

# User Access Control System based on ESP32 Technology

**Vinícius Penckowski**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering (Electrical Engineering branch).

Work oriented by:

Prof. Dr. Paulo Jorge Pinto Leitão

Prof. Dr. Frederic Conrad Janzen

Bragança

2020



# User Access Control System based on ESP32 Technology

**Vinícius Penckowski**

Dissertation presented to the School of Technology and Management of Bragança to obtain the Master Degree in Industrial Engineering (Electrical Engineering branch).

Work oriented by:

Prof. Dr. Paulo Jorge Pinto Leitão

Prof. Dr. Frederic Conrad Janzen

Bragança

2020



# Acknowledgement

I would first like to thank my thesis advisors Paulo Jorge Pinto Leitão of the *Instituto Politécnico de Bragança* (IPB), and Frederic Conrad Janzen of the *Universidade Tecnológica Federal do Paraná* (UTFPR-PG), for all the support, guidance and ideas given that made this project become possible to be developed.

I would also like to thank my parents: Mauro Francisco Penckowski and Solange Maria Penckowski, for all the opportunities given, for the encouragement and inspiration throughout my years of study, as well as for the base idea and the financial support for the development of this project.

Lastly, my sincere thanks to the *Universidade Tecnológica Federal do Paraná* (UTFPR-PG) and to *Instituto Politécnico de Bragança* (IPB), and all the professors for all the knowledge and opportunities given.



# Abstract

Access Control Systems are systems that are capable of controlling user access with permission-based databases. The majority of commercial Access Control Systems nowadays, even the expansive ones, lacks many advanced features, such as the possibility to control and configure multiple sectors over Wi-Fi (including illumination), using scheduling based permissions, and without any additional servers.

This project aims to develop an Access Control System costing under US\$15, capable of registering and allowing (or denying) the access of users in multiple sectors, using up to 49 modules interconnected over Wi-Fi (one being the main module, and the other the secondary modules), using web-based graphical interfaces, allowing a centralized and practical way of configuring and setting databases.

The modules use low-range RFID tags to identify users, and are able to control electrical locks, illumination and micro-switches of it's corresponding sector, and also notify adjacent sectors of entries and exits.

To keep the project easy to use, all the settings and databases can be accessed, filtered and edited in a graphical web interface (HTML5 and CSS) provided by an internal web-server running at the ESP32 controllers, and available to authenticated users.

The result is a low cost Access Control System that is fast, reliable and easy to use product, presenting advanced features, such as multi-sector control and with wireless (Wi-Fi) communication.

**Keywords:** Access control, ESP32, Wi-Fi.



# Contents

<b>Acknowledgement</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	2
1.2 Objectives . . . . .	3
1.3 Document Structure . . . . .	4
<b>2 Context and Technologies</b>	<b>5</b>
2.1 Commercial products . . . . .	5
2.2 Related work . . . . .	6
2.2.1 ESP32 based Projects . . . . .	6
2.2.2 ESP8266 based Projects . . . . .	7
2.2.3 Arduino based projects . . . . .	8
2.2.4 Other microcontroller based projects . . . . .	8
2.3 Protocols . . . . .	9
2.3.1 SPI - Serial Peripheral Interface . . . . .	9
2.3.2 I2C - Inter-Integrated Circuit . . . . .	10
2.3.3 IEEE 802.11 (Wi-Fi) . . . . .	12
2.3.4 HTTP - Hypertext Transfer Protocol . . . . .	14
2.3.5 RFID . . . . .	15

2.4 Components used . . . . .	16
2.4.1 ESP32 . . . . .	16
2.4.2 Relay module (2 channels) . . . . .	17
2.4.3 RC522 - RFID . . . . .	18
2.4.4 DS3231 - RTC . . . . .	19
2.4.5 Micro-SD module . . . . .	20
2.4.6 Active buzzer . . . . .	21
2.5 Methods . . . . .	22
2.5.1 Arduino IDE . . . . .	22
2.5.2 Geany . . . . .	23
2.5.3 HTML - HyperText Markup Language . . . . .	24
2.5.4 CSS - Cascading Style Sheet . . . . .	25
2.5.5 JS - JavaScript . . . . .	25
2.5.6 C (Programming Language) . . . . .	26
2.5.7 Databases and configuration files . . . . .	27
<b>3 Architecture and specifications</b>	<b>29</b>
3.1 System specifications . . . . .	29
3.1.1 General configuration . . . . .	30
3.1.2 Users . . . . .	30
3.1.3 Sectors . . . . .	30
3.1.4 Schedules . . . . .	31
3.1.5 Records . . . . .	31
3.2 Architecture . . . . .	32
3.2.1 Modules . . . . .	32
3.2.2 Intercommunication and databases . . . . .	34
<b>4 System development</b>	<b>35</b>
4.1 Layout of components . . . . .	35
4.2 PCB development and prototype . . . . .	38

4.3	General operation . . . . .	41
4.3.1	Initialization . . . . .	41
4.3.2	Loop . . . . .	43
4.3.3	Server (core 01) . . . . .	44
4.3.4	Navigation bar . . . . .	46
4.3.5	Home . . . . .	47
4.3.6	Records and analysis . . . . .	48
4.3.7	Users . . . . .	51
4.3.8	Sectors . . . . .	54
4.3.9	Schedules . . . . .	57
4.3.10	Settings . . . . .	61
4.3.11	Interconnection between modules . . . . .	69
<b>5</b>	<b>Tests and discussions</b>	<b>75</b>
5.1	Speed tests . . . . .	75
5.1.1	Boot time . . . . .	75
5.1.2	File modification . . . . .	76
5.1.3	Web-server . . . . .	77
5.2	Intercommunication test . . . . .	80
5.2.1	Files . . . . .	80
5.2.2	Information . . . . .	82
5.3	Security . . . . .	84
5.4	Market analysis . . . . .	85
<b>6</b>	<b>Conclusion</b>	<b>87</b>
6.1	Project analysis . . . . .	87
6.2	Difficulties . . . . .	88
6.3	Suggested future improvements . . . . .	89

# List of Figures

2.1	Form of connection of the SPI link with multiple slaves . . . . .	9
2.2	Connections for multiple masters and slaves . . . . .	11
2.3	Pinout and appearance of the ESP32 30-pin devkit [18] . . . . .	16
2.4	Pinout and appearance of the 2-channel relay module . . . . .	17
2.5	Pinout and appearance of the RFID module - RC522 . . . . .	18
2.6	Pinout and appearance of the RTC module - DS3231 . . . . .	19
2.7	Pinout and appearance of the micro-SD card module . . . . .	20
2.8	Appearance of the active buzzer . . . . .	21
2.9	Arduino IDE with empty project . . . . .	22
2.10	Geany interface with empty project . . . . .	23
3.1	Intercommunication architecture . . . . .	32
3.2	Architecture specification . . . . .	33
3.3	Implemented architecture . . . . .	33
3.4	Databases architecture . . . . .	34
4.1	Connections between modules on the main module . . . . .	37
4.2	Prototype developed on breadboard . . . . .	38
4.3	Computational version of the PCB tracks . . . . .	39
4.4	Copper part of the printed circuit board . . . . .	39
4.5	Final prototype of the PCB project . . . . .	40
4.6	The four secondary modules and the main one . . . . .	40
4.7	Simplified flowchart of setup function . . . . .	42

4.8	Simplified flowchart of loop function of Core 02) . . . . .	43
4.9	Main panel navigation bar . . . . .	46
4.10	Cookie saved to remember the navigation bar of the last page accessed . . . . .	46
4.11	Home of the panel without using an external memory card. . . . .	47
4.12	Information received from the server to the home page. . . . .	47
4.13	Records page . . . . .	48
4.14	Record analysis page . . . . .	49
4.15	Records pagination . . . . .	50
4.16	Record analysis interface . . . . .	50
4.17	Users page . . . . .	51
4.18	Add users window . . . . .	52
4.19	User edit window . . . . .	52
4.20	File that provides information for the users page . . . . .	53
4.21	Sector page . . . . .	54
4.22	Sector add interface . . . . .	55
4.23	Sector edit interface . . . . .	55
4.24	File that provides information for the sectors page . . . . .	56
4.25	Schedules page . . . . .	57
4.26	Schedules pagination . . . . .	58
4.27	Loading interface for schedules page . . . . .	58
4.28	Edit and add schedules . . . . .	59
4.29	Database format for schedules page . . . . .	60
4.30	Settings navigation bar . . . . .	61
4.31	Network settings page . . . . .	62
4.32	Network database file . . . . .	62
4.33	Sector settings page . . . . .	64
4.34	Sector database file . . . . .	64
4.35	Date and time settings page . . . . .	66
4.36	Date and time database file . . . . .	66

4.37	File backup and restore page . . . . .	67
4.38	Advanced settings page . . . . .	68
4.39	Intercommunication between modules using normal cyclic mode. . . . .	70
4.40	Intercommunication between modules using smart cyclic mode. . . . .	71
4.41	Intercommunication between modules using update in case of access. . . . .	72
4.42	Intercommunication between modules during event record . . . . .	73
5.1	Boot time . . . . .	76
5.2	Download speed for larger files. . . . .	77
5.3	Time to load the Schedules page for the first time . . . . .	78
5.4	Time to load the Schedules page with cache . . . . .	79
5.5	Broadcast used in Smart Mode to update files . . . . .	80
5.6	Request send by the secondary modules to check for file changes . . . . .	80
5.7	Response code 200 when requesting the file “usuarios.csv” . . . . .	81
5.8	Response code 304 when requesting the file “setores.csv” . . . . .	81
5.9	Example of real usage of the system . . . . .	82
5.10	Network activity of real usage of the system . . . . .	83
5.11	Example of the recorded events . . . . .	84

# Acronyms

AES	<i>Advanced Encryption Standard.</i>
CCMP	<i>Chaining Message Authentication Code Proto-</i> <i>col.</i>
CPU	<i>Central Processing Unit.</i>
GMT	<i>Greenwich Mean Time.</i>
HF	<i>High Frequency.</i>
IoT	<i>Internet of Things.</i>
LF	<i>Low Frequency.</i>
MAC	<i>Media Access Control.</i>
MISO	<i>Master Output Slave Input.</i>
MOSI	<i>Master Output Slave Input.</i>
PCB	<i>Printed Circuit Board.</i>
RFID	<i>Radio Frequency Identification.</i>
SS	<i>Slave Select.</i>
SSID	<i>Service Set Identifier.</i>
TKIP	<i>Temporal Key Integrity Protocol.</i>
UHF	<i>Ultra High Frequency.</i>



# Chapter 1

## Introduction

An Access Control System is a device capable of allowing or denying access to someone in a given location or sector. More advanced systems are also able to take care of several users, knowing their names, identifiers and expiration dates, in addition to the possibility to configure permission schedules and control more than one sector.

Using this type of systems, it's possible to determine patterns in user behaviours, improve the control over events and guarantee that unauthorized users won't access places that they're not allowed to.

The market for this type of system is quite wide and varied, especially today, when concerns about safety, and behaviour analysis are increasing. In addition, requirements such as presence and punctuality also become easier to control when using this systems.

To be able to identify which user is requesting access, there are many technologies that can be used. One of the most common is the RFID (*Radio Frequency Identification*), that is a low cost and easy to use technology based on tags that has their own ID and are able to store information. By using passive RFID cards, there is no need for batteries on the tag, and also no need of physical contact between the tag and the RFID reader.

Some examples of the places that this type of systems can be implemented internally are university, schools, factories, industries, shops in general, commercial buildings, hotels, hospitals and even inside residences.

Also, with the emergence of IoT (*Internet of Things*) devices, concepts such as connectivity has a high importance nowadays. The use of this technology allowing different types of systems and devices to be able to intercommunicate and exchange information.

The work presented in this dissertation is a project that focus on using low cost and highly available components to develop an Access Control System for multiple sectors (in this case, spaces mainly inside buildings, such as rooms, laboratories, warehouses, refectories, offices, and many others), with identification based on low-range RFID tags and intercommunication over Wi-Fi.

## 1.1 Motivation

The main motivation for the development of this project was the high price of commercial Access Control Systems, and the lack of advanced features in some of them, such as scheduling permissions, being able to control multiple sectors over Wi-Fi, centralized and practical way to configure databases, No external hardware needed (such as additional servers), web-page based interface, and many others.

This motivation emerged during a research in the market for a similar system to be applied in a Brazilian company named "Eletro Energia Motores Elétricos LTDA.", which requested a system that contains all of the features cited above. However, no products with the requirements proposed were found at the US\$10 to US\$20 price range.

This way, developing an Access Control System that contains all the mentioned features, keeping a price under US\$15, with the use IoT technology to allow integration with other systems and based on Wi-Fi is a promising product to be developed and used in many applications, as wireless networks are present in a lot of businesses of the most varied types.

Also, the passive RFID technology has became an easy and cheap way to identify users, with tags costing less than US\$0.2 and being available in different sizes and formats.

In addition, the area of Access Control Systems has a high importance nowadays, when questions involving privacy and security are on the rise. To solve this, more companies

are adopting these types of security systems, thus being a market that tends to grow more and more.

## 1.2 Objectives

The general objective of this project is to develop a low cost and high functionality Access Control System for multiple sectors, using RFID cards to identify users, record access tries and allow or deny access at certain sectors based on permission schedules. The project has four main specific objectives, in which the development of the system was based:

The first one is the inter-connectivity, so that the Access Control System that can control multiple rooms (sectors) communicating over Wi-Fi. To do this, one of the modules will be the main one (server), containing all the databases for the secondary modules. In case of a network failure, the secondary modules must be capable of continue to work and keep recording the events based on their internal copy of the databases.

The second specific objective of this project is to keep it at a cost under 15US\$ (including the electronic components, PCB and power supply only), without the need of external devices for databases manipulation (such as computer running servers), by using wildly available and conventional components, such as the ESP32, a powerful micro-controller with built-in Wi-Fi module.

The third specific objective is to develop a highly customizable system, capable to control other peripherals, such as electrical locks, room illumination, and receive signals from micro-switches installed on doors.

The fourth specific objective of the project is to develop an easy-to-use and user-friendly interface in HTML, CSS and Javascript, that can show all the databases and allow an easy way of editing them using any device connected to the same network that has a browser.

## 1.3 Document Structure

This document has been divided into the following chapters:

- **§1 - Introduction:** Presents an introduction for Access Control Systems, as well as the motivation, objectives and applications of the project;
- **§2 - Context and Technologies:** Presents the related work, as well as general explanations about the communication protocols, components and methodology used in this project.
- **§3 - Architecture and specifications:** Presents the architecture and specifications about hardware, inter-connectivity and databases of the system.
- **§4 - System development:** Presents the development stages of project assembly and programming, including some functions performed by the system, as well as the graphical interfaces developed.
- **§5 - Tests and discussions:** Real-world tests and speed tests done after the completion of development of the project, as well as market analysis, and the results obtained.
- **§6 - Conclusion:** Final conclusions with objectives analysis, difficulties and suggestions for future improvements.

# Chapter 2

## Context and Technologies

In **Chapter 2**, some related work are presented for a better understanding of what already exists and what can be improved, including a price-based comparison with commercial products. Then, the protocols, technologies and additional software used to develop this project are presented.

### 2.1 Commercial products

It is important to note that the price of this project (arround 15 US\$) is based only on the cost of the components used (shown in 4.1), not including the time taken to develop this project, neither additional commercial costs (such as manufacturing the modules, marketing, taxes, transport or profits).

The majority of products in the market at the same price range of this project (from US\$10/module to US\$30/module) aren't able to be accessed via Wi-Fi nor control multiple sectors without cables. Most of them aren't even capable of registering the user's names, expiration dates, sector names or permission schedules.

Increasing the price range from US\$30/module to US\$60/module, a few Wi-Fi modules starts to appear, but many aren't capable of creating permission schedules, and most of them depends on an active separate web-server (another computer running specific software, or an online server) to be able to inter-communicate and manipulate the databases

and configurations of the whole system.

At the range above US\$60/module (up to 4x more than the cost of this project), up to US\$100/module, there are still some modules without Wi-Fi and without scheduling permissions, and those that has it, usually depends on external software and hardware. Even at this price, it's rare to find an Access Control System that is capable of providing intelligent intercommunication between sectors ( illumination control for example) or that can work even if the main module (server) or network stops working.

## 2.2 Related work

When searching for Access Control Systems, many papers, projects and products are available nowadays. Each of them has it's own particularities, including different features and protocols for different usages. However, no low-cost system that contains all the features of this project was found, due to different needs and applications of those other projects. Also, it's worth noting that not all the features of the other projects are included in this project, mainly to to keep the low cost of the system.

### 2.2.1 ESP32 based Projects

The most recent project found is the “Sistema de controle de acesso utilizando autenticação por RFID e gerenciamento por meio de software WEB” [1], an Access Control System that is capable of logging access attempts of the registered users of a single sector. In this project, the back-end was developed using the framework Lumen.

An external server (Amazon EC2 virtual machine) is used to deal with all the process that involves the registration and changes of tags databases. This implies in more processing power and memory available. But it also creates a downside: the proposed system needs to be connected to the Amazon Server to operate with all the developed features.

## 2.2.2 ESP8266 based Projects

One of the projects that has the most similarities with the proposed project of this dissertation is the ESP-RFID (<https://github.com/esprfid/esp-rfid>), an open source Access Control System that uses an ESP8266, RFID reader and relay modules to control electrical locks. Furthermore, the project uses an internal asynchronous web-server to provide the graphical interface and databases for the administrator.

The ESP-RFID project presents an easy to use interface and multiple functionalities for a basic access control of a single sector, that allows the administrator to access configurations, manage the users registered and visualize the logs and records. It also allows a wide range of relays to be used, and a pin-based configuration, being an easy to configure and use for multiple different setups.

Some of the features that the ESP-RFID project lack are the ability to allow scheduling times of allowance (the project can only set a user as allowed or denied for that sector) and the lack of intercommunication between multiple modules, as the project is made to be used as a single-sector system. Also, there are a few components that would make the project expand it's applications, such as a RTC module (to keep date and time persistent in case of a reboot, without needing an active internet connection for NTP usage) and a micro-sd support for extended memory for logs.

Other project using ESP8266 is the “PROTOTYPE FOR A PATRIMONIAL CONTROL AUTOMATION SYSTEM USING RFID TECHNOLOGY” [2], which even though is not a system to control access from users, it is capable of controlling the movement of objects between sectors using RFID technology and also indicate it's position. The project uses a MC522 RFID, which is used not only for reading the tags, but also for writing information about the correspondent product or object.

The process of storing information inside the RFID tag is a good alternative for a multi-sector access control system, and it's also widely used in parking lots. The downside of this method is the possibility of data corruption when using contact-less readers that don't hold the card for enough time for the process of writing and checking the data.

### **2.2.3 Arduino based projects**

Another very interesting project is presented in “RFID-Based Monitoring And Access Control System For Parliamentary Campus” [3], which aligns a low cost device with multiple applications by using an MC522 RFID reader, an arduino uno and a Zigbee module for communication (IEEE 802.15.4). This project used the Zigbee for being a low cost, low consumption and effective way of communication, supporting up to 255 nodes.

With the technological advances seen on the last years, the price of an Wi-Fi module such as the ESP32 is almost the same as the combination of an Arduino Uno + Zigbee module. Also, the ESP32 offers much more processing power and storage capabilities.

Another project is presented in “Based Kindergarten Intelligence Security System” [4], a project developed in 2012 for usage in schools. This system provides a secure way to know which students were present in the school every day by using RFID tags for identification. If the system detects that any child is absent, the project sends a a SMS (“Your Child is Absent Today”) to his or her parents via a GSM module that is connected to the Arduino. The system is also capable of limiting the entries and exits of the students, as each of the RFID cards are only allowed to work once per day.

### **2.2.4 Other microcontroller based projects**

One of the most complex projects found in the area of Access Control Systems that was implemented in universities is the “RFID Based Security and Access Control System” [5], developed in 2014. This project uses a combination of RFID tags and a camera to identify the users and allow or deny access to the room. The microcontroller used is the AT89C52, for being a low cost and low power consumption device.

The communication between the main controller and the Computer System of the “RFID Based Security and Access Control System” [5] happens over Serial Connection. The project uses a GSM module to notify the security van in case of an unauthorized access attempt.

## 2.3 Protocols

### 2.3.1 SPI - Serial Peripheral Interface

SPI is a synchronous serial communication protocol, generally used for short distances. This protocol allows a full duplex communication between the master and the slaves, as well as the use of more than one “Slave” devices with only one master. The addressing is done using an input pin on the slaves SS (*Slave Select*), as it happens exclusively via hardware. This protocol uses 4 lines for communication, which are:

- CLK: Clock line provided by the master for synchronous communication;
- MOSI (*Master Output Slave Input*)
- MISO (*Master Output Slave Input*)
- SS:line for choosing the slave.

For applications where there is a need for multiple slaves on the same bus, the connection shown in the figure 2.1 is recommended.

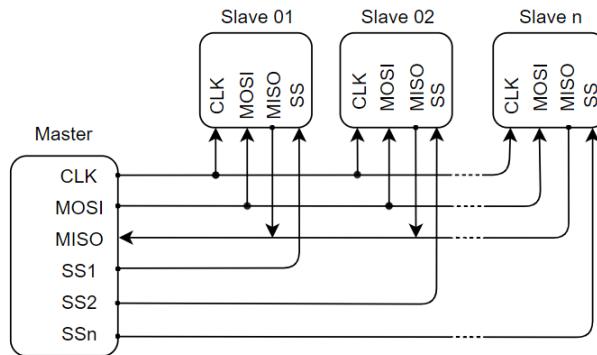


Figure 2.1: Form of connection of the SPI link with multiple slaves

The data transfer is done as follows[6]:

1. After the clock is generated by the master device, one of the SS lines is selected;
2. The slave device can send information to the master via the MISO line, while the master can send information via the MOSI line;
3. At the end of the transmission, the master device disables the SS line of the device whose communication has ended.

Thus, it is possible to verify that there is no implementation of an addressing mechanism via software, this being done via hardware. In addition, there is no check for data receipt and no flow control.

