Federal University of Technology - Paraná – Brazil – UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering
Integration Workshop 3 (ELEX22) – S71 – 2024/1

Technical Report
ParaSunflower

Estevão Vieira Gomes – estevao@alunos.utfpr.edu.br
Guilherme Libério de Souza e Silva – guilhermesilva.2018@alunos.utfpr.edu.br
João Rauli Sarmento – joaosarmento@alunos.utfpr.edu.br
Leonardo Bueno Fischer – leonardofischer@alunos.utfpr.edu.br
Lucas Iuri dos Santos – lucassantos.1994@alunos.utfpr.edu.br
Rafael Josef de Araújo Cordeiro Merling – rafaelmerling@alunos.utfpr.edu.br

Junho de 2024

Abstract

Many outdoor places, be it public or private, in big cities suffer with lack of shading area. With this issue in mind we developed a project for the Integration Workshop 3 course that offer a solution: a rentable parasol.

The main idea is that the user can use an app to localize and rent for a period of time a parasol in a fixed location. To make the product more attractive the user can control the parasol's position with app commands or set it to move automatically to follow the sun's position. Another feature is that all the system is powered by solar cells, making it more environmentally friendly.

In this report we present every step of the development, with the components used, schematics, the setbacks (with the solutions found) and the results obtained.

1 Introduction

This project was developed for the Integration Workshop 3 subject of the Computer Engineering course at UTFPR Curitiba.

This project has the intention to solve the following problem: outdoor places may have large open areas with little or no shades, usually provided only by trees. A way to deal with this problem is the use of a parasol, however, bringing your own to a public park is usually a nuisance, since they are usually heavy and, differently from beaches, it is not easy to dig the hole to put the parasol. Furthermore, if the sun moves and you lose your shade, you have no option other than diggin another hole to reposition the parasol.

Sun-following systems have been developed for many purposes (e.g. measuring radiation and optimizing solar energy generation). [1] proposed such a system that served as reference for solving the aforementioned problem.

With all of this being said, the proposed solution consists on the creation of a parasol renting system, providing easy access to parasols on public parks and making sure of a shade on the area. Also, the parasol is able to follow the sunlight for the user while rented, which guarantees the maximum amount of sunlight covered. Another feature of the project offered to the user while renting a parasol, is to be able to provide a USB output, which will be able to charge a mobile phone or any electronical device that can be charged through that kind of output, making use of the sunlight energy.

1.1 Overview

We now present a high level view of the project with its main components and their relationships. The following block diagram illustrates them.

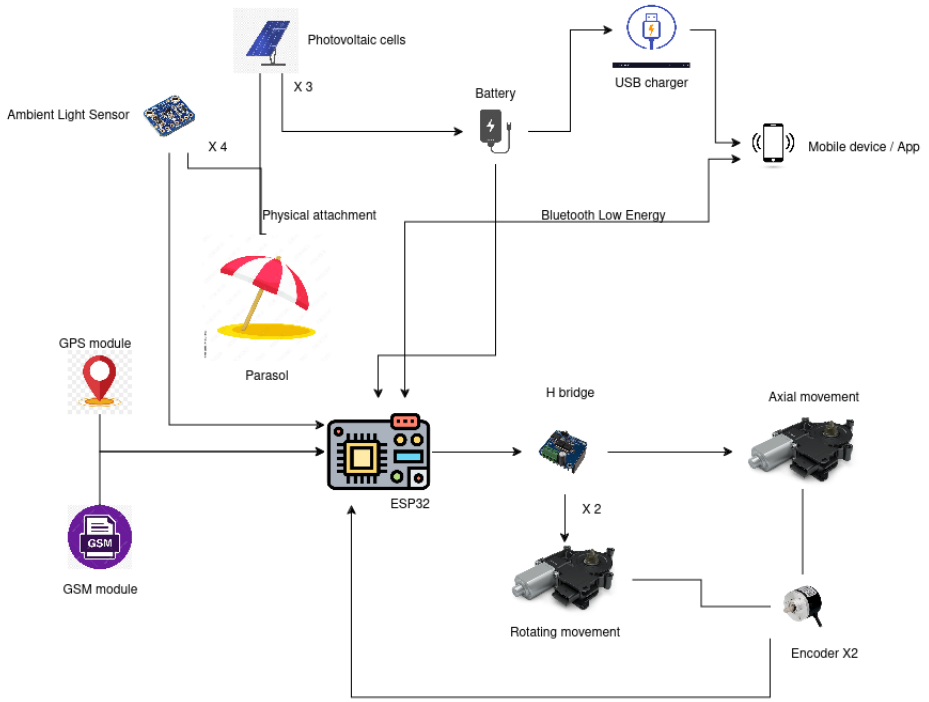


Figure 1: Block diagram of the project

The project main piece is the parasol, but it is only physically attached to the photovoltaic cells and luminosity sensors. The parasol is placed on a support, along with the batteries, electronic components and motors, that will be further explained in the mechanical development.

