



University of  
Stavanger

Faculty of Science  
and Technology

Stavanger, January 13, 2025

## ELE610 Applied Robot Technology, V-2025

### ABB robot assignment 2

In the second ABB robot assignment you should make a simple drawing program for ABB robot IRB 140, the one UiS has named Rudolf. Use the RAPID documentation extensively during the work on this assignment.

Approval of the assignment can be achieved by demonstrating the robot program to the teacher, and then submit a report including RAPID code on *canvas*. Only the RAPID code needs to be submitted, note that this should be a txt-file, not the mod-file that Windows will interpret as a video file. Possibly, you may submit the RAPID code in a pdf-file and perhaps also include comments or questions. Make sure that the relevant RAPID code is clearly marked, for example by using Courier font. Nothing but RAPID code needs to be included in the report.

## 2 Drawing program for Rudolf

You don't need to start from scratch in this assignment, a pack-and-go file that contains a station similar to the actual laboratory in E458 can be used as a starting point, download Pack and Go file, [UiS\\_E458\\_nov18.rspag ↗](#), and make sure that file extension is **rspag**.

### 2.1 Open pack-and-go-file

A Pack and Go file is a single file that packages the station data along with the related virtual controllers for archiving and for sharing station data with other users. A Pack and Go file can be created (saved) from the {File}-tab and [Share]-sidebar.

The Pack and Go file, [UiS\\_E458\\_nov18.rspag ↗](#) contains the two robots, Rudolf and Norbert, the conveyor belt between them, the table along the windows and also one table beside each robot. The Pack and Go file was

first made more than 10 years ago, and it has been updated several times to reflect changes in our laboratory and in software, the current is from November 2018 and should work as intended with RobotStudio version 2019 and 2020. However, for the RobotStudio version 2022 there are some problems in Norbert simulation, the puck is not released from the gripper as it used to be. I work on how to solve this, but have yet not found a good solution. Until this is fixed the simulation can only be used to check robot movements, not to show that pucks are moved. On the actual robot the operation of the gripper should work well.

The following list is for opening the Pack and Go file on RobotStudio version 2022 and using version 6.14 of RobotWare. This will create a new project based on the Pack-and-Go file.

- a. Start RobotStudio and open Pack and Go file, [UiS\\_E458\\_nov18.rspag](#) ↗; {File}-tab, [Open]-sidebar,  and locate the `rspag`-file that you should have downloaded to your PC, and start the open-wizard.
- b. Select  and select the target catalog where you want to store the project, typically `R:\Projects\RS2`.
- c. Select  and you use files from your local PC for duplicate files. I am not quite sure of this choice but it is probably better than to use any older versions from the Pack and Go file. Select .
- d. For selection of Virtual Controllers there may be several options, you may select one of the installed versions of RobotWare. I think that the newest one, version 6.14.01.00, is the proper choice here. Note that a virtual controller must be selected for both Rudolf and Norbert, use the same.
- e. Now you will see a window with the selected choices before you continue, this should be as shown in Figure 1. Select .
- f. A warning may be displayed. Select .
- g. Finish unpacking by pressing . Activate the {Home}-tab, and your screen should look something like figure 2.
- h. Investigate the station, in particular the trees in {Layout} and {Paths&Targets} both in {Home}-tab, and the programs in {RAPID}-tab, both for Norbert and Rudolf in the {Controller}-tab to the left.
- i. Run a simulation of the simple paths that are already programmed. You should see Rudolf moving in a simple path above the table, and Norbert picking the puck and moving it. Unfortunately the gripper will not release the puck, it can be done manually by pausing the simulation in

## Unpack & Work

### Ready to unpack

Review the settings below, then click Finish to unpack and open the station

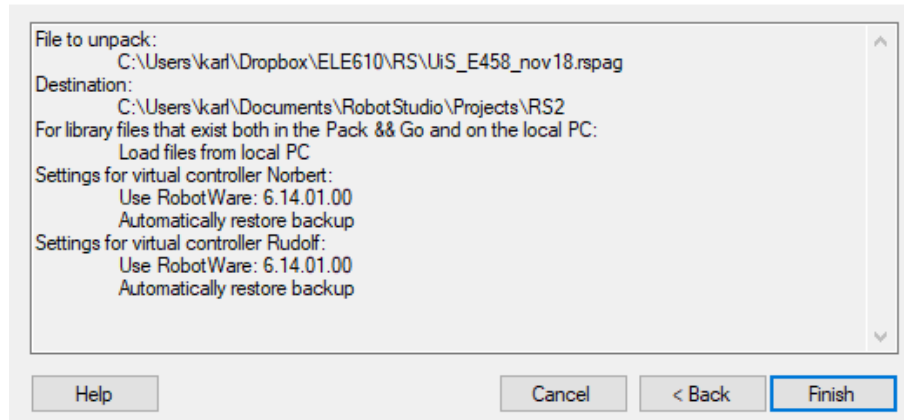


Figure 1: The window with the selected choices before you continue unpacking the Pack and Go file.

the correct position, and then right-click on the puck, in {Home}-tab and {Layout}-tab, and detach it from the object it is connected to (the gripper).

