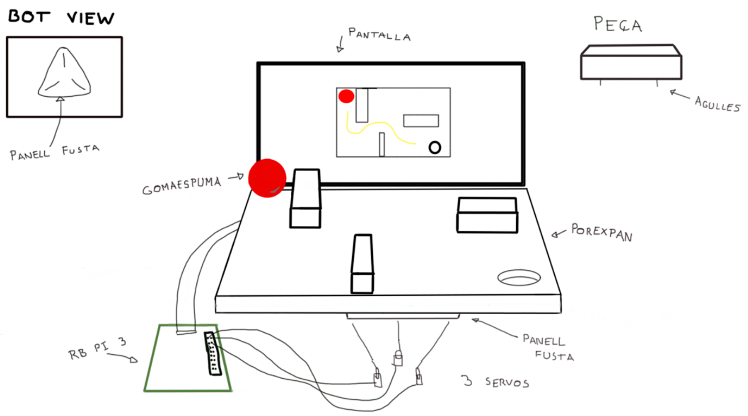


BALLBOT

This project aims to recreate the maze game with a ball. This ball will move with the movement of the board, in order to reach the final destination.

PROJECT SPRINT #3
DATE: 4th May 2021

Juan Carlos Martinez Moreno	1566936
Victor Bosch Pueyo	1566583
Alvaro Caravaca Hernández	1566685
Luis Fernando Paz Galeano	1567369

Table of Contents

BALLBOT

This project aims to recreate the maze game with a ball. This ball will move with the movement of the board, in order to reach the final destination.

Project description

This project aims to recreate the maze game with a ball. This ball will move with the movement of the board, in order to reach the final destination.

To give it a different point of view, it will be the same BALLBOT who moves the board, and it will be the player who designs the circuit. Therefore, the robot must be able to adapt to the board each time the user modifies it.

In addition, there will be a screen showing the circuit that captures the camera, and the circuit that the robot has calculated to take the ball to its final destination will be drawn in real time.