Since the project is sun-powered we have the photovoltaic cells attached to the main battery for charging purposes. The main battery is then responsible for powering the main ESP32 microcontroller and the USB charging module for mobile devices.

The ESP32 is the core of the project since it is involved in all communication and motor control. We have a Global Positioning System and a Global System for Mobile communicatioon modules connected to it to feed (through the GSM) the ParaSunflower position (through the GPS) to a remote server maintained by the owners. Thus, with multiple parasols installed on a park or another public area, the user can locate the closest available parasol using the smartphone App.

By installing the ParaSunflower app, the user can fiind an available parasol and rent it using the unique QR Code present in each unit. The renting allows for the user to adjust the parasol either manually (adjusting the rotation and deflectioon angle through app commands) or automatically (when the parasol directs itself towards the sun using its four luminance sensors). Also, the user has access to an USB charging port during the reenting period. The rental payment is completed using PIX technology, available in any bank account in the smartphone.

The ESP32 has a connection to the mobile device that has the app installed through its in-built Bluetooth Low Energy functionality. The app sends commands to manually adjust the ParaSunflower position, rent more time, toggle between manual and automatic movement. And the ESP32 will send acknowledgment messages to the app.

The four luminosity sensors attached to the parasol are connected to the ESP32, they are used to get luminosity data from diametrally opposed positions. These data are used to activate the motors.

There are two (window lift) motors used, one for rotational movement and another for radial movement. These motors were chosen because of their providing torque, needed for the parasol movement, that could not be provided by other motors (e.g. stepper motors) in the same price range. The motors have one encoder attached to each of them, these encoders are connected to the ESP32 to provide information about the movement of the motors to avoid problems like multiple rotations that would cause disorder in the cables.

Between the motors and the ESP32 there are two H bridges that are gonna drive the motor movement according the direction provided by the ESP32 (and the luminosity sensors).

On Figure 2 there is a visual representation of the communication channels on the device and how they will operate through their working time.

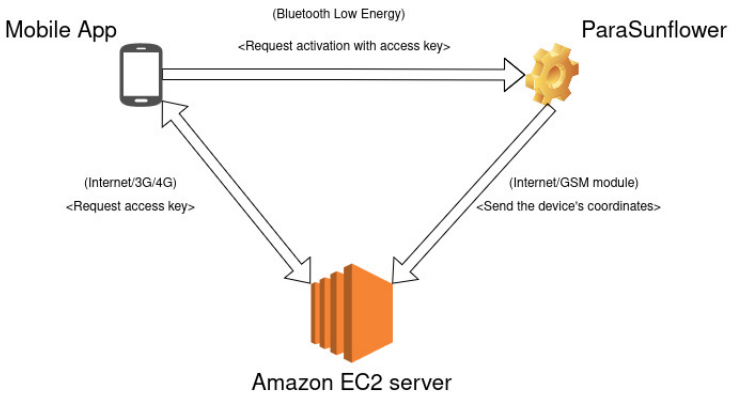


Figure 2: Communication channels of the project

2 Project Specification

The project was divided into three main development parts, Mechanical, Electronical and Software, each one of them has their own set of Functional Requirements presented on the following sections: 2.2, 2.3 and 2.4. Also there is the need for some Anti-Requirements, presented on the section 2.5. For the optional requirements, it is possible to check the page on the blog [2]

2.1 Overall Requirements

- The project considers 2 actors, the user and the owner. The requirements are written considering the actions (and responsibilities) of both actors.

2.2 Electronical Functional Requirements

- EFR01 - The parasol must be able to gather information about the sun's position and set its own position, according to the sun's automatically.
- EFR02 - The parasol must communicate with the mobile app.
- EFR03 - The parasol must have an USB output able to recharge mobile devices.
- EFR04 - The parasol must have its power supplied by a battery.
- EFR05 - The system must use encoder to know the position of the parasun's parts.

2.3 Mechanical Functional Requirements

- MFR01 - The parasol must be able to align with the sunlight from 9AM to 3PM.

- MFR02 - The parasol must be able to rotate 360º on its own axis.
- MFR03 - The parasol must be able to move radially, formin up to a 45º angle with the normal line.
- MFR04 - The parasol must be able to perform the both the rotational and radial movement automatically.
- MFR05 - The parasol must be always open.
- MFR06 - The parasol must have a box on its base to hold the electronical components and circuitry.
- MFR07 - The parasol must have a platform coupled on its top.
- MFR08 - The parasol must be attached to a rotatory base.
- MFR09 - Both motors responsible for the movements (one for the rotational and the other for the radial), after the reductions, must have a 5RPM +- 2.5RPM velocity.

