A Humanoid Robot System for Train Ticket Checking

*Project thesis with Code https://gitlab.lrz.de/ge97zoj/humanoid_ws22, Humanoid Robotic Systems, Technical University of Munich

Chongyu Zhang, Tao Ma, Yinglei Song, Yi Zhang

Chair of Cognitive Systems, Technical University of Munich

***Abstract*—The railway transportation system is constantly seeking ways to improve the passenger experience and streamline operations. The ticket checking process is an area in need of improvement, as it is often slow and congested during busy periods. This paper proposes a solution to these issues by introducing a humanoid robot system for train ticket checking. Therefore this proposes of the project a ticket checker based on NAO robot to meet the above need and it will serve as an automated approach towards a robotics real-time interaction with the human body. The system consists of a humanoid robot equipped with computer vision and natural language processing capabilities. The robot is capable of detecting and recognizing train tickets, verifying the ticket information against the database, and communicating with passengers to inform them of their ticket status. The system has been tested in a simulated train station environment at ICS-LAB from Technical University of Munich (TUM) and has demonstrated high accuracy in ticket verification and effective communication with passengers. The introduction of this humanoid robot system is expected to improve the efficiency and convenience of the ticket checking process and enhance the passenger experience at train stations.**

***Index Terms*—Keywords: Artificial Intelligence, Robotic, Computer Vision, Humanoid Robot System, Control, Natural Language Processing**

I. Introduction

COVID brought with it a series of hygiene measures aimed at reducing contact between people as much as possible. One of these measures was the rapid introduction of smart ticketing systems for accessing areas, with a particular focus on public transport. In fact, using a unified and digital system in this sector is one of the most interesting applications of future technologies for public transport. Smart tickets are digital alternatives to printed tickets that involve various solutions sharing the same digital systems [1]. There are several types of smart tickets, including mobile tickets, e-tickets, and m-tickets. Mobile tickets are stored on a smartphone after purchase, while E-tickets are issued in PDF format and have a Barcode. All of the tickets need the QR Code or Barcode to store the ticket and passenger information [2]. In order to reduce the time required for train ticket checking, we want to design a Train Ticket Checking by using the NAO Robot, which will help ticket checker to check the ticket with QR code and the passenger information when the passengers first getting in the train and getting off the train. During this period, there will be a series of reactions such as bowing head and blinking eyes and can answer the series of questions raised by passengers.

II. Description and Method

First of all, the NAO Robot in the Figure 1 will take will go for a certain distance and then he will turn his head towards the passenger. After the waiting time if there are passenger getting on board, he will speak the words of welcome and try to say that the passengers to show the QR code of the purchased ticket or the digital ticket stored in mobile Phone. Then if the passengers on the train or on the trip have some question about the entire trip or information about the destination such as the time, the time of arrival, train timetable about the nect station and the weather information. And if the train is arrived at the destination, the NAO Robot will say that the destination has already arrived and can ask the leaving passengers for now for the valid tickets again to check, whether the station is true and make the passenger get off the train. And the person counting can also be executed at the same time by watching the face/ see the QR Code of the passengers. If there is a new passengers coming up then execute a new cycle to check the ticket and passenger's personal information. At end of the trip in the simulated environment (Train will runs from start station Munich to the final station Hamburg) the robot will remind all the rest passengers to get off the train.

A. System Components and Architecture

To achieve our goal, our robot must finish the following vision, motion and speech tasks. Its affiliation is well reflected in Figure 2. For visual work we need Face recognition and QR code recognition components, while during the motion of NAO the robot's joints, arms and head are involved in the locomotion. The function of speaking at the end needs to be realized by relying on the elements of answering questions and checking tickets and conducting a certain degree of guidance. During locomotion and conversation, some robot joints may react to better interact with humans. This includes a series of shaking of the head and some movements of the arm after answering the question, provided that these gestures and movements are defined in the python services.

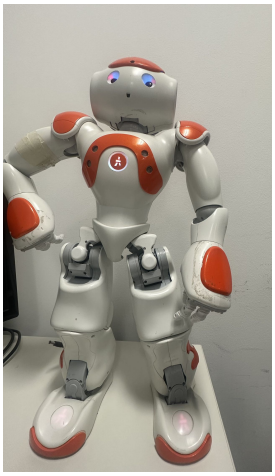

Fig. 1. The NAO Robot as Ticket Checker of initial state.

