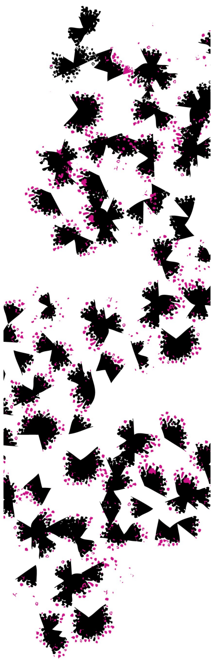
ROBOTICS AND MECHATRONICS



Implementing EyePi architecture on R3D3

P.A.J. (Pascale) van de Ven

BSc Report

Committee:
Dr.ir. E.C. Dertien
Dr.ir. J.F. Broenink
J.M. Linssen, MSc

February 2017

003RAM2017
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

MIRA BIOMEDICAL TECHNOLOGY AND TECHNICAL MEDICINE

CTIT

Contents

1 Introduction

1.1 Concept

The world is changing to a more automatic world with a lot of robots. Robots not only replace people, but also support humans more and more in several aspects like in health care, by receiving people and by giving necessary information. For better understanding between human and robot, human-like (emotional) responses are added (Breazeal, 2001). Human-like communication is needed for a better human-robot experience (Dautenhahn, 2007).

The social and lifelike interaction is combined in the design of the R3D3. R3D3 takes the form of a receptionist robot that can welcome visitors and have conversations with visitors of example at museums, shops or office buildings. R3D3 (Rolling Receptionist Robot with Double Dutch Dialogue) is a lifelike robot combined with an avatar present on a tablet as shown in figure 1.1. R3D3 can recognize visitors and can communicate in natural Dutch speech.

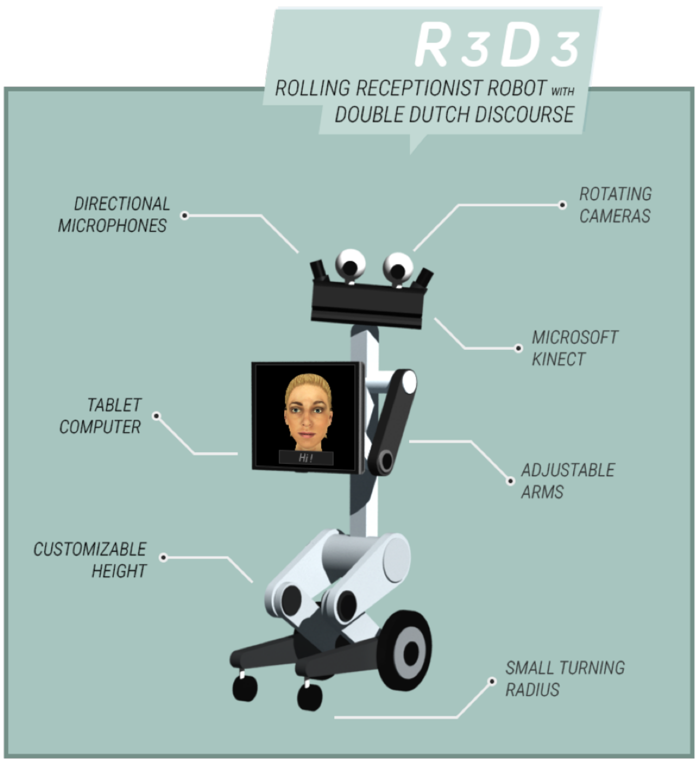


Figure 1.1: Concept of the R3D3 (LEO, 2016)

The development of R3D3 is part of the Dutch national COMMIT/ project. The project is led by the Human Media Interaction (HMI) group. HMI takes care of the natural dialogues in combination with the avatar. RaM is responsible for the actual construction of the robot. The main other partner is the Amsterdam-based company VicarVision that has developed the FaceReader software that uses computer vision to detect emotions from facial expressions of people interacting with the new robot. Other partners include the Dutch Police Academy, NEMO Science Center and iretail.solutions, who provide the use cases. (LEO, 2016)

For a lifelike and social robot with emotions, the EyePi concept (more information can be found in section 2) will be implemented. EyePi is a robot that has a neck with a display for eyes. EyePi can show emotions with his eyes in combination with the use of the neck. The breathing and blinking aspect is implemented, to facilitate the interaction between the user and robot. EyePi is developed at the Robotics and Mechatronics group by van de Vijver (2016) and Oosterkamp (2015). The EyePi architecture will be implemented in R3D3 to have a lifelike interaction with visitors. More information about the EyePi can be found in chapter 2.

1.2 Main goal

The main goal of this project is to develop the software and hardware to control the robot-frame. To achieve this goal a construction is built and the frame is controlled with a combination of software and hardware. To design R3D3 as a receptionist robot, autonomous behavior is needed. The EyePi is a part of this design and the autonomous behavior.

The robot should have a human-like appearance with a head, shoulders, body and legs (with knees). The frame should be height adjustable which will be accomplished with a construction that has (preferably) a visual appearance like human knees. The tablet should be adjustable in height, so arms are suggested, complaint with the human appearance of the system. (den Uyl et al., 2016) Next to the adaptability of R3D3 it should be safely to use in all kind of environments. For example it should be safe for children fingers.

To adjust the frame, software should be implemented to control the whole robot and should be able to communicate with the software of HMI and VicarVision. The existing software of the EyePi should be combined with software so the tablet can move (arms) and the height could be adjusted.

1.3 Approach

In this research the first prototype of the R3D3 will be developed. The frame of R3D3 will be built. The setup of the EyePi can be copied and the existing software of the EyePi will be extended with the new servos and motors. The software of the EyePi will be updated so the autonomous behavior with the camera mounted on the base of the EyePi does not influence the behavior of R3D3. The software will be extended to be able to have all the functionalities R3D3 should have (see chapter 3.3).

1.4 Report outline

This report starts with an analysis of the existing EyePi and ASAP software in chapter 2. After this related work will be analyzed in chapter 3. Which will be followed by the available toolkit. The analysis include the requirements for the R3D3.

In chapter 4 the design of the R3D3 will be discussed. The implementation of the hardware and software will be discussed which result in the final system.

The results will be discussed in chapter 5. In the last chapter, chapter 6, the final conclusion will be given and recommendations will be mentioned.

2 Background

As introduced in chapter 1 the hardware of EyePi can be copied and the software can be an be used as a base for R3D3. In this chapter an overview of the EyePi system is given and the AsapRealizer software of HMI will be explained and how this software is connected to the robot platform.

2.1 EyePi

EyePi has been developed by van de Vijver (2016) and Oosterkamp (2015). The main goal for development of the EyePi was to have a simple design that has a lifelike and fluent interaction which can show emotions. The role that the robot fulfills in social interaction have been transformed from puppet to actor. EyePi is shown in figure 2.1. EyePi can show emotions on the LED matrix display. It has a neck that consists of 3 Dynamixel[1] servo motors which can move independently. In this way the head can pitch, nod and zoom. Combining the emotion on the display with the movement of the neck results in a lifelike robot.

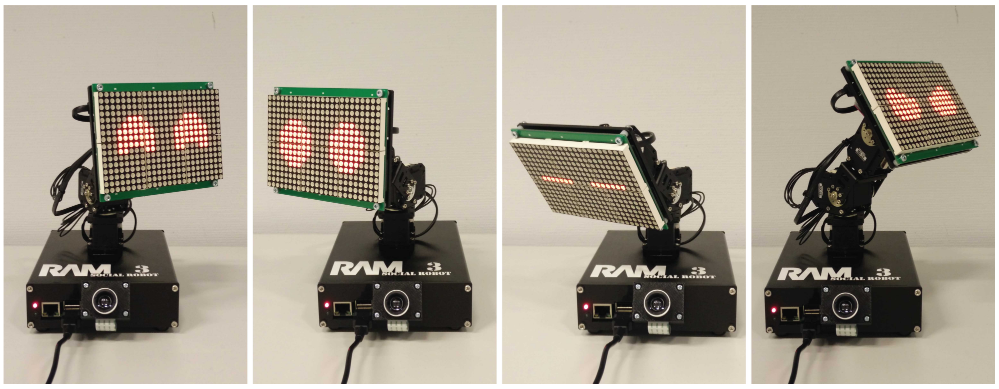


Figure 2.1: EyePi (van de Vijver, 2016)

EyePi has an autonomous behavior performing a breathing motion and blinking action next to the emotion it is showing. The system is designed in a way that the controllable movements (MIDI or external) are mixed with the existing breathing motions and blinking actions. An overview how this is done, is shown in figure 2.2.

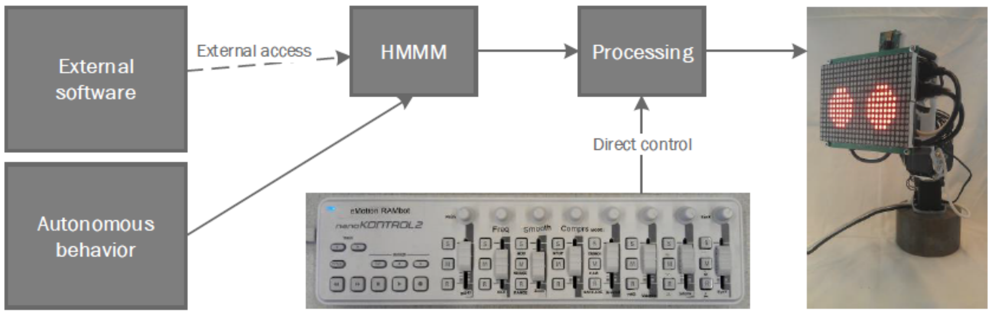


Figure 2.2: EyePi system overview (van de Vijver, 2016)

The EyePi has two modes. The first one is the manual control mode. In this mode the EyePi can be controlled by the sliders and buttons on the MIDI-controller. This controller is directly

[1]http://en.robotis.com/

connected to the Raspberry Pi. The other mode is the autonomous mode. In this mode the autonomous behavior is mixed with the external input. The autonomous behavior depends on the motion detection of the camera. In this mode external software can influence the behavior of EyePi next the input of the Pi Camera. These external inputs will be mixed correctly with the autonomous behavior to be fluent and lifelike. To achieve this, Heterogeneous Multilevel Multimodal Mixing (HMMM) is introduced. In this part the inputs on sequence, gaze direction, emotion and motion parameters are mixed for the most salient point.

The processing of all this is done on a Raspberry Pi, which runs ROS (Robot Operating System). An overview of the software structure can be found in figure 2.3.

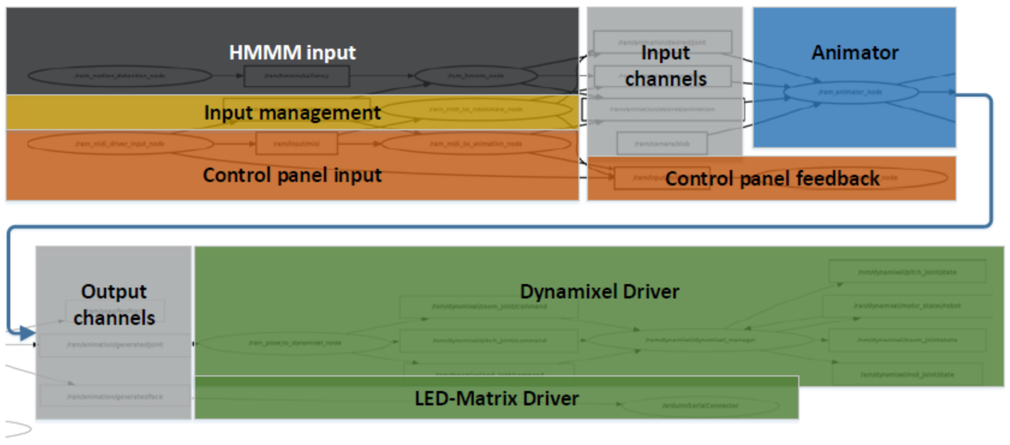


Figure 2.3: EyePi software structure (van de Vijver, 2016)

There are three external input channels which can be used. The inputs are ROS-topics (Saliency, Emotion and Sequence). On these topics external input can be published and this will be mixed in the HMMM component of the EyePi.

2.2 AsapRealizer

From the Human Media Interaction (HMI) research chair, the AsapRealizer part of the ASAP (Articulated Social Agents Platform) software is used. ASAP plans the behavior during interaction. Several inputs and outputs can be connected to the ASAP software. To communicate with the Raspberry Pi a central message broker is used (see figure 2.4). In this case Apollo Broker. The communication is done by using a body containing a XML-message with the correct content.

