Cyber-Physical System Security

Lab worksheet two

Objectives of Lab

- Set up and configure the water treatment scene in FactoryIO and OpenPLC.
- Set up and configure the bakery scene in FactoryIO and OpenPLC.
- Explore the two scenes and the sensors that they use.

Background

This lab follows on from our last tutorial, where we learnt how to set up and configure the different components required for a basic factory scene within FactoryIO. In this worksheet you will learn to create the water disinfection process from within a water treatment plant, and to create the mixing and baking process from within a bread factory. You will set up the ladder logic to control each of these scenes and explore the different sensors within each scene that allow the correct processes to run. The two processes we are creating will allow you to see the difference between continuous processes and batch processes, including understanding the different impacts that an attack can have on each type of process.

The FactoryIO scenes and .st files you will build are also available from http://www.cems.uwe.ac.uk/~pa-legg/uwecyber/cpss/

System requirements and Prerequisites

- Completion of lab one
- Raspberry Pi 3B+ running Raspbian Operating System (32-Bit Released 11.01.2021)
- OpenPLC Runtime V3
- OpenPLC Editor V1.0
- FactoryIO V2.4.6

Task One – Configure the water treatment

Step 1 – Creating the FactoryIO Scene

For our water treatment scene, we are going to take one of FactoryIO's pre-built scenes and modify it. The scene we will use for this is scene 3 – Filling Tank (Timers). The scene can be found under the scenes option once FactoryIO has opened.

Upon opening the scene, it will look similar to the image shown in Figure 1. It consists of a water tank, and a column with an electric switchboard. The switchboard has a pre-built start and stop button, and a digital display.

Figure 1: Scene 3 - Filling Tank (Timers)

To modify this scene for the water treatment scene, you will need to remove the stop button and add an extra digital display. Once you have done this it should look like the image in Figure 2, although the exact positioning of the buttons and display does not matter.

Figure 2: Water treatment scene

To make the start button and display true to our purpose, re-name the start button from 'Filling' to 'Start'. Name the first digital display 'Chlorine /100mg' and the second display 'PH Level'.

Figure 3: Water treatment display labels

Step 2 – Configuring the FactoryIO Server and Addresses

If you completed the previous lab sheet, then the majority of the server will already be set up correctly for the water treatment scene. Return to the driver screen before making your way to the configuration panel. Once configuration is open, you will need to change the register outputs count from zero to two, as shown in Figure 4 below.

Figure 4: Water treatment server configuration

Having configured the FactoryIO server to have the correct amount of holding registers, we now need to assign each of our parts to an address. Figure 5 contains the addressing scheme we used within FactoryIO for the water treatment scene.

Figure 5: Water treatment addressing

Step 3 – Creating the OpenPLC Ladder Logic

Prior to creating the ladder logic for the water treatment plant, we need to add our variables and set the equivalent locations to those we set in FactoryIO. Table 1 shows the addresses used in relation to the FactoryIO addresses.

Table 1: OpenPLC water treatment addressing

FactoryIO Name	OpenPLC Name	FactoryIO Address	OpenPLC Address
Start	I_PbFill	Input 1	%IX100.1
Tank 1 (Fill Valve)	O_FillValve	Coil 0	%QX100.0
Tank 1 (Discharge Valve)	O_DischargeValve	Coil 1	%QX100.1
PH Level	PH_Level	Holding Reg 1	%QW101.1
Chlorine/100mg	Chlorine_Level	Holding Reg 0	%QW100.1

Alongside our addressed variables, we will have other Boolean variables that help the program determine what process needs to be activated next. Our ladder logic will not follow the traditional style of sitting on rails, it will instead be blocks of subsequent code, that each trigger each other in a specific order.

The first block of code below, is the logic used to start our tank filling upon the start button being pressed. It uses a timer on function to trigger the fill valve being open for 70 seconds, if the tank is not currently discharging. This event is also triggered if the restart variable is triggered.

