Testbed for virtual commissioning of PLC system

Jakub Nosek, and Jakub Arm

Brno University of Technology, Czech Republic

E-mail: 221007@vut.cz, jakub.arm@vut.cz

Abstract— The work deals with the creation of a testbed for the virtual commissioning of a PLC system. The testbed consists of Unity, a virtual PLC, a Rest API server and a C # application. The production process is simulated at Unity using a Siemens virtual PLC. The data is sent from Unity in JSON format to the Rest API server, where it is read by a C # application and the program is modified if necessary. After editing, the simulation restarts. The testbed is intended to optimize the PLC application, i.e.crisis treatment, collision avoidance and line tact reduction.

Keywords—PLC, C#, Unity, JSON, Simulation

1. INTRODUCTION

The testbed containing a virtual commissioning PLC system is to be used for automated testing and even debugging of PLC applications. The data for PLC program repairs are taken from a dynamic simulation of real technology, which is connected to a virtual PLC. At this point, we assume that we will optimize the following aspects of the PLC program:

- timer values
- collision treatment of technological parts
- line cycle time.

The work focuses on the reworking of the project from the subject BPC-PPA by Dr. Arm. The original intention was to program a virtual PLC using Codesys software, because it is free and contains a virtual PLC. The aim of the simulation was to use a robot to place six boxes on the main conveyor, where the boxes were filled with cans and then placed on a shelf according to the student's ID. The project was available to students as an exe file.

In [1], the process of optimizing the production process based on video analysis is presented. In our case, however, we proceed by first creating a credible virtual model of the production process (the so-called passive digital twin) and modifying the PLC application according to the operation in the virtual environment.

Mathematical technology models can be used to verify the correctness of the PLC application. However, this approach requires ensuring the credibility of such a model. In addition, it also suffers from problems arising from nondeterministic communication between the virtual PLC and the model, as stated [2]. Our approach uses a dynamic physical model, which also takes into account the physical properties of objects (mass, friction, inertia).

2. TESTBED DESIGN

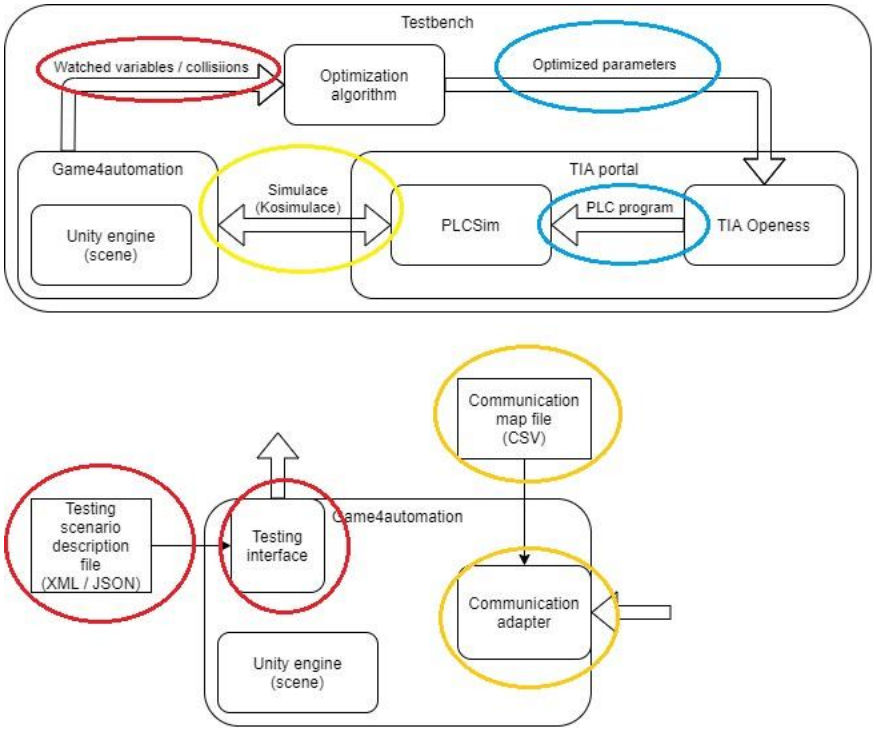

Figure 1: Schema of testbed

The task of the test interface will be to transfer the monitored parameters to the newly created application, where the data will be processed and then applied to the PLC.

AutomationML will be used to describe the testing scenario, according to which the individual industrial components, their behavior and interconnections will be described.