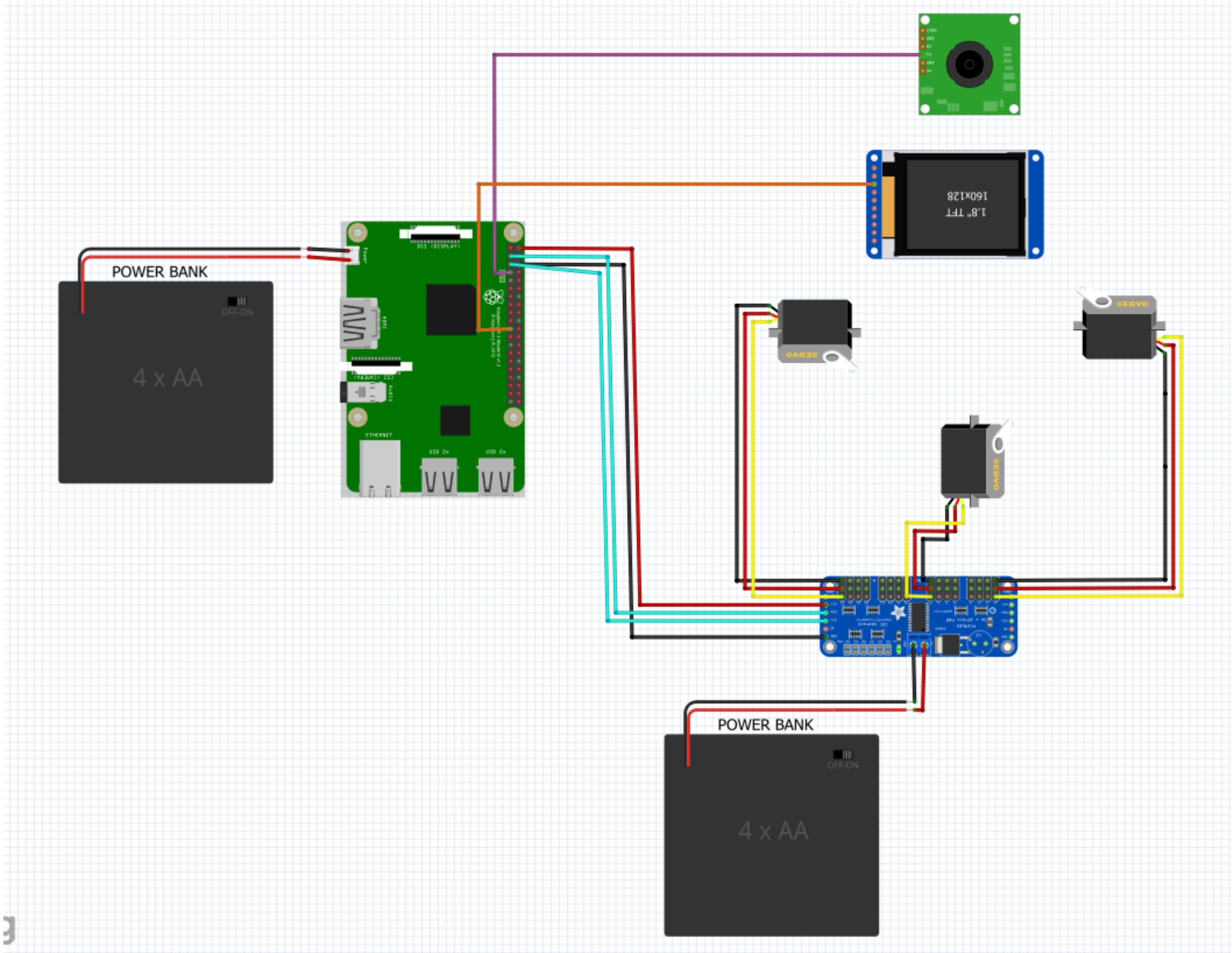
Electronic components

This is the list of the used components:

- *RASPBERRY PI 3*
- *CAMERA*
- *SD*
- *SERVOMOTOR (3)*
