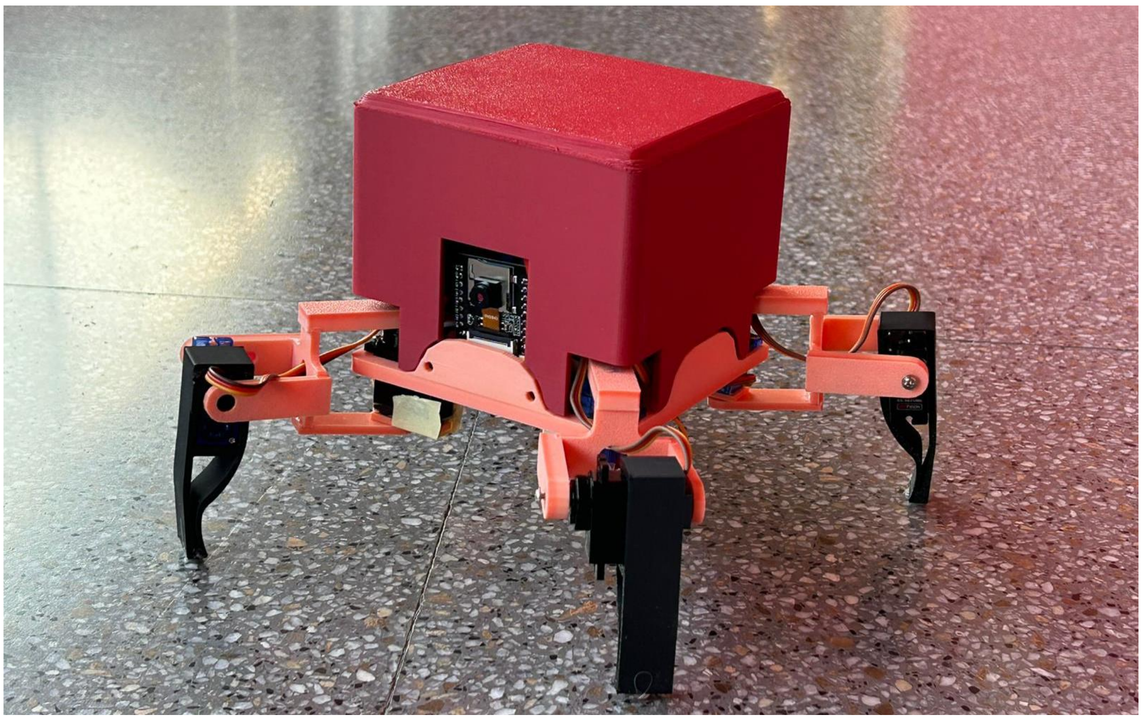
Spider-Robot

Maria Schreiber, Sergi Piguillem, Roger Camps & Òscar Guerrero



Degree in Mechatronics Engineering

Subject: Integrated Projects II

Vic, June of 2025

Table of contents

Summary

Title: Spider Robot

Authors: Maria Schreiber, Sergi Piguillem, Roger Camps & Òscar Guerrero

Subject: Integrated Projects II

Date: Friday, June 6th, 2025

Keywords: quadruped robot, esp32 cam.

Robotic spiders are bio-inspired machines designed to mimic the movements of real arachnids. This project focuses on mechanical and electrical design, to achieve a functional robotic spider with multi-directional movements. It uses servomotors and an ESP32 microcontroller to recreate the movement of a spider. The design consists of four legs and eight servomotors that are assembled to the legs. This will control the movement of the spider; this idea was developed, but the final version operates with four of the eight servomotors. The design consists of four legs and eight servomotors that are assembled to the legs, this will control the movement of the spider, this idea was developed, but the final version operates with four of the eight servomotors, and the initial code was also adapted to achieve the wanted results. Mechanical ~~part~~ were designed and printed, manually they were assembled, the electrical system is developed with Arduino, using an ESP32-CAM board, all the system is powered by a battery. The movement will depend on the command that sends the program, so it is remotely controlled. During the project there are different improvements made to make it work efficiently. This project combines mechanical design, embedded programming, and electronic control.

1. Introduction

The ability of bio-inspired robots to mimic the effectiveness and adaptability of nature has drawn a lot of interest in recent years. Spider-like robots are one such area of research; their multi-legged design allows them to navigate difficult terrain. Potential uses for these robots include environmental monitoring, search and rescue operations, and educational platforms.

The T8X Spider Robot by Robugtix, which has 26 servo motors and sophisticated kinematics, and the OpenDog project, which employs 3D-printed components to simulate quadruped mobility, are two examples of commercial and academic projects that have sought to mimic arachnid locomotion. These solutions are frequently too expensive or complicated, though.

With the use of widely available technology like ESP32-CAM and SG90 servos, this project seeks to provide a low-cost, simplified quadruped spider robot with realistic movement and remote video capabilities. The design improves accessibility for educational and hobbyist applications by striking a balance between affordability, simplicity, and performance.

2. Objectives

Building on the concepts presented in the preceding part, the project's goals were established to guarantee that the finished prototype satisfies requirements for flexibility, control, stability, and effectiveness. Technical choices are guided by these goals at every stage of the development process.

The most important aspect of the project is to make a robot that looks like a real spider, at least that has the shape of a spider to be able to simulate the movement and can also have a vision mechanism to see where it is going to, so that it can be controlled remotely.

Therefore, the main objective of the spider robot is to have a similarity to a spider and have accurate movement remotely control.

Moreover, the specific objectives of it are:

- Four legs spider.
- It has a vision mechanism.
- It has a compact design with dimensions under 20 cm in width and length.
- Have a battery that will have a useful life of at least 30 minutes with continuous operation.

3. Development of the project

This project is divided into three main parts, mechanical, electrical and programming. Each part of the project must remain faithful to the main concept of the project, a quadruple spider, as in the end each part will be integrated to form the complete robotic spider, all the development must be carried out having in mind the remaining areas that will interact with each other. In this section it is going to be explained how each component, mechanical, electronic and software has been developed to achieve a final functional result.

3.1. Analysis of the problem and possible solutions

To start the project there are several problems related to stability, lack of grip, coordination and precise control of the legs that it is necessary to have in mind. One of the biggest problems was controlling all the eight servomotors with the ESP32-CAM, but modifying the software, it could move with only the necessary ones. Another issue was to coordinate all the servomotors to move correctly depending on the wanted direction, related to this was the lack of grip on the legs. This issue makes the spider can move but not from the spot. Trought this section all these problems have been analyzed, and different solutions were made to achieve efficiency and correct function of the spider.

3.2. Description of the design

The spider robot project's design phase is essential since it establishes the composition, operation, and integration of every subsystem. The mechanical, electrical, and control components that comprise the robot are explained in detail in this section. The goal is to create a small, light, and sturdy robotic platform that can move like a real spider and can hold all the required parts, including servomotors, the ESP32-CAM, and the battery system.

During the design phase, modularity, ease of manufacture, and effective use of available space were prioritized. To optimize part shape and guarantee appropriate fit and performance, 3D modelling and simulation techniques were employed. The finished prototype is appropriate for testing as well as practical uses since it strikes a balance between form and function. From the mechanical framework to the software architecture that controls its behavior, each subsection will explore the main facets of the robot's structure.

3.2.1. Mechanical design

The spider robot's mechanical design is centered on building a small, light, and sturdy structure that can sustain its own weight while moving dynamically. The design incorporates a central body that is optimized for component placement, articulated legs with several degrees of flexibility, and a protective shell for both safety and aesthetics, all of which are inspired by the anatomy of actual spiders. Every component was thoughtfully created to guarantee modularity, simplicity of assembly, and effective communication with the control and electrical systems.

3.2.1.1. Mechanical parts

Body:

We have done this body, because it is not heavy and there is the perfect space to put all the electronic components and space for all the wiring. In addition, there are holes in the base, for hexagonal separator to make another base above the battery to put the rest of the components.

Also, to place the leg's, the better option was to put them in each corner of the spider, because it allows to obtain better stability to make the movement and the leg's will have the necessary space to complete the whole movement. Next, we put the camera (ESP – 32 Cam) in front of the spider, because from there it is perfect to have a better vision of the environment, and the camera is set there on a slide.

Despite this, there are four square holes in two sites of the base, to pass all the servomotors wiring under the spider to be more hidden and to have all the wiring much tidier.

Finally, the big space in the middle of the spider is used to put the battery to supply all the power of the components.

All the components of the spider are organized to be all the weight of the components in the middle of it is body to have the center of gravity there and not to have problems when all the legs are moving.

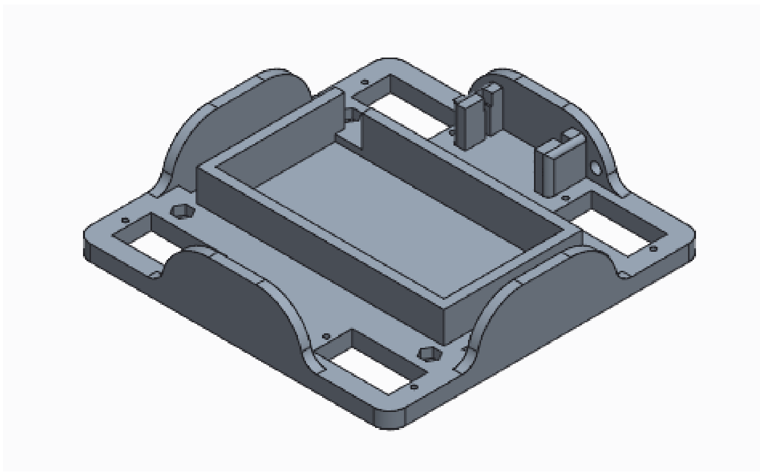


Image 1. Spider body

This is the base of the spider, it is a (100 x 100) mm size, this is the minimum possible to have enough space for all the electronics.

Legs:

The legs have four distinct designs since they are made up of six components, two of which are duplicates. Two servos actuate each leg, enabling multidirectional movement. The robot's adaptability and effective force transfer from the servos to the ground are provided by this design.

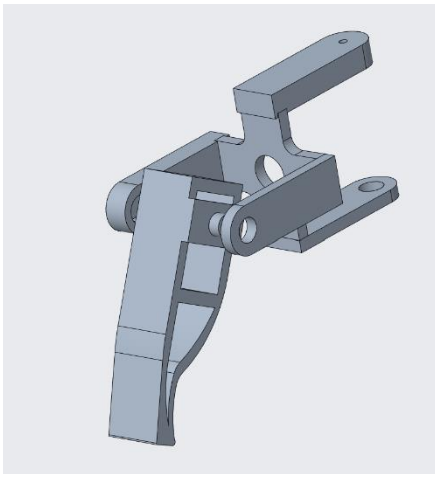


Image 2. Spider legs

The leg structure is divided into two main parts:

- **Femur**: Responsible for movement along the X-plane. It connects the inner servo to the outer section.
- **Tibia**: Makes contact with the ground and is connected to the outer servo, which allows movement along the Y-plane.