The Rest API server working with the JSON data format was chosen for the communication of the test application with the simulation in Unity for the purpose of data sharing. The principle of data collection works by storing the sample time, all inputs and outputs between the PLC and the simulation, the number of collisions and information about all collisions in the given frame (collision position and names of collided objects). This data is saved as a new element in the List. After the selected time in seconds, data is uploaded to the API server, which contains information that a new, possible restart of the simulation from the test application, the number of elements in the List and the List itself, which has the previously mentioned information about individual frames.

The task of the C # application is to process data from the simulation in Unity and subsequently rework the program and its parameters using an optimization algorithm. Then it starts the simulation again with the new parameters.

According to tests, Modbus TCP will be used as a communication adapter, for better response. In case of problems, it will be a backup variant S7 with a real PLC.

3. IMPLEMENTATION

Codesys software has been replaced by TIA Portal V16. Codesys SoftPLC communicated with Unity via Modbus TCP at IP address 127.0.0.1, which is the computer's local address on localhost. When programming in the TIA Portal, it was not possible to set the virtual PLC in the PLCSIM V16 software to the IP address 127.0.0.1, because Siemens does not allow this address, as Codesys. In general, it is not possible for the basic PLCSIM V16 program to communicate with the Modbus TCP communication protocol because it does not have the required virtual adapter. Therefore, PLCSIM Advanced, which includes this feature, was used. A project with a virtual PLC SIMATIC S7 / 1512C-1 PN was created for this solution in the TIA Portal. Subsequently, the IP address in Unity changed. Unfortunately, this was not enough and the code for Modbus TCP had to be modified. The original version of Modbus TCP read from the Holding registrers and the Input registers. Codesys can distinguish this, but TIA Portal cannot and uses everything as Holding registers. The Modbus TCP code has changed that Unity will

process everything as Holding registers, only the first 10 Words will be taken as Unity output and the next 10 Words as Unity input.

Two communication protocols were considered for the testbed. With virtual PLC Modbus TCP and with real PLC S7. The response was measured in Unity for both communication protocols. The code was modified in the script to save the results to a text file. A string variable was created in the InterfaceThreadedBaseClass script, which stores the response time from the CommCycleMs variable as a new line each time the program iterates. This recording takes place only after the tenth iteration due to the fact that the first measured values may not be valid due to the loading of the environment. The measurement starts at the beginning of the simulation. In 30,000 iterations, the measurement results are saved in a text file. As a result, there are 29,899 stored measurement results. At the end of the text file, the minimum and maximum of the measured values, the average value and the variance are found. The minimum and maximum values found were found during the program run. The individual measurement values in each iteration were stored in the temporary field, and the variance and average value were calculated in the last iteration. For Modbus TCP on the virtual PLC, the minimum value was 1 ms, the maximum value was 12 ms, the arithmetic mean was 1,91 ms, the variance was 0,39 ms, and the measurement uncertainty was 0,007206 ms. For S7 on the real PLC, the minimum value was 50 ms, the maximum value was 143 ms, the arithmetic mean was 98,57 ms, the variance was 9,42 ms, and the measurement uncertainty was 0,035548 ms.

4. WORK PROGRESS

The entire program in Unity is currently being redesigned so that it can share data in a suitable format with the Rest API server and work with Siemens PLCs via Modbus TCP. A PLC program with HMI visualization was created for the simulation. The Rest API server was created to have the same data structure as the simulation. The main goal of the Rest API server is that after receiving new data, it stores it in a global variable until it is overwritten again by simulation. The C # application is currently being worked on, but it is already communicating with the Rest API server and collecting data. At the same time, PLC communication with simulation in Unity was measured.

