UPC - Universidad Politécnica de Catalunya

January 18^{th}, 2023 - Barcelona, Spain

820771 Control and Automation for the Efficient Use of Energy

Final Project Report

Group 6
Alice Scalamandrè
Laura Melo Amaro
Sara Kiprijanova

Contents

List of Figures

List of Tables

1 Introduction

As a consequence of Ukraine's invasion by the Russian military forces on the 22^{nd} of February of 2022, a global energy market disruption started. The new geopolitical and energy market panorama requires the EU and the world, to drastically accelerate the Energy Transition, and separate themselves from unreliable suppliers and environmentally negative energy sources - fossil fuels. For the EU Member States reducing energy consumption is seen as one of the key measures to reduce energy bills and tackle supply issues.

In light of Ukraine's Invasion, the European Commission elaborated the REPowerEU plan, which focuses on becoming independent from Russian oil and gas, while at the same time fast forward the green transition. A broad set of measures to respond to this goal are put into place, such as diversification of energy supplies/suppliers, an accelerated roll-out of renewable energy to replace fossil fuels in homes, industry and power generation, and energy savings. This last is considered the quickest and cheapest to address the current energy crisis while reducing energy bills. In September 2022, the European Commission presented some measures to address the high energy prices. The strongest targets are related to the reduction of electricity demand. The first one is the reduction of 10% of the monthly gross electricity consumption when compared to an average reference period [1]. The second measure states an obligation for each Member State to reduce gross electricity consumption during peak price hours - at least 5% on average per hour (compared with forecasted consumption) [1].

Aligned with the regulation and plans put into place in recent months, the concepts of Energy Flexibility and Demand-Side Management (DSM) are being touted as vital key business cases to ensure the reliability of the grid network. DSM can be implemented at different levels, from residential to industrial, and the main objective is to help decrease energy costs, provide more efficient energy security and reduce the environmental impact of energy production. The implementation is usually put into place by independent service providers, utilities or governance institutions[2].

A critical and evolutionary way of thinking about energy demand management is vital for the acceleration of the Energy Transition. An educated society will lead to a more resilient environment and economy. Ensuring producer and consumer responsibility, while developing educational networks and sharing experiences among stakeholders will be essential to the rise of energy efficiency strategies.

2 Project Description

Aligned with the current energy context the objective of this project is to develop and optimize a demand-side management tool. The replacement of a large proportion of our transport fleet by EVs will have massive knock-on effects on our electricity systems, which may lead to additional pressure on the electrical grid. This pressure, if not addressed properly, may originate in grid congestion or even blackouts. The fast growth of EVs and their uncontrolled and non-optimized charging can conduct to increased electricity bills for end-users. From the demand-side management perspective, this issue must be addressed, both from the economic and practical side of it, to facilitate and promote the integration of EVs in urban areas.

The main problems that this project intends to solve are:

- User side: High electricity costs due to a non-optimised charging schedule.
- Grid side: overload caused by high demand during peak periods, which may lead to power losses. Integration of EVs negatively affects the voltage stability of the grid, which depends on the location, penetration level and EV charging time.

Therefore, the final purpose of this project is the development of a smart charging automated tool to optimize the charging schedule for householder users, minimize electricity costs and contribute to the peak shaving on the Spanish power grid.

The system focus on the time period when the user is at home, and the car is parked and plugged in. By sensing the vehicle's presence at the selected area (i.e. garage or parking slot) the system will send a notification to the user's phone requesting the number of hours the user wants the car to charge. With open-source data collected from the cloud regarding the Day-Ahead market prices, the system will check the following "cheapest hours" and automatically charge the car during that optimal period of time. The user will be informed through a notification in his personal phone account. Additionally, the tool will be able to measure, through a current sensor, the amount of power that is been fed into the car's battery. This tool will support the uptake of EVs in an effective transition rather than a chaotic disruption.

2.1 Project Goals

This project intends to enable energy, social, economic and environmental goals. The smart charging tool that is presented in this report is part of an emerging concept that wants to improve the user's experience by unlocking insights into charging behaviour and energy consumption, allowing the drivers to have control of the charging process. The main goals associated with this project are the following,

1. **Increased Grid flexibility:** this tool helps to stabilise the grid by avoiding peak electricity demand and allows EV owners to be more flexible with their charging schedules. The capacity to delay or direct charging allows for better flexibility of the system and limits the requirement for expensive network upgrades. EV smart charging guarantees that the supply and demand of electricity are better balanced, as it is intended to move electricity utilization away from existing peaks -peak shaving.
2. **Reduced electricity bills:** smart charging also results in lower electricity bills for EV owners. By charging at off-peak times, when the electricity prices are cheaper, the end-user will be saving money.

3. **Reduced Carbon Footprint:** smart charging helps reducing the overall carbon footprint by avoiding peak electricity demand periods, usually associated with non-renewable power sources. In this way, the energy used to supply the battery will most likely come from less polluting sources.
4. **Improved Battery Health and Lifecycle:** lithium-ion batteries age with each charge-discharge cycle, losing some of their capacity over time. Smart charging helps optimise EV battery health, by avoiding deep discharges, which maximizes the battery's lifespan.
5. **Improved EV Charging Experience:** By enabling EV owners to program their charging schedules remotely through an online program, without having to manually connect or disconnect their vehicles. Smart charging improves user experience, by making it more convenient and efficient.

The goals presented are expected to be reached fully after the project reaches larger scale adoption.

3 Datasheet of the load

For this project, the initial implementation scenario suggests charging an EV as a power load; however, for developing and testing the setup, a scaled-down version was used - a portable power bank. Therefore, Table 1 presents and compares the Load data in both cases: the case of charging an EV and charging a portable power bank.

Table 1: Datasheet of the load: EV and power bank. Reference for the Real life Scenario [3].

Load Datasheet			
Experimental Scenario: Portable Power Bank			**Real life Scenario: Nissan LEAF (2018)**
5V/2A	**Power input**	**Charging Method**	3.6kW
10,000mAh (∼37Wh)	**Battery capacity**		40kWh
4-6 hours	**Charging time**		12 hours

The value of 3.6 kW was chosen because home charging points typically have a power rating of 3.6kW-7kW. Additionally, the value of 12 charging hours referes to a 0% State Of Charge (SoC) to a 100% SoC. Nevertheless, for the sake of the project, the SoC of the battery at the arrival time was always assumed to be around 30%, moreover the charging is programmed to stop at a SoC of 80%. Therefore, the charging time is considered to be just of 6 hours instead of 12. This decision was made because keeping the battery between 20% and 80% SoC extend the battery life.

4 Project Management

In this section, the structure of the meetings, the product backlog and the risk matrix will be discussed.

A weekly meeting seemed necessary in order to achieve all the project goals within the given timelines. Each meeting was structured in a similar way: icebreaker, summary of past

achievements, check of product backlog, completion of the most important task, addition of new tasks, if necessary, and division of future work.

4.1 Product Backlog

As presented by the Scrum definition, "the Product Backlog is an emergent, ordered list of what is needed to improve the product" [4].

As mentioned before, a fixed meeting time was dedicated to the organization of the product backlog and Trello was identified as a useful tool for fulfilling this task. As it can be seen from figure 1, the board was divided into four columns: Backlog, In progress, Ready and Done, and each task was categorized with different types of tags.

