

Bachelor's thesis (AMK)

Machine- and Production Technologies

Machine Automation

2015

Teemu Kuuskoski

# GAMIFYING PLC EDUCATION



TURUN AMMATTIKORKEAKOULU  
TURKU UNIVERSITY OF APPLIED SCIENCES

Author Teemu Kuuskoski

## GAMIFYING PLC EDUCATION

The purpose of a engineering thesis is to a design simple logic guided automation system for the Turku University of Applied Sciences for education purposes. The system was executed as a simulation with Visual Components 3D Automate platform. The control system was Beckhoff TwinCAT 3 logic control. The connecting interface between the program was a 3D Automate PLC add-on.

The other part of the thesis is to consider how, gamification methods can be used to help the students to learn simulation and logic programming more efficiently.

The progress of the project is to first make a layout of a production line with Visual Components, where defective products are removed from the main conveyor with a pneumatic pusher. Then design TwinCAT 3 logics to control this pneumatics pusher and make sure that the right products are terminated. In the last part, two programs are connected and check that the project result is as intended.

At the end of the thesis, the project conclusions are introduced and the challenges of the thesis are observed, such as how did the use of the programs go along and was the project successful.

### KEYWORDS:

Simulation, Visual Components, logic program, TwinCAT 3, gamification, pneumatic pusher, linear conveyor

Teemu Kuuskoski

## LOGIIKKAOHJELMOINNIN OPETUKSEN PELILLISTÄMINEN

Opinnäytetyön tarkoituksena oli suunnitella yksinkertainen logiikkaohjattu automaatiojärjestelmä Turun ammattikorkeakoululle opetuskäyttöön. Järjestelmä toteutettiin simulaationa Visual Components 3D Automate -alustalla. Ohjausjärjestelmänä käytettiin Beckhoff TwinCAT -3-logiikkaohjausta. Rajapintana ohjelmien välillä toimi 3D Automate PLC add-on -lisäosa. Opinnäytetyön toisessa osassa tavoitteena on selvittää, miten pelillistämismenetelmiä voisi käyttää tukemaan oppilaiden oppimista simulaatiossa sekä logiikka ohjelmoinnissa.

Työ aloitettiin ensin suunnittelemalla Visual Componentsilla automaattilinjasto, jossa vialliset tuotteet poistettiin pneumaattisen työntäjän kanssa. Seuraavaksi suunniteltiin logiikka TwinCAT 3 -ohjelmalla, joka ohjaa pneumaattista työntäjää. Viimeisessä osassa nämä kaksi ohjelmaa yhdistettiin ja varmistettiin, että ohjelmat toimivat odotetulla tavalla.

Lopussa esitellään projektista johdettavat päätelmät sekä tarkastellaan opinnäytetyön edetessä esiintyneitä haasteita, kuten miten ohjelmien käyttö sujui sekä oliko projekti onnistunut.

### ASIASANAT:

Simulaatio, Visual Components, TwinCAT 3, pelillistäminen, automaatti linjasto

# CONTENT

<b>LIST OF ABBREVIATIONS (OR) SYMBOLS</b>	<b>8</b>
<b>1 INTRODUCTION</b>	<b>6</b>
1.1 Objective of my thesis	6
1.2 Structure of the thesis	6
<b>2 GAMIFICATION</b>	<b>7</b>
2.1 The principle	7
2.2 History of gamification	7
2.3 Gamification around us	8
2.3.1 Recycle Bank	9
2.3.2 Khan Academy	9
2.3.3 Memrise	10
2.4 Why do we play games?	11
2.5 Benefits of gamification	12
2.5.1 Motivation	12
2.5.2 Flow	12
2.5.3 Feedback	13
2.5.4 Enjoyment	14
2.5.5 Competition	15
2.6 Using gamification in the project	15
2.6.1 Points	15
2.6.2 Levels	16
2.6.3 Badges	16
2.6.4 Leaderboards	17
<b>3 THE TOOLS</b>	<b>18</b>
3.1 Simulation	18
3.1.1 Advantages and disadvantages of simulation	18
3.2 Visual Components	19

3.2.1 Application	20
3.2.2 Building simulations	21
3.2.3 Programming simulations	22
3.2.4 Interfaces with other programs	23
3.2.5 Usability	23
3.3 Programmable Logic Controllers	24
3.3.1 Differences with PLC and Computers	24
3.4 TwinCAT 3	24
3.4.1 Licences	24
3.4.2 Structured text (ST)	25
3.4.3 Data types	27
3.4.4 Function Block Diagram (FBD)	28
3.4.5 Function blocks	29
3.4.6 Ladder Logic Diagram (LD)	30
<b>4 THE MAIN PROJECT</b>	<b>32</b>
4.1 Visual Components Simulation	32
4.1.1 Adding components to the 3D world	32
4.1.2 Connecting the components together	38
4.2 TwinCAT logic simulation	43
4.3 Connection between Visual Components and TwinCAT	54
4.3.1 Adding PLC_IN and PLC_OUT signals	54
4.3.2 Adding signal interface	56
4.3.3 Modifying the Python code for the Pusher Conveyor	58
4.4 Connection between Visual Components and TwinCAT	61
4.4.1 Activating configuration	61
4.4.2 Adding connection part inside 3D create	62
4.5 Testing the project	66
4.5.1 Walkthrough of the PLC program	66
4.5.2 The RESET function	69
<b>5 RESULTS AND REFLECTIONS</b>	<b>70</b>
<b>SOURCE MATERIAL</b>	<b>72</b>
<b>PICTURE SOURCES</b>	<b>74</b>

## PICTURES

Picture 1: Example of Memrize memorization photo.	10
Picture 2: Example of the grading in Memrise.	11
Picture 3: Pacman videogame.	11
Picture 4: Almost everybody knows Pong.	11
Picture 5: Example of a points meter.	16
Picture 6: Example set of badges.	17
Picture 7: Basic pick and place automatic conveyor line with Panasonic robot simulated by Visual Components.	20
Picture 8: Basic starting view of Visual Components.	21
Picture 9: Differences between Java language and Python language.	22
Picture 10: RSL in Visual Components.	23
Picture 11: ST example (PLC Academy 2015).	25
Picture 12: Example from the main project.	26
Picture 13: Function Block Diagram Example (Beckhoff 2016).	28
Picture 14: A general function of a function block.	29
Picture 15: Timer on-delay (TON) function block(Beckhoff 2016).	29
Picture 16: A count up function block (Beckhoff 2016).	30
Picture 17: Ladder diagram has a structure of an electric circuit.	30
Picture 18: Finding components in eCAT.	33
Picture 19: Adding the Advanced Feeder to the 3D world origin.	34
Picture 20: Final result of changing the view.	35
Picture 21: Chosen product active with red lines.	36
Picture 22: Making sure that the Conveyor was copied.	37
Picture 23: Final result after adding all the components.	38
Picture 24: Green line for connecting components.	38
Picture 25: Connected conveyor line.	39
Picture 26: Pusher Conveyor working without logics.	40
Picture 27: Finding where to change the distribution for the products.	41
Picture 28: Example Distribution Book.	41
Picture 29: The changes to the variables.	42
Picture 30: Vcid code for getting right colors and components.	42
Picture 31: Changing the component 1 and 2 search id for the eCat.	43
Picture 32: Opening new project.	43
Picture 33: PLC program open.	44
Picture 34: Deleting old existing MAIN (PRG).	45
Picture 35: Adding Global Variable List.	46
Picture 36: PLC_out and PLC_IN global variables added.	46
Picture 37: Adding MAIN (PRG) with LD.	47
Picture 38: MAIN (PRG) variables added.	47
Picture 39: Adding FB_Cylinder function block.	48
Picture 40: Variables for the FB_Cylinder.	49
Picture 41: Adding an empty function block.	50
Picture 42: Adding the variables to the function block of the MAIN (PRG).	51
Picture 43: TON function block added with variables.	52
Picture 44: Variables added to the CTU function block.	53
Picture 45: Adding RESET to the CTU function block.	53
Picture 46: Adding RESET input and RESET variable to the MAIN (PRG).	54
Picture 47: Adding PLC_OUT and PLC_IN signal.	55
Picture 48: Connecting the two signals to the PushScript	56
Picture 49: Finding Signal Interface.	57

Picture 50: Signal Interface changes.	58
Picture 51: Finding the PushScript of the PusherConveyor.	59
Picture 52: PLC_IN and PLC_OUT variables added to the Python code.	60
Picture 53: Python program added.	61
Picture 54: Activating configuration on the TwinCAT.	62
Picture 55: Adding TwinCAT connection to 3D Create.	63
Picture 56: Adding main file in PLC add-on.	63
Picture 57: Pairing the 3D Create signals with the PLC signals.	64
Picture 58: The connect button.	65
Picture 59: Changing simulation mode from virtual to real time.	66
Picture 60: The Global Variables should flicker to TRUE.	67
Picture 61: Signal run through in MAIN (PRG).	67
Picture 62: The run through of the signal in TON function block.	68
Picture 63: DutyCycleCounterCTU function block run through.	68
Picture 64: The final outcome.	69
Picture 65: The RESET signal changes to "TRUE".	70

## FIGURES

Figure 1. Google trend for search of the work of gamification	7
Figure 2: Gaming age groups division	14

## **LIST OF ABBREVIATIONS (OR) SYMBOLS**

PLC	Programmable logic controller
I/O	Input/Output
CAD	Computer Aided Design
QR code	Quick Response code
D.I.C.E	Design, Innovate, Communicate, Entertain
RSL	Robot Sequence Language
Vcid	Virtual Components Identification
FBD	Function Block Diagram
POU	Program Organization Unit
LD	Ladder Logic Diagram
Var	Variable
ST	Structured text
TON	Timer on-delay
CTU	Count up
CV	Current counter Value



# 1 INTRODUCTION

This thesis is made for the Turku University of Applied Sciences and the subject is about researching how to do a certain school assignment easier to understand and more entertaining with gamification.

## 1.1 Objective of my thesis

Objective of this thesis is to make an easy to understand guide to teaching Programmable Logic Controllers (PLC) for new students. Gamification methods are used to make the learning fun and interesting. PLC is a challenging subject for students because its not particularly visually pleasing. Also before it wasn't possible to practice at home before, since the PLC program was a commercial program but now with the new version it has been made free to use. Before the user needed a licence for the PLC program, but now it has been made free to use with one week licence that can be continued indefinitely.

## 1.2 Structure of the thesis

The thesis will first explain in the next chapter what is gamification and how to use it as a teaching method. From that, I will describe the two computer programs that I will be using: Visual Components 3D Create for designing an automatic conveyor line and TwinCAT 3 for designing the PLC. The 4th chapter will be about making the conveyor simulation and the PLC program. The main point of this chapter is how to make them connected in a way, that the PLC is commanding the automatic conveyor. The last chapter is about how did I achieve my goals and how effective the method was.