This form of communication has some advantages, including:

- High transmission speeds (MHz);
- Support for multiple slaves, limited by the number of digital ports used as SS.

However, some of the disadvantages are:

- Use of more cables for communication (total of 04 lines);
- Use of additional digital outputs for the slave selection line (SS);
- The master controls all communications, so that slave-slave communication is not possible directly [6];
- Limit of only one master;
- Low communication distances.

### 2.3.2 I2C - Inter-Integrated Circuit

The I2C protocol is a synchronous communication specification, developed by Philips in 1982, whose initial objective was the communication between CPU (*Central Processing Unit*) and peripheral chips on televisions [7].

The architecture of this protocol includes the ability to have more than one master on the same data lines, as well as multiple slaves. In addition, only two cables are required for communication [8], namely:

- SCL (Serial clock): Provides the clock for synchronous communication;
- SDA (Serial Data): data lines for all the devices.

This lower number of data lines facilitates the assembly of projects with this protocol. However, data transfer is more complex when compared to other protocols used, being performed as follows:

1. Start frame: indicates the start of the transmission;
2. Address frame: indicating which device should receive or send the information;

3. The last bit of the address will indicate whether it's a read or write request;
4. Data, whether in the master-slave sense, or vice versa, and this choice is made in the previous package (read or write);
5. After receiving and sending all data, a stop signal is sent to indicate the end of the communication.

It is important to note that each byte sent in this protocol includes an additional bit of Acknowledgement, in order to verify the correct receipt of the data.

The connection between master-slave is simple, being easily expanded to several masters and/or slaves, as shown in the figure 2.2

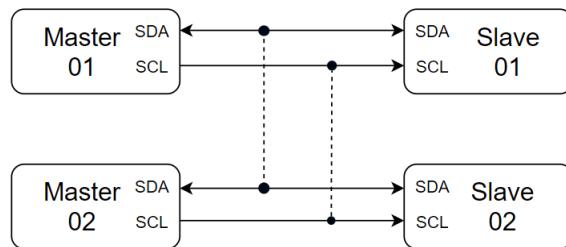


Figure 2.2: Connections for multiple masters and slaves

Therefore, this protocol has as a great highlight the possibility of interconnecting several masters and slaves. Some other advantages of this protocol are:

- Use of fewer data cables (only 2)
- Addresses are set via software;
- Easy to add and remove devices;
- Easy error detection;
- Support up to 1008 slaves (use of 10bits for addressing) [9].

However, the protocol has some disadvantages, such as:

- Lower transmission speeds, usually with the clock operating in the range of a few kHz;
- More complex software for receiving and sending data;
- Recommended use at low distances.

### 2.3.3 IEEE 802.11 (Wi-Fi)

IEEE 802.11, popularly known as Wi-Fi (registered trademark of the Wi-Fi Alliance), is a set of specifications that must be followed in order to standardize wireless local area networks. This technology represented a huge advance in the way devices connect to the internet wirelessly, as it proved to be a very flexible and viable technology.

The operation of a Wi-Fi network happens over radio waves transmitted by a device, usually the router, so that other devices within its range can connect if the credentials are correct.

These radio waves are divided into up to 14 different channels for Wi-Fi networks that operate on the base frequency of 2.4GHz, which are called channel 01 through channel 14 (in some places this number is lower, as in North America, which goes to channel 11, and Europa, which goes to channel 13). This division into channels aims to avoid interference from other networks and devices, in order to offer a less noisy signal.

It is important to highlight that normally choosing channels 1, 6 and 11 tends to be better choices, as it avoids the phenomenon known as channel overlap, which is basically a consequence of overlapping frequencies of the other channels [10].

Another important point of Wi-Fi networks is security, even more so today, where several devices use this technology to exchange important information. Among all the ways available to configure network security, the following stand out:

- Open network: Network where a password is not required for authentication
- Protection by MAC (*Media Access Control*) address :Network where only registered MAC addresses can access. Low usage because it is vulnerable to MAC cloning.
- Hidden network: Hides the name of the network. Not often used because it is easily discovered with Wi-Fi scan programs.
- WEP (Wired Equivalent Privacy): Developed in the 1990s, it is a type of protocol with password protection. For encryption, 64-bit or 128-bit keys can be used [11]. However, several security problems have been found and this protocol is being less used nowadays. [12]

- WPA (Wi-Fi Protected Access): Protocol introduced in 2003 to solve problems in the WEP protocol. With this protocol, a different key is generated for each information package using TKIP (*Temporal Key Integrity Protocol*) in addition to the use of 256-bit encryption keys [11]. However, security issues have been discovered and this protocol is being replaced by WPA2. [12]
- WPA2 (Wi-Fi Protected Access 2): Launched in 2004, the protocol presents several improvements in the field of security and creation of cryptographic keys, which is the model most used in residential applications. In this protocol, the use of RC4 and TKIP is replaced by CCMP (*Chaining Message Authentication Code Protocol*) and AES (*Advanced Encryption Standard*), which provide greater security for the network. [13]

Among the advantages of this communication protocol, the following stand out:

- Possibility of multiple devices connected;
- Easy to configure and use;
- High popularity (supporting multiple types of devices);
- Easily expandable;
- High mobility, usually used in small devices and without the need for cables;
- Low cost of product development using Wi-Fi nowadays.

However, some disadvantages also exists, including :

- Even though security has improved in the last few years, bruteforcing the key of a network using a captured handshake is possible, however it's necessary devices with high computational power.
- Range dependent of the router.

In view of the large number of advantages, it is easy to understand why this technology has become so popular in recent years.

#### **2.3.4 HTTP - Hypertext Transfer Protocol**

The HTTP protocol is a communication protocol used as one of the bases for the transfer of information on the internet.

This form of communication is done in a client-server manner, where the client can request files from a server, which then respond the request to the client. It is important to highlight that it is always the client who initiates requests [14].

The requests are usually composed of 5 basic elements, which are:[14]

- HTTP Method: Niorm GET, POST, DELETE, PUT, OPTIONS or HEAD, however, more methods are available;
- The internal server path for that specific file;
- The version of the HTTP protocol;
- Headers in general, which can indicate important information about cache and file versions;
- Body of data, in case of sending data to a server.

The responses have the following elements:[14]

- Version of the HTTP protocol;
- Status code on the requested file and server, and status information in summary form regarding the code received;
- HTTP headers referring to the request;
- Body of data, when necessary.

In this way, the HTTP protocol becomes quite functional, given that it is capable of controlling several features, such as cache, authentications, sessions, proxys, among others, without losing simplicity in the way of communication and the high possibility of expansion.

### 2.3.5 RFID

RFID is a radio frequency based identification technology. Among the most used frequencies are:

- LF (*Low Frequency*): In the range between 125 and 134kHz
- HF (*High Frequency*): In the 13.56MHz band
- UHF (*Ultra High Frequency*): In the 433MHz band or between 860 and 960MHz
- Microwave: 2.45 GHz and 5.8 GHz

In this technology, a transceiver is used to request the RFID tag code, and the tag responds with a certain signal, which can represent not only its identification code, but also stored information.

Regarding the type of tags, there are 2 main classes:

- Passive: Do not require the use of batteries [15], so the reader is responsible for powering the tag through radio waves. Thus, there is no need to recharge or change batteries, a factor that greatly reduces the dimensions and cost of this technology.
- Active: Includes the use of batteries for electrical supply to the circuit of the tag [15]. Thus, there is a need to change or recharge the batteries after a certain period of use. As an advantage over the passive, it is possible to reach greater distances using this class of tags [16].

However, even with all this diversity that technology offers, some problems are present, which stand out:

- In the case of small active tags, the short battery life;
- In the case of passive tags, the small reading range (this problem also exists in some active tags).
- The price, even though low, is still higher than some other technologies (for example, such as bar codes, for supermarket applications)
- Security issues, where some types of tags can be cloned.

## 2.4 Components used

### 2.4.1 ESP32

ESP32 is a powerful low-cost microprocessor with integrated Wi-Fi and BLE modules, a dual core processor with clocks up to 240MHz and a total of 48 pins for general usage.

Some of the main features used in this project of this micro controller are: [17] [18]:

- Integrated Wi-Fi module (IEEE 802.11 standard), with support for operation as Access point, Client and Access Point + Station;
- Dual-core Processor with 32 bit architecture (up to 240MHz);
- 512KB SRAM memory and 448KB ROM;
- Support for external flash memory (typically 4MB);
- Possibility of generating PWM on all output pins;
- Ability to change SPI and I2C interface pins via software.

In order to simplify the use of the module, some Development Kits were launched, one of the most recommended being the 30-pin DevKit V1, which combines a small size and low cost, and it is partially breadboard friendly, as shown in figure 2.3.

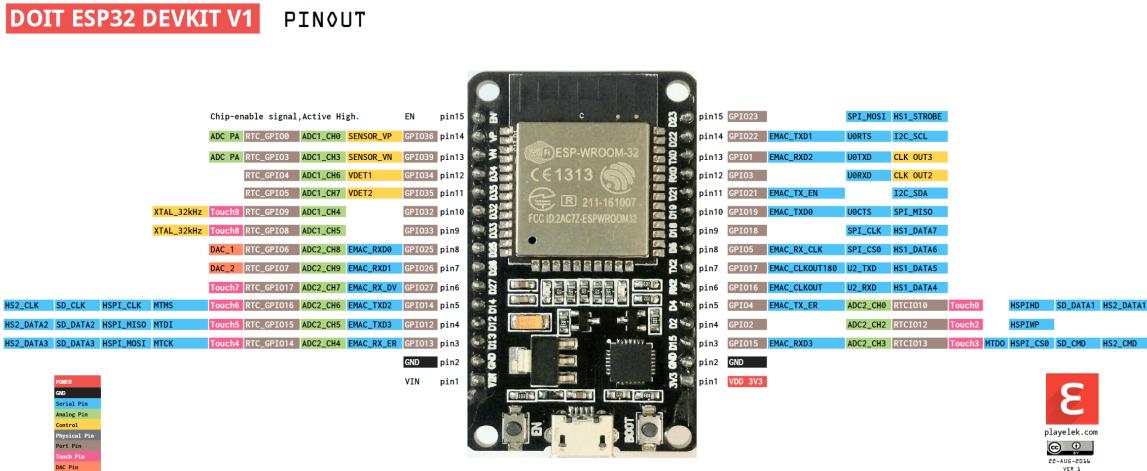


Figure 2.3: Pinout and appearance of the ESP32 30-pin devkit [18]

In this development kit, a total of 25 digital pins are available for use, in addition to supporting input voltages between 5V and 12V, and a 4MB flash memory. Also, the

use of ESP32 is simplified due to the CP2102 chip present in this development kit, which allows the chip to be reflashed via the USB port.

### 2.4.2 Relay module (2 channels)

The 2-channel relay module makes it possible to connect higher power loads (up to 220VAC/10A) via 3.3V digital signals.

In the case of this relay, the load control is activated with a low signal pulse (below 2V), thus being an active-low type module, making it possible to use it with 5V supply and 3.3V control signal.

In the case of the module represented in the image 2.4, it is possible to control two loads separately, as it contains 2 separate channels.

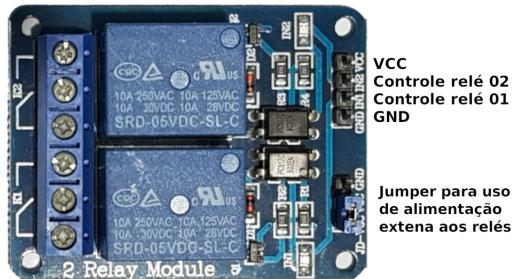


Figure 2.4: Pinout and appearance of the 2-channel relay module

In addition, it is possible to use external power for the relays by removing the jumper described in the image 2.4, or the possibility of checking the status of the relays of this module using the leds as indicators.

However, this component has a higher current consumption when compared to the other components (on average 150mA at 5V), since each relay (SRD-05VDC) used typically consumes 72mA when activated [19].

The response time is between 5ms and 10ms, which is sufficient for applications that do not require instantaneous responses.

In this project, all control units have a 2-channel relay module (according to §4.1) for electrical closing and illumination control, which can be disabled via software.

### 2.4.3 RC522 - RFID

The RC522 is a 13.56MHz RFID reader module that has the MFRC522, from NXP, as the base chip. Using this module, it is possible to read RFID cards and tags wirelessly, combining a low energy consumption with a low cost product.

The MFRC522 chip supports I2C, SPI and UART [20] communications. However, the RC522 board layout is based on the SPI protocol, with the pinouts shown in the figure 2.5.

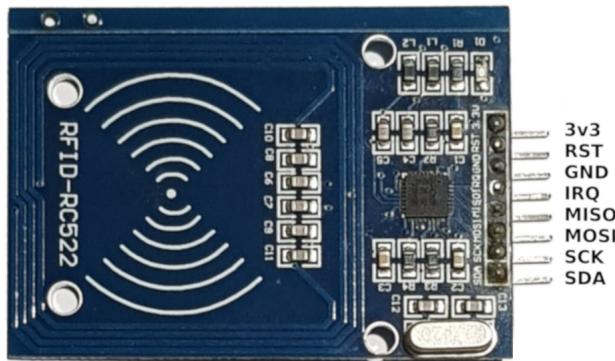


Figure 2.5: Pinout and appearance of the RFID module - RC522

In addition, some other features make this module highly attractive for use in prototypes, such as:

- Easy to use in projects with microcontrollers such as Arduinos and ESP32;
- Works with 3.3V microcontrollers;
- Low cost (< 2US\$);
- Low power consumption (<30mA);
- Supports several card technologies such as Mifare1 S50, S70 Mifare1, Mifare Ultra-Light, Mifare Pro and Mifare Desfire;
- Small dimensions and low weight;
- Breadboard friendly.

In this project, this component is used as a way of identifying users from the tags for inputs and outputs (according to §4.1).

#### 2.4.4 DS3231 - RTC

The DS3231 module is a high precision real time clock, with an integrated oscillator crystal, and a low energy consumption [21].

It is possible to use a CR2032 battery so that, in case of a power failure, date and time information is maintained, as shown in figure 2.6.



Figure 2.6: Pinout and appearance of the RTC module - DS3231

This module provides time information with precision of seconds, and is also capable of providing the minutes, hours, days, month and year, in addition to automatically making corrections such as leap years and months less than 31 days.

It also has some very interesting features, such as:

- Works with voltages between 3.3V and 5V
- Low current consumed (500nA from battery, 1.5mA from external source)
- Circuit for detecting external power failure
- Able to count years from 2000 to 2099
- 32kB AT24C32 memory chip (for additional storage)
- I2C communication, facilitating connections
- Breadboard friendly
- Small, lightweight (8g) and low cost module
- Presence of 2ppm (approximately 1 minute of error per year).

In this project, only the main module uses this module, in order to offer an alternative way to obtain the date and time, according to §4.1.

## 2.4.5 Micro-SD module

The card module developed by LC Technology supports the use of Micro SD and Micro SDHC memory cards. The amount of memory supported will depend on the microcontroller and libraries used, making it a very powerful and simple to use module.

It uses the SPI communication protocol to provide a high speed for data transmission, as well as being able to be directly connected to a 3.3V power supply and signal.

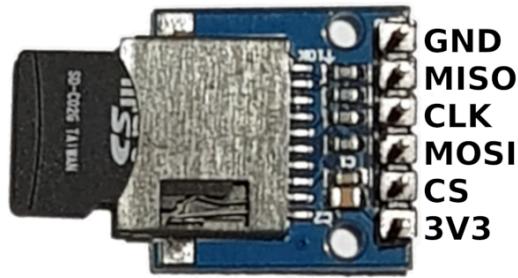


Figure 2.7: Pinout and appearance of the micro-SD card module

In addition, this module has several advantages, such as:

- Operation directly at 3.3V (without the need for voltage regulators or circuits for Level Shifting).
- Reduced size compared to other modules with similar characteristics
- Allows for easy insertion and removal of the memory card
- Easy use and application in projects with microcontrollers
- Support for Micro SD and Micro SDHC cards.
- Low cost, small dimensions and light weight
- Breadboard friendly

In this project, only the main module uses this component in order to expand the amount of memory available to store the records, according to §4.1.

#### 2.4.6 Active buzzer

The buzzer is an electronic device capable of making sounds. In the case of the active buzzer, the sound is generated by an internal electronic circuit, so that when the buzzer is energized, it will beep.

Thus, it is an ideal component for use in alerts, alarms and signaling. It has only 2 terminals, one of which is positive, the other negative, as shown in figure 2.8.

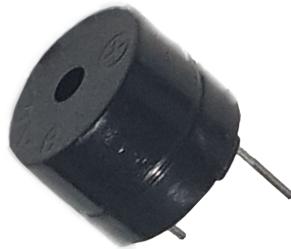


Figure 2.8: Appearance of the active buzzer

The operation of buzzers is based on the Piezoelectric effect, in which, briefly, the application of a mechanical effort is capable of causing the appearance of a potential difference in some types of crystals. In the case of buzzers, this effect is the opposite, where the application of tension causes a deformation in the internal crystal of the buzzer, making it possible to emit sounds based on the displacement that the crystal is subjected to.

This is a small and low cost component, but it has great applications due to the fact that it is capable of emitting sound. In this project, all modules have an active buzzer (according to §4.1) for indications and alerts in general, which can be disabled via software.

## 2.5 Methods

### 2.5.1 Arduino IDE

The Arduino IDE is an open-source [22] Integrated Development Environment developed in Java and based on open-source software (such as Processing). This IDE provides an easy-to-use interface and a fast way to compile and upload code to an arduino board with only a few clicks.

The figure 2.9 shows a general view of the Arduino IDE:

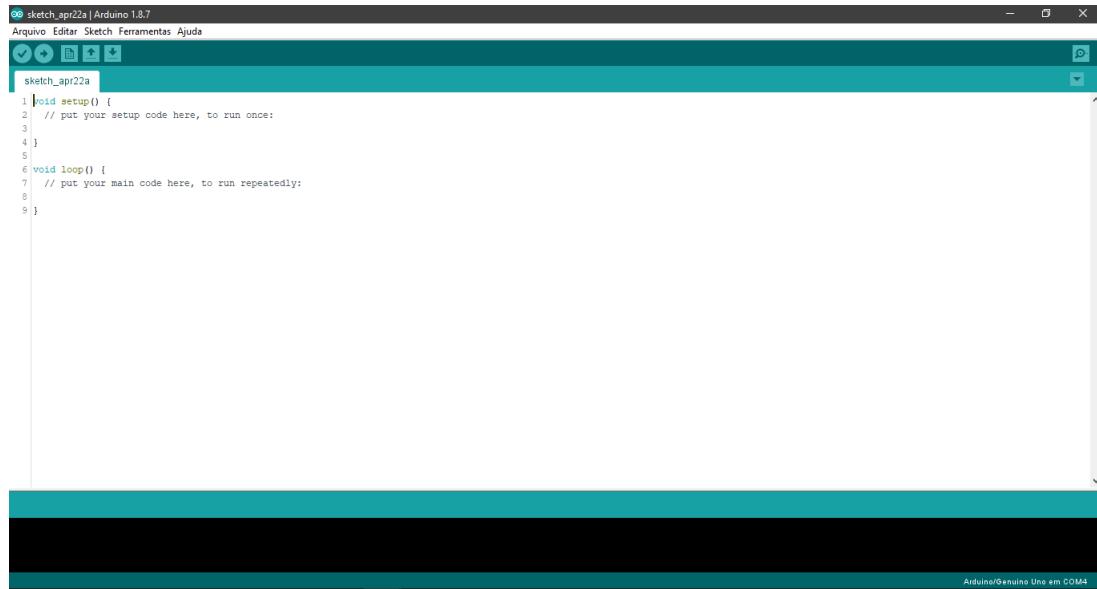


Figure 2.9: Arduino IDE with empty project

For this project, the Arduino IDE was used as a text editor, compiler and bridge to send the compiled binaries to the ESP32 of all the code in C programming language.

To be able to use the Arduino IDE to interface with the ESP32, it is necessary to install the tools provided by ESPRESSIF. This is a simple process done by adding the URL [https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json) to the Additional Board Manager URLs field (located at the preferences window of Arduino IDE).

## 2.5.2 Geany

Geany is a fast, small and easily customizable programming text editor that supports many filetypes, including all the programming languages and files used in this project [23].

The figure 2.10 shows a general view of the Geany interface:

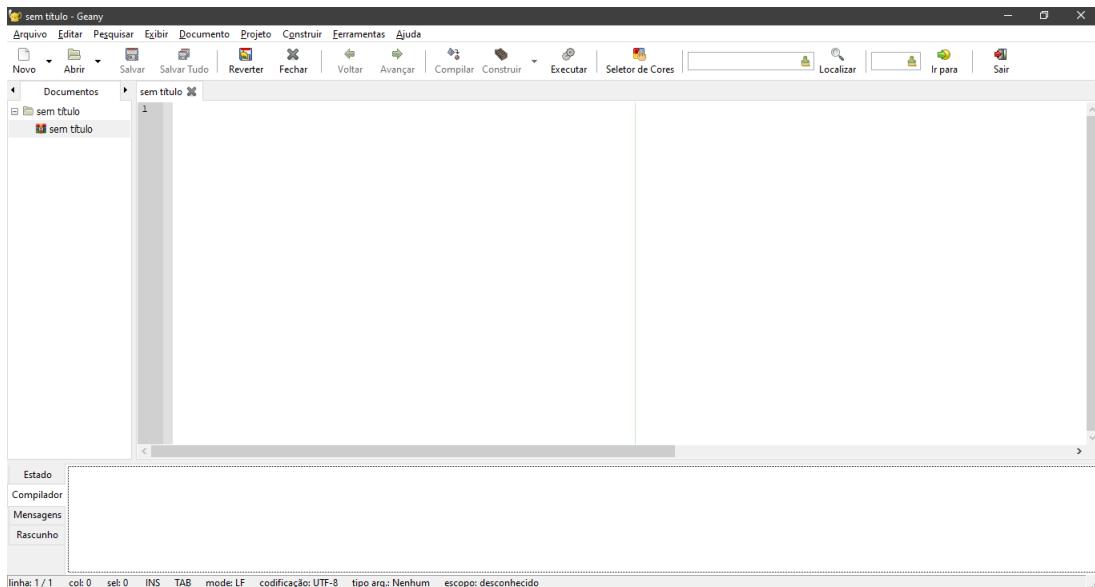


Figure 2.10: Geany interface with empty project

For this project, Geany was chosen due to it's organized and simple interface, with multiple features, such as:

- Auto-closing of HTML tags;
- Syntax highlighting;
- Auto-completion and lists for symbol names;
- Built-in system to compiler;
- Project management sidebar;
- Easy HTML execution (F5 key) in external web-browser;

For this project, Geany was used as a text editor for the HTML, CSS and Javascript files, as well as reading and creating the structure for the databases and configuration files (CSV).