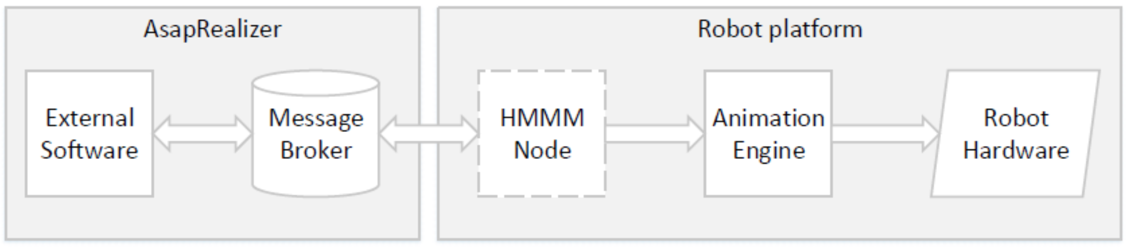


Figure 2.4: Block diagram of external communication (van de Vijver, 2016)

3 Analysis

In this chapter related work to R3D3 will be mentioned. The available toolkit, hardware and software, are discussed. These will result in requirements for the R3D3 project.

3.1 Related work

All kind of robots are created, to help or replace people. In some robots human-like traits are added to make the communication between robot and people better and easier (Fong et al., 2003). Some examples of related work to the R3D3 are described below.

R3D3 combines the appearance of a robot with a virtual human on a tablet. There are several robots developed that use a tablet to show information. An example is Pepper (see figure 3.1a). Pepper is a humanoid robot that recognizes emotions using his 3D camera and voice recognition. These information is processed with an "emotion engine" which allows the robot to respond accordingly. Pepper is 1.2-meter tall that rolls around on a wheeled base and has a face in combination with a tablet to communicate (information) and express emotions. (IEEE, 2014) Another example is Kompaï (see figure 3.1b), which is a robot that can help frail people and caregivers. It can monitor health and use the screen to show useful information for that moment. Using a Kinect camera the surrounding is monitored to be able to help or call for emergency help. (Robotics, 2017)

The FURo-S (see figure 3.1c) gives the same impression as the design of the R3D3 with a face in combination with a tablet. Still there are some differences. The head consists of a screen on which a human face (avatar) is showed. The tablet is used to show information. The height of the FURo-S can not be adjusted. There are some similarities, like the voice recognition and the possibility to talk. The height of the arms (and the tablet) can be controlled. (Robot, 2016)

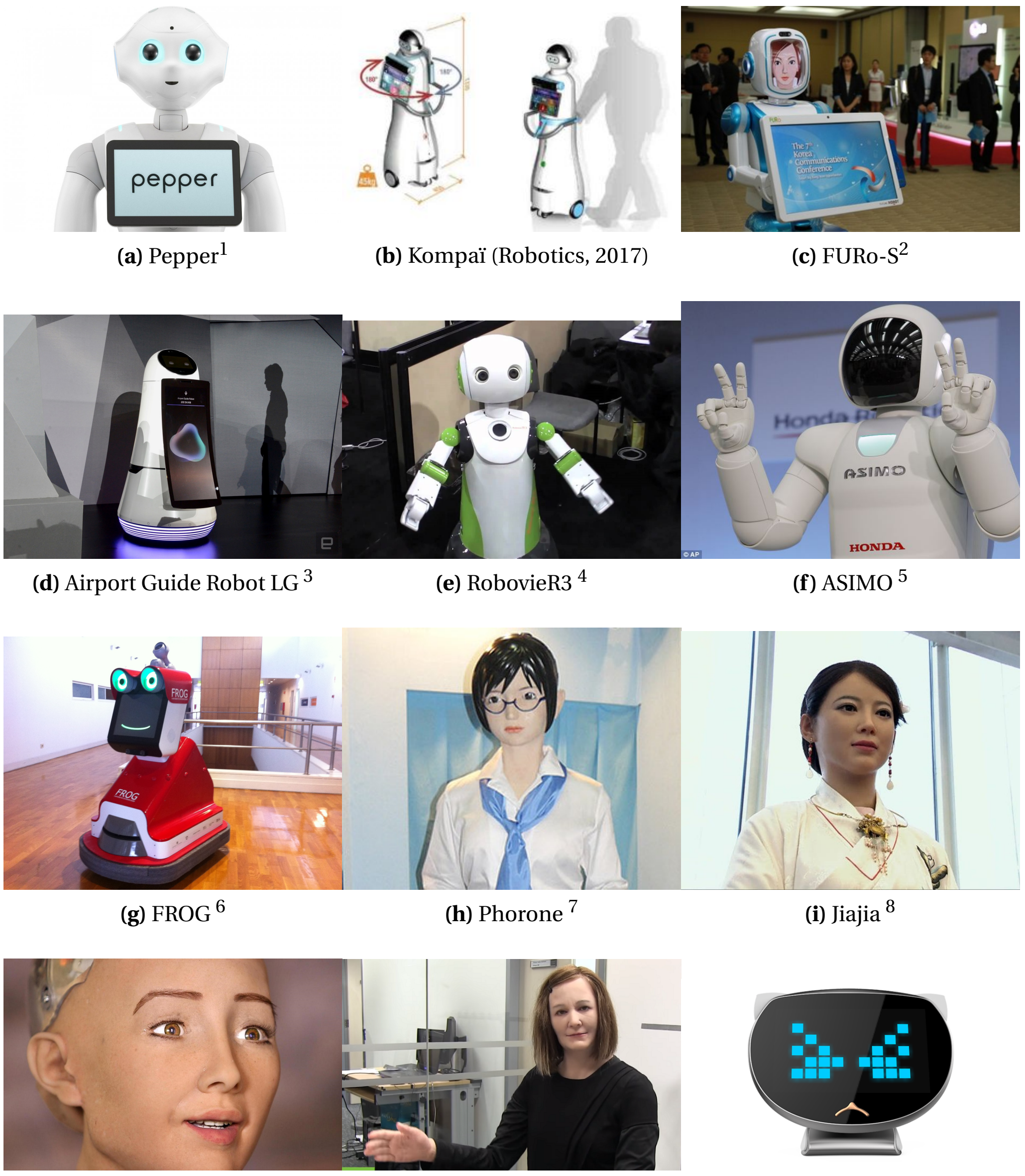
LG developed a robot that can answer questions in combination with information on a screen. This is the Airport Guide Robot (see figure 3.1d). It has the possibility to wear the screen as a backpack or on the front side, to show clearly information to the person in front of the robot. An addition is the scanner for airline tickets. With the information of the ticket the Airport Guide Robot can tell everything about the boarding time, gate location and the weather forecast of the destination of the traveler. (Sijbrands, 2017)

The Robovie R3 (see figure 3.1e) is another humanoid robot that is developed to assist people. It stands about $108cm$ tall, can see with 2 cameras (as eyes) and has 2 microphones and a speaker to communicate. (Beciri, 2010) It can carry for example your shoppings but can also give information about nearby products and can guide people (even Robovie can hold hands). (Savov, 2010). Honda designed the ASIMO (see figure 3.1f), a humanoid robot and is designed to assist people in their daily lives. ASIMO walks and has arm and hands to be able to grep stuff. (Honda, 2017)

At the University of Twente at the Human Media Interaction group FROG (see figure 3.1g) is developed. This is developed as an outdoor guide robot. FROG makes use of a screen and does have eyes to be more friendly. The robot can talk about the sights during the guide tour. (Frogrobot, 2016)

Like R3D3 there are a lot of receptionist robots developed and robots that have human-like traits. The exact implementation of these robots are totally different per use. An example is Phorone (see figure 3.1h), a personal robotic secretary. She is lifesized and can deliver human-like expressions and can even talk. (Bayas, 2008) Other human-like robots are Jiajia (see figure 3.1i), Sophia (see figure 3.1j) and Nadine (see figure 3.1k). These robots are designed to look like a human. The face has a 'skin' with has multiple degrees of freedom to show expressions and emotions.

EyePi is designed to show emotions and has a neck, to act human-like. More robots use screens to display eyes and use these eyes to show emotions, like the XIBOT (see figure 3.1l). The XIBOT is an screen that can display eyes but takes also the form of a life assistant and can be used for video chat. (Huang, 2017)



(a) Pepper[1] **(b)** Kompaï (Robotics, 2017) **(c)** FURo-S[2]

(d) Airport Guide Robot LG [3] **(e)** RobovieR3 [4] **(f)** ASIMO [5]

(g) FROG [6] **(h)** Phorone [7] **(i)** Jiajia [8]

(j) Sophia [9] **(k)** Nadine [10] **(l)** XIBOT [11]

Figure 3.1: Different designs of robots for social interaction

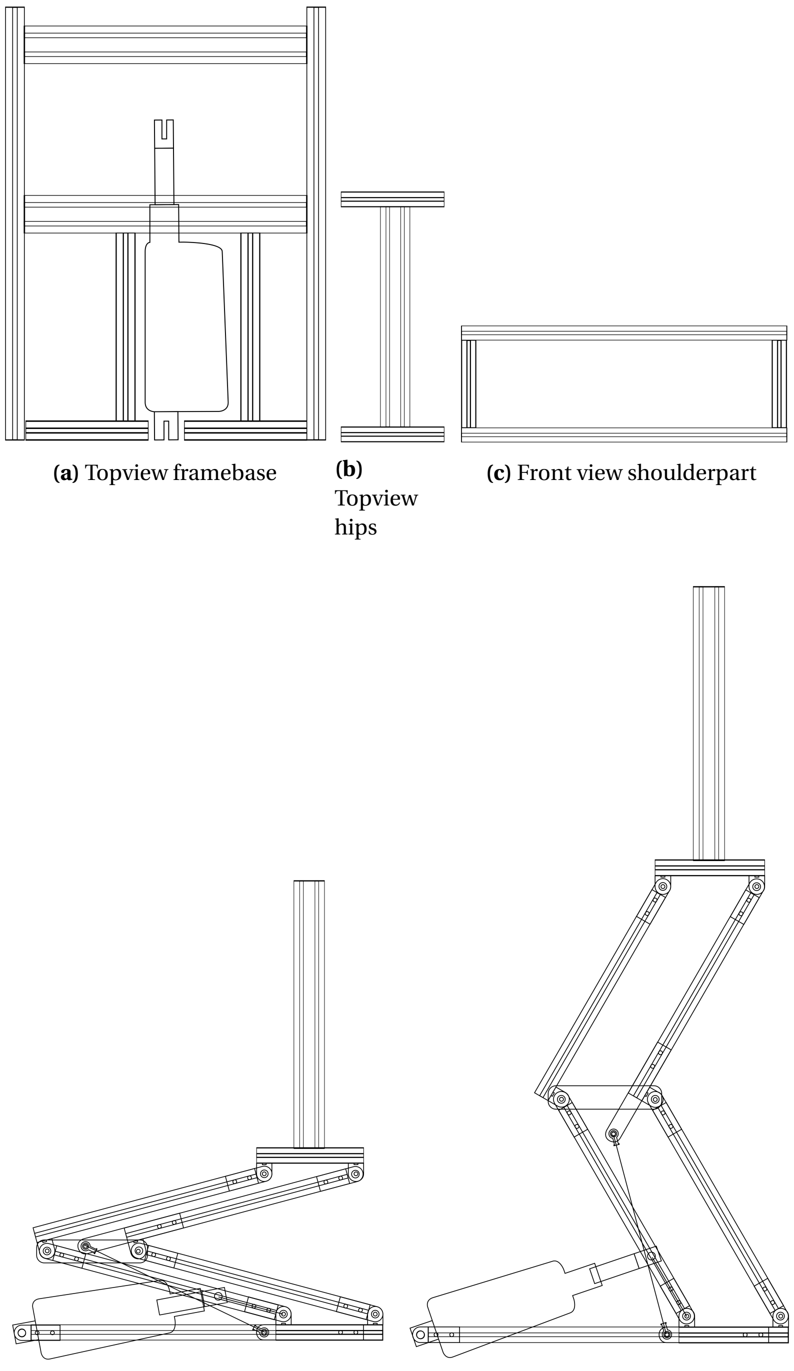
The selection, given in figure 3.1, is a selection of already developed robots. These robots have something in common with the design of R3D3 like the tablet-face combination or the form of a receptionist. There are more robots that have the possibility to communicate with people or help them, like the x.AI, Amigo etc.

3.2 Available toolkit

3.2.1 Design frame

At the start of the R3D3 project a drawing for the frame of the first prototype of R3D3 had already been made. This drawing was based on the concept of R3D3 as shown in figure 1.1

and can be found in figure 3.2. The head will be mounted on top of the shoulder part, the Kinect between the top and bottom parts and one Dynamixel servo motor on each side of the shoulder part.


(a) Topview framebase **(b)** Topview hips **(c)** Front view shoulderpart

(d) Sideview frame low **(e)** Sideview frame high

Figure 3.2: Technical drawing of R3D3 frame

As can be seen from the technical drawing (figure 3.2) the frame does not have a base to move with. For the first prototype of R3D3 a wheelchair base (from an earlier project, named Degenkrab (Dertien, 2007)) is used (see figure 3.3). This base is modified in such a way that it can be remote-controlled or the joystick on the base itself can be used.