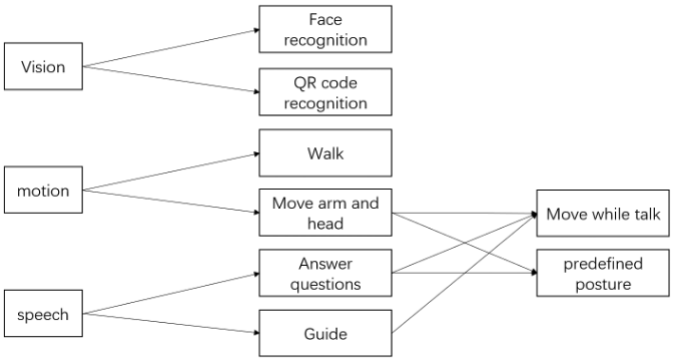

Fig. 2. System Components for finishing overall tasks.

Our state machine in the Figure 3 shows the model used to automatically choose the service and action over time, where the service and action is in one of a finite number of states and can transition from one state to another in response to events or external inputs. The mode switch state machine will automatically switch the state according to the input and send information to the output device during the ticket checking and call interaction process. Our Ticket Checking system consists of a camera, a face detection subsystem, a mode selector, three inputs and two outputs. The camera captures images and sends them to the face detection subsystem, which is responsible for detecting faces within the images. Then according to the judgment test results and code running results, the corresponding mode to be executed will be automatically selected in the main program. The mode selector has three inputs: tactile, camera, and audio. It has inputs and outputs because external to the robot can receive information from different human-computer interaction interfaces to help make the decision to choose the mode. It means that it can receive input signals from these sources and selects the appropriate mode of operation and give the feedback as output. The two outputs of the system are the ability to move joints and speaking.

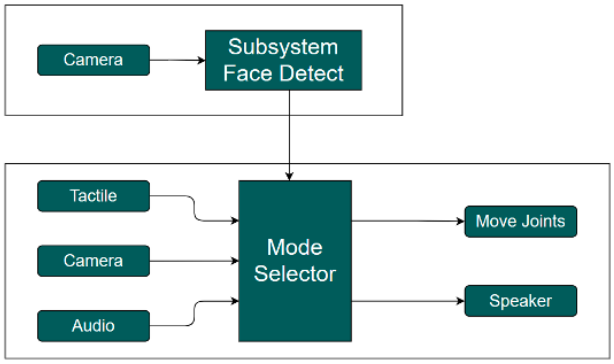

Fig. 3. System Architecture and state machine to switch the modes.

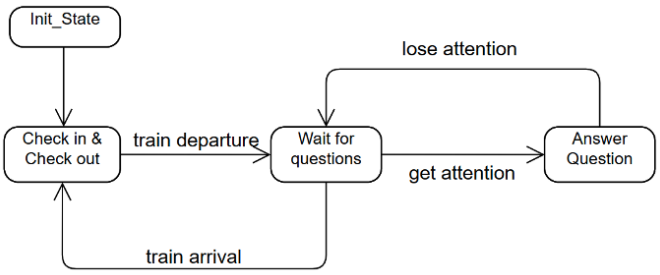

Fig. 4. The Mode Switch of Ticket Checking System.

In according to our Algorithm structure, when the initial state has passengers on board, our robot will move to the designated position and when the robot has finished all preparations, the function of the camera will be called for and collect the current portrait in the camera. By comparing with the portrait in the service and the keyword of voice recognition, it can be determined whether the passengers and tickets are true and correct, then give the defined answer to the passenger. The robot remains in place while the train is moving. If the passenger has a question, the module where the robot communicates with the passenger will be entered. The two callback functions for voice response recognition will be called. The key string of the question mentioned, such as the "north" and "south weather", "station", "time" and other keywords, will be heard, and then query and give a response according to the information defined by the function. When the train arrives at the station, it should switch to the third state and detect the number of passengers. Our detection of the number of people is based on the found ticket information, that is, the facial information of the passengers. This is the whole process of state and switching, the process is shown in Figure 4, and the code is shown below in the algorithm 1.