## 2 GAMIFICATION

### 2.1 The principle

“Gamification is used to change appropriate, important, and serious work matter in various ways that are more attractive and easier to comprehend. In the center of gamification is a presumption that points, small rewards and achievements will encourage and motivate to make certain work task” (Ängeslevä 2014, 52). In other words, gamification means using game design elements in a non-game context to achieve some kind of accomplishment. To understand the impact of the gamification is now and in the the future Jesse Schell a CEO of Schell games had a speech in 2010 D.I.C.E summit that gathers around video game executives to talk about trends and innovations in gaming industry (Jesse Schell 2010). The speech gives the listener a good understanding that the gamification method usage is limitless.

### 2.2 History of gamification

Gamification is a quite new concept even though gaming has been a part of our free time for over 30 years. The gamification term was first introduced in 2003 by Nick Pelling (Pelling 2015) who was a computer programmer and an inventor but the term didn't come popular until 2010.

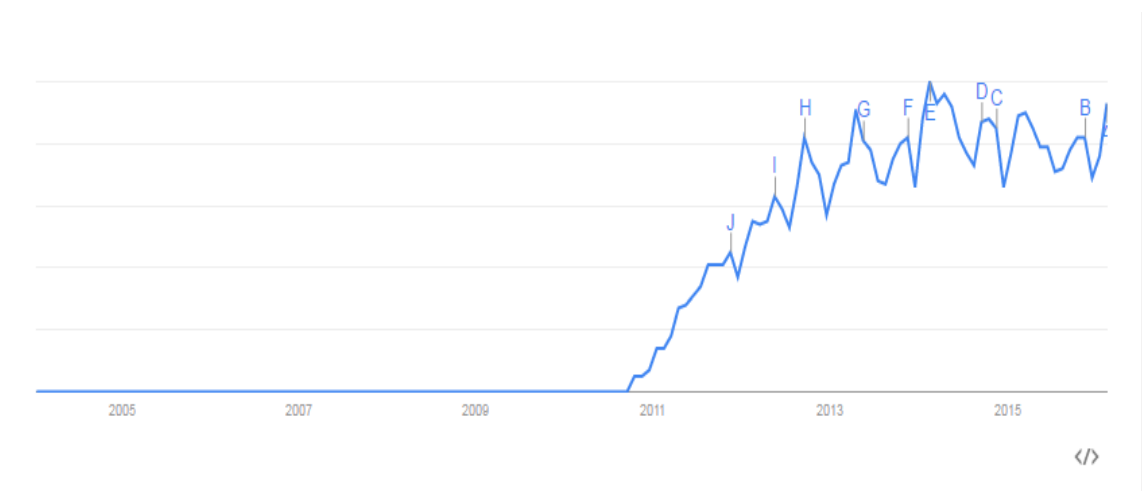


Figure 1. Google trend for search of the work of gamification.

In Figure 1. Can be seen that the interest for gamification has risen greatly after the year 2010 as shown with Google trend that shows how much certain word has been searched from Google.

### 2.3 Gamification around us

People with smartphones in their hands can be seen everywhere, using their phones to play some Candy Crush Saga or maybe defending their bases in Clash of Clans. They challenge other friends with game requests, after they got a new highscore in a new addictive game or perhaps work together to build a stunning new house in Minecraft through the internet. People like to distance them sometimes from reality and games help with that.

The interest in health has also increased now, because of the bracelets that tell you when to eat and do a workout efficiently. It gives the user the feel of accomplishment by checking the workout schedule from the internet calendar when the bracelet automatically syncs the information via Wi-Fi. The interest in fitness apps has also increased and one of the most popular ones is Fitocracy. The app gives you tools to plan your workout as alone or in a group. Fitocracy motivates people with achievements for doing workout tasks, leveling up which gives the person bragging rights, quests which give the user may tasks to do in a week and challenges and duels where the users have contest against one person (duel) or large groups are competing against each other to be the best (Physicaltherapyweb, 2016)

Last year's Turku exhibition (Turun Messut) had an IT company giving a chance to win prizes if you could find all the QR codes around the exhibition halls. The treasure hunt did make the contest much more interesting for people compared with the traditional way of putting name on the raffle ticket.

For Medicine students, you can find a game that helps you to memorize bones in the human body. Whack-a-bone game gives you a certain body part where you have to first add the bones to the right places, e.g. in your arm. In the end, you have hit the bones on right order that the program gives and be careful not to lose your stamina meter which, if depleted is a game over ([www.anatomyarcade.com](http://www.anatomyarcade.com) 2016).

### 2.3.1 Recycle Bank

The finalist of Gamification World conference for the best gamification project. There the objective is to popularize recycling by giving points which can be redeemed for discounts on over 4000 reward partners. They have made this possible by making partnerships through communities, waste haulers, local businesses and the national brands. You will achieve points by going through levels of questions and answers, which give you an insight of a greener living practices and vow to follow them (Recycle Bank 2016).

### 2.3.2 Khan Academy

Khan Academy is a US non-profit company for education organization, with a goal of giving a free global education all around the world through the internet. It was created and founded 2009 by Salman Khan with 3 MIT degrees. In the beginning Khan academy offered mostly mathematics but since then they have added other subjects like history, biology, computer science and a lot of others. The structure of studies on Khan Academy consists of about 10 minute long Youtube videos in where you would have different amount of parts depending on the subject. The 10 minute is important because it has been shown that about 25 minutes is the optimal learning time and after that you should have a small break(Lobdell 2011). This way multiple videos can be watched through out the day and learning remains efficient. After the videos, you will be asked questions about the subject to test yourself about it. The gamification shows in Khan Academy by getting points and hundreds of badges of a different kind to show the users accomplishments (Khan Academy 2016).

### 2.3.3 Memrise

Memrise is a language learning program where you learn new languages. The main way you learn a new language is memorization. The program will first give a new word and a picture or sentence (meme) how you will remember that. All the suggestions for remembering the word are made by the community so anybody can help each other for learning. “Ich” for example (German for I) it gives you picture of a t-shirt shown in Picture 1.



Picture 1: Example of Memrise memorization photo.

This picture will give you a connection with “Ich” and then your brain should make a connection with the word and in this case the picture. I actually used this program for learning a Korean alphabet before my exchange student time (Memrise 2016). After each short practice session, the users learning will be graded like shown in Picture 2.

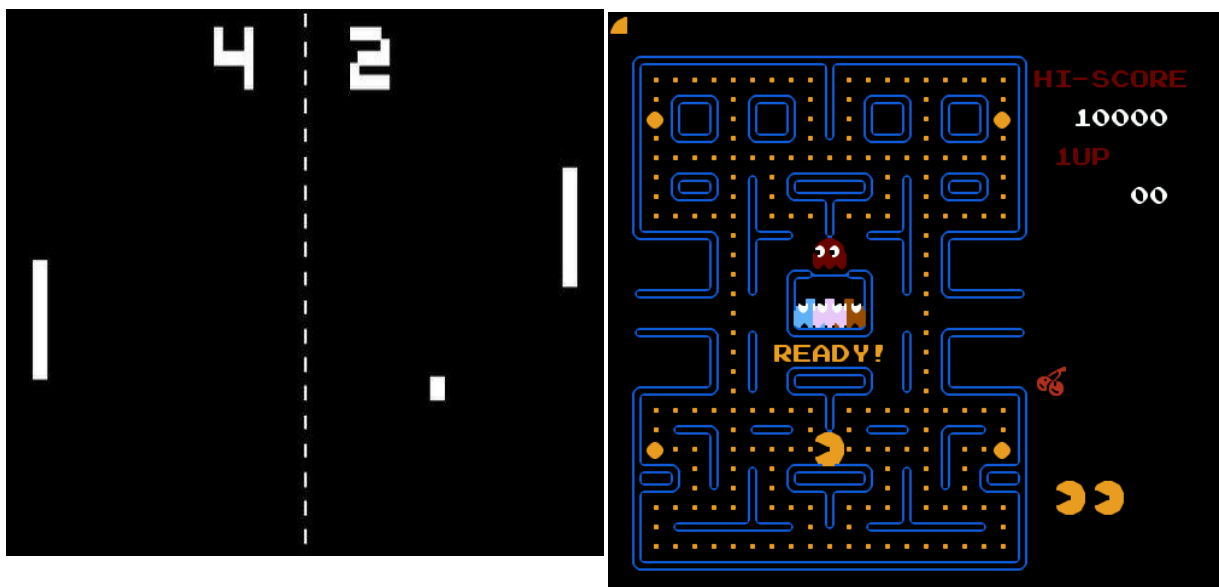
✓ Correct answers	28	1,225 pts
⌚ Speed	3min 55	52 pts
⚡ Accuracy	96%	312 pts
Total points		1589 pts

Picture 2: Example of the grading in Memrise.

This gives the user a fast feedback how did he or she do in the practice, this gives the motivation to achieve better score next time.

#### 2.4 Why do we play games?

Games have been a common leisure time since the 70's golden age of Amiga and Commodore 64. Community at that time was mesmerized about this new kind of entertainment that brought families together around your living room television. They were simple games with usually only joystick and one button as controlling methods. ONGELMA KUVIEN KANSSA!!!!



Picture 4: Almost everybody knows Pong. Picture 3: Pacman videogame.

Pictures 3. and 4. feature examples of early videogames. why do people like to play? Philosophy doctor Lauri Järvillehto says that “Popular games are affecting gamers inner motivation by simulating autonomy, competence and fellowship emotion”(Järvillehto 2014, 122).

## 2.5 Benefits of gamification

It has been shown that gamification is enhancing people learning, company's income with made purchases, the publicity of products by getting points when sharing product information. Nevertheless why does it work?

### 2.5.1 Motivation

The main point of motivation is to understand why people are doing something. There are motivations of two kind: extrinsic motivations and intrinsic motivations. Extrinsic motivations (external) are coming from outside. For example a medal from sports, cash incentives or to lose 10 kilos before summer season. Intrinsic motivations are motivations that come within and make you feel good about yourself. Let us make an assumption that you like to knit and you want to do more complex works, just to improve your skills. In Intrinsic motivations, the personal growth is important (Duggan & Shoup 2013, 26).

### 2.5.2 Flow

Flow is a humans optimal condition, where the user acts effortlessly, get done things and push them forward. Flow state usually demands four conditions:

- Perfect concentration for the task at hand.
- Tangible feedback from the task.
- The task has a clear goal and purpose.
- The skills and the requirements of the mission are in balance.

In flow state, the person is so immersed in the action, that not even one conscious thought comes up in to the mind. Researches have proven that the changes to the people environment disturb the flow experience. If these changes can be removed, by using ear plugs when studying in a noisy place for example, the perfect immersion is achievable. Flow demands continuous feedback about the progress. Without the feedback, it is impossible to know if the task produces positive results. Therefore the task demands a more conscious thought process, but then again that disturbs to keeping up the flow status. Adding a goal gives the task a structure and an end goal, but also a clear message when the task has been completed (Järvillehto 2014, 40).

