UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS

HUMAN-ROBOT INTERACTION (HRI) WITH ROBOT PEPPER

Authors:
Hernan Indíbil de la Cruz Calvo.
Alejandro Zornoza Martínez.

April, 2018

Índice de contenido

1. INTRODUCTION **1**

2. OBJECTIVES **2**

3. PRELIMINARIES **2**

3.1. CHOREOGRAPHE 2

3.2. HOW TO CONNECT TO PEPPER 7

4. SPEECH RECOGNITION **7**

4.1. ANIMATIONS 8

4.2. MOVEMENT 10

5. FACE RECOGNITION **12**

6. DIALOGUE **15**

7. CONCLUSIONS **21**

8. REFERENCES **23**

ANNEXES **24**

ANNEX I ENLACES DE CONTENIDO 24

1. INTRODUCTION

Actually with the evolution of technology and advances in recognition and speech algorithms, the capacibility of the robots increment so fast, that it is the reason why a lot of companies are working on robot development, robots will provide humans a lot of tools that they could make their life more easier.

With the advance accomplished in the HRI(human robot interaction), we can provide to the robots habilities that help them to interact with humans and exchange information between them.

The HRI(human robot interaction) is a multidisciplinary field with a lot of contributions from artificial intelligences, mecanics advances, robotics design, natural language understanding and social science.

These set of science are working all time from improve the capabilities of the robots, making them strongers with better motors, making them more intelligent, making them more social and finally improving their natural language understanding and better in people recognition.

We have learnt during practices how the robot acts depending diferent conditions and enviroments and now we are able to try to work with one of the most advanced robot in HRI that it is called Pepper.

Pepper is a human-shaped robot, that it was design for being a genuine day-to-day companion, which it has the hability to perceive emotions and answer depending the type of them.

The body of Pepper it is preparied for communicate with the people with the most natural and intuitive body language, giving it a friendly actitude and voice.

With the memory capability the user can personalise the robot downloading the software applications and adapts himself to user tastes and habits. We have to remember that his habilities are provided by researchers groups from around the world and the joing of all of them permits it design.

In this project we are going to try to development our personal behaviours for Pepper and learning who a robot with these caracteristics acts, especially in face recognition, speak dialogs, hear words and move around following special orders defined by us.

2. OBJECTIVES

3. PRELIMINARIES

3.1. CHOREOGRAPHE

We have to provide to the reader the enough knowledge about the development environment called choregraphe. We use it in this project to perform the human-robot interaction with Robot Pepper. One of the benefits of using this enviroment it is that the user do not need a lot of knowledge about programming in code. The reason is simple, it provides a toolbox diagram enviroment where the user can move and connect boxes that are defined to simulate the robot behavior.

We will provide to the reader a tiny guide of using this enviroment and how we use to design the robot behaviors.

First of all, we have to show the first interface page of choregraphe, but we have to remember first that the choregraphe installation package include other importants tools that we have to tell to the reader.

The installation package includes:

- Choregraphe 2.5.5 development tool: That it is the enviroment for design. We will explain more about it in other sections after that.
- NAO Documentation 2.5.5 : All relevant information about robots, enviroment, how to install, and uses are included there. It is importante documentation that the user can use to perform his hability while he is working with the robots, it includes documentation about Pepper, Nao and Romeo.
- Memory Backup 2.5.5: It provides the posibility of safe the robot state when we are working on it, it can save previusly installations and we can recover them if the user think it it necesary. This tool have two principal uses, the first one, save data installed on the robot and the second one restore data from the user computer to robot memory.
- Monitor 2.5.5: It provides the tools for monitoring and take information about memory and cameras data, we did not use it for this project.

Well, to start with we are going to explain all the information about the Choregraphe 2.5.5 development tool, and we will teach all the steps that we follow for make the project tasks.

First of all, we open the Choregraphe 2.5.5 development tool and we can see the interface that we provide in the image 1

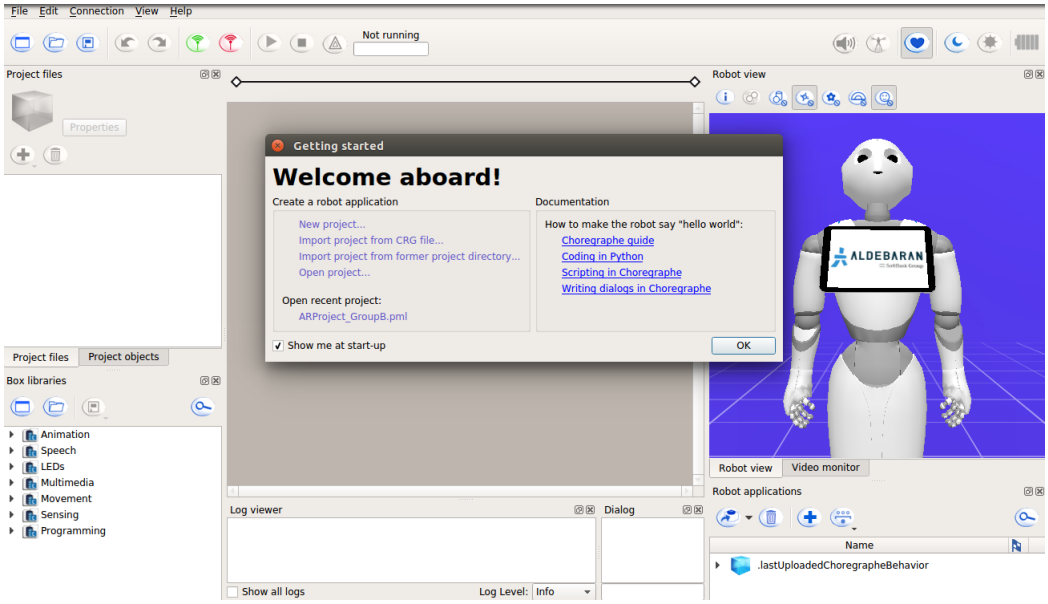

Figure 1: Image of the Choregraphe main menu interface.

At the Getting started window we can see diferent options:

- Create a robot application:
  - New project.
  - Import project from CRG file.
  - Import project from former project directory.
  - Open project.
- Documentation:
  - Choregraphe guide.
  - Coding in Python.
  - Scripting in Choregraphe.
  - Writing dialogs in Choregraphe.

In addition the interface contains a top bar, one left bar and one right bar.

The top bar 2 contains the options for:

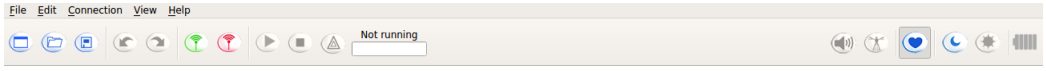

Figure 2: Image of the top bar.

- The top part contains:
  - File options.
  - Edit options.
  - Connection options.

- View options.
- Help options.

The bottom part of this bar provides:

- The first rectangle 3 contains: Create project option, open recent project option, save project.

Figure 3: Image of the top bar first rectangle.

- The second rectangle 4 provides the option of backward or forward in the steps that we do.

Figure 4: Image of the top bar second rectangle.

- The third rectangle 5 provides the posibility of connect and disconnect to a robot or a simulate robot.

Figure 5: Image of the top bar third rectangle.

- The fourth rectangle 6 provides options for upload to the robot and play, stop, debugs and errors output, the state running, not running, volume options and animation mode.

Figure 6: Image of the top bar fourth rectangle.

- The last rectangles 7 provides options for turn autonomous life on/off, rest the robot, wake up the robot and battery information about the robot.

Figure 7: Image of the top bar fiveth rectangle.

Also, we are going to continue giving information about other parts of the main menu interface, at the left side we can see this menu blocks 8.

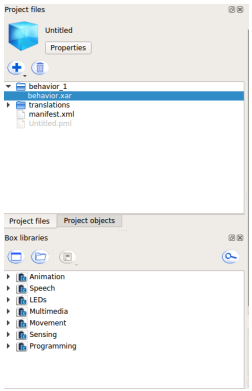

Figure 8: Image of the left bar.

In this part the reader can see two different zones: the first one, at the top it provides information about project files, in the plus mark, you can create new behaviors, dialogs, planar moves, directories, or import files from other selected directories.

The last one provides the user box libraries which will help the user to program the robot behaviors. It have 7 categories called:

- Animation.
- Speech.
- LEDs.
- Multimedia.
- Movement.
- Sensing.
- Programming.