### 2.5.3 HTML - HyperText Markup Language

The Hypertext Markup Language (HTML) is one of the most common markup languages used to develop web pages. This language is used to describe to the web browser the structure of a page, including data about the appearance of the document that will be rendered in the client's computer.

The use of HTML facilitates the process of structuring a web page that contains multiple types of elements (such as text, images, tables, buttons...) by allowing the developer to use structural semantics to create structured documents [24].

The HTML elements are represented by tags, which label pieces of content. These tags are used by the browsers to render the page structure and elements correctly. Some of the HTML elements used in this project are:

- <head> - Indicates the headers of the HTML page;
- <script> - Allows the scripts to be written directly in the HTML file
- <style> - Allows CSS styling to be written directly in the HTML file
- <body> - Contains all the graphical elements of a HTML page;
- <h1> - Text formatted as a title in the rendered page;
- <div> - Container usually used for grouping multiple HTML elements.
- <label> - Used as a caption for an item in the user interface
- <input> - Inputs of text/options at the graphical interface;
- <button> - Buttons that the user can interact with;
- <form> - Group of multiple elements with fields for user input;
- <table> - Used to create a table to show information organized by rows and lines;
- <br> - Indicates a line break;
- <a> - Used to create a clickable hyperlink.

Adittionaly to all the advantages cited above, this language was chosen for it's popularity and the wide range of functionalities provided. Also, HTML can embedded with other languages, such as CSS for better styling and JS, for interactive functions.

For this project, each graphical interface (presented in 4.3.3) has it's own HTML page.

#### **2.5.4 CSS - Cascading Style Sheet**

The CSS is a language capable of describing how the elements of a HTML page should be displayed at the web-browser of a client [25].

The use of a global CSS file for all the webpages of a system allows the developer to create multiple pages with a similar style without the need to rewrite the desired characteristic of the structure in every document. This also allow all the webpages to keep a pattern and simplifies the process of changing characteristics of them simultaneously.

The process of loading an external CSS file inside a HTML page is very simple and done with one line of code in the HTML headers, as shown in the following example (file.css represents the path and name of the CSS file):

```
<link href="file.css" rel="stylesheet" type="text/css">
```

It's important to note that a webpage can have it's own CSS specifications without a separate file. This is done by using a HTML block called `<style>`. Also, A HTML element can have a individual styling. This is done by using the indicator `"style=`" inside the element's delimiter.

For this project, the default styling for the majority of elements are inside a global CSS file, loaded by all the HTML pages. Some page-specific styling options are present in some HTML files, as well as some inline element styling (for example, on different colors for the same type of element).

#### **2.5.5 JS - JavaScript**

Javascript is a high-level programming language, usually compiled in real-time that can be integrated with HTML. This programming language has multiple functionalities, (such as prototype-based object-orientation and first-class functions), making it a very powerful tool to be used in web-pages.

One of the main advantages of using JavaScript as a programming language to control the HTML files is it's high integration with the elements of a HTML page. This makes possible, for example, to execute a JS function by simply pressing a HTML button [26].

The process of loading an external JavaScript file inside a HTML page is very simple and done with one line of code in the HTML headers, as shown in the following example (file.js represents the path and name of the JS file):

```
<script src="file.js"></script>
```

The `<script>` element can also be used for JS code to be written inside a HTML file. For this project, some JS scripts are built-in inside HTML pages (to avoid overloading the webserver with multiple requests at the same time), while other scripts are in separated from the HTML files for better organization and smaller file sizes.

### 2.5.6 C (Programming Language)

C is a procedural programming language developed between 1972 and 1973 for computers, that is vastly used to program micro-controllers nowadays. This programming language has multiple features that was used in this project, including:

- Full set of control flow primitives for conditions and loops (such as if, else, for, while, switch...);
- Allows easy arithmetic and logic operations and comparisons;
- Allows the use of Functions to execute tasks and return values;
- Support for Arrays and matrices of elements;
- Easy Low-level access to computer memory with pointers;
- The code files can be separately compiled and be linked together afterwards;
- Static data typing (with implicit conversions possible).

The C language was chosen to be used in the ESP32 programming part of this project due to it's high integration with the majority of libraries made for Arduino. Also, because it is a pre-compiled language, it allows micro-controllers to execute the majority of tasks faster than real-time compiled languages.

Another advantage of C programming for ESP32 is the large support offered by Espressif ®in the arduino-esp32 project (Arduino core for the ESP32), available at <https://github.com/espressif/arduino-esp32>.

### 2.5.7 Databases and configuration files

For this project, all the databases and configuration files are of the “.csv” (Comma-separated values) filetype. As CSV file format is not completely standardized, it is possible to adapt the contents and structure of this type of file to the desired application.

For data structuring in this project, the CSV files are separated by tabs (“\t”), for the data of a individual dataset, and newlines (“\n\r”) for different datasets inside the same database.

This datatype was chosen due to some advantages, such as:

- Easy to read and write, as it’s a plain text file;
- Easy to load directly to arrays and matrices (for the ESP32 processing and for the graphical interface);
- No additional libraries needed to manipulate this files;
- Easy to download and manipulate at almost any text editing program;
- Compatibility with many datasheet manipulation programs;
- Great performance for reading and editing;

The downside of using this filetype is that most of the process of finding and editing a specific line should be developed for each one of the databases.

The configuration files have a similar structure and uses the same filetype, the difference is that as the datasets are composed by only one data each, they are divided in newlines only, as shown in 4.3.10.



# Chapter 3

## Architecture and specifications

In **Chapter 3**, a general overview about the proposed specifications and architecture of this project are presented, including both hardware and software parts, as well as general capabilities about inter-connectivity, events logging (records), settings and all databases used in the system, such as users, sectors and schedules.

### 3.1 System specifications

The general specifications of the project are:

- Maximum number of registered users: 499;
- Multiple sector control, supporting up to 49 sectors simultaneously with 49 modules;
- Maximum number of schedules: up to 24.500 (500 per sector);
- Maximum number records/logs: 500.000 with extended storage;
- Illumination and electrical locks control, as well as micro-switch support.
- Centralized and easy to use web-based graphical interface in the main module;
- Intercommunication between multiple modules for user movements, as well as data and file exchange.

The maximum number of sectors and users was limited to allow all the information to be stored in the internal RAM of the ESP32, allowing a better responsiveness of the system. With the chosen specifications, 90% of the available static RAM memory was

used (other processes, such as hash encryption and network related functions uses the heap memory, which is dynamically allocated).

Detailed specifications are presented in the following subsections.

### 3.1.1 General configuration

The system is capable of connecting to Wi-Fi that uses open, WEP, WPA and WPA2 (personal) security modes. It's also capable of using static IP address or DHCP.

For the external components, it supports electrical locks that works with relays (both active-high and active-low); illumination control that doesn't exceeds 250V/10A. For micro-switches, both active-high and active-low are supported. The buzzer can be disabled for usage in quiet ambients.

To keep date and time updated, it's possible to configure the system to use NTP when internet is available, or use the internal date and time RTC module.

### 3.1.2 Users

Given that several of the applications to this system requires a considerable number of registered users, a total of **499 users** can be registered in the system, which include information such as:

- User identification code (between 01 and 499);
- Full name of the user;
- User identification UID number;
- User expiration date.

Those information can be changed using the graphical interface of the main module.

### 3.1.3 Sectors

The number of sectors is limited based on the ESP32's processing power and available memory. For the project, this number was limited to **49 sectors**, so each main module can support up to 48 active secondary modules.

In the case of sectors, it is possible to register the following information:

- Sector code (between 01 and 49);
- Sector name;
- IP of the central responsible for the sector.

### 3.1.4 Schedules

It is possible to register up to **24.500 schedules per week** (500 schedules per sector per week), a number that should be sufficient for most of the project's applications.

- Indication of which sector is allowed;
- Indication of which user is allowed;
- Initial date and time of the permission (weekly);
- End date and time of permission (weekly).

### 3.1.5 Records

The records are saved in a “.csv” format file, with data separations using tabs for data, and a new line for the records, which can be:

- Allowed access attempt (Entry/Exit);
- Denied Access attempt;
- Alarm triggered;
- Alarm disarmed;
- System reboot.

The main module stores all records from all modules. For this, a Micro-SD card can be used, in order to avoid limitations due to the internal storage space of ESP32 (detailed in §2.4.1). To keep the stability of the system, the limit of **1.500 records** was chosen for analysis in real time on the web interface (more details in §4.3), and up to **500.000 records** to extract via backup.

The secondary modules also has in its internal memory a records file to register the events of the corresponding sector and of the next sector, supporting up to 30.000 records.

## 3.2 Architecture

### 3.2.1 Modules

The system has two types of modules, both capable of controlling the access of users:

- **Main module:** Only 1 main module per system. This module contains all the databases, records and graphical interfaces of the system. This module is also responsible for serving databases and general information to the secondary modules. Also, it's the only module capable of changing the users, sectors and schedules databases via its web-based graphical interface.
- **Secondary modules:** Optional, limited to 48 secondary modules. These modules only have their own databases, logs and it's individual settings page, and is responsible for requesting databases and notifying events to the main and adjacent modules. They are only capable of changing its own records and settings files.

The modules work in a server-client communication over Wi-Fi, supporting up to 49 active modules simultaneously, as shown in figure 3.1.

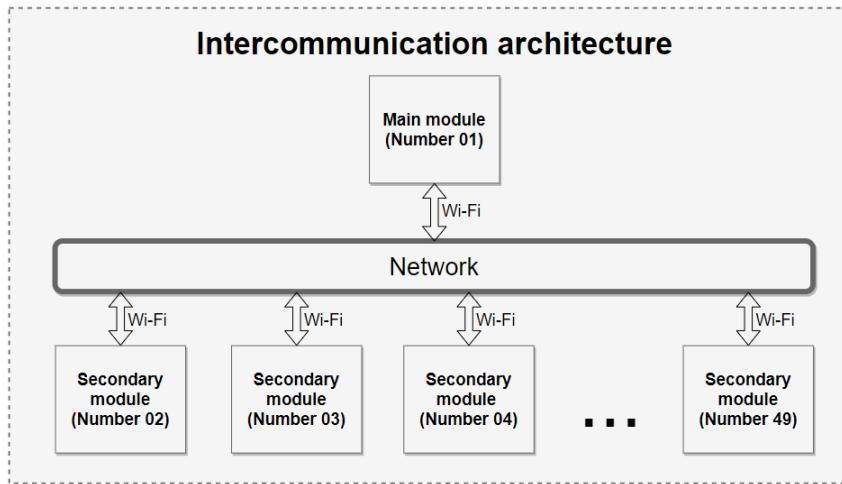


Figure 3.1: Intercommunication architecture

A generic concept of the system is shown in figure 3.2.

The system components chosen for this project are 1x ESP32 (DevKit), 2x RFID RC522 (one for entries and other for exits), 1x Dual-channel relay module (for illumination

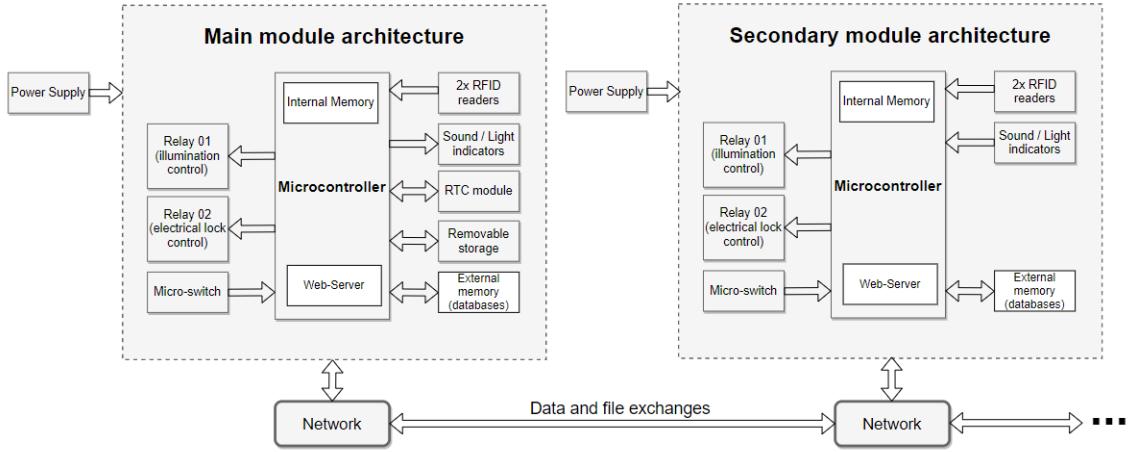


Figure 3.2: Architecture specification

and electrical lock control), 1x Buzzer, 1x Input for micro-switches, 1x RTC DS3231 (main module only), 1x Micro-SD card slot (main module only), and are grouped together as shown in figure 3.3.

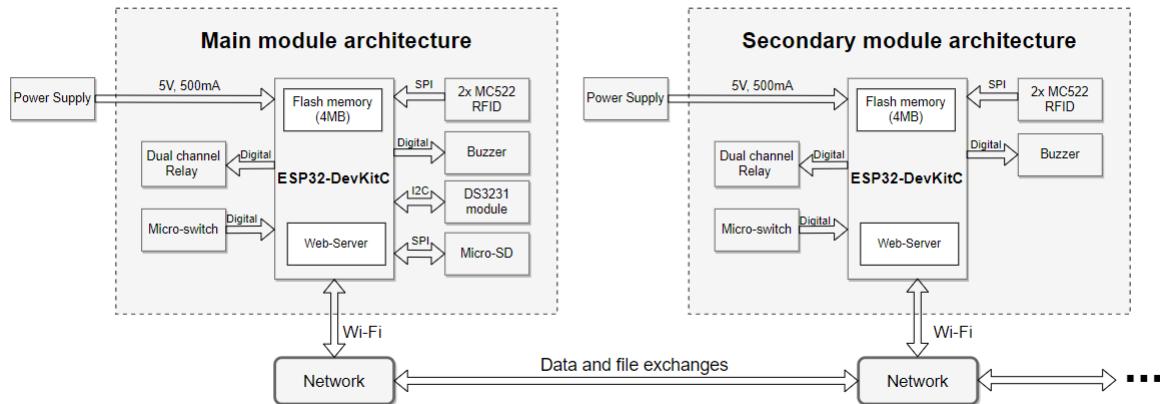


Figure 3.3: Implemented architecture

The RC522 module was chosen due to its low cost and low range, ensuring that only users that intentionally approximate their tags (within 3 centimeters of the reader) will activate the system and record the event.

### 3.2.2 Intercommunication and databases

The main module works as the server, providing all the databases to the the secondary modules (clients).

An external secure Wi-Fi network with internet is required for full system functionality, but the main module is capable of generating it's own local network for smaller ambients (limited to 4 secondary modules). If the security of the external network cannot be guaranteed, the system provides a "Secure Mode" function, explained at §4.3.10.5.

Two secondary modules can intercommunicate directly using the information provided by the sectors database, which contains all the information needed for any module to know which one is responsible for each sector. This intercommunication informs the movements of users between sectors, which allows the modules to be able to control the illumination of a sector, even if the user has left one sector through the entry of another one (example at 5.2.2).

All the databases and files used in the web-server are stored in the internal 4MB flash memory of the ESP32. The databases are saved in “.csv” filetype, as described in 2.5.7.

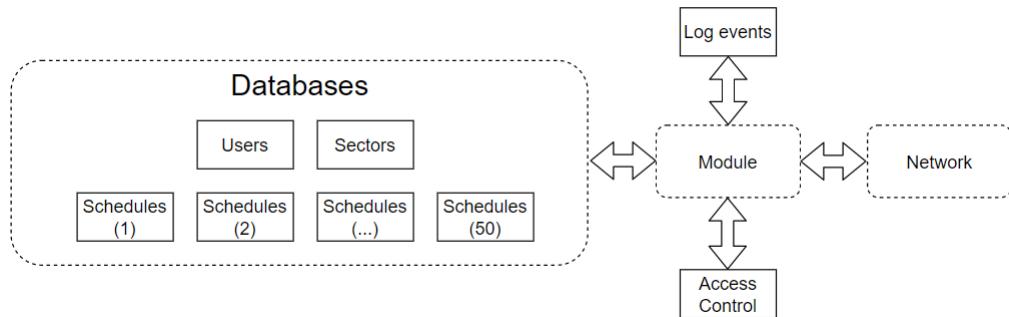


Figure 3.4: Databases architecture

# **Chapter 4**

## **System development**

In **Chapter 4**, an overview about the development of the project is presented, including the hardware part (layouts, assembly and final prototype), as well as the software part (including the some routines done by the system during and after booting, the web-server, intercommunication and the graphical interfaces).

### **4.1 Layout of components**

The components and modules used in this project are connected in order to take advantage of the maximum available pins of the ESP32 microprocessor, avoiding those which usage are not recommended.

In summary, each module is composed by:

- 1x ESP32 (DevKit) - Used for information processing and network communication;
- 2x RFID RC522 - Used for reading RFID cards (one module for entry, and another for exit);
- 1x Relay module - Used to control electrical locks and lighting;
- 1x Buzzer - Used for alarms and indications (access allowed with 1 beep, access denied with 2 beeps).
- 1x Borne/button - Used to simulate the entry of a user (can be connected to a micro-switch installed on the door);

- 1x RTC DS3231 - Used to obtain date and time offline;
- 1x Micro-SD card module (main module only) - Used to increase the available storage space.

Thus, the pinouts used for the modules and other components were based on the DevKit layout, in order to facilitate the future development of PCB (*Printed Circuit Board*), and are arranged as follows:

- Pin 01 - Pin used for DEBUGING (TX)
- Pin 02 - Development board LED (DevKit)
- Pin 03 - Pin used for DEBUGING (RX)
- Pin 04 - SS line for the micro-SD card (SPI communication)
- Pin 13 - Relay control (electric lock);
- Pin 14 - Relay control (illumination);
- Pin 18 - MISO (SPI communication)
- Pin 19 - MOSI (SPI communication)
- Pin 21 - SCK (SPI communication)
- Pin 22 - SS of RC522 01 (SPI communication)
- Pin 23 - SS of RC522 02 (SPI communication)
- Pin 25 - Input terminal (for micro-switches)
- Pin 26 - SDA (I2C Communication)
- Pin 27 - SCL (I2C Communication)
- Pin 32 - RST (for MC522 module initialization)
- Pin 33 - Buzzer

It is important to note that some of the recommended standard pins have been changed via software, as is the case of SPI and I2C communication. This change was made in order to use sequential pins to match the RC522 pinout and to avoid crossing paths in the manufacturing of the PCB. Due to the high processing power of the ESP32, all pins that supports digital output can easily operate on all protocols used in this project.

Even though there are two types of modules in this system (main and secondary),

the connections and pinouts are the same, the only differences in the hardware part are that the secondary modules do not have the micro-SD card modules nor the RTC-DS3231 module, which are exclusive to the main module.

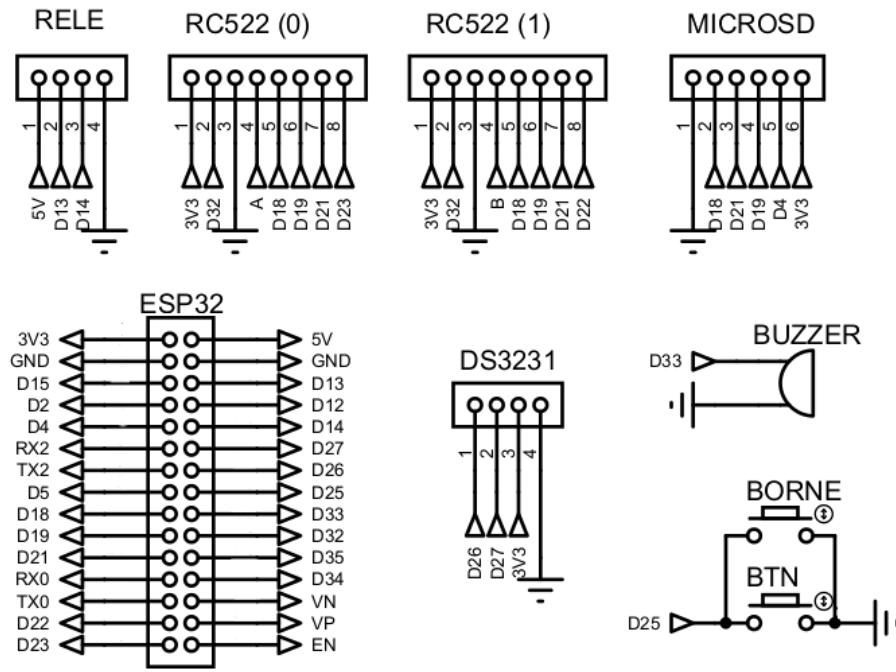


Figure 4.1: Connections between modules on the main module

In addition, the following system inputs and outputs can be configured:

- Relay control: Allows use in “default open”, “default closed” or “disabled”. It is also possible to configure a maximum timeout for the relay that controls the locks;
- Micro-switch control: Allows use in “default open”, “default closed” or “disabled”. In addition, it is possible to configure the maximum time that the micro-switch can be active before triggering the alarm;
- Buzzer: Allows "enabled" or "disabled" mode;

The power source of the project is made through the micro-USB connector of the ESP32 development board, with 5VDC. This power will be used directly in the relay module (which operates at 5V). The other components use the 3.3V voltage generated by the regulator of the development board used.

