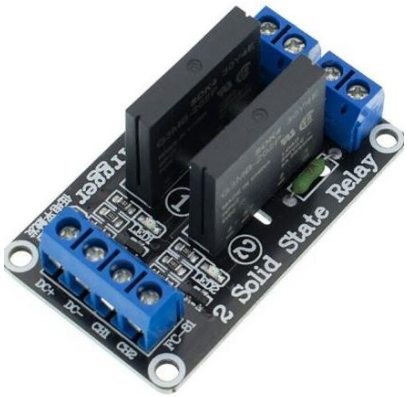

Figure 15. *Solid State Relay.*

2.2.7. Domotic Application

There was the need for an application to give the user the ability to control household devices and with an interface to see each sensor's measurements. Among these, two applications stand out: Home Assistant and OpenHAB.

While there are minimal differences between them, Home Assistant holds a notable advantage over OpenHAB due to its extensive array of open-source programming options. Additionally, Home Assistant is recognized for its robust automation engine, user-friendly interface, and powerful integrations, yet not by a significant margin. Therefore, Home Assistant will be employed for this project to maintain alignment with the original concept.

2.2.8. Communication Protocol

After careful consideration, we opted for the MQTT protocol to allow communication between the device and the server. MQTT is extensively utilized in the IoT industry due to its lightweight and efficient nature[11]. This efficiency enables devices to conserve energy more effectively in comparison to other protocols such as HTTP[12]. MQTT achieves this by employing a smaller packet size and lower overhead than its counterparts.

3. Proposed Home Automation System

This chapter provides the proposed system architecture. It begins with an introduction to the system overview, which includes a description of the main components and the interaction between them. The next sections focus on each component that was chosen and explain why were they chosen in the previous chapter.

3.1. System Overview

After researching all the alternatives for the elements to use in this project, **Figure *16*** represents, the block diagram to be implemented for this work. It also represents the working state of this project.

