Building of Smart Plugs to Energy Efficiency in the Residential Load Management

William Rodrigues Silva - a52488

Thesis presented to the Superior School of Technology and Management (ESTiG) of Bragança as part of a Double Degree Program with the Federal Center of Technology of Minas Gerais (CEFET-MG) for obtaining the Master's Degree in Industrial Engineering.

Work oriented by:

Prof. Dr. José Luís Sousa de Magalhães Lima

Prof. Thadeu Vinicius de Brito

Prof. Dr. Luis Claudio Gambôa Lopes

Bragança

2022-2023

Building of Smart Plugs to Energy Efficiency in the Residential Load Management

William Rodrigues Silva - a52488

Thesis presented to the Superior School of Technology and Management (ESTiG) of Bragança as part of a Double Degree Program with the Federal Center of Technology of Minas Gerais (CEFET-MG) for obtaining the Master's Degree in Industrial Engineering

Work oriented by:

Prof. Dr. José Luís Sousa de Magalhães Lima

Prof. Thadeu Vinicius de Brito

Prof. Dr. Luis Claudio Gambôa Lopes

Bragança

2022-2023

Acknowledgment

I thank God to all blessing and health and strength that I was given to be here and CEFET-MG to make this exchange happen.

My fianceé and future wife, Karen Barbosa Cesar to support me since the beginning of engineering bachelor, especially this period spent far away from her and home, always available to talk in bad moments while I have been living in Portugal. She also has her contribution to this thesis.

My family, especially my brother and my parents that always support since the beginning of my life always encouraging and supporting me in so many things.

My co-supervisor Thadeu Vinicius de Brito from CeDRI that was always open to help me in so many questions that appears in my thesis and also calming me down with good conversations.

My supervisor José Luís Sousa de Magalhães Lima to receive me very well since I beginning my studies in IPB and even being busy, was very open to do meetings and help me in this work.

To my professor and co-supervisor from my university in Brazil, Luis Claudio Gamboa Lopes that, as same way, always be able to help me by online meetings and for his knowledge transmitted in some subjects in my course in Brazil.

To Paloma Greiciana de Souza Dias, my friend since the beginning of my bachelor's degree in Brazil, which together with me, helps to move this work ahead and help me with many questions.

Last, but not least, thank all of my friends that I met here in Portugal from CEFET-MG, other parts of Brazil, and the international friendships that I made here.

Abstract

Giving a background of the importance of Energy Efficiency and Renewable Sources, the present work aims to develop a low-cost smart plug module that could self-manage loads automatically in a residence. The system uses renewable energy by controlling current based on zero-crossing synchronism and switching using a semiconductor TRIAC. On this, the power could be customized according to the device requirements, being controlled remotely and being able to communicate with a supervisory using the Wi-Fi connection and MQTT publish/subscriber protocol to change data. Thus, it also discusses alternatives to use this surplus energy in the photovoltaic system context, giving highlighting resistive loads which can deal better with the high current and voltage variation due to switching. After test different microncontrollers and reach a final version, it has been made a basic supervisory, especially for tests that contemplate the sending and receiving data.

Keywords: Smart Plugs, Energy Efficiency, Renewable Sources, Internet of Things

Contents

List of Tables

List of Figures

Acronyms

ABESCO *Associação Brasileira das Empresas de Serviços de Conservação de Energia.*

ATS Automatic Transfer Switch.

DIAC Diode for Alternating Current.

EDP *Energias de Portugal.*

EEPROM Electrically-Erasable Programmable Read-Only Memory.

ESTiG *Escola Superior de Tecnologia e Gestão.*

GSM Global System for Mobile Communications 2G.

IoT Internet of Things.

IPB *Instituto Politécnico de Bragança.*

ISR Interrupt Service Routine.

LCAR *Laboratório de Controle, Automação e Robótica.*

MPPT Maximum Power Point Tracking.

MQTT MQ Telemetry Transport.

PCB Printed Circuit Board.

PUSC Production Unit for Self-Consumption.

SCR Silicon Controlled Rectifier.

SRAM Static Random Acess Memory.

TRIAC Triode for Alternating Current.

ZCT Zero-crossing Trigger.

Chapter 1

Introduction

1.1 Context

With the technological development that has been seen in the last years, turns increasingly necessary for electrical energy usage. However, the majority of sources used in the world for generation are non-renewables, which means it owns limitations in nature or/and are big carbonic gas emitters like mineral coal and natural gas. Still, billions of people do not have access to essential electrical energy services [1].

Furthermore, companies also are affected by energy waste which can be comprised by research made by the *Associação Brasileira das Empresas de Serviços de Conservação de Energia* (ABESCO) in 2015 which showed that approximately 10% of the energy generated in Brazil, a country predominantly supplied by hydroelectric, is wasted. This percentage represents R$12.64 billion which R$2.79 billion are from the commercials end-users, R$5.51 billion from residential end-users, and others from industries and less impactful end-users [2].

In Portugal's context, the number of domestic consumers is growing up in comparison with other consumer types that keep stable or falling. It can be observed analyzing Figure 1.1 [3].

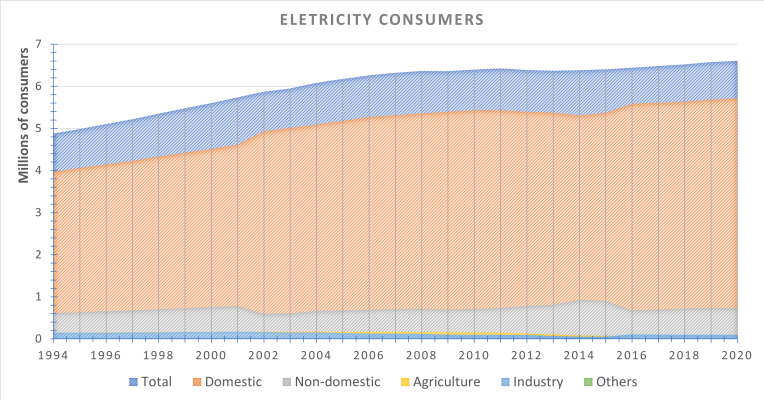


Figure 1.1: Total electricity in Portugal consumers and divided by type [3]

The electrical energy loss is also growing up, especially between 2011 and 2013 still keeping a high level compared to the years before [4]. Figure 1.2 brings up this information.

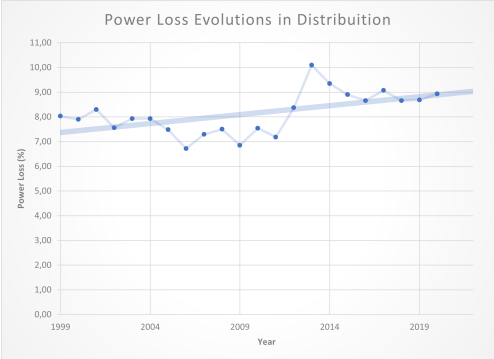


Figure 1.2: Power loss evolution in Portugal between 1999 and 2020 and linear preview [4]

Renewable sources have been proving even more that is the main way to deal with those issues and to reach goals to reduce environmental damages. It has been implemented in several countries in the world through the development of national policies, increasing the capacity of production and the share of energy. But, some countries like India that has a very strong demand due to the high population turning this energy supply intense as long as the country has more than 50% of its energy matrix supplied by coal and since now, also give alternatives to keep a sustainable consumption [5].

1.2 Motivation

To reduce consumption in the long run, some consumers had been choosing self-consumption as an alternative, especially through solar power, which is growing even more in Portugal [6], [7] as it is omnipresent, freely available, environment friendly, and has less operational and maintenance costs [8]. But, with the variations of self-generation and considering that not always the consumer will stay at home when it is at full production generating a surplus energy. On this, some alternatives can be chosen to redirect this extra energy that follows ahead as explored by [9].

The first way is to inject this energy into the grid, which on days with high radiation levels, does not change the energy that the consumer will utilize, and, in this case, this energy is wasted, being not a profitable alternative energetically for the consumer.

The second one is to limit the photovoltaic produce power, which consists of control of the inverters through a reduction of the DC voltage reference to, consequently, reduce the power. On this, even if the panels have the capacity to generate more power, this power would not be available to use turning it also non-profitable.

The third is about the storage of energy through the connections of batteries in the grid to use it later. This option is very interesting energetically but in a short period it is very costly, and, in some cases, it can not compensate especially for a low-power user. The charge and discharge of batteries generate losses that can compromise production. Follow ahead a table with some batteries and some of their characteristics.

A fourth alternative is to sell this surplus energy to the concessionaire which, in Portugal, the price depends on a contract celebrated between the trader and the person that desires to send it.

There are some conditions that sales can only be made in a Production Unit for Self-Consumption (PUSC) above 350 W and with a bidirectional counter that registers this surplus every 15 minutes [10]. Selling the surplus already increases the profitability of self-consumption, but through low values compared to paid buying the electricity of the concessionaire as shown in Table 1.1 keeps around 0.04 €/kWh. If it has a surplus

power generated between this break, there is no benefit to consumer and it returns to concessionaire.

Company	**Simple**		**Bi-Hourly**			**Tri-Hourly**			
	P	E	P	OE	Em	P	R	F	Em
EDP	0.2168	0,1461	0.2762	0.1832	0.0844	0.2550	0.2768	0.1402	0.0859
Galp	0.1419	0.1370	0.1419	0.1660	0.0820	-	-	-	-

Table 1.1: Standard Pricing for 3.4 kVA Residential Consumer. P = Power; E = Energy; Em = Empty; OE = Out of Empty; R = Rush Hour; F = Full Hour [11]

The last alternative, which is the main point of this work, is building an intelligent system that could shift this surplus energy produced during the day to a load that can be useful and/or return comfort to the resident giving some benefits like the effective cost, easy implementation and quick payback to the user. This strategy could be implemented in association with a storage system as mentioned or not, being the last alternative more difficult to absorb all energy as long as it demands more flexible loads to operate during the day, which, in most cases, is the period of full generation.

1.3 Objective

This could succeed through the development of a low-cost smart plug model able to control the power through a microcontroller interacting with a supervisory system and work on this mentioned energetic efficiency alternative. Therefore, this work abroad some electronics and programming techniques to develop this device that should be useful in the electrical energy management.

The Chapter 2 brings up the technologies present nowadays and ends with some works in smart plugs applications. Chapter 3 is present the schema of the entire project and the plug and also the description and purpose of each element that compose it. Chapter 4 is related the steps to make the prototype, about the tests in each model and a topic relative to the cost. Chapter 5 shows a review of the project, a discussion about it, and how it could be improved.

Chapter 2

Technologies Associated to the Proposed Work

Here are the interesting topics of this work, focusing on the implications discussed in the introduction, starting with a context about the photovoltaic system installation in Section 2.1, discussing ways to shift the surplus electrical energy generated by the system in Section 2.2, and, finally, introducing the topic of smart plugs through the existing devices present in Section 2.3 and some works that have been conducted on them in Section 2.4.

2.1 Photovoltaic System Context

In order to get better efficiency from a renewable source, it is always important to drain the maximum amount of power that a source is generating. For this, control devices are essential for tracking back the maximum power and choosing the best conditions.

To maximize the efficiency of self-consumption based on photovoltaic generation, it is common to implement a control module for the power flow from the generation to the batteries and the inverter known as Maximum Power Point Tracking (MPPT) which has the function of getting from the generation the maximum of voltage and current that depends on the amount of irradiation and temperature, and there are several techniques

for reaching this objective [12]. Its installation on the grid is demonstrated in Figure 2.1 ahead.

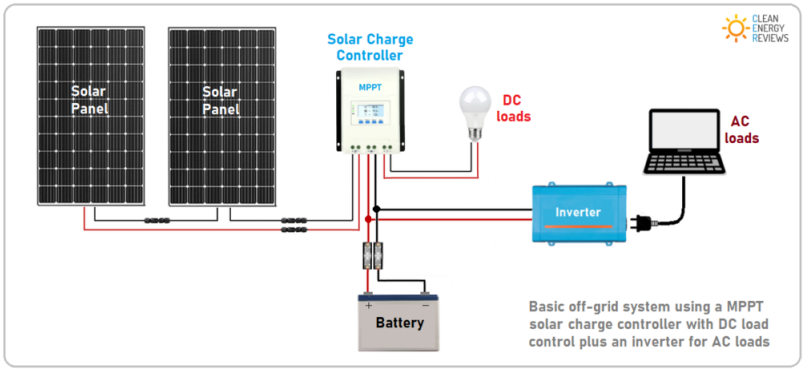


Figure 2.1: MPPT installation [13]

Broadly speaking, observing the graphs in Figure 2.2, this device tracks the peak voltage and current and also manages the maximum values of voltage and current combined to send to the batteries which consequently means the point of maximum power that the source is able to yield as can be observed in the Graph power versus current. To track this ideal point, several techniques can be applied as reported by [12].

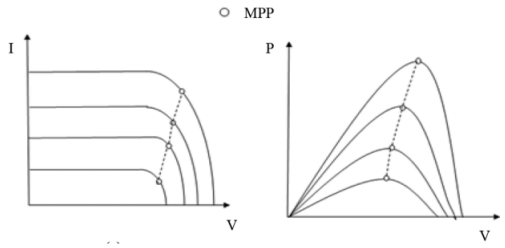


Figure 2.2: Maximum Power Point graphics [12]

Reliability is also crucial for distribution systems operation due to loss that could cause additional costs that directly implies on client satisfaction [14].

