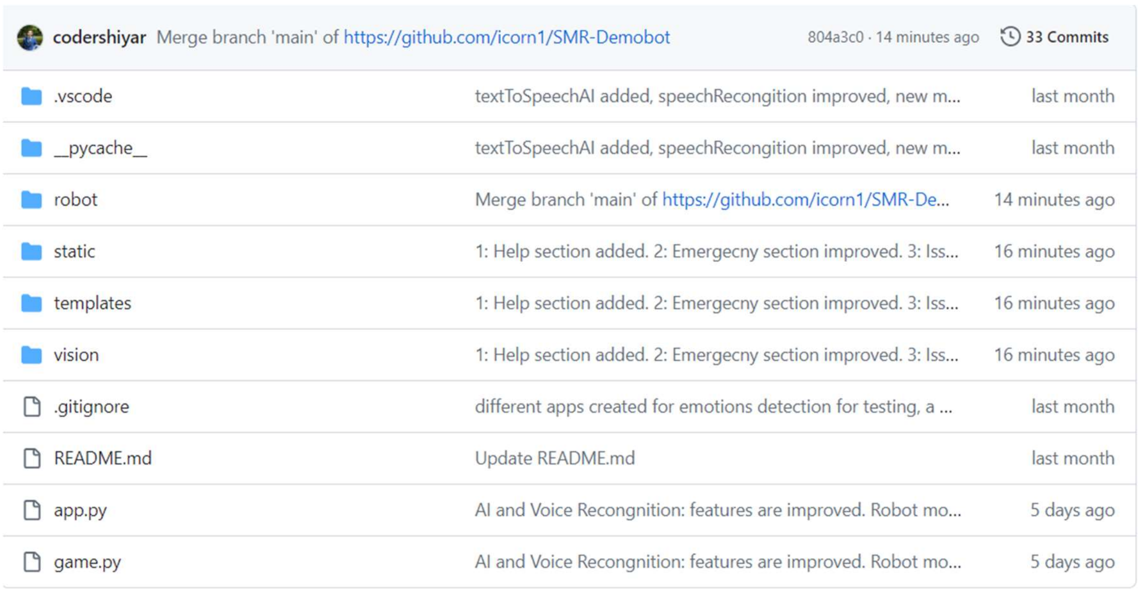
Promobot Project Documentation

Guillermo Forcén, Shiyar Jamo, Ixent Cornella

1. GitHub Repository

1.1 Contents

The GitHub repository is organized in the following folders and programs:



codershiyar Merge branch 'main' of https://github.com/icorn1/SMR-Demobot 804a3c0 · 14 minutes ago 33 Commits

.vscode	textToSpeechAI added, speechRecongition improved, new m...	last month
__pycache__	textToSpeechAI added, speechRecongition improved, new m...	last month
robot	Merge branch 'main' of https://github.com/icorn1/SMR-De...	14 minutes ago
static	1: Help section added. 2: Emergecny section improved. 3: Iss...	16 minutes ago
templates	1: Help section added. 2: Emergecny section improved. 3: Iss...	16 minutes ago
vision	1: Help section added. 2: Emergecny section improved. 3: Iss...	16 minutes ago
.gitignore	different apps created for emotions detection for testing, a ...	last month
README.md	Update README.md	last month
app.py	AI and Voice Recongnition: features are improved. Robot mo...	5 days ago
game.py	AI and Voice Recongnition: features are improved. Robot mo...	5 days ago

Figure 1: Screenshot of the repository in GitHub (Source: authors)

Our project is an HTML+CSS+JavaScript Web application for the UI, which is handled via a Flask server using Python, running on port 5000. There is UDP communication handling to control the robots. Precisely, *app.py* is the main program for this repository.

1.2 How it is structured

The *.vscode* and *__pycache__* folders should be ignored, and so is *game.py*.

Arduino folder contains Arduino code that can be tested separately.

robot folder contains the source code of the Kawasaki robots. Keep in mind that this code is only there to store it, and you do not actually needed to run the web application. Instead, what you ought to do with the code is upload it to the Kawasaki robot (in case the robots don't have the code uploaded). All of the files are written in Kawasaki Programming Code (AS).

static folder contains all of the static elements, such as videos, CSS, music, etc.

templates contains the HTML code.

vision contains the python modules for the vision part to run properly.

app.py is the main python handler and the program that needs to be executed in order to start the server.

1.3 How to run the dashboard & get everything working

The first step is to try to run app.py. There are many requirements to be installed, for example, tensorflow. Make sure to install all of them. Once you can run app.py, make sure your setup is the following:

- Laptop with the web application
- Ethernet switch with 3 cables; one to each robot, and one to the laptop.
- USB-C connection from laptop to camera
- USB-A connection from Arduino to laptop
- HDMI connection to the TV (optional)
- Make sure your laptop has power

In case the connection to the robots is failing:

- Make sure your laptop’s Ethernet adapter is on the same network as the robots (192.168.0.X)
- Attempt to ping the robots.

Once you make that sure everything is working, and that the connection to the robots is successful, you can begin to use the application.

There are two URLs to use the web application:

127.0.0.1:5000

127.0.0.1:5000/tv

2. Dashboard / UI

2.1 Contents and features

The dashboard has been programmed to include the following functions:

- **Rock-Paper-Scissors game**: The right robot does the rock-paper-scissors motion three times, and then the robot's choice is seen on the TV and the hand does the selected pose.
- **Dance mode**: The robots can dance in a synchronized motion, to the beat of a family-friendly song.
- **Pong game** (optional feature): The robots can play a classic game of pong if the voice command ”Play Pong” is called. This feature is labeled as optional because it was not planned and is not fully finished.
- **People detection**: With the use of AI and the Intel camera, the detection of people in front of the robot can be done and shown in the camera for the user to see itself.
- **Movement detection**: With the use of AI and the Intel camera, the user can see their exoskeleton outlined on the TV.
- **Feeling detection**: Again, using AI, the detection of people's emotion is done for the user to interact again with the robot.
- **TV Mode**: Finally, we have the TV Mode, which makes both robots go to the corner and straighten themselves, so that the TV is fully visible and advertisements can be shown.

The other functions that are not written on the touchscreen but are rather built implicitly are:

- **Speech Recognition**: We've implemented speech recognition and voice commands so that the robot can be used without using the touchscreen. Captions are also written down on the TV.

- **Emergency mode**: An emergency mode (which puts the robot on hold) can be pressed and also called with speech recognition, and then unpressed so the robots can resume their motions.
- **Hello robot**: We have implemented the ’Say Hello’ feature in the robot. Now, the robot automatically greets people when it detects someone saying ’hello’.

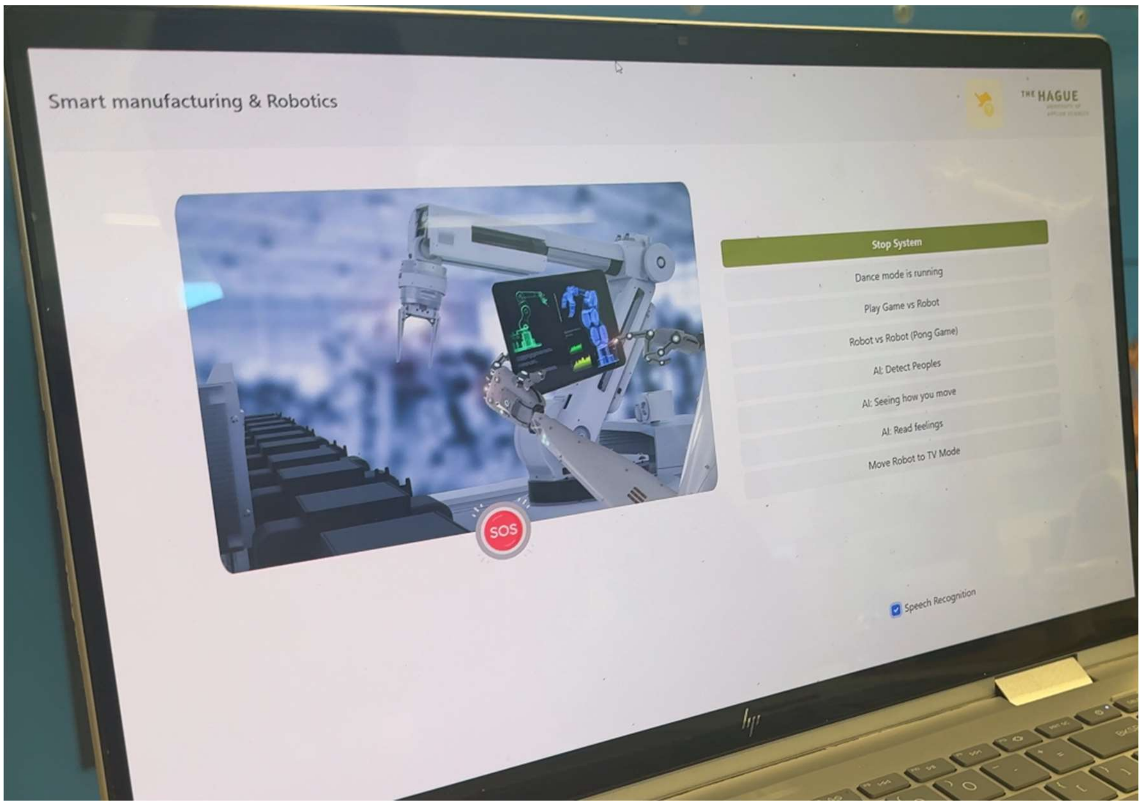


Figure 2: The Dashboard (Source: authors)

2.2 Programming the dashboard

To program the dashboard you’ll need to edit the HTML and JavaScript, just like any other web application built this way. Keep in mind that if you need to change the robot code, it has to be uploaded to the robot each time you change it, and for that you can refer to the section 4 of this documentation.

3. Setup

3.1 How it was mounted

The robots where mounted on individual cages before being on this setup. Then, the original plan was to mount them on a professional setup, custom made, with bendable glass.

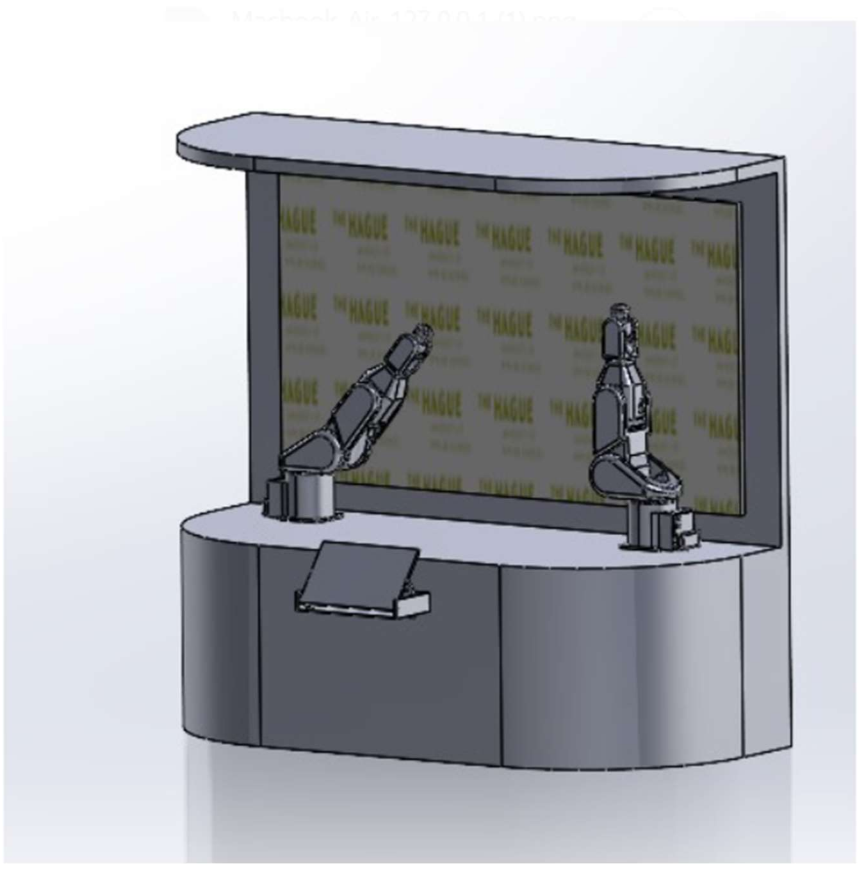


Figure 3: Original setup 3D design (source: authors)

Then, we scrapped that idea since the resources to build it where rather hard to get, and would take more time than we had for the project. Therefore, we focused on creating a temporal setup, the one on the robot lab. The distance between the robots is almost equal as the original plan was.

