UNIVERSITÉ LIBRE DE BRUXELLES

Bruface
Brussels Faculty of Engineering

VRIJE UNIVERSITEIT BRUSSEL

Development of an user's preference detection system for multiple social robots in HRI: Technical document for developer and user

Xuan Nhan Nghiem

Master thesis submitted under the supervision of
Prof. dr. ir. Bram Vanderborght
Prof. dr. ir. Dirk Lefeber

The co-supervision of
Ir. Hoang-Long Cao
Ir. Albert De Beir

Faculty of Engineering Sciences
Electromechanical Engineering
Mechatronics-Construction

Academic year
2016-2017

Contents

Chapter 1

Structure of the program (developer)

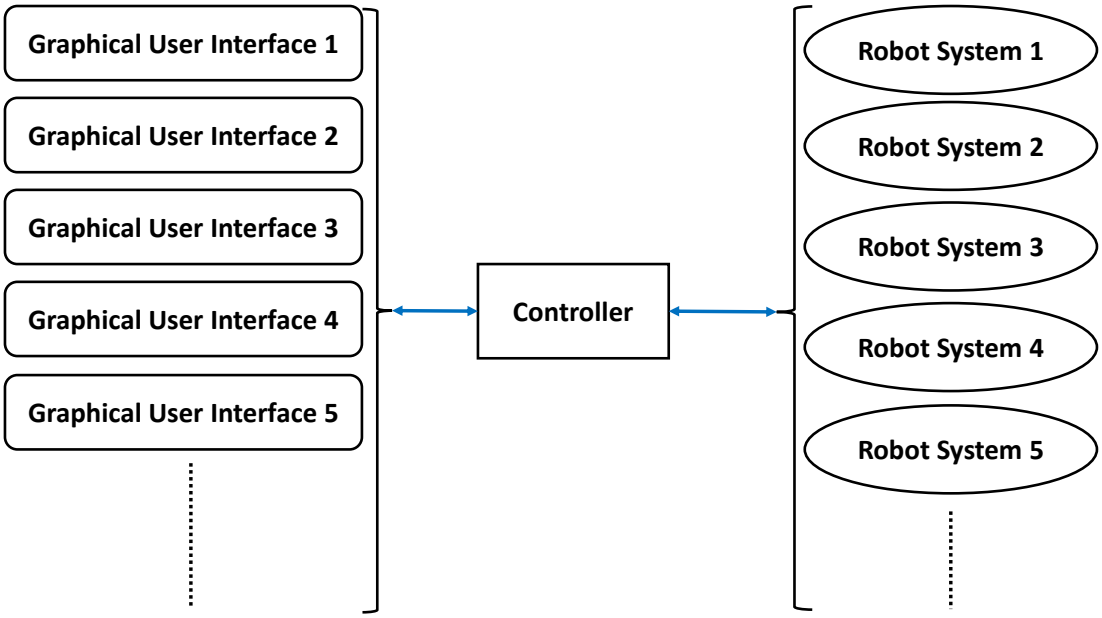


Figure 1.1: A total independence between the GUI and the robot platform will allow to modify, add or remove a part from one side much easier.

However, for now, the program is separated in two different python projects. Indeed, there are two separated codes, one for each GUI (see figure 1.2). The fact is that a large part of the code is identical for both GUI as only the Python files *data.py*, *view.py* and *logic.py* contain a few different functions between the two GUI. A fusion of the 2 projects could be an interesting improvement to really reach the structure of figure 1.1.

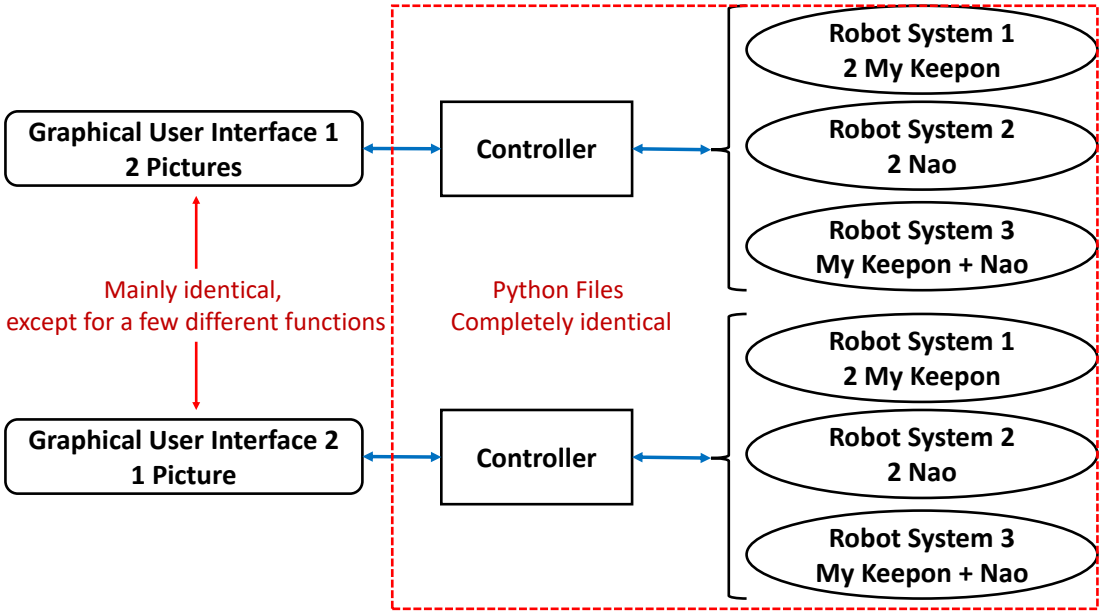


Figure 1.2: Current global structure of the detection system.

The program is separated in five principal parts:

1. **Data**: all the data from the Excel file are read and stored by the python file *data.py*
2. **GUI**: the graphical interface with which the participant will directly interact, managed by two python files, *view.py* and *logic.py* (see section 1.2 for more details).
3. **Controller**: the link between the GUI and the robot commands, provided by the python file *controller.py* (see section 1.3 for more details).
4. **Robot**: the instructions to the robots, one python file per robot which mean for now *keepon.py* and *nao.py* (see section 1.4 for more details).
5. **Post-questionnaire**: a second graphical interface on which the participant can complete a questionnaire on the experiment, generated by the python file *postQuestionnaire.py* (see section 1.5 for more details).

To know which module you need to install to use the program, please see section 2.1 for more information.

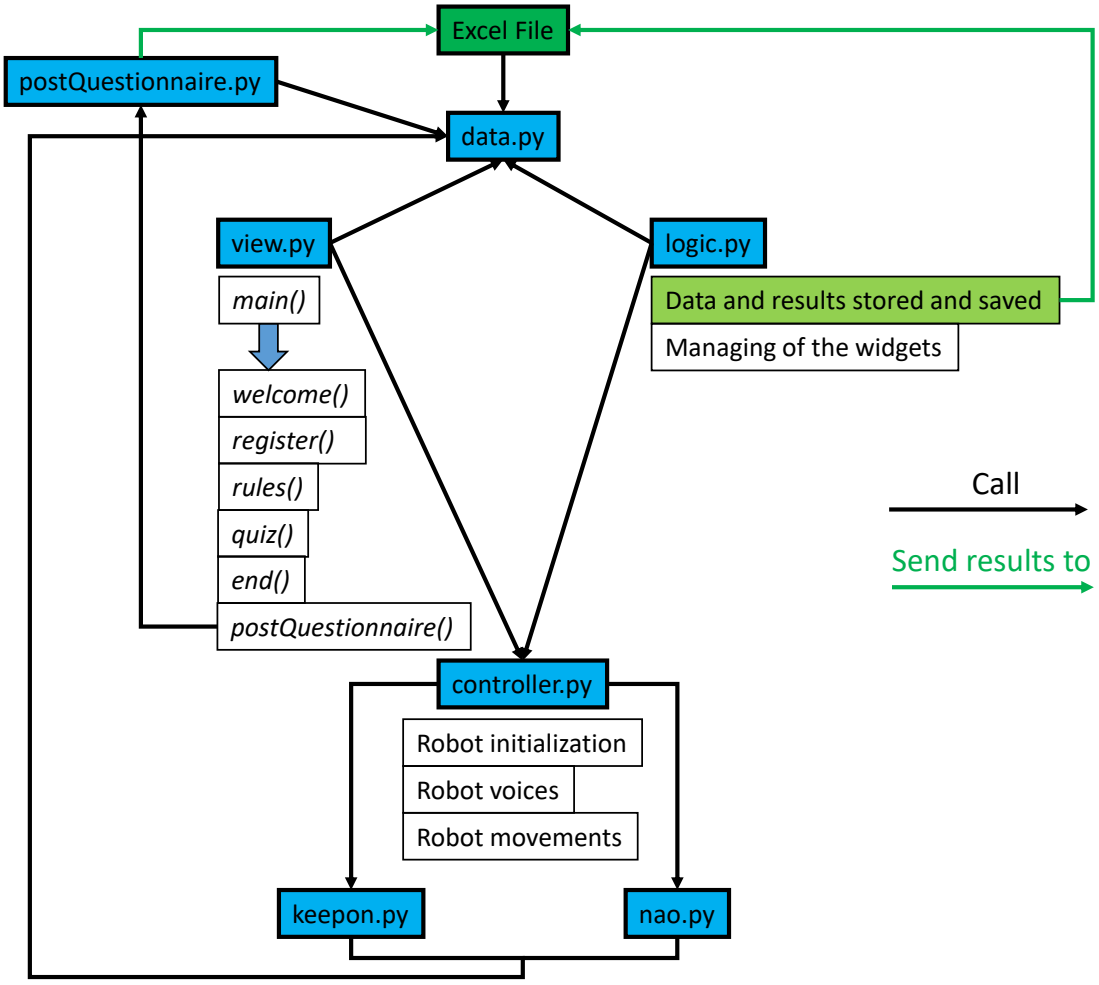


Figure 1.3: General flowchart showing the link between the python files (in blue). The most important functions/functionalities are also written for some files to highlight their role in the program. The arrows and cases in green concern the data and results of the experiment to save. This flowchart can be associated for both graphical interface. There are some differences between them, especially in the "Data" and "GUI" parts but none about the general structure.

