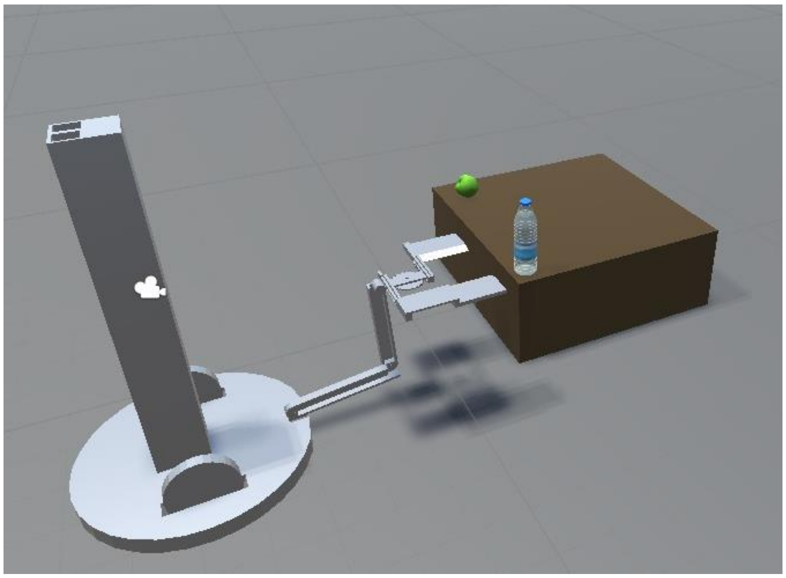
RPL - Computer Engineering. UAB (Barcelona) 2019-2020



CAREBOT

Assistance robot for the elderly and/or reduced mobility people.

PROJECT SPRINT #5.
DATE: 26 May 2020

Laura Roldán Vilardell
Rubén Vera Garcia
Pau Cano Ribé
Víctor Pérez Jiménez

Table of Contents

Sprint #5
Date: 26 May 2020

CAREBOT

Assistance robot for the elderly and/or reduced mobility people.

Project description

Carebot is a robot focused on the care of elderly and mobility-impaired people. Its users are able to communicate with Carebot by stating clear commands out loud. The robot will follow the command as much as possible, if such command is one of the commands available to the user.

The robot will be able to move around the house when given the order, positioning itself via a QR-location system. It will be able to locate and identificate some objects, which it will be able to pick up and handle to the user once the user has requested such.

For these reasons, it's specially useful retrieving objects that have fallen on the ground or on hard-to-reach places, as well as carrying objects from room to room that might be too heavy for the user.

Electronic components

This is the list of the proposed components:

- Preamp Electret Microphone-PRO-0055
- Ultrasonic distance sensor HC-SR04 - SEN-0054
- Mini player MP3 - PRO-0111
- Micro servo SG90 - SRV-0016
- Module Wi-Fi ESP8266 (ESP-01) - ERF-0019

- Adjustable Voltage Regulator (LM317) - CMP-0006
- Speaker - 3W - PRO-0162
- Camera 2MP OV2640 ARDUINO - B0011
- 9376 MOTOR MINI 4.5V A 9V - 141
- EF92A MICRO SERVO 1808 - EF09082
- AVR ARDUINO UNO REV 3
- DFRobot_Speaker_v1.0_SKU__FIT0449

Software Architecture

The following image shows the software architecture used in this robot, divided in the internal logic, the input, and the actuators.
The internal logic is divided in 6 main modules: the image recognition module, the speech recognition module, the command processing module, the movement module, the arm module, and the central processing module.
Both the central processing module and the image recognition module are divided in two sub modules: the position processor module and processor module; and, the object recognition module and the QR processor module; respectively.

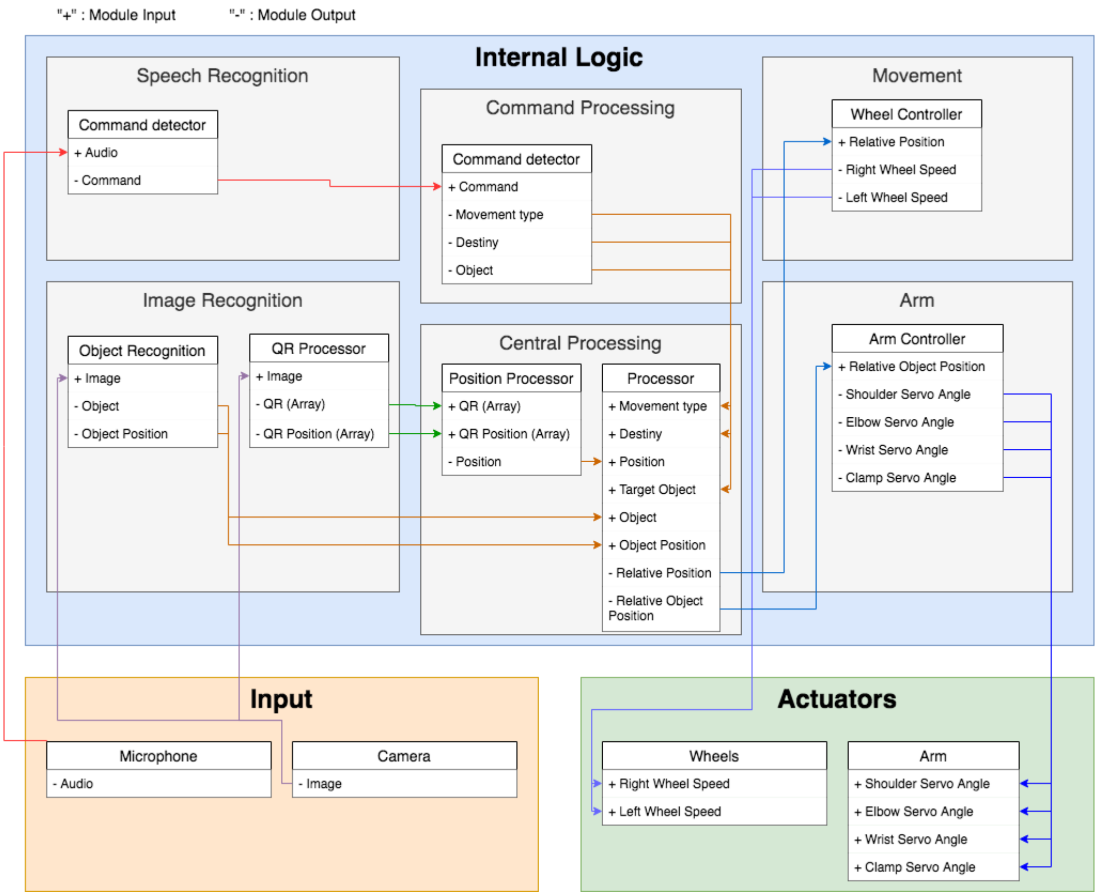


Image Recognition module: this module is responsible for processing the image input that the camera generates. It's divided in two modules: Object recognition module and QR processor module. The ***Object recognition module*** tries to match the image received in the camera to a database of objects known to the robot, and, if such object is found, the object, and its position is used as module output. The ***QR processor module*** tries to find QR codes in the image from the camera. If such QR code is found, its ID and position is used as module output.

The object recognition part works thanks a pretrained neural network: yolo v3. This is a model that uses a convolutional neural network to detect objects from an image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

The architecture consists of few initial convolutional layers that are responsible for extracting features from the image, while the full connection layers predict the probability of the output and the coordinates of the object.

Also, in this module, the distance from the center of the image to each corner of each bounding box is calculated, because is an information that we will use in the simulation to know how many degrees the robot must turn to take the object.

This module is programmed with Python and uses the ImageAI library to control the network.

Speech Recognition module: this module is responsible for interpreting the words spoken by the user and transforming them into commands, which are used as the module output. It is implemented inside the Unity environment.

This module works with the Microsoft API, System.Speech.Recognition and is programmed with CSharp.

Command Processing module: this module receives as input the commands produced by the speech recognition module. It processes these commands, determining whether it’s a command regarding an object, or a movement.

Central Processing module: this module is responsible for the processing of the outputs of the other modules explained until now. It is separated in two sub modules: the *position processor* module, and the *processor* module. The position processor module uses the output of the QR processor module to determine the current position of the robot. The processor module uses the output of the position processor, command processor, and object recognition modules to determine which movements should be made by the wheels and by the arm.

Movement module: this module controls the main movement of the robot, adjusting the speed of the right and left wheel motors, using the output of the processor module to determine how much it should rotate or move.

