Stavanger, June 19, 2026

Universitetet
i Stavanger

Det teknisk-
naturvitenskapelige fakultet

ELE610 Applied Robot Technology, 2026

ABB robot assignment 1

In this first task you will be introduced to the ABB RobotStudio software package and the fairly comprehensive documentation that comes with it. The first 11 sections of this assignment should be done, sections 1.12 is optional but should be done if you have time.

Approval of the assignment can be achieved by showing the simulation to the teacher, and the group should submit, on *canvas*, the RAPID code. Note that the code file, `*.mod`, should be renamed into `*.txt`, since Windows interpret `.mod` as a video file.

1 Introduction to RobotStudio

You can use RobotStudio as installed on PC on your desk (work-place) in E462 or E464. We should have everything correctly installed there when the course starts. You may also want to install RobotStudio on your own **Windows** laptop computer, thus allowing all group members to work on RobotStudio concurrently and also to work on RobotStudio elsewhere (at home).

The two PCs closest to the robots in laboratory E458 also have RobotStudio installed, actually several older versions of RobotStudio are installed on them, so be aware of which version you start. These PCs, or your own Windows laptop PC with RobotStudio installed, can be used to load RAPID programs onto the physical robots. Then the program can be started using the physical FlexPendant connected to the robot.

Karl Skretting, Institutt for data- og elektroteknologi (IDE), Universitetet i Stavanger (UiS), 4036 Stavanger.
Direkte 51 83 20 16. E-post: karl.skretting@uis.no.

1.1 Install RobotStudio on your laptop

To install RobotStudio on your own laptop, the ABB RobotStudio Technical data sheet ↗ says you should have Windows 10 or later. You can **not** install RobotStudio on an Apple computer. On your Windows laptop computer you can install Thus, most of you will prefer to install ABB RobotStudio Suite ↗.

Download a huge (approximate 2 GB) zip-file from ABB download page ↗, perhaps you have to register to get a link to the file.[1] Normally you select the newest version, which now is RobotStudio Version 2026.2, release date: May 27, 2026. Older versions should also work well, I (Karl) have RobotStudio 2025.3 (64-bit) Version 25.3.11371.0 on my own laptop and I don't think I need to update this. Unpack zip-file into a (temporary) directory. Locate and run `Setup.exe`, and follow the installation wizard. Select full installation and use standard options. RobotWare can be installed from the Add-Ins Gallery within RobotStudio, see section 1.1.2. ABB allows users to try RobotStudio, with limited options, without a license. To activate a license see section 1.1.3.

The program will be installed, perhaps in directory `C:/Programfiles(x86)/ABB/RobotStudio2026`. The program `RobotStudio.exe` should be in the `Bin` directory here, and you will probably want to make a shortcut to this program file from the desktop of your computer. You may start the program now and then get a window like shown in figure 1. But before you continue to create your project you should check out the three things in the following subsections. When you have decided where to store your projects, and have installed the RobotWare controller and have activated the license you create a new project.

1.1.1 Storing of projects and path problems

Important terms here are stations and projects, but also RAPID programs stored i modules and Virtual Controllers are important to understand. For complete information see RobotStudio Help, in particular section "Building Stations" part "Understanding stations and projects". A brief explanation is that a station is a digital twin of a real robot cell. A Project is the higher-level container that stores: One or more Stations, Robot controllers and backups, RAPID code files, System modules, Configurations, Libraries and assets and Simulation settings. I think a good approach is to store each RS assignment as a project, and then different stations, and RAPID programs can be stored as work progress. The assignments may not yet be completely up to date on this, as parts from earlier years may still be present. I will try to update the assignments as we go along and unclear (misleading) explanations are revealed.

[1] The zip-file may also be available on the PCs in E462 and E464, on `C:/ELE610/Prog/RobotStudio.yyyy.v.zip` where `yyyy.v` is year and version number, Asbjørn should check this.

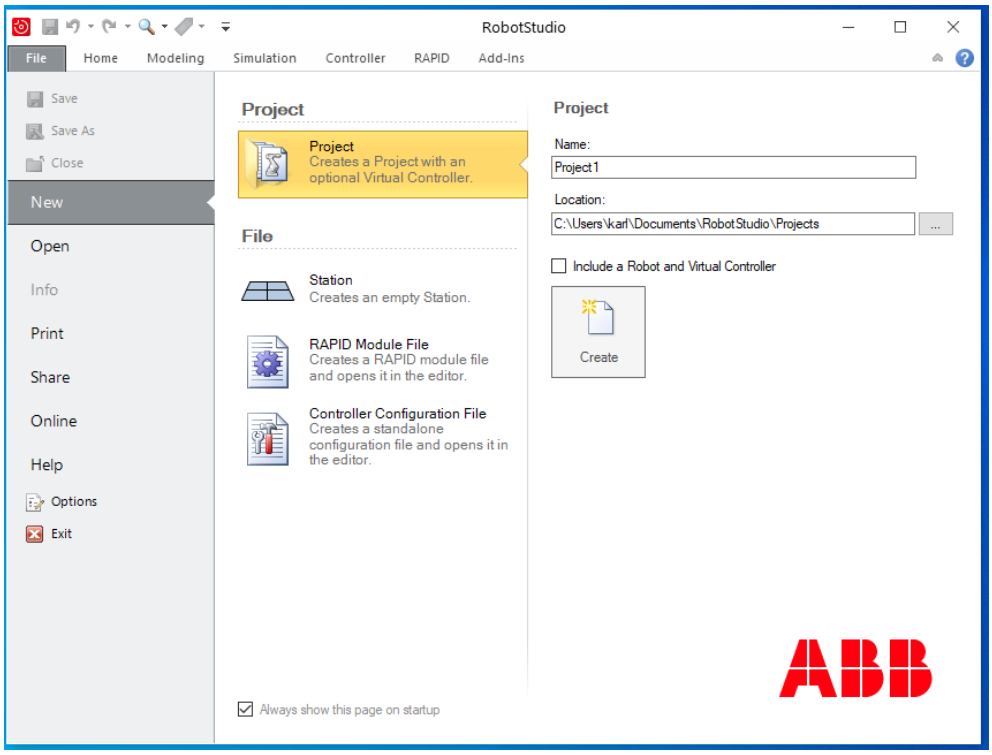

Figure 1: The RobotStudio start window as it is in older (2022) version. Version 2025 is the same, and I guess this is so also for 2026 version.

After a project is created you should look at the project directory tree structure and see the files created. Can you locate the RAPID program module, it is a simple text file that can be viewed in any text editor. Also look at the project directory after you have saved different stages of the station and RAPID program. The file paths used when RobotStudio stores files are[2] more restrictive that the Windows OS. There is, or may be, a problem using network-paths in file names. Also, several characters in RobotStudio file names (and directory names) are prohibited, most notable spaces and perhaps special non-ANSII-127 letters. Below are two suggestions for solving, or rather avoiding, this problem:

[2]Perhaps the latest versions of RobotStudio handle paths "as it should be", but at least in older versions there are problems.