Aiming to manage the power source, and ensure the system reliability, there is a device capable to switch the supply to the concessionaire when the batteries automatically are low, known as Automatic Transfer Switch (ATS), monitoring both electrical power sources (grid and photovoltaic) and transfer of retransfer power to a load [15], [16].

This device, commonly incorporated in the solar system, has relays capable to do this mechanism, meanwhile, as it deals through the batteries level checkage, it is not able to deal with blackouts and needs to be adapted for this with the addition of some devices.

2.2 Loads to Shift the Surplus Energy

It is necessary to realize research about which loads will be used to consume the supply energy coming from the photovoltaic system. A common solution for this is to shift this energy for heating to a water heater that some companies already sell for this purpose. The water heater powered by the Portuguese company *Energias de Portugal* (EDP) shown in Figure 2.3 is already projected for this purpose [17].

Figure 2.3: Water Heater powered by EDP [17]

With this device, the company promises a 60% reduction in water heating costs [17], showing that as long as consumers save energy, they can also save for comfort.

A very recent research topic that could integrate these forms to spend energy, is reallocating this surplus or part of it to the production of green hydrogen through the electrolysis process. The element can be obtained through renewable sources like wind and solar, and many studies have examined this topic recently, such as in [18] and [19]. The research [18] relates that increased photovoltaic efficiency could reduce drastically the cost of hydrogen production. This application could be useful in the future to fuel vehicles and also to have an alternative to microgeneration.

It is also important to observe the limitations of the energy selling presented in the previous section and work on it, especially on the 15 minutes of break in which the alternate source could generate a meaningful power value.

2.3 Smart Plugs in Present

Smart devices, according to several kinds of literature, can be defined as "a context-aware electronic which have the capacity to perform the autonomous computing and connecting with other devices wire or remotely through data exchange" [20]. On this, it is almost inevitable to approach this subject without inserting it into the context of the Internet of Things (IoT) paradigm, which consists of multiple definitions as follows:

- "A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'Things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network" [21].
- "Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environmental, and user contexts" [22].

Analyzing this context, there are some examples of smart plugs present nowadays exemplified in Figure 2.4.

Figure 2.4: Mi Smart Plug Wi-Fi [23]

Those ones usually have a native smartphone application for controlling through ON/OFF and set timers as demonstrated in Figure 2.5.

Figure 2.5: Application used to control Mi Smart Plug Wi-Fi [23]

The present work aims to increase it with more functions turning it more smart and autonomous to fulfill the main objective of this work which consists of monitoring the surplus energy by sensors turning consumption more efficient.

In reference to power control, it has been researched in the industry, it can be found some dimmer modules are ready to communicate with microcontrollers. Also, exists a final device known as a smart dimmer that can communicate wirelessly with a smartphone application where is possible to control it remotely. It also contains a rotative button for manual control as demonstrated in a model presented in Figure 2.6.

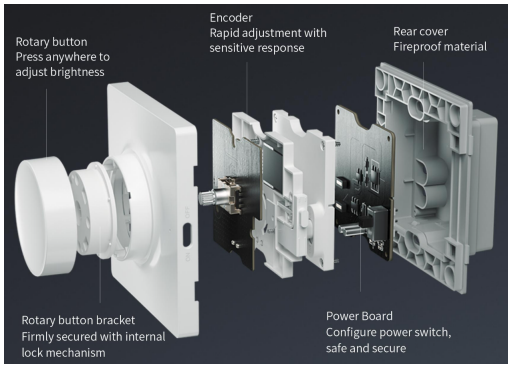


Figure 2.6: Yeelight Wireless Smart Dimmer [24]

This device shows the principal function of the proposal work which is power control.

Otherwise, it does not contemplate an integration with a supervisory, as mentioned before, that could be improved even more with machine learning, for example. On this, the load drive could be done based on several variables like climate conditions, user behavior, etc. The smartness of the system could be even more explored reaching the definition of smart devices and the IoT appliance.

2.4 Works in Smart Plug development

Works like [25], proposed the elaboration of a smart plug that communicates through Wi-Fi connection, uses of ESP32 microcontroller as an information center, a relay as an actuator, voltage and current sensor to get the measures and all the control and supervision are made by a web application.

This system can measure the instantaneous voltage, current and power, energy consumption, and electricity bills. As an action, it had a virtual button to turn on or off the plug via app as the current smart plugs do.

In [26], a similar proposal has been applied, which is used the wireless protocol ZigBee. This work develops smart plugs and a condition monitoring circuity, with the function of group all the information and communicate with the plugs. This circuit has been composed by sensors that aim to control the temperature of a refrigerator and a lamp brightness through two temperature sensors, one for the room and another for the refrigerator, in addition to a luminosity sensor. Therefore, through an algorithm flowchart defined by the code, the system can handle the loads actuation automatically.

A special mention to the work [27] presents a device that could replace the traditional ATS using the Arduino Mega 2560, Triode for Alternating Current (TRIAC), Bluetooth and Global System for Mobile Communications 2G (GSM) modules, and LEDs and LCD display as indicators. The results were promising showing that this model could, in a high speed, switch and transfer power during 2.58 milliseconds and concluding that the use of semiconductors in this project offers security and fast response facing the current models composed by mechanical contactors and relays.

Those related works presented in this section have a common aspect that sustains the development of this present prototype. The low-cost microcontrollers like ATMega328 and ESP can be easily incorporated into electrical energy management, providing efficiency and reducing the costs of the application. Beyond the attributes present in the mentioned works, this one focus on power control and the possibility to interact with a supervisory that could be integrated into machine learning to turn the system completely autonomous and smarter.

Chapter 3

Modeling and Tools

Based on this background, this work aims to develop a concept of a plug capable to utilize the maximum of the energy coming from renewable sources. It will shift the energy that would send back to the grid through the appliance of this output power in some resistive load made by the smart plug. Using the overproduction of energy coming from photovoltaic panels, it could be used completely all the energy generated and the necessary amount that load requires. In resume, the final objective is to build an intelligent electrical management system that works on energy efficiency.

It aims efficiency not only in the loads connected to the photovoltaic system but also in the loads connected to the grid like automatic dimmer lamps according to variations of the surroundings as innovation compared to the existing models.

The main idea of this work is to develop the smart plugs that control the out current intensity using TRIAC. The ideal current level will be calculated and sent from the microcontroller ESP that realizes the measurement to the plug ESP32 working on MQ Telemetry Transport (MQTT) protocol. A complete report of the system will be read by a monitoring center, which will manipulate in order to show on a screen to the user the status of the system, and customize the levels of power. The focus is the development of the hardware of the plug explaining step-by-step the building and the software that controls each plug. All the description of the system is shown in Figure 3.1 below.

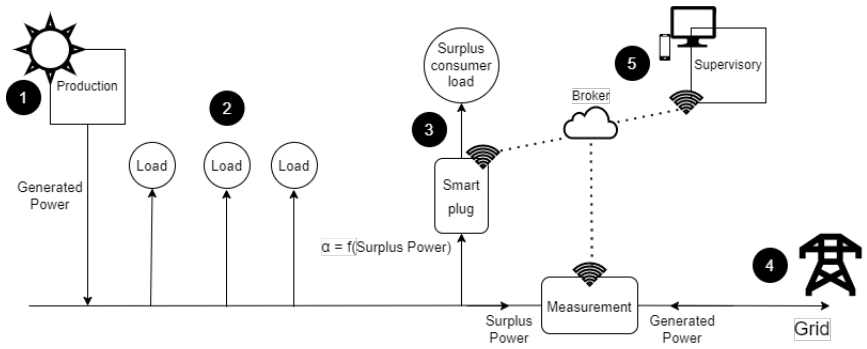


Figure 3.1: A System Overview

In ①, comprises the photovoltaic system generation and transmission since the capture of solar irradiation until the inverter output to AC current that can be described.

The ② demonstrate all the loads connected to conventional plugs in a residence which will be fed with the energy generated.

In ③ is the focus of this project that is the smart plug in discussion, which in it is connected the load to consume the surplus energy. It can be notice the MQTT protocol between ③ and ⑤, which the broker will act like a data register that could be accessed by the plug. All the MQTT working is detailed ahead on the Session 3.2.2. The Wi-Fi connection is fundamental to work on it, which is illustrated in each device that utilizes it. As can be observed in this part, the TRIAC trigger angle is function of the surplus power which this calculus is made in the code of the measurement ESP32.

In ④, is represented the electrical energy that is available from the concessionaire and its measurement. The power flow will define if it has surplus energy that is sent back to the grid, that will turn on the plug, or is working as expected with the concessionaire only feeding the residence. To get this power surplus measure, it has been used voltage and current sensors and another microcontroller ESP to communicate with the plug.

Last but not least, in the ⑤, a supervisory will receive all the consumption data, and then it can show all it on a screen and allows to the user customize all the loads of the residence fed by smart plugs and define the maximum power to them.

Focusing on the topic ④, there is a brief description of the necessary elements to the constructive part of the smart plug in Figure 3.2 in blocks with the application and a

description mention the components utilized.

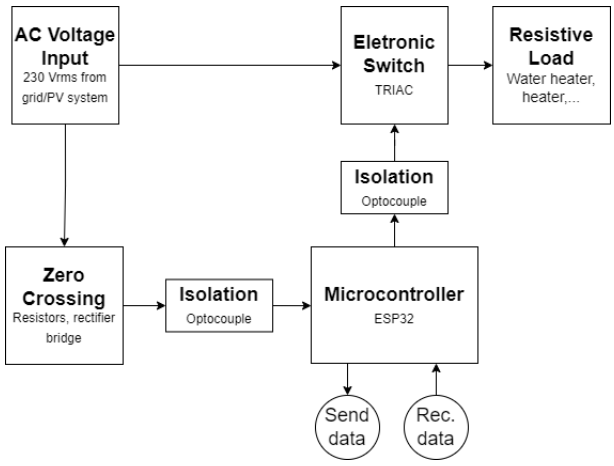


Figure 3.2: Plug Architecture

This section, as follows, brings up all those necessary materials for this application divided into two principal sessions: Hardware and Software. The hardware comprises all the physical and constructive parts like the microcontroller and the electronic components that compose the smart plug Printed Circuit Board (PCB). About the software, it explains how the microcontroller will work through the code developed, an explanation of the code, and some libraries and functions that are crucial to the plug working correctly.

3.1 Hardware

This present section will abroad all the theoric basis to develop the plug prototype through the description of each component used and some mention of other components that could be useful to comprise other ones.

3.1.1 ESP32

Espressif ESP32 is a low-cost microcontroller from Espressif Systems that integrates a Wi-Fi and Bluetooth connection compared with Arduino. This device can also be highlighted by your robustness able to work in industrial environments with an operational

temperature ranging from –40 to +125 degrees Celsius, ultra-low power consumption and a high-level integration [28]. Those features make the ESP32 an ideal device to work on IoT applications like this.

This microcontroller is accompanied by peripheral devices that turn possible to connect to a device like a computer as a source and a data transfer to load the sketch previously programmed. Thus, is provided by inputs and outputs analogic and digitals that make it possible to connect physically to circuits. Figure 3.3 is shown a fundamental microcontroller layout.

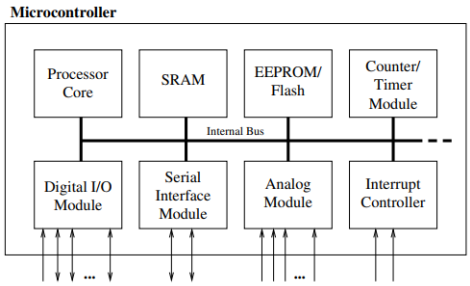


Figure 3.3: Microcontroller layout [29]

As the processor core performs all calculations, it is guided by the software loaded on it. The processor receives information from the inputs and sends it through the internal bus, and the outputs are sent through the same bus. Memory (Static Random Acess Memory (SRAM), Electrically-Erasable Programmable Read-Only Memory (EEPROM)/Flash) will also interact with the processor core [29].

To timestamp events, events counting and all that involves time counting, the Timer Module is very useful for this, especially for this work that requires constant synchronism with time.

Through those characteristics, this device becomes the most important in the project because it can develop, store, and run all the logical implementations and communicate with other devices over Wi-Fi, facing power applications as long as it has appropriate components joining it.

3.1.2 Diode

A diode is a semiconductor component with the principal function to allow current in only one way in an electrical circuit. This happens by its composition of two layers named N and P which in one way allows the displacement of the electrons and in the other way, acts like a barrier that resists this displacement. This component has several applications in electronics [30].

The diode is also the basis to comprise the other semiconductors used on the hardware detailed ahead because of its similar composition.

Rectifier Bridge

Englobe one of the principal application of the diodes being an essential component used in AC-DC converters composed of four diodes that manipulate an input AC current to generate a DC current in output. This component is especially used in full-wave rectifiers which mirror the negative part of the sinusoidal wave and turn it positive [30]. It has been used in the one shown in Figure 3.4. Your specifications are detailed in [31].

Figure 3.4: Rectifier Bridge B250C1000 [32]

It is fundamental to the zero-crossing application that will be discussed ahead in which the voltage changes each time the wave falls or rises.

3.1.3 DIAC

The Diode for Alternating Current (DIAC) is a semiconductor that acts like an AC diode allowing the current to flow in both ways. Its composition and operation can be explained as two diodes in antiparallel as shown in Figure 3.8 [33].

