Report

• Introduction

• Theoretical background

• Instructions to run the script

• Wireshark captures all the supported communications modes

• Diagrams

• Conclusions

• Narrative to glue the previous components

****

**Industrial Informatics**

**Assignment 01**

by:

**Dong Le** - 151057054

**Carlos Gustavo Ramos Rodríguez** - 151890619

**Rune Lemey** - 151910450

**Date: 15.11.2023**

**Table of contents**

[Introduction 4](#_Toc129815314)

[Methodology 5](#_Toc129815315)

[Theoretical Background 5](#_Toc129815316)

[Instructions 7](#_Toc129815317)

[Results & Diagrams 9](#_Toc129815318)

[General 9](#_Toc129815319)

[Read Coils 10](#_Toc129815320)

[Read Discrete Inputs 11](#_Toc129815321)

[Read Holding Registers 13](#_Toc129815322)

[Read Input Registers 14](#_Toc129815323)

[Write Single Coil 16](#_Toc129815324)

[Write Single Register 17](#_Toc129815325)

[Write Multiple Coils 19](#_Toc129815326)

[Write Multiple Registers 20](#_Toc129815327)

[Conclusion 23](#_Toc129815328)

[References 24](#_Toc129815329)

[Appendices 25](#_Toc129815330)

# Introduction

This assignment focusses on orchestrating industrial processes using a systematic approach, where the focal points include the utilization of REST API, Flask framework, JSON data exchange, Object-Oriented programming, and Postman for API testing. The objectives of this assignment are not only to implement a technical solution to an open problem but also to provide insights into the intricate design phase, emphasizing the significance of UML diagrams in the process.

As we navigate through this assignment, we understand more about controlling industrial processes and asynchronous systems. Additionally, our application involves visualizing and comprehending the role of Web Services in the integration of automation systems. Through practical applications of Object-Oriented Programming, we will not only tackle orchestration tasks efficiently but also gain valuable hands-on experience in using Python for industrial automation challenges.

# Methodology

## Even-Driven Programming

The application was used event-based method, which is a paradigm where the flow of the program is determined by events. Some of benefits of using event-driven system are:

* Asynchronous processing: it allows the system to handle multiple tasks simultaneously.
* Scalability: Compared to synchronous processing, event-based system is more scalable, it can handle large numbers of events.
* Modularity: each component can focus on specific tasks and communicate via events, so it makes the program easy to maintain.

In this application, the group applied this method to implement solutions. To be specific, the orchestrator program handled the flow by process event triggering such as user input, zone changes, robot events.

By planning the design phase carefully, the schema for events is clear. It supports the coding process effectively, and ensures all scenarios are covered.

## Web UI

By using Flask and html, a web was build, it helps user submit order, subscribe events, and keep track of orders.

A screenshot of a survey

Description automatically generated

Figure 1: Web UI

## Instructions

To manage to operate the station, a list of actions user needs to follow.

Action list for the start:

* Subscribe to the conveyor and robot.
* Calibrate the robot.
* Submit orders.

Action list for workstation 1:

* Place pallet.

Action list for workstation 5:

* Remove the pallet.

Action list for the end:

* Unsubscribe.
* Shutdown station.

# Diagrams

## Flow Diagrams

The following diagrams shows the flow of program when events trigger.

The rectangle figures are events, this can either be that the state of a zone has changed or that a pallet transfers to a different zone. Another figure shown is the diamond shaped figure. These are questions that need to be asked. This can have 2 outcomes, if the answer to the question is yes then the arrow will have green color. When the answer to the question is no, then the arrow will be red. The last figure that is the circle. This means that nothing needs to be done in the program.

**When user submit order on Web UI:**

A diagram of a pallet

Description automatically generated

Figure 2: user input flowchart

**When Event Z1\_Changed trigger**

**A diagram of a process

Description automatically generated**

Figure 3: Z1\_Changed flowchart.

**When Event Z2\_Changed trigger**

A diagram of a pallet

Description automatically generated

Figure 4: Z2\_Changed flowchart.

**When Event Z3\_Changed trigger**

A diagram of a pallet

Description automatically generated

Figure 5: Z3\_Changed flowchart.

**When Event Z4\_Changed trigger:**

A diagram of a process flow

Description automatically generated

Figure 6: Z4\_Changed flowchart.