- The more **simple** way:
  Use C-disk on your own computer or on the PC in laboratory, you may store your *projects* under `C:\ELE610\RobotStudio`.

- The more **flexible** way:
  You may still want to store your files in a particular directory other than `C:\ELE610\RobotStudio`. This can be on your UiS network directory (F-disk or OneDrive) `..\Documents\ELE610\RobotStudio` or somewhere on your DropBox (or any other cloud?) directory, like `..\Dropbox\ELE610\RobotStudio`. Be aware that UiS network and cloud directories sometimes(?) contain special characters that violate the strict RobotStudio limitations. The application may hide this, so even if the path name looks fine there may be problems in RobotStudio.

  If you want to store (user) files in a particular directory where the path violate the strict RobotStudio limitations this can be solved by using the `SUBST` command in a (command prompt) window:
  `C:\> subst R: C:\Users\username\Documents\ELE610\RobotStudio`
  This should give you easy access to your RobotStudio files. Also, this is the way used in these ABB assignments to refer to (local) RobotStudio files; the directory where you save your RobotStudio data is simply referred to as `R:\`.

It is recommended that you either save (update/overwrite) the same project, or save everything as a new project. However, it is still possible to save individual files, ex. RAPID mod-files, anywhere. This may be wanted when you want to transfer a program (or a module) to an actual robot. If you want to copy a project to another PC this is more complicated than just copying some files, RobotStudio tools (function), Share pack-and-go file, should be used (correctly, thus you need to know these tools).

NOTE! When you save a RAPID module as a text file somewhere, the short cut is Ctrl + S, the file you see in the RobotStudio editor will be the saved file, and not the file connected to the Station `T_ROB1`. If you continue to edit this file it will not have any effect on the Station. You should close the file in the editor, and reopen the file connected to the Station `T_ROB1`. To save the (station RAPID program module) file in the editor to the station (denoted as Apply All) the short cut is Ctrl + Shift + S.

1.1.2 RobotWare Controller

A *controller* is needed to debug or simulate (run) a program. RobotWare is the virtual controller that comes with RobotStudio, it is installed from within RobotStudio. To do this an Add-In is needed:
From the startup screen, figure 1, select {Add-Ins}-tab. You may have to press

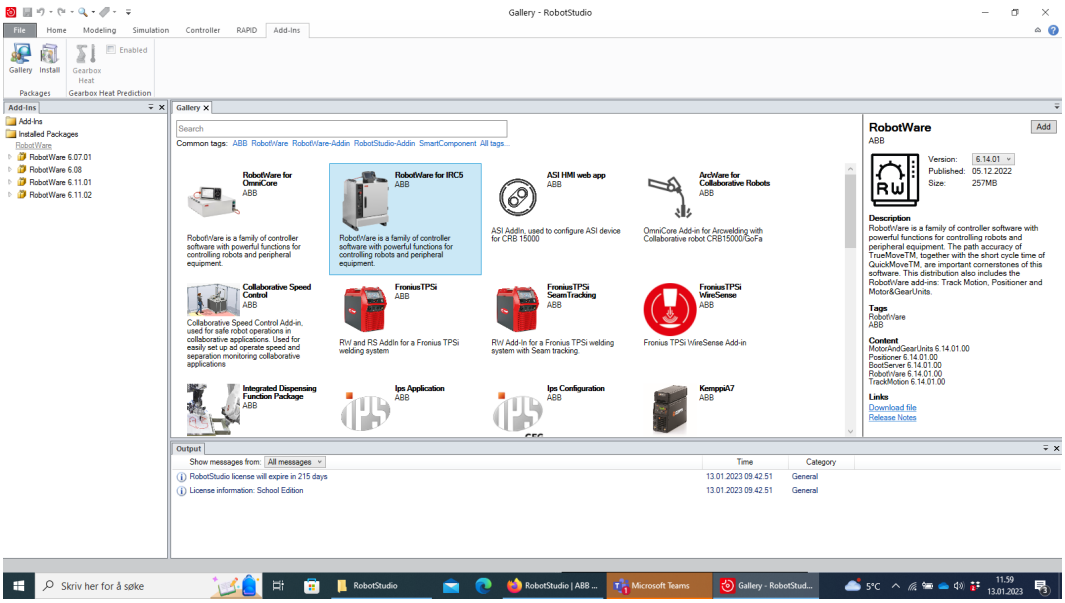

Figure 2: The RobotStudio Add-In window. You notice that 4 older versions of RobotWare are installed, but the newest one 6.14.04 is not installed yet.

the Gallery-icon to the left, and you see a screen like in figure 2. Select the [RobotWare for IRC5] Add-In. Details for this Add-In is shown in the right side-bar of the window, you may see that it is already installed and a Remove button is in the upper right part of the window. If it is not installed a Add button is shown, press on this to install the Add-In. A green bar in the lower right part of the window show installation progress. When it is added you can go back to the {Add-Ins}-tab.

When you install RobotStudio, version 2023.2 or newer, all the documentation is not included. In particular documentation on RobotWare which include RAPID instructions, functions and data types is not installed. This must also be installed as an Add-In, add the RobotWare Documentation Package for IRC5. Then the documentation should be available under File-Help-IRC5 Documentation part.

1.1.3 License

From the startup screen, figure 1, the {File}-tab is the active tab. Select [Help] (in the left sidebar) and you will see license information in the right part of the window, this is as shown in figure 3. To activate a license select [Manage Licenses], the dialog should be as in figure 4. Select Activate Wizard and and you get a new dialog box as in figure 5. Here you should first specify a network license server and in the next dialog window enter the UiS license server, `rs.lm.ux.uis.no`, then when RobotStudio is restarted the license is

