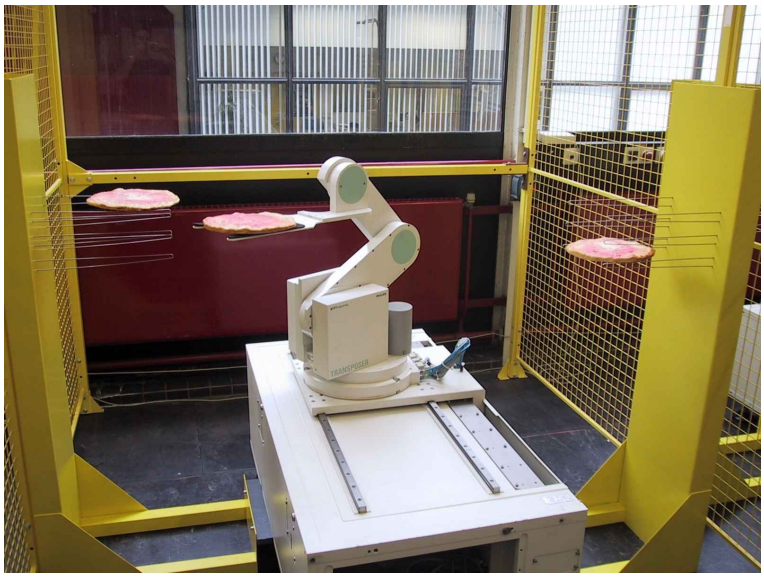
Integration Project Systems and Control (4SC00)



M.J.G. van de Molengraft
J.P. Heldens

February 2013

cstwiki.wtb.tue.nl

Control of an industrial robot

The robot

The industrial robot that will be used in this project was originally designed by Philips. At the Philips factory it was used in the production of LCD-screens. The robot got a second life in education projects at the Department of Mechanical Engineering of the Eindhoven University of Technology. Figures 1 and 2 show a photo and a schematic drawing of the robot. In the schematic the degrees of freedom are indicated. The motors for the horizontal and vertical fork (1) displacement are located inside part (3). The axles inside the arm (2) are not directly driven, but serve as a mechanical transmission. Also there is a motor for the rotation of part (3) relative to the table (4) and a motor for the translation of the table (4) on the base (5).

Figure 1: photo of the robot

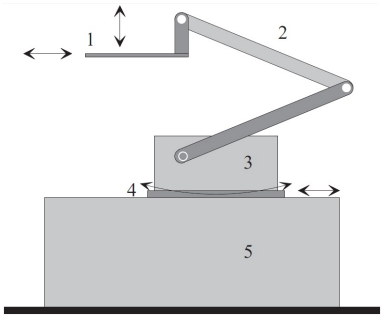


Figure 2: schematic drawing of the robot

Position measurements are carried out with optical incremental encoders attached to the axles of the motors. The relation between the rotation of the motor and the displacement of the fork is linear.

Table 1 shows the limits on position for the different degrees of freedom. Those positions are relative to an initial point [0,0,0,0] for each motor.

Degree of freedom	Minimum position	Maximum position
Vertical fork displacement	-0.2418 [m]	-0.0268 [m]
Horizontal fork displacement	0.0231 [m]	0.5301 [m]
Rotation part 3	0.0108 [rad]	5.5708 [rad]
Table translation	0.0694 [m]	0.4494 [m]

Table 1: maximum positions for all degrees of freedom

An EtherCat stack is built into the base of the robot. The EtherCat hardware acquires the encoder information and controls the amplifiers to drive the motors.

Project goal

In this project the robot will be used to transport pizzas. On both sides of the robot shelves are positioned which can hold up to three pizzas. The pizzas have to be transported from one shelve to the other as fast as possible. At the end of the project a contest will take place.

The challenge of the project is to use as much knowledge from previous control courses as you can for the purpose of creating the winning control design. Hopefully, by the end of the project, you will have discovered the value of the varied earlier acquired knowledge on system analysis, modeling and control design. Especially, in the execution of the work for this project an important role is reserved for experimentation on the real robot.

