K.Anter | M.Greiner | J.Vatter | J.Weghake
Direction: H.Schnaubelt

SoftBank Robotics

NAO 6

DO IT

TECHNIK LPE

sprachliches + naturwissenschaftlich-technologisches
bertha-von-suttner
gymnasium neu-ulm

CREATIVE PROJECT IDEAS

Do it NAO⁶ - Creative Project Ideas
1st edition 2019 - Updated version December, 2020.

Authors: Kai Anter, Marcel Greiner, Jonas Vatter, Jannes Weghake
Direction: Heike Schnaubelt (OStR, senior councilor of studies)
Editor: Tony Schuster (StR, councilor of studies)
Program contribution: Caspar Sachsenmaier

All program examples and screenshots for NAO⁶ were tested and taken in the environment Choregraphe (Version 2.8.6.23).
It is possible to transfer the content to earlier NAO generations, however their correct functioning cannot be guaranteed. The same applies the other way around.
In this book, "NAO" always refers to the sixth generation of NAO.

TECHNIK LPE

Technik-LPE GmbH
info@technik-lpe.com
www.technik-lpe.de

Friedrichsdorfer Landstraße 64
69412 Eberbach / Germany
Phone: +49 6271 944650-1
Fax: +49 6271 944650-2

We,
the team of authors comprised of **Kai Anter**, **Marcel Greiner**, **Jonas Vatter**, and **Jannes Weghake**, under the direction of **Heike Schnaubelt (OStRin)**, are pleased to present our book "Do it NAO^6 - Creative Project Ideas".

It was created as part of our study skills seminar "The real and virtual STEM classroom of the future" in cooperation with the company **Technik-LPE GmbH**.

Our target group:

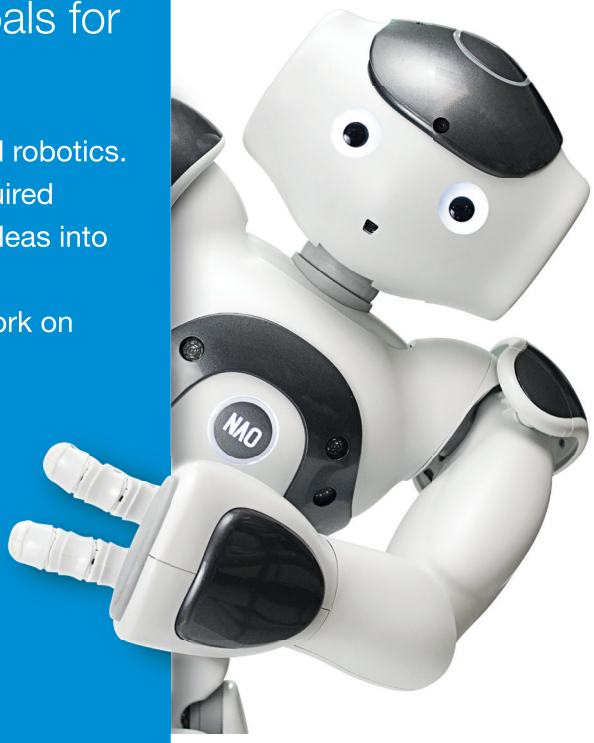
Anyone who is interested – anyone at all!

We have set ourselves the following goals for the book:

- NAO^6 is intended as an ideal introduction to humanoid robotics.
- This book, NAO^6 with a laptop/notebook, and the required materials are all you need to put the creative project ideas into practice.
- We want to encourage independent and structured work on creative project ideas in the STEM field.

We will achieve these goals through:

- Clear illustrations
- Step-by-step instructions
- Practical exercises with suggested solutions



Together with this book you will receive a link to our virtual archive from Technik-LPE GmbH. This contains all material templates and suggested solutions for all the projects.

We hope we can inspire you to explore this new universe and that this book will be of help to you in doing so.
Have fun discovering this new NAO-verse!

Neu-Ulm, April 2019

Your author team

Kai Anter | Marcel Greiner | Jonas Vatter | Jannes Weghake

Preface

Project ideas

Idea	Application	Time	Difficulty	Chapter
NAO shopping	NAO as a sales assistant	Short	Easy	1
	NAO as a shopping helper	Long	Advanced	12
	NAO goes shopping	Medium	Extreme	17
NAO as teacher	NAO recognizes famous faces	Short	Easy	3
	NAO as a calculator	Medium	Advanced	14
	NAO as a language teacher	Long	Advanced	11
	NAO learns to draw and write	Long	Extreme	16
NAOmarks	NAO recognizes traffic signs	Medium	Easy	2
	NAO plays rock-paper-scissors	Medium	Easy	6
	Remember that?	Medium	Advanced	9
	Playing dice with NAO	Long	Extreme	18
NAO remote control	Steering NAO	Medium	Easy	8
	NAO helps tidy up	Long	Advanced	10
	Template: NAO remote control	Long	Extreme	20
Sport with NAO	NAO recognizes a red ball	Medium	Easy	5
	Playing ball with NAO	Medium	Easy	4
	Gymnastics with NAO	Long	Easy	7
	NAO plays soccer	Long	Extreme	15
Entertain-ment	NAO tells a joke	Medium	Advanced	13
	Your own Basic Channel!	Long	Extreme	19

Tips and tricks

Here are several tips and tricks formulated to use NAO efficiently and properly. For more information refer to the NAO6 Pocket Guide ***https://www.softbankrobotics.com/emea/en/support/nao-6/1-meet-nao***

1. Do not work with your NAO on tables or other raised surfaces.
2. Disconnect the Ethernet and charging cables from NAO after you have transferred the program (risk of tripping).
3. It is possible to use NAO while it is in charge with limitations in movements.
4. Do not set NAO's movement speed above 80% as it could overbalance (recommended values: 60% - 70%).
5. Always let your NAO move on an even and stable surface; otherwise it is more likely to fall over and do not leave NAO unattended for a longer period of time.
6. Do not lift NAO while it is sitting as this triggers the Fall Detection. (The Fall Detection is a procedure that is always active on NAO. If NAO falls, it prevents severe damage. NAO goes into a defensice position very quickly. Your fingers could be trapped when this happens.)
7. Only grab NAO by holding at the chest level placing your hands under the arms.
8. Use the Pose Library with caution. If you want NAO to stand up, use the Stand Up box. (The Pose Library is a collection of several positions that NAO can assume. You will learn about the Stand Up box later in C.2.1.5.)
9. Only store NAO in the styrofoam box or the transport case that is available separately, or lean it against a wall with a piece of fabric in between (risk of tipping, damage).
10. When NAO is switched off, the battery needs around 2 hours to charge from 0%to 100%, when switched on around 2.5 hours.
11. NAO needs around 2-3 minutes to start up.
12. The applications are designed for using a physical robot, however many applications can run on simulated robots.
13. Keep the Styrofoam protection provided as a storage for NAO.
14. Use the latest version of Choregraphe (2.8.6.23 or above). Make sure your NAO 6 is upgraded to the latest Naoqi version (2.8.6.23 or above). Examples and exercises have been tested on this version of the robot. If you use previous version of NAO or Choregraphe, you may notice differences in executing the examples.

Tips for troubleshooting

If a program is not running as planned, ask yourself the following questions:

General

Are you connected to a "real" NAO?
Do you have the latest version of your project open?
Did you try reconnecting to NAO?
Did you try reconnecting to the network?
Did you try restarting Choregraphe?
Did you restart NAO?
Did you try restarting the network?
Have you replaced umlauts, accents, or special characters in the path for Choregraphe projects or in file names?
If NAO doesn't stay in still, have you deactivated Autonomous Life?

Programming with Choregraphe

- ☐ Have you been through the program step by step?
  Tip: It often helps to explain to a person or object (like a rubber duck or soft toy) how the program works and which individual steps are taken. This helps to uncover errors.
- ☐ Are the boxes connected correctly?
- ☐ Is there a connection missing somewhere?
- ☐ Are the parameters of the box correct?
- ☐ If a certain box doesn't work, have you tried deleting the box and recreating it?
- ☐ Are the key parameters in the Memory boxes (Insert Data, Get Data) correct?
- ☐ Are the key parameters in the Event boxes (Subscribe to Event, Raise Event) correct?
- ☐ Was the right camera selected (top/bottom)?

Timelines

- ☐ Have the frame names been written correctly in the Goto boxes? Are there any typos in the frames?
- ☐ Were the keyframes set correctly?
- ☐ Are the keyframes in the right places?
- ☐ Are the Stop/Play boxes in the right place?

Vision Recognition

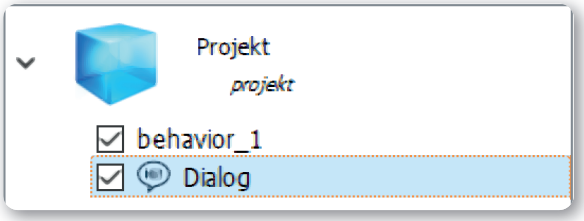
Did you send the Vision Recognition Database to NAO?
Is there enough light in the room?
Have you tried learning and saving the object again?
Is the alignment of the object correct?

Python

- ☐ Is there a self.onStopped() at the end in the Python source code that ends the box and issues a signal at the onStopped output?
- ☐ Is there a "self." missing in the Python source code?
- ☐ Is your source code indented correctly?
- ☐ Have you written the variable and method names correctly?

Dialog

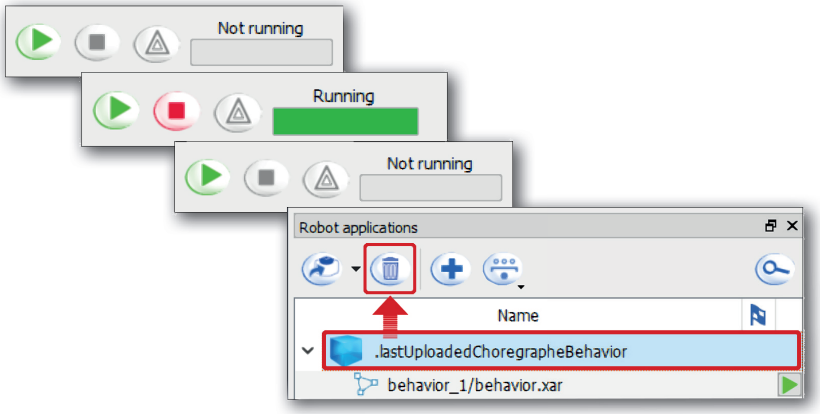
- ☐ Are there parentheses, colons, tildes (~) missing anywhere?



Screen 1.3.1 - Dialog under "Project Properties"

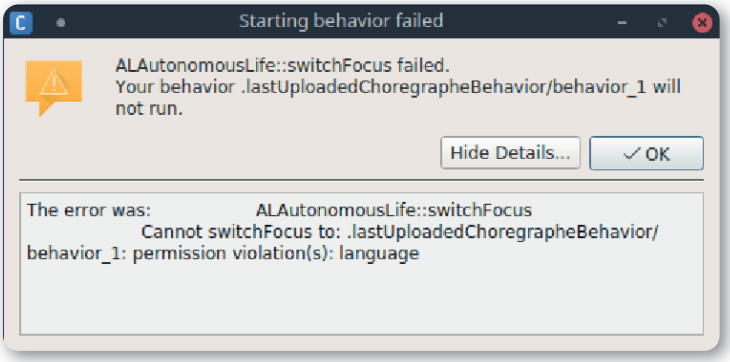
- ☐ Was the dialog selected in the "Project Properties"?
- ☐ If you are unable to insert a dialog in your language, was this language selected in the supported languages in the project properties?

Special cases



Screen 1.3.2 - Upload error

- ☐ If you click on the green start arrow and the bar to the right briefly turns green and then switches back to "Not running", you can try to delete ".lastUploaded-ChoregrapheBehavior" under "Robot applications" and upload your program again.



Screen 1.3.3 - Error "Starting behavior failed"

- ☐ If this error appears, you need to select the language currently used by NAO under Properties > Supported languages. American English should mostly suffice. If the error still appears, you can select all installed languages for NAO.

NOTES

Structure of the projects

The structure of the projects is described below. This structure is used for every project idea.

Overview

The overview gives you a brief insight into the project. After a short introduction to the topic and a picture, there is a brief description of the project or game.

Objects required and preparation

This section lists the objects required for the project. If you need to prepare something, the steps are explained. If there are digital materials for a project, they can be found in the virtual archive.

Suggested solution

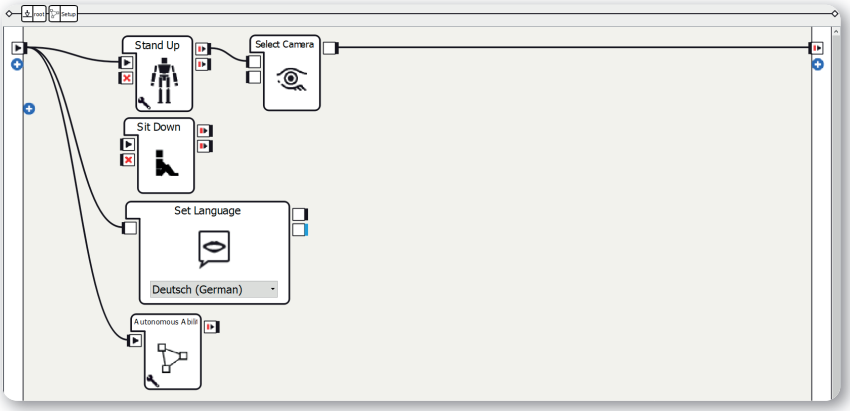
Under the "Suggested solution" heading is a step-by-step explanation of how to program the project.

Most projects have a Setup diagram and a Start Condition diagram. The structure of these diagrams in the individual projects is very similar. In the projects, you only need to make small changes.

The basic framework

In the basic framework you can see the rough flow of the program. Mostly, Diagram boxes are used to keep the project clear and easy to follow.

The Setup diagram



Screen 1.4.1 - The Setup diagram

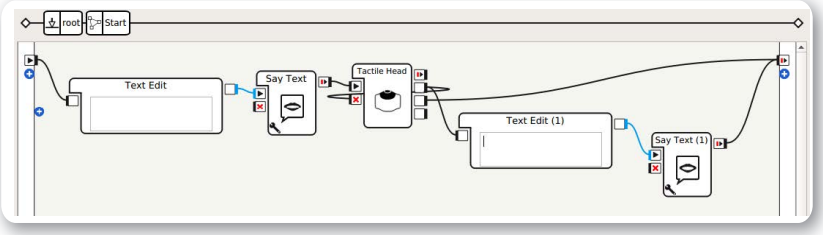
Here are the boxes that need to be run when the program is started. The structure of the Setup diagram is the same in most projects, however changes are sometimes needed. These changes are explained in the individual projects.

You can create a general Setup diagram as follows:

1. Create a Stand Up or Sit Down box
2. Create a Select Camera box
3. Create a Set Language box
4. Set the language in the Set Language box to your language
5. Create an Autonomous Abilities box
6. Make connections as shown in the screenshot

The Stand Up or Sit Down box makes NAO stand or sit. The Select Camera box activates the top or bottom camera, depending on the input connected. The Set Language box sets the language to the one you chose so that you can play with NAO in your own language. The Autonomous Abilities box activates or deactivates NAO's automated processes (e.g. blinking or idle movements).

Start Condition diagram



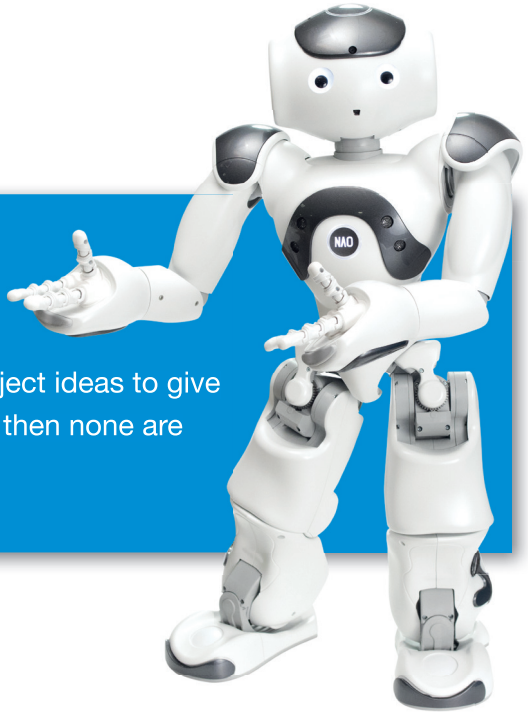
Screen 1.4.2 - The Start Condition diagram

The Start Condition diagram starts or stops the project. The sample texts you can write in the Text Edit boxes are described in the individual project ideas.

You can create a general Start Condition diagram as follows:

1. Create a Text Edit box
2. Create a Say Text box
3. Create a Tactile Head box
4. Connect the frontTouched output with the onStop input of the Tactile Head box
5. Create another Text Edit box
6. Create another Say Text box
7. Make connections as shown in the screenshot

In this general Start Condition diagram, the program is started by touching the front head sensor and stopped by touching the middle head sensor. If you want, you can use other pressure sensors, speech recognition, a Naomark, or a smile (using the Get Smile box). It's up to you what you choose.



More project ideas

Depending on the project, there are more project ideas to give you inspiration. If you don't see this heading, then none are listed for this particular project.

NOTES

1 NAO as a sales assistant

Overview

Can robots replace human sales assistants in retail? Robots can access databases and the Internet while people have the advantage of charisma and feelings.

The "NAO as a sales assistant" project is a simple question-and-answer game between NAO and a customer. After a few questions, the customer has found the product they are looking for.

Objects required and preparation

Image 2.1.2.1 - Objects required

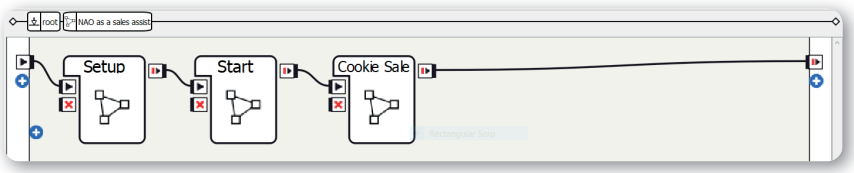
For this project you need:

- Optional: Cookie packages or containers
- Optional: A sales stand

You don't need any objects but to make it more realistic, you can use suitable props such as cookie packages and a sales stand.

Suggested solution

Basic framework



Screen 2.1.3.1 - Basic framework

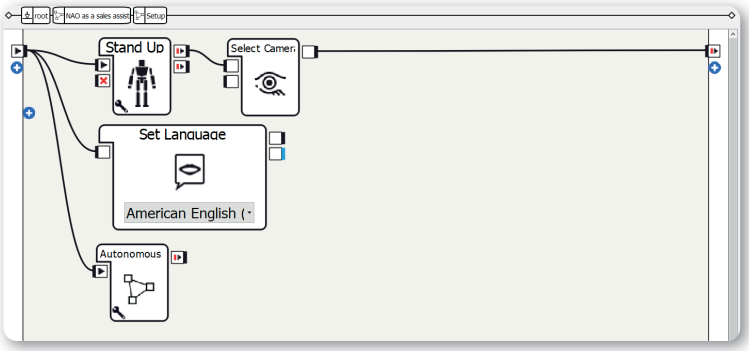
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start, and Cookie Sale
3. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram, NAO reacts to the signal word "NAO".
- Then NAO introduces itself and asks whether you would like chocolate or not.
- If you answer "no", it offers you cookies without chocolate.
- If you answer "yes", it asks whether you would like dark, white, or milk chocolate.
- It offers the recommended type of cookie depending on the answer.

The Setup diagram

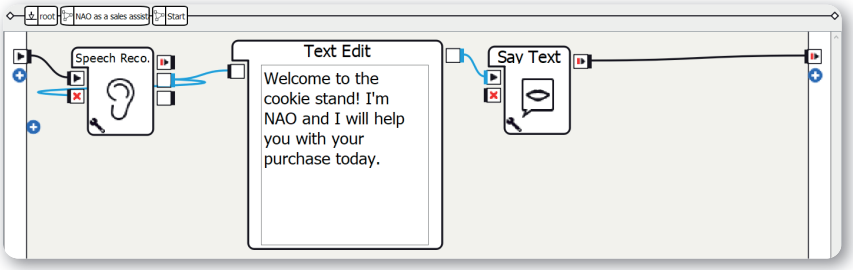


Screen 2.1.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start diagram



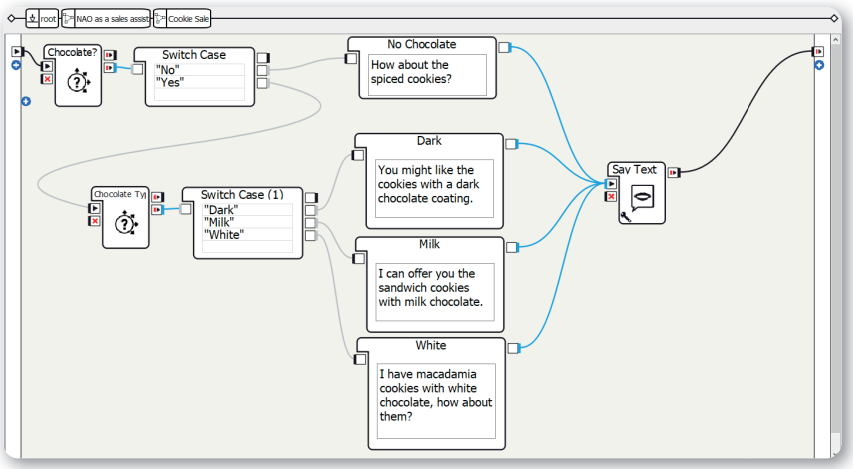
Screen 2.1.3.3 - The Start diagram

Method:

1. Create a Speech Recognition box
2. Set the Word List parameter to "NAO"
3. Create a Text Edit box
4. Set the text to something like "Welcome to the cookie stand! I'm NAO and I will help you with your purchase today."
5. Create a Say Text box
6. Make connections as shown in the screenshot

In this section, NAO welcomes the customer. The Say Text box then says the text entered.

The Cookie Sale diagram



Screen 2.1.3.4 - The Cookie Sale diagram

Method:

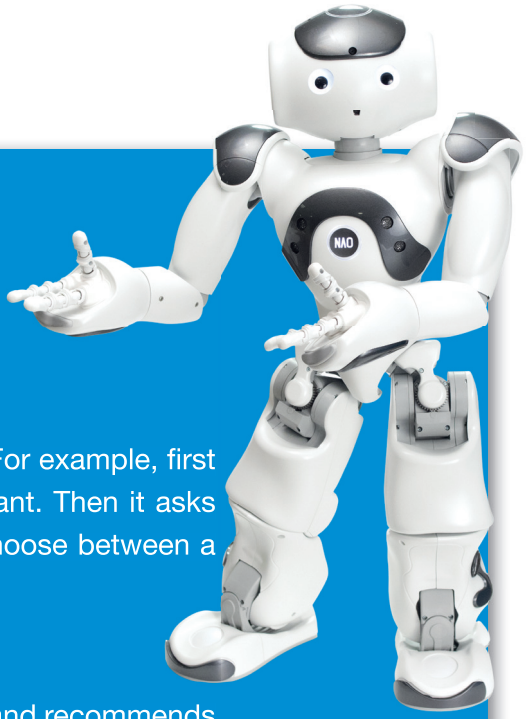
1. Create a Choice box and do the following:
    a. Under Localized Text, select the language you want and enter the question, here: "Do you like chocolate?"
    b. Under Choice, select the language you want and enter possible answers, here: "Yes" and "No"

2. Create a Switch Case box and enter the cases "Yes" and "No"
3. Connect the Switch Case box to the Choice box for chocolate yes/no
4. Create a Text Edit box and connect it to the "No" case of the Switch Case box
5. Set the text to something like "How about the spiced cookies?"
6. Create a Say Text box
7. Create another Choice box, connect it to the "Yes" case of the Switch Case box, and do the following:
    a. Under Localized Text, select the language you want and enter the question, here: "Which type of chocolate do you prefer? Dark, milk, or white?"
    b. Under Choice, select the language you want and enter the possible answers, here: "Dark", "Milk", and "White"

8. Create a Switch Case box and enter the cases "Dark", "Milk", and "White"
9. Connect the Switch Case box to the Choice box for the type of chocolate
10. Create three Text Edit boxes and connect them to the three cases of the Switch Case box for the type of chocolate
11. Set the texts to something like:
    a. Dark: "You might like the cookies with a dark chocolate coating."
    b. Milk: "I can offer you the sandwich cookies with milk chocolate."
    c. White: "I have macadamia cookies with white chocolate, how about them?"

12. Create a Say Text box
13. Make the rest of the connections as shown in the screenshot

This is where most of the interaction takes place: NAO asks whether you want chocolate or not. If you don't want chocolate, it offers spiced cookies. If you do, NAO asks what kind of chocolate you would like. It offers different types of cookie depending on the answer.



More project ideas

NAO the ice cream seller

You can program NAO as an ice cream seller. For example, first it asks how many scoops of ice cream you want. Then it asks what flavors you would like. Lastly, you can choose between a cone and a cup.

NAO the shoe seller

NAO presents you with different types of shoe and recommends different models. For example, first it can ask what shoe size you have. Then it asks whether you want the shoes for everyday, sport, or a special occasion. Finally, it asks for your favorite color. The customer is given a selection of shoes based on these criteria.

NOTES

2 NAO recognizes traffic signs

Overview

Fully automated and safe driving by robots and algorithms is still a thing of the future. But can robots play the role of a driving instructor and teach people the rules of the road?

In the project "NAO recognizes traffic signs", NAO can use Naomarks to record the meaning of a traffic sign and explain its function.

Objects required and preparation

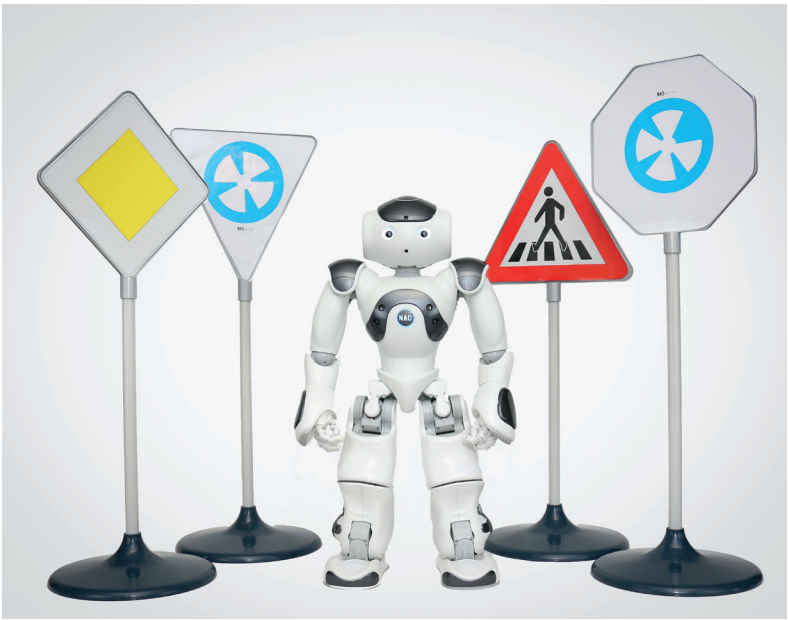


Image 2.2.2.1 - Objects required

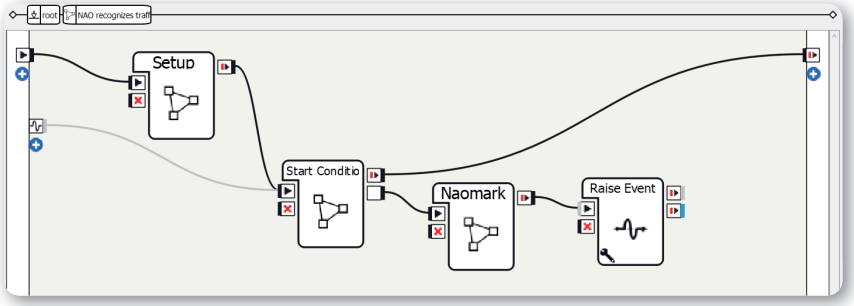
For this project you need:

- Printed Naomarks (recommendation: A4 format)
- Traffic signs, e.g. to stand up or as a printout

First, select several Naomarks and print them out. If you don't have any traffic signs to stand up, you can print out or draw a few traffic signs. Now stick the Naomarks onto the backs of the traffic signs. It makes sense to laminate the traffic signs so that you can reuse them.

Suggested solution

Basic framework



Screen 2.2.3.1 - Basic framework

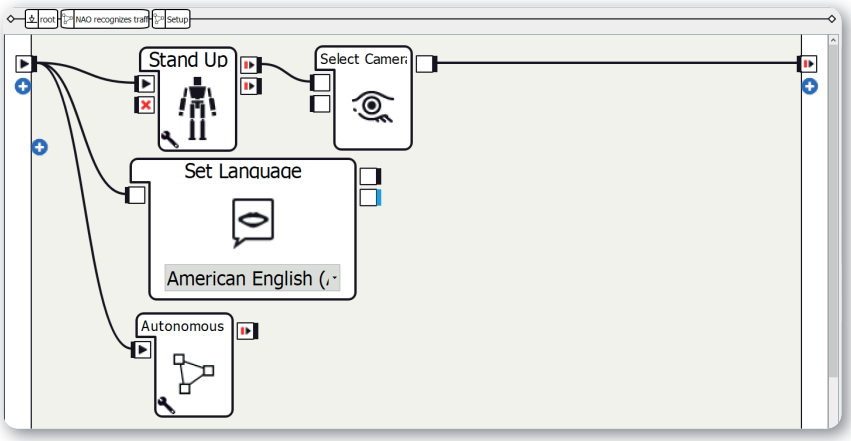
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, and Naomark
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Create a Raise Event box and set the key parameter to something like "trafficsign/again"
5. Add an output from ALMemory on the left, name of the event in this suggested solution: "trafficsign/again"
6. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, the Naomark is recognized in the Naomark diagram and the type of sign is explained.
- Then an event is raised that invokes the Start Condition diagram again.
- This allows you to play the game again.