## 4.2 PCB development and prototype

The development of the project was initially made on a breadboard, with the purpose of an easy configuration and available pin tests. Bearing in mind that the ESP32 development kit used is not completely breadboard friendly, as it is not possible to use all the pins on a conventional breadboard (10 pins width), a special breadboard with 12 pins was used for the initial tests, as shown in figure 4.2.

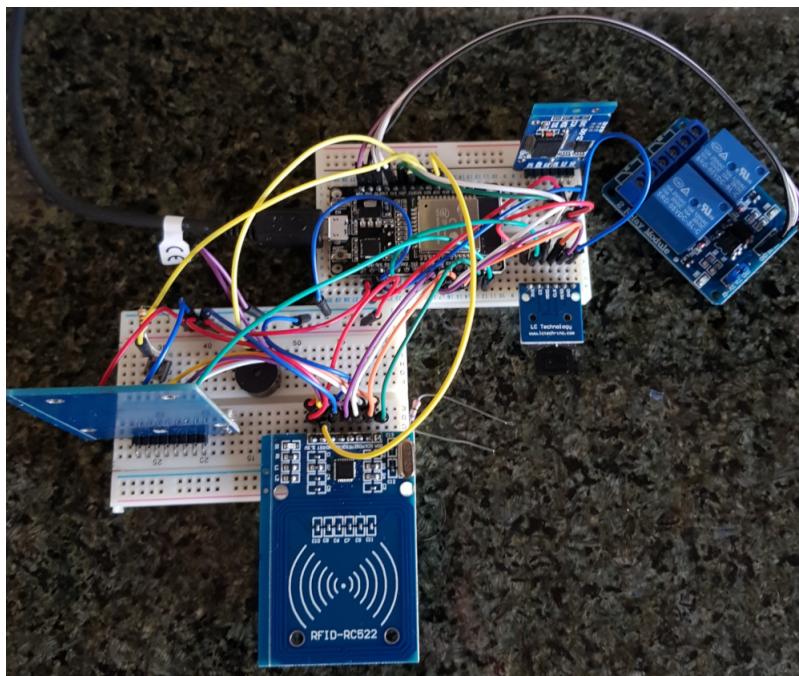


Figure 4.2: Prototype developed on breadboard

However, it was necessary to migrate to the PCB due to the devices that uses SPI communication, which began to present contact problems and possible interference, caused by the high speed communication (8MHz).

Thus, according to the arrangement of the components presented in §4.1, it was possible to develop a simple small PCB (15cm x 8.5cm). The same PCB is used in the main module and in the secondary modules.

The trails and islands were scaled in order to occupy the largest available space, without compromising the processes of welding and placement of components. In addition,

two different hole sizes were used, considering that some components have larger terminals than other, as shown in figure 4.3. In addition, grounding mesh was used on the negative terminal of the components to reduce the area of copper removed.

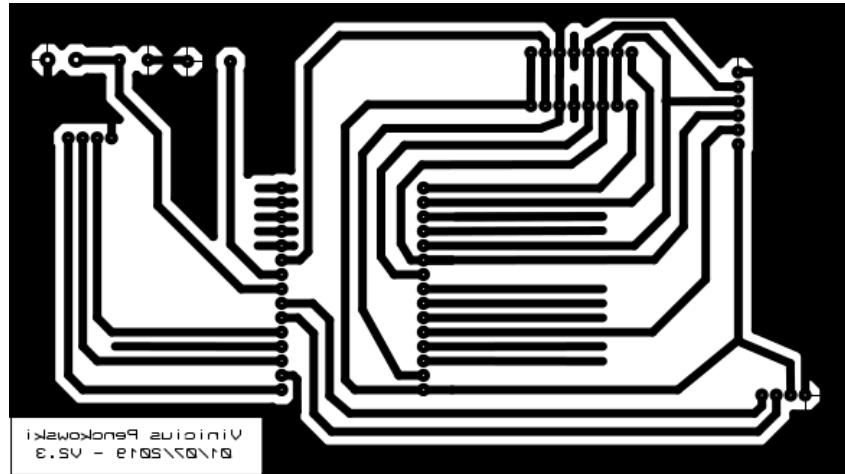


Figure 4.3: Computational version of the PCB tracks

In total, five printed circuit boards were made according to the PCB in figure 4.4. Some unused pins have tracks that are a few centimeters long to prevent the islands from being damaged during welding.

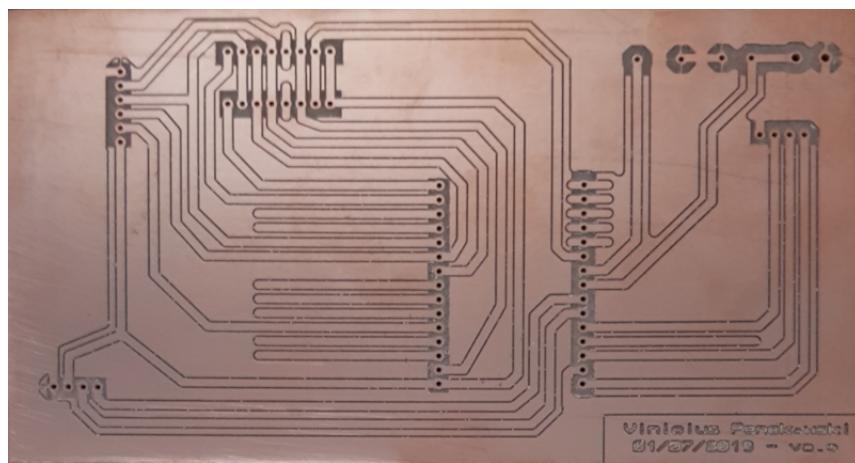


Figure 4.4: Copper part of the printed circuit board

In addition, only regions close to where the welded terminals would be inserted had the extra copper removed to avoid problems during welding.

Then, female terminals were soldered to the plate, in order to allow the removal of the most expensive components in case of failures. With the welded terminals, the components were then inserted and resulted in the final prototype shown in the figure 4.5.



Figure 4.5: Final prototype of the PCB project

It is worth mentioning that the second RC522 module is directly connected to the board. However, for real application outside a door, it could be connected using a set of male-female cables.

The secondary modules have a similar layout, with differences only in not having the DS3231 and micro SD card modules installed, as shown in figure 4.6.

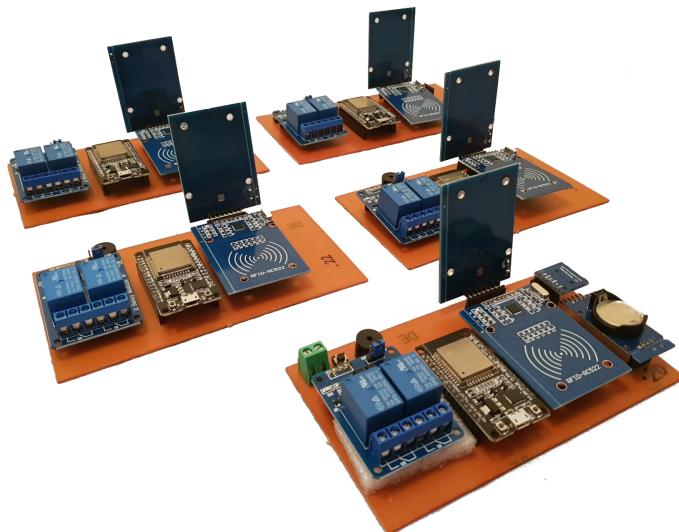


Figure 4.6: The four secondary modules and the main one

## 4.3 General operation

### 4.3.1 Initialization

During the initialization of the control panel, several actions are taken to guarantee the functioning of the system.

The first action performed initializes the ESP32 internal memory and the memory card (if it exists), in order to collect the configuration files that will be used during the system initialization.

The system then checks whether the configuration files exist. If they do not exist, files with the default settings are generated. If they exist, these settings are loaded. More information about the existing settings is described in §4.3.10.

With the settings loaded, it's then possible to define the output types for the pins and for the components, such as the RC522, relays, buzzer, micro-switch, and also initialize the DS3231 module (if it's present).

Then, the module's Wi-Fi communication is initiated, based on the loaded configurations, defining how the module should work (connect to a network, or generate a network), the network credentials, and other information such as IP, gateway, netmask and DNS.

After that, the asynchronous server is started, where more than 20 pages interconnects the graphical interface and the webserver (detailed in §4.3.3). In addition, all GUI pages and files required for HTML pages, such as scripts and databases becomes available. When all pages are initialized, the default HTTP headers of all files are instantiated.

The memory used by users is cleared and the existence of the user database is checked. If it does not exist, a file with null information is generated for each user. If it exists, the informations (user name, RFID tag identifier and expiration time) are loaded into variables in RAM to speed up the use during the RFID authentication process.

Subsequently, the system reads sector information from the system's database. If it does not exist, a file with null information is generated for each of the sectors. If it exists, the informations (sector name and IP of the module responsible for this sector)

are loaded into variables in RAM for quick use during the recording of events and inter-communications (detailed in §4.3.11).

The system then reads the schedule database for its respective sector and for the next sector (for which the module is responsible) and saves in RAM the required data.

With all the necessary information loaded from the internal memory, the system generates a 16-byte SHA1-HMAC Hash for each file, which will be used in the future to verify changes in the databases. These Hashes are saved in RAM memory and are sent via HTTP to the main module, which checks whether it is necessary to update the file (detailed in §4.3.11).

Finally, the system checks whether the connection was successful. If not, the ESP32 starts an Access Point. If successful, the date and time is collected from the internet via NTP or RTC module.

The secondary modules also checks file hashes with the main module to confirm that the databases are correct and up to date. Otherwise, the databases are updated and reloaded in the internal variables.

After initialization is completed, the ESP32 DevKit's LED is turned off until further processing is required. A simplified flowchart of the setup is shown in figure 4.7.

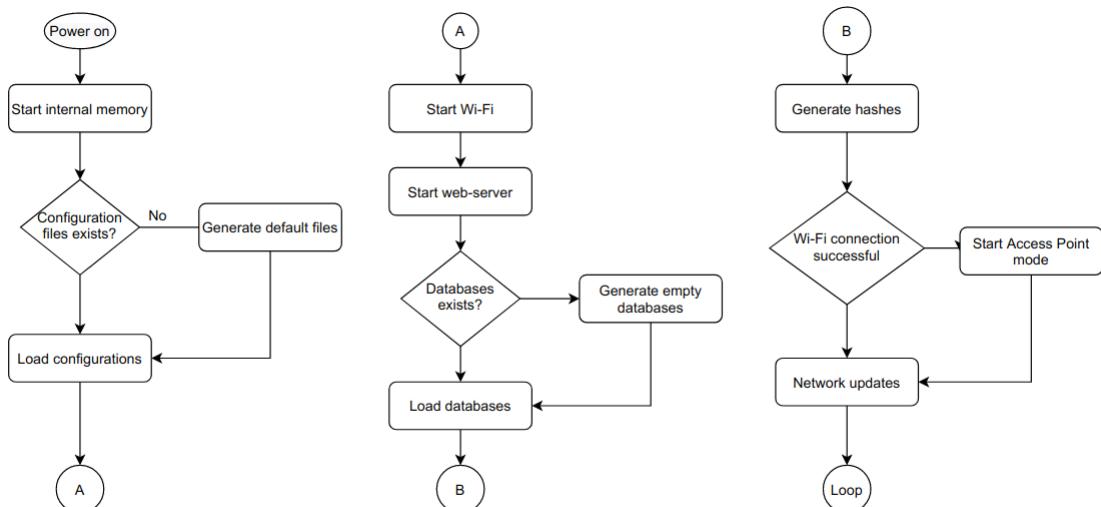


Figure 4.7: Simplified flowchart of setup function

### 4.3.2 Loop

After booting, the system is placed in loop. As that the ESP32 processor has 2 cores that can be used individually, the first one is responsible for all network and server processes, as well as functions that collects data from the HTML graphical interface.

The other functions are executed in the second core, as shown in figure 4.8. Among them, are the reading and processing the RC522 modules status. In case a tag is detected, it's identifier is read and compared with the database to check if it corresponds to any user. If positive, a function to check if the user is authorized is called, in order to allow or deny the user's entry and record the occurrence. Then, the function that controls the external components is called, in order to check the number of people in the sector for illumination control, electrical lock control and status of the door's micro-switch.

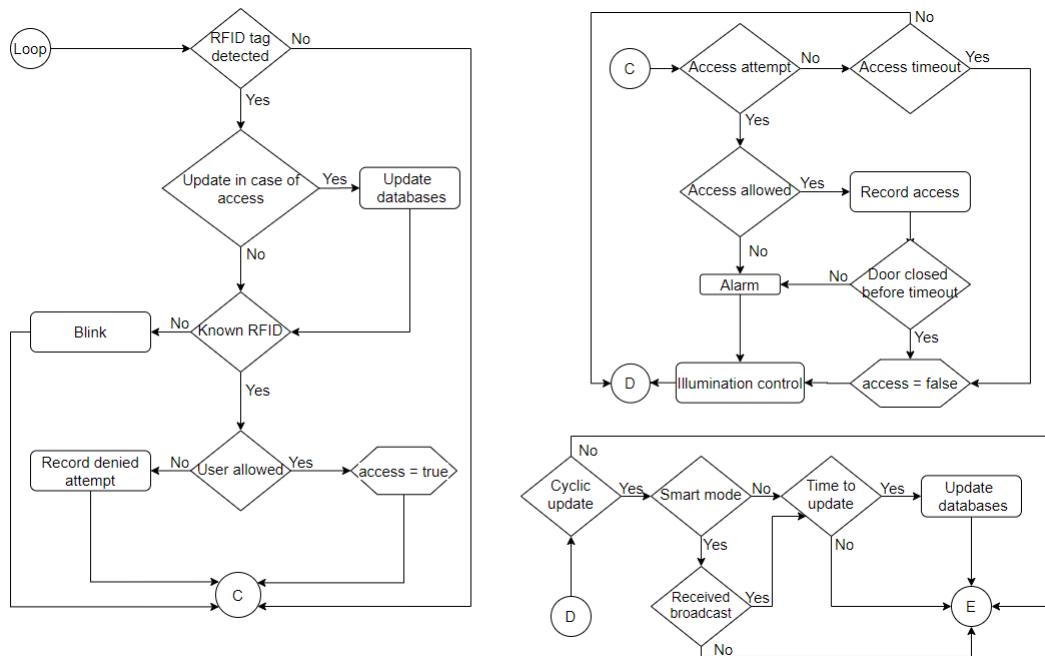


Figure 4.8: Simplified flowchart of loop function of Core 02)

Then, the update functions runs, and an optimization process runs to check if the core 01 has requested core 02 to run any intensive processing task (such as reading or editing databases or settings) in order to avoid slowing down network functionality, and to improve the performance of the server.

### 4.3.3 Server (core 01)

For functions related to providing files and data an asynchronous server is executed internally in the ESP32. This server, in addition to providing the necessary files for the graphical interfaces, also bridges the communication between the user and the ESP32, and between modules.

Some of the pages that perform this “bridge” are:

- “/adicionarUsuario” - Receives via GET request the data of the user number to be edited or created, such as name, UID and expiration date;
- “/adicionarSetor” - Receives through GET request the data of the sector number to be edited or created, the name of the sector and its IP address;
- “/adicionarHorario” - Receives through the GET request all the information for the schedules databases (such as sector, user, and allowance times);
- “/limparRegistros” - Request to delete stored records;
- “/limparUsuarios” - Request to delete user databases;
- “/limparSetores” - Request to delete sector databases;
- “/limparHorarios” - Request to delete all the schedules databases;
- “/status” - Indicates the current system status in all modules, used in the system status page, and in the sectors web-page (to indicate if the module is working) (§4.3.8);
- “/info” - Provides information about memory for the system’s home page (detailed in §4.3.5);
- “/NTP” - Provides the EPOCH time for module synchronization.
- “/backup” - Requests through a GET type request that a system backup or restoration be made with files from the micro-SD card.
- “/solicitarSetor” - Used for intercommunication between modules to define which module becomes responsible for each of the sectors (detailed in §4.3.11).
- “/registrar” - Used for intercommunication between modules for registering and accounting users by sector (detailed in §4.3.11);

- “/configRede” - Receives the network configuration data from the module via GET request (detailed in §4.3.10);
- “/configNTP” - Receives the date and time configuration data for the system via GET request (detailed in §4.3.10);
- “/configSetor” - Receives configuration data for the sector via GET request (detailed in §4.3.10);
- “/reboot” - Reboots the system.

In order to be able to access any of the system pages, it is necessary to login with an username and password via web browser. This authentication is also used by the modules to be able to communicate with the others, so the password of all modules must be the same and known by the operator. The configuration of passwords is described in §4.3.10.

In addition, all the files necessary for the graphical interface are then made available by the web-server, including all the HTML, CSS, JS files, and also the databases and configuration files in CSV. All of these files also requires authentication to be accessed.

Static files, such as HTML, CSS and JS use HTTP headers to inform to the browser that the files must be saved permanently in cache (to avoid needing to reload the same files at each request).

For most of the non-static files, such as databases, a 16-byte hash identifier is generated and sent with the file. Thus, when requesting these files, the browser (or another module) sends this hash to the server, which checks if there was any changes with the files based in this hash. In case of changes, the server responds with the HTTP 200 code (OK), and sends the new file and the new hash. If the file has not been modified, the server responds with HTTP code 304 (not modified), and the browser (or secondary module) uses the existing file in memory.

The hash identifier is also used for integrity and security checks, as it uses SHA1-HMAC with a cryptography key. This key needs to be the same at all the modules, so they can check if the file is safe to be used.

#### 4.3.4 Navigation bar

In order to facilitate navigation between the system pages, a “Navigation bar” was implemented, which is always present at the top of the screen and presents the main pages of the graphical interfaces.

It is only present in the main module, since the secondary ones have access only to the “Settings” page.

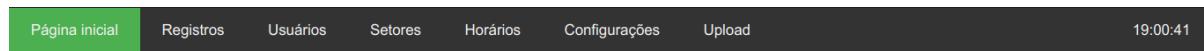


Figure 4.9: Main panel navigation bar

The navigation bar presents the main pages of the system, and keeps the currently active page in green. In the right corner of the bar, it is possible to check the time of the module, in order to check if it's correct, and if needed, it can be changed in the settings tab (§4.3.10).

In addition, a system of cookies is used to register the last page visited, so when reloading the page, the same tab is loaded. This is done using a variable called num is used which indicates the number of the last page accessed (from 0 to 6, corresponding to the pages indicated in the navigation bar).

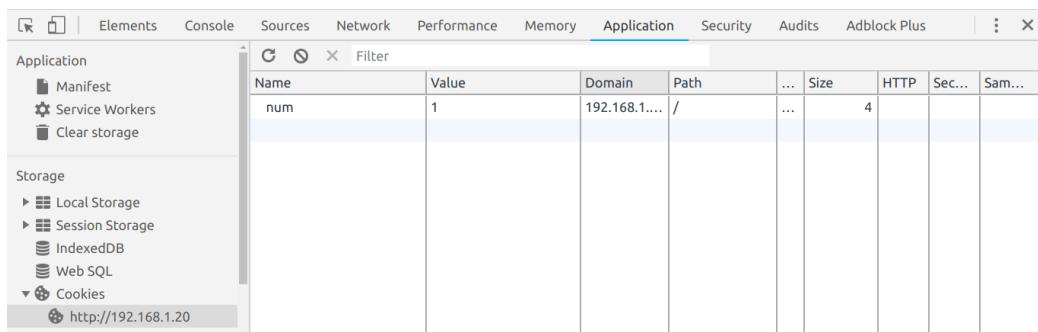


Figure 4.10: Cookie saved to remember the navigation bar of the last page accessed

The Upload page is available in this project for direct upload of files to the main server. In the case of commercial use of the system, this page must be disabled to avoid security problems and improper files.

### 4.3.5 Home

It is the first page of the panel, and it is the default startup page after the system configuration. The first information on the page is the title (“Bem vindo a central de controle de acesso!”), and the IP of the module.

Then, some important information is presented, such as the amount of internal memory available (for records and times), the amount of memory available on the external memory card, if installed, and the available heap memory, the which is responsible for keeping the server stable.



Figure 4.11: Home of the panel without using an external memory card.

On this page it is also possible to request that the system be restarted via software.

All information passed through this page is also available through the “/info” location, which is the data source for this page. These data are arranged so that the first column indicates the available memory and the second the total memory, and the lines respectively represent the internal, external and heap memories, as shown in figure 4.12.

Name	x	Headers	Preview	Response	Cookies	Timing
home.htm	1	843 2820				
home.js	2	0 0				
geral.css	3	171076 264436				
info						

4 requests | 8.6 KB transferred

Line 1, Column 1

Figure 4.12: Information received from the server to the home page.

#### 4.3.6 Records and analysis

The second page of the system is the Records page, which contains the following system logs:

- Access attempt allowed (Entry/Exit);
- Access attempt denied;
- Alarm triggered;
- Alarm disarmed;
- System restart.

The page has the title “Registros do Sistema”, followed by three buttons, which allow, respectively, the records to be downloaded in the .csv format, the records to be deleted, and the records to be analyzed.

Then, there is the possibility of filtering the records based on the type of record, user and sector, as well as filtering by dates of occurrence. If any filtering is requested, the table is updated in real time. It's also possible to select the order of the events by crescent or decrescent order.

