POLITECNICO DI TORINO

Master’s degree in Mechatronic Engineering

Fluid Automation

TASK 2

REPORT

Student:

Muratore Luigi **s333098**

Prof.:

Luigi Mazza

2023/2024

CONTENT

Task requirements

In our project, we needed to develop an automatic conveyor system to process and store steel plates coming from a steel mill.

DESCRIPTION

In the design of our machine, we have two conveyors, a press and a storage station. The top conveyor can transport three steel plates simultaneously, which are dropped onto a second conveyor to form a stack.
As the stack is complete, the top conveyor stops to allow for the extension of a pneumatic cylinder. Such a cylinder presses on the stack for 3 seconds before retracting.
The stack is then transported to a wooden crate for shipping, while the top conveyor is reactivated to build the next stack.
The wooden crate at the end can store the stacks of plates in two columns, with a maximum of three stacks each. Two openings at the top allow for the stacks to be lowered in the correct column: as soon as the first column is full, the second opening is activated to direct the incoming stacks to the second column, while the first opening is closed off. When the whole crate is full, it is ready to be shipped so it must be transported to the designated storage area of the plant.

SOLUTIONS

To simulate the boxes on the conveyors and their motion, we linked the linear displacement of pneumatics cylinders with the linear movements of the boxes.
Essentially, we have created two boxes for each transition, wherein one box is visible and linked to a cylinder displacement, causing it to move when the cylinder goes out-stroke. While the second box remains stationary and replaces the first box, making it visible when the cylinder goes in-stroke.
Since no more than three plates could be loaded on the first conveyor at the same time due to weight limits, we just created three pistons for the first conveyor so that there were at most three boxes visible simultaneously.

Another requirement was that the operator shall be, actually, able to change the number of plates that each stack is made of.

To handle this … we created an input section where the operator could insert the number of pieces for the stack at each cycle by means of a numeric pad.

We structured the whole system in two main cycles:
The first one is the *STACK-CYCLE,* the one that goes from the input of the operator through the creation of the stack, the press step up to the stacking of the pressed pieces in the storage area.
The second cycle is the COMPLETE-CYCLE, and it is composed of six stack-cycles because the storage area can accept up to six stacks before filling up. Then there is the need to ship the pallet and set a NEW CYCLE.

To manage the whole moving process, we divided it into phases, from 1 to 5, considering also the phase-back of each one, so from 1-back to 5-back. We have ten phases overall. The rest of the processes are handled by animations and visibility functions.
The firsts six phases are responsible for the three boxes on the first conveyor (from *phase1* to *phase3_back*). Then we have the phase for the press step (*phase4* and *phase4_back)*, and in the end the phases related to the pressed stack on the second conveyor (*phase5* and *phase5_back*).

According to the requirements, the incoming plates had to be loaded on the first conveyor according to a clock signal generated by a Pulse.
So, we create a loop on the *phase1* (the one related to the visibility and motion of the first box) so that, with a variable *piece*, send a pulse that recalls *phase1* as many times as the number of the pieces that the operator gave as input.

So, we developed the automatic control to implement the task with:
- CODESYS (programming)
- Fluid Sim (plant)

using LADDER CODE and STRUCTURED TEXT languages in proper well organized function blocks.
Moreover, we developed a clear and understandable HMI interface in CODESYS with all the necessary conditions needed to simulate the real plant.

FluidSIM

__FluidSIM__ is a simulation software developed by __Festo__ Didactic, designed for the creation and analysis of pneumatics, hydraulics, and electrical circuits.

We used FluidSIM to build the plant in terms of **pneumatic circuit** and **PLC interface.**
We utilized the **Library Explorer** to gather components and all necessary items, making use of different sections, in particular:

- **Pneumatic** → Actuators, Directional Valves, Flow control valves, Sensors
    - double acting cylinders
    - 5/2 directional valve with 2 stable positions electrically switched by solenoids on both sides to control each cylinder.
    - 2/2 directional valve with 1 stable position, pneumatically controlled
    - 3/2 directional valve with 1 stable position, electrically switched by solenoids to manage the emergency and stop mode.

    - sensors (sensor ref. Bidirectional)
        - A0, A1, B0, B1, C0, C1, P0, P1, D0 to detect the in-stroke or out-stroke of the pistons

    - displacement encoders:
        - conv1, conv2, conv3, press_piston, conv4 that provide motion feedback for linear measurement of the movement of pistons

    - restrictors

- **Electrical Controls** (IEC Standard) → Power supply, Output components, Switches and Contact
- **EasyPort/OPC/DDE**

As already said, we used pneumatic pistons to simulate the movements of the boxes, we linked each box to the linear movement of the respective piston:

- The first piston simulates the first horizontal movement.
- The second piston simulates the second horizontal movement.
- The third piston simulates the third horizontal movement.