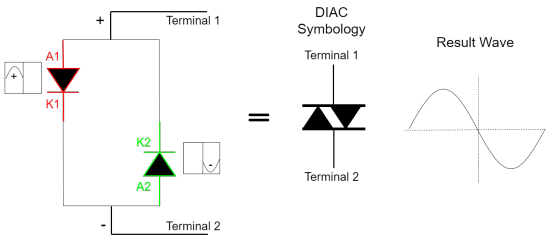


Figure 3.5: DIAC represented by diodes operation and symbology

This component could sound useless at first impression because, it could simply use no components to get the waveform shown. But, it helps to avoid noises in the circuit since it has inferior operation limits that are described by your reaction curve in Figure 3.6 known as breakdown voltage.

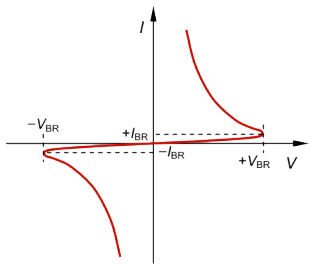


Figure 3.6: DIAC reaction curve [34]

As can be seen in the graph, when is applied a voltage between $-V_{BR}$ and $+V_{BR}$, there is no current flowing through the component operating like an open switch and, on other hand, when this breakdown voltage is over, the current is allowed to flow and the DIAC behave ideally like a short-circuit.

The comprehension of this component is important to see how works the component photo DIAC, used on the development of the plug which your proposal is described ahead in Section 3.1.5.

3.1.4 TRIAC

The TRIAC, represented in Figure 3.7, is a semiconductor with a composition similar to DIAC with the addition of a terminal gate which, since it has a voltage pulse in a semicycle interval flowing to this gate, the circuit current flow is allowed.

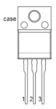


Figure 3.7: TRIAC BT 137X Series [35]

Its composition can be described as two Silicon Controlled Rectifier (SCR) thyristors in antiparallel as shown in Figure 3.8. While the SCR allows flowing current in only one way since having a pulse in the gate to start, with TRIAC is possible to do it for both semicycles [33].

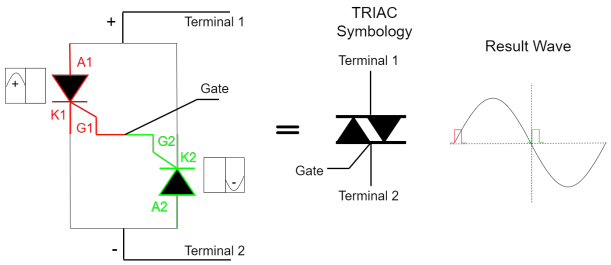


Figure 3.8: TRIAC represented by SCR thyristors operation and symbology

It is commonly applied in circuits of current control like lamp dimmers and heaters control. Controlling the current, consequently, it controls the power which is one of the manly purposes of the plug. The basic schema is shown below in Figure 3.9.

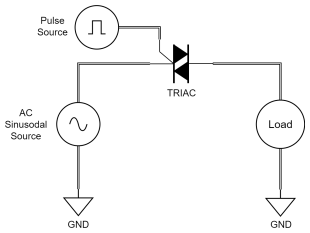
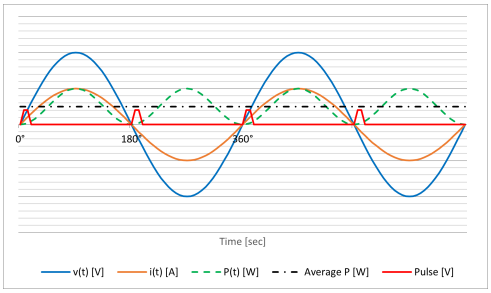
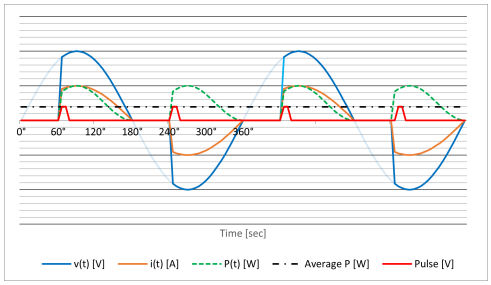


Figure 3.9: Basic circuit with TRIAC

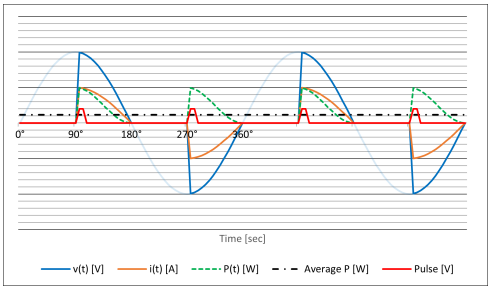
This level could be implemented through two methods: by phase angle and by pulse train. For this work, it has been used the first one which consists in inject current in a specific phase angle on a period cutting the current in the time corresponding to this angle, as shown the Figure 3.10 for different values of alpha.



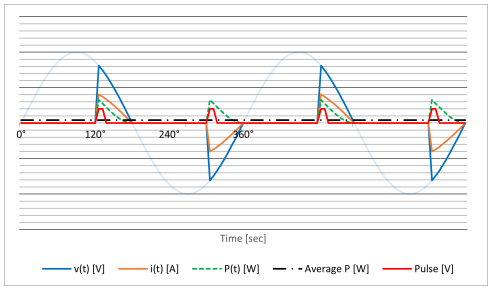
(a) With angle trigger of 0 degrees



(b) With angle trigger of 60 degrees



(c) With angle trigger of 90 degrees



(d) With angle trigger of 120 degrees

Figure 3.10: Resistive circuit behavior for different trigger angles injected on TRIAC

It can be observed in Figure 3.10 what happens if a TRIAC is connected to a resistive load as the Figure 3.9 that as long as the trigger angle is growing, the wave is "cut" more affecting the resulting power causing your reduction.

This application, as can be noticed by the synchronism of the voltage and current, is totally safe for resistive loads, but not the same for capacitive and inductive loads because, in case of the inductive ones, the blunt variation of current in the instant of the pulse, can elevate the voltage instantaneous to high values damaging the circuit.

3.1.5 Optocoupler

It is a component made of an infrared LED encapsulated with a photosensitive component in an integrated circuit. It is very useful to isolate circuits with a high difference in power

operation. One of the optocouplers used in the current project is shown in Figure 3.11.

Figure 3.11: Optocouple 4N25 [36]

It operates like the transistor in switch mode which, when a current flows through the infrared LED, a current is induced from the collector to the emitter in form of light through the light detected by the base of the transistor. This provides safe isolation between the command circuit and the load circuit avoiding damaging the microcontroller and the PCB that composes the plug in the zero-crossing application. Its symbology is shown in Figure 3.11.

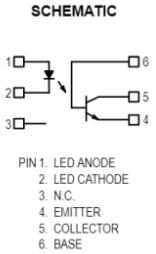


Figure 3.12: 4N25 Schematic [36].

About the zero-crossing application, between the diode terminals (1, 2) is connected the rectifier side and in the collector (5) and emitter (4) pins, is connected the microcontroller side. The working of this application is better described in the Session 4.

Other configuration of this component joins the DIAC and transistor functions. Since as flow current is through the DIAC part, the transistor is polarized and allows current flow through its terminals. Its appearance is the same as the optocoupler, encapsulated on an integrated circuit and its symbology is represented in Figure 3.13. On this, it acts as a TRIAC switched by light.

However, TRIAC phase angle control method can also result in increased harmonic distortion in the power supply system, which can cause interference with other devices and systems [33]. For this, it is necessary the addition of an anti-harmonic filter.

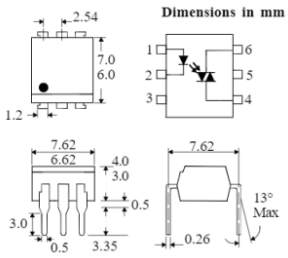


Figure 3.13: MOC3020 Schematic and dimensions [37]

3.2 Software

This section discusses a brief background to implement the programming code of the plug and explains how it communicates with other devices.

3.2.1 Arduino IDE

To develop, compile and execute the program that contains the logic of the plug, an environment is necessary to do. Both Arduino and ESP32 are compatible with the Arduino IDE, a development software destined to run and compile programming codes, at first, to the microcontroller Arduino that will be expanded for all ESP models. Figure 3.14 shows the interface of the Arduino IDE 2.0 bringing an improvement structured in comparison with the old version.

Figure 3.14: Arduino IDE interface [38]

Ahead, there is a brief overview of this *software* according with [38]:

- **Verify / Upload:** compile and upload the code to Arduino Board.
- **Select Board & Port:** detect Arduino boards automatically show up here, along with the port number.
- **Sketchbook:** where are the sketches locally stored on the computer. Additionally, it is possible to sync with the Arduino Cloud, and also obtain sketches from the online environment.
- **Boards Manager:** browse through Arduino and third-party packages that can be installed.
- **Library Manager:** browse through thousands of Arduino libraries, made by Arduino and its community.
- **Debugger:** test and debug programs in real-time.
- **Search:** search for keywords in the code.
- **Open Serial Monitor:** open the Serial Monitor tool, as a new tab in the console.

As can be noticed, it brings a friendly interface with the port and device selection. With two native functions to run the script uploaded by the user: `void loop()`, which repeatedly runs the code inside the curly brackets, and `void setup()` to run once aiming to initialize variables and declare inputs and outputs.

It is also possible to choose a variety of boards from both Arduino and ESP ones and select which port is connected to deal with the different microcontrollers that were used on the tests of this work.

3.2.2 MQTT Protocol

Developed by Andy Stanford-Clark (from IBM) and Arlen Nipper (from Arcom, now Cirrus Link) in 1999, MQTT is a client-server publish/subscriber messaging transport

protocol. This protocol is widely used in the IoT contexts due its simple implementation, lightweight, bandwidth efficiency, and low battery consumption.

The publish/subscriber architecture works through the decoupling between the client that sends the message (publisher) and the client that receives the message (subscriber) making them not have direct with each other. The connection between them is handled by a third element called broker which will receive the publishers' messages and distribute them to the correct subscribers [39]. Figure 3.15 is shown this architecture.

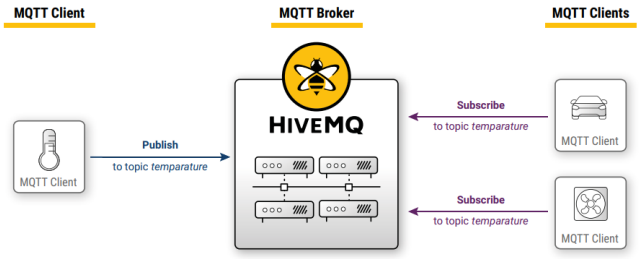


Figure 3.15: How pub/sub architecture works [39]

This protocol was fundamental to communicating the plug with the measurement of power surplus to receive the power percentage from the supervisory.

In this work context, the plug will be one client which will work as a publisher and also a subscriber meaning that it will access and receive data from the broker, the intermediary that will filter the messages. The other client is the supervisory that also will be a publisher and a subscriber interacting with the plug through the broker.

Those data consist of the power percentage from the total available set and send from the supervisory to the plug and the real percentage of power that the plug will send back to the supervisory.

Chapter 4

Development and Implementation

After a short background in the previous chapter, this one joins this knowledge to develop the controlled smart plug explaining with more details, if necessary, the function of each component in the circuit present in Section 4.1. With the circuit schema, Section 4.2 brings the making of the project. All the logic behind the programming is present in Section 4.3. At the end, Section 4.4 brings up the effective cost of implementation and a comparison with existent products.

4.1 Developing of the Electronic Circuit

The electronic circuit schema of the smart plug has been developed with the most of components mentioned in Section 3 through an analysis of each step of the necessary manipulation of the electrical energy based on the model used in the tests stage. In this chapter, it is presented step by step of the electronic circuit.

To start the development is necessary to get the AC voltage coming from the electrical grid which, in Portugal, is 230V with a frequency of 50 Hertz, and an incandescent lamp as a load to practical effects because its resistive characteristic and facility to observe the current intensity. Figure 4.1 shows the circuit made in Multisim of the source and lamp to be controlled.

As the purpose of this work is necessary to insert a component that is capable of

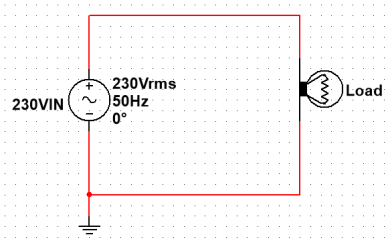


Figure 4.1: Circuit to be controlled.

controlling the power of the load and, admitting that the voltage in the load terminals is always the same and knowing that the power is the product of voltage and current, the focus becomes to control the current. As seen in the introduction of the components shown in Chapter 3, the TRIAC is a component that has this function which through a sync circuit in the gate, allows or not the current flow in the lamp in both ways. A BT137X was chosen as one appropriate model to work in the conditions of voltage, current and power required as can be checked in its datasheet. Figure 4.2 shows how this component as an addition to the previous circuit.

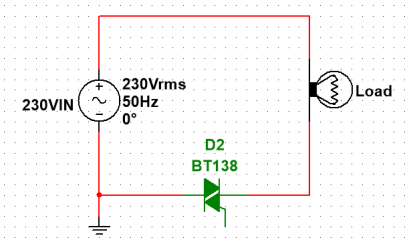


Figure 4.2: Circuit with the switch element TRIAC

After that, as seen, a circuit to control the current injected in the gate of TRIAC has to be developed. Firstly, it should have a component that acts like a switch triggering the current in each semicycle of the alternating voltage coming from the electrical network that results in the reduction of the root mean square value of the current and, consequently, of the load power. This switchable component could be a transistor or, for better isolation between the control circuit and the power circuit, an optocoupler. This optocoupler is made of a transistor junction which has been used an optocoupler DIAC MOC3020M in the output. DIACs are commonly used in together with TRIAC to prevent low noise

voltages under the voltage rupture as mentioned in Section 3.1.5. To avoid damage to the component, it was added resistors to limit the current that flows through the component according to the specifications in its datasheet shown in Figure 4.3.