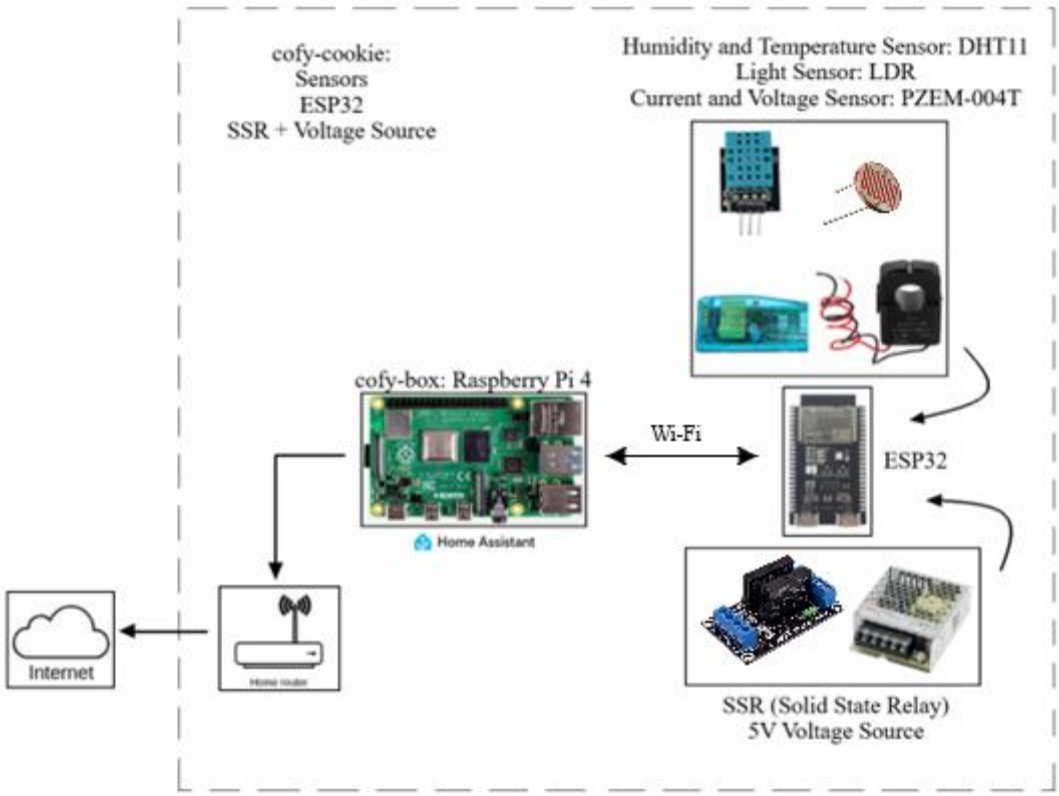

Figure 16. *System Overview.*

The previous figure gives an image of the system itself. The top right of the figure shows the cofy-cookie that contains the sensors. These communicate with the ESP32, which is located right under them in the figure. The ESP32 then sends the information through Wi-Fi to the Raspberry Pi 4 that has the home assistant application running, allowing the user to see the measurements taken by the sensors, and to send back information for the cofy-cookie with the SSR (bottom right of the figure). These allow the user to take action based on the information. The Raspberry Pi 4 is also connected to a home router, configured by the authors, to connect it to the internet as seen on the left side of the figure.

3.2. Cofy-Box

The cofy-box serves as the intermediary between the user and the cofy-cookie, facilitating communication. It consists of a *Raspberry Pi 4*, a single-board computer, with a Home Assistant image to oversee all cofy-cookie measurements, installed in an SD Card that must have at least 32GB of space. The decision to utilize a single-board computer for the cofy-box is driven by the goal of reducing expenses, conserving energy, and optimizing memory usage in handling incoming data from the cofy-cookie and user interactions.

3.3. Cofy-cookie

To have a working project that can be compared with the schematics previously presented, and as this work is an open-source project, this section explained how each component connects to others to have a functional cofy-cookie. Arduino IDE will be used to program both ESP32.

3.3.1. ESP32-S2

The microcontroller used for the system will be the ESP32-S2, as said before. **Figure 17** shows the overview of each block of the MCU.

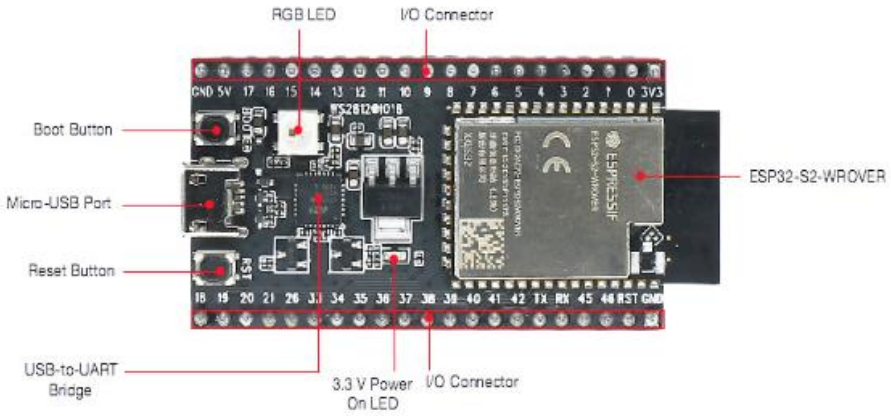

Figure 17. *ESP32-S2 Overview.*

Figure 18 shows the pinout of the chip, where it is possible to see where each pin is located, and the availability of so many GPIO ports, as well as a 3.3V pin and a 5V pin, 1 reset pin, and 2 grounds. It is not possible to see in the figure, but there is an Rx in pin 44 and a Tx in pin 43.