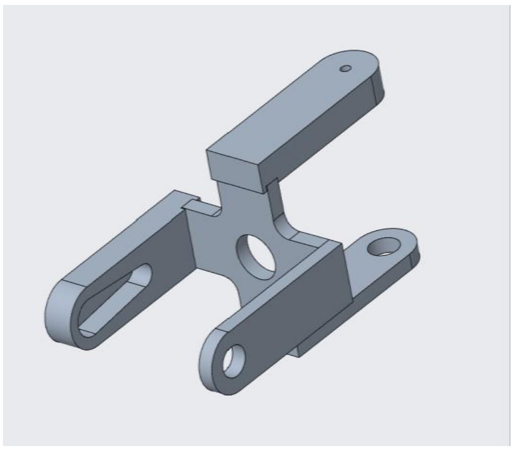


Image 3. Femur

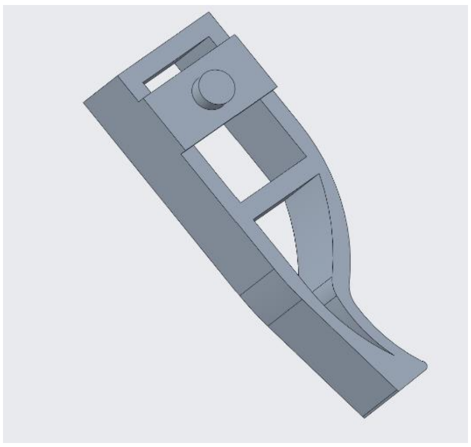


Image 4. Tibia

This combination results in robust legs with a full range of motion, capable of supporting the robot’s body effectively.

Servo - leg supports:

These supports are used to attach the legs to the servos mounted on the body. Their main function is to keep the leg aligned beneath the servo axis. Since the upper side of the femur is connected to the servo horn, the support ensures that the bottom connection rotates along the same axis, maintaining structural consistency and movement precision.

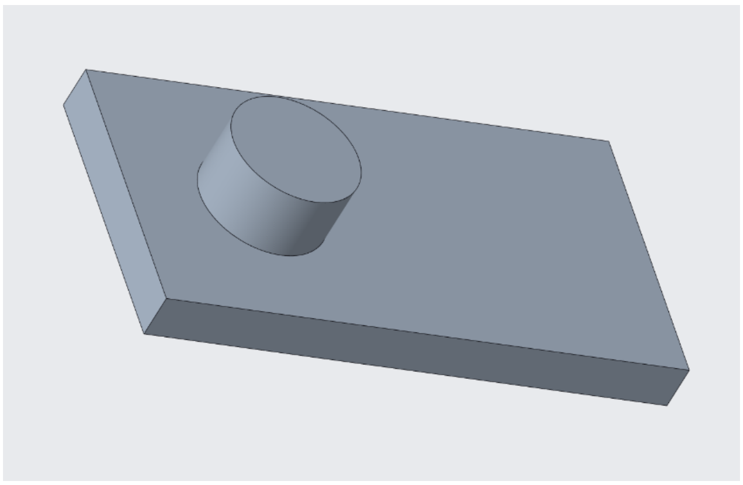


Image 5. Servo supports

Shell:

The shell is designed to cover and protect all the internal electronics while providing a clean and aesthetic external appearance. It fits perfectly on top of the base, ensuring a secure and seamless integration with the rest of the structure.

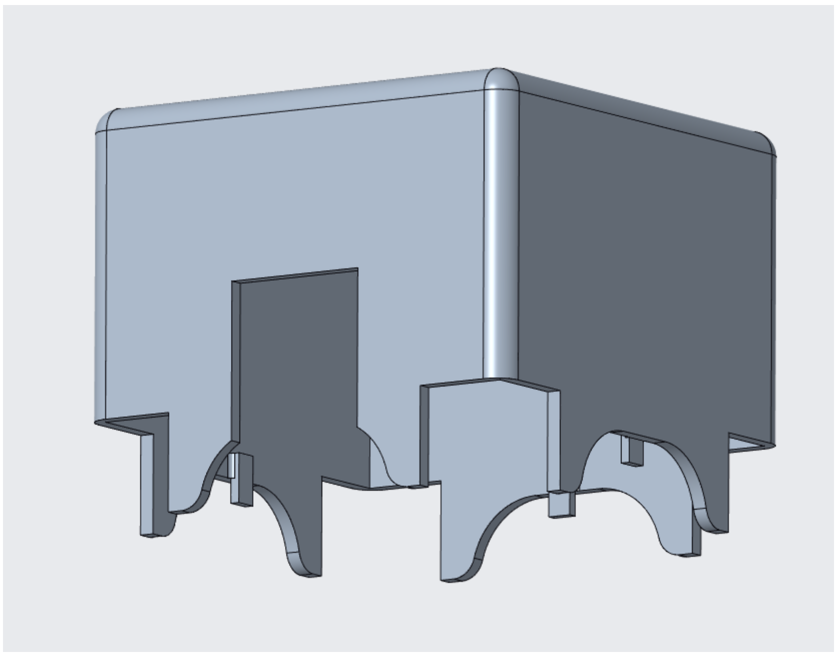


Image 6. Shell

Spider Robot:

The complete spider robot is composed of three main structural elements: the base body, four articulated legs, and the top shell.

Each part is carefully designed to integrate with the others, resulting in a compact, stable, and functional robotic system. The legs are mounted at the corners of the body to ensure maximum balance and mobility, while the central positioning of the battery and electronics contributes to a low and centered center of gravity. This configuration improves performance during locomotion and minimizes the risk of tipping or imbalance during complex movements.

The shell not only protects the internal components but also enhances the visual appeal of the robot, giving it a polished and professional look.

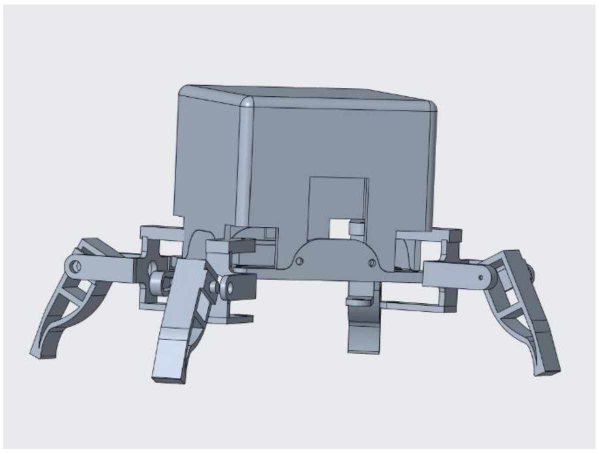


Image 7. Spider robot

3.2.1.2. Material selection and manufacturing method

Possible materials to manufacture.

These are the three possible materials to manufacture to do the construction of the spider.

- Wood → Thin Wood and Oak Wood.
- Steel
- 3D print → They are several types of materials for threading to be used.

These three materials are the most used for manufacturing this type of project. Because these are the three best materials to be manufactured easily and all have a good resistance to weight.

Wood

There are two types of wood to use:

- Thin Wood: This wood isn't very heavy compared to other types of wood, this wood can be useful, because it isn't heavy and it would be easy to modify to make the different parts of the spider robot. But the problem is that it is not as flexible as we need to manufacture each part of the spider.
- Oak Wood: The oak wood makes great stability for the spider and it's almost impossible to break the different parts of the spider if this wood is used. But using all the structure with this wood will make the spider robot much heavier and will make it difficult to do all the movement for the servomotors.

Steel

- Using steel isn't a great idea, because it is a little difficult to do all the different parts of the spider's (body and legs. Also, it will make it heavier, but not as much as oak wood.

3D Print

- Doing all the elements of the spider robot using 3D print, it will make not very heavy and very flexible to manufacture all the diverse parts, because we can use this application, PTC Creo, that is possible to do all the different parts easily and print it. Additionally, 3D print has the necessary stability, because all the pieces done with 3D print are enough resistant.

In conclusion, using steel would be a great idea if it was not difficult to manufacture compared to the rest of materials, moreover, it would be heavy to move it with servomotors.

On the other hand, using wood would have the same problem as steel, because it is heavy, but not as difficult to manufacture. Using thin wood will be good to use for adding new parts.

Finally, the material chosen, for all the possible flexibility to design and manufacture, also the weight of it, is to use 3D Printing, specifically the material PLA, because it is one of the materials they have, and it is tough to have imperfections on the outline to be have more precision. To design all the distinct components, we will use PTC Creo, because it is the application used in previous subjects.

3.2.1.2. Technical drawings

Base:

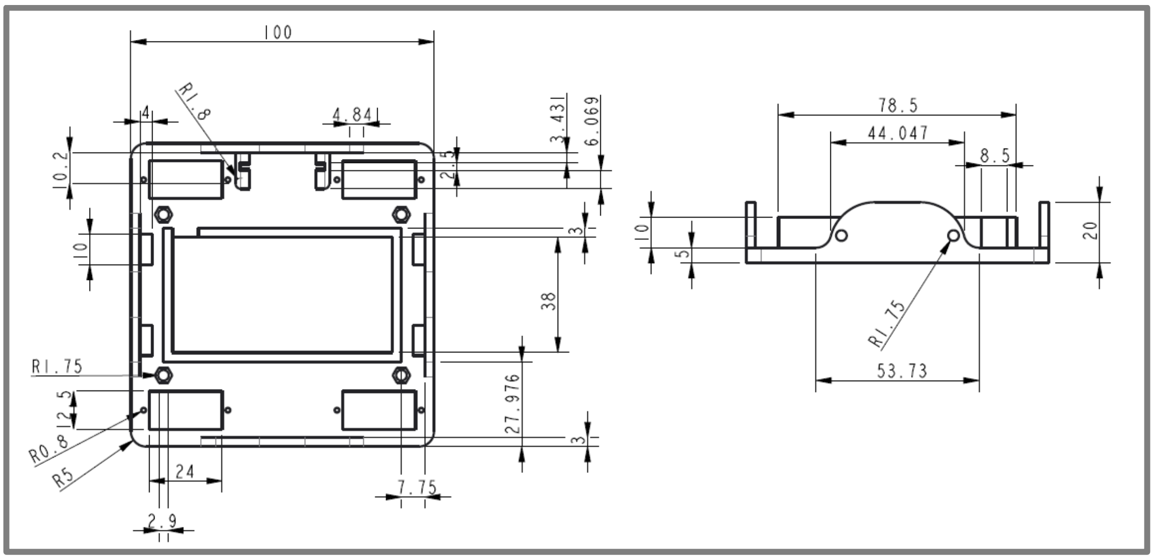


Image 8. Spider base plans

Servo – Leg Suport:

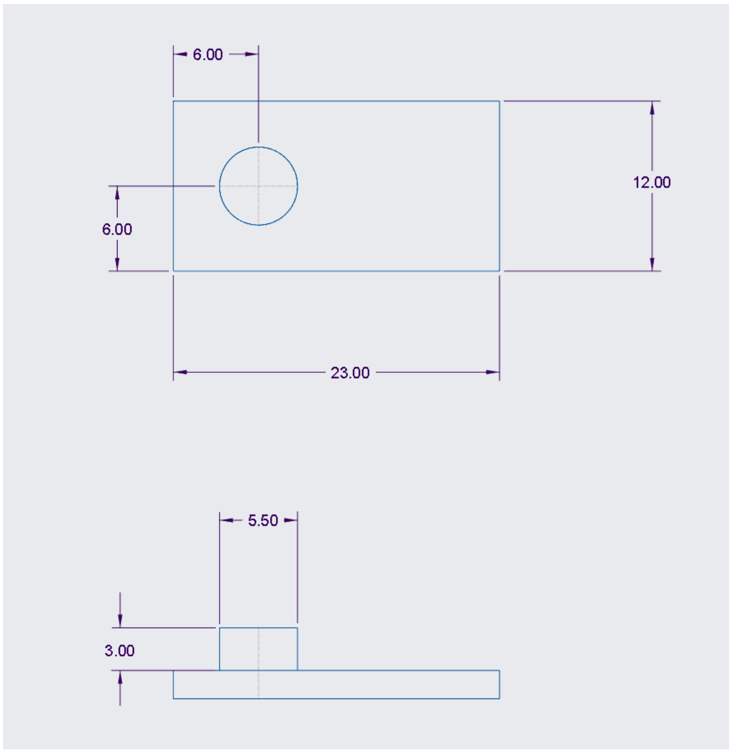


Image 9. Servo legs support plans

Leg Parts:

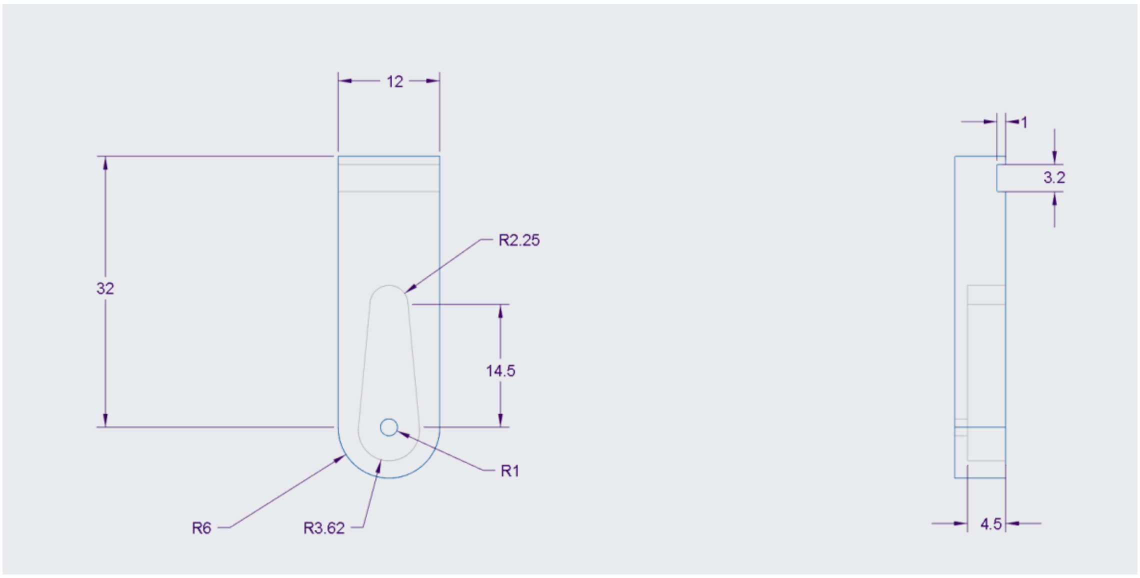


Image 10. Leg's plans

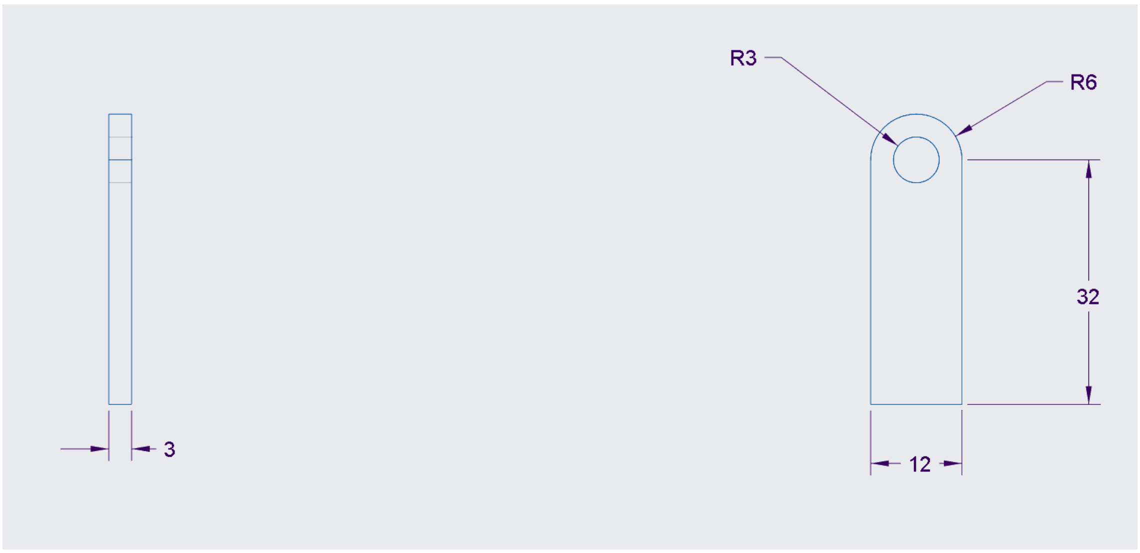
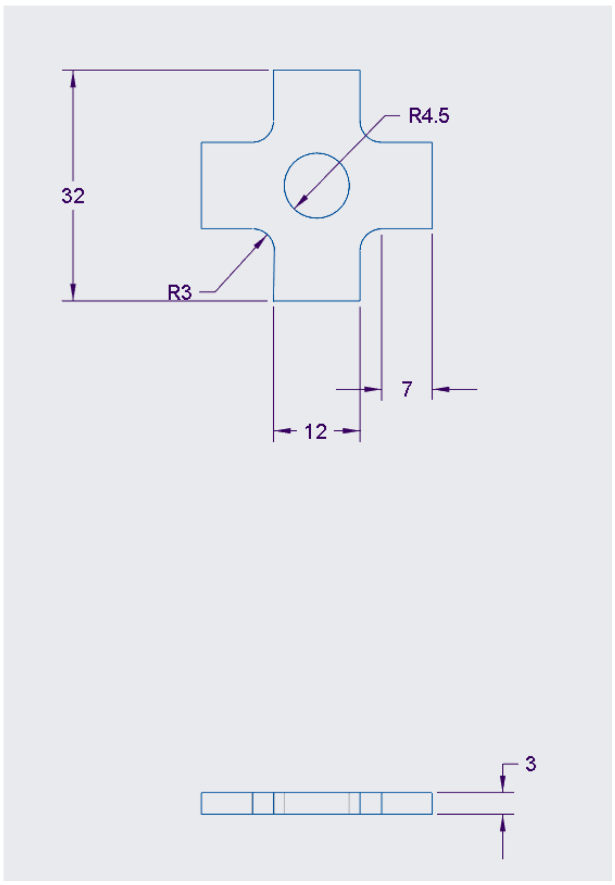


Image 11. Leg's plans



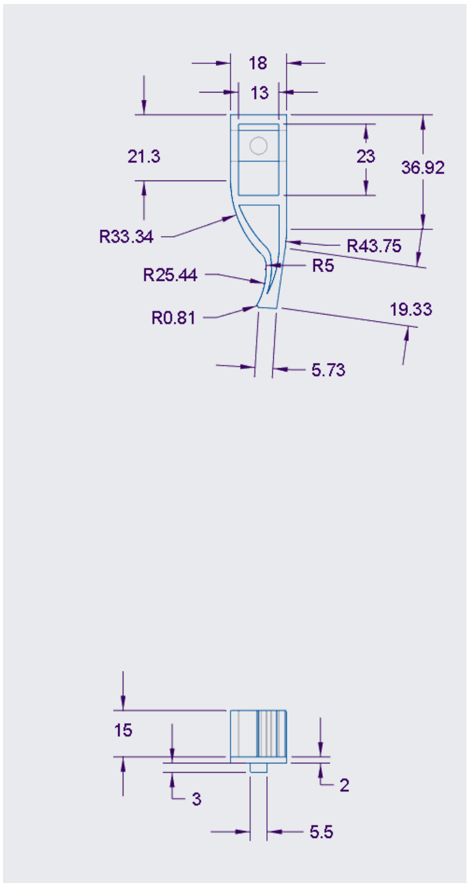


Image 12. Leg's plans Image 13. Leg's plans

Shell:

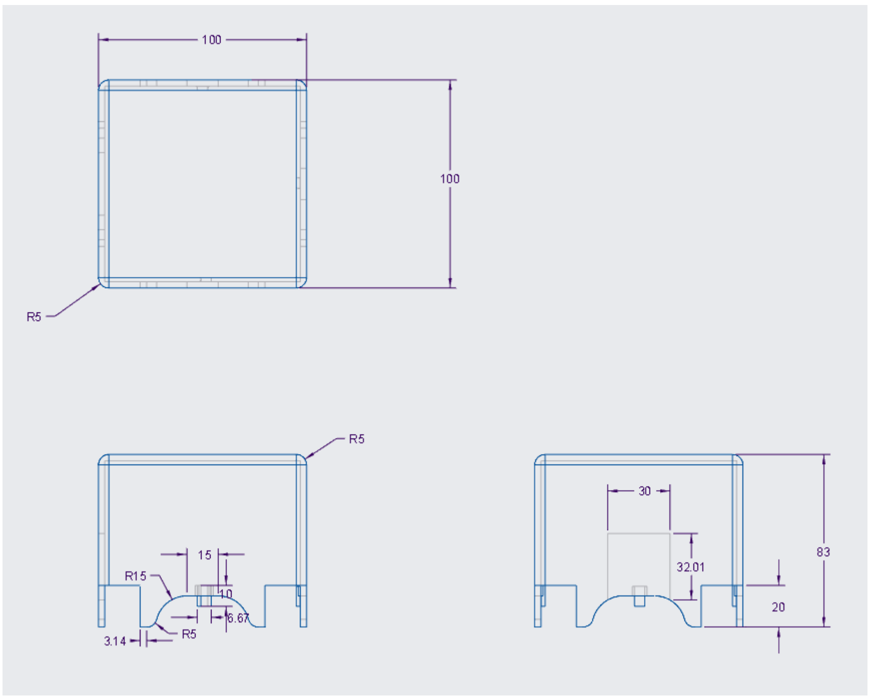


Image 14. Shell plans

3.2.1.3. Assembly Process

The body, legs, servo holders, and protective shell were among the custom-designed parts that were 3D printed before the mechanical parts were assembled. To guarantee snug fitting and low friction during motion, each component was thoroughly cleaned and post-processed after printing.

The following were the assembling steps:

1. **Body preparation**: Two layers were secured using hexagonal standoffs; the upper layer held wiring and electronics, while the lower layer held the battery.
2. **Leg modules** were equipped with two SG90 servomotors for each leg, which was made up of a segment of tibia and a femur. Adhesive was used to join the femur fragments. Using screws and servo horns, these were fastened to the motors.
3. **Leg mounting**: Next, 3D-printed servo brackets were used to mount the constructed legs on the main platform's four corners.
4. **Installation of the shell**: At last, the upper shell was installed, lining up with the base and securing the electronics within while allowing for accessibility and ventilation.

3.2.2. Electrical design

For the electrical design it will be necessary different materials, to make the movement of the legs servomotors will be used, as the spider is planned to be small and to not be heavy, the servomotors will be the micro servomotors SG90, this ones are perfect to fit in the legs of the project, also they will have the necessary power to move the structure and the weight of it, this ones have an operating voltage of 4.8~6 Volts, when the servo motors are resting the amperage will be of 5 to 10 mA, in case of movement without load it will be from 120 to 150mA, when the movement is with maxim load it will reach 650 mA, with average load is from 200 to 300 mA.