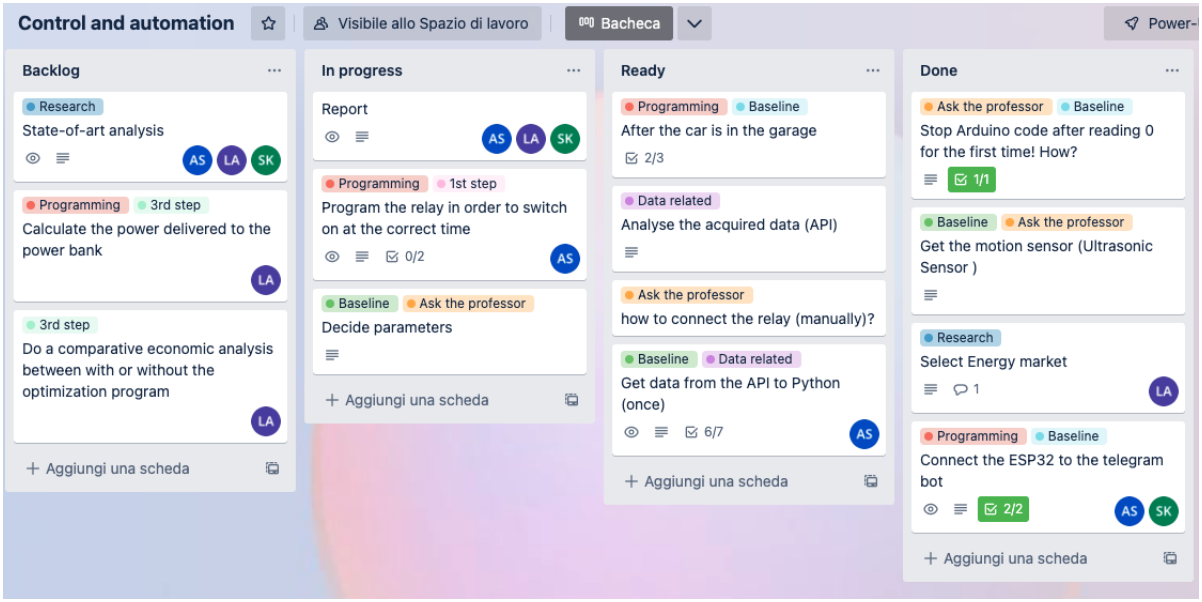


Figure 1: Trello board used for work management.

4.2 Risk Assessment

The project presented different risks and each of them was analyzed depending on its severity and probability, as shown in table 2. The severity and probability levels are identified as explained by [5].

The severity levels are:

- Marginal: the hazard may either be controlled, or would commonly result in less than minor, illness, injury or system damage.
- Moderate: The hazard may commonly cause severe injury or illness or major system damage, requiring immediate corrective action.
- Critical: The hazard may commonly cause death or major system loss, requiring immediate cessation of the unsafe activity or operation.

On the other hand, the probability levels are:

- Improbable: Unlikely but possible to occur during standard operations
- Occasional: Likely to occur some time during standard operations
- Probable: Likely to occur often during standard operations

Table 2: Project risk matrix.

	Critical	Moderate	Marginal
Probable	Problems in the electronics connections	Lack of data	Sensor malfunction
Occasional	Poor performances due to errors in the code	Delayed transmission of electricity prices	Lack of cooperation and distribution of work
Improbable	Exposed electronics equipment with high voltage	Third party API changes	Conflicts between team members

For each risk it was also identified a correspondent mitigation strategy.

- **Problems in the electronics connections**: continuous testing and proof of concept with the help of peers and the professor
- **Poor performaces due to errors in the code**: continuous testing and proof of concept with the help of peers and the professor
- **Exposed electronics equipment with high voltage**: careful handling of the devices
- **Lack of data**: adapting a different strategy (ex. aquiring data from another source)
- **Delayed transmission of electricity prices**: continuous testing
- **Third party API changes**: adapting a different strategy (ex. change the API or find new access for the same API)
- **Sensor malfunction**: request for backup
- **Lack of cooperation and distribution of work**: have one meeting a week and task division using Trello
- **Conflicts between team members**: open communication and feedback sessions

5 Sensors

In this section, the properties of the sensors and electronic devices used for the project will be presented, as well as their electronic diagrams. The project setup used 2 sensors: an ultrasonic sensor for proximity detection and a current transformer for power consumption determination. At the same time, a single-channel relay module was used to control the power consumption of the load.

5.1 Motion detection

The main trigger of the overall process happening in the project is the determination of a vehicle at a particular spot - in this case, a home garage. To do so, an ultrasonic sensor for

motion detection is placed on the floor of the parking spot.

5.1.1 Sensor description

The sensor used is a URM09 analog ultrasonic sensor and it measures the distance to an object using ultrasonic sound waves.

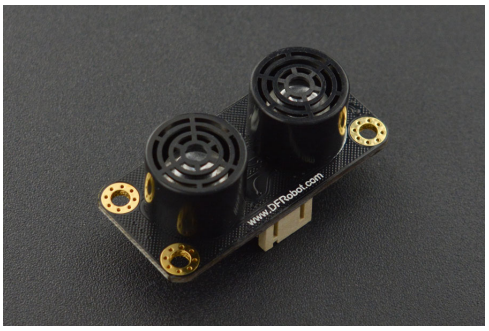


Figure 2: URM09 Ultrasonic Sensor. Source: dfrobot.com.

This sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. The sensor determines the distance to the targeted object by measuring the time lapses between the sending and receiving of the ultrasonic pulse.

5.1.2 Wiring diagram

The sensor has a three-pin output:

1. **Analog sensor pin**, Input pin to read analog signal, connected to ESP32 pin A0;
2. **VCC**, Supply input for powering the sensor, 5V (The chosen setup used the 3.3V ESP32 output for improved reading accuracy);
3. **GND**, 0V reference pin;

The wiring diagram is presented in Figure 3.

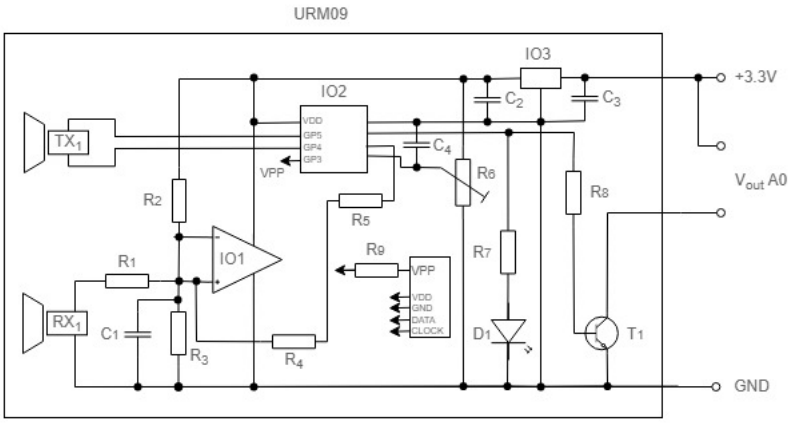


Figure 3: Electrical scheme: URM09 Ultrasonic Sensor and wiring diagram with ESP32.

5.1.3 Signal conversion

URM09 reads analog signals and therefore to be able to understand it, it is needed to convert this value to the desired output - the distance between the sensor and the vehicle. To do so, there are 2 values defined:

1. **The maximum range of the sensor**, 500 cm (in code: MAX RANGE), defined by the manufacturer;
2. **The analog-to-digital accuracy of the ESP32**, 4095 (in code: ADC SOLUTION) - The voltage measured is assigned to a value between 0 and 4095, in which 0V corresponds to 0, and 3.3V corresponds to 4095. Any voltage between 0 V and 3.3 V will be given the corresponding value in between - for this reason, the 3.3V output of the ESP32 is used instead of the V one.

The conversion is done according to the formula:

$$Distance = AnalogSignal * \frac{MaxRange}{ADC_{SOLUTION}} \tag{1}$$

The converted value is displayed in centimeters, and for the purpose of the project, the trigger distance to initiate the following activities is 0cm.

5.2 Charging control

To increase the economic benefit of the vehicle owner, as well as to support the power grid by charging in lower-demand hours, it is necessary to control the charging times of the parked vehicle. To do so, a simple relay is installed between the AC power supply and the vehicle.

5.2.1 Device description

The relay used in the project is a 5V single-channel relay module. It is an electromechanical device capable of controlling the power flow by opening or closing the contacts of a switch. The relay uses a small electric current to energize a coil, which later attracts the contacts of a switch and pulls them together when activated, and a spring pushes them apart when the coil is not energized.

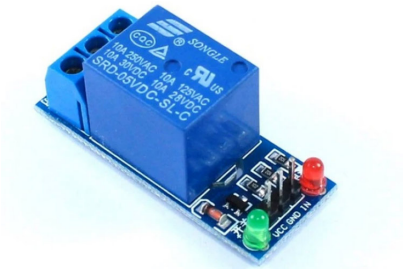


Figure 4: 5V single-channel relay module. Source: components101.com.

5.2.2 Wiring diagram

The single-module relay has 3 pin outputs for the microcontroller and 3 outputs for the AC circuit. Their properties are the following:

1. **Relay trigger**, Input pin to activate the relay, connected to ESP32 pin D2;
2. **VCC**, Supply input for powering the relay coil, 5V;
3. **GND**, 0V reference pin;
4. **COM**, Common terminal of the relay where the AC power supply is connected;
5. **NO**, Normally open terminal of the relay;
6. **NC**, Normally closed terminal of the relay, where the load is connected.

