

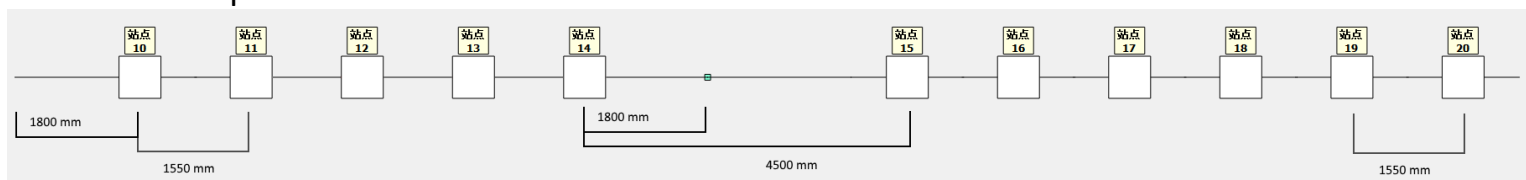
AGV Load and Unload (random movement)

Target of this exercise is to handle an AGV executing a load/unload task, moving between two station groups, where left group is for load and right group is for unload. You can see the attached video (see chapter 4 Video).

1. 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) and 15-20 (unload).
- 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. Remember that agv maximum speed is 1 m/s on all the map.

2. 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. 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), choose one loading station at random and register a loading mission.
- If agv is loaded, choose a random unloading station, register, and do the unload mission.

- After a complete unload, set a wait timer of 5 seconds, during which agv is not taking any other mission. Print current timer value with **OnNextMissionDebugMessage** inside **OnNextMission** predefined function. You can use this code:

```
string str_time  
str_time = StrFormat("%.1f s", SecsToTimeout(timerGoWaitStation))
```

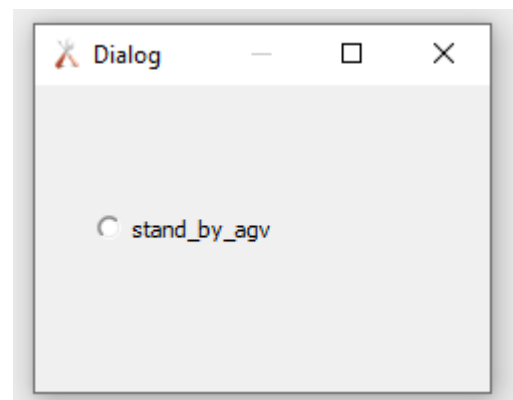
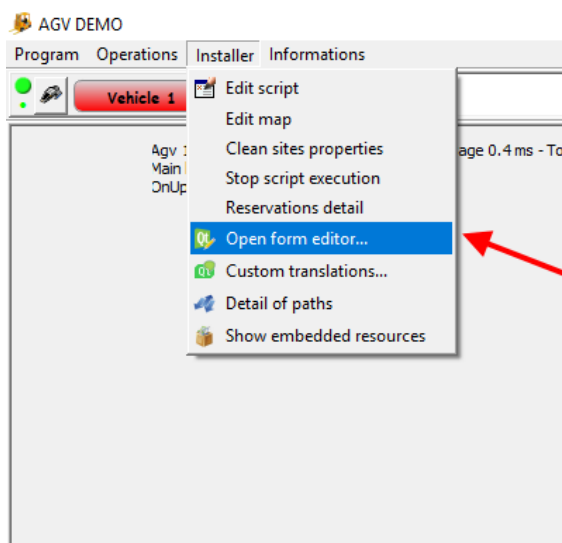

To set a timer, use *TimeoutS()* or *TimeoutMs()* commands; use *IsTimeout()* to check it.
- According to a user flag called “stand_by_agv” (see chapter 3 *Debug*), a forced parking movement may be requested; if this flag is true, the next mission will be a movement to the standby point. This flag priority is higher than load/unload missions (even a loaded agv will go to the standby point instead of going to unload). No abort of current mission is needed. When the flag goes to false, normal load/unload task may work.

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.

3. Debug

Start AGVM and make the system work. See agv moving and try to disable some stations to see that agv must avoid choosing them as target.

Make a user panel with AGVM Qt Designer in order to allow user to handle a “stand_by_agv” flag (see **XForm** structure to handle communication with user panels in the xScript) and test its behavior.



To load or unload the agv, use this code in the loading/unloading macro. Remember that this piece of code is good in emulation only. During real work R3 program must write this status bit.

```
; load agv
if (EMUAGV_IsAgvEmulated(uAgv))
    vInfo.uStatus = vInfo.uStatus | VST_LOAD_ONBOARD
    EMUAGV_SetFlags(uAgv, vInfo.uStatus)
endif

; unload agv
if (EMUAGV_IsAgvEmulated(uAgv))
    vInfo.uStatus = vInfo.uStatus & ~VST_LOAD_ONBOARD
    EMUAGV_SetFlags(uAgv, vInfo.uStatus)
Endif
```

4. Video

In the video you can see the agv working and performing its load/unload task. When the “stand_by_agv” flag is switched on, the agv next mission is a motion to the standby point, then it remains there. You can also see the 5s timer happening after a complete unload mission. Graphically there is a picture of a box on the agv when it is loaded.

To make this feature, you can use a picture of your own and use this code:

```
; set the picture to the loading station when the product is ready
SetSiteImage(machine_id[i], "pic_load.png")
; reset the station picture to empty
SetSiteImage(machine_id[i], "")
; load the agv from the load station (also move the picture from station to agv)
AgvExecLoad(uAgv, iPar1)
; download the agv to the unload station (take out the picture)
AgvExecUnload(uAgv, iPar1)
```

Using this method, check that this bit is set false in OnApplicationStart function:

mpar.bPalletBloccaPercorso = false

You can also write the old but deprecated way:

mpar.setloadunitobstructspath(false)

Other way to handle pictures is to use **AgvSetAgvLoadInfo** and **AgvSetLoadImage** xScript commands, but they are suggested for emulation only.

```
; load agv
AgvSetAgvLoadInfo(uAgv, true, 0)
AgvSetLoadImage(uAgv, PIC_LOAD_ROBOX, true)
```

```
; unload agv
AgvSetAgvLoadInfo(uAgv, false, 0)
```