We will explain better in future sections giving relevant information about the utility of these categories and the different types of boxes that they provide.

On the other side, the right bar 9, it provides a video image of the robot where the user can see what is doing the robot doing, objects that are in front of him, and the currently distance of them. It is a place where you can test the robot behavior and the user can validate and test the designed behaviors.

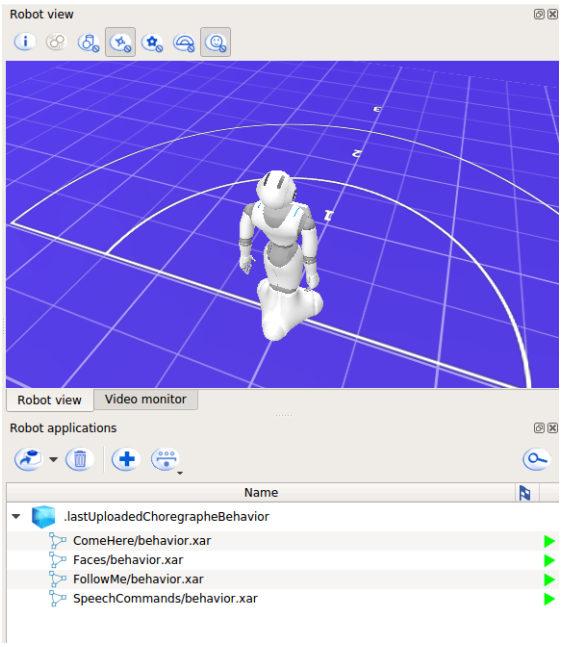

Figure 9: Image of the right bar with robot simulation.

At the bottom, the user can see packages that are installed into the robot and options for install new ones or delete one of them.

Finally,in the middle 10, we have the space for the block designing.

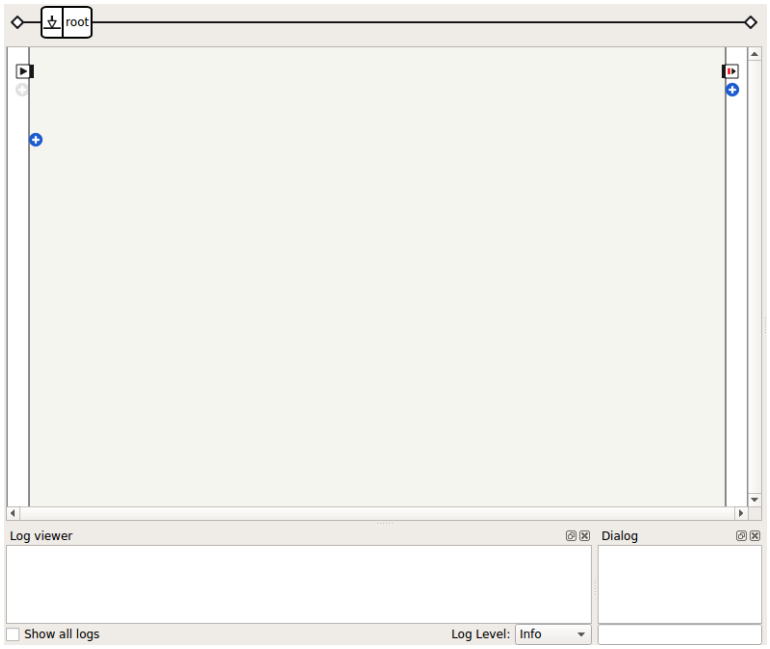

Figure 10: Image of the block diagram design section.

At the top the reader can see timelines blocks, it contains block that will show other block designing space. In the middle the user can drop blocks for making his behaviours, he have to connect the first one to the left arrow and the last one in the other side, at the right arrow.

At the bottom the reader can see the log viewer for see error details, he can use the dialog text for writting messages to the robot, it provides to the user that he do not need to talk if he want to test for example a speech recognition block.

3.2. HOW TO CONNECT TO PEPPER

In order to start working Choregraphe we need to connect with Pepper. First we need to be connected to the same network as the robot, in our case the network is called "legoland". Once we are connected to the network, we can connect Choregraphe with Pepper. That can be done either by pressing Connection -> Connect to..., by pressing the green button in 5 depicting an antenna with waves or by pressing Ctrl + Shift + C.

Now we need to know the IP address that Pepper is using. We can know that by touching the button behind the tablet the robot is holding. We just need to write the IP address and the port the robot says in the Choregraphe screen shown in 11.

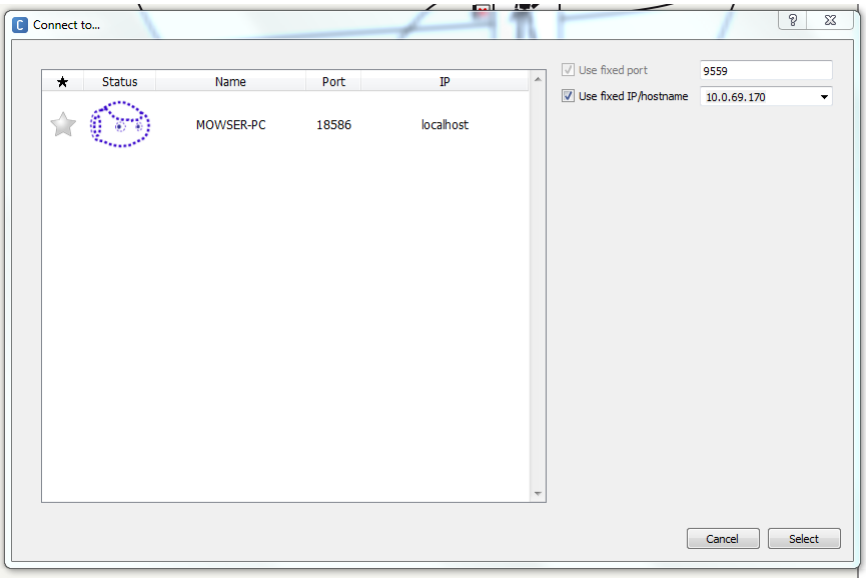

Figure 11: Connect to... screen in Choregraphe.

After that we only need to press the "Select" button and we will be connected to the robot.

To disconnect we can use either Connection -> Disconnect, the red button in 5 or Ctrl + Shift + D.

4. SPEECH RECOGNITION

One of the most interesting things in Choregraphe is the "Speech Recognition" block in 12.

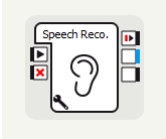

Figure 12: Speech Recognition block.

They allow us to interact with the robot with our voice.

Inputs:

- onStart: type “Bang”, which means it will not treat the information received, it is just a signal that starts the box behaviour. It is the one marked with the play symbol.
- onStop: type “Bang”. Stops the box behaviour. Marked with a red cross.

Outputs:

- onStopped: type “Bang”. Sends a signal after the box behaviour has stopped. Marked with a vertical red line with a black play symbol on its right side.
- wordRecognized: type “String”. Sends the word the robot has recognized.
- onNothing: type “Bang”. Sends a signal if Pepper has not heard anything during a certain timelapse.

The most important parameters are:

- Word list: list of strings separated by semicolons the robot will try to recognize when that box behaviour is on.
- Confidence threshold: when the robot hears something, it assigns a confidence value to the word it has recognized. If that value does not reach the value set in this parameter, the word will not be recognized.

4.1. ANIMATIONS

There are a few animations already defined, as you can see in 13.

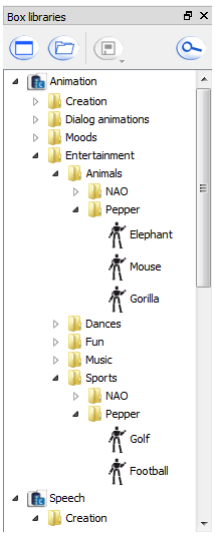

Figure 13: Some predefined animation blocks.

In our example, we decided to use “Gorilla” and “Football” animations.

Inputs:

- onStart: type “Bang”. Starts the box behaviour.
- onStop: type “Bang”. Stops the box behaviour.

Outputs:

- onStopped: type “Bang”. Sends a signal after the box behaviour has stopped.

We have created a behaviour called “SpeechCommands” that has to perform an animation when it is said so, which diagram is the one being shown in 14 .