### 2.5.3 Feedback

To make the user think that he or she is achieving something there has to be good feedback from the gamified product. There are three different kinds of feedback systems: Instant feedback, continuing feedback and cumulative feedback. Instant feedback means that the user gets immediate feedback what he or she has done. For example in the Angry birds when hitting the green pigs or eating a ghost in Pacman. With this feedback, the user knows that they have achieved something, which in affects the flow status. Continuous feedback means consistent mechanism, which notifies the user, that they are making progress. Most common examples from the continuous feedback are points or the chanting of a football game that is getting louder and louder. Cumulative Feedback stands for player's lasting growth acknowledgement during gameplay. The most distinct example of cumulative feedback is the user passing levels. Passing a new level could be recognised with new badges or equipment in games. All three feedback systems improve learning. Instant feedback keeps the user in flow status and upholds the interest: the user feels capable of doing new achievements. This also sustains dopamine metabolism, which results from that dopamine system stimulates always when we accomplish something new. Continuing feedback keeps the user interest by showing that they are advancing to the target. Finally cumulative feedback functions giving the know-how needs gratification by showing



clearly and unambiguous, that the user is moving ahead also in the bigger picture (Järvilehto 2014, 142-143).

#### 2.5.4 Enjoyment

People have their own preferences what they enjoy to do and now the rise of gaming in different age groups is giving a change to use games as a learning method. Figure 2. Shows the distribution of age and gender within game players.

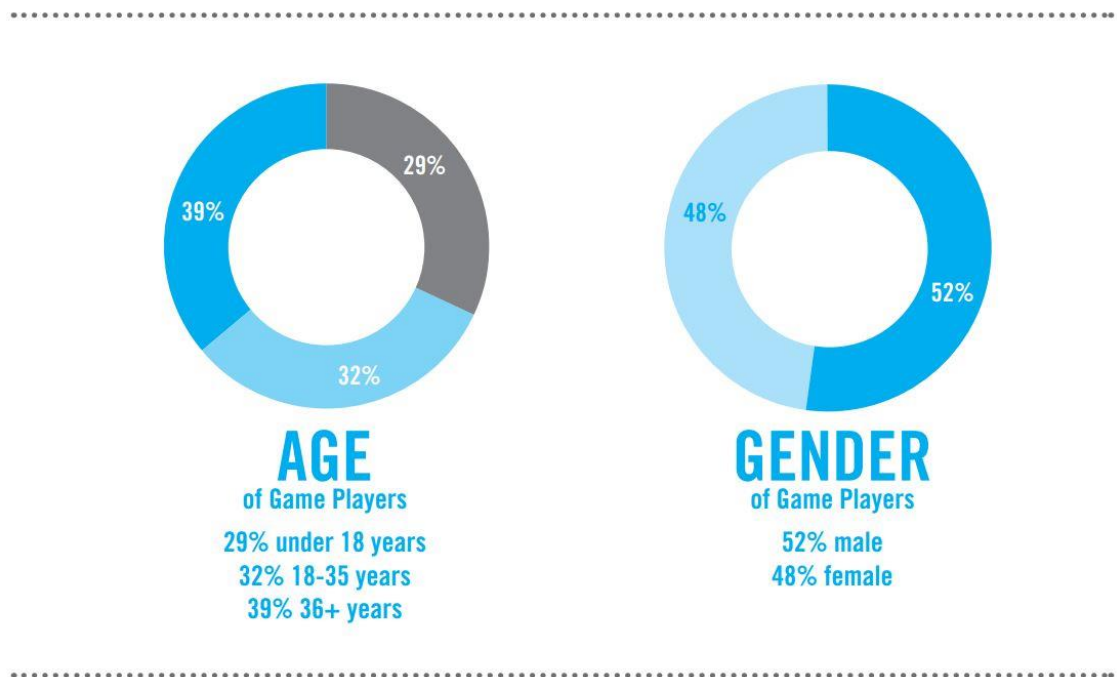


Figure 2: Gaming age groups division.

In a research about the connection of gaming with added learning incentives the results were positive. Michael Giannako (2013) and Ming Yueh Hwang and colleagues (2013) had a research where students play games alone. In Giannako's research 41 students play GemGame-learning game in where he investigated the enjoyment and happiness that the game gives and also students intention to play the game. The learning game was simple, which was based on traditional gaming mechanics. The research result was that the enjoyment produced by the game did affect positively on the individuals learning (Krokfors et al. 2014).

### 2.5.5 Competition

Competing is a basic feeling that gives a huge sensation of achievement and keeps you pushing forward closer to your goal. This competitiveness can be obtained by simple leaderboards. Leaderboards will give the user instant feedback on how he is doing against his colleagues, friends and everybody else. The goal for the user to win the leaderboard is fame and bragging rights which boost users confidence. Leaderboards can be used to count many different parameters: videos shared, likes accumulated in Facebook, the fastest time or the most usual points just to mention a few. There are certain rules for building effective leaderboards. It has to be motivational for user that he or she cares to use the precious free time to achieve something. There are three solutions for this: using different time frames, shrinking the context and show “me”. The time frame solution divides the leaderboards into different categories e.g daily top users, weekly top users and all time top users. That way the user has many means to compete. Shrinking the context means having smaller groups that are competing e.g friends or work colleagues. The third principle is to show who is under or in front of you in the leaderboard, which gives the user his position in the board and what to do to rise or to prevent going down in the leaderboards (Paharia 2013, p.79-80).

## 2.6 Using gamification in the project

Using gamification in the project would be measured with an add-on program on a USB stick that connects to the computer and it follows how the user is doing. The program would be self programmable itself since it seeks just certain signals what happens on your computer. So it would be possible to gamify different kinds of projects.

### 2.6.1 Points

Points would be gradually collected while following the guide and getting through new obstacles for the user. Adding for example your very first component to the 3D world will give you 100 points or adding a new PLC program in TwinCAT.

### 2.6.2 Levels

The user will achieve levels with the given points that he or she obtains. Every-time the user gets points they will be added to a level bar that shows their current level and when they achieve next, as demonstrated in Picture 5.



Picture 5: Example of a points meter.

It shows the user clearly how much they have points and how much they need for the next level, which instead promotes motivation to proceed with the task.

### 2.6.3 Badges

Badges give the user bragging rights against other users and good feeling about themselves. They will pop up with certain specific actions you have done in the work but with a broader sense than points. They are a visual representation of an achievement as seen on Picture 6.



Picture 6: Example set of badges.

In this project, badges would be given for example when opening first time TwinCAT there would be a pop up badge for “Logic adventure begins” with an image of a Himalaya mountain which signifies that the user is just starting to learn to use Logics and the main goal is to climb the “Mountain” to achieve competence in using TwinCAT. The more the designer uses creativity with naming or in the design of the badge, the more the user is motivated to get them. There would be also secret badges that are marked by name, but the exact conditions are not revealed. That would give the user a purpose to try different things inside the programs to get that mysterious badge.

#### 2.6.4 Leaderboards

There would be leaderboards of a different kind that the user can see. There would be Top 10 user list that shows the best-leveled users, “Me” leaderboard shows the users spot on the leaderboard and a couple of users that are under he or she and also on top of the user. There could be also a leaderboard that is class friend oriented.

### 3 THE TOOLS

I will be using a Visual Components 3D create program for designing the automatic conveyor with all the moving parts and Beckhoffs TwinCAT 3 for making PLC program that control the parts in the Visual Components.

#### 3.1 Simulation

These days a simulation is an irreplaceable tool for industrial engineers every day. The simulations primal meaning is “the imitation of the operation of a real-world process or a system over time ”(Banks 1998, 3). In this thesis case, let us think about building an automatic bottling line to the Coca Cola company, it would take a lot of work hours to design the conveyor line and ordering the supplies and if there is even one calculation mistake it has to be solved to make it work as intended. With simulation software its possible to design the product already virtually and check that it for certainly will work without problems.

##### 3.1.1 Advantages and disadvantages of simulation

The advantages of simulation:

- **Testing a design prior:** Simulation software gives a big benefit of testing the best solution before ordering hardware.
- **Time manipulation:** by speeding or slowing time you can inspect what is happening in the simulation at any given time.
- **Understanding why something happens:** making a replica with simulation helps to understand why certain phenomenon happened.
- **Diagnosing problems:** gives a better understanding about the systems (e.g factory) for the user.
- **Identifying constraints:** manufacturer’s nightmare is bottlenecks in the production but by performing bottleneck analysis you can discover why there are delays.

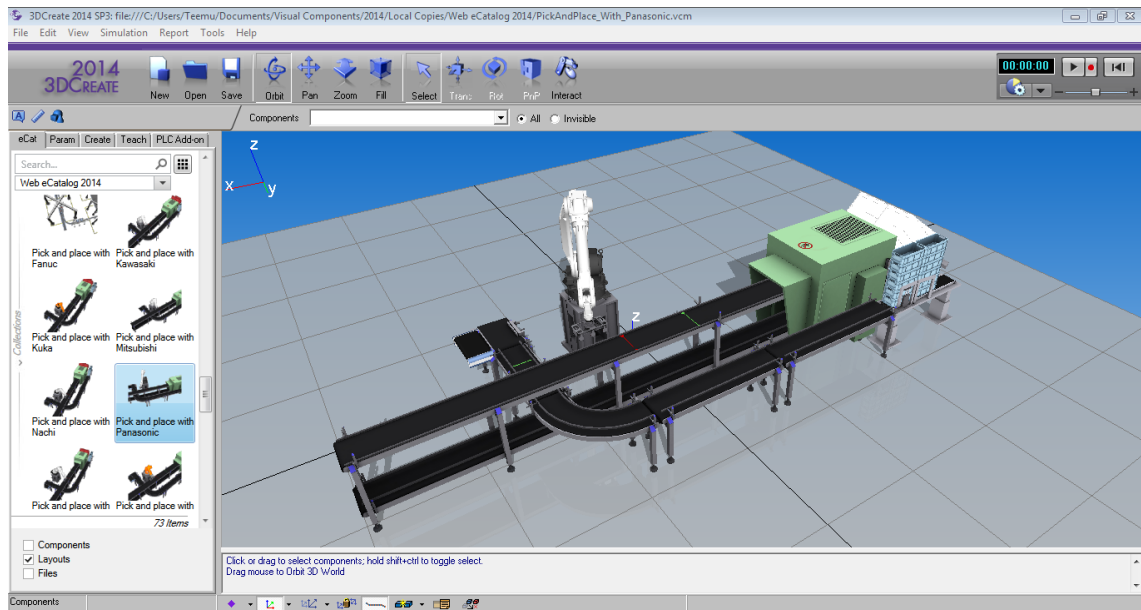
- **Visualizing:** with visualizing you can see the facility or organization running. This gives more insight than just looking at CAD drawings for the user.
- **Redesigning:** simulation software ability to be able to redesign the system gives the user full of options to change the system for the future.
- **Smart investment:** the training for the simulation software is far less costly than redesigning existing installation.
- **Training the workforce:** by being able to make mistakes the users can learn to operate better which is much cheaper than learning on the job.