Arm module: this module controls the position of the arm based on the relative object position received from the Processor module, and adjusts the servomotors accordingly.

Of these modules, all of them, except the QR Recognition module, have been finished to some degree. We can separate these modules as being arduino-ready, or being done, but not arduino ready, due to being centered on the simulation.

As such, these modules are:

- Arduino Ready:
    - Object Recognition module
    - Speech Recognition module
    - Arm movement module
- Simulation Ready:
    - Command Processing module
    - Central Processing module
    - Movement module

Amazing contributions

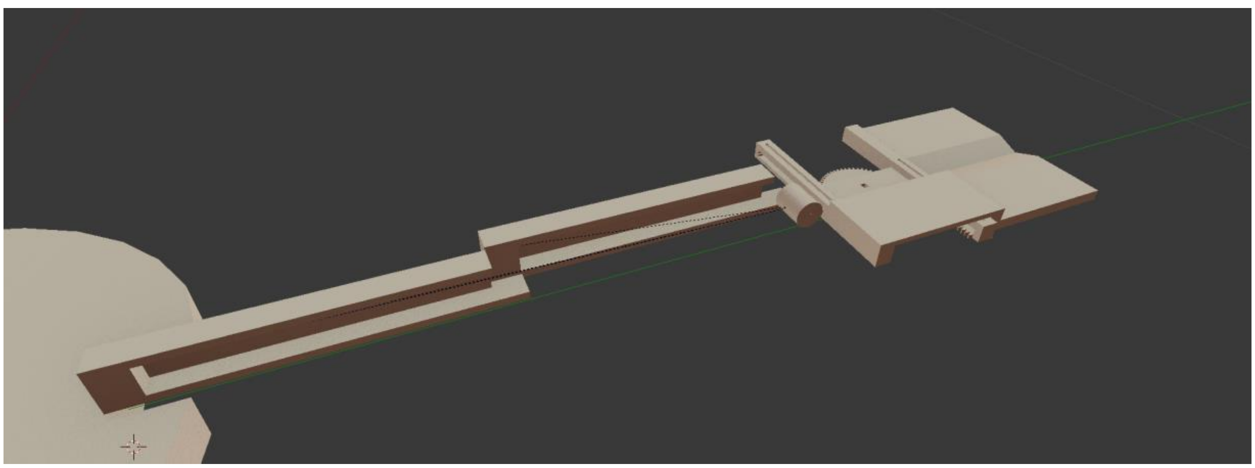
Carebot is a project that aims to accompany senior citizens. It aims to help those people with reduced mobility and who live alone, during their day to day. Especially during these turbulent times that we are living in, where the elderly barely have a human help by their side, the Carebot is a helper for these people. Thanks to its robotic arm and its ability to recognize voice commands, it will be able to grab an object chosen by the user and bring it to his position.

We think that this prototype is very useful for our society, and in addition, it is a robot that would be used for the day-to-day of the elderly. If we achieve all the goals that we set, we would get a prototype that would be able to do small things, but will have an ambition for the future, because with more knowledge, effort and money, it could become a very useful element in the real life.

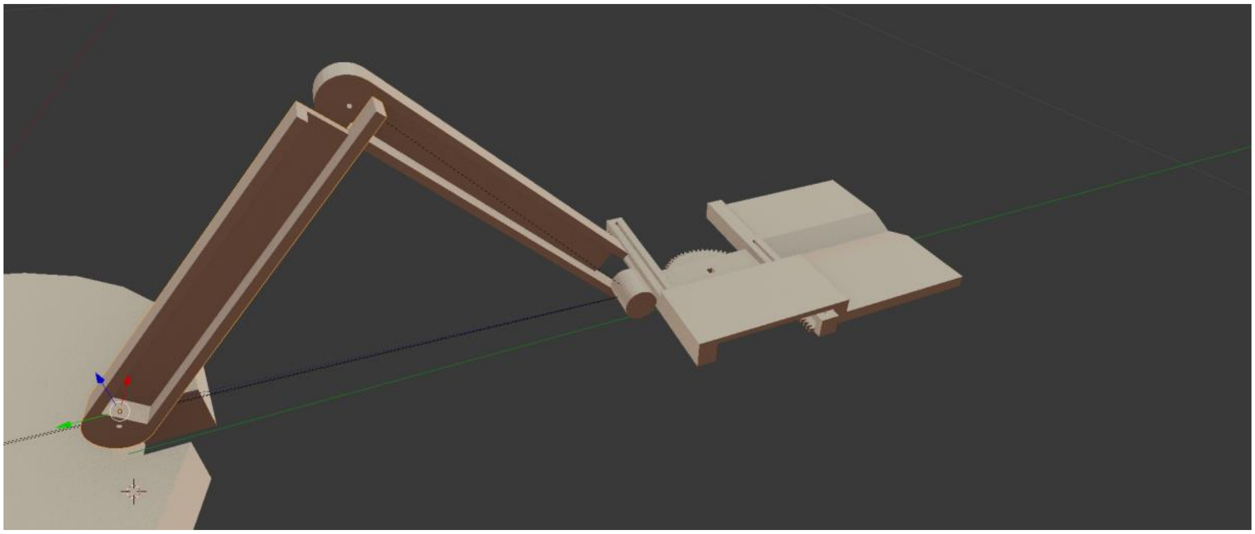
Extra components and 3D pieces

- 3D pieces for the arm
- 3D pieces for the body

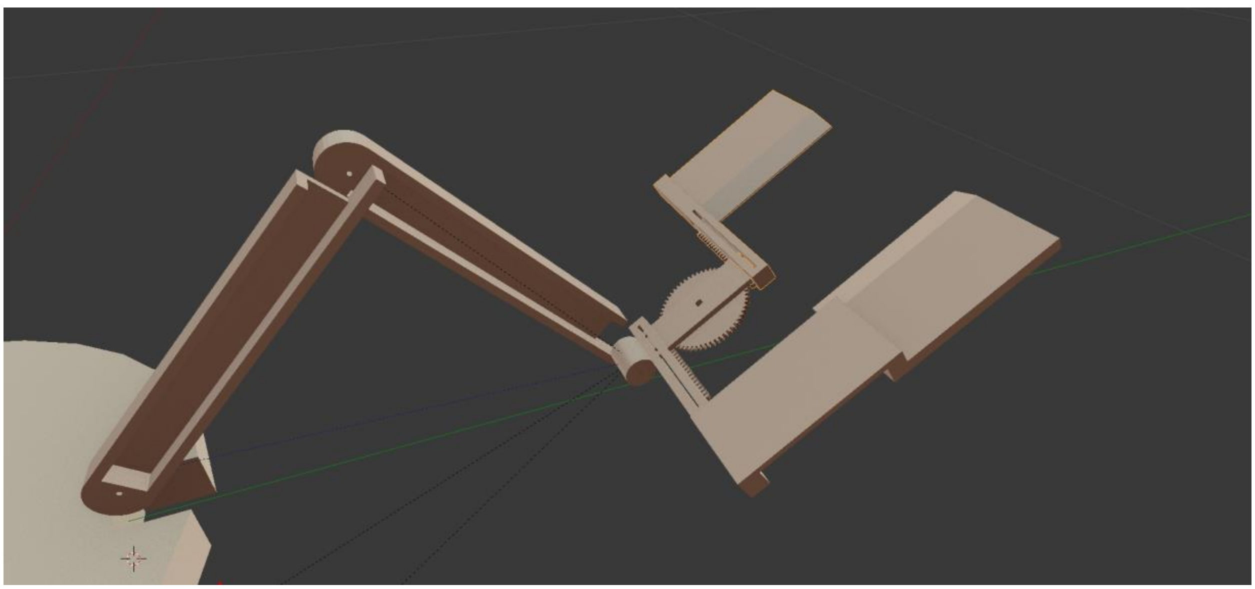
The 3D pieces for the arm are used as arm, lower arm, and clamp of the robot. The clamp works using a servo attached to a gear, that, when rotated, moves the clamp parts opposedly, joining, or separating, them. Such movements can be seen on the pictures below



Robotic arm on base position

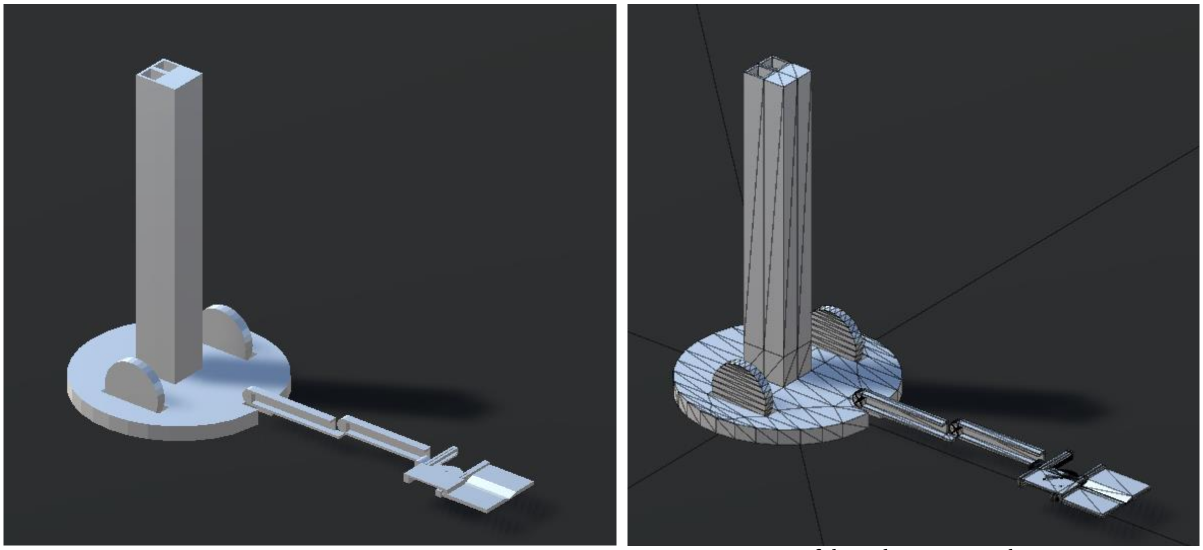


Robotic arm with elbow rotation

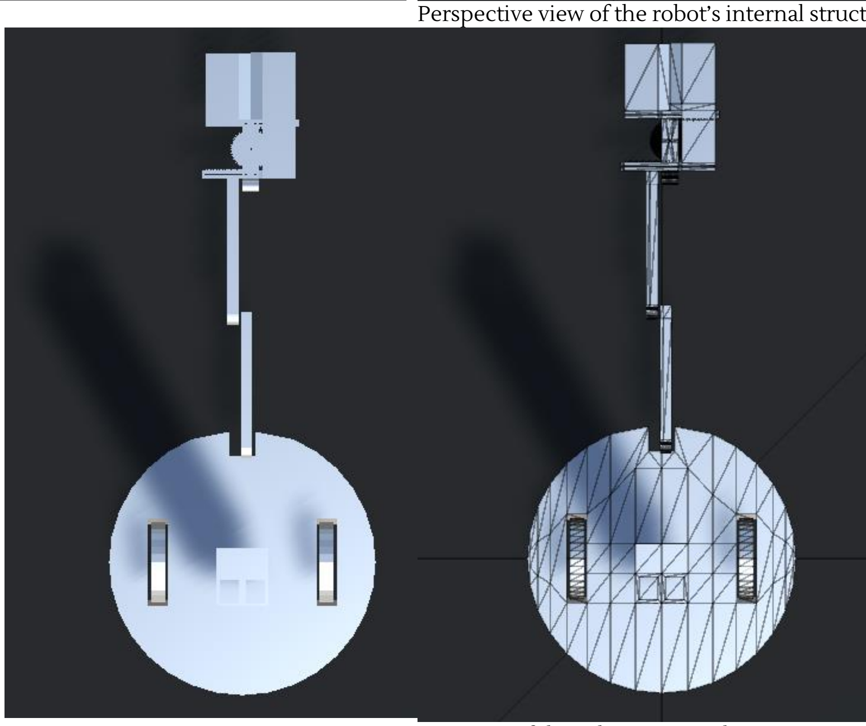


Robotic arm with elbow and clamp rotation, and clamp opened

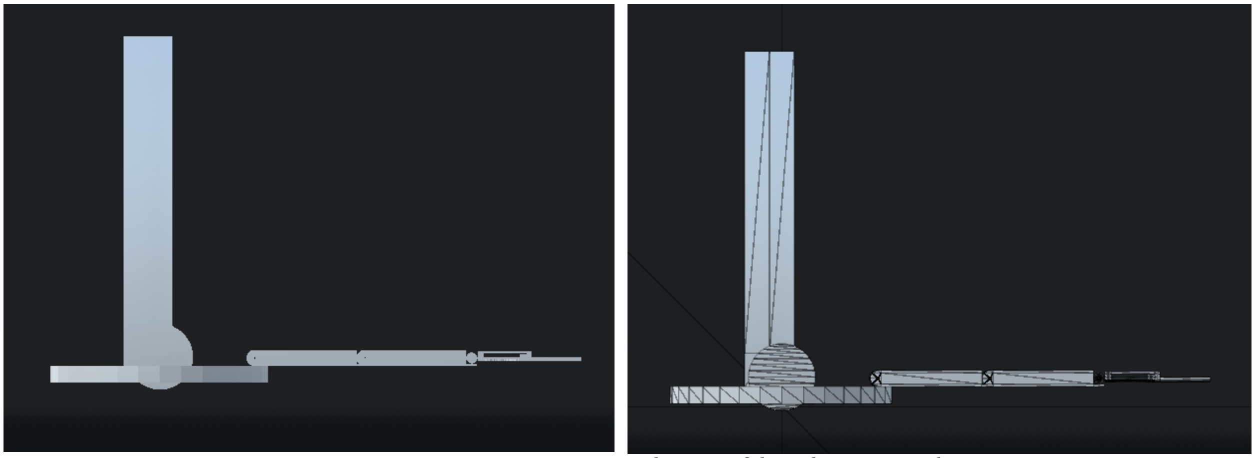
The rest of the 3D printed parts will be used as the main structure of the