Tipo	Nome	Setor	Data	Entrada	Saída	Duração
Permitido*	Usuário 1	Setor 1	16/07/2019	-	19:37:56	-
Permitido	Usuário 2	Setor 2	16/07/2019	19:37:56	19:38:37	00:00:41
Negado	Usuário 2	Setor 4	16/07/2019	19:37:59	-	-
Permitido	Usuário 1	Setor 3	16/07/2019	19:38:37	19:39:54	00:01:17
Permitido	Usuário 3	Setor 3	16/07/2019	19:38:37	19:44:01	00:05:24
Permitido	Usuário 1	Setor 2	16/07/2019	19:39:55	19:40:22	00:00:27
Alarme*	-	Setor 2	16/07/2019	19:39:57	-	-
Permitido	Usuário 1	Setor 4	16/07/2019	19:40:22	19:40:42	00:00:20
Permitido	Usuário 5	Setor 4	16/07/2019	19:40:25	19:40:45	00:00:20
Permitido	Usuário 1	Setor 5	16/07/2019	19:40:45	19:41:52	00:01:07
Permitido	Usuário 2	Setor 2	16/07/2019	19:43:01	19:48:03	00:05:02

Figure 4.13: Records page

Then comes the spreadsheet, which join the correspondent events, so that for each

entry, the respective exit is found and both are grouped on the same line, then the duration of the event is calculated. In addition, the spreadsheet shows actions that require attention in different colors (such as alarms and denied accesses).

When selecting the option “Download records”, the file obtained shows each event saved separately. This way, the permanence calculation doesn’t appear. This choice was due to the processing limitations of ESP32 with large files.

The option “Detailed analysis” opens a floating window on the page and allows the selection of a user to be analyzed, and the range of days. With that, a table that indicates the total permanence by sector, and the number of occurrences is presented. In addition, alarms, incomplete accesses (entries without exits, or exits without respective entries), and improper access attempts are also indicated.

With this data, the top 10 sectors where the selected user had the longest permanences are used to generate a graph in order to provide a better visualization of that user’s behavior. The generated graph also shows the percentage values for each of the sectors.

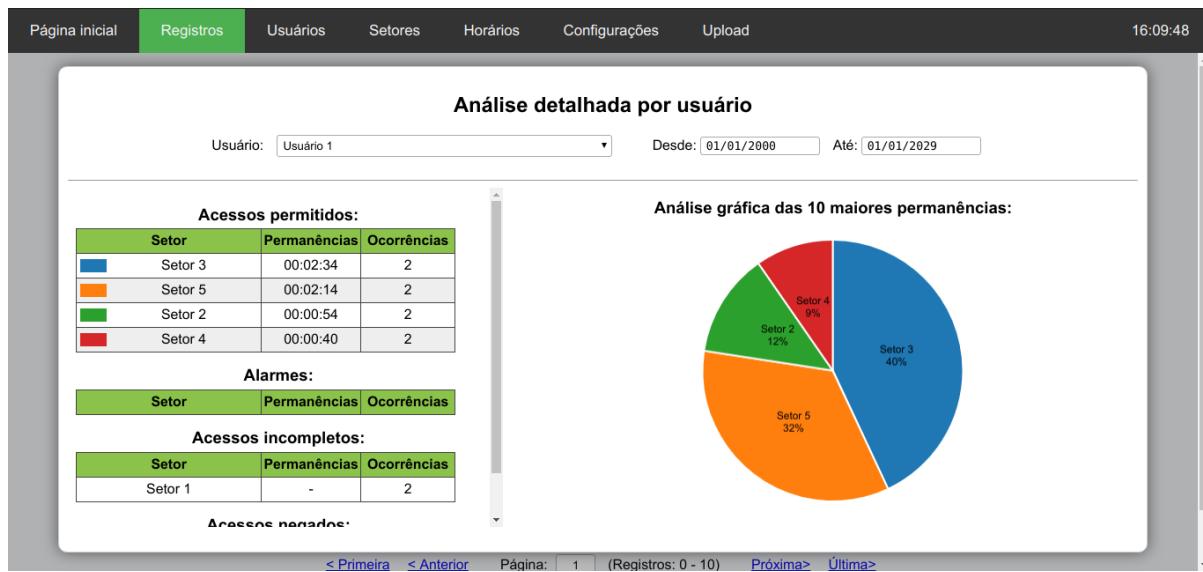


Figure 4.14: Record analysis page

Due to the fact that the module is capable of displaying up to 1,500 records, a pagination was implemented, where it is possible to choose between 10, 25, 50, 100 or 200 records per page, with the objective of not overloading the graphical interface, as shown

in the image 4.15, which is located at the bottom of the page.

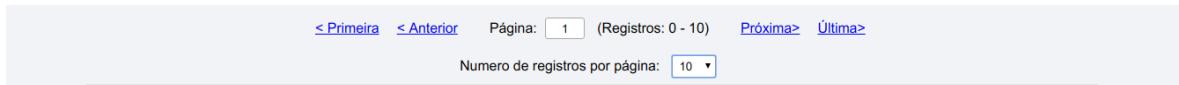


Figure 4.15: Records pagination

For the operation of this page, the file “Registro.csv” is loaded, as shown in the image 4.16. This file is then divided into a data matrix, which is analyzed for the purpose of correcting formatting, grouping entries and exits, calculating permanences and coloring each line accordingly.

A special hash of the records file is requested once every 2 seconds, and if the hash changes, the records are reloaded. This way, if any event occurs, the page automatically refreshes its data, and new events are displayed.

Name	Headers	Preview	Response	Cookies	Timing
registro.h...					
geral.css					
Chart.js					
chartjs-pl...					
chartjs-pl...					
registro.js					
registro.csv					
usuarios.csv					
setores.csv					
	1   Tipo Nome Setor Data Entrada Saida Duração - 2   Permitido Usuário 1 Setor 1 16/07/2019 - 19:37:56 - 3   Permitido Usuário 2 Setor 2 16/07/2019 19:37:56 - - 4   Negado Usuário 2 Setor 4 16/07/2019 19:37:59 - - 5   Permitido Usuário 2 Setor 2 16/07/2019 - 19:38:37 - 6   Permitido Usuário 1 Setor 3 16/07/2019 19:38:37 - - 7   Permitido Usuário 3 Setor 3 16/07/2019 19:38:37 - - 8   Permitido Usuário 1 Setor 3 16/07/2019 - 19:39:54 - 9   Permitido Usuário 1 Setor 2 16/07/2019 19:39:55 - - 10   Alarme - Setor 2 16/07/2019 19:39:57 - - 11   Permitido Usuário 1 Setor 2 16/07/2019 - 19:40:22 - 12   Permitido Usuário 1 Setor 4 16/07/2019 19:40:22 - - 13   Permitido Usuário 5 Setor 4 16/07/2019 19:40:25 - - 14   Permitido Usuário 1 Setor 4 16/07/2019 - 19:40:42 - 15   Permitido Usuário 5 Setor 4 16/07/2019 - 19:40:45 - 16   Permitido Usuário 1 Setor 5 16/07/2019 19:40:45 - - 17   Permitido Usuário 1 Setor 5 16/07/2019 - 19:41:52 - 18   Permitido Usuário 2 Setor 2 16/07/2019 19:43:01 - - 19   Negado Usuário 2 Setor 4 16/07/2019 19:43:28 - - 20   Permitido Usuário 3 Setor 3 16/07/2019 - 19:44:01 - 21   Permitido Usuário 2 Setor 2 16/07/2019 - 19:48:03 - -				
9 requests   ...	Line 1, Column 1				

Figure 4.16: Record analysis interface

For the development of this page, more than 500 lines of code were produced, including HTML, CSS and mainly JavaScript.

### 4.3.7 Users

The third page of the system is the users web-page, in which it is possible to add, edit and remove users from the users database.

This page is composed of the title “Usuários Registrados”, followed by three buttons, which respectively allow: Add a new user, download the file from the user database or delete all users.

Then, it is possible to perform a search filter based on the user code, name or the corresponding RFID card identifier. The spreadsheet is updated in real time if any filtering is requested.

If an RFID tag is detected at the main module while this page is open, the RFID identifier is automatically inserted in the search field and the user is located. This feature is ideal for finding out to which user a particular tag belongs.

Código	Nome	Cartão	Expira	Editar	Remover
1	Usuário 1	EB 36 95 35	2029-01-10	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
2	Usuário 2	50 A0 B3 1E	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
3	Usuário 3	52 B0 B4 2F	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
4	Usuário 4	59 1D 38 5A	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
5	Usuário 5	AA BB CC EE	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
6	Usuário 6	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
7	Usuário 7	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
8	Usuário 8	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
9	Usuário 9	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
10	Usuário 10	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
11	Usuário 11	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
12	Usuário 12	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>
13	Usuário 13	AA BB CC DD	2029-01-01	<input type="button" value="Editar"/>	<input type="button" value="Remover"/>

Figure 4.17: Users page

The length of the user names is limited to 30 characters due to limitations of the internal memory available in RAM of the modules.

To add new users, the “Adicionar Usuário” button can be used, which results in the popup shown in figure 4.18 to select the number of the user, followed by the 4.19 window.

A new user can be also added by passing a card that has not yet been registered at the main module RFID reader.

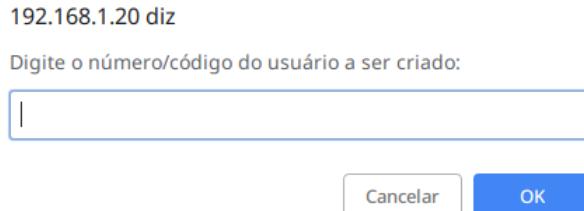


Figure 4.18: Add users window

In each line of the table, there is the option to edit individually or remove the corresponding user. When selecting the option to edit, a floating window is opened on the screen (figure 4.19), allowing information such as username, RFID tag identifier or expiration date to be changed.

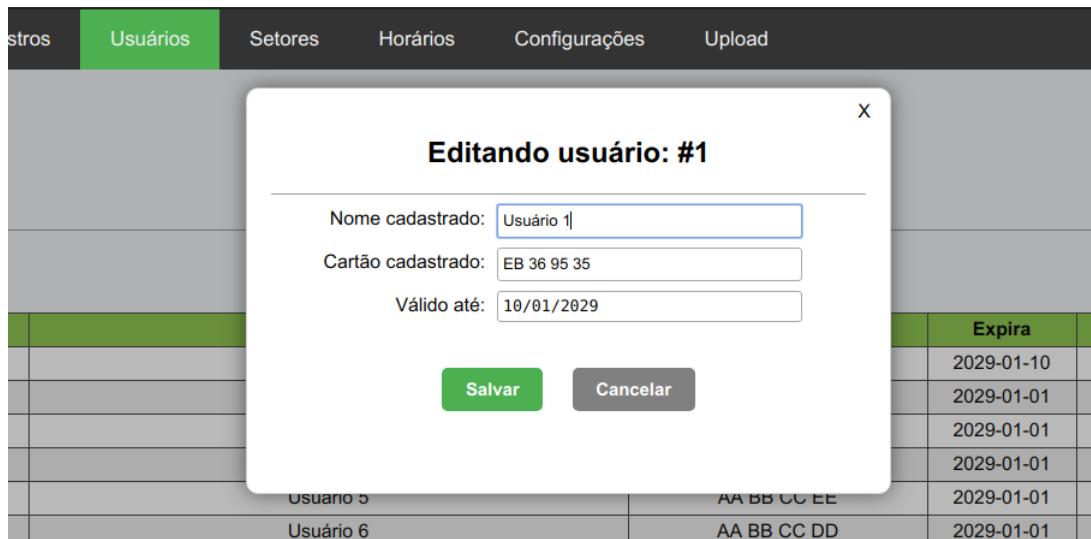


Figure 4.19: User edit window

When saving any modifications, the system verifies if there were no repetitions in names or tags, and if the information is valid. If so, this new data is sent to the server that requests changes in the ESP32 RAM variables, and are than saved in the database file, allowing the page to reload and show the new values in the spreadsheet. If any of

the information is not valid, a warning page is displayed with the message “Error: The number must be between 1 and 499, the names and even the RFIDs cannot be repeated”.

The data used on the users page comes from the file “users.csv”, which can be downloaded through the button “Baixar Usuários”. This file is loaded on the page (figure 4.20), and its data is divided into a matrix. This matrix is then analyzed in JavaScript and the spreadsheet is generated, splitting the data in 5 rows, indicating the code of the user, the name of the user, the RFID tag, expiration date and buttons for editing and removing an user from the databases.

The expiration date is given in seconds since the year 2000 (modified EPOCH system), so this information must be converted before being displayed. In case the expiration date of a user has expired, its information is kept, however the line of the table is colored in red.

Name	X	Headers	Preview	Response	Cookies	Timing
192.168.1...	1	0	0			
geral.css	2	1	Usuário 1	EB 36 95 35 916012800		
home.htm	3	2	Usuário 2	50 A0 B3 1E 915235200		
usuarios.h.	4	3	Usuário 3	52 B0 B4 2F 915235200		
geral.css	5	4	Usuário 4	59 1D 38 5A 915235200		
ultimoC	6	5	Usuário 5	AA BB CC EE 915235200		
usuarios.cs	7	6	Usuário 6	AA BB CC DD 915242400		
ultimoC	8	7	Usuário 7	AA BB CC DD 915242400		
ultimoC	9	8	Usuário 8	AA BB CC DD 915242400		
ultimoC	10	9	Usuário 9	AA BB CC DD 915242400		
ultimoC	11	10	Usuário 10	AA BB CC DD 915242400		
ultimoC	12	11	Usuário 11	AA BB CC DD 915242400		
ultimoC	13	12	Usuário 12	AA BB CC DD 915242400		
NTP	14	13	Usuário 13	AA BB CC DD 915242400		
ultimoC	15	14	Usuário 14	AA BB CC DD 915242400		
ultimoC	16	15	Usuário 15	AA BB CC DD 915242400		
ultimoC	17	16	Usuário 16	AA BB CC DD 915242400		
ultimoC	18	17	Usuário 17	AA BB CC DD 915242400		
ultimoC	19	18	Usuário 18	AA BB CC DD 915242400		
ultimoC	20	19	Usuário 19	AA BB CC DD 915242400		
ultimoC	21	20	Usuário 20	AA BB CC DD 915242400		
ultimoC	22	21	Usuário 21	AA BB CC DD 915242400		
ultimoC	23	22	Usuário 22	AA BB CC DD 915242400		
61 requests ...			Line 1, Column 1			

Figure 4.20: File that provides information for the users page

The file “users.csv” is part of the files whose hash is generated at startup. Thus, it is sent via headers during transmission as a ETAG of the file. Thus, if there was no change

in the file, the browser loads the same from the cache itself, thus consuming less module processing and speeding up the page loading.

#### 4.3.8 Sectors

The fourth page of the system is where it is possible to get and edit the sectors database. Right after the title “Setores Registrados”, there are three buttons, which respectively allow the addition of new sectors (“Adicionar setores”), download the sectors database (“Baixar setores”) and delete all sectors (“Limpar setores”).

Then, a filtering label allows the search for a specific sector by its registered name. The spreadsheet is updated in real time in case any filtering is requested.

The screenshot shows a web application interface for managing sectors. At the top, there is a navigation bar with tabs: Página inicial, Registros, Usuários, **Setores** (which is highlighted in green), Horários, Configurações, and Upload. To the right of the tabs, the time is displayed as 20:14:54. Below the navigation bar, the main content area has a title "Setores cadastrados". Underneath the title are three buttons: "Adicionar setores" (blue), "Baixar setores" (blue), and "Limpar setores" (red). A search input field labeled "Filtros de busca: Busca por setor..." is also present. The main content is a table with the following data:

Código	Setores	IP da central	Status	Editar
1	Setor 1			[Edit] [Remover]
2	Setor 2	<a href="#">192.168.1.20</a>	[Green]	[Edit] [Remover]
3	Setor 3	<a href="#">192.168.1.21</a>	[Red]	[Edit] [Remover]
4	Setor 4	<a href="#">192.168.1.22</a>	[Green]	[Edit] [Remover]
5	Setor 5	<a href="#">192.168.1.23</a>	[Green]	[Edit] [Remover]
6	Setor 6	<a href="#">192.168.1.24</a>	[Green]	[Edit] [Remover]
7	Setor 7			[Edit] [Remover]
8	Setor 8			[Edit] [Remover]
9	Setor 9			[Edit] [Remover]
10	Setor 10			[Edit] [Remover]
11	Setor 11			[Edit] [Remover]
12	Setor 12			[Edit] [Remover]
13	Setor 13			[Edit] [Remover]
14	Setor 14			[Edit] [Remover]

Figure 4.21: Sector page

The spreadsheet contains five rows. The first indicates the sector code. The second indicates the name of the sector, which can be changed on the page itself through the “Editar” button. The third indicates the IP address of the module responsible for this sector, which can be easily accessed by clicking on it. The fourth row indicates the status of the module, which can be: green (online), red (offline), or gray (no module registered in this sector). The last row allows the editing and removal of a sector.

The status indicators are updated sequentially, taking between 100ms to 1000ms each, depending on the status of the module. In this way, it is possible to monitor in real time the status of all registered modules. To check if the modules are online, a request is sent via HTTP with a GET parameter to wait for response. If the response is correct, the indicator in the fourth row turns green, otherwise it turns red.

When requesting the addition of a new sector, the window of the figure 4.22, so that the sector number is inserted. The rest of the sector creation process becomes the same as the edition process (figure 4.23), it is only necessary to insert the sector name and save, so that the sector information can be changed in the module database .



Figure 4.22: Sector add interface

When clicking on the option to edit a sector, the floating image window 4.23 appears, allowing the sector name to be changed.



Figure 4.23: Sector edit interface

The size of the sector names is limited to 30 characters due to limitations of the internal RAM memory available in the modules.

When requesting to change the name of a sector, the information is checked to verify that an existing name has not been used, or that the sector value does not exceed the maximum number of 49 sectors. If everything is right, the database is changed. Otherwise, the following warning message is displayed: “Error: The number must be between 1 and 49, and the sector names cannot be repeated.”

The data used in the sectors page comes from the file “sectors.csv”, which can be downloaded through the button “Download Sectors”. This file is loaded on the page (figure 4.24), and its data is divided into a matrix. This matrix is then analyzed and the page spreadsheet is then generated.

Name		Headers	Preview	Response	Cookies	Timing
setores.htm	1	0				
NTP	2	1 Setor 1				
geral.css	3	2 Setor 2 192.168.1.20				
sectores.csv	4	3 Setor 3 192.168.1.21				
	5	4 Setor 4 192.168.1.22				
	6	5 Setor 5 192.168.1.23				
	7	6 Setor 6 192.168.1.24				
	8	7 Setor 7				
	9	8 Setor 8				
	10	9 Setor 9				
	11	10 Setor 10				
	12	11 Setor 11				
	13	12 Setor 12				
	14	13 Setor 13				
	15	14 Setor 14				
	16	15 Setor 15				
	17	16 Setor 16				
	18	17 Setor 17				
	19	18 Setor 18				
	20	19 Setor 19				
	21	20 Setor 20				
	22	21 Setor 21				
	23	22 Setor 22				
4 requests   ...		Line 1, Column 1				

Figure 4.24: File that provides information for the sectors page

The file “sectors.csv” is also part of the files whose hash is generated at startup. Thus, it is sent via headers during transmission as a ETAG of the file. If there was no change in the file, the browser loads the same from the cache itself, thus consuming less module processing and speeding up the loading of the sectors page.

#### 4.3.9 Schedules

The fifth page of the system is the interface for adding, editing and removing schedules. Below the title “Horários Permitidos” there are two buttons, the first being “Adicionar horários” (used to add new schedules) and the second “Limpar horários” (used to clear all the schedules).

Then three filtering options are available through selectors. The first one allows to filter by sector. The second selector makes it possible to filter by user. The third allows to choose the day of the weekday to be analyzed. The spreadsheet is updated in real time in case any filters are applied. Multiple filters can be used at the same time.

In order for these selectors to be generated, the page needs to load the users and sectors databases, thus being the system page that loads the largest amount of files for full operation.

The spreadsheet with the information is generated, and shows all the permission times of all modules (up to 49 schedules databases files loaded and decoded).

Setor	Nome do usuário	Horário inicial	Horário final	Editar
Setor 1	Usuário 2	Liberado	Liberado	[Editar] [Remover]
Setor 2	Usuário 3	Liberado	Liberado	[Editar] [Remover]
Setor 3	Usuário 1	Liberado	Liberado	[Editar] [Remover]
Setor 4	Usuário 2	Liberado	Liberado	[Editar] [Remover]
Setor 4	Usuário 3	Liberado	Liberado	[Editar] [Remover]
Setor 5	Todos os usuários	Liberado	Liberado	[Editar] [Remover]
Setor 1	Usuário 1	Seg - 10:20	Seg - 11:40	[Editar] [Remover]
Setor 1	Usuário 3	Seg - 10:20	Seg - 11:00	[Editar] [Remover]
Setor 1	Usuário 4	Seg - 11:00	Qua - 15:00	[Editar] [Remover]
Setor 2	Usuário 1	Ter - 11:00	Sáb - 22:30	[Editar] [Remover]
Setor 3	Usuário 10	Ter - 11:30	Qui - 00:00	[Editar] [Remover]
Setor 4	Usuário 3	Ter - 12:00	Sáb - 23:59	[Editar] [Remover]

Figure 4.25: Schedules page

The spreadsheet consists of five rows, four with information, which respectively indicate the time sector, the user, the start time and the end time of the permission. The fifth

row allows easy editing and deletion of a specific time. The system only allows permissions that the starting time is earlier than the end time compared to Sunday 00:00. This happens because the schedules happens on a weekly basis, resetting on Saturday 23:59.

Due to the possibility of having up to 24.500 hours registered in total (500 schedules in each of the 49 sectors), a paging system similar to the one existing on the records page was implemented, as shown in figure 4.26.

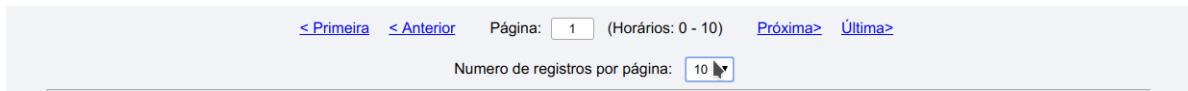


Figure 4.26: Schedules pagination

Due to the fact that several database files must be loaded, a loading interface was implemented for aesthetic reasons, which indicates which file is being loaded, as shown in figure 4.27



Figure 4.27: Loading interface for schedules page

All database files (times, users and sectors) use ETAG based on the 16-byte hash, to avoid overloading the ESP32 server. In addition, schedules are requested synchronously without blocking the graphical interface. This choice was due to limitations of the ESP32 in processing multiple requests for files in a short period of time.