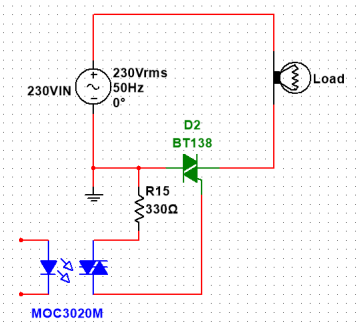


Figure 4.3: Circuit with a switchable component in the gate: the optocoupler by DIAC

To do the control is necessary a source which must be synchronized with the grid voltage to switch the gate of the TRIAC. With a microcontroller, is possible to manage the circuit reading values of other components and act in the right time to switch the gate.

In Figure 4.4 is shown the increment of this device that could be an Arduino or an ESP depending on the goals that will be more detailed ahead. Choosing the ESP32 and connecting the digital pin output 12 with 0 or 3,3 volts the input of the optocoupler depending on the time instant. Plus, a source of 5V has been applied to the circuit to supply the microcontroller, and, with this voltage, is possible to scale the resistor according to the limits of MOC3020M.

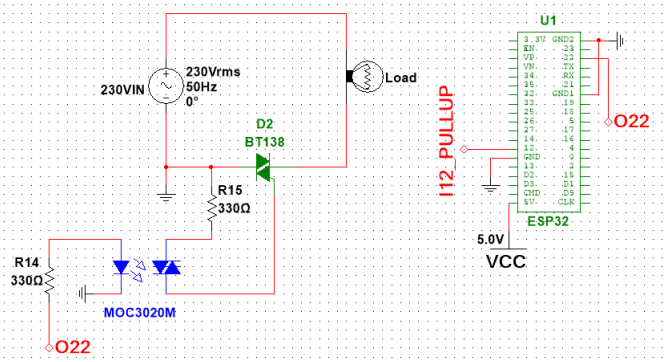


Figure 4.4: Circuit controlled by microcontroller ESP

After that, is necessary to something able to detect each semicicle to get the reference point and start to count the time to turn on and off the gate of TRIAC at a specific trigger angle defined by the user.

A common technique for this purpose is known as **Zero-crossing Trigger (ZCT)** which is intended to prove when a line voltage crosses the zero voltage by using a comparison circuit or gate control technology [40]. In this case, it was used a comparison circuit with a full wave rectifier and an optocoupler. The rectifier will double the frequency of the grid allowing each instant in which the voltage crosses zero volts to be detectable. The optocoupler will isolate the power and control circuit and sense each moment that wave crosses zero. This feature allows to the microcontroller catch the exact moment of it.

As described before, the optocoupler is an ideal component to do the isolation between the power and control circuit. On this, the zero value can be read by the pulses generated in the output of the optocoupler 4N25 that is connected to a digital input of ESP32 (pin 22 chosen). Before supply the 4N25, it was inserted resistors with high values to reduce the alternate voltage that is applied to this component avoiding damage.

This circuit composes a basic dimmer similar to found in some electronic literatures [30] but with an adaptation to be triggered by a microcontroller instead manually through a potentiometer as commonly shown.

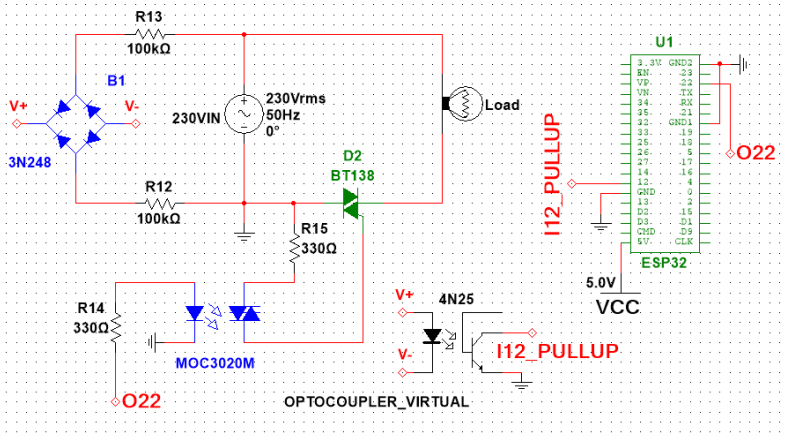


Figure 4.5: Circuit with optocouple to detect the change of semicicle

For connection, it has been placed the output of the rectifier bridge in the input of the 4N25, get its output 5V, and read the output using a pin of the microcontroller which has an internal pull-up resistor. This internal pull-up will help to avoid ambiguity in the reading of values which, when the phototransistor has no current flowing, it is connected directly to an internal voltage with resistance (high level). On other hand, if there is current, it flows connecting the input pin to detect zero volts by the GND connection (low level).

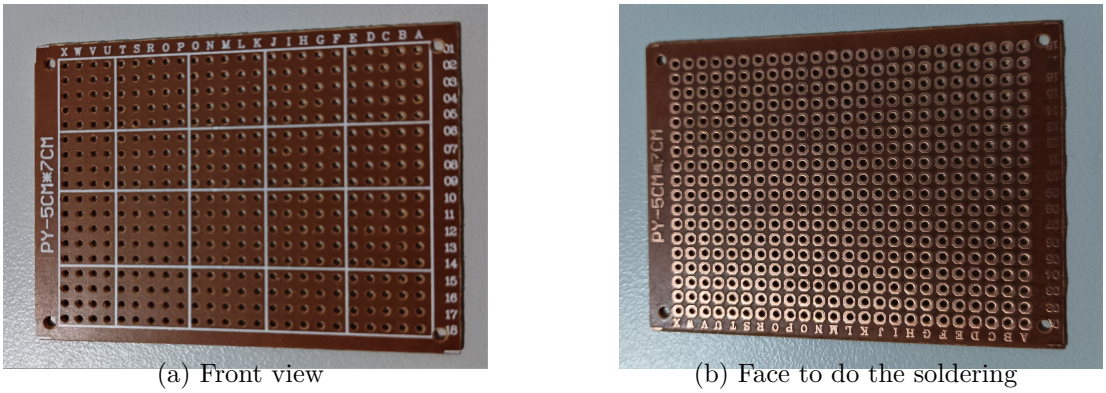
4.2 Prototype Making

As a result of the step-by-step description in the previous section, it is possible to define a material list of the necessary components to develop the plug presented in Table 4.1.

id	**Amount**	**Component**
1	1	ESP32 Wroom 32E
2	2	Pitch Single Row Female Pin Socket 19 ways
3	1	Universal Board 5x7 cm
4	2	KRE Conector
5		Jumpers
6	2	Resistor $10k\Omega$ $3W$
7	2	Resistor $330k\Omega$ $0,25W$
8	1	Rectifier Bridge
9	1	TRIAC BT137-600D
10	1	Optocouple 4N25
11	1	Optocouple MOC3020
12	2	Integrated Circuit Socket 6 ways
13	1	Plug Adapter
14	4	Wire 25 cm
15	1	Plastic Box

Table 4.1: Material list

For the final version, it has been building a compact prototype to fit in real plugs and also reduce the number of components of the test board. For this, it has been used a universal PCB model of Figure 4.6 with the dimension 5cm x 7cm for soldering the microcontroller ESP and the components.



(a) Front view (b) Face to do the soldering

Figure 4.6: Universal Printed Circuit Board to fix the plug components

To fix the components on the board of Figure 4.6, the JBC desoldering module and soldering station, which are installed on stand 8 of *Laboratório de Controle, Automação e Robótica* (LCAR) at *Instituto Politécnico de Bragança* (IPB), have been used, as shown in the images of the soldering process in Figure 4.7.

(a) Desoldering module JBC (b) Components soldered in the Universal Board

Figure 4.7: Soldering process

After the welding process, the continuity with the help of a multimeter in the appropriate function for this purpose has been tested, as shown in Figure 4.8 to attest that it is all fine do not present any short-circuit or bad connection.

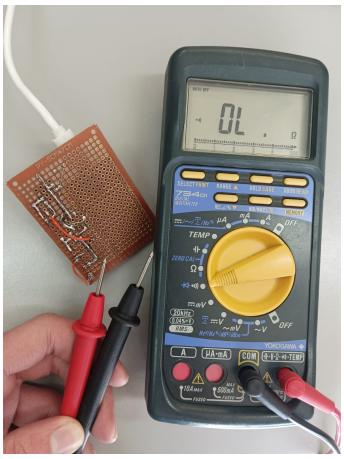


Figure 4.8: Continuity test using a multimeter

Then, it has been stripped and placed the wires to connect the adapter and supply the circuit with energy from conventional plugs and wires to connect the load.

After all the connections were prepared, it has been placed the PCB, aiming for protection and aesthetics, inside a plastic case previously pierced for the wires as shown in Figure 4.9.

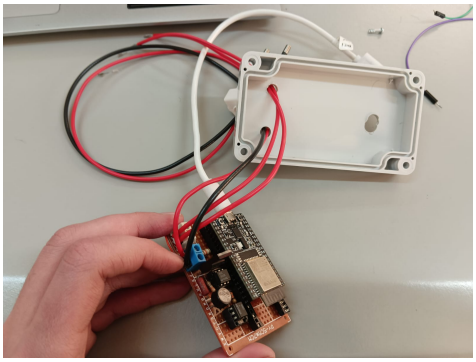


Figure 4.9: Insertion of a case

After some adjustments of soldering and wire changing due to bad contact, it reaches the final prototype shown in Figure 4.10 which shows the smart plug working in an incandescent lamp.

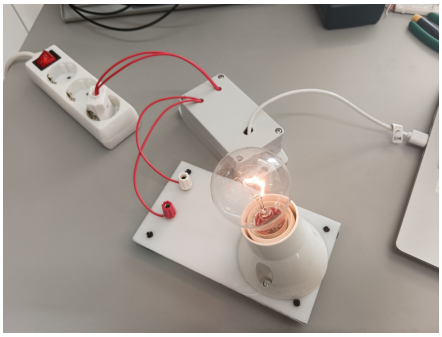


Figure 4.10: Final prototype in work

4.3 Developing and Testing of the Code

This section presents the logic that implies in the code development like formulas and important circuit behaviors. It is currently referenced by Appendix A where is the programming codes tested in the application. For testing, a green PCB model with the

dimmer application presented before has been used. Each model has also been described, the test validations, and a discussion about it.

At first, the tests have been started with Arduino Uno for two main reasons: the test board of TRIAC model available has the fitting for Arduino, and for first tests offline with the microcontroller does not need a Wi-Fi connection.

After that, it has been providing an ESP32 model which has the same Arduino format which was tested with the same code applied before and increasing the possibility of connection to the project and tests to receive data.

The final step is to provide a new PCB model aiming to reduce the number of components turning the project more compact. For this, it also has been reduced the microcontroller size migrating to an ESP32 model Wroom 32E which will be shown ahead.

4.3.1 ATMega Arduino Version

As previously mentioned, it has been started the development of code uploading and running on Arduino Uno R3 as the first step for this project.

To start, it was first declared the pin variables previously defined by the board construction in Figure 4.5: pin 3 for detecting the zero-crossing as a `INPUT_PULLUP` avoiding ambiguously and the pin 8 to activate the gate of TRIAC allowing the current flow as an `OUTPUT`. As related in Section 3, the pin is declared in the setup function.

To read the 4N25 output for each time that the voltage crosses to zero value, the gate starts counting the right time to activate the TRIAC according to the percentage allowed, but, this function can be ignored depending on which part of the code the microprocessor is reading and there is a time processing problem.

For this, an Interrupt Service Routine (ISR) can be used to do some actions which immediately stop the current execution running in a loop and execute the function defined inside the `attachInterrupt` scope helping to solve synchronism problems [41]. It has been fundamental to get precision of the activation of the TRIAC gate that is impossible to reach without ISR.

The function `atachInterrupt` has three parameters that must be declared, respectively: the input which will stop the current execution, the function that turns out to be executed, and the detection method. It exists three basic modes of detection: rising, falling, and change. The rising one will stop the code when detects the signal input declared in the first parameter growing up, the falling when detect it is decreasing and the change mode calls the function always when the input switch from high to low and vice versa. Figure 4.11 illustrates each mode's switching moment from the sequential read of the code to run the interrupt function.

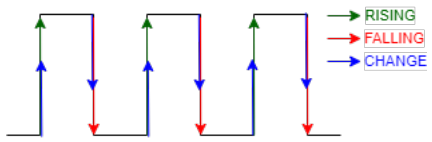


Figure 4.11: Detection moment of each mode

After applying the ISR in the code, it has been calculated the time necessary to wait for the activation of the gate. The triggering frequency is double the grid frequency as seen. On this, is obtained the percentage of this value to determine the time delay matching to the power percentage defined as can be calculated by Equation 4.1. The level is inversely proportional to the time: how much more is the level, faster is the triggering time, and more short is the time delay to trig the gate. On this, it can be stored in a variable or directly inserted in the delay function that is going to be detailed moreover.

$$t_{delay} = \frac{T}{2} \times \frac{100 - level}{100} \tag{4.1}$$

In practical terms, it turns necessary to set an adjustment in relation to the zero crossing sensitivity because the signal generated in the output is not synchronized with the exact moment that the voltage cross through zero as can be observed in a test oscilloscope ahead. This time has to be considered in the calculus and obtained with the help of the cursor as shown in Figure 4.12.