The Setup diagram

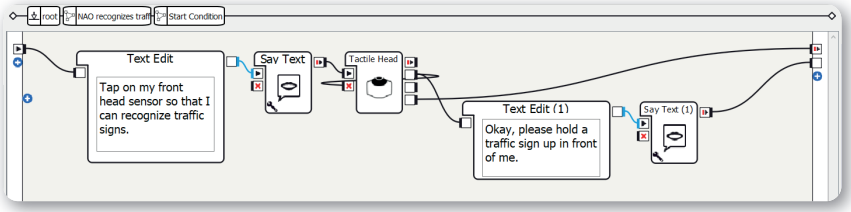


Screen 2.2.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram

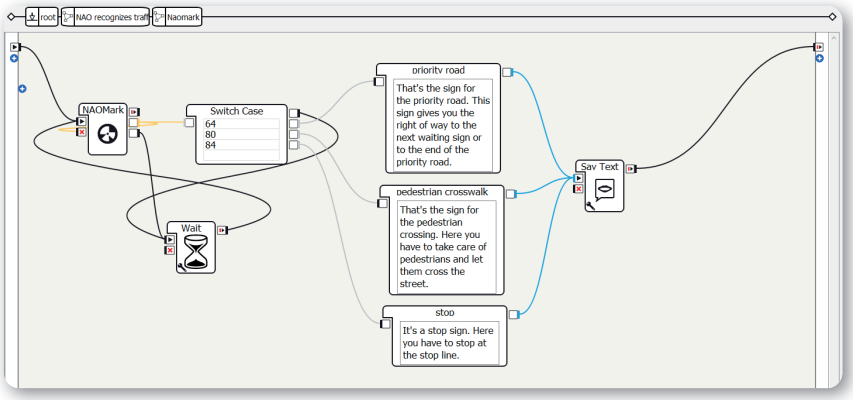


Screen 2.2.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Tap on my front head sensor so that I can recognize traffic signs."
2. Set the text of the right Text Edit box to something like "Okay, please hold a traffic sign up in front of me."

The Naomark diagram



Screen 2.2.3.4 - The Naomark diagram

Method:

1. Create a NAOMark box
2. Create a Wait box
3. Create a Switch Case box
4. Enter the numbers of the Naomarks you selected into the Switch Case box
5. Create a Text Edit box for each case of the Switch Case box
6. Set the text of one Text Edit box to a description or explanation of a traffic sign
7. Connect the Text Edit box to the relevant case of the Switch Case box
8. Example: The stop sign has the Naomark number 84. This means that you have to connect the Text Edit box to the explanation of the stop sign with the case 84 in the Switch Case box.
9. Repeat steps 6 and 7 for all traffic signs
10. Create a Say Text box
11. Make connections as shown in the screenshot

In this program, the Naomarks are used to detect the traffic sign and say the corresponding explanation.

More project ideas

Recognize more traffic signs

You can expand the project by incorporating more explanations for traffic signs.

Recognize traffic signs using object recognition

Instead of Naomarks, you can also use object recognition to detect the shape and color of the traffic sign.

NOTES

3 NAO recognizes famous faces

Overview

Facial recognition by computers and algorithms is getting better, faster, and more efficient all the time. At events, for example, facial recognition can be used to see which people enter the venue.

You can also use this facial recognition yourself with NAO! In the project "NAO recognizes famous faces", you save the faces of famous people in NAO and can make it recognize them.

Objects required and preparation

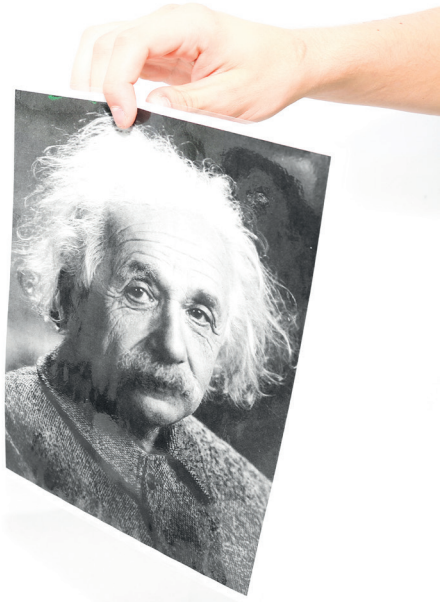


Image 2.3.2.1 - Objects required

For this project you need:

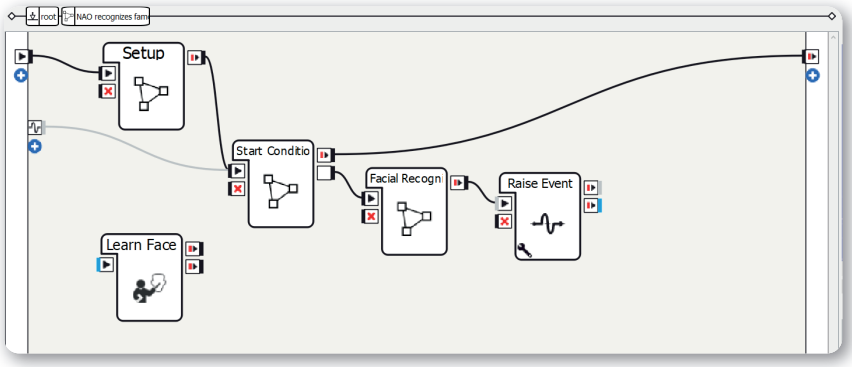
- Printed pictures of famous faces

To perform this project, you need pictures of famous people. The suggested solution uses Albert Einstein (shown as an example in the "Objects required" image), Bertha von Suttner, and Stephen Hawking. You can find pictures of these famous people on the Internet. Pay attention to the copyright and licensing rights of the pictures!

Once you have printed out the faces, you need to save them using the Learn Face box. If you can't remember how to save faces, you can look it up in our first book ***Learn it NAO⁶ – The Basics***. Keep a note of the names you give the faces, as you will need them again later!

Suggested solution

Basic framework



Screen 2.3.3.1 - Basic framework

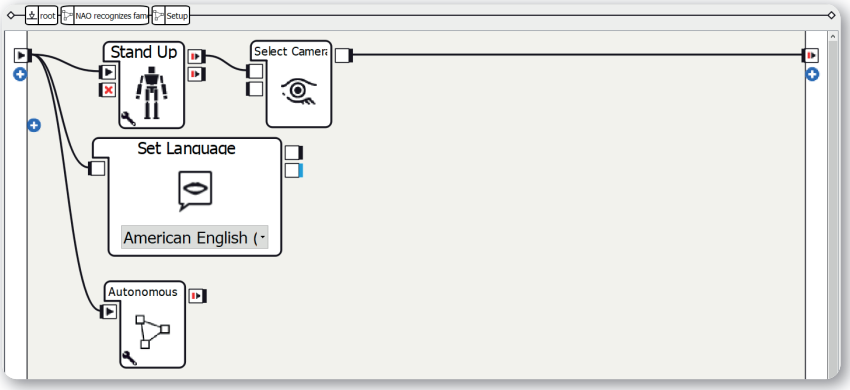
Method:

1. Create a Learn Face box
2. Create three Diagram boxes
3. Rename the Diagram boxes as Setup, Start Condition, and Facial Recognition
4. Add an output with the data type "Bang" to the Start Condition diagram
5. Create a Raise Event box and set the key parameter to something like "famousfaces/again"
6. Add an output from ALMemory on the left, name of the event in this suggested solution: "famousfaces/again"
7. Make connections as shown in the screenshot

Sequence of the program:

- First you need to use the Learn Face box to save the faces of the famous people, otherwise NAO can't recognize them.
- Afterward, when you start the program, facial recognition is started after the Setup and Start Condition diagrams.
- After the facial recognition, the Raise Event box is activated so that you can play the game again.

The Setup diagram

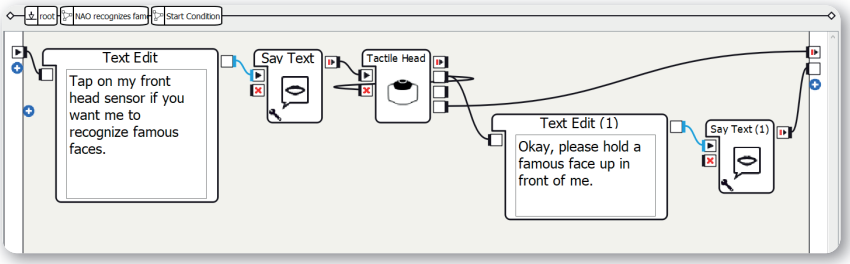


Screen 2.3.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram

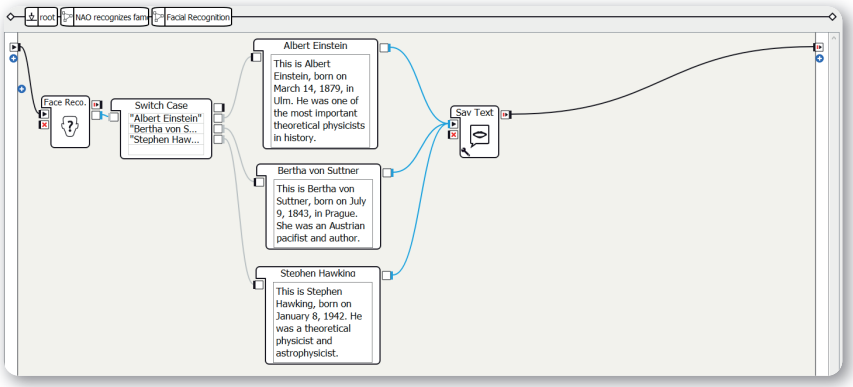


Screen 2.3.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Tap on my front head sensor if you want me to recognize famous faces."
2. Set the text of the right Text Edit box to something like "Okay, please hold a famous face up in front of me."

The Facial Recognition diagram



Screen 2.3.3.4 - The Facial Recognition diagram

Method:

1. Create a Face Reco. box
2. Create a Switch Case box
3. Add the famous faces Albtert Einstein, Bertha von Suttner, and Stephen Hawking
4. Create three Text Edit boxes
5. Enter explanations for the famous people:
   a. "This is Albert Einstein, born on March 14, 1879, in Ulm. He was one of the most important theoretical physicists in history."
   b. "This is Bertha von Suttner, born on July 9, 1843, in Prague. She was an Austrian pacifist and author."
   c. "This is Stephen Hawking, born on January 8, 1942. He was a theoretical physicist and astrophysicist."
6. Connect the explanations to the relevant case for the famous person in the Switch Case box
7. Create a Say Text box
8. Make connections as shown in the screenshot

Once a face has been recognized by the Face Reco. box, the Switch Case box determines which famous person it is. Then the relevant Text Edit box is issued with the information about the person.

More project ideas

NAO knows people

Save all the faces of your family or friends in NAO using photos and surprise them when NAO recognizes them the first time he meets them.

NOTES

4 Playing ball with NAO

Overview

Robots are being used in more and more professions. But can they raise a child? Can they take over the role of parents and teachers?

In the project "Playing ball with NAO", NAO is a playmate with whom you can play with a red ball. You roll a ball to NAO and NAO rolls it back to you. This can be repeated as often as you want.

Objects required and preparation

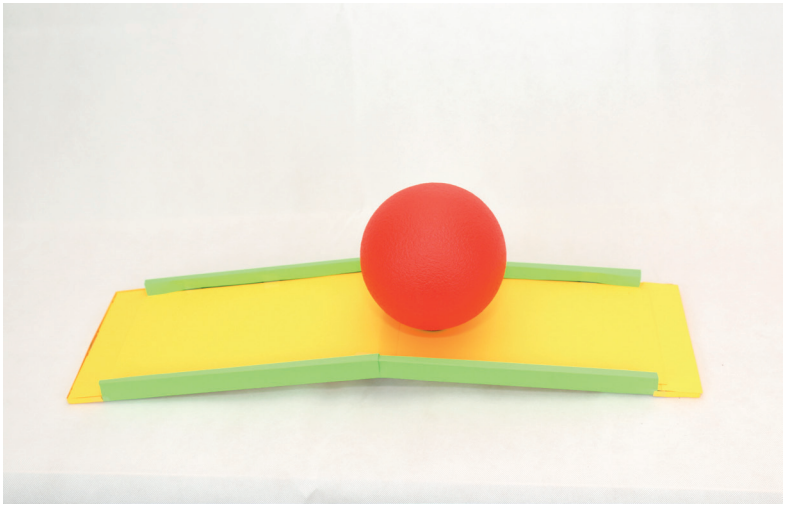


Image 2.4.2.1 - Objects required

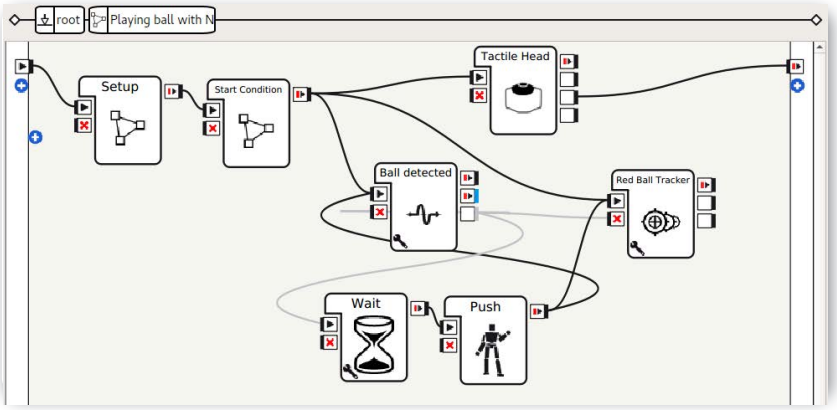
For this project you need:

- A red ball (minimum diameter: 10 cm; optimum diameter: 15 cm)
- A flat ramp (maximum height: 5 cm)

For this game you need a flat ramp so that the ball can roll to NAO more easily. If you don't have a flat ramp, you can make one out of building blocks or cardboard.

Suggested solution

Basic framework



Screen 2.4.3.1 - Basic framework

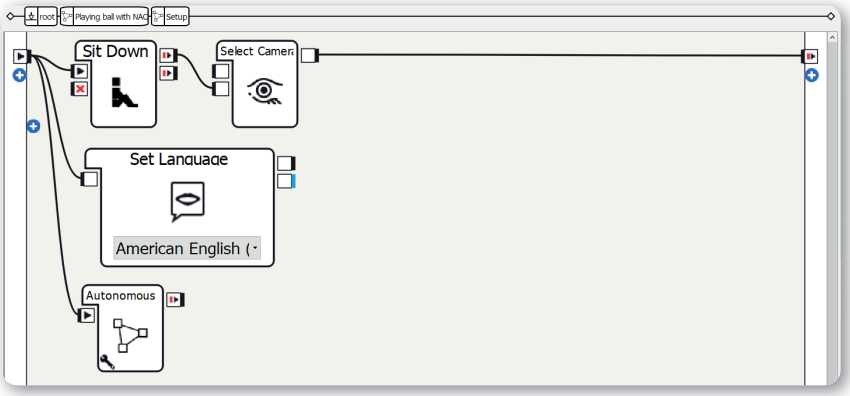
Method:

1. Create two Diagram boxes and a Timeline box
2. Rename the Diagram boxes as Setup and Start Condition
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Rename the Timeline box as Push
5. Create a Tactile Head box
6. Create a Subscribe to Event box (in the screenshot: "Ball detected")
7. Set the event parameter of the Subscribe to Event box to “redBallDetected”
8. Create a Wait box
9. Set the "Timeout (s)" parameter to 2
10. Create a Red Ball Tracker box
11. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, NAO looks down and reacts if it detects a ball.
- After two seconds it then rolls the ball away with its hands and looks for the red ball again.
- You can end the game by tapping on NAO's middle head sensor.

The Setup diagram

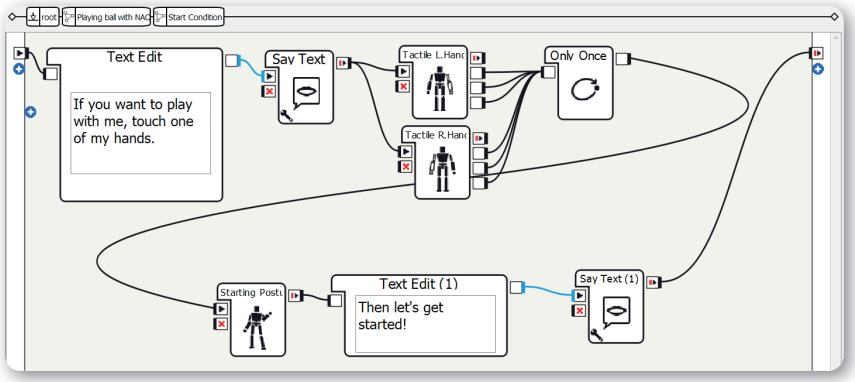


Screen 2.4.3.2 - The Setup diagram

The structure of the Setup diagram is explained the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use a Sit Down box
2. Use the bottom input of the Select Camera box
3. **Only** activate the Autonomous Blinking parameter in the Autonomous Abilities box; deactivate the other three.

The Start Condition diagram



Screen 2.4.3.3 - The Start Condition diagram

This Start Condition diagram differs from the general Start Condition diagram as the touch sensors on the hands are used in this project.

Method:

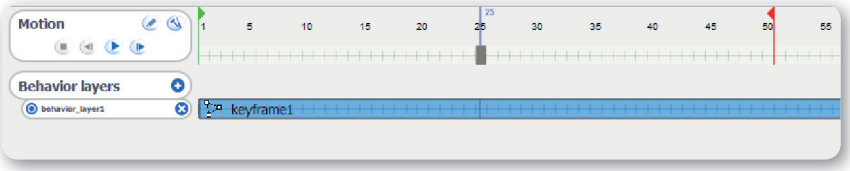
1. Create a Text Edit box
2. Set the text to something like

   "If you want to play with me, touch one of my hands."
3. Create a Say Text box
4. Create a Tactile L. Hand and a Tactile R. Hand box
5. Create an Only Once box
6. Create a Timeline box (in the screenshot: Starting Posture)
7. Create another Text Edit box
8. Set the text to something like "Then let's get started!"
9. Create a Say Text box
10. Make connections as shown in the screenshot

To play, NAO stretches its legs right out and leans forward slightly so that it can reach the ball. You need to set the following posture with the help of the Timeline box:

Image 2.4.3.3.1 - NAO's posture

The Starting Posture timeline



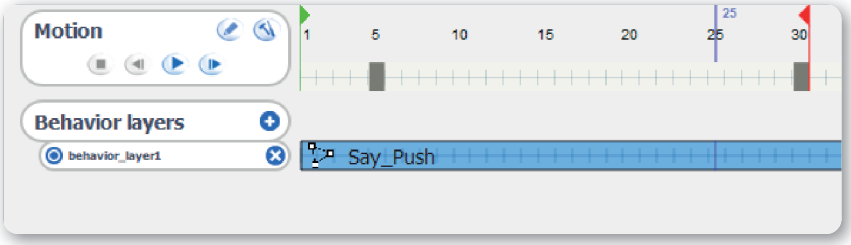
Screen 2.4.3.4 - The Assume Position timeline

Method:

1. Switch on the Animation Mode
2. Put NAO into a suitable position: Legs stretched out, body leaning forward slightly, arms down (see the photo "NAO's posture")
3. Save the position of the whole body (head, arms, and legs) in a keyframe

This is roughly what your Starting Posture timeline should look like. The only keyframe in this timeline is NAO's posture. In this step you need to make sure that you save the position of the whole body. You can do this by pressing the F8 key on the keyboard, for example.

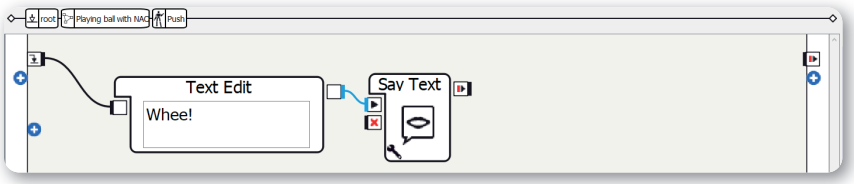
The Roll timeline



Screen 2.4.3.5 - The Roll timeline

Method:

1. Switch on Animation Mode
2. Click on the keyframe from the Assume Position timeline so that NAO is in the right position
3. Save only the position of the arms at frame number 30
4. Lift the arms up so that NAO can roll the ball
5. Save only the position of the arms at frame number 5
6. The Say_Push keyframe:



Screen 2.4.3.5.1 - The Say_Roll keyframe

a. Create a Text Edit box
b. Set the text to something like "Whee!"
c. Create a Say Text box
d. Make connections as shown in the screenshot

In this timeline NAO rolls the ball away and over the ramp. You may need to try out and adjust the animation several times here. You can save the position of the arms by pressing the F10 key on the keyboard, for example.

You can test whether the animation works using the triangle under "Motion". Here are a few tips:

- If NAO is not leaning forward far enough, for example, you don't need to replace both keyframes. Instead you can simply alter the starting position in the Start Condition diagram.
- If NAO doesn't roll the ball hard enough, you can try moving the keyframe with the raised arms 1-3 frames to the left. This means that the movement is performed more quickly.
- If NAO starts the animation too soon, you can make NAO look further down in the Assume Position timeline or extend the waiting time in the Wait box.
- If NAO doesn't detect the ball at all, you can try using another red ball, illuminating the ball, or making NAO look a little higher.

5 NAO detects a red ball

Overview

Thanks to artificial intelligence, many things can now be simulated. In the future, it may even be possible to simulate a human brain. For example, you can teach a computer, like a small child, which objects have which names and where they are. You can try this out in a very simplified form with your NAO!

You can use the program "NAO detects a red ball" to make NAO detect where a red ball is located in relation to NAO.

Objects required and preparation

Image 2.5.2.1 - Objects required

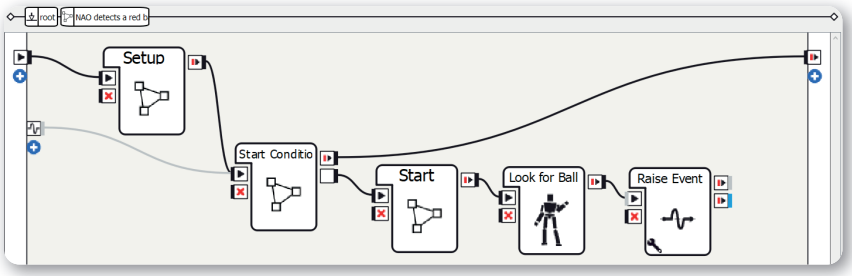
For this project you need:

- A red ball (minimum diameter: 5 cm)
- See-through cups with a large diameter
  or
- Marks on a surface

In this program NAO looks right, forward, and left. So that the other player knows where the ball needs to be placed, you can use cups or marks (distance 20 - 40 cm). The marks/cups should be to the right, in front of, and to the left of NAO. First you should check whether NAO can detect the ball with the bottom camera.

Suggested solution

Basic framework



Screen 2.5.3.1 - Basic framework

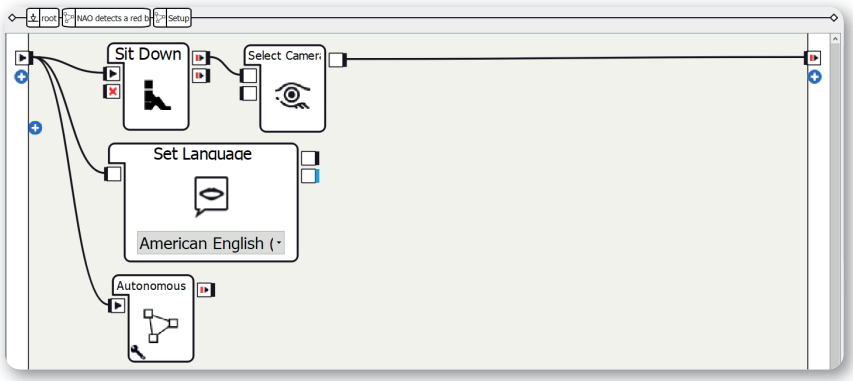
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, and Start
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Create a Timeline box
5. Rename the Timeline box as Look for Ball
6. Create a Raise Event box and set the key parameter to something like "detectball/again"
7. Add an output from ALMemory on the left, name of the event in this suggested solution: "detectball/again"
8. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, the other player positions the ball during the Start diagram.
- In the Look for Ball timeline, NAO looks for the ball.
- Then an event is raised that invokes the Start Condition diagram again.
- This allows you to play the game again.

The Setup diagram

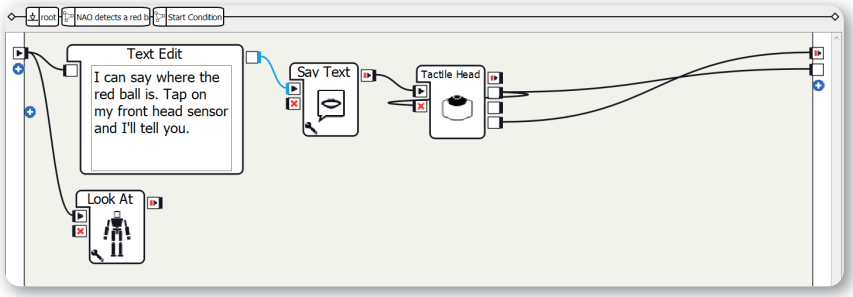


Screen 2.5.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use a Sit Down box
2. Only activate the Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three.

The Start Condition diagram

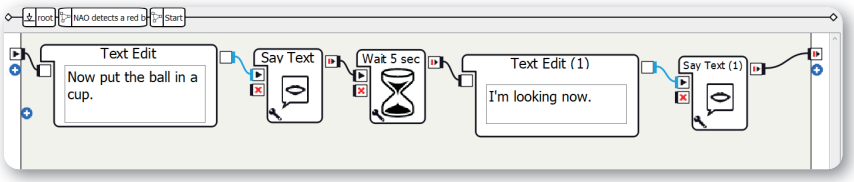


Screen 2.5.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text in the left Text Edit box to something like "I can say where the red ball is. Tap on my front head sensor and I'll tell you."
2. Create a Look At box
3. Set the Z-parameter of the Look At box to 5
4. Make connections as shown in the screenshot

The Start diagram



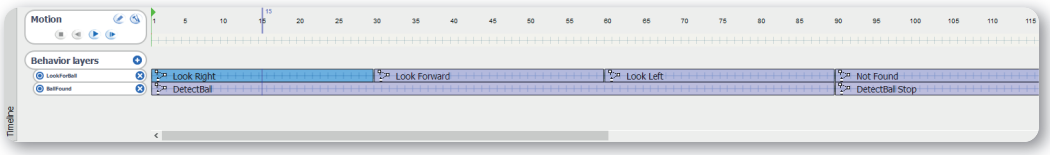
Screen 2.5.3.4 - The Start diagram

Method:

1. Create a Say box
2. Set the text to something like "Now put the ball in a cup."
3. Create a Wait box
4. Set the Timeout parameter of the Wait box to 5
5. Create a Say box
6. Set the text to something like "I'm looking now."
7. Make connections as shown in the screenshot

In this diagram NAO says that the other player can put the red ball on one of the three marks or cups. After waiting about 5 seconds, NAO tells the other player that it is now looking.