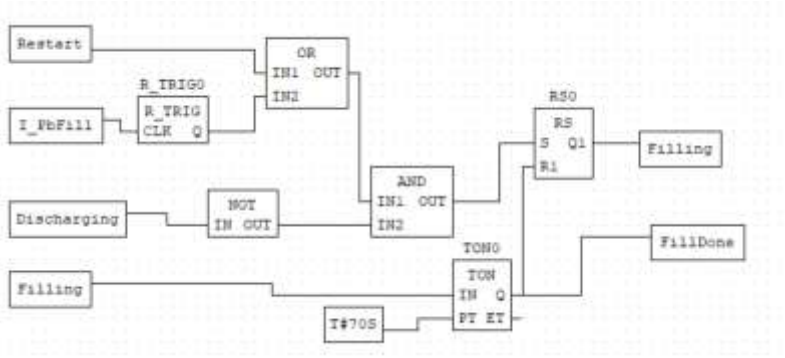

Figure 6: Water treatment starting/restarting block

Our next block of code is what is triggered when the tank filling is complete. It looks rather complicated but uses a set of nested timer on functions and count up and down functions to complete the purification process. Once the tank is full, we want to decrease the PH level from 7 to 5, and increase the chlorine level from 0/100mg to 4/100mg. This then needs to sit at this level for 8 seconds, prior to returning each variable back to it original state, which is safe for human consumption. Once the purification process is done, we use this to trigger the discharge process shown in Figure 8.

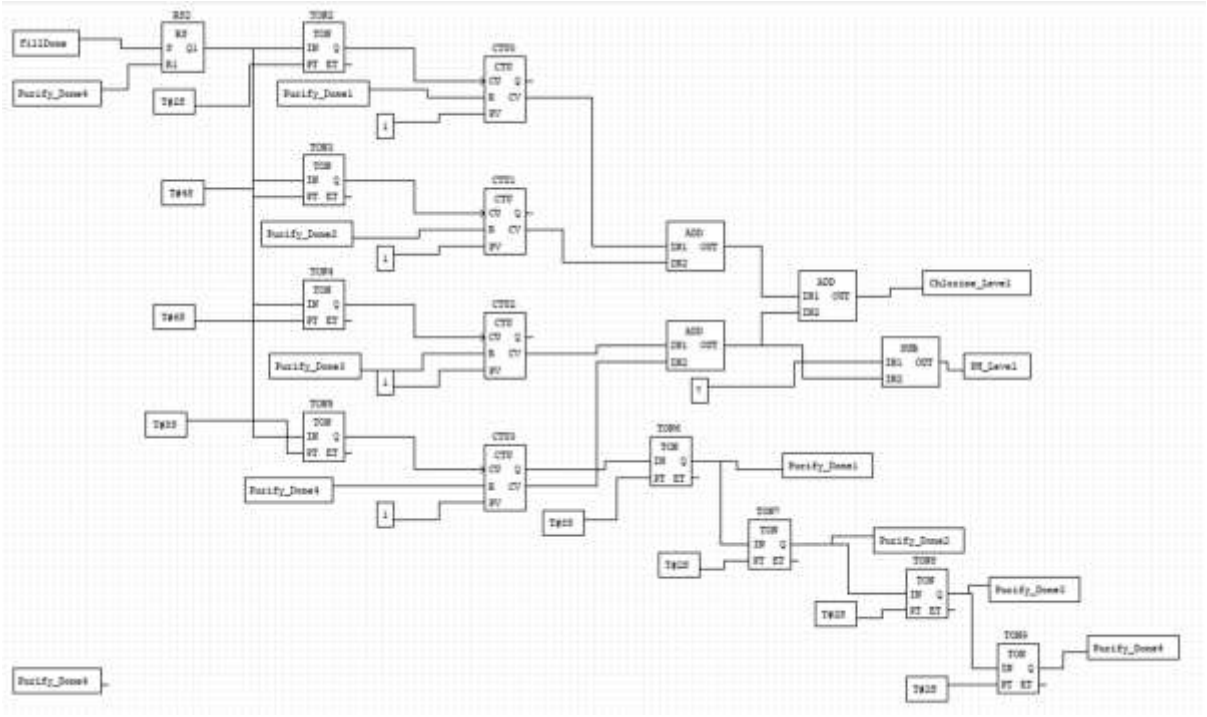

Figure 7: Water treatment purification block

The block shown below is used to discharge our water tank for 90 seconds, prior to restarting the entire purification process. It makes use of another timer on block to control the time, and uses a restart block to set the discharging variable to false once the process has complete.