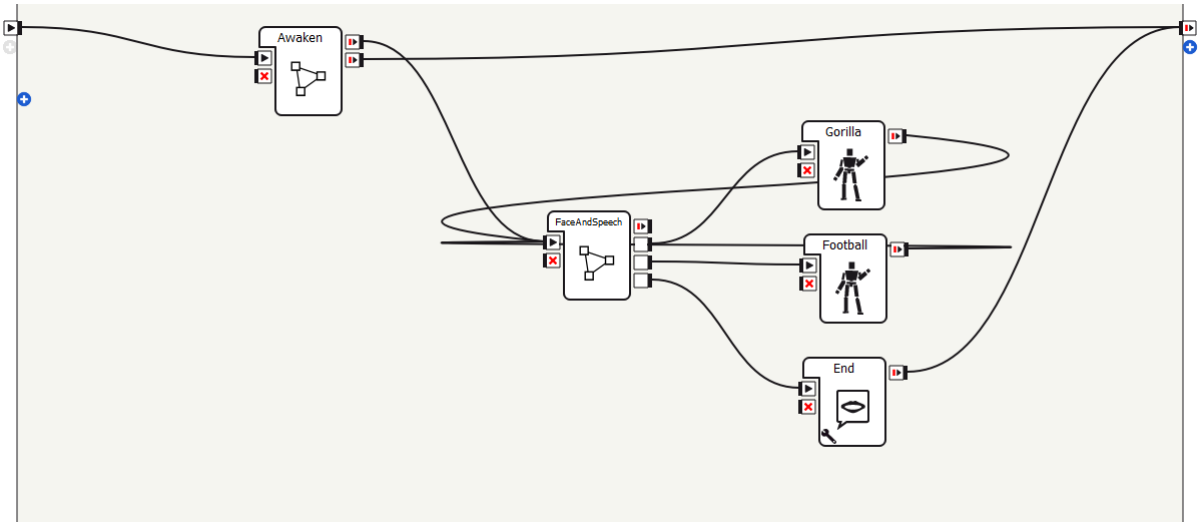

Figure 14: SpeechCommands behaviour diagram.

Firts we need to “Wake Up” the robot, otherwise it will not be able to perform anything.

We created a block composed by more blocks in order to do that, called “Awaken” block.

Inputs:

- onStart: type “Bang”.

Outputs:

- onSuccess: type “Bang”. Sends a signal if the robot woke up correctly.
- onFailure: type “Bang”. Sends a signal if the robot failed to wake up.

As we can see in **??**, the block is composed by a “Stand up” box, some “Set Language” boxes and a “Say” box. The “Stand up” box wakes up the robot, starts the motors and puts Pepper to its default position.

If the robot woke up correctly then it will say a message. Else it will send a signal through its onFailure output.

After that, we have the “FaceAndSpeech” block, which diagram is shown in 15 . Inputs:

- onStart: type “Bang”.
- onStop: type “Bang”.

Outputs:

- onStopped: type “Bang”.
- An type “Bang” output for each function the robot has to be able to perform. That is to say, if we want it to say “Goodbye” after the user says “Bye”, “See you” or any other farewell, there will only be an output for all the farewells the robot can hear.

- End: type “Bang”. When the robot is told to stop, this output will send a signal.

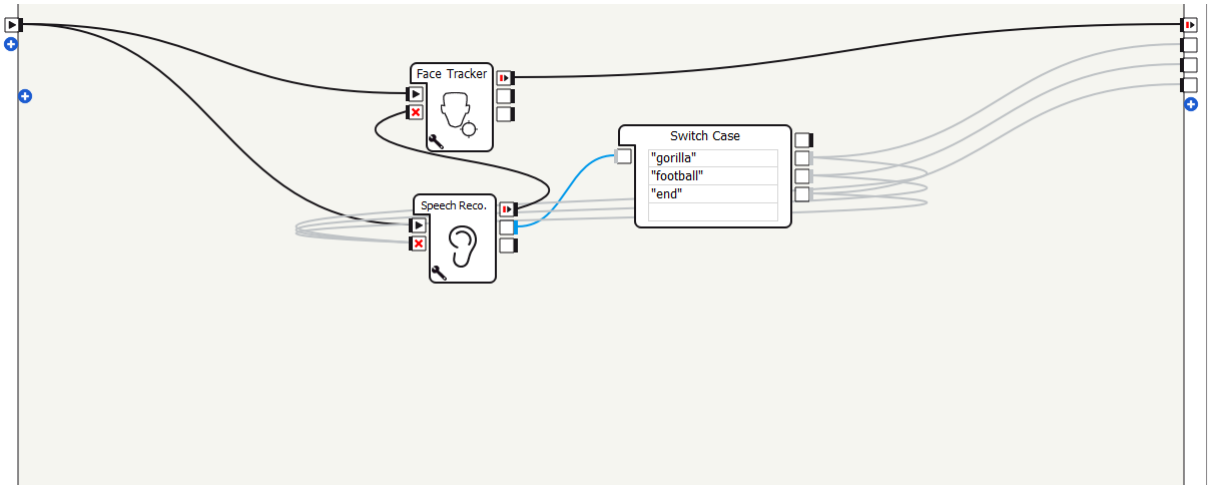

Figure 15: FaceAndSpeech box diagram.

This block has a Face Tracking box, a Speech Recognition box and a Switch box. The Speech Recognition box sends any word it recognizes to the Switch box that sends a signal to the correct output and stops both the tracking and the speech recognition.

Since we wanted to make the robot more realistic, we added a Face Tracking box to make it follow the face of the person talking to it.

The Speech Recognition box has in its wordlist the words “Football” and “Gorilla”. The Switch box has only three cases, one for each word in the wordlist and another one for the “end” word. If someone writes “end” in Pepper’s console, this case will be triggered.

After the FaceAndSpeech box we have the animations and the message the robot says when it is told to stop.

4.2. MOVEMENT

We created the “FollowCome” behaviour, that implements both the “Follow me” and “Come here” voice commands. The diagram, that is very similar to that in 14 , can be seen in 16 .

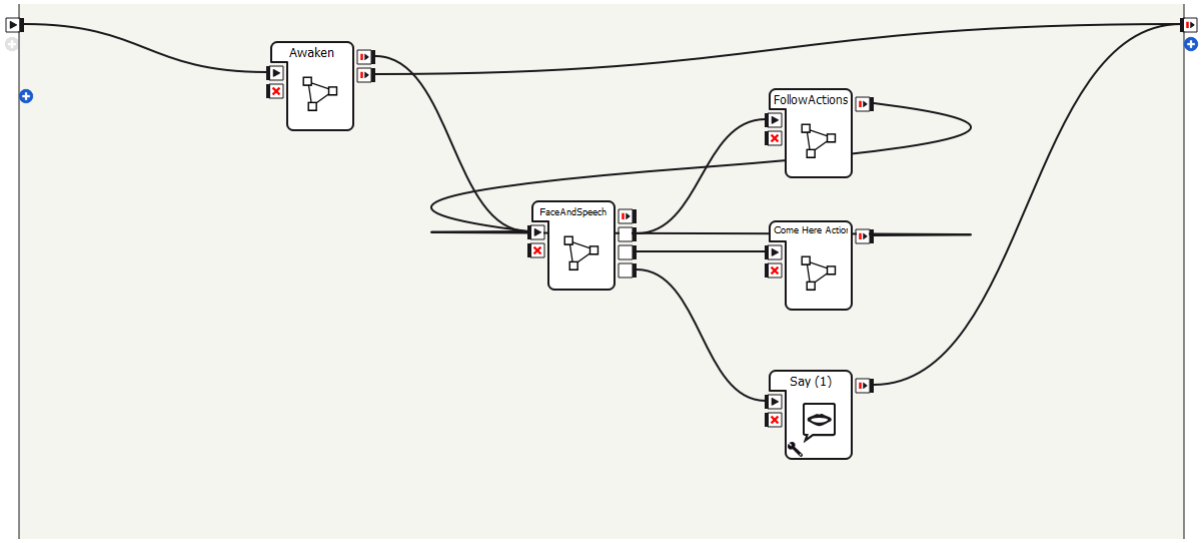

Figure 16: FollowCome behaviour diagram.

The only changes are inside FaceAndSpeech block and the animations have been replaced by new boxes.

FaceAndSpeech box had its wordlist and cases changed to use the new voice commands: “Follow me” and “Come here”.
Instead of animations, we now have two new blocks: “FollowActions” and “Come Here Actions”.