The disadvantages of a simulation:

- **It requires a special training:** learning to use the software takes time and experience.
- **Simulation modeling can be time consuming and expensive:** skimping on resources for modeling and analysis may result in a model that is insufficient.
- **Inappropriate use of simulation:** using simulation when not necessarily needed. Simulating for example waiting lines in a supermarket. Its more preferable to use analytical solution (Banks 1998, 10-12).

### 3.2 Visual Components

Visual Components is one of the commercial 3D simulation design programs for making automatic conveyors with robotics. It shows the material flow easily and it is used for planning and optimizing factory layouts and production lines as shown in picture 7.



Picture 7: Basic pick and place automatic conveyor line with Panasonic robot simulated by Visual Components.

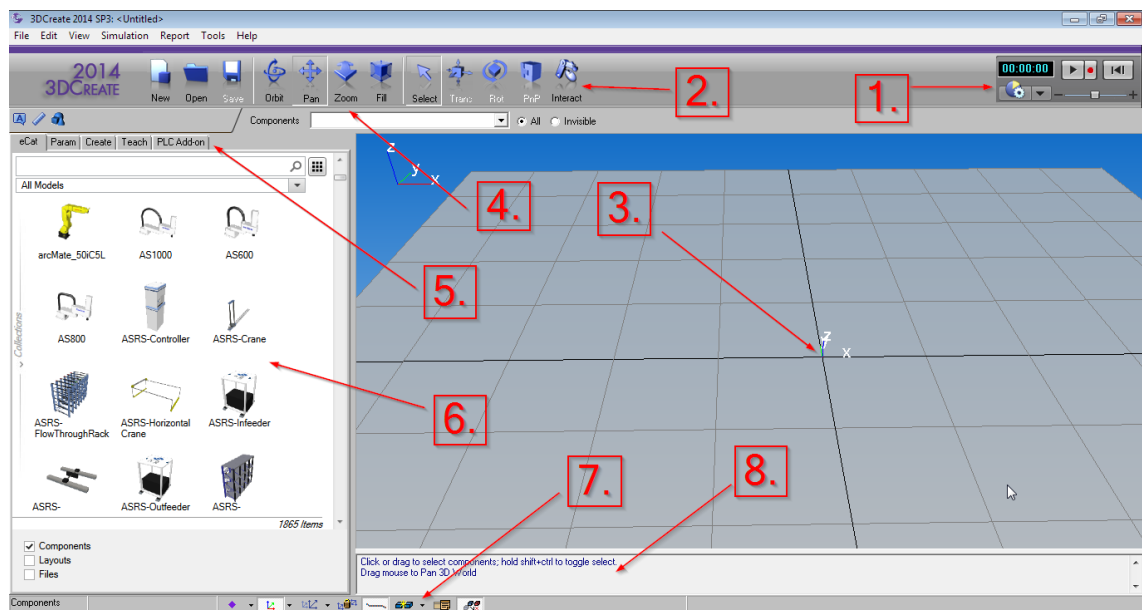
Visual Components has extensive premade components (electronic catalogue, eCat) that can be snapped together easily, think of it as a building simulator for Lego. You can also make your own components because all the objects are geometric shapes or bring your own designs from CAD files. Robotics is programmed with RSL-language (Robot Sequence Language) with a combination of Python programming language.

### 3.2.1 Application

There are two main applications for the Visual Components program: Visualization and simulation. Visualization always gives you an up to date view of your designed production line. You can add and delete existing components by using the components library of the software. Simulation creates an accurate virtual version of the production line which allows to analyze the efficiency of the virtual product line before it is even built in real life.

### 3.2.2 Building simulations

Making a simulation with Visual Components has been made to be as simple as possible. You have a great number of components to accomplish your own design goal. By adding first object by dragging or by double clicking it to move it to the origin of the 3D world (e.g Advanced Feeder) you can snap other components to it to make a fully functioning automatic conveyor. Picture 8 explains all the tools when opening 3D Create.



Picture 8: Basic starting view of Visual Components.

1. Speed increase, speed decrease, play, stop, record and reset controls.
2. Connection and moving tools for components: Translate, Rotate, Plug and Play components and Interact with components.
3. 3D world and origin
4. Interaction with 3D worlds camera: Orbit, Pan, Zoom and Fill.
5. Tool Tabs: eCat, Parameters changing components basic parameters), Create (creating new geometry, Behaviour and Parameter), Teach (Teaching robotic movement) and PLC Add-on (connecting Visual Components to an external PLC program).
6. eCat



7. Status bar: Option to turn off certain views in 3D world.
8. Message board: Text board for feedback what is happening.

### 3.2.3 Programming simulations

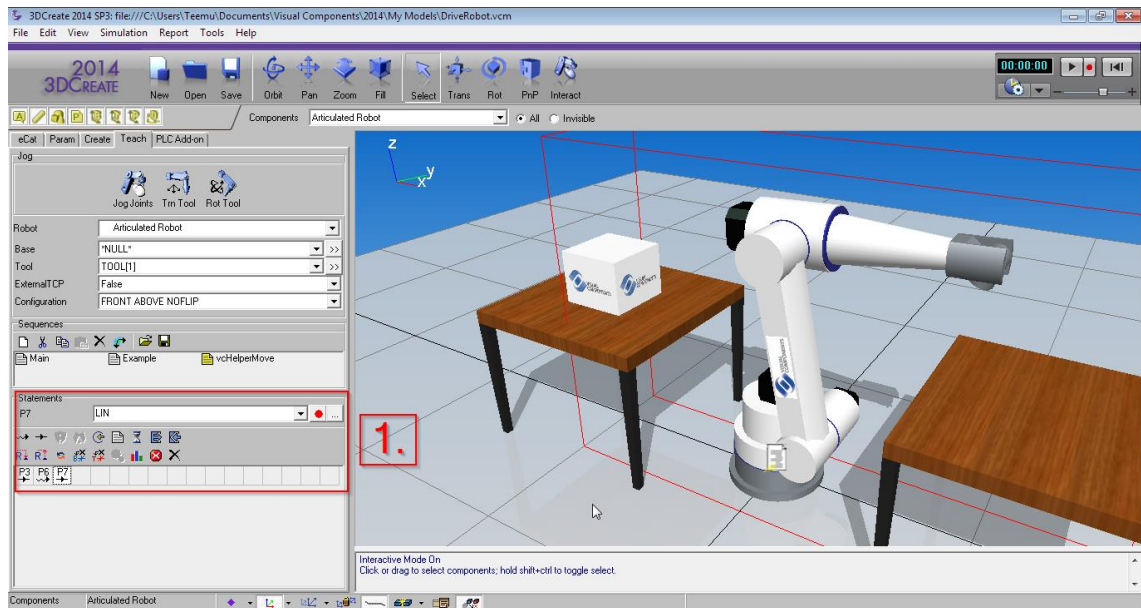
Python language is used in Visual Components for adding uses for applications and add more complicated simulations easily. The language is an interpreted, object oriented programming language that is easily readable, because it often uses English keywords. Other programming languages use punctuation which makes it somewhat harder to understand. (Python overview 2016)



Picture 9: Differences between Java language and Python language.

Picture 9. demonstrates how easily understandable Python programming language is.

Robot sequence language is based on RRS II standard, that doesn't have to contain control flow statements, such as "if-then-else" or "loops" (Visual Components, 2004, 53). RLS is used to make a point to point moving with a robot, which is made by moving the robots joints in a certain position and then saving the move route(1.).



Picture 10: RSL in Visual Components.

In Picture 10. We have added three different moving trajectory for the robot.

### 3.2.4 Interfaces with other programs

Visual Components is versatile platform for many different interfaces:

- Mitsubishi GX Simulator 2
- Siemens S7
- Stäubli Controller add-on
- Media Toolkit
- Beckhoff TwinCAT
- PLC add-on using OPC connection

### 3.2.5 Usability

The Visual Components is really easy to use for simulation purposes. The drag and drop mentality of the components is really easy to assimilate. There are great tutorials in the Visual Components community page that will help getting additional information about the simulation program.

### 3.3 Programmable Logic Controllers

Programmable logic controllers (PLC) are devices which are equipped with a microprocessor, that guides and regulates machines and processes function with a logic program. Switches, contacts and sensors, that are added to it to transmit information on the devices operation status and of the processes measurement values. With these information logic programs are used to guide for example relays, lamps, magnet valves, motors and cylinders (Värjä & Mikkola 1995, 5).

#### 3.3.1 Differences with PLC and Computers

PLC and computers resemble each other, but they are made for different work tasks. Computers are made for calculating and displaying tasks, when instead PLC is designed for controlling tasks and industrial circumstances. The differences for comparing PLC to a computer are (Bolton 2015, 6):

- PLCs are made to withstand harder conditions than basic computer. PLC can cope with noise, humidity, vibrations and temperature.
- PLC has an expandable rack, that makes it easily customizable for larger number of inputs and outputs.
- They are easy to program and have a straightforward programming language. It is focused on logic and switching operations.
- PCs are better at long term data storage and analysis
- PCs crash more easily compared to PLCs

### 3.4 TwinCAT 3

TwinCAT is an engineering tool developed by Beckhoff for PLC programming. TwinCAT 2 needed a licence to work but Beckhoff have designed TwinCAT 3 to be technically free to use.

#### 3.4.1 Licences

There are two types of licences: temporary licence and paid licence. The main difference between these is that in order to keep, for example your product line

running, you need the paid licence that it is continuously working and the temporary licence is a great way to learn how to do the PLC programming. This is especially great for schools because it is a free program and also you don't need any tangible hardware to use because everything is virtualized.

### 3.4.2 Structured text (ST)

Structured text is another standardized programming language that is used in TwinCAT. ST has a certain flow or structure how it is written and it has to be precise on that, as demonstrated in Picture 11.

```
PROGRAM stexample
  VAR
    x : BOOL;
  END_VAR
  x := TRUE;
  REPEAT
    x := FALSE;
  UNTIL x := FALSE;
  END_REPEAT;
END_PROGRAM;
```

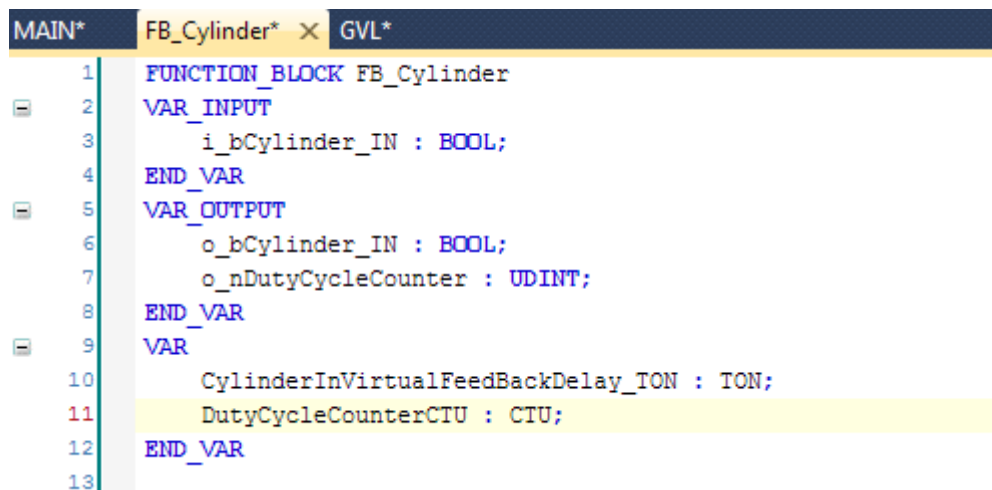
Picture 11: ST example (PLC Academy 2015).

