Project final report

Igus robot arm industrial project

Rik van Heesewijk, Luuk Aarts, Dimitar Dyulgerov, Jorn Kersten
6-15-2023

Table of Contents

Introduction 2
Research 2
Visualization research 2
Unity digital twinning Communication software 2
Findings TwinCat ..... **Error! Bookmark not defined.**
CANopen research 2
Analysis 3
Requirements 3
Design 3
Tests 3

Introduction

In this file you will find a brief description about our findings and what we did. After each section of text, you will find a link to the appropriate file in our project. In such a file you will find further information about what we investigated or recorded.
You start with a piece about our research, in it you will find all kinds of different research on different topics. Then comes the analysis section, in which you will find briefly what we did to record the assignment and find out what we needed to do for this project. After that comes the requirements chapter, where we briefly explain the requirements and how we arrived at them. The requirements chapter is followed by the design document, which contains further information about how code communicates with each other and how classes communicate with each other. Finally, there is the tests chapter, where we briefly talk about the tests we did.

Research

In the following sections you will find what we researched and for what purpose. You will also find the result found and how we got to it.

Visualization research

We had to use a tool to create a visualization of a conveyor belt. Besides having to work visually, it also had to be able to communicate with TwinCAT on a Beckhoff PLC. In addition, the program had to be simple enough so that anyone could get along with it and install it easily.
Therefore, we created four different research questions:

- Can this tool communicate with the Beckhoff plc?
- Can this tool be used on bad PCs?
- Is the tool easy to learn?
- Is the tool easy to use?

In the end, I researched these four questions for three different visualization tools: Unity, Unreal Engine and Simatic WinCC. After researching these questions, Simatic WinCC fell by the wayside; there wasn't much to find about it, nor was it clearly documented. So the choice was between Unity and Unreal Engine, with Unreal Engine having some lesser features and in addition I had already worked with Unity. Therefore, I finally chose Unity. If you want to read more about this research, go to the Unity research file.

Unity digital twinning Communication software

To communicate between the visualization software and the PLC we had to a protocol that worked. We researched what protocols where available and what the difference between them where. We finally went with Ads because it was the easiest to use and fit all of our needs perfectly well. These findings can be found in the Unity digital twinning Communication software document.

PLC software research

TwinCAT 3 is used to program the Beckhoff PLC. Our general findings about TwinCAT 3, how it can be used and if there is any other usable software, is/are described in the Findings_TwinCat document. The gist of the source code and implementation details are described in the TwinCAT 3 source code documentation document.

CANopen research

One of the initial project requirements was using CANopen to communicate with the robot arm. Due to encountering unexpected issues, described in Ethernet or CANopen and TwinCAT 3 source code

documentation Therefore, as can be concluded from the aforementioned document, CANopen was abandoned in favour of a socket client/server communication.

Robot communication research

Since our main goal is to pick up virtual packages while by moving the robot, we had to somehow move the robot from one position to the next. Therefor we had to know how to control the robot by using the Igus communication protocol. By creating some simple research questions, we got to easily search for each smaller obstacle so we would eventually have an clear answer for the main question at hand.

- How to connect with the Igus robot?
- What is the message format for the Igus robot?
- What is the Igus robot communication sequence?
- How to make the Igus robot move?

These questions helped us understand and get to know the communication protocol created for the Igus robot arm. We found that some of the publicly stated options for moving the arm did not work so we searched for some different options. While testing we found a solution to finally move the robot as we wanted. All detailed research documentation can be found in the Findings_IgusRobotCommunication file.

Analysis

To find out exactly what our assignment was, we asked the client about what we needed to do and make. From this came some clear answers.
We had to control a robot arm from a Beckhoff PLC, using CANopen. There also had to be a visualization of a conveyor belt, which could then control the robot arm via the Beckhoff PLC to "pick up" cubes on the conveyor belt. The visualization must then send a location of a cube on the conveyor belt to the PLC, then it causes the robot arm to move to that position.
If you want to read more about our analysis, read the Project plan.

Requirements

Based on the interviews and the analysis file, we created the requirements for this project. The requirements were fairly easy to draft. There had to be a visualization of a conveyor belt, it had to pass coordinates of a cube on the belt to a Beckhoff PLC. This then had to send the coordinates to a robot arm which would then go to that coordinate and "move" the cube.
If you want to see the requirements more clearly, read the Project Requirements.

Design

We had to design a number of different solutions; we described them in the Design document. This contains what we did to arrive at the design, how it works, and what it is for. In designing and coming up with these diagrams, we learned quite a lot.
If you want to know more about the different diagrams, read on in the Design document.

Tests

To see if everything works well with each other, we have set up a number of tests. With these tests we want to ensure that all components both stand-alone and linked together are performing the correct tasks and using the correct data. We first tested everything separately, then we linked everything together to find out if everything is working as expected. We then wrote down any special

observations and fixed them so that they work. If the problems were too big or irrelevant, we did nothing with them.
If you want to see the tests a little more in depth, read on in the Test document.