System identification

To design a good controller you need knowledge about the system to be controlled. This knowledge can be acquired by doing measurements and experiments. For system identification you can think of identification in both time and frequency domain (transfer functions). Think in advance about what you want to know about the system and especially about how you can measure this. For this purpose run simulations with a simple model of the system. Only implement your model on the real system after doing simulations. Some Matlab commands which can be helpful for system identification can be found in the Matlab Signal Processing toolbox. Some examples are: cohere.m and tfe.m

Requisites

You need a computer running real-time Linux with Matlab/Simulink with SOEM EtherCat support installed. A fully functional computer will be supplied with the robot. For running simulations and experiments you need the following files:
- pizza_empty.mdl
- pizza_empty_sim.mdl
- ref3ma_8axes.mdl

For every group of students there is a folder on the computer with these files. This folder can be found in /home/pizza/groupX (where X is your group number). Along with these files some C-files are located in that folder. Those C-files take care of the connection between the software and the robot and should not be altered or removed. The latest information and documentation about the project can be found at the wiki page of the project: cstwiki.wtb.tue.nl. The reference generator (Ref3) can be used to create third order reference trajectories. To carry out an experiment the Simulink model has to be compiled with the use of RTW to create a Real-Time Application (RTA). Compiling of a Simulink model can be done from within the model. During the compilation process a RTW C-code is generated from the model. This C-code will be used for the RTA. The compilation process can be monitored in the Matlab Command window. At last the RTA is linked to the EtherCat stack with the use of Wintarget.

Simulink model for experimentation with the robot

The Simulink model for control of the robot, shown in figure 3, comprises the pizza-interface block and a start/stop block. The start/stop block starts the experiment. The pizza-interface block enables communication between the robot and the model.

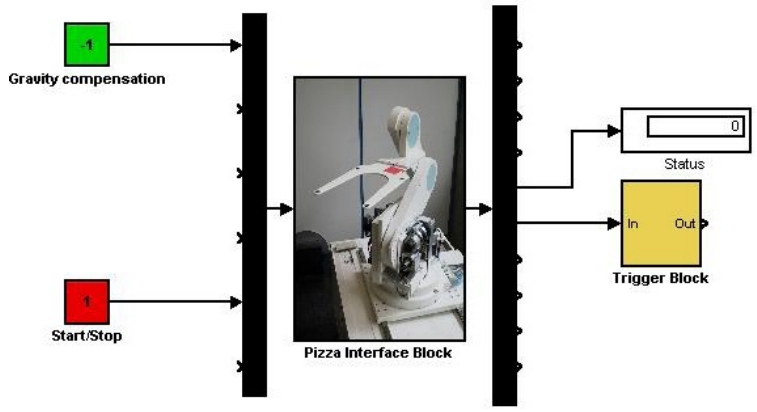


Figure 3: Simulink model for experiments

The inputs to the robot have the following functions and quantities:
Input 1: motor 1 for vertical fork displacement [V]
Input 2: motor 2 for horizontal fork displacement [V]
Input 3: motor 3 for rotation [V]
Input 4: motor 4 for table translation [V]
Input 5: input for activation by the start/stop block

And the outputs:
Output 1: vertical displacement of the fork caused by motor 1 [m]
Output 2: horizontal displacement of the fork caused by motor 2 [m]
Output 3: rotation by motor 3 [rad]
Output 4: translation by motor 4 [m]
Output 5: status of the robot. The numbers indicate the following situations:

- 0: Ready to be started by the start/stop block
- 1: Initialization phase
- 2: Initialization phase
- 3: Robot is ready for use, only in this state you can take control of the robot
- 4: Airbag limit of vertical fork movement has been exceeded. The virtual 'airbag' is the range the robot needs to safely come to a stop when approaching its end position. When this limit has been exceeded the experiment is stopped to avoid damage to the robot.
- 5: Airbag limit horizontal fork movement has been exceeded
- 6: Airbag limit rotation has been exceeded
- 7: Airbag limit table translation has been exceeded
- 8: Speed limit vertical fork movement has been exceeded
- 9: Speed limit horizontal fork movement has been exceeded
- 10: Speed limit rotation has been exceeded
- 11: Speed limit table translation has been exceeded