- *SCREEN*

Ballbot

Hardware Scheme

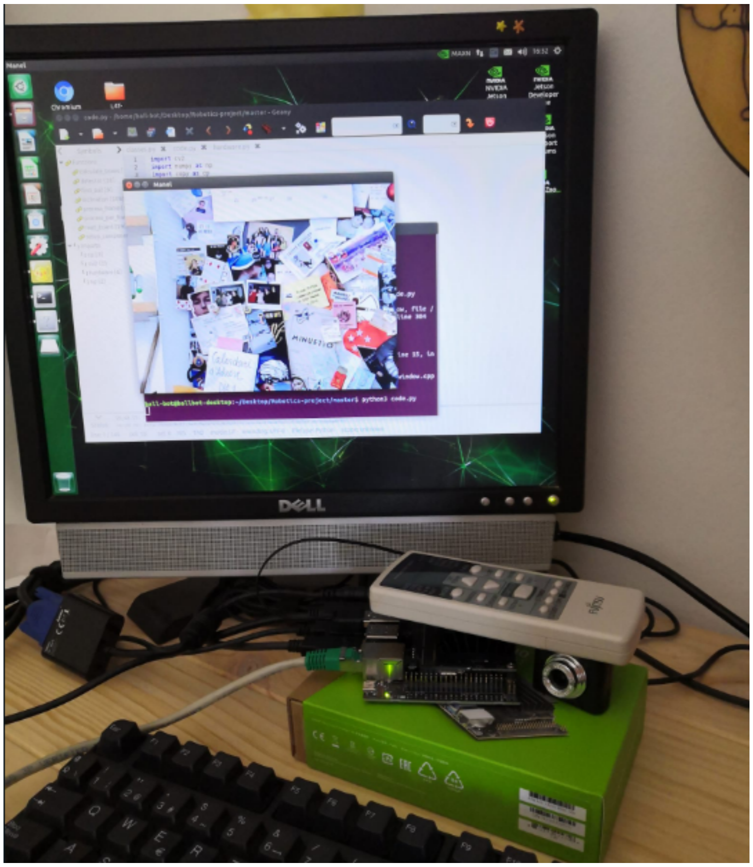


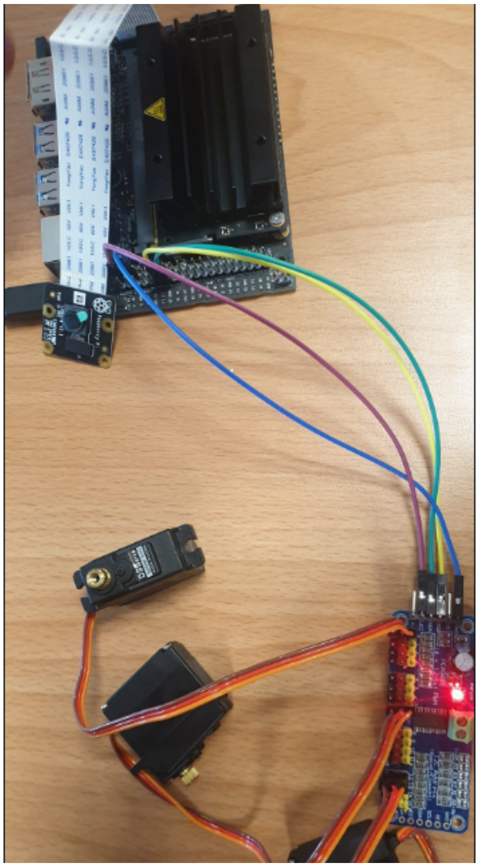
The connections will go as follow:

- *Both power supplies will be connected to a Power Bank*
- *Camera connected to Raspberry Pi camera module*
- *Screen connected to Raspberry Pi display module*
- *Raspberry Pi pin 6 (GND) to PCA9665 GND pin*
- *Raspberry Pi pin 2 (5V) to PCA9665 VCC pin*
- *Raspberry Pi pin 3 (GPIO 2) to PCA9665 SDA pin*
- *Raspberry Pi pin 5 (GPIO 3) to PCA9665 SCL pin*
- *3 servos connected to PCA9665 0, 8 and 15*

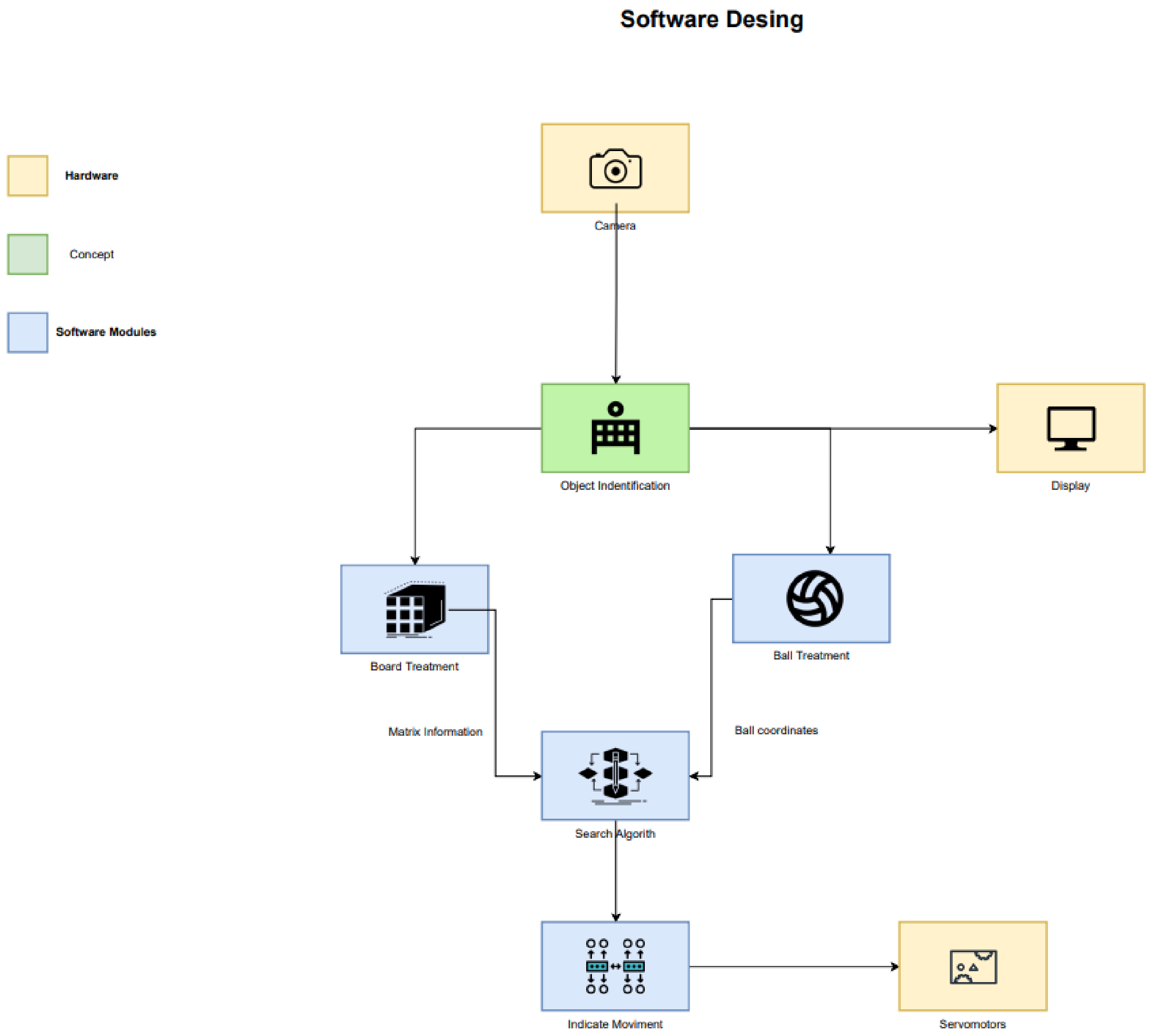
Ballbot

To create the final robot scheme, we have connected the different components.
It has been configured to use the *nvidia jetson*, which is the robot's main computer.
We have also assembled the servos and the camera.
Note:
At this point on a physical level we were unable to begin the project because the camera provided had an excessive amount of zoom, which required the labyrinth to be at a considerable distance in order to focus.
We have requested a change of camera, with which it will be necessary to reconfigure and connect the components based on this new modification.





Software Architecture



- *Board treatment: Identify the structure of the maze (position and orientation of the walls, identification of start and goal points).*
- *Ball treatment: identifying the ball's position; necessary for calculating the path to follow.*

Ballbot

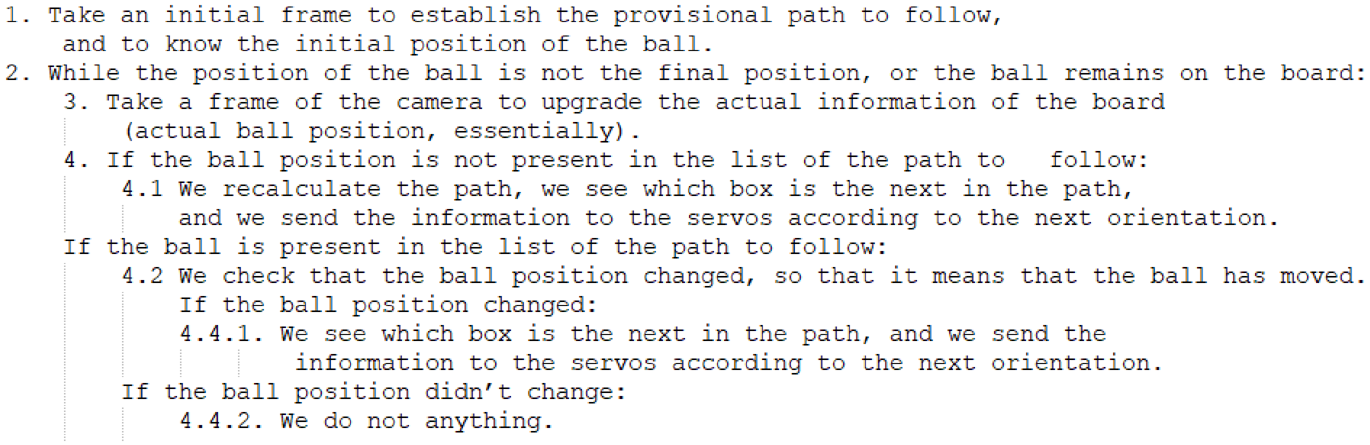
- *Search algorithm: determine the path the ball must follow to reach the goal point (there are different types of search algorithms, for example: backtracking, branch&bound).*
- *The module indicates to the servos what type of movement they need to perform (orientation, force, etc.)*

- ***Update**: We have changed the flow of the function as we have considered modifications to the function designed to implement the board resolution algorithm.*

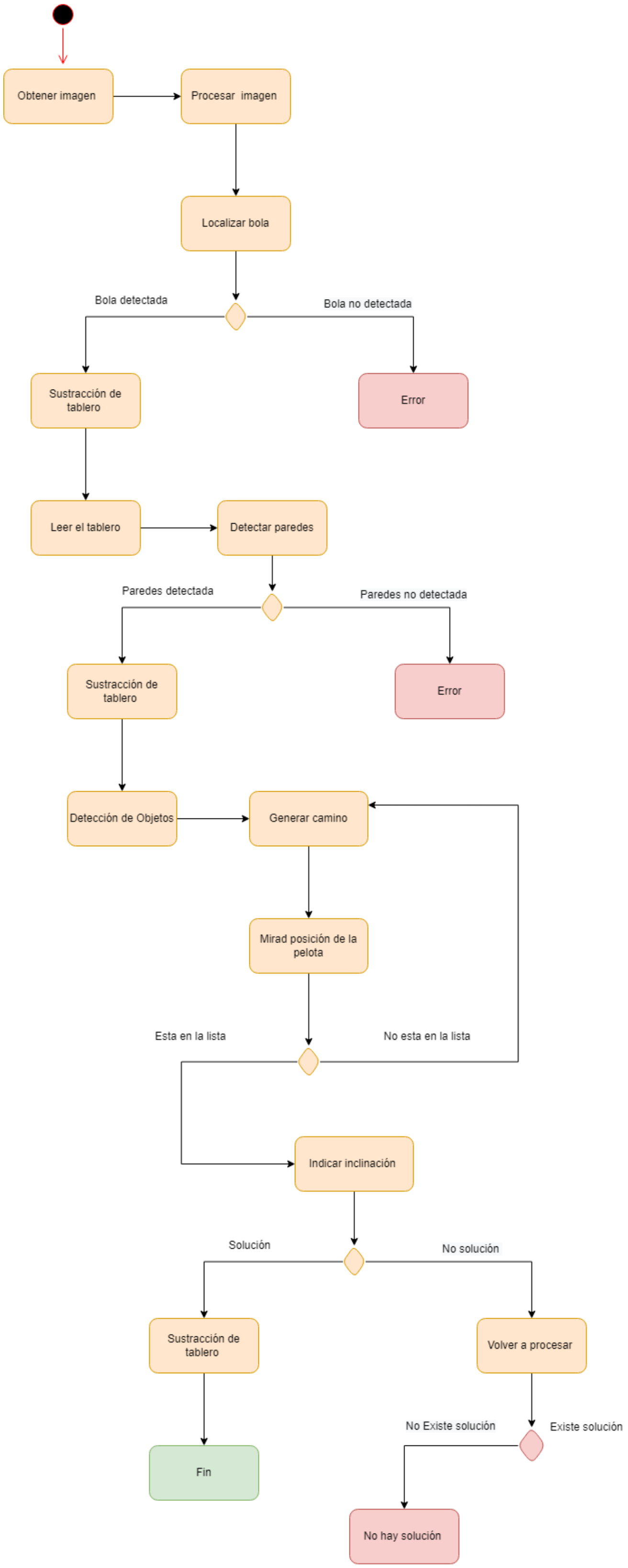
- *We have gone on to represent the board at a smaller size to better size the squares, as this will give more room to detect the walls of the board, based on the set of nodes that make up the wall in question.*

- *We have also started to implement the functions of the treatment of the physical components (hardware), specifically the use of servo motors.*

The pseudocode that follows our robot to execute the task of moving the ball to the final position is:

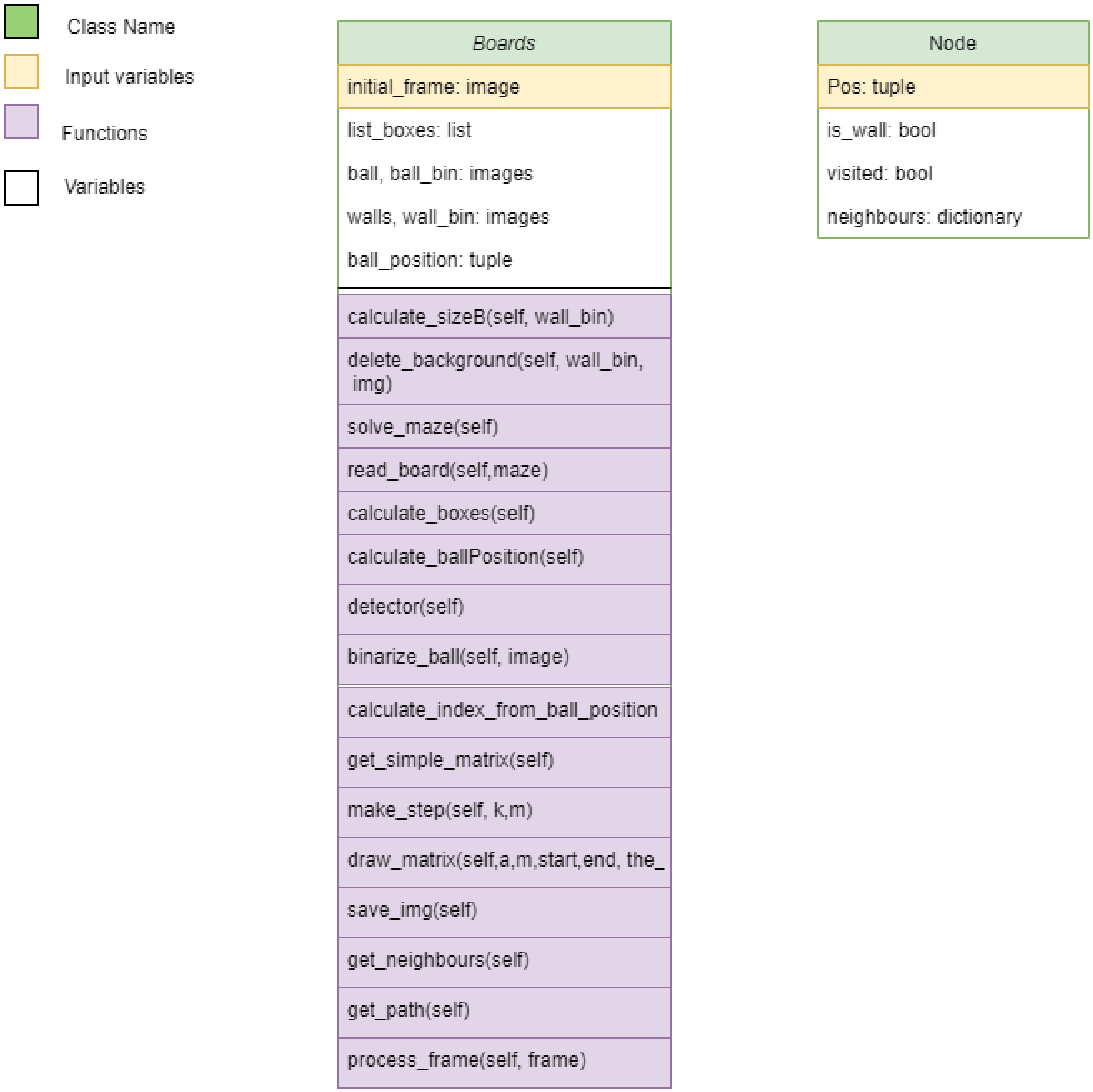


```
1. Take an initial frame to establish the provisional path to follow,
    and to know the initial position of the ball.
2. While the position of the ball is not the final position, or the ball remains on the board:
    3. Take a frame of the camera to upgrade the actual information of the board
        (actual ball position, essentially).
    4. If the ball position is not present in the list of the path to   follow:
        4.1 We recalculate the path, we see which box is the next in the path,
            and we send the information to the servos according to the next orientation.
    If the ball is present in the list of the path to follow:
        4.2 We check that the ball position changed, so that it means that the ball has moved.
            If the ball position changed:
            4.4.1. We see which box is the next in the path, and we send the
                    information to the servos according to the next orientation.
        If the ball position didn’t change:
            4.4.2. We do not anything.
```


Obtener imagen
Procesar imagen
Localizar bola
Bola detectada
Bola no detectada
Sustracción de tablero
Error
Leer el tablero
Detectar paredes
Paredes detectada
Paredes no detectada
Sustracción de tablero
Error
Detección de Objetos
Generar camino
Mirad posición de la pelota
Esta en la lista
No esta en la lista
Indicar inclinación
Solución
No solución
Sustracción de tablero
Volver a procesar
Fin
No Existe solución
Existe solución
No hay solución

Ballbot

Class Diagram



🟩 Class Name

🟨 Input variables

🟪 Functions

□ Variables

Boards

- initial_frame: image
- list_boxes: list
- ball, ball_bin: images
- walls, wall_bin: images
- ball_position: tuple
- calculate_sizeB(self, wall_bin)
- delete_background(self, wall_bin, img)
- solve_maze(self)
- read_board(self,maze)
- calculate_boxes(self)
- calculate_ballPosition(self)
- detector(self)
- binarize_ball(self, image)
- calculate_index_from_ball_position
- get_simple_matrix(self)
- make_step(self, k,m)
- draw_matrix(self,a,m,start,end, the_
- save_img(self)
- get_neighbours(self)
- get_path(self)
- process_frame(self, frame)

Node

- Pos: tuple
- is_wall: bool
- visited: bool
- neighbours: dictionary

Amazing contributions

The following features make our project unique:

- *The structure of the maze is built through user interaction.*
- *Mazes can be solved in an infinite number of ways.*
- *The robot's ability to guide the ball to the goal.*
- *Control of the edges, because the ball cannot go out of the board.*
- *Device for displaying the path to be taken, or the path that is currently being taken.*

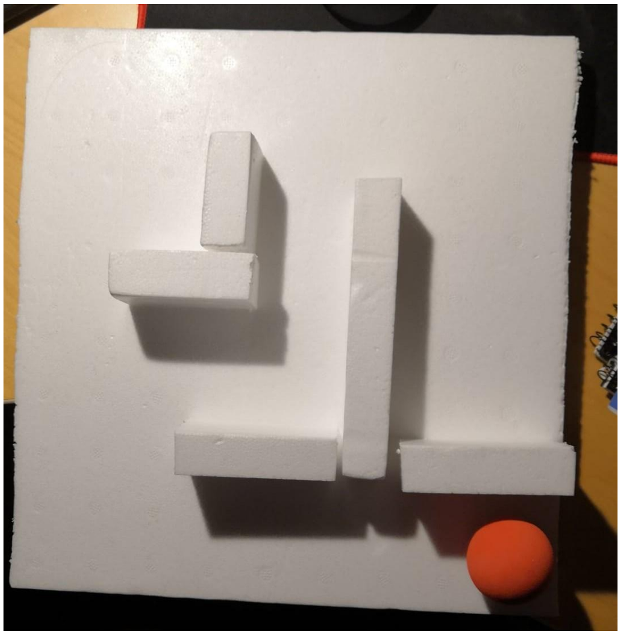
Our project deserves at least a grade of 8-9 for its originality, since it combines two different ideas: an existing project that involves balancing a ball on a platform, and a labyrinth in which the ball will move to reach the goal. A big challenge here is combining algorithms that solve different problems very quickly with computer vision to continuously see the labyrinth and determine the optimal path. Also, we will deal with some physical restrictions, like the material of the board (porexpan, so soft...) or the walls, that can be broken.

Extra components and 3D pieces

- *POREXPAN BOARD*
- *POREXPAN PIECES*
- *WOOD PLANK*
- *RED FOAM BALL*
- *CAMERA SUPPORT*

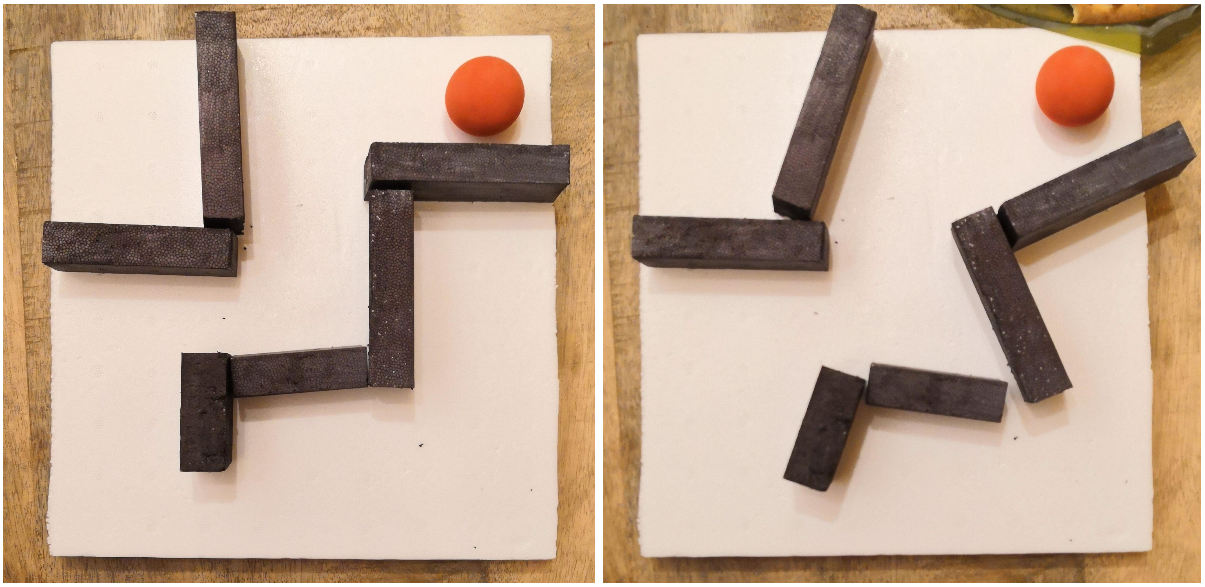
The porexpan board will be used as the main board of the BALLBOT. In there, the porexpan pieces will stick together through needles. The pieces will be the walls of the labyrinth, and it will be the player who creates the entire labyrinth using those pieces. The material has to be porexpan so that all the pieces can be removed from the board easily.

The red foam ball will be the ball that the robot will move by the inclinations of the board in order to get into the final destination.