The Look for Ball timeline

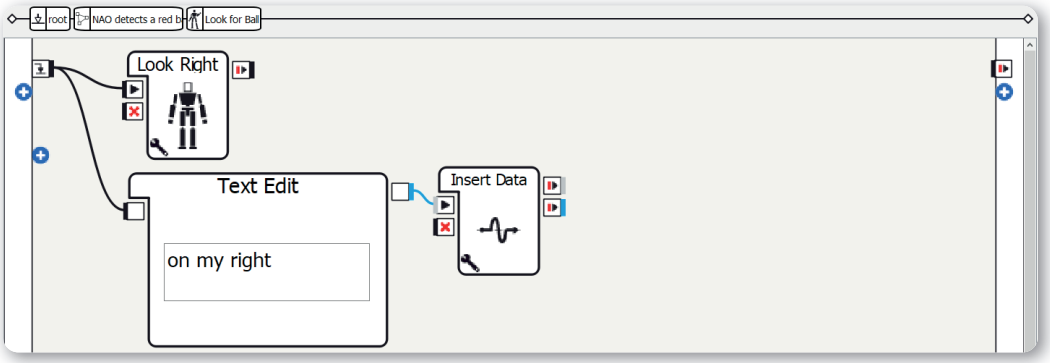


Screen 2.5.3.5 - The Look for Ball timeline

Method:

1. On the left, rename the existing behavior layer to something like LookForBall
2. Add three keyframes to LookForBall
3. Rename the keyframes as: Look Right, Look Forward, Look Left, Not Found

4. The Look Right keyframe:



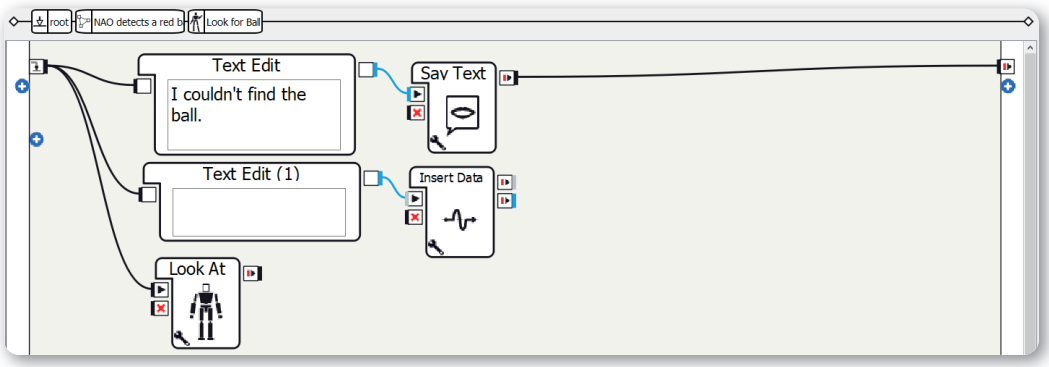
Screen 2.5.3.5.1 - The Look Right keyframe

a. Create a Look At box
b. Set the parameters: X = 1; Y = -10; Z = -3
c. Create a Text Edit box
d. Set the text to "on my right"
e. Create an Insert Data box
f. Set the key of the Insert Data box to something like "detectball/direction"
g. Make connections as shown in the screenshot

5. The Look Forward keyframe:
   a. The structure is the same as for the previous keyframe
   b. Set the parameters of the Look At box: X = 1; Y = 0; Z = -3
   c. Set the text of the Text Edit box to "in the middle"

6. The Look Left keyframe:
   a. The structure is the same as for the previous keyframe
   b. Set the parameters of the Look At box: X = 1; Y = -10; Z = -3
   c. Set the text of the Text Edit box to "on my left"

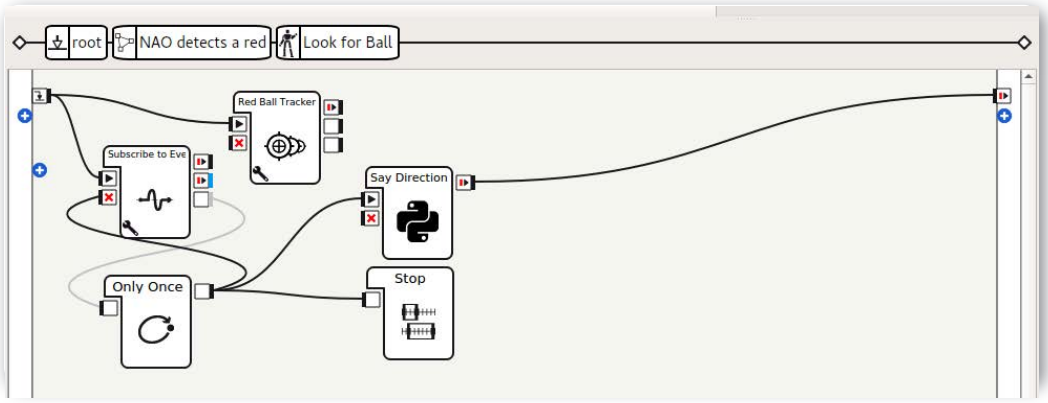
7. The Not Found keyframe:



Screen 2.5.3.5.2 - The Not Found keyframe

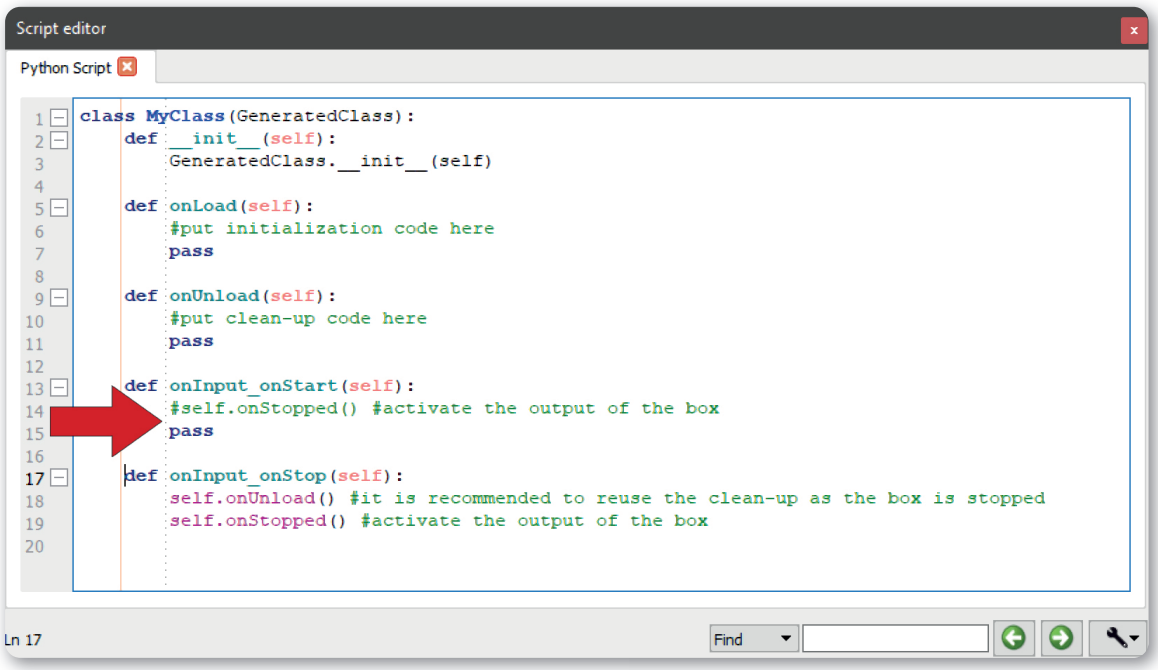
a. Create a Text Edit box
b. Set the text to something like "I couldn't find the ball."
c. Create a Say Text box
d. Create another Text Edit box
e. Create an Insert Data box
f. Use the same key as in the previous keyframes
g. Create a Look At box
h. Set the parameters: X = 1; Y = 0; Z = 5
i. Make connections as shown in the screenshot

8. On the left, add a behavior layer and rename it to something like BallFound
9. Add the DetectBall Stop keyframe to the BallFound behavior layer
10. Now you have two keyframes: DetectBall and DetectBall Stop
11. The DetectBall keyframe:



Screen 2.5.3.5.3 - The DetectBall keyframe

a. Create a Subscribe to Event box
b. Create a Red Ball Tracker
c. Set the key to redBallDetected
d. Create an Only Once box
e. Create a Stop box
f. Create a Python Script box



Screen 2.5.3.5.4 – The Python Script box

f. Add the following source code to the method "onInput_onStart(self)":

```
def onInput_onStart(self):
    self.mem = self.session().service("ALMemory")
    self.tts = self.session().service("ALTextToSpeech")
    direction = self.mem.getData("detectball/direction")
    text = "This ball is " + direction
    self.tts.say(text)
    self.onStopped()
```

g. Make connections as shown in the screenshot

12. DetectBall Stop keyframe: This keyframe remains empty, don't add anything
13. Set the keyframes Not Found and DetectBall Stop to the same frame number (directly above one another)

The timeline makes NAO look right, forward, and left. If NAO sees a red ball, it immediately stops looking and says where the ball is.

NOTES

6 NAO plays rock-paper-scissors

Overview

Digital playmates in kindergarten – is it a good thing? Will robots be able to replace human pre-school teachers in the future? You can take one small step toward this vision yourself.

Using the program "NAO plays rock-paper-scissors" you can use Naomarks to play rock-paper-scissors with your NAO.

Objects required and preparation

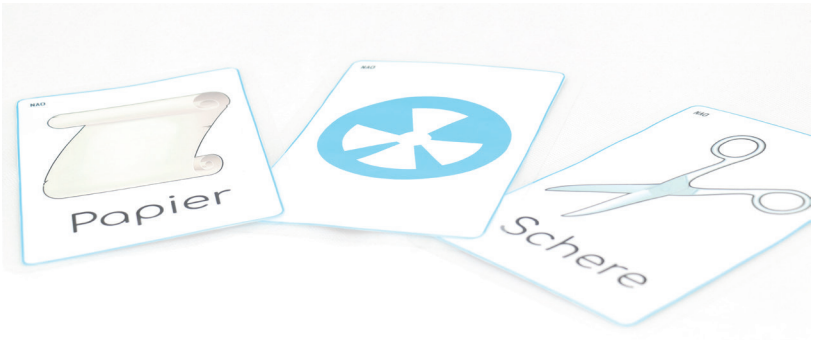


Image 2.6.2.1 - Objects required

For this project you need:

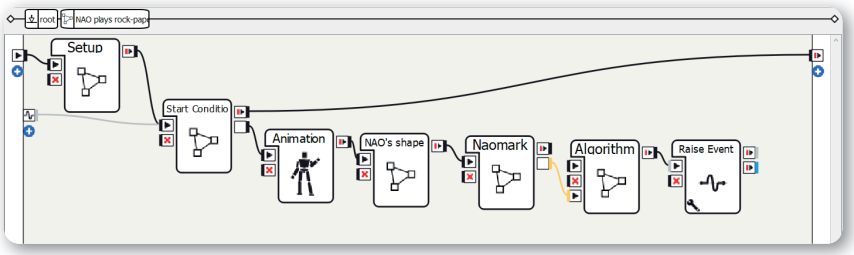
- Printed Naomarks (recommendation: A5 format)
- Printed or drawn symbols for rock, paper, and scissors

First you need to assign a Naomark to each symbol (rock, paper, scissors). It doesn't matter which Naomark you choose. It's a good idea to note down the pairs as they are important later on when you are programming. In this suggested solution Naomark number 64 is used for scissors, 80 for rock, and 119 for paper.

Now print out the Naomarks and glue or draw the respective symbols onto the backs of them. It makes sense to laminate the Naomarks so that you can reuse them.

Suggested solution

Basic framework



Screen 2.6.3.1 - Basic framework

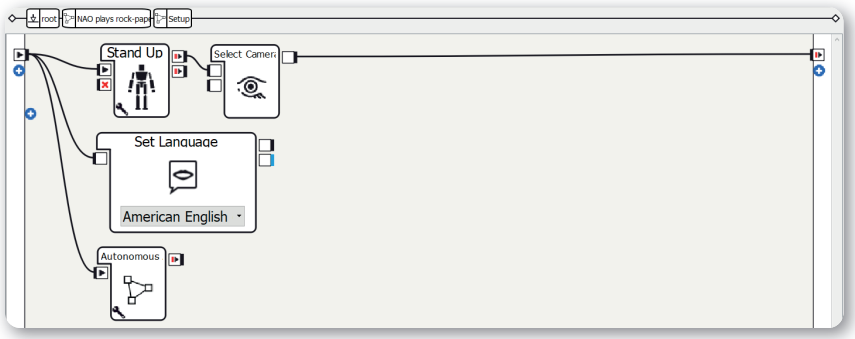
Method:

1. Create five Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, NAO's shape, Algorithm, and Naomark
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Create a Timeline box and set the name as Animation
5. Create an output with the data type "Number" to the Naomark diagram
6. Add an input with the data type "Number" and the nature "Start" to the Algorithm diagram
7. Create a Raise Event box and set the key parameter to something like "rockpaperscissors/again"
8. Add an output from ALMemory on the left, name of the event in this suggested solution: "rockpaperscissors/again"
9. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, the shape is determined at random by NAO and the other player's shape is detected.
- Finally, the Python Script box determines whether the other person won or lost.
- Then an event is raised that invokes the Start Condition diagram again.
- This allows you to play the game again.

The Setup diagram

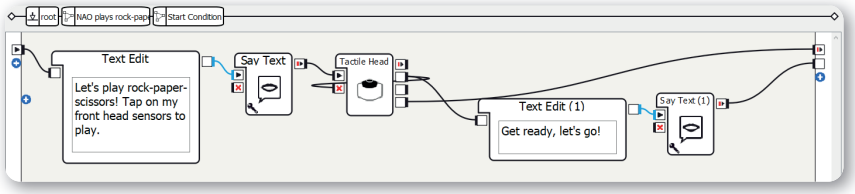


Screen 2.6.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram



Screen 2.6.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

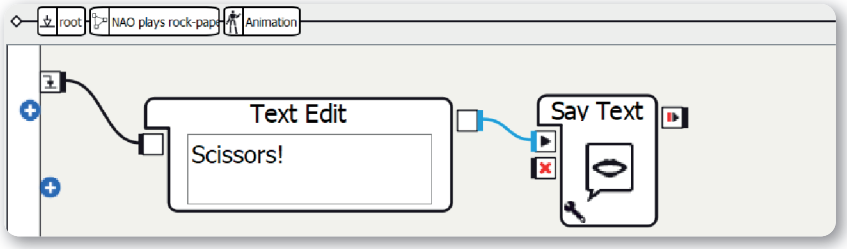
1. Set the text of the left Text Edit box to something like "Let's play rock-paper-scissors! Tap on my front head sensor to play."
2. Set the text of the right Text Edit box to something like "Get ready, let's go!"

The Animation timeline

Screen 2.6.3.4 - The Animation timeline

Method:

1. Rename the first keyframe as "Rock"
2. Set the start of the first keyframe to frame 25
3. Create two more keyframes: "Paper" at frame 50 and "Scissors" at frame 75
4. Now you have three keyframes: Rock, Paper, Scissors
5. The Scissors keyframe:



Screen 2.6.3.4.1 - The Scissors keyframe

   a. Create a Text Edit box
   b. Set the text to "Scissors!"
   c. Create a Say Text box
   d. Make connections as shown in the screenshot

6. The Rock and Paper keyframes:
   a. Make these the same as the Scissors keyframe, only with the text "Rock" and "Paper"

Now you need to create the keyframes for the animation and set them accordingly:

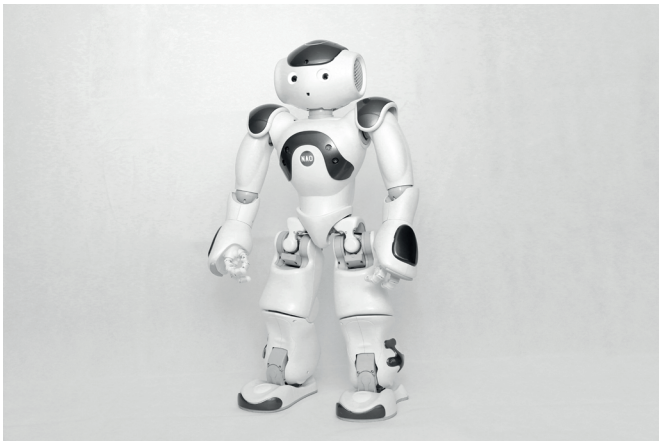


Image 2.6.3.4.2 - Standing position

Posture Stand at frame 100

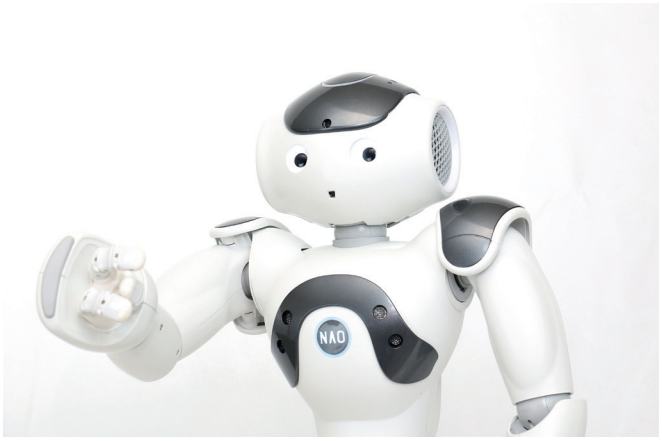


Image 2.6.3.4.3 - Raised hand

Raised hand at frames
25, 50, and 75

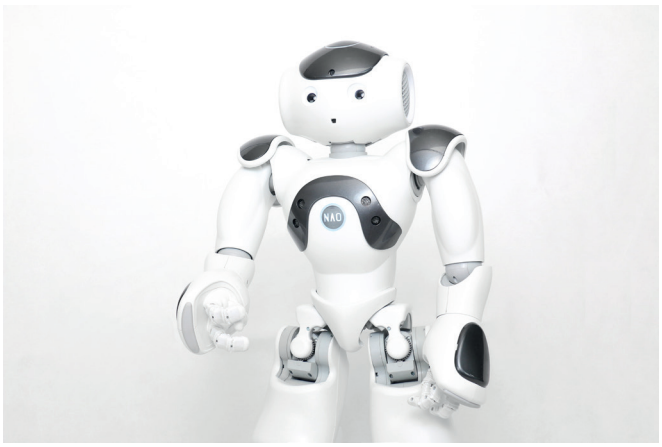
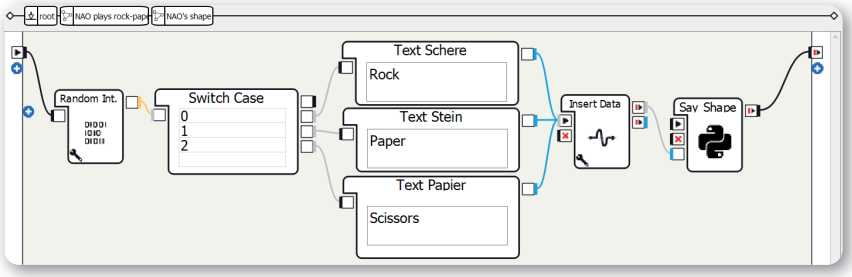


Image 2.6.3.4.3 - Lower hand

Lower hand at frames
35, 60, and 85

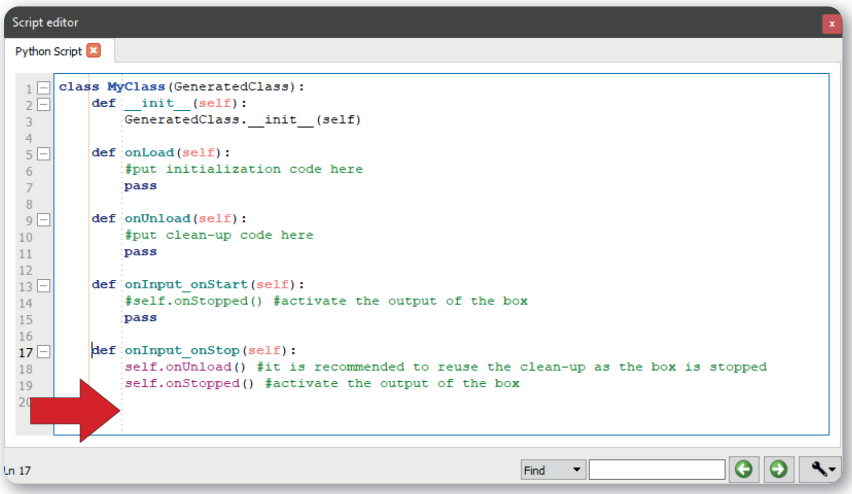
The Choose NAO's shape diagram



Screen 2.6.3.5 - The Choose Symbol diagram

Method:

1. Create a Random Int. box
2. Set the parameters of the Random Int. box: "Min value" to 0 and "Max value" to 2
3. Create a Switch Case box and create three cases: 0; 1; 2
4. Assign symbols to these three numbers, here: 0 ≙ Rock; 1 ≙ Paper; 2 ≙ Scissors
5. Add an Insert Data box and set the key parameter to "rockpaperscissors/naomark"
6. Create a Python Script box
7. Add a Naomark input with the data type "String" to the Python Script box



Screen 2.6.3.6 - The Python Script box

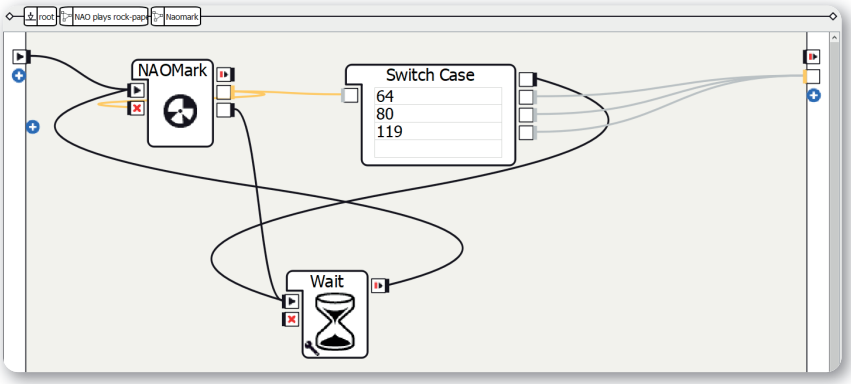
8. Create the method onInput_Naomark(self, p) and add the following source code:

```
def onInput_Naomark(self, p):

    self.tts=self.session().service("ALTextToSpeech")
    self.tts.say("I have " + p + ".")
    self.onStopped()
```

9. Make connections as shown in the screenshot

NAO's shape is generated randomly here and saved under the key rockpaperscissors/naomark. Then NAO says which shape it has so that the other player can be sure that NAO is not a cheat.

The Naomark diagram



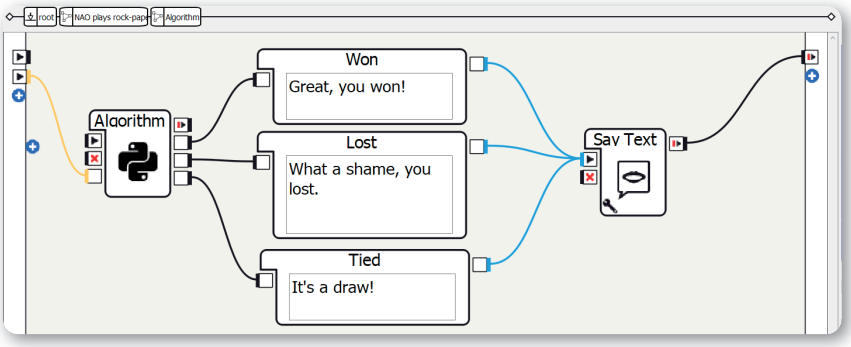
Screen 2.6.3.7 - The Naomark diagram

Method:

1. Create a NAOMark box
2. Create a Switch Case box and insert the three Naomark symbols that you chose
3. Create a Wait box and set the "Timeout" parameter to something like 0.5 seconds
4. Make connections as shown in the screenshot

Here, the other player's shape is detected and NAO checks whether the Naomark number detected belongs to the Naomark numbers you specified before. If this is not the case, the Naomark is not accepted. If it is one of them, we proceed to the next box.

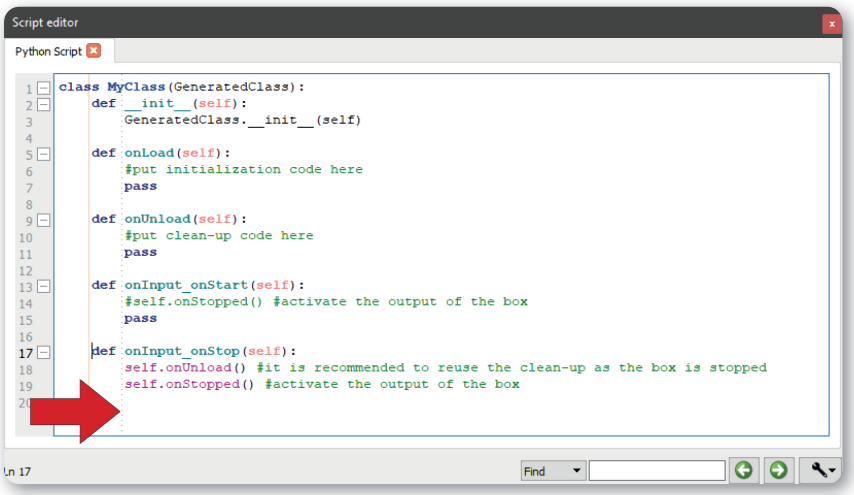
The Algorithm diagram



Screen 2.6.3.8 - The Algorithm diagram

Method:

1. Create a Python Script box
2. Add a Naomark input with the data type "Number" to the Python Script box
3. Add three outputs "Won", "Lost", and "Drawn" with the data type "Bang" to the Python Script box



Screen 2.6.3.9 - The Python Script box

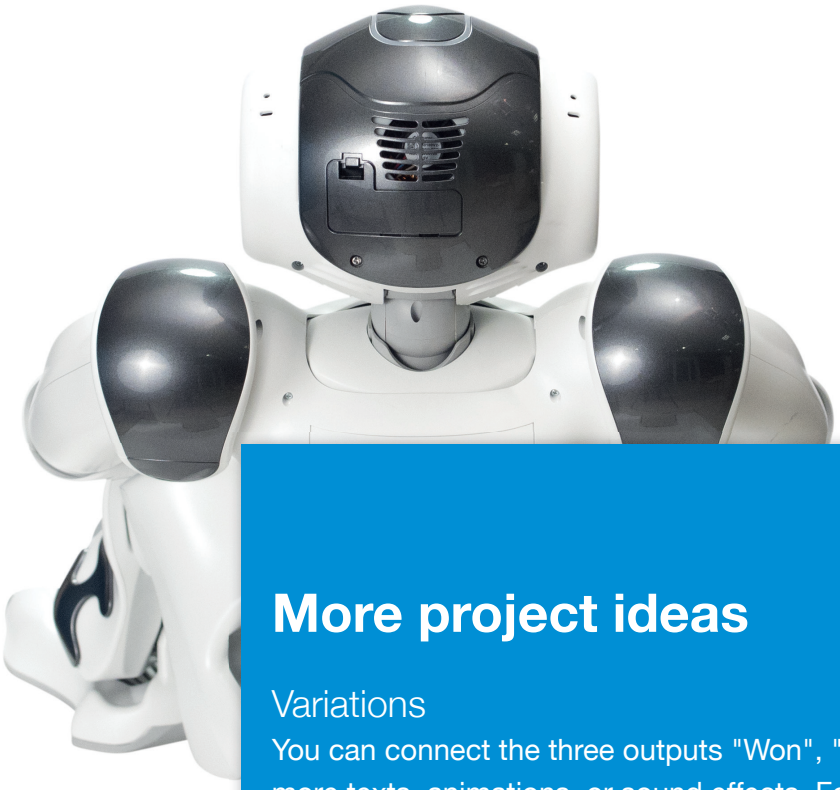
4. Create the method "onInput_Naomark(self, p)" and add the following source code:

```
def onInput_Naomark(self, p):
    self.memory = self.session().service("ALMemory")
    self.naomarkshape = self.memory.getData("rockpaperscissors/
    naomark")

    # Compare Naomark input with NAO's shape
    self.naomark = p

    if self.naomark == 64: # Naomark number for scissors
          if self.naomarkshape == "Scissors":
               self.drawn()

          elif self.naomarkshape == "Rock":
               self.lost()

          elif self.naomark == "Paper":
               self.won()

     elif self.naomark == 80: # Naomark number for rock
          if self.naomarkshape == "Scissors":
               self.won()

          elif self.naomarkshape == "Rock":
               self.drawn()

          elif self.naomarkshape == "Paper":
               self.lost()

     elif self.naomark == 119: # Naomark number for paper
          if self.naomarkshape == "Scissors":
               self.lost()

          elif self.naomarkshape == "Rock":
               self.won()

          elif self.naomarkshape == "Paper":
               self.drawn()

     self.onStopped()
```

5. Create three Text Edit boxes
6. Set the texts as follows:
    a. "Great, you won!"
    b. "What a shame, you lost."
    c. "It's a draw!"

7. Create a Say Text box
8. Make connections as shown in the screenshot

This Python Script box determines whether the other player won or lost or whether there is a draw. This is then transmitted by the output of "Won", for example, being activated by the method "self.won()".



More project ideas

Variations

You can connect the three outputs "Won", "Lost", and "Drawn" with more texts, animations, or sound effects. For example, NAO could celebrate when the other player has won and play an animation.

7 Gymnastics with NAO

Overview

Will robots be able to take over our jobs in the future, for example in caring for the elderly?

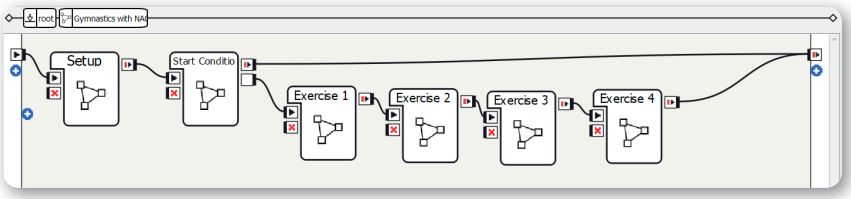
In the project "Gymnastics with NAO", NAO takes on the role of an old people's nurse who can help you do physical exercises every day.

Objects required and preparation

You don't need anything extra for this project.

Suggested solution

Basic framework



Screen 2.7.3.1 - Basic framework

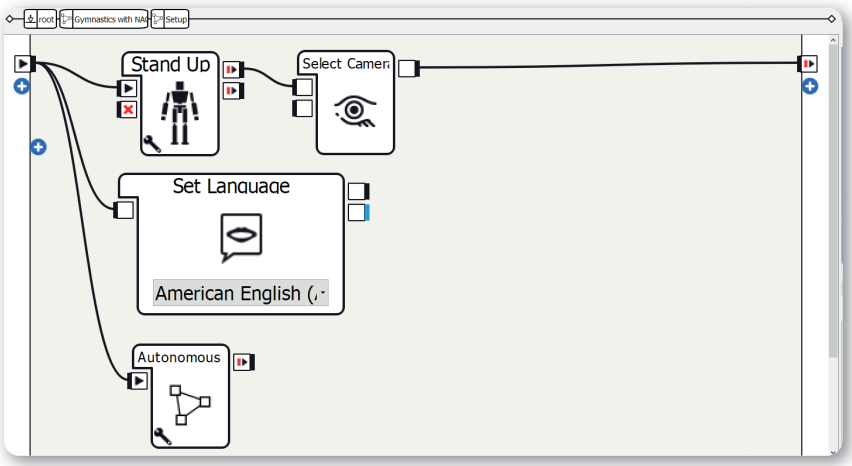
Method:

1. Create two Diagram boxes
2. Rename the Diagram boxes as Setup and Start Condition
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Create a diagram for the template of the exercise sequences
5. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, the exercise sequences are carried out one after the other.
- The exercise and the repetitions are explained in the movement diagrams.
- In the screenshot there are several diagrams for different movements, however it is sufficient for you to create just one diagram to start with.