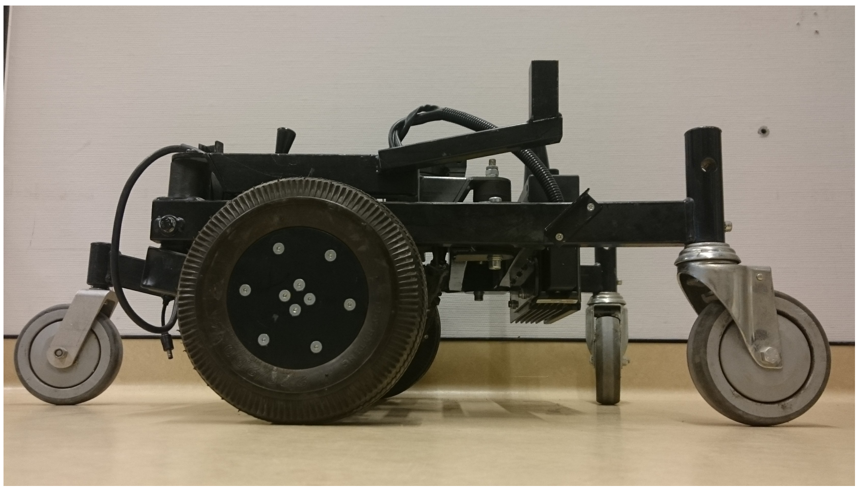


Figure 3.3: Wheelchair base

For the movement of the knees an actuator is available, namely the LINAK LA28. This actuator can deliver a maximum force of $3500N$, which is sufficient for this setup. The maximum speed is $46mm/s$ when $24V$ is applied. The maximum stroke length is $60mm$.

3.2.2 EyePi

The EyePi uses ROS (Robot Operating System) on a Raspberry Pi for the processing. The overview of the ROS-topics and ROS-nodes can be found in figure 2.3. The existing software can be used as a basis for the software of R3D3. There are some parts that do not need to be changed like the face animation part. The number of joints has to be extended to accommodate two arms (see figure 3.4 and figure 3.5). As can be seen (and described in chapter 2) there are 3 Dynamixel servo motors used for the EyePi (pitch_joint, zoom_joint and nod_joint).

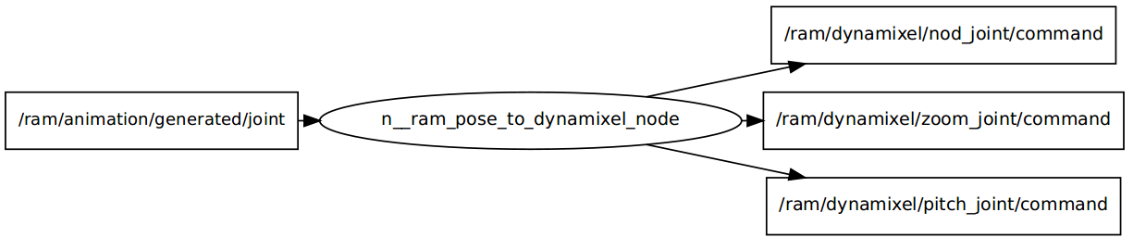

/ram/animation/generated/joint
n__ram_pose_to_dynamixel_node
/ram/dynamixel/nod_joint/command
/ram/dynamixel/zoom_joint/command
/ram/dynamixel/pitch_joint/command

Figure 3.4: Rosnode ram_pose_to_dynamixel_node with topics of EyePi software

The height control will be implemented in the existing software. The actuator is controlled using a micro controller board (Arduino) which uses its internal ADC to sense the current (through the actuator). This Arduino board requires one additional serial port next to the existing one that is used for the dot matrix display of the face.

To be able to use R3D3 in both modes, the midi controller will be updated with the new controllable parts of the robot. This can be done in the ram_midi_to_animation_node (see figure 3.6).

The standard mode during initializing is the manual mode for EyePi. In this case the MIDI panel can be used as a controller. R3D3 should be able to work without the use of a MIDI-controller. Due to the fact that R3D3 has to listen to the external software (act autonomously) the initialized mode will be changed to the autonomous mode.

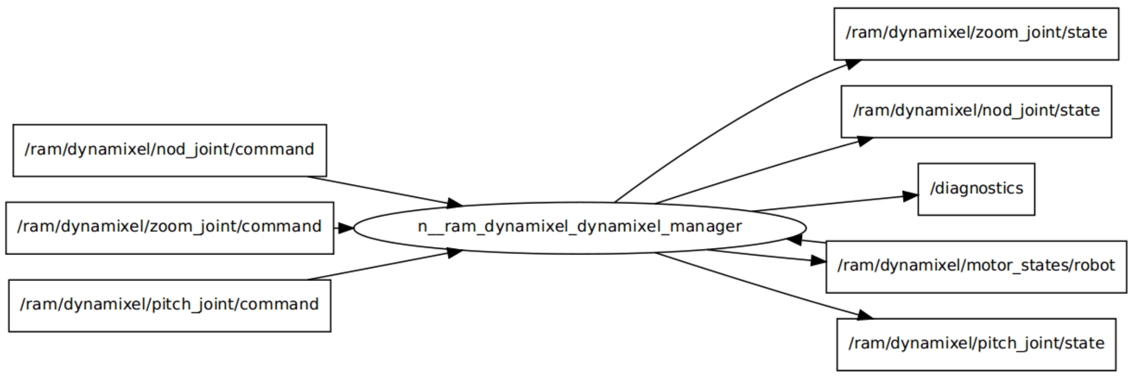

/ram/dynamixel/nod_joint/command
/ram/dynamixel/zoom_joint/command
/ram/dynamixel/pitch_joint/command
n__ram_dynamixel_dynamixel_manager
/ram/dynamixel/zoom_joint/state
/ram/dynamixel/nod_joint/state
/diagnostics
/ram/dynamixel/motor_states/robot
/ram/dynamixel/pitch_joint/state

Figure 3.5: Rosnode ram_dynamixel_dynamixel_manager with topics of EyePi software

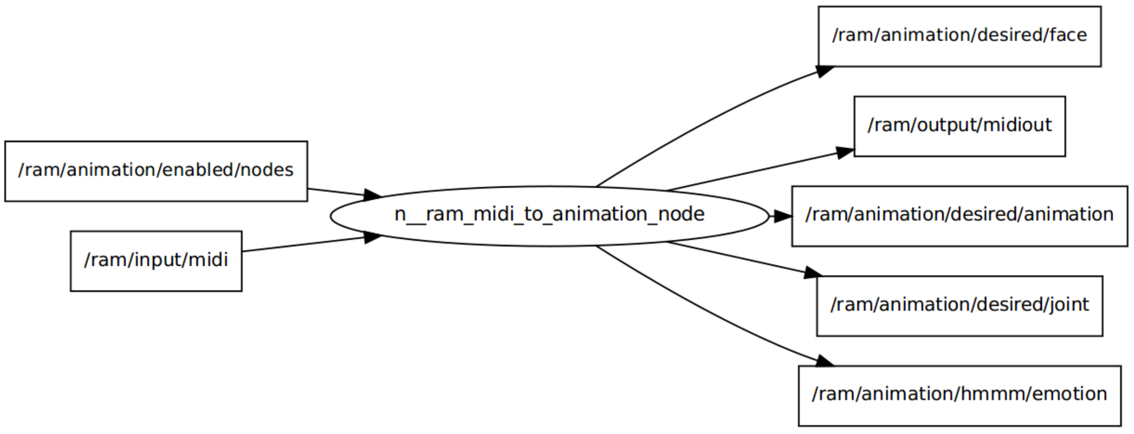

/ram/animation/enabled/nodes
/ram/input/midi
n__ram_midi_to_animation_node
/ram/animation/desired/face
/ram/output/midiout
/ram/animation/desired/animation
/ram/animation/desired/joint
/ram/animation/hmmm/emotion

Figure 3.6: Rosnode ram_midi_to_animation_node with topics of EyePi software

To be able to be open for control from the outside the existing topics will be extended with new once for the arm and height control. The current ROS-topics on the EyePi are:

- /ram/animation/hmmm/saliency
- /ram/animation/hmmm/emotion
- /ram/animation/hmmm/sequence

Next to the nodes and topics that has to be added, the motion detection part of the EyePi will be disabled, due to the fact that not a Pi Camera is used but a Kinect. This Kinect will not be connected directly to the Raspberry Pi, but via a message broker (Apollo Broker).

3.3 Requirements

Using the goal and the analysis stated before, the following requirements per group are listed.

3.3.1 Hardware

- The look of the robot should invite users to interact with it. (Linssen et al., 2016)
- R3D3 should be light, flexible and yielding, perceptive and sensitive, and not too quick in behavior, in order to be inherently safe in environments with unpredictable behavior from visitors, including children. (den Uyl et al., 2016)

- The sizes are roughly humanoid as well, the fully extended system reaches 180 cm in order to allow interaction with adults of average size. For interaction with children it should be able to move down (crouch) to 120 cm. (den Uyl et al., 2016)
- The system should operate primarily on flat floors. (den Uyl et al., 2016)
- The robot body should follow the mimetic constraint, i.e. be as human-like as possible in appearance. (den Uyl et al., 2016)
- The head should be placed on a natural location, including neck/shoulders. (den Uyl et al., 2016)
- Here a single semi-circular lightweight angular activator could be used for tilting the tablet between say -30 and +30 degrees around vertical position. (den Uyl et al., 2016)
- Main function of the head is to attract and direct attention, to be the first focus of attention upon meeting the robot and shift (or actively direct) this focus to the Discourse system discussed later. (den Uyl et al., 2016)
- The electronic hardware should be reliable.

3.3.2 Software

- ROS should support a dialogue agent which governs the behaviour of both the robot and the virtual human. (den Uyl et al., 2016)
- External software must be able to control the robot.
- The software should be reliable.
- The software of the EyePi must be implemented correctly for the same behavior.
- The resulting behavior should be fluent.

4 Implementations

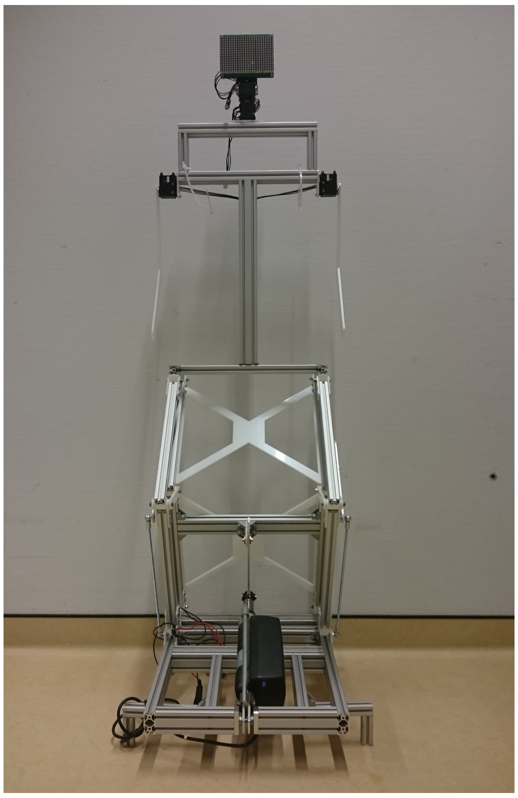
This chapter focus on the implementation of the hardware and software of R3D3. The changes that had to be made in the software of the EyePi are discussed and an overview of the total system with the external software is given.

4.1 Hardware

The hardware of R3D3 consists of a frame made with standard aluminum extruded rods (by BOIKON[1]) and the electronics needed to control the frame. The setup of the EyePi is copied and mounted on top of the frame.

4.1.1 Frame

The frame of R3D3 is build as stated in section 3.2.1. The shoulder part (see figure 3.2c) is mounted on top op the body. On top of this schoulder part the EyePi is mounted and On each side a Dynamixel servo motor is added for the arms. The result can be seen in figure 4.1. The frame is build from BOIKON. This material is easy to adjust and new parts can easily be added and mounted.


(a) Frame frontview

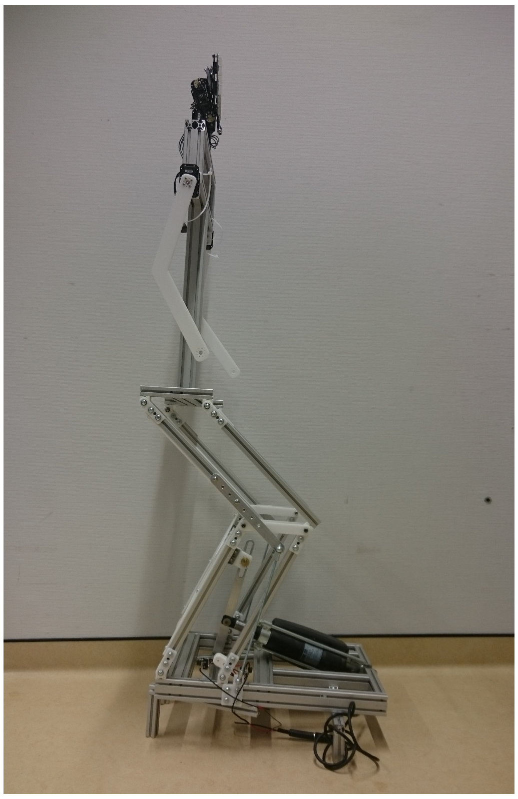

(b) Frame sideview

Figure 4.1: Overview of frame R3D3

For the height adjustment a LINAK LA28 actuator is used. This actuator has a small stroke (60mm) and does not have build-in end stops. To enlarge the arm a lever is added which is made of aluminum to be able to lift the whole frame. Due to the fact that the actuator consists

[1]https://www.boikon.nl/

of a worm wheel, the axis will still rotate if the actuator is at the begin or end of the stroke. A sturdy lever is needed to prevent that the lever breaks in case the actuator still turn at the begin or end of the stroke. The lever is mounted at the lower part of the legs.