B. Face Recognition

Here are the details and optimizations for face detection. This part is written as a separate ROS node in Python. In order to

Algorithm 1 Case Switching

$Variable \leftarrow InitialValue$
$Walking$
while $OK \leftarrow ROS$ **do**
  $WaitingForTicket \leftarrow SwitchCase1$
  $CheckingTicket$
  $Callbackfunction$
  $WaitingForQuestions \leftarrow SwitchCase2$
  $QueryingandAnsweringQuestions$
  $Callbackfunction$
  $NewStation \leftarrow SwitchCase3$
  $CountingPassengerNumber$
  $Callbackfunction$
end while

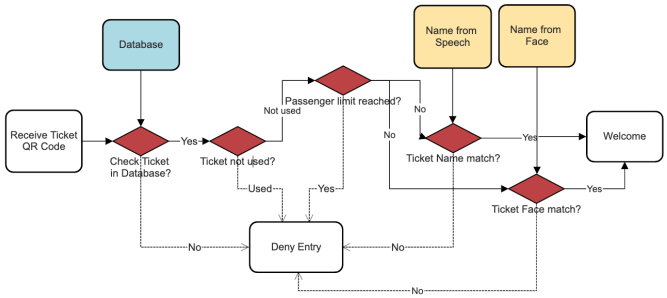

Fig. 5. Flowchart ticket checking system.

save computational power on desktop machine, we decided to run this node on our laptop. The relationship between desktop and laptop are ROS master and slave.

We use an external Python package: face-recognition 1.3.0 [3]. We first load photos of our four team members, and extract face encoding from each photo. These encodings are stored in an array. Once a new frame from NAO top camera comes, we find the face encodings in this frame, and compare them with known face encodings in predefined array. We assign names of closest matches to each face encodings in the frame.

We use std_msgs/Int8MultiArray message for publishing the names of faces in the frame. This array contains four elements. Each of them represent one person. The values of elements are only 1 and 0. Value 1 means we detected this person, while 0 means not. This array needs to be reset to zeros before each ROS spin.

During testing, we observed some problems. Due to low resolution of image and illumination, sometimes recognition results were not stable. This reflected flashing bounding box of face. The crucial part of this problem is that if we happen to receive a unstable recognition while verifying identity of a passenger, result will show there are no people in this frame. This leads to a failure of whole system. In order to tackle this problem, we designed a 2D-array which can contain recognition results five consecutive frames. When we want to publish the name of faces, we add results of five frames together to see who is the most possible person in the image. Finally the value of the name with highest score is set to 1, while others remains zeros. Through this method, we are able to produce stable predictions of faces under various kinds of illuminations. The potential drawback is that we only detect one face in one frame. Since the robot only verify one person's identity at the same time, this would not matter to much.

Face recognition module is also used for attention tracking. This is easily done by summing the values of all the elements in Int8MultiArray message. If result is 0, then robot will switch to next state. If result is not 0, then robot will continue the Q&A state.

C. Count Number of Passengers

Initially we wanted to use top camera of NAO to count the passenger number in real time, but the problem was that face recognition results were not reliable at the distance where all four faces could be included. Therefor we decided to count this number when we scan the tickets. It is realized by checking a used_tag of ticket. 0 means the ticket is not used. It's the initial value. If we have a ticket with tag equal to 0, we add one to passenger number, then set the tag to 1. This means passenger getting on the train. If we have tag equal to 1, we subtract one from passenger number, then set the tag to 2. This means passenger getting off the train. A ticket with 2 as used_tag is not valid anymore.

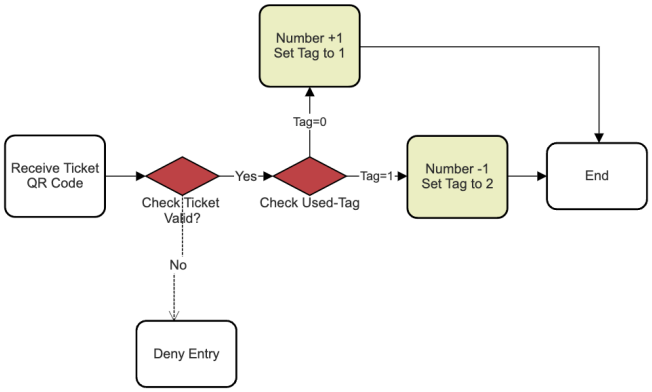

Fig. 6. Flowchart passenger counting system.

D. Speech Recognition

Whenever one person is recognized, the state will be selected and switched in the following way. The purpose of this chart is to hear, check information and to give a voice reply. The speak mode split into two python services. By using the speak_server.py "service" the robot can do our predefined motion. In the hear_server.py we defined the mode 1 for answer the question, input will be the question from passenger and return the key word from question, mode 2 for check the name from passenger, input will be the name from passenger voice and return the name of passenger. In according to the Figure 4. the robot hear ask at first for checking the ticket or ask whether

the passenger has a question, if the robot hear some words (after processing published as messages) should go to the main loop. The Function in the Ticket Checker and Main should be executed to check the ticket and give a feedback. And the ROS messages can also be generated on the command window. And the function answer_questions could be called to give a reply for the human asked question. Like the Figure 7, when the keywords received by the robot are correctly recognized and processed, they will be compared, and then the required voice function will be broadcast by the robot. If the voice is not recognized or is incorrect, you can make a second attempt or directly give the next step according to the situation (if the ticket cannot be recognized, you cannot take the train.). During the execution, the function (waitforticket) will be called and the 2 python services will be called so that the robot can react with blinking red eyes and nodding and shaking the head.