The Setup diagram

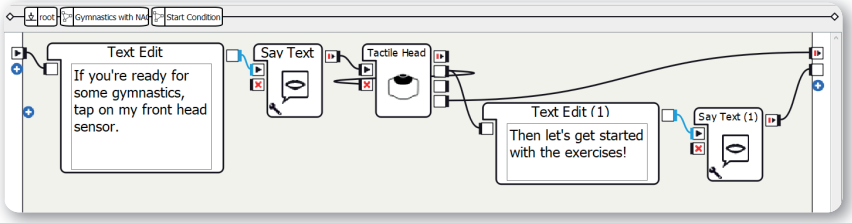


Screen 2.7.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. Only activate the Autonomous Blinking parameter of the Autonomous Abilities box; deactivate the other four

The Start Condition diagram

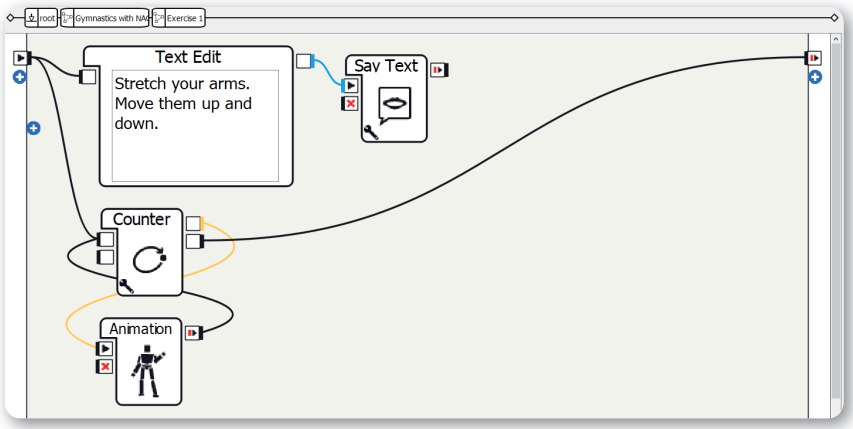


Screen 2.7.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the left Text Edit box to something like "If you’re ready for some gymnastics, tap on my front head sensor."
2. Set the text of the right Text Edit box to something like "Then let's get started with the exercises!"

The Exercise diagram



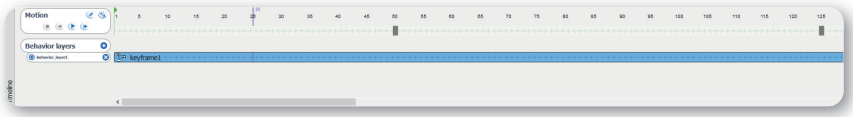
Screen 2.7.3.4 - The Exercise diagram

Method:

1. Create a Text Edit box
2. Set the text to a description of the movement
3. Create a Say Text box
4. Create a Counter box
5. Set the "Final value" parameter to the number of repetitions +1
   (so if you want 5 repetitions, you need to set the parameter to 6)
6. Create a Timeline box
7. Make connections as shown in the screenshot

This is the template for the exercises. You can now copy this diagram as many times as you want to create different exercise sequences.

The Animation timeline



Screen 2.7.3.5 - The Animation timeline

In this timeline you create your exercises using keyframes. You only need to perform the movement once, as it is repeated using the Counter box.

More movement exercises

To create more movements, all you have to do is adjust the Animation timeline and the description.
Here are a few movement examples that are suitable for older people:

Stretch your arms, move them up and down.

Image 2.7.3.6.1 - Arms up

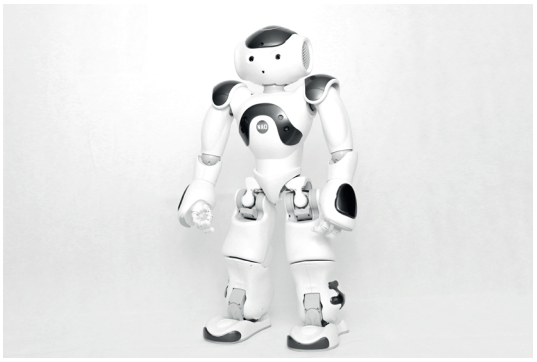


Image 2.7.3.6.2 - Arms down

Put your arms in front of your chest and swing them to the sides.

Image 2.7.3.7.1 - Arms in front of the chest

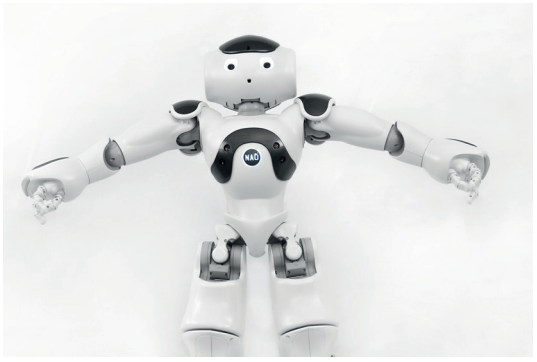


Image 2.7.3.7.2 - Arms to the side

Stretch your arms forward and turn your palms up and down.

Image 2.7.3.8.1 - Palms down

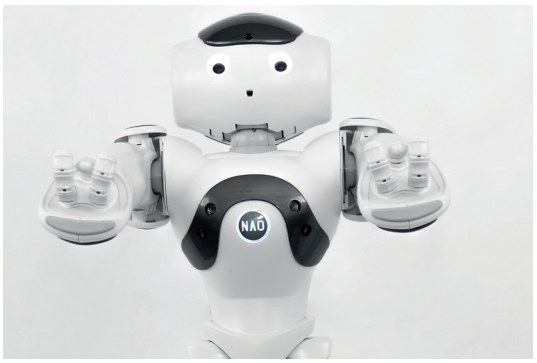


Image 2.7.3.8.2 - Palms up

Bend and stretch your arms up one after the other.
You could describe this movement as "picking apples".

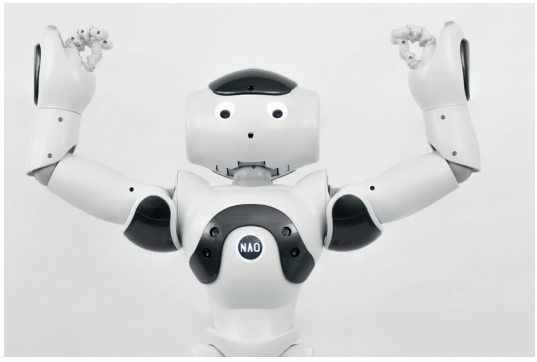


Image 2.7.3.9.1 - Arms to the side

Image 2.7.3.9.2 - Stretch your left arm up

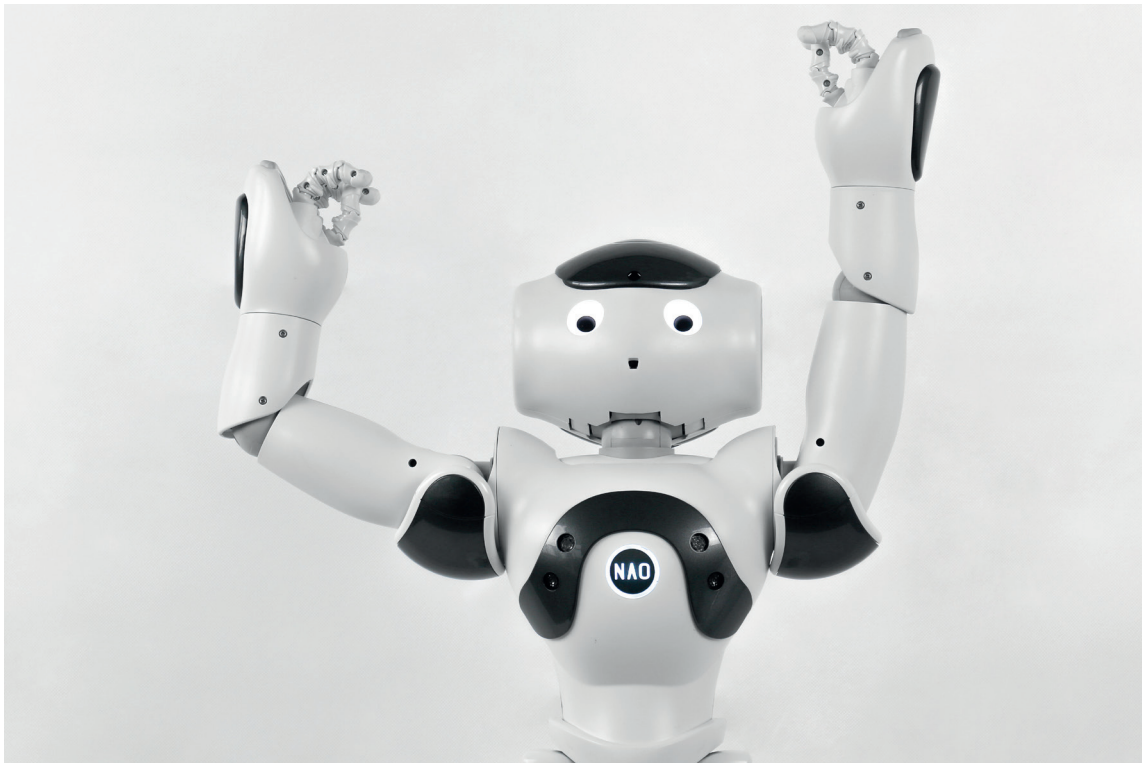


Image 2.7.3.9.3 - Stretch your right arm up

More project ideas

Mirror therapy

Mirror therapy was invented in 1996 by Vilayanur S. Ramachandran. In patients who have had a limb amputated, mirrors are used to reflect the remaining limb, creating the illusion that the amputated limb is still there.
In this program, NAO holds an orange ball in its left hand. The person opposite now has to follow NAO's movements with their own hand.

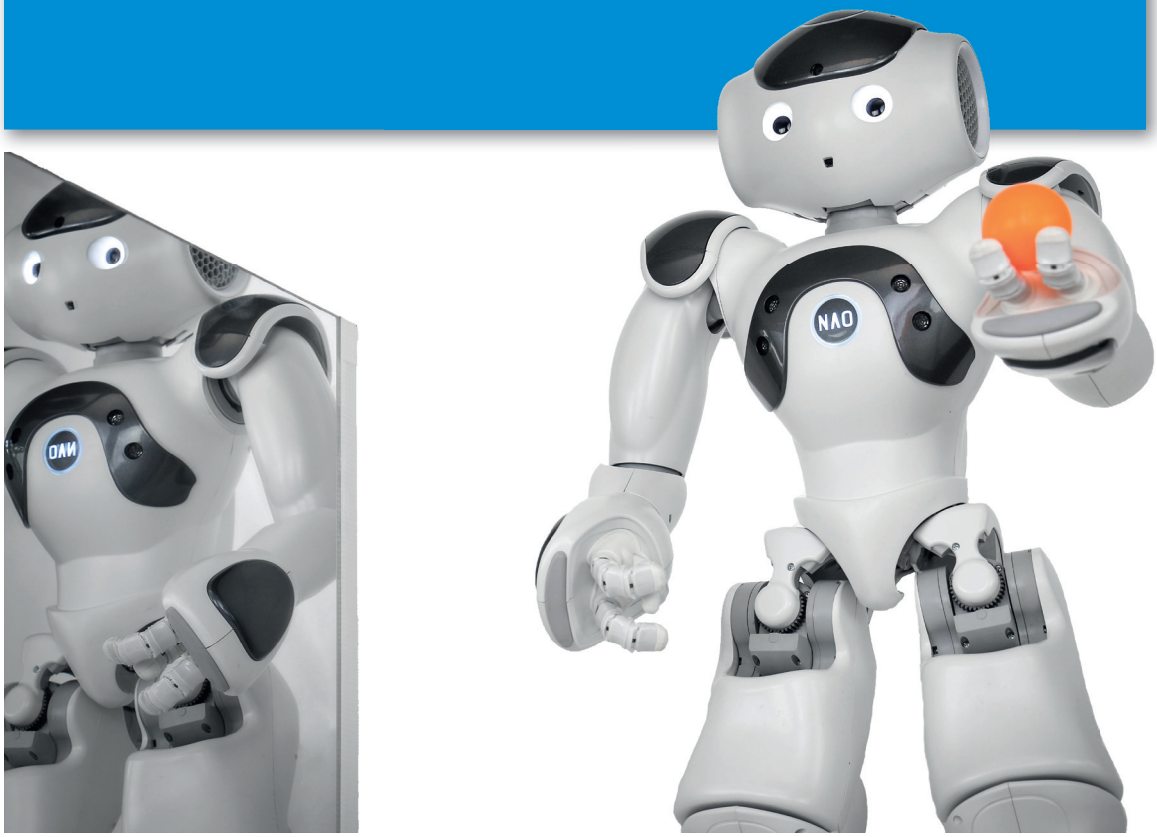


Image 2.7.4.1 - Mirror therapy

8 Steering NAO

Overview

NAO may appear complicated to control at first glance with all those methods and variables, but you can actually control it very easily using a signaling disc.

In the project "Steering NAO" you can use Naomarks to signal to NAO whether it should walk forward or backward. Plus, NAO can turn to the right and left.

Note: NAO could slip and fall on a bad surface (e.g. carpet, uneven, sloping, or slippery surface). Always stay close by so that you can catch NAO at any time if you need to.

Objects required and preparation

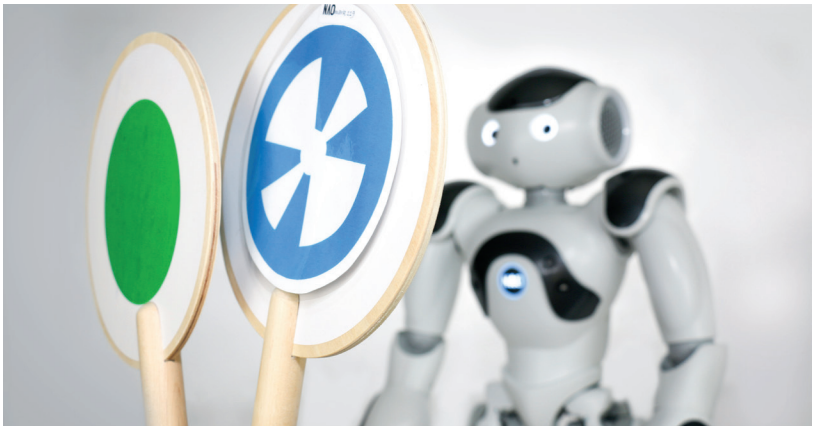


Image 2.8.2.1 - Objects required

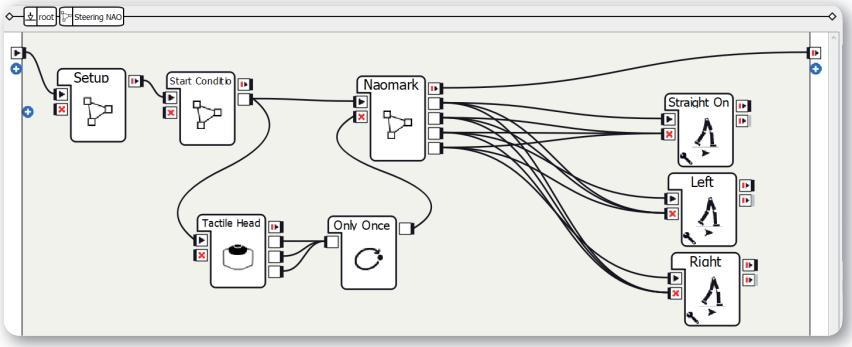
For this project you need:

- Printed Naomarks (recommendation: A4 format)
- Optional: Signaling discs

Naomarks are used to control NAO in this program. To do this, choose four different Naomarks and take note of the Naomark numbers, which are used for programming later. Now print them out. You can go one step further and stick the Naomarks to signaling discs. If you want, you can also laminate them so that you can use them again.

Suggested solution

Basic framework



Screen 2.8.3.1 - Basic framework

Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, and Naomark
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Add four outputs with the data type "Bang" to the Naomark diagram:
    a. "Go"
    b. "Stop"
    c. "TurnLeft"
    d. "TurnRight"
5. Create a Tactile Head box
6. Create an Only Once box
7. Create three Move To boxes
8. Rename the Move To boxes to Straight On, Left, and Right
9. Set the parameters of the Straight On box
    a. Distance X: 2
    b. Distance Y: 0
    c. Theta: 0

10. Set the parameters of the Left box:
    a. Distance X: 0
    b. Distance Y: 0
    c. Theta: 180

11. Set the parameters of the Left box:
    a. Distance X: 0
    b. Distance Y: 0
    c. Theta: -180

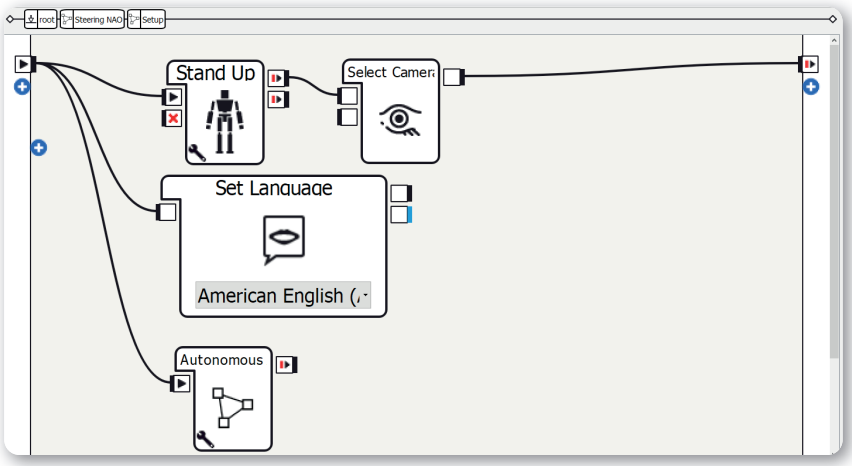
12. Connections from the Naomark diagram to the Move To boxes:
    a. The Go output: Connect to onStart of the Straight On box; connect to onStop of the other two Move To boxes
    b. The Stop output: Connect to onStop of all three Move To boxes
    c. The TurnLeft output: Connect to onStart of the Left box; connect to onStop of the other two Move To boxes
    d. The TurnRight output: Connect to onStart of the Right box; connect to onStop of the other two Move To boxes

13. Make the rest of the connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, NAO looks for Naomarks.
- If it detects one, all other movements are stopped and only the movement associated with the Naomark is carried out.
- "If you want to stop the program, tap on Nao's front or middle head sensor. On this project the Tactil Head box next to the Only Once box will control the program stopping. Touch one of these two sensor will activate the Only Once box that will in turn activate the onStop input of the Naomark box which activates its onStopped output. In fact the onStopped output of the Start Condition box is not linked to anything."

The Setup diagram

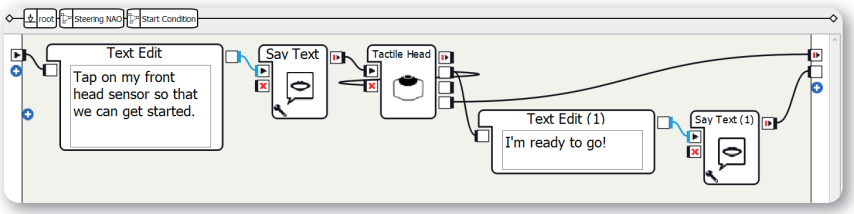


Screen 2.8.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. Only activate the Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three.

The Start Condition diagram

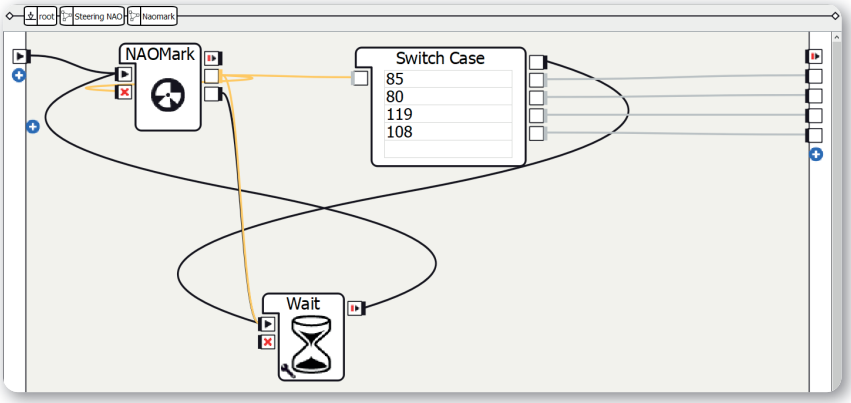


Screen 2.8.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Tap on my front head sensor so that we can get started."
2. Set the text of the right Text Edit box to something like "I'm ready to go!"

The Naomark diagram



Screen 2.8.3.4 - The Naomark diagram

Method:

1. Create a NAOMark box
2. Create a Wait box
3. Set the Timeout parameter to 0.5 seconds, for example
4. Create a Switch Case box
5. Enter the four Naomark numbers you selected into the Switch Case box
6. Make connections as shown in the screenshot

After the program is started, NAO continuously looks for Naomarks. If it finds one, the corresponding movement is carried out or all movements are stopped. If the Naomark number is not in the Switch Case box, NAO looks for a Naomark again.

More project ideas

Obstacle course

A full obstacle course can be built using Naomarks, where NAO can detect and avoid obstacles and complete the course.

NOTES

9 Remember that?

Overview

NAO can be used in old people's homes as entertainment for elderly people and assistance for the nurses. This program has already been tested successfully in an old people's home and the residents really enjoyed it!

In the game "Remember that?" you show NAO a card. NAO detects the object on the card and asks the other player which object it is and whether they can think of a story associated with it.

Objects required and preparation

Image 2.9.2.1 - Objects required

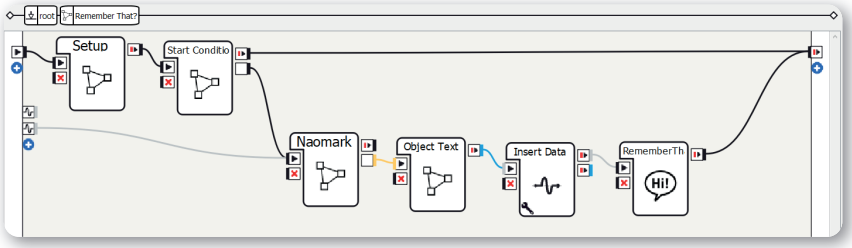
For this project you need:

- Printed out objects
- Printed Naomarks (recommendation: A5 format)

To perform this project, you need at least two different objects assigned to Naomarks.
There are a few sample cards available for you to print out in the virtual archive.
Pay attention to the copyright and licensing rights of the pictures!

Suggested solution

Basic framework



Screen 2.9.3.1 - Basic framework

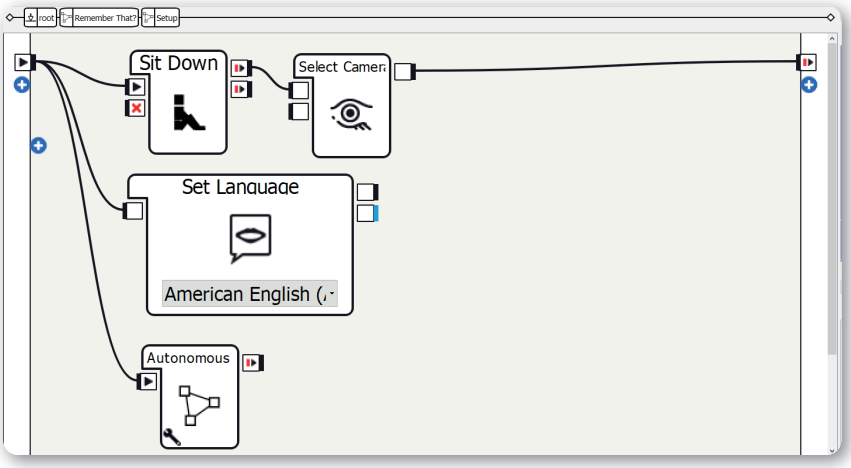
Method:

1. Select the relevant language on the left from among the supported languages in the Project Properties
2. Create four Diagram boxes
3. Rename the Diagram boxes as Setup, Start Condition, Naomark, and Object Text
4. Add an output with the data type "Bang" to the Start Condition diagram
5. Add an output with the data type "Number" to the Naomark diagram
6. On the Object Text diagram, change the data type of the onStart input to "Number"
7. On the Object Text diagram, change the data type of the onStopped output to "String"
8. Create an Insert Data box
9. Set the key parameter of the Insert Data box to something like rememberthat/objecttext
10. Create a new Dialog box with topic in the relevant language
11. Add an output from ALMemory on the left, name of the event in this suggested solution: rememberthat/again
12. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, NAO looks for a Naomark.
- After it has found one it determines which object the Naomark number belongs to.
- The name of the object is saved and in the dialog, NAO holds a conversation with the other player.

The Setup diagram

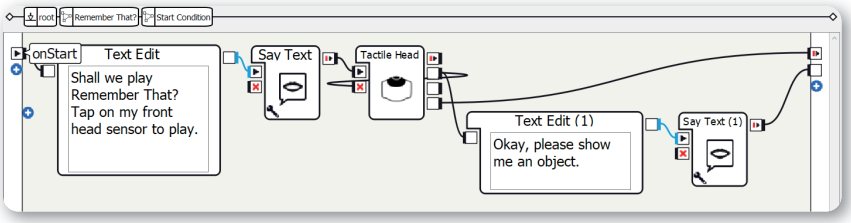


Screen 2.9.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use a Sit Down box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram



Screen 2.9.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Shall we play Remember That? Tap on my front head sensor to play."
2. Set the text of the right Text Edit box to something like "Okay, please show me an object."

The Naomark diagram

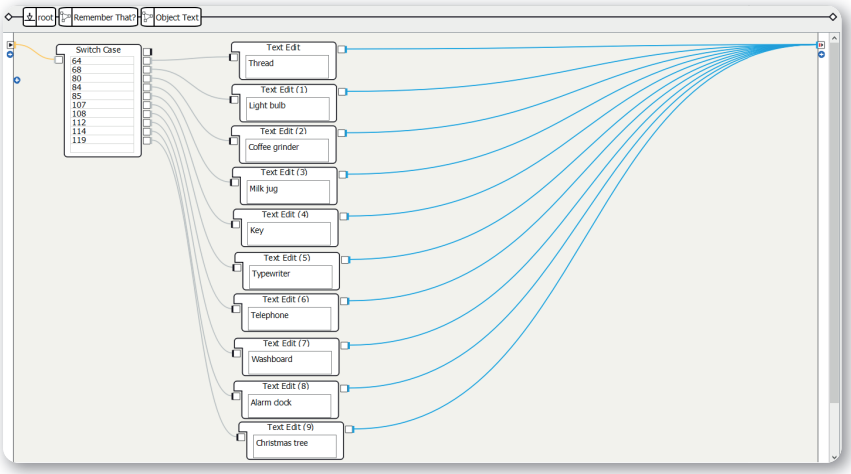
Screen 2.9.3.4 - The Naomark diagram

Method:

1. Create a NAOMark box
2. Create a Wait box
3. Make connections as shown in the screenshot

In this diagram, NAO detects the Naomark number, which is transferred to the next diagram via the output. If no Naomark was found, the NAOMark box is restarted.

The Object Text diagram



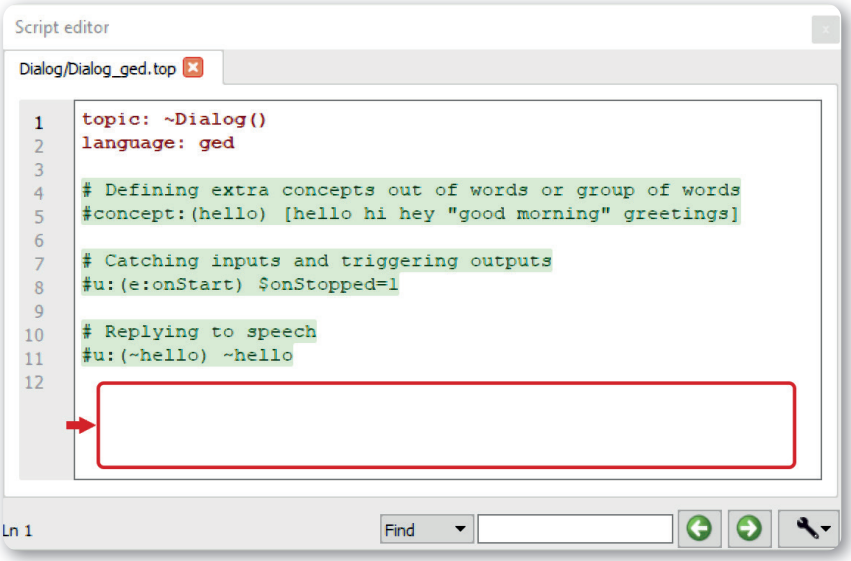
Screen 2.9.3.5 - The Object Text diagram

Method:

1. Create a Switch Case box and for each individual object, create a Text Edit box
2. Write the names of the objects in the respective Text Edit box
3. Enter the assigned Naomark numbers for the objects in the Switch Case box and connect them to the Text Edit box
4. Connect the Text Edit box to the "String" data type output of the diagram

Each individual Naomark number is now assigned to an object. The name of the object is then issued.