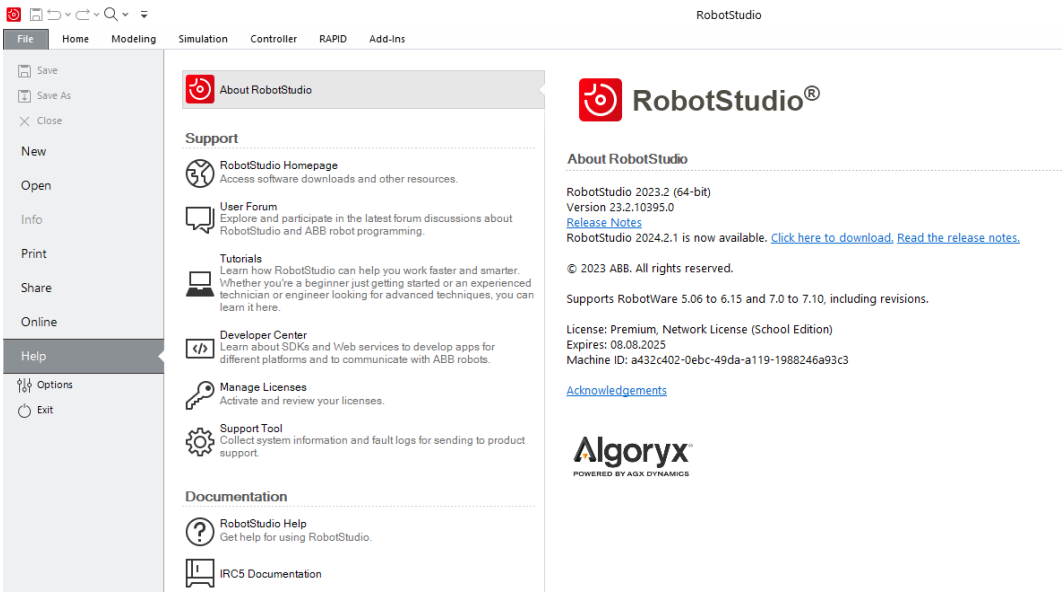

Figure 3: The RobotStudio Help window. Notice the documentation shown at the bottom in this image, and the About RobotStudio part to the right. Both version and license information is shown here.

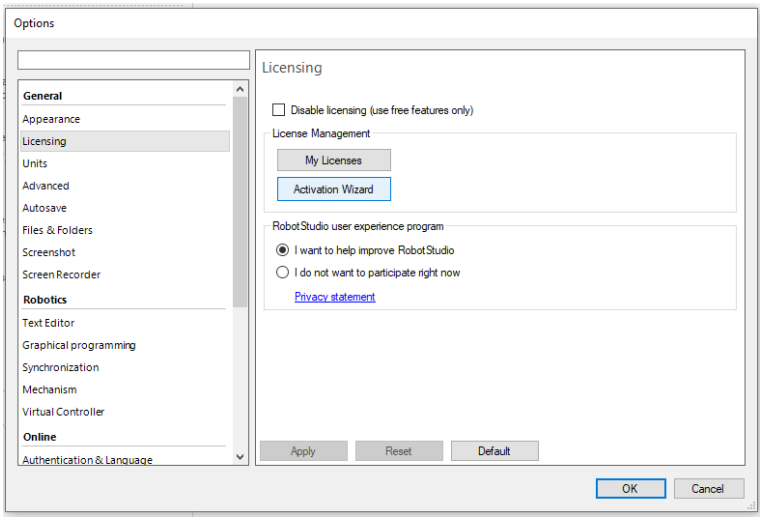

Figure 4: The RobotStudio Manage Licenses Options window

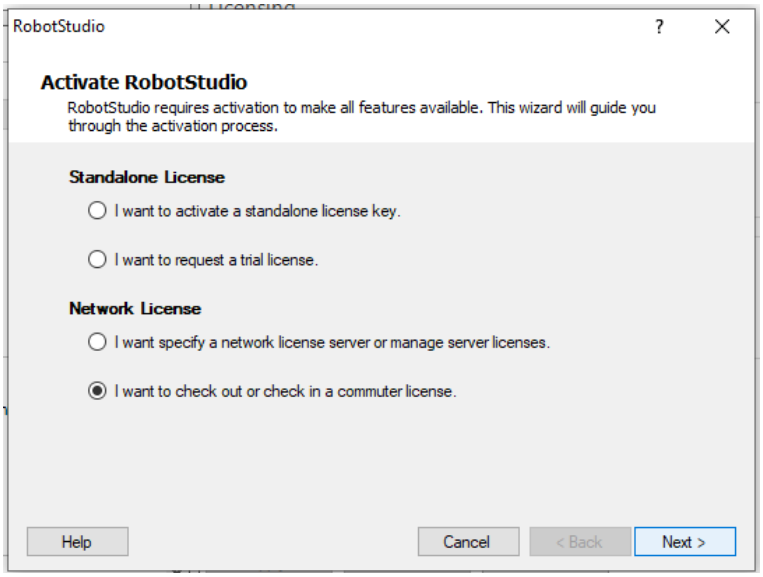

Figure 5: The RobotStudio Activate window

activated.

To be able to use RobotStudio on a laptop from anywhere you have to start Licensing Activation Wizard ({Home}-tab, Help item in leftmost part, and click on Mange Licenses to get Options window) figure 5, and on the bottom of the first dialog check "I want to check out or to check in a commuter license", figure 6, and check out a commuter license for the wanted number of days, I suggest 30 days.

1.1.4 Create new project

Restart RobotStudio and select "New" in the left part of the window, figure 1. Select project name, perhaps `RS1`. Give the location as where to store your projects. Check the "Include a Robot and Virtual Controller" check-box in the window and some new fields appear. If the controller location is correct[3] you select robot model as "IRB 140 6kg 0.81m and newest RobotWare version, this will update the Controller name. Leave "Custom options" and "Create from backup" open. Finally press the large Create icon. You have to confirm the IRB-140 robot in a popup-window. When this is done the window should be like in figure 7. If the lower right field in this window is green you may continue to learn RobotStudio.

[3]If not correct a yellow warning sign is shown and you need to correct this before you continue.

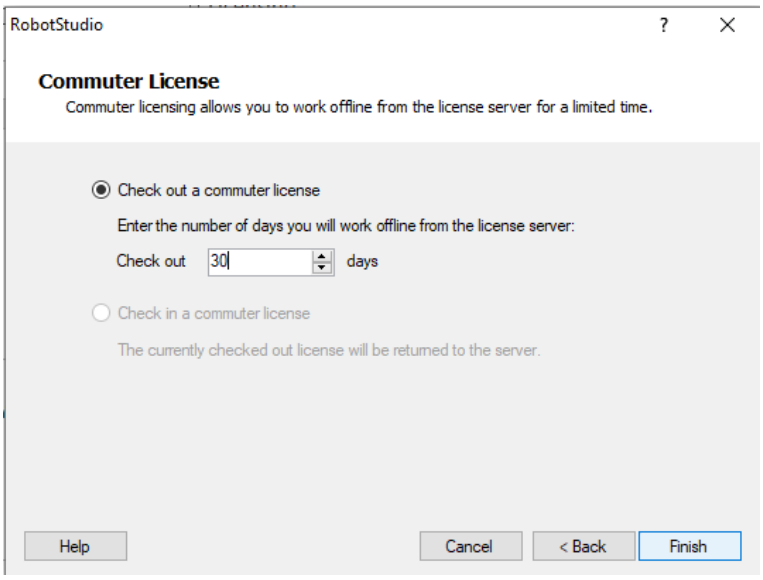

Figure 6: The RobotStudio Activate window

1.2 Learn RobotStudio

To learn RobotStudio is very ambitious point to include here. It is not realistic to expect that you will be a RobotStudio expert after some few hours of study, however I hope you will get an overview of the program, and get sufficient knowledge to search for, and find, relevant documentation for the tasks to do in the RS assignments.