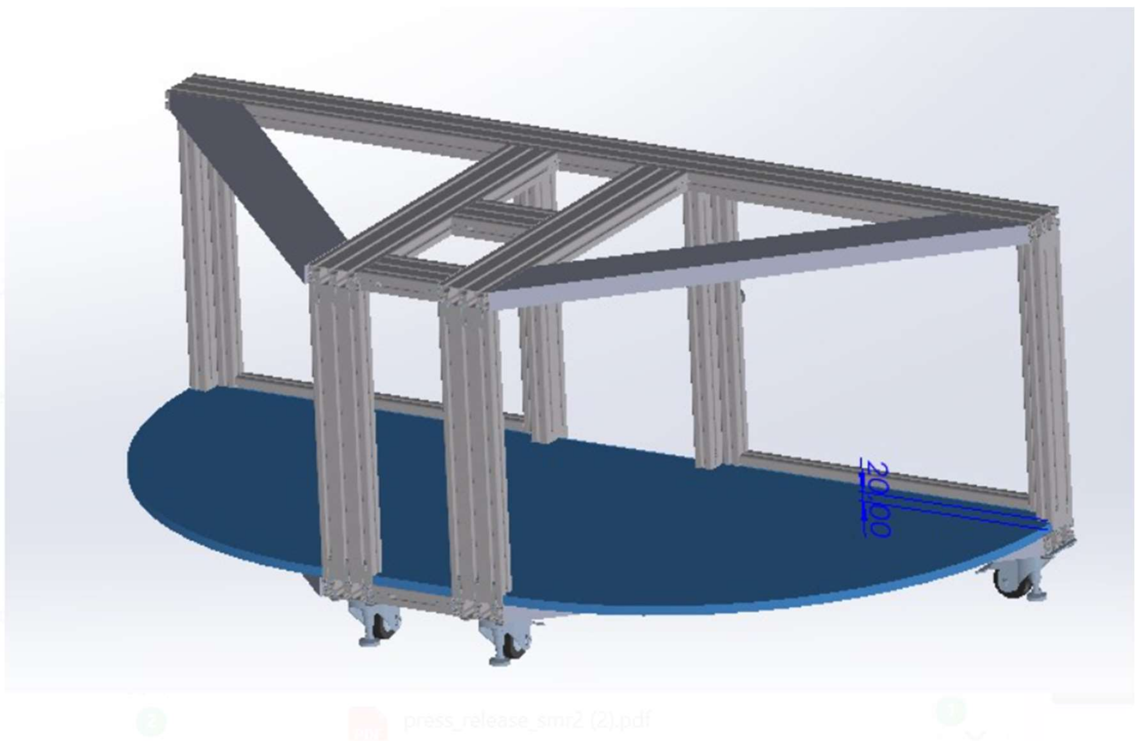


Figure 4: Profile selection to craft the structure for the setup (source: authors)

3.2 Important remarks

If you need to move the robots, keep in mind that one of them has a taller base than the others, to be precise, 10 cm taller. For that, we elevated the robots so that they could match heights. However, you ought to remember that if you need to change the setup. It is not possible to remove the base extension from the robot (we tried but it seemed as if the robot was mounted so it could never be removed from its original setup).

Furthermore, the robot with the camera (the one on the right from the users POV), is significantly slower than the other robot. You'll need to play with the speed parameters (SPEED, ACCEL, DECEL, ACCURACY) to match their speeds.

4. Kawasaki Robots

4.1 Brief introduction to how Kawasaki Robots works

Kawasaki robots have their own language: AS language. This language is similar to assembly code, and has a steep learning curve. It is strongly suggested that you take a week just to get familiar with how the code works and the robot behaves.

To control the Kawasaki robots with your PC and upload code to them, you should use KIDE. KIDE is a complicated IDE at first but once you get the hang of it, it becomes better. To connect to the robot, click the bolt icon and select the IP of the robot. To get all of the programs of the robot, select the first icon of these four:

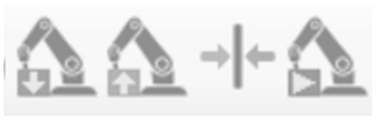


Figure 4: Top corner icons of KIDE (they should be colored when connection is established)

Once you do that you'll have access to all of the programs uploaded to the robot. Make sure that, every time you edit a program, you upload it to the robot before using it. You have to click the second icon, but the smaller one (with the program selected). You might encounter error 6 (in that case, you have to go to the teach pendant and select the program options, then click "Cancel Register"), or error -100 if there was a syntax error (we just restarted the robot at this point).

There are two types of programs that can be ran by the robots:

Regular programs (anything that ends with .as)

Program Control (anything that ends with .pc).

The main difference between these two is that Program Control executes in the background, and regular programs in the foreground. It can be run at the same time as a regular program, and that allows you to control the flow of information, for example, we used a .pc program to run a regular program, and then keep using that .pc program to listen to a socket and if there was a message there, stop the program (like an emergency button). .pc programs can be useful if you need to run multiple programs.

4.2 Our Kawasaki Robot code

Robot one (the one on the right from the users POV) has the following code uploaded for this project:

main_robot1.as	This program is not meant to be executed, but is rather the main foreground program
udp_emergency.as	This program is used to stop the robot in case of emergency
bg_control1.pc	This is the program to run. It will run both main_robot1 and udp_emergency in the background.

Robot two (the one on the left from the users POV) has the following code uploaded for this project:

main_robot2.as	Same as robot 1, but built for the other robot.
udp_emergency.as	This program is used to stop the robot in case of emergency
bg_control2.pc	This is the program to run. It will run both main_robot2 and udp_emergency in the background.

You can check all of this code in GitHub.

4.3 References and important manuals

Please, check these manuals on the Kawasaki robots. Everything that you need will be explained in detail there. Also, refer to the robotexchange practicum for some useful commands.

MANULS PAGE:

https://drive.google.com/drive/folders/1WUVhSoy2YsOI5mfBlVynOUJjK-DIkwT9

ROBOTEXCHANGE:

https://www.robotexchange.io/t/kawasaki-robot-hands-on/637