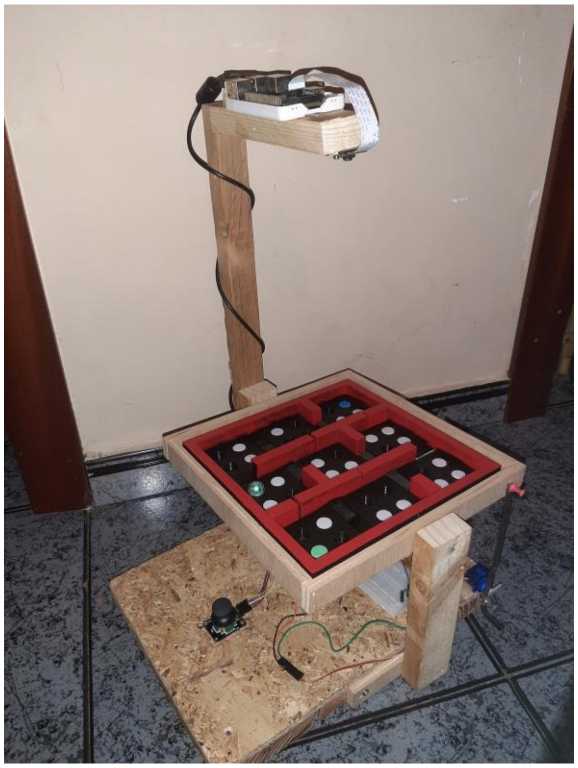

TiltBot

Interactive maze-solving entertainment device.

FINAL SPRINT
DATE: 12 June 2024

Jaime Martinez Coca - 1634495
Robert Pearson Ruiz - 1630916
Sergio López Parejo - 1634093
David Fuentes Insa - 1637892

Table of Contents

Final Sprint
Date: 12 June 2024

TiltBot

Interactive maze-solving entertainment device.

Project description

TiltBot is an entertainment device where you try to get a ball from one side of a maze to the opposite side, just by tilting it. Along the way, you'll need to dodge holes, or you'll have to start over! Using it is simple: you can tilt the maze platform where the ball is located with a joystick.

If crossing the maze is too difficult, TiltBot has an extra feature. You can switch to an automatic mode, where AI will solve the maze, showing you one of the possible paths to the end. (For now, the algorithm is somewhat limited. See the Limitations section.)

Additionally, to increase the amount of fun, the maze format can be customized. You can open or close holes, as well as restructure the walls to choose your own difficulty.

Electronic components

This is the list of the used components:

- *Arduino UNO R3*
- *Raspberrypi 3B+*
- *Joystick*
- *x2 Microservo*
- *Camera Raspberrypi v2*

Action Module

This module is the one in charge of the actual movement of the labyrinth that allows the puzzle to be solved.

Action Hardware

In order to move the structure, we make use of two different servo motors, one for the outer layer, and the other for the inner layer. Each of these servos move in different directions, allowing the structure to tilt in any angle needed. For a deeper understanding check Structure.

To be able to control manually these servos we have also connected a joystick, which moves both motors accordingly to the direction and amount of pressure applied on it.

All of this is interconnected by an Arduino UNO R3 module.

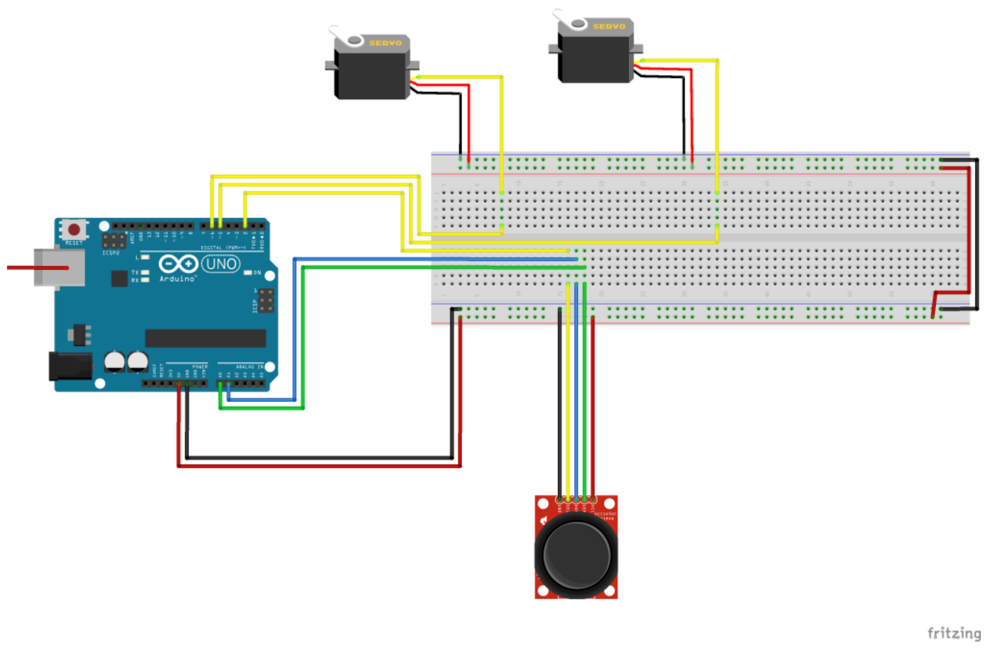

Figure 1. Action module HW circuit.

- *Servo_1 SEL is connected to digital pin 6*
- *Servo_2 SEL is connected to digital pin 5*
- *Joystick SEL is connected to digital pin 2*
- *Joystick VER is connected to digital pin A0*
- *Joystick HOR is connected to digital pin A1*

Action Software

To make the servo motors move, we have implemented some code in the Arduino.

The code is divided into two sections, one for manual movement (using the joystick) and another for automated movement using the Raspberry Pi.

To start, we initialize the game mode in manual mode and enter the main program loop.

In this main loop, we check if the joystick button has been pressed. If so, we switch modes (if we were in manual mode, we switch to automatic mode, and vice versa).

Then, if we are in manual mode, we simply read the tilt angles on the X and Y axes of the joystick and translate them into corresponding angles on the two servo motors.

If, on the other hand, we are in automatic mode, a signal is sent to the Raspberry Pi to indicate that we are starting the automated algorithm. Then, we enter an internal loop where the Arduino will wait for movements from the servo motors until receiving a completion message.

When it receives this completion message, it will exit the loop, return to manual mode, and go back to the main loop.

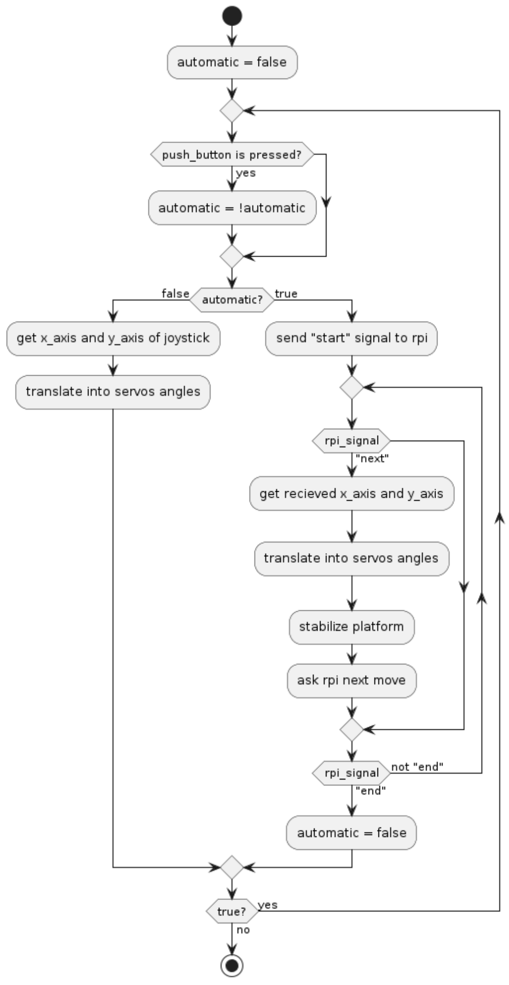

Figure 2. Action Module SW flux diagram.

Vision Module

The vision module is responsible for the automated maze-solving part. The idea is to move the ball across the maze following the walls, so it does not fall into any holes.

When requested by the Action module, it will take a photograph of the maze from above and find a path from a starting point to an endpoint. It will then give instructions to the action module to try to move the ball along the path.

Vision Hardware

The hardware part is very simple. It consists of a Raspberry Pi 3B+ connected to a camera positioned above the center of the maze, and also connected to an Arduino UNO R3.

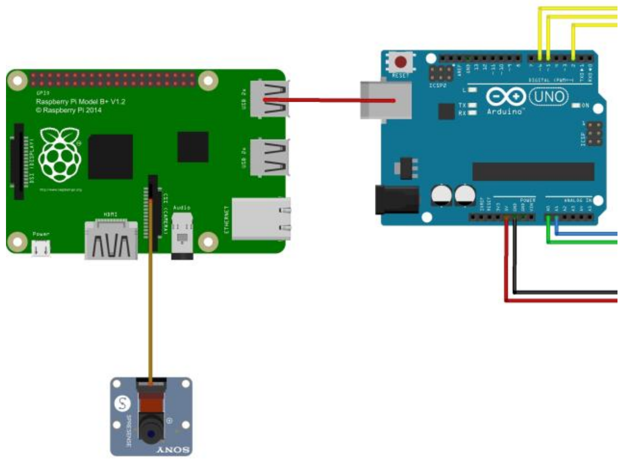

Figure 3. Vision Module HW.

Vision Software

The code waits for a start signal from the Arduino to begin the resolution algorithm. When it receives the signal, it takes a photograph of the maze. From this image the Raspberry detects the walls, the starting point, and the ending point using the color of the pieces to differentiate them.

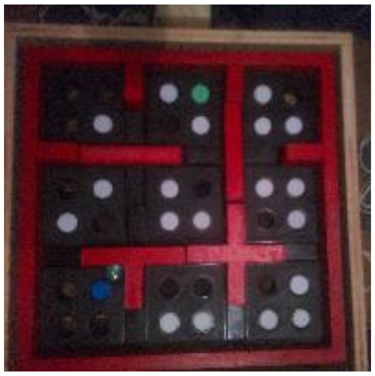

Figure 4. Initial photo of example.

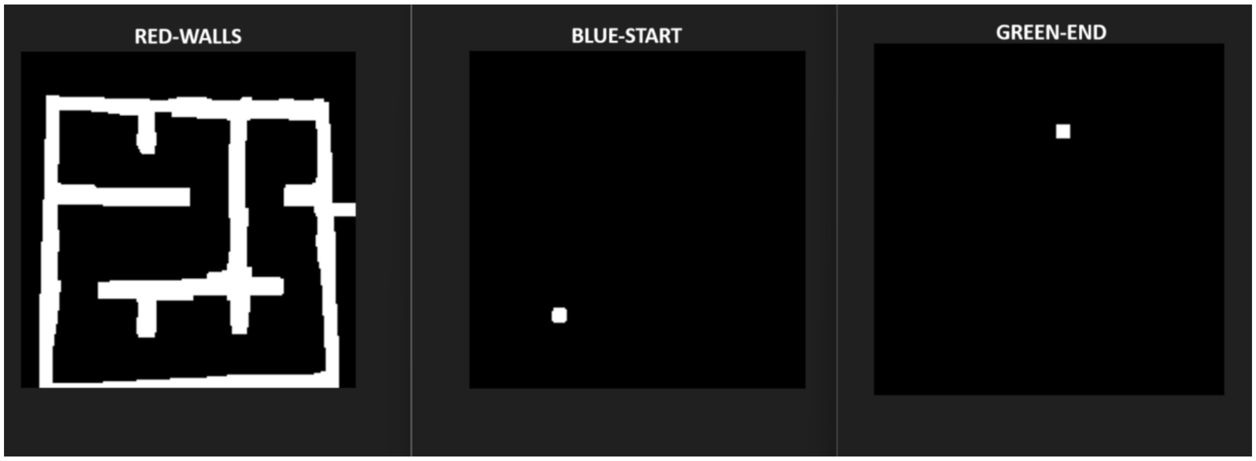

Figure 5. Detected parts of the example.