1.1 Data

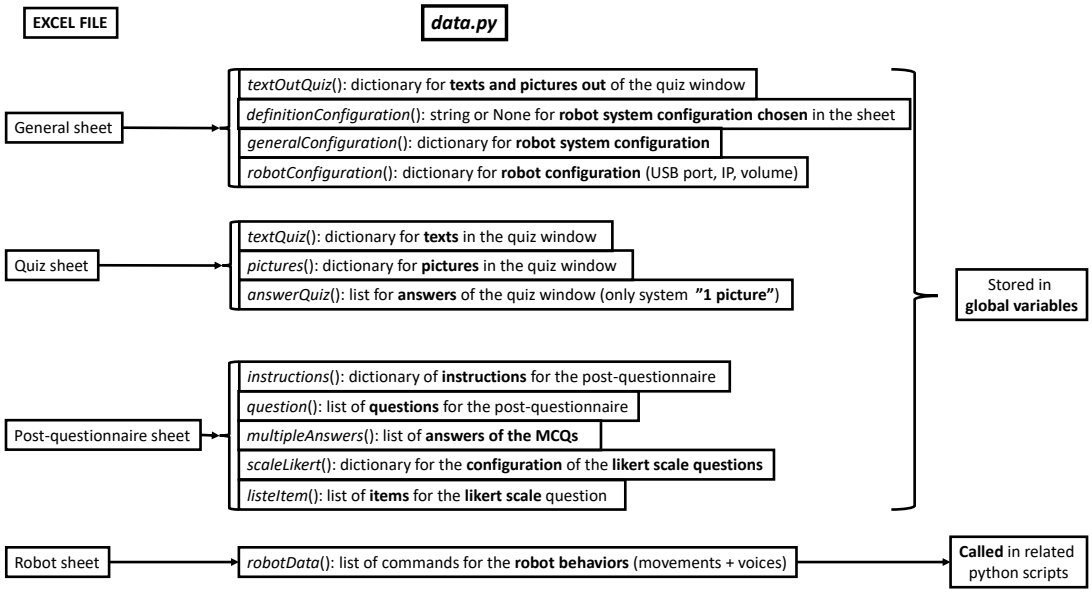


Figure 1.4: This diagram shows the most important functions defined in *data.py*, from where and which data they read and how they are stored in the program.

Exceptionally, the use of global variables was the most convenient way for this part. The values assigned to those variables are just information extracted from the Excel file (and reorganized in a more practical format, e.g. list or dictionary) that the other parts of the program need to display the wanted elements but these global variables will never be modified.

1.2 Graphical User Interface

1.2.1 view.py

This script contains three parts: the creation of windows for the experiment, the creation of alert windows and the main function to launch the whole program. The flowchart can be found in figure 1.5.

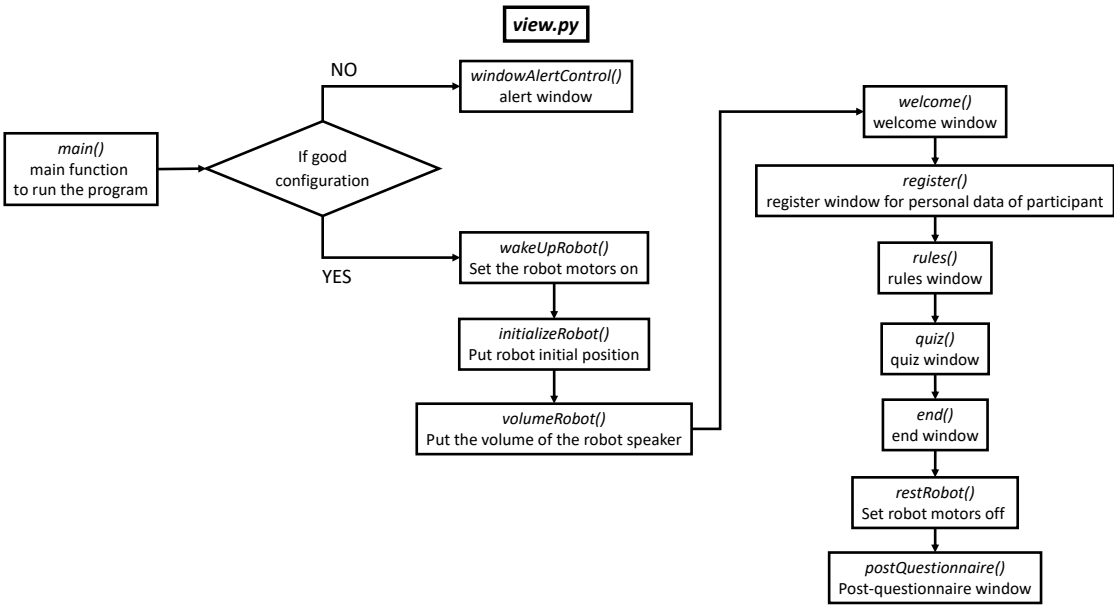


Figure 1.5: Flowchart of the python script *view.py*.

1.2.2 logic.py

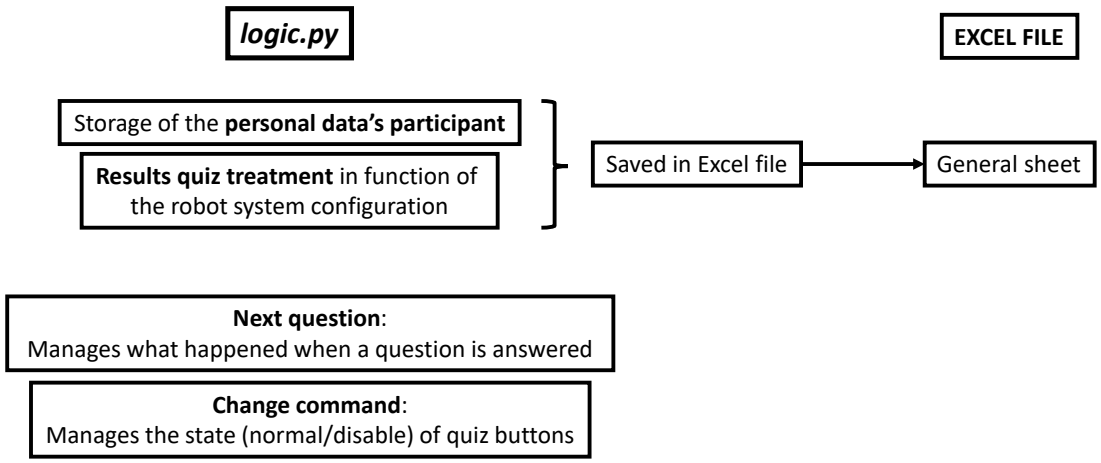


Figure 1.6: Flowchart of the python script *logic.py*. Only the main functionalities are presented here and not the functions.

Concretely, *logic.py* attends to:

- do some verification's about the inputs from the participant for the personal data

- configure some aesthetic points, e.g. size of the pictures
- manage the events for the interactive widgets, in particular the button(s) during the quiz
- save all the inputs from the participant (i.e. personal data, answers for the quiz & answers for the post-questionnaire) and to write them in the correct sheet of the Excel file. General information like the date and hour of the experiment and the type of robot platform are also saved in the same Excel sheet for an easier visibility

1.3 Controller

The "Controller" part does the link between the graphical interface itself and the commands for the robots.

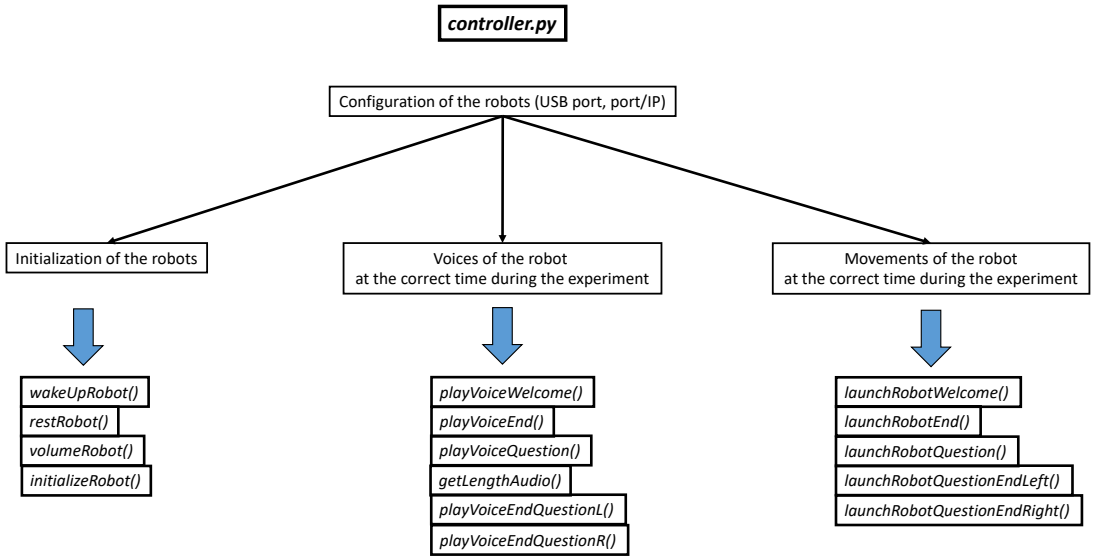


Figure 1.7: Flowchart of the python file *controller.py*.

1.4 Robots

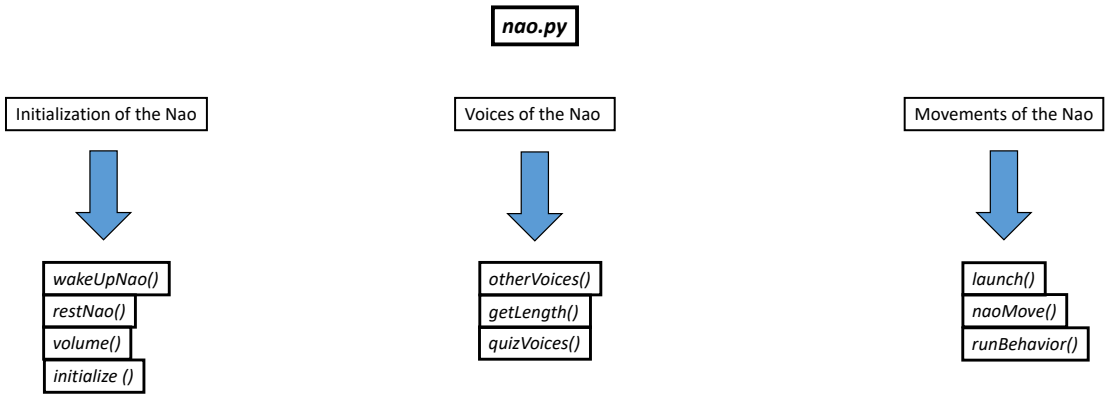


Figure 1.8: Flowchart of the python script *nao.py*.

For each part of the experiment where the robot can interact with the participant (welcome, quiz and end window for now), two sorts of instructions can be given in the Excel tables: the name of a **behavior** or a **number**. The latter represents the time in second that the experimenter would like that the robot waits before executing the next instruction (which can be a behavior or another waiting time). For example, the question 1 has the following instructions: [2,behavior_1,behavior_2,3, 5,behavior_3]. It means that the robot will first wait 2 seconds, then execute the behavior_1, behavior_2, wait 3 seconds, 5 seconds and finally execute behavior_3. This particular configuration allows to synchronize each movement of the robot with its dialogue.

The voices are exactly defined in the same way than the movements in the Excel file, except the fact that here, it is not the name of a behavior but the audio file name with the extension which is written.

1.4.1 My Keepon robot

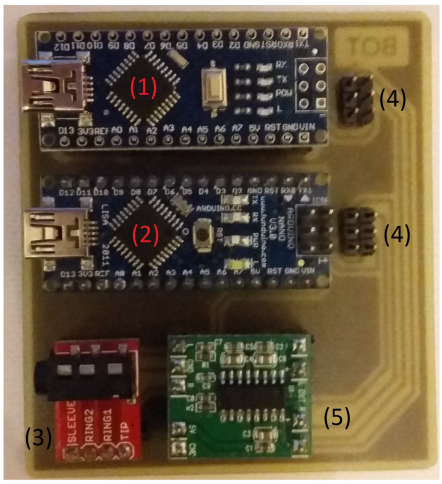


Figure 1.9: PCB with all the components to use simultaneously two My Keepon robots: (1) Arduino Nano for one Keepon, (2) Arduino Nano for the other Keepon, (3) speaker, (4) 2x6 pins, (5) amplifier.

The programming to control the My Keepon is quite simple. It consists to send the instructions (*bytes* type) to the Arduino Nano which will execute the related behaviors to the My Keepon. In fact, only a few steps are necessary to use My Keepon:

1. Setup the communication between the computer and the Arduino Nano which is done through USB connection (using of the module *serial*);
2. Send the wanted instructions to the robot with the method *write*.

About the voices, a specific python module needs to be used to play the audio files with the computer. Several modules exists to do so and it was decided to install *pyglet* because it accepts a wide panel of audio file formats.

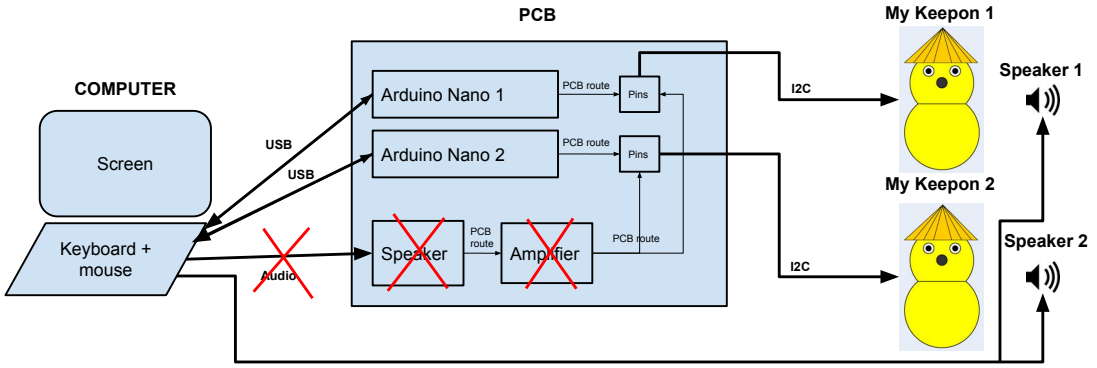


Figure 1.10: Connection between the computer and the My Keepon robots via the PCB

1.4.2 Nao robot

Aldebaran-Robotics provides several tools to fully exploit the Nao robot. The most interesting for this work are the following:

- Choregraphe: this is a software which provides a friendly and powerful graphical interface to configure and program a Nao robot. It can also generate a simulated Nao to test behaviors.
- SDK: several SDK (*Software Development Kit*) have been developed to allow the programming of the robot. For now, there exists the SDK in Python, C++, Javascript and Java.

Choregraphe Only the main functions of the software have been exploited for the master thesis and many tools will not be described in this section. Choregraphe has been used to configure the Nao robots, to adapt and load existing behaviors into the robot and to simulate the robot.

Python The control of Nao with Python is again quite easy. At the opposite of My Keepon, the communication between the computer and the Nao can be performed completely by wifi. It should be initialized by encoding the port and IP address of the robot. Then, one just needs to use the right NAOqi APIs to get access to the related methods and be able to control the robot as desired. With these APIs, all the functionalities of Nao are accessible for the programmer from Python. As part of the master thesis, only a few APIs have been used: "ALMotion" to wake up (activate the motors) and put at rest the robot, "ALAudioPlayer" to play audio files loaded on the robot and "ALBehaviorManager" to play the behaviors defined in Choregraphe and also loaded on the robot.

1.5 Post-questionnaire

This last part is launched just after the end of the experiment. It aims to collect the participant's opinion on the experiment and the interaction of the robots. It was programmed such that the questionnaire can proposed 4 types of questions:

- Open questions
- Multiple choice questions where several answers can be chosen
- Multiple choice questions where a single answer can be chosen
- Questions with Likert scale answer

The participant's answers are then saved in the Excel file, in the specific sheet.

Chapter 2

User guide of the detection system

This appendix aimed to give the necessary instructions to use correctly the Excel file and the Python code that will then permit to perform new experiments. The Excel file named ***Quiz_2_Pictures.xlsx*** is for the GUI 2 Pictures while the file named ***Quiz_Multiple_Answers.xlsx*** is for the GUI 1 Picture. The configuration of the two GUI is similar and then will be described at the same time. The Python codes and the related Excel file must be placed in **the same folder**. Also, when you create a name for a file/behavior, don't put any space! For example, "Picture 1" is forbidden, you must write "Picture_1", or "Picture1", or something similar.

2.1 Python Codes

The Python codes is shared between 7 files, all placed in the same folder: *data.py*, *view.py*, *logic.py*, *controller.py*, *nao.py*, *keepon.py* and *postQuestionnaire.py*.

The version used is Python 2.7 because the SDK of Nao is only designed for this version. Several integrated modules are used: tkinter (GUI), time, thread and math. Also, to run correctly the program, the following modules need to be installed:

- **pillow** (for the pictures)
- **openpyxl** (to read and modify Excel)
- **pyglet** (for the multimedia files). Moreover, **AVbin** (for audio files) must be downloaded and the file *avbin.dll* must be placed in the same folder than the codes
- **pyserial** (for the connection with the PCB of My Keepon)

A very useful tool to install these modules is **pip**. With that, the installation is done just by typing the command *pip install moduleName* in the Command Prompt (cmd.exe) or terminal and the installation will be done automatically. However, if you do not want to use pip, you can still search on the Internet and install in the usual way.