As can be observed, this measurement is about the time that the zero-crossing detection by the 4N25 (in yellow) starts to rise and when the voltage signal (in blue) crosses

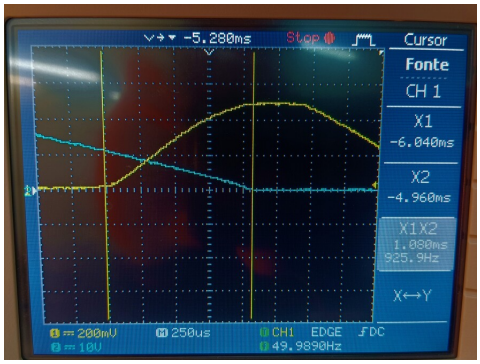


Figure 4.12: Measurement of the pulse width since the zero-crossing detection until the real zero-crossing.

the zero that results in 1.080 microseconds applied on a variable in line 6 of the code of the Appendix A.1. In this example, an alpha angle has been settled that explains the cause of the wave stabilizing on zero.

On this, to have the precision for the zero value, working in the rising edge, is necessary the addition of 1080 microseconds called "*rising delay*" in the code to calculate the time correctly. Then, the delay time described in Equation 4.1 turns into Equation 4.2 with the increase of this time delay before the percentage calculus.

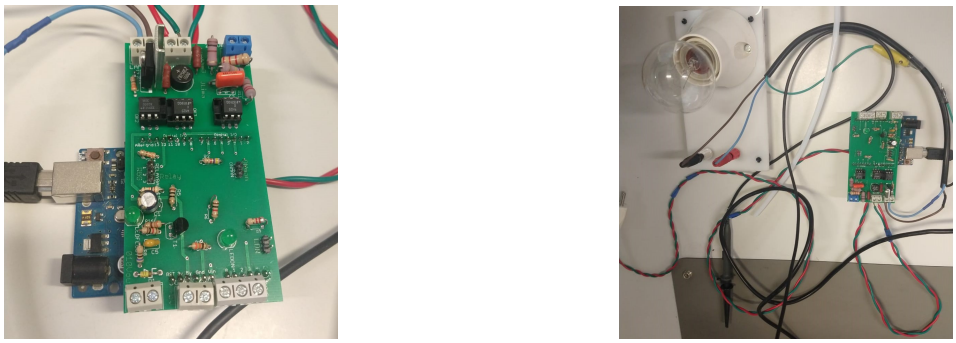
$$t_{delay} = \left(\frac{T}{2} + risingDelay\right) \times \frac{100 - level}{100} \tag{4.2}$$

Inside the interruption function, it has been calculated the delay time to activate the TRIAC gate considering Equation 4.1 plus the adjustment mentioned before. That is because the time will start to count after this time between the rising of the detection wave read by the microcontroller and the real moment that the wave crosses the zero.

Lastly, the MQTT protocol was applied to access the necessary level of current to send to the TRIAC through the triggering angle. This value was previously calculated in the input of the system through a sensor to get the power values and converted in an ESP32 to the percentage of current allowed in the plug ESP32.

It was made the tests with the developed code with Arduino with an existent green

PCB model of Figure 4.13 containing the dimmer application previously mentioned. Figure 4.13 ahead also shows the prototype and an incandescent lamp as a resistive test load. The reason to use Arduino is that the green test board in question showed in Figure 4.13 had the appropriate fitting made for it and, at the first moment, only test the control part without connection.



(a) Smart Plug test PCB coupled with Arduino (b) Incandescent lamp as a test load connected

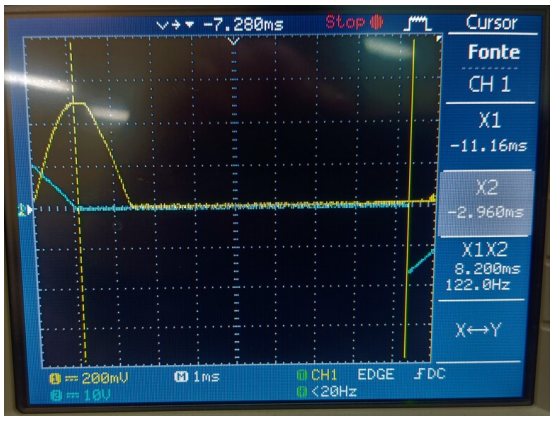
Figure 4.13: SmartPlug model with Arduino

The tests on Arduino Uno R3 were executed running the code present in Appendix A by changing the value of the variable level in line 12 for different percentage values.

Firstly, it is necessary to comprise the characteristic of the zero-crossing by the analysis previously explained in Section 4 by the measurement of the moment of rising detection and the real zero-crossing demonstrated in Figure 4.12

After that, the evaluation tests start with applying an angle alpha that matches 20% of the maximum power as exactly described in Appendix A.1. The results are shown in Figure 4.14 ahead.

(a) Load waveform (blue) and zero-crossing (yellow) for 20%



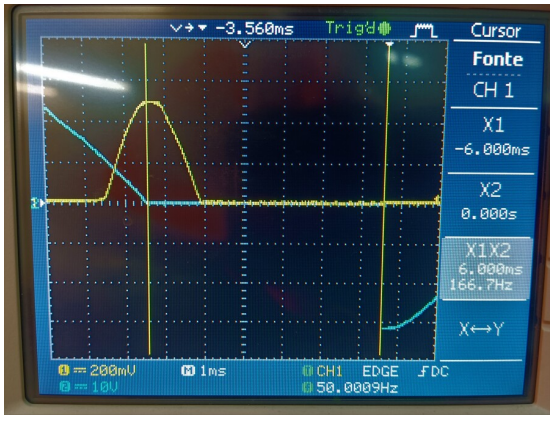
(b) Time delay to trig the gate detailed for 20%

Figure 4.14: Tests with 20% of the power

As observed in Figure 4.14(b), the delay time since the real zero-crossing is approximated of the expected result of 8 ms which corresponds to 80% of the time turning off and turning on in the 20% left of the semicicle.

Proceeding with the angle tests, an alpha angle of 40 degrees was applied through the change of value for the variable *level* in line 12 of the current programming code to 40. Figure 4.15 ahead shows the results.

(a) Load waveform (blue) and zero-crossing (yellow) for 40%



(b) Time delay to trig the gate detailed for 40%

Figure 4.15: Tests with 40% of the power

To this value of alpha, it shows a more reliable value matching with the expected one. The current flow in 40% of the semicicle can be noticed by the 6.000 milliseconds of waiting and the 4.000 milliseconds left to activate.

After that, testing with 60% of the available power results in the waveforms presented in Figure 4.16 ahead.

(a) Load waveform (blue) and zero-crossing (yellow) for 60%

(b) Time delay to trig the gate detailed for 60%

Figure 4.16: Tests with 60% of the power

Those tests show that the *delay* function works correctly in Arduino showing consistent values with a correct calibration to be found.

4.3.2 Wemos D1 Mini ESP32 Version

After that, as treated in the beginning of this chapter, it has been substituted in place of Arduino Uno, this embedded model of ESP32 with the same fitting to migrate the application to allow a Wi-Fi connection to receive the values wirelessly. Figure 4.17 shows the model utilized.

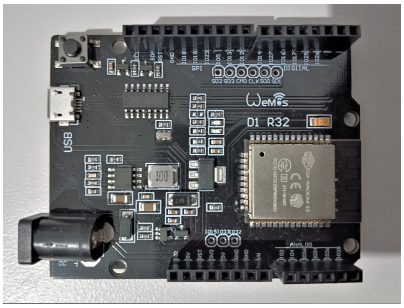


Figure 4.17: Wemos D1 Mini ESP32

In the tests ahead, it is going to be described that using *delay* function is not a big deal. Then, in the Wemos D1 Mini ESP32 microcontroller tests has been used timers aiming more precision in time counting. This library has several functions to configure and deal with the timers present in the ESP32 construction. It has been used four of them: `timerBegin()`, `timerAttachInterrupt()`, `timerAlarmWrite()`, and `timerAlarmEnable()` [42].

It runs the ISR function `ISR_zeroCrossing()` when it detects the interruption by the zero-crossing, which determines the time parameter for turning the trigger from high to low. To ensure that this line will be executed continuously, any interruption has been disabled for this declaration.

After that, execute the function dedicated to setting the timer parameters to determine the waiting time until the gate of TRIAC receives the pulse from the output microcontroller ESP32 and also determine which pulse is allowed. The function `setTimer()` waiting time is calculated constantly in the `void loop()` using the `map()` function replacing the Equation 4.1.

Then, after the timer finishes the counting, the microcontroller trig the gate through the function `ISR_turnPinHigh()` and basically at the same instant, sets the timer to turn to low the gate by the function `setTimer()` again. As followed to done in `ISR_zeroCrossing()`, it has been stopped the interruption actions.

So, it starts the function `setTimer()` to count a short-time correspondent at a pulse. To finish, this triggering cycle, the trigger pin is set to low through the function `ISR_turnPinHigh()`. Then, it continues to run the loop function always calculating the time delay based on

the level defined.

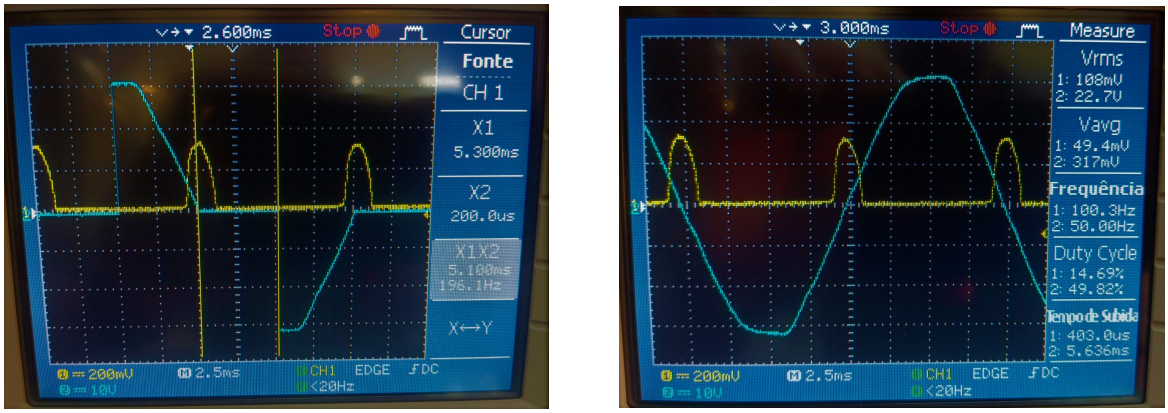
To send power and consumption parameters to the plug, it has also been stored the time instant of the zero crossing detection and the start of the gate observed in lines 55 and 74 of the code and used in a specific function to do the calculus.

After the tests, a new code has been written to connect the plug to the internet and prepare it to send and receive data via Wi-Fi and communicate with the MQTT broker. In order to achieve this, it has been imported two libraries: `WiFi.h` and `PubSubClient.h` previously declared at the beginning of the code.

On this, it has been elaborated functions to keep the connections and receive the data from the supervisory (publisher) to import to the plug (subscriber) so that it could be read and processed by the plug. Doing this, it is established the remote control via Wi-Fi connection through MQTT protocol. The next step is the development of the supervisory to send data to the plug constantly and be able to also receive data.

Applying the same code, modifying the pinout and the speed rate as can be observed in the code of A.2, the result was not promising being necessary to change the code for this microcontroller.

Then, compiling, uploading, and running the code present in A.2 elaborated using timers to the Wemos D1 Mini ESP32, has satisfactory results showing the expected wave for different power levels as can be observed ahead in Figure 4.18.



(a) Triggering of 60% (b) Triggering of 100%

Figure 4.18: Tests with ESP32 Wroom 32E after applying Timers

It has been necessary to make some adjustments to match the level to the exact time of activation through the calibration of the *minValue* and *maxValue* analyzing the zero crossing behavior like done before.

The final results were practically satisfactory only having a calibration issue to be solved by the adjustment of the variables *minValue* and *maxValue*.

4.3.3 ESP32 Wroom 32E (Final Version)

To final version, aim to fit into a smaller board turning the prototype more compact, as explained in Section 4.2, it has been used the model Espressif Wroom ESP32-E of Figure 4.19.

Figure 4.19: ESP32 Wroom 32E

There are basically no changes in the code in comparison with the previous version

shown that still utilizes the test board. So, the final code can be resumed to the flowchart presented in Figure 4.20.