They both have the same inputs and outputs, none other than onStart, onStop and onStopped.

The FollowActions box, which diagram is shown in 17, starts with the robot saying what the user has to do in order to stop the robot.

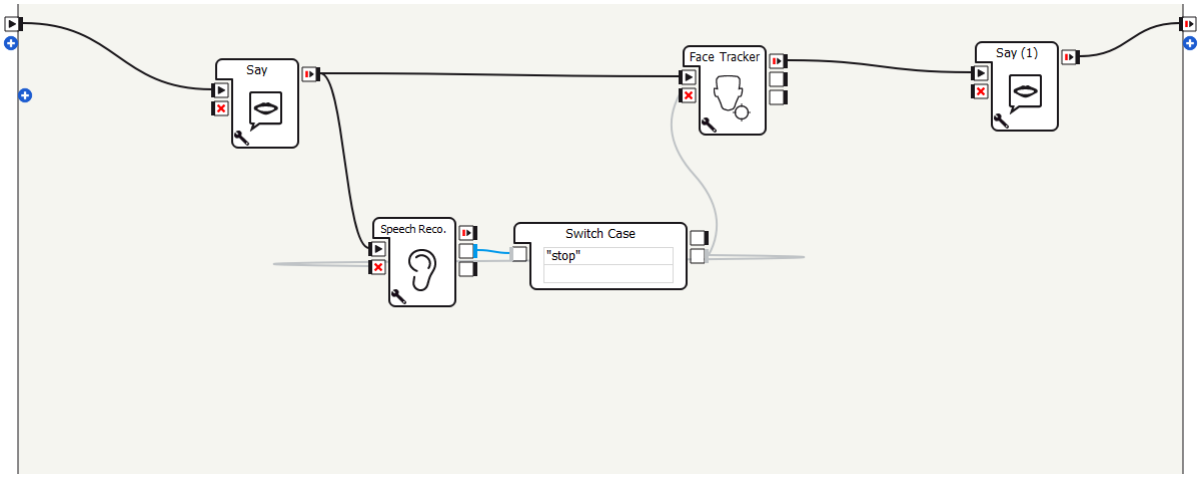

Figure 17: FollowActions box diagram.

Then, it starts simmultaneously a Face Tracker box and a Speech Recognition box with two words in its wordlist: “stop” and “blergh”. Only having one word in its wordlist will make the robot try to hear only that word, making a lot of false positives.

After that box, a Switch box will stop everything when the robot hears “stop”, and will do nothing otherwise.

The Face Tracker box will make the robot move to the position of the face it is tracking.

When the robot is stopped, it says a phrase to let the user have some feedback.

The "Come Here Actions" box is more simple, as can be seen in 18.

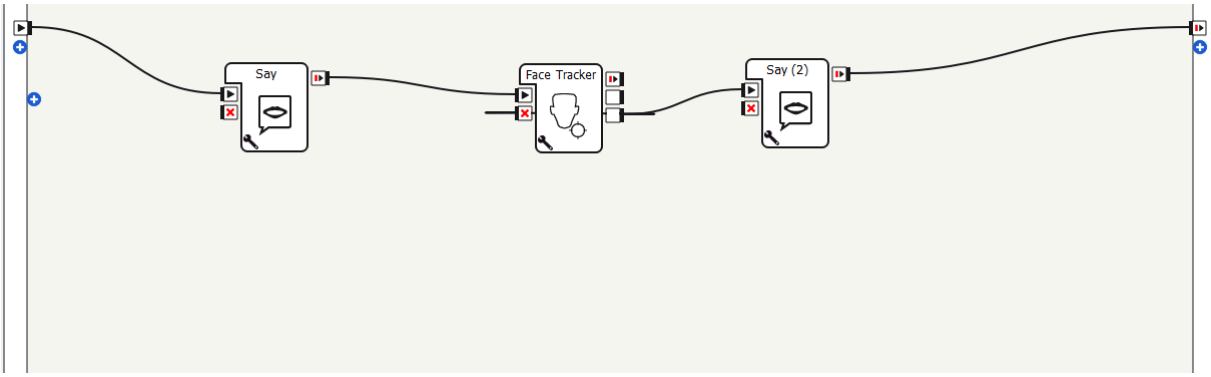

Figure 18: Come Here Actions box diagram.

It starts and end with say blocks, to let the user have some feedback.

In the middle of those blocks, there is a face tracking box that moves the robot to the position of the user and stops when it reaches its target.

One important thing to take in account when using Choregraphe is that all the tracking and speech recognition blocks of the same type in the same diagram will be activated simmultaneously.

That is to say: if you put two different Speech Recognition boxes in the same diagram, when you start one of them the other will start too, and if you stop one of them the other will also be stopped.

The same happens to Face Tracking blocks. That's why we created complex boxes with smaller behaviours that only need one tracking box of each type as much. It helps us organize our work as well as fixing that problem.

5. FACE RECOGNITION

Pepper has a Face Learning module, that lets her to learn a face and to recognize it later. Subsequent learnings of the same face allows Pepper to speed up the recognizing proccess.

We have created the behaviour "Faces", shown in 19, that lets Pepper greet someone unknown to her and ask his/her name, in order to store the face. When Pepper meets someone whose face is already stored, she will greet that person by his/her name.

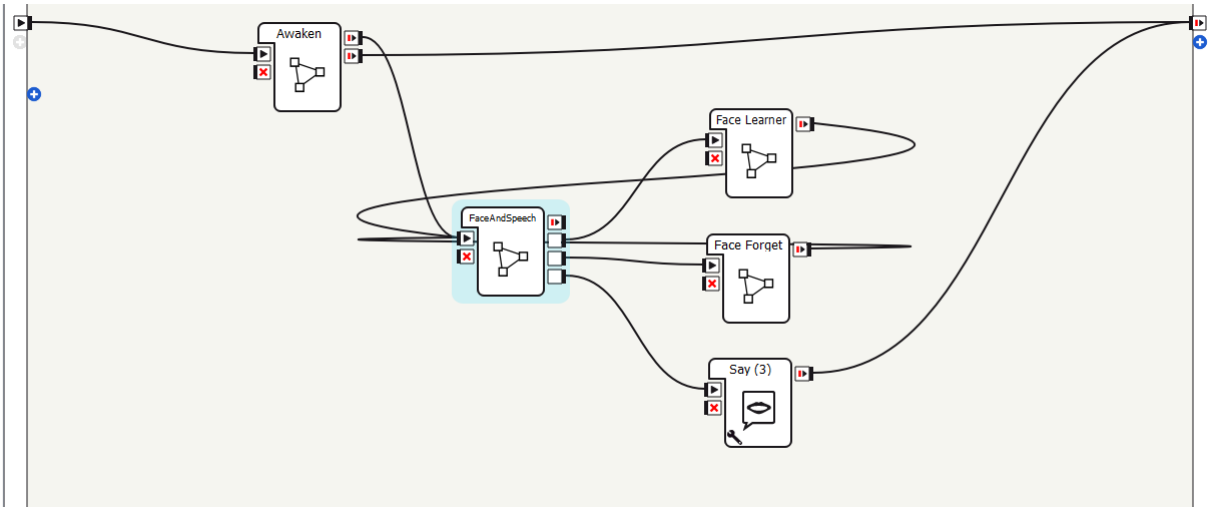

Figure 19: Faces behaviour diagram.

Again, the diagram is very similar to those of the previous behaviours.

We had to change the wordlist and cases in FaceAndSpeech box, as well as the blocks that execute after it outputs something.

We have two new blocks, both with onStart and onStop inputs and the onStopped output that restarts the FaceAndSpeech box like in the previous behaviours.

When the user greets the robot, the “Face Learner” box is executed, that has the diagram shown in 20 .