Finally, to use the Nao with Python, the Python SDK must be installed. The software Choregraphe must also be installed.

2.2 Integration of external files (picture and audio)

All the pictures which are not part of the quiz must be placed in the same folder than the codes. Then, their name must be written at the corresponding places in the Excel

file (see section 2.4). Concerning the voices, they must be all placed in the folder "Voices" which is in the same folder than the Python codes.

According to the documentation, you can use with pillow more than 30 different format for the pictures, including PNG, JPEG and GIF. You can check the website https://pillow.readthedocs.io/en/latest/handbook/image-file-formats.html#image-file-formats for more information about that.

According to the documentation, you can use with pyglet different formats for the audio files, including MP3, WAV and WMA. You can check the website https://pyglet.readthedocs.io/en/pyglet-1.2-maintenance/programming_guide/media.html#supported-media-types for more information about that.

2.2.1 GUI with 2 Pictures

With this GUI, you will obviously need two pictures per question. The pictures at the left side must be placed in the folder named "ImagesLeft" and the pictures at the right side must be placed in the folder named "ImagesRight".

2.2.2 GUI with 1 Picture

With this GUI, you will need 1 picture per question. The pictures must then be placed in the folder named "Pictures".

2.3 Definition of robot behaviors

The definition of a behavior for a sequence is done by the following structure.

Let's imagine that you want to define the movements of the robot for the question one. You will have to give a series of instructions that can be composed of a number (for waiting time in second) or a movement. For example, [movement_1, 3.4, 4, movement_2, 5, movement_3]. It means that the robot will execute movement_1, wait during 3.4 seconds, waiting during 4 seconds, execute movement_2, waiting during 5 seconds and finally execute movement_3. **The structure is exactly the same than for the definition of the dialogues with the audio files.**

2.3.1 My Keepon

Here are the possible behavior of the My Keepon. The instructions written in the Excel file (see section 2.4) must have **exactly the same syntax** (without the quotation marks):

- "SPEED [PAN, TILT, PONSIDE] <0...255>;"

- "MOVE PAN <-100...100>;"
- "MOVE PAN <-100...100>;"
- "MOVE TILT <-100...100>;"
- "MOVE SIDE [CYCLE, CENTERFROMLEFT, RIGHT, CENTERFROMRIGHT, LEFT];"
- "MOVE PON [UP, HALFDOWN, DOWN, HALFUP];"
- "MOVE STOP;"

2.3.2 Nao

The definition for Nao is a bit more complicated but allow to do much more behaviors. To do so, you must use Choregraphe. On the figure 2.1, you can see an illustration of the software. Here is a little description:

- (1): the configuration of Nao is done in this part, i.e. the wi-fi connection, the initialization of the joints, the volume of the sounds, etc.
- (2): this is the current project where are defined all the behaviors necessary for the study
- (3): a series of basic behaviors are regrouped in libraries. Additional libraries can added. For the needs of the master thesis, a libraries composed of many behaviors has been provided
- (4): this area attends to provide an easy tool to modify and test a (series of) behavior on a simulated or real Nao
- (5): this window shows the results on a Nao while running a behavior. Generally used to simulate a Nao robot, it also plays the behaviors while it is running on a real robot
- (6): this window shows the projects which have already been loaded on the Nao robot. Hence, it is important to not forget to load the updates of the project defined in (2)

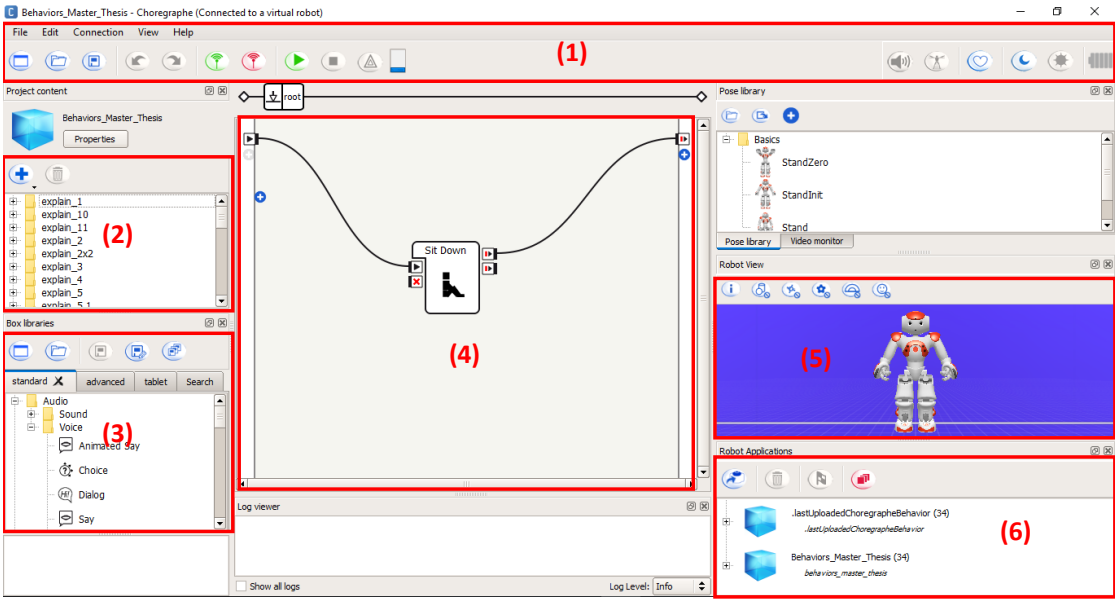


Figure 2.1: Interface of the software Choregraphe to control the Nao robot. (1) configuration of Nao (i.e. wi-fi connection, initialization of the joints, volume of the sounds, etc.); (2) current project with saved behaviors; (3) a series of predefined behaviors that can be exploited; (4) area to easily modify and test a (series of) behavior on a simulated or real Nao; (5) production of a behavior running on a Nao (simulated or real); (6) projects already loaded on the Nao robot.

All the behaviors must be saved in the project "Behaviors_Master_Thesis". To open the project in Choregraphe, you must open the name with the same name but with the extension .pml (this file is in fact itself in the folder "Behaviors_Master_Thesis"). Here is the way to add, modify or remove behaviors for Nao when you have opened the right project:

1. Connect to your Nao

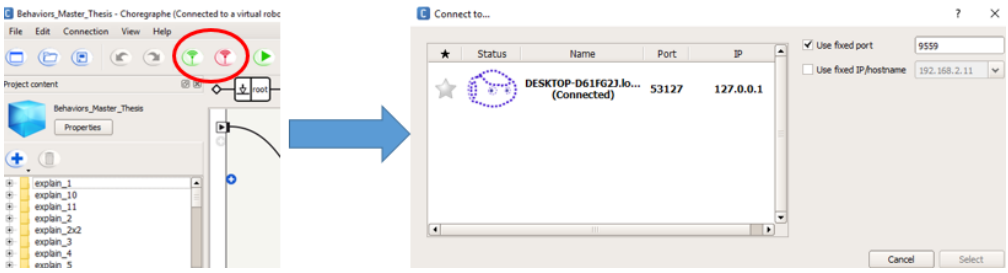


Figure 2.2: Wifi Connection

2. Create your behavior, click on the right button and give a name to the behavior.

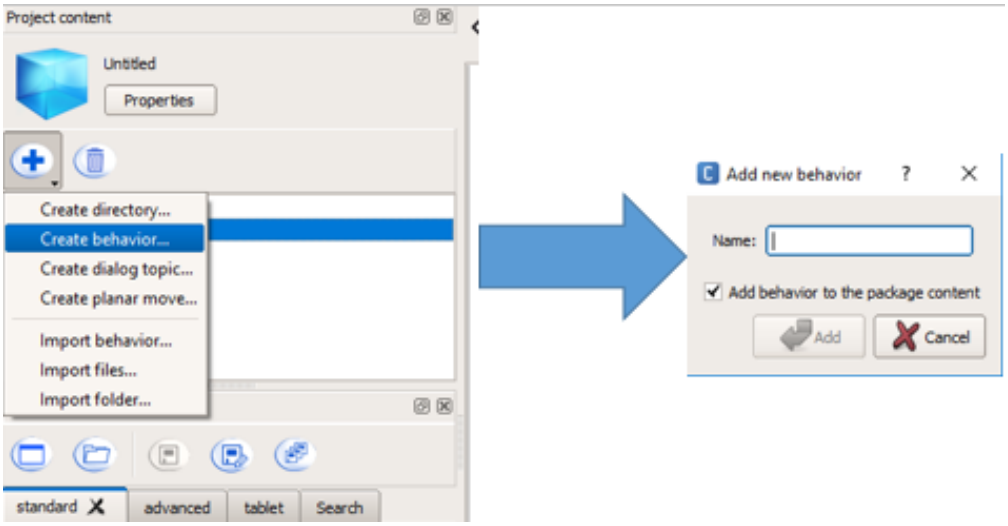


Figure 2.3: Creation of a new behavior for the project.

3. From the libraries, extract the wanted behavior place him in the frame to run a behavior. Make the right connection. Then save. If you want to create another behavior, repeat point 1.