The Dialog box



Screen 2.9.3.6 - The Dialog box

Method:

1. Open the .top file and add the following source code:

```
concept: (praise) ^rand[super "very good" "well done!"]
concept: (interesting) ^rand["That was interesting!" "Great story!"]
concept: (yes) [yes yeah gladly definitely]
concept: (no) [no nope "not really"]
concept: (objecttext) [
      "Thread $rememberthat/objecttext==Thread"
      "Lightbulb $rememberthat/objecttext==Lightbulb"
      "Coffeegrinder $rememberthat/objecttext==Coffeegrinder"
      "Milkjug $rememberthat/objecttext==Milkjug"
      "Key $rememberthat/objecttext==Key"
      "Typewriter $rememberthat/objecttext==Typewriter"
      "Telephone $rememberthat/objecttext==Telephone"
      "Washboard $rememberthat/objecttext==Washboard"
      "Alarmclock $rememberthat/objecttext==Alarmclock"
      "Christmastree $rememberthat/objecttext==Christmastree"]
#Detection of the word from the variable

u: (e:onStart) ^nextProposal

proposal: What do you see on the card?
      u1: (~objecttext) ~praise ^nextProposal

proposal: Can you think of a story about a $rememberthat/objecttext
            that you would like to tell me?
      u1: (~yes) Then you can start telling it! When you're done,just
                 tap on my head.
# Wait for a head sensor event
           u2: (e:FrontTactilTouched) ~interesting ^nextProposal
           u2: (e:MiddleTactilTouched) ~interesting ^nextProposal
           u2: (e:RearTactilTouched) ~interesting ^nextProposal
      u1: (~no) That's okay. ^nextProposal

proposal: Do you want to carry on playing or stop?
# The ALMemory-Event "rememberthat/again" is activated here.
      u1: ("continue playing") Okay, then choose a new card and hold it
          in front of me so that I can see the back, please. $rememberthat/
          again=1
      u1: (stop) Okay. Thank you for playing with me!$onStopped=1
```

The Dialog box is used so that NAO says the name of the object detected and asks for a story from the other player's life. If the other player can't think of a story, they are asked if they want to carry on playing or stop.

Note: If you want to add other objects, you need to add them to the source code of the dialog.

More project ideas

Expand the dialog

You can expand the dialog with more questions or information, for example.

NOTES

10 NAO helps tidy up

Overview

Clean and tidy surroundings are important in life. If you need some help with this, NAO can help you out!

In the project "NAO helps tidy up" you can show NAO an object and it tells you which container it needs to go in (for vegetables, fruit, or other).

Objects required and preparation

Image 2.10.2.1 - Objects required 1

Image 2.10.2.2 - Objects required 2

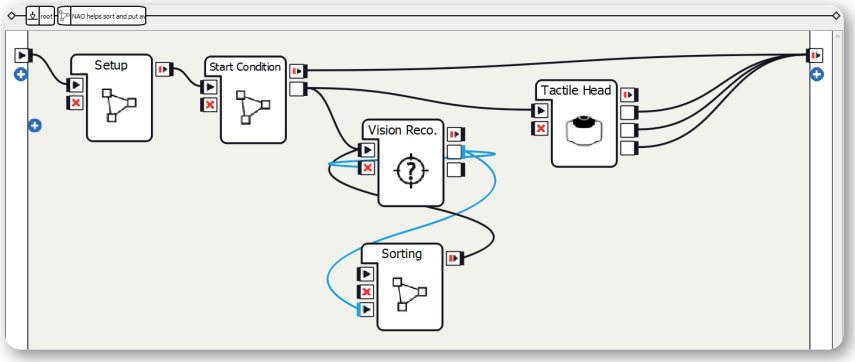
For this project you need:

- Fruit, vegetables, and other objects (also possible with printouts)
- Three containers

First you need to save your objects in a Vision Recognition Database and upload them to NAO. Pay attention to the copyright and licensing rights of the pictures! Learn more at section C.3.1.15 The “Vision Recognition” box of “Learn It”.

Suggested solution

Basic framework



Screen 2.10.3.1 - Basic framework

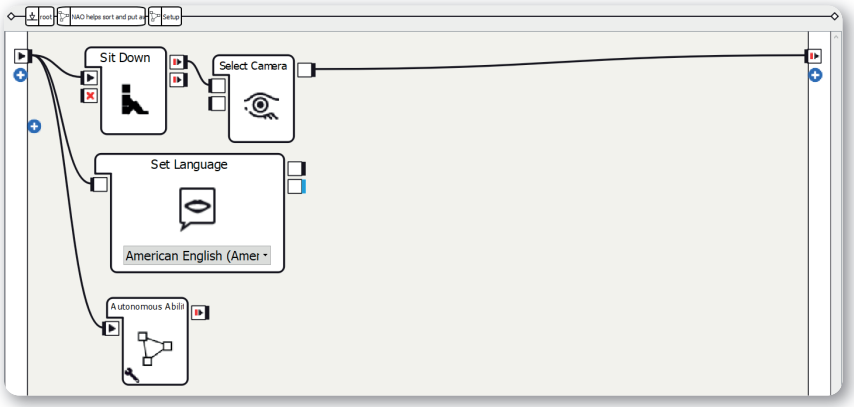
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, and Sorting
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Add an input with the data type "String" and the nature onStart to the Sorting diagram
5. Create a Vision Reco. box
6. Create a Tactile Head box
7. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, the Vision Reco. box is started.
- When an object has been recognized, the Sorting diagram determines whether the object is fruit, vegetable, or other.
- Then the Vision Reco. box is started again.
- You can end the program by tapping on a head sensor.

The Setup diagram

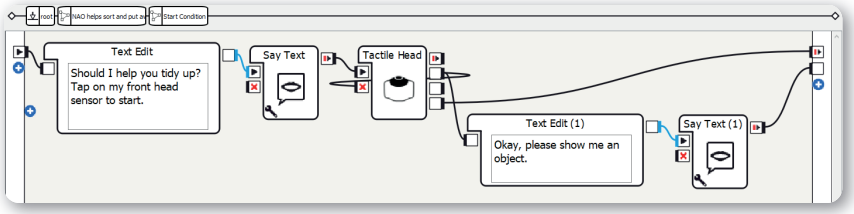


Screen 2.10.3.2 - The Setup diagram

The structure of the Setup diagram is explained in "Structure of the projects". In this project you need to be aware of the following points:

1. Use a Sit Down box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram

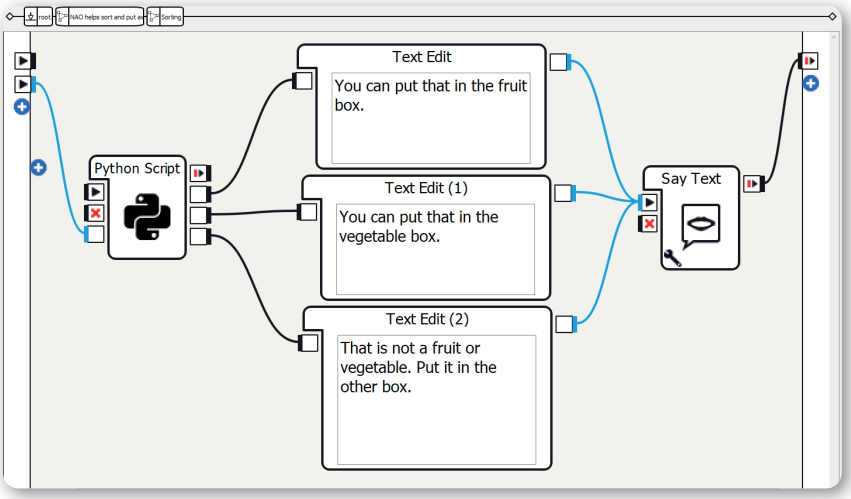


Screen 2.10.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Should I help you tidy up? Tap on my front head sensor to start."
2. Set the text of the right Text Edit box to something like "Okay, please show me an object."

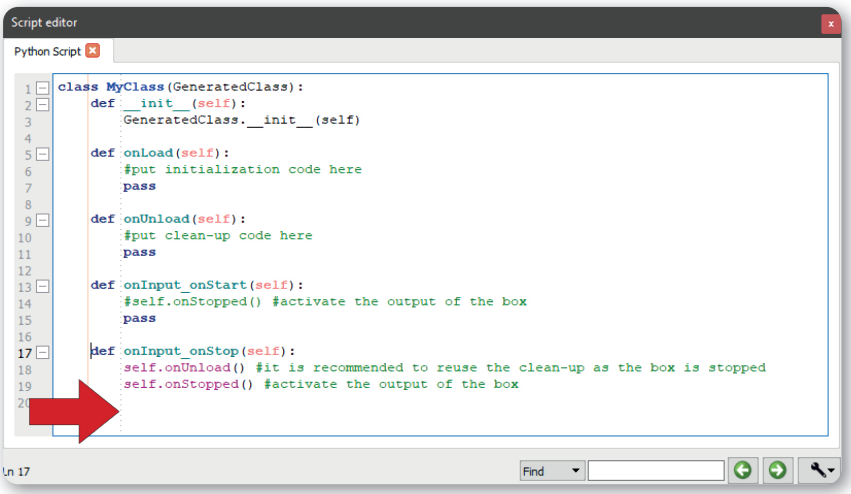
The Sorting diagram



Screen 2.10.3.4 - The Sorting diagram

Method:

1. Create a Python Script box
2. Add an input with the name "objectrecognition" and the data type "String" to the Python Script box
3. Add three outputs with the data type "Bang" to the Python Script box; output names: "fruit", "vegetable", "other"



Screen 2.10.3.4.1 - The Python Script box

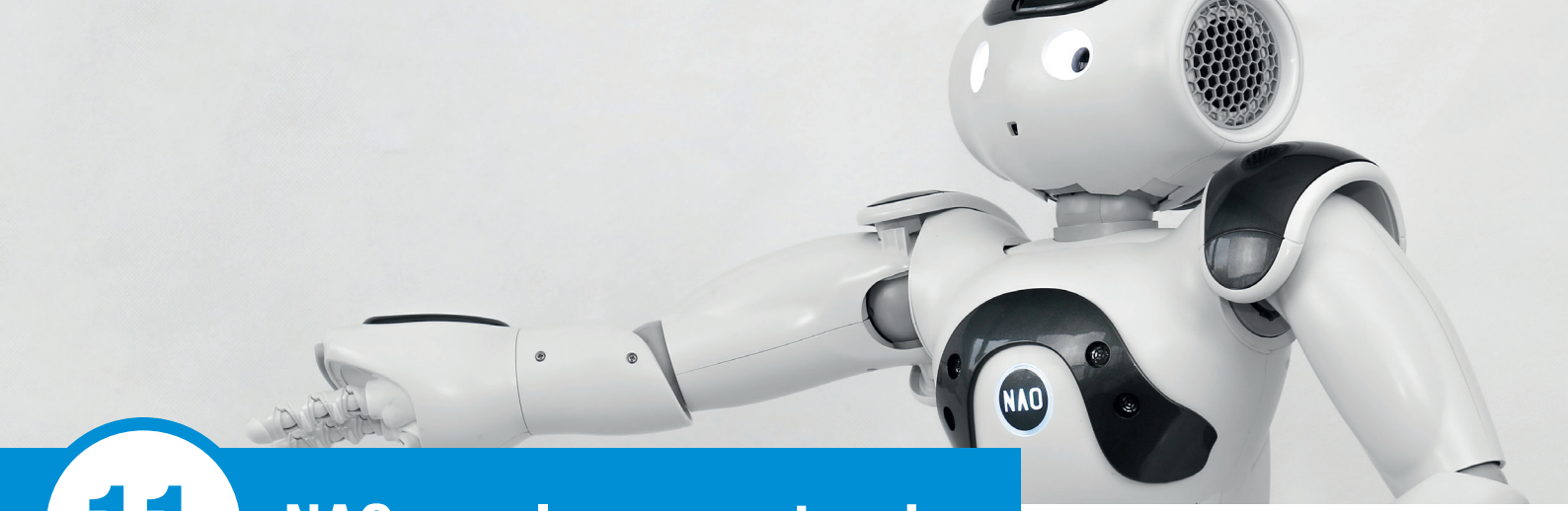
4. Create the method onInput_objectrecognition(self, p) and add the following source code:

```
def onInput_objectrecognition(self, p):
    object = p
    self.tts.say(object)
    time.sleep(1)

    if object == "Apple":
        self.fruit()

    elif object == "Orange":
        self.fruit()

    elif object == "Strawberry":
        self.fruit()

    elif object == "Pepper":
        self.vegetable()

    elif object == "Tomato":
        self.vegetable()

    elif object == "Pumpkin":
        self.vegetable()

    elif object == "Chocolate":
        self.other()

    elif object == "Bread":
        self.other()

    self.onStopped()
    self.onUnload()
```

5. Add three Text Edit boxes and set something like the following texts:
    a. Text for Fruit: "You can put that in the fruit box."
    b. Text for Vegetable: "You can put that in the vegetable box."
    c. Text for Other: "That is not a fruit or vegetable. Put it in the other box."

6. Create a Say Text box
7. Make connections as shown in the screenshot

Explanation:
The Python Script box now determines which category the object should be sorted into.



11 NAO as a language teacher

Overview

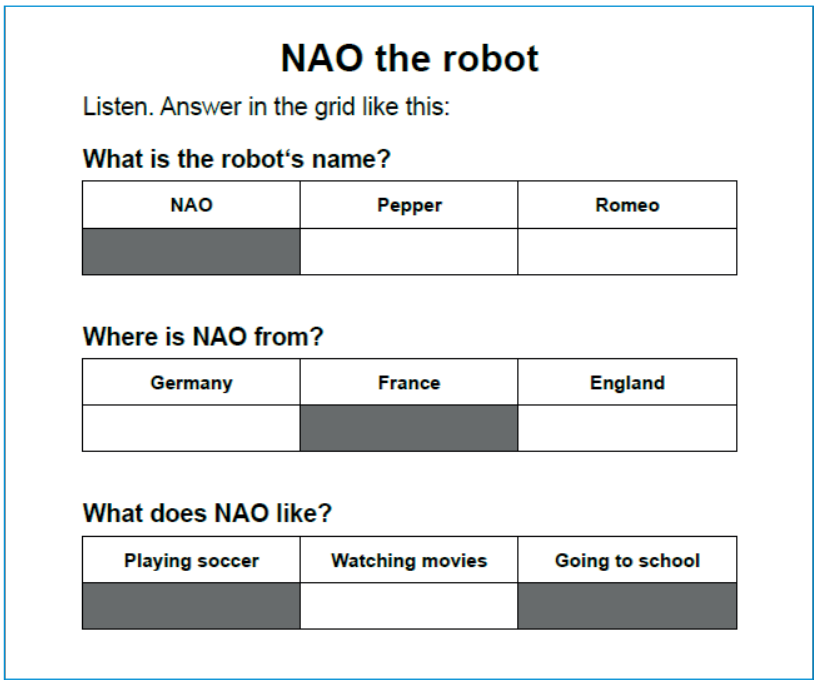
Robots can be used in more and more professions, but can they be as efficient as teachers in teaching languages?

You can try this out and decide for yourself in the program "NAO as a language teacher". Not only can NAO explain vocabulary and grammar, it can also simulate listening comprehension exercises and correct them. You will also learn different possibilities for programming a lesson.

Objects required

For this project you will need:

- German pack installed



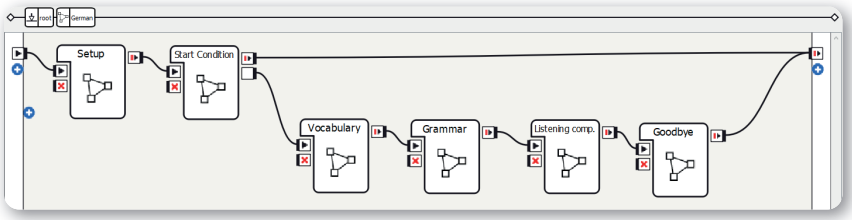
Screen 2.11.2.1 - Objects required

For this project you need:

- A preprepared worksheet (see screenshot "Objects required")

For the lesson it is practical to have worksheets for notes, vocabulary, and listening comprehension. You can design your own handout for your lesson with NAO or use the template from the virtual archive that is used in programming.
To correct the worksheets you can scan the solution with the completed table into a Vision Recognition Database. Uploading Vision Recognition Database to NAO is mandatory!
Refer to section C.3.1.15 The “Vision Recognition” box of “Learn It” book.

Suggested solution



Screen 2.11.3.1 - Basic framework

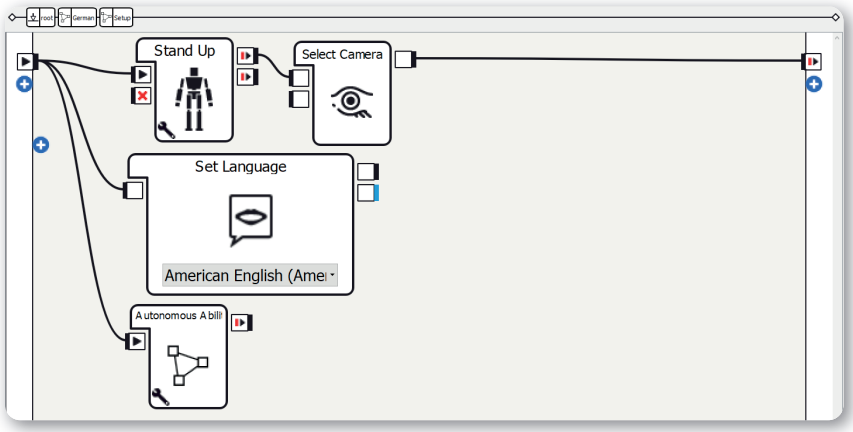
Method:

1. Create six Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, Vocabulary, Grammar, Listening comp., and Goodbye
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, individual lesson modules follow.
- You can change the sequence however you like.
- At the end, NAO says goodbye to the class.

The Setup diagram

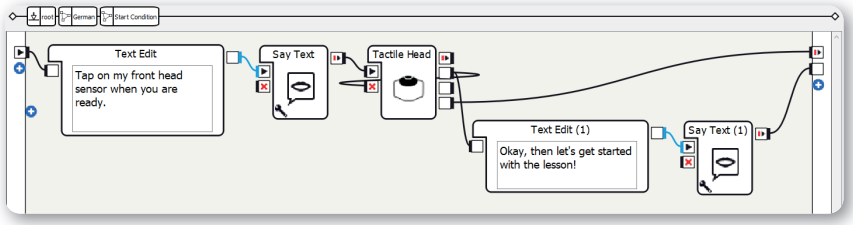


Screen 2.11.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram

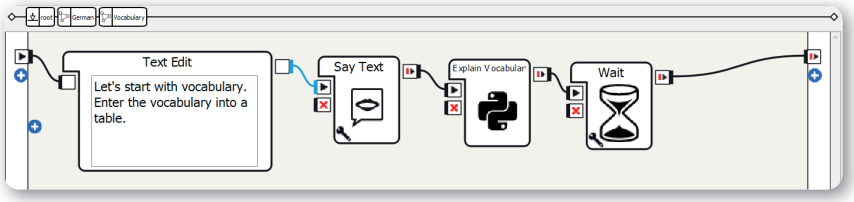


Screen 2.11.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Tap on my front head sensor when you are ready."
2. Set the text of the right Text Edit box to something like "Okay, then let's get started with the lesson!"

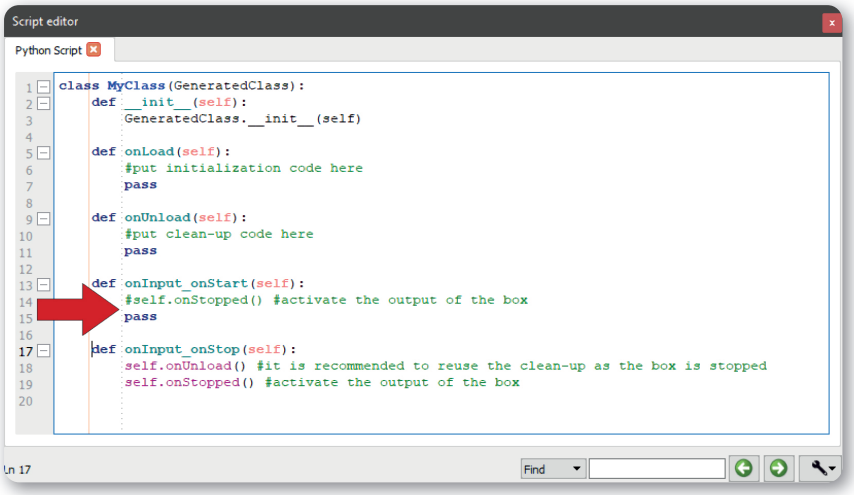
The Vocabulary diagram



Screen 2.11.3.4 - The Vocabulary diagram

Method:

1. Create a Text Edit box
2. Set the text to something like "Let's start with vocabulary. Enter the vocabulary into a table."
3. Create a Say Text box
4. Create a Python Script box



Screen 2.11.3.4.1 - The Python Script box

5. Add the following source code to the method "onInput_onStart(self)":

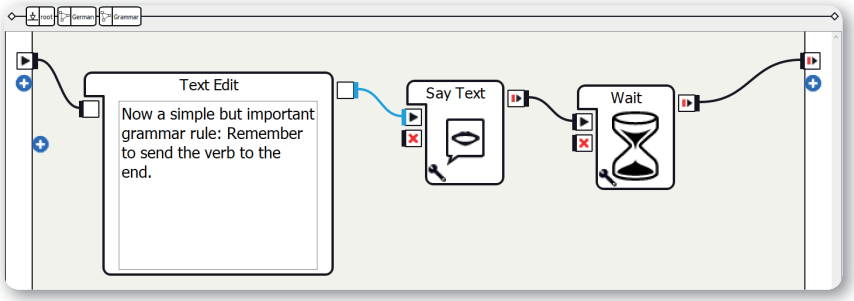
```
def onInput_onStart(self):
    # Preparation
    import time
    tts = self.session().service("ALTextToSpeech")
    tts.setParameter("speed", 70)

    # Vocabulary 1
    tts.setLanguage("German") tts.say("zeichnen")
```


```
tts.setLanguage("English")
tts.say("means to draw in English.")

time.sleep(2)
tts.say("An example sentence would be:")

tts.setLanguage("German")
tts.say("Zeichnen wir ein Bild!")

#Wait
time.sleep(6)

# Vocabulary 2
tts.setLanguage("German")
tts.say("treffen")

tts.setLanguage("English")
tts.say("means to meet in English.")
time.sleep(2)
tts.say("For example, you can say:")

tts.setLanguage("German")
tts.say("Treffen wir uns im Park!")

#Wait
time.sleep(6)

# End
tts.setLanguage("English")
tts.say("That's all the vocabulary for today.")

# Programend
self.onStopped()
```

6. Make connections as shown in the screenshot

Python is used here to say a German word and then the English translation.

The Grammar diagram



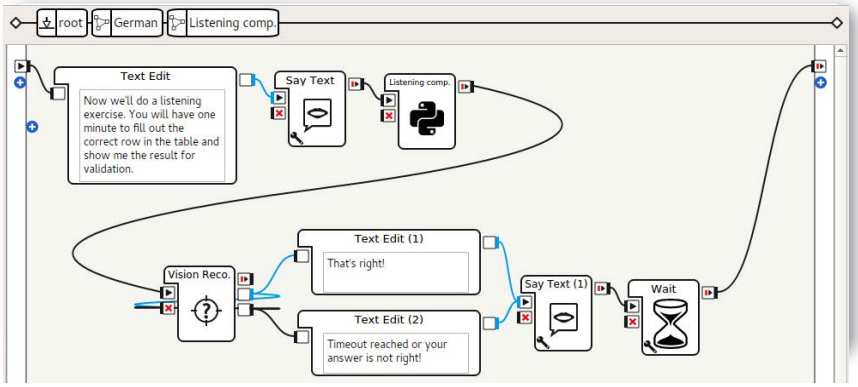
Screen 2.11.3.5 - The Grammar diagram

Method:

1. Create a Text Edit box
2. Set the text to something like "Now a simple but important grammar rule: Remember to send the verb to the end."
3. Create a Say Text box
4. Create a Wait box
5. Set the Timeout parameter to 3 seconds, for example
6. Make connections as shown in the screenshot

In this diagram NAO explains a grammar rule. If you want, you can expand it. If you want to write anything in German, do it phonetically so that you don't have to set NAO's language to German. Otherwise, the German will be pronounced incorrectly.

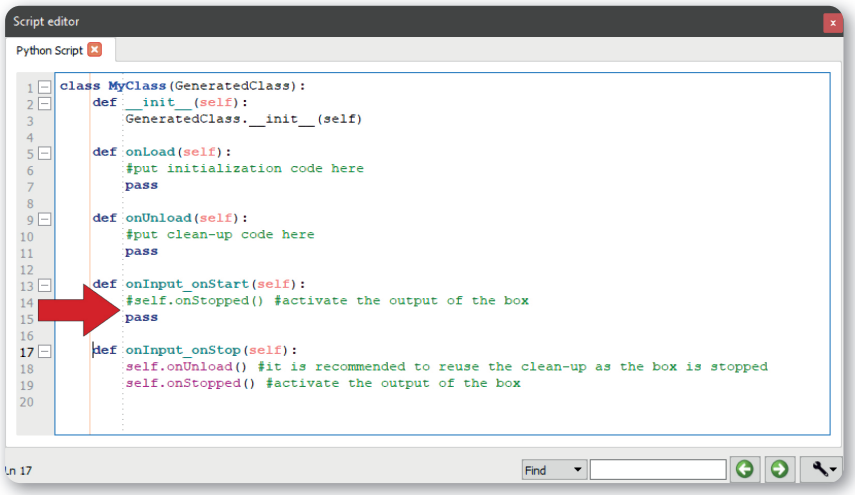
The Listening Comprehension diagram



Screen 2.11.3.6 - The Listening Comprehension diagram

Method:

1. Create a Text Edit box
2. Now we'll do a listening exercise. You will have one minute to fill out the correct row in the table and show me the result for validation.
3. Create a Say Text box
4. Create a Python Script box

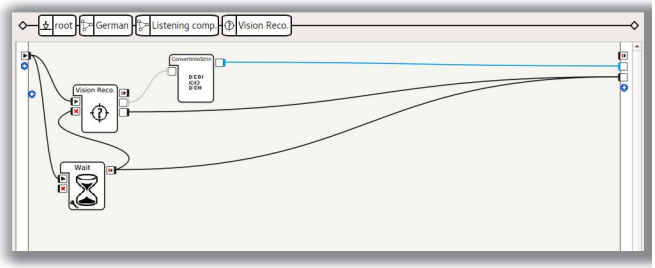


creen 2.11.3.6.1 - The Python Script box

5. Add the following source code to the method onInput_onStart(self):

```
def onInput_onStart(self):
    # Preparations
    import time
    tts = self.session().service("ALTextToSpeech")
    tts.setLanguage("German")
    tts.setParameter("speed", 70)

    # Listening Comprehension
    tts.say("Hallo! Ich heiße NAO.")
    time.sleep(1)
    tts.say("Ich bin ein Roboter!")
    time.sleep(1)
    tts.say("Ich komme aus Frankreich.")
    time.sleep(1)
    tts.say("Ich wohne in Deutschland.")
    time.sleep(1)
    tts.say("Ich spiele gern Fußball und gehe gern in die
    Schule mit meinen Freunden.")
    tts.setLanguage("English")

    # End source code
    self.onStopped()
```

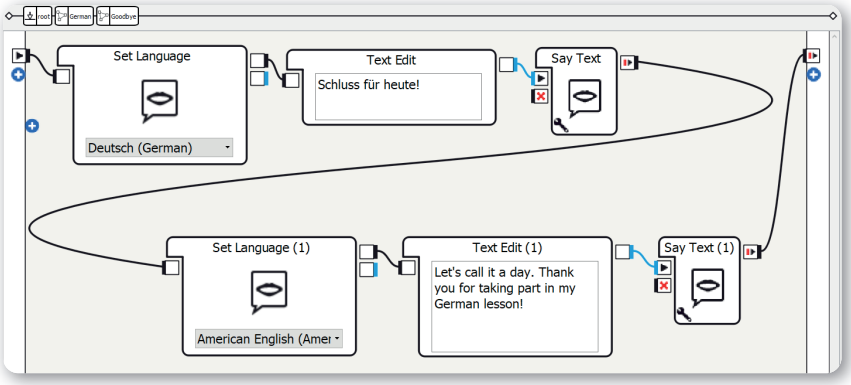
6. Create a Vision Reco. box
   6.1. Open the Vision Reco box and create a Wait box, adjust the Timeout parameter to 60 seconds and make connections as shown on the screenshot on the right



7. Create two Text Edit box
8. Set the text of the text boxes to "That's right" and "Timeout reached or your answer is not right!"
9. Create a Wait box
10. Set the Timeout parameter to 3, for example
11. Make connections as shown in the screenshot

This listening comprehension mostly uses the Python Script box containing the text to be spoken. After the listening comprehension, NAO's object recognition is used to make corrections. To enable this, you first need to save the suggested solutions in a Vision Recognition Database and upload them to NAO. The object recognition only recognizes the sheet if the solution is correct.
Refer to section C.3.1.15 The “Vision Recognition” box of “Learn It” book for more information.

The Goodbye diagram



Screen 2.11.3.7 - The Goodbye diagram

Method:

1. Create a Set Language box
2. Set the language to Deutsch (German)
3. Create a Text Edit box
4. Set the text to "Schluss für heute!"
5. Create a Say Text box
6. Create another Set Language box
7. Set the language to American English
8. Create a Text Edit box
9. Set the text to something like "Let's call it a day. Thank you for taking part in my German lesson!"
10. Create a Say Text box
11. Make connections as shown in the screenshot

At the end of the lesson, NAO says goodbye in German and English.

In this project the content is hard coded. You could improve the code separating the content in a file. This will be presented in project 11 - Nao tells a joke in which the content is read from a csv file."

More project ideas

Expanding the lesson modules

You can use the boxes listed above in any order you like. Of course, you can incorporate variations with several Text Edit boxes, a Random Int box, and a Switch Case box or write a different text for the listening comprehension.

Lesson in other languages

If your NAO has other language packages and you can speak one of these languages, you can design a lesson in this language! The Virtual Archive contains a lesson in Japanese.

12 NAO as a shopping helper

Overview

Major corporations can use big data to give us recommendations for products or suggest alternatives to articles we have searched for. Is NAO also able to help us to shop?

In the project "NAO as a shopping helper", it will help you to shop. First, tell NAO which product you are looking for. Then show NAO a selection of products one after the other until NAO recognizes the product you want and tells you.