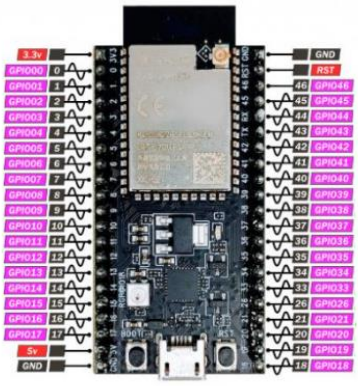

Figure 18. *ESP32-S2 Pinout.*

3.3.2. ESP32-C3

In this project, it was allowed to work not only with an ESP32-S2 but also with an ESP32-C3. This work as it does not have many components to connect to the microcontroller it is easy to use a micro with reduced dimensions as this one. As the programming and the choice of every pinout is equal for each ESP it was decided to prove that this project does not only work for a specific type of microcontroller.

As it is possible to see in **Figure *19***, there are much fewer GPIO pins in this microcontroller than in the ESP32-S2 but there are enough for the project that is being designed because it will only be necessary to use two of the GPIO pins, 3.3V, ground, Tx and Rx pins.

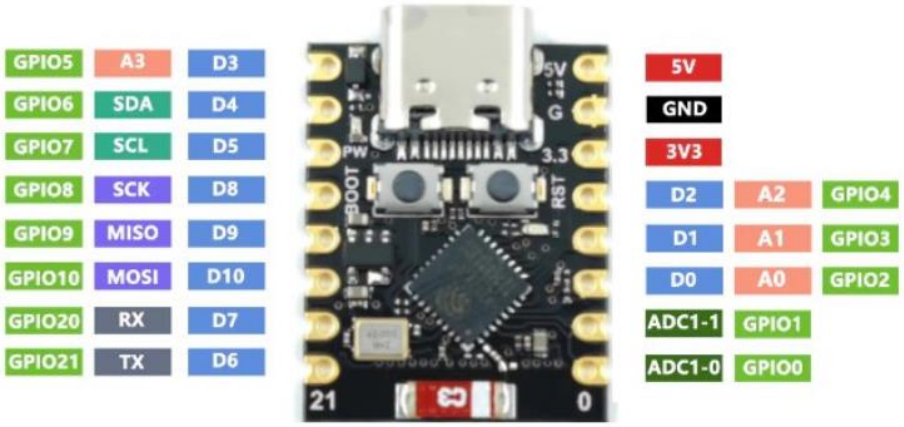

Figure 19. *ESP32-C3 Pinout.*

3.3.3. LDR

The Light Dependent Resistor (LDR) is just another special type of Resistor and hence has no polarity so both pins can be connected in any direction. **Figure *20*** shows an LDR.

Figure 20. *Light Dependent Resistor.*

3.3.4. DHT11

The sensor chosen to measure temperature and humidity was the *DHT11*. This sensor comes with 3 pins to connect to the microcontroller. **Figure *21*** shows the pins and what each pin does. Pin1 from left to right is the VCC, the power supply from 3.3V to 5.5V. Pin2 is the Data pin which outputs both temperature and Humidity through serial data and connects to any GPIO. Lastly, Pin3 is the ground connecting the sensor to the ground of the circuit.

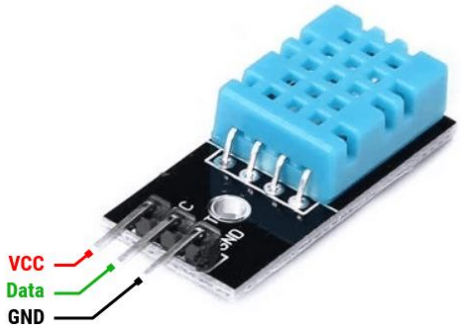

Figure 21. *DHT11 Pinout.*

3.3.5. PZEM-004T V3

The *PZEM-004T 3* will be used to measure current. In ***Figure 22*****,** from bottom to top, the left pins are: Pin1 is the ground, which connects the sensor to the ground of the circuit. Pin2 is the Tx pin that will connect to the Rx pin of the microcontroller, for communication. Pin3 is the Rx pin that will connect to the Tx pin of the microcontroller, and Pin4 is the power supply of 3,3 - 5V. The pins on the right side of the component from top to bottom are: Pin1 and Pin2 connected to the load terminals, being Pin1 connected to the load part that goes through the current transformer. Pin 3 and Pin 4 are the cables that make the connection between the *PZEM-004T V3* and the current transformer.