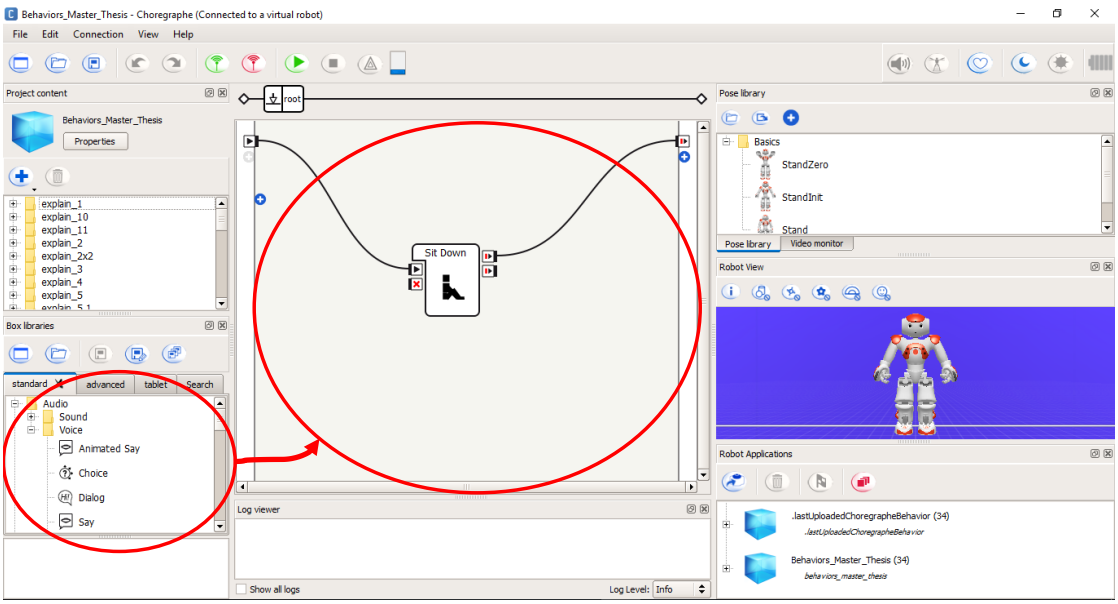


Figure 2.4: Definition of a behavior.

4. Now, you must upload the new version of the project (with the new behavior) into your robot.

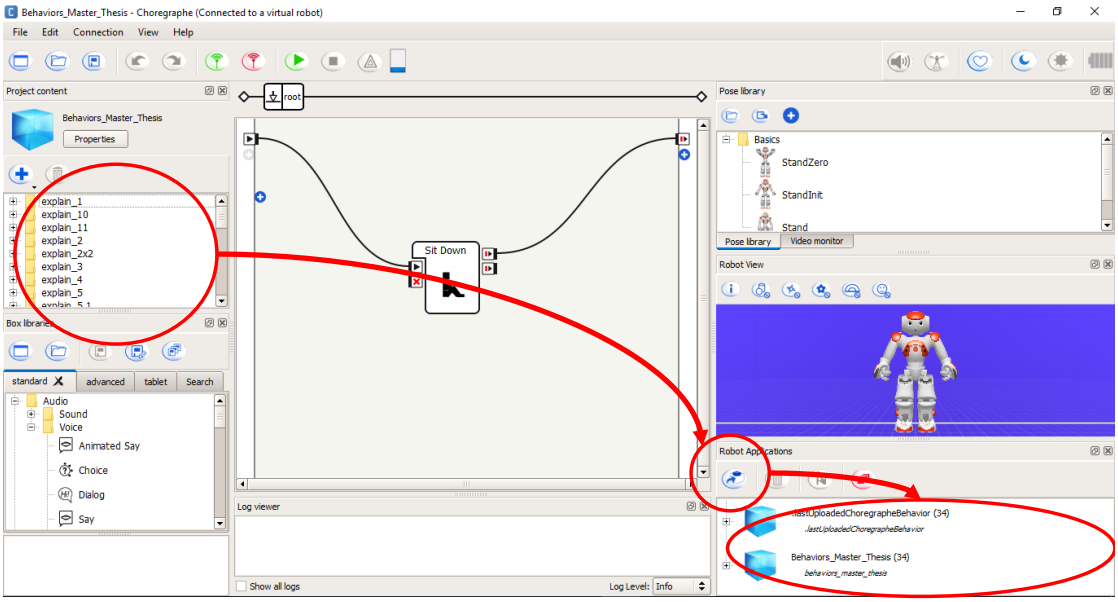


Figure 2.5: Upload of the project into the robot.

5. About the audio files, you must also upload them into the robot. When you will do the file transfer, it could ask you the password of the robot and then, you will be able to upload manually your audio files. Be careful to upload them in the folder "voices".

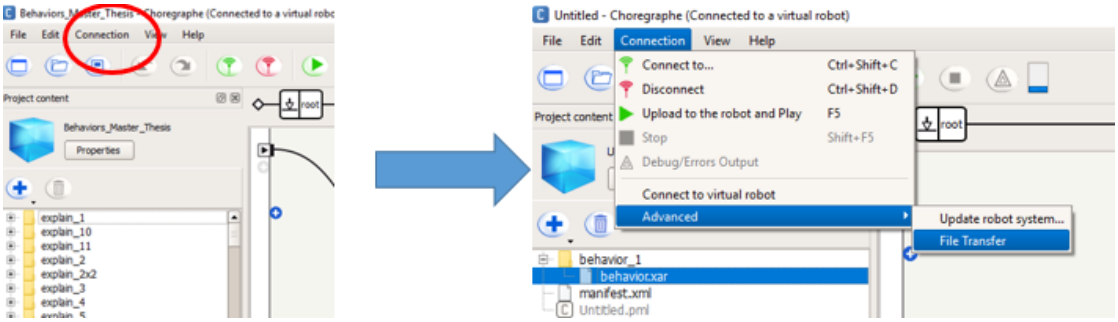


Figure 2.6: Upload of the audio files into the robot.

Now, you can play your behavior on the robot! Some other tips:

- To run a behavior, click on play and you will see the behavior on the screen. If you are using a real robot, you must first "Wake up" at the right top, which means turn on the motors. You can also fix the sound volume. When you finished with the real robot, don't forget to bring it back at rest.

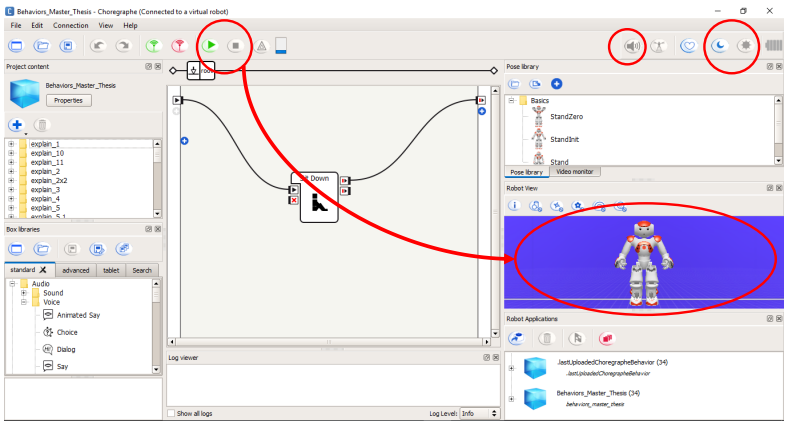


Figure 2.7: Running a behavior.

- If you decide to create a **new** project, you have to be careful about the ID of the application. That is the latter that you have to write in you Python code if you want to play the behaviors of this project.

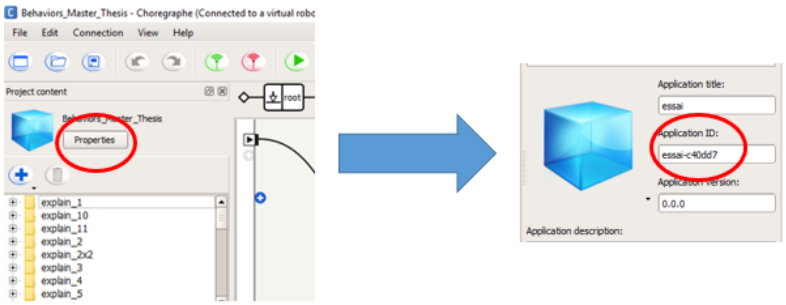


Figure 2.8: Creation of a new project.

2.4 Excel File

The Excel file is composed of 11 sheets which can be regrouped in 6 parts which are described in this section. When it is asked to add a picture/audio file, you have to write the name of the picture/audio file file with **the extension** (e.g. .png, .gif, .mp3...). For each sheet, there is one or more screenshot to illustrates their content. Colored

frames have been added to show the difference sections of each sheet and **these frames designate the only parts that you can modify for your experiment, the modification of the rest being possible only if the appropriate modification in th Python code are correctly performed (except if the contrary is mentioned).**

2.4.1 General sheet

The General sheet regrouped two types of data that have to be defined:

- **The general configuration**: it implies the robot position (e.g. which side is placed the male robot) and the robot platform (e.g. 2 Keepon, 2 Nao, and so on) with additional options to facilitate the tests during implementation. To choose which configuration you want, you must put the number "**1**" at the corresponding cell for both part and blank cell for the other. Below, there are also two other cells named "First number question i-1" and "Last number question n". There are also only for the development. In a real experiment, these two options are deactivated, and the program will calculated the total number of questions with the number of questions written in the Quiz sheet (see section 2.4.2)
- **The general content of the graphical interface**: all the texts and pictures present outside the quiz window (i.e. for now, welcome, personal data, rules and end).