**When Event Z5\_Changed trigger:**

A diagram of a pallet

Description automatically generated

Figure 7: Z5\_Changed flowchart

**When Event DrawEndExecution trigger:**

A diagram of a task

Description automatically generated

Figure 8: DrawEndExecution flowchart

## Use Case Diagram

Figure 9 displays the Case Diagram, here to main this happens. First, the operator needs to select/submit an order for the pallet, when this has happened the order will be displayed. The operator also needs to place a pallet, this will trigger a signal that the state of Z1 has changed.

The second user interaction is when the pallet arrives at Z5, and thus the state of zone 5 changed. That the pallet then needs to be removed by the operator.

A diagram of a process

Description automatically generated

Figure 9: Use Case Diagram

## Class Diagram

The Class diagram shows the application has 3 different classes. First there's the FlaskApp this stays for the code main code that invokes the class Orchestrator to work, meaning that FlaskApp depends on the Orchestrator. The Orchestrator is the main class on the program, this has a Composition relationship with Pallet since this program uses the values from the other class.

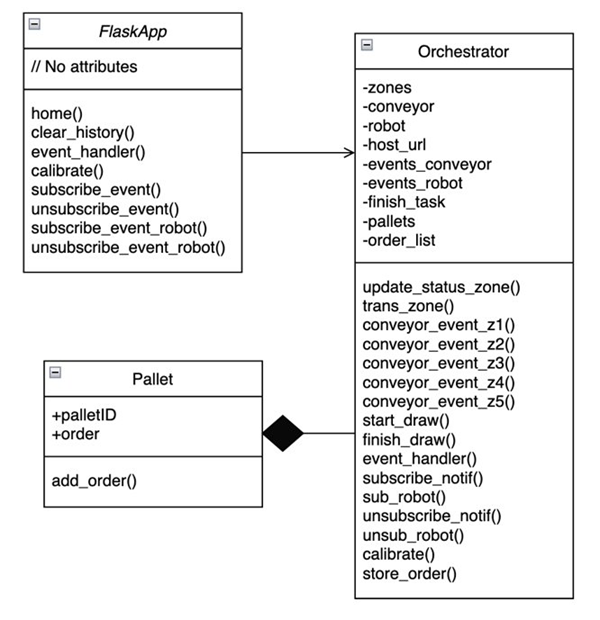


Figure 10: Class Diagram

# Weakness and Future Improvements

The biggest reliability problem that the system can have, is component failure such as sensors do not detect pallets, robots not working, unfunctional conveyors. When this happens, the whole production will be postponed till the problem is fixed. This can be partly solved by applying a fault and error section in the code. For example, when a pallet takes too much time to transfer than it should be, or the robot stop without draw something for a set time, then an alarm will go on. When the alarm is activated, a person will be notified and can come to the system to solve the problem.

Another weakness is that the application only receives one and draw one order. A solution for this is to compare the event DrawEndExecution with the number of drawings in ordered set. If the number of events is not equal to order drawings, then robot keeps draw the next draw in the order set, otherwise, it finish the order.

# Conclusion

In the implementation of the assignment, we encountered some challenges in logic. However, thanks to the design phase, we made the flow charts diagrams to ensure all scenarios are covered. After that the programming follows the define design, we understanded the usefulness of design phase, we enjoyed the process of working based on diagrams and found it rewarding in the end.

The advantages of Object-Oriented programming are undeniable, the application becomes nicer and more clean, easy to maintain. This assignment has only five zones/pallets, but if there are more pallets, the program is still manageable with OOP and easy to expand vertically.

One challenge we faced was using Flask, html, and Web Services. We do not have experience not only in web software application, but also using Flask. However, we found several tutorials online and they are easy to follow. Even though our web UI is not excellent, we enjoyed making it and be proud with the result as beginners. Moreover, thanks to Flask so the application can handle asynchronous operation with delay.

Overall, the implementation of the first assignment was an excellent opportunity for us to learn about the RESTful web services. We found that working with RESTful API required a deeper understanding of methods, python, Flask, and json, which was a valuable learning experience. We believe that the exercise has enhanced our knowledge and support us for future works, courses.

# References

We followed your lectures and materials :D

Gracias Señor!

# Appendices

Github: