Python script that models communication between components in a concrete 3D printing system

Author

Keyvan Rahmani Monfared

March 24, 2024

Table of contents

Problem Definition

The task is to write a Python script that models communication between components in a concrete 3D printing system.The components of the system are a robot, a concrete pump, a concrete flow valve, an operating panel and an user interface (Figure 1). They are connected to a control PLC through which they communicate. There are a number of input and output signals from each component to the PLC, for example component state, connection status etc. The robot controls the activity of the pump and valve by sending commands through the PLC. The pump, robot and the valve also have an option of being activated and deactivated manually through physical switches on the operating panel and signals from a user interface.

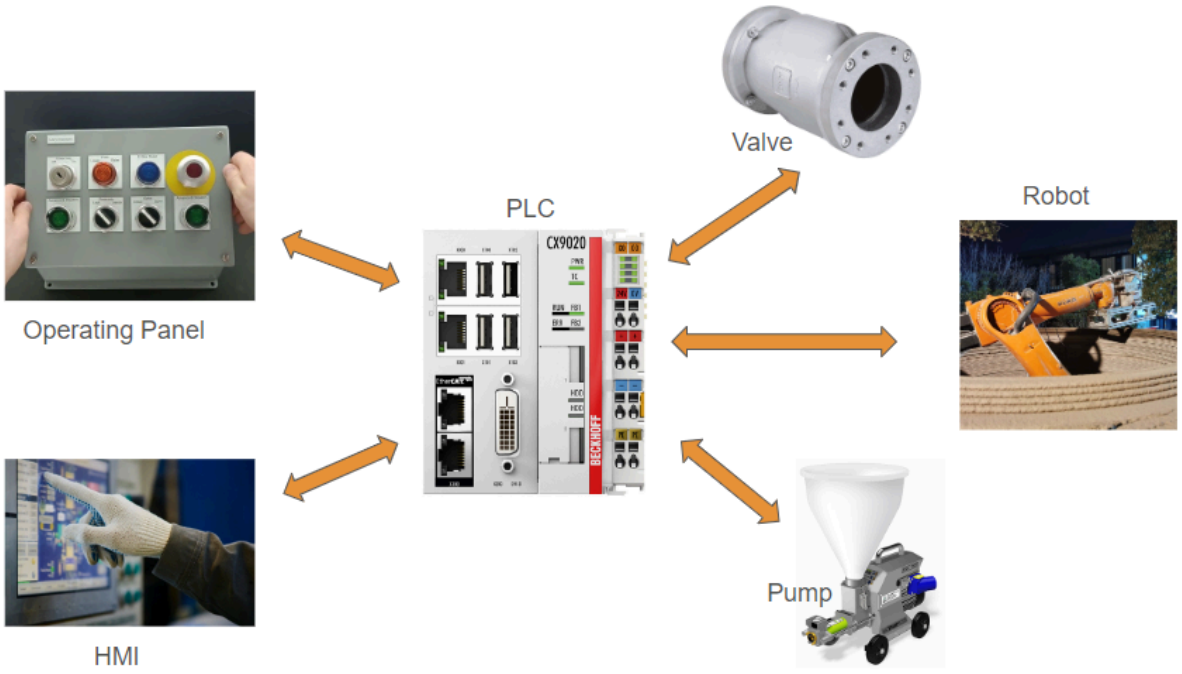

Figure 1: Different components of the concrete 3D printer system

PLC Inputs

The list of the PLC inputs is shown in Table 1. Input addresses begin with the letter 'I', followed by the specific component or device responsible for generating the signal. The subsequent details pertain to the functionality of the input address. For example 'I_Switch_valve_keep_open' begins with 'I' indicating input address. ‘Switch’ indicates that this input address belongs to a switch. The rest ‘valve_keep_open’ explains what this switch is used for.