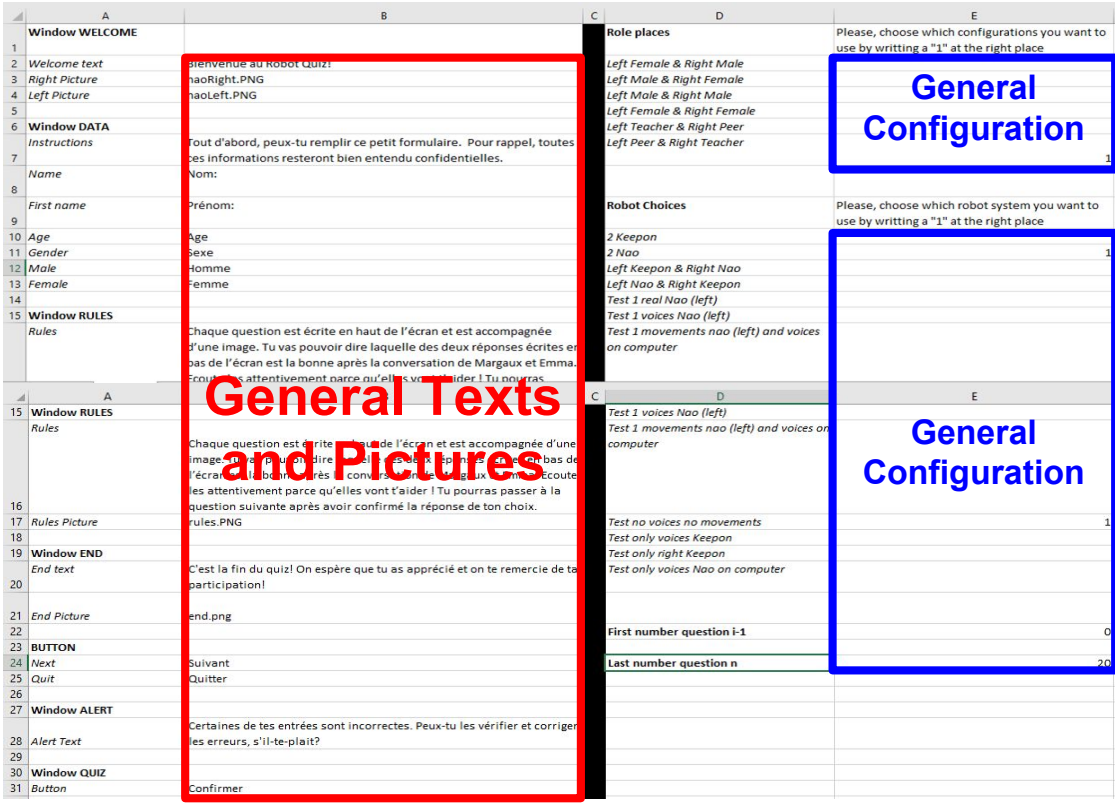
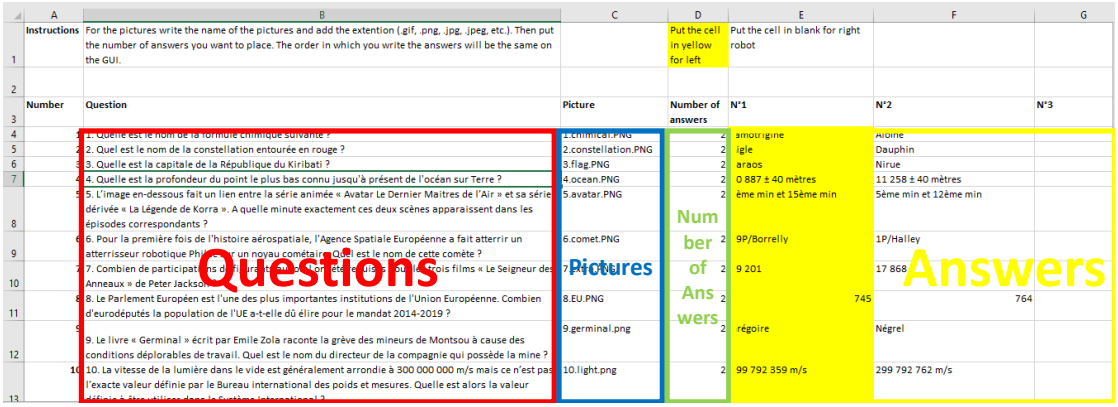


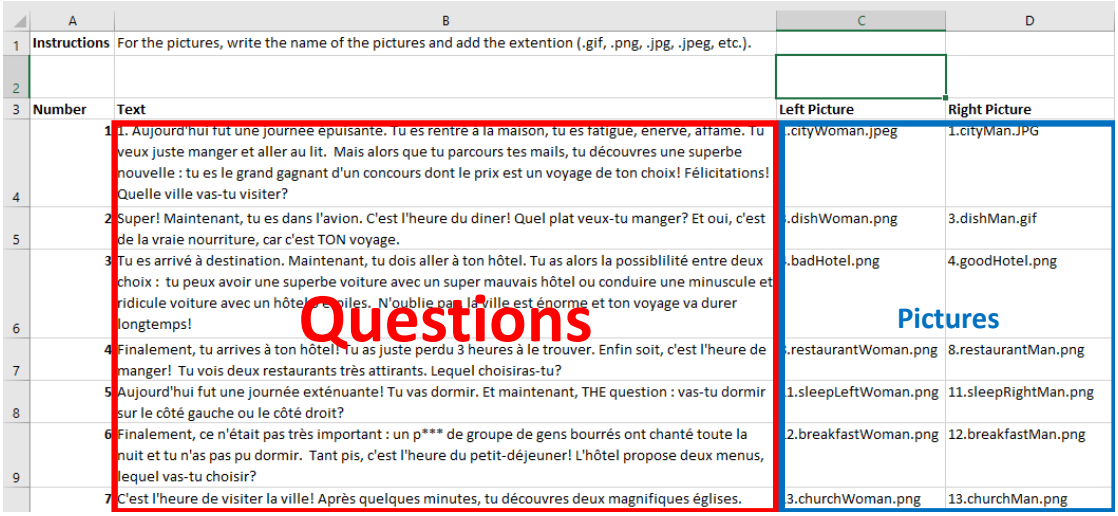
Figure 2.9: Screenshot of the General sheet, where the red frame is dedicated to the text and picture for the GUI and the blue frame is for the configuration of the robot platform.

For each frame from the figure 2.9, the first column is to say what kind of information should be placed and the second column is where to write the information. As a reminder, it is essential to **not touch to the first column** of each frame, except if the corresponding changes are also done in the Python codes.

2.4.2 Quiz sheet



(a) Partial screenshot of the Quiz sheet for the GUI 1 picture. Note that the attribution of the answers to one of the two robots is done with the definition of the cell color (yellow for the right robot and none for the left robot).



(b) Partial screenshot of the Quiz sheet for the GUI "2 pictures", more simple than for the other GUI, as there are only two answers: the two pictures.

Figure 2.10: Partial screenshots of the quiz sheet for both GUI

This sheet is the core of the content of the quiz. Here will be defined the questions, pictures and/or answers for the experiment. On the figure 2.10 can be found a screenshot of the quiz sheet for the GUI 1 Picture (2.10a) and the GUI 2 Pictures (2.10b).

The red frame is for the questions. This is this list that will determine the total number of questions for the whole program. You can add, modify and remove as many as questions that you want. To do that, you can so add or remove the corresponding

rows. The number of the first column A is not important, it is just to have a visual overview of the quiz. However, the program will read all the cells until it encounters a blank one. It means that if you let a blank cell between two questions, **all the questions below the blank cell will not be taking into account. This rule is also applied to the other frames.**

The blue frame is for the pictures. As expected, there is one column for the GUI 1 Picture and 2 columns for the GUI 2 Pictures, one for right robot and the other for the left robot. Remind that you have to write the name of your picture file with the extension.