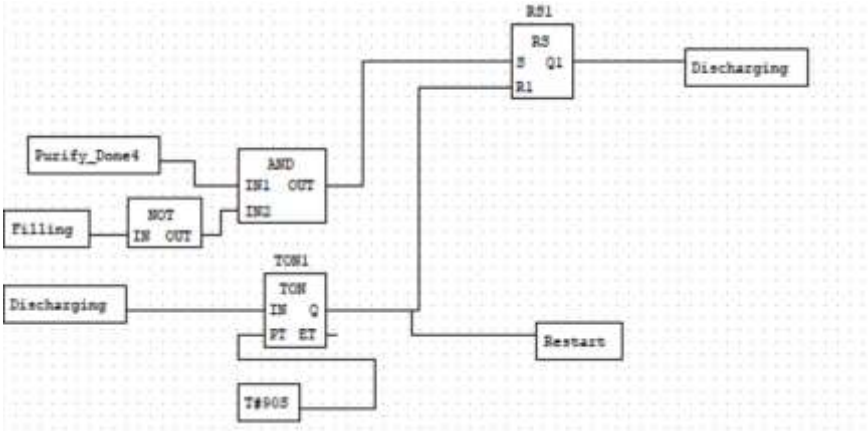

Figure 8: Water treatment discharging block

The final block of code within this ladder program is to link the filling and discharging variables to our filling and discharging valves. This is shown in Figure 9.

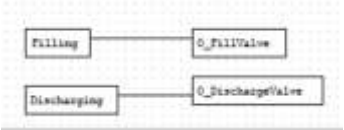

Figure 9: Water treatment variable link block

Do not forget to save this logic as a .st file so that it can be used with OpenPLC Runtime.

Step 4 – Configuring the OpenPLC slave device

To enable us to control our factory with this logic, we need to edit our slave device details on our OpenPLC Runtime instance. As we did with the FactoryIO server details, we need to set change Runtimes slave device to have 2 holding registers. Save the device with those settings and you should be able to activate the PLC to control the water treatment scenario.

Task Two – Configure the bakery

Step 1 – Creating the FactoryIO Scene

To create the bakery scene, we are going to open a completely new scene. Once you open a new scene, you will see an empty factory floor. To begin our scene, you will need to start on the far-left side of the factory and add a column with an electrical switchboard. You will need to place a start and a stop button on the switchboard, as this will allow us to control our batch process.

To create the beginning of our production line, we need to implement a conveyor belt, and a part emitter. You will need to change the emitter configuration to allow it to be controller defined, which can be done by right clicking the part once placed. The initial conveyor will eb longer than our other ingredient conveyors, to allow the space for the production to begin. At the end of the conveyor, you should implement a stop, and a small distance prior to that should be a diffuse sensor. For ease of configuration and programming later, we have named these after the ingredient being used in that section. Table 1 shows the exact

parts we have used in our bakery set-up, excluding the column, electrical switchboard, and buttons.