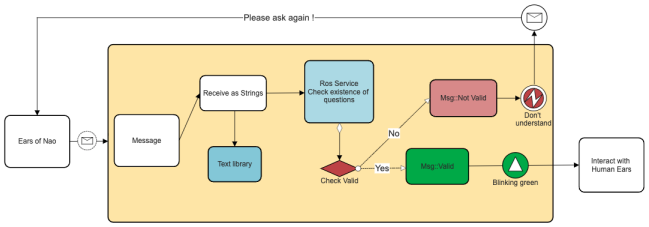

Fig. 7. Speech Flow of Passenger NAO Speech Communication.

III. Experiment and Results

The aforementioned NAO [4] is a small humanoid robot, around 60 cm tall. It includes:

- Two cameras on the head and the tactile touch button
- An inertial navigation unit (accelerometer and gyroscope)
- Internet connectivity over Ethernet cable or WLAN
- Speakers and Ears like Audio
- around 60 minutes battery life.

We chose the official NAOqi Python SDK [5] as the framework for our implementation. In addition, we utilized third-party libraries Face-recognition 1.3.0 [3] and NumPy [6].

We designed our experiments in the following way: As shown in the Figure 8, we have four passengers, the four passengers are Jack, who will take the train from the first station Munich to Hannover in the compartment 1, Mike, who will take the train from Ingolstadt to Hamburg in the compartment 1, Tom, who will take the train from Fulda to Hamburg in the compartment 2 and John who will take the train from Hannover to Hamburg in the compartment 1. Therefore, the total number of people recorded in compartment 1 is 2 persons, that is, Mike and Jack at the beginning of boarding. Then when Tom gets on the bus and enters the ticket check in compartment 1, he needs to be reminded that compartment 1 is full, so he has to go to compartment 2 to find his seat. When Jack gets off at the Hannover station, the number will decrease by 1, but when John gets on the bus, the number will become 2 (FULL).

Passenger	Trip Start → End	Seat in
1 Jack	Munich → Hannover	Compartment 1
2 Mike	Ingolstadt → Hamburg	Compartment 1
3 Tom	Fulda → Hamburg	Compartment 2
4 John	Hannover → Hamburg	Compartment 1

Fig. 8. Passenger Travel Arrangement in Experimental Environment.

In order to test the comprehensiveness of our robot's functions, we also arranged other experiments to test whether it is possible to clearly identify the authenticity of ticket information and the authenticity of personnel information under some error conditions in the Figure 9. Details as follow: If the passenger Mike travels according to his original travel plan, that is, the second row in Figure 9 of the experimental plan, then the information should be completely correct. If he arrange travel according to the first, third, and fourth line in the Figure 9, he will get a voice prompt broadcast by the robot, like ticket is incorrect or the QR code is incorrect or the site information is incorrect. This is the state we expect to achieve.

Passenger	Trip Start → End	Seat in	"Error Case"
1 Mike	Ingolstadt → Hamburg	Compartment 1	With wrong ticket
2 Mike	Ingolstadt → Hamburg	Compartment 1	Correct
3 Mike	Ingolstadt → Hamburg	Compartment 1	With wrong name
4 Mike	Ingolstadt → Hamburg	Compartment 1	With wrong station

1. We assume that our robot will always broadcast the information of the station so that passengers will not get off at the wrong place

Fig. 9. Passenger Travel Arrangement under error conditions.

After testing, the test results have initially reached our expected goal. In the initial state before the start of ticket checking in the Figure 10, the robot can return to its original position, start to move and run for a certain distance, then face passengers, and broadcast vehicle information, train number information, and remind passengers of ticket checking precautions and ticket checking start. Before passing through each station, the robot will prompt the passenger to arrive at the station and ask the passenger to show the ticket to confirm the information that the passenger wants to get off. When the train arrives at the terminal, the robot can sit down slowly like the Figure 10, broadcast the announcement of the arrival of the vehicle letter again, and remind the passengers to pay attention to the ticket check and wish a good journey.