The wiring diagram is presented in Figure 5.

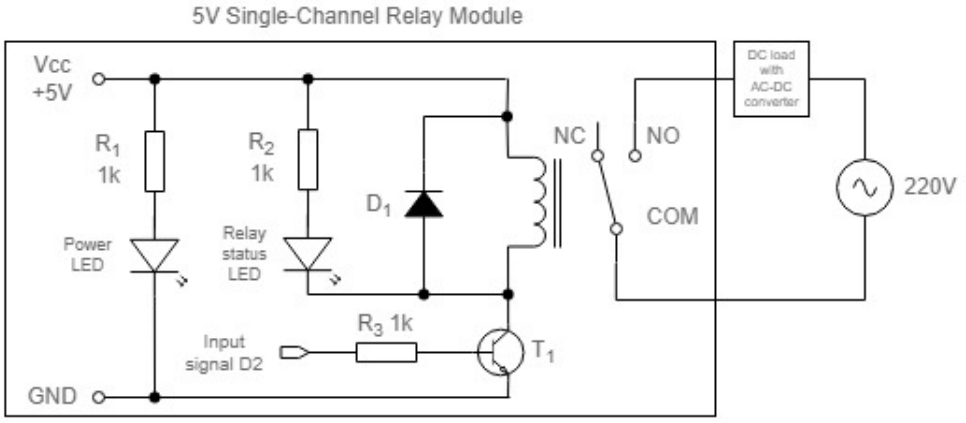


Figure 5: Electrical scheme: single module relay and wiring diagram with ESP32.

5.2.3 Switch control

There are several modes of operation of the relay that is determined by the initial choice of input and output pins and wiring. These choices will decide which orders from the ESP32 will activate or deactivate the relay, which means that power will either be delivered to or cut from the load.

As seen from the wiring diagram, for the project, the selected mode of the relay is HIGH-level trigger normally closed mode - this means that the IN terminal is connected to 5V source of the ESP32 and the switch is open. In this setup, the initial state of the relay is OFF, so the vehicle is not charging. To trigger the switch, the D2 pin needs to be programmed to HIGH, which will close the switch and allow power flow to the vehicle.

5.3 Power consumption

As one of the goals of the project is to determine the economical benefit of charging during off-peak times when electricity is cheaper, it is necessary to track the power consumed by the vehicle while charging. Therefore, a current transformer is installed on the vehicle charging cable, before the current is delivered to the onboard AC-DC power converter.

5.3.1 Sensor description

The sensor used to measure the current to charge the vehicle is SCT-013 split core current transformer. This sensor measures the intensity of a current that crosses a conductor without needing to cut or modify the conductor itself. Using the SCT-013 with a processor such as the ESP32, it is possible to measure the intensity or power consumed by the charging vehicle.

Figure 6: Electrical scheme: SCT013 and wiring diagram with ESP32.

The SCT-013 reads and provides an analog measurement proportional to the intensity that a circuit crosses. The measurement is made by electromagnetic induction - the sensor has a split core that allows the user to turn it on to wrap electrical equipment without having to cut it off. This model provides the measurement as a voltage output, which is preferable to use as the connection is simpler.

5.3.2 Wiring diagram

As the analog input pins of the Arduino can only read positive voltages, it is necessary to modify the output voltage of the sensor as it can provide negative values. This was done by this summing a DC voltage offset of 2.5 V in the signal and placing two resistors to provide a middle voltage point between GND and Vcc.

The wiring diagram of the CT is presented in Figure 7.

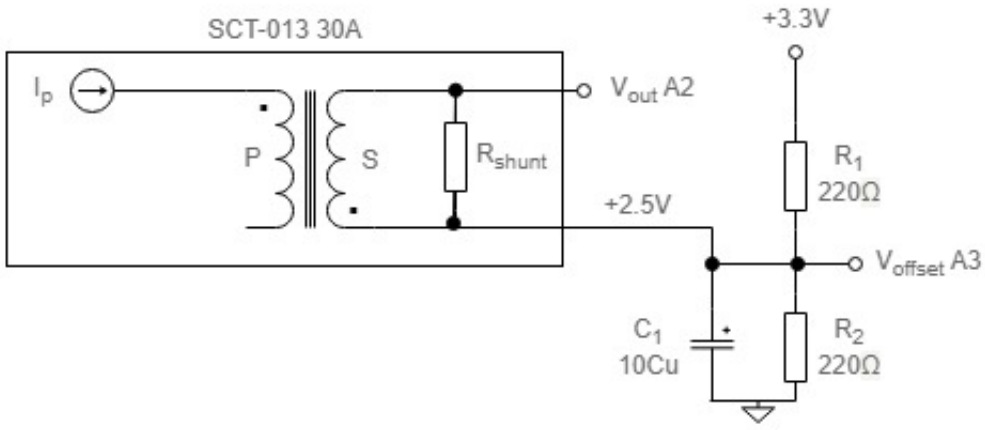


Figure 7: Electrical scheme: SCT013 and wiring diagram with ESP32.

5.3.3 Signal conversion

The CT is connected to 2 analog inputs of the ESP32: A3 is taken as a signal-reading pin and A2 as a reference pin for the voltage. As the necessary output needed is the power consumed, these signals need to be converted to a digital value, which in a later stage will be modified to find the root mean square (RMS) value.

Similar to the signal transformation process for the proximity measurement sensor, there are several values that help understand the analog values read:

- The voltage output of the ESP32, 3.3V (in code: ESP VOLTAGE);
- The analog-to-digital accuracy of the ESP32, 4095

The conversion is done according to the formulas:

$$VoltageRead = AnalogValueRead \frac{ESP_{VOLTAGE}}{ADC_{SOLUTION}} \tag{2}$$

$$VoltageRef = AnalogValueRef \frac{ESP_{VOLTAGE}}{ADC_{SOLUTION}} \tag{3}$$

The converted values are displayed in volts and are now ready to be further transformed to calculate the power consumption by the vehicle.

6 Workflow

This section serves the purpose of presenting the workflow of the project, as seen in Figure 8. Nevertheless, further explanation of the communication protocol between the participants and the data flow will be provided in section 7.

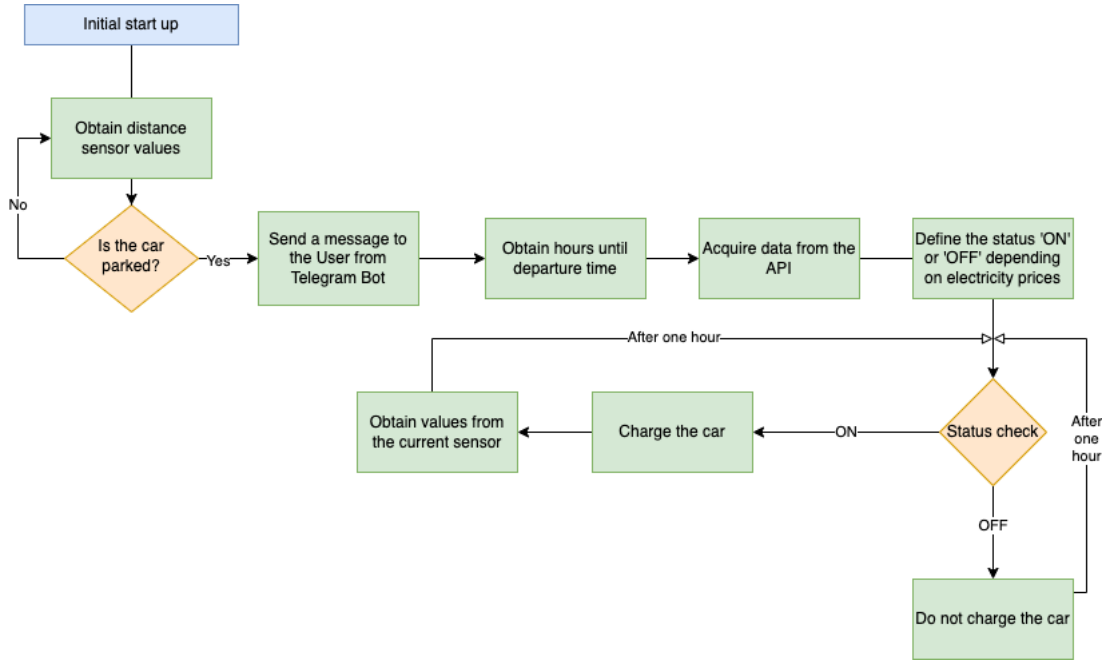


Figure 8: Workflow chart.

7 Communication Scheme

In this section, the analysis of the communication scheme and data flow between each participant will be carried out. There are 7 participants in the general system setup that interact with each other in different ways:

1. **User**, or the driver of the EV and owner of the system;
2. **Load**, or the EV;
3. **Microcontroller**, or the ESP32 FireBeetle;
4. **Computer**;
5. **Sensors**, the set of sensors explained in section 5;
6. **Cloud**, where API data on energy prices is retrieved;
7. **Control device**, or the relay;

The communication scheme between the participants is presented in figure 9.

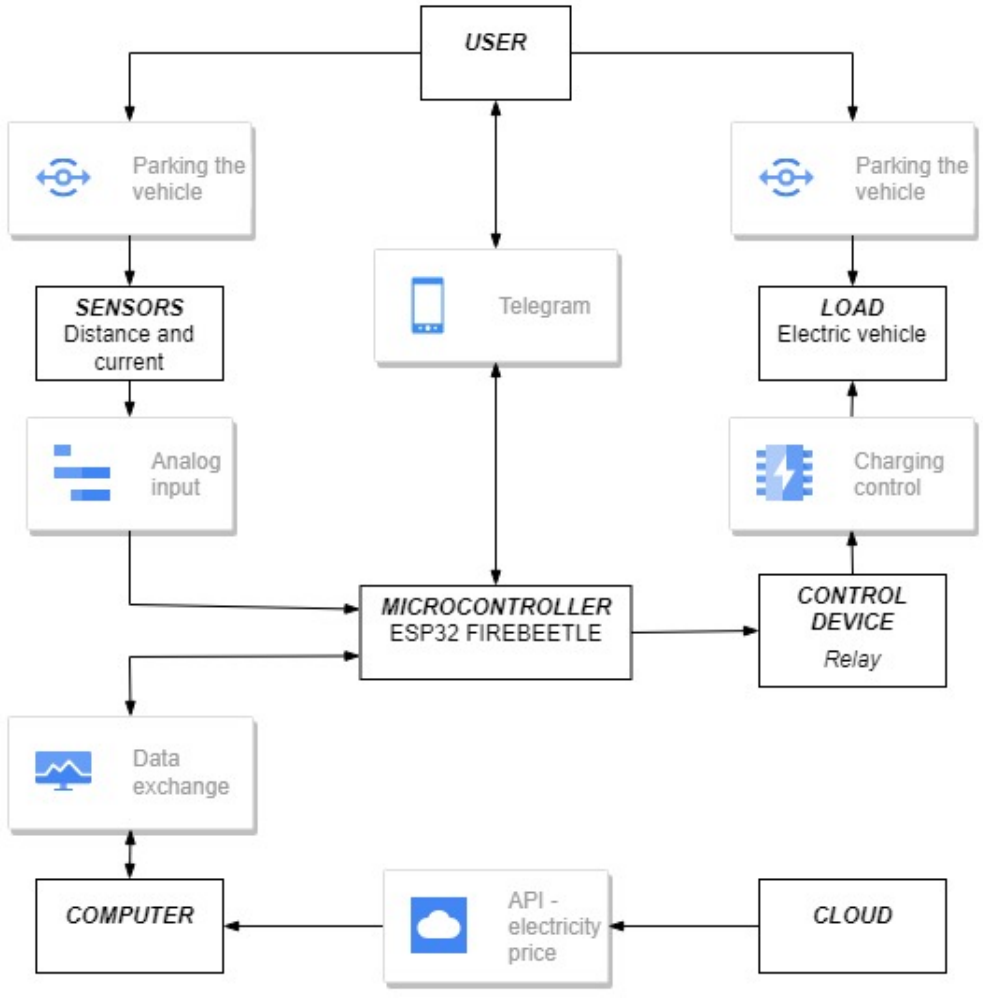


Figure 9: Communication scheme between each participant in the system setup.

In the following part, the interaction between each participant is analyzed.

7.1 User

The User plays the initial role of triggering the system by parking the EV in the selected parking spot, where a distance sensor is installed.

7.1.1 User to distance sensor

The distance sensor returns values between 0-500cm. Having a vehicle parked on the spot will return a value of 0cm, which triggers a Telegram bot to interact with the User regarding the amount of time the vehicle will stay parked. A variable of type bool is defined to store information on whether the car is parked or not so that the sensor will stop measuring the distance after the car is in the parking spot.

Code section: Sensing a car presence

```
bool stopPrint = false;
if(stopPrint == false){
  Serial.print("Distance: ");
  Serial.print(distCar,0);
  Serial.println(";");
  delay(2500);
  if(distCar==0) {
    stopPrint = true;}
```

Once a zero value is detected, a Telegram Bot is activated establishing interaction with the user.

7.1.2 User to Microcontroller

A new Telegram Bot was created, @ChargingAssistant, which can be controlled by a single user ID. The purpose of the bot is to collect information regarding the parked time of the car and send this information to the Microcontroller, which will send it to the Computer for the purpose of API data retrieving.

The following code to establish User-Microcontroller communication was used:

```
bool carDetectedOld = false;
bool carDetectedNew = false;
if(distCar==0){
    carDetectedNew = true;
}
// Sending a message from the bot to the User
if(carDetectedNew != carDetectedOld){
    bot.sendMessage(CHAT_ID, "Your car is now parked. How many hours will you
    stay at home?", "");
    carDetectedOld = true;
}
// Sending a message from the User to the bot
if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
```

```
14      while(numNewMessages) {
          bot.sendMessage(CHAT_ID, "Thanks, enjoy your stay at home!");
16        handleNewMessages(numNewMessages);
          numNewMessages = bot.getUpdates(bot.last_message_received + 1);
18      }
        lastTimeBotRan = millis();
20    }
```

7.2 Microcontroller

The analog sensors used send data to the Microcontroller, which processes it and triggers further actions, such as API data retrieving or switch control.

7.2.1 Distance sensor to ESP32

The communication protocol was already explained in sections 5.1.3 and 7.1.1.

7.2.2 Current transformer to ESP32

When the vehicle is charging, the CT measures the current through the charging cable. The analog signal is processed by the ESP32 according to the process explained in 5.3.3. To avoid big mistakes in readings, a time interval of 5 seconds was defined in which the average values of the readings from A3 and A2 were calculated. These values are used to calculate the instantaneous current and its RMS value, and the filtered current every 20ms.

Due to the length of the Arduino IDE code executing this, it can be found in the Annexes, section A.1.

7.3 Microcontroller

The ESP32 plays an important role in receiving, processing and sending data to the rest of the scheme participants. It is also the device that controls the relay to enable or disable power flow to the Load.

7.3.1 ESP32 to Computer

The ESP32 sends the following data to the Computer:

- Distance between the car and the sensor, only when equal to zero;
- User input to Telegram, in the form of number in hours;
- Current, flowing through the charging cable.

The Computer uses the zero distance input as a trigger to retrieve API data from the cloud regarding energy prices for the next X hours (depending on user input) and creates a charging schedule for the vehicle.

To establish communication with the ESP32, a simple code in Python is used:

```
# Creating communications object with Arduino using Serial
arduino = serial.Serial('COM6', 9600)
print("Communication established.")
```

For further analysis, the code can be found in the Annexes, Section A.2.

7.3.2 ESP32 to Relay

The ESP32 can assign either HIGH or LOW voltage to the digital pin to which the Relay is connected to, which will either activate or de-activate the switch, as explained in section 5.2.3.

The following code written in Python to control the Relay was used:

```
if (Serial.available() > 0) {
  relayControl = Serial.read();
  if (relayControl == 'H') {
  digitalWrite(RELAY_PIN, HIGH);
  else if (relayControl == 'L') {
  digitalWrite(RELAY_PIN, LOW);
  }
```

8 Results

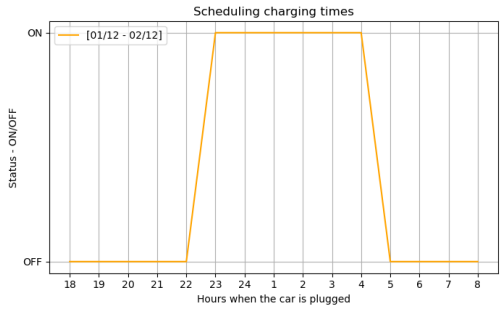
This section discusses the project's outcomes, such as the scheduling charging time, the help provided to allow peak-shaving and an analysis of the economic benefits.

