

AGV Load and Unload (winders 2)

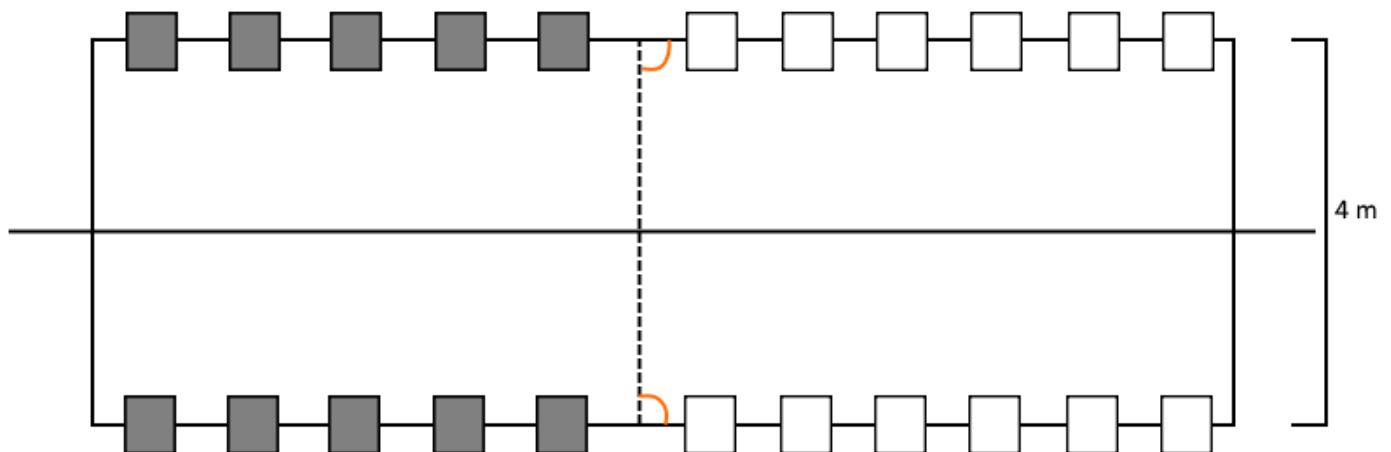
Target of this exercise is to practice with a plant design and focusing on how to find a clever solution in terms of xScript, map configuration and agv number.

1. Winders plant

In this example we start working with the plant used in the previous exercise but adding more complexity: let's copy and paste, to make two parallel lines, with a distance of 4 meters and connected by side vertical lines, as shown in **picture 1**.

To improve agv motion possibilities, we can add a central horizontal line (empty, with no station). If you want, you can also add one vertical line (dotted line in **picture 1**) to allow agv to change line between load and unload station groups. The orange part is where there is no space enough for a curved path, therefore agv must perform a rotation. All the other crosses can be curved.

A connection between central vertical and horizontal line is allowed too.



Picture 1 – plant scheme

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 the scheme shown in **picture 1**.

System handling is up to you: you can set line directions, add some user point at your convenience and define your own station numbers.

Create some agvs in the RAT project to be played. Since the plant is double of previous exercise map, let's start with two agvs, then maybe you will decide to add more ones.

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 perform the load/unload missions in this winder plant. Provided data are:

TIME_SHIFT_S 35; [s] time to shift station timers during init.
; 1st winder will finish a bobbin in 35 s, 2nd winder in 70 s, 3rd in 105 s....
; this values for each winder line; A user delta can be added to time shift
TIME_READY_PRODUCT_S 200; [s] winding time to create a full bobbin

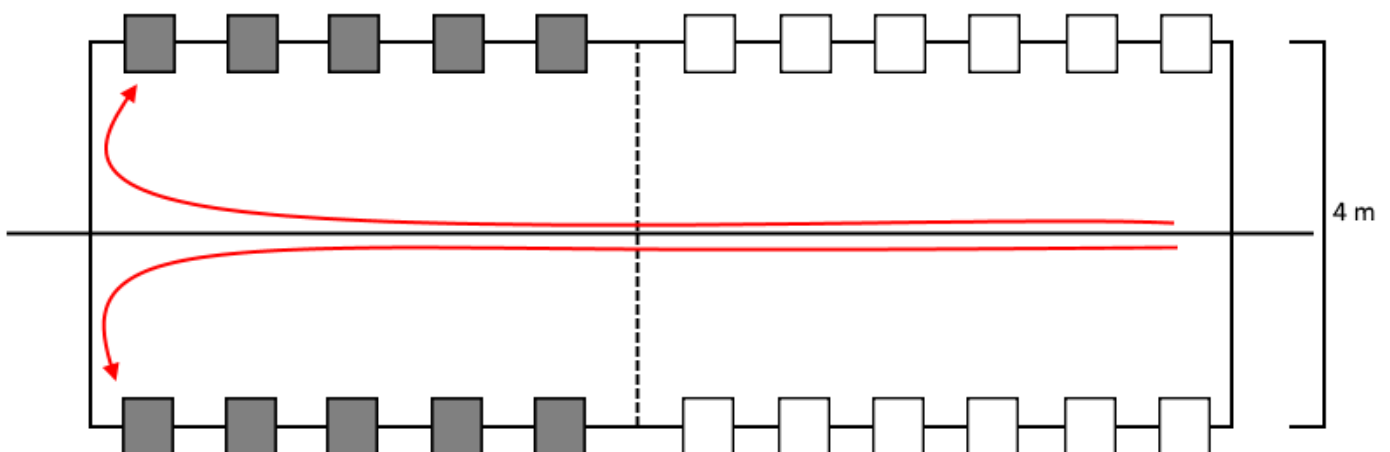
Using a winder script to handle winder machines may be a good idea. If the winder machine completes the second bobbin and agv hasn't taken the first one yet, it's an error. So, you must show this in some way, like putting an alarm picture on the station (you can use **SetSiteImage** command).

Since loading station requests are defined by winder timers, you can just choose the unload schedule: write your own code to search the best unload station for each agv.

How to avoid more agvs to choose the same unload station? Try to use point properties, like **PROP_RESERVED_TO_AGV** (see **IntProperty** command).

4. Debug

Start AGVM and make the system work. Try to find suitable rules for the map (like compulsory directions and maybe use some generic points). How about using the central line as shown in **picture 2**, just to go leftwards?



Picture 2 – a map example where central line is used to send agvs leftwards only.

If necessary, add more agvs in the plant in order to perfectly fulfill the factory production needs. Remember that agv maximum speed is 1 m/s on all the map, but on the central horizontal line it is possible to reach 1,2 m/s. Auto acceleration in $0,3 \text{ m/s}^2$ and rotation speed is 25 deg/s. Load and unload time are 5 seconds each.

To set the parameters you can do as follows:

```
if (EMUAGV_IsAgvEmulated(i))  
    EMUAGV_SetRegisterValue(i, Reg_NVRR, NVRR_MAX_VEL_ROT_AUTO, 25.0) ; deg/s  
    EMUAGV_SetRegisterValue(i, Reg_NVRR, NVRR_MAX_VEL_TRA_AUTO, 1.2) ; m/s  
    EMUAGV_SetRegisterValue(i, Reg_NVRR, NVRR_ACC_TR_AUTO, 0.3) ; m/s^2  
endif
```