University of Stavanger

Faculty of Science and Technology

Stavanger, July 1, 2021

ELE610 Robot Technology, autumn 2021

Preliminary issue, the actual exercise will be available in August after proof reading.

ABB robot assignment 1

In this first task you will be familiar with the ABB RobotStudio software package and the fairly comprehensive documentation that comes with. The first 11 sections of this assignment should be done, sections 1.12 and 1.13 are 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*, a report including RAPID code, note that this should be renamed into a txt-file (not the mod-file that Windows will interpret as a video file). The report may alternatively be a pdf-file that also includes comments or questions (in addition to RAPID code shown in Verbatim or Courier font).

1 Introduction to RobotStudio

1.1 Install RobotStudio

Most of you will prefer to install ABB RobotStudio on your own laptop computer. To install RobotStudio on your own laptop, it must have Windows 7, Windows 8.1, or Windows 10. I have only tested the last versions of RobotStudio on Windows 10. You can also use RobotStudio on the PCs on your desk (work-place) in room E462 or E464. I hope we have everything correctly installed when the course starts.

The two PCs closest to the robots in E458 also have RobotStudio (but perhaps an earlier version) installed and can be used when running the programs on the physical robots.

Karl Skretting, Department of Electrical Engineering and Computer Science (IDE), University of Stavanger (UiS), N-4036 Stavanger. E-mail: karl.skretting@uis.no.

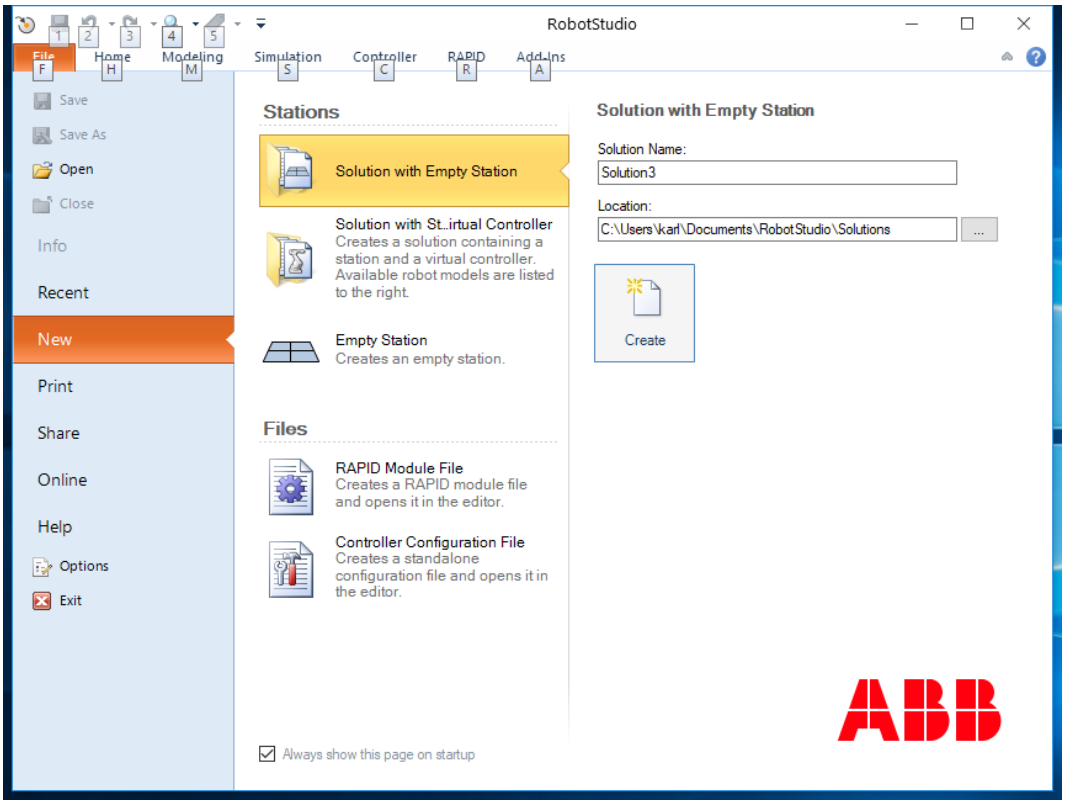

Figure 1: The RobotStudio start window.

Install RobotStudio on a PC
It is an advantage that each group member has a laptop where RobotStudion is installed, thus allowing all group members to work on RobotStudio concurrently and also to work on RobotStudio elsewhere (at home). You should install the correct (last) version of RobotStudio and RobotWare virtual controller on your (laptop) computer. The huge (approximate 2 GB) zip-file can be downloaded from ABB web page.