robot. It will consist of a circular platform, with rectangular holes for the wheels, and a square pillar, where the camera and microphone will be attached. On top of the platform, there will also be attached the robotic arm.



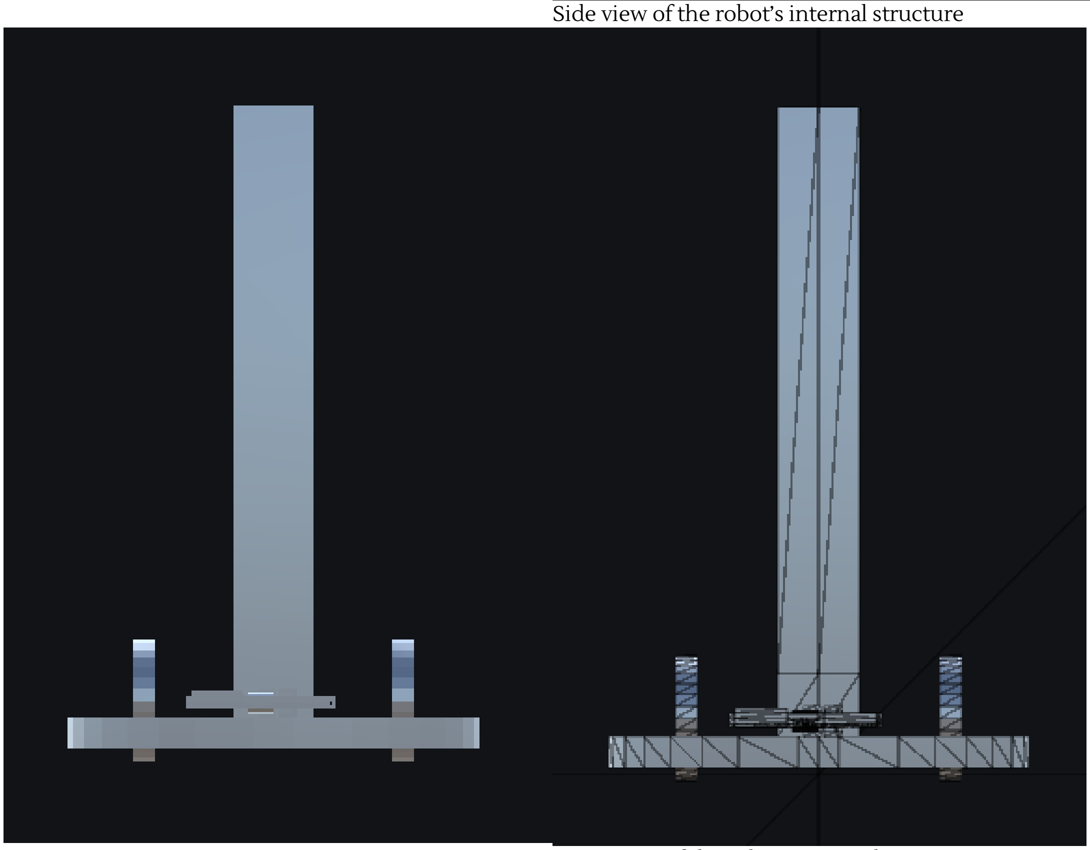
Perspective view of the robot's internal structure