Inputs			
	Address	Signal Source	Comment
1	I_Swith_Pump	Operating Panel	On-Off switch to turn the pump on and off
2	I_UI_Pump	User Interface	On-Off switch to turn the pump on and off
3	I_Swith_Valve	Operating Panel	On-Off switch to turn the valve on and off
4	I_UI_Valve	User Interface	On-Off switch to turn the valve on and off
5	I_Robot_Start_Pump	Robot	Command the pump to start
6	I_Robot_Stop_Pump	Robot	Command the pump to stop
7	I_Robot_Start_Valve	Robot	Command the valve to start
8	I_Robot_Stop_Valve	Robot	Command the valve to stop
9	I_Swith_Robot	Operating Panel	On-Off switch to turn the robot on and off
10	I_UI_Robot	User Interface	On-Off switch to turn the robot on and off
11	I_Switch_valve_keep_open	Operating Panel	On-Off switch to keep the valve open. his signal overrides any other command.

Table 1: PLC inputs

Process Flowchart

Figure 2 indicates the diagrammatic representation of the separate steps of the process in sequential order. As can be seen this flowchart contains multiple functions. The flowchart of these functions is shown in Figure 3 to Figure 8.

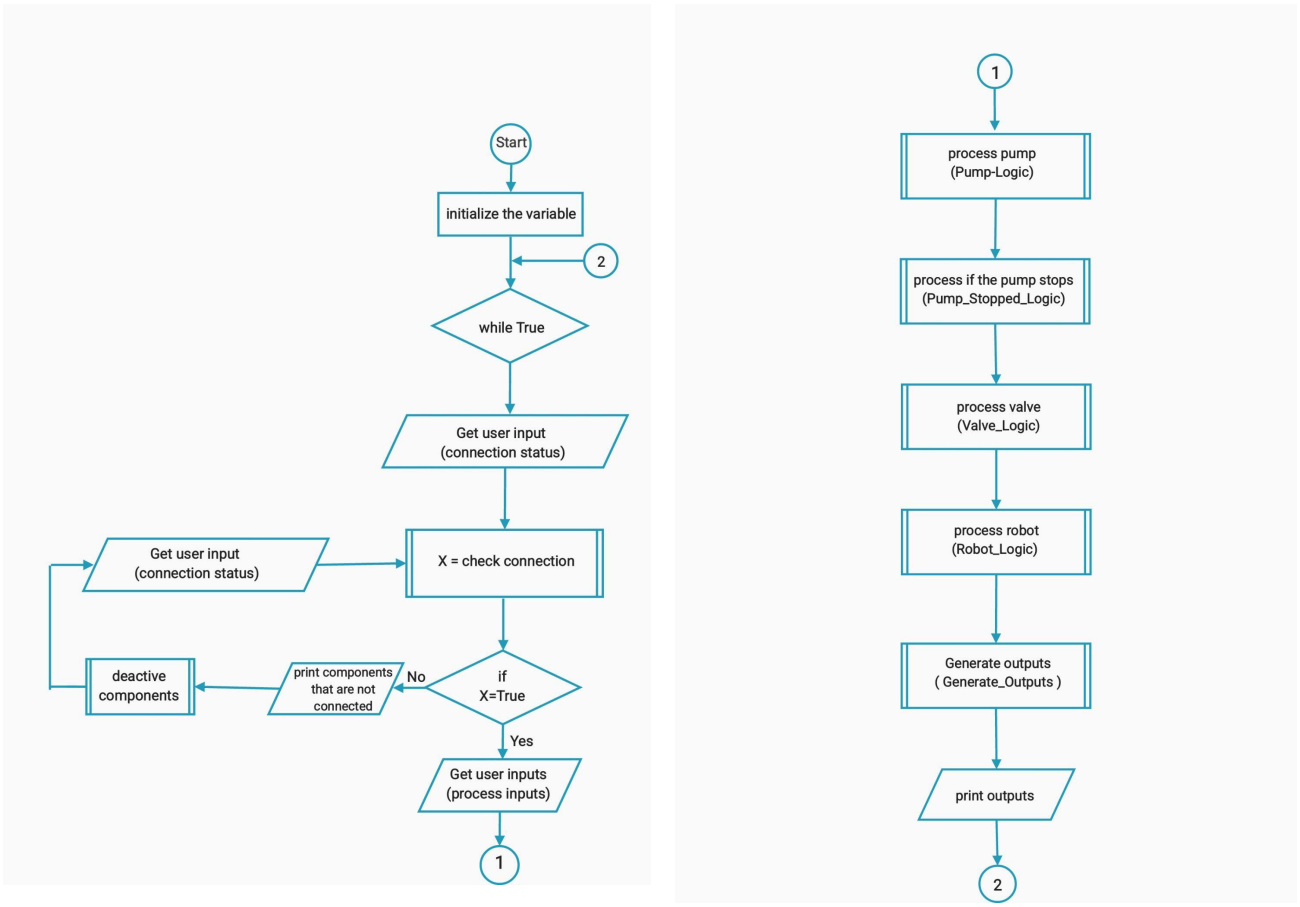

Figure 2: Flowchart of the process

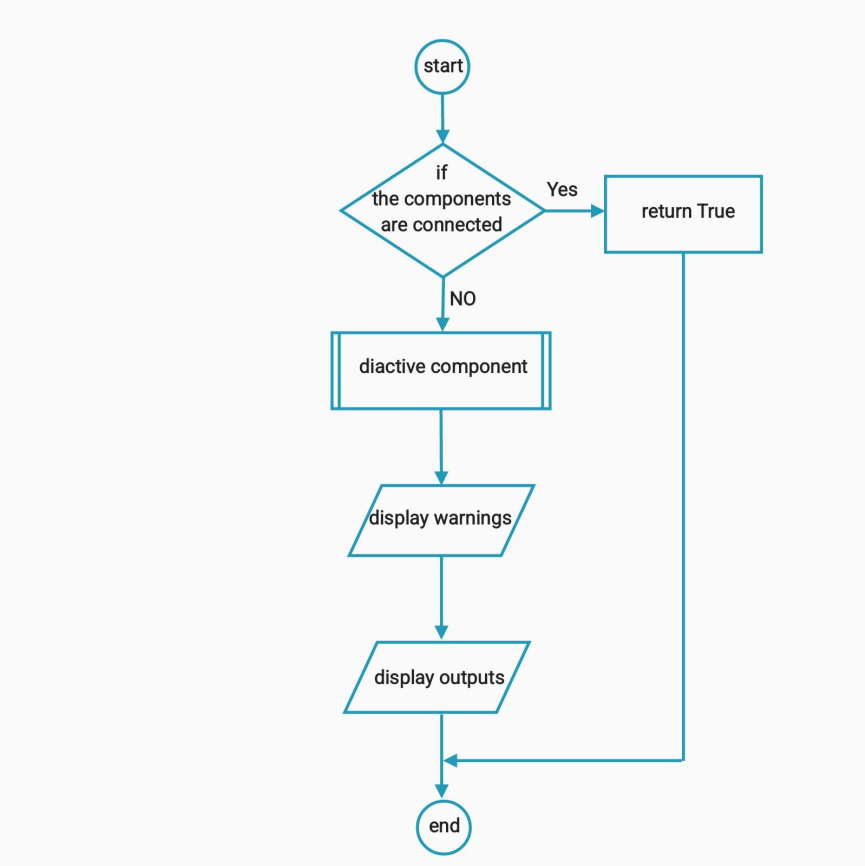

Figure 3: Flowchart representation of the Check_Connection function

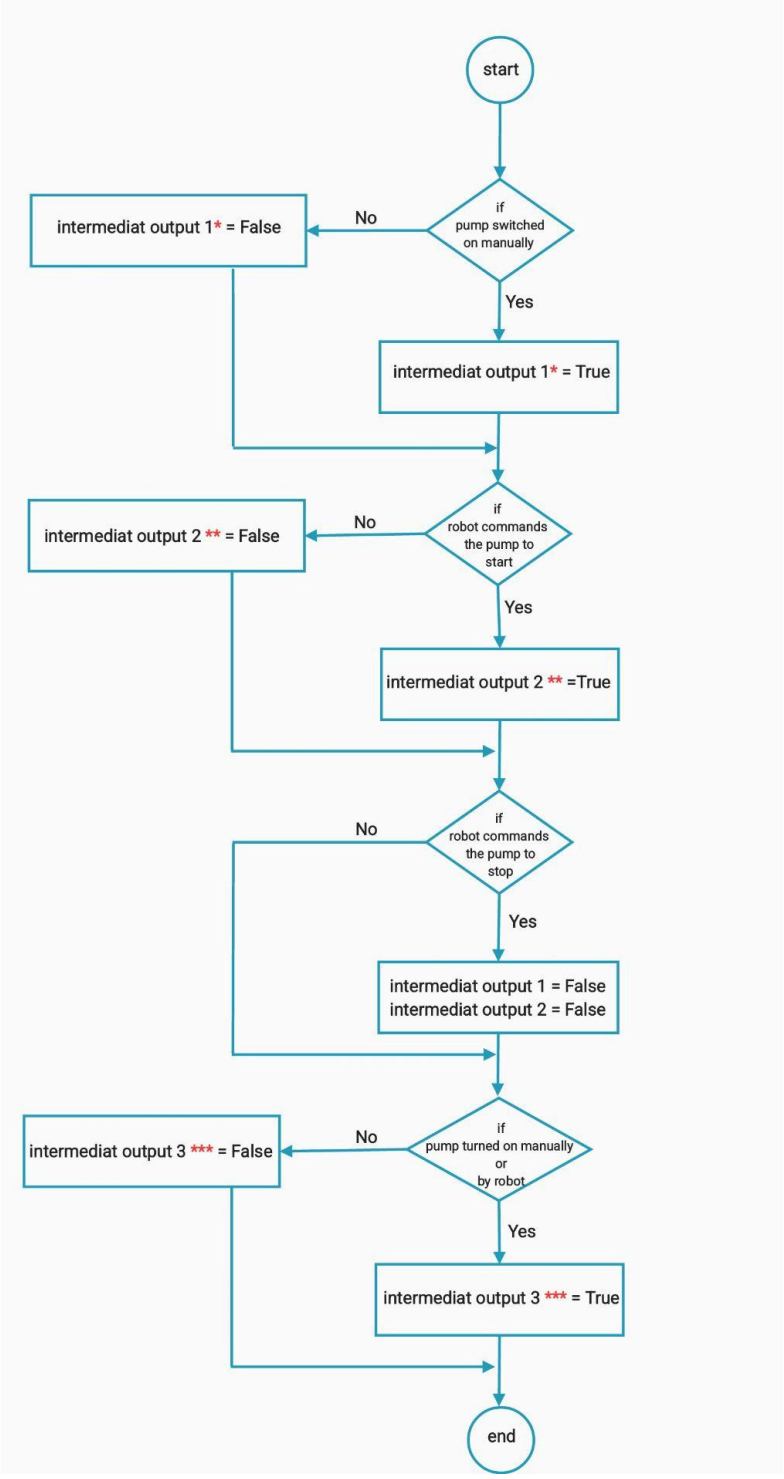

Figure 4: Flowchart representation of t the Pump Logic function

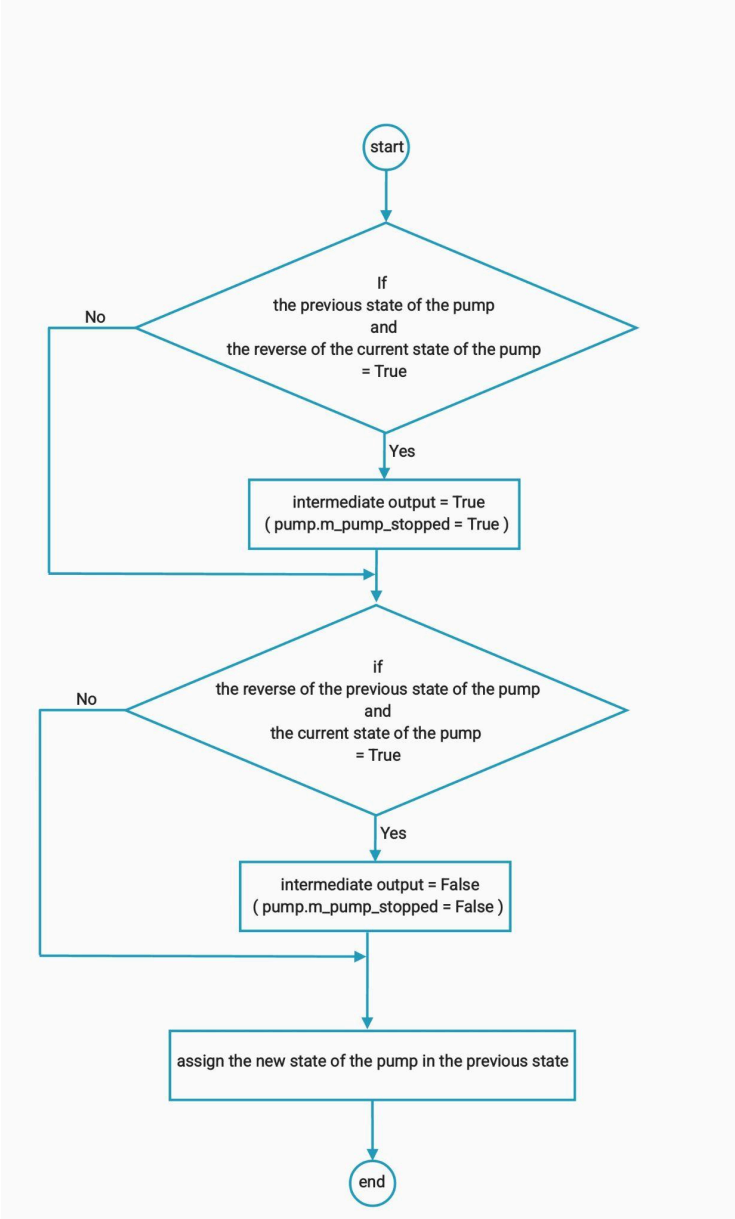

Figure 5: Flowchart representation of the Pump_Stopped_Logic function

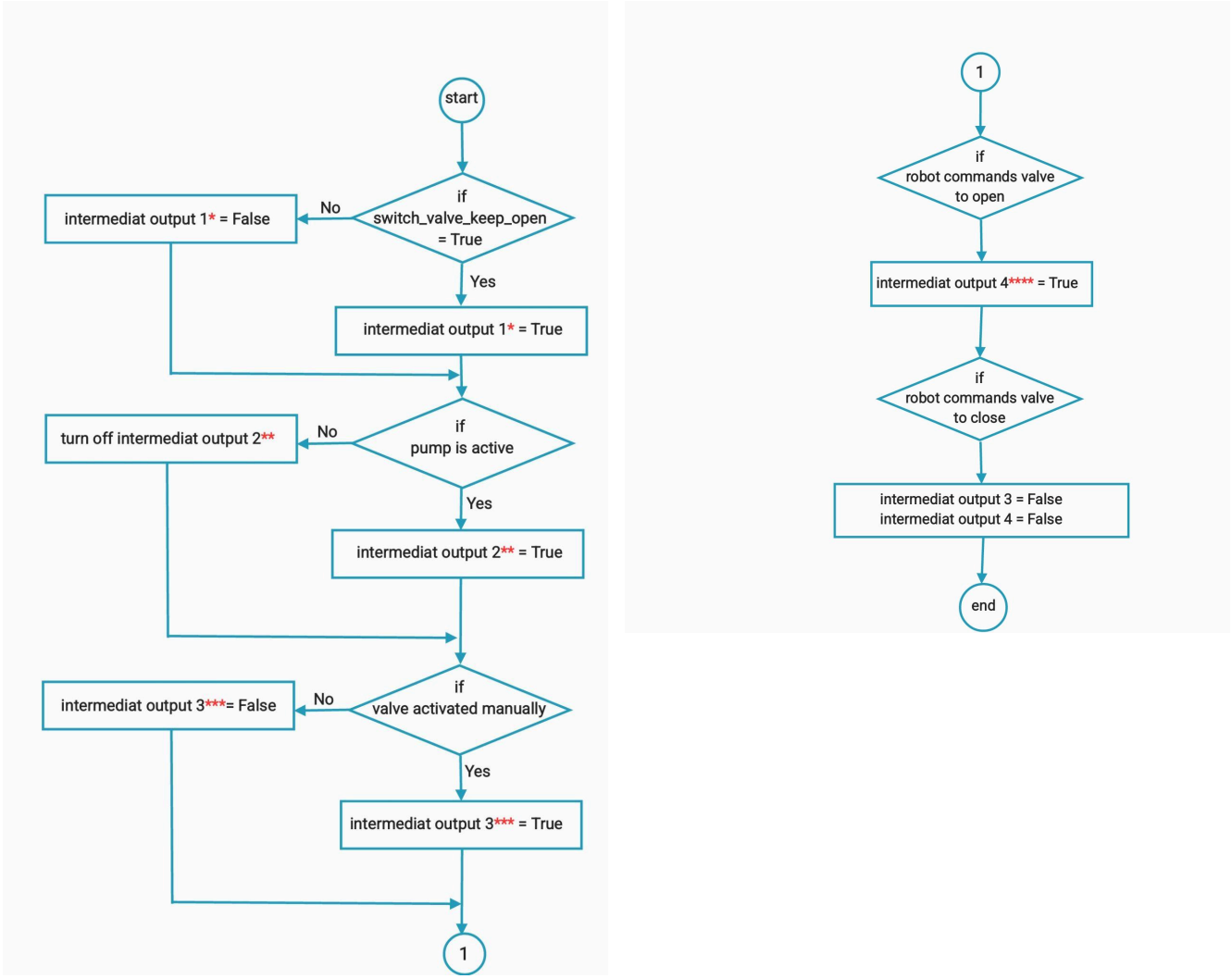

Figure 7: Flowchart representation of the Valve_Logic function

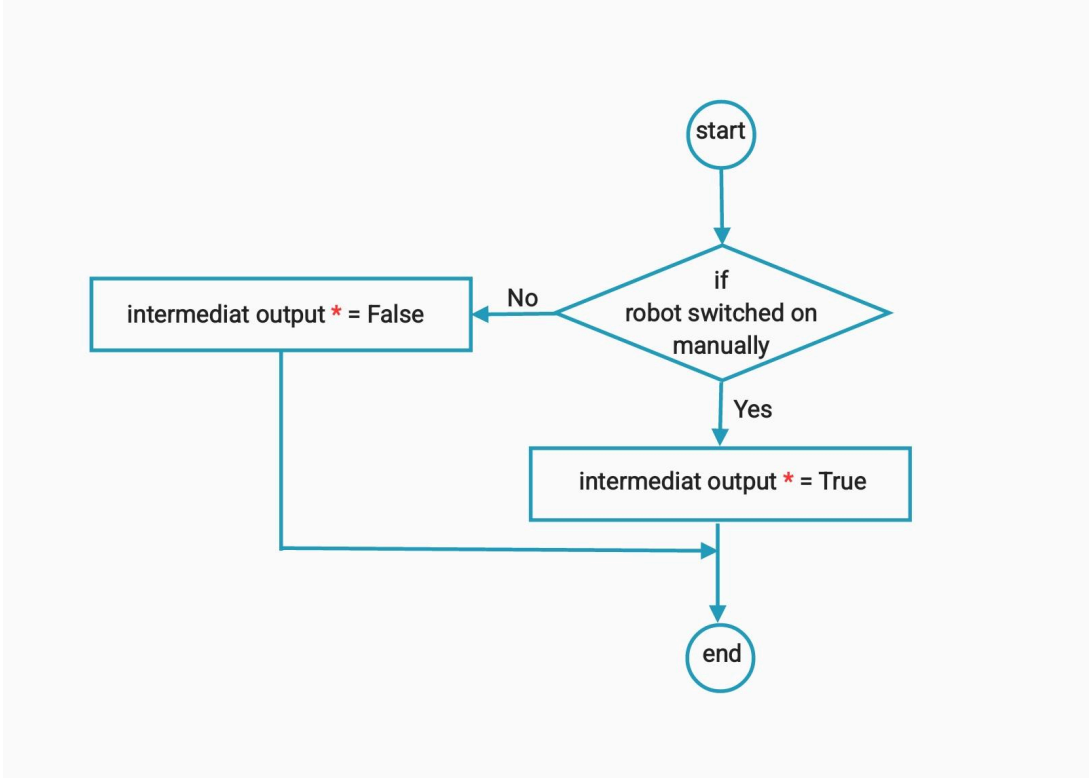

Figure 8: Flowchart representation of the Robot_Logic function

How to Test and Work with the Programme

At the beginning of each cycle, the program checks the connection of components (pump, robot, valve). If the connection was established, it reads the inputs. But if all the components are not connected, it checks the connections again until all the components are connected. For simulation, the user determines which components are connected and which are not connected. The user enters an array with three elements. These elements show the