You probably want to update the file description, in {File}-tab, before you store the project. To store click on the disk-icon (store symbol) in upper left part of the RobotStudio window.

## 2.2 Clean up

In this assignment you only need Rudolf, the table beside Rudolf, and the pen attached to Rudolf. Delete everything not needed and check that simulation still works. You don't need the path for Rudolf, just keep one point and let it be 30 mm above the table center. In `main()` simply have on move to this point.

## 2.3 Add A4 sheet

An A4 sheet is wanted on the table, table size is 600 mm  $\times$  800 mm, height is 728 mm 8 mm above `wobj0` (robot base). The sheet is modeled; {Modeling}-tab, [Surface] and [Surface Rectangle]. Its size should be 297 mm and  $\times$  210 mm, its color white and name **A4sheet**. Place the sheet center 0.1 mm above

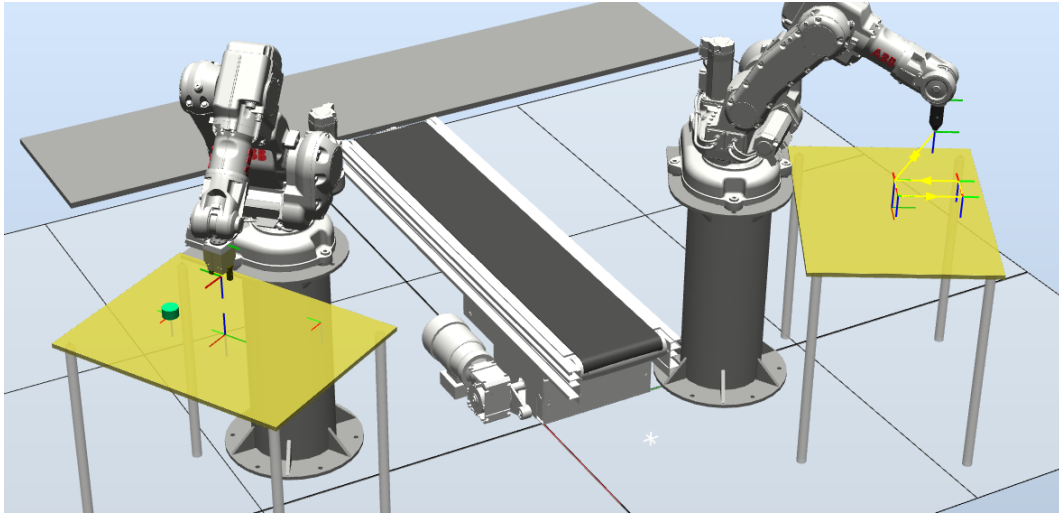


Figure 2: The model of UiS E458 robot cell in the Pack and Go file. Robot Norbert to the left (north) and Rudolf to the right (south).

the center of the table, let the long side of the sheet and the long side of the table be parallel lines. Now you can simply copy the work object `wobjTableR` to `wobjA4` and lift it 0.1 mm up in the RAPID code. All paths to draw on the A4 sheet should use work object `wobjA4`, and if the sheet is moved you just have to move the work object accordingly. For the sheet in the station this can be done by attaching `wobjA4` to the sheet. The `wobjTableR` is not used any more but you can still have it in the RAPID code, or you may comment it out. Make sure that station and RAPID code is synchronized. Check that simulation place the pen 30 mm above the sheet.

You can now store the station; {File}-tab and [Save station] or if a new name is wanted [Save station as].

Finally, move the sheet to another position on the table, synchronize, and check that simulation moves the pen 30 mm above the sheet. Move the sheet back to the center of the table, synchronize, and check again that simulation moves the pen 30 mm above the sheet.