Top view of the robot's internal structure



Side view of the robot’s internal structure



Front view of the robot’s internal structure

Foreseen risks and contingency plan

Risk #	Description	Probability (High/Medium/Low)	Impact (High/Medium/Low)	Contingency plan
1	Not being able to test the camera module	High	High	Test module with pre generated images. Assume it is working correctly until proven otherwise
2	Interconnection between third-party programs	Medium	High	Use other compatible third-party programs
3	Problems with the Internet Connection	Low	High	Connect robot via USB cable to computer, use computer as access node to internet
4	Clamp not grabbing objects correctly	Low	Medium	Adjust arm kinematics variables, giving offset if needed
5	Arm not able to lift objects due to their weight	Low	Medium	Reduce arm components weight and use higher power servomotors
6	Not able to recognize	Medium	Medium	List of recognized

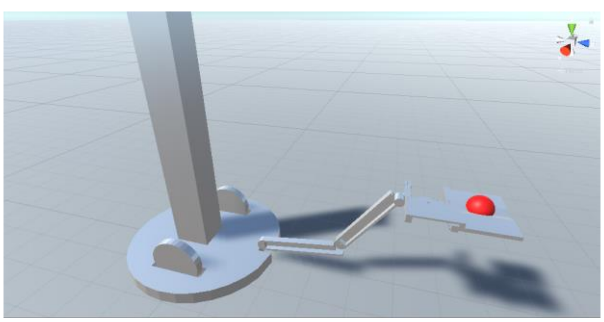
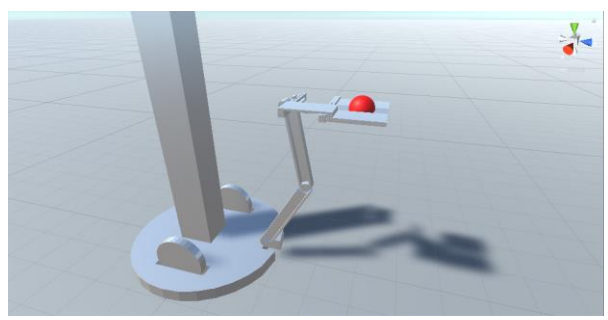
	similar objects and differentiate them			objects limited and different between them
7	Too much distance to see the qr codes of movement	Medium	Medium	Reduce the area of movement of the robot in a smaller room.

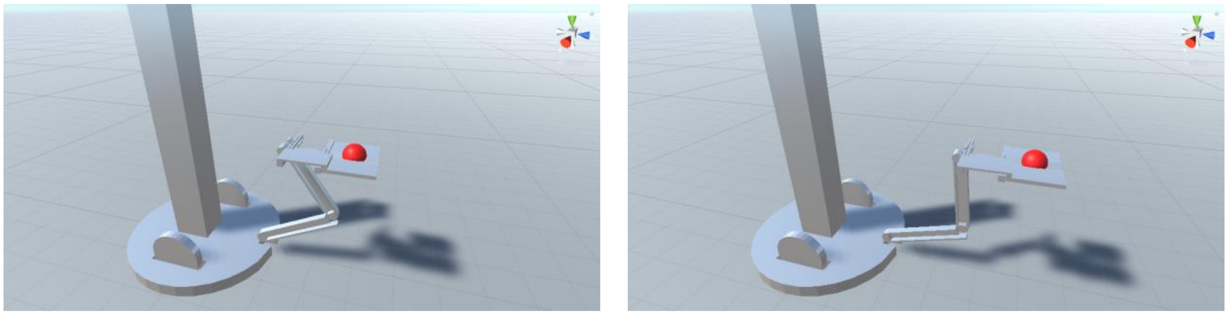
Strategy for validation, testing and simulation

Testing a robot of this kind without an actual physical model is a hard task to accomplish. That's why we have reached the conclusion modular, software-oriented tests are our best bet, but there will be some modules tested in group.