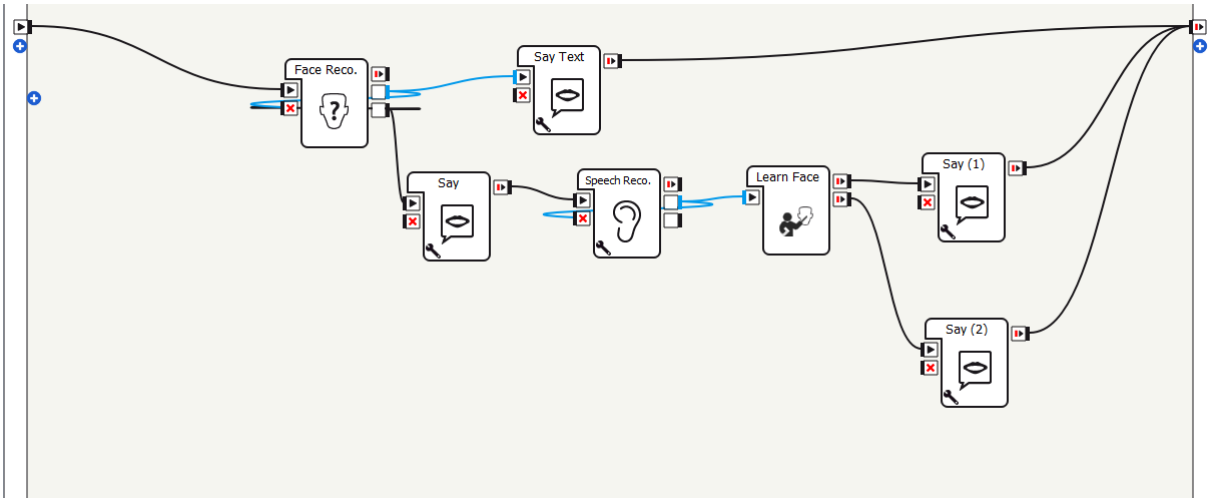

Figure 20: Face Learner box diagram.

It starts trying to recognize the user's face with the "Face Recognition" box. If it succeded in its task, the robot will greet the user and will say the user's name.

Otherwise, the robot will ask for the user's name and will start a Speech Recognition box. The wordlist of this box has a list of possible user names. After hearing a name, the robot will learn the user's face with the heard name.

If it succeded in its task, the robot will say "Nice to meet you". Otherwise, the robot will answer with a quote of a poem written by William Blake.

The other box, which diagram is the one appearing in 21, is called "Face Forget" and lets us make the robot forget a certain face.

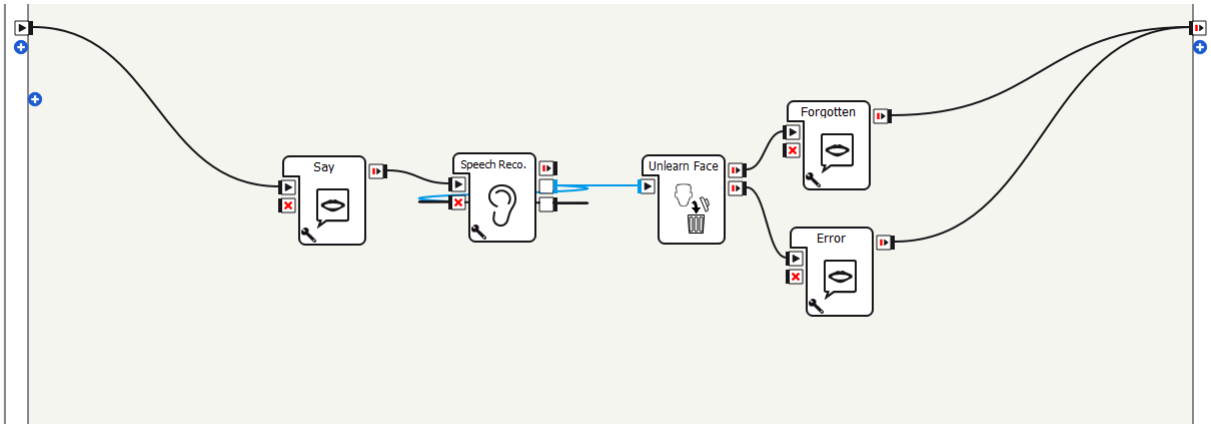

Figure 21: Face Forget box diagram.

The robot will ask for the name of the face the user wants it to forget. Again, a Speech Recognition box with a list of names as its wordlist will be activated. The robot will forget the face whose name the user has said using the "Unlearn Face" block.

Finally, the robot will tell the user whether it was able to forget the face or not.

6. DIALOGUE

The last behaviour is a dialogue in which the robot interacts with a user talking and with animations. Its diagram is the one that appears in 22 .

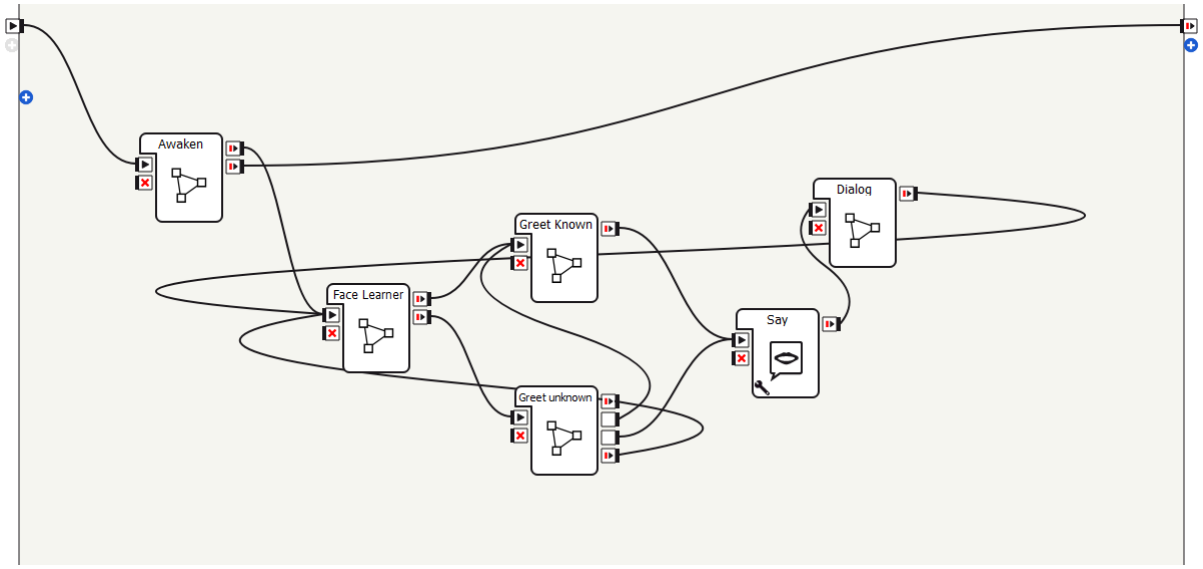

Figure 22: Dialogue behaviour diagram.

The Awaken box is different from its counterpart in the other behaviours, as can be seen in 23 .

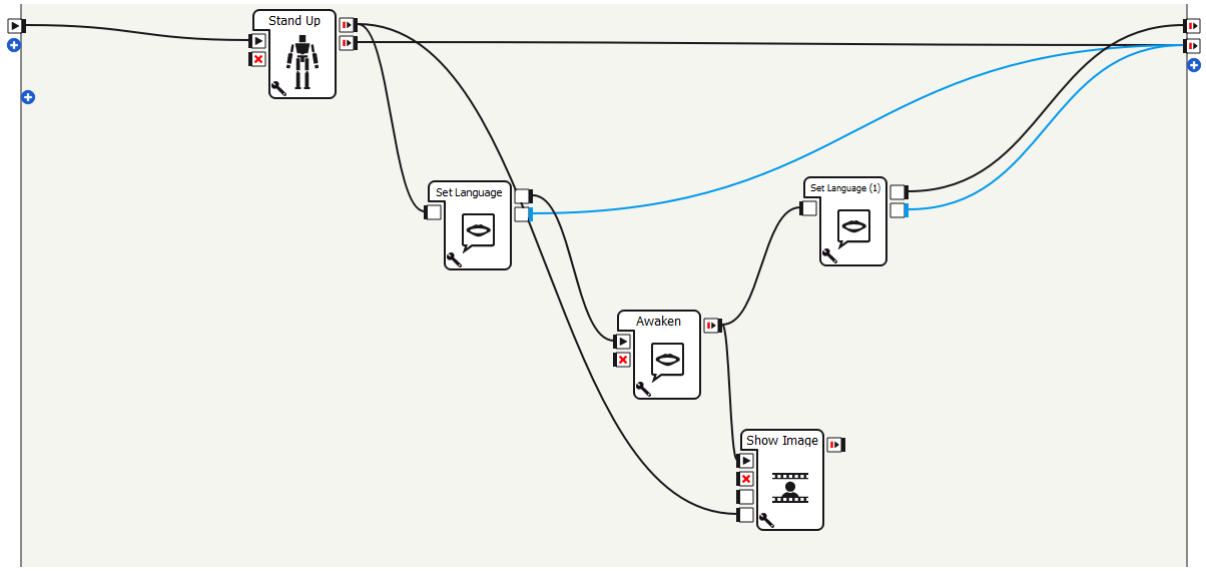

Figure 23: Awaken box diagram in Dialogue behaviour.