2.4 Software Funcitonal Requirements

- SFR01 - The app must provide the option for the user to make the parasol automatically set its own position.
- SFR02 - The app must be able to connect with the remote server
- SFR03 - The must be able to read QR Code.
- SFR04 - The app must have the option to select the amount of time the parasol will be unlocked.
- SFR05 - The app must provide the option to add more time
- SFR06 - The app must be able to accept payments
- SFR07 - The app must clearly inform the price for every payment.
- SFR08 - The app must clearly show how much rented time the parasol has left.
- SFR09 - The app must send a notification to the user 5 minutes before the time is expired.
- SFR10 - The remote server must be able to process payments
- SFR11 - The remote server must warn the app in case of a payment being accepted.
- SFR12 - The remote server must monitor current weather conditions.

- SFR13 - The app must inform the user if the normal charge mode is available on the parasol.
- SFR14 - When the using conditions change to unsuitable, the app must send the user a notification.

2.5 Anti-Requirements

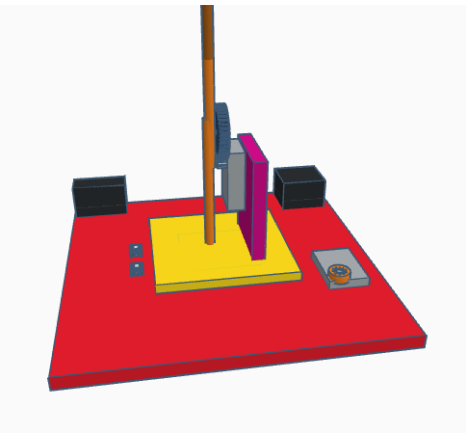
- AR01 - The parasol will NOT communicate with other instances of itself.
- AR02 - The parasol will NOT have a digital display.
- AR03 - The parasol will NOT monitor wind speed nor rain through sensors.
- AR04 - The app will NOT have a Sign Up Feature.
- AR05 - The system will NOT accept any payment method other than PIX.
- AR06 - The parasol will NOT measure temperature.
- AR07 - The system will NOT distinguish between sunny and cloudy days.
- AR08 - The app will NOT have custom rent time, only preset ones.
- AR09 - The app will NOT work on IOS or nay other operational system that isn't Android.
- AR10 - The parasol will NOT have an AC outlet for phone charging.
- AR11 - The parasol will NOT be accompanied by tables or chairs.
- AR12 - The parasol will NOT be usable on beaches or sandy areas.
- AR13 - The parasol's DC motors will NOT be able to move all at the same time.
- AR14 - The parasol will NOT rotate faster than 5^{o} per second.
- AR15 - The parasol will NOT be movable by the user.
- AR16 - There will NOT be an APP for the owner, all communications with the owner will happen by way of SMS messages.
- AR17 - The parasol wll NOT be able to work on weathers with winds faster than 19km/h (3rd degree in the Beafourt Scale).

3 Development

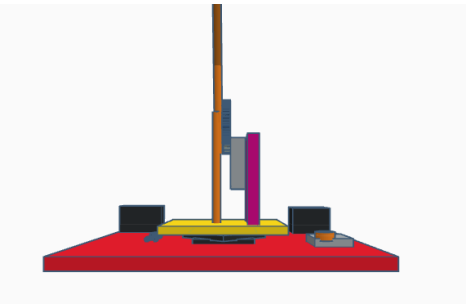
This section describes the project's development process, describing the technical details as well as some difficulties and how they were handled. Further details on the development progress can be obtained in the project's blog.

3.1 Mechanic

For the initial planning of the mechanical structure, it was necessary to consider the kind of movements the adjustments would require. With that in mind, it was planned to make the structure a dual-axis kind, with one of them being a rotational one, where the parasol rotates on his own axis and the other being a radial one, where the parasol can adjust its angle towards the normal line, this created the need for a base that is stable enough to deal with those movements. The sketches of the base were made on Tinkercad[3] and can be seen on the figures 3a and 3b.