8.1 Scheduling charging time

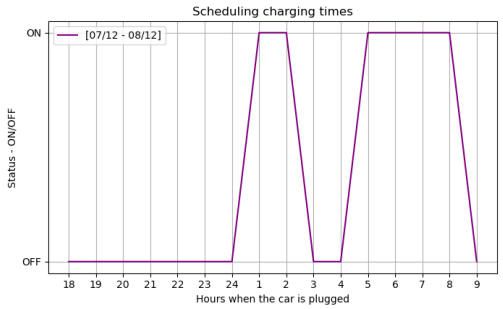
One of the aims of the developed smart plug is to help customers save money on their electricity bills. The electricity prices are retrieved from REE API once the user arrives home, as explained in Section 7.3.1.

The schedule of the charging times is made accordingly to the cheapest six hours between the time the user arrives home and leaves the house as explained in Section 3. In the following cases, 18:00 was assumed as the arrival time and 08:00 of the following day as the departure time.

Figure 10a and figure 10b show the scheduled charging time for two different days of the month of December 2022 that was taken as a sample. The difference between the scheduling depends on the electricity prices, as can be seen from the tables 3a and 3b.



(a) Scheduling between 01/12/22 and 02/12/22.



(b) Scheduling between 07/12/22 and 08/12/22.

Figure 10: Charging time scheduling between 18:00 and 08:00.

Hour	Price [€/MW]	Status
18:00	351.30	OFF
19:00	354.90	OFF
20:00	359.34	OFF
21:00	347.31	OFF
22:00	289.53	OFF
23:00	277.12	**ON**
00:00	279.07	**ON**
01:00	281.67	**ON**
02:00	282.55	**ON**
03:00	287.80	**ON**
04:00	289.22	**ON**
05:00	293.89	OFF
06:00	290.26	OFF
07:00	295.79	OFF
08:00	313.02	OFF

(a) Scheduling between 01/12/22 and 02/12/22.

Hour	Price [€/MW]	Status
18:00	401.66	OFF
19:00	398.20	OFF
20:00	386.29	OFF
21:00	381.29	OFF
22:00	334.85	OFF
23:00	327.66	OFF
00:00	273.87	OFF
01:00	260.96	**ON**
02:00	260.53	**ON**
03:00	263.89	OFF
04:00	263.55	OFF
05:00	263.33	**ON**
06:00	260.79	**ON**
07:00	257.37	**ON**
08:00	254.55	**ON**

(b) Scheduling between 07/12/22 and 08/12/22.

Table 3: Electricity prices and correspondent charging time scheduling between 18:00 and 08:00.

8.2 Peak shaving

One advantage of the product is to help peak shaving, as it can be seen in figure 11. In fact, without the smart plug, the car would charge as soon as it arrives, a period of high demand. Having a different scheduling for charging helps shifting the load to the hours with lower demand, helping the grid and avoiding overloading. Nevertheless, is important to highlight that, even if improbable, there might be some days when the hours with the lowest electricity prices are also the ones with higher demand. In that case, the algorithm would not help peak shaving but only working on load shifting, looking for the cheapest hours.

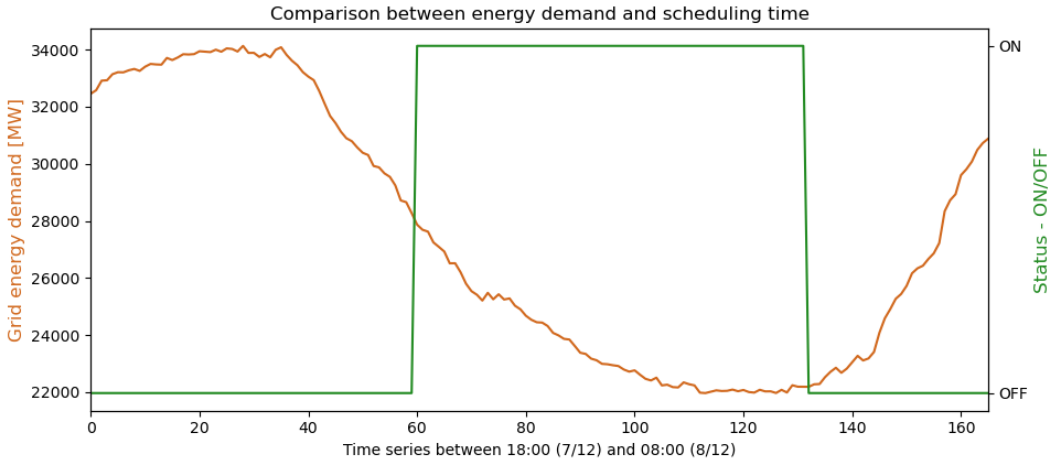


Figure 11: Example of how the smart plug helps peak shaving. Sample day: 07/12-08/12.

8.3 Economic Analysis - REE API

Another advantage of the product can be verified when analyzing the difference between the monthly electricity bill with the smart plug and without. As before, the analysis will focus on the month of December 2022.

The economic analysis was made considering multiple assumptions. Firstly, the customer's arrival and departure time, were considered to be always the same during the course of the whole month, at 18:00 and 08:00.

As shown in figure 12, the difference between the two scenarios is not always constant. Over the whole month, it was found that the monthly electricity bill for EV charging without the smart plug is around 162,6 €, while using the smart plug it would be around 106,8 €. This would sum up to a saving of around 55.7 €, 34% of the initial cost.

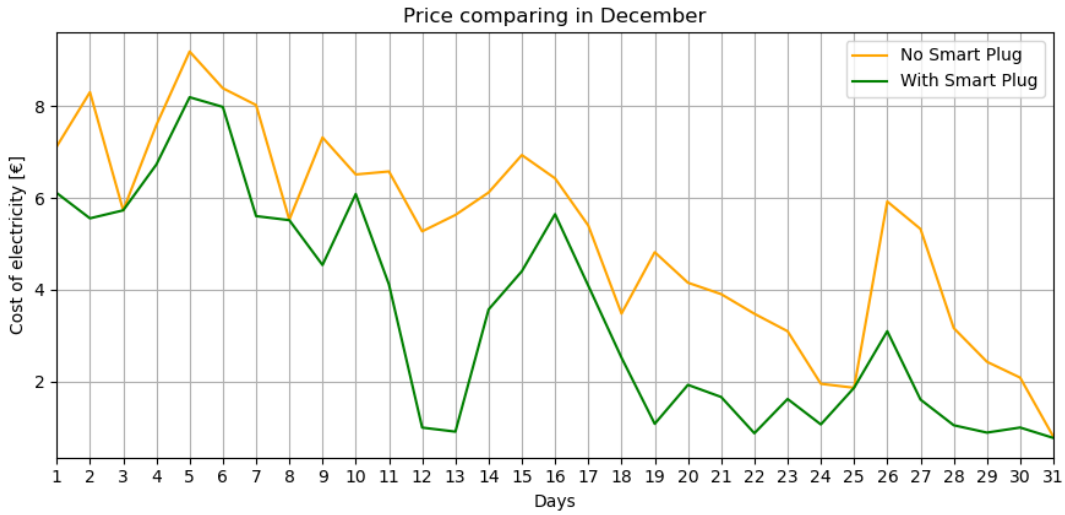


Figure 12: Comparing the price of electricity in December in the two different scenarios.

9 Discussion

This section will highlight the challenges faced during the course of the project, the actions taken to overcome them, and the consequential learning. Additionally the limitations and the improvements that can be carried on in the future. Finally, the business model will be discussed.

9.1 Challenges and Learning

The main challenge faced was the inability to test the product on a real EV. This resulted in many assumptions, such as the type of car used and the consequential energy fed into the vehicle and the total charging time. Testing the product on an actual EV would have ensured the reality of the data and the scheduling of the charging time. The problem was overcome by simulating the EV with a power bank and finding average data on the energy and charging time for a specific type of car, as highlighted in Section 3.

Another challenge that was encountered was the fact that the latest energy price that can be acquired from the API used is at midnight of the same day of the acquisition. This results in the impossibility of testing the algorithm continuously during the night. The problem was overcome by doing tryouts during the day. Nevertheless, this is definitely something that need to be improved for better usage of the product.

Finally, the group faced a lack of previous knowledge in the field, so a major learning was the understanding of the communication protocols between the different devices and data transfer.