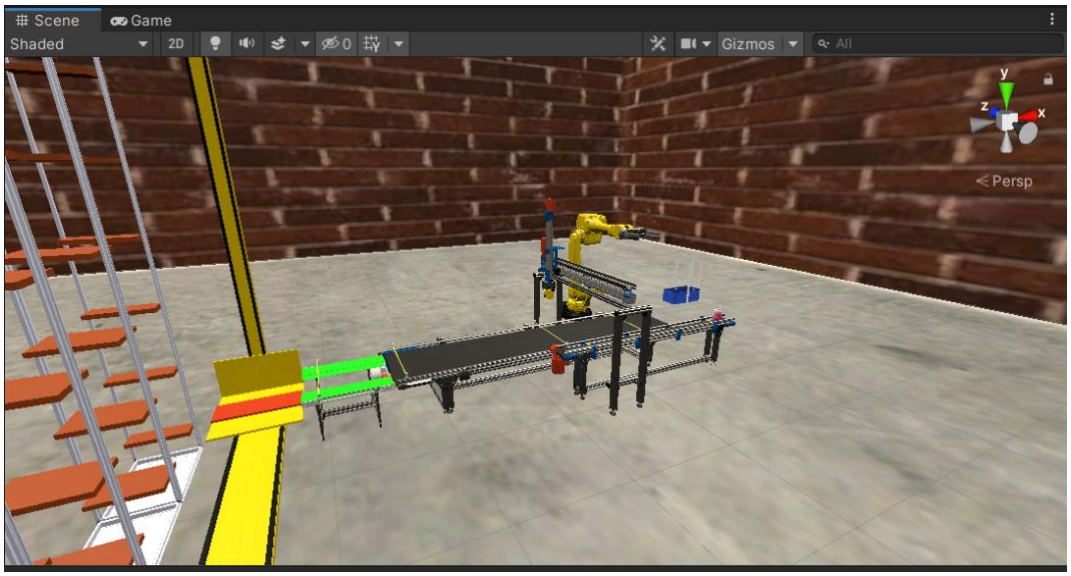

Figure 2: Simulation in Unity

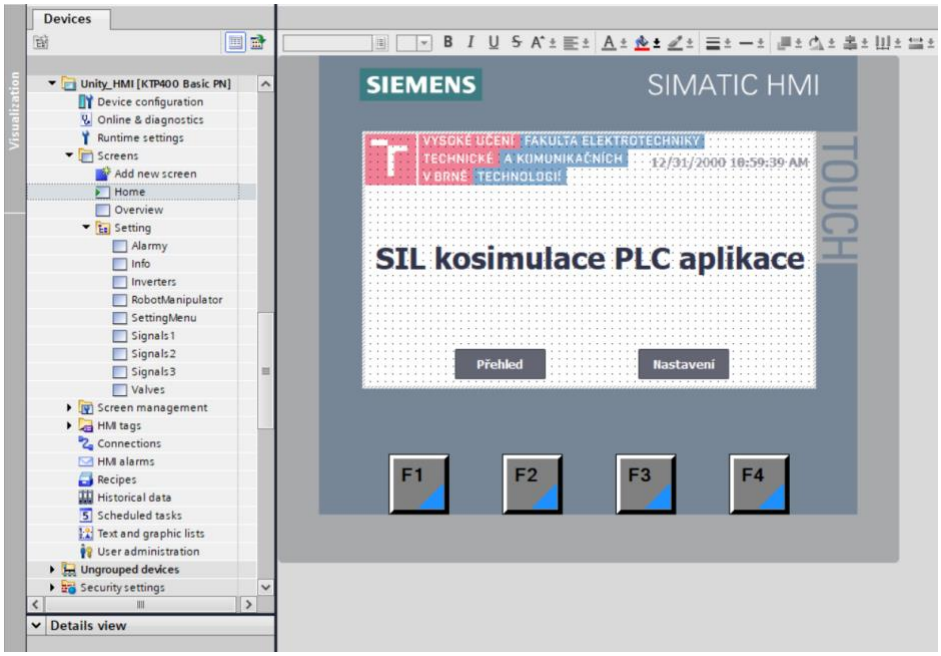

Figure 3: HMI Main screen

5. CONCLUSION

A conclusion was drawn from the measurement results of the communication protocols. For Modbus TCP, the minimum value was 1 ms, the maximum value was 12 ms, and the arithmetic mean was 1,91 ms. For S7, the minimum value was 50 ms, the maximum value was 143 ms, the arithmetic mean was 98,57 ms, the variance was 9,42 ms, and the measurement uncertainty was 0,035548 ms. It follows that Modbus TCP with a virtual PLC is clearly the fastest. It is clear that a real PLC that uses its own processor can never compare to the response speed of a virtual PLC on the same device where the simulation is taking place. It is necessary to realize that communication is not real-time, so in the simulation there is always a small delay.

I have currently completed one of the last important tasks, which is the communication between the simulation in Unity and the test application. I have successfully tested this and will continue to make minor improvements. Even though this is not the main goal of the work, I will try to create at least a basic optimization algorithm.

ACKNOWLEDGMENT

This work was supported by Brno University of Technology and was carried out with the support of the project FEKT-S-20-6205 Research in automation, cybernetics and artificial intelligence for Industry 4.0.

REFERENCES

[1] R. Beloiu, "Virtual Commissioning of Wheel Robot Processing," *2021 12th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*, 2021, pp. 1-4, doi: 10.1109/ATEE52255.2021.9425077.

[2] H. Carlsson, B. Svensson, F. Danielsson and B. Lennartson, "Methods for Reliable Simulation-Based PLC Code Verification," in *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 267-278, May 2012, doi: 10.1109/TII.2011.2182653.