Table 2: Bakery Parts

Assigned Name of Part	Type of Part
Emitter 1	Emitter
Flour Conveyor	Belt Conveyor (4m)
Flour Diff	Diffuse Sensor
Flour Stop	Roller Stop
Yeast Conveyor	Belt Conveyor (2m)
Yeast Diff	Diffuse Sensor
Yeast Stop	Roller Stop
Mixing Conveyor	Belt Conveyor (2m)
Mixing Sensor	Diffuse Sensor
Salt and Sugar Conveyor	Belt Conveyor (2m)
SS Diff	Diffuse Sensor
Salt and Sugar Stop	Roller Stop
Additives Conveyor	Belt Conveyor (2m)
Additives Diff	Diffuse Sensor
Additives Stop	Roller Stop
Dry Mixing Conveyor	Belt Conveyor (2m)
Dry Mixing Sensor	Retroreflective Sensor
Dry Mixing Stop	Roller Stop
Oil Conveyor	Belt Conveyor (2m)
Oil Sensor	Diffuse Sensor
Oil Stop	Roller Stop
Water Conveyor	Belt Conveyor (2m)
Water Sensor	Diffuse Sensor
Water Stop	Roller Stop
Wet Mixing Conveyor	Belt Conveyor (2m)
Wet Mixing Sensor	Retroreflective Sensor
Wet Mixing Stop	Roller Stop
Splitting Conveyor	Belt Conveyor (4m)
Splitting Sensor	Retroreflective Sensor
Split	Remover
Emitter 2	Emitter
Proofing Oven Conveyor	Belt Conveyor (4m)
Proofing Oven Sensor	Retroreflective Sensor
Baking Oven Sensor	Diffuse Sensor
Connector Conveyor	Curved Belt Conveyor
N/A	Aligner 1
N/A	Aligner 2

Baking Oven Conveyor	Belt Conveyor (6m)
Stop Conveyor	Belt Conveyor (2m)
End Mixing and Baking	Remover

As with the first emitter, the second will also need to be set to controller defined. The list of parts within Table 1, shows the order that the parts are assembled in. The following images will show how our bakery scene looks, to assist with visualising how it should look.

Figure 10: Bakery section one

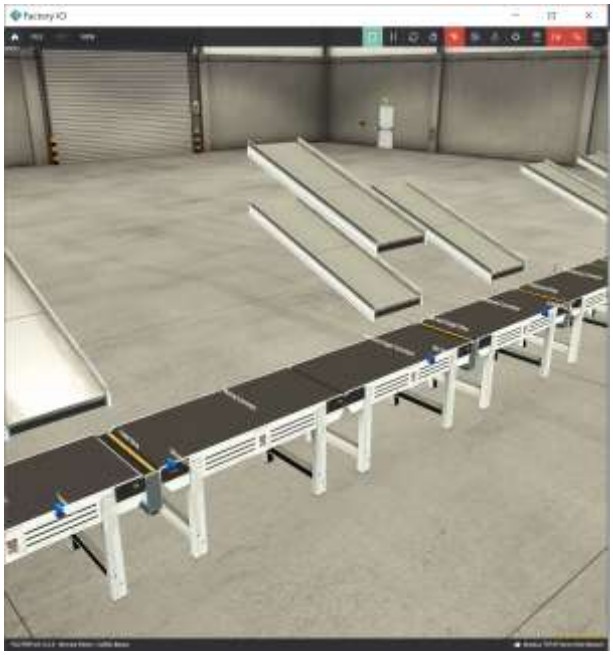

Figure 11: Bakery section two

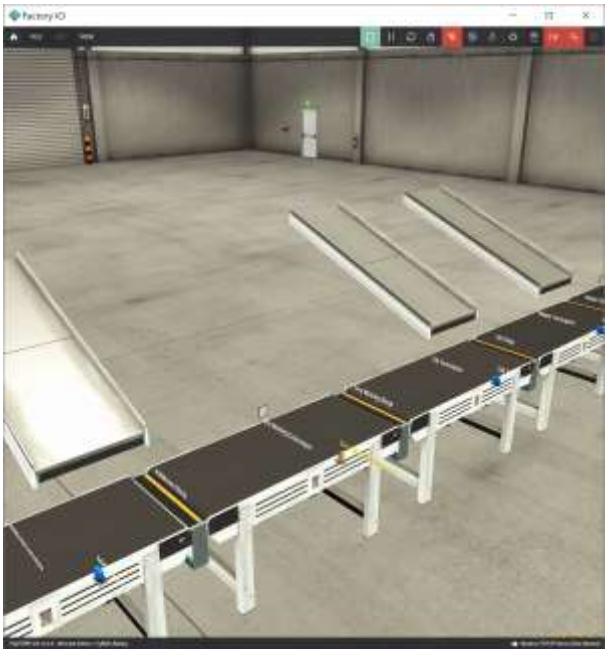

Figure 12: Bakery section three

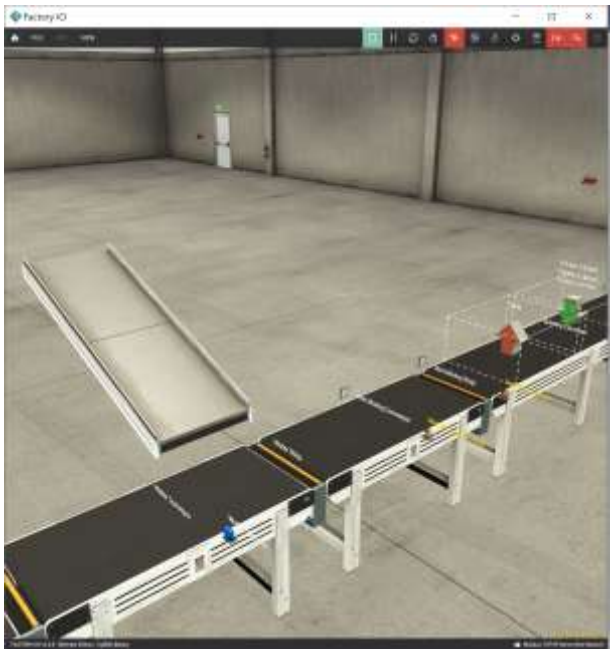

Figure 13: Bakery section four

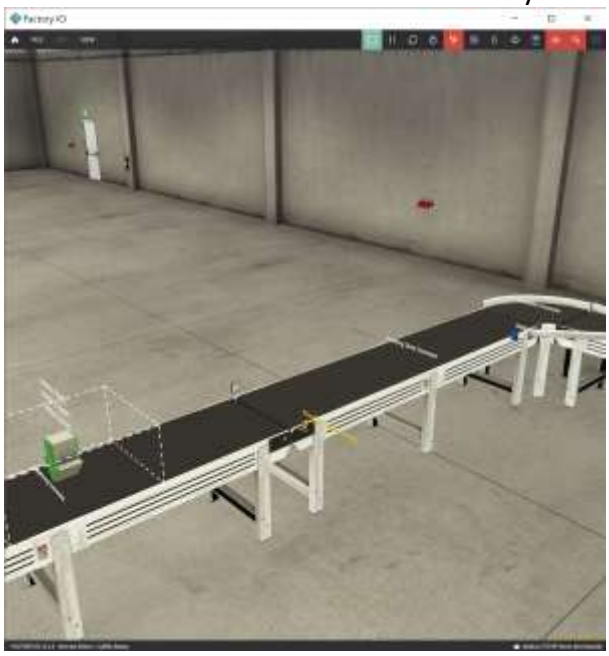

Figure 14: Bakery section five

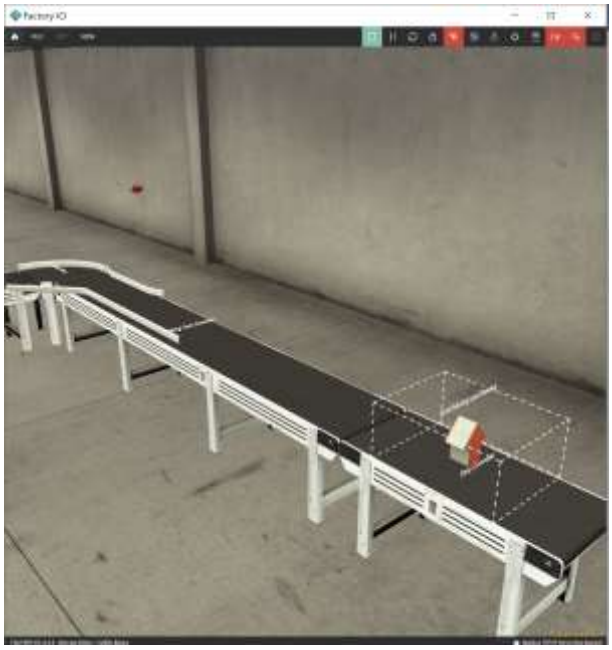

Figure 15: Bakery section six

Figure 16: Bakery - full scene

As you can see within the images, we have used chute conveyors to illustrate where the ingredients would typically come out.