Output 6: Time after initialization. Use this to start your own controller and trajectory.
Output 7: Output voltage motor 1
Output 8: Output voltage motor 2
Output 9: Output voltage motor 3
Output 10: Output voltage motor 4

Simulation model

The Simulink model for simulations contains the same blocks as the model for experiments as shown in figure 4. The first 10 outputs of the pizza-interface block are identical to the ones in the experimental model. However, the simulation version has 8 additional outputs:
Output 11: vertical displacement of the fork caused by motor 1 [m]
Output 12: horizontal displacement of the fork caused by motor 2 [m]
Output 13: rotation by motor 3 [rad]
Output 14: translation by motor 4 [m]
Output 15: speed of vertical fork movement [m/s]
Output 16: speed of horizontal fork movement [m/s]
Output 17: speed of rotation [rad/s]
Output 18: speed of table translation [m/s]

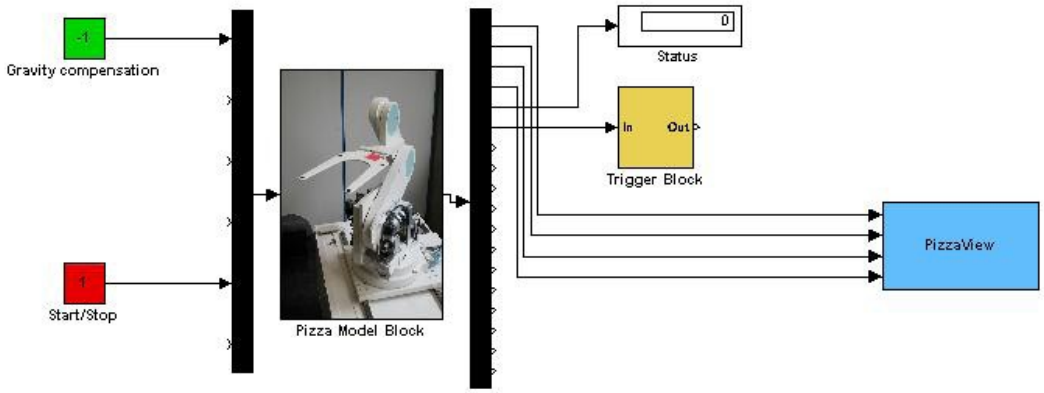


Figure 4: Simulink model for simulations

Experimenting and safety

Always run simulations before carrying out an experiment on the robot. Make sensible choices for input signals, start with 'weak' controllers, one axis at a time. An emergency button is attached to the robot. Always keep this one nearby when running an experiment so you will be able to intervene. Make sure no one is near the robot when starting it.
During the experiment the robot will always start with a homing procedure. This procedure initializes the encoders and moves the robot to the following coordinates:
• motor 1: -0.1 m
• motor 2: 0.1 m
• motor 3: 2.4 rad
• motor 4: 0.2 m
From this position your experiment can be started. During the homing phase output 5 will indicate the initialization is in progress by showing the number 1 or 2. As soon as the robot is ready for use, it will show status 3 and also trigger output 6. Put a trigger/enable block in your controller and trajectory blocks and connect these to output 6 of the pizza-interface block. This ensures both your controller and trajectory will be enabled once the robot is ready for the experiment.

To carry out an experiment take the following steps:

Turn on the robot:
• Put the main switch in upright position
• Enable the switch box, the signal light will light up
• Make sure the emergency button is not pushed down

Turn on the computer. Log in with username 'pizza' and password 'pizza1'.