Objects required and preparation

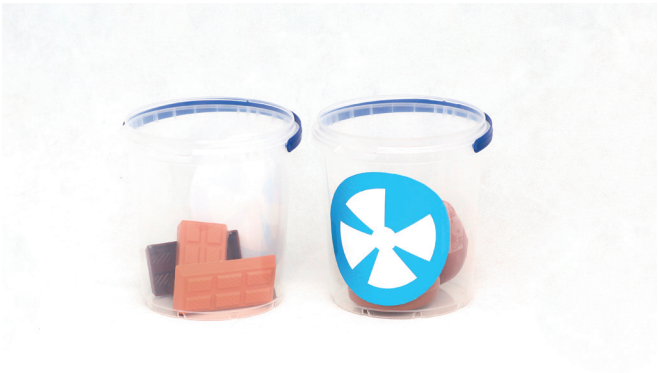


Image 2.12.2.1 - Objects required

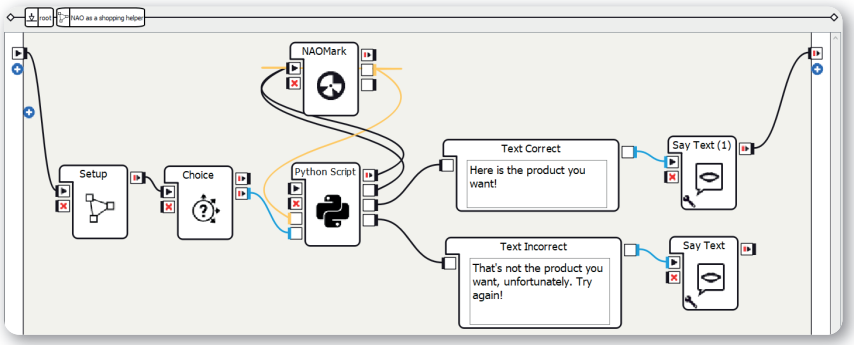
For this project you need:

- At least two objects with Naomarks

Use at least two different objects with Naomarks assigned to them for the project. It is important that you note down the Naomark number. In this suggested solution, 84 is used for donuts and 80 for chocolate.

Suggested solution

Basic framework



Screen 2.12.3.1 - Basic framework

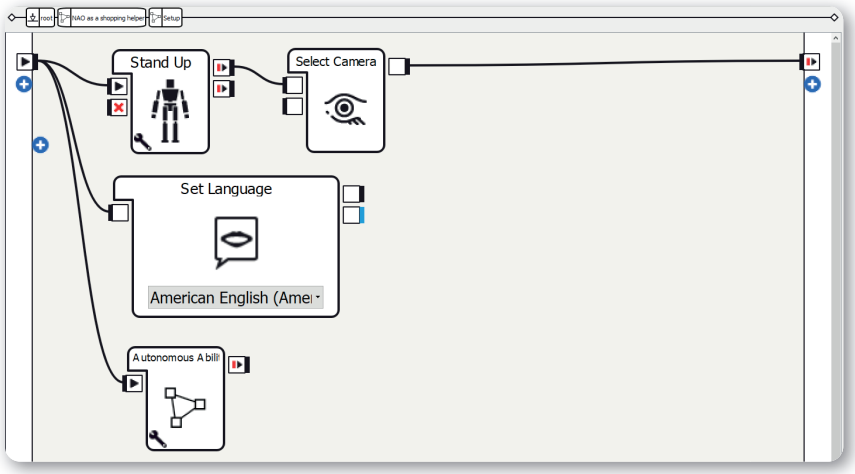
Method:

1. Create a Diagram box and rename it as Setup
2. Create a Choice box and do the following:
    a. Under Localized Text, select the language you want and enter the question, here: "What do you prefer? Chocolate or donuts?"
    b. Under Choice, select the language you want and enter possible answers, here: "Chocolate" and "Donuts"
3. Create a Python Script box
4. Add an input with the name "input_naomark" and the data type "Number" to the Python Script box
5. Add an input with the name "input_choice" and the data type "String" to the Python Script box
6. Add an output with the name "output_correct" and the data type "Bang" to the Python Script box
7. Add an output with the name "output_incorrect" and the data type "Bang" to the Python Script box
8. Create a NAOMark box
9. Create two Text Edit boxes
10. Set the texts to something like:
    a. Correct: "Here is the product you want!"
    b. Incorrect: "That's not the product you want, unfortunately. Try again!"
11. Create two Say Text boxes
12. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram, NAO asks the other person whether they would prefer donuts or chocolate.
- The answer is processed in the Python Script box.
- Depending on whether you show NAO the correct or incorrect object, it will answer either "Correct" (see 10a) or "Incorrect" (see 10b).

The Setup diagram

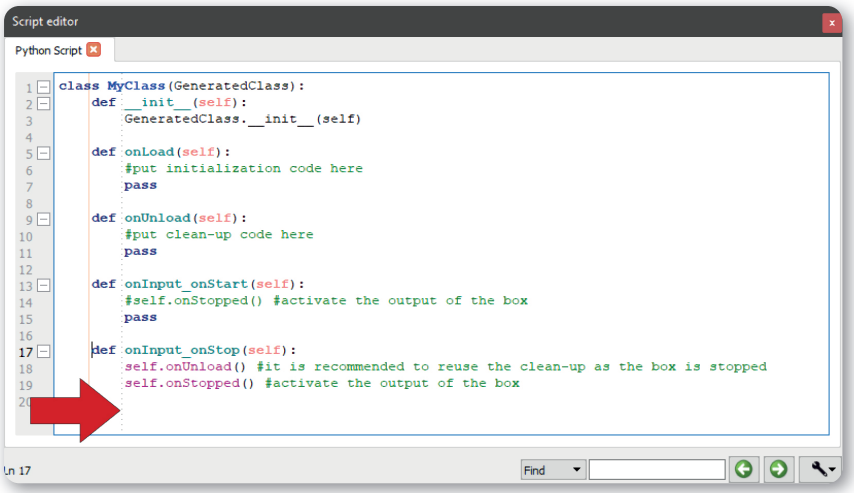


Screen 2.12.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate the** Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three**.**

The Python Script box



Screen 2.12.3.3 - The Python Script box

Method:

1. Create the method onInput_input_naomark(self, p) and add the following source code:

```
def onInput_input_naomark(self, p):
    self.naomark = p
    if self.naomark == self.number: #compare
        self.correctProduct()
    else:
        self.incorrectProduct()
```

2. Create the method onInput_input_choice(self, i) and add the following source code:

```
def onInput_input_choice(self, i):
        self.choice = i
        if self.choice == "Donuts":
            self.number = 84
        elif self.choice == "Chocolate":
            self.number = 80
    self.output_naomark()
```

3. Create the method correctProduct(self) and add the following source code:

```
def correctProduct(self):
    self.output_correct()
```

4. Create the method incorrectProduct(self) and add the following source code:

```
def incorrectProduct(self):
    self.output_naomark()
    self.output_incorrect()
    self.onStopped()
```

More project ideas

More options

You can expand the program by providing more options. Note: If you do this, you also need to edit the source code in the Python Script box!

NOTES

13 NAO tells a joke

Overview

When you're in a bad mood a good joke can cheer you up.

In the project "NAO tells a joke", a table containing a list of jokes is used to help NAO make you laugh.

Objects required and preparation

Diagram 2.13.2.1 - Objects required

For this project you need:

- A CSV file containing jokes

So that you don't have to create a Text Edit box for each joke, you can use a CSV file to make the project easier to follow. You do need to write a little Python source code, but you can increase the number of jokes much more easily.

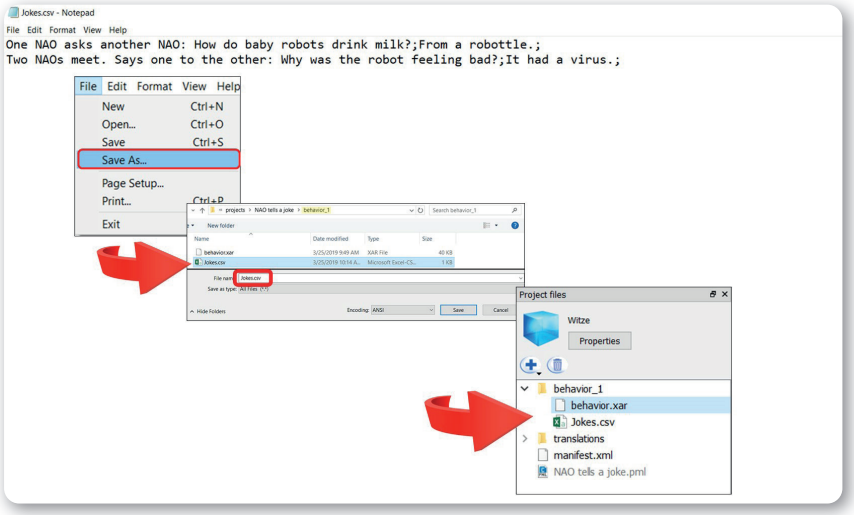
You can create a CSV file with a simple text editor. Think of a CSV file as a table: The rows of the table are the lines in the text; the columns are separated by commas or semicolons. Semicolons are used here for separation, as commas are used in the sentences.

```
1 One NAO asks another NAO: How do baby robots drink milk?;From a robottle.;
2 Two NAOs meet. Says one to the other: Why was the robot feeling bad?;It had a virus.;
```

Screen 2.13.2.2 - Structure of the CSV file

Your jokes should be structured as shown in the screenshot above, like this:

Joke1_Part1;Joke1_Part2;
Joke2_Part1;Joke2_Part2;
Joke3_…



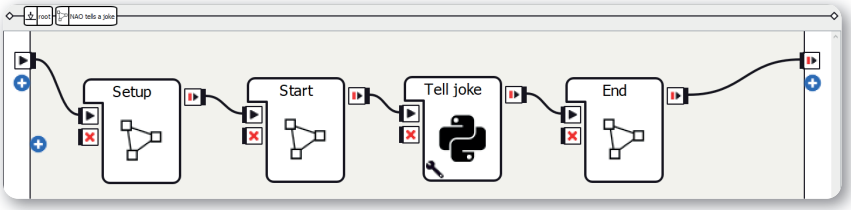
Screen 2.13.2.3 - Save CSV file

Once you have written down your jokes, do the following:

- Click on "Save as..."
- Set the name of the file to "Jokes"
- Set the file extension to .csv
- Save the file in the project folder under the Behavior folder

Suggested solution

Basic framework



Screen 2.13.3.1 - Basic framework

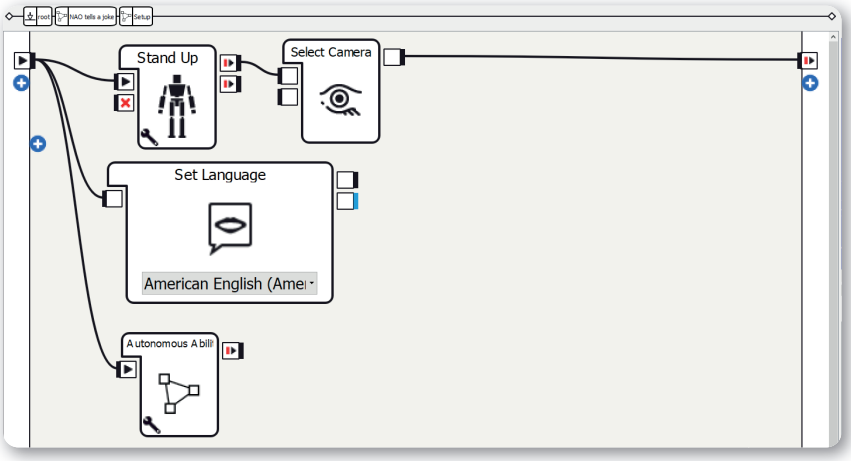
Method:

1. Create three Diagram boxes and a Python Script box
2. Rename the Diagram boxes as Setup, Start, and End
3. Add a parameter with the name "Name of the CSV file" to the Python Script box
4. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram, NAO says in the Start diagram that it is going to tell a joke.
- In the Python Script box, the CSV file is now opened, the individual jokes are loaded, and a random joke is selected and told.
- Afterward, NAO laughs and the program is ended.

The Setup diagram

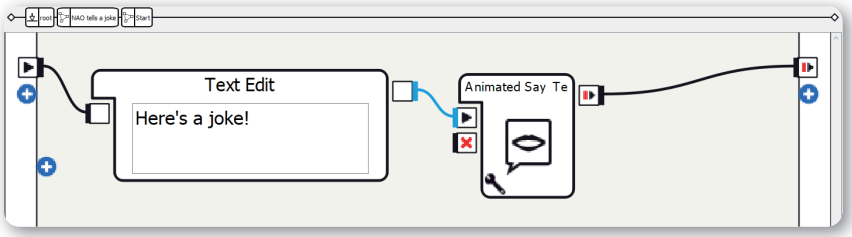


Screen 2.13.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate** Autonomous Blinking, Background Movement, and Speaking Movement in the parameters of the Autonomous Abilities box; deactivate the other two.

The Start diagram



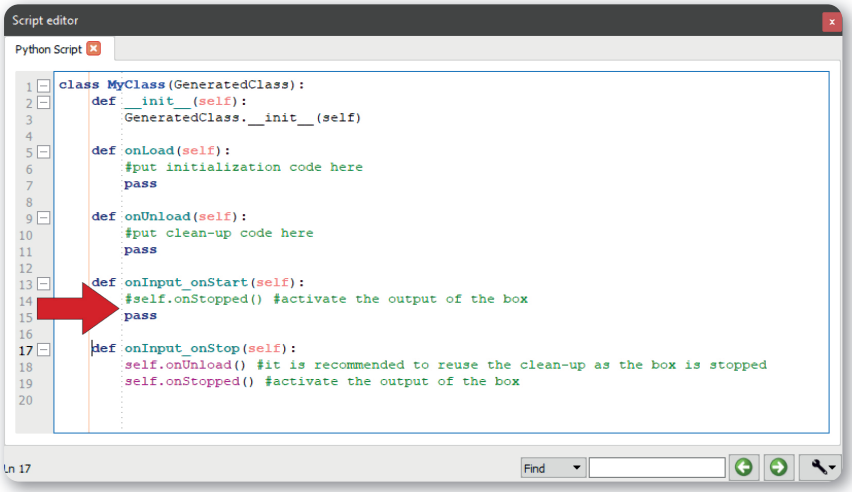
Screen 2.13.3.3 - The Start diagram

Method:

1. Create a Text Edit box
2. Set the text to something like "Here's a joke!"
3. Create a Animated Say Text box
4. Make connections as shown in the screenshot

At the start of the program NAO says that it is going to tell a joke. If you want, you can expand this section yourself, for example with an animation.

The Python Script box



Screen 2.13.3.4 - The Python Script box

Method:

1. Add the following source code to the method onInput_onStart(self):

```
def onInput_onStart(self):
    import csv
    import random as rand
    import time

    self.tts = self.session().service("ALTextToSpeech")

    # If there is something in the parameter "Name of
    the CSV file", the path can be created
    if self.getParameter("Name of the CSV fi le") != "":
        # Save path for the joke in variable
        jokepath = os.path.join(self.behaviorAbsolutePath(),
        self.getParameter("Name of the CSV file"))

    # Otherwise the Python Box should be stopped
    else:
        self.logger.error("Script not executable: parameter
        empty")
        self.onUnload()
        self.onStopped()

    self.jokes = []
    # Jokes from CSV table using a CSV parser as field in a
    field
    with open(jokepath, 'r') as jokefile:
        # csv-file-reader
        reader = csv.reader(jokefile, delimiter=";")
        for row in reader:
            self.jokes.append(row)

    # Random joke is selected from the list (between 0 and
    length of the list -1)
    jokenumber = rand.randint(0, len(self.jokes)-1)

    # First part of the joke is loading (CSV filed, index
    0) and being told
    joke_part1 = self.jokes[jokenumber][0]
```

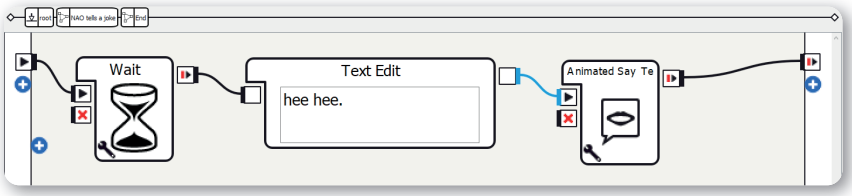
```
self.tts.say(joke_part1)

# Second part of the joke is loading (CSV field, index
1).
# If there is a second part this will be told after a
pause.
joke_part2 = self.jokes[jokenumber][1]
if joke_part2 != "":
    time.sleep(1.5)
    self.tts.say(joke_part2)

# End source code
self.onUnload()
self.onStopped()
```

First of all, the method checks whether the "Name of the CSV file" parameter you created in the Python Script box contains anything. If this is not the case, the box is stopped.

Then a CSV parser is used to insert the CSV table into a field with the following format: [[joke1_part1, joke1_part2], [joke2_part1, joke2_part2], ...]

A joke is then selected using a random natural number and the parts of the joke are told one after the other.

The End diagram



Screen 2.13.4.5 - The End diagram

Method:

1. Create a Wait box
2. Create a Text Edit box
3. Set text to something like "hee hee."
4. Create an Animated Say Text box
5. Make connections as shown in the screenshot

As NAO found the joke so funny, it laughs at the end.

More project ideas

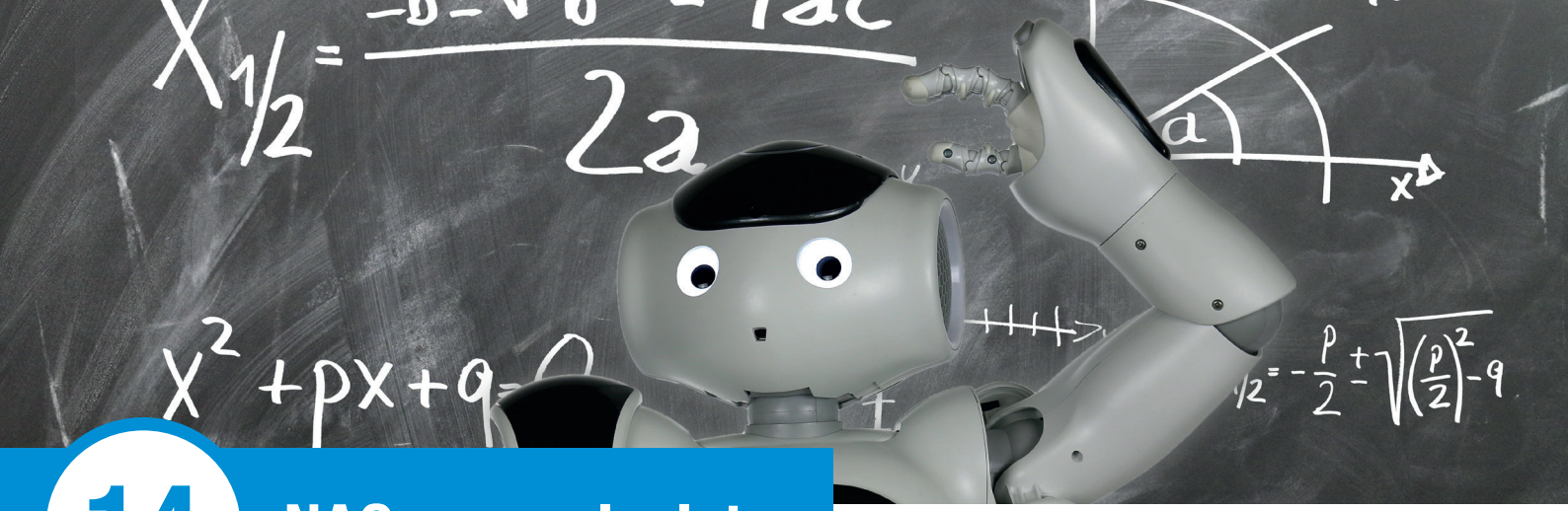
Incorporate sound effects

At the end of the program you can incorporate a sound effect (e.g. "Ba-dum-tss").

NAO as a language teacher

In project 11 "NAO as a language teacher" you can use a CSV file for the vocabulary, for example. The German vocabulary and its English translation are in one row so that you can access them.

NOTES



14 NAO as a calculator

Overview

You're too lazy to work it out yourself or use a calculator? Then just ask NAO! This program is used to calculate simple terms in seconds!

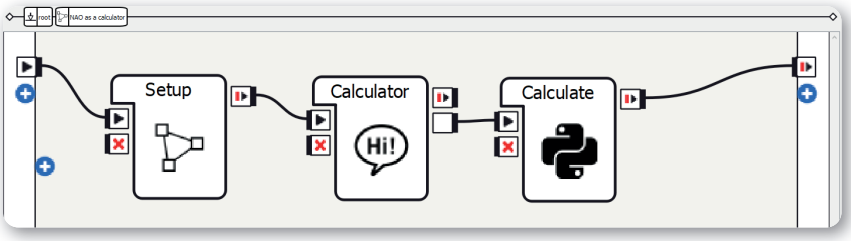
In the project "NAO as a calculator", NAO determines the values of simple mathematical terms for you. Voice recognition enables NAO to understand terms like "two times two" or "three plus one" and gives the result.

Objects required and preparation

You don't need anything extra for this project.

Suggested solution

Basic framework



Screen 2.14.3.1 - Basic framework

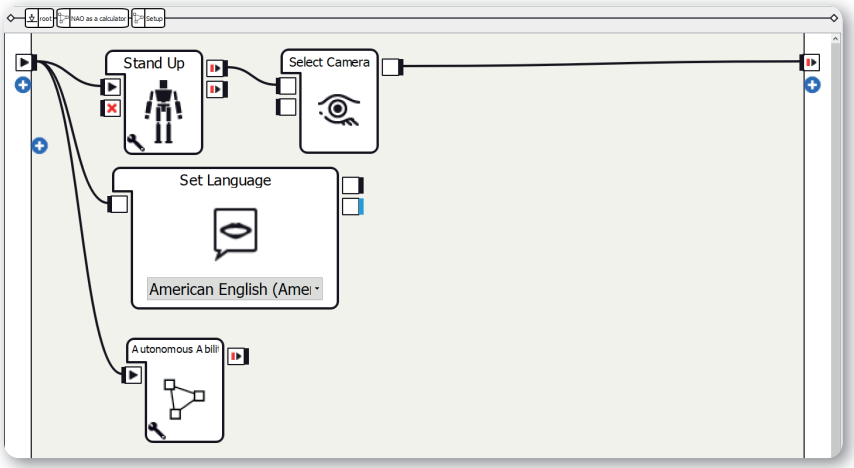
Method:

1. Select the relevant language on the left from among the supported languages in the Project Properties
2. Create a Diagram box and rename it as Setup
3. Create a new Dialog box with topic in the relevant language
4. Create a Python Script box
5. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram, NAO asks you to say a mathematical term.
- The result is then calculated in the Python Script box and said.

The Setup diagram

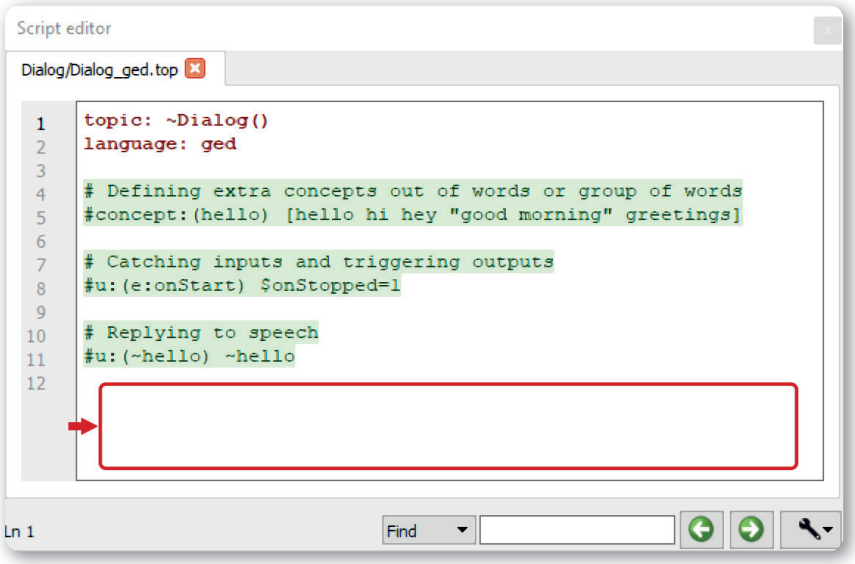


Screen 2.14.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only** activate the Autonomous Blinking and Background Movement parameters in the Autonomous Abilities box; deactivate the other three.

The Dialog box



Screen 2.14.3.3 - The Dialog box

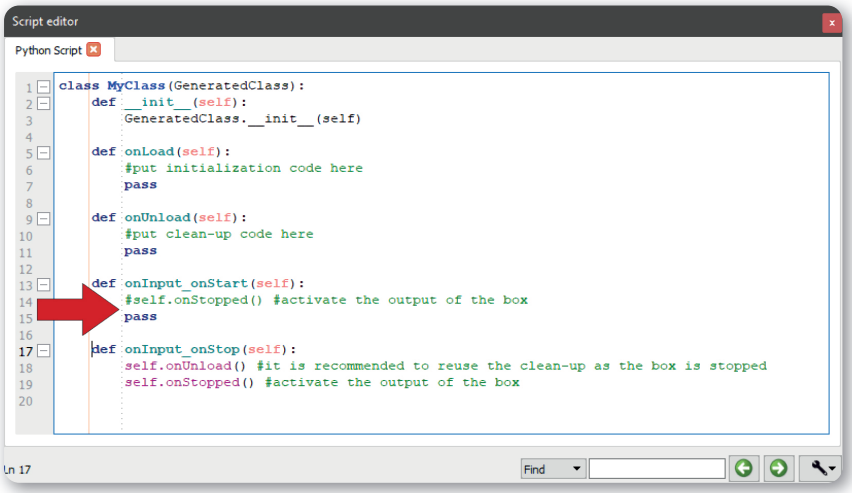
Method:

1. Add the following source code to the topic you have created:

```
concept: (Numbers) [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25]
concept: (ArithmeticOperator) [plus minus times "divided
by" "to the power of"]

# Insert numbers and arithmetic operators in ALMemory then
activate output
u:(e:onStart) Give me a mathematical term.
    u1:(_~Numbers _~ArithmeticOperator _~Numbers) Your
    term is $1 $2 $3 . $number1=$1 $arithmeticoperator=$2
    $number2=$3 $output=1
```

NAO registers your term with the Dialog box. You determine which numbers NAO can recognize in "concept: (Numbers)". Here, they are the numbers 1 - 25. If you want, you can also expand the numbers, e.g. up to 100.

The Python Script box



Screen 2.14.3.4 - The Python Script box

Method:

1. Add the following source code to the method "onInput_onStart(self)":

```
def onInput_onStart(self):
    mem = self.session().service("ALMemory")
    tts = self.session().service("ALTextToSpeech")

    number1 = mem.getData("number1") number2 =
    mem.getData("number2")

    arithmeticoperator = mem.getData("arithmeticoperator")
    if arithmeticoperator == "plus":
        result = int(number1)+int(number2)
    elif arithmeticoperator == "minus":
        result = int(number1)-int(number2)
    elif arithmeticoperator == "times":
        result = int(number1)*int(number2)
    elif arithmeticoperator == "divided by":
        result = int(number1)/int(number2)
    elif arithmeticoperator == "to the power of":
        result = int(number1)**int(number2)

    tts.say(number1 + " " + arithmeticoperator + " " +
    number2 + " equals " + str(result))
    self.onStopped()
    self.onUnload()
```

First you set the ALMemory and ALTextToSpeech modules in variables so that you can access their methods. Then the two numbers are called up from NAO's memory and the value is determined depending on the arithmetic operator. The result is said using TextToSpeech.

NOTES

15 NAO plays soccer

Overview

The global soccer industry is enormous – but can robots replace our beloved human soccer players? Would that still be entertaining? You can try it out for yourself in this project!

In the project "NAO plays soccer", NAO's aim is to score a goal. To do this, it first needs to detect the ball, move toward it, and then shoot it at the goal.
In this project you need to find the right parameters –depending on the structure of the pitch or the size of the ball.

Objects required and preparation

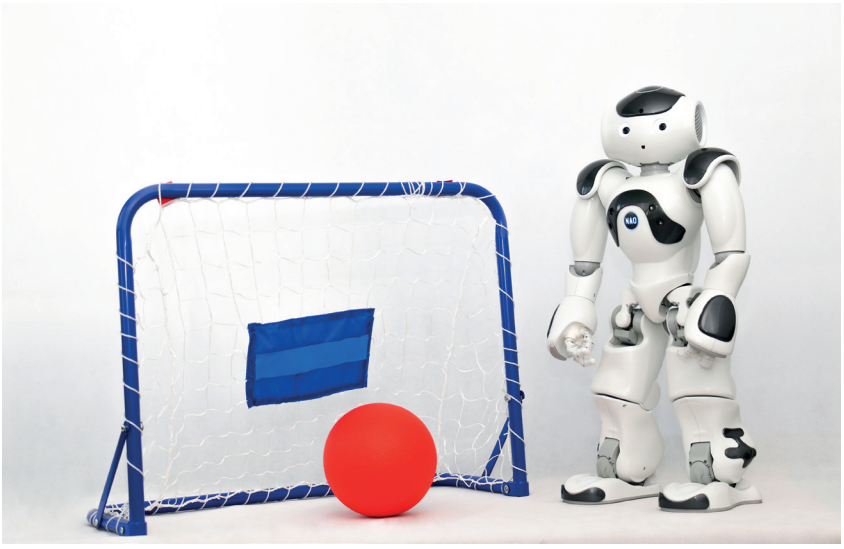


Image 2.15.2.1 - Objects required

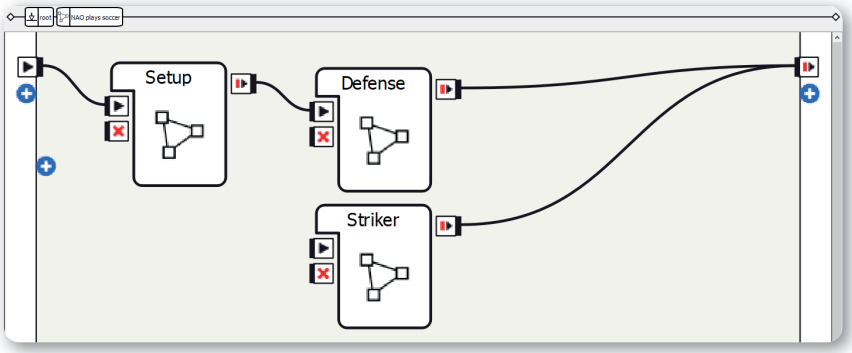
For this project you need:

- A red ball (minimum diameter: 10 cm; optimum diameter: 15 cm)
- A flat, non-slip surface with enough space
- A small goal or goal markers

The red ball is used as a soccer ball as this means we can work with the Red Ball Tracker box. In order for NAO to really be able to run around, you need a large, flat, non-slip surface. It would also be good if you had a small soccer goal, but goal markers consisting of two objects are also okay.