(a) Common view of the base



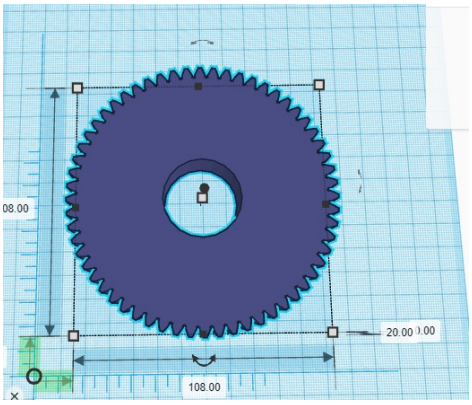
(b) Front view of the base

Figure 3: Design of the base

The base has two layers: a bigger one to create the work room for the other layer while giving the initial stability and space to add some spare weights to increase the stability of the base on the ground; and a second, smaller one consisting on the wooden base, that has all parts responsible for the radial movement on it's top. Between the wooden bases are the rotatory base and a gear, which connects with the motor on the side, initially planned to be through a bicicle crown and a chain. On the smaller base there is a support to hold the motor responsible for the radial movement with its gear is connected to the mechanical arm.

Figure 4: Both bases coupled alongside the mechanical arm

Due to some issues with the bicicle crown, where the chain became loose after a couple turns after every single test, the concept needed to be changed, so it was replaced for a gear system, placing an intermediary gear in between the motor's gear and the mains rotatory gear. The design can be seen on Figure 5a and the implementation on Figure 5b.



(a) Design of the intermediary gear

(b) Intermediary gear coupled on the base

Figure 5: Intermediary gear design and coupling on the base

For the build itself, everything with the exception of the rotatory base and the mechanical arm was made using wood, since it would be easier to manipulate the material and make it have the necessary size. Each motion is performed by its own DC motor, which is one used by electric windows in cars (the motor

used can be seen on figure 7. Those are very powerfull and alongside the gears they generate a torque that is more than enough for any movement of the parasol.

Figure 6: Gear responsible for radial movement coupled to mechanical arm

Figure 7: DC Motor used on the project

3.2 Electronic

The firsts steps in the eletronical design of the ParaSunFlower were to map all the GPIOs of the microtroller of choice, being the ESP32 DEVKIT V1[4], and designing the circuit using KiCAD[5], for such tasks the production team first had to know every components that were to be used through the development. During repeated analysis some changes were made and some components were

changed, the final list of materials is as follows.

- 1x ESP32 DEVKIT V1
- 2x BTS7960 H-Bridges
- 4x BH1750 Light Sensors
- 2x KY-040 Rotacional Encoders
- 1x GPS NEO-6M module
- 1x GSM GPRS Sim900 module
- 3x 6V Photovoltaic Cells
- 1x USB charging module
- 1x HTZ5L battery
- 1x 12v 7Ah Elgin Nobreak battery
- 1x LM317 Voltage regulator
- 1x L7805 Voltage regulator
- 1x BC548 Transistor
- 1x CD4051 Multiplexer

In addition to the itens listed above, there are also a variety of resistors, capacitors and diodes used in the design to atain the proper functionalities of the circuit.

As for the KiCAD portion of the project, the final design is presented on Figure 8.

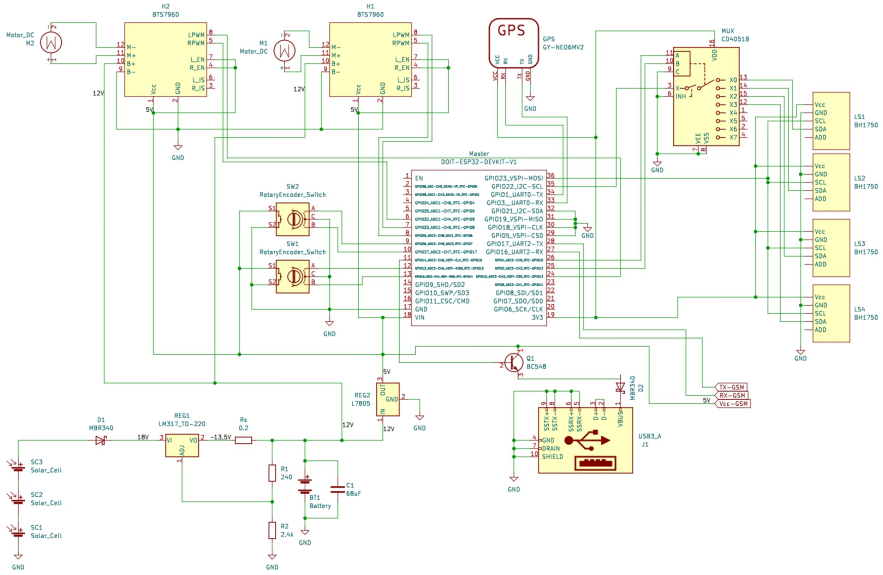


Figure 8: Schematic used on the main PCB, made on KiCAD

Opting for a PCB to bring the circuit to life, a colleague was contacted to print it, the design was also made in KiCAD, as showed in Figure 9.

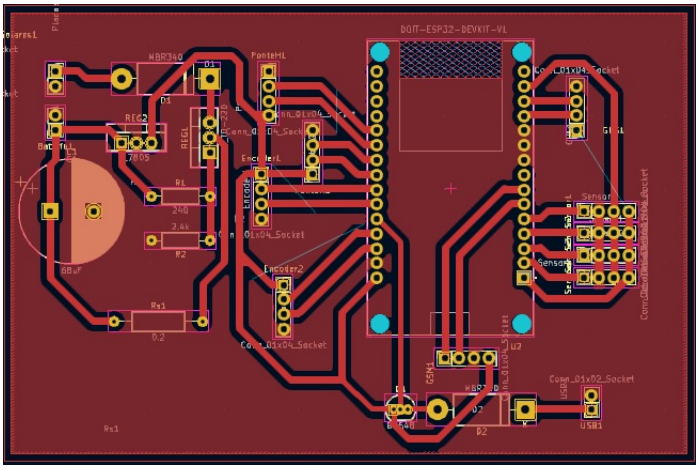


Figure 9: PCB Design

Due to some problems, described in section 3.2.5, the team had to also use a Universal PCB especifically for the multiplexer.

Most of the pins in the ESP32 are being used in this project, as showed on figure 10, each pin's functions is described in the following sections

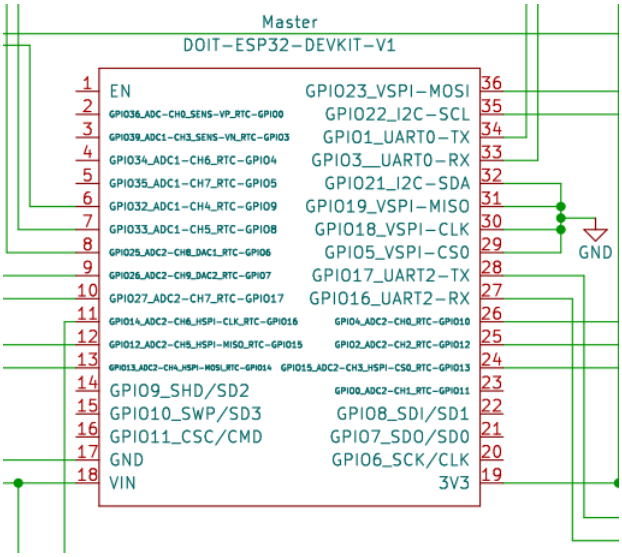
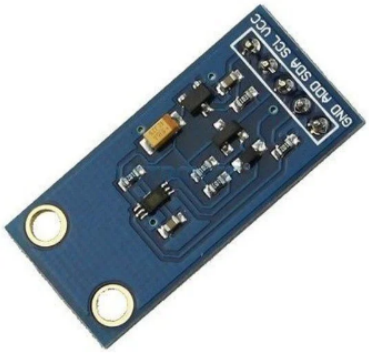


Figure 10: ESP32 Pin Planning

3.2.1 H-Bridges

The H-bridge model chosen for the Parasunflower was the BTS7960[6], a robust component able to take 43A of current and 50V. Mapped to GPIOs 15, 25, 33, and 32, the H-bridges were more than enough to control the powerful DC motors used to move the parasol in both its axis (Figure 11a).

(a) H-Bridge



(b) Luminosity sensor

Figure 11: Components BTS7960 (H-Bridge) and BH1750FVI (Luminosity sensor)

3.2.2 Luminosity Sensors

Positioned atop the parasol are all 4 BH1750[7] luminosity sensors (Figure 11b), being able to measure up to 64 thousand lumens, with a maximum resolution of 0,5 lumen, these sensors make the whole project work. Each sensor's clock are mapped to GPIO22, the default I2C clock pin, coupled with the CD4051 Multiplexer so all four signals can be read by GPIO23 (Figure 12), the ESP32 is able to use these measurements to control the parasol's movement relative to the sun, as shown in the section 3.3.1.

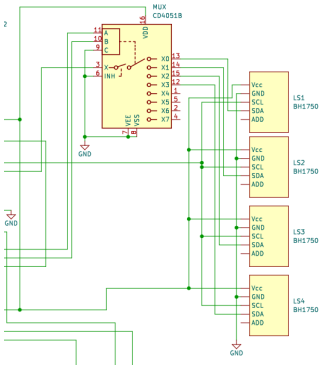


Figure 12: Sensors and Multiplexer Circuit Design

3.2.3 GPS and GSM

Since both modules use UART communication they are both mapped to the default UART ports of the ESP32, GPIOs 1, 3, 16 e 17. They're used to determine the Parasunflower's latitude and longitude and send this information to the server in order to show the parasol's position in the app's map.