To keep the body horizontal a construction of two parallelograms is used. Due to the fact that the actuator is mounted at the lower parallelogram, the upper part of the frame is mounted with a linkage system to the lower part of the frame. This part is not parallel to the existing leg structure to have the possibility use the knees in the frame. At the backside of the legs two crosses of acrylic are added for more stability in the legs. With this construction the legs always move parallel to each other.

4.1.2 Electronics

The height control of the frame is done with the Raspberry Pi in combination with an Arduino Uno that is connected to a H-Bridge. In this case the VNH2SP30 from ST Microelectronics is used on a board designed by M. Schwirtz (RaM,2014). The actuator is used at $12V$ with a duty cycle of 100%. Although the actuator is primarily used at 100% duty cycle, using PWM it is possible to change the speed and have a suitable ramp-up/ramp-down of the motor's velocity. Due to the fact that the actuator does not have build-in end stops, the micro switches are used in combination with diodes to interrupt the circuit. These are hardware switches, which are also connected to the Arduino Uno to be able to use it in the code of the height adjustment. The overview of this system can be found in figure 4.2. For each connection with the micro switch a voltage divider is used to go from $12V$ to less than $5V$ to be able to use the Arduino for detection. The value for R1 is equal to 390Ω and for R2 is 820Ω.

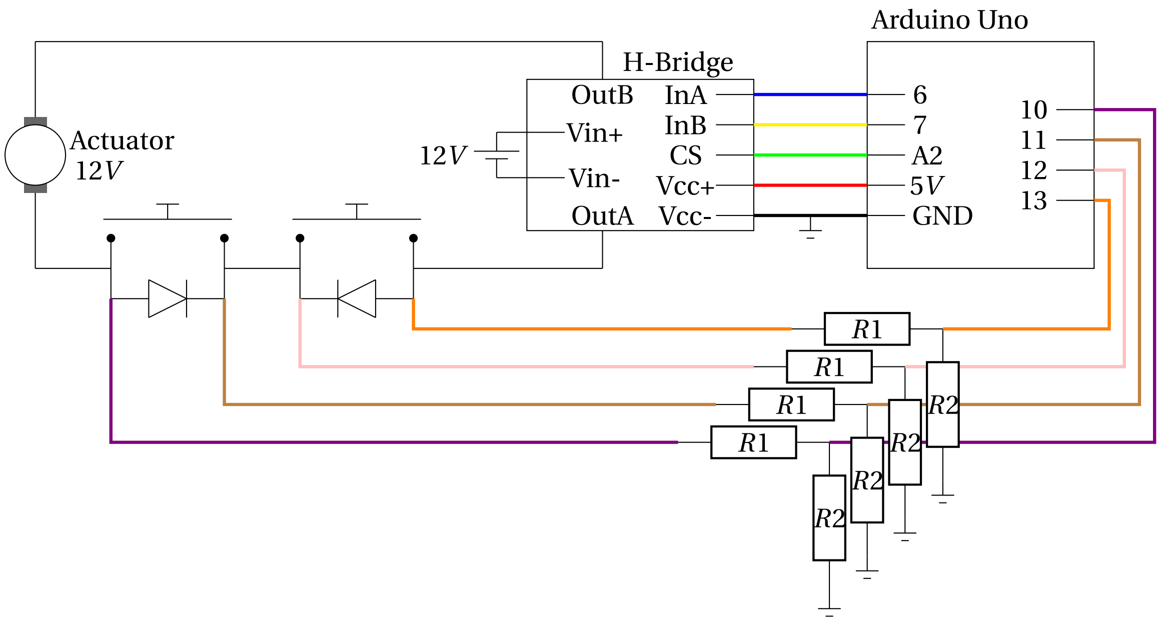


Figure 4.2: Electronics used for the height control

The motors of the wheelchair base run on $24V$. To have the freedom to move without cables, two Lead Crystal Batteries of $12V$ ($10Ah$) are connected in series. This results in $24V$ with $10Ah$, which means that the wheelchair base can be used without adjustments.

To be able to rotate the arms of R3D3 2 Dynamixel MX-28 [2] are used. There are added to the existing Dynamixel bus. In this way the Dynamixels can be controlled from the Raspberry Pi in combination with a USB - RS-485 converter.

[2]http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28.htm

4.2 Software

The software of R3D3 consists of the ROS implementation on the Raspberry Pi and the code on the two Arduino boards. These Arduino boards have a serial connection to the Raspberry Pi. The software on the Arduino Micro that is used to control the LED matrix display has not changed.

4.2.1 Raspberry Pi

As described in section 3.2.2 the software of the EyePi can be used as a base for R3D3. New code is added as will be explained further in this section. Some changes on the existing EyePi code are made. The motion detection is disabled because the Pi Camera is not used. Without disabling the motion detection, the ram_hmmm_node node does not function in a right way in the autonomous mode. The mixing in the HMMM part needs a desired saliency point to mix with the external input. Without a set point for the saliency there is no possibility to control R3D3 from external software. This motion detection part is disabled using a ROS parameter in the launch file of the R3D3. In case the motion detection is set to false the saliency point is set to the middle of the display and no input needed from the Pi Camera.

EyePi does boot in manual boot, but R3D3 should act autonomously and is connected to external software. The initialized mode is changes using a global parameter in the launch file. This parameter (node_enabled) is used in the file ram_midi_driver_input_node for the correct initializing. Even when there is no MIDI-controller connected to the Raspberry Pi the initialized mode is the autonomous mode.

The Dynamixel configuration files are extended for the two shoulder joints. To be able to control these joints the existing Joint_State ROS message should contain two more variables for both shoulder. There are two possibilities to control the arms. The first one is in manual control mode with the MIDI-controller in the ram_hmmm_node (see figure 4.3) which gives a desired joint. One slider is used to control both arms. The other possibility to control the arms is in autonomous mode. To handle the information, an extra node is added to ROS as can be seen in figure 4.4 and a desired joint is published. The message contains a value between 0 and $\frac{1}{2}$ which correspond to a value (given in radians) between 0 and $\frac{\pi}{2}$. Chosen is to link the two shoulder Dynamixels in such a way that they always have the same position. This is to prevent the tablet from harm (torsional force) which might occur if the arms do not follow a mirror path. In the ram_animator_node (see figure 4.5) the two joints get the same value for the position and the position of the joints is generated. Before this node the arms are treaded as one joint.

This extra information from the extra Dynamixels result in more Dynamixel outputs of the ram_pose_to_dynamixel_node which can be found in figure 4.6. These outputs are managed by the Dynamixel Manager to create the states for all joints.

For the height adjustment an extra node is added to ROS (see figure 4.7). The software for the height adjustment of the Raspberry Pi only receives the message from the MIDI-controller or from the external software and publish this information on the ROS topic /ram/animation/-generated/height. The Arduino Uno uses this information for the actual adjustment as will be described in section 4.2.2. From the MIDI-controller or from the external software an integer between 0 and 2 is received which corresponds to a specific height (as will be explained in section 4.2.2).

The total overview of ROS on the Raspberry Pi can be found in figure 4.8. In this overview the link with the two Arduino boards can be seen. The ArduinoUnoSerial is the connection with the Arduino Uno that is used for the height adjustment and the ArduinoSerialConnector is used for the face of EyePi.

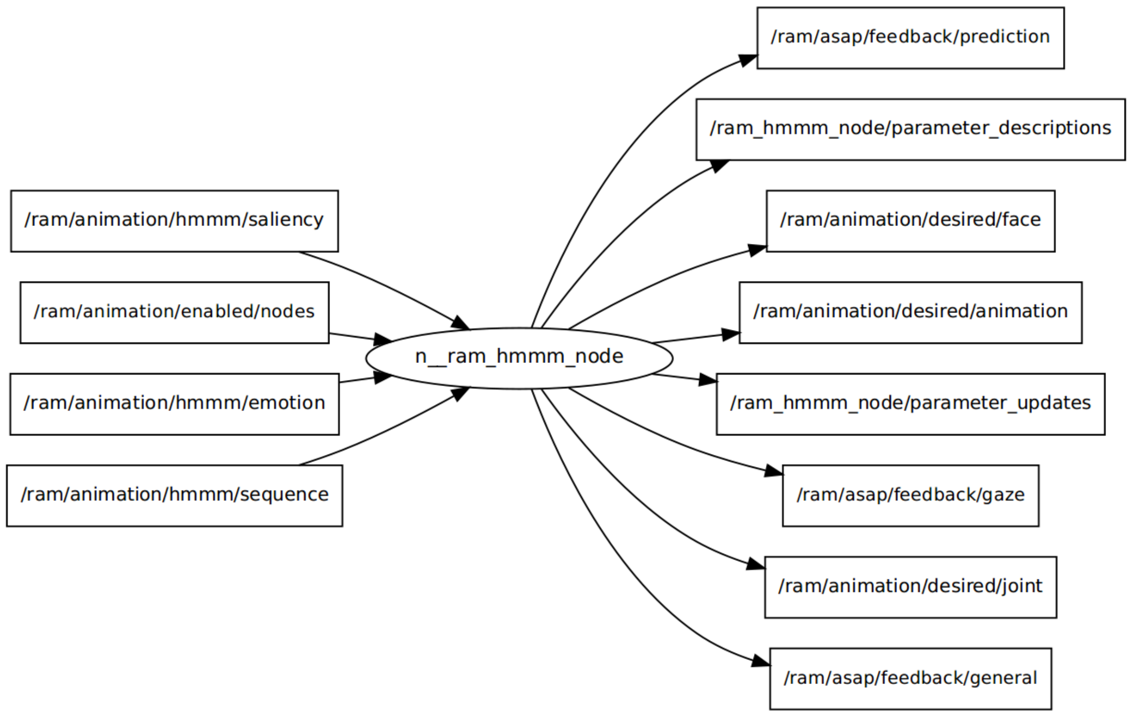

/ram/animation/hmmm/saliency
/ram/animation/enabled/nodes
/ram/animation/hmmm/emotion
/ram/animation/hmmm/sequence
n__ram_hmmm_node
/ram/asap/feedback/prediction
/ram_hmmm_node/parameter_descriptions
/ram/animation/desired/face
/ram/animation/desired/animation
/ram_hmmm_node/parameter_updates
/ram/asap/feedback/gaze
/ram/animation/desired/joint
/ram/asap/feedback/general

Figure 4.3: Rosnode ram_hmmm_node with topics of R3D3 software

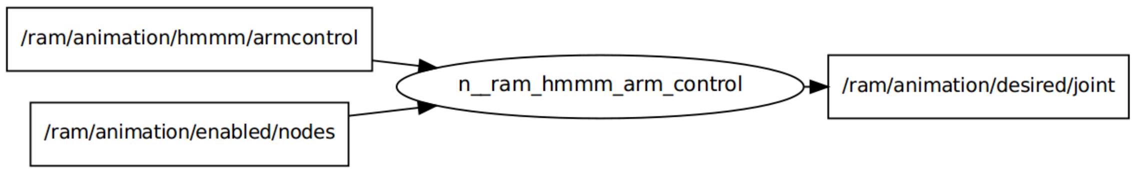

/ram/animation/hmmm/armcontrol
/ram/animation/enabled/nodes
n__ram_hmmm_arm_control
/ram/animation/desired/joint

Figure 4.4: Rosnode ram_hmmm_arm_control with topics of R3D3 software

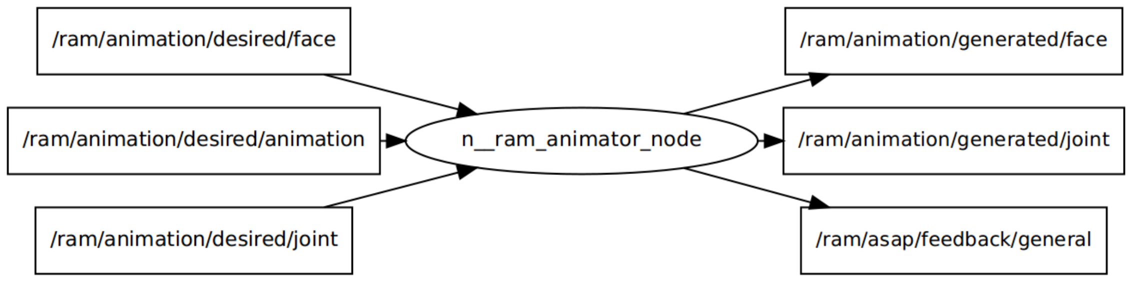

/ram/animation/desired/face
/ram/animation/desired/animation
/ram/animation/desired/joint
n__ram_animator_node
/ram/animation/generated/face
/ram/animation/generated/joint
/ram/asap/feedback/general