Suggested solution

Basic framework



Screen 2.15.3.1 - Basic framework

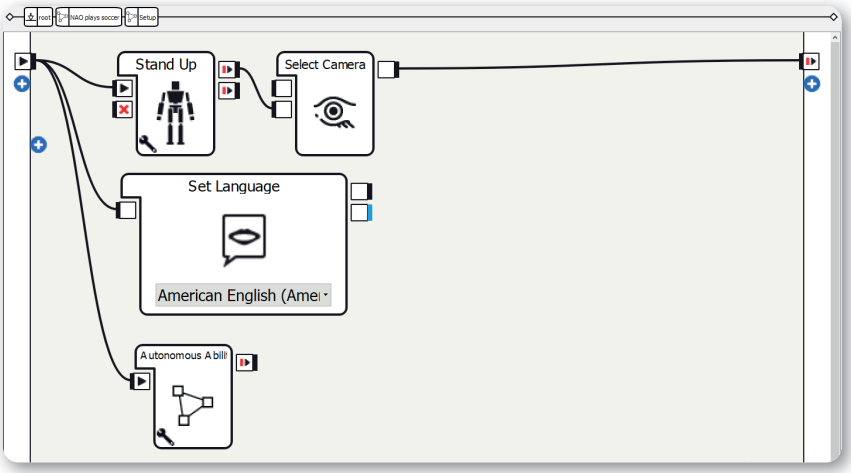
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Defense, and Striker
3. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram, the Defense or Striker diagram is started depending on the connection.
- Generally, NAO moves to the ball using the Red Ball Tracker box.

The Setup diagram

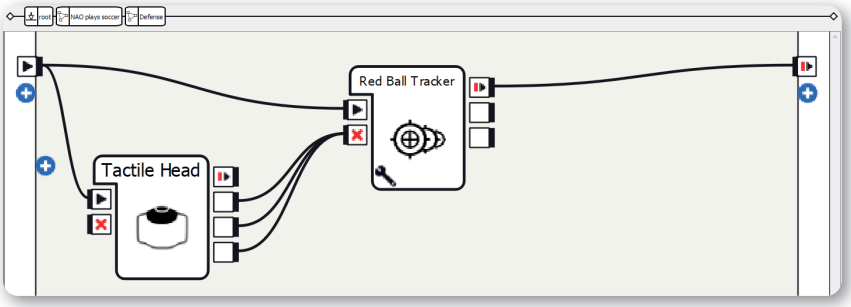


Screen 2.15.3.2 - The Setup box

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. Use the bottom input of the Select Camera box
3. **Only** activate Autonomous Blinking in the parameters of the Autonomous Abilities box; deactivate the other four

The Defense diagram



Screen 2.15.3.3 - The Defense diagram

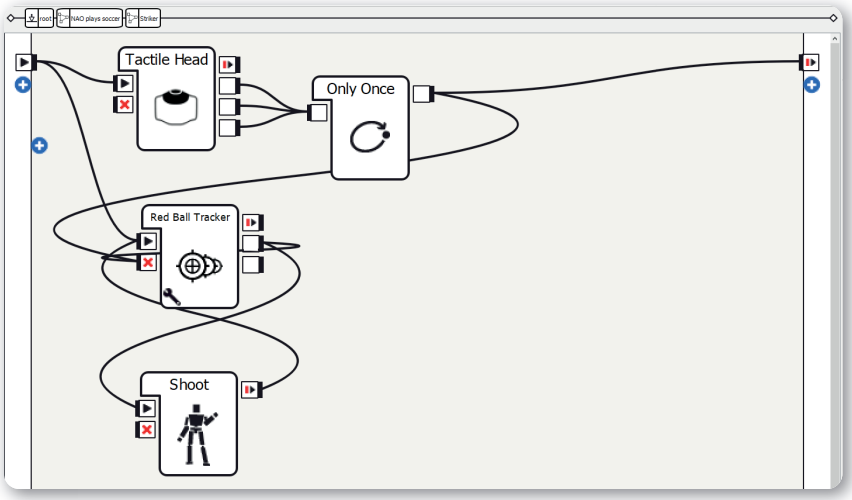
Method:

1. Create a Tactile Head box
2. Create a Red Ball Tracker box
3. Set the parameters of the Red Ball Tracker box:
    a. Diameter: In the example: 0.1 (use the diameter of your ball!)
    b. Distance X: 0.01
    c. Threshold X: 0.1
    d. Distance Y: 0
    e. Threshold Y: 0
    f. Theta: 0
    g. Threshold Theta: 0

4. Make connections as shown in the screenshot

When the diagram is started, NAO looks for a red ball again. Once it finds one, NAO follows the ball. Once it reaches the ball, NAO defends the ball by standing in front of it. You can stop NAO at any time by tapping on its head sensors.

The Striker diagram



Screen 2.15.3.4 - The Striker diagram

Method:

1. Create a Tactile Head box
2. Create an Only Once box
3. Create a Red Ball Tracker box
4. Set the parameters of the Red Ball Tracker box:
    a. Diameter: In the example: 0.1 (use the diameter of your ball!)
    b. Distance X: 0.05
    c. Threshold X: 0.15
    d. Distance Y: 0.2
    e. Threshold Y: 0.15
    f. Theta: 0
    g. Threshold Theta: 1
5. Create a Timeline box
6. In the Timeline box, create an animation for shooting:
    a. Leave some distance between the keyframes as otherwise NAO could fall over.
    b. Always stay within arm's reach of NAO so that you can catch it if it falls!
    c. Example sequence for the movement:
        - Bend left knee and push off with the right leg
        - Quickly pull in the right leg
        - Kick with the right leg
        - Squat for stability
        - Stand up after 2-3 seconds
7. Connect the output of the Only Once box to the onStopped input of the Red Ball Tracker box
8. Make the rest of the connections as shown in the screenshot

The striker is like the defense player with added shooting animation. When the diagram is started, NAO looks for a red ball. Once it finds one, NAO follows the ball until it reaches it. Then the Red Ball Tracker box is stopped and the shooting animation is carried out, which will hopefully get the ball into the goal. You can stop NAO at any time by tapping on its head sensors.

More project ideas

Goalkeeper

NAO can also be a goalkeeper: For example, NAO could sit down quickly once it sees a red ball.

Penalty

For the penalty you can use the same shooting animation as for the striker. Make sure that you position NAO close enough to the ball though.

16 NAO learns to draw and write

Overview

In the Middle Ages, monks had to spend days and weeks writing out books such as the Bible. Since the invention of the printing press, this has no longer been necessary. But how well can robots perform this writing task?

In this project you can teach NAO to draw and write. NAO draws an arrow pointing to a printout of NAO's head. And NAO writes its own name.
Note: You need a lot of patience for this project as you will have to repeat some steps several times to get a good result.

Objects required and preparation

Image 2.16.2.1 - Objects required 1

Image 2.16.2.2 - Table variation 1

Image 2.16.2.3 - Table variation 2

Image 2.16.2.4 - Table variation 3

For this project you need:

- Something to write with (recommended: pencil)
- A writing surface (dimensions in the suggested solution: height 39 cm, page length 50 cm)
- Elastic bands (not recommended: adhesive tape, as it could leave marks)
- A printout of NAO's head (width: 10 cm)

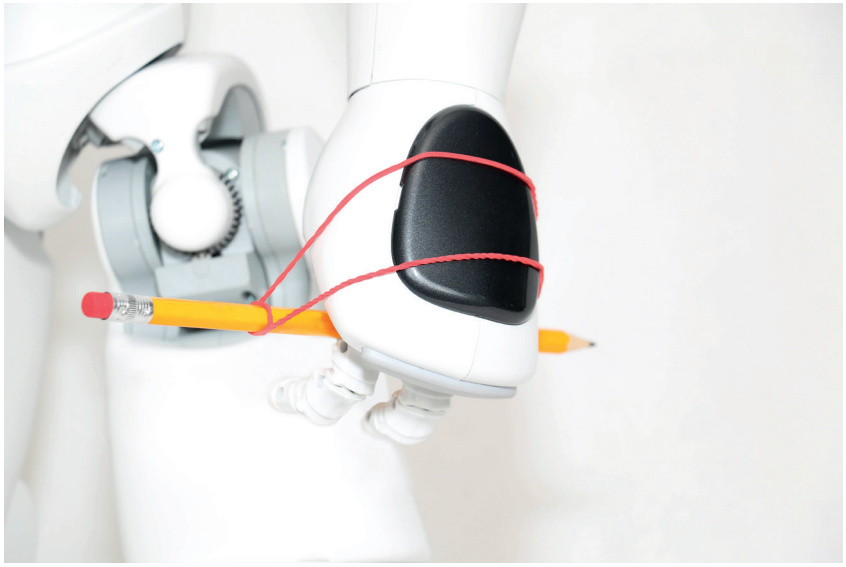
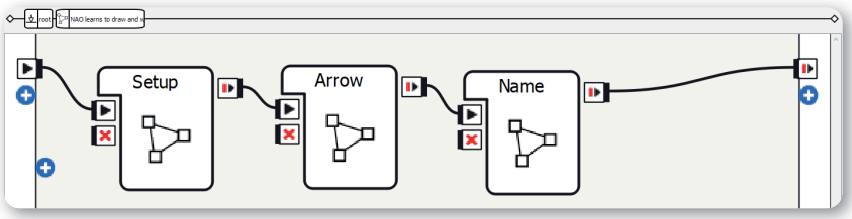


Image 2.16.2.5 – Fastening

You can fasten a pencil to NAO's hand using an elastic band as shown in the "Fastening" image above.

Suggested solution

Basic framework



Screen 2.16.3.1 - Basic framework

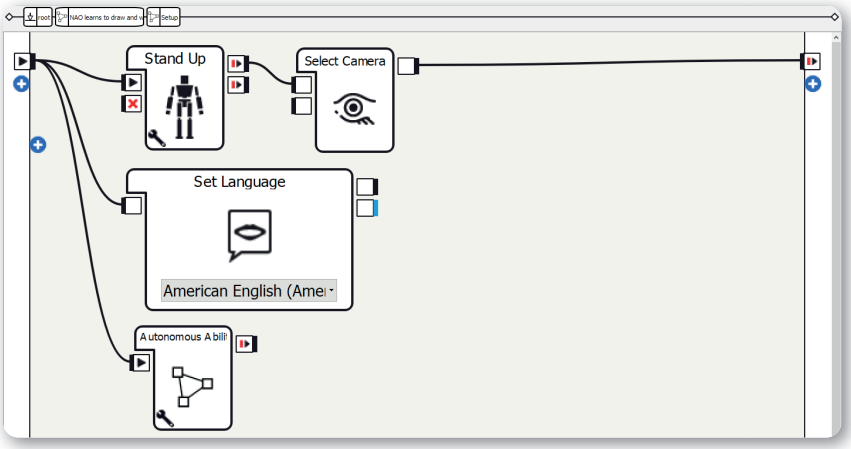
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Arrow, and Name
3. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram the arrow is drawn and then the name is written.
- Note that the two tasks are performed with different hands!

The Setup diagram

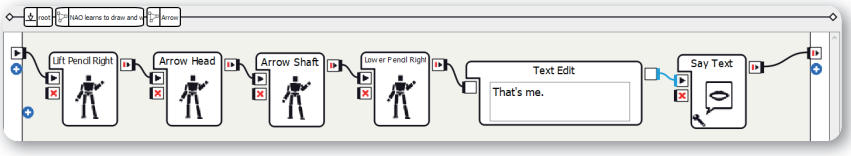


Screen 2.16.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only** activate Autonomous Blinking in the parameters of the Autonomous Abilities box; deactivate the other four

The Arrow diagram



Screen 2.16.3.3 - The Arrow diagram

Method:

1. Create four Timeline boxes:
   a. Lift Pencil Right: In this diagram the pencil is lifted sideways above the tabletop (this is necessary so that NAO doesn't bang into the table)
   b. Arrow Head
   c. Arrow Shaft
   d. Lower Pencil Right: This animation is the Lift Pencil Right animation in reverse

2. Create a Text Edit box
3. Set the text to something like "That's me."
4. Create a Say Text box
5. Make connections as shown in the screenshot

In this diagram NAO draws an arrow pointing to NAO's head. Once the arrow has been drawn, NAO says that this is NAO's head.

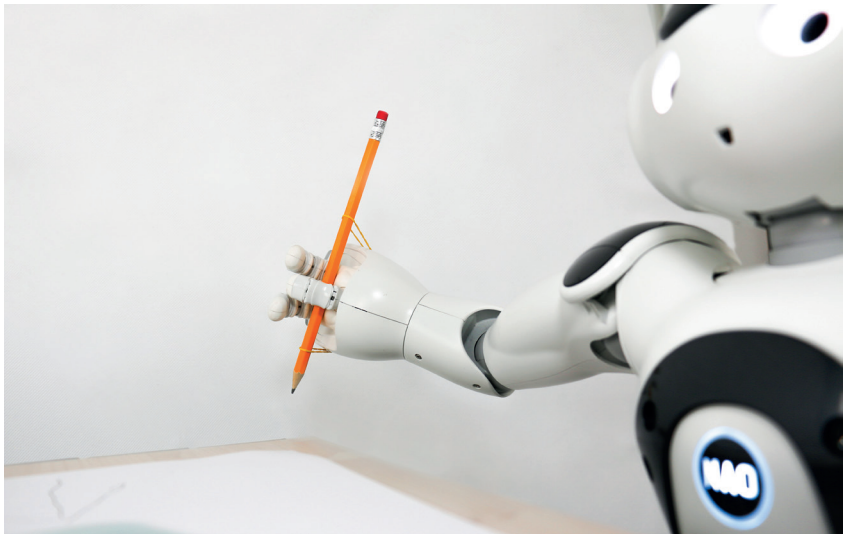
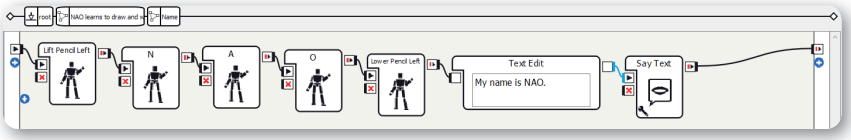


Image 2.16.3.4 - Lifting sideways

You need to create the keyframes for the animations depending on the height of your table, otherwise NAO writes in the air or bangs into the tabletop.

The Name diagram



Screen 2.16.3.5 - The Name diagram

Method:

1. Create five Timeline boxes:
    a. Lift Pencil Left: In this diagram the pencil is lifted sideways above the tabletop (this is necessary so that NAO doesn't bang into the table)
    b. N
    c. A
    d. O
    e. Lower Pencil Left: This animation is the Lift Pencil Left animation in reverse
2. Create a Text Edit box
3. Set the text to something like "My name is NAO."
4. Create a Say Text box
5. Make connections as shown in the screenshot

In this diagram, NAO writes its own name with its left hand. After it has written its name, NAO introduces itself.
You need to create the keyframes for the animations to match the height of your table.

More project ideas

Your own name!

It's not just its own name that NAO can write, it can write yours, too!

NOTES

17 NAO goes shopping

Overview

Digital assistants are becoming more and more popular. They can wake you up, list your appointments for the day, and tell jokes. Yet these are just virtual tasks that they can carry out. Can we go a step further and have them perform tasks in the real world such as shopping in a supermarket?

NAO stands with its left hand on the shopping cart, pulls a flower into the cart using a loop with its right hand, pushes the shopping cart further, sees cookies, doesn't want to eat cookies, and then goes to the checkout.

Objects required and preparation

Image 2.17.2.1 - Objects required

For this project you need:

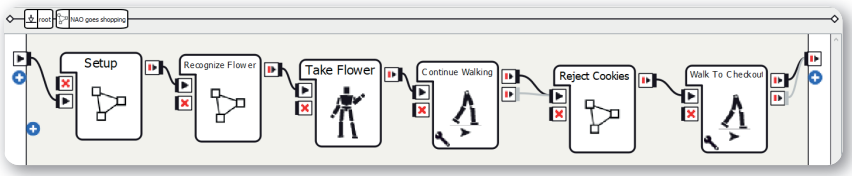
- A sales stand or shelf (height in the suggested solution: 35 cm)
- An object to buy with a loop attached
- Some wire, pipe cleaner, string, etc.
- A shopping cart to push (height in the suggested solution: 23 cm)

The sales stand should be slightly lower than NAO's shoulder. To enable NAO to grip the object securely, its fi ngers are not used; instead its hand passes through a loop. Make a loop from your chosen material that is a few centimeters larger than NAO's hand. Then attach it to an object to buy. Your sales stand can look like the one in the "Objects required" image above.

As NAO also needs to recognize the object, a Vision Recognition Database is required. To create this, put NAO in the start position and save the object. Don't forget to upload it! Refer to section C.3.1.15 The “Vision Recognition” box of “Learn It” book.

Suggested solution

Basic framework



Screen 2.17.3.1 - Basic framework

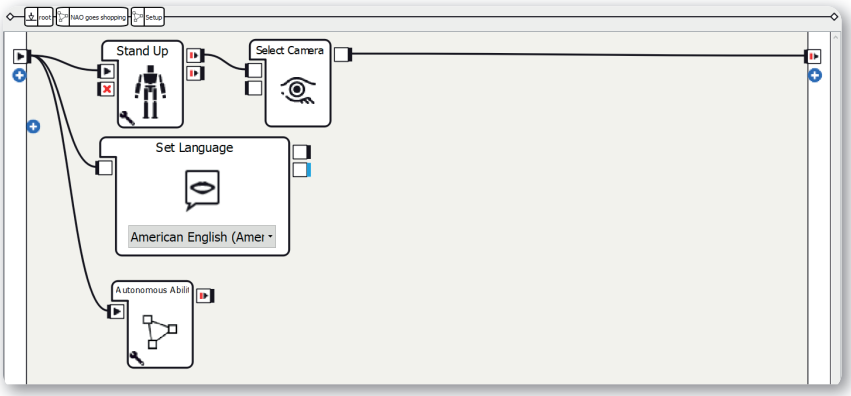
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Recognize Flower, and Reject Cookies
3. Create a Timeline box
4. Rename the Timeline box as Take Flower
5. Create two Move To boxes
6. Rename the Move To boxes as Continue Walking and Walk To Checkout
7. Set the X-parameter of the Continue Walking box to 0.3
8. Set the X-parameter of the Walk To Checkout box to 1
9. Deactivate "Arms movement enabled" in both Move To boxes
10. Make connections as shown in the screenshot

Sequence of the program:

- Once NAO recognizes the flower, the Timeline is used to put it in the cart.
- Then NAO continues pushing the shopping cart, says that it doesn't want to buy any cookies today, and finally walks to the checkout.

The Setup diagram

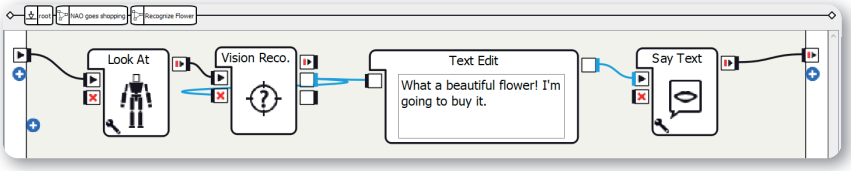


Screen 2.17.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. Only activate Autonomous Blinking in the parameters of the Autonomous Abilities box; deactivate the other four

The Recognize Flower diagram

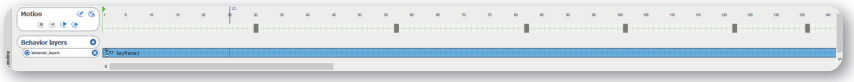


Screen 2.17.3.3 - The Recognize Flower diagram

Method:

1. Create a Look At box
2. Set the Y-parameter so that the object is recognized (here: -1)
3. Create a Vision Reco. box
4. Create a Text Edit box
5. Set the text to something like "What a beautiful flower! I'm going to buy it."
6. Create a Say Text box
7. Make connections as shown in the screenshot

NAO looks at the object here and then recognizes it. If you have problems with the object recognition you can check the troubleshooting section.
The Take Flower timeline



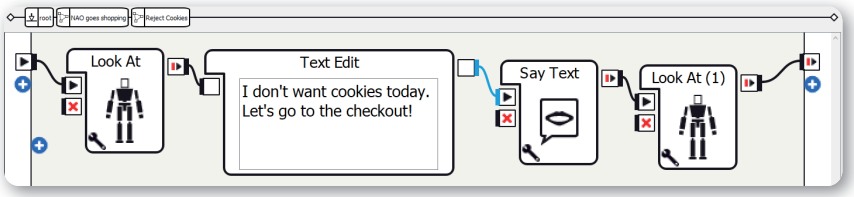
Screen 2.17.3.4 - The Take Flower timeline

Method:

1. Set the keyframes for the movement:
    a. NAO puts its hand through the loop
    b. NAO lifts up the object
    c. NAO moves its arm to the shopping cart
    d. NAO puts down the object
    e. NAO moves its arm to its side
    f. Last keyframe: Suitable arm position for pushing the shopping cart

In this timeline the animation is played that is used to take the object.

The Reject Cookies diagram



Screen 2.17.3.5 - The Reject Cookies timeline

Method:

1. Create a Look At box
2. Set the Y-parameter so that NAO looks at the other object (here: -1)
3. Create a Text Edit box
4. Set the text to something like "I don't want cookies today. Let's go to the checkout!"
5. Create a Say Text box
6. Create a Look At box
7. Make connections as shown in the screenshot

After pushing the shopping cart a short way, NAO looks at the cookies but doesn't feel like eating them today. So NAO walks to the checkout.

18 Playing dice with NAO

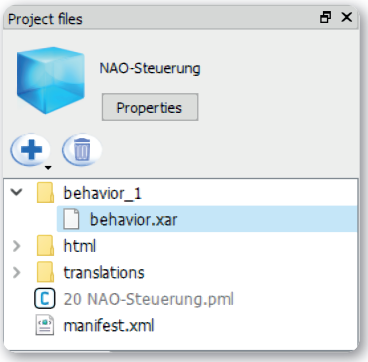
Overview

Do you want to show people all the cool programs you've created for NAO? You can choose them with a die!

First of all, the player throws the die with the Naomarks. Then you show NAO the top of the die and it performs the corresponding program.

Objects required and preparation

Image 2.18.2.1 - Objects required 1



Screen 2.18.2.2 – Objects required 2

For this project you need:

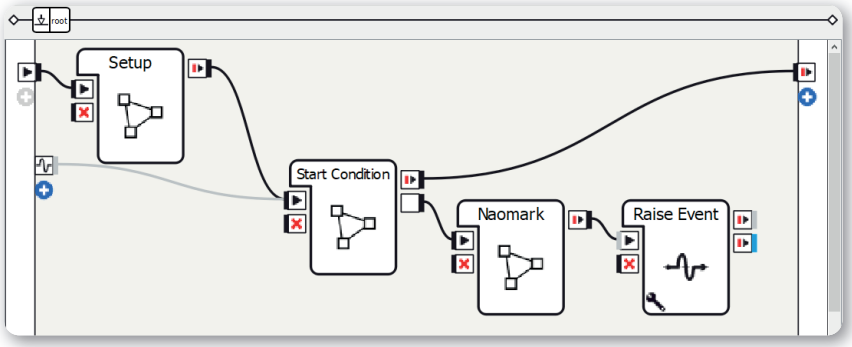
- A die
- Printed Naomarks (recommendation: A5 format)
- Several Choregraphe projects that are installed on NAO

For this you need a die with different actions on the sides for NAO to carry out. For example, you can write on a paper die with a pencil or use a die with cards you can change. It makes sense to laminate the cards so that you can reuse them.

You can install projects on NAO by clicking on the button at the top left in the "Robot applications" (see Screen 2.18.2.2 – Objects required 2).

Suggested solution

Basic framework



Screen 2.18.3.1 - Basic framework

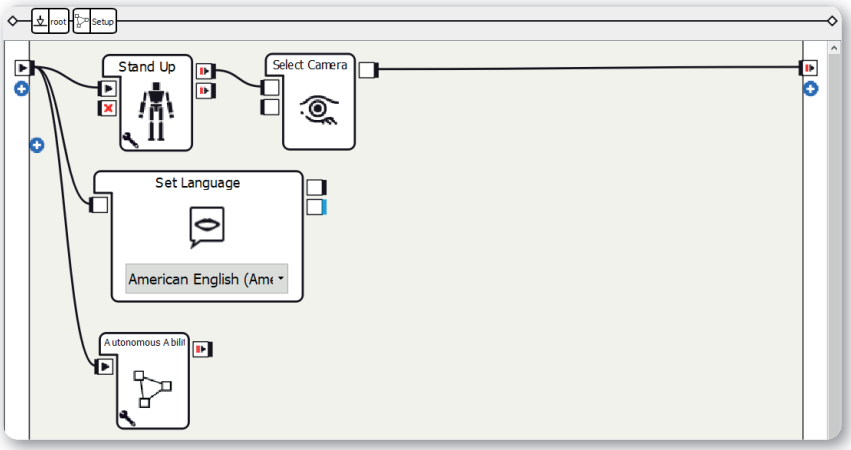
Method:

1. Create three Diagram boxes
2. Rename the Diagram boxes as Setup, Start Condition, and Naomark
3. Add an output with the data type "Bang" to the Start Condition diagram
4. Create a Raise Event box and set the key parameter to something like "dicegame/again"
5. Add an output from ALMemory on the left, name of the event in this suggested solution: "dicegame/again"
6. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup and Start Condition diagrams, NAO looks for a Naomark.
- If it finds one, another diagram or behavior is started.
- When it is finished, the game starts again and NAO looks for another Naomark.

The Setup diagram

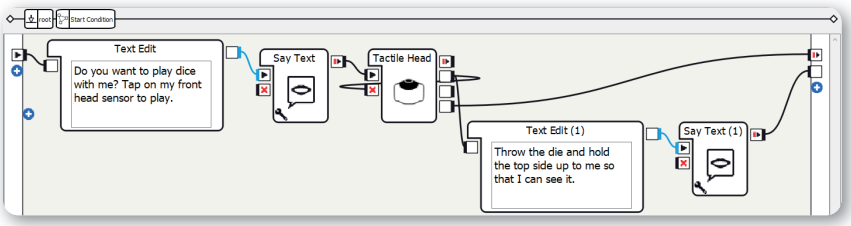


Screen 2.18.3.2 - The Setup diagram

The structure of the Setup diagram is explained in the subsection “The Setup diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Create a Stand Up box
2. **Only activate** Autonomous Blinking and Background Movement in the parameters of the Autonomous Abilities box; deactivate the other three**.**

The Start Condition diagram

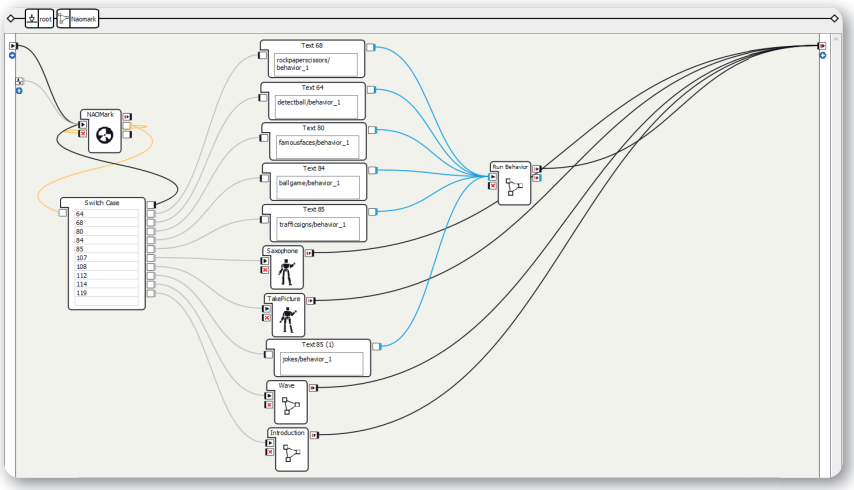


Screen 2.18.3.3 - The Start Condition diagram

The structure of the Start Condition diagram is explained in the subsection “Start Condition diagram” of the section "Structure of the projects". In this project you need to be aware of the following points:

1. Set the text of the left Text Edit box to something like "Do you want to play dice with me? Tap on my front head sensor to play."
2. Set the text of the right Text Edit box to something like "Throw the die and hold the top side up to me so that I can see it."