In case of Picture 11, the program of the example is inside “PROGRAM stexample” and “END\_PROGRAM”. The program doesn't stop to the END\_PROGRAM, but PLC scan cycle starts again from the top. Indentation for example to the “x := FALSE;” means that the sub-paragraph belongs to the “REPEAT” paragraph. After the user is done with the sub-paragraph, the next line will be unintended back to the beginning of the next line to a new paragraph. There are a lot of semicolons, colons and other symbols. These symbols have a purpose in the TS, some of them are operators, functions, statements or variables. The basic rules of ST:

- **All statements are divided by semicolons:** Semicolons end the statement.
- **The language is not case sensitive:** It is not mandatory to use upper- and lowercase letters. Eventhough it helps readability.

- **Spaces have no function:** again used for readability.

When writing ST the user has to think how well can another user understand the written text. For example variable names should show what they are, like Boolean or integer.



```

1  FUNCTION_BLOCK FB_Cylinder
2  VAR_INPUT
3      i_bCylinder_IN : BOOL;
4  END_VAR
5  VAR_OUTPUT
6      o_bCylinder_IN : BOOL;
7      o_nDutyCycleCounter : UDINT;
8  END_VAR
9  VAR
10     CylinderInVirtualFeedBackDelay_TON : TON;
11     DutyCycleCounterCTU : CTU;
12 END_VAR
13

```

Picture 12: Example from the main project.

In the example shown in Picture 12, it is possible to see basic rules of a different kind of text to make the ST more easily readable. Firstly, thinking how to name them, in this case function block program, the name starts with capitalized “FB” for function block. After that there is an underscore “\_”, which is a divider between a name and program/variables etc. “i\_bCylinder\_IN” is divided into four parts: i (input), b (Boolean), Cylinder (name) and IN (signifies the move of the Pusher-Convey-or). Another example, “o\_nDutyCycleCounter”. In this variable are also 4 parts: o (output), \_ (divider), n (numerical) because this is for the calculation of the rejected products and DutyCycleCounter (name), notice that every word starts with a capitalized letter. After the program is uploaded, the computer translates it to a language that the PLC understands. A statement is the user telling to PLC what to do.

A : BOOL;

This is an example of a statement. The user is commanding the PLC to make a variable (A) that should be a BOOL type (more on that in chapter 3.4.3). The PLC understands that this is one statement because it reads the semicolon at the end of the line. Statement means the same as in spoken language, for example a company can give a statement (PLC Academy 2015).

### 3.4.3 Data types

Data type means a set of data that has certain values that have specified characteristics (Rouse 2005). They are also divided into an integer and a float; whole number and fractional number. These numbers below mean that the programming has to be designed from the beginning to fit in certain limits, if the programmer wants to do an automatic calculator he or she has to think how long does the calculator count. Would BYTE (0-255) be enough or is it too small, because the program will stop at the count at 255 (Koehler 2013). The usual data types are:

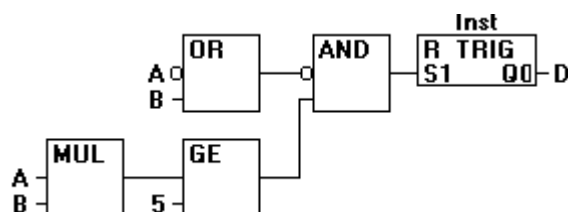
- **BOOL:** signal is TRUE or FALSE (ON and OFF). Memory use 8 Bit.
- **BYTE:** integer which uses numbers from 0 to 255. Memory use 8 Bit.
- **WORD:** integer which uses numbers from 0 to 65535. Memory use 16 Bit.
- **DWORD:** integer which uses numbers from 0 to 4294967295. Memory use 32 Bit.
- **SINT:** short signed integer which uses numbers from -128 to 127. Memory use 8 Bit.
- **USINT:** unsigned short integer data type which uses numbers from 0 to 255. Memory use 8 Bit.
- **INT:** signed integer which uses numbers from -32768 to 32767. Memory use 16 Bit.
- **UINT:** unsigned integer which uses numbers from 0 to 65535. Memory use 16 Bit.
- **DINT:** signed integer which uses numbers from -2147483648 to 2147483648. Memory use 32 Bit.
- **UDINT:** unsigned integer which uses numbers from 0 to 4294967295. Memory use 32 Bit.

- **REAL**: floating point which uses numbers from  $\sim 3.402823 \times 10^{38}$  to  $\sim 3.402823 \times 10^{38}$ . Memory use 32 Bit.
- **STRING**: variable that contain any string of characters. Memory usage depends on how many characters there are.
- **TIME**: definition of time from T#0ms to T#71582m47s295ms. Memory use 32 Bit.
- **TIME\_OF\_DAY**: time of day from TOD#00:00 to TOD#1193:02:47.295. Memory use 32 Bit.
- **DATE**: day of the year from D#1970-01-01 to D#2106-02-06. Memory use 32 Bit.
- **DATE\_AND\_TIME**: day and time from DT#1970-01-01-00:00 to DT#2106-02-06-06:28:15. Memory use 32 Bit (Beckhoff 2016)

#### 3.4.4 Function Block Diagram (FBD)

The function block diagram is a graphically oriented programming language. It works with a list of a network whereby each network contains a structure which represents either:

- a logical or arithmetic expression
- the call of a function block
- a jump
- return instruction (Beckhoff 2016).

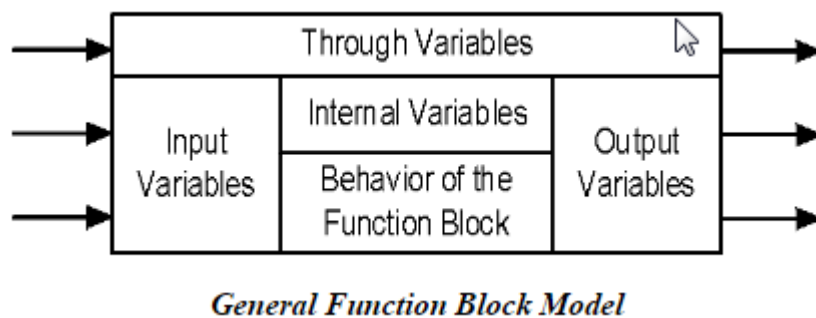


Picture 13: Function Block Diagram Example (Beckhoff 2016).

Picture 13. shows a basic function block diagram. FBDs are read from left to right and the lines represent the signal pathways.

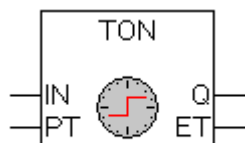
### 3.4.5 Function blocks

A basic function block consists of input variables (left side) and output variables (right side) through variables, internal variables and a behavior description (picture 14) .



Picture 14: A general function of a function block.

Input variables can only be inserted from externally. Inside of the FB they can be only read, but output variables can be read and written from inside and only can be only read from outside. Here are a couple of example function blocks (Heverhagen et. al.) :

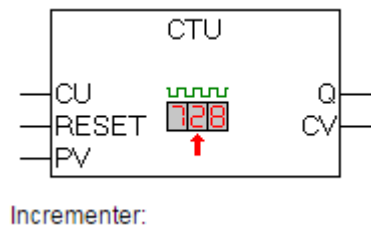


Timer on-delay

Picture 15: Timer on-delay (TON) function block(Beckhoff 2016).

Picture 15 has a picture of TON FB. This block is used as a timer for certain PLC projects.



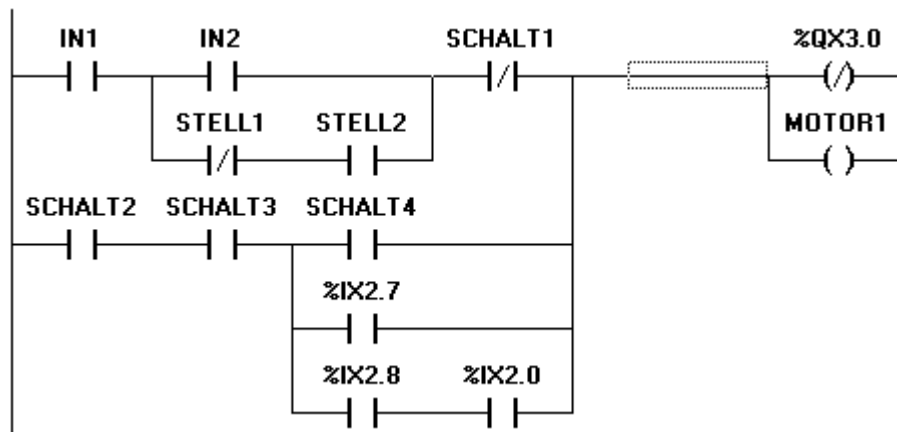


Picture 16: A count up function block (Beckhoff 2016).

CTU is used to count up something. In this project it will be used to count rejected products.

### 3.4.6 Ladder Logic Diagram (LD)

The second graphics oriented programming language is Ladder Logic Diagram, but LD has a different kind of a structure as shown in Picture 17:



Picture 17: Ladder diagram has a structure of an electric circuit.

The LD is suitable for two things:

- Constructing logical switches.
- Creating networks as in FBD.

This makes LD advantageous for controlling the call of other POU's (more on POU's later). LD consists of a series of networks. On Left and right side of the

Picture 17. can be seen a vertical current line that limits the network. Inside of the left and right network limiter is the circuit diagram consisting of contacts, coils and connection lines.

Networks have contacts ( represented by two parallel lines `| |` ) that pass on from left to right, the condition “ON” or “OFF” correspond to the Boolean values TRUE and FALSE. Each contact has a Boolean variable. If the variable is TRUE, then condition is passed forward on the connecting line. Unless its TRUE, then the connection receives the value OFF. Every contact has to transmit the value “ON”, otherwise the connection won’t work. A contact can be also negative with a “slash” between the straight lines `|/|`.

Coils are on the right side of the network, which are portrayed by a parenthesis `()`. A coils function is to transmit the values of the connections and copy it in a befitting Boolean variable. Coils can also be negated with the “slash” `(/)`, then it copies the negative value.