connection of pump, valve and robot respectively. For example, the array [1 , 1, 0] shows that the pump and valve are connected and the robot is not connected. The program also asks the user to enter the array of PLC inputs that are shown in Table 1. For example [1,0,0,0,0,0,0,0,0,0] indicates that I_Swith_Pump is on and the other inputs are off.

Test 1

If any of the components is not connected to the control PLC, all other components should be disabled until the connection state is restored.

Test procedure

At the beginning of each cycle, the program checks the connection of components (pump, robot, valve). The program asks the user to determine which components are connected and which are not connected. For the test, set one of the connections to 0. For example, the array [1 , 1, 0] shows that the pump and valve are connected and the robot is not connected.

Test 2

If the pump stops, the robot should also be stopped until the pump operation is restored.

Test procedure:

Follow the steps below and enter the array indicated in each step in the program console.

1) Enter the array of connections: [1 , 1, 1].
2) Start the pump and the robot : [1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1, 0 , 0].
As a result, the pump, valve and robot are turned on.
3) Enter the array of connections : [1 , 1, 1].
4) Stop the pump but leave the robot switch on : [0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1, 0 , 0].
It can be seen in the output, although the robot switch is on, but because the pump is turned off, the robot is also turned off.
5) Enter the array of connections : [1 , 1, 1].
6) Again leave the pump stopped and the robot's switch on : [0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1, 0 , 0].
In the output, it can be seen that the robot remains off.
7) Enter the array of connections : [1 , 1, 1]
8) Start the pump : [1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1, 0 , 0].
At the output, the pump and the robot will start again.

Test 3

The concrete flow valve should always be open if the pump is active, even if the robot commands it to close.

Test procedure

Follow the steps below and enter the array indicated in each step in the program console.

