# ROBOTSTUDIO

## Creating a Robot Station

Initially we start with an empty robot station in RobotStudio, where we are given the option to add an existing ABB robot system. For our particular cell, we are using the IRB2400.



Figure 1: Selecting the Robot Controller for our Station.

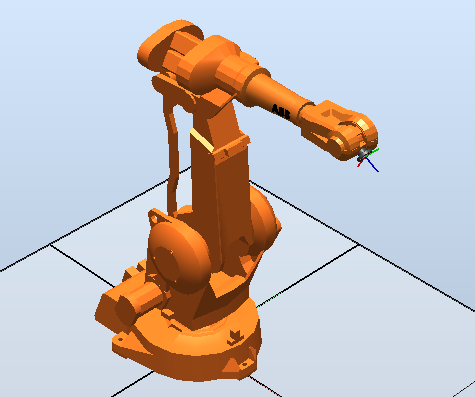


Figure 2: Station with the imported IRB2400

Now that we have imported the robot, we can now work on importing the other components of the cell, such as the table and dispensers. In order to do this we have been given the technical drawings of all the needed components, including their dimensions which enable us to model each part in modelling software such as SolidWorks or CATIA. Once each model has been constructed and checked for compliance with the given dimensions, it is exported as a geometry file such as a .wrl or .stl file, which can be parsed by RobotStudio and used as a part in our station. Because we want our components to eventually have their behaviours controlled by signals and properties, we add the models as components of their own Smart Components.

Once the component has been imported into the station, its position is set in compliance to the given cell dimensions, as to create a complete imitation of the real robot cell that is present in L220.

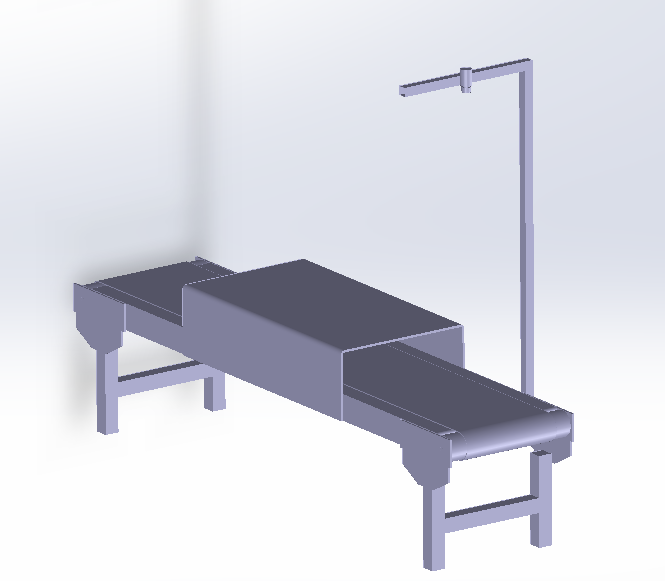
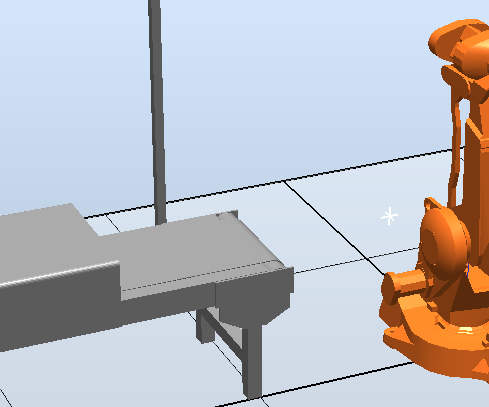
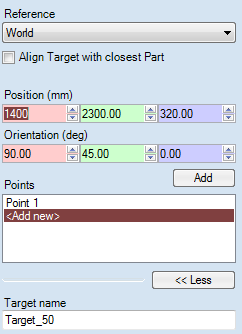
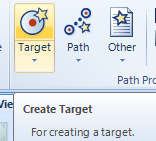
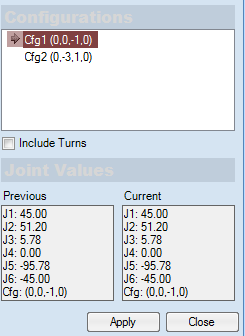
 

Figure 3: A finished model of the conveyer constructed in SolidWorks (left), which was then imported into the existing Robot Station (right)

This process is repeated until all components are present in the Robot Station. Once this has been achieved, and the configuration of the robot cell complies with the provided cell dimensions, it is time to incorporate events.

## Robot Tasks

“Teaching” the robot how to perform desired tasks is greatly simplified in RobotStudio and can be achieved in a number of ways. The first step is to set a series of robot targets (robtargets) and paths which are needed for the robot to carry out specific tasks in specific ways. Robtargets are defined position data for a robot and external axis. They define the position of the tool centre expressed in mm, the orientation of the tool in the form of a quaternion, the axis configuration of the robot (as there can be more than one possibility) and the position of the external axis. We can teach our robot these points in a number of ways, including entering the coordinates ourselves, or jogging the robot manually to the needed targets and saving them.   
To add a target of known coordinates, we click on the ‘Target’ icon, where we can then define a series of coordinates for targets that we desire.

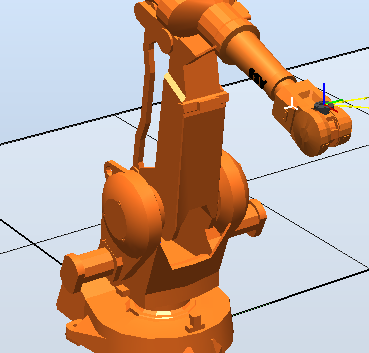
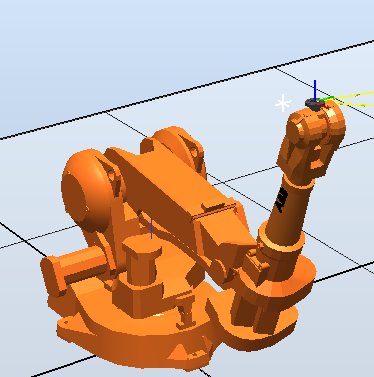
 OR 

Figure 4: Process for constructing new robtargets.