The Naomark diagram



Screen 2.18.3.4 - The Naomark diagram

Method:

1. Create a NAOMark box
2. Create a Switch Case box
3. Insert the Naomark numbers into the Switch Case box
4. Add a box for each case, examples:
   a. Connect a Text Edit box with a Run Behavior box to start a behavior installed on NAO
   b. A Timeline box with an animation
   c. A Diagram box with a small program
5. Make connections as shown in the screenshot

Once a Naomark has been recognized, a specific program is started.
You can decide for yourself which programs to assign to the various Naomarks. It is also worth incorporating your projects from the previous chapters of this book here.
Each application is defined by its "Application ID".You can modify this ID in the "Properties" of your project. For instance, this project "Playing dice with NAO" has the ID "dicewithnao", as you can see in the Properties. This allows us to call a certain behavior from another application, inside our project. As you can see in the "Project Files" of Choregraphe, a behavior ("behavior.xar" file) is located in the "behavior_1" directory. Thus, if want to run this behavior, we would then specify the Application ID and the folder containing our behavior.
For instance: "dicewithnao/behavior_1"
This is the same for every other applications installed on your robot: app_id/behavior_directory
This is what we are doing in this project, taking as parameter "p" this behavior:

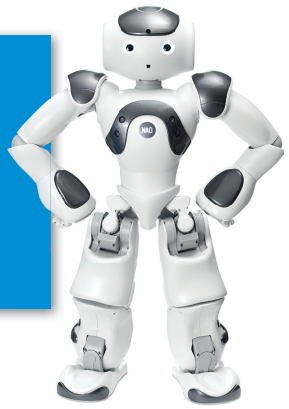
- rockpaperscissors/behavior_1
- detectball/behavior_1
- famousfaces/behavior_1

etc.

More project ideas

Card game

Instead of a die you can use cards turned over on a table.



19 Build Your Own Dialog!

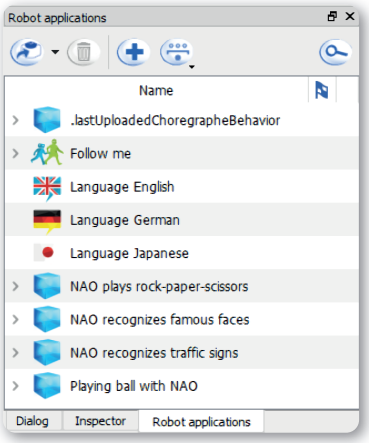
Overview

In this project you will program your own dialog to interact with NAO. This program can be used as entertainment at home or as a source of information in public places, for example. In the future you could be able to have a conversation with NAO just as you would with a person.

There are endless possibilities available to you for designing your Dialog. You can talk to or play with NAO and much more!

Refer to section C 3.1 Dialog box of the "Learn It" book, to revise dialog capabilities.

Objects required and preparation



Screen 2.19.2.1 - Objects required

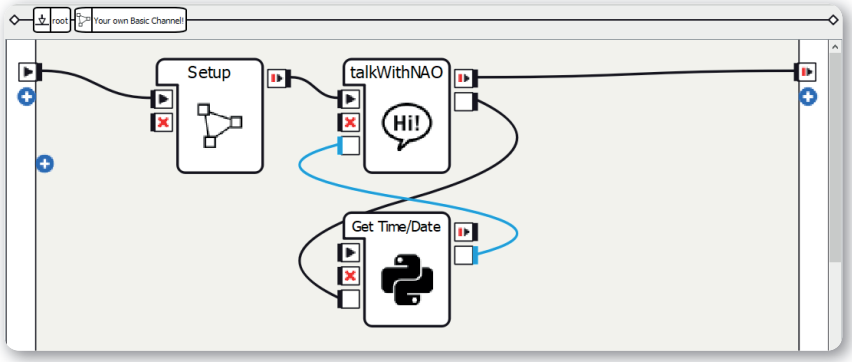
For this project you need:

- Optional: Several Choregraphe projects that are installed on NAO

As a general rule you don't need to prepare anything for the Dialog, however if you want to start your own programs you need to install them on NAO first. They are then displayed in the "Robot applications".
You can install projects on NAO by clicking on the button at the top left in the "Robot applications" (see Screen 2.18.2.2 – Objects required).

Suggested solution

Basic framework



Screen 2.19.3.1 - Basic framework

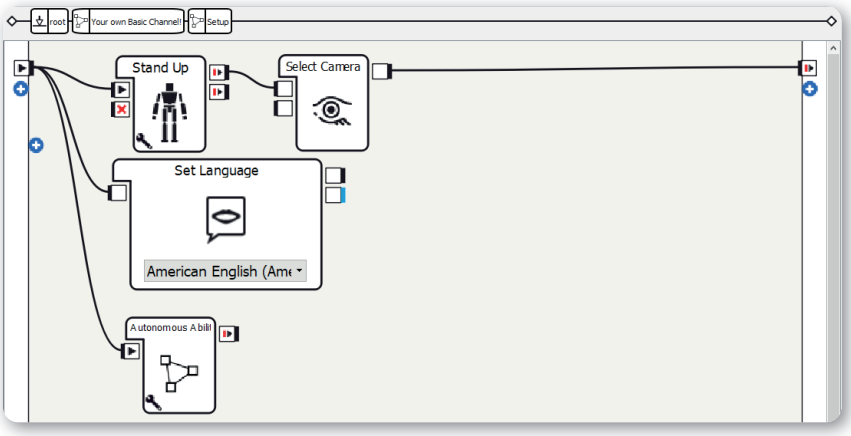
Method:

1. Select the relevant language on the left from among the supported languages in the Project Properties
2. Create a Diagram box and set the name as Setup
3. Create a new Dialog box with topic in the relevant language
4. Add a currentTime input with the "String" data type and a getTime output with the "Bang" data type to the Dialog box
5. Create a Python Script box
6. Add a getTime input with the data type "Bang" to the Python Script box
7. Add a currentTime output with the data type "String" to the Python Script box
8. Make connections as shown in the screenshot

Sequence of the program:

- After the Setup diagram the Dialog box is started so that you can talk to NAO.
- The Python Script box is needed so that NAO can tell the time if it is asked.

The Setup diagram

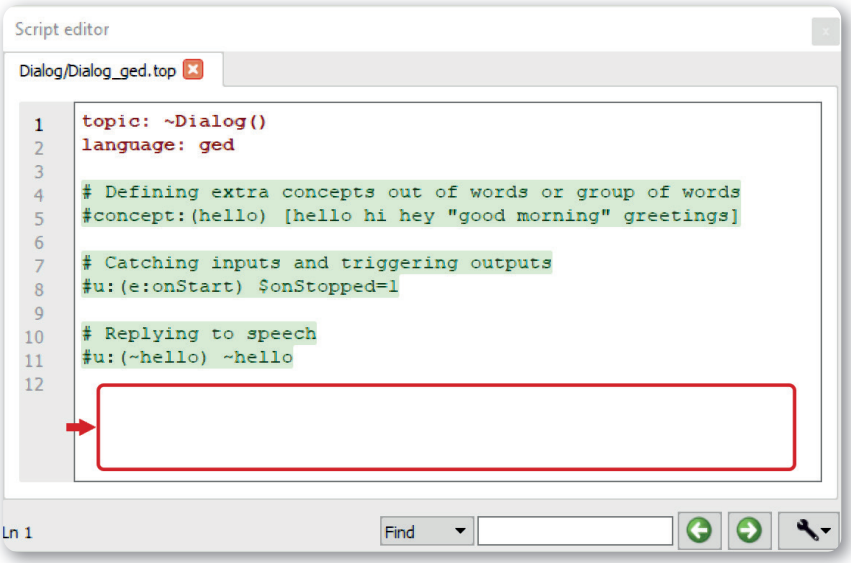


Screen 2.19.3.2 - The Setup diagram

The structure of the Setup diagram is explained in "Structure of the projects". In this project you need to be aware of the following points:

1. Use the Stand Up box
2. **Only activate** Autonomous Blinking and Background Movement in the parameters of the Autonomous Abilities box; deactivate the other three.

The Dialog box



Screen 2.19.3.3 - The Dialog box

Here is a small example of a dialog:

Method:

1. Add the following source code to the topic you have created:

```
# Concepts
concept: (hello) [hello "good morning" hi]
concept: (naohello) ^rand[hello hey "hi there" "nice to see you!
"]

#Start
u: (e: onStart) Hello, I am NAO.

#Smalltalk
u: (~hello) ~naohello
u: (["how are you {doing}" "are you well"]) I'm  fine, and you?
      u1: ([good super]) Super, I'm glad!
      u1: ([not so good "not good") That's a shame.
u: (bye {NAO}) Bye! $onStopped=1
u: (stop) Bye! $onStopped=1

#Info: Information about NAO
u: (what is your name) My name is NAO, nice to meet you!
u:(where do you live) I live at the SoftBank Robotics office in
Paris.
u: (can you swim)I can surf the internet but I can't swim yet
unfortunately.
# Interaction with NAO
u: (what time is it) $getTime=1
u: (e: currentTime) $currentTime

u: ("rock paper scissors") Okay.
^start(rockpaperscissors/behavior_1)
^wait(rockpaperscissors/behavior_1) What else do you want to do?
```

You can use this dialog to talk to NAO. However, you can also start installed programs using the path (in the source code: start(behaviorpath)). If you want to perform more complex sequences, you can do this using inputs and outputs of the Dialog box. You can expand this dialog however you like, for example by extending the Smalltalk section or talking to NAO about a specific topic (e.g. programming).

From a dialog you can execute behaviors or animations using ^run, ^start, ^wait and ^stop

These functions take the behavior's path prefixed with application id as a parameter.

The application id is set in the Properties of your application from the "Project files" panel.

The behaviour path is the directory behavior_1 in this project.

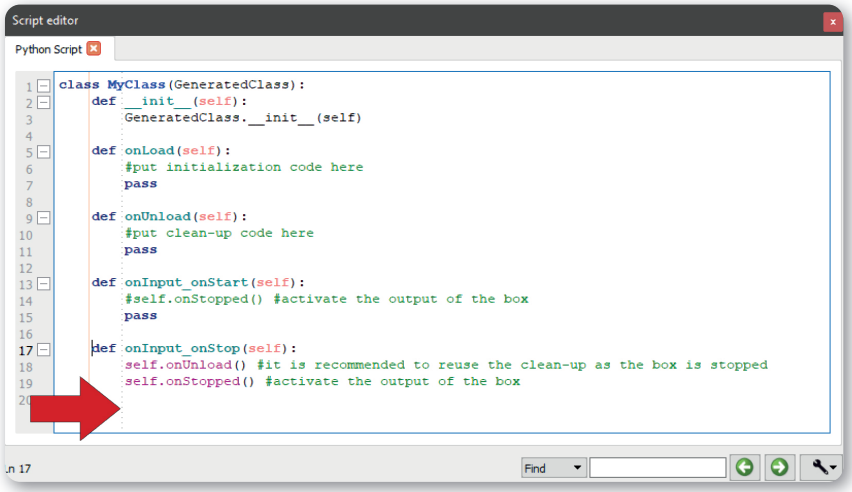
Details on functions:

^run: Suspends the speech, runs an animation and resumes the speech.

^start: Starts behaviors or animations.

^wait: Waits until a running behavior or an animation has finished.
^stop: Stops a running behavior or an animation.
More information here: ***https://developer.softbankrobotics.com/nao6/naoqi-developer-guide/naoqi-apis/naoqi-audio/alanimatedspeech#annotated-text***

The Python Script box



Screen 2.19.3.4 - The Python Script box

Method:

1. Add the following source code to the method onInput_getTime(self):

```
def onInput_getTime(self):
  # Get date
  currentTime = datetime.datetime.now()

  output_text = "It is " + str(currentTime.hour) + " " +
  str(currentTime.minute) + "."
  self.currentTime(output_text)
  self.onStopped()
```

In this source code, the current time is determined before the time to be spoken is assembled and converted into a string. Then the string is issued and said in the Dialog box.

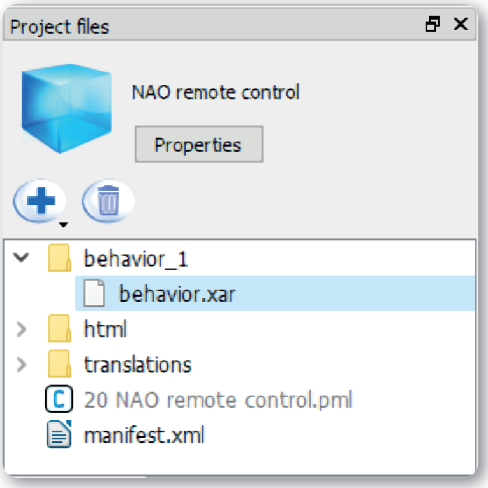
NOTES

20 Template: NAO remote control

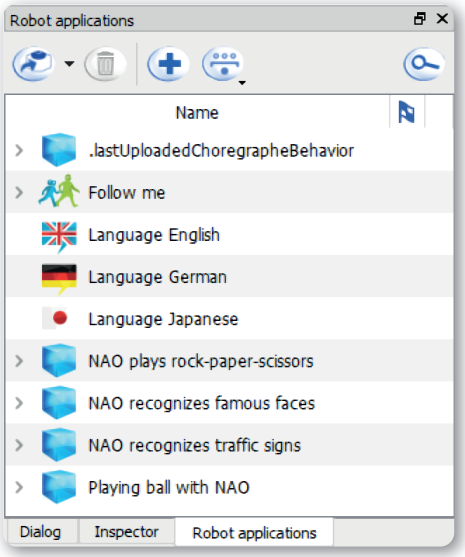
Overview

If your notebook or laptop is heavy and the battery doesn't last very long, it can be difficult to spend long periods presenting NAO to an audience with Choregraphe, for instance. But did you know that you can also control NAO and send commands via a web browser? This chapter shows you what such a website can look like.

Objects required and preparation



Screen 2.20.2.1 - The control project template



Screen 2.20.2.2 - Robot applications

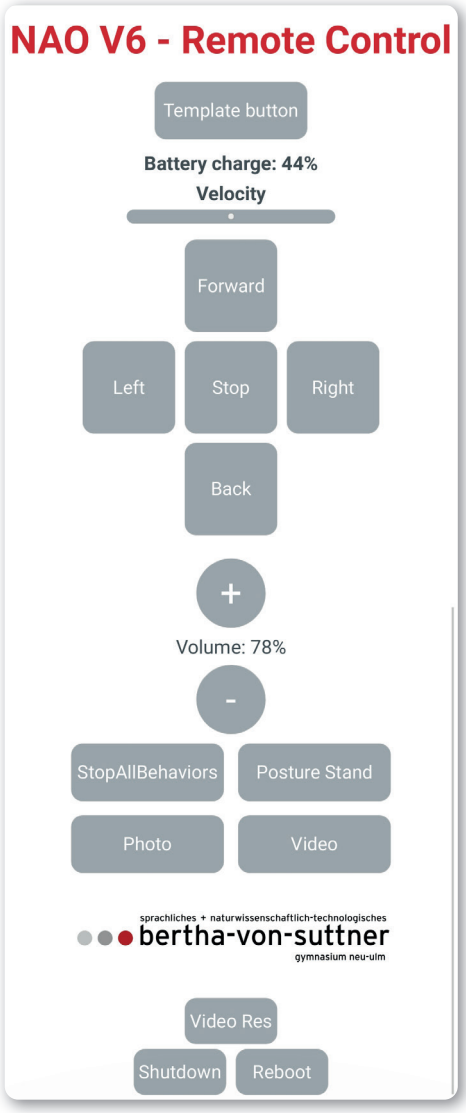
For this project you need:

- The control project template from the virtual archive
- Optional: Several Choregraphe projects that are installed on NAO

The template can be found in the Virtual Archive. You can install the control project on your NAO by clicking on the button at the top left in the "Robot applications" (see Screen 2.20.2.2 – Robot applications).

This project is structured differently from the ones in the previous chapters as the flow diagram from Choregraphe is not used. You will work exclusively in the html folder of the project. To save you building the control website from scratch, you can use the project template. The template already contains the html folder with the HTML, JavaScript, and CSS files.

Functions of the remote control



Screen 2.20.1.1 - NAO V6 control

Here is an explanation of the buttons:

- Template button: This is the template for a control button. You can find out how to program this button below.
- Battery level: The current battery level is shown here.
- Speed slider: You can set NAO's walking speed here. The further to the right the slider is, the faster NAO walks. On a poor surface you need to make sure that the speed is not too high. That means that the slider shouldn't be to the right of the middle.
- Forward, Left, Right, Back: You can use these buttons to make NAO walk forward/ backward and turn.
- Stop: You can press this button to stop NAO's movement.
- +, -: You can make the volume louder/quieter here.
- StopAllBehaviors: This stops all behaviors that are currently running on NAO. Don't worry, nothing bad can happen here!
- Posture Stand: This button moves NAO into the standing position. WARNING: If NAO is sitting down, you shouldn't use this button!
- Photo: You can take a photo with this button.
- Video: You can start and stop recording a video here.
- Video Res: Here you can change the resolution of the video recording.
- Shutdown: Here you can shutdown NAO.
- Reboot: Here you can reboot NAO.

You can access the remote control by typing
“[IP-ADDRESS]/apps/[Application ID]/index.html” into the address bar.
Accessing the photos and videos:
The picture and the video taken are stored on Nao hard drive. You can access Nao's hard drive using ssh or ftp. You can use a tool such as Filezilla. To connect to Nao. You can use the following credentials: "***nao***" as a user and "***nao***" as a password which are set by default. You get the IP address pressing briefly the chest button once.
The pictures and videos are stored in the directory /home/nao/recordings/cameras/

The name of pictures are Photo_YYYYMMDD_HHMMSS.jpg.

The videos and the sound are recorded in separate files. The file names are based on the same pattern with distinct extensions .avi for images and .wav for sound Video_YYYYMMDD_HHMMSS.avi and Video_YYYYMMDD_HHMMSS.wav.
YYYY, MM and DD stands for year, month and day.
HH, MM and SS stands for hour, minute and second.
This is defined in functions declared in the file html/javascript.js of the project.

Suggested solution

The index.html file

Screen 2.20.3.1 - The index.html file

The HTML file is the bare bones of the website. If you want a new button, you can insert the following source code into the template:

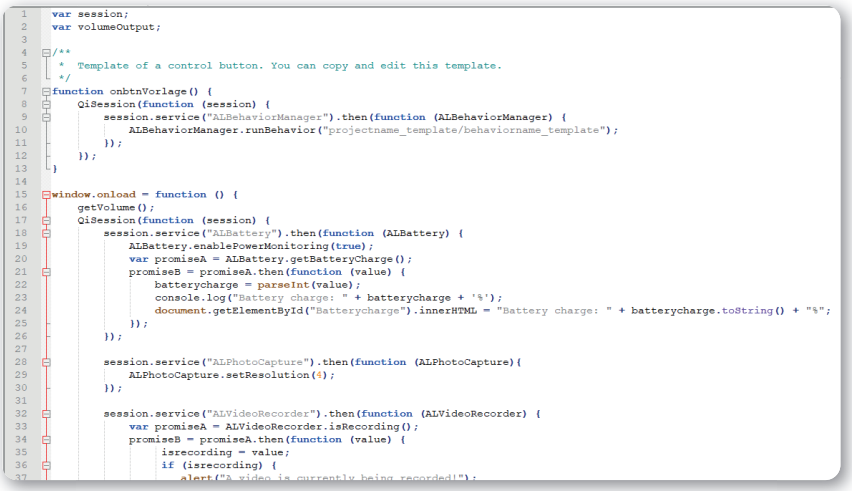
```
<button id="btnTemplate" onclick="onBtnTemplate"
class="controlBtn">Templatebutton</button>
```

Here is a brief description of the attributes from left to right:

- id attribute: The value of this attribute is a unique name for the button. The name should be clear and state what the button does.
- onclick attribute: The value of this attribute is the name of the JavaScript method to be invoked when the button is pressed.
- class attribute: The value of this attribute is the class name of the button. You can use classes to ensure that the control buttons have a uniform design.
- Text between <button> and </button>: You can see this text on the website.

Now you have inserted a button into the website but it doesn't do anything yet. To make it do something, you need to use the JavaScript programming language.

The javascript.js file

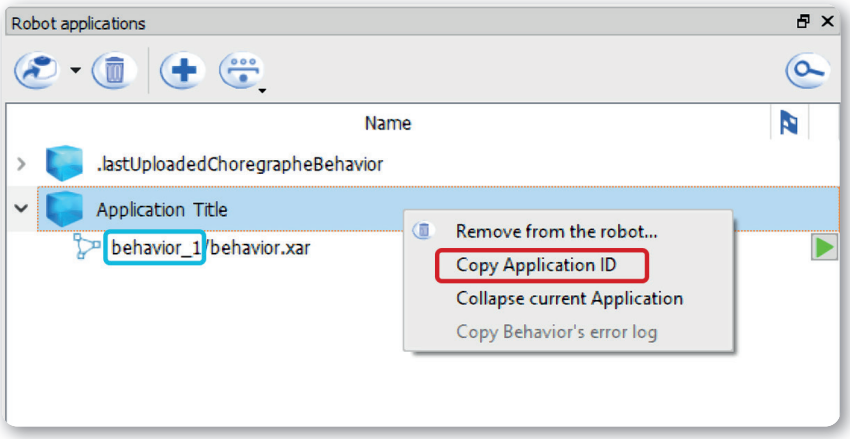


```
var session;
var volumeOutput;

/**
 *  Template of a control button. You can copy and edit this template.
 */
function onbtnVorlage() {
    QiSession(function (session) {
        session.service("ALBehaviorManager").then(function (ALBehaviorManager) {
            ALBehaviorManager.runBehavior("projectname_template/behaviorname_template");
        });
    });
}

window.onload = function () {
    getVolume();
    QiSession(function (session) {
        session.service("ALBattery").then(function (ALBattery) {
            ALBattery.enablePowerMonitoring(true);
            var promiseA = ALBattery.getBatteryCharge();
            promiseB = promiseA.then(function (value) {
                batterycharge = parseInt(value);
                console.log("Battery charge: " + batterycharge + '%');
                document.getElementById("Batterycharge").innerHTML = "Battery charge: " + batterycharge.toString() + "%";
            });
        });

        session.service("ALPhotoCapture").then(function (ALPhotoCapture){
            ALPhotoCapture.setResolution(4);
        });

        session.service("ALVideoRecorder").then(function (ALVideoRecorder) {
            var promiseA = ALVideoRecorder.isRecording();
            promiseB = promiseA.then(function (value) {
                isrecording = value;
                if (isrecording) {
                    alert("A video is currently being recorded!");
```

Screen 2.20.3.2 - The javascript.js file

You can use JavaScript to give your buttons different functions If you want your button to start a behavior installed on NAO, you can use the following source code:

```
function onbtnTemplate() {
    QiSession(function (session) {
        session.service("ALBehaviorManager").then(function
        (ALBehaviorManager) {
        ALBehaviorManager.runBehavior("projectname_template/
        behaviorname_template");
        });
    });
}
```



Screen 2.20.3.3 – Copy project name

Insert the path of the project on NAO into the transfer parameters of ALBehaviorManager.runBehavior("projectname_template/behaviorname_template"). You can copy the project name by right-clicking on your project and selecting "Copy Application ID". The name of the behavior is to the left of the slash.

The stylesheet.css file

```
@charset "utf-8";

body {
    background-color: #FFFFFF;
    text-align: center;
    font-size: 260%;
    position: relative;
    font-style: normal;
    font-family: "Gill Sans", "Gill Sans MT", "Myriad Pro", "DejaVu Sans Condensed", Helvetica, Arial, "sans-serif";
    -webkit-user-select: none;
    user-select: none;
}

H1 {
    text-align: center;
    color: #CB2027;
    margin-top: 20px;
    margin-bottom: 20px;
}

H2 {
    font-size: 100%;
    text-align: center;
    color: #485155;
    margin-top: 10px;
    margin-bottom: 5px;
}

Button {
    background-color: #A4ACB1;
    width: 200px;
    height: 200px;
    color: white;
    text-align: center;
    font-size: 40px;
    margin: 5px 5px 5px 5px;
    border-radius: 25px;
}

Button:hover {
    background: #CB2027;
```

Screen 2.20.3.4 - The stylesheet.css file

This file is the "clothing" for the website. It determines what the building blocks from the HTML file should look like. You can use the template file for your control website. If you want to decide for yourself how your control website should look, you can watch tutorials on the subject online.

NOTES

Acknowledgments

Now you have reached the end of our book. If you read and worked through the book carefully, you should now have a comprehensive basis for realizing your own projects. This book is a milestone, not just for you but also for us. Realizing our dream of writing our own book has been an enriching experience for us.
Teamwork was another important aspect in this cooperative project. As in every major collaboration, mutual trust is essential for working successfully and it meant that through this really close cooperation, we evolved from classmates to friends.

We would like to express our warmest thanks for all the support we received from:

***Heike Schnaubelt** (OStR, senior councilor of stud-ies) (Direction)*	for her encouragement, ideas, perseverance, management, and above all for arranging this project.
***Tony Schuster** (StR, coun-cilor of studies) (Editor)*	for his critical and helpful analysis.
***Mark Lörz** (OStD, senior director of studies)* *(Headmaster)*	for his openness and open-mindedness in realizing the project.
Markus Nerlich *(Managing Director, Technik-LPE GmbH)*	for enabling this project.
Christiane Schulz *(Head of Marketing & Sales, Technik-LPE GmbH)*	for our constant contact through the joint cooperation.
Markus Wagner *(Graphic designer, Technik-LPE GmbH)*	for his advice and compiling the content of our book.
***Louis-Gabriel Pouillot** (Software engineer - Developer Experience SBRE)*	for his technical expertise and solutions needed to complete the project.
***Clement Gossart** (FAE, SBRE)*	for his work and field experience.
***Dev saransh Sood** (Marketing assistant SBRE)*	for his work on this update, and the modifications.

Finally, thank you for reading our book and working through it – hopefully you're now motivated to take a look at our second book "Do it NAO[6] - Creative Project Ideas".

Kai Anter Marcel Greiner Jonas Vatter Jannes Weghake

Acknowledgments

Last year's study skills seminar "The humanoid NAO robot – its use as an avatar, in STEM subjects, and in challenges" (direction: Heike Schnaubelt - OStRin) for her inspiring ideas with NAO5:

- K. Äda: "Industrial robot – NAO in industry"
- J. Bischof: "Cyborgs – blurring the lines between man and machine; use of neuroimplants using the example of the humanoid NAO robot"
- A. Hospes: "The utopia of autonomous driving - a comparison with the NAO"
- R. Kölle: "Robots in the classroom - NAO as a Japanese teacher"
- D. Oppold: "NAO soccer"
- M. Prager: "Robots and their use in old-age care using the example of the humanoid NAO robot"
- O. Rammensee: "Robots make art"
- A. Schmidt: "The humanoid NAO robot as a language teacher using the example of an English lesson (Grade 5)"
- B. Schnitzler: "NAO as a sales assistant"
- C. Schubert: "Robots in surgery; NAO in the operating theater"
- A. Trodler: "NAO robot as a "therapist" in mirror therapy"

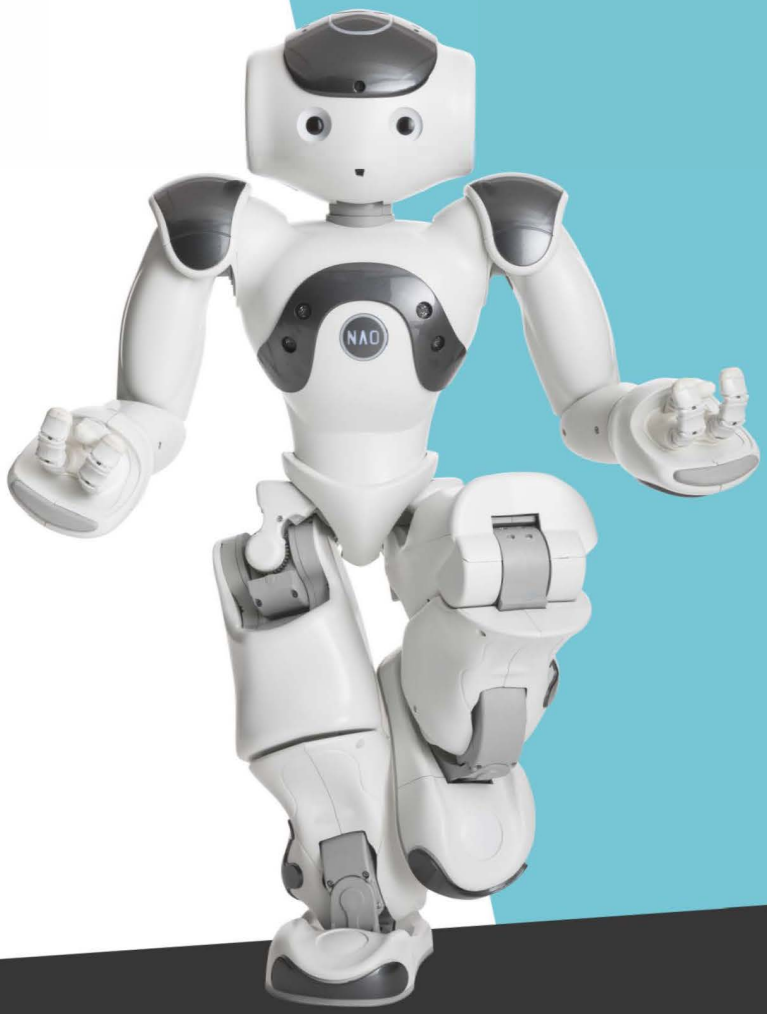
Caspar Sachsenmaier for providing the NAO control program.

Finally, thank you to you for reading our book and working through it – hopefully you're now motivated to tackle your own project ideas.

SoftBank Robotics

EXPAND YOUR COMPETENCE

SCIENCE |
TECHNOLOGY |
ENGINEERING |
MATHEMATICS |



More information:
www.softbankrobotics.com/emea

Contact:
emea-sales@softbankrobotics.com
Documentation:
developer.softbankrobotics.com/nao6

SoftBank Robotics Europe
NAO Robot

@SBREurope
@NaoRobot
@PepperTheRobot

SoftBank Robotics Europe

@sbreurope

SoftBank Robotics Europe