

AGV Load and Unload (winders)

Target of this exercise is to learn to write your own xScript file to be used as library for your own functions and structures. In this example you will write a file to handle a winding machine system.

1. Winders

In this example we work with an agv serving some winders. Every winder is producing a coil of plastic wire (**figure 1**, left), then when the bobbin is ready, the agv can go and take it out from the winder (load mission) and bring it to a collecting station (unload mission).

The winding machine never stops: when the bobbin is ready, the winder machine starts building a second coil on the other bar (**figure 1**, middle). It is necessary that agv comes to take out a finished bobbin before a due time, after which the second bobbin will grow up too much and crush the other one, making damages to the machine (**figure 1**, right).

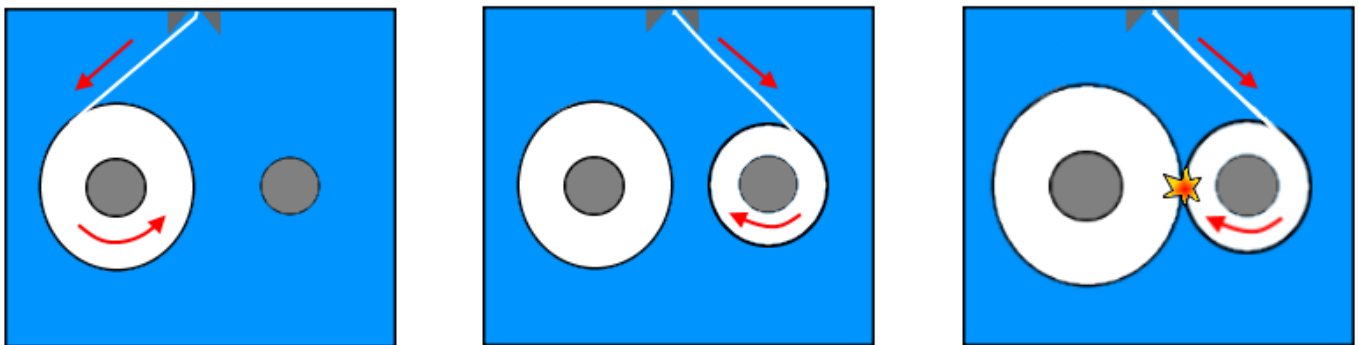
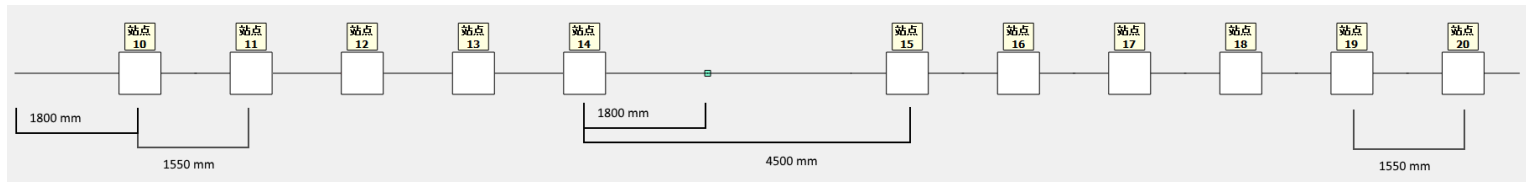


Figure 1: winder making a coil of wire (left); winder finishes one bobbin and starts the second coil (middle); after too much time the bobbins crush (right).

2. Draw a map with RAT (Robox Agv Tool)

Make a folder to store this project and save your xml map file in a subfolder called “map”. Open RAT and draw a map respecting following specifications (same as previous exercise):

- 1 horizontal line 22 m long.
- 11 stations (user points) numbered from 10 to 20, divided in two groups: 10-14 (load winders) and 15-20 (unload stations).
- Station 10 is set at quote 1800 mm of the line.
- In every group, station distance is 1550 mm constant.
- Distance between station 14 and 15 is 4500 mm.
- A generic point called “Standby point” is set at 1800 mm after station 14 (right side). This point code number is 200.



Create 1 agv in the RAT project to be played in this sample. Agv maximum speed is 1 m/s.

3. AGV Manager project

In the project folder you should make your AGVM project. If you want, you can start from BasePrj_AGVM, that you can find online in Robox OneDrive AGV folder or from your previous exercise code. Set AGV as emulated, since in this sample there is no Robox controller connected to a real agv.

You must write a xScript that makes the agv to do the following things:

- If the agv is empty (check **uStatus** of **XVehicleInfo** struct), check the winders: if there is at least a winder which has finished its bobbin, go and perform a loading mission. If more winders are ready, choose the one with less remaining time.
- If agv is loaded, choose a random unloading station and do the unload mission.

Even if there is no controller, please make all necessary MACROS for all the missions, in order to make a correct structure for the real agv. Unused macros can have a comment, a timer or just return true. Load and unload time are 5 seconds each.

In this example we just introduce this kind of system, so we don't care about the bad case of bobbins collision. Anyway, keep it in mind because it's very important on real systems (see AGVM *tutorial 3* to know more).

4. Winders' script

In order to handle the winders, you must write a sXcript_winder.xs. To handle each winder, use a struct (**Object**) in which you keep memory of station number, time to finish the product and a product ready flag. An additional flag may be used to assert if agv is coming; it will be useful for further samples with more agv.

Some interesting functions that you may write are:

- Init: used at AGVM start to set initial conditions and data.
- Update: synchronous update of timers.
- searchUrgent: used to know the most urgent winder request.
- timeToProduct: to know how much time is remaining until bobbin is complete.

Remember to store these functions in a class (again an **Object** of xScript) and use them in your main script. Other functions of your own are welcome.

5. Debug

Start AGVM and make the system work. Try to find suitable winder timers values in order not to saturate the system. You can act on bobbins production time and shift time among different winders. For all the stations it is necessary to write remaining time on agv manager screen: you can do it in a text box (**NewText** and **SetText** commands) or directly in the stations' labels (**SetSiteText** command).

Remember that on real systems these time data are provided by the customer and our job is to develop a good agv system to serve and fulfill the factory.