When selecting the option of editing or creating a schedule, the floating window of the figure 4.28 is displayed. If the schedules are kept blank, the system considers this schedule to be of the "Total Allowance" type, so any time turns into access allowed for that user and sector. If the hour fields are filled, the authorization only happens within the chosen time range for that sector and user.

To change sectors or users, selectors are used in order to only allow values within the database of sectors and users to be used, avoiding incompatibilities and errors on the page or in internal processing.

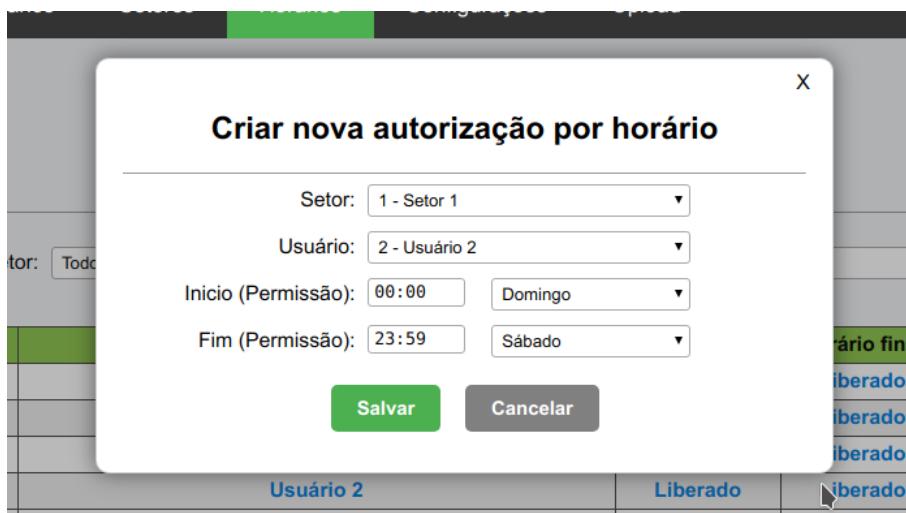


Figure 4.28: Edit and add schedules

The data used for the spreadsheet on the page comes from the time databases of each sector. An example of sector 01 schedules is shown in the figure 4.29. In this case, it is possible to note that the lines have a fixed size (24 bytes), so that it is possible to edit the file without having to completely rewrite it, speeding up the process of editing schedules.

In addition, the fields indicate the users and sectors numerically. To be displayed according to the figure 4.25 it is necessary an additional processing step, which converts the numbers of users into their names.

The number of the file represents the number of the sector. Within this file, three rows are used, the first indicating the permission user number, the second row indicating the starting time of the permission and the third row indicating the end time of the

permission. These times are represented in minutes since Sunday 00:00, as a way to standardize the file size.

Name		Headers	Preview	Response	»
horarios.htm			1 002 00000	10079	-0001
geral.css			2 001 02060	02140	-0001
horarios.js			3 004 02100	05220	-0001
setores.csv			4 003 02060	02100	-0001
favicon.ico			5		
usuarios.csv					
H1.csv					
H2.csv					
H3.csv					
H4.csv					
H5.csv					
H6.csv					
NTP					
59 requests   69.5 KB transferred   Line 1, Column 1					

Figure 4.29: Database format for schedules page

Thus, whenever a user requests access to a module, the system calculates the current time in minutes since 00:00 on Sunday and compares it with the values of the databases. Knowing that the number of users does not exceed 3 digits, and that with this form of time representation, the maximum value is 10079 (equivalent to 23:59 on Saturday), it was possible to make this database the only one of the system with a number of characters per line fixed without wasting too much storage.

Due to the large number of databases on this page, it is not possible to download all of them simultaneously. However, it is possible to download them individually by accessing the file's address. It is also possible to backup of all the schedules on a micro-SD card using the settings page, at tab “Arquivos” (detailed in §4.3.10).

Due to the high processing power required for the analysis of all the multiple databases, more than 500 lines of code were developed in JavaScript, C and C ++, making it one of the most complex functionalities to be implemented.

### 4.3.10 Settings

The sixth page of the system centralizes the settings and configurations of the system. This page is also the only one available in the secondary modules.

The page consists of the title, followed by the IP address of the module that is being configured. Then, an internal navigation bar is used to separate and organize all existing configurations, as shown in figure 4.30. The existing tabs contains the settings related to: Network, Sectors, Date and time, Files (backup and restore) and Administrative access.



Figure 4.30: Settings navigation bar

#### 4.3.10.1 Network Settings

The first settings tab refers to the network settings of the module (figure 4.31). In this tab, the following information can be changed:

- Modo do Wi-Fi - In this selector, it is possible to configure the ESP32 to connect to a Wi-Fi network, or to generate a Wi-Fi network;
- Nome da rede - SSID (*Service Set Identifier*) of the generated or connected network;
- Senha da rede - Password of the generated or connected network;
- IP da central - Static IP used by the module on the network;
- Máscara de rede - Netmask of the network;
- Gateway da rede - Gateway of the connected network;
- DNS da rede - DNS that should be used for internet access;
- IP da central principal - IP of the main module. If the module is the main one, this field is disabled;
- Alta velocidade - ESP32 high speed mode, which speeds up the response time of requests, but can cause overheating.

In addition to the settings, the page features two buttons, one for saving changes and one for canceling changes. In case of saving, the panel is restarted with the new settings.

Otherwise, the page is reloaded with the previous values.

The screenshot shows a web-based configuration interface for a device. At the top, there is a navigation bar with tabs: Página inicial, Registros, Usuários, Setores, Horários, Configurações (which is highlighted in green), and Upload. To the right of the navigation bar is the timestamp 14:51:45. Below the navigation bar, the title "Configurações gerais" is displayed, followed by the IP address "(192.168.1.20)". A sub-navigation bar below the title has tabs: Rede (highlighted in blue), Setores, Data e hora, Arquivos, and Avançado. The main content area contains several input fields for network settings:

- Modo do Wi-Fi: Conectar ao Wifi
- Nome da rede: Nome\_da\_rede
- Senha da rede: ..... (redacted)
- IP da central: 192.168.1.20
- Máscara de rede: 255.255.255.0
- Gateway da rede: 192.168.1.1
- DNS da rede: 8.8.8.8
- IP da central principal: (empty)
- Alta velocidade: Ativado

At the bottom of the form are two buttons: "Salvar" (Save) and "Cancelar" (Cancel).

Figure 4.31: Network settings page

All information presented on this page is provided by a configuration file, as shown in figure 4.32, which indicates each of the configuration options respectively. In the case of selectors, the number indicates the position of the selector.

The screenshot shows a browser's developer tools Network tab. The table lists various files and their details:

Name	X	Headers	Preview	Response	»
config.htm				1 0	
geral.css				2 Nome_da_rede	
config.js				3 Senha_da_rede	
configRede.csv				4 192.168.1.20	
configNTP.csv				5 255.255.255.0	
setores.csv				6 192.168.1.1	
configADM.csv				7 8.8.8.8	
configSetor.csv				8	
				9 1	
				10	

At the bottom of the table, it says "8 requests | 24.9 KB transferred" and "Line 1, Column 1".

Figure 4.32: Network database file

#### 4.3.10.2 Sector settings

The second settings tab refers to the sector settings whose module is responsible, as shown in the figure 4.33. In it, the following information can be changed:

- Registrar evento ao sair - Allows exits from the sector to also be registered as entries in the next sector.
- Setor ao entrar - Sector whose center is responsible. It is only possible for one module to be responsible for each of the sectors. The selection is made based on a selector generated with the sectors database.
- Setor ao sair - Sector whose entry is registered on leaving the sector configuring previously. The selection is made based on a selector generated with the sectors database. If the option “Log event on exit” is disabled, this selector is also inactive.
- Controle de fecho - Allows the use of electrical locks on sector doors through the relay. It is possible to disable the relay, or to enable it as default high or default low.
- Controle de iluminação - Allows the control of lights of the sector through the relay. It is possible to disable the relay, or to enable it as default high or default low.
- Sensor de entrada - Allows the use of micro-switches to detect movements between sectors. It is possible to disable it, or enable it as a default high or default low.
- Tempo limite para entrada - Defines the maximum time, in seconds, between the card swipe and the user entering the sector. After this time, the access is blocked again.
- Tempo limite no sensor - Defines the maximum time, in seconds, that a user can remain in the sensor (of the door) before triggering the alarm (in order to prevent the door from being kept open for longer than necessary)
- Efeitos sonoros - Allows the sound effects of the buzzer to be activated or deactivated according to the needs of the sector.
- Armazenamento - Defines the storage location of the module.

In addition to the settings, the page also has two buttons, one for saving changes

and one for canceling changes. When saving, changes are implemented and the primary module is also notified of the changes.

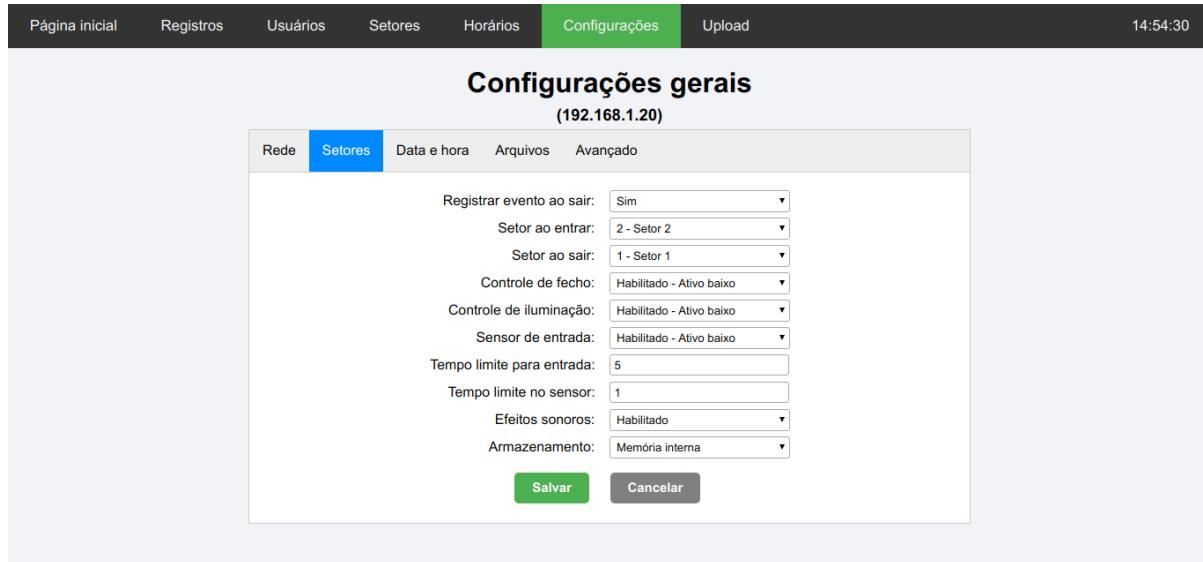


Figure 4.33: Sector settings page

All information presented on this page is provided by a configuration file, as shown in figure 4.34, which indicates each of the configuration options respectively. In the case of selectors, the number indicates the position of the selector.

Name	X	Headers	Preview	Response	>>
config.htm	1			1	
geral.css	2			2	
config.js	3			1	
configRede.csv	4			1	
configNTP.csv	5			1	
setores.csv	6			1	
configADM.csv	7			5	
configSetor.csv	8			1	
	9			1	
	10			0	
	11				

8 requests | 24.9 KB transferred Line 1, Column 1

Figure 4.34: Sector database file

#### 4.3.10.3 Date and time settings

The third settings tab refers to the date and time settings, as shown in figure 4.35. In it, the following information is requested:

- Usar NTP - Allows the use of NTP (clock synchronization protocol over the network). For this feature to work correctly, the module needs an internet connection, or a local NTP server on the network.
- Servidor NTP - Indicates the address of the NTP server, either local or online. This option can only be configured if the “Usar NTP” setting is active.
- Fuso Horário - Represents the time zone in relation to the GMT (*Greenwich Mean Time*) time. This option can only be configured if the “Usar NTP” setting is active.
- Horário de verão - Selector that allows enabling or disabling daylight saving time for the module. This option can only be configured if the “Usar NTP” setting is active.
- Inserir data - Allows manual insertion of the date in the format DD/MM/YYYY. This option can only be configured if the “Usar NTP” setting is inactive.
- Inserir hora - Allows manual entry of time in HH:MM:SS format. This option can only be configured if the “Usar NTP” setting is inactive.

For the use of NTP over the internet, it is necessary to configure a valid DNS in the network tab. In case the NTP is inactive, the main module uses the RTC-DS3231 module to maintain the time, and the secondary modules access the “/NTP” page of the main to collect and sync the time.

The modules can be configured in different ways. However, the records are made based on the time of the module that the event happened, therefore, it is recommended that all use the same parameters.

In addition to the settings, the page also has two buttons, one for saving changes and one for canceling changes. When saving, changes are implemented without requiring a restart. If the “Usar NTP” setting is disabled, a new button appears, allowing the module to collect the time of the browser that is accessing the page, in order to synchronize with

the device.



Figure 4.35: Date and time settings page

All information presented on this page is provided by a configuration file, as shown in figure 4.36, which indicates each of the configuration options respectively. In the case of selectors, the number indicates the position of the selector.

Name	X Headers Preview Response >>
config.htm	1 1
geral.css	2 pool.ntp.org
config.js	3 -3
configRede.csv	4 0
configNTP.csv	5
setores.csv	
configADM.csv	
configSetor.csv	

8 requests | 24.9 KB transferred Line 1, Column 1

Figure 4.36: Date and time database file

#### 4.3.10.4 Files

The fourth settings tab allows backups and restores of configurations and databases in the system, as shown in the figure 4.37.



Figure 4.37: File backup and restore page

The following functions are available:

- Download: Records, users and sectors;
- Backup and restore;
- Delete: Records, users, sectors and schedules.

The Backup and restore functions require a micro-SD memory card connected to the module to be performed. During the backup, the system status page is shown and indicates which file is being processed.

In addition, this tab also allows the download and removal of the database files individually, a useful feature in secondary modules, which do not have the other system interfaces to perform these operations.

In case of database corruption, or problems to load the graphical interfaces (caused by a file), it is possible to clean them on this page without the need to load them, as a way to solve the problem.

#### 4.3.10.5 Advanced

In the fifth tab of the page, it is possible to set the administrator's password for accessing the graphical interfaces, configuration files and databases. In addition, these credentials are also used for intercommunication between modules, so that all must use the same password to exchange files and information.

It's also possible to change the cryptography key and enable "Secure Mode". When this mode is active, the databases and settings can be only changed when the module is accessed via a Wi-Fi network generated by the ESP32, so all the communication happens securely using the encryption of a separate WPA2 network. When accessing the module via the Wi-Fi network that the module is connected to, the databases can only be read.

In this tab, the following information is requested:

- Alterar senha administrativa - Field used to change the admin password.
- Modo Seguro - Field used to enable or disable secure mode.
- Alterar chave de criptografia - Field used to change the cryptography key.

When saving, all the passwords are compared internally to the original ones, and the changes are implemented and the system is rebooted.



Figure 4.38: Advanced settings page

By default the username and password “admin”, “admin” are used, and allow access to the settings of the panel until the credentials are changed.

#### 4.3.11 Interconnection between modules

In order to make it possible to intercommunicate all the modules in case of a database changes, new record events or when there is a movement between sectors, it is necessary to implement some form of synchronization of files and information between the modules.

In order to optimize the databases update process, the secondary modules generates hashes of their files and send them as “ETags” for comparison before receiving the database files. This way, in case the databases are already updated, the main module responds with code 304 (Not-Modified), and the secondary module continues to use the files already in the internal memory. Thus, only if the databases has changed the files will be sent (and the server will respond with code 200).

For the databases intercommunication, secondary modules requests the database files from the primary module through two types of update, the “Cyclic update” and the “Update in case of access”.

In the case of **Cyclic Update**, when booting the system, the modules determine the number of total active modules registered and enumerates each one. With this information, it is possible to know the total update period in seconds, which is equal to the number of total modules( $T$ ). Thus, it is possible to determine when each module must be updated based on the individual number (according to the formula 4.1).

$$N = T\%t \quad (4.1)$$

- N - Individual number of a module;
- T - Total of active modules;
- t - Instant of time (EPOCH).

For the following graphical examples (figures 4.39, 4.40, 4.41), the first module (Module 0) is the main one, and the other three (Module 01, 02, 03) represents the secondary ones. The file that changed is only needed for the modules 01 and 03 (for example a schedule that only affects them). Each blue connection represents checking for changes of the 4 databases (“usuarios.csv”, “setores.csv”, schedules of the sector, schedules of the next

sector to allow exiting).

The image 4.39 represents an example of what happens when a file changes using the Cyclic Update in normal mode.

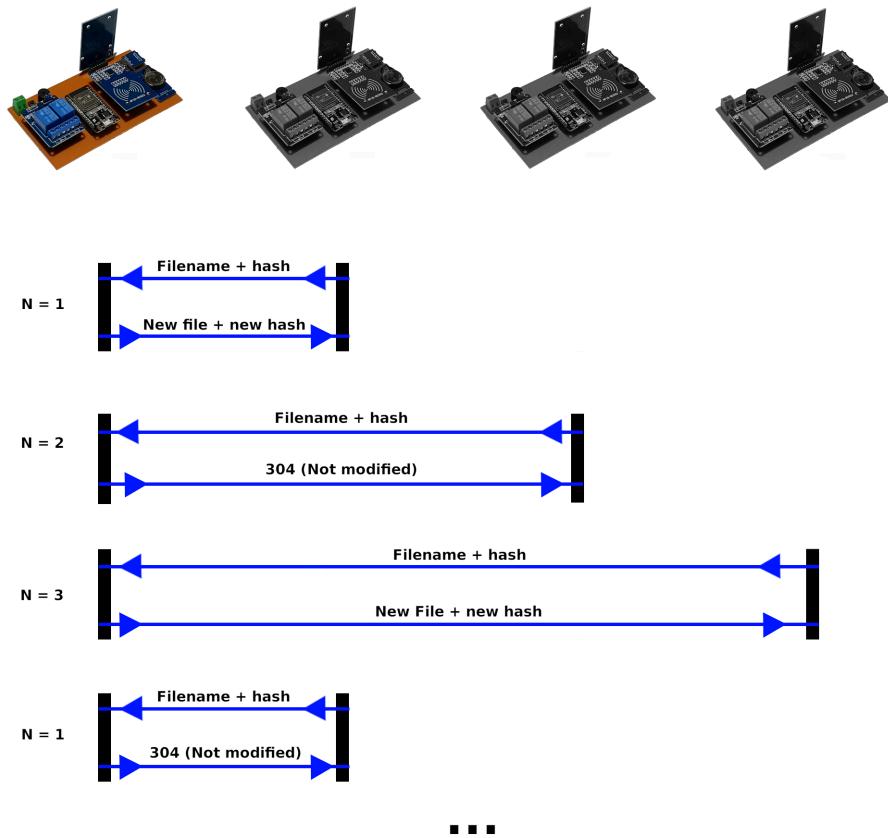


Figure 4.39: Intercommunication between modules using normal cyclic mode.

In this example, when  $N=1$ , the module 01 checks if any files needs to be updated, and the server responds with the new file. When  $N=2$ , the module 02 checks if any files needs to be updated, and the server responds 304 (file not-modified). When  $N=3$ , the module 03 checks if any files needs to be updated, and the server responds with the new file. Then the cycle restarts, this time all the modules are updated and the server will respond 304 (not-modified) to the requests.

A Smart Cyclic Update mode is also present, and uses less requests when the files aren't modified. It uses a broadcast from the main module to indicate the number of the database that changed. This way, only the modules that needs these databases will

request them.

The Smart Cyclic Update is represented in image 4.40.

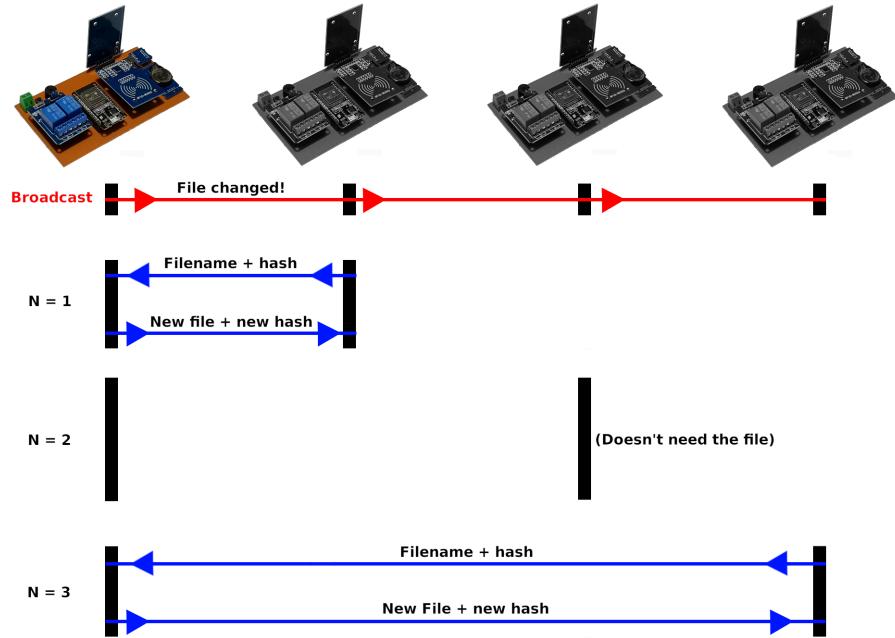


Figure 4.40: Intercommunication between modules using smart cyclic mode.