For installation on a standalone PC, follow the installation wizard, `Setup.exe` in unzipped file. Select full installation and use standard options. Do not activate the license the first time the wizard asks, skip this. Activate the UiS network license for RobotStudio after installation. When RobotStudio is started, and the PC is connected to the UiS network locally, preferably wireless, from the startup screen, figure 1, select {File}-tab (top left) and [Help] (in the left sidebar) and [Manage Licenses] (at the bottom of the Support section). Enter our license server, `rs.lm.ux.uis.no`, then RobotStudio is restarted (and activated for 90 days).

Run RobotStudio on E462 or E464 PCs or a laptop
After installation it should be simply to press the (correct) icon on the desktop and RobotStudio should start. The start window is shown in figure 1. The

top line shows the 'Quick Access Toolbar' with some icons. The second line is the main menu line, here are tabs that set which main state the program has, initially the {File}-tab is active and the {File}-window is shown below the main menu line. This window has a menu on the left that select what is shown in the rest of the window. Especially the [Help] option is useful as it shows the RobotStudio version, supported RobotWare versions, and license information, and more important gives access to the comprehensive documentation. When you work with RobotStudio the active tab is often the {Home}-tab or the {RAPID}-tab.

Controller (RobotWare) installation and issues
A *controller* is needed to program or simulate a program. RobotWare is the virtual controller that come with RobotStudio and is basically the same version as RobotStudio. RobotWare is installed from within RobotStudio. To do this an Add-In is needed: From the startup screen, figure 1, select {Add-Ins}-tab and 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. This should only be needed to be done once. You may want to make a new virtual controller now, or you can postpone this a little bit. From the {Controller}-tab in the main menu line you select Installation Manager 7, or Installation Manager 6, and follow the instructions to install a new virtual controller which can later be used. *This can be confusing.*

From the {Home}-tab in the main menu line you should connect a virtual controller to the station using the [Virtual Controller]-menu (third from left on top of window). Here you can also create a new virtual controller (if not done from the {Controller}-tab). When you (successfully) add a controller to the station, the controller status field (which appear at the very lower right of the window) should turn from red to yellow to green. A dialog window may appear asking you to select from two robot variants, just select the top one and continue. See documentation for more on controller installation and use.

Ideally, the version of RobotStudio and RobotWare should match, but that's not a problem to have more versions of RobotWare installed simultaneously on the same PC. Real controllers that run on the robots, Rudolf and Norbert, are now RobotWare version 6.01. It should be no problem that this is different from the version you have installed. What is a problem is to import old Pack-And-Go files, they require the same version of RobotWare that was used when they were created. Therefore, when needed you should also install earlier versions of RobotWare on your laptop. This is an issue that have cause some troubles earlier, as I don't know exactly what is needed to avoid problems. Often, there has been no problems at all, but sometimes we have had some problems, and I don't rule out that there may still be problems.

Storing of files
Another thing I think can be rather confusing is how RobotStudio stores, and recover, files (of different versions) related to solutions (projects) and stations (robots) and parts included therein. It is a huge and complex (?) structure appropriate for large factories. If you want to get some experience on how RobotStudio keeps track of your changes, and on how UiS network system actually works, you should read a little bit documentation and the just try it. Don't be afraid, the worst thing that can happed is to get lost and then you have to start over again. Thus, it is always wise to make copies, at least of the RAPID code files `*.mod`, on a disk you trust, on your own laptop or a memory stick. To help you, or confuse you, some previous problems are mentioned below, and a little about how they can be resolved or circumvented.

There is, or may be, a problem using network-paths in filenames. Each user, and each installation, may want to store (user) files in a particular catalog and sub-catalogs below. One way to solve (avoid) this is to use the `SUBST` command in a "ledetekst" (command prompt) window. You may want to store the files on your UiS network catalog (F-disk) `F:\Documents\RobotStudio` which may be the same as `C:\Users\username\Documents\RobotStudio` or `C:\Users\username\MyDocuments\RobotStudio`. Or you may want to store the files on your local computer (C-disk) `C:\ELE610\RobotStudio` or on your DropBox (or any other cloud?) catalog somewhere `...\Dropbox\ELE610\RobotStudio`. Anyway, using `SUBST` command in a command prompt window like

```
C:\> subst R: C:\Users\username\Documents\RobotStudio
```

should give you easy access to your RobotStudio files. It also gives me a way to refer to (local) RobotStudio files; the catalog where you save your RobotStudio data is simply referred to as `R:\`.