Figure 4.5: Rosnode ram_animator_node with topics of R3D3 software

4.2.2 Arduino

The Arduino is connected via ROS Serial to the Raspberry Pi and subscribes to the ROS topic /ram/animation/generated/height (see figure 4.7). On this topic the desired height, given in a certain position (0 to 4), is published. The Arduino uses this information to go the correct height. The end stops are used to define the lowest state (0) and the highest state (4). From there on the other state are set by time. The correct direction of the actuator is set by the InA

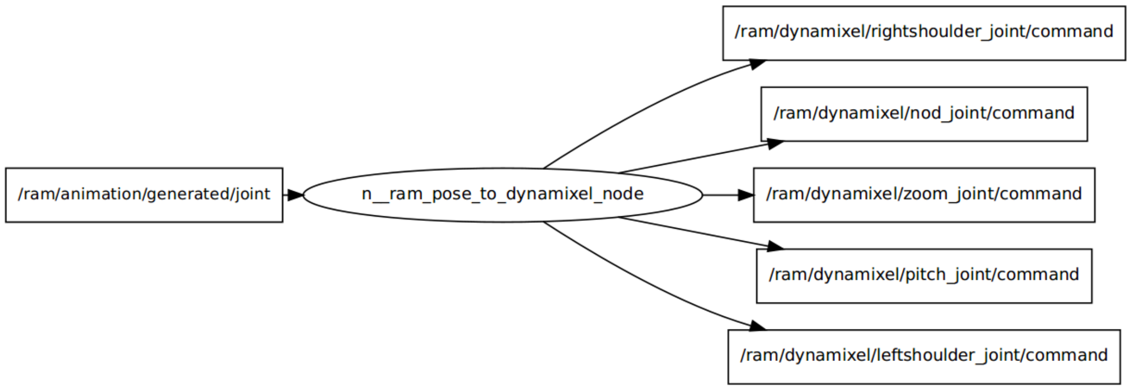

/ram/animation/generated/joint
n__ram_pose_to_dynamixel_node
/ram/dynamixel/rightshoulder_joint/command
/ram/dynamixel/nod_joint/command
/ram/dynamixel/zoom_joint/command
/ram/dynamixel/pitch_joint/command
/ram/dynamixel/leftshoulder_joint/command

Figure 4.6: Rosnode ram_pose_to_dynamixel_node with topics of R3D3 software

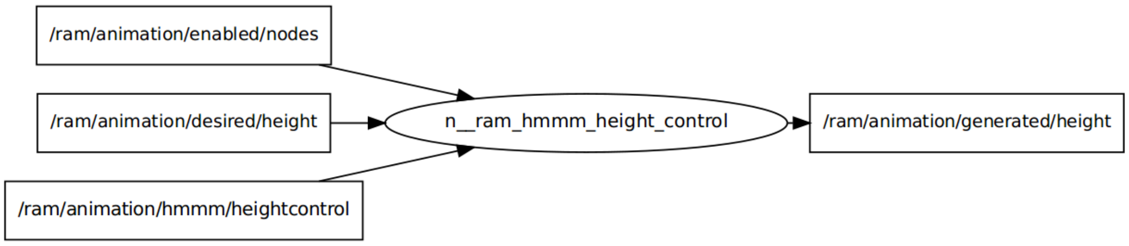

/ram/animation/enabled/nodes
/ram/animation/desired/height
/ram/animation/hmmm/heightcontrol
n__ram_hmmm_height_control
/ram/animation/generated/height

Figure 4.7: Rosnode ram_hmmm_height_control with topics of R3D3 software

and InB inputs of the H-Bridge. To be sure that not too much current is going through the motor the current sense of the H-Bridge is used. If the motor is at one of it ends the current sense will be high and the motor is powered off. This also works if the motor locks. This current sense is an extra protection to be sure the frame of R3D3 will not brake down if the motor reaches one of the ends.

4.3 Final system

In the final system the existing software and the new software functionalities are combined. R3D3 has been constructed using the existing EyePi as the face and a Kinect is mounted on the shoulder part for the FaceReader software of VicarVision. The communication with the external software will be implemented. An overview of the total system including the hardware and the software in combination with the external software from HMI and VicarVision can be found in figure 4.9.

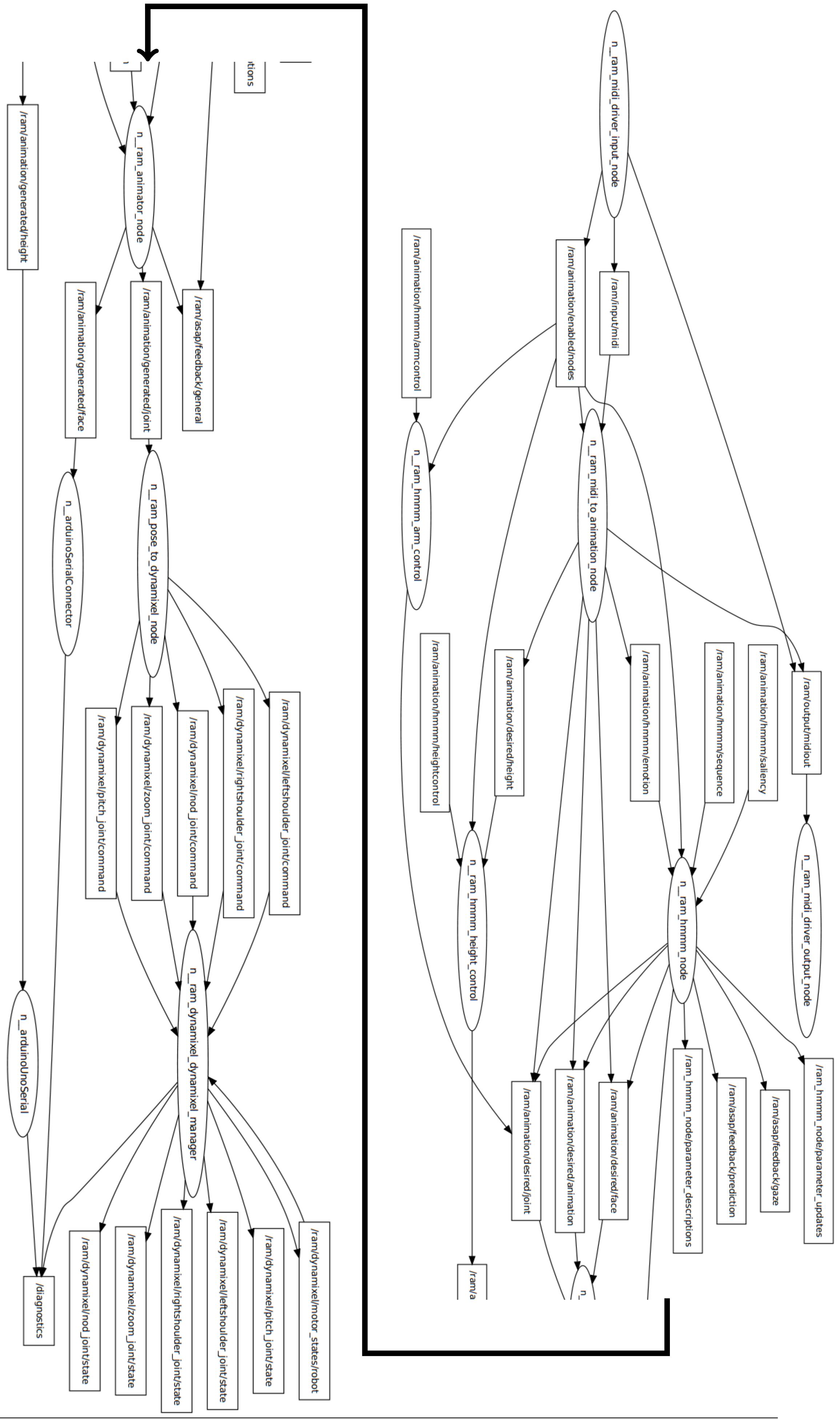

n_ram_midi_driver_input_node
/ram/animation/enabled/nodes
/ram/input/midi
/ram/animation/hmmm/armcontrol
n_ram_hmmm_arm_control
n_ram_midi_to_animation_node
/ram/output/midiout
n_ram_midi_driver_output_node
/ram/animation/hmmm/heightcontrol
/ram/animation/desired/height
/ram/animation/hmmm/emotion
/ram/animation/hmmm/sequence
/ram/animation/hmmm/saliency
n_ram_hmmm_height_control
n_ram_hmmm_node
/ram/animation/desired/joint
/ram/animation/desired/animation
/ram/animation/desired/face
/ram_hmmm_node/parameter_descriptions
/ram/asap/feedback/prediction
/ram/asap/feedback/gaze
/ram_hmmm_node/parameter_updates
/ram/a
n_
tions
n_ram_animator_node
/ram/animation/generated/height
/ram/animation/generated/face
/ram/animation/generated/joint
/ram/asap/feedback/general
n_arduinoSerialConnector
n_ram_pose_to_dynamixel_node
/ram/dynamixel/pitch_joint/command
/ram/dynamixel/zoom_joint/command
/ram/dynamixel/nod_joint/command
/ram/dynamixel/rightshoulder_joint/command
/ram/dynamixel/leftshoulder_joint/command
n_ram_dynamixel_dynamixel_manager
n_arduinoUnoSerial
/diagnostics
/ram/dynamixel/nod_joint/state
/ram/dynamixel/zoom_joint/state
/ram/dynamixel/rightshoulder_joint/state
/ram/dynamixel/leftshoulder_joint/state
/ram/dynamixel/pitch_joint/state
/ram/dynamixel/motor_states/robot

Figure 4.8: R3D3 software structure

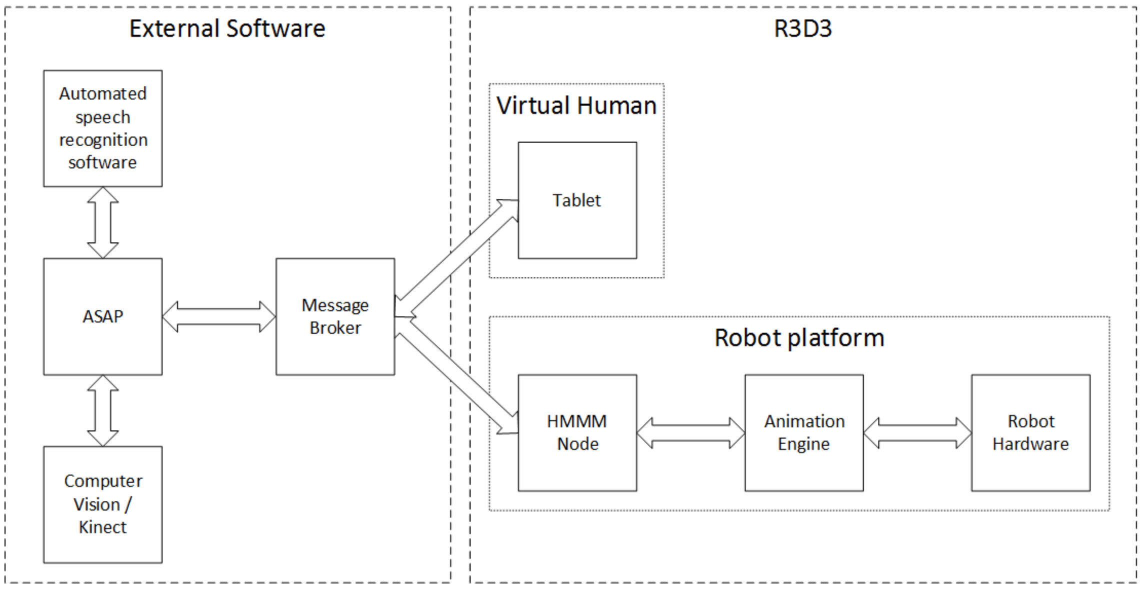


Figure 4.9: Block diagram overview of total R3D3 system

5 Results

5.1 Performance

Figure 5.1 shows the current design of R3D3 including the legs, knee mechanism, body, shoulder base (with Kinect), arms and head. The weight of the frame equals $11.5kg$ and the weight of the wheel chair base equals $38.4kg$, including the batteries. This results in a total weight of $49.9kg$. The shoulder part is relatively big in compare to the rest of the body. This is necessary to mount the Kinect and the EyePi, but does not follow the mimetic constraint.

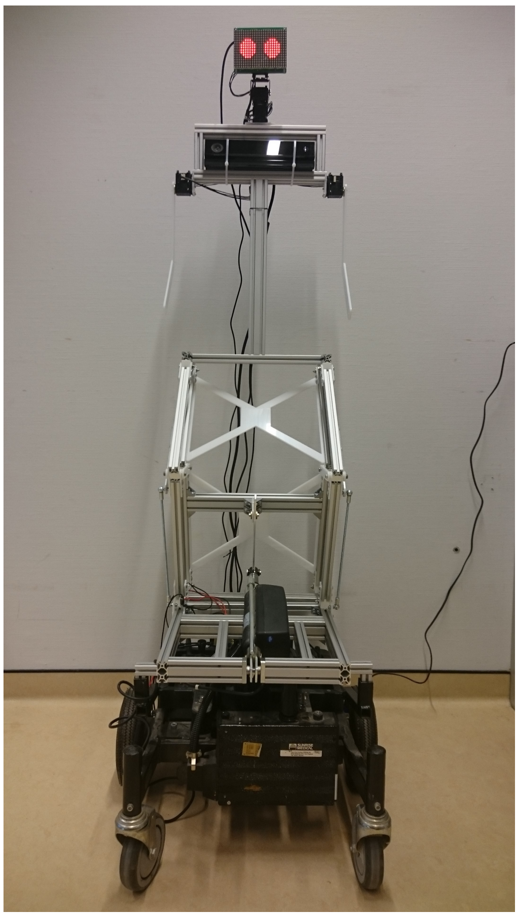
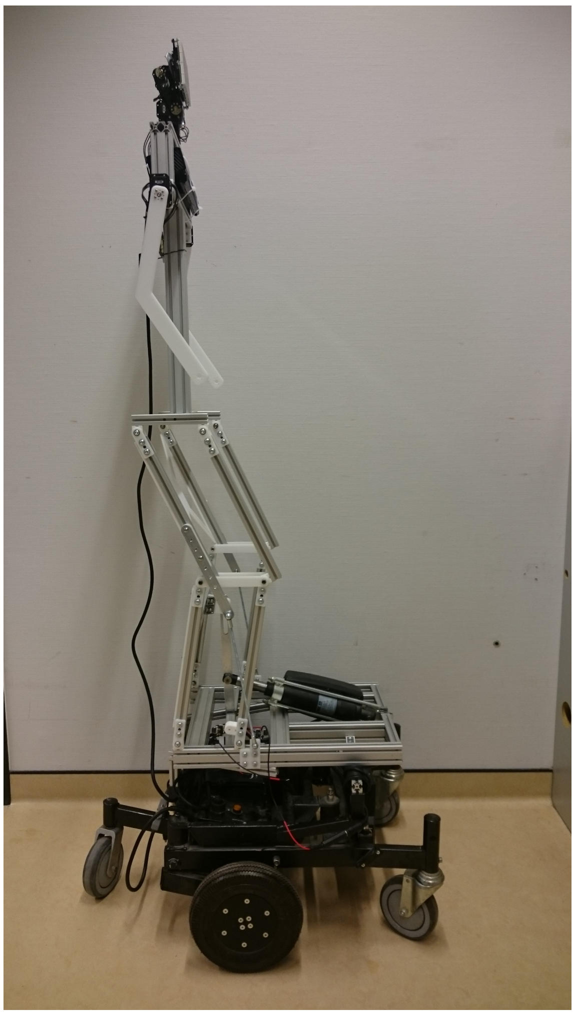


Figure 5.1: R3D3

The height adjustment is done with the actuator that is mounted on the lower legs by a lever. The knees connect the upper legs with the lower legs. With the current prototype of R3D3 a maximum height of $171cm$ and a minimum height of $152cm$ is reached. The difference between the heights is showed in figure 5.2. The height of the frame is measured compared to the stroke length of the actuator, which can be found in figure 5.3. The stroke length is linear compared to the height of the frame. The velocity of the actuator during an up- or down motion can be seen in figure 5.4 and appears to be linear. As can be seen, going down takes less time compared to going up. This can be explained by the fact that going up means pressing the weight of the frame against the gravity.

The control of the height is done using the Arduino. This Arduino is connected to a H-Bridge to turn the motor in the right way. The implemented code on the Arduino subscribes to the ROS-topic ram_animation_generated_height where a value between 0 and 2 is published. These


(a) Highest point (171 cm)

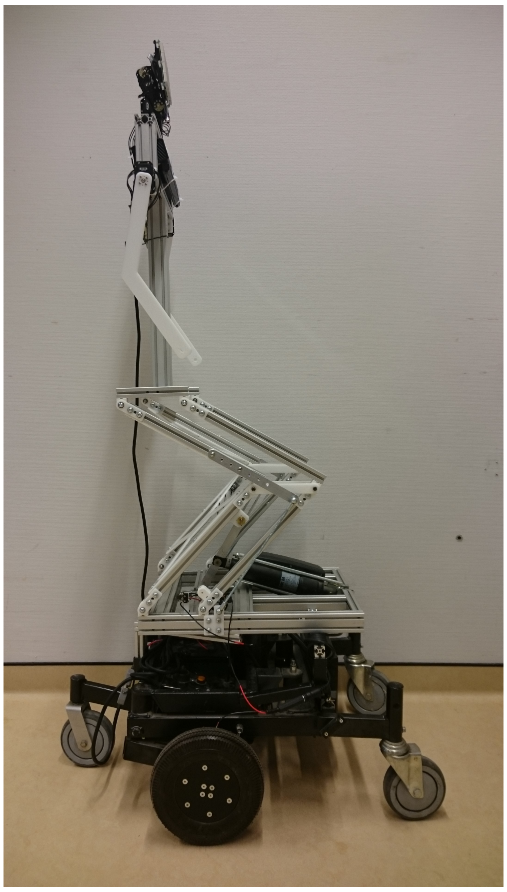

(b) Lowest point (152 cm)

Figure 5.2: Difference in height of R3D3

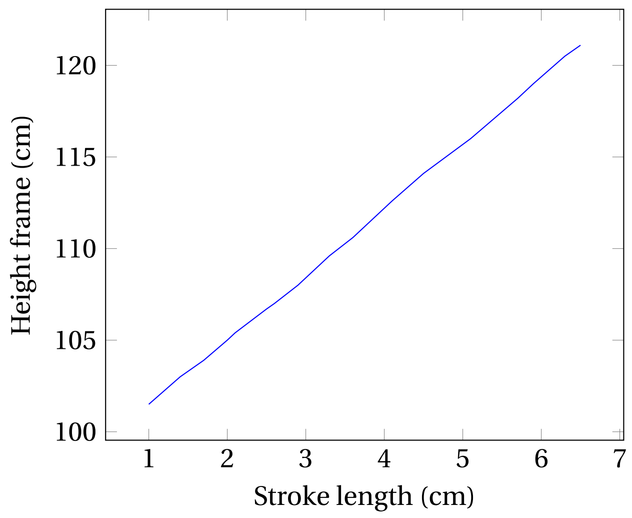


Figure 5.3: Height of frame compare to stroke length

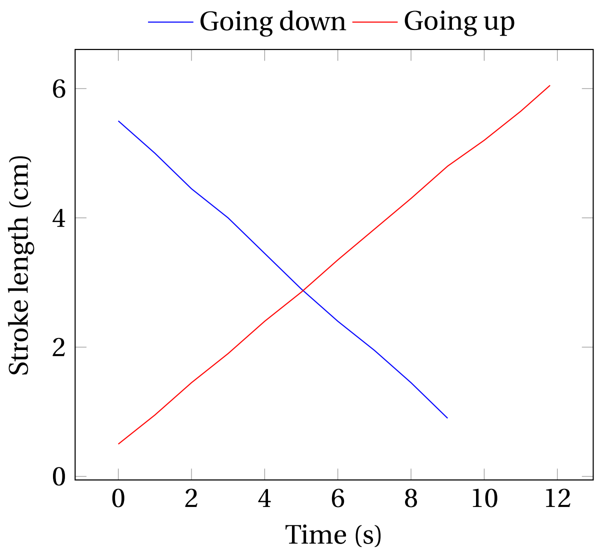


Figure 5.4: Speed of actuator

values correspond to down, stop and up respectively. These operations give no problem and go fluent. The current sense is used as an extra protection to be sure the frame will not break. The option to set a number of positions (0 to 4) did not work out due to the fact that the delay in the Arduino will be to big. The delay is needed to be sure that the actuator is at the right position. This (big) delay results in a connection loss with the Raspberry Pi.

The arms of R3D3 are mounted on each side of the shoulder part with Dynamixel servo motors. These servo motors can perform a relatively fluent motion (without tablet mass). The value of the servos are always the same. During the initialization of the Dynamixel servo motors, the two servos are set one by one. This causes a small delay between the two arms and this results in unsynchronized arms for a couple of seconds. The same holds for shutting down the system. Currently the arms are made of acrylic, which is probably not stiff enough to hold a tablet. If the arms are stiff enough there is enough room to absorb the delay between the two arms and the tablet will not break. The minimum and maximum angle of the arms can be adjusted in the initiation files of the Dynamixel servo motors. The current minimum angle is 0 degrees and the maximum angle is 90 degrees. The mentioned -30 to 30 degrees in the requirements (see section 3.3) will not fit in the current design. The minimum angle is -10 degrees (seen from vertical position).

5.2 User evaluation

R3D3 has been designed to work with visitors of the clients (NEMO, Dutch Police Academy and iretail.solutions). To test the idea of R3D3 the robot has been evaluated during a number of demonstration sessions testing user interaction. The interaction with people has been tested. The first demonstration was given during an 'innovation day' of the Police Academy in Apeldoorn. The report of this event can be found in appendix A. At that time the R3D3 design was not completed yet. The tablet was not available, so a stand alone monitor was used. Due to the fact that everything was connected to the grid, R3D3 was not able to drive around. The connection between the input of the Kinect and face recognition software and the EyePi was not fully implemented. Chosen was to use R3D3 in the manual mode. Still the visitors could get an idea of what R3D3 could do. The reactions of the visitors were positive. They liked the fact that R3D3 really had a human-like face and the breathing aspect was received well. Despite the dialogue options were very limited, the visitors have a nice dialogue with the virtual human. Most of the people did not have any problems talking to a virtual person.

Another event was the RaM open day on the 24th of January 2017. During the innovation day at the Police Adacemy there were only adults, but on the open day of RaM there were mainly children. In this case it was necessary to adjust the height. This was done manually due to the fact that the ASAP software and Apollo Broker were not available. There was no tablet with virtual human present. Still the reactions where very positive. Especially the children really liked the fact that the robot had a human-like body, but still looked like a robot. The breathing and the blinking of the robot was received very positive. The first reaction of one of the small guys was "Hij ademt! Ow en hij knippert ook, wat schattig." (Translation: He's breathing! Oh and he also blinks, how cute.) For the children it was no problem to communicate with the robot and they like the fact that they could see R3D3 physically and ask questions.

5.3 Development evaluation

As described before, the code of the EyePi is used as a base for the code of R3D3. At the beginning is took a lot of time to understand the existing code. This was due to the fact that ROS was completely new, but also due to the fact that a lot of code on the EyePi is hard-coded. For example every joint get a (calculated) position variable, but there were filled per joint. This could be done with a loop. In this case the new joints will not be added hard coded, but will just be assigned by the loop. This resulted in some difficulties adding extra joints. Some code had to be rewritten in ram_animator_node and ram_pose_to_dynamixel_node.

Next to this the motion detection part of the EyePi was always enabled for the saliency. But this will be disabled in the R3D3 case, due to the fact that no Pi Camera is used. The only solution (without rewriting the whole code) was to set the set point of the X and Y component of the saliency to the half of the resolution of the matrix display. This set point is mixed with the external input in the ram_hmmm_node. If the weight of the external input is high enough the set point does not matter, but the code of the EyePi is still working without the use of the Pi Camera.

6 Conclusion and recommendations

6.1 Conclusion

The requirements for the hardware and software stated in section 3.3 will be discussed here.

6.1.1 Hardware

- *The look of the robot should invite users to interact with it. (Linssen et al., 2016)*
  Despite the robot is still the first prototype, the reactions are positive. People do not have any problem talking to the robot, even children not.

- *R3D3 should be light, flexible and yielding, perceptive and sensitive, and not too quick in behavior, in order to be inherently safe in environments with unpredictable behavior from visitors, including children. (den Uyl et al., 2016)*
  The total weight of R3D3 is $49.9kg$. Especially the wheel chair base is heavy. This is good for the stability, but is not the best solution for in every environment (especially if there are children). The base does not move fast and the speed of the movement of the arms and height can be adjusted. Without a shell the robot is not child friendly due to the sharp corners and the open joints.

- *The sizes are roughly humanoid as well, the fully extended system reaches $180cm$ in order to allow interaction with adults of average size. For interaction with children it should be able to move down (crouch) to $120cm$. (den Uyl et al., 2016)*
  This prototype of R3D3 reaches a maximum height of $171cm$ and a minimum of $152cm$. This is not as much as wanted. This is the result of the actuator that is used in this prototype. An actuator with a longer stroke in combination with a good lever will fix the height problem.

- *The system should operate primarily on flat floors. (den Uyl et al., 2016)*
  The wheel chair base can move on flat floors. Even small ramps are no problem. This wheel chair base is not part of the design of R3D3, but a temporary solution.

- *The robot body should follow the mimetic constraint, i.e. be as human-like as possible in appearance. (den Uyl et al., 2016)*
  R3D3 has a human-like appearance due to the build of the frame. It has legs with knees, a body, shoulders, arms, a neck and a face with the breathing and blinking aspect.