In this example, the first step is a broadcast sent by module 0 to indicate which files changed. Then, when  $N=1$ , the module 01 knows that it needs to be updated and asks for the files, and the server responds with the new database. When  $N=2$ , the module 02 already knows no file has changed and doesn't ask. When  $N=3$ , the module 03 knows that it needs to be updated and asks for the files, and the server responds with the new databases. Then the cycle stops until another broadcast is sent.

In **update in case of access**, data is requested whenever a card is detected. In this case, the following update order is used:

- When detecting a card, the user databases are updated to verify that the card is registered with one of the users;
- If so, the panel updates the schedules database of the attempted access sector (either the inbound or outbound sector).
- The sector database is then updated, so that it is possible to record the allowed or

denied event, and to communicate the next movement centers that occurred, for the control of the lighting relay modules and for records.

In the example of 4.41, only when an access attempt is made, the modules check if any of the files has changed.

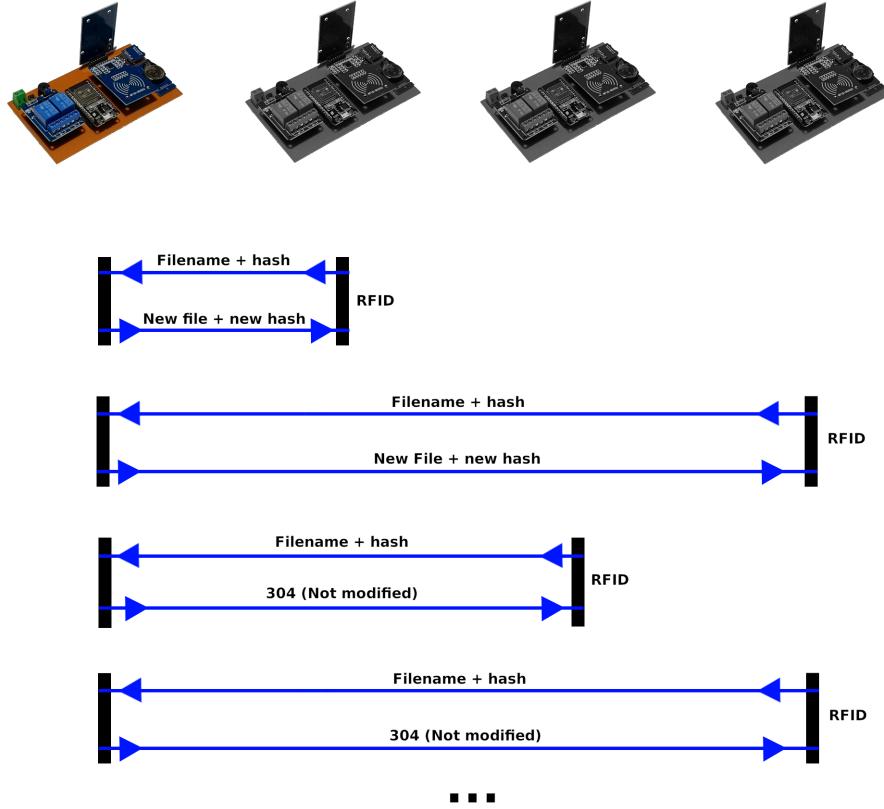


Figure 4.41: Intercommunication between modules using update in case of access.

The information intercommunication happens between two secondary modules in real time, without the primary module being used. This saves processing power from the main module and speeds up the process. To make this possible, the sectors database has to know the IP of all the modules responsible for each sector.

The most used information refers to the movement of users (the entry or exit of a user in a sector), and it's used to communicate to the next sector if the lights may be turned on or off, and to record the event in both modules.

In the intercommunication of real-time events, the module where the event happened

sends to the module responsible for the event (e.g.: next sector) the following parameters: (figure 4.42)

- Type of the record (TIPO): Numbers between -2 and 3, representing the type of the event (respectively: alarm, stop alarm, reboot, entry, exit, denied attempt);
- User: The number of the user of the event;
- Sector (SETOR): The number of the sector of the event;
- Time of the event (HORA): EPOCH time of the event
- Hash: An integrity and security hash using all the above data to check if the request is reliable.

If the communication was successful, the destination module responds with a code 200. If the communication fails, the sender module tries again (limited to 5 tries).

```

263 853.091064483 192.168.12.23 192.168.12.20 HTTP 239 GET /NTP HTTP/1.1
264 853.097354428 192.168.12.20 192.168.12.23 HTTP 201 HTTP/1.1 200 OK (text/plain)

► Frame 18: 329 bytes on wire (2632 bits), 329 bytes captured (2632 bits) on interface 0
► Ethernet II, Src: Expressi_f4:82:70 (30:ae:a4:f4:82:70), Dst: Expressi_f3:40:ec (30:ae:a4:f3:40:ec)
► Internet Protocol Version 4, Src: 192.168.12.21, Dst: 192.168.12.20
► Transmission Control Protocol, Src Port: 61847, Dst Port: 80, Seq: 1, Ack: 1, Len: 275
▼ Hypertext Transfer Protocol
  ▼ GET /registrar?tipo=1&user=1&setor=2&hora=1585578886&hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9 HTTP/1.1\r\n
    [Expert Info (Chat/Sequence): GET /registrar?tipo=1&user=1&setor=2&hora=1585578886&hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9 HTTP/1.1\r\n]
    [Severity level: Chat]
    [Group: Sequence]
    Request Method: GET
  ▶ Request URI: /registrar?tipo=1&user=1&setor=2&hora=1585578886&hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9
    Request URI Path: /registrar
    Request URI Query: tipo=1&user=1&setor=2&hora=1585578886&hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9
      Request URI Query Parameter: tipo=1
      Request URI Query Parameter: user=1
      Request URI Query Parameter: setor=2
      Request URI Query Parameter: hora=1585578886
      Request URI Query Parameter: hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9
    Request Version: HTTP/1.1
  Host: 192.168.12.20\r\n
  User-Agent: ESP32HTTPClient\r\n
  Connection: keep-alive\r\n
  Accept-Encoding: identity;q=1,chunked;q=0.1,*;q=0\r\n
  Authorization: Basic YWRtaW46YWRtaW4=\r\n
  \r\n
  [Full request URI: http://192.168.12.20/registrar?tipo=1&user=1&setor=2&hora=1585578886&hash=caba4dbcfa24b02ffd2523cec2363bd5db316cd9]
  [HTTP request 1/1]
  [Response in frame: 19]

```

Figure 4.42: Intercommunication between modules during event record

In the case of new events to be recorded, the main module is also notified to be able to record the event too. This way, the main module will record all the events that happened in all the sectors.



# **Chapter 5**

## **Tests and discussions**

In **Chapter 5**, a test simulating a real-world usage of the system is presented, including speed tests for booting time, file manipulation, and web-server responses, as well as discussions about the network packets generated by the system in each situation, and a market analysis for access control systems.

### **5.1 Speed tests**

For all the speed tests of this section, the main module is working at it's full capacity to make sure the results correspond to the worst scenario in terms of databases size. This means that the system is using databases with a total of 499 users, 49 sectors and 24500 schedules (500 schedules per sector).

#### **5.1.1 Boot time**

The boot time of the system includes the time taken to:

- Configure the ESP pins and components (RFID, SD card, RTC and relay);
- Load the 4 configuration files (Network, sectors, time and passwords);
- Checks if the module will run as main or secondary module;
- Start the web-server;

- Load the databases (users, sectors, schedules);
- Generate the hash for all the databases;
- Connect to the Wi-Fi network;
- Update date and time from the internet (NTP);

```

23:30:34.629 -> Iniciando...
23:30:35.060 -> Lendo configurações de setor...
23:30:35.060 -> Lendo configurações de rede...
23:30:35.159 -> Lendo configurações administrativas e chaves de criptografia...
23:30:35.623 -> Iniciando servidor...
23:30:35.623 -> Coletando usuários...
23:30:35.855 -> Coletando setores...
23:30:35.955 -> Coletando horários...
23:30:36.054 -> Coletando SHA...
23:30:36.783 -> Checando rede... -> 192.168.12.20
23:30:36.783 -> Lendo configurações de data e hora...
23:30:36.883 -> Hora atualizada! -> MDD=1350
23:30:36.916 -> Iniciado!      Tempo de boot: 2298ms

```

Figure 5.1: Boot time

All this actions, are done in 2.3 seconds (figure 5.1). This means that, in case of a reboot, or energy restoration, the modules will be online and running in under 3 seconds, an impressive time that was achieved after a lot of improvements.

### 5.1.2 File modification

The time taken to read, write and generate the hash of the files was also measured and represents the time used by the system when the system needs to get information of a file, or change it. These approximate times are shown in table 5.1:

Table 5.1: Time taken to process files.

	Read speed	Write Speed	Hashing (SHA1-HMAC)
Users database	~200ms	~1,2s	~40ms
Sectors database	~100ms	~800ms	~40ms
Schedules database	~100ms	~500ms	~40ms
Configuration files	~100ms	~200ms	-

The configuration files doesn't generate hashes due to it's reduced size.

The size of the databases and datasets used in the system are shown in table 5.2:

Table 5.2: Databases and datasets size

	Users	Sectors	Schedules	Records
Maximum datasets	499	49	500 per sector	500.000
Datatype	csv	csv	csv	csv
Maximum dataset size	62 B	60B	24B	100B
Maximum database size	~31kB	~3kB	12kB (per sector)	~50MB

The ESP32 flash memory installed in this module has 4MB, allowing all the databases (including all the 49 schedules databases in the main module) to be stored internally. The records must be stored in an external micro-SD card if it exceeds the available internal memory of the system.

### 5.1.3 Web-server

The web-server was also highly improved, since it's used not only for the user's graphical interface, but also for modules intercommunication.

This improvements can be seen in the loading times of the pages. For example, one of the most resource intensive pages is the schedules page, which needs all the 49 schedules database files, and also the users and sectors database. It also needs to download the HTML, CSS and the javascript files that controls the page.

When downloading large files (such as the records file), the average download speed is arround 300KB/s (2.4 Mbps) as shown in figure 5.2.

```
Conectando-se a 192.168.12.20:80... conectado.
A requisição HTTP foi enviada, aguardando resposta... 200 OK
Tamanho: 99967 (98K) [text/plain]
Salvando em: "registro.csv"

registro.csv      100%[=====] 97,62K   307KB/s   em 0,3s

2020-03-30 17:38:21 (307 KB/s) - "registro.csv" salvo [99967/99967]
```

Figure 5.2: Download speed for larger files.

The download speed depends on multiple factors, such as the network dimensions, the devices connected, the bandwidth available and the available resources of the network. A separate network was generated using a computer to avoid external interference during the speed tests.

When loading the page for the first time (no cache in destination browser), the page takes only 4 seconds to load and process all the 58 files (figure 5.3).

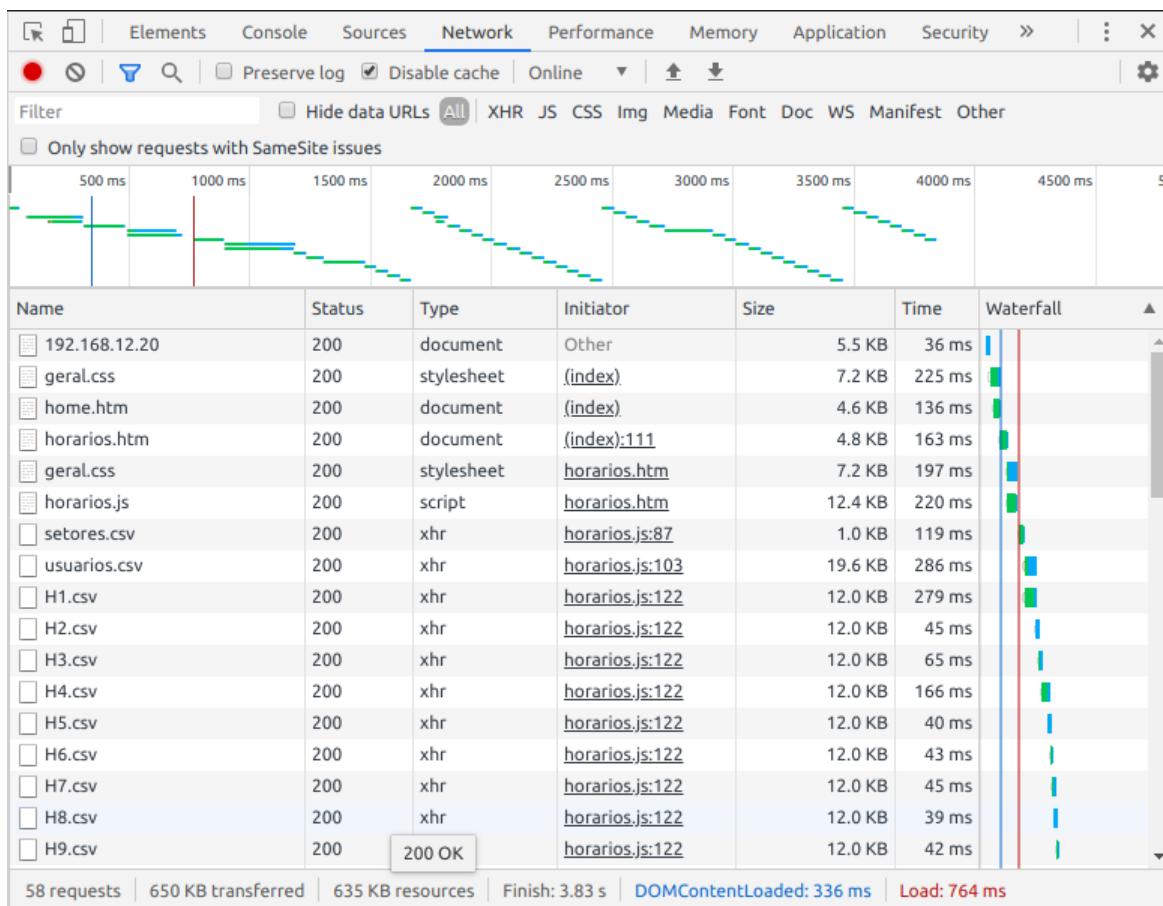


Figure 5.3: Time to load the Schedules page for the first time

When reloading the page, only changed files need to be downloaded. The change of the files is checked with the web-server via ETags. In case the file hasn't been changed, the web-server responds with 304 code, taking only around 16ms per file. This way, the same page can be loaded in under 1.5 seconds when using the browser's cache (figure 5.4).

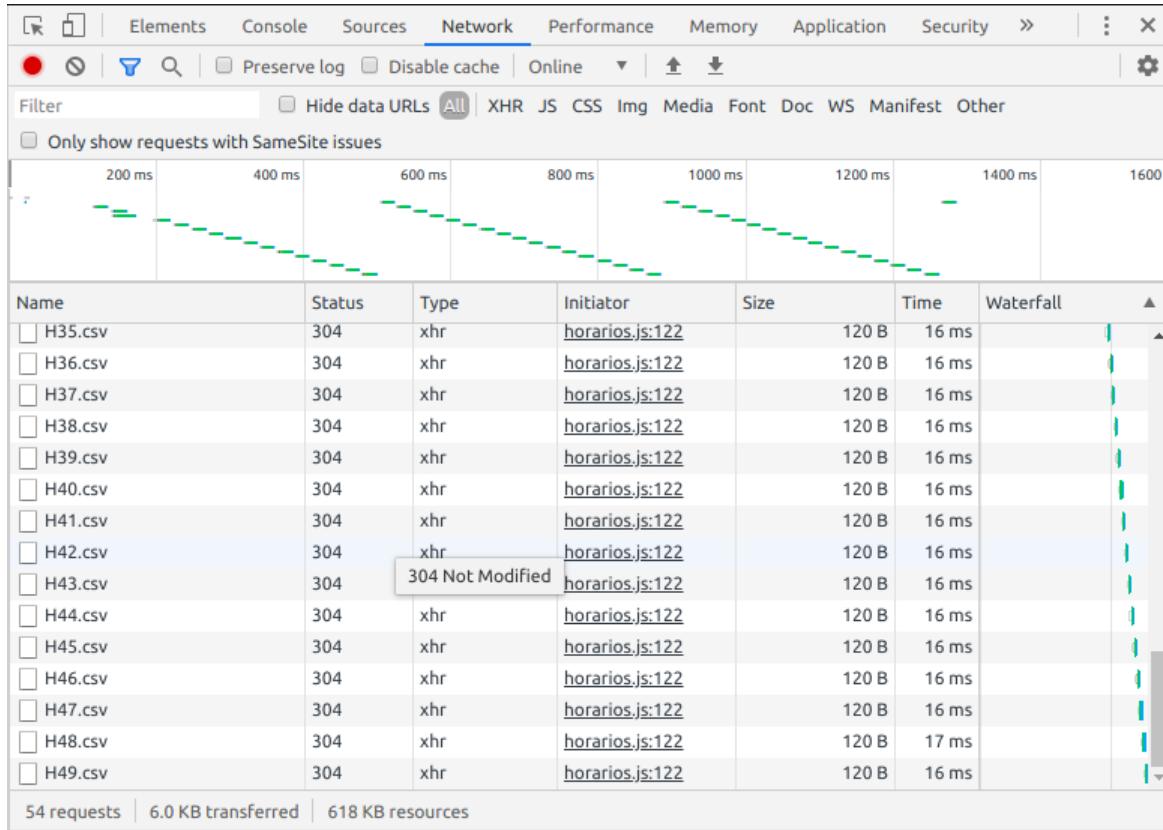


Figure 5.4: Time to load the Schedules page with cache

It's possible to predict the average download time of any database with any amount of data using formula 5.1.

$$t[ms] = (size[B]/300) + TTFB + M. \quad (5.1)$$

Where:

- t - Time taken to download a file (microseconds);
- size - Size of the file to be downloaded (Bytes);
- TTFB - Time to first byte, depends on the network and the time taken by the ESP32 to find the file (usually between 15 and 100ms).

If the file is already ETag-cached, the "size" can be considered equals to zero, and the time taken to the system respond is approximately equal to the TTFB (as the server will only respond with a 304 message). This effect can be seen in figure 5.4.

## 5.2 Intercommunication test

### 5.2.1 Files

When using Smart mode for files update, the first step of the communication between modules is the broadcast sent by the main module indicating that a file has changed. This broadcast is shown in 5.5, and it's composed by two bytes in hexadecimal that, when converted to ASCII, indicates the number of the file that has changed. The broadcast is sent to the address "239.1.2.3" via UDP.

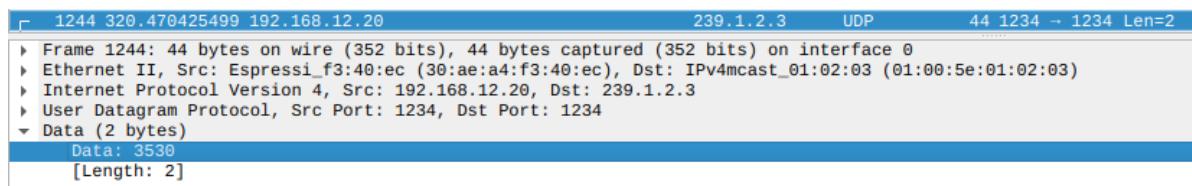


Figure 5.5: Broadcast used in Smart Mode to update files

When requesting a file, the secondary modules uses a simple GET request with the path of the file needed, the login credentials and a "If-None-Match" containing the hash of the file. (figure 5.6)

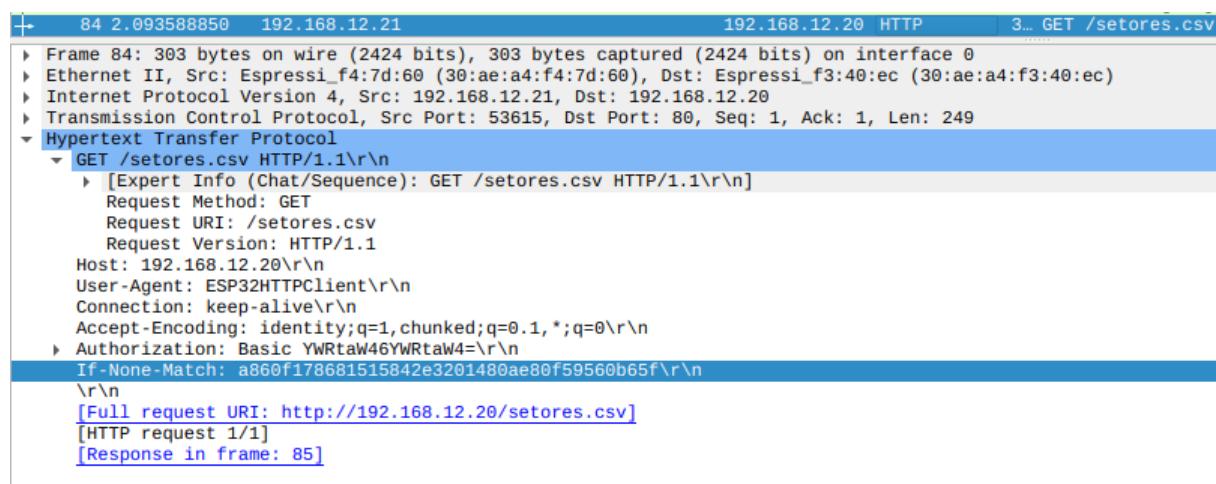


Figure 5.6: Request send by the secondary modules to check for file changes

If the credentials are correct and hash sent by the client doesn't match with the server, the main module sends the new file and new hash (ETag) (figure 5.7). This hash is then

generated again at the secondary modules and checked if it matches with the hash sent by the main module o check the file integrity and security.

```
+ 1331 320.834633669 192.168.12.20          192.168.12.21 HTTP      1.. HTTP/1.1 200 OK
▶ Frame 1331: 1416 bytes on wire (11328 bits), 1416 bytes captured (11328 bits) on interface 0
▶ Ethernet II, Src: Expressi_f3:40:ec (30:ae:a4:f3:40:ec), Dst: Expressi_f4:7d:60 (30:ae:a4:f4:7d:60)
▶ Internet Protocol Version 4, Src: 192.168.12.20, Dst: 192.168.12.21
▶ Transmission Control Protocol, Src Port: 54036, Seq: 18669, Ack: 252, Len: 1362
▶ [14 Reassembled TCP Segments (20030 bytes): #1303(1436), #1304(1436), #1305(1436), #1309(1436), #1312(1436),
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 200 OK\r\n
  ▶ Content-Length: 19750\r\n
  ▶ Content-Type: text/plain\r\n
  ▶ Access-Control-Allow-Origin: *\r\n
  ▶ Content-Disposition: inline; filename="usuarios.csv"\r\n
  ▶ Cache-Control: if-none-match\r\n
  ▶ ETag: c1ec5de5966f5fcfc6e19d63a0862d2fbb84c5\r\n
  ▶ Versao: 0\r\n
  ▶ Connection: close\r\n
  ▶ Accept-Ranges: none\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.082155604 seconds]
  [Request in frame: 1285]
  [Request URI: http://192.168.12.20/usuarios.csv]
  File Data: 19750 bytes
▼ Line-based text data: text/plain (500 lines)
```

Figure 5.7: Response code 200 when requesting the file “usuarios.csv”

If the credentials are correct and the hash matches with the file, the server responds with a 304 code, indicating that the file hasn’t changed (figure 5.8).

```
+ 85 2.111454846 192.168.12.20          192.168.12.21 HTTP      1.. HTTP/1.1 304 Not Modified
▶ Frame 85: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 0
▶ Ethernet II, Src: Expressi_f3:40:ec (30:ae:a4:f3:40:ec), Dst: Expressi_f4:7d:60 (30:ae:a4:f4:7d:60)
▶ Internet Protocol Version 4, Src: 192.168.12.20, Dst: 192.168.12.21
▶ Transmission Control Protocol, Src Port: 80, Dst Port: 53615, Seq: 1, Ack: 250, Len: 120
▼ Hypertext Transfer Protocol
  ▶ HTTP/1.1 304 Not Modified\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 304 Not Modified\r\n]
    ▶ Response Version: HTTP/1.1
    ▶ Status Code: 304
    ▶ [Status Code Description: Not Modified]
    ▶ Response Phrase: Not Modified
  ▶ Content-Length: 0\r\n
  ▶ Access-Control-Allow-Origin: *\r\n
  ▶ Connection: close\r\n
  ▶ Accept-Ranges: none\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.017865996 seconds]
  [Request in frame: 84]
  [Request URI: http://192.168.12.20/setores.csv]
```

Figure 5.8: Response code 304 when requesting the file “setores.csv”

Other web-server responses configured are:

- 401 (Unauthorized) - When the credentials are incorrect, or when the logout button is used (at the graphical interface)
- 404 (Not Found) - When the requested file doesn’t exists

### 5.2.2 Information

For the tests of this example, the modules were configured to allow access to the user "Usuário 01" at the "Setores 01, 02, 03 and 04". The user "Usuário 02" is only allowed to access the Sector 01. The update mode selected is Cyclic Update, with Smart Mode enabled. The displacement of the modules and sectors are represented in figure 5.9. The main module is the first one ("Setor 01"), and the others are the secondary modules.