You should examine [Help] item, in the left sidebar under the {File}-tab, and browse the comprehensive RobotStudio documentation. The RAPID documentation inside the IRC5 documentation box is very relevant, in particular the RAPID overview document and the document on Instructions, Functions and Data types. In the RAPID overview document the first parts of Basic RAPID programming and the the first parts of Motion and I/O programming are especially useful. You will need to refer to the documentation all the time when you use RobotStudio, at least for the first years.

You create a **project** and include the robot as in section 1.1.4, and the screen may look like in figure 7. A project can contain several stations and program modules. Storing files is more explained in section 1.1.1, and as a consequence in this document the RobotStudio document directory is renamed (`SUBST`) as `R:\` and thus this project location is denoted as `R:\Projects\RS1`.

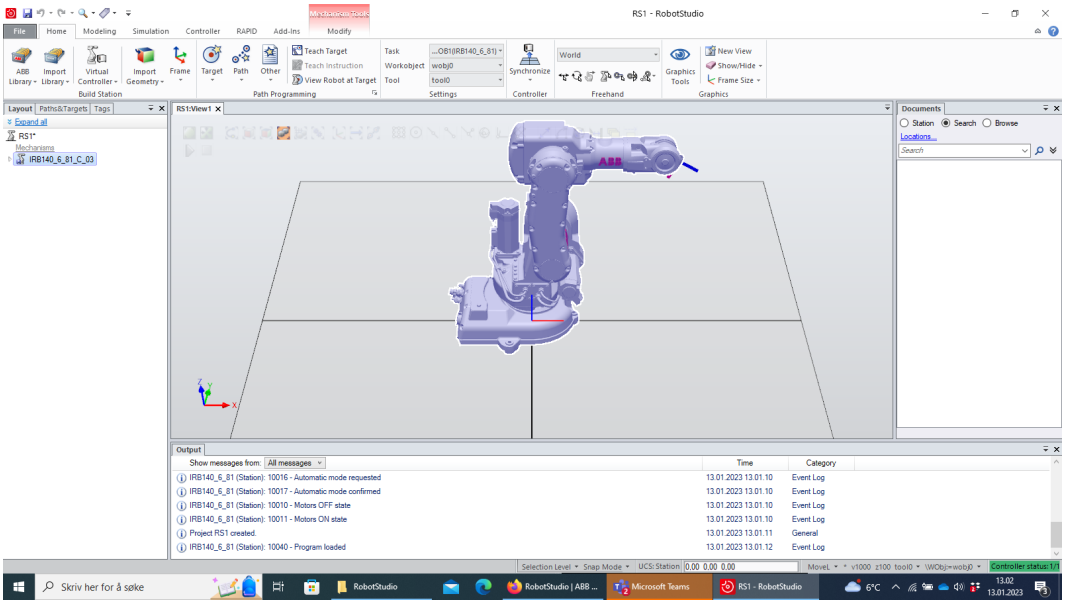

Figure 7: The RobotStudio window after a new project has been created. This window has many parts, from top to bottom we have:
1) The *title line* (top line) contains the general menu part to the left, the project name centered, and some general Windows actions to the right.
2) The *tab line* contains {File}-tab, {Home}-tab and more tabs to the left and Help-elements to the right.
3) The *tool ribbon*. This can be hidden (minimized) by the small arrow-up symbol in the right part of the tab line. Clicking this will hide the tool ribbon below it, and change symbol to an arrow-down symbol, clicking once more the tool ribbon is shown again. This is good to know if the tool ribbon disappear.
4) The *work area* is the main part of the window. The layout can be changed; The customize symbol (bar above triangle down) in general menu on title line can set this to Default Layout as shown in this figure. Here, the robot station is shown, also code may be shown in work area. Notice the (weak) station *view menu* in upper left corner of work area.
To the left of work area a tree-view of various things (depending on context) can be shown, here the {Layout}-sidebar is shown. To the right an overview of documents can be shown. The output part below work area shows messages. Double click on a message to get more information.
5) The *status line* is on the bottom of the window, just above the Windows-line at the bottom of the screen. Notice the green background for the controller status in the rightmost position here. If this is red or yellow RobotWare Controller is not connected, see section 1.1.2.

From a window like in figure 7 you may rotate and move the view of the robot, use Ctrl and Shift and mouse. The wheel on the mouse (if present) can be used to zoom in and out. You can give a project description in [Info] on {File}-tab. To save project and end RobotStudio; {File}-tab and [Save Project] and {File}-tab and [Exit].

1.3 Attach tool to the robot

Start RobotStudio again and open the project; {File}-tab and [Open] and you probably find your project on the top of the list Recent. A tool should now be attached to the robot. The tool may be from a standard library of tools (Import Library - Equipment), or it can be specially designed for the customer. To design a tool in a proper way in RobotStudion is not a trivial task. Here, you should first use the tool for the UiS pen. If there is time, you may try an alternative tool in the end of this assignment.

Download library file `UISpenholder.rslib` ↗. This is actually a zip-file, and the web browser may rename it when you download the file, make sure the extension is `rslib`. In RobotStudio import the library file; {Home}-tab, [Import Library] and [Browse for Library] and locate the rslib-file and open it. The tool should appear in the {Layout}-sidebar just below the IRB-140 robot. You can now drag the tool onto the robot in the {Layout}-sidebar, **not** the graphics in {RS1:View1}-window. Update the position and the pen should be attached to the hand of the robot in {RS1:View1}-window. You may right click on the tool in the {Layout}-sidebar and a menu should appear, look at this but don't change anything yet.

1.4 Include table and work object in station

Include the propeller table to the station; {Home}-tab, [Import Library] (the small down-triangle in the lower right) and [Training Objects]. The propeller table should be at the bottom of the list and appear in the {RS1:View1}-window when it is selected. You can now activate the "Snap Object"-item (3-by-3 grid), the view menu is weakly shown in the top of the work area, {RS1:View1}-window, perhaps you need to press the expand-symbol to the right of view menu first. A small ball is now shown at the closest object (point) near the tip of the mouse pointer (in the {RS1:View1}-window). When you left-click on the mouse this point is selected and the coordinates are shown in the status line.

