UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Exercise Sheet 3

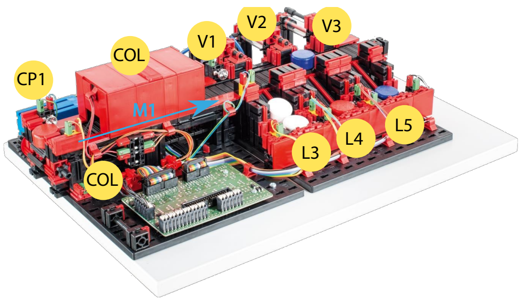
Process monitoring

Lecture *Real-Time Systems*, Summer semester 2021

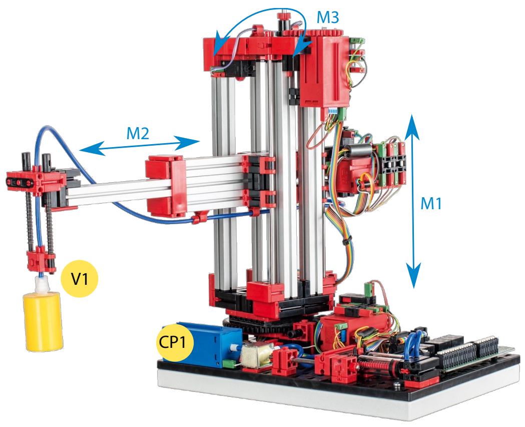
Dr. Javad Ghofrani
A discussion forum for the exercise can be found at: `moodle.uni-luebeck.de`.

In this exercise the control of the factory simulation is completed. Therefore the stations shown in 0.1a and 0.1b have to be implemented.

Use the TIA-Portal and SIMIT templates from the Moodle.



(a) Sorting Line With Color Detection - SLD



(b) Vacuum Gripping Robot - VGR

Figure 0.1: Stations - Not all signals are shown. Check the SIMIT simulation for missing signals.

The following signal tags are used to control the vacuum gripping robot:

	Tag	**Type**	**True**	**False**
Reference switches	S1 - S3	NO	triggered	untriggered
Motors	M1 - M3	-	turning	stoped
Pneumatic valves	V1	-	opened	closed
Air compressor	CP1	-	on	off
Encoder	E1 - E3	DInt	-	-
Encoder enable	E1_EN - E3_EN	Bool	enabled	disabled

In TIA Portal, the tags have the prefix **"VGR_"**. Control signals for motors also have a suffix that indicates the direction. A comment is also provided for each tag, which explains its meaning in more detail.

The vacuum gripping robot is used to pick up tokens at the high bay warehouse, to transport them to the multi processing station and to bring tokens from the sorting line back to the warehouse.

The robot consist of three axis which are dirven by motors. The motors are equipped with encoders to provide a position feedback. At the tip of the robot is a suction cup, which is controlled by a valve.

Initially the robot moves to its home position where the encoders are enabled. Starting there the robots position can be tracked by the encoder values **E1 - vertical movement**, **E2 - horizontal movement** and **E3 - rotation**. The home position of each axis is reached, when the cosponsoring reference switch is triggered.
After homing, all containers in the high bay warehouse are requested one after another and their contents transferred from the high bay warehouse to the multi processing station, where they are processed. After processing the sorting line with color detection sorts the tokens based on their color. It therefore moves the tokens into the detection chamber and reds the color sensor. The white tokens are moved to the first bay, the red ones to the second and the blue ones to the third bay. From there the vacuum gripping robot takes the tokens and brings them back to the high bay warehouse, where they are stored.

Encoders: To enable an encoder the corresponding axis has to be moved to its reference switch. When the switch is triggered the encoder enable signal has to be set to true. The encoder then returns the position referenced to its home position.

Moving the robot: To pick up a token from a specific position first the vertical axis has to be moved to its reference switch, to avoid collisions. Then the other axis are moved to their target position and lastly the vertical axis is lowered to its target position. The vertical axis is controlled with **M1**, the horizontal axis with **M2** and the rotation with **M3**.