Step 2 – Configuring the FactoryIO Addresses

Following on from the water treatment set-up, we will need to further reconfigure the server to make room for all of our parts. As such, you will need to adjust the digital inputs to 16, digital outputs to 26 and the register outputs to 8. This will give us sufficient room for all our components.

Figure 17: FactoryIO bakery server configuration

Having configured the FactoryIO server to have the correct amount of holding registers, we now need to assign each of our parts to an address. Figure 18 contains the addressing scheme we used within FactoryIO for the bakery scene. It is important to recognise that each emitter, has three corresponding parts when it comes to addressing. One part to emit, and the other two to decide the base and part that it emits to the factory.

Figure 18: FactoryIO bakery addressing

It is important to note here that our addressing system with OpenPLC will not match exactly with the addresses written here. This is due to OpenPLC reserving all addresses with an 8 or a 9 in them, so those shown at those addresses will need to be adjusted accordingly.

Step 3 – Creating the OpenPLC Ladder Logic

To create the ladder logic for the baker scene, this first thing to do is to add all the initial variables and their addresses based on the parts we used in FactoryIO. To help with the addressing, we have named each variable similarly in the ladder logic as we did in FactoryIO, and Table 2, shown below, will show the part name and its corresponding addresses.

Table 3: OpenPLC bakery scene addressing

Part Name (FactoryIO)	Part Name (OpenPLC)	Addressing (FactoryIO)	Addressing (OpenPLC)
Start	I_Start	Input 1	%IX100.1
Stop	I_Stop	Input 2	%IX100.2
Flour Conveyor	Conveyor_Flour	Coil 0	%QX100.0
Yeast Conveyor	Conveyor_Yeast	Coil 1	%QX100.1
Salt and Sugar Conveyor	Conveyor_SS	Coil 2	%QX100.2
Additives Conveyor	Conveyor_Additives	Coil 3	%QX100.3
Oil Conveyor	Conveyor_Oil	Coil 4	%QX100.4

Water Conveyor	Conveyor_Water	Coil 5	%QX100.5
Emitter 1 (Emit)	SpawnA_1	Coil 6	%QX100.6
Emitter 1 (Base)	SpawnA_2	Holding Reg 0	%QW100.1
Emitter 1 (Part)	SpawnA_3	Holding Reg 1	%QW101.1
Emitter 2 (Emit)	SpawnB_1	Coil 20	%QX102.4
Emitter 2 (Part)	SpawnB_2	Holding Reg 2	%QW102.1
Emitter 2 (Base)	SpawnB_3	Holding Reg 3	%QW103.1
Flour Diff	FlourDiff	Input 3	%IX100.3
Yeast Diff	YeastDiff	Input 4	%IX100.4
Mixing Sensor	MixingSensor	Input 5	%IX100.5
SS Diff	SSDiff	Input 6	%IX100.6
Additives Diff	AdditivesDiff	Input 7	%IX100.7
Dry Mixing Sensor	DryMixingSensor	Input 8	%IX101.0
Oil Sensor	OilSensor	Input 9	%IX101.1
Water Sensor	WaterSensor	Input 10	%IX101.2
Wet Mixing Sensor	WetMixing Sensor	Input 11	%IX101.3
Splitting Sensor	SplittingSensor	Input 12	%IX101.4
Proofing Oven Sensor	ProofingOvenSensor	Input 13	%IX101.4
Baking Oven Sensor	BakingOvenSensor	Input 14	%IX101.5
Flour Stop	FlourStop	Coil 7	%QX100.7
Yeast Stop	YeastStop	Coil 8	%QX101.0
Mixing Conveyor	MixingConveyor	Coil 9	%QX101.1
Salt and Sugar Stop	SSStop	Coil 10	%QX101.2
Additives Stop	AdditivesStop	Coil 11	%QX101.3
Dry Mixing Conveyor	DryMixingConveyor	Coil 12	%QX101.4
Dry Mixing Stop	DryMixingStop	Coil 13	%QX101.5
Oil Stop	OilStop	Coil 14	%QX101.6
Water Stop	WaterStop	Coil 15	%QX101.7
Wet Mixing Conveyor	WetMixingConveyor	Coil 16	%QX102.0
Wet Mixing Stop	WetMixingStop	Coil 17	%QX102.1
Splitting Conveyor	SplittingConveyor	Coil 18	%QX102.2
Split	Split	Coil 19	%QX102.3
Proofing Oven Conveyor	ProofingOvenConveyor	Coil 21	%QX102.5
Connector Conveyor	ConnectorConveyor	Coil 22	%QX102.6
Baking Oven Conveyor	BakingOvenConveyor	Coil 23	%QX102.7
Stop Conveyor	StopConveyor	Coil 24	%QX103.0
End Mixing and Baking	EndMixing	Coil 25	%QX103.1