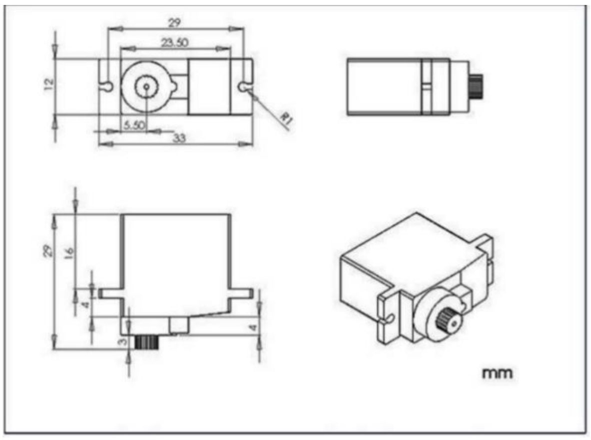


Image 15: Servomotor SG90

To control the spider the use of an ESP32 will be implemented, in this case the voltage of operation is 3,3 V, the amperage at maximum discharge is 60mA, and 500mA in maximum charge, the operation current is 102mA in average.

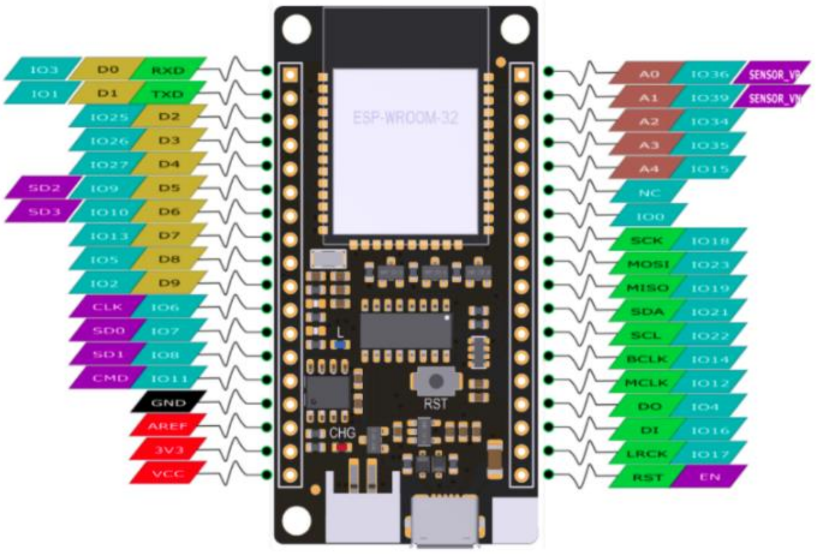


Image 16: ESP32

The system will include an ESP32-CAM, the selected one has an operating voltage of 3,3V, at rest the amperage is 10mA, if it is working without Wi-Fi is 80 to150mA, in case of active Wi-Fi is from 180 to 320 mA, when it is streaming video is 500 to 600mA.

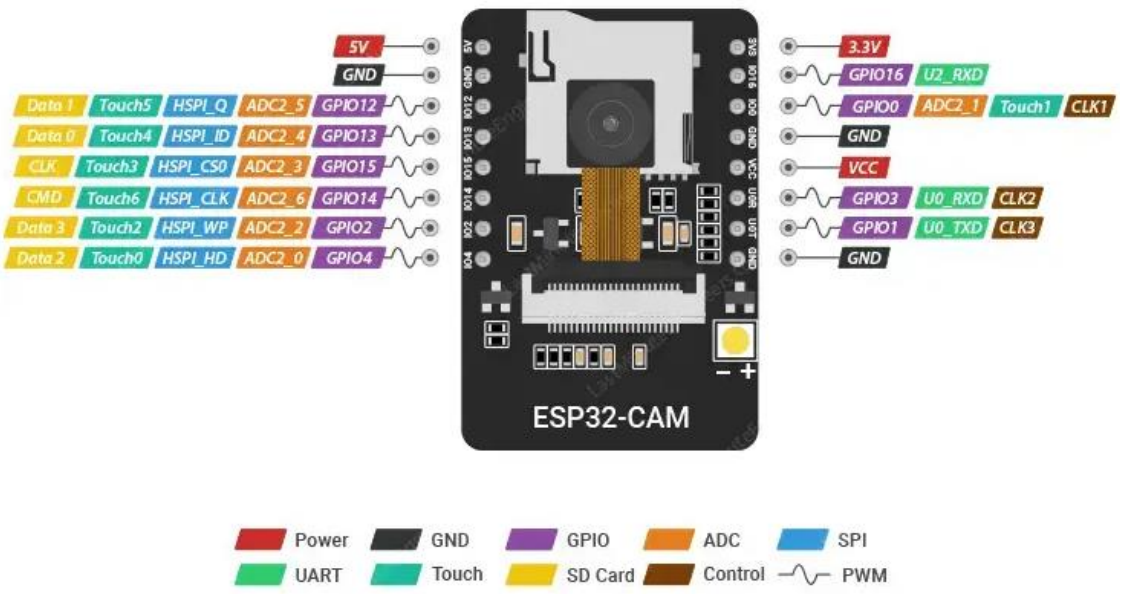


Image 17: ESP32-CAM

When all the electronics devices are known it is necessary to make calculations to know the battery that will be necessary to supply all the components. Using the values found in the various datasheets, it will be considered the worst-case scenario. The average current will be 1200 mA for all the servomotors, in case of the camera and ESP32 will take 230mA, with this value it can be calculated the capacity that it is necessary for the battery, once it is calculated we know that the capacity of the battery will have to be at least 1430mAh.

$$C = I * t = (230mA + 1200mA) * 1hour = 1530mAh$$

Once the materials are selected and all the elements that will appear on the project, the electrical diagram can be done. As the diagram shows, to complete the project it is necessary to have an ESP32, eight servomotors SG90, a regulator of 5 volts and a battery of 7,4 volts. It can be observed in the diagram that the battery will be connected to the 5 volts regulator, where there is an output and input, positive and negative, once these connections are done, it will be connected de output to a previously soldered PCB, arranged in four columns and twelve rows. The first column will be welded vertically and used for the ground connection, the second column, also in a vertical configuration will be where the voltage connections are, the last two columns will be connected horizontally in each row as the PWM. The next connections that can be seen in the diagram are the servomotors, each one has a ground, voltage and PWM connection, every servomotor will be connected to the PCB, horizontally, this way the ground, voltage and PWM will be connected correctly to it, the remain column is used to make the connections to the ESP32, each servomotor is connected to a different pin, those pin will be the same that the ones of the program, to the ESP32 will also go a connection of the 5 volts of the regulator, and the ground, (as shown in Annex 7.2: Electrical diagram)

3.2.3. Control and software

- Movement of the servos:

The control of the spider is going to be divided in two different parts: first it is necessary to control the different legs of it and their movement, this way the spider can simulate how a spider walks, the other part is the camera, in this program the implementation of the legs will be necessary.

First it is necessary to test all the servo motors to check their correct operation, once it is check and the assembly with the structure, it will be checked their different angles with a different code, this way we can make sure it is going to move the way it needs to be.

The servomotors will be divided in two, the ones that will be connected to the platform, and the ones that will be connected to the one that is already connected to the platform. The ones connected to the platform the axis will be vertical, and it will move 90 degrees, the other ones, the axis will be horizontal, and their movement will also be 90 degrees. In the next table it shows the different servos that will be used, their attach to the ESP32, the degree they will start at, the degree they will move to, and the finish degree (the same where they start):

Attach	Servo	Start	Move	Finish
12	1	90º	0º	90º
13	2	90º	0º	90º
14	3	90º	0º	90º
15	4	0º	90º	0º
2	5	90º	0º	90º
16	6	130º	50º	130º
0	7	90º	0º	90º
14	8	0º	90º	0º

Table 1. Characteristics of each servomotor

To make it walk the way it is needed it is necessary to install the library of servo of the ESP32, then define the eight servos that will work, then it is necessary to define using #define the first position of each one with their respective initial angle.

Once the definitions are done, the class Spider is created. This class will represent the movement of the spider, inside this will be public variables. These variables store the angles where each servo wants to go. Inside the class there is the constructor for the class, it calls zero() which is the method to set all the servos to their initial position, this function puts all the public variables to the initial position defined at the beginning of the code, then it synchronizes them with the current position values, calls tick() to update all the servos.

The tick function is to compare each servo's current position with the position where it must move to, and moves it to the target, then it updates the position with write(pos..). Then the done bool is created, this one check if all the servos have reached their target position, if it is it returns true, in the end the creation of private variables of the current position of each servo is done for smooth control.

Once the class is done, the creation of an object of the spider class to control the robot is made. First the movement state variables are done, this one keeps track of which phase of movement the robot is in for each direction, they all start at 0. Then the movement functions are created, there are four different functions, one to go to the front, back, left and right, the way they function is with a switch, each one works with their movement state variable, then it goes inside the switch and to the case 0, once it is inside it changes to one the variable and it goes to the case 1, this makes the command to move the legs it calls to the position written, then the if check if the movement is done, if it is it changes the movement variable to the next number, this case 2, if it is not done it returns. In each void the move order changes, and where they have to move, to move forward first it is called the servos 1, 3, 2 and 4 while to move back first is called 6, 8, 7 and 5. Then it is created a void to stop the spider, this will put all the servos to the first position defined ant the beginning of the code, the next void is a loop to make the movements, first it is necessary to create two variables, one of the direction, and the other one will be the old direction, the void will be working with an if, in case the old direction is different to the direction it will reset all the states of movement, and will change the old direction to the new direction, this direction is a number that will receive of the pressed buttons, to make the move, there is another switch, as the void created before there will be five directions, to move forward, back, left, right or to stop, depending on the direction received, it will activate the switch and the case with the direction received, in the case there is the tick and the function where it will have to move.

- Streaming and communication interface:

1. Web Interface Generation

The ESP32-CAM hosts a web page stored in program memory and served over HTTP. This page includes:

- An <img> tag acting as a dynamic image window.
- Five buttons implemented in HTML and controlled through JavaScript.

2. Streaming Method

Instead of using MJPEG (which can monopolize the processor and block other tasks), the program uses a snapshot-based streaming technique:

- The <img> tag's src attribute is repeatedly updated by JavaScript.
- A JavaScript function uses setInterval to request a new image from the URL /capture every few hundred milliseconds.
- Each request returns a JPEG frame, which is then shown as a live feed.

This frame-by-frame capture strategy allows the microcontroller to stay responsive and handle additional tasks like servo control, making it ideal for multitasking on the ESP32-CAM.

3. Robot Control Buttons

The interface also features five buttons, each linked to a specific movement command (e.g., forward, backward, left, right, stop). Each button:

- Triggers a JavaScript function when clicked.
- This function sends an HTTP GET request to the ESP32-CAM using fetch() with a custom endpoint (like /forward).
- The ESP32-CAM parses the URL in the server handler and triggers the appropriate servo action accordingly.

The GitHub repository contains the full source code and pertinent project files:
https://github.com/Integrated-Project-2-2025-UVic-UCC/Group-1

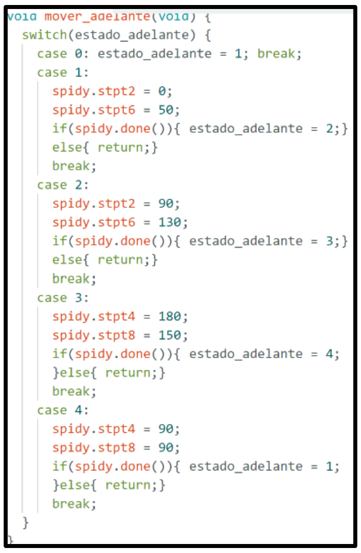
3.3 Functional testing of the prototype

Once the prototype is done, it is necessary to verify the entire system's correct operation. To start the testing, the different parts of the spider must be assembled, this includes the connection of the servomotors to their positions, then the code has to be uploaded to the ESP32, all the electrical connections must be done, once the mechanical and electrical assembly are done, the spider can be powered by the battery to observe the result.

This testing must ensure the correct respond of the servomotors, making the spider move and that it can support the weight of the body. At the first testing the spider is going to move forward, it can be verified whether the sider is capable of walking, to make it possible the code was modified to coordinate the legs so it can move the way it is demanded, the result was that the servomotors didn't move correctly, because of the angles set at the code, once that was changed the legs moved the way it was determined, but it wouldn't move due to a lack of grip on the surface. Once the forward movement was done, all the other moves were tested. As it was in the code, the legs did their respective movements.

When the test with the ESP32 was done, the ESP32-CAM was introduced, changing the ESP32 for it, but the prototype did not pass the functional tests with this board, the issue was from the limited number of available pins, this was insufficient to make all the servomotors work.

Primarily the legs moved in diagonal pairs; to make it work a change in the control strategy was made. The system will only use four servomotors instead of eight. With this change the movement was adapted to work in a horizontal way, for the forward movements first the two servomotors at the front will move, then the two of the back will do the same. This code is uploaded in all directions, including left, right, and backward. Once it was tested, all the legs moved according to the code and there were no errors detected.



```
void mover_adelante(void) {
  switch(estado_adelante) {
    case 0: estado_adelante = 1; break;
    case 1:
      spidy.stpt2 = 0;
      spidy.stpt6 = 50;
      if(spidy.done()){ estado_adelante = 2;}
      else{ return;}
      break;
    case 2:
      spidy.stpt2 = 90;
      spidy.stpt6 = 130;
      if(spidy.done()){ estado_adelante = 3;}
      else{ return;}
      break;
    case 3:
      spidy.stpt4 = 180;
      spidy.stpt8 = 150;
      if(spidy.done()){ estado_adelante = 4;
      }else{ return;}
      break;
    case 4:
      spidy.stpt4 = 90;
      spidy.stpt8 = 90;
      if(spidy.done()){ estado_adelante = 1;
      }else{ return;}
      break;
  }
}
```

Image 18: Modified code

4. Cost of the project

All of the major resources used in the Spider Robot project's development are included in the cost analysis. These include the cost of purchasing materials and components, the expenses associated with mechanical, electrical, and software tasks, and human resources (measured in hours spent). This part computes the total amount of money needed to complete the project, determines the economic worth of the work completed, and gives a summary of how team members and tasks are allocated their efforts. Finding the most resource-intensive components, determining the economic viability, and analysing the profitability of possible mass manufacturing are the goals of this cost analysis.

4.1. Hours Worked

4.1.1. Worked hours per member and total group hours.

Sprints\ Member	**Òscar Guerrero**	**Maria Schreiber**	**Sergi Piguillem**	**Roger Camps**
Sprint 1	5:30 h	8 h	8:30 h	8 h
Sprint 2	9 h	11 h	8:30 h	9 h
Sprint 3	9 h	10 h	8:30 h	9 h
Sprint 4	10 h	11:30 h	9 h	12 h
Sprint 5	8 h	10 h	8 h	10 h
Sprint 6	20 h	14 h	14 h	14 h
Sprint 7	21 h	18 h	19 h	18 h
Total hours per member	82:30 h	82:30 h	75 h	80 h
Total group hours	318 h			

Table 2. Hours worked in each sprint

4.1.2. Mechanical tasks

Tasks\Member	Òscar Guerrero	Maria Schreiber	Sergi Piguillem	Roger Camps
Mechanical assembly testing	9 h	17 h	22 h	7 h
Body design			5 h	20 h
Design of the leg system			17:30 h	6 h
Project specifications			3 h	1 h
Material selection				3 h
Conceptual design				4 h
Manufacturing method				1 h
Stability simulation	5 h	8 h	5 h	5 h
Weight Optimization				1:30 h
Design of the battery and electronics mounting system				4 h
Structural testing on different surfaces	2 h	3 h	3 h	2 h
Total mechanical hours	154 h			

Table 3. Hours worked in mechanical tasks

4.1.3. Electrical tasks

Tasks\Member	Òscar Guerrero	Maria Schreiber	Sergi Piguillem	Roger Camps
Integration of a camera for real – time vision	4 h			
Power supply and system stability testing	2h	8 h	2h	
Test the battery and regulator	6 h	4 h	2h	
Electrical diagram	4h	14 h		
Energy consumption study	2h	10 h	2h	
Battery selection	4 h	4 h	4h	
Selection and study of servomotors	3 h	2 h	7 h	
Wiring and connections design	8h	12 h	4 h	
Electrical system protection	2h	8 h	2h	
Design and feasibility of a custom PCB	2h	8 h		
Total hours	130 h			

Table 4. Hours worked in electrical tasks

4.1.4. Programming

Tasks\Member	Òscar Guerrero	Maria Schreiber	Sergi Piguillem	Roger Camps
Integration of a communication system	2 h	4h		
Real – time video streaming	6 h			
Test the ESP32 - Cam	5 h			
Study of microcontrollers to use and modules	4 h	15 h		
Movement control	9 h	20 h		2 h
Implementation of obstacle detection				
Code optimization for energy efficiency	2 h	9h		
Development of a control interface	4 h			6 h
Testing and debugging	13 h			
Optimization of system response time	2 h			
Test the material	3 h	2h		
Total hours	108 h			

Table 5. Hours worked in preogramming tasks

4.1.5. Management

Tasks\Member	Òscar Guerrero	Maria Schreiber	Sergi Piguillem	Roger Camps
Milestone 0 - Presentation	2 h	2 h	1:30 h	1 h
Github	3 h	1h		2 h
Mid – term presentation	2 h	2 h	2 h	2 h
Milestone 1 – Shopping list	2 h	3 h	3 h	1 h
Sprint reports	3:30 h	2 h		7:30 h
Memory	6 h	14 h	4 h	13 h
Final presentation				
Total management hours	74:30 h			

Table 6. Hours worked in management tasks

4.2. Budget of the project

Materials	Cost for unit	Units	Total cost	Provider	Website
Servomotor (sg90)	3 €	8	24 €	AZDelivery	Microservo
Servomotor (sg90S)	4 €	5	20 €	UVic	
Battery (7.4V 2Ah)	13 €	2	25,99 €	Yangers	Battery
Camera Wi-Fi ESP32	14,99 €	1	14,99 €	Binghe	ESP32-Cam
Voltage regulator (5V)	2,99 €	1	2,99 €	ElectroHo	Regulator
3D Print PLA	16 €	1	16 €	UVic	
Wooden board	1,81 €	1	1,81 €	UVic	
TOTAL	105,78€				

Table 7. Budget of each component

To obtain the total cost of the project, the material cost needs to be added to the labor cost. For engineering services, the price is set at a standard cost of 30€ per hour. To ensure that this cost is in line with industry norms and accurately reflects the value of our professional work, this rate has been set based on the average hourly cost for engineers in Spain. Therefore, the total labor cost of the project is 9.540€. Counting in the material cost as well the total cost of the project comes out to 9.645,78€

To ensure the profitability of this project, a break-even analysis has been carried out. This analysis will give information regarding how many spider robots must be sold to make a profit and count this project as viable. Since the 9.540 are project development costs, it will have repercussions throughout all the built robots, meaning the only fixed cost is the material cost of 106€. Adding a 30% margin to the cost, the price of the product comes to 138€.

As seen in the figure below (Image 19), the break-even point is 300 units sold, meaning once 300 units are sold there will start to be a profit. The cost per spider once 300 units are sold is 137,8€ and every spider above that will be a lower cost reaching an asymptote of 106€ (the material cost of the spider).

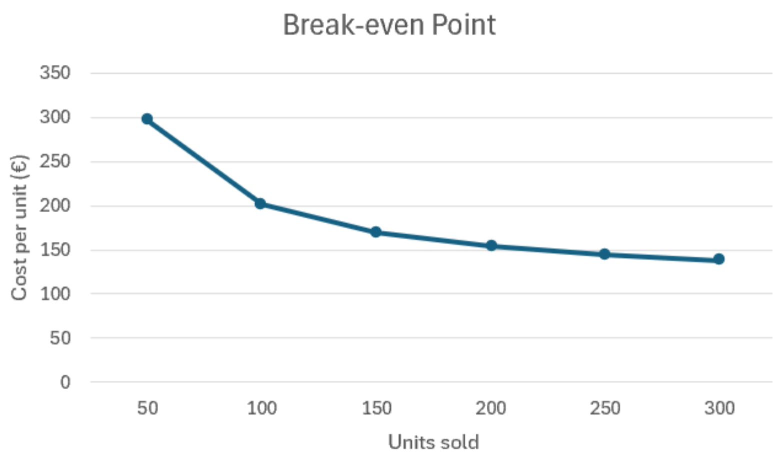


Image 19: Cost per units sold

5. Conclusions

During the project some changes must be made to achieve the proper functioning of the spider, since the starter code and mechanical design had to change, despite the initial problems related to mechanical, where the first platform didn't have all the supports of the materials that were going to be on top of it, and the legs didn't have enough grip to make the body move, and on the electrical design, there weren't enough servomotors pins on the ESP32-CAM, the changes that were made leaded to a functional spider.

The final design, even though it was not the initial one, achieved movement of the legs with four of the eight servos, allowing the robot to walk in any direction, forward, left, right and backward. Although the mechanical, electronical, and programming part had to be changed from the initial one, the robot had the compact size determinate at the beginning, and the battery can operate effectively within the 15 minutes wanted. The different tests showed a functional walking spider.

6. Bibliography

Robugtix, T8X Spider Robot, Robugtix, (2024) from

https://www.robugtix.com/t8x

James Bruton, OpenDog Project, YouTube, (2022) from
https://www.youtube.com/user/jamesbruton

Arduino, ESP32Servo, Kevin Harrington (2025) from

https://docs.arduino.cc/libraries/esp32servo/

AZDelivery (2025), Micro Servo Motor 9G from

https://www.amazon.es/dp/B07CYZK379

Binghe (2025), ESP32-CAM from

https://www.amazon.es/dp/B0D3D8RMC5

Yangers (2025), 7.4V Battery from

https://www.amazon.es/dp/B08P6DB121

ElectroHobby (2025), DC-DC Regulator from

https://www.electrohobby.es/convertidor/151-convertidor-dc-dc-regulable-5a-xl4015.html

ESP32-CAM, Naylamp Mechatronics SAC (2023) from:
https://naylampmechatronics.com/espressif-esp/700-esp32-cam-con-camara-ov2640-esp32-wifi-base-ch340.html

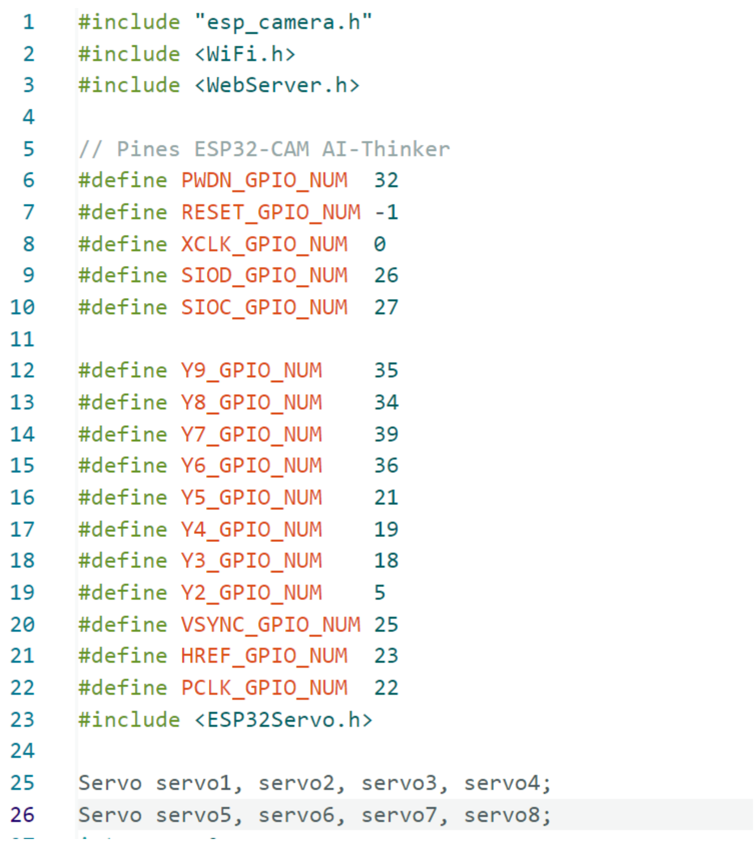
Battery capacity, Nature's Generator from:
https://naturesgenerator.com/blogs/news/how-to-measure-battery-capacity?srsltid=AfmBOorP06CTvTiYV6YFeHxgS37RSpJnvKaujZtnvRZr_HzMK5kSpNjq

ESP32 CAM Pinout Reference:

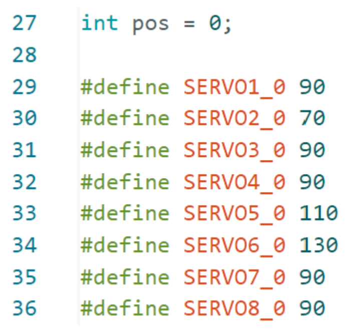
https://www.cabotinoso.es/referencia-de-configuracion-de-pines-de-cam-esp32/

7. Annex

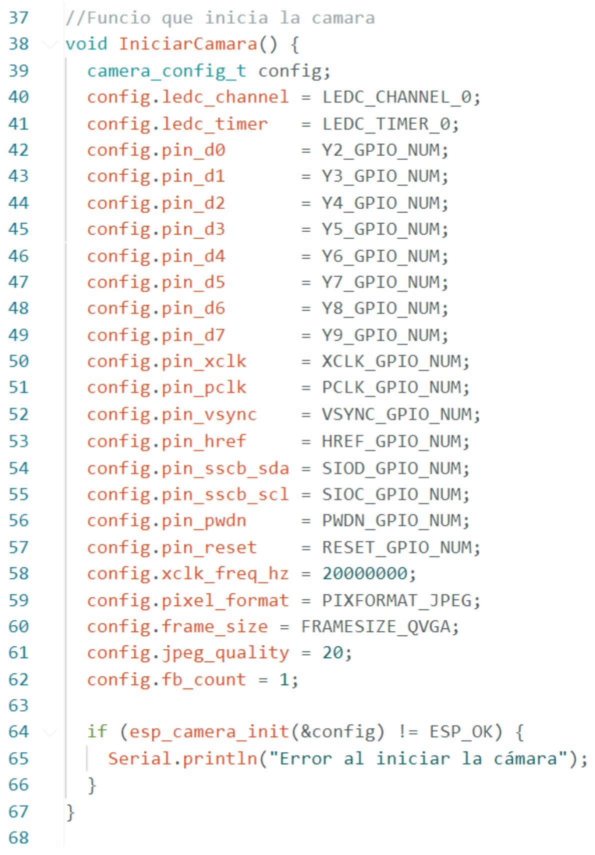
7.1. Arduino codes



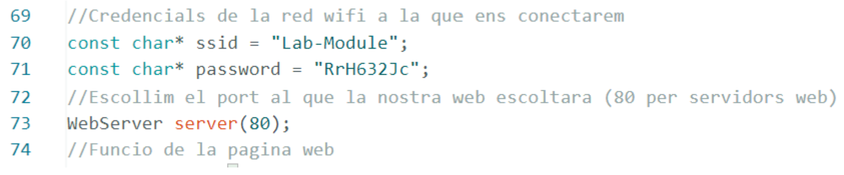
```
#include "esp_camera.h"
#include <WiFi.h>
#include <WebServer.h>

// Pines ESP32-CAM AI-Thinker
#define PWDN_GPIO_NUM  32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM  0
#define SIOD_GPIO_NUM  26
#define SIOC_GPIO_NUM  27

#define Y9_GPIO_NUM    35
#define Y8_GPIO_NUM    34
#define Y7_GPIO_NUM    39
#define Y6_GPIO_NUM    36
#define Y5_GPIO_NUM    21
#define Y4_GPIO_NUM    19
#define Y3_GPIO_NUM    18
#define Y2_GPIO_NUM    5
#define VSYNC_GPIO_NUM 25
#define HREF_GPIO_NUM  23
#define PCLK_GPIO_NUM  22
#include <ESP32Servo.h>

Servo servo1, servo2, servo3, servo4;
Servo servo5, servo6, servo7, servo8;
```



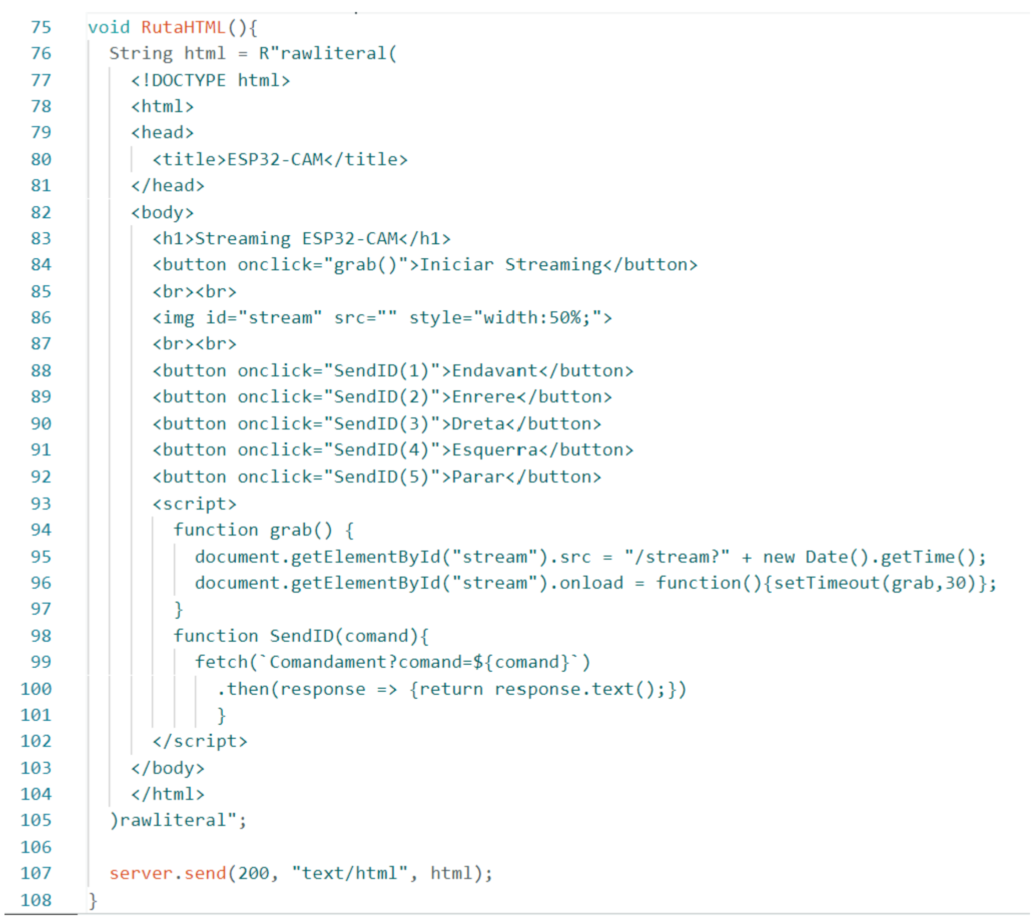
```
int pos = 0;

#define SERVO1_0 90
#define SERVO2_0 70
#define SERVO3_0 90
#define SERVO4_0 90
#define SERVO5_0 110
#define SERVO6_0 130
#define SERVO7_0 90
#define SERVO8_0 90
```



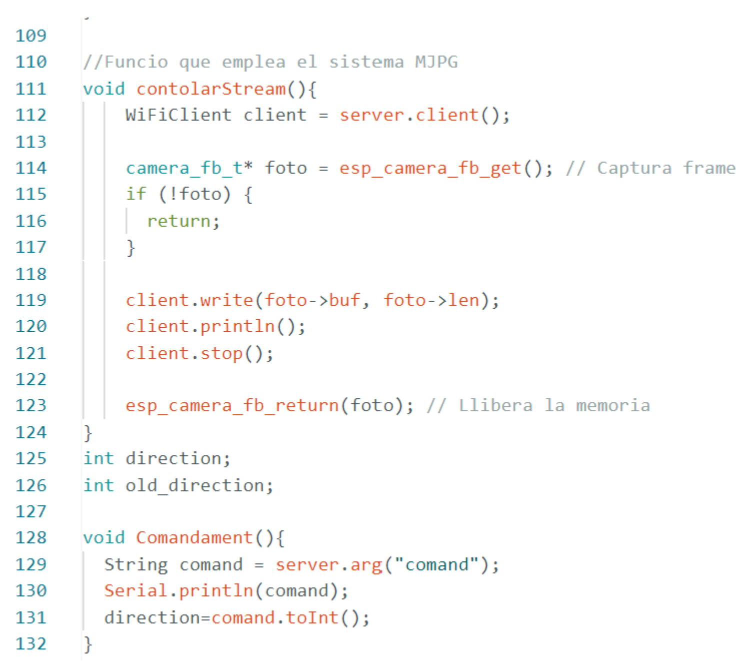
```
//Funcio que inicia la camara
void IniciarCamara() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer   = LEDC_TIMER_0;
  config.pin_d0       = Y2_GPIO_NUM;
  config.pin_d1       = Y3_GPIO_NUM;
  config.pin_d2       = Y4_GPIO_NUM;
  config.pin_d3       = Y5_GPIO_NUM;
  config.pin_d4       = Y6_GPIO_NUM;
  config.pin_d5       = Y7_GPIO_NUM;
  config.pin_d6       = Y8_GPIO_NUM;
  config.pin_d7       = Y9_GPIO_NUM;
  config.pin_xclk     = XCLK_GPIO_NUM;
  config.pin_pclk     = PCLK_GPIO_NUM;
  config.pin_vsync    = VSYNC_GPIO_NUM;
  config.pin_href     = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn     = PWDN_GPIO_NUM;
  config.pin_reset    = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;
  config.frame_size = FRAMESIZE_QVGA;
  config.jpeg_quality = 20;
  config.fb_count = 1;

  if (esp_camera_init(&config) != ESP_OK) {
    Serial.println("Error al iniciar la cámara");
  }
}