For modules regarding the interaction with specific hardware, testing is done using third-party software, such as Unity3D.

The arm movement module has been tested in Unity3D, using a simple object as a target for the arm. We've tested that the arm rotates each of its parts correctly to get the clamp as closely as possible to the base of the target object.



Unity3D screenshots showing the correct movement of the arm

The Object Recognition, at the beginning, was tested using the computer's camera by analyzing the objects depicted in the pictures. It worked independently from the other modules.

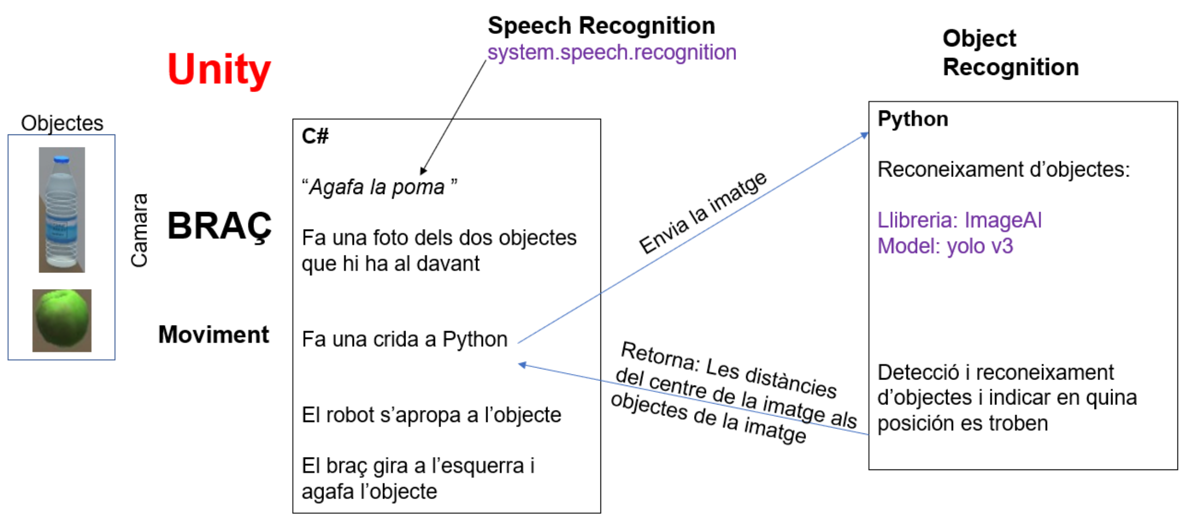
The same way as before, the Voice Recognition was tested using a serial communication between C#, where the module is implemented, and Arduino. In this case, we tested the commands with a simple demo, that consisted of turning on and off a led. Later, this module is integrated in the simulation environment successfully.

It's been possible to do a group test of the arm module, the object detection module, the voice control module, and a simple movement module. This test is done inside Unity3D. The main robot controller uses a modified voice detection module to detect which commands are spoken to the robot. When the robot is ordered to take an object from those that are in the test (apple or bottle), the main controller calls a Python file, which processes the image printed by Unity's camera, and returns which objects are on screen, and their pixel coordinates. Using this information, and data about the camera's technical details such as the focal length and sensor size, the distance from the camera to the object, and it's relative horizontal and vertical positions are calculated. Once this informations is calculated, a simple movement module rotates, if necessary, and approaches, if necessary, the robot to the desired object, and, when nearby, grabs the object with the arm's clamp, and moves it up, approaching it to what in real life would be the user.

We've seen that using this approach, the bottle is almost always able to be taken, while the apple fails more usually, do to it being confused with a sports ball by the neural network used in the object detection module. Most work should be put into the neural network to avoid this issue.

If the robot, instead, is ordered to just "move", it will move towards the green things in the image, which could be used to make the robot follow a laser pointer.

To sum up, the next figure shows the final result of our simulation:



References

This project has been inspired by the following Internet projects:

- https://rlpengineeringschooluab2018.wordpress.com/2018/05/29/pixi-lamp/
- https://rlpengineeringschooluab2019.wordpress.com/2019/05/29/athenea