Finally, there is **the green frame where the number of each questions must be defined and the yellow frame where are written the answers**. These two last frames are not present in the GUI 2 Pictures. **It is very important that the number of answers given in the green frame corresponds to the number of answers written in the yellow frame** as the program will read this number to see how far it should read the corresponding row to get the answers. Also, notice that the attribution of the answers to one or the other robot is done via cell color (yellow for the left robot and blank of the right robot). **Other colors cannot be used in this context.**

2.4.3 Movements Robot sheets

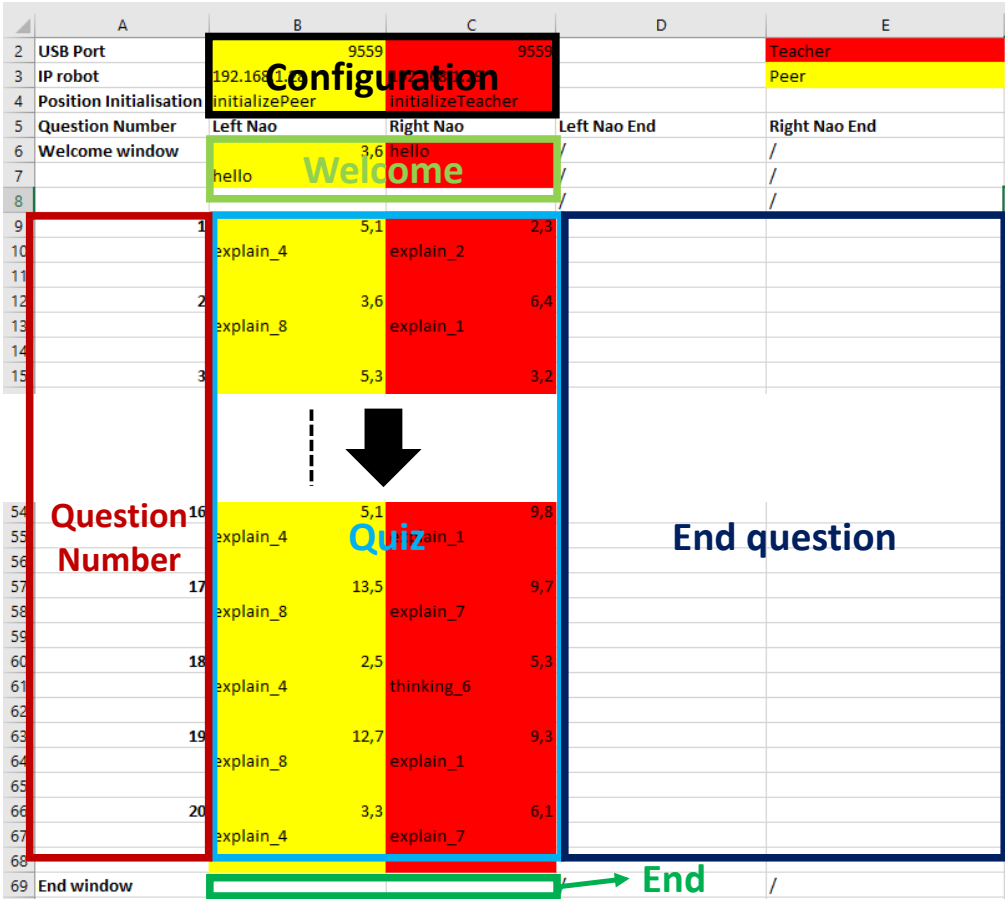


Figure 2.11: Partial screenshot of the sheet for the movements of the Nao robots. Their configuration (IP, port and initial position) and their movements for the experiment are defined in this sheet. The cell colors were just put for simplicity to clearly show which part is for which robot and these colors are not used in the code.

There are one sheet per type of robot, i.e. two here: Keepon sheet and Nao sheet. All the movements for the whole experiment are defined in those sheets. **For now, they are only for the welcome, quiz and end window.** The way to define the movements is exactly the same for both robot.It means that in fact, the sheet for the My Keepon and the one for the Nao are extremely similar, except for the robot configuration (USB for My Keepon, port and IP for Nao). Furthermore, the structure is also identical for both GUI. That is why there is only one screenshot of the Nao sheet at the figure 2.11 in this section. Note that the color cells are absolutely not important here, there were just put to facilitate the reading but they can be changed

and removed without any impact for the program.

On the figure 2.11, there 5 main different frames (the last one will be talked a bit later): the black to configure the robots and their initial position, the light green for the welcome window, the normal green for the end window and the two blue for the quiz. The column B is for the left robot and the column C is for the right robot. About the quiz part, there are so two subpart: the light blue frame which is devoted to the movement for the questions themselves and the dark blue frame which are the movement when the question is answered. For example, if at the end of a question, the participant has chosen the answer of the right robot, the latter will execute the behavior defined in the column E while the left robot will do nothing. Also, **be careful to make a perfect correspondence between the number of questions defined in these sheets and the number of questions defined in the quiz sheet** (section 2.4.2).

Finally, it remains the **definition of the behaviors**: for each sequence (the welcome, each question and the end), you have to write is a series of instructions that can be composed of number(s) (for a waiting time) and/or defined movement(s) (see further details in section 1.4). Typically, on the figure 2.11, you can see for the question 1 for the left Nao that it is written the number "5.1" followed by the behavior "explain_4". It means that when the question 1 is launched, the left Nao will first wait 5.1 seconds before executing the behavior explain_4 which has been loaded on the robot. In this example, there only two commands per sequence but you can add more commands if needed. Contrary to the other sheets, the dark red frame with the question numbers is important must be updated following you needs. This is through the column A that the program will be able to know when a series of commands must be executed.

2.4.4 Voices Robot sheets

The voices of the robots are defined exactly in the same structure than with the movements, i.e. one sheet per types of robot (Keepon Voices and Nao Voices sheet), identical for both GUI and composed of series of instructions number(s) and/or audio file(s) (with their extension). That can be observed on the figure 2.12, again for Nao only. The frame functions are identical to the the one described in section 2.4.3 for the movements sheet. The only difference is that the sound volume can be fixed for the Nao while it is not the case for the My Keepon. Again, the cell color has no importance here.

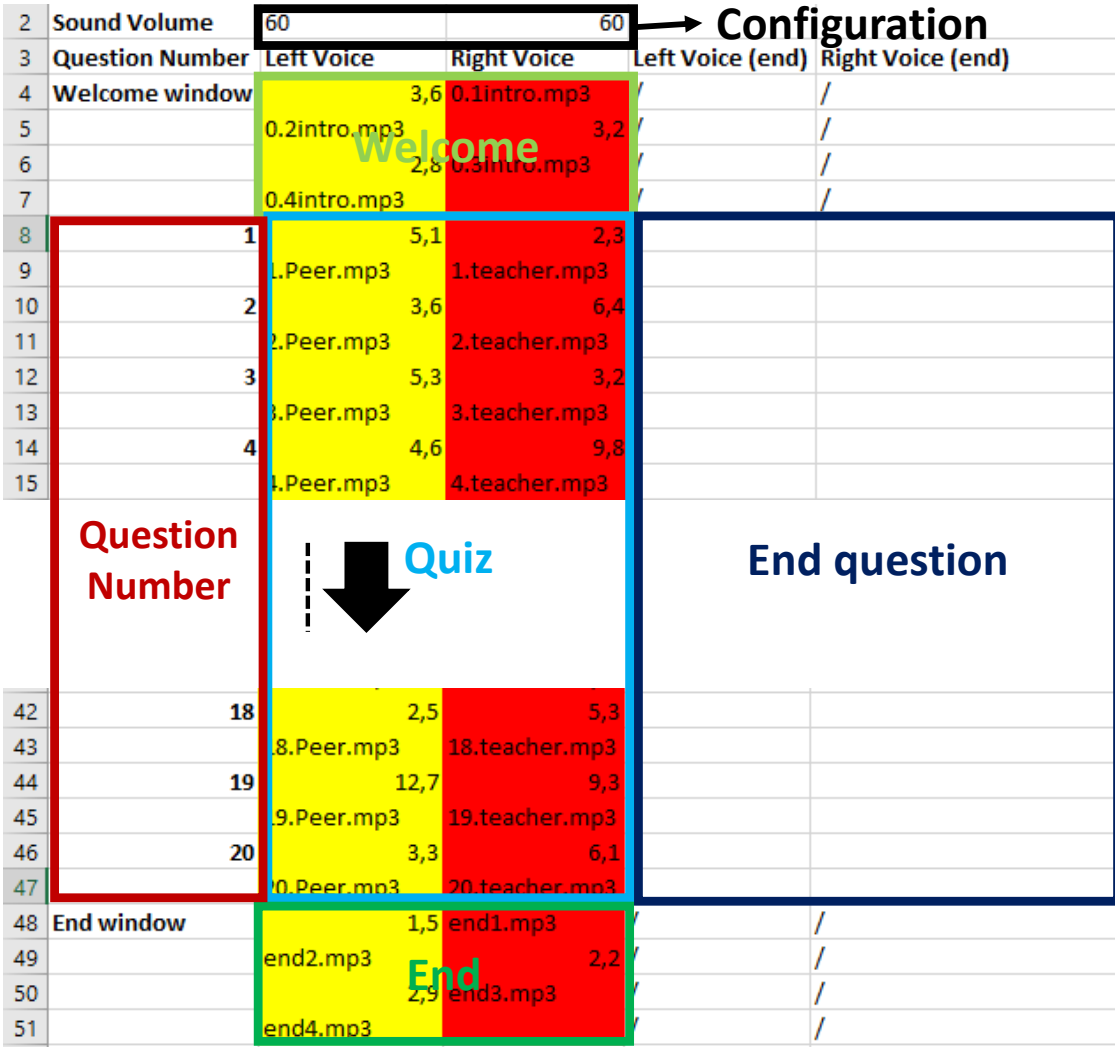
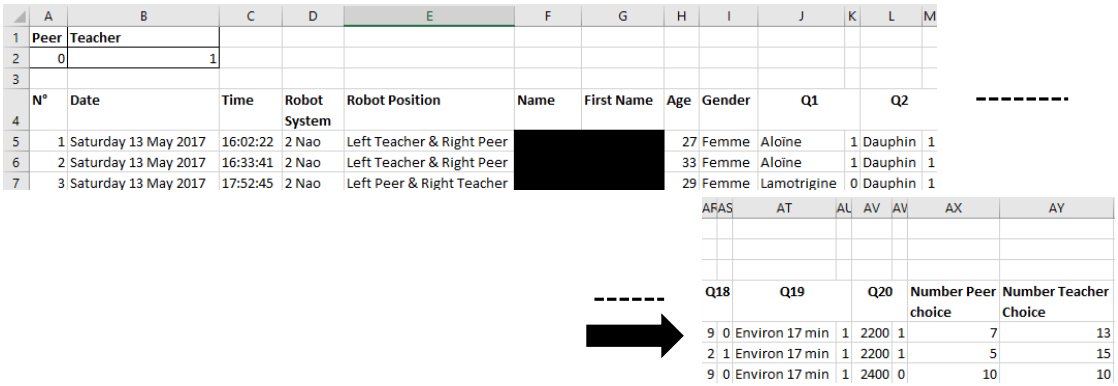


Figure 2.12: Partial screenshot of the sheet for the voices of the robots.