9.2 Improvements on the System

As already mentioned in Section 9.1, the data acquisition from the API is a limitation due to the fact that the latest provided electricity price is at midnight. In order to improve the algorithm, it could be interesting to forecast the unavailable electricity prices until departure

time, based on the latest week or the latest month. This could be then backed up by requiring electricity prices from the API once more at midnight.

Another improvement would be to store the values acquired from the current sensor, in order to develop a better economic analysis. Additionally, data visualization could be improved by creating a user interface where the monthly reports of customer savings could be shown.

Finally, the main limitation of the developed code is that it can run just once; in fact, if the car leaves the garage it would be necessary to run the code again to make the process work. The improvement that could be done would be to do a continuous check on the distance sensor and when the distance value is again different from zero (the car left the garage), activate a reset function that would allow repeating the process without manually running the code again.

9.3 Business Model

A smart EV charging demand-side management software must take into consideration the creation of a sustainable and profitable business model that can also enable the widespread adoption of individual electric vehicles. The model that will be presented will consider both the customer needs and the business costs, and provide a clear path for profitability while providing incentives for customers to use the EV charging infrastructure. Ensuring that this software tool is able to generate sufficient revenue to remain viable is of the utmost importance. Additionally, it will take into account the associated costs, revenue streams, a description of the different customer segments, key partners, and potential opportunities to increase efficiency and reduce expenses, in order to move to the next stage - scaling up.

9.3.1 Value Proposition

The key value proposition of this smart EV charging demand-side management tool is the optimization of the cost of charging electric vehicles by actively managing the demand, and therefore ensuring that EVs are charged when electricity is most affordable. This tool can also help mitigate peak demand charges and reduce the strain on the electricity grid. Additionally, this tool is capable of providing additional features such as real-time notifications and reporting.

9.3.2 Customer Segment Profile

Analyzing the customer segment target is extremely important when developing an effective business model. Taking into consideration that the our software tool is provided as a service, different customer profiles were identified. In the following table 4, a summary of the customer segments is presented.

Table 4: Customer segment profiles.

Customer Segments	
Residential Costumers	- Homeowners - Multi-family Dwellings
Business & Commercial Customers	
Utility Companies	
Public Sector	

This demand-side management tool has a diversified group of customer profiles. The business strategy that will be used is a segmented market model strategy. Segmented diversified market strategy refers to a business model that targets specific customer segments with specialized products and services. In this case, by segmenting the market into different subsets of customers with similar needs, from residential customers to the public sector, it is possible to develop a tailored service strategy that matches each customer specific interests. Below a more detailed description of each market segment is presented.

1. **Residential Costumers:** This segment considers homeowners and multi-family dwellings (i.e. apartment buildings) that already have or are considering installing a smart electric vehicle charging station in their homes. These consumers are mostly motivated by convenience, the desire to save money on fuel costs or reduce their overall carbon footprint. The regulatory framework for EV owners in Spain also promotes the growth of this segment. By outsourcing a smart charging EV tool, they are able to reach their goals.

2. **Business & Commercial Costumers:** This segment is mostly motivated by providing their customers and employees with more convenient charging options, reducing their energy costs, improving the efficiency of their EV fleets, or taking advantage of the data analytic capabilities to better understand their energy use. Also taking advantage of the environmental benefits and government incentives associated with EVs. This smart charging software tool ensures the safety and reliability of their EVs fleets by ensuring the proper charging of batteries and preventing overload.

3. **Utility Companies:** this segment has a high motivation for better managing their electrical grid and reducing demand for power during peak hours. This smart charging EVs software tool allows the utility company to more accurately monitor and manage the charging of EVs, and this way prevents power outages and reduces the cost of electricity for their clients.

4. **Public Sector:** the main motivation for the public sector to acquire this smart EV charging software tool is to improve the management of the growing demand for EV charging infrastructure in cities and municipalities. This tool allows governments to track and optimize the charging of EVs, and this ensures that the current infrastructure is being efficiently used. Additionally, this tool helps governments to identify and fast manage future problems, allowing them to respond quickly to changes in the EV charging landscape. Finally, it also aligns with European and National environmental goals, as a way of reducing GHG emissions, improving air quality and supporting public health.

By diversifying the market segments, this business model also intends to increase market penetration; increase customer loyalty, by offering tailored solutions; reduce the risk of over-reliance on a single market segment, allowing the risk to be spread by multiple segments; increase profitability; and promote brand awareness and recognition, by creating a strong brand identity.

9.3.3 Cost Structure

The cost structure of the smart EV charging demand-side management tool is based on the type of service that is intended to be offered. Figure 13, shows a representative graph of the cost structure associated with our business model.

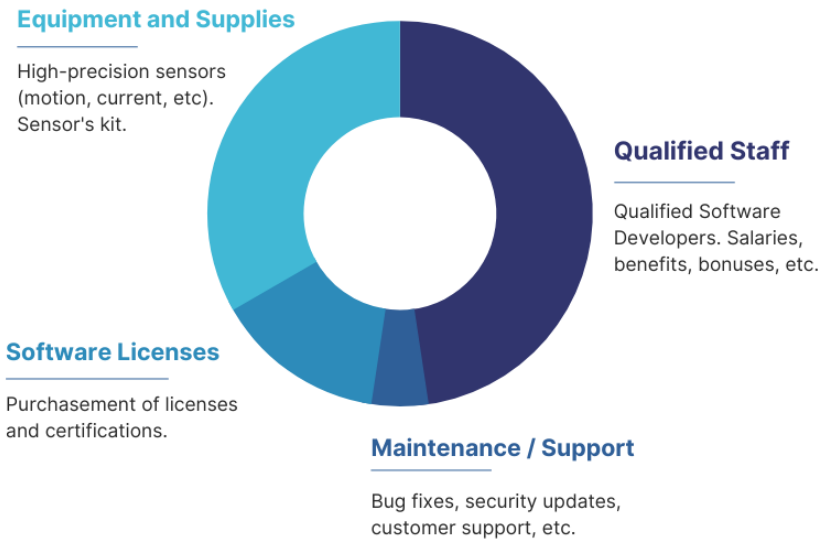


Figure 13: Cost Structure for the smart EV charging demand-side software tool.

This business model focus on providing a smart charging software tool as a service. Therefore, the main costs should be associated with software development. To have competitive software, it is crucial that the main investment goes to qualified staff, so most software developers, but also accounting and legal teams, and perhaps third-party consulting services. Since the software tool relies on a distance and current sensor to perform in the most accurate way, it is crucial to have high-precision sensor suppliers. The purchase of these sensors and the consequence installation represent a significant percentage of the company's costs.

Lastly, software licensing and ongoing maintenance and support of the system is a relevant part of the company's costs. The purchase of licenses to use certain software programs and platforms, and the performance of customer support, such as bug fixing and security updates represents costs for the company.

9.3.4 Revenue Stream

Revenue streams are a central part of all business models because they are the sources of income for a business. Revenue streams are the ways to pay investment and operational expenses and reinvest in the company. In this business model, the streams come in form of subscriptions-based services, per-charge fees, data monetization services and utility partnerships.

Figure 14 shows the different potential revenue streams according to each customer segment.

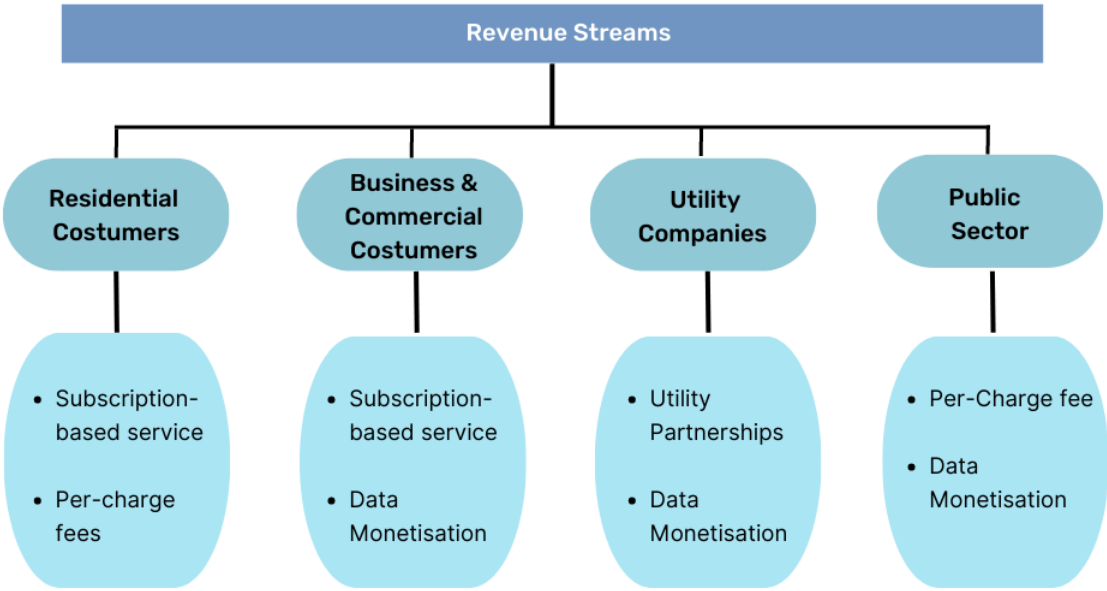


Figure 14: Revenue Stream Scheme, according to each customer segment.