- *The head should be placed on a natural location, including neck/shoulders. (den Uyl et al., 2016)*
  EyePi, the head, is placed on top of the shoulder part (where the Kinect is mounted). The EyePi has a neck consisting of 3 Dynamixel servo motors, which is mounted on top of the shoulder part. On each side of the shoulder part an arm is mounted. These could be mounted on the middle of the shoulder side part to have a more natural spot.

- *Here a single semi-circular lightweight angular activator could be used for tilting the tablet between say -30 and +30 degrees around vertical position. (den Uyl et al., 2016)*
  For the actuation of the tablet two Dynamixel servo motors are used in combination with arms made from acrylic. The minimum and maximum angle of the Dynamixel can be adjusted and are now configured for 0 to 90 degrees.

- *Main function of the head is to attract and direct attention, to be the first focus of attention upon meeting the robot and shift (or actively direct) this focus to the Discourse system discussed later. (den Uyl et al., 2016)*

The head (EyePi) is noted by the blinking and breathing part in combination with the emotion it is showing. Especially children were attracted by the head. During the demonstration at the Police Academy it was clear that people first saw and talked about the robot and then saw the virtual human for further interaction. The head has the possibility to look to the virtual human on the tablet to let the visitor focus on the virtual human.

- *The electronic hardware should be reliable.*
  The Raspberry Pi and the Arduino have no problem running continuously. If these two are mounted well they will be reliable. The micro switches are used as hardware end stops now. These are reliable to use in this situation.

6.1.2 Software

- *ROS should support a dialogue agent which governs the behaviour of both the robot and the virtual human. (den Uyl et al., 2016)*
  ROS can handle XML-messages which abstracts the ROS-message. Together with ASAP and Apollo broker, a connection between the robot and the virtual human is possible.

- *External software must be able to control the robot.*
  Using ASAP and a message broker (Apollo Broker) XML messages can be send to the Raspberry Pi. ROS can handle these messages if the format is correct (as described in appendix B.2).

- *The software should be reliable.*
  The Raspberry Pi and the Arduino can run continuously. The communication between these to is no problem as long as the Arduino does not have long delays in the code. If the delays are too long the communication is lost and ROS restarts the communication with the Arduino. Using the software for going up, down and stand still works without any problems.

- *The software of the EyePi must be implemented correctly for the same behavior.*
  The software of the EyePi is used as a base for the code of R3D3. The changes that were needed for the correct implementation of R3D3 are done in the EyePi software in such a way that the behavior of EyePi is not changed.

- *The resulting behavior should be fluent.*
  The behavior of EyePi is fluent and natural (van de Vijver, 2016). The behavior of EyePi has not changed, so is still fluent and natural (lifelike). The arms and knees can be adjusted fluently using the MIDI-controller or XML-messages.

6.2 Recommendations

There is always room for improvement, especially due to the fact that the first prototype of R3D3 is build. The recommendations are as follows:

- *Autonomous driving*
  R3D3 can now be driven by the remote control. But autonomous driving is needed for autonomous behavior. This can be achieved by extending the current controller with an extra serial input to be able to communicate with ROS.

- *Free running wheels*
  The wheel chair base of R3D3 could not easily be moved by hand, but only by lifting the whole base. For manual movement a (light) base with free running wheels would be recommended.

- *Docking station*
  The battery of R3D3 is charged with an external charger. It would be nice to create a docking station for charging and storing R3D3. Combining these can be done using a magnetic charger, so no cables are needed.

- *Shell*
  The prototype of R3D3 is build from BOIKON. This results in a sturdy body, but this looks not very friendly. The frame is not children friendly due to raw parts. A shell could be added for a more friendly appearance and for users' safety.

- *Height adjustment*
  To get the minimum height of $120cm$ and a maximum of $180cm$ the actuator should be replaced with one with a longer stroke. If the stroke is long enough the actuator will work without a lever, but it is no problem to use a lever in the construction. If another actuator is used, build-in end stops are recommended. In this case less hardware is needed and this will result in more reliability. The size of the actuator does matter as well for the base construction. Preferred is a small actuator with a bigger stroke with build-in endstops.

- *Arm control*
  The arms should hold a tablet or screen. The current arms are not stiff enough to hold a tablet, so the arms should be made from another material. The control should be updated due to the fact that the initialization of the servo motors of the arms is done one by one. Connecting the two servo motors in master-slave configuration could solve this.

- *Mixing behavior*
  The current control of the robot is transparent, so R3D3 is controlled as a puppet. To have a more autonomous behavior without the use of the external software it is an option to mix the height and arm control inputs in combination with the existing EyePi behavior to change the role of R3D3 to actor.

- *Shutdown button*
  Without the MIDI-controller, R3D3 can not be powered off. An external shutdown button is recommended because R3D3 will operate in autonomous mode.

- *Extra DOF tablet*
  The tablet can now be tilted around vertical position of the frame. For a better user experience two extra Dynamixel servo motors can be added to the 'hands' to be able to tilt the tablet for a better view.

A Report of the demonstration session at the Dutch Police Academy

This section is adapted from (Linssen et al., 2017).

On the 9th of December 2016, the Police Academy hosted an innovation day for police personnel to find out about current and future technological innovations that are of interest to law enforcement. Among these were computer vision applications for surveillance, eagle-drone interaction and R3D3. Kasper van Zon of VicarVision, Pascale van de Ven from the department of Robotics and Mechatronics (UT), and Jeroen Linssen from the department of Human Media Interaction (UT) gave a demo of the first fully integrated prototype of R3D3.

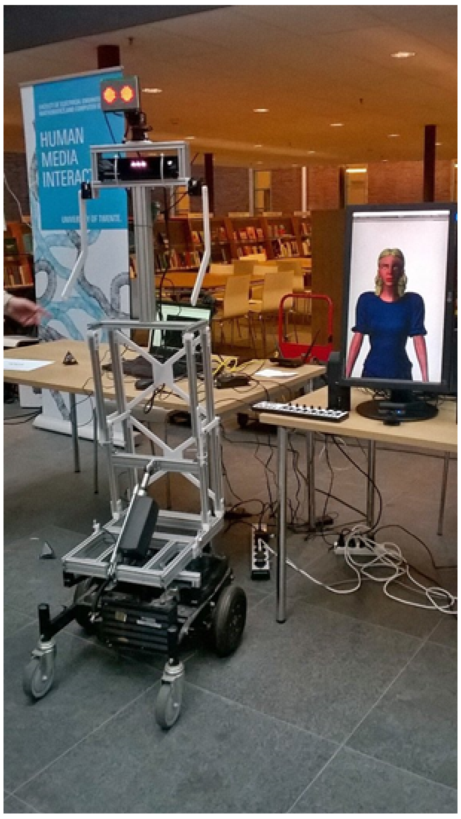


Figure A.1: Overview of frame R3D3

This prototype consisted of a life-size robot, which features the head of EyePi, previously developed at RAM, a Kinect for computer vision, and a virtual human. The robot could not yet hold a tablet in between its hands (as intended), so instead the virtual human was displayed on a monitor standing alongside the robot. Still, the prototype was able to perform several key functions:

- Detect people and their characteristics, using computer vision;
- Pick up and transform people's speech to text using automated speech recognition (ASR);
- Control the behavior of the robot: head movements, emoting, arm movement;
- Control the behavior of the virtual human: speaking, emoting;
- Match questions of people to relevant answers.

In the demo, people could interact with R3D3 by talking to it in Dutch. For this, we wrote a small database consisting of question-answer pairs about the R3D3 project and the technologies involved. The utterances would be recognized as one of the questions in its database. Then, the virtual human responded with a matching answer; or, if a question was not recognized, she replied by asking the person to rephrase that question.

Overall, reactions were positive: visitors were intrigued by all the work that went into creating this combination of technologies. They inspected how the robot head moved, what VicarVision's computer vision told them about their appearance, and how the virtual human responded and tried to understand what they said. This was a challenge at times, especially with the crowded environment in which multiple demos were showcased. Still, the virtual human was able to give a correct response with the first attempt half of the time.

The question-answer pairs we wrote were still very limited in nature, leading to short interactions. This is something we will address in the coming quarter, expanding the dialogue model and diversifying the possible interactions. Furthermore, we will integrate the computer vision in the dialogue model to a larger extent and perform another iteration of the robot/virtual human duo design.

B User Quick start

B.1 Boot and access R3D3

R3D3 automatically boot if power is applied to the Raspberry Pi (and the servos). R3D3 can be used in two modes, which are:

- Autonomous mode
- Manual mode

Standard R3D3 boots in autonomous mode, so it has the possibility to communicate with external software. Between these modes can be switched using the MIDI-controller CYCLE button.

R3D3 can be accessed connected to the same network via SSH. The user name is pi and the password is $SocialRobot$. In the launch file the initial boot modes can be changed. The system can safely switched off using the SHUTDOWN button on the MIDI-controller.

B.2 External communication

To communicate with R3D3 using the external software the autonomous mode is set. Then R3D3 can be accessed and information can be send to the following ROS-topics:

- Saliency (ROS-topic: /ram/animation/hmmm/saliency) (Same as EyePi)
- Emotion (ROS-topic: /ram/animation/hmmm/emotion) (Same as EyePi)
- Sequence (ROS-topic: /ram/animation/hmmm/sequence) (Same as EyePi)
- Height (ROS-topic: /ram/animation/hmmm/heightcontrol)
- Arms (ROS-topic: /ram/animation/hmmm/armcontrol)

Using an Apollo Broker, these topics can be accessed. This broker is connected to the ASAP software, Kinect and the automated speed recognition software. The message broker is the handler for a the information ASAP sends.

The messages that ROS can handle are of the XML-format. The correct format and content is explained in the following 5 subsections.

B.2.1 Saliency

This section is adapted from (van de Vijver, 2016).

It is possible to send saliency maps to the EyePi, which contains information about the detected salient points. The map is communicated with the message template defined in Codeblock B.1. Multiple salient points can and should be communicated in the same message. An update rate of at least 15Hz (which is the same as the internal motion detection) is preferred.

The locationX and locationY attributes must have a value $>= -1$ and $<= 1$. The weight attribute must have a value of $>= 0$ and $<= 10,000$. Internally, the location values are mapped to an 640x640 frame, which is located in front of the robot.

```
string inputId
uint8 deliberate
float32[] locationsX
float32[] locationsY
uint16[] weights
```

Codeblock B.1: Saliency message definition

The $inputId$ parameter is used to distinguish the separate input streams. It should have an static id for every connection and it is used internally to calculate the interest decay.

The *deliberate* parameter defines if the saliency map should be treated as an autonomously generated map, or as a deliberate one. When a map is marked as deliberate, it can not trigger an autonomous shock reaction.

The Apollo example (Codeblock B.2) communicates two salient points and their weights (in a single update) to the toolkit. In the example the salient points are $(-0.6, -0.6)$ and $(0.9, 0.9)$, with a weight of respectively 1000 and 700. They are being send with the id *walle*1 and are set to be deliberate.

```
<data>
      <inputId type="str">walle1</inputId>
      <deliberate type="int">1</deliberate>
      <locationsX type="tuple">
              <value type="float">-0.6</value>
              <value type="float">0.9</value>
      </locationsX>
      <locationsY type="tuple">
              <value type="float">-0.6</value>
              <value type="float">0.9</value>
      </locationsY>
      <weights type="tuple">
              <value type="int">1000</value>
              <value type="int">700</value>
      </weights>
</data>
```

Codeblock B.2: Saliency Apollo message example

Note that when there are no updates, the salient point will be removed after a configurable timeout. When a point keeps getting reported, the saliency will be updated with the new data. When a point is used, it will lose saliency after getting an initial bonus. After loosing focus, a penalty will be given, after which the saliency can recover.

Every time a new most salient point is calculated, a gaze feedback message will be published containing the salient point data.

B.2.2 Emotion

This section is adapted from (van de Vijver, 2016).

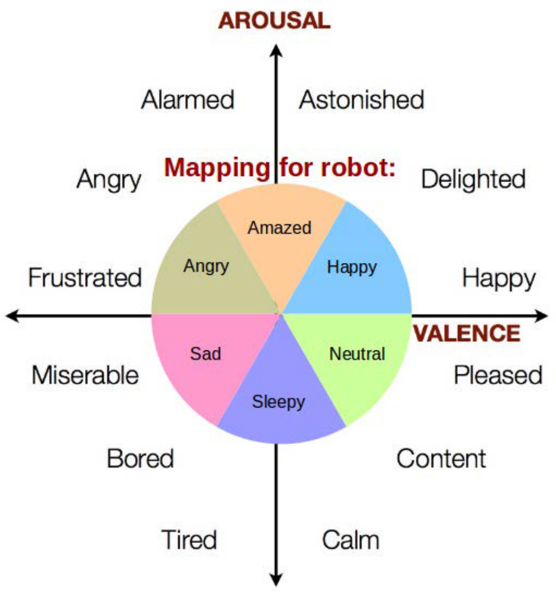


Figure B.1: Arousal and valence to robot emotion mapping

The robot emotion is defined in valence and arousal, which is illustrated in Figure B.1. By adjusting the values, the exact emotion can be communicated. Currently, new values will be mixed with the current value, allowing emotion fading. However, this only works for small changes. If the change in emotion is large enough (which is configurable), the emotion change will be done directly.

The message is defined as given in Codeblock B.3, with an Apollo example in Codeblock B.4. The arousal and valence values can be anything between -1 and 1. The weights can be used to speed up the mixing process if required. By default, the MIDI-controller uses a weight of 40 for both properties.

```
string id
float32 arousal
uint16 arousalWeight
float32 valence
uint16 valenceWeight
```

Codeblock B.3: Emotion message definition

```
<data>
      <id type="str">#uniqid</id>
      <arousal type="float">0</arousal>
      <arousalWeight type="int">10</arousalWeight>
      <valence type="float">0</valence>
      <valenceWeight type="int">10</valenceWeight>
</data>
```

Codeblock B.4: Emotion Apollo message example

The incoming emotion message will trigger a feedback message of the acknowledged type.

B.2.3 Sequence

This section is adapted from (van de Vijver, 2016).

The last communication method can be used to request predefined sequences. The definition is given in Codeblock B.5. The following sequences are supported:

1. Dance
2. Nod
3. Shake
4. Point at the left
5. Point here
6. Point at the right

```
string id
uint8 sequence
float64 request
uint8 requestType
uint8 requestCount
```

Codeblock B.5: Sequence message definition

For the timing request the following types are supported:

1. Start at.
2. Stroke at.
3. End at.
4. Predict timing.

Note that the 'Predict timing' type will only return the predicted timing as a special feedback message, but it will not process the sequence any further. The format of the message if given in Codeblock B.5.

The value given in the *request* attribute is parsed as seconds and is added to the current time to form the requested time. With *requestCount* it is possible to specify how many times a sequence needs to be repeated.

An Apollo example is given in Codeblock B.6, which requests that the EyePi executes one nod in 400ms.

```
<data>
      <id type="str">sequence_dance_1</id>
      <sequence type="int">2</sequence>
      <request type="float">400</request>
      <requestType type="int">1</requestType>
      <requestCount type="int">1</requestCount>
</data>
```

Codeblock B.6: Sequence Apollo message example

Sequences requests can trigger two types of planning feedback, containing either acknowledged or rejected. After a sequence has been acknowledged and planned, there will be progress feedback during the execution of the sequence, on the start, stroke and end synchronization points of the sequence.

B.2.4 Heightcontrol

The height of R3D3 is defined in a certain direction, as can be seen in codeblock B.7. The *direction* attribute must have a value of 0, 1 or 2. Equal to, down, don't move and up respectively.

```
int8 direction
```

Codeblock B.7: Height control message definition

The Apollo example (given in codeblock B.8) communicates a direction. In this example the direction is up.

```
<data>
      <direction type="int">2</direction>
</data>
```

Codeblock B.8: Height control Apollo message example

B.2.5 Armcontrol

The height of the arms (left and right together) of R3D3 is defined as a position, as can be seen in codeblock B.7. The *position* attribute must have a value between 0 and 1/2. Which correspond to radians.

```
float64 position
```

Codeblock B.9: Arm control message definition

The example of an Apollo message can be found in codeblock B.10. The example gives a value of 0.25, so the arms are halfway.

```
<data>
      <position type="float">0.25</position>
</data>
```

Codeblock B.10: Arm control Apollo message example

C Developer guide

The software of R3D3 is build up from the ramSocialRobot.launch file. In the launch file all the necessary components for R3D3 are launched, from parameters, nodes to servos. First the global parameters are set. Then the nodes (that are always necessary) are launched followed by the Dynamixel manager and controller launch files. The Dynamixel manager contains the information about the range of possible IDs and the baud rate. The controller launch file contains the names of the joints which are initialized in the config.yml file. Then the nodes for the motion detection and the HMMM are initialized. The motion detection is disabled in the case of R3D3, as described in section C.2.

An overview of important functions of the software are given to know what to adjust for extra functions of the robot.

C.1 Mode selection

The initialized mode is set in the ramSocialRobot.launch file with a global parameter node_enabled. If this parameter is set to 'True' R3D3 initialize in the autonomous mode otherwise in the manual mode. Using the MIDI-controller the mode can be switched using the CYCLE button.

C.2 Motion detection

The motion detection of the EyePi is integrated in the HMMM block. A Pi camera is used to detect motion. Due to the fact that no Pi Camera is used a parameter in the ramSocialRobot.launch file is used to disable the motion detection part. This parameter (saliency_enable) is used in the ram_hmmm_node to set the saliency to a fixed point. Setting this parameter to 'False' set the saliency to a fixed point, otherwise the input of the Pi Camera is used.

C.3 Extra Dynamixels

Adding extra Dynamixels to the existing bus consist of some changes in the code. First the limits of the Dynamixel should be set using the Dynamixel Wizard in the RoboPlus program. Connect the Dynamixel with the USB2Dynamixel to a Windows computer. Make sure the ID is set to another ID than the already used ones (11, 12, 143, 14 and 15 in the R3D3 case). The minimum and maximum angle (CW and CCW Angle Limit) are set. The baudrate should be set to 1 (10000000bps) to be able to communicate with the other Dynamixels on the bus and the Raspberry Pi.

After the setup in the Dynamixel itself the Dynamixel servo motor can be connected to the existing bus. By updating the config.yml file in the ram_dynamixel_config package the joint exists for ROS. In this configuration file the name, speed, ID, initial value and maximum and minimum are set. These minimum and maximum should be in the range of CW and CCW Angle Limits of the Dynamixel itself. Otherwise it will never reach the asked position. Add this information to the shutdown_config.yml file for a save shutdown procedure. This file is used if the system is powered off. Add the name of the joints to the controller.launch and the shutdown_controller.launch files to use the information in the configuration files. Now the Dynamixel motor servos can be used with the JointState message. This message can contain an array with the name and position of each servo.

C.4 Rosserial

To be able to communicate with Arduino over a serial connection the package rosserial is used. This is a protocol for wrapping standard ROS serialized messages an multiplexing multiple topics and services over a character device [1]. In the case of R3D3 the rosserial_python package is used. In the ramSocialRobot.launch file a node for the serial is created with an unique name and port. In the serial_node.py file the baud rate is set. If the baud rate of multiple Arduino is the same, the same serial_node.py file can be used.

C.4.1 Arduino header files

To be able to communicate with the correct ROS-message a header file is created for the specific message by executing the following command on the Raspberry Pi:

```
rosrun rosserial_arduino make_libraries.py <destination folder>
```

To be able to compile the Arduino code, copy the header files to the Arduino/libraries/ros_lib folder on the computer where the Arduino code is compiled.

C.5 MIDI-controller

The MIDI-controller is used to manual control the robot. The MIDI-controller has several sliders and buttons that can be used for controlling the robot. These inputs are set in the ram_midi_to_animation_node. The value of the slider or button can be published to a ROS-topic. The button layout as used for R3D3 can be found in figure C.1.

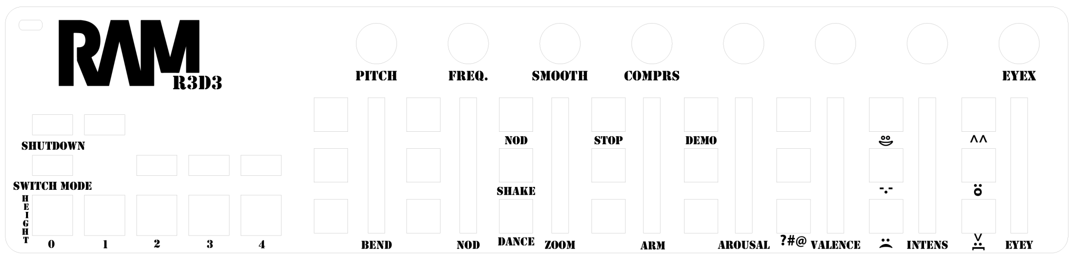


Figure C.1: Overview of MIDI-controller

C.6 Apollo Bridge

To connect to the Apollo Bridge a node with the asap_ros_bridge is launched. This is done in the ramSocialRobot.launch file. To be able to communicate the IP-adres in the relay.py file is updated to the IP-aders of the message broker. Now ROS is able to communicate with the Apollo Bridge and so the external software.

[1] http://wiki.ros.org/rosserial

D Image references

1. Pepper:
   http://www.digitaltrends.com/cool-tech/man-arrested-for-assaulting-pepper-the-robot-that-can-read-your-emotions/ (viewed on 27-02-2017)

2. FURo-S:
   http://newlaunches.com/archives/furo_robot_capable_of_emotional_interactions_unveiled_at_the_7th_korea_communications_conference_press_party.php (viewed on 27-02-2017)

3. Airport Guide Robot LG:
   https://www.engadget.com/2017/01/04/lg-robots-at-ces-2017/ (viewed on 27-02-2017)

4. RobovieR3:
   https://www.youtube.com/watch?v=LY_bPsszaH8 (viewed on 27-02-2017)

5. ASIMO:
   http://www.dailymail.co.uk/sciencetech/article-2059135/Asimo-humanoid-robot-A-drinks-waiter-dont-need-tip.html (viewed on 27-02-2017)

6. FROG:
   http://www.utnieuws.nl/nieuws/59900/Robot_heet_koning_welkom_en_zoeft_over_campus (viewed on 27-02-2017)

7. Phoron:
   https://www.dailygizmo.tv/2008/04/04/phorone/ (viewed on 27-02-2017)

8. Jiajia:
   http://nos.nl/artikel/2139381-video-jiajia-lijkt-een-mens-maar-is-toch-echt-robot.html (viewed on 27-02-2017)

9. Sophia:
   http://www.cnbc.com/2016/03/16/could-you-fall-in-love-with-this-robot.html (viewed on 27-02-2017)

10. Nadine:
    http://www.telegraph.co.uk/science/2016/03/12/meet-nadine-the-worlds-most-human-like-robot/ (viewed on 27-02-2017)

11. XIBOT:
    http://www.instash.com/xibot-will-be-your-eager-robotic-helper-and-friend/ (viewed on 27-02-2017)

Bibliography

Bayas, L. S. (2008), Phorone.
http://www.trendhunter.com/trends/robotics-phorone-the-secretary

Beciri, D. (2010), Robovie R3 robot is much better yet cheaper than previous version.
http://www.robaid.com/robotics/robovie-r3-robot-is-much-better-yet-cheaper-than-previous-version.htm

Breazeal, C. (2001), Affective interaction between humans and robots, in *European Conference on Artificial Life*, Springer, pp. 582–591.

Dautenhahn, K. (2007), Socially intelligent robots: dimensions of human–robot interaction.

Dertien, E. (2007), Horseshoe crab.
http://edwindertien.nl/outside/horseshoe-crab/

Fong, T., I. Nourbakhsh and K. Dautenhahn (2003), A survey of socially interactive robots.

Frogrobot (2016), The FP7 FROG project.
https://www.frogrobot.eu/wordpress/

Honda (2017), Say Hello to ASIMO.
http://www.honda.com/mobility/say-hello-to-asimo

Huang, J. (2017), XIBOT: Your New Robotic Family Member.
https://www.indiegogo.com/projects/xibot-your-new-robotic-family-member--2#/

IEEE, S. (2014), How Aldebaran Robotics Built Its Friendly Humanoid Robot, Pepper.
http://spectrum.ieee.org/robotics/home-robots/how-aldebaran-robotics-built-its-friendly-humanoid-robot-pepper

LEO (2016), R3D3 Rolling Receptionist Robot Double Dutch Dialogue.
http://www.leorobotics.nl/news/r3d3-rolling-receptionist-robot-double-dutch-dialogue

Linssen, J., M. Thuene, R. Kuijpers, D. de Jong, B. Landman and R. Boelsma (2016), R3D3 D1: User requirements and expectations.

Linssen, J., P. van de Ven, E. Dertien and K. van Zon (2017), R3D3 D11: Integrated internal test system.

Oosterkamp, J. (2015), Creating a toolkit for human-robot interaction.
http://essay.utwente.nl/67511/

Robot, F. (2016), FURo-S.
http://www.myfuro.com/furo-s/service-feature/

Robotics, K. (2017), Kompaï Robotics.
https://kompai.com/

Savov, V. (2010), Robovie R3 all set to assist, freak out elderly and handicapped shoppers this November.
https://www.engadget.com/2010/05/05/robovie-r3-all-set-to-assist-freak-out-elderly-and-handicapped/

Sijbrands, C. (2017), De LG Airport Guide Robot assisteert je op het vliegveld en beantwoord al je vragen.
http://numrush.nl/2017/01/05/lg-airport-guide-robot/

den Uyl, M., H. Tangelder, J. Linssen, L. van der Werff, R. Kuijpers, P. Bison and E. Dertien (2016), R3D3 D2: System design and constraints.

van de Vijver, B. (2016), A Human Robot Interaction Toolkit with Heterogeneous Multilevel Multimodal Mixing.
http://essay.utwente.nl/71171/