2.4.5 Results sheets

This part regrouped all the personal data and results obtained after the experiment. It is structured such that one type of experiment has its own sheet to have a clear separation between the studied variables (e.g. one sheet for gendered voices, one sheet for teacher versus peer...). There are for now three results sheets. Again, there is an illustration for only one sheet on the figure 2.13 because the structure is identical for any types of studies. However, there is a small difference concerning the types of GUI used: if it is the GUI 1 Picture (as on the figure 2.13), the answer and the corresponding robot is saved for each question while in the GUI 2 Pictures, only the chosen robot is saved as there are only two possibilities.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Peer	Teacher											
2	0	1											
3													
4	N°	Date	Time	Robot System	Robot Position	Name	First Name	Age	Gender	Q1		Q2	
5	1	Saturday 13 May 2017	16:02:22	2 Nao	Left Teacher & Right Peer	[redacted]		27	Femme	Aloïne	1	Dauphin	1
6	2	Saturday 13 May 2017	16:33:41	2 Nao	Left Teacher & Right Peer			33	Femme	Aloïne	1	Dauphin	1
7	3	Saturday 13 May 2017	17:52:45	2 Nao	Left Peer & Right Teacher			29	Femme	Lamotrigine	0	Dauphin	1

AR	AS	AT	AU	AV	AW	AX	AY
Q18		Q19		Q20		Number Peer choice	Number Teacher Choice
9	0	Environ 17 min	1	2200	1	7	13
2	1	Environ 17 min	1	2200	1	5	15
9	0	Environ 17 min	1	2400	0	10	10

Figure 2.13: Partial screenshot of the sheet for the data and results of the experiment. This picture concerns the tests performed for the master thesis (Peer versus Teacher). The cells related to the names and first names of the participants have been hidden to preserve confidentiality. In this situation, 1 means that the answer of the teacher robot has been chosen while 0 means that the answer of the peer robot has been chosen.

2.4.6 Post-questionnaire sheets

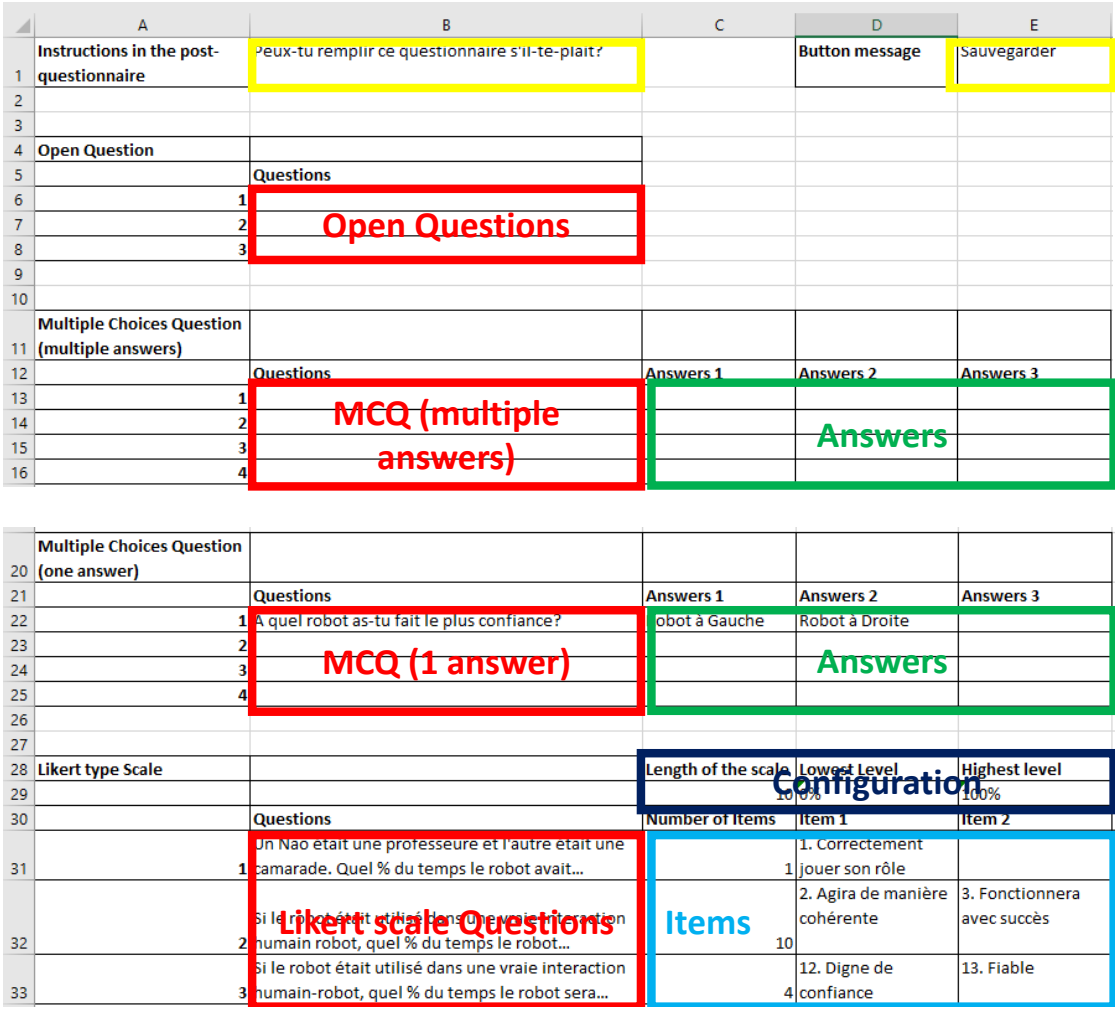
This last part concerns the eventual post-questionnaire that the participant might have to complete to give his/her opinion on different facets of the experiment. There are two sheets, one to write the content of the questionnaire (see figure 2.14a) and the other to save the participant's answers (see figure 2.14b).

For now, the post-questionnaire can accept open questions, multiple choices questions where several answers can be chosen at the same time, multiple choices questions where only one answer can be chosen at the same time, and questions with Likert scale (cfr. figure **??**).

On the figure 2.14a:

- **Yellow frames**: Instructions message that can be changed if needed
- **Red frames**: To write the questions. It can be added as many questions as wanted for each types of questions.
- Green frames: To write the multiple answers for the MCQs.
- **Dark blue frame**: To configure the likert scale questions (number of level for the items, which is the highest and the lowest).
- **Light blue frame**: For each question, the number of items must be defined and then on the same row, the corresponding items are put. The first column is primordial as it is this number that the program will read to know how far it

should read to extract all the items for the question. Finally, each likert question (and so items) will be presented two times on the GUI: one for each robot.



(a) Partial screenshot of the sheet for the questions of the post-questionnaire.

N°	Date	Time	Robot System	Robot Position	Name	First Name	Age	Gender	MCQ (1 answer)	Robot Type	1. Correctement jouer son rôle	2. Agira de manière cohérente
1	Saturday 13 May 2017	16:12:06	2 Nao	Left Teacher & Right Peer	[redacted]		27	Femme	Robot à Gauche	ROBOT A GAUCHE = PROFESSEURE	10	8
1	Saturday 13 May 2017	16:12:06	2 Nao	Left Teacher & Right Peer			27	Femme	Robot à Gauche	ROBOT A DROITE = CAMARADE	10	7
2	Saturday 13 May 2017	16:33:41	2 Nao	Left Teacher & Right Peer			33	Femme	Robot à Gauche	ROBOT A GAUCHE = PROFESSEURE	8	8
2	Saturday 13 May 2017	16:33:41	2 Nao	Left Teacher & Right Peer			33	Femme	Robot à Gauche	ROBOT A DROITE = CAMARADE	6	7
3	Saturday 13 May 2017	17:52:45	2 Nao	Left Peer & Right Teacher			29	Femme	Robot à Droite	ROBOT A GAUCHE = CAMARADE	7	8
3	Saturday 13 May 2017	17:52:45	2 Nao	Left Peer & Right Teacher			29	Femme	Robot à Droite	ROBOT A DROITE = PROFESSEURE	8	7

(b) Partial screenshot of the sheet for the answers of the post-questionnaire. The cells related to the names and first names of the participants have been hidden to preserve confidentiality. Notice that there are two rows per participant. Indeed, the items is presented for both robots, so each row corresponds to one of the two robots.

Figure 2.14: Partial screenshot of the two sheets related to the post-questionnaire.