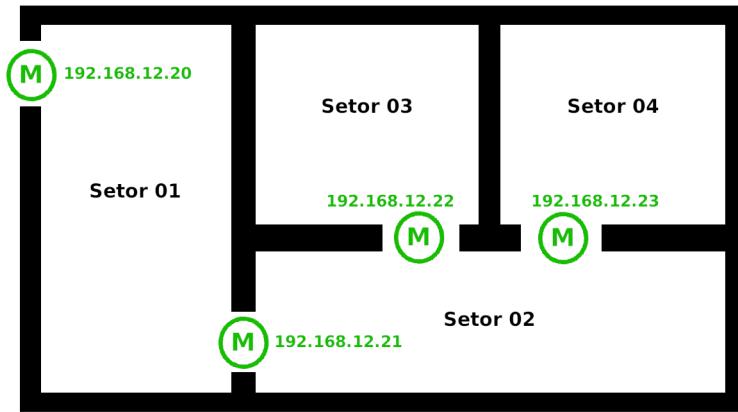


Figure 5.9: Example of real usage of the system

Every time a movement involves more than one sector, network packets are generated to notify the main and adjacent modules. For the example below, the packets are shown in figure 5.10, and are composed of pairs of data information (respectively request and response).

- Movement 01 - "Usuário 01" enters the "Setor 01" - No packets are exchanged, since all the movements only affects the sector controlled by the main module;
- Movement 02 - "Usuário 01" enters the "Setor 02" (exiting from "Setor 01") - Packets 09/10, 18/19;
- Movement 03 - "Usuário 01" enters the "Setor 04" (exiting from "Setor 02") - Packets 36/37, 45/46, 53/54;
- Movement 04 - "Usuário 01" enters the "Setor 02" (exiting from "Setor 04") - Packets 62/63, 70/71, 78/80;

- Movement 05 - "Usuário 01" enters the "Setor 01" (exiting from "Setor 02") - Packets 89/90, 97/98;
- Movement 06 - "Usuário 01" exits the "Setor 01" - No packets are exchanged, since all the movements only affects the sector controlled by the main module;
- Movement 07 - "Usuário 02" enters the "Setor 01" - No packets are exchanged, since all the movements only affects the sector controlled by the main module;
- Movement 08 - "Usuário 02" attempts to access the "Setor 02" - Packets 112/113;
- Movement 09 - "Usuário 02" exits the "Setor 01" - No packets are exchanged, since all the movements only affects the sector controlled by the main module;

As an example for better understanding the packets, when the "Movement 03" happens, the module responsible for the "Setor 04" needs to inform the module that controls "Setor 02" that someone has left the sector and who it was (packets 45/46), so the illumination of the "Setor 02" can be turned off if there is nobody there. Also, the main module needs to be notified of the exiting of a user from "Setor 02" (packets 36/37) and the entry in "Setor 04" (packets 53/54) to record these events.

No.	Source	Destination	Info
9	192.168.12.21	192.168.12.20	GET /registrar?tipo=2&user=1&setor=1&hora=1585578885&hash=
10	192.168.12.20	192.168.12.21	HTTP/1.1 200 OK (text/html)
18	192.168.12.21	192.168.12.20	GET /registrar?tipo=1&user=1&setor=2&hora=1585578886&hash=
19	192.168.12.20	192.168.12.21	HTTP/1.1 200 OK (text/html)
36	192.168.12.23	192.168.12.20	GET /registrar?tipo=2&user=1&setor=2&hora=1585578904&hash=
37	192.168.12.20	192.168.12.23	HTTP/1.1 200 OK (text/html)
45	192.168.12.23	192.168.12.21	GET /registrar?tipo=2&user=1&setor=2&hora=1585578904&hash=
46	192.168.12.21	192.168.12.23	HTTP/1.1 200 OK (text/html)
53	192.168.12.23	192.168.12.20	GET /registrar?tipo=1&user=1&setor=4&hora=1585578904&hash=
54	192.168.12.20	192.168.12.23	HTTP/1.1 200 OK (text/html)
62	192.168.12.23	192.168.12.20	GET /registrar?tipo=2&user=1&setor=4&hora=1585578936&hash=
63	192.168.12.20	192.168.12.23	HTTP/1.1 200 OK (text/html)
70	192.168.12.23	192.168.12.20	GET /registrar?tipo=1&user=1&setor=2&hora=1585578936&hash=
71	192.168.12.20	192.168.12.23	HTTP/1.1 200 OK (text/html)
78	192.168.12.23	192.168.12.21	GET /registrar?tipo=1&user=1&setor=2&hora=1585578936&hash=
80	192.168.12.21	192.168.12.23	HTTP/1.1 200 OK (text/html)
89	192.168.12.21	192.168.12.20	GET /registrar?tipo=2&user=1&setor=2&hora=1585578951&hash=
90	192.168.12.20	192.168.12.21	HTTP/1.1 200 OK (text/html)
97	192.168.12.21	192.168.12.20	GET /registrar?tipo=1&user=1&setor=1&hora=1585578951&hash=
98	192.168.12.20	192.168.12.21	HTTP/1.1 200 OK (text/html)
112	192.168.12.21	192.168.12.20	GET /registrar?tipo=3&user=2&setor=2&hora=1585578982&hash=
113	192.168.12.20	192.168.12.21	HTTP/1.1 200 OK (text/html)

Figure 5.10: Network activity of real usage of the system

The final events records are shown in the records page of the main module(figure 5.11).

<b>Tipo</b>	<b>Nome</b>	<b>Setor</b>	<b>Data</b>	<b>Entrada</b>	<b>Saída</b>	<b>Duração</b>
Permitido	Usuário 1	Setor 1	30/03/2020	14:34:33	14:34:45	00:00:12
Permitido	Usuário 1	Setor 2	30/03/2020	14:34:46	14:35:05	00:00:19
Permitido	Usuário 1	Setor 4	30/03/2020	14:35:05	14:35:37	00:00:32
Permitido	Usuário 1	Setor 2	30/03/2020	14:35:37	14:35:51	00:00:14
Permitido	Usuário 1	Setor 1	30/03/2020	14:35:51	14:36:05	00:00:14
Permitido	Usuário 2	Setor 1	30/03/2020	14:36:11	14:36:35	00:00:24
Negado	<b>Usuário 2</b>	<b>Setor 2</b>	<b>30/03/2020</b>	<b>14:36:22</b>	-	-

Figure 5.11: Example of the recorded events

### 5.3 Security

In terms of security, there are a few points that may be highlighted. The first one is the buffer overflow protection for databases. This is done using size-limited char arrays instead of the usual (for ESP32) Strings. Also, the data received via web requests are also verified before the system changes any variables in RAM or in the databases.

The second point is that there is no asynchronous web-server for ESP32 that supports HTTPS (SSL/TLS) and that has all the functionality needed for this project. This results in data being sent in plain-text with a Basic Authentication method for login with cookies.

The possible use of HTTPS also implies on many other problems, such as self-signed certificates, security warnings or DNS routing, since this system is planned to be used inside a local network.

To use HTTP in a more secure way, all the files and intercommunication that happens between the modules has an encrypted hash. This hash is used in the modules to check if the request was made by a trusted source (that knows the encryption key of the hash). Also, the databases also contain an encrypted ETag hash for integrity and security check.

To solve the authentication security of the system, a Secure Mode was developed. This way, when this mode is active, accessing the graphical user interface by the externally generated Wi-Fi network only allow the visualization of the databases and files, not being possible to request changes in those files. The only way to request files changes is by accessing the module using its own generated network, which has all the protection and encryption provided by the WPA2 cyphers and encryption.

Knowing that nowadays the UID and all the information of a RFID tag can be cloned, the system is recommended to be used only in the inside of places that are already restricted to known and reliable users. Even though this method of authentication has it's flaws, it requires specific hardware and access to the UID of the wanted tag to become a security problem. This way, the use of RFID can be considered safer than many metallic keys or holed cards (used in some hotels), as those can be cloned with a single picture.

In case of Denial of Service(DOS) attacks, simulated via multiple simultaneous file requests, the TCP stack gets full and the system tries to respond all the requests. If the attack continues and the web-server isn't capable of responding to all the requests, the watchdog timer triggers and the system reboots.

## 5.4 Market analysis

The final cost of production of this project in low scale, as done during it's development is around US\$15/module. When comparing this project with the majority of the available products at the market, it's possible to split them in two groups: the products at a similar price range, and the higher-cost products.

The majority of the similar cost products (up to 30US\$/module) usually lacks multiple features found in this project, such as:

- Wi-Fi communication;
- Multi-sector control;
- Schedulling system;
- Illumination control with intercommunication between modules;
- Web-based graphical interface with filters;
- No additional hardware/software needed;
- Micro-switch support to check the door state;
- Expiration date for users
- Timeouts for open doors;
- Events grouping in graphical interface (corresponding entries and exits);

- Realtime spreadsheets and graphs for users behaviour.

It's important to note that some low cost products has a numerical keypad for passwords, configurations and databases manipulation. This is not used in this project as all the configurations are done via the web-based graphical interface, and the system uses only RFID tags to allow or deny the access.

Also, some of them supports more users than this project. This limitation exists to keep the size of the databases smaller than the available free memory, as they are sent via Wi-Fi to all the modules, and those then calculate the hashes of the files for integrity and security checks. Also, some of the products only allow a basic register of users, without full names or expiration dates, as well as not needing to keep the databases synchronized with other modules.

When comparing with higher cost products, still many of them don't present all the cited above features of this project (such as an interconnected illumination control and wireless connectivity). However, those products also usually comes with some advantages, such as:

- Built-in display;
- Facial recognition / fingerprints scanners;
- More memory (for logs, records and databases);
- Online access of the databases and records;
- Support for a wider variety of electrical locks.

When using Access Control Systems for multiple sectors, the price can be a determinant factor. As an example, if a company needs to control 30 sectors, will spend around US\$450 by using this project. This price can easily be higher than US\$1500 when using other commercial products, some with them lacking some features present in this project.

This way, even though this project has some particularities (including both advantages and disadvantages), it presents a lot of advanced features and a low price, allowing this project to easily compete with other low-cost products, and even with the higher-cost products available at the market.

# Chapter 6

## Conclusion

In **Chapter 6**, both the general and specific objectives of this project are analysed, and the difficulties found during the development of the system is presented. Finally, some suggestions for future improvements are proposed.

### 6.1 Project analysis

The development of this project shows that it's possible to create an Access Control System that contains multiple advanced features (such as multi-sector control, scheduling times, illumination control, web-based graphical interface, and many other) using standard components, being a good alternative to the market products available nowadays.

This was possible due to the advances in the low-cost components used in this project, such as the ESP32, as well as the easy access to information needed to use those components and protocols.

The general objective of this project was achieved: developing an Access Control System based on ESP32 that is capable of controlling multiple sectors. Also, the specific objectives were achieved:

- **Inter-connectivity:** All the modules are capable of exchanging information and files in a secure and optimized way, without the need of external servers, by using hashes and HTTP headers.

- **Low-Cost:** It became possible to keep the project price under the US\$15 by using conventional components highly available in the market;
- **Customization:** The system supports the use of a wide range of electrical locks and micro-switches and is capable of controlling illumination. It's also possible to set how long the doors can be kept open before the alarm triggers;
- **User friendly interface:** The final result of the graphical interface was kept simple and intuitive to be easily accessed through any modern internet browser (with HTML5 support), and allows the administrator to view and edit the databases of the system (information about the registered users, sectors and schedules), and also analyze or download the records, as well as configure the modules.

The system has also proven that it's possible to use conventional and low cost components to create an Access Control System that includes many functionalities that are only present in commercial products that costs much more, such as Wi-Fi communication, multi-sector control and weekly schedules for access allowance.

Also, the performance of the final system was better than initially expected, with the system booting and being ready to be used in under 3 seconds. Furthermore, most of the communication between modules and browser happens in fractions of a second. The time taken to read and write files were also improved, making the system be capable of handling multiple databases without losses in stability.

Lastly, this project shows how the technology is evolving rapidly and getting each day cheaper and more powerful. This is seen in devices such as the ESP32, being able to easily handle complex projects, supporting multiple protocols, components and technologies.

## 6.2 Difficulties

The main difficulty of the development of this project is how big and complex the system became. The end project has more than 4.000 lines of code (C, C++, HTML, JS and CSS) produced by the author. It also contains more than 50 databases (with more than 25.000 datasets in total), 5 configuration files, 35 web-server pages (10 pages with

graphical interface and 25 used for requests and other data acquisition).

The second main difficulty is the development of a system that it composed by multiple modules, which need to exchange information and files in a highly optimized way, to avoid overloading the main module (server). This inter-communication requires both the server and the client to be synchronized to guarantee the security of the data sent.

Other difficulties include some data corruptions in the internal ESP32 memory during firmware upgrades, wrong memory addressing for files in the file-system, limitations of the ESP32 memory and the learning of techniques to improve the performance of the system.

Also, learning and using for the first time multiple protocols and technologies, such as HTTP Headers, keyed-hashes with message authentication code, SPI lines sharing, HTML, JS and CSS to develop a complex project is a challenging experience.

### 6.3 Suggested future improvements

For future improvements of the Access Control System, some of the suggestions are:

- **HTTPS implementation** - When the asynchronous web-server support it, implementation of HTTPS to improve the security of the system intercommunication by using the encryption and integrity check provided by TLS/SSL;
- **Increase in system's specifications** - Increase in the number of records, users, sectors and schedules that can be saved in the modules by improving the memory management;
- **External cover and PCB improvements** - Manufacturing of an external cover or box of the modules, as well as a smaller and double-sided PCB;
- **User Interface over internet** - When HTTPS gets implemented, the access of the system outside of the local network (via internet), providing an easy and secure way to configure the system from anywhere;
- **DOS Attack Protection** - Better limitation of HTTP requests in the web-server to avoid 0-day exploits caused by buffer-overflows and avoid unnecessary reboots;

- **Additional authentication methods** - The use of multiple authentication methods, such as facial recognition, to ensure that the card wasn't cloned or stolen;
- **Notifications** - Pop-ups in the graphical interface to show real time events, such as user movements, denied accesses and alarms;
- **Mobile web-pages** - Development of a mobile version of the web-pages, to improve user usability in smaller screens.

# Bibliography

- [1] V. R. Gonçalves, “Sistema de controle de acesso utilizando autenticação por rfid e gerenciamento por meio de software web.”, 2019.
- [2] E. F. Weles, “Protótipo para um sistema de automação de controle patrimonial utilizando tecnologia rfid”, *Revista Brasileira de Mecatrônica*, vol. 1, no. 4, pp. 1–10, 2019.
- [3] S. T. R. Htun, S. S. Y. Mon, and H. M. Tun, “Rfid-based monitoring and access control system for parliamentary campus”, *Intl. J. Sci. Technol. Res*, vol. 4, 2015.
- [4] Z. Fang, L. Wei, W. Chen, and Y. He, “A rfid-based kindergarten intelligence security system”, in *2012 IEEE Ninth International Conference on e-Business Engineering*, IEEE, 2012, pp. 321–326.
- [5] U. Farooq, M. Hasan, M. Amar, A. Hanif, and M. U. Asad, “Rfid based security and access control system”, *International Journal of Engineering and Technology*, pp. 309–314, Jan. 2014. DOI: 10.7763/IJET.2014.V6.718.
- [6] S. Electronics, *Serial peripheral interface (spi)*. [Online]. Available: <https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi>.
- [7] J. M. Flynn, “Understanding and using the i2c bus”, *EE Times-India*, vol. 1, no. 1, pp. 01–02, 1997. DOI: [ftp://ftp.propeller-chip.com/PCMProp/Chapter\\_10/Docs/i2c/I2C\\_bus.pdf](ftp://ftp.propeller-chip.com/PCMProp/Chapter_10/Docs/i2c/I2C_bus.pdf).
- [8] H. S. Mendonça, *Spi e i2c*. [Online]. Available: <https://paginas.fe.up.pt/~hsm/docencia/comp/spi-e-i2c/>.

- [9] L. George, *I2c – inter-integrated circuit*. [Online]. Available: <https://electrosome.com/i2c/>.
- [10] CISCO, *Enterprise mobility 8.5 design guide*. [Online]. Available: [https://www.cisco.com/c/en/us/td/docs/wireless/controller/8-5/Enterprise-Mobility-8-5-Design-Guide/Enterprise\\_Mobility\\_8-5\\_Deployment\\_Guide/wlanrf.html](https://www.cisco.com/c/en/us/td/docs/wireless/controller/8-5/Enterprise-Mobility-8-5-Design-Guide/Enterprise_Mobility_8-5_Deployment_Guide/wlanrf.html).
- [11] J. Scarpati, *Wireless security protocols: The difference between wep, wpa, wpa2*. [Online]. Available: <https://searchnetworking.techtarget.com/feature/Wireless-encryption-basics-Understanding-WEP-WPA-and-WPA2>.
- [12] A. Sari and M. Karay, “Comparative analysis of wireless security protocols: Wep vs wpa”, *International Journal of Communications, Network and System Sciences*, vol. 8, pp. 483–491, Dec. 2015. DOI: 10.4236/ijcns.2015.812043.
- [13] F. Katz, “Wpa vs. wpa2: Is wpa2 really an improvement on wpa?”, Jul. 2019.
- [14] M. e colaboradores individuais, *Uma visão geral do http*. [Online]. Available: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>.
- [15] R. Want, “An introduction to rfid technology”, *IEEE Pervasive Computing*, vol. 5, no. 01, pp. 25–33, Jan. 2006, ISSN: 1536-1268. DOI: 10.1109/MPRV.2006.2.
- [16] S. E. Sarma, S. A. Weis, and D. W. Engels, “Rfid systems and security and privacy implications”, in *Cryptographic Hardware and Embedded Systems - CHES 2002*, B. S. Kaliski, ç. K. Koç, and C. Paar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 454–469, ISBN: 978-3-540-36400-9.
- [17] Espressif, *Esp32 series datasheet*. [Online]. Available: [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf).
- [18] M. Lab, *Esp32 pinout – how to use gpio pins?* [Online]. Available: <https://microcontrollerslab.com/esp32-pinout-use-gpio-pins/>.
- [19] S. RELAY, *Srd-05vdc-sl-c datasheet*. [Online]. Available: <https://datasheetspdf.com/pdf-file/720556/Songle/SRD-05VDC-SL-C/1>.

- [20] NXP, *Datasheet - mfrc522*. [Online]. Available: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>.
- [21] M. integrated, *Ds3231*. [Online]. Available: <https://datasheets.maximintegrated.com/en/ds/DS3231.pdf>.
- [22] ARDUINO, *Arduino - software*. [Online]. Available: <https://www.arduino.cc/en/main/software>.
- [23] T. G. contributors, *Geany - the flyweight ide*. [Online]. Available: <https://www.geany.org/>.
- [24] Mozilla, *Html: Linguagem de marcação de hipertexto*. [Online]. Available: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.
- [25] ——, *Css*. [Online]. Available: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.
- [26] ——, *Javascript*. [Online]. Available: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>.