3.2.4 Batteries and Solar Cells

The whole system is powered by two 12V batteries, which are charged by a series association of three Photovoltaic Cells, each being able to supply 6V. A voltage regulator is positioned after the cells so the 18V are regulated to 13.5V in order to properly charge the batteries.

3.2.5 Challenges and Solutions

During development the team faced a couple of challenges regarding the eletronical project and design, a total of 8 versions of the KiCAD design were made to accommodate all the changes from the circuit.

At first, LDR sensors were to be used, but consulting with other professors it was made clear that they would not be enough; in order to solve this prob-

lem the BH1750 was found, a highly accurate light sensor which returns light measurements in Lumens, delivering exactly what the project demanded.

Another big challenge and setback was the limited number of I2Cs addresses of the BH1750 sensor, having only two addresses while four were needed, this problem was tackled again and again until finally being solved by using a CD4051 Multiplexer IC to loop between the signal of each sensor, a solution that, while not simple, proved to be effective lefting GPIO15 available.

The team also faced a problem with the GPIOs of the ESP32, during the testing phase of the BTS7960 H-Bridges, it was found that the GPIO35 specifically is the only pin which is uncapable of generating PWM signals, luckly GPIO15 was available because of the modifications in the light sensors circuitry (as described above), so a simple change in the pin mapping was made.

Additionally the PCB printing process was at fault, in order to solve it, the group had to carve some incomplete trails with a stylet, a simple yet arduous job. Furthermore, problably the worst problem to uncover were the faulty wires going up to the light sensors: for still unknown reasons only the cold-colored wires extracted from a network cable were not retrieving any data, among a plethora of failed atempts, the only solution found was changing the cold-colored wires to hot-colored ones.

3.3 Software

3.3.1 Firmware

The ESP32 code was designed to be easy to implement, read and make modifications on the fly, a couple of specific libraries were used, namely BH1750.h for the light sensors, ESP32Encoder.h for the rotary encoder and the esp32 native libraries for the Bluetooth Low Energy.

The most interesting part of the code is the logic behind the sun following routine, as described in the flowchart presented in Figure 13.

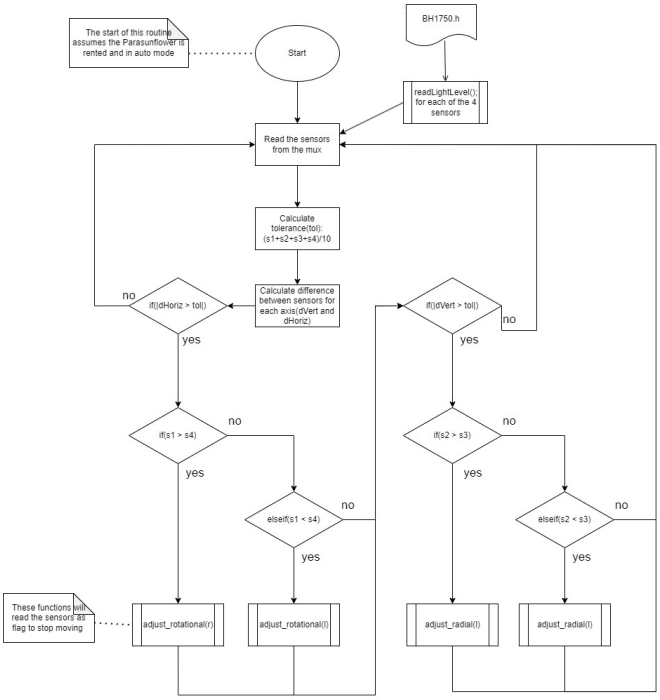


Figure 13: Sun-following routine flowchart

It's worth mentioning that the tolerance is calculated based on the mean amount of light received by the sensors, the treshold of movement being set at 10% of that mean.

3.3.2 App

For the Android app project the chosen technology for implementation was React Native[8] using Expo and Typescript[9], due to the similarity to web development, modules and libraries availability, easy to find documentation and a big active community.

For the navigation between the app screens the react-navigation[10] module was used, while zustand[11] library was used to mantain track of several variables according to App State changes. The permissions required to use the app include acess to GPS location, camera and bluetooth scan and connect.

There were nine total screens created, but only seven used for the final version, the reason for that difference being that two screens were made for development testing purposes. The first screen in the flow is a map screen that displays a Google Maps component with the user's current location (Figure 14) and markers in the parasols' locations (Figure 15) obtained through a GET HTTP request to the server. Also there is a text display and a button to proceed to the next screen.