Run a program You first run a simulation of the program, and in some cases like this first assignment this is all you do. To run a program on the actual robot the computer must be attached to the same local network as the robot is. If you use a laptop it is simple to carry the laptop to E458, connect to the local network there and then from RobotStudio connect to the actual controller (robot) and then copy the program into its memory. If simulation is on a E462 or E464 PC the program (not pack-and-go file) should be stored on a memory stick which is easy to carry to E458 where you then use one of the PCs close to the windows and robots.

1.2 Learn RobotStudio

Both a 32 bit and a 64 bit version of the software are available, I am told that the best is to use the 32 bit version but have actually not noticed any difference. Our ABB specialist (Morten Mossige) says that the 32 bit version may be slightly better tested and includes a few additional features, which

we probably do not use. There is also no speed gain for the 64-bit version. One may be notified that a newer version is available, even if one has installed the latest version. But there is no need to answer 'Download' here since the installed version is good enough. Simply, reply 'Do not show again' and hope that this message will not be repeated every startup time.

Examine [Help], and browse the comprehensive RobotStudio documentation. RAPID documentation may be "hidden" inside IRC5 documentation box. The most relevant documents are
[RobotStudio Help], [Introduction to RAPID] and [RAPID Reference].

Make (create) a new solution, i.e. project. A solution can contain several stations and program modules. From {File}-tab, [New] option, select [Solution with Empty Station] and give the solution an appropriate name and store it somewhere (location)[1]. Ex.: Solution Name: 'Lab1', Location: `R:\`, and the project files should be stored in `R:\Solutions\Lab1`. Press [Create] button and the main RobotStudio window will appear. It has several parts, look at all parts and documentation and try to get an overview of what each part is for. Press [ABB LIbrary] in {Home}-tab and select robot [IRB 140] in dialog window, we have [IRB 140 6kg 0.81m]. The robot should appear in {View1} and in the lower right part of the window 'Controller status' should become green. Rotate and move the view of the robot, use `Ctrl` and `Shift` and mouse. You can now store the station; {File}-tab, you may fill out the information properties as appropriate, and the select [Save station As], and give name 'Lab1_2'. End RobotStudio; {File}-tab and [Exit]. Examine the directories and files created, perhaps you will get something like what I got (some years ago): Approx. 50 MB in 60 files in 28 directories. The solution stored in a directory; `R:\Lab1`, the file `R:\Lab1\Lab1.rssln` indicate that this is a solution directory. The station file is `R:\Lab1\Stations\Lab1\_2.rsstn`.

1.3 Attach tool to the robot

Start RobotStudio again and open station stored above; {File}-tab and [Recent]. A tool may be attached to the robot. The tool may be from a standard library of tools, 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]. The tool should appear in the {Layout}-sidebar. You can now drag the tool onto the robot, just above in the {Layout}-

[1]The location on disk is denoted as folder, directory or catalog, and given by a path

sidebar, update the position and the pen should be attached to the hand of the robot in {View1} window. You may right click on the tool in the {Layout}-sidebar and a menu should appear.

1.4 Include table in station

Now, include the propeller table to the station; {Home}-tab, [Import Library] and [Equipment]. The propeller table should be at the bottom of the list. You can now store the station; {File}-tab and [Save station] or if a new name is wanted [Save station as], name may be 'Lab1_4'.

1.5 Define work object

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. Thus the work object should be the part that the robot process or move. In {Home}-tab a part of the menu 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, in {File}-tab [Help], for the difference between 'User Frame' and 'Object Frame'.

Define the new work object using {Home}-tab, [Other] and [Create Workobject]. Left side bar will show the {Create Workobject} dialog window, see figure 2. Here you see the {Create Workobject}-tab at the middle left part, the name is already given as `wobjTable` and the `Frame by points` text has been clicked, and thus the small [v] button has appeared just right of this text. Now, click the small [v] button and the three point input dialog appears, see figure 3. We want the x-axis along one long edge of the table and the y-axis along one short edge of the table and to do that the first given point should give the origin of the work object, the second point the direction along the x-axis and the third point the y-axis. To fill in values, click on first point in dialog window figure 2 (upper red box), the switch to the {View1}-window and zoom in on the table, select [Snap Object] from the see through symbols at the top of the {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). Finish by pressing [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. If the axis system is not as you want it, delete it and try again. In the menu of the {Home}-tab the value of [Workobject] should be updated as well.