To create our ladder logic, we first need to understand exactly what we want the scene to do. Upon the user pressing the start button, we want all the ingredient conveyors (Flour, Yeast, Salt and Sugar, Additives, Oil and Water) to activate and the connecting conveyor, stop conveyor and final remover to also activate. With these conveyors, and our other ones we have multiple sensors, which need to trigger stops and our other conveyors. The following list describes what we want to happen when each sensor is triggered. The sensors are listed in order of how they are set up in the factory, so are in the correct chronological order.

- Flour sensor triggered = flour stop activated for 12 seconds
- Yeast sensor triggered = yeast stop activated for 6 seconds
- Mixer sensor triggered = mixer stop activated for 12 seconds
- Salt and sugar sensor triggered = salt and sugar stop activated for 8 seconds
- Additives sensor triggered = additives stop activated for 6 seconds
- Dry mixer sensor triggered = dry mixing stop activated for 14 seconds
- Oil sensor triggered = oil stop activated for 7 seconds
- Water sensor triggered = water stop activated for 9 seconds
- Wet mixer triggered = wet mixing stop activated for 49 seconds and splitting conveyor activated
- Splitting sensor triggered = activate second emitter and first remover
- Proofing oven sensor triggered = activate proofing oven
- Baking over sensor triggered = activate baking oven

Due to the way OpenPLC works with the variables we have set, traditional ladder logic using rails has not been used. Instead, we have used order-based ladder logic blocks. The following figures depict our ladder logic, in the correct order and will give a brief description on what each block is doing.

At the very top of our program, we set our variables for the emitters. One of these is shown below, and the binary figures correlate to the parts we want the emitter to produce. The new variable you can see is to link it to be triggered by another logic block further within the program. Figure ** is for our second emitter, but the first emitter uses the same logic, with a larger box being set and a different trigger variable.

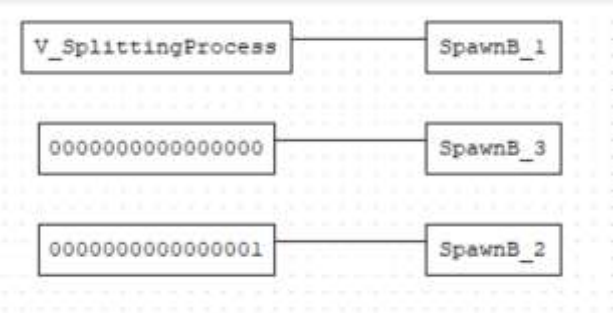

Figure 19: Bakery emitter code block

We then have four identical blocks of code, each triggering different variables. These blocks use a reset block to create the start and stop logic for each of our processes that start upon the batch process being started, such as our ingredient conveyors.