The four top corners of the table should have coordinates (425, 150, 308), (825, 150, 308), (825, -150, 308), and (425, -150, 308). Let us here denote these points as P_1, P_2, P_3, and P_4 respectively. We will now create a coordinate system (work object) centered at P_4 and x-axis towards P_3 and y-axis towards

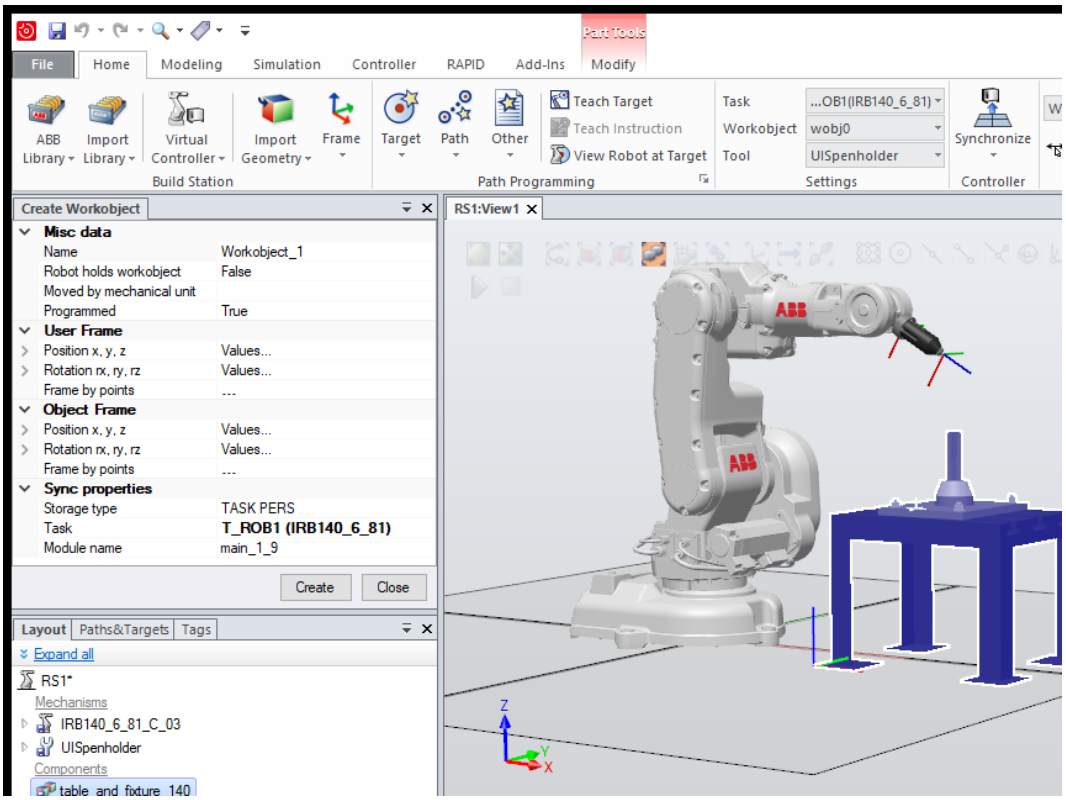

Figure 8: The upper left part of RobotStudio window when creating a work object.

P_1 and thus z-axis upwards. Thus, the x-axis is along one long edge of the table and the y-axis along one short edge of the table.

In RobotStudio a work object is used to define all robot movements, the path a robot should follow is given by points related to a work object. In {Home}-tab the Setting part of the tool-ribbon shown in Fig. 7, contains three selection boxes stacked on top of each other: [Task], [Workobject] and [Tool]. Probably [Workobject] shows [wobj0]. Now we want the table as our work object with name `wobjTable`. See RAPID Data types documentation for the difference between `User Frame` and `Object Frame`.

Define the new work object using {Home}-tab, [Other] in tool ribbon, and [Create Workobject]. Left side bar will show the {Create Workobject} dialog window, see figure 8. Give the name as `wobjTable` and click on the User Frame `Frame by points` text, and then click on the small [v] button that appeared just right of this text. Now the three point input dialog appears, see figure 9.

The first given point should be the origin of the work object, the second point the direction along the x-axis and the third point gives the direction of the y-axis. To fill in values, click on first point in dialog window figure 8 (upper

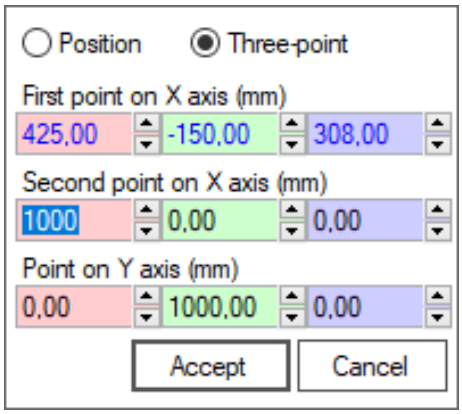

Figure 9: The RobotStudio three point input dialog window. The first point, origin (P_4), is already entered.

red box), then switch to the work area {RS1:View1}-window and zoom in on the table, select [Snap Object] from the see through symbols at the top of the {RS1:View1}-window, and click on the origin of the work object (top of corner of table), then give the other two points (two other table corners). You should notice that the z-values for all three points are 308 [mm]. Finish by pressing Accept and Create and the work object is created and shown as an axis cross on the table in the {View1} window. The red line point in x-direction, the green line in y-direction and the blue line in z-direction. In the tool ribbon below {Home}-tab the value of [Workobject] should be updated as well. If the work object (axis system) is not as you want it, delete it and try again.

The work object should be attached to table, so when table is moved the work object should follow. In the {Paths&Targets}-tab in the left sidebar (still under {Home}-tab) you can expand the tree, locate `wobjTable` and right click on it. In the menu select [Attach to] and chose [table_and_fixture_140] without updating the position. Check that when you move the table the work object follows.

When the table now is moved the work object should follow. Place the table with the long edge towards the robot, 100 mm closer to the robot and still centered in front of the robot. Notice that the top size of the table is 400×300 [mm], but the local reference is on the outer corner of the plate of one of the legs, 25 [mm] outside the top (and 308 [mm] below). Rotate and translate the table ({Home}-tab and {Layout}-tab and right-click on table) to get the wanted position for both the table and the work object. The work object `wobjTable` should be on table corner left and close to the robot, denoted as P_4 above. The world coordinates of the of this point, (origin of `wobjTable`) should be (325, 200, 308). P_1 should have world-coordinates (625, 200, 308).

You can now store the project; click on the disk-icon (store symbol) in the

general menu on the title line of the RobotStudio window. You may also store a copy of the station up to this point, {File}-tab and "Save Station As", give the file name and save the station, ex. as RS1-4. Continue to work on the project RS1.

1.5 Define path

The robot has still not moved, but this will happen soon. In the tool ribbon under the {Home}-tab there is a section named `Freehand`, here is a `Jog Joint` symbol+text. Click on this and then on the robot in the work area and move a selected axis by dragging it. Also try the `Move and Rotate` option to change position of the table.