An image depicting the Sun will appear in the screen after the robot wakes up and says its wake up message.

The image is the one shown in 24 .

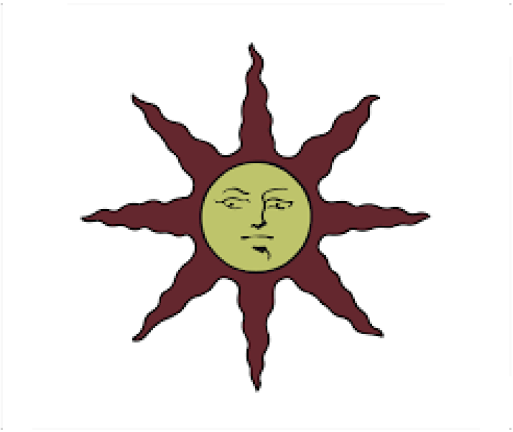

Figure 24: Sun image shown by the robot.

In order to reduce the amount of time it takes to load the image, its preloaded after the robot wakes up and displayed after the message is said.

After the Awaken box, a new box is executed: the "Face Learner" box. It's very similar to the one in 20, but in this case we take advantage of the fact that you can relearn a face multiple times to speed up the recognition phase.

As can be seen in 25, the main difference between this FaceLearner box and the other is that each time Pepper recognizes a face, it will relearn it.

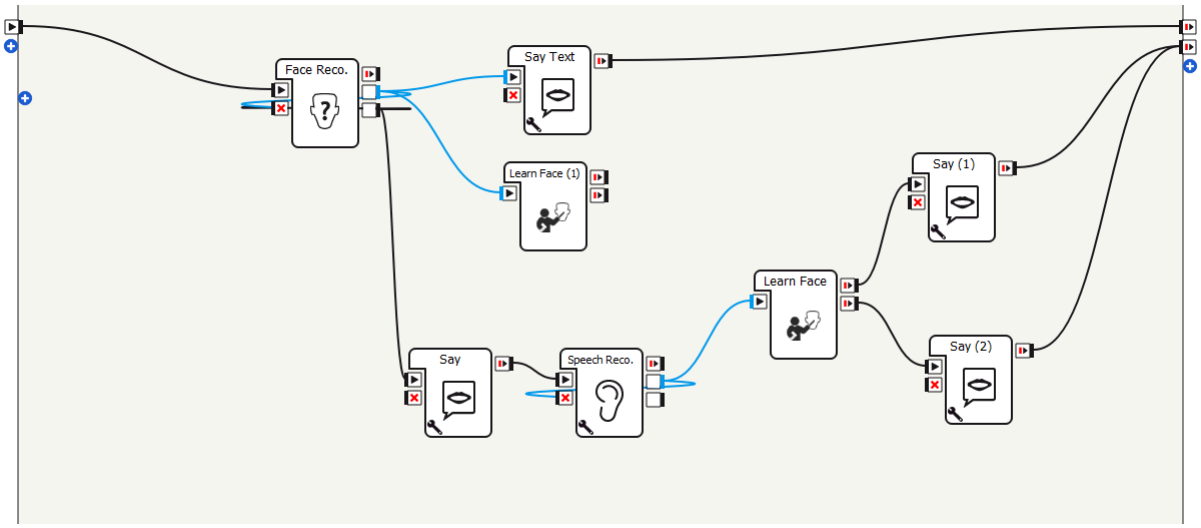

Figure 25: FaceLearner box diagram in Dialogue behaviour.

Using the face learning we let the robot to start a conversation by greeting someone known to Pepper or by asking someone's name.

The other difference between this box and the previous one is that this one has a different output if Pepper detected a new face or not.

If the face is new, the “Greet Known” box will be executed. Otherwise the “Greet unknown” box will be executed.

The diagram of Greet unknown box is the one appearing in26.

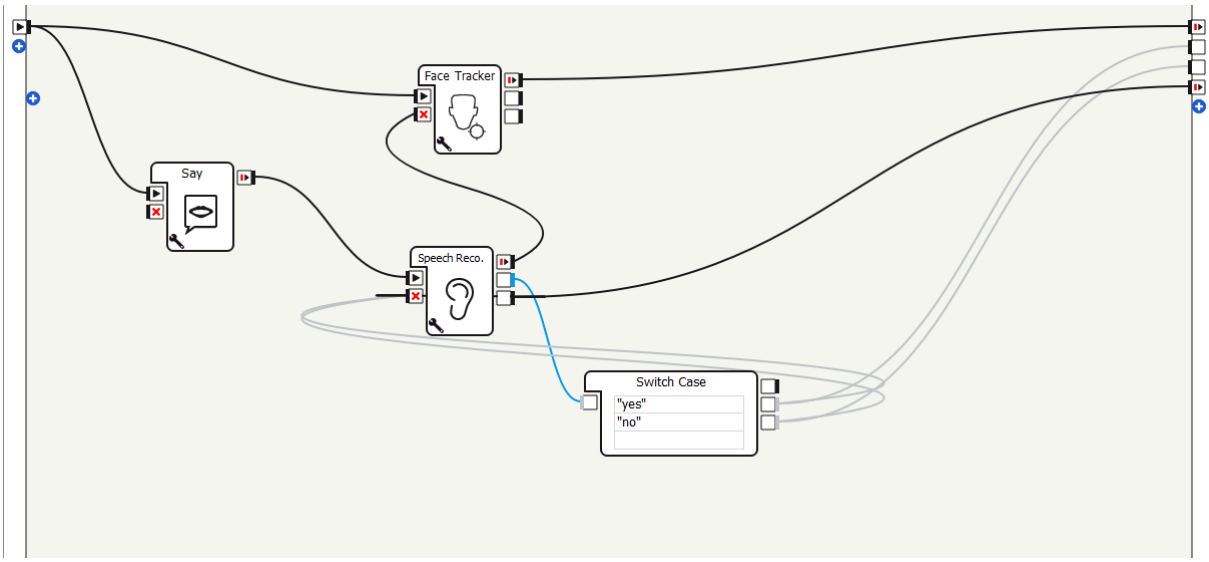

Figure 26: Greet unknown box diagram.

First the robot will ask the user whether he/she has ever thought about praising the Sun.

If the answer is “No”, the robot will ask the player to talk about the Sun and the Dialog box will start.

If the answer is “Yes”, the Greet Known box will be executed. The diagram of Greet Known box is the one appearing in 27.

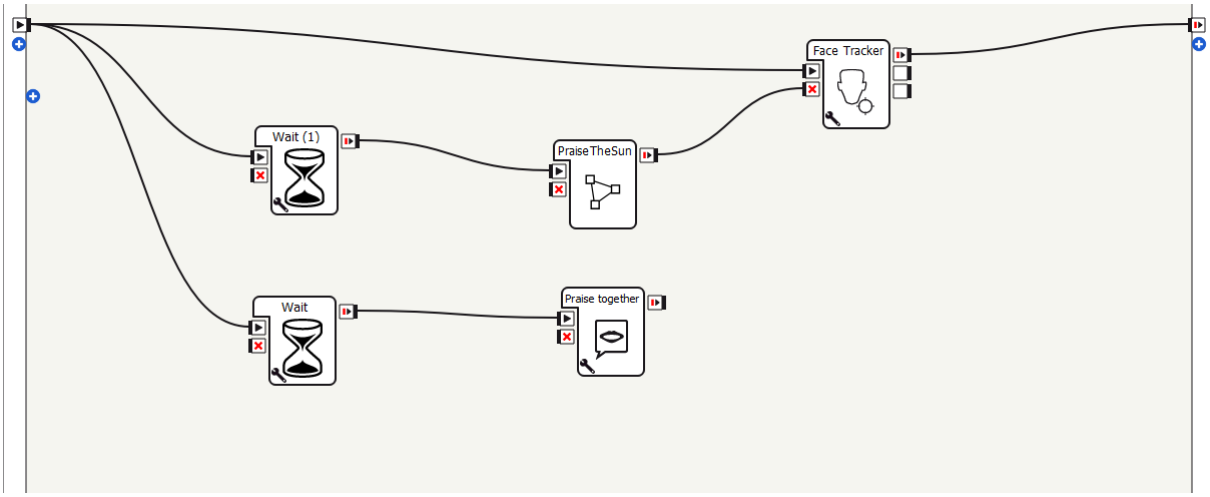

Figure 27: Greet Known box diagram.