## 2.4 Draw a rectangle

We want to draw a rectangle on the sheet. This can be done in many ways, but the resulting drawing (path) should be the desired shape for all different methods used. Four target points and a path could be made in the station, and then synchronized to the RAPID code as was done in assignment 1. The points and the path could be written directly into the RAPID code, and synchronized to the station. To write code gives the flexibility to define parametrized functions

(procedures) to draw the rectangle. Here, you should write code directly in RAPID. A good way to organize your code may be to let the main module initially be like below

```

1 MODULE main_2_4
2     ! Module for ELE610 assignment 2 section 4, UiS KS Nov. 30,
    2018
3
4     ! wobj in the middle of the table
5     TASK PERS wobjdata wobjTableR:=[FALSE,TRUE
    , "", [[150,500,8],[1,0,0,0]], [[0,0,0],[1,0,0,0]]];
6     TASK PERS wobjdata wobjA4:=[FALSE,TRUE
    , "", [[150,500,8.1],[1,0,0,0]], [[0,0,0],[1,0,0,0]]];
7     PERS tooldata tSimplePen:=[TRUE
    , [[0,0,110],[1,0,0,0]], [1,[0,0,1],[1,0,0,0],0,0,0]];
8
9     CONST jointtarget jCalibPos := [[60,30,0,0,0,0],[9E9,9E9,9
    E9,9E9,9E9,9E9]];
10    CONST robtarget Target_10:=[[0,0,30],[0,0,1,0],[0,0,0,0],[9
    E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
11
12    PROC main()
13        ! MoveAbsJ jCalibPos, v1000, z10, tSimplePen;
14        MoveL Target_10, v500, z10, tSimplePen\WObj:=wobjA4;
15        section24a;
16        MoveL Target_10, v500, z10, tSimplePen\WObj:=wobjA4;
17    ENDPROC
18
19    PROC section24a()
20        ! instructions for this section comes below
21    ENDPROC
22
23 ENDMODULE

```

In the example above I tried to use a simple initial position, `jCalibPos`, but it turned out not to be that simple so the offensive line is just commented out above. I don't know exactly what is wrong or how to correct it, the error messages in RobotStudio are sometimes helpful, but not always, see figure 3. It may very well be that no error will occur if you try this.

- a. First you should simply define the target points needed to draw a rectangle of length 70 mm and height 50 mm with one corner in origin and orientation as the A4 sheet. Copy and change the line where `Target_10` is defined, and add `MoveL` instructions to `main()` function. Synchronize and check simulation. To better view the path the tool center point has followed it can be shown in a selected color; {Simulation}-tab, [TCP trace] and a dialog window appears in the left side bar. Make sure that speed and zone parameters are appropriate.
- b. To add flexibility a rectangle drawing procedure should be made, see RAPID documentation as in section 1.2 in assignment [rs1.pdf](#) ↗. The

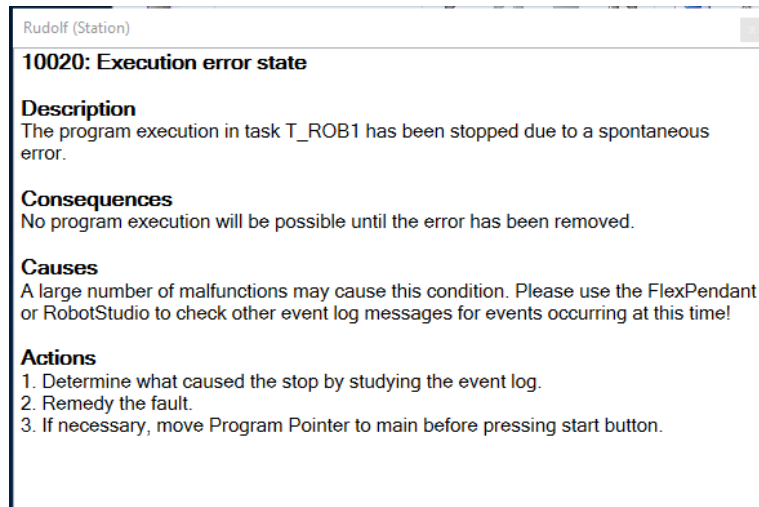


Figure 3: Example of a not very helpful error message.

procedure should take one `robtarg`, one corner point of the rectangle, and `length` and `width`, two numbers, as arguments. The procedure should move fast to a point 10 mm above the given point, slowly down, draw the rectangle aligned with the sheet, then move up 10 mm again and return. The function `Offs()` will be useful, see RAPID documentation. Use this function to draw three rectangles within each other, sizes  $70 \times 50$ ,  $80 \times 60$ , and  $90 \times 70$ . Synchronize and check simulation. Make sure that speed and zone parameters are appropriate.

The following simple RAPID code example can be helpful.

```
PROC goUpDown(robtarg p1, num times, num dist)
  ! move fast to dist above the given robtarget
  MoveL Offs(p1, 0, 0, dist),v200,z1,tSimplePen\WObj:=wobjA4;
  FOR i FROM 1 TO times DO
    ! move slowly down to the point
    MoveL p1, v50,z1,tSimplePen\WObj:=wobjA4;
    ! move faster up again
    MoveL Offs(p1, 0, 0, dist), v100,z1,tSimplePen\WObj:=wobjA4;
  ENDFOR
ENDPROC
```

This example function can be called from main or another procedure by  
`goUpDown target_10, 2, 40;`

## 2.5 Draw triangles and circles

Make a function `drawTriangle` that draws an equiangular triangle with center in a `robtarget` given as an input argument, and side length  $s$  as the second input argument. One side should be parallel with the x-axis of the work object `wobjA4`. Note that distance from center to each of the corners then is  $\frac{1}{3}\sqrt{3}s$ , and that lower left corner has coordinates  $(-\frac{1}{2}s, -\frac{1}{6}\sqrt{3}s)$  relative to the center.