Figure 22. *PZEM-004T V3 Pinout.*

In the PZEM's manual, there is a table that shows the results of the measurements made by the device. ***Table 4*** shows these results.

Register address	Description	Resolution
0x0000	Voltage value	1LSB correspond to 0.1V
0x0001	Current value low 16 bits	1LSB correspond to 0.001A
0x0002	Current value high 16 bits	
0x0003	Power value low 16 bits	1LSB correspond to 0.1W
0x0004	Power value high 16 bits	
0x0005	Energy value low 16 bits	1LSB correspond to 1Wh
0x0006	Energy value high 16 bits	
0x0007	Frequency value	1LSB correspond to 0.1Hz
0x0008	Power factor value	1LSB correspond to 0.01
0x0009	Alarm status	0xFFFF is alarm, 0x0000is not alarm

Table 4. *Measurements Results.*

3.3.6. Solid State Relay

The Solid State Relay that is being used in this project has two channels, so it is possible to manage two loads at the same time. In ***Figure 23***, the left side pins are: Pin 1 represents channel 1, which can connect to any digital port of the microcontroller, and Pin 2 represents channel 2, which can also connect to any digital port of the ESP. Pin3 is responsible for giving energy to this device as it represents VCC which can operate between 3,3V and 5V, and Pin 4 connects to the ground of the circuit. On the right side pins, the two terminals of each channel are used like a traditional switch.

Figure 23. *Solid State Relay.*

3.4 Proposed Electronic Circuit

After consideration of each component's pins and ways of communication, the proposed circuit schematic can be observed in ***Figure 24***, mounted on two breadboards, and tested.