The robot will follow the user with its head with a Face Tracker box and will ask the player to praise the Sun together.

After that the “PraiseTheSun” box will be executed.

That box follows the diagram shown in 28.

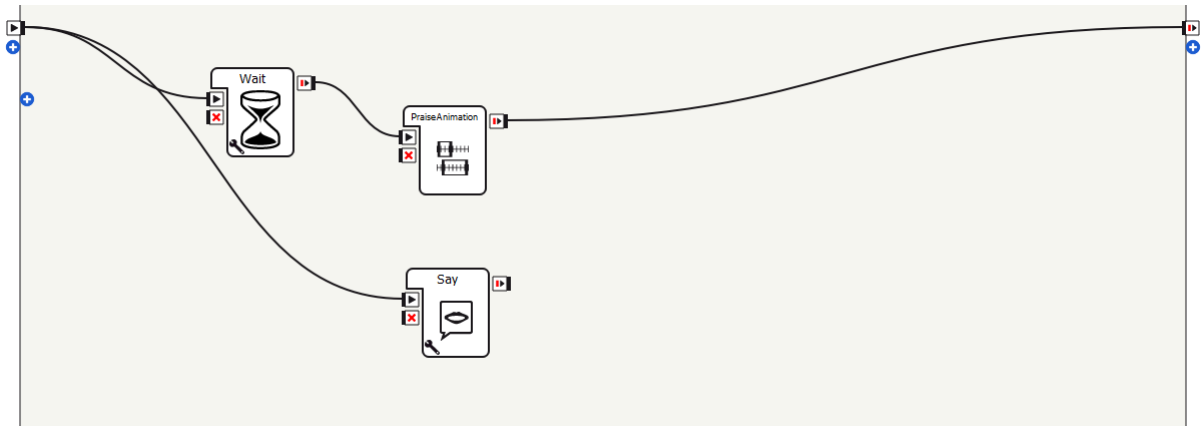

Figure 28: PraiseTheSun box diagram.

First the robot will say “Praise the Sun”, and then an animation of the robot praising the sun will start, leaving the robot in the same position as in the figure 29 .

Figure 29: Praise the Sun animation.

When the PraiseTheSun box ends, the Greet Known will end too.

After that, the same boxes that will be executed if you answer “No” in the Greet unknown box will be executed, leading also to the Dialog box.

The Dialog box has a complex behaviour as can be seen in the figure 30.

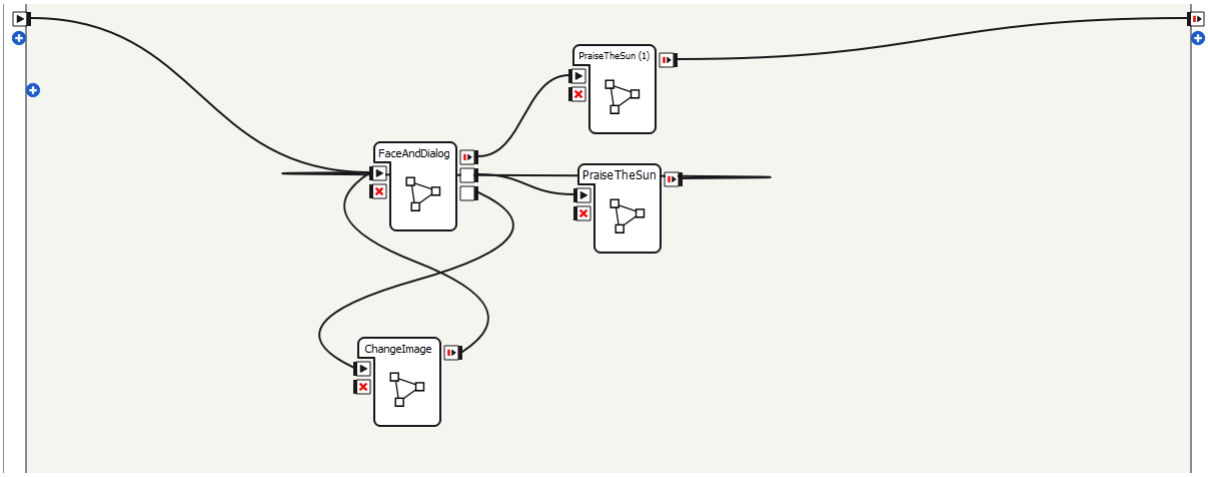

Figure 30: Dialog box diagram.

First the "FaceAndDialog" box is executed. Its diagram is 31 .

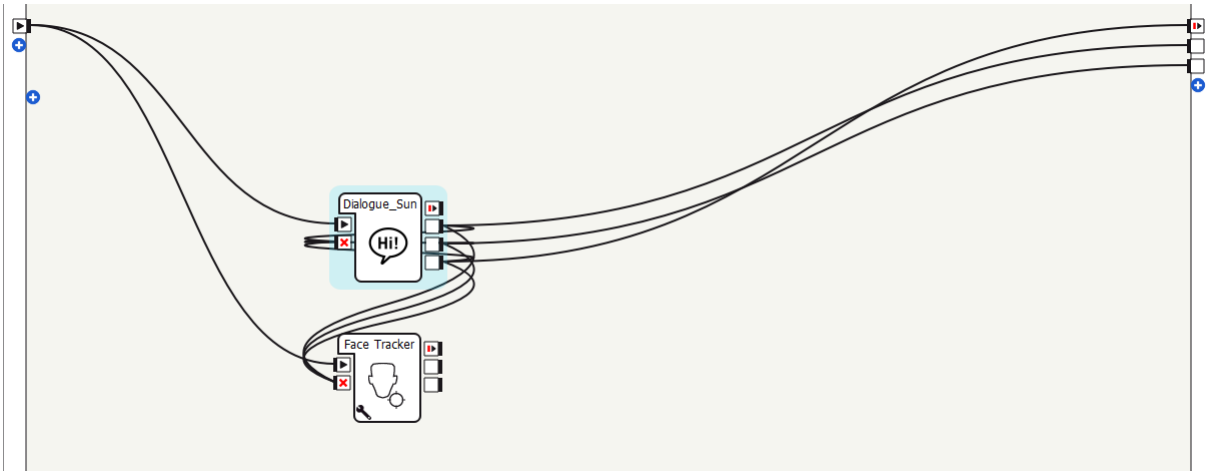

Figure 31: FaceAndDialog box diagram.

It executes the next dialog:

../src/Dialogue_Sun/Dialogue_Sun_enu.top

In this dialog, the robot can talk about what is the Sun, how to praise it, why to praise it, where to praise it, who is Gwyn, Gwynevere and Gwyndolin, and finally you can ask Pepper what can she do/help and you can say goodbye to her.

While talking, there is a Face Tracker block that follows the user's head, to add more realism.

There are three special dialog options.

The farewell option will send a signal through the onStopped output of the FaceAndDialog option. It will start a PraiseTheSun box and the Dialog box will be stopped.

The "how to praise" option will send a signal through the "Praise" output, and will start a PraiseTheSun box. After its behaviour is completed, the FaceAndDialog will be restarted.

The “where to praise” option will send a signal through the “Where” output, starting the “ChangeImage” box. After its behaviour is completed, the FaceAndDialog will be restarted.

The ChangeImage box, with the diagram that appears in 32, changes the image displayed in the tablet to the map of the ESIIAB shown in 33 while saying the best place to praise the Sun inside of the ESIIAB.