```



```
//Credencials de la red wifi a la que ens conectarem
const char* ssid = "Lab-Module";
const char* password = "RrH632Jc";
//Escollim el port al que la nostra web escoltara (80 per servidors web)
WebServer server(80);
//Funcio de la pagina web
```

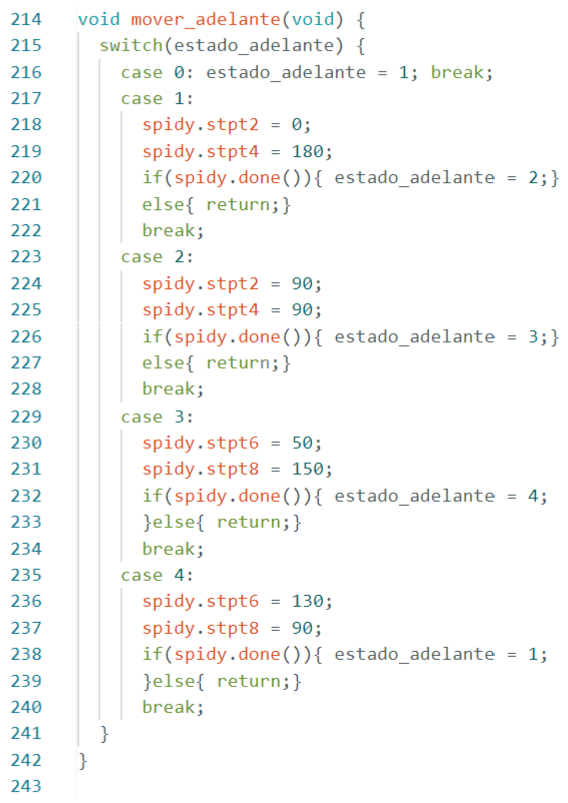


```
void RutaHTML(){
  String html = R"rawliteral(
    <!DOCTYPE html>
    <html>
    <head>
      <title>ESP32-CAM</title>
    </head>
    <body>
      <h1>Streaming ESP32-CAM</h1>
      <button onclick="grab()">Iniciar Streaming</button>
      <br><br>
      <img id="stream" src="" style="width:50%;">
      <br><br>
      <button onclick="SendID(1)">Endavant</button>
      <button onclick="SendID(2)">Enrere</button>
      <button onclick="SendID(3)">Dreta</button>
      <button onclick="SendID(4)">Esquerra</button>
      <button onclick="SendID(5)">Parar</button>
      <script>
        function grab() {
          document.getElementById("stream").src = "/stream?" + new Date().getTime();
          document.getElementById("stream").onload = function(){setTimeout(grab,30)};
        }
        function SendID(comand){
          fetch(`Comandament?comand=${comand}`)
            .then(response => {return response.text();})
            }
      </script>
    </body>
    </html>
  )rawliteral";

  server.send(200, "text/html", html);
}
```

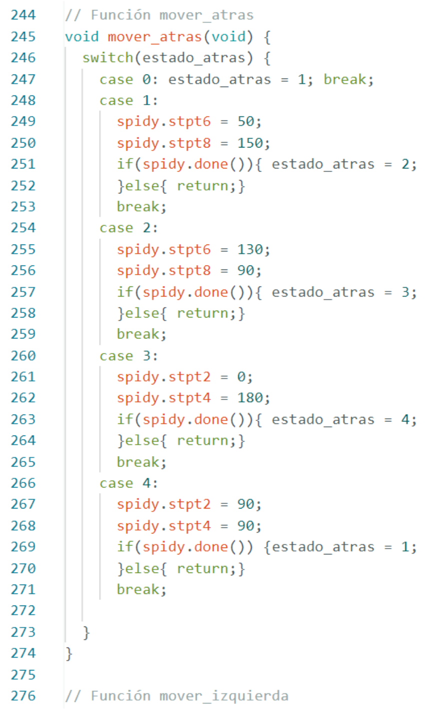


```

//Funcio que emplea el sistema MJPG
void contolarStream(){
    WiFiClient client = server.client();

    camera_fb_t* foto = esp_camera_fb_get(); // Captura frame
    if (!foto) {
      return;
    }

    client.write(foto->buf, foto->len);
    client.println();
    client.stop();

    esp_camera_fb_return(foto); // Llibera la memoria
}
int direction;
int old_direction;

void Comandament(){
  String comand = server.arg("comand");
  Serial.println(comand);
  direction=comand.toInt();
}
```

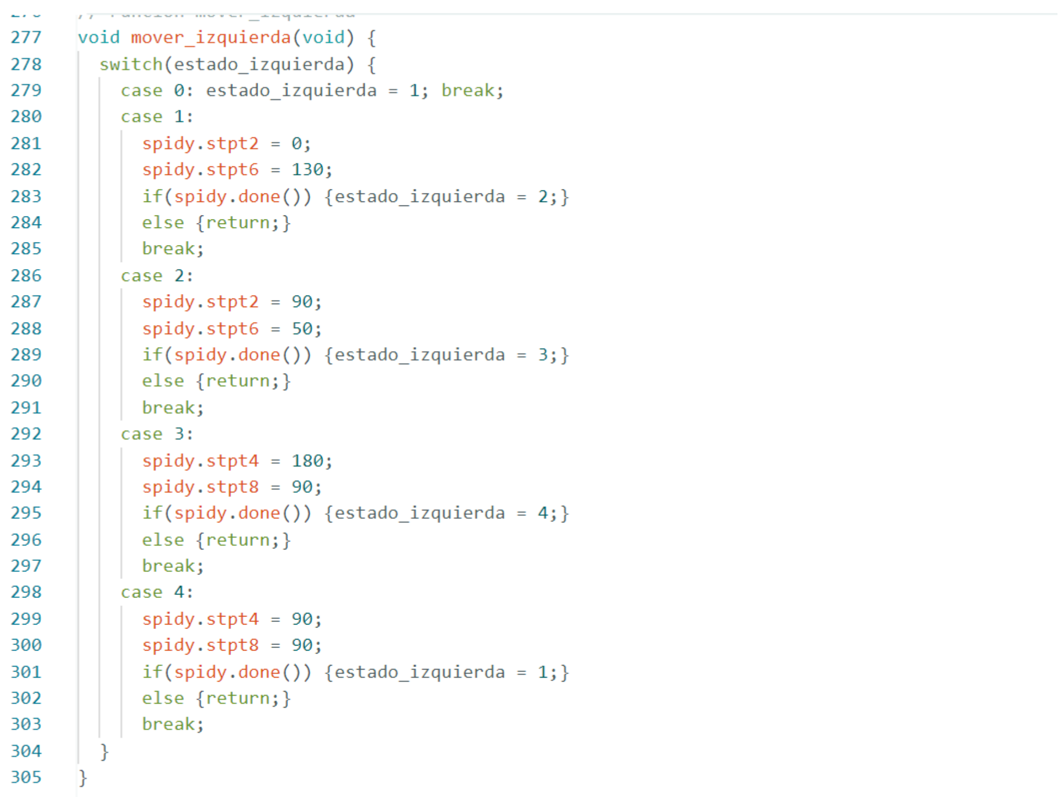
```
class Spider {
  public:
    int stpt1, stpt2, stpt3, stpt4;
    int stpt5, stpt6, stpt7, stpt8;

    Spider() {
      zero();
    }

    void zero() {
      stpt1 = SERVO1_0;
      stpt2 = SERVO2_0;
      stpt3 = SERVO3_0;
      stpt4 = SERVO4_0;
      stpt5 = SERVO5_0;
      stpt6 = SERVO6_0;
      stpt7 = SERVO7_0;
      stpt8= SERVO8_0;
      pos1 = stpt1;
      pos2 = stpt2;
      pos3 = stpt3;
      pos4 = stpt4;
      pos5 = stpt5;
      pos6 = stpt6;
      pos7 = stpt7;
      pos8 = stpt8;
      tick();
      delay(1000);
    }

    void tick() {
      if (pos1<stpt1) pos1++;
      else if (pos1>stpt1) pos1--;
      if (pos2<stpt2) pos2++;
      else if (pos2>stpt2) pos2--;
      if (pos3<stpt3) pos3++;
      else if (pos3>stpt3) pos3--;
      if (pos4<stpt4) pos4++;
      else if (pos4>stpt4) pos4--;
      if (pos5<stpt5) pos5++;
      else if (pos5>stpt5) pos5--;
      if (pos6<stpt6) pos6++;
      else if (pos6>stpt6) pos6--;
      if (pos7<stpt7) pos7++;
      else if (pos7>stpt7) pos7--;
      if (pos8<stpt8) pos8++;
      else if (pos8>stpt8) pos8--;

      servo1.write(pos1);
      servo2.write(pos2);
      servo3.write(pos3);
      servo4.write(pos4);
      servo5.write(pos5);
      servo6.write(pos6);
      servo7.write(pos7);
      servo8.write(pos8);

      delay(10);
    }

    bool done() {
      return pos1==stpt1 && pos2==stpt2 && pos3==stpt3 && pos4==stpt4 && pos5==stpt5 && pos6==stpt6 && pos7==stpt7 && pos8==stpt8;
    }

  private:
    int pos1, pos2, pos3, pos4;
    int pos5, pos6, pos7, pos8;

};



Spider spidy;

//estats
int estado_adelante = 0;
int estado_atras = 0;
int estado_izquierda = 0;
int estado_derecha = 0;

// mover_adelante
```



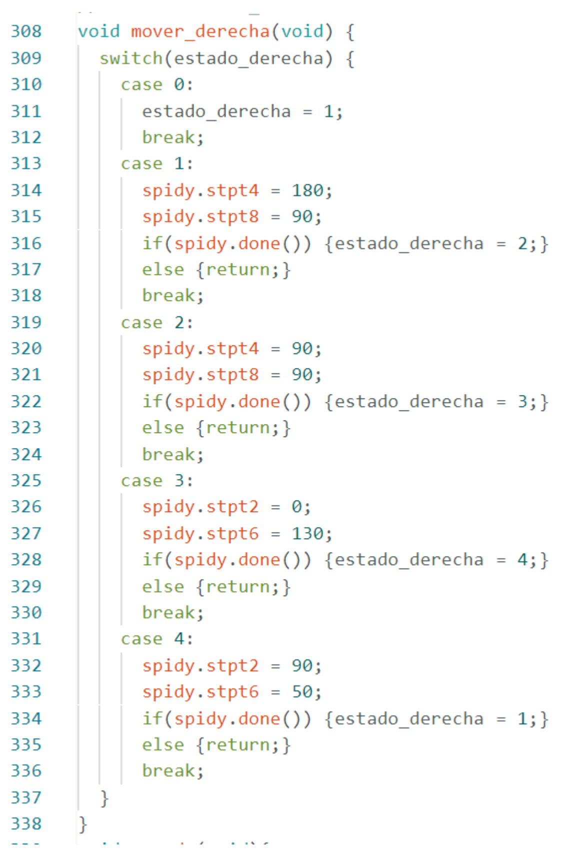
```
void mover_adelante(void) {
  switch(estado_adelante) {
    case 0: estado_adelante = 1; break;
    case 1:
      spidy.stpt2 = 0;
      spidy.stpt4 = 180;
      if(spidy.done()){ estado_adelante = 2;}
      else{ return;}
      break;
    case 2:
      spidy.stpt2 = 90;
      spidy.stpt4 = 90;
      if(spidy.done()){ estado_adelante = 3;}
      else{ return;}
      break;
    case 3:
      spidy.stpt6 = 50;
      spidy.stpt8 = 150;
      if(spidy.done()){ estado_adelante = 4;
      }else{ return;}
      break;
    case 4:
      spidy.stpt6 = 130;
      spidy.stpt8 = 90;
      if(spidy.done()){ estado_adelante = 1;
      }else{ return;}
      break;
  }
}