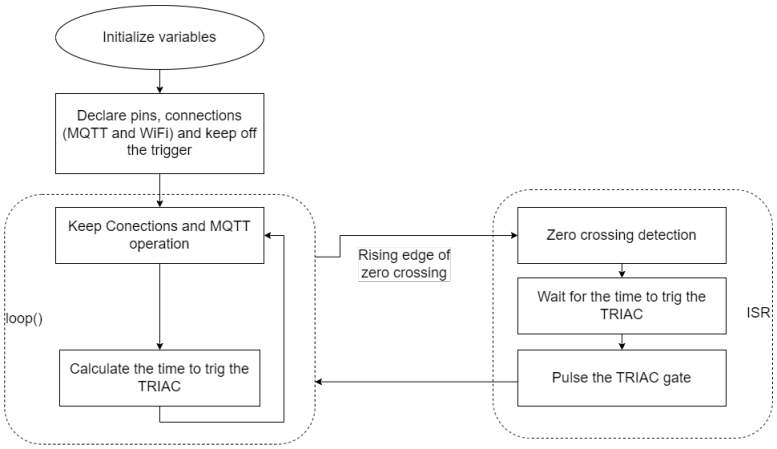
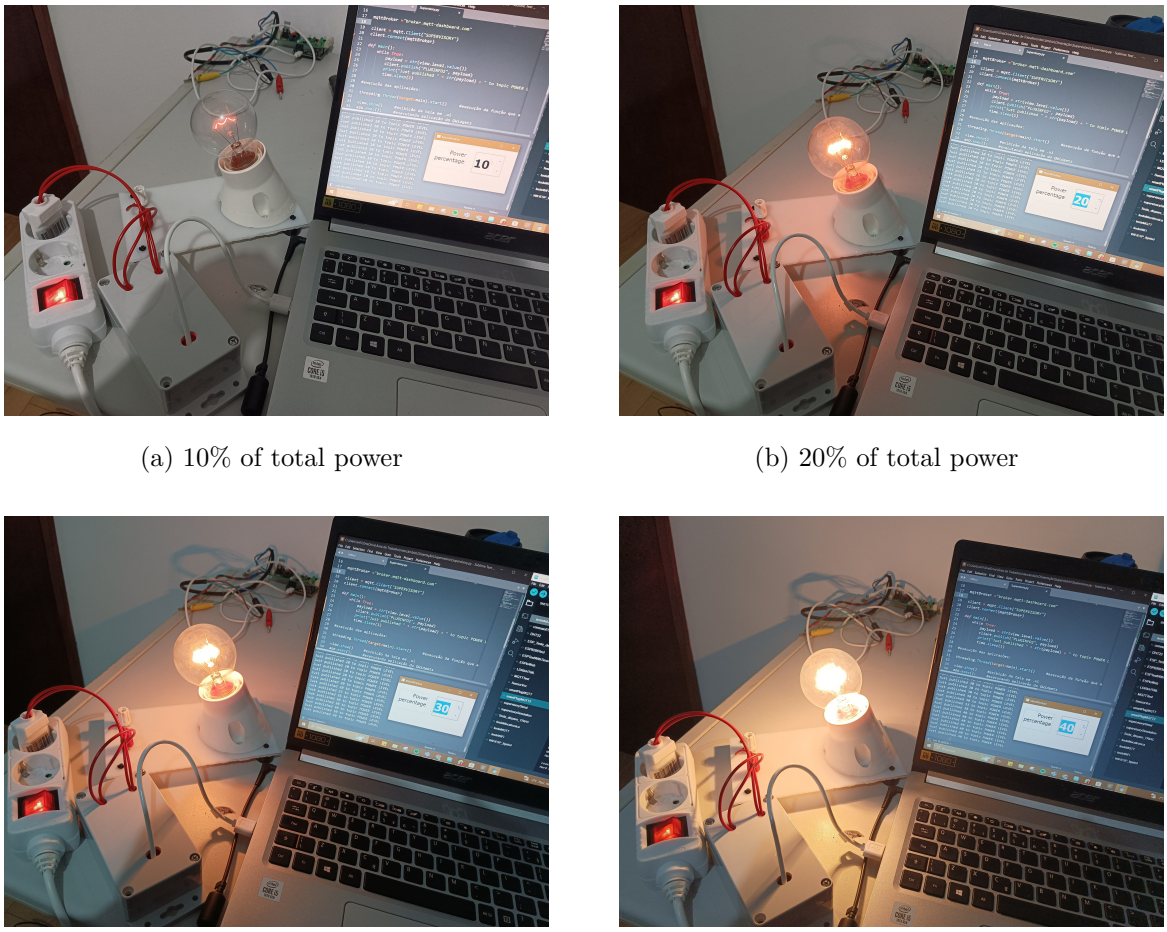


Figure 4.20: Code Flowchart

Compiling and uploading the same code for this ESP32 model, it was observed that the lamp has been blinking constantly is not desirable.

Then, analyzing its datasheet, it was noticed that pins 22 and 23 could have better results using their internal pull-ups and after changing the pin, it shows similar results as the Wemos turning the final prototype ready to be developed as related in Section 4.2.

To test the communication between a supervisory and the plug, it has been developed a basic interface based on Python programming language available in Appendix A.6. The percentage values are sent by the edit text field in QTDesign where is draw the interface and the Visual Studio Code to debug and execute the code. Figure 4.21 ahead is detailed in the interface showing different power levels indicated by lamp brightness.



(a) 10% of total power

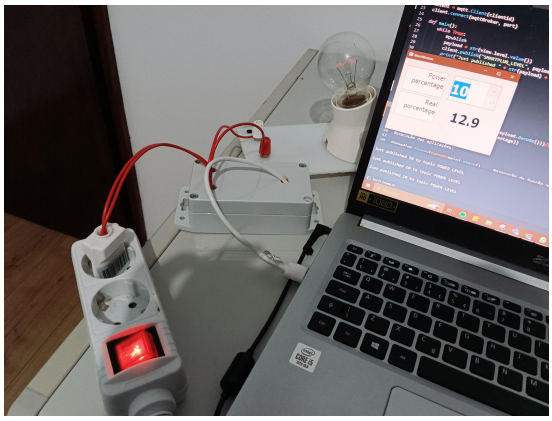
(b) 20% of total power

(c) 30% of total power

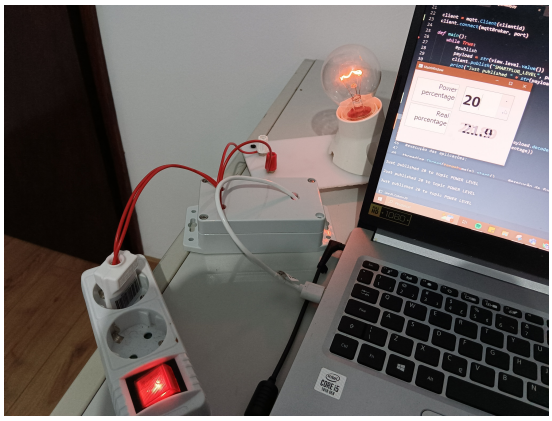
(d) 40% of total power

Figure 4.21: Tests with supervisory in Python answering commands

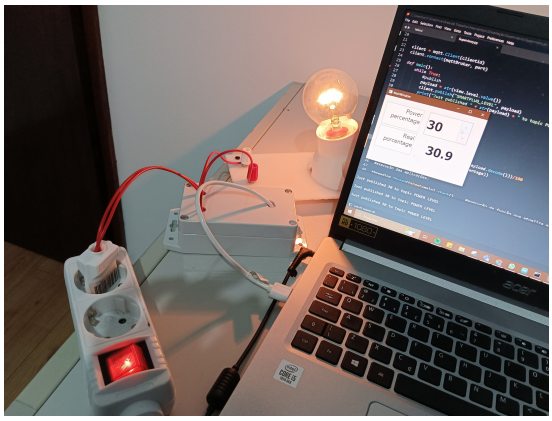
After that, it was increased the possibility of the plug sending information to the supervisory by the addition of publisher functions that are detailed in the final code in Appendix A.5. On this, it was used the variable `timeOff` to collect the real-time that the plug waits to activate the trigger and transform this variable to percentage, it is possible to compare the value send to the real one and based on this, it was changed the `minValue` and `maxValue` parameters in lines 10 and 11 of the code of Appendix A.5 until reach approximated values how shows the Figure 4.22 (600 and 9600, respectively). The final version of the supervisory interface developed in QTDesign generates an HTML code that will be treated by the python libraries described in Appendix A.7.



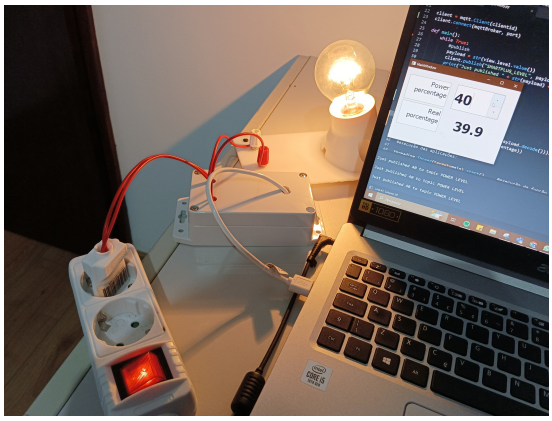
(a) 10% of total power



(b) 20% of total power



(c) 30% of total power



(d) 40% of total power

Figure 4.22: Tests with supervisory in Python receiving the real power percentage

By those tests, it shows satisfactory results that ensure the plug is able to receive values from a supervisory being possible to be controlled and supervised remotely.

4.4 Pricing Comparison

For the purpose of attesting the low-cost feature, the prices of the components were researched and compared with those of some smart plugs and dimmers. Table 4.2 brings this information based on PTRobotics [43].

id	**Amount**	**Component**	Price (€)
1	1	ESP32 Wroom 32E	16.61
2	2	Pitch Single Row Female Pin Socket 19 ways	2.52
3	1	Universal Board 5x7 cm	1.17
4	2	KRE Conector	0.62
5		Jumpers	≈ 0
6	2	Resistor $10k\Omega$ $3W$	0.50
7	2	Resistor $330k\Omega$ $0,25W$	0.30
8	1	Rectifier Bridge	0.31
9	1	TRIAC BT137-600D	0.86
10	1	Optocouple 4N25	0.37
11	1	Optocouple MOC3020	0.43
12	2	Integrated Circuit Socket 6 ways	0.36
13	1	Plug Adapter	1.27
14	0.5m	Wire 25 cm	4.49
		Total	**29.81**

Table 4.2: Cost Sheet

Comparing the final price with an interrupter regulator seen in marketplaces like [44], which supports maximum power of 150W it reaches a value of €24.35 against the €29.81 of Table 4.2.

Another product to be compared is the smart dimmer by Smartify which support until 1000 W with the price of €29.99 [45]. This one integrates voice commands being compatible with the principal voice assistant.

At first, this prototype seems more expensive, but the possibilities of improving this application with the same hardware turn this difference in price negligible.

Chapter 5

Conclusion and Future Works

Recapitulating the main objectives of this work, it could be listed in these topics:

- Report the importance of renewable sources and give an efficient alternative energetically;
- Do a case study of which loads could take advantage of the surplus energy of photovoltaic system generation since it is helpful for the consumer;
- Build a low-cost model of a smart plug able to send and receive data from a supervisory system that can read the surplus and send this power value to the plug.

The first topic was explored in Chapter 1 giving a general context of the importance of renewable sources for the planet and thus the appliance of efficiency on it. Some obstacles were found in energy selling pricing because it is not regulated.

Based on the photovoltaic system scenario, some alternatives were found to shift this surplus energy. For power control, it is more convenient to handle only resistive loads. This could be extended in future works to cover all load types, but these other loads can continue to be managed with traditional ON/OFF controls. Hydrogen production is another potential future application that could be examined in future research.

After, the smart plug development passed by several tests to reach a low-cost model showing limitations especially in synchronizing the grid frequency with the programming

code but, it was fundamental to report that some ways to implement it could be not recommended in similar applications. The final version shows satisfactory responses to the supervisory commands and is able to improve in the future giving attention to the security of the circuit avoiding undesirable current peaks.

The plug is prepared to receive the percentage of power available after the surplus is measured and processed by the supervisory which will provide the necessary level from the total power.

Lastly, it has been done a comparison of the price of a device in the marketplace similar to this work. It presents similar prices, but the prototype has more possibilities for improvement with the same hardware and integrates more devices.

In summary, it was possible to develop a device that can actuate efficiently using electronics, microcontrollers, and the internet of things the knowledge that could be increased lately.

Bibliography

[1] K. Kaygusuz, "Energy for sustainable development: A case of developing countries," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 2, pp. 1116–1126, 2012, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2011.11.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032111005491.

[2] ABESCO, *Desperdício de energia gera perdas de r$ 12,6 bilhões*, http://www.abesco.com.br/novidade/desperdicio-de-energia-gera-perdas-de-r-126-bilhoes/, Oct. 2015.

[3] ERSE, "Parâmetros de regulação para o período 2022 a 2025," Tech. Rep., Oct. 2021.

[4] P. DGEG/MAAC, "Consumidores de electricidade: Total e por tipo de consumo," Tech. Rep., May 2022.

[5] S. Paul, T. Dey, P. Saha, S. Dey, and R. Sen, "Review on the development scenario of renewable energy in different country," in *2021 Innovations in Energy Management and Renewable Resources(52042)*, 2021, pp. 1–2. DOI: 10.1109/IEMRE52042.2021.9386748.

[6] J. de Negócios, *Portugal é competitivo na energia solar*, 2022. [Online]. Available: https://www.jornaldenegocios.pt/negocios-iniciativas/detalhe/portugal-e-competitivo-na-energia-solar.

[7] IOL, *Maior parque solar de portugal terá capacidade para alimentar 110 mil casas*, 2022. [Online]. Available: https://away.iol.pt/energia/solar/maior-parque-solar-de-portugal-tera-capacidade-para-alimentar-110-mil-casas/20221225/63a43c5e0cf2254fb289c1ee.

[8] M. T. Ameli, S. Moslehpour, and M. Shamlo, "Economical load distribution in power networks that include hybrid solar power plants," *Electric Power Systems Research*, vol. 78, no. 7, pp. 1147–1152, 2008, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2007.09.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378779607001952.