Set and Reset coils are represented by `(S)` or `(R)`. A Set coil never writes over the value TRUE in the appropriate Boolean variable. It always keeps the value TRUE. Reset coil does the same thing except it keeps FALSE (Beckhoff 2016).

## 4 THE MAIN PROJECT

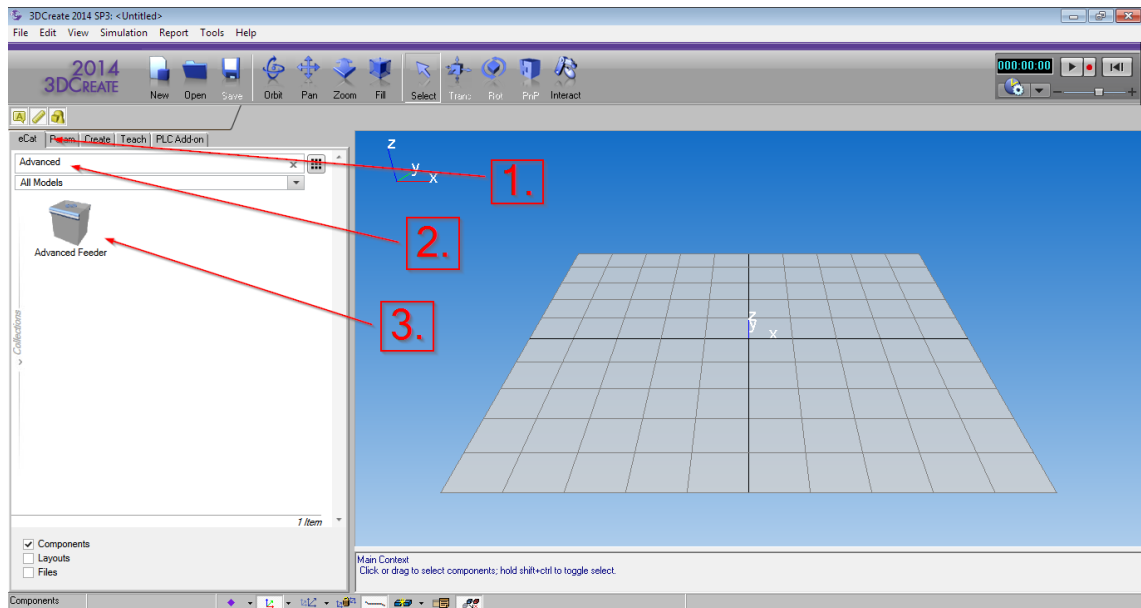
The main project starts with making an automatic conveyor simulation first in Visual Components. The conveyor has a pusher that removes one of two products from the conveyor. There will be two different products, green and red, products arriving to the line with certain probability and in this case it removes the red products that are "defective". After that the logics will be made with TwinCAT, which controls the pusher in real-time. In the PLC program will be added a virtual feedback delay to imitate a real life situation with delay and cycle counter that counts the defective products. Finally in part three will be made the connection between the two simulation programs.

### 4.1 Visual Components Simulation

In the first part of the simulation, the components will be added to the 3D world, connecting the components as an automatic conveyor and testing in the end that it works as intended.

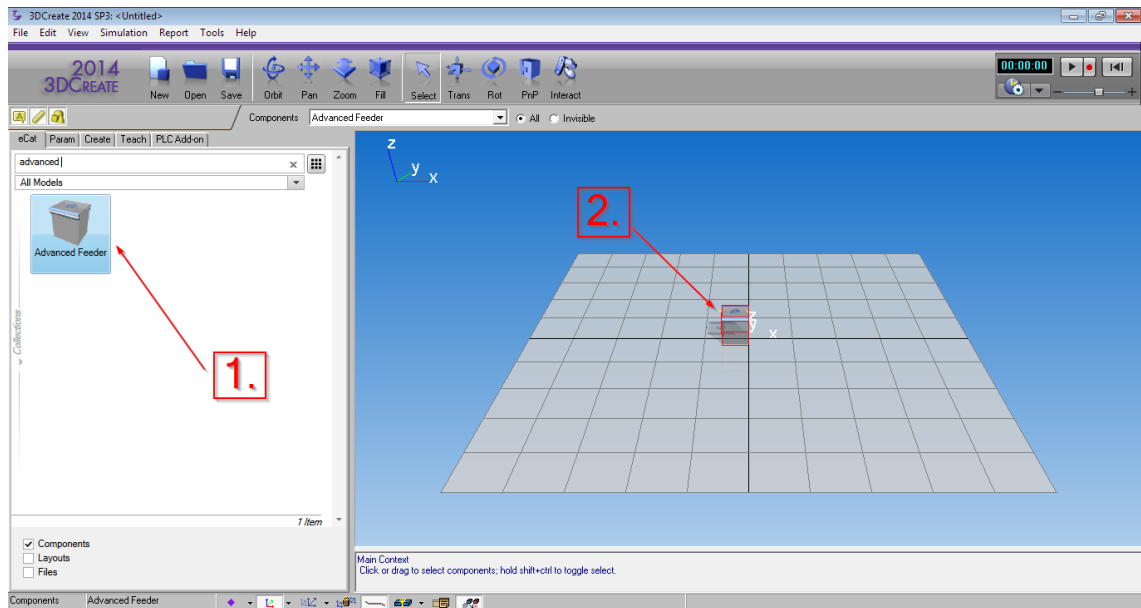
#### 4.1.1 Adding components to the 3D world

First let's open the Visual Components 3D Create simulation program which starts with an empty 3D space (picture18). Next we add the needed parts for the simulation: Advanced Feeder X 1 (creates the products), Pusher Conveyor X 1 (removes the defective product from the conveyor), Conveyor X 3 (moves the component forward). The components can be found in the eCAT tab (1.) and when starting to write in the search field (2.) the eCAT automatically shows the right component (3.).



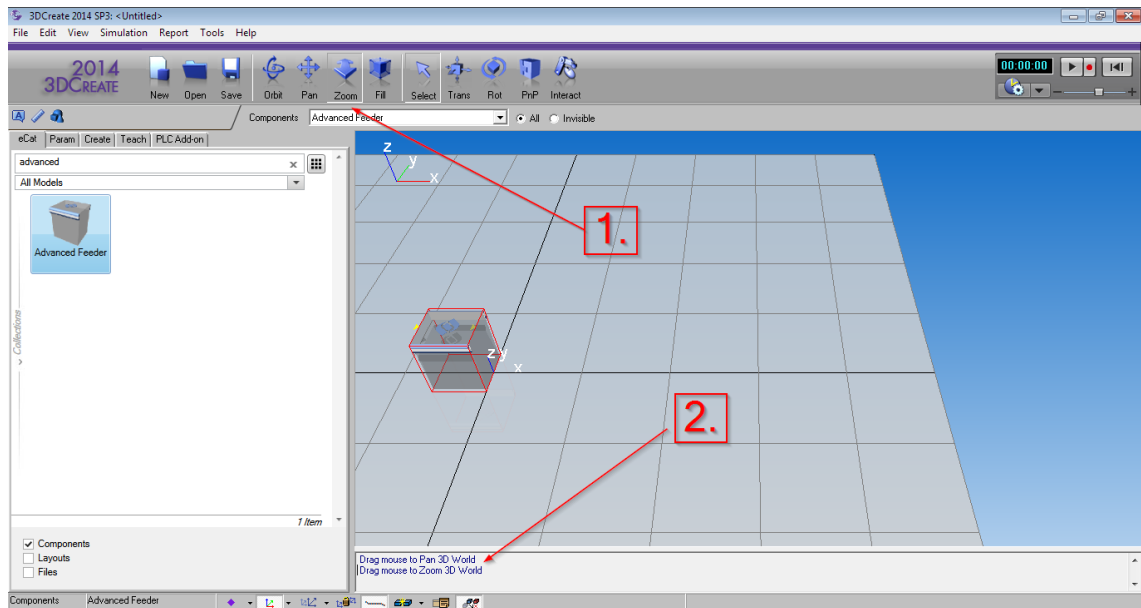
Picture 18: Finding components in eCAT.

Now that the first component have been found, the following step is to add them to the 3D world (picture 19). There are two options how to add the components to the 3D world, double clicking the left mouse button so that the component is automatically situated in the 3D world origin or dragging the components to a designated spot. For the Advanced Feeder, the double click method will be used (1.) which moves the component to the 3D world origin (2.).



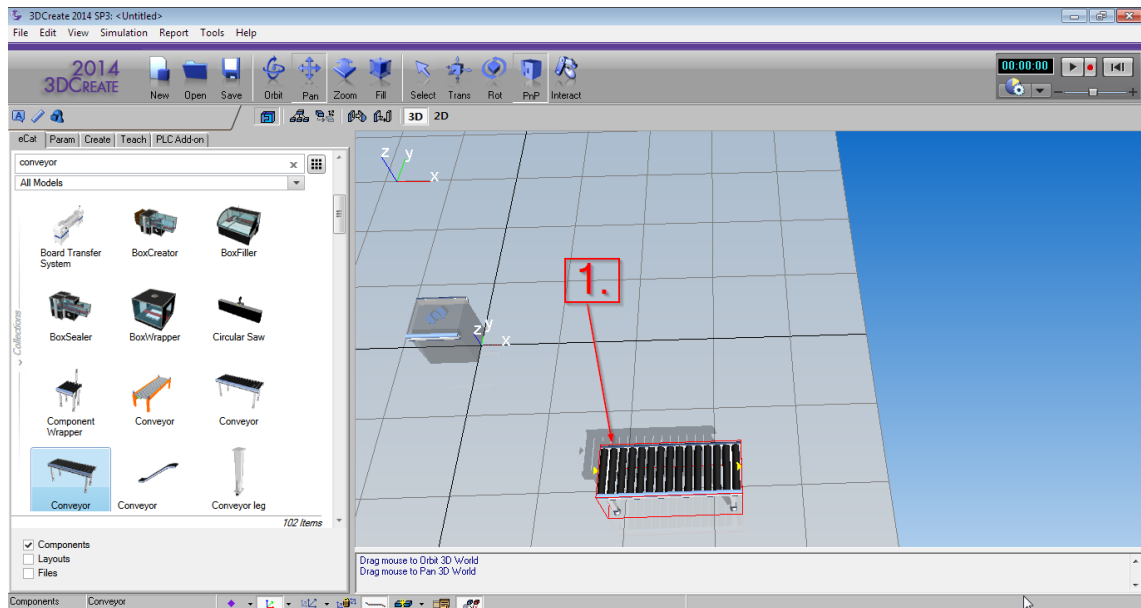
Picture 19: Adding the Advanced Feeder to the 3D world origin.

Advanced Feeder has been now added and to get a better look at the component the 3D world view tools will be used. There are Orbit, Pan, Zoom and Fill choices (1.). The Orbit tilts the view around the origin, Pan moves the view at a 2D level, and Zoom makes the view closer or far away from the user and Fill changes the view to accommodate the whole component as close as possible. In this case Orbit, Pan and Zoom will be used. Every time a tool is chosen with mouse the message board under the 3D world shows how to use the tools. For changing the views, it shows to drag the mouse to change the view (2.). Choose Orbit with left mouse button and change the view more upward, Move the view with Pan so that the right side of the Advanced Feeder only shows, because all the other components will come on this side. Finally Zoom to have a closer look at the Advanced Feeder (picture 20).