```



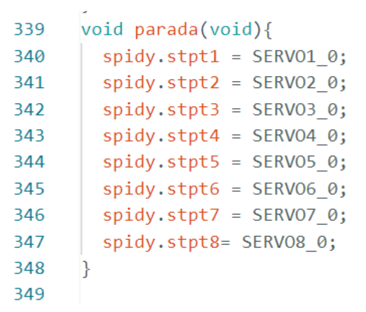
```
// Función mover_atras
void mover_atras(void) {
  switch(estado_atras) {
    case 0: estado_atras = 1; break;
    case 1:
      spidy.stpt6 = 50;
      spidy.stpt8 = 150;
      if(spidy.done()){ estado_atras = 2;
      }else{ return;}
      break;
    case 2:
      spidy.stpt6 = 130;
      spidy.stpt8 = 90;
      if(spidy.done()){ estado_atras = 3;
      }else{ return;}
      break;
    case 3:
      spidy.stpt2 = 0;
      spidy.stpt4 = 180;
      if(spidy.done()){ estado_atras = 4;
      }else{ return;}
      break;
    case 4:
      spidy.stpt2 = 90;
      spidy.stpt4 = 90;
      if(spidy.done()) {estado_atras = 1;
      }else{ return;}
      break;

  }
}

// Función mover_izquierda
```



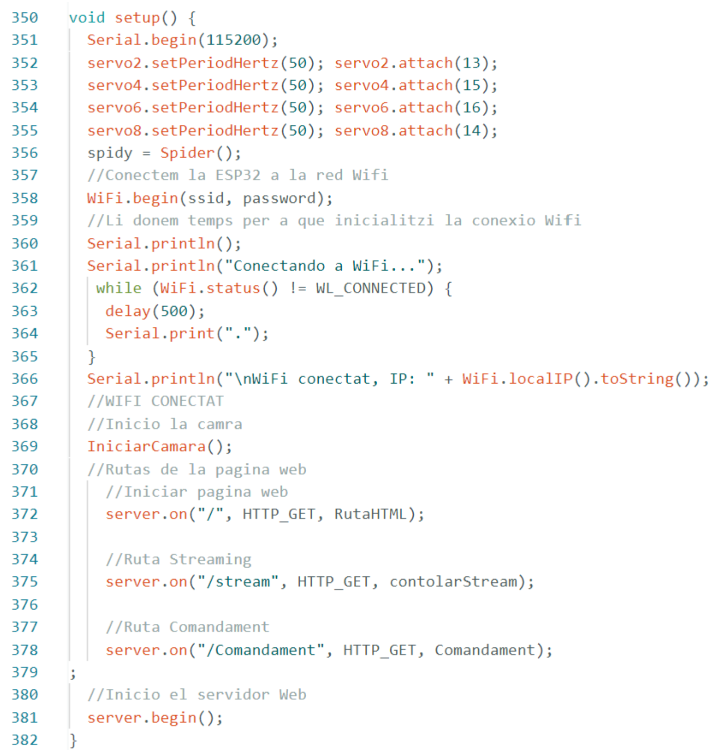
```
void mover_izquierda(void) {
  switch(estado_izquierda) {
    case 0: estado_izquierda = 1; break;
    case 1:
      spidy.stpt2 = 0;
      spidy.stpt6 = 130;
      if(spidy.done()) {estado_izquierda = 2;}
      else {return;}
      break;
    case 2:
      spidy.stpt2 = 90;
      spidy.stpt6 = 50;
      if(spidy.done()) {estado_izquierda = 3;}
      else {return;}
      break;
    case 3:
      spidy.stpt4 = 180;
      spidy.stpt8 = 90;
      if(spidy.done()) {estado_izquierda = 4;}
      else {return;}
      break;
    case 4:
      spidy.stpt4 = 90;
      spidy.stpt8 = 90;
      if(spidy.done()) {estado_izquierda = 1;}
      else {return;}
      break;
  }
}

// Función mover_derecha
```



```
void mover_derecha(void) {
  switch(estado_derecha) {
    case 0:
      estado_derecha = 1;
      break;
    case 1:
      spidy.stpt4 = 180;
      spidy.stpt8 = 90;
      if(spidy.done()) {estado_derecha = 2;}
      else {return;}
      break;
    case 2:
      spidy.stpt4 = 90;
      spidy.stpt8 = 90;
      if(spidy.done()) {estado_derecha = 3;}
      else {return;}
      break;
    case 3:
      spidy.stpt2 = 0;
      spidy.stpt6 = 130;
      if(spidy.done()) {estado_derecha = 4;}
      else {return;}
      break;
    case 4:
      spidy.stpt2 = 90;
      spidy.stpt6 = 50;
      if(spidy.done()) {estado_derecha = 1;}
      else {return;}
      break;
  }
}
```



```
void parada(void){
  spidy.stpt1 = SERVO1_0;
  spidy.stpt2 = SERVO2_0;
  spidy.stpt3 = SERVO3_0;
  spidy.stpt4 = SERVO4_0;
  spidy.stpt5 = SERVO5_0;
  spidy.stpt6 = SERVO6_0;
  spidy.stpt7 = SERVO7_0;
  spidy.stpt8= SERVO8_0;
}

```

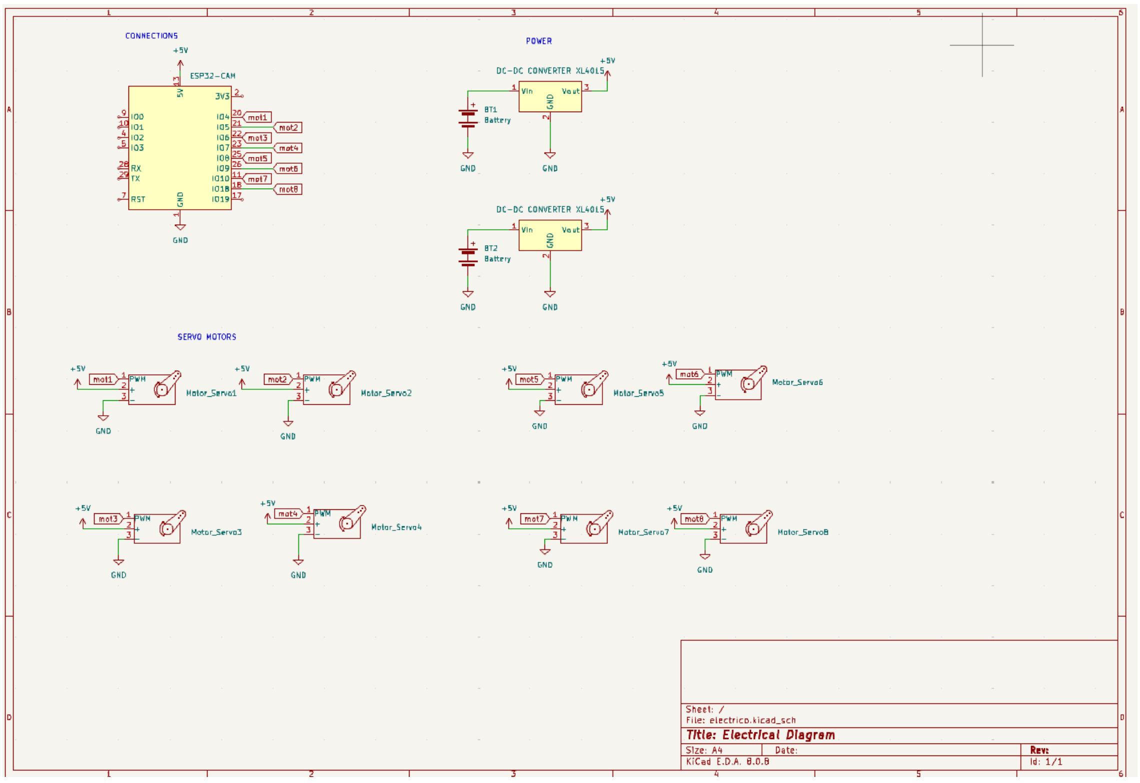


```
void setup() {
  Serial.begin(115200);
  servo2.setPeriodHertz(50); servo2.attach(13);
  servo4.setPeriodHertz(50); servo4.attach(15);
  servo6.setPeriodHertz(50); servo6.attach(16);
  servo8.setPeriodHertz(50); servo8.attach(14);
  spidy = Spider();
  //Conectem la ESP32 a la red Wifi
  WiFi.begin(ssid, password);
  //Li donem temps per a que inicialitzi la conexio Wifi
  Serial.println();
  Serial.println("Conectando a WiFi...");
   while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi conectat, IP: " + WiFi.localIP().toString());
  //WIFI CONECTAT
  //Inicio la camra
  IniciarCamara();
  //Rutas de la pagina web
    //Iniciar pagina web
    server.on("/", HTTP_GET, RutaHTML);

    //Ruta Streaming
    server.on("/stream", HTTP_GET, contolarStream);

    //Ruta Comandament
    server.on("/Comandament", HTTP_GET, Comandament);
;
  //Inicio el servidor Web
  server.begin();
}
```

```

void loop_movement() { //switch
  if(old_direction != direction) {
    // reset all states
    estado_adelante = 0;
    estado_atras = 0;
    estado_izquierda = 0;
    estado_derecha = 0;
  }
  old_direction = direction;
  Serial.println(direction);
  switch(direction){
    case 1:
      mover_adelante();
      break;
    case 2:
      mover_atras();
      break;
    case 3:
      mover_derecha();
      break;
    case 4:
      mover_izquierda();
      break;
    case 5:
      parada();
      break;

  }

}
```

```

void loop() {
  server.handleClient();
  spidy.tick();
  loop_movement();
}
```

7.2. Electrical diagram


CONNECTIONS
+5V
ESP32-CAM
5V
3V3
IO0
IO1
IO2
IO3
RX
TX
RST
GND
IO4
IO5
IO6
IO7
IO8
IO9
IO10
IO18
IO19
mot1
mot2
mot3
mot4
mot5
mot6
mot7
mot8
GND
POWER
DC-DC CONVERTER XL4015
Vin
Vout
GND
BT1
Battery
BT2
Battery
+5V
GND
SERVO MOTORS
PWM
Motor_Servo1
Motor_Servo2
Motor_Servo3
Motor_Servo4
Motor_Servo5
Motor_Servo6
Motor_Servo7
Motor_Servo8
Sheet: /
File: electrico.kicad_sch
Title: Electrical Diagram
Size: A4
Date:
Rev:
KiCad E.D.A. 8.0.8
Id: 1/1