Identifying and managing the right revenue streams is essential for the success of a business. As follows, a brief explanation of each revenue stream is presented.

- **Subscription-based service:** charging station owners (or EV charging station companies) can subscribe to this smart charging software tool and are charged a monthly or annual subscription fee for continuous access to the services.
- **Per-charge fees:** charging station owners (or EV charging station companies) pay an additional fee for each charge that is completed and optimized by the software.
- **Data Monetisation:** by collecting the data about the user charging habits, it is possible to sell it to utilities or to the public sector. This data can help to better understand their clients charging patterns and optimize the grid and the public infrastructure.
- **Utility Partnerships:** charging station owners that use this smart software tool can partner with their local utility to offer customer discounted rates. When partnerships like this happen, a percentage of those contracts is paid directly to the company.

The present business model intends to have multiple revenue streams. With this, it is possible to share the risk, and understand which activities are more profitable and re-invest in those. With different sources of income, it is possible to better manage the cash flows and plan future expenses.

9.3.5 Scaling up

Scaling up is a natural expected step of any business. The timing depends on the goals of the business model and the resources available. The process of growth can be achieved through careful planning and strategic decisions. In the first stage, it is intended to increase the clients' portfolio, and introduce new updates and extra services to the smart charging software tool. Optimizing the streamlined processes, like the automation of certain aspects of the business, for example, sensor updates.

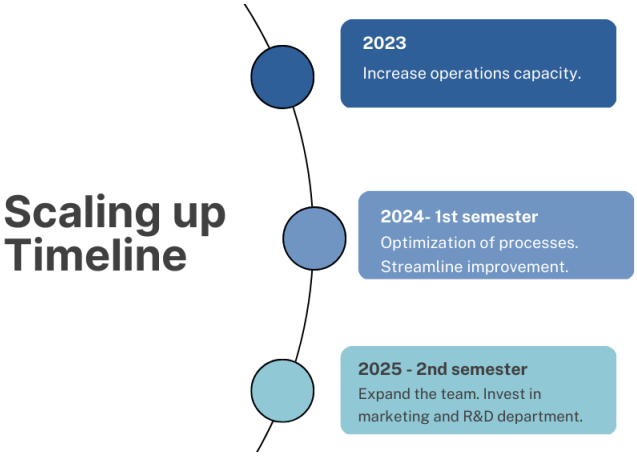


Figure 15: Scaling up - timeline.

Once these goals are achieved, the intention is to expand the resources, hiring people with skills and experience to help the growth, while being remunerated accordingly. The creation of a RD department will allow the company to improve its offerings, forecast electricity prices, or offer a monthly report to the user on his savings is some of the main improvements that would be conducted. The assumed timeline is intended to be supported by public and private investment, alongside an expected profitable annual revenue.

10 Conclusion

As initially stated, the purpose of building the smart charging station was to initially reduce the electricity bills in households, as well as contribute to peak-shaving by charging the EVs in selected hours with low energy demand and lower electricity prices. Being able to test the setup in a scaled-down environment, it can be concluded that it does have a potential for a scale-up in the initially stated situation, such as a household. At the moment, the project faced plenty of limitations due to the short time frame of project execution and the limited resources (such as not having an EV to test on or using low-quality sensors that produce errors when measuring or malfunctioning). However, seeing the results produced by the charging schedule (Section 8.1) accompanied by the economic analysis (Section 8.3), it does confirm that scheduled charging reduces the monthly electricity bill by EUR 50, which is a solid number. At the same time, looking at Section 8.2, it can be seen that the algorithm not only selects the lowest-price hours but also the hours when there is lower energy demand, so the power grid benefits as well. To conclude, although the project is based on plenty of assumptions, the results show that there is a potential for implementing it in the suggested initial environment and testing it with an actual EV and improved API data can produce satisfactory results.

References

[1] European Comission. "COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE EUROPEAN COUNCIL, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE COMMITTEE OF THE REGIONS". In: *SWD(2022) 230 final* (2022).

[2] Andy Ford, Aaron Gillich, and Pegah Mirzania. "28 - Sustainable Energy and Energy Efficient Technologies". In: *Future Energy (Third Edition)*. Ed. by Trevor M. Letcher. Third Edition. Elsevier, 2020. URL: https://www.sciencedirect.com/science/article/pii/B9780081028865000281.

[3] *Guide to the Nissan LEAF 2018*. Aug. 2022. URL: https://pod-point.com/guides/vehicles/nissan/2018/leaf.

[4] Scrum.org. *What is a product backlog?* 2022. URL: https://www.scrum.org/resources/what-is-a-product-backlog.

[5] *Levels of a risk matrix*. June 2022. URL: https://www.vectorsolutions.com/resources/blogs/levels-of-a-risk-matrix/.