[9] V. Ignatova, *Producing too much solar power? here's how you can manage the excess*, https://blog.se.com/building-management/2019/09/12/producing-too-much-solar-power-heres-how-you-can-manage-the-excess/, Sep. 2019.

[10] G. Energy, *É rentável a venda excedente do autoconsumo?* https://goldenergy.pt/blog/autoconsumo/venda-excedente-do-autoconsumo/, Aug. 2022.

[11] ERSE, *Lista de preços de ofertas comerciais*, https://www.erse.pt/simuladores/lista-de-precos-de-ofertas-comerciais/, Jan. 2023.

[12] B. Subudhi and R. Pradhan, "A comparative study on maximum power point tracking techniques for photovoltaic power systems," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 89–98, 2013. DOI: 10.1109/TSTE.2012.2202294.

[13] C. E. Rewiews, "Mppt solar charge controllers explained," Tech. Rep., Aug. 2022.

[14] R. E. Brown and H. L. Willis, *Electric Power Distribution Reliability*. Marcel Dekker, 2008, ISBN: 0-8247-0798-2.

[15] D. L. Ransom, "Choosing the correct transfer switch," *IEEE Transactions on Industry Applications*, vol. 49, no. 6, pp. 2820–2824, 2013. DOI: 10.1109/TIA.2013.2265376.

[16] R. Abeuov, N. Batseva, and D. Kuptsova, "Synchronous automatic transfer switch by programmed trajectory movement control for distributed generation," in *2021 XVIII International Scientific Technical Conference Alternating Current Electric Drives (ACED)*, 2021, pp. 1–4. DOI: 10.1109/ACED50605.2021.9462274.

[17] EDP, *Termoacumulador solar edp*, https://www.edp.pt/particulares/servicos/termoacumulador-solar/.

[18] R. Boudries and A. Khellaf, "Techno-economic study of solar electrolytic hydrogen production at high temperature," in *2020 6th International Symposium on New and Renewable Energy (SIENR)*, 2021, pp. 1–5. DOI: 10.1109/SIENR50924.2021.9631906.

[19] Q. Wang and L. Pan, "Study on floating offshore wind power hydrogen production and hydrogen production ship," in *2022 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)*, 2022, pp. 899–905. DOI: 10.1109/ICAICA54878.2022.9844498.

[20] M. Silverio-Fernández, S. Renukappa, and S. Suresh, "What is a smart device?" *Vis. in Eng.*, vol. 6, pp. 1–10, May 2018. DOI: https://doi.org/10.1186/s40327-018-0063-8.

[21] R. v. Kranenburg, *The Internet of Things: A Critique of Ambient Technology and the All-seeing Network of RFID*. Institute of Network Cultures, 2008.

[22] i. o. w. t. w. g. R. o. t. E. E. Networked enterprise & RFID INFSO G. 2 micro & nanosystems, *Internet of things in 2020, Roadmap for the future*. European Association on Smart Systems Integration, May 2008.

[23] X. U. States, *Mi smart plug wi-fi*, https://www.mi.com/us/mj-socket/.

[24] Yeelight, *Yeelight wireless smart dimmer bluetooth wall switch*, https://www.yeelight.com/pages/products.html/dimmer.

[25] M. Rokonuzzaman, M. K. Mishu, M. R. Islam, M. I. Hossain, M. Shakeri, and N. Amin, "Design and implementation of an iot-enabled smart plug socket for home energy management," in *IEEE 2021 The 5th International Conference on Smart Grid and Smart Cities (ICSGSC)*, IEEE Computer Society, 2021, pp. 50–54, ISBN: 978-0-7381-3315-7. DOI: 10.1109/ICSGSC52434.2021.9490420.

[26] E. Al-Hassan, H. Shareef, M. M. Islam, A. Wahyudie, and A. A. Abdrabou, "Improved smart power socket for monitoring and controlling electrical home appliances," *IEEE Acess*, vol. 6, pp. 49 292–49 305, Sep. 2018. DOI: 10.1109/ACCESS.2018.2868788.

[27] M. Alembong, I. Essiet, and Y. Sun, "Swift automatic transfer switch based on arduino mega 2560,triacs bluetooth and gsm," in *2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET)*, 2021, pp. 1–6. DOI: 10.1109/SeFet48154.2021.9375773.

[28] E. Systems, *Esp32*, https://www.espressif.com/en/products/socs/esp32.

[29] G. Gridling and B. Weiss, *Introduction to Microcontrollers*, 1.4. Vienna University of Technology, Institute of Computer Engineering and Embedded Computing Systems Group, 2007, ISBN: 978-85-64574-21-2.

[30] R. L. Boylestad and L. Nashelsky, *Electronic devices and circuit theory*, 11st. Pearson, 2013, ISBN: 978-85-64574-21-2.

[31] W.-T. E. P. Semiconductors, *B40c1000 – b500c1000 1.0a bridge rectifier*, https://pdf1.alldatasheet.com/datasheet-pdf/view/33908/WTE/B250C1000.html, 2002.

[32] L. Componentes, *B250c1000 puente rectificador 600v, 1a[b250c1000]*, https://www.lh.pe/b250c1000-puente-rectificador-600v-p-6838.html.

[33] P. Semiconductors, *Power Semiconductor Applications*. Philips Semiconductors, 1994.

[34] ElectronicaPT, *Diac*, https://www.electronica-pt.com/diac.

[35] P. S. .-. P. Specification, *Triacs - bt137x series*, https://pdf1.alldatasheet.com/datasheet-pdf/view/16775/PHILIPS/BT137X.html, Sep. 1997.

[36] I. Motorola, *Semiconductor technical data - 6-pin dip optoisolators transistor output*, https://pdf1.alldatasheet.com/datasheet-pdf/view/2846/MOTOROLA/4N25.html, 1995.

[37] I. Components, *Optically coupled bilateral switch non-zero crossing triac*, https://pdf1.alldatasheet.com/datasheet-pdf/view/166664/ISOCOM/MOC3020.html, Dec. 2002.

[38] A. Docs, *Getting started with arduino ide 2.0*, https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2.

[39] HiveMQ, *MQTT & MQTT 5 Essentials: A comprehensive overview of MQTT facts and features for beginners and experts alike*, Vol. 1. HiveMQ, 2020, ISBN: 9783-00-067913-1.

[40] J. Wan, Z. Zeng, and M. Huang, "Emi impaction by the triac zero crossing detection trigger," in *2017 IEEE 5th International Symposium on Electromagnetic Compatibility (EMC-Beijing)*, 2017, pp. 1–5. DOI: 10.1109/EMC-B.2017.8260374.

[41] Arduino, *Attachinterrupt()*, https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/.

[42] Espressif, *Timer*, https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/timer.html#timeralarmwrite.

[43] PTRobotics, *Ptrobotics - especialistas em componentes electrónicos*, https://www.ptrobotics.com/.

[44] efectoLED, *Regulador interruptor conmutado universal led triac*, https://www.efectoled.com/pt/comprar-reguladores-de-potencia/13696-regulador-led-triac-300w-com-controle-remoto-ir.html?utm_source=google&utm_medium=pmax&utm_campaign=PT_Pro_Componentes_electronicos_y_

luces_de_emergencia_Max_Valor&gclid=Cj0KCQiA6fafBhC1ARIsAIJjL8khw-KKBui6aTTS04EAz3tYErt6ux80blE93s44wGVEHH_8jJSfVDEaAq9WEALw_wcB.

[45] SmartFy, *Interruptor inteligente de luz wifi dimmer smartify - preto*, https://smartify.pt/products/interruptor-inteligente-de-luz-wifi-dimmer-smartify-preto?_pos=8&_sid=0e41f0274&_ss=r.

Appendix A

Code Steps

A.1 Phase Angle Method Code for Arduino

```
1  //naming pins
2
3  const unsigned int trigger = 8; //write in gate TRIAC BT138 (D2)
4  const unsigned int detector = 3;  //read from optocouple 4N25 (OK03)
5
6  float RISING_DELAY = 1080;  //obtained experimentally
7
8  //fix frequency of energy network and the period (1 cicle):
9  float freq = 50;     //portuguese grid frequency
10 float T = (1/freq)*1000000;    //period converted in us
11
12 volatile float level = 20;  //percentage of total power desired
13
14 void setup() {
15   // put your setup code here, to run once:
16   pinMode(detector, INPUT_PULLUP);
17   pinMode(trigger, OUTPUT);
18   Serial.begin(9600);
19
20   digitalWrite(trigger, LOW);
```

```
21
22    //attachInterrupt(interruption, function to run, mode)
23    attachInterrupt(digitalPinToInterrupt(detector), triggerActivate, RISING);
24  }
25
26
27  void loop() {
28    // put your main code here, to run repeatedly:
29
30  }
31
32  //when the detector goes LOW to HIGH
33  void triggerActivate(){
34
35    //wait the time matching to the alpha angle to allow the current
36    //dealing to microsseconds to more precision
37
38    if(level<17) RISING_DELAY=0; //correction to avoid blink
39
40    //calculating the delay time to allow current
41    delayMicroseconds(((T/2)+RISING_DELAY)*((100-level)/100));
42
43    //pulse
44    digitalWrite(trigger, HIGH);
45    delayMicroseconds(700);
46    digitalWrite(trigger, LOW);
47  }
```

A.2 Phase Angle Method Code for ESP32

```
1  float RISING_DELAY = 1080;
2
3  //naming pins
```

```
4
5  uint8_t trigger = 12; //write in gate TRIAC BT138 (D2)
6  uint8_t detector = 25; //read from optocouple 4N25 (OK03) the zero crossing
7
8  //fix frequency of energy network and the period (1 cicle):
9  float freq = 100;
10 float T = (1/freq)*1000000;     //period converted in us
11 uint_fast32_t timeActivation = 0;
12
13 volatile float level = 50;
14
15 void setup() {
16   // put your setup code here, to run once:
17   pinMode(detector, INPUT_PULLUP);
18   pinMode(trigger, OUTPUT);
19   Serial.begin(115200);
20
21   digitalWrite(trigger, LOW);
22   //attachInterrupt(interruption, function to run, mode)
23
24   attachInterrupt(digitalPinToInterrupt(detector), triggerActivate, RISING);
25 }
26
27
28
29 void loop() {
30   // put your main code here, to run repeatedly:
31   //fix frequency of energy network and the period (1 cicle):
32
33   timeActivation = ((T/2)+RISING_DELAY)*((100-level)/100);
34 }
35
36 //wait the time matching to the alpha angle to allow the current
37 //dealing to microsseconds to more precision
38
```

```
39 //when the detector goes LOW to HIGH
40 void triggerActivate(){
41   if(level<17) RISING_DELAY=0;
42
43   delayMicroseconds(timeActivation);
44   //pulse
45   digitalWrite(trigger, HIGH);
46   delayMicroseconds(700);
47   digitalWrite(trigger, LOW);
48 }
49 }
```

A.3 Using ESP32 Timers

```
1  //naming pins
2
3  const unsigned int trigger = 8; //write in gate TRIAC BT138 (D2)
4  const unsigned int detector = 3;  //read from optocouple 4N25 (OK03)
5
6  float RISING_DELAY = 1080;  //obtained experimentally
7
8  //fix frequency of energy network and the period (1 cicle):
9  float freq = 50;    //portuguese grid frequency
10 float T = (1/freq)*1000000;   //period converted in us
11
12 volatile float level = 20;  //percentage of total power desired
13
14 void setup() {
15   // put your setup code here, to run once:
16   pinMode(detector, INPUT_PULLUP);
17   pinMode(trigger, OUTPUT);
18   Serial.begin(9600);
19
```

```
20    digitalWrite(trigger, LOW);
21
22    //attachInterrupt(interruption, function to run, mode)
23    attachInterrupt(digitalPinToInterrupt(detector), triggerActivate, RISING);
24  }
25
26
27  void loop() {
28    // put your main code here, to run repeatedly:
29
30  }
31
32  //when the detector goes LOW to HIGH
33  void triggerActivate(){
34
35    //wait the time matching to the alpha angle to allow the current
36    //dealing to microsseconds to more precision
37
38    if(level<17) RISING_DELAY=0; //correction to avoid blink
39
40    //calculating the delay time to allow current
41    delayMicroseconds(((T/2)+RISING_DELAY)*((100-level)/100));
42
43    //pulse
44    digitalWrite(trigger, HIGH);
45    delayMicroseconds(700);
46    digitalWrite(trigger, LOW);
47  }
```