The work object should be attached to table, so when table is moved the work

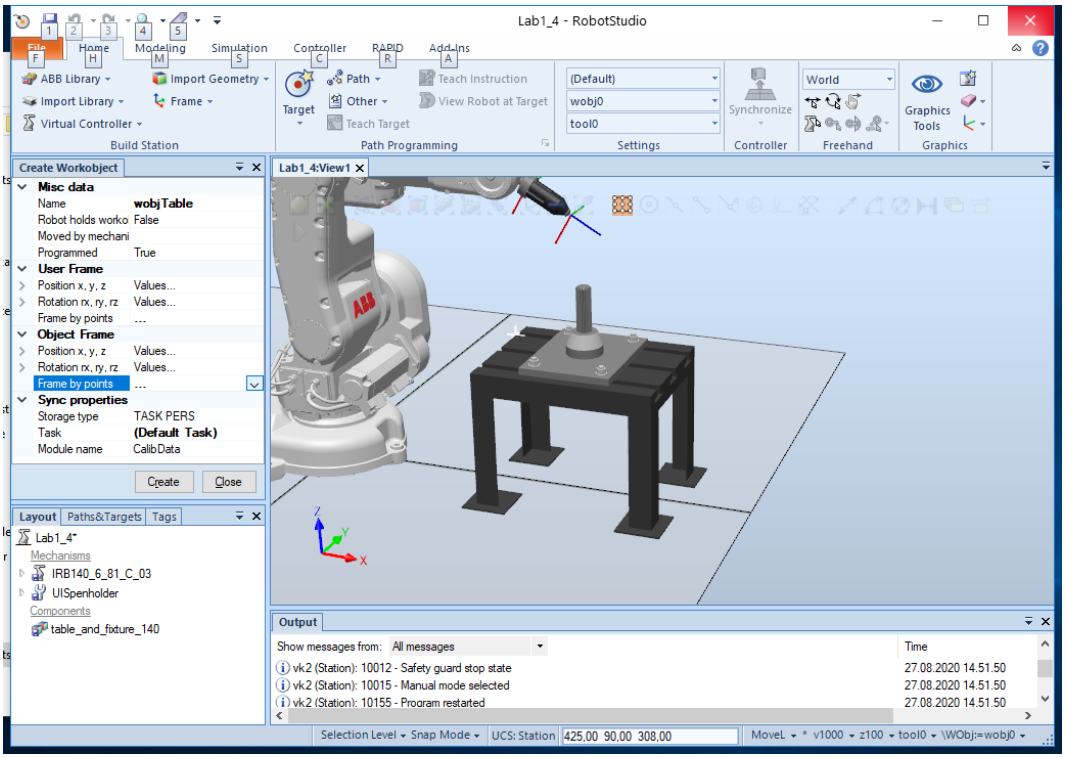

Figure 2: The RobotStudio window when creating a work object.

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. Place the table with the long edge towards the robot, 100 mm closer to the robot and still centered in front of the robot. You can now store the station; {File}-tab and [Save station] or if a new name is wanted [Save station as].

1.6 Define path

The robot has still not moved, but this will happen soon. In the menu of the {Home}-tab there is a section named 'Freehand', here a small robot symbol is for 'Jog Joint'. Click on this and then on the robot and move a selected axis by dragging it. Also try the 'Jog Linear' and 'Jog Reorient' options.

Now is the time to define the target points. The first point we want to define is the upper left point (related to the robot) on the top of the table; In the {Home}-tab [Target] select [Create Target] and a dialog window appears in the left side bar. Use work object as reference, click in red box (Position) and then on the corner of the table to fill inn values. When the tool move

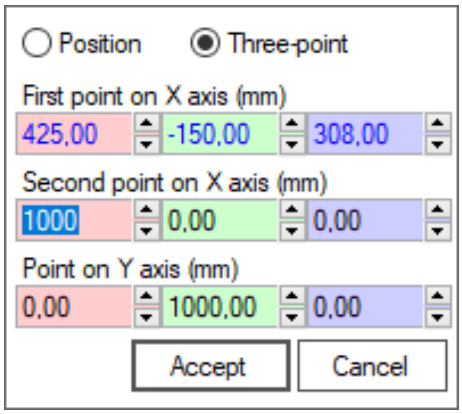

Figure 3: The RobotStudio three point input dialog window.

to this point the robot tries to align the axis system of the tool to the axis system of the point, for this to be possible the target point z-axis should point downwards. Thus orientation in red box should be 180 degrees, and perhaps also orientation i z-axis should be -90 degrees for even better alignment. Name the point 'Target_10', perhaps you need to click the More button first, and then click Create button. In the {Paths&Targets}-tab in the left sidebar (still under {Home}-tab) you can expand the tree, locate 'wobjTable' and find the target point under this branch. Now go clockwise and add the other three corners of the table as 'Target_20', 'Target_30' and 'Target_40' (origin for 'wobjTable').

Now is the time to define the path. In the {Home}-tab [Path] 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.