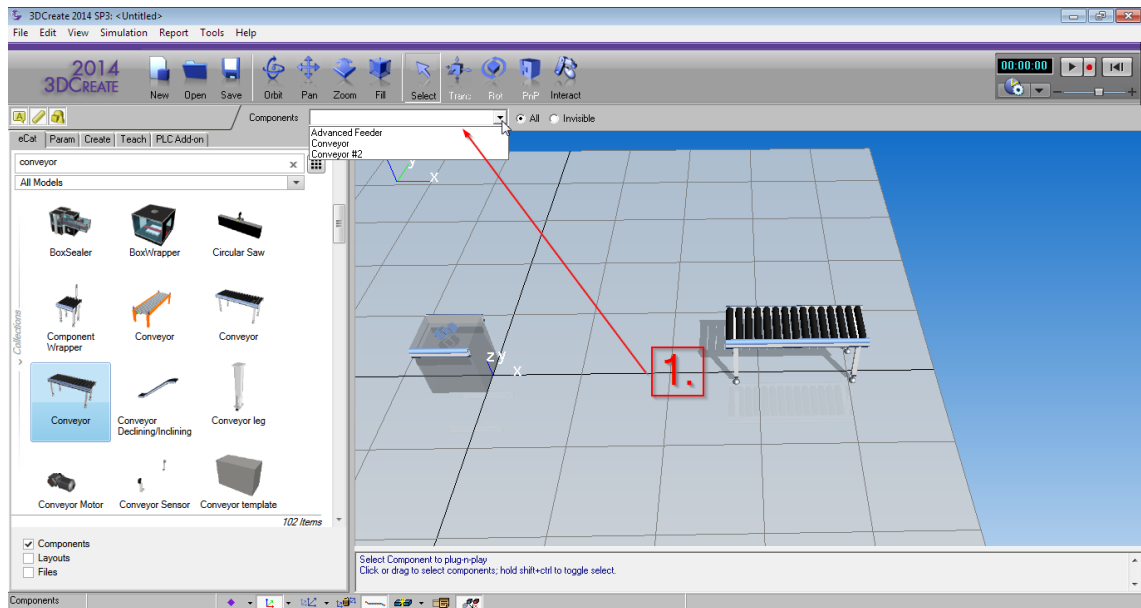
Picture 20: Final result of changing the view.

Now that the advanced Feeder is in its final position the rest of the components can be added. This time the dragging method will be used. Finding the conveyor goes the same way as Advanced Feeder. Typing “Conveyor” in eCat gives a lot more options for components this time, it is possible to find it by scrolling down slightly. By holding the left mouse button, drag the Conveyor to the 3D world. Easier method for adding the two more Conveyors is with copying and pasteing. Choose the Conveyor that was just added to the 3D world by clicking it. Red outer lines on a component indicate which component is currently chosen (1.) (picture 21).



Picture 21: Chosen product active with red lines.

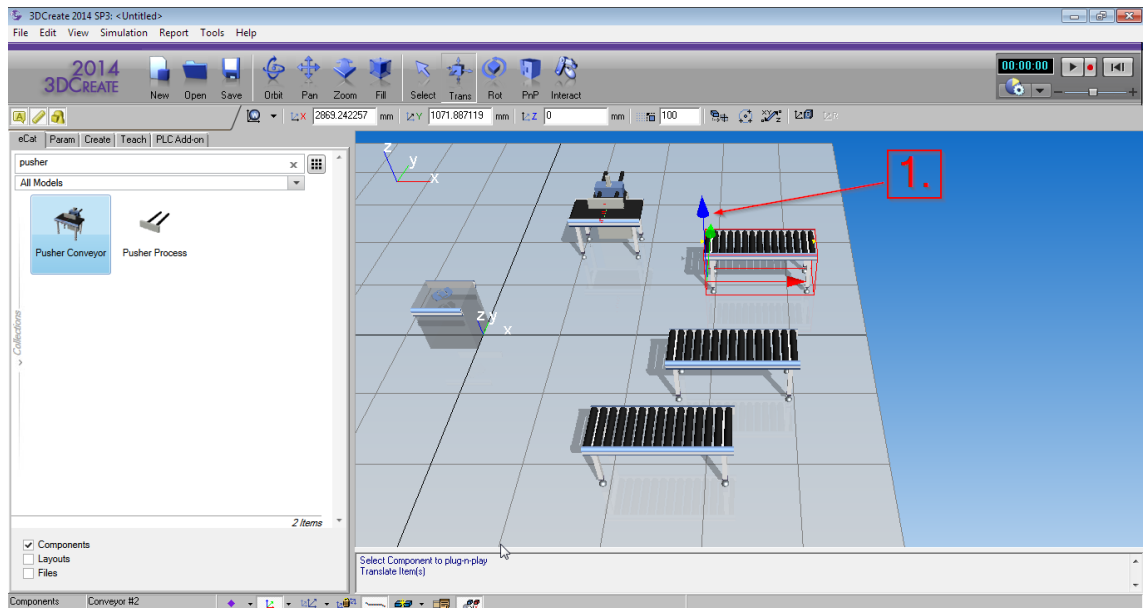
After making sure that the component is chosen, press CTRL + C which copies the component. Click the 3D world, to make sure that the component is no more active and press Ctrl + V. This makes a copy of the product. Notice that the copy was created on top of the already existing component. By checking Components drop down list (1.) it is possible to see all the added components in the 3D world. Components on top of each other are also shown (picture 22).



Picture 22: Making sure that the Conveyor was copied.

After making sure that the copy is there, make another copy again with Ctrl + V when you don't have any chosen components. Moving the Conveyors apart happens when first choosing the Conveyor #3 from the drop down list that makes the Conveyor again highlighted with red and then choosing Trans button. The Trans button gives the option to move any components in the 3D world. When Trans is active and there is a chosen component, it shows red, green and blue arrows which signify in which direction the component can be moved. This also shows that Trans is active (1.). Drag the Conveyor apart by dragging it, pressing Esc makes the Trans button inactive and do the same with Conveyor #2. The last component is Pusher Conveyor which again is dragged to the 3D world (picture 23).

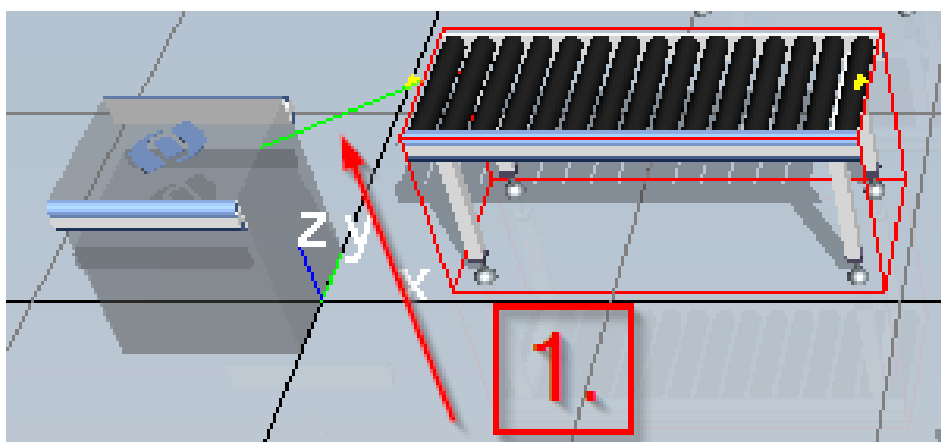




Picture 23: Final result after adding all the components.

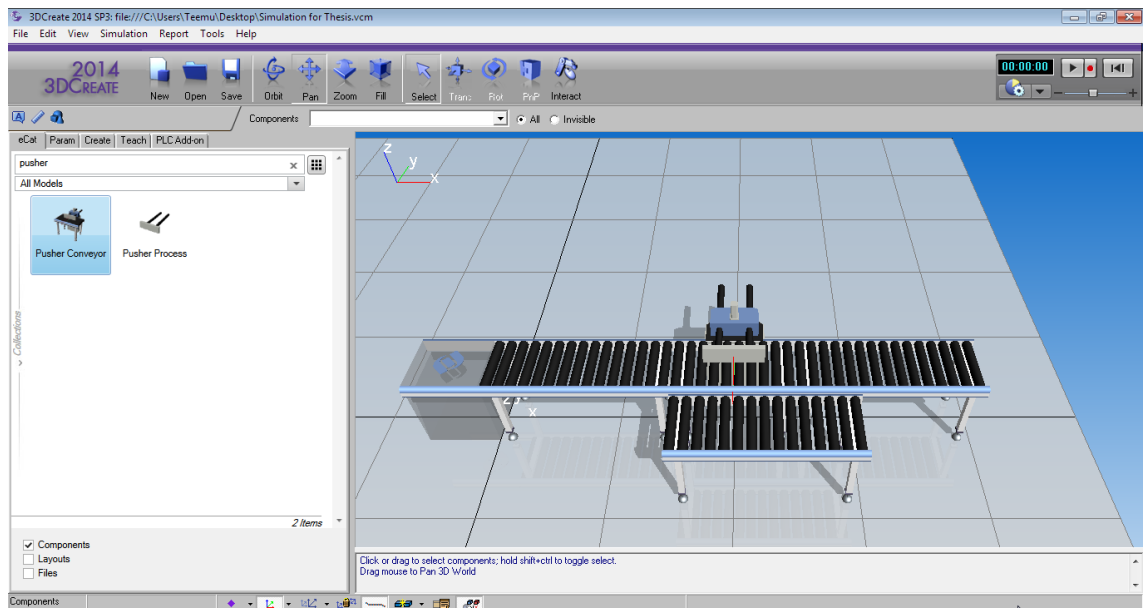
#### 4.1.2 Connecting the components together

In this section, the components will be connected together and we also test that the conveyor works. New command will be introduced: PnP (Plug and Play components). By choosing one of the Conveyors and pressing PnP makes it in “connection” mode. Dragging the Conveyor now close to the Advanced Feeder shows a green line (1.) that means that these two components can be connected (picture 24).