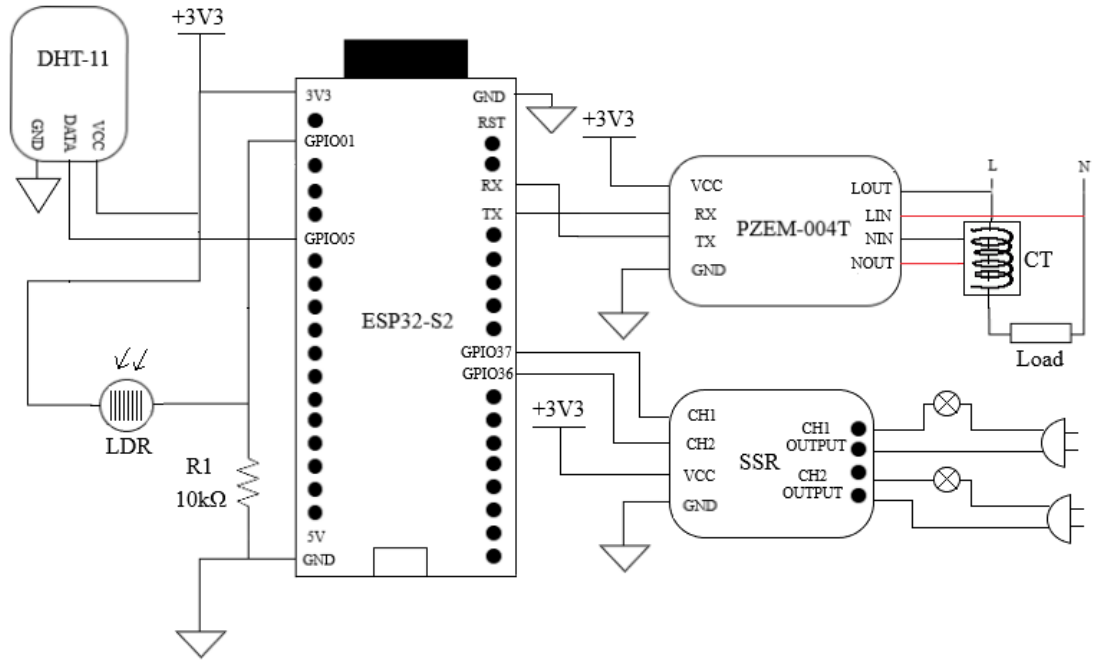

Figure 24. *Electronic Circuit using the ESP32-S2.*

The system will also be implemented using an ESP32-C3, as shown in ***Figure 25***. It is the same circuit, on a different microcontroller. The MCU's are both from the same manufacturer and their development tools are the same. Thus, it is possible to reuse the same development to have alternative implementations.

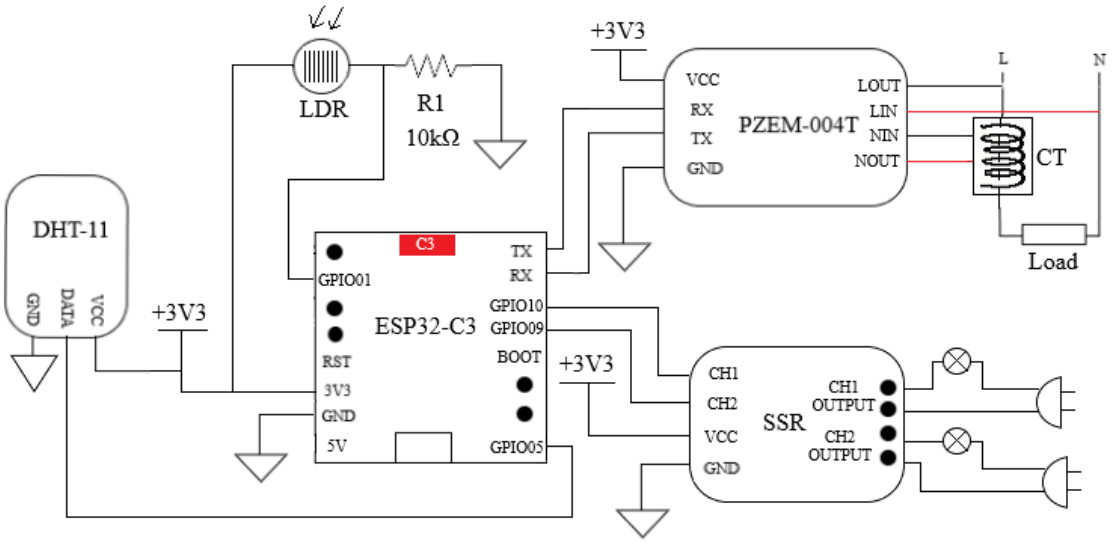

Figure 25. *Electronic Circuit using the ESP32-C3.*

4. Preliminary Results

As for the first results of this project, there were two demos made. The first one is the blinking LED, which was tested on the ESP32-S2. This project was made to gain confidence with the project flow of the MCU tools. The LED of the microcontroller blinked every 0.5 seconds and was successful. The second demo was to test the DHT11 and was also successful giving the temperature and humidity, and even showing when the temperature differs. This second demo can be seen in **Figure 26**.

```
Temperature: 27 °C      Humidity: 17 %
Temperature: 28 °C      Humidity: 17 %
Temperature: 28 °C      Humidity: 17 %
Temperature: 28 °C      Humidity: 17 %
Temperature: 28 °C      Humidity: 17 %
Temperature: 27 °C      Humidity: 17 %
Temperature: 28 °C      Humidity: 17 %
```

Figure 26. *DHT11 Demo on ESP32-S2.*

As a first test of trying to run the home assistant on the Raspberry Pi 4, there was a successful attempt by turning the server on running the home assistant, and connecting it to the HUE Lights of one of the authors making it possible to control the lights using the home assistant. This trial is shown in **Figure 27**.