1) Enter the array of connections : [1 , 1, 1].
2) Start the pump: [1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0, 0 , 0].
It can be seen that even though we did not open the valve, when the pump is turned on, the valve is also turned on.
3) Enter the array of connections : [1 , 1, 1].
4) Pump is on, robot commands the valve to close: [1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0, 0 , 0].
The output shows that even though the robot commands the valve to close, the valve remains open.
5) Enter the array of connections: [1 , 1, 1]
6) Stop the pump, and the robot commands the valve to stop:
[0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0, 0 , 0].
In the output, we see that the valve is closed.

Test 4

The pump and the valve should have an option of being activated and deactivated manually (for example, using a physical switch or signal from a UI, but command from the robot should have precedence over the manual operation signals.

Test procedure

Follow the steps below and enter the array indicated in each step in the program console.

1) Enter the array of connections: [1 , 1, 1].
2) Open the valve manually: [0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0].
3) Enter the array of connections: [1 , 1, 1].
4) Robot commands the valve to close, and operator opens the valve manually:
[0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0].
In the output we see that the valve is closed because the robot signal overrides other signals.

Test 5

The valve should have the option of being kept open by a signal that overrides any other command.

Test procedure

Follow the steps below and enter the array indicated in each step in the program console.

1) Enter the array of connections : [1 , 1, 1].

2) Keep the valve open by I_switch_valve_keep_open:

[0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1].

3) Enter the array of connections : [1 , 1, 1].

4) Robot commands the valve to close, but I_switch_valve_keep_open is active:

[0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1].

The output shows that although the robot commands the valve to close , the valve remains open.