Now is the time to **define the target points**. Each corner of the table, points P_1 to P_4, are defined as target points. Under {Home}-tab in the tool ribbon [Target] item (in group Build Station) select [Create Target] and a dialog window appears in the left side bar. Use work object `wobjTable` as reference. When the red "Position" field is selected and "Add New"-line in "Points"-part in the "Create target"-dialog window is active (blue), you can simply click on the four points P_1 to P_4 in the {RS1:View}-tab, the "Snap Object" should still be active, and the four points are added to the Points-list. When this is done press Create and Close, the four points should now have been added to `wobjTable` in the {Paths&Targets}-tab.

In the {Paths&Targets}-tab in the left sidebar you can expand the tree, locate `wobjTable` and find the target point under this branch. The orientation of the four points can be changed simultaneously: select the points, right-click and select "Modify Target" and "Rotate" on the popup-menu. In the Dialog-window that appears, let "Reference" be "Local" and rotate 180 degrees around x-axis, and perhaps also 90 degrees around z-axis. Then close the dialog window.

Now is the time to **define the path**, i.e. the track the robot tool point should follow. Under {Home}-tab in the tool ribbon [Path] item select [Empty Path] and `Path_10` path appears under the `Paths&Procedures` branch in the {Paths&Targets}-tab in the left sidebar. The four target points can now be dragged and dropped into the path, start and end with `Target_10`. The Path is also shown as a yellow curve in the {View1} window. We note that the curve is smooth in the corners and do not touch the points, except for the start and end point. We want a smooth path, but still want to move closer to the target points. Right click on all of the `MoveL` target points in the path, one by one or all at once, select [Modify Instruction] and [Zone] and select `z5`. You should see that the path, the yellow curve in the {View1} window, is modified.

1.6 Check path

Now we want to check the path. First if the robot can reach all target points; Right click on a target point in the `wobjTable` branch of the tree in the {Paths&Targets}-tab in the left sidebar. Select [View Robot at Target] and the robot and its tool should be moved to this point in the {View1} window. Do this for all points.

Even if the path seems to be good there may still be problems, as not only the target points must be reachable but also all points along the path must be reachable. To check this a program must be made and a simulation must be run as explained in next subsection.

1.7 Make RAPID program

A RAPID program can be generated from the station. Under {Home}-tab in the tool ribbon [Synchronize] item select [Synchronize to RAPID], select all modules in the dialog window that appears to `Module1`, make sure that all boxes for synchronizing are checked and then click OK button.

Select the {RAPID}-tab in the tab line and expand the {Controller}-tab in the left sidebar and double click on the program module `Module1`. This module now appears in the work area as a new tab. Observe how the `UISpenholder-`, the `wobjTable-` and `robtarget`-variables are defined. Also note that `PROC Path_10()` is defined but `PROC main()` is empty. Add a line in `main` function like below

```
1 !Add your code here
2 Path_10;
```

You may also update the comments with some relevant information. Look closer at all code in this file and try to understand the instructions used, the RAPID syntax may look a little bit awkward but this is how it is. In particular, look at the `MoveL` instruction in RAPID documentation.

You should now apply the changes (store): Under {RAPID}-tab click on tool ribbon [Apply] to apply (save) the changes done in RAPID code. You should now synchronize program back to the Station: Under {RAPID}-tab on the tab line, select [Synchronize to Station] under the [Synchronize] item in the tool ribbon, set all modules in the dialog window that appears to `Module1` and make sure that all boxes for synchronizing are checked and then click OK button.

You can now store the project; click on the disk-icon (store symbol) in the general menu on the title line. You may also make an additional copy of the RAPID-code, this is what you will need if you want to run the program on

an actual robot. Under the {RAPID}-tab on left sidebar the module `Module1` should be a branch of the Controller tree. Right click on this (and select Save module as) and store it somewhere under the name `RS1_8.mod`. Note that the tab in the RAPID-code-window now says `RS1_7.mod`, thus if you make more changes here they are only applied to the copy (if stored) and not to the `T_ROB1/Module1` connected to the project. I recommend that you close this tab (i.e the copy) and open the connected RAPID-code module by clicking on it again, in the expanded {Controller}-tab in the sidebar, where it should still be named as `Module1`.

If the program code consists of several modules, all modules, and a pgf-file, can be stored in a program directory. This directory is what to copy and use when a program should be moved to our actual robot, the station corresponds to the physical environment of the robot and can not be copied into the real physical world, a similar environment must be built if it is wanted. To save the complete program right click on `T_ROB1` in the expanded {Controller}-tab in the left sidebar and select "Save Program As".

1.8 Simulation

You may now go to simulation. Under the {Simulation}-tab on the tab line, select tool [Play] on the tool ribbon and hopefully the program starts and the simulation runs as expected. Alternatively, you can click on the Play symbol on the view menu (in work area). To slow down the simulation: Select [Simulation Control], a small symbol to the right of text [Simulation Control] below the Play symbol in tool bar. Set simulation speed to 20% in the dialog window. Run simulation again. You may also see the other options for [Simulation Control], perhaps even try some. You can now store the project/station, ex. as RS1_8.

1.9 Another path

Add a new path `Path_20` consisting of four new target points `Target_50`, `Target_60`, `Target_70` and `Target_80`, each point is the corner of the (metal) block bolted to the table. Follow the procedure used for `Path_10`. Add `Path_20` to `PROC main()` and run simulation again.

1.10 Use collision set

Simulation can test if collisions will happen, both by close inspection of the robot path during simulation and by the use of collision sets. In the {Simulation}-tab on the main menu, select [Create Collision Set] and a new collision set will

appear in the tree under {Layout}-tab in the sidebar. Drag and drop both the `UISpenholder` and `table_and_fixture_140` into ObjectsA and ObjectsB of this collision set respectively. Run simulation again and note that a collision happens, the pen touches the table.

Move `wobjTable` 1 mm up in the {Paths&Targets}-tab in the left sidebar of the {Home}-tab. The Synchronize to RAPID, note the change in the RAPID code, and run simulation again. This time hopefully without any collision. You can now store the project; click on the disk-icon (store symbol) in upper left part of the RobotStudio window. You may also store a copy of the project up to this point, {File}-tab and "Save Project As", change the description and save the project copy, ex. as RS1-11. Continue to work on the project RS1.

If everything is done and you have documented that each of you have used more than 15 hours on this assignment you may show the simulation to the teacher, submit your RAPID code in *canvas* and the assignment is done if your work is accepted. If you still have some time left try the next two sections too.

1.11 Another path

Add a path `Path_30` that moves the pen 10 mm above the center of each of the four bolts. The center of each bolt is 17 mm above the block and thus 32 mm above the table. On each bolt the pen should be moved down 12 mm (activating a collision) and then move the pen up again before continuing to a position 10 mm above the center of next bolt. You should define the points as the center of the top of each bolt, and use the RAPID `Offs()` function to define the path. Note that the the points should be defined relative to a work object aligned with the top of the table, like the other target points. When the collision test is run again the work object should again be 1 mm above the table, thus collision will be detected only when the tool press down on the bolts.