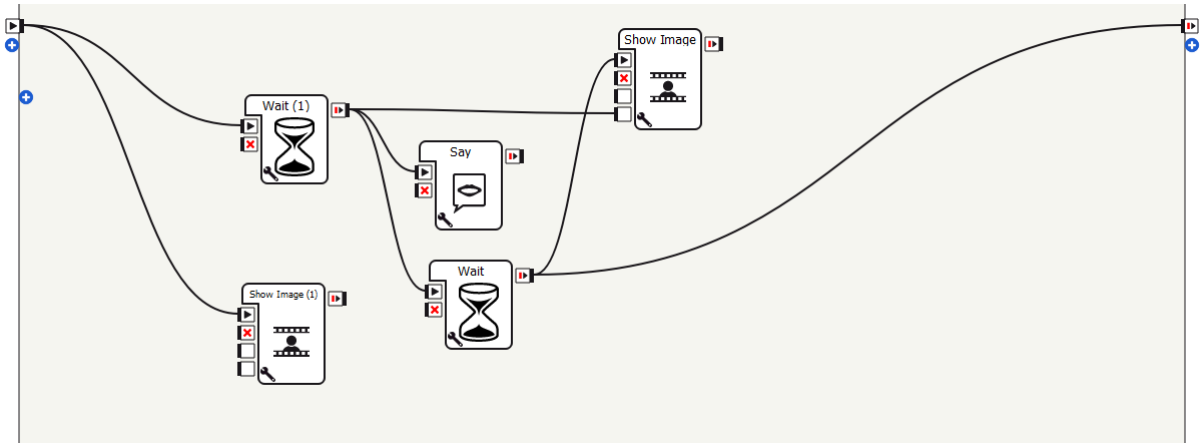

Figure 32: ChangeImage box diagram.

After the farewell option was used and the robot has said goodbye, the Dialog box ends and the FaceLearner box will be run again.

7. CONCLUSIONS

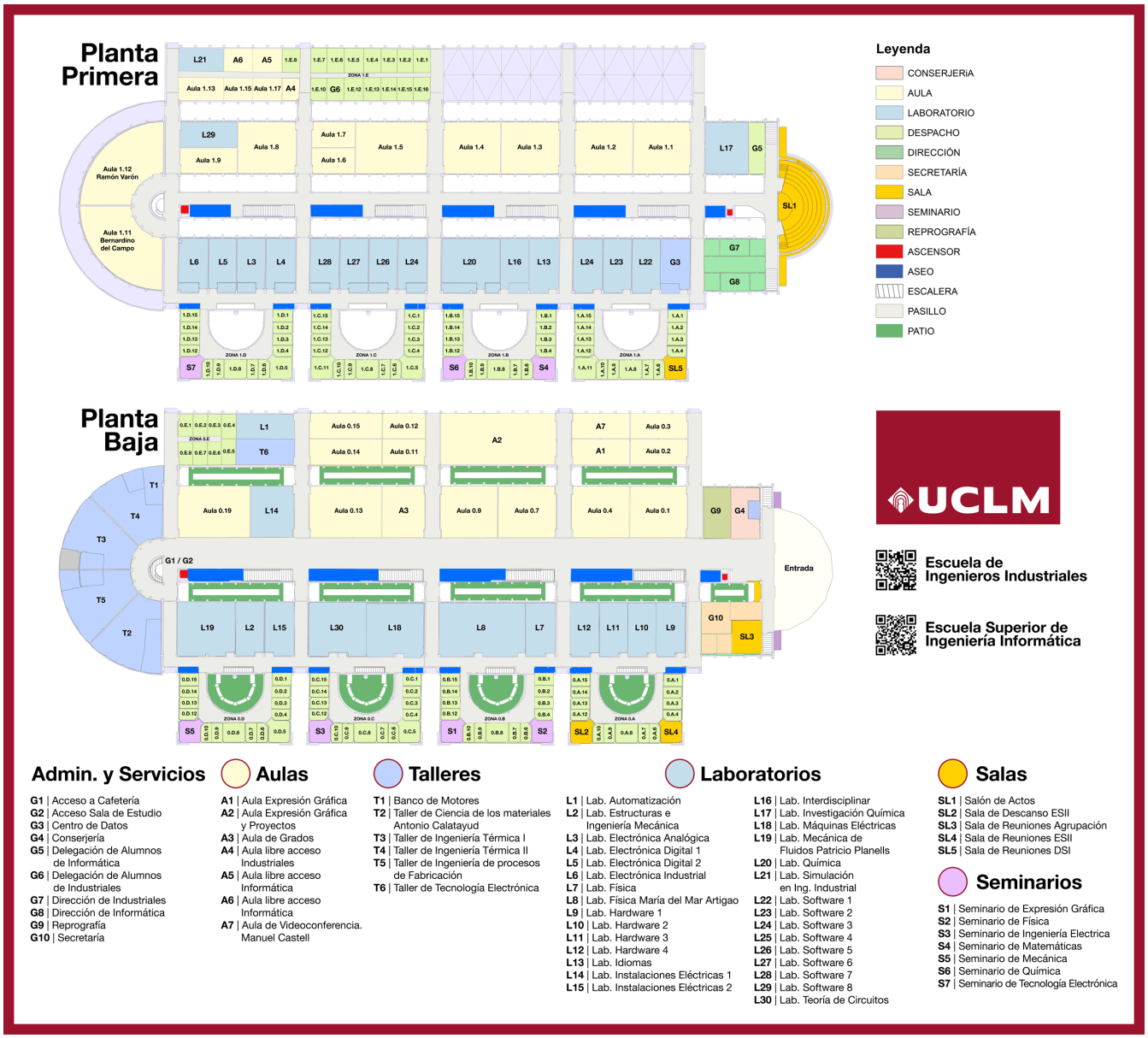

Figure 33: ESIIAB map shown by the robot.

7. REFERENCES

[1] Apache Cordova, cordova.apache.org. Consultado el 10 de Abril del 2018. https://cordova.apache.org/docs/en/latest/

[2] ArcGIS for Developers, developers.arcgis.com. Consultado el 14 de Abril del 2018. https://developers.arcgis.com/

[3] Bouskela, Mauricio., et al (2016),*La ruta hacia las Smart Cities: Migrando de una gestión tradicional a la ciudad inteligente*. Banco Interamericano de Desarrollo.

[4] Dirección General de Tráfico, www.dgt.es. Consultado el 13 de Abril de 2018. http://revista.dgt.es/es/categorias/distracciones-moviles.shtml

[5] IBM Cloudant, www.ibm.com. Consultado el 14 de Abril del 2018. https://www.ibm.com/cloud/cloudant

[6] IBM Bluemix, www.ibm.com. Consultado el 14 de Abril del 2018. https://www.ibm.com/cloud-computing/bluemix/es

[7] Ionic, www.ionicframework.com. Consultado el 11 de Abril del 2018. https://ionicframework.com/docs/

[8] Marco de Desarrollo de la Junta de Andalucia, *Conceptos sobre Escalabilidad*, Consultado el 10 de Abril de 2018. http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/220

[9] Node-RED, www.nodered.org. Consultado el 16 de Abril del 2018. https://nodered.org/docs/platforms/bluemix

[10] Pandorafms, *Bases de Datos NoSQL*, pandorafms.org. Consultado el 12 de Abril de 2018. https://blog.pandorafms.org/es/bases-de-datos-nosql/

[11] Universidad de Castilla-la Mancha. www.uclm.es. Consultado el 9 de Abril de 2018. https://eventos.uclm.es/19833/detail/hack-for-albacity.html

[12] URBIOTICA (2008), www.urbiotica.com. Consultado el 18 de Abril del 2018. http://www.urbiotica.com/producto/u-spot/

[13] W3Schools (1998), www.w3schools.com. Consultado el 15 de Abril del 2018. https://www.w3schools.com/nodejs/default.asp

ANNEXES

ANNEX I ENLACES DE CONTENIDO

- **Enlace a la WebApp:**
  `https://git.eu-gb.bluemix.net/HernanIndibil.LaCruz/Hackaton-Skyhold-Web`
- **(Aplicación de mapas en ejecución):**
  `https://hackaton-skyhold-web.eu-gb.mybluemix.net/`
- **Documentación (presentación y memoria):**
  `https://git.eu-gb.bluemix.net/HernanIndibil.LaCruz/Hackaton-Skyhold-Web/tree/master/DOCS`
- **Node-RED:**
  `https://git.eu-gb.bluemix.net/HernanIndibil.LaCruz/Hackaton-Skyhold-RED`
- **(Editor de Node-RED):**
  `https://hackaton-skyhold-red.eu-gb.mybluemix.net/red/`
- **Repositorio de la Aplicación:**
  `https://github.com/Mowstyl/Hackaton-Skyhold-App`

ACLARACIÓN: *Los diferentes servicios serán públicos o se proveerán los accesos necesarios a los mismos.*