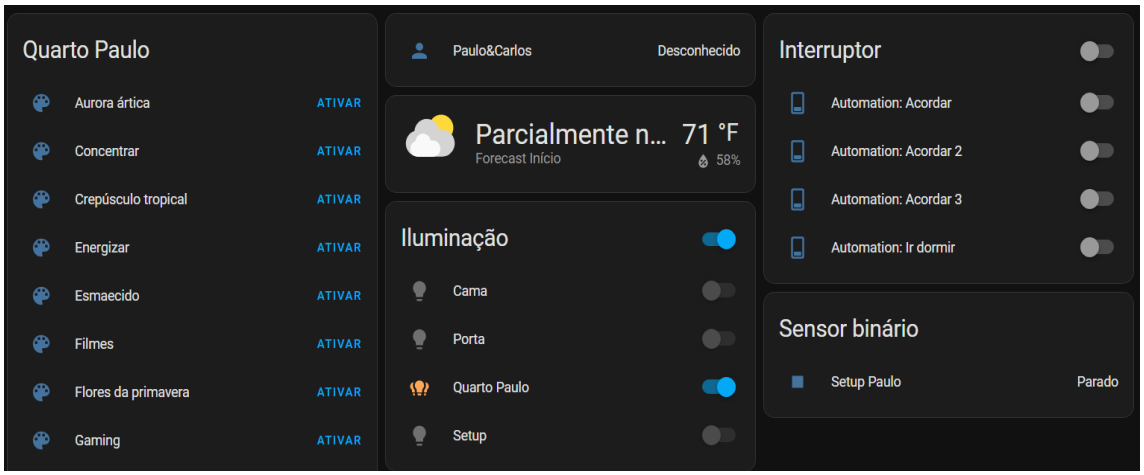

Figure 27. *Home Assistant Trial.*

The figure shows in the middle which light the user wants to turn on or off, in the left side shows the preset lights to turn on every light. The right side is a different kind of automation like what hour to turn on the lights.

5. Conclusion & Future Work

This report indicates the suggested solution for the project.

This project initiated a survey of the required hardware specifications and defined specific milestones to achieve the goal. Then proceed to the implementation of the server and establish connections between the server and other components of the project, considering energy efficiency.

The system then needs to be tested to verify the connectivity and communication between the server and each component. There is also the need to conduct tests to ensure the accuracy of measurement readings, and if the cofy-cookie is actively doing what the user wants it to do.

The expectation is that a prototype is tested and verified, to make this possible.

As for future work, there is a necessity for a demo of each component running on the ESP32. The circuit needs to be developed for the different sensors, as well as the firmware for the cofy-cookie according to the project specificities and compatibility with the Home Assistant. Additionally, the ESP32 and the server need to be connected, with a deeper study of the internet protocol necessary to perform the connection. Finally, the whole system will be tested and presented.

6. Project Calendarization

This chapter details the project's current timeline, taking into account modifications that have been made since the beginning of the project. It serves as a roadmap for what remains to be done, following the future work referenced in the previous chapter.

Figure 28 shows the work plan for the project, divided by tasks.