Picking a token: To be able to pick up a token the compressor has to be turned on **(CP1)**. First the robot is moved to the tokens position, then the suction cup is turned on **(V1)** for at least two seconds before the robot can continue to move. After turning off the vacuum it takes two seconds for a token to detach from the suction cup.

The exact procedure at the individual stations is described in the following subtasks. The *pipelining* principle from the previous exercises is again used. Several workpieces can be processed or transported in parallel in the different stations. A predefined interface is used for the stations to communicate.

Exercise 1 VGR Control

(a) Position Control - any programming language

Create a function or function block that controls the VGRs movement. The function/function block gets the target height, target rotation and target extension of the robots axis as an input and outputs a true after the target position has been reached. To be able to precisely move to a position **VGR_VMOT** has to be controlled similar to exercise sheet one. When ever an axis is close to its target position it has to decelerate by decreasing the motor voltage. After the axis has reached its desired position, the voltage can be increased again.

(b) Station Control - GRAPH

Create a function block that handles the control of the VGR.
It needs three **"StationInterfaces"** as an input and one as an output. A second output with the type **"HBWInterface"** is also needed. Edit *"/Factory_Control[FB2]"* to match your naming scheme.

In its init-state the robot has to perform its homing procedure.
After homing a list of requests to the HBW has to be created. Implement a function that returns an array of the data type **"HBWRequest"**. This array has to contain get-requests for every container in the HBW and can be in any order. You can find the data type definition with comments under *"PLC data types/HBWRequest"*.

The task of the VGR controller is to process the list of requests, reducing waiting times for high throughput. For example, an empty container at the exit of the HBW can be used directly if a token is in one of the bays of the SLD, instead of sending it back to the warehouse first. Use interface **"HBWInterface"** and **"StationInterface"** to synchronise with other stations.

	Height	Rotation	Extension
HBW	2400	5400	660
MPS	1200	3540	3450
SLD white	150	1800	1400
SLD red	150	1470	1600
SLD blue	150	1220	2200

(a) List of relevant positions

Color	Sensor value	Return value
Nothing	19266	0
White	5724	1
Red	11501	2
Blue	17568	3

(b) Color mapping

Note:
You might need to use a two dimensional array to keep track which bay in the high bay warehouse is empty. 2D arrays can be created by adding a second dimension to a 1D array (e.g. Array[0..2, 0..2] of Int).
Counting in GRAPH can be done by using a static variable and using SCL commands in a step. "S1 N counter:= counter +1" can directly by written as an action of a state.

Exercise 2 Completing SLD

In this task you will complete the sorting system with color detection.

For this purpose you have to implement the mapping of the sensor values to colors. As soon as a token is under the color sensor, the sensor returns the values from the table above. The sensor values, are noisy and fluctuate depending on ambient light and the exact positioning of the token under the sensor.

Implement a function that maps sensor values (**SLD_COL**) to color intervals, always selecting the color with the smallest difference to the sensor value. The colors are coded as in the table above. Call this function in the *"CalcColor"* step of the *SLD_Control*-FB and assign the return value to the variable **"#tokenColor"**.

Exercise 3 Simulation

Simulate your PLC programme in SIMIT. Check that all stations are working.

Notes on the simulation:
The simulation starts with tokens of random color in every container in the HBW. The color is not shown until it is scanned in the SLD.

The simulation turned rather The simulation has become quite complex on the software side and therefore probably contains some bugs. So if a token disappears or gets stuck, this may be due to the simulation. But often this happens if the specified interfaces are not used correctly. Please first check if the handshakes between the stations are done correctly before asking for help.

Tokens at the output of a station disapear an are visible in the next stations view. This is a wanted behaviour.
Some problems may occur if the sim-speed slider is set to greater than one.

As on the to the previous task sheet, the simulation is computed on the PLC itself to minimise waiting times when starting the SIMIT simulation. You can reset the simulation by stopping and restarting the PLC in online mode. This requires, that the reset switch in SIMIT is switched on before the PLC is stopped. The reset switch must be switched off again after starting the PLC. It is still possible to reset the simulation by pressing the Play and Stop buttons in SIMIT.