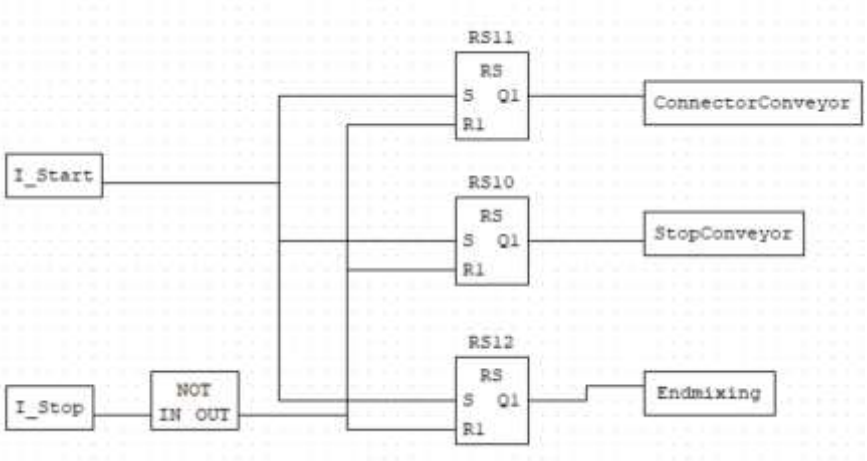

Figure 20: Bakery starting blocks

Following this, we have blocks that activate the different processes once a sensor has been triggered. The blocks use a time pulse to activate the stops or the conveyors for a given period. These blocks are repeated and modified to complete all the necessary processes in our bakery.

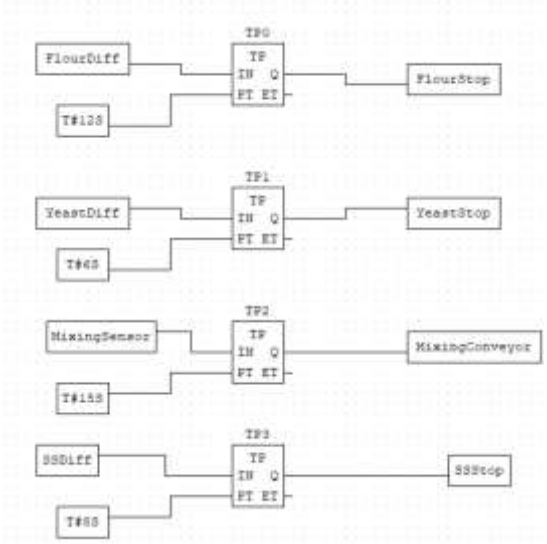

Figure 21: Bakery process blocks

Do not forget to save the logic as a .st file for use on the OpenPLC instance.

Step 4 – Configuring the OpenPLC slave device

To enable us to control our factory with this logic, we need to edit our slave device details on our OpenPLC Runtime instance. As we did with the FactoryIO server details, we need to set Runtime's slave device to have 16 discrete inputs, 26 coils and 4 holding registers. Save the device with those settings and you should be able to activate the PLC to control the bakery scenario.

Task Three – Explore the water treatment scene and sensors

Having built the water treatment scene, take some time to experiment with the factory. What happens if you were to force a value on one of the actuators? Does it make a difference if you change the timing values on the filling or discharging?

Task Four – Explore the bakery scene and sensors

Similarly, to the water treatment scene, take some time to full explore the bakery scene. Can you think of any small changes you could make to prevent part of the scene from working correctly? Could you implement further sensors within the scene?

Takeaways

Having completed this worksheet you should have been able to build both a bakery and a water treatment scenario, and create some ladder logic that controls it. You should have an understanding of the amount of sensors that go into a factory and be able to set these different factories up correctly to be controlled by OpenPLC.

Further reading

Cyber-Physical Systems Security Knowledge Area Issue 1.0 – available from https://www.cybok.org/knowledgebase/

FactoryIO – available from https://docs.factoryio.com/

FactoryIO Emitter details – available from https://docs.factoryio.com/manual/parts/emitter/