Then, we find all the corners, and calculate which of those corners are closer to the start and end points.

Once we have these closer corners, we search for a path across the corners of the labyrinth.

In this search we must consider that we are moving a ball, so when we move it to a "internal" corner, it will continue moving until it collides with the next wall.

If there is a valid path, then this is returned, and sends point by point the movements to the Arduino. From one point to the next one the Raspberry waits for the correct signal from the Arduino.

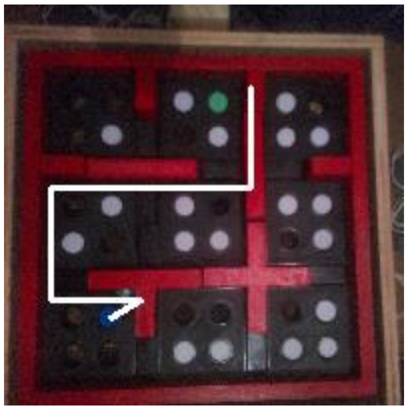

Figure 6. Valid path.

Vision Software Limitations

The pathfinding algorithm uses the walls to move the ball to the end. For this reason, there are some maze layouts that it is unable to solve, which are those where there is no way to access the final area through the walls. An example it cannot solve is the following.

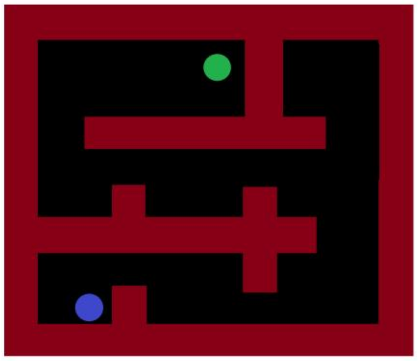

Figure 7. Unresolvable maze.

The rest of the cases work correctly, at least on perfect images.

However, when applying the algorithm to real images, it does not work in 100% of the cases. Through testing, we have concluded that this is due to, among other things, imperfections in the materials/structure.

Structure

The structure is one of the most important parts of this project. To create a customizable maze, we had to come up with a structure that would allow us to rearrange where the walls that delimit the path of the maze should be, and the holes through which it should avoid falling. We designed the platform of our maze with holes where different pieces of walls fit and holes where plugs fit that allow the ball to pass over it. In addition to this, the structure has an exterior frame that will allow the platform to rotate on an axis. This has been achieved by means of a servomotor that will be attached to this frame by means of an elongated metal plate. As for the movement in the other axis of the inner platform, we have placed another servo attached to the bottom of the inner platform of the maze and joined to the outer frame by another plate. This allows us to obtain the desired movement in both axes.

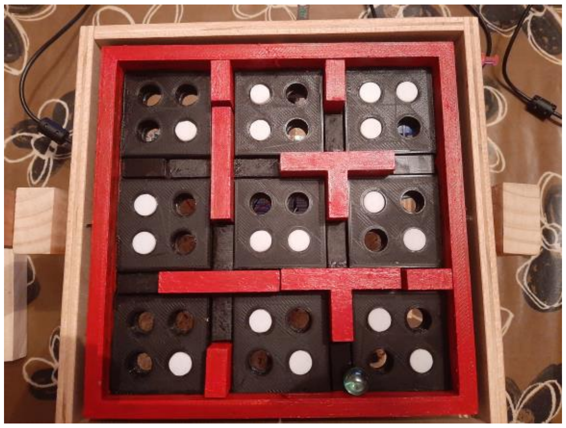

Figure 8. Final platform sample.