Once these robtargets are formed, in order for them to be of any use to us we need to be able to incorporate them in to our controller’s program modules. We can do this simply by dragging the formed robtargets into a path (created by clicking the ‘Path’ icon), and then synchronising our current station to a virtual controller (‘Synchronise’ icon). Robotstudio automatically create a small program with default velocities that can be run to visualise the movement of the robot’s new path. However, in our project, creating paths is not needed for us, as the movement and tasks are all given as signals from our RAPID program.

## Robot Station Simulation Logic

Now that we have a functioning robot station, and we have a developed RAPID program which gives instructions to our robot station, we can work on implementing the simulation aspects. That is, our simulation does not automatically account for gravity, nor does it even account for the interaction between two surfaces. Without the implementation of station logic, our tines would pass right through the pallet, and our pins would fall straight through the dispenser and the table, and the ground.  
Robotstudio allows us to utilise a number of components, which help us add realism and functionality to our simulation. These include various sensors, position manipulators, and attachers. For example, in order for our end effector to carry the pallet via its tines from the table to the conveyer belt, we need to be able to know when the tines are in contact with the pallet, which triggers the pallet to move with the tines. Once the tines tilt back, the pallet tilts as well, and additionally, moves back into the base of the tines to simulate the real case of the pallet sliding back due to gravity. Once the pallet is placed down on the conveyer, we need to know, and then trigger an event that detaches the pallet from the tines and leaves the pallet in place. This is a simple case of station logic.

Similarly, the event of grabbing a pin from the dispensers may not seem like a difficult task to simulate, but in order to create a realistic imitation, a lot of careful work is required. Firstly, pins are required to drop out of the dispenser if there is room to do so. Components called ‘Plane Sensors’ are located on the surface of the ledges where the pins would rest. The sensed part is identified to us, and prohibits the creation of another pin. Additionally, if the plane sensor was to deactivate, that is, if there was no pin on the ledge, another pin would be created, using a ‘source component’ that creates a new pre-set geometry (a new pin), which is then moved downwards through space until it comes in contact with the plane sensor, where it is told to stop.

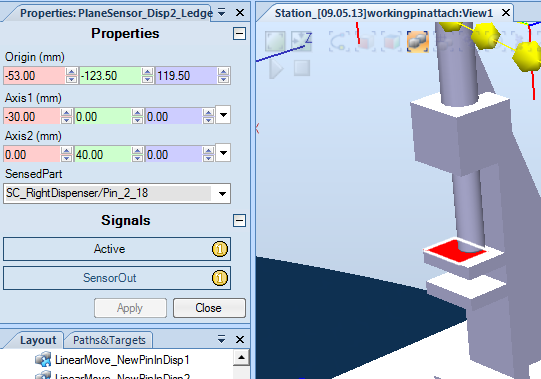


Figure 5: The Plane Sensor for Dispenser 1, displaying its active state, and its Sensed Part.

From this point, the gripper is moved to the location of the pin that it has been instructed to grab, as told by the RAPID program. A ‘Positioner’ component moves the position of the gripper to an open state, before moving in towards and around the target pin. A ‘Linear Sensor’ component, much like a plane sensor’, which is located in the centre of the gripper, senses the present pin component, and relays the identified pin to an ‘Attacher’ component, which locks the position of the pin in its current position relative to the gripper, essentially simulating a physical ‘grab’.  
Now that the pin has been retrieved, the robot positions the tool head to a location slightly above the pallet, as instructed by the RAPID program. The pin is then lowered into the chosen hole. At this point, all simultaneously, a linear sensor located near the side of the gripper detects that the pin has indeed been placed in a hole, which triggers the opening of the gripper via another positioner component, which further triggers the deattachment of the pin from the gripper. At the same time, a linear sensor/attacher couple located in the pallet hole becomes activated by the pin’s presence, and then pin is finally locked into the hole of the pallet.

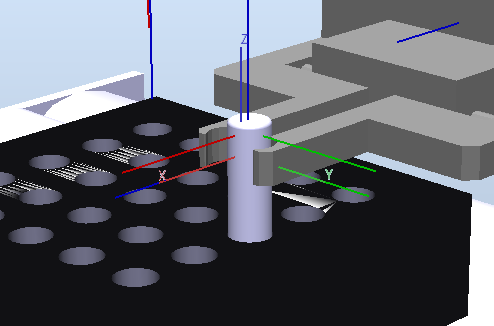


Figure 6: The Plane Sensor for Dispenser 1, displaying its active state, and its Sensed Part.

This example covers the majority of what is involved in the rest of the station logic. The remaining tasks will be outlined through flow-charts.

Getting Pin Out of Pallet

START

END

Pin?

End effector moved to safe position

Assign sensed pin to attach to gripper

Detach pin from pallet

Open Gripper

No pin attached.

NO

YES

Running Conveyer with Pallet

Linear Move pallet out

Linear Move pallet in

Move pallet in or out?

START

IN OUT

END

Placing Pin in to a Dispenser

END

LinearMove pin down into dispenser

Detach Pin

Open Gripper

Move pin above dispenser 1

Open Gripper

Detach Pin

LinearMove pin down into dispenser

Move pin above dispenser 2

2

Dispenser 1 or 2?

Move to dispensers

Pin in Gripper?

“No pin in gripper”

START

NO

YES

1 2