This path should also be added to the main menu. Also add to main a `moveL` instruction before `Path_10` and after `Path_30`, here use a target point 20 mm above the shaft centered on the table. The last instruction of the RAPID program may cause a Corner Path Warning, figure 10. This is not serious, and can easily be suppressed. Check simulation.

You should now store a copy of the project up to this point, as much changes will be done when the tool is changed in the next section. Continue to work on the project RS1.

Check simulation. You can now store the program and the station again. You should now show the simulation to the teacher, and submit your code in *canvas* and the assignment is done.

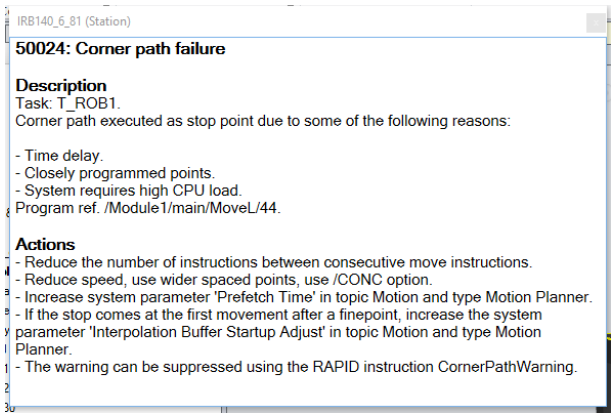

Figure 10: Corner Path Warning may occur in last instruction of a program, perhaps elsewhere as well. This is an example of a warning message that is not easy to remove, as there may be several reasons and several ways to fix the program.

1.12 Alternative tool

This part is optional. You should only do it if you have more time, the assignment should be 15-20 hours.

In section 1.3 you used the tool `UISpenholder`. Now you can replace this by the [Pen] from "Training Objects" in {Home}-tab [Import Library] and [Equipment]. Note that this tool is more difficult to use, and that it may happen that you need to reorient or reconfigure some of the target points. Try your best to solve this. Check simulation. You can now store the program and the station again.

You should now show the simulation to the teacher, submit your code in *canvas* and the assignment is done.

1.13 Noen begrep i RobotStudio

This section is only in Norwegian. All students have the complete RobotStudio documentation available from RobotStudio, Help-section under {File}-tab. The RobotStudio Help document has in appendix B a Terminology section. This has much more terms explained than the few terms included below. Even so, some explanation in Norwegian might be helpful for some students.

Det er viktig å få riktig forståelse av sentrale begreper brukt i RobotStudio. Begrepene er greit forklart på engelsk i dokumentasjonen, men som en liten hjelp til norskspråklige studenter følger en kort norsk forklaring.

Robot: En fysisk robot (manipulatorarm).

Controller: Styreskap med elektronikk og programvare som styrer roboten.

System: Et robotsystem er den fysiske roboten med styreskap og tilhørende programvare. I ELE610 bruker vi to robotsystem på E458, Rudolf og Norbert. Flere robotsystem som jobber sammen kalles gjerne ei *robotcelle*.

Station: I RobotStudio er en stasjon en *modell* av den fysiske roboten. Stasjonen inkluderer også (deler av) omgivelsene og verktøy og arbeidsstykker knyttet til roboten. Det kan være litt forvirrende at en stasjon også innholder {Paths&Targets} som beskriver baner og målpunkt, men de må være med for å vises i modellen. Disse kan synkroniseres med baner og målpunkt i RAPID program, de må være i programmet for å kunne utføres av den virtuelle kontrolleren.

Virtual Controller: Dette er et program, eller en programdel i RobotStudio som kalles RobotWare. Denne gjør det mulig å styre både den fysiske roboten og modellen av denne, altså simulere bevegelser i stasjonen.

Program: Et RAPID program består av en eller flere moduler som er lastet inn i en kontroller (fysisk eller virtuell) og som styrer kontrolleren og dermed roboten. Generelt skal et og samme program kunne brukes både på fysisk og virtuell kontroller og gi samme resultat, men det kan være noen ulikheter i grensesnitt, definisjon og virkemåte for IO-signal. For å flytte et program, men ikke stasjonen, fra en PC til en annen PC (eller til selve roboten på E458) er det hensiktsmessig å lagre hele programmet, alle moduler som inngå i programmet, på en egen katalog: Høyreklikk på programnavnet i treet i venstre del av skjermen og velg [Save Program As].

Module: Dette er en del av et program. For et program skal det være definert en, og kun en, funksjon `main()`. Generelt må ingen navn være definert flere ganger i de moduler som inngår i et program. Typisk for

vår bruk så er det hensiktsmessig å samle det meste av programmet i en modul, gjerne med `main_x_y.mod` der (x,y) er (øvingsnummer, delnummer). Merk at main-funksjonen uansett må ha navn `main()`.

Det er ofte hensiktsmessig å lagre en modul i ei fil (med samme navn som modulen), høyreklikk på modulnavnet i treet i venstre del av skjermen og velg [Save Module As]. En kan også trykke `Ctrl` + `S` i editoren (når en har endringer som ikke er lagret), men nå må en følge med, editoren skifter nå til å vise lagret fil og **ikke modul-fila** som tilhører stasjonen, lukker en fila som editoren viser og åpner fila tilhørende stasjonen ser en at denne **ikke** er oppdatert. For å unngå dette må en lagre modul i stasjonen først, [Apply] i [Controller] delen under {RAPID}-fane. For å lagre alle moduler i stasjonen [Apply All] så kan en bruke taskekombinasjonen `Ctrl` + `Shift` + `S`. En må altså passe på at fila som vises og redigeres i editor er modul-fila som tilhører stasjonen.

Project: Et prosjekt i RobotStudio består av de filer som lagres som en enhet, det vil si i en katalog med underkataloger under `R:\RobotStudio\Projects`.

Solution: Tidligere ble et prosjekt kalla ei løsning. Ei løsning (solution) i RobotStudio består av både en stasjon, en (eller flere) virtuell(e) kontroller(e), og program. En kan si at dette tilsvarer det fysiske system.

FlexPendant: Et betjeningspanel, skjerm med styreknapper, som henger i enden av en tjukk kabel tilknyttet kontrolleren for hver robot. I tidligere versjoner ble tilsvarende enhet kalla *TeachPendant*. Det er også en virtuell FlexPendant i RobotStudio.

RAPID Programmeringsspråket som brukes i RobotStudio. Hjelpetekst er tilgjengelig under {File}-fane, {Help} i venstre del og dokumentasjon *RAPID Instructions, Functions and Data type*. Bruk denne så ofte dere har bruk for den.