Ballbot

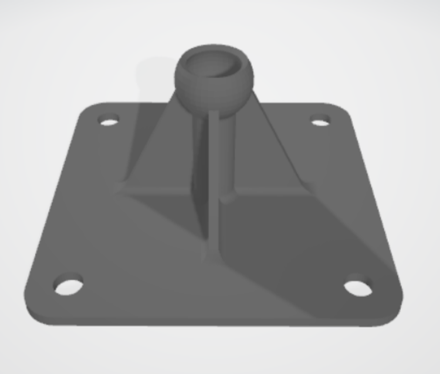
Prototype parts version 1.0



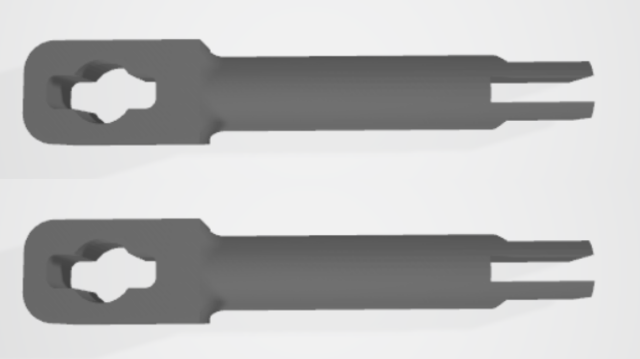
We developed a rather realistic board prototype for this part of the project.

With this prototype, we are executing the various treatments that will be necessary before, during, and after the application of the search algorithm that we will use to solve the labyrinth.

Base

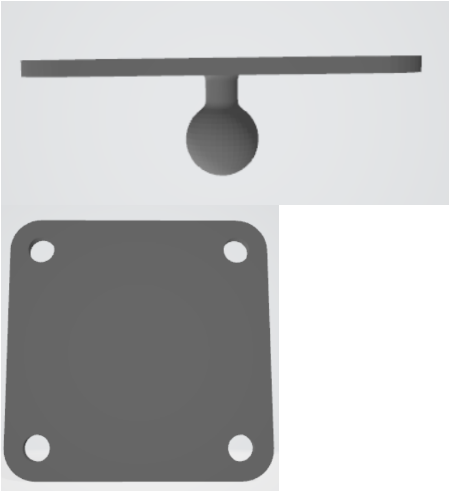


Axis of the servos

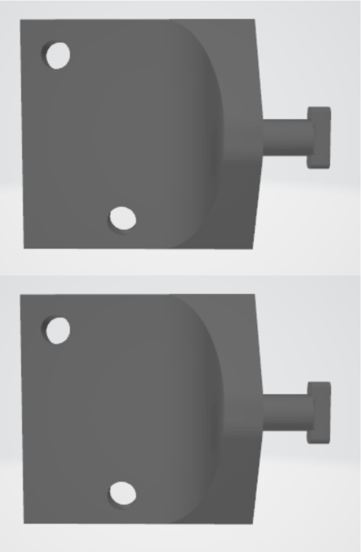


Ballbot

Base with the rotation ball



Fixed bridles



Simulation Strategy

In order to create the simulation of the robot, we would use the Coppelia simulator. With three servo motors we can simulate the board treatment. With this, we can introduce a ball with realistic physics and understand the movement of it.

Finally, we will put a camera to implement that module.

Ballbot

Foreseen risks and contingency plan

Risk #	**Description**	**Probability** (High/Medium/Low)	**Impact** (High/Medium/Low)	**Contingency plan**
1	That the ball leaves the board while the board is moving.	Hight	Hight	When the robot is not detecting the ball, stop the execution of the robot. A possible solution will be putting walls on the edges of the board.
2	Non-robust algorithm that doesn't calculate the path that leads to a possible solution.	Medium	High	Have other search methods with more features. (Dynamic Programming). If any methods find a solution, display an error message on the screen.
3	That the material of the maze board is very heavy.	Medium	High	Change the board material type to a lighter one or make the board smaller.
4	Robot components do not provide enough	Low	High	Immediately notify the budget department (Teaching team).

Ballbot

	performance in case the complexity of the robot increases.			
5	Don't show the path of the ball on the screen.	Medium	Low	Go with another type of feedback.
6	That the agent takes a long time to solve the game.	Medium	High	Have other search methods with more features. (Dynamic Programming).

References

This project has been inspired by the following Internet projects:

https://la-tecnologia.com/tips-arduino/equilibrador-de-bola-con-plataforma-stewart/