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

1.7 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 of the {Home}-tab. 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.

1.8 Make RAPID program

A RAPID program can be generated from the station. Select the {RAPID}-tab on the main menu, select [Synchronize] and [Synchronize to RAPID], 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. Expand the tree in the {Controller}-tab in the left sidebar of the {RAPID}-tab and double click on the program module 'Module1', this module now appears in the main section of the window. Add a line in `main` function like below

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

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 may now synchronize program to Station. You can also same the program, and all modules used in the program; in the {Controller}-tab in the left sidebar of the {RAPID}-tab right click on the program 'T_ROB1' and select [Save Program As] and give name of the folder (directory) where you want all the modules to be saved. This folder 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.

1.9 Simulation

You may now go to simulation. Select the {Simulation}-tab on the main menu, select [Play] and hopefully the program starts and the simulation runs as expected. To slow down the simulation: Select [Simulation Control], a small symbol to the right of text [Simulation Control] below the Play symbol in menu, then 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 station; {File}-tab and [Save station] or if a new name

is wanted [Save station as].

1.10 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.

1.11 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' 2 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 may now store the program and the station again.

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 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.

1.12 Another path

Add a path 'Path_30' that moves the pen 10 mm above the center of each of the four bolts, on each bolt the pen should be moved down 12 mm (activating a collision) and then move the pen up again before continuing to the next bolt. This path should also be added to the main menu. Also add a 'moveL' instruction before 'Path_10' and after 'Path_30', here use a target point 20 mm above the shaft centered on the table. Check simulation. You can now store the program and the station again.

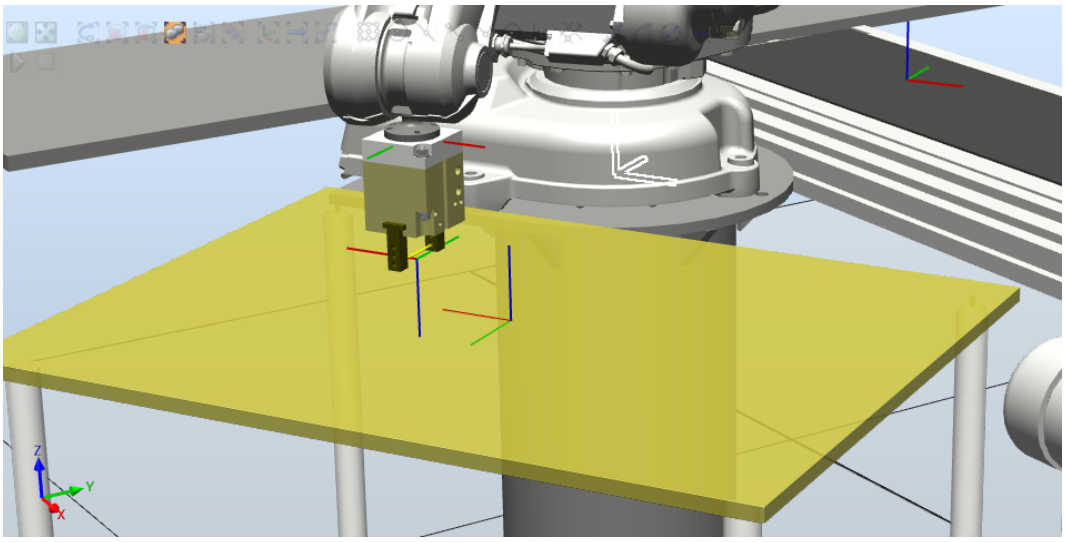

Figure 4: Figure shows Norbert and the table in standard position by its side. The x- and y-axis, red and green lines, for tool 'tool0' is shown above the gripper, at the end of the robot hand. The work object 'wobj0' is shown (weak) in the robot base. The work object 'wobjTableN' is shown in the middle of the table, also axis for the tool 'tGripper' is shown. The axis cross on the conveyor belt is where 'wobjTableR' will be when (erroneously) used with Norbert.

1.13 Alternative tool

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 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.

1.14 Noen begrep i RobotStudio

This section is only in Norwegian. All students have the complete RodotStudio documentation available from RobotStudio, Help-section under {File}-tab, but 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. Vi har to robotsystem på E458, Rudolf og Norbert. Flere robotsystem som jobber sammen kalles gjerne ei *robotcelle*.

Station: I RobotStudio er en station 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, som gjør det mulig å styre, simulere bevegelser, i en modell.

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' der (x,y) er (øvingsnummer, delnummer). 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 den 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`.

Solution: En 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 4.

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 4.

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 4 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 $\theta = g \cdot \pi/360$.