Synkronisering: I {RAPID}-fane er det et ikon for synkronisering mellom RAPID-kode og punkt, baner, verktøy og *work object* i modellen, altså *Station*. Dette kan gjøres begge veier, og kan være ganske forvirrende på den måte at ikke alle endringer en gjør i RAPID kan overføres til modellen (*Station*) og tilsvarende andre veien. Det kan også være feil som gjør at synkronisering ikke er mulig, disse er ikke alltid like enkle å finne. En skal også være klar over at programmerte baner ikke vil overføres fra RAPID til modell, baner i modellen må være med navngitte punkt. Det kan også være nødvendig å utvide treet som viser hvilke elementer som skal synkroniseres for å se hvilken modul hvert enkelt element synkroniseres til eller fra. Kanskje bør en kun velge noen enkelte greiner i treet som skal synkroniseres.

PCSDK Et (C#, C++, ...) interface (grensesnitt) for tilgang til kjørende RAPID-program fra andre utviklingsverktøy. For eksempel kan Matlab via PCSDK lese og endre verdier på variabler i et RAPID-program *mens programmet kjører*.

robtarget En dataklasse som brukes i RAPID (og PCSDK). Denne beskriver et mål-punkt sin posisjon og ønsket orientering for verktøyet i dette. Når roboten går til et punkt vil han forsøke å plassere verktøyet sitt aksekors (tooldata) samme sted og med samme orientering som punktet sitt aksekors som er punktkoordinater relativt til gjeldende *work object*.

tooldata En dataklasse som brukes i RAPID (og PCSDK). Denne beskriver et verktøy, med plassering i forhold til senter i robotens ytterste ledd og orientering i forhold til retning for robotens ytterste ledd.. `tool0` er et *tooldata* som alltid er definert in system module `BASE`, se figur 11.

work object En dataklasse som brukes i RAPID (og PCSDK). Dette angir koordinatsystem som `robtarget` forholder seg til. `wobj0` er et *work object* som alltid er definert in system module `BASE`, se figur 11.

pos Dataklasse for posisjon som brukes i RAPID (og PCSDK). Denne er en del av både *tooldata* (relativt til `tool0`), *work object* (relativt til `wobj0`, robotens base) og *robtarget* (relativt til brukt *work object*). Posisjon gis inn med 3 tall som er forflyttning lang x-akse, y-akse og z-akse i millimeter. Eksempel for `wobjTableN` i figur 11 er `[150,-500,8]`.

orient Dataklasse for orientering (retning) som brukes i RAPID (og PCSDK). Denne er en del av både *tooldata* (relativt til `tool0`), *work object* (relativt til `wobj0`, robotens base) og *robtarget* (relativt til brukt *work object*). Orientering gis inn med 4 tall som er representasjon på quaternion ↗-form, eksempel `[q1,q2,q3,q4]` der en har $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$. Ingen rotasjon er gitt som `[1,0,0,0]`, rotasjon 180 grader om x-akse er `[0,1,0,0]`, rotasjon 180 grader om y-akse er `[0,0,1,0]` og rotasjon 180 grader om z-akse er `[0,0,0,1]`. Rotasjon g grader om x-akse er `[c,s,0,0]`, rotasjon g grader om y-akse er `[c,0,s,0]` og rotasjon g grader om z-akse er `[c,0,0,s]`, der `c` og `s` er henholdsvis cosinus og sinus til **halve** vinkelen $\theta = g \cdot \pi/360$.

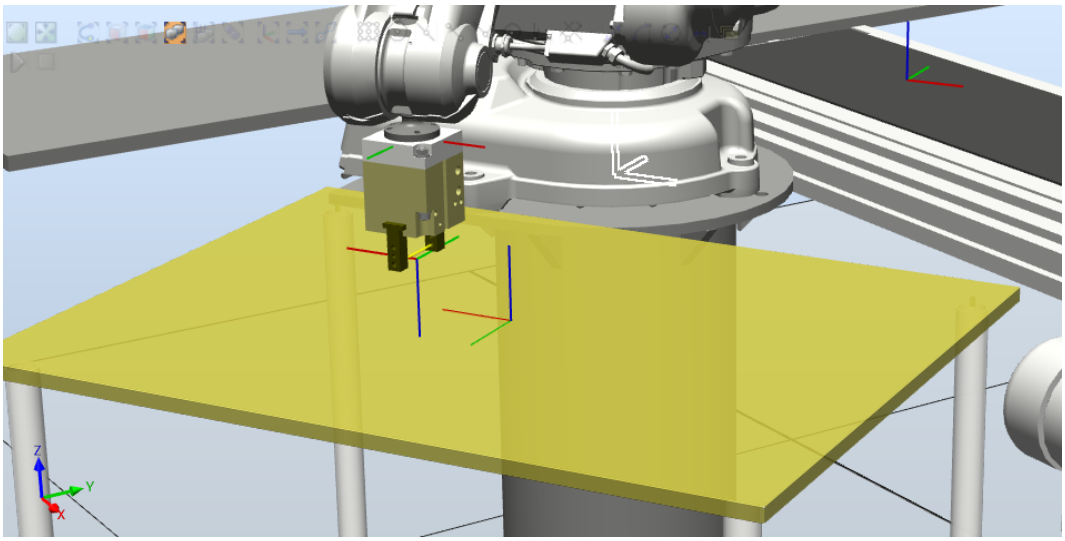

Figure 11: Figure shows Norbert and the table in standard position by its side. The x- and y-axis, red and green lines, for tool `tool0` is shown above the gripper, at the end of the robot hand. The work object `wobj0` is shown (weak) in the robot base. The work object `wobjTableN` is shown in the middle of the table, also axis for the tool `tGripper` is shown between the fingers of the tool. `tGripper` is as `wobj0` rotated 180 degrees around z-axis and translated along z-axis. The axis cross on the conveyor belt is where `wobjTableR` will be when (erroneously) used with Norbert. This is used when a similar table is placed on the other side of the robot, as it normally is for Rudolf.

Figur 11 viser roboten Norbert og bordet i standard posisjon, slik det vanligvis er plassert på høyre side av roboten. De små rød, grønne, og blå aksekorsene viser henholsvis x- og y- og z-akser for verktøy (tool) og arbeidsobjekt (work object) definert i aktiv kode. Her ser en `tool0` over griperen, i enden av robothanda og `tGripper` mellom fingrene på griperen. En ser at `tGripper` er rotert 180 grader om z-akse og flytta litt framover langs z-aksen i forhold til `tool0`. En ser `wobj0` svakt og uten farger inni robotens senter bunn (base), `wobjTableN` viser midt på bordet, legg merke til at z-akse peker oppover, mens z-akse for verktøy peker nedover, eller framover i forhold til robotens hånd.