To start and configure Matlab:
• Double click on the Matlab icon and enter the password
• Change current directory to: /home/pizza/groupX (where X is your group number)
• Open pizza_empty.mdl or your own saved model

To start the experiment:
• In the model, type Ctrl+B to build the Real-Time Application
• Open a terminal
• To go to the right directory, type: cd /home/pizza/groupX
• To start the RTA, type: sudo ./pizza_empty -w
• Enter the password if prompted
• In your model, click Simulation ⇒ Connect To Target
• Wait a few seconds and then click Simulation ⇒ Start Real-Time Code
• Give the start/stop block the value 1 (one)

To stop the experiment:
• Give the start/stop block the value 0 (zero)
• Important: wait until the robot is in its downward position!
• Click Simulation ⇒ Stop Real-Time Code
• Save your data and shut down the computer. Make sure to turn off the robot

Saving data

To collect data during the experiment use an Out block (Simulink ⇒ Sinks ⇒ Out). Other data storage functions like To File or To Workspace can be used in the simulation file but will not work during a real-time experiment. After stopping the experiment a data structure 'yout' with your signals will be saved to pizza_empty.mat in your current Matlab directory. To save your data to a USB-stick plug in the USB-stick and open a file browser by double clicking on the Nautilus icon on the Desktop. Browse to your files in /home/pizza/groupX, copy them and save them to the USB-stick.

Linux commands

Below some useful Linux commands are listed which can be used in a Terminal window:
• cd to change the directory, e.g. cd /home/pizza/group2
• cp to copy files, e.g. cp pizza_empty.mdl /home/pizza/group2/backup
• rm to remove files, e.g. rm pizza_empty.mdl
• ls to list the files in the directory, e.g. ls
• sudo to execute a command with administrator rights, e.g. sudo ls
• sudo su to permanently gain administrator rights, e.g. sudo su
• matlab to start Matlab, e.g. sudo Matlab
• nautilus to start a file browser, e.g. sudo nautilus

Frequently asked questions

What is the sampling frequency of the model?
The sampling frequency of the model is 500 Hz and should not be changed.

What input values can I send to the pizza-interface block and what do they mean?
Inputs 1 and 2 have a range of -6.25 to 6.25 Volts. Inputs 3 and 4 range from -10 to 10 Volts. These voltages are linearly converted by the amplifiers to motor currents which have a relation to motor torque which is also assumed to be linear.

Why do the input voltages to inputs 1 to 4 not correspond to the values of outputs 7 to 10?
This discrepancy is caused by a scaling factor in the software and of course the system dynamics.

Why does the robot start its homing procedure sometimes to the front instead of to the back?
When powering up the robot the encoders do not yet know the robot position. When the robot is very close to the front it may accidentally start homing forward. When this happens, stop the robot, shut it down, manually move it more to the back and restart the experiment.

Axis 4 does not move, or the homing procedure gets stuck when the robot has moved all the way to the back. How can I fix this?
First of all, check if you executed all the necessary steps correctly. If you did, turn off the robot and manually put it somewhere in the middle. Turn the robot on and start your experiment again. If the same problem occurs, you were probably too rough with the robot and now it is broken. Ask your supervisor for help.

When I put noise on the robot for an identification experiment, should I see the robot shaking or hear awkward noises?
NO, immediately press the emergency button in such case!

Why can't I access or copy files in the Nautilus file browser?
You may not have the right permissions because you started a file browser without administrator rights. Starting Nautilus with the Desktop icon should give you administrator rights. You can also start a file browser with administrator rights from a terminal by typing: sudo nautilus.

Can I use dctools blocks to create a controller?
Yes you can. However, the dctools blocks are created with the wrong sampling frequency and will cause an error if you do not correct this. To change the sampling frequency of a dctools block, take the following steps:
• Right click on the dctools block and choose 'Look Under Mask'
• In the S-function parameters field, change the sampling interval from 0.001 to 0.002 (to match the model sampling frequency of 500 Hz)
• Repeat this for every instant of a dctools block used

Why doesn't the robot move at all?
The group before you left the robot with the emergency button pressed.

Good luck!