Then make a function that draws a circle with center in a `robtarget` given as an input argument, and radius as the second input argument. Use the `MoveC` instruction, see RAPID documentation.

Modify the `main()` function to draw a rectangle of size  $100\text{ mm} \times 100\text{ mm}$  centered at origin of work object `wobjA4`, lower left corner of rectangle is then  $(-50, -50)$ . Then, at each corner of the rectangle, draw a triangle with sides 25 mm and a circle with radius such that the perimeter goes through the triangle corners, I think this gives  $r = \frac{25}{3}\sqrt{3}$ . Synchronize and check simulation. Make sure that speed and zone parameters are appropriate.

## 2.6 Program control

Look up RAPID documentation on instructions `FOR`, `GO TO`, `IF` and `WHILE`. Make a function `drawShapes` that, centered in a given `robtarget`, draws a given number of squares and circles, the first square should have sides 120 mm, the first circle should have radius 60 mm, thus fitting perfectly inside the square. The next square should fit perfectly inside the last circle, and so on each object fitting inside the previous one. The number of objects to draw should be given as an input argument to the function. Check simulation.

## 2.7 User control

Look up RAPID documentation on instructions `TEST`, `TPWrite`, `TPReadFK` and `TPReadNum`. A virtual FlexPendant can be displayed during simulation, and used almost as the actual robot FlexPendant. The simulation can be started pressing  symbol on virtual FlexPendant. Use the virtual FlexPendant to select which program part to run; `section24a`, `section24b`, `section25` or `section26`. For `section26` you should also be able to select number of shapes to draw inside each other. Synchronize and check simulation and store the program. The main module should be submitted in *canvas* as txt-file.

The simulation should be shown to teacher. When this is approved you are ready to run the program on Rudolf, as explained in next section.

## 2.8 Run program on Rudolf

The first time you use the robot the teacher, Ståle or Karl, should join you and teach you the essential security routines. When you know these, you may work on the laboratory without the teacher but not alone.

Before you can run the program you should check that the correct tool is mounted on the robot, if not this must be done. Get acquainted to the robot by controlling the robot from the FlexPendant (jogging). Place a table next to Rudolf and place a sheet on the table, it may be an A3 sheet even if the model in the station is an A4 sheet. To control the robot from your program you should use your own laptop or one of the computers in E458 (username: Robot, password: abb). If you use your own laptop, make sure that you use the **robot** wireless network (password: NorbertRudolf).

Start RobotStudio but you do not need to load a station (model), and connect to physical robot Rudolf: {Controller}-tab and [Add Controller]. You have to give the address to Rudolf 152.94.160.197 in the dialog window that appears. By the way, Norbert has address 152.94.160.198. Click [Rudolf] and  button and the actual robot should be in the left column. Acquire write access: {Controller}-tab and [Request Write Access] and confirm on the physical FlexPendant. Then delete program already loaded into the robot: click on **Program** in the controller tree in the left column, then [Delete] in the menu.

The program can now be copied from your computer, or a memory stick, into the controller: click on **Program** in the controller tree in the left column, then [Load] in the menu. Locate the program, a pgf-file in the catalog where it was store. Activate the program after it has been loaded into the {Rapid}-tab: + to store and ++ to apply. Run program step by step: Press  symbol on FlexPendant. After the instruction that place the pen 30 mm above the center of the sheet you should check that the location of the sheet (actually its work object) in the station (model) and the actual location of the sheet are the same. To check the height, use one of the pucks which has 30 mm height and test if the pen position relative to the puck placed on the sheet center is as it should be. If not it may be needed to adjust the work object **wobjA4** for the actual robot. When this is done, the program should run as in the simulation,  symbol on FlexPendant. Check this, then show it to the teacher.



## 2.9 Triple pen and more shapes

This part is optional. You should only do it if you have more time, the assignment should be 15-20 hours. As the physical triple pen is now out of order, this part should only be simulation in RobotStudio.

Try to replace the simple pen with the UiS triple pen, library file `TriplePen.rslib` ↗ and program module `TriplePen.mod` ↗. You may draw the same shapes as in section 2.7, perhaps also add an option in the menu to let you select which pen to use. You may also add your own shapes. Enjoy.