- The fourth piston simulates the vertical movement of the press.
- The fifth one simulates the fourth horizontal movement.

We used Programmable logic controller (PLC) to handle all the information regarding inputs and outputs and program actions related to a specific situation.

We connected the normally-open make switches to the sensors mounted on the pistons' rod: A0, A1, B0, B1, C0, C1, P0, P1, D0 and D1.
So, when one of the sensors is activated, an electrical signal comes to the respective PLC's port so that it can be handled in CODESYS to perform some function knowing where the signal is.
The solenoids are the devices that carry the signal to the valves, enabling them to commutate and manage the out-stroke or in-stroke of a cylinder. They also allow us to control the valve at the beginning of the circuit, allowing air to enter the circuit or blocking it in an emergency.
The displacement encoder takes data related to the position of the piston and so allows us to move the boxes in the simulated environment.

Once we have defined all the inputs and all the outputs, we passed these data to CODESYS to be able to manage all the system macro actions, routines, and subroutines.

CODESYS

POUs

In CODESYS, a POU (Program Organization Unit) is a core element used to organize and structure the program. It refers to individual programming units that include Programs, Functions, and Function Blocks.

We created six different POUs based on the requirements of our projects:

- Main PLC_PRG
- Action execution
- Cycle management
- Cylinder displacement calculation
- Emergency
- HMI visibility

Action execution

The action execution POU is essential to link a pre-defined phase in CODESYS to an action of a cylinder in FluidSIM. We did it in Ladder Code
We firstly defined the input variables that are all the phases, and the ports of the PLC in FluidSIM related to a solenoid.
In this way, to connect the phase1 to the port AB0.0 of the PLC that is connected to the solenoids conv1+ and so the phase1 will perform the out-stroke of the piston conveyor1.
We did it for all the phases, so:

Phase1	→ out-stroke of piston conveyor1
Phase1-back	→ in-stroke of piston conveyor1
Phase2	→ out-stroke of piston conveyor2
Phase2-back	→ in-stroke of piston conveyor2
Phase3	→ out-stroke of piston conveyor3
Phase3-back	→ in-stroke of piston conveyor3
Phase4	→ out-stroke of piston press
Phase4-back	→ in-stroke of piston press
Phase5	→ out-stroke of piston conveyor4
Phase5-back	→ in-stroke of piston conveyor4

Cycle management

This is a very important POU, because it is responsible for managing the performance of the loop cycles.
Here we defined the position and the timing of each phase, the pulse signal for the loading of the plates on the first conveyor, the press phase, and all the counters and timers that check if the storage area is full, if the shipping has been delivered. We used all the information related to the limit switch mounted on the pistons on FLuidSIM to better supervise all the operations.

In this case we coded the POU in Ladder Code.
At the beginning we have 6 absolute variables: *new cycle, full, shipping, finish, okay* and *reset*.
If the new cycle and reset are set the first network sets the variable okay and reset the variable reset, we will see later more details about these variables.
The second network starts effectively the loop, taking as input a variable ***n***, that is the number of the plates of the stack set by the operator in the HMI.
If n is greater than zero and the *start* is pressed either from the HMI or from the PLC, it is performed a check on the variables defined before, (so if okay and finish are set, if the piston conveyor1 is fully in-stroke and the limit switch is activate EB0.0, if the storage station is not full and the shipping phase is not set) then the phase1 is performed, the motor1 (associated to the first conveyor) is turned on and there is the reset of the variable *finish* (it will be set at the end of the cycle-stack so to know if the machine is ready to compute another cycle or not yet).
The third network is used to count the pieces that have been loaded on the conveyor by means of a counter, so when the variable n (the one defined by the operator) is equal to the quantity of pieces loaded on the conveyor we need to stop the loading, and in this case, we set a variable *piece*.
Moving on, we have the set of the sequential phases, only if the limit switches related to the in-stroke or out-stroke position of the corresponding piston is activated.
If we have phase1 and the limit switch at the end of the piston is energized, it means that the piston reached the fully out-stroke position, we can perform the phase1-back that takes back the piston to the in-stroke position.
The most important condition is here, the phase1-back sets the execution of phase2 but in parallel it also sets the phase1 again if the variable *piece* is not set.

Before, we said that the variable pieces would be set if the operator input was equal to how many pieces were loaded on the conveyor. If it wasn't, we need to call phase 1, and load as many pieces as needed.
At the same moment is called the phase2 and in the same way the phase2-back, phase3 and phase3-back.
The phase3 is the last phase related to the boxes on the first conveyor. So, putting a variable that calls the phase1 n times, we created a loop around the firsts 3 phases up to phase3-back, and the system can carry on only if the piece n has reached the phase3, that’s why we have a check with a counter also here.
The phase4 is the one related to the press, so a piston goes out-stroke also in simulation and stays pressing the stack for 3 seconds, indeed we put a timer of 3 seconds before setting the phase4-back to take back the piston.
The phase5 and phase5-back take the pressed stack from the press station to the stacking area, with the same mechanism of a piston.
Here we have another counter, that is essential to count how many pressed stacks have come to the stacking area, the variable max-stack is pre-defined.
From the requirements the stacking area can hold a maximum of 6 pressed stacks, after them, we need to ship the pallet.
So, recalling the phases, we have that from the phase1 to the phase5-back is performed the loop ***stack-cycle***, this loop can be performed at most 6 times before shipping the pallet and resetting everything.
So, to better organize the data we create the variable ***finish***, that is set at every ending loop of the stack-cycle, and it is reset at every start.
If the counter reaches the max-stack (6) of collected stacks, it sets the variable ***full***.
From this moment automatically it will set the variable shipping that, by means of animation and visibilities, it will show a shipment that will take 6 seconds, and after that it will show a delivered check.
Only from this moment we will have the possibility to click on **NEW-CYCLE**, that will reset all the variables, counter or timer, setting the variable reset, and it will allow us to compute a new stack-cycle from the beginning.

Cylinder displacement calculation

The cylinder displacement calculation is a POU written in Ladder Code, necessary to convert the linear position of the pistons in a linear function and then in a scale from 0 to 100 to better handle the animations.

The displacement encoder reads the value of the position of the piston. They send these data from the analog card of the PLC to CODESYS. By means of some simple formulas, we are able to have a linear scale of the position.
To then perform animations, as translation, in a better way.
We firstly convert the data from the analog card of the PLC to an integer number, then we use maths operators blocks to perform calculations, as a divider or subtractor and so on, to have the displacement as output.

Emergency

We created a dedicated POU to handle the most dangerous situations, the ones that, for example, need to stop the whole system instantaneously.
We defined three states: *normal-mode, stop-state, emergency-state*.
At the beginning of the cycle or whenever the *start* button is clicked, either from the HMI or from the PLC, the *stop-state* and *emergency-state* are set to FALSE or 0.
If the *emergency* button is clicked, either from the HMI or from the PLC, the *emergency-state* is set to TRUE or 1.
The same for *stop-state*, but with the stop button.
Here we handled these two states directly on the pneumatic circuit, placing two 3/2 valves with 1 stable position that regulate the source and the exhaust of air in the circuit.
In the *emergency-state* both of the valves are de-energized, in order to block the air both in input and output, since the emergency needs to stop the system instantaneously.
The *stop-state*, instead, blocks only the source of air in the circuit to let the exhaust open to unload the air, in this way the cycle will end in the next phase instead instantaneously.
We did it by sending commands directly to the PLC port where the solenoids of the valves are connected.

HMI visibility

Here we handled all the possible animations, in terms of translation and visibilities.
We coded this POU in Structured Text.
As for the other POUs we firstly set the input variables that were needed to activate a particular condition and then, we coded them in if-cycles.

We dealt with visibility of the three plates on the first conveyor, with the big box that contains the plates coming from the first conveyor, the storage area and the shipping and delivered station.

Main PLC_PRG

The MAIN POU is the most important POU of the whole project, because it takes all the information from the other POUs, and it organizes them to let them work together.
Firstly, we have that the *start* button, either from the HMI or from the PLC sets the variable ***run*** that, in its turn, opens the valves *source* and *exhaust* of the pneumatic circuit.
They also reset *stop* and *emergency*, while *stop* and *emergency* reset *run*.
The main POU is responsible for linking the input and output of *POU-blocks*, as in this case with the *HMI_visibility, cycle_management, action_execution, Emergency* and *cylinder_displacement*.
On the left we have the inputs that enter to the block of the POU and on the right the output that exit from it.

HMI - Human Machine Interface

As far as the Human-Machine Interface (HMI) is concerned, we built a simple panel composed of three sections from where we can control the whole system:

1. Input
2. Sensors
3. Modes

We used:

- Push-buttons
- Selectors
- Lamps
- Input number pad

Human-Machine Interface (HMI) is very important in order to make a connection between the programming side and animations.
Thanks to that we can have a representation, physically simulated, to what the system would do in the real world if everything was in these conditions.

INPUT

The first section is related to the input: at the beginning of each stack-cycle the operator is required to insert the number of pieces that the stack will be formed by.
Clicking on the box pieces a number pad appears, and the operator can insert a number (we set a minimum of 1 and a maximum of 10 to make the simulation more dynamic)
At the end of every stack-cycle the blue lamp will turn on indicating that the machine is ready to perform a new stack-cycle.

SENSORS

The yellow lamps indicate the status of the motors, they turn on to indicate that motors are moving.
The lamps are also present in correspondence of the motor on the plant.

MODES

According to the requirements the push-buttons are responsible to start the cycle, stop the cycle either in the first next available phase or immediately in case of emergency and start a new cycle after having shipped a complete pallet.
The lamps indicate the status of the corresponding scenario.

When a push-button is pressed it sends a signal to the variable associated with CODESYS.

- When START is pressed the motor 1 is activated and the conveyor 1 starts to move. At the same time, the phases related to the boxes are activated, and so we will see as many boxes as we set on the input appear on the first conveyor once at time and max 3 at the same time.
- When STOP is pressed the air in the pneumatics circuit is stopped and the cycle will end in the next available phase. There is an orange lamp indicating the stop scenario.
- The EMERGENCY selector is used in emergency scenarios to stop all the cycles instantaneously by means of a selector. We also have a lamp that allows us to check the emergency status (red lamp: emergency).
  The emergency is essential, and it has the highest priority. Whenever the selector is switched the whole system is stopped, all the conveyors are stopped, all the boxes are stopped. We can then activate the conveyors again and start the cycles with the boxes only if the switch is placed in the original position.
- NEW CYCLE is used when we have 6 stacks ready, shipped and delivered, and so we need to reset all the variables and make available a new cycle. It is available only if we have ended the previous cycle to 6 stacks.

Animations

To better visualize our plant, we built the whole system from scratch using the *visualization toolbox* of CODESYS.
We used *rectangles, ellipsoids, lines, counters, lamps, buttons* and so on.
We also added pictures to make the simulation more real, for example for the emergency area, for the HMI panel or for the shipping and delivery station.

To make a complete simulation we added animations, as for the boxes but also for the stoking, the shipping and the delivering stations.
We had timers and counters, and we mixed everything programming visibilities and linking variables.

For example, as far as the lamps are concerned, they are simply linked to variables of the MAIN POU. While the plates are linked both to the visibility variable to be visible only when it is necessary and to the displacement variable to be able to move and simulate the conveyor.

TESTS

Finally, we can perform some tests.

[VIDEO min. 21]

SINGLE CYCLE

For example, for a single stack-cycle, putting 4 plates as input.
So, by means of the numeric pad it is inserted 4, and since the blue lamp is turned on (the machine is ready) it is clicked the push-button START.
In this interface it is shown the PLC circuit on the right, pneumatic circuits on its left.
While in the most left side of the screen there is the simulated plant with all the variables associated to the cycles in the POUs.

[VIDEO min. 22]

COMPLETE CYCLE

Another test could be the test of a complete cycle, so performing 6 stack-cycles in order to test all the functionalities of the simulation.
In this case, it is inserted a different value for the plates of the stack at each cycle in order to check the cycle behaviour with different inputs.
The interface is the same, with the PLC circuit, the pneumatic circuits, the simulated plant and all the variables.
The first section of the variables is related to the phases, so we can have a check of the active phases during the simulation.
Firstly, there is the variable *n*, the input variable, that takes the value from the numeric pad, it changes at each cycle according to the choice of the operator.
As we can notice, the variable *stack-collected* starts from 0 incrementing by one at each *stack-cycle*. While the *stack-pieces* counts the plates coming from the first conveyor and arriving at the press station, the same as the *current-piece* variable, but counting the plates loaded on the first conveyor in the *phase1*.
Then there are all the variables related to the timers, so the one of the press phase and the one of the shipping station.
Moving on there is the variable *max-stack* is a constant pre-defined, according to the requirements it is 6.
Then the variables *piece, okay, finish, full, new-cycle* that allow the timing of the phases and starts of the cycles.

For example, we can see the variable piece become TRUE when the plates in the box under the press reach n, so we don't need any more plates. It allows for the continuance of the cycle with the press phase.
Or at the end of a stack-cycle, after putting the pressed stack in the stacking area, the variable *finish* becomes set to allow the start of a new cycle.
Full is set when we have 6 pressed stacks in the stacking area, as in this case, so the variable shipping becomes true, and the shipping-timers starts.
After the delivered check, we can click on NEW-CYCLE to reset all the variables and start a new pallet.

STOP & EMERGENCY

We can also perform a test on STOP and EMERGENCY scenarios.
As already said, we handled these two states directly on the pneumatic circuit, placing two 3/2 valves with 1 stable position that regulate the source and the exhaust of air in the circuit.

The *stop-state* blocks only the source of air in the circuit to let the exhaust open to unload the air, in this way the cycle will end in the next available phase.
There is an orange lamp indicating the stop scenario.

In the *emergency-state,* instead, using the selector, both valves are de-energized, in order to block the air both in input and output, since the emergency needs to stop the system instantaneously. Also here, there is a lamp that allows us to check the emergency status, the red lamp.

After one of them we can resume the cycle, clicking on START.