If a normal human face is detected during the journey, a stable red detection frame will be displayed and the name of the passenger will be displayed below, so that the terminal staff can detect it (Figure 11 shows the face information of Jack).

The results of the tests show that the humanoid robot system is highly accurate in ticket verification and effective in communicating with passengers. The robot is able to detect and recognize train tickets with a high degree of accuracy and

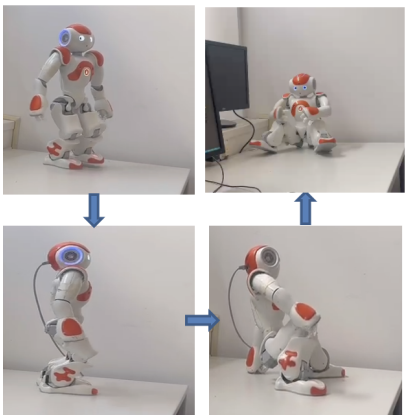

Fig. 10. The NAO Robot can walk for several distance and face to passenger and in the end of the trip sit down like a human.

Fig. 11. Face recognition: The passenger can be fully correct and clear detected and recognized.

verify the ticket information against a database in real-time. The communication between the robot and passengers was found to be clear and effective, with passengers easily understanding the status of their tickets. But in the real environment, we need to further supplement the situation and other states in our program. We have tested that the time of each test cycle is about 15s, which does not include time for follow-up questions as questions are varied and take different amounts of time. After optimization, the human face can be detected very well, and the viewfinder frame of the human face will be stably maintained and detected. Due to the limited experimental environment and the lack of communication with the main station of the Deutsche Bahn, the broadcast information of the voice function is limited to the scope defined by the text library. But all defined text recognition and detection can be well implemented, so that passengers can clearly hear and understand the robot's broadcast and corresponding reply information. Simulated design error conditions and misjudgments worked well, can also detect defined errors in Figure 9 and make feedback as shown in Figure 13.

A. *Time measurement*

The Figure 14 shows the results of the dialogue time measurement after a total of 6 trials according to the schedule. In all the tests experienced at each station and the interaction of each passenger, the ticket checking time spent by our

Fig. 12. In order to detect whether the counting function of people is normal, we defined the case of Tom. He can show the ticket but The robot reacts, shaking its head and say that Tom should go to the next second compartment because the compartment 1 has already been full.

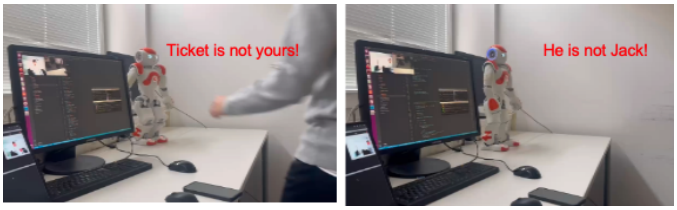

Fig. 13. The invalid case shows that the ticket is not belonging to the passenger or the ticket information is not correct or the name and recognized face can not match together.

robot is about 13 s per person, but 7 to 8 s of it are for voice communication and prompts. If the train information is only broadcast once during the real ticket checking process in the future, then the ticket checking efficiency will be greatly improved. However, it will save a lot of time in the questioning session of passengers, because the robot can know and understand more and more extensive information than the flight attendant, and can quickly query and give a reply, in our experiment around 10 s for each question.

Nr.	Driving Record	Entire Checking Time				With Asking Question
		F.D.	S.D.	T.D.	L.D.	
1	03.02.2023	14 s	16 s	15 s	12 s	+ 10 s each question
2	04.02.2023	15 s	13 s	15 s	14 s	+ 10 s each question
3	05.02.2023	13 s	17 s	16 s	11 s	+ 10 s each question
4	05.02.2023	14 s	16 s	14 s	14 s	+ 10 s each question
5	06.02.2023	15 s	15 s	14 s	15 s	+ 10 s each question
6	06.02.2023	16 s	14 s	13 s	13 s	+ 10 s each question

F.D.: Munich to Ingolstadt; S.D.: Ingolstadt to Fulda; T.D.: Fulda to Hannover; L.D.: Hannover to Hamburg

Fig. 14. The time required to react to the interaction between the passenger and the automatic ticket inspection robot.