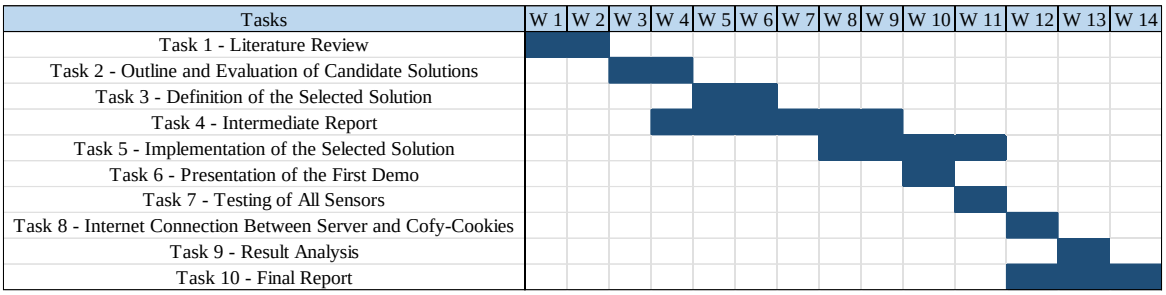

Figure 28. *Work Plan.*

7. References

[1] Aylott, Ben, et al. DELIVERABLE D3.1 Final Design/Specification Project Title: Smart Building Ecosystem for Energy Communities.

[2] Srivastava D., Kesarwani A., & Dubey S. (2018). Measurement of Temperature and Humidity by using Arduino Tool and DHT11. *International Research Journal of Engineering and Technology*, 5(12), 876-878. www.irjet.net

[3] Harahap, P., Pasaribu, F. I., & Adam, M. (2020). Prototype Measuring Device for Electric Load in Households Using the Pzem-004T Sensor. Budapest International Research in Exact Sciences (BirEx) Journal, 2(3), 347-361.

[4] Pratama, Erik Wahyu, and Agus Kiswantono. "Electrical Analysis Using ESP-32 Module in Realtime." JEECS (Journal of Electrical Engineering and Computer Sciences), vol. 7, no. 2, 13 Jan. 2023, pp. 1273–1284, https://doi.org/10.54732/jeecs.v7i2.21.

[5] Rocha, Miguel, and Pedro Silva. IoT System for pH Monitoring in Industrial Facilities. July 2023.

[6] Peña, Eric, and Mary Grace Legaspi. "Uart: A hardware communication protocol understanding universal asynchronous receiver/transmitter." Visit Analog 54.4 (2020): 1-5.

[7] "Orange Pi Zero Plus vs Raspberry Pi 4 Model B: What Is the Difference?" VERSUS, versus.com/en/orange-pi-zero-plus-vs-raspberry-pi-4-model-b. Accessed 20 Mar. 2024.

[8] "Chip Series Comparison - ESP32-S3 - — ESP-IDF Programming Guide V5.0 Documentation." Docs.espressif.com, docs.espressif.com/projects/esp-idf/en/v5.0/esp32s3/hw-reference/chip-series-comparison.html. Accessed 20 Mar. 2024.

[9] JackSoldanoJacksoldano.com. "Sensor Comparison: DHT11 vs DHT22 vs BME680 vs DS18B20." Instructables, www.instructables.com/Sensor-Comparison-DHT11-Vs-DHT22-Vs-BME680-Vs-DS18/. Accessed 21 Mar. 2024.

[10] M1. "Solid State Relay vs. Mechanical Relay - What Is Different?" GEYA Electrical Equipment Supply, 18 Oct. 2022, www.geya.net/solid-state-relay-vs-mechanical-relay-what-is-different/.

[11] Khan, M.A.; Khan, M.A.; Jan, S.U.; Ahmad, J.; Jamal, S.S.; Shah, A.A.; Pitropakis, N.; Buchanan, W.J. A Deep Learning-Based Intrusion Detection System for MQTT Enabled IoT. Sensors 2021, 21, 7016. https://doi.org/10.3390/s21217016

[12] Yifeng Liu and Eyhab Al-Masri. 2022. Slow Subscribers: a novel IoT-MQTT based denial of service attack. Cluster Computing 26, 6 (Dec 2023), 3973–3984. https://doi.org/10.1007/s10586-022-03788-9

[13] S. Saxena, S. Jain, D. Arora and P. Sharma, "Implications of MQTT Connectivity Protocol for IoT based Device Automation using Home Assistant and OpenHAB," 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2019, pp. 475-480.