Extra components and 3D pieces

- Labyrinth base
- Plugs
- x4 exterior walls
- x4 exterior floor pieces
- x2 cross shaped walls
- x2 T shaped walls
- x2 I shaped walls

- We made a platform that has nine main isles of floor. Among them, four central X-shaped holes remain in the center of the platform, and eight small holes are located on the exterior side. Also, each piece of the floor has four holes in it, that can be fitted with our plugs in order to make the maze either easier or harder.

- Regarding the plugs, they are all the same, and they are of such a height that allows the ball to pass over it effortlessly. These plugs allow us to change the location of the holes, as well as modifying the difficulty of the maze, increasing or decreasing the number of wholes through which the ball can fall through.

- All the different types of walls and floors are also made so that we can change the shape and paths of the maze. Each piece has either floor, wall or both depending on the piece. On the one hand the floor parts are only as tall as the labyrinth base, so that the ball can go over these pieces without any struggle. On the other hand, the wall parts are made taller, so that the ball can not go pass them, giving the shape of the path that it needs to do in order to get to the end.

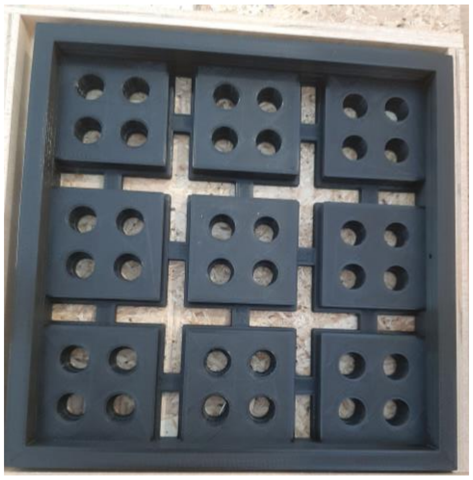
Figure 9. 3D platform piece.

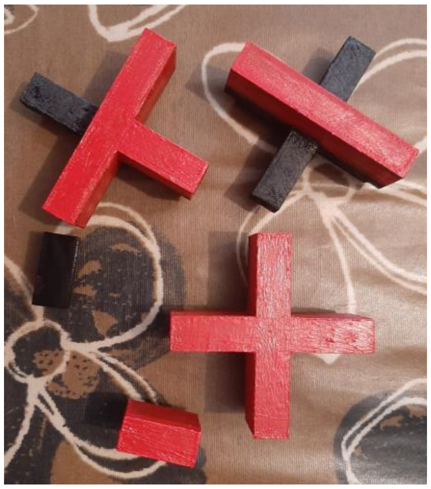
Figure 10. 3D wall pieces.

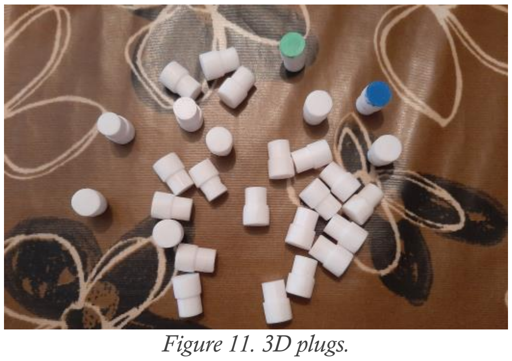
Figure 11. 3D plugs.

Foreseen risks and contingency plan

Risk #	**Description**	**Probability** (High/ Medium/ Low)	**Impact** (High/ Medium/ Low**)**	**Contingency plan**
1	Computer vision does not always solve the maze (it finds a path, but the ball deviates / it does not find a path	High	Medium	Perfecting the materials with millimetric precision and improving the image quality.

Table 1. Foreseen risks and contingency plans.

References

This project has been inspired by the following Internet projects:

Original idea for the project by Antonio Álvarez:
https://www.youtube.com/watch?v=PMSr5L0SD24

Tutorial - Usage of Joysticks with Arduino: https://programarfacil.com/blog/arduino-blog/joystick-con-arduino/

Tutorial - Usage of Servos with Arduino: https://programarfacil.com/blog/arduino-blog/servomotor-con-arduino/

Tutorial - Send data from Arduino to Raspberry (Serial connection):
https://www.youtube.com/watch?v=-3swby4ryU4