The next screen (Figure 16) consists of a camera component to read the QR code displayed on the parasol containing it's ID. Once the QR code is sucessfully read the app automatically navigates to the next screen (Figure 17) where there are time options and prices for each, consisting of 10, 30, 45 and 60 minutes with a price of R$0,25 per minute, once the user selects the desired option it proceeds to a screen (Figure 18) displaying the total price to be paid and a button to generate a PIX code, which once clicked makes a POST HTTP request to the server sending the parasol's ID and time selected and receiving back the PIX code. It also makes the app go to the next screen (Figure 19) where the code generated is displayed along with a button to copy it to clipboard and a five minute timer, in this screen the app makes requests(polling) to the *check payment* API endpoint, once there is a status different from pending the app proceeds accordingly: if cancelled, it goes back to the generate PIX code screen, if returns that the weather conditions are not appropriate it returns to the first screen and if approved proceeds to the time remaining and controls screen.

In this screen there are some information being displayed: there is a timer with the total time payed for, if not connected to the parasol via bluetooth a text is displayed requesting the user to connect to it using a button located in the lower part of the screen alongside an "Add Time" button which leads to the same flow described before starting from time options. If not connected via bluetooth, the other buttons remain disabled, and once it is the app sends a command to the ESP signalizing that the rent process was successful and the buttons become available. The parasunflower starts on auto mode, displaying only a button to

Figure 14: User Location on Map

Figure 15: Parasol Marker on Map

switch to manual mode, once switched more buttons show up, giving the options to rotate left and right and to radially move up and down, also a button to switch back to auto is displayed. The different situations of this screen can be seen in Figures 20, 21, 22 and 23.

As for the main challenges during the implementation there were many issues encountered, among them some problems with the initial implementation using Javascript instead of Typescript, which made it harder to perceive errors due to the lack of some pre-compilation checks only present in Typescript, also for the implementation of the bluetooth integration that all sources found used it instead of Javascript. Furthermore some conflicts between modules caused delays in the development and when in time for non-development builds some problems also appeared with permissions, server communication and credentials for the Google Maps API, which requires a development licence in Play Store to work in internal distribution builds.

3.3.3 Server

The server was projected and made using the Python's Fast API[12] framework and the PostgreSQL[13] database, the integration with the PIX payment was done using the Mercado Pago API[14], everything was deployed in an AWS EC2[15] service. The following endpoints were created on the API:

- Create Rental: Receives the ID of the parasol and the time selected by the user on the app. This endpoint will check the wind conditions using the OpenWeatherAPI to know if the weather conditions are suitable for the renting and insert the new rental on the database. At the end it will return

Figure 16: QR Code Reading

Figure 17: Time options

Figure 18: PIX generation

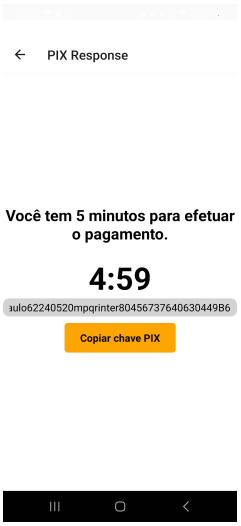


Figure 19: PIX response

Figure 20: Controls auto mode with bluetooth off

Figure 21: Controls auto mode with bluetooth on

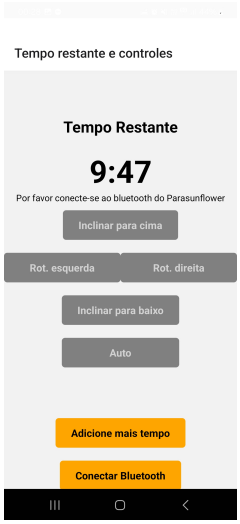


Figure 22: Controls manual mode with bluetooth off

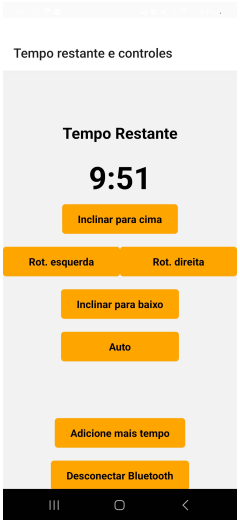


Figure 23: Controls manual mode with bluetooth on

the copy and paste PIX code from the Mercado Pago API alongside the rental ID.

- Check Payment: Receives the ID of the rental. This endpoint checks the condition of the PIX payment on the Mercado Pago API, if it is considered as approved, the rental is updated on the database with the date of the approvement and it returns the expiration date alongside the payment status and payment ID, otherwise it just returns the payment status and payment ID.
- Create Time Addition Payment: Receives the rental ID and the time selected by the user on the app. It updates the condition of the rental on the database
- Get Parasuns Positions: It doesn't receives any parameters. This endpoint gets a list of the parasuns registered on the database and returns then.
- Create Location Entry: Receives the latitude, longitude and the ID of the parasol. This endpoint inserts the new parasol with its coordinates on the database

With this, the server is able to make the proper communications with the device for the rental and locating routines.