A.4 Phase Angle Method Code for ESP32

```
1  //naming pins
2  #define trigger 12
```

```
3  #define zeroCrossing 23
4
5  //to stop interruptions
6  portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
7
8  //timers
9  hw_timer_t *timer = NULL;
10
11 //limits to the delay detection
12 float minTime = 0+700;
13 float maxTime = 10000-1300;
14
15 //for calculus (power, conspumtion...)
16 float crossingTime = 0; //register the moment of the zero crossing
17 float pinHighTime = 0; //register the moment of rising edge of TRIAC
18 float timeOff = 0;  //to calculate the real waiting time
19
20 //to calculate the waiting  time to trig the TRIAC
21 uint64_t timeActivation = 0;
22
23 volatile float level = 0;
24
25 //ISR to regist the zero-crossing
26 ICACHE_RAM_ATTR void ISR_zeroCross()  {
27   //if(currentBrightness == IDLE) return;
28   portENTER_CRITICAL_ISR(&mux); // turn off interruptions for a while
29     crossingTime = micros();  //catch this moment
30     setTimer(timeActivation, &ISR_turnPinHigh); //time delay for triggering
31   portEXIT_CRITICAL_ISR(&mux); // turn on the interruptions
32 }
33
34 //set the timer parameters to rising (ISR_turnPinHigh)
35 //and pulse duration (ISR_turnPinLow)
36 ICACHE_RAM_ATTR void setTimer(uint64_t timeConfig, void (*ISR)(void)){
37   timer = timerBegin(0, 80, true);
```

```
38    timerAttachInterrupt(timer, ISR, true);
39    timerAlarmWrite(timer, timeConfig, false);
40    timerAlarmEnable(timer);
41    portEXIT_CRITICAL_ISR(&mux);
42  }
43
44  //rising trigger pulse
45  ICACHE_RAM_ATTR void ISR_turnPinHigh(){
46    portENTER_CRITICAL_ISR(&mux);
47      digitalWrite(trigger, HIGH);
48      pinHighTime = micros(); //catch this moment
49      timeOff = pinHighTime-crossingTime;
50      setTimer(70, &ISR_turnPinLow);
51    portEXIT_CRITICAL_ISR(&mux);
52  }
53
54  //falling trigger pulse
55  ICACHE_RAM_ATTR void ISR_turnPinLow(){
56    portENTER_CRITICAL_ISR(&mux);
57      digitalWrite(trigger, LOW);
58    portEXIT_CRITICAL_ISR(&mux);
59  }
60
61  void setup() {
62    // put your setup code here, to run once:
63    pinMode(zeroCrossing, INPUT_PULLUP);
64    pinMode(trigger, OUTPUT);
65    Serial.begin(115200);
66
67    digitalWrite(trigger, LOW);
68
69    //attachInterrupt(interruption, function to run, mode)
70    attachInterrupt(digitalPinToInterrupt(zeroCross), ISR_zeroCross, RISING);
71  }
72
```

```
73 void loop() {
74   // put your main code here, to run repeatedly:
75
76   level = 15;
77   level = constrain(level, 0, 100); //limit the level between 0 and 100
78
79   //do the proportion between time and level
80   timeActivation = map(level, 0, 100, maxTime, minTime);
81
82   //Serial.println(timeOff);
83 }
```

A.5 Final Code

```
1  //libraries
2  #include <WiFi.h>
3  #include <PubSubClient.h>
4
5  //naming pins
6  #define trigger 12
7  #define zeroCrossing 25
8
9  //limits to the delay detection
10 unsigned long minTime = 600;
11 unsigned long maxTime = 9600;
12
13 //to stop interruptions
14 portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
15
16 //timer
17 hw_timer_t * timer
18
19 //store the time delay to pulse the gate
```

```
20 unsigned long timeActivation = 0;
21
22 //store the percentage of power intended and the previous
23 volatile float level = 0;
24 volatile float previousLevel = 0;
25
26 //----------------------WiFi attributes----------------------
27 const char* SSID = "ssid";
28 const char* PASSWORD =  "password";
29
30 WiFiClient Wificlient;   //storage the data of the connected client
31 //-----------------------------------------------------------
32
33 //----------------------MQTT attributes----------------------
34 const char* BROKER_MQTT = "broker.mqtt-dashboard.com";
35 int BROKER_PORT = 1883;
36
37 #define ID_MQTT "ESP_PLUG" //unique for each device
38 #define TOPIC_SUBSCRIBER "GRID_INFO"  //same between the pub and sub
39 PubSubClient MQTT(Wificlient);
40 //-----------------------------------------------------------
41
42 //-----------------------functions-----------------------
43 void connectWiFi();
44 void connectMQTT();
45 void keepConnections();
46 void receivePackage(char* topic, byte* payload, unsigned int length);
47 void sendPackage();
48 //-----------------------------------------------------------
49
50 //----------------------interruptions----------------------
51 //ISR to regist the zero-cross
52 ICACHE_RAM_ATTR void ISR_zeroCross()  {
53   portENTER_CRITICAL_ISR(&mux); // turn off interruptions for a while
54     crossingTime = micros();  //catch this moment
```

```
55      setTimer(timeActivation, &ISR_turnPinHigh); //time delay for triggering
56    portEXIT_CRITICAL_ISR(&mux); // turn on the interruptions
57  }
58
59  //set the timer parameters to rising (ISR_turnPinHigh)
60  //and pulse duration (ISR_turnPinLow)
61  ICACHE_RAM_ATTR void setTimer(uint64_t timeConfig, void (*ISR)(void)){
62    timer = timerBegin(0, 80, true);
63    timerAttachInterrupt(timer, ISR, true);
64    timerAlarmWrite(timer, timeConfig, false);
65    timerAlarmEnable(timer);
66    portEXIT_CRITICAL_ISR(&mux);
67  }
68
69  //rising trigger pulse
70  ICACHE_RAM_ATTR void ISR_turnPinHigh(){
71    portENTER_CRITICAL_ISR(&mux);
72      digitalWrite(trigger, HIGH);
73      pinHighTime = micros(); //catch this moment
74      timeOff = pinHighTime-crossingTime;
75      setTimer(70, &ISR_turnPinLow);
76    portEXIT_CRITICAL_ISR(&mux);
77  }
78
79  //falling trigger pulse
80  ICACHE_RAM_ATTR void ISR_turnPinLow(){
81    portENTER_CRITICAL_ISR(&mux);
82      digitalWrite(trigger, LOW);
83    portEXIT_CRITICAL_ISR(&mux);
84  }
85  //-------------------------------------------------------
86
87  //---------------mainly arduino functions----------------
88  void setup() {
89    // put your setup code here, to run once:
```

```
90    pinMode(zeroCrossing, INPUT_PULLUP);  //detect the zero cross
91    pinMode(trigger, OUTPUT); //pin to pulse the gate
92    Serial.begin(115200); //speed rate
93
94    digitalWrite(trigger, LOW);   //start low before calculate
95
96    WiFi.begin(SSID, PASSWORD);
97    connectWiFi();
98   MQTT.setServer(BROKER_MQTT, BROKER_PORT);
99   MQTT.setCallback(receivePackage);
100
101 //attachInterrupt(interruption, function to run, mode)
102   attachInterrupt(digitalPinToInterrupt(zeroCross), ISR_zeroCross, RISING);
103 }
104
105 void loop() {
106   // put your main code here, to run repeatedly:
107   keepConnections();
108   sendPackage();
109   MQTT.loop();
110
111   level = constrain(level, 0, 100); //limit the level between 0 and 100%
112
113   //do the proportion between time and level
114   timeActivation = map(level, 0, 100, maxTime, minTime);
115
116   //Serial.println(timeOff);
117 }
118 //--------------------------------------------------------
119
120 //------------Wi-Fi and MQTT implementations-------------
121 void connectWiFi(){
122     if (WiFi.status() == WL_CONNECTED) {
123      return;
124     }
```

```
125
126     while(WiFi.status() != WL_CONNECTED){
127       delay(500);
128       Serial.print(".");
129       level = 100;
130     }
131
132     Serial.println("␣");
133     Serial.print("Connected␣to␣WiFi␣");
134     Serial.print(SSID);
135     Serial.print("␣|␣IP␣Address:␣");
136     Serial.println(WiFi.localIP());
137 }
138
139 void connectMQTT(){
140   while (!MQTT.connected()){
141     Serial.print("Connecting␣to␣Broker␣");
142     Serial.println(BROKER_MQTT);
143
144     if(MQTT.connect(ID_MQTT)){
145       Serial.println("Connection␣to␣Broker␣successfully!");
146       MQTT.subscribe(TOPIC_SUBSCRIBER);
147     }
148
149     else{
150       Serial.println("No␣connection␣found.␣Trying␣to␣reconnect...");
151       delay(1000);
152     }
153   }
154 }
155
156 void keepConnections(){
157   if(!MQTT.connected()){
158     connectMQTT();
159   }
```

```
160   connectWiFi();
161 }
162
163 void receivePackage(char* topic, byte* payload, unsigned int length){
164   previousLevel = level;
165
166   String msg; //string to store the package message
167
168   for(int i = 0; i < length; i++){
169     char c = (char)payload[i];
170     msg += c;
171   }
172
173   //converting to float to apply in the input of the TRIAC
174   level = msg.toFloat();
175 }
176
177 void sendPackage(){
178   char dif[5]={0};
179
180   sprintf(dif,"%.2d", timeOff);     //convert to string
181
182  MQTT.publish(TOPIC_PUBLISHER, dif);
183 }
184 //-------------------------------------------------------------
```

A.6 Supervisory Test

```
1
2 import string
3 from PyQt5 import  uic, QtCore, QtGui, QtWidgets
4 from PyQt5.QtGui import QPixmap
5 from PyQt5.uic import loadUi
```

```
6
7  import time          #implement delays
8  import threading     #run tasks "in parallel"
9  from paho.mqtt import client as mqtt
10 from random import randrange, uniform
11
12 app=QtWidgets.QApplication([])       #QTWidgets class in a variable
13 view=uic.loadUi("view.ui")       #storage and loading of the view
14
15 mqttBroker ="broker.mqtt-dashboard.com"
16
17 client = mqtt.Client("SUPERVISORY")
18 client.connect(mqttBroker)
19
20 def main():
21     while True:
22         #publish
23         payload = str(view.level.value())
24         client.publish("SMARTPLUG_LEVEL", payload)
25         print("Just published " + str(payload) + " to topic POWER LEVEL")
26
27         #subscribe
28         client.subscribe(topic1)
29         client.on_message = on_message
30         time.sleep(0.5)
31         print(" ")
32
33 def on_message(client, userdata, msg):
34     realPercentage = (10000-float(msg.payload.decode()))/100
35     view.realLevel.setText(str(realPercentage))
36
37 #"parallel" functions to run
38 threading.Thread(target=main).start()
39 threading.Thread(target=client.loop_start).start()
40
```

```
41 view.show()      #exibhition of the view
42 app.exec();      #running QTWidgets applications
```

A.7 Supervisory Interface

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ui version="4.0">
3   <class>MainWindow</class>
4   <widget class="QMainWindow" name="MainWindow">
5    <property name="geometry">
6     <rect>
7      <x>0</x>
8      <y>0</y>
9      <width>504</width>
10     <height>339</height>
11    </rect>
12   </property>
13   <property name="windowTitle">
14    <string>MainWindow</string>
15   </property>
16   <widget class="QWidget" name="centralwidget">
17    <widget class="QTextBrowser" name="textBrowser">
18     <property name="geometry">
19      <rect>
20       <x>20</x>
21       <y>30</y>
22       <width>191</width>
23       <height>111</height>
24      </rect>
25     </property>
26     <property name="html">
27      <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
            &quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
```

```
28 &lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot
       ;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
29 p, li { white-space: pre-wrap; }
30 &lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg
       2'; font-size:7.8pt; font-weight:400; font-style:normal;&quot;&gt;
31 &lt;p align=&quot;right&quot; style=&quot; margin-top:0px; margin-bottom:0
       px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0
       px;&quot;&gt;&lt;span style=&quot; font-size:18pt;&quot;&gt;Power
       percentage&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
32       </property>
33      </widget>
34      <widget class="QSpinBox" name="level">
35       <property name="geometry">
36        <rect>
37         <x>240</x>
38         <y>30</y>
39         <width>211</width>
40         <height>111</height>
41        </rect>
42       </property>
43       <property name="font">
44        <font>
45         <pointsize>36</pointsize>
46         <weight>75</weight>
47         <bold>true</bold>
48        </font>
49       </property>
50       <property name="maximum">
51        <number>100</number>
52       </property>
53      </widget>
54      <widget class="QTextBrowser" name="textBrowser_3">
55       <property name="geometry">
56        <rect>
57         <x>20</x>
```

```
58          <y>150</y>
59          <width>191</width>
60          <height>111</height>
61         </rect>
62        </property>
63        <property name="html">
64         <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot;
               &quot;http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
65 &lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot
       ;1&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
66 p, li { white-space: pre-wrap; }
67 &lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'MS Shell Dlg
       2'; font-size:7.8pt; font-weight:400; font-style:normal;&quot;&gt;
68 &lt;p align=&quot;right&quot; style=&quot; margin-top:0px; margin-bottom:0
       px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0
       px;&quot;&gt;&lt;span style=&quot; font-size:18pt;&quot;&gt;Real
       porcentage&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
69        </property>
70       </widget>
71       <widget class="QLabel" name="realLevel">
72        <property name="geometry">
73         <rect>
74          <x>240</x>
75          <y>160</y>
76          <width>211</width>
77          <height>101</height>
78         </rect>
79        </property>
80        <property name="font">
81         <font>
82          <family>MS Shell Dlg 2</family>
83          <pointsize>36</pointsize>
84          <weight>75</weight>
85          <bold>true</bold>
86         </font>
```

```
87      </property>
88      <property name="layoutDirection">
89       <enum>Qt::LeftToRight</enum>
90      </property>
91      <property name="styleSheet">
92       <string notr="true">color: rgb(0, 0, 0);</string>
93      </property>
94      <property name="text">
95       <string>—</string>
96      </property>
97      <property name="alignment">
98       <set>Qt::AlignCenter</set>
99      </property>
100     <property name="margin">
101      <number>0</number>
102     </property>
103    </widget>
104   </widget>
105   <widget class="QMenuBar" name="menubar">
106    <property name="geometry">
107     <rect>
108      <x>0</x>
109      <y>0</y>
110      <width>504</width>
111      <height>26</height>
112     </rect>
113    </property>
114   </widget>
115   <widget class="QStatusBar" name="statusbar"/>
116  </widget>
117  <resources/>
118  <connections/>
119 </ui>
```