A Appendix

A.1 Arduino IDE code

```
///// GENERAL VALUES
#define ADC_SOLUTION (4095.0) // ADC accuracy of ESP32 is 12bit
#define ESP_VOLTAGE (3.3) // 3.3V output of ESP32

///// DISTANCE VARIABLES
#define MAX_RANGE (500) // The max measurement vaule of the module is 500cm
const int signalPin = A0; // Input analog pin

///// CURRENT VARIABLES
// Measure the consumed current by a load with the SCT013-30A CT
// Pins
const int sensorPin = A3;
const int refPin = A2;
const int Rshunt = 33.3;
// Time variables
unsigned long time_now = 0;
unsigned long time1_ant = 0, time2_ant = 0;
unsigned long count = 0;
float sum1 = 0, sum2 = 0;
double Ifilt = 0.0;
// Auxiliary variables
unsigned long time_ant = 0, difTime = 0, act_time = 0;
const int sampleDuration = 20;
int count_integral = 0;
double rawSquaredSum = 0;
double Iant = 0;
// Constant grid frequency (50 Hz)
double freq = 50;
// Transformer reduction relationship
double n_trafo = 1000;
// Measured current variable
double Irms = 0;

///// TELEGRAM VARIABLES
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <ArduinoJson.h>

const char* ssid = "MIWIFI_fXEC"; // Needs to be changed for different WiFi
const char* password = "YUFJYFhf";  // Needs to be changed for different WiFi

#define BOTtoken "5894687503:AAG-kLyVIJFXcaGAJPe864e-8p8cHSzA6xA" // Telegram
    Bot
#define CHAT_ID "1529642773"  // Sara's Telegram ID

WiFiClientSecure client;
UniversalTelegramBot bot(BOTtoken, client);

int botRequestDelay = 5000;
unsigned long lastTimeBotRan;

// Handle what happens new messages from user are received
```

```
53 void handleNewMessages(int numNewMessages) {
   for (int i=0; i<numNewMessages; i++) {
55   String chat_id = String(bot.messages[i].chat_id);
     if (chat_id != CHAT_ID){
57     bot.sendMessage(chat_id, "Unauthorized user", ""); // If other user tries
    to message the bot
       continue;
59   }
     // Print the received message
61   String text = bot.messages[i].text;
     Serial.println(text);
63   String from_name = bot.messages[i].from_name;
   }
65 }

67 ///// RELAY VARIABLES
#define RELAY_PIN D2
69 bool carDetectedOld = false;
bool carDetectedNew = false;
71 bool stopPrint = false;
char relayControl;
73
///// SETUP
75 void setup() {
Serial.begin(9600);
77
///// MOTION SETUP
79 float distCar, sensity_t;

81 ///// CURRENT SETUP

83 ///// TELEGRAM SETUP
// Establishing WiFi connection
85 Serial.print("Connecting Wifi: ");
Serial.println(ssid);
87
WiFi.mode(WIFI_STA);
89 WiFi.begin(ssid, password);
client.setCACert(TELEGRAM_CERTIFICATE_ROOT);
91
while (WiFi.status() != WL_CONNECTED) {
93   Serial.print(".");
   delay(500);
95 }

97 Serial.println("");
Serial.println("WiFi connected");
99 Serial.print("IP address: ");
Serial.println(WiFi.localIP());
101
bot.sendMessage(CHAT_ID, "Welcome home!", "");
103
///// RELAY SETUP
105 // Initialize digital pin as an output
pinMode(RELAY_PIN, OUTPUT);
107 }
```

```
109 ///// L O O P
    void loop() {
111
    ///// M O T I O N  L O O P
113 // Read the value from the sensor:
    float sensity_t = analogRead(signalPin);
115 float distCar = sensity_t * MAX_RANGE / ADC_SOLUTION;
    //Serial.print(distCar);
117 //Serial.println(""); // Value sent to the computer

119 if(stopPrint == false){
      //Serial.print("Distance: ");
121   //Serial.print(distCar,0);
      //Serial.println(";");
123   delay(2500);
      if(distCar==0) {
125     stopPrint = true;
      }
127 }

129 ///// T E L E G R A M  L O O P
    if(distCar==0){
131     carDetectedNew = true;
    }
133
    if(carDetectedNew != carDetectedOld){
135     bot.sendMessage(CHAT_ID, "Your car is now parked. How many hours will you
        stay at home?", "");
        carDetectedOld = true;
137 }

139 // Sending a message from the User to the bot
    if (millis() > lastTimeBotRan + botRequestDelay)  {
141     int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

143     while(numNewMessages) {
          bot.sendMessage(CHAT_ID, "Thanks, enjoy your stay at home!");
145       handleNewMessages(numNewMessages);
          numNewMessages = bot.getUpdates(bot.last_message_received + 1);
147     }
        lastTimeBotRan = millis();
149   }

151 ///// R E L A Y  L O O P
    if (Serial.available() > 0) {
153   relayControl = Serial.read();
      if (relayControl == 'H') {
155   digitalWrite(RELAY_PIN, HIGH);
      }
157     else if (relayControl == 'L') {
        digitalWrite(RELAY_PIN, LOW);
159   }
    }
161
    ///// C U R R E N T  L O O P
163 act_time = micros();
      difTime = act_time - time_ant;
```

```
165   int RawValue = 0;
      if (difTime >= 1000) {
167     time_ant = act_time + (difTime - 1000);

169     // Read the ADC input from the sensor and the voltage reference point
        int ADC_sensor = analogRead(sensorPin);
171     int ADC_ref = analogRead(refPin);

173     // Convert the ADC input measured to voltage values
        double V_sens = ADC_sensor * ESP_VOLTAGE / ADC_SOLUTION;
175     double V_ref = ADC_ref * ESP_VOLTAGE / ADC_SOLUTION;

177     // Calculate the instantaneous current using the voltage diference and the
       burder resistor value
        double Iinst =  n_trafo * (V_sens - V_ref) / Rshunt;
179
        // Calculate the integral
181      rawSquaredSum += Iinst * Iinst * 0.001;

183     // Count 20 ms
        count_integral++;
185   }

187   // Each 20 ms, calculte the RMS
      if (count_integral >= sampleDuration)
189   {
        // Calculate the RMS
191     Irms = sqrt(freq * rawSquaredSum);
        // Counter and integral reset
193     count_integral = 0;
        rawSquaredSum = 0;
195
        // Low-pass filter
197     Ifilt = 0.95 * Iant + 0.05 * Irms;
        Iant = Ifilt;
199
        // Calculate the average power
201     double Pavg = Ifilt * 230.0;
      }
203     // Read time in ms
        time_now = millis();
205     // Each 1 second, measure the A2 and A3 ports
        if (time_now - time1_ant > 1000) {
207       // Increment the time counter
          count++;
209       // Acumulate the ADC measurements each second, to calculate latter and
       average value each 5 seconds
          sum1 += Ifilt;
211       sum2 += (Ifilt * 230.0);
          // Udpate the "1 second" time flag
213       time1_ant = time_now;
        }
215
        /* Each 5 seconds, calculate the average value of the A2 and A3 measurments,
        and the state of the relay output pin
217     and write the values with the serial port using semicolons to separate them
       */
```

```
    if (time_now - time2_ant > 5000) {
      //Serial.print("Current: ");
      //Serial.print(sum1/count);
      //Serial.print("; Power: ");
      Serial.print(sum2/count); // Value sent to the computer
      //Serial.println(";");
      // Reset the variables to calculate the avarage results
      sum1 = 0;
      sum2 = 0;
      // Reset the time counter and update the "5 second" time flag
      count = 0;
      time2_ant = time_now;
    }
}
```

A.2 Python code

```
# Importing libraries
import serial
import requests
import json
import time
from datetime import datetime, timedelta
import numpy as np
import pandas as pd

# API start date definition
start_date = datetime.now()
start_hour = datetime.now().hour

# Hours untill midnight
max_hours_ahead = 24 - start_hour

# API end date definition to see the prices ahead
end_date = start_date + timedelta(hours=max_hours_ahead)

# Transform into strings with a minute resolution
end_date = end_date.strftime('%Y-%m-%dT%H:%M')
start_date = start_date.strftime('%Y-%m-%dT%H:%M')

# Get data from API
endpoint = 'https://apidatos.ree.es'
get_archives = '/en/datos/mercados/precios-mercados-tiempo-real'
headers = {'Accept': 'application/json',
           'Content-Type': 'application/json',
           'Host': 'apidatos.ree.es'}
params = {'start_date': start_date, 'end_date': end_date, 'time_trunc':'hour'}
response = requests.get(endpoint+get_archives, headers=headers, params=params)
data_json = response.json()

# Data forecasted
spot_price_values = data_json["included"][1]["attributes"]["values"]
spot_price = []

```

```
# Create a vector just with price values
for time_period in spot_price_values:
    spot_price.append(time_period['value'])

# Creation of the hours vector
hours_vect = list(range(max_hours_ahead)) + np.ones(max_hours_ahead) * int(
    start_hour)

# Creating a DataFrame with the needed values
# The default status is 'OFF' and will be changed later
data = {'Hour':hours_vect, 'Price':spot_price, 'Status':['OFF'] *
    max_hours_ahead}
df=pd.DataFrame(data)

# Helping variable to enter specific rows of the df
loop_number = 0

# Creating communications object with Arduino using Serial
arduino = serial.Serial('COM6', 9600)
print("Communication established.")

try:
    while True:
        # Specify for how much time the EV needs to be charging
        charging_time = 6

        now = datetime.time
        # READ DATA
        # Check if there is new info from the Arduino and read it
        data_bytes = arduino.readline()

        # Decoding the message into UTF-8
        data_ide = data_bytes.decode("utf-8")
        # Retrieving the value of hours by the User in Telegram
        user_value = int(data_ide)
        # Check if those amount of hours go beyond midnight
        if user_value > max_hours_ahead:
            user_value = max_hours_ahead
        if user_value < charging_time:
            charging_time = user_value

        # Take just the next x hours
        df1 = df.head(user_value)

        # Parsing the dataframe
        for index, row in df1.iterrows() :
            # Looking for the lowest prices (the same amount of charging time)
            if index in df1.nsmallest(charging_time, ['Price']).index.values:
                # Changing the Status from 'OFF' to 'ON'
                df1.at[index,'Status']='ON'

        # Print the df to check the charging schedule
        print(df1)

        # Reset the timer for the new loop
        previous_time = datetime.now()
        starttime = time.time()
```

```
93
        for index, row in df1.iterrows():
95          actual_time = datetime.now()
            # Communication between the Computer and Relay to enable car
    charging
97          if row['Status']=='ON':
                arduino.write('H'.encode())
99              print ('Your car is charging.')
                data_bytes = arduino.readline()
101
                # Decoding the message into UTF-8
103             data_ide = data_bytes.decode("utf-8")
                user_value = int(data_ide)
105             print(user_value)

107         else:
                arduino.write('L'.encode())
109             print ('Waiting for lower electricity price.')

111         # Waiting time before entering another loop, equivalent to 1h of
    real time
            time.sleep(30.0)
113
# Handling KeyboardInterrupt by the end-user (CTRL+C)
115 except KeyboardInterrupt:
# Closing communications port
117     arduino.close()
        print('Communications closed.')
```