4 Results

Some pictures of the device working are presented on Figure 24. This section will also cover the budget of the project and the task report of the group, with the estimated hours and the spent hours on the project

4.1 Budget

The final budget can be seen at the Figure 25. Totalizing R$1449,00, the most expressive costs that can be pointed is the Parasol itself, which turned out being quite more expensive than originally thought, alongside the set of bateries (motorcycle and nobreak), the H-Bridges and luminosity sensors for a more robust system. Even though the metal support for the parasol is the most expensive on the list, the team already knew from the beginning its value and how crucial it would be to make the project viable.

(a) ParasunFlower top view

(b) Parasun aiming directly at the sun

Figure 24: ParasunFlower at parque Barigui

	Price	Total
Parasol	R$ 160,00	R$ 1.449,00
Metal Support	R$ 259,00	
2x ESP32	R$ 80,00	
Battery Charger Kit	R$ 70,00	
Photovoltaic Cell(Extra)	R$ 94,00	
3 x Motor De Vidro Elétrico Universal Mo	R$ 90,00	
GSM Module ESP32	R$ 120,00	
Bateria Moto HTZ5L	R$ 140,00	
4x Luminosity Sensor	R$ 60,00	
2x Driver Ponte H BTS7960	R$ 120,00	
Usb 3.0 female module	R$ 6,00	
Charger TP4056	R$ 15,00	
Módulo GPS NEO-6M	R$ 40,00	
Bateria Selada Elgin 12v 7ah Nobreak	R$ 60,00	
Base Giratória Cadeira De Barco	R$ 70,00	
3x Magnetic Encoder	R$ 45,00	
PCB Printing	R$ 20,00	

Figure 25: Final Budget

4.2 Schedule

Considering Table 1 we can see that we even with a couple of issues and setbacks, we were able to be under of the total time estimated with the 30% error margin.

Category	Time Estimated + 30% (h)	Time Spent(h)
Mechanic	134,55	124,55
Electronic	292,5	250
Software	192,4	202
Integration	102,7	97
Presentation, Video, Technical Report	68,9	98,5
Total	791,05	772,05

Table 1: Task Schedule

5 Conclusion

Making a final rewind on everything that happened during the development of the project it is possible to get some points of relevance. One of them is the necessity of a good first plan and the necessity for this plan to be flexible and be able to change, either for planned risks or last minute problems that might happen, for example, a task taking more time than initially estimated.

The other is to understand how the team works and adapt the communication in a way that every member can be on the same page. Because with this strategy it gets easier to delegate the functions to the members and to know who is delayed, on time or ahead and make the necessary adjustments if a necessity occurs.

A lot of issues faced on the development were already expected, the main one is the mechanical development due to the lack of knowledgement and experience on the sector. But also, the difficulty on some debugs due to amount of connections that was necessary to get the necessary functionalities for the project.

In a final conclusion, we were able to develop a sun following parasun that can be rented by anyone in public parks and provide them with a easy to use solution for their lack of shade, giving people a maximum shaded area at all times or the option to manually adjust their parasun position without even having to get up from their chairs.

References

[1] P Roth, A Georgiev, and H Boudinov. Cheap two axis sun following device. *Energy conversion and management*, 46(7-8):1179–1192, 2005.

[2] Project's functional requirements. `https://scarlet-meeting-99f.notion.site/Requirements-4e7d898a18884e95bdda89f65f8b41b2`.

[3] Tinkercad official page. `https://www.tinkercad.com/`.

[4] ESP32 DataSheet. `https://www.alldatasheet.com/datasheet-pdf/pdf/1148023/ESPRESSIF/ESP32.html`.

[5] KiCad. `https://www.kicad.org/`.

[6] BTS7960 Datasheet. `https://www.alldatasheet.com/datasheet-pdf/pdf/152657/INFINEON/BTS7960.html`.

[7] BH1750 Datasheet. `https://www.alldatasheet.com/datasheet-pdf/pdf/338083/ROHM/BH1750FVI.html`.

[8] React Native. `https://reactnative.dev/docs/getting-started`.

[9] TypeScript. `https://www.typescriptlang.org/docs/`.

[10] React Navigation. https://reactnavigation.org/.

[11] Zustand. https://docs.pmnd.rs/zustand.

[12] FastAPI. https://fastapi.tiangolo.com/.

[13] PostgreSQL. https://www.postgresql.org/.

[14] Mercado Pago API. https://www.mercadopago.com.br/developers/pt/reference.

[15] Amazon EC2. https://aws.amazon.com/pt/ec2/.