IV. Drawback and Conclusion

Firstly, the motion and motion planing for the NAO Robot on board is a bit difficult, because the The space in the carriage is very narrow, and the collision between the robot and the

personnel during the movement is unavoidable. So we just used the installed basic packages, in order to adapt to all common situations. After a period of time, the motor of the robot will heat up and cause uncertain failures, making the robot's legs insensitive or losing control of flexibility. So we set the degree of freedom of the robot in the early stage of work, and tried a variety of methods to make him walk stably, but due to system reasons, libraries similar to colcon could not be installed, so we used the most basic walking method to achieve a simple walking function and fixing the robot to face the passenger for ticket checking and communication. (In addition, we have tried to convert the coordinate system and get the passenger coordinates of the Aruco Marker information, and then try to make the robot walk towards the passenger, but due to the robot itself, the specific position after the coordinate conversion cannot be correctly reached. In the end we didn't do it this way for a limited time.)

Secondly, the camera of NAO Robot sometimes is not possible to clearly and accurately capture the exact information of passengers due to light intensity and limited camera resolution. After our adjustment and optimization this type of problems also occur sometimes.

Thirdly, it's evident that the abundance of sensors and interfaces of NAO make it a desirable development platform for e.g. Walking, Detecting tasks or Working as assistant. However, its weak CPU and limited RAM pose a challenge, requiring the programs running on the robot to be optimized for resource efficiency. With this consideration and the reason of EU Data Nondisclosure Agreement, we were not able to connect to the real Deutsche Bahn system internal network to get real valid real-time information from all the trains. So we tried to write a text library by ourselves so that the information in it can be well stored, and then it can be well called and compared locally.

There still remains a vast room for improvement. Some of the most interesting topics for future work will now presented. The first important improvement should be the self-localization and motion planing if we could update the version of Hardware. Currently our robot is completely unaware of its position in the train environment, but if such information about the position of passengers, obstacles like luggage and seats could be obtained, it could be leveraged to make path planning more effective and precise to provide more efficient and comprehensive services. Secondly, if the NAO Robot could access the Deutsche Bahn system or could make one connection with it, at that time, it is possible to arbitrarily detect the tickets of all trains and provide diversified voice services according to the train itinerary, and it is hoped that some emergency notifications will be broadcast through the connection with the Deutsche Bahn system audio equipment. In general, the integration of NAO robot based Ticket Checking System into the in ICS LAB simulated train environment has the potential to bring about significant improvements to the efficiency and convenience of various processes, such as ticket checking. At the same time, it can also greatly reduce the human costs.

REFERENCES

[1] RailRecipe. (2022) Train Chart Preparation Time All You Need to Know. Retrieved from https://www.railrecipe.com/blog/train-chart-preparation-time.

[2] Smart Ticketing Alliance (STA). (2022) About our work and vison Our founders, our board, our partners. Retrieved from https://www.smart-ticketing.org/about-us.

[3] Face-recognition 1.3.0. (2023) Face Recognition. Retrieved from https://pypi.org/project/face-recognition/.

[4] Tuna, Gurkan & Tuna, Ayse & ahmetoğlu, Emine & Kuscu, Hilmi. (2019). A survey on the use of humanoid robots in primary education: Prospects, research challenges and future research directions. In *Cypriot Journal of Educational Sciences. 14. 361-373. 10.18844/cjes.v14i3.3291.*

[5] NAOqi developer guide. (2021) SOFTBANK ROBOTICS DOCUMENTATION. Retrieved from http://doc.aldebaran.com/2-8/index_dev_guide.html.

[6] Travis E. Oliphant (2015) Guide to NumPy (2nd. ed.). In *CreateSpace Independent Publishing Platform, North Charleston, SC, USA.*

APPENDIX

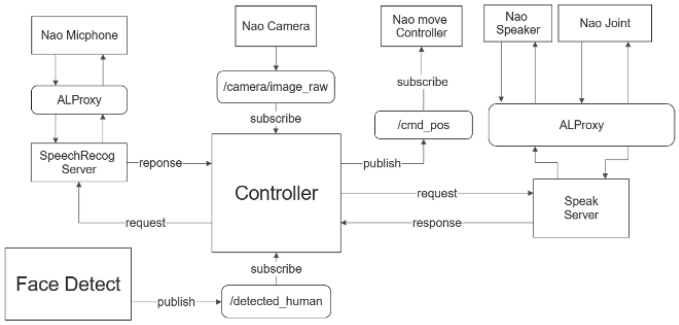

Fig. 15. This is our entire modules communication chart and implemented state machine. It shows how the elements of one part communicate and inform with the other, when to publish information and when to receive information and how to link all system components together.