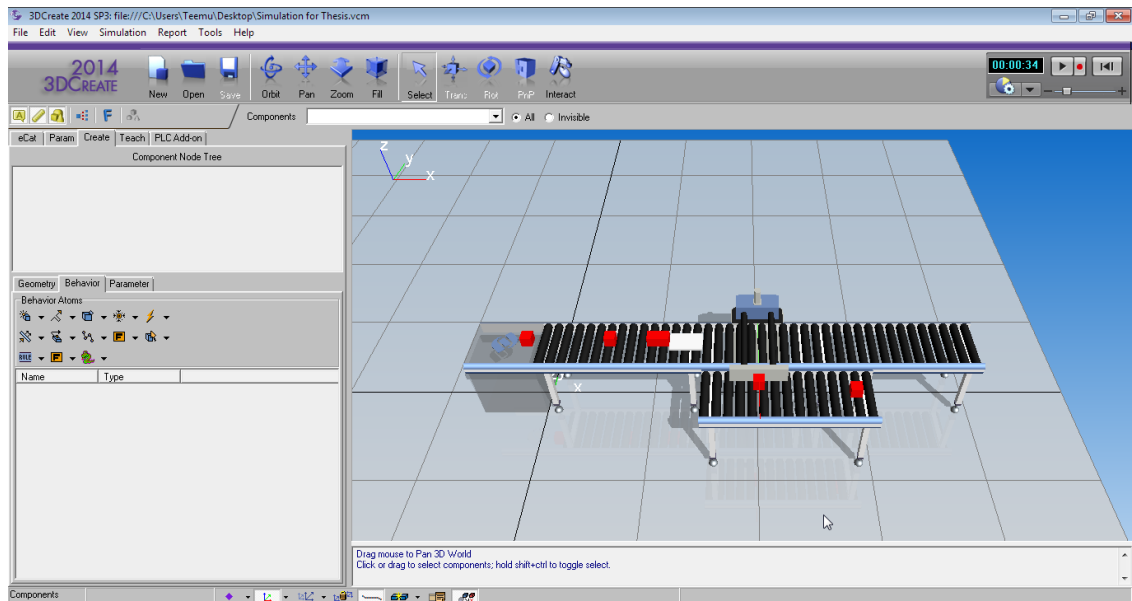
Picture 24: Green line for connecting components.

Moving the Conveyor even closer makes two components “snap” together. After the first connection (pressing Esc and on to the 3D world unselects the component), add the Pusher Conveyor on the right side of the Conveyor, the second Conveyor on the right side of the Pusher Conveyor and finally the third Conveyor in front of the Pusher Conveyor (picture 25).



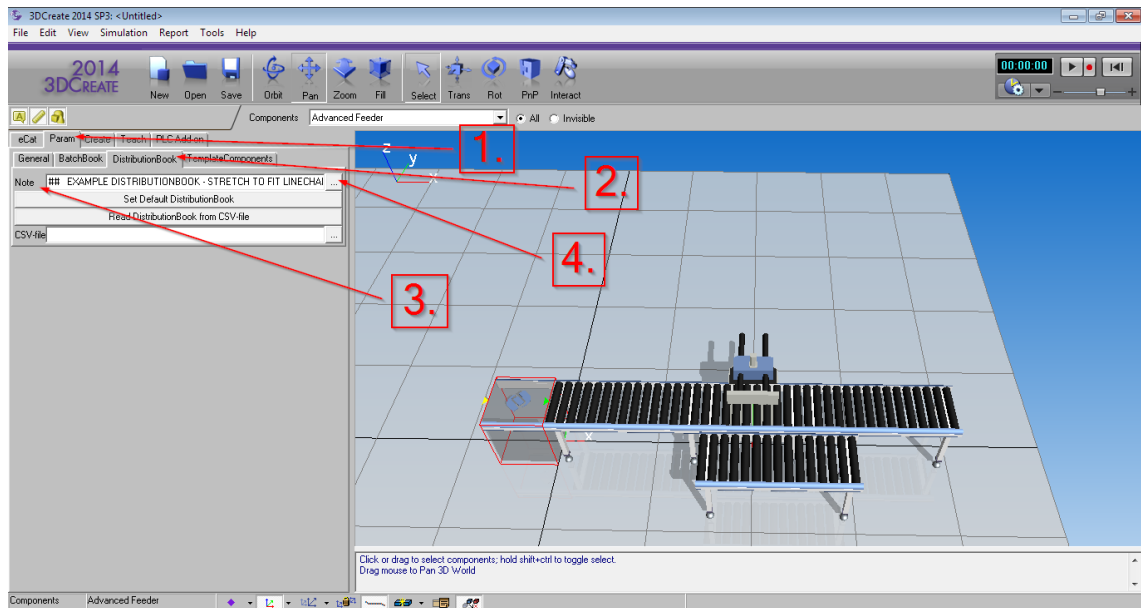
Picture 25: Connected conveyor line.

After this, testing is the next step. On the right top corner are the buttons for controlling the “playback” of the now made conveyor. Pressing play shows the Advanced Feeder “feeding” three different products to the conveyor line. The user can also see how the Pusher Conveyor should work. It will push the red “defective” products out of the main line (picture 26).



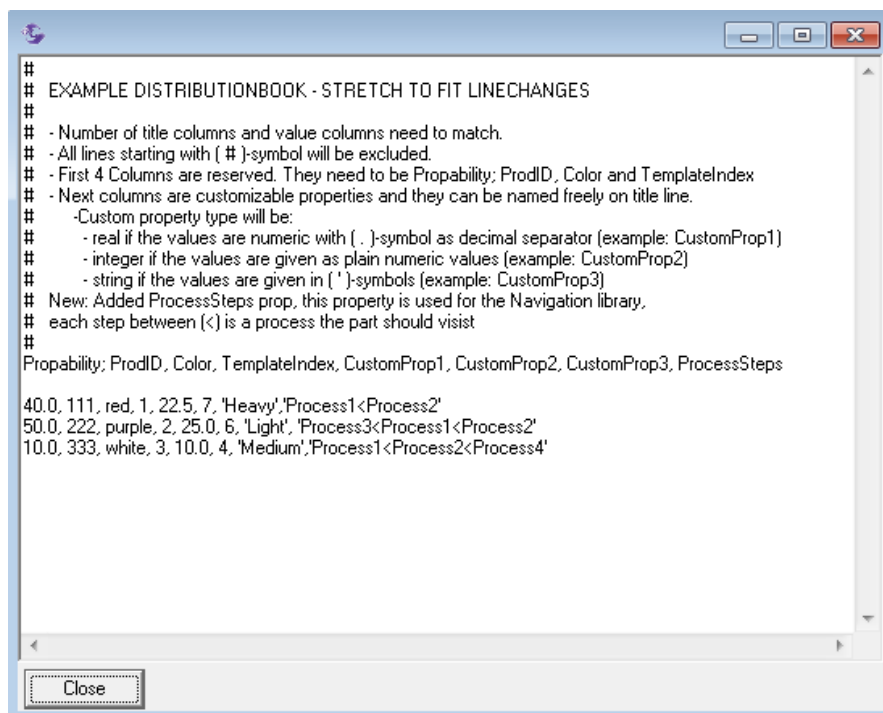
Picture 26: Pusher Conveyor working without logics.

Notice that this is only working at virtual time and the main point of this project is that the conveyor line works in real time with TwinCAT logic, which does the work of the pusher. Now that it is made certain that the conveyor works there will be a couple of changes for the Advanced Feeder. Now the Advanced Feeder is creating three products but it will be changed to so that 80% of the time it will create green products and 20% of the time it will create red ones. The Advanced Feeder has to be active and then choosing Param tab (1.) (beside the eCat tab) will give choices to change the product creation. Under Param tabs is another tab called "DistributionBook" (2.) and on the end of the "Note" line (3.) there is a three dotted button (4.) that gives the option to change the parameters (picture 27).



Picture 27: Finding where to change the distribution for the products.

Pressing the three dotted button opens up a small pop up (picture 28), which shows all the information of the three products that the Advanced Feeder is now creating.



Picture 28: Example Distribution Book.

In this pop up window, all the rows that are marked with “#” are just comments so they don’t affect the program. After that there are all the variables, that all can be changed to suit the need of the user. In the upper part, are the names of the variables and the lower three rows are the variables for those. Every variable has been separated with “,” and in the next part is important to be precise with the typing, to make sure error’s don’t prevent program from working properly. Replacing the variables with the next picture 29 makes the Advanced Feeder create green and red products.

```
Propability; ProdID, Color, TemplateIndex, CustomProp1, CustomProp2, CustomProp3, ProcessSteps
```

```
20.0, 111, red, 1, 22.5, 7, 'Heavy', 'Process1<Process2'
```

```
80.0, 222, green, 2, 25.0, 6, 'Light', 'Process3<Process1<Process2'
```

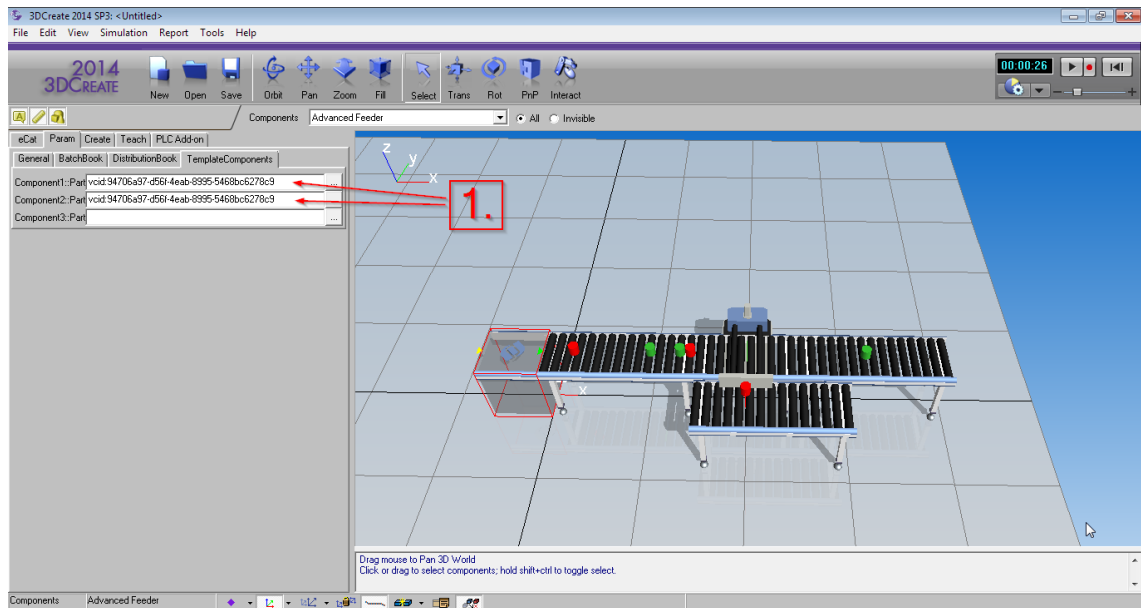
Picture 29: The changes to the variables.

The important parts are the first three: Probability, ProdID and Color. After changing the variables, there is still one more thing to do. We have to make sure, that Visual Components looks for the product from the right place, in this case eCat. The next tab beside the DistributionBook tab is TemplateComponents. On the Component1 and Component2 (1.) adding the next text finds the right colors and components. Delete the text from the Component3(picture 30).

```
vcid:94706a97-d56f-4eab-8995-5468bc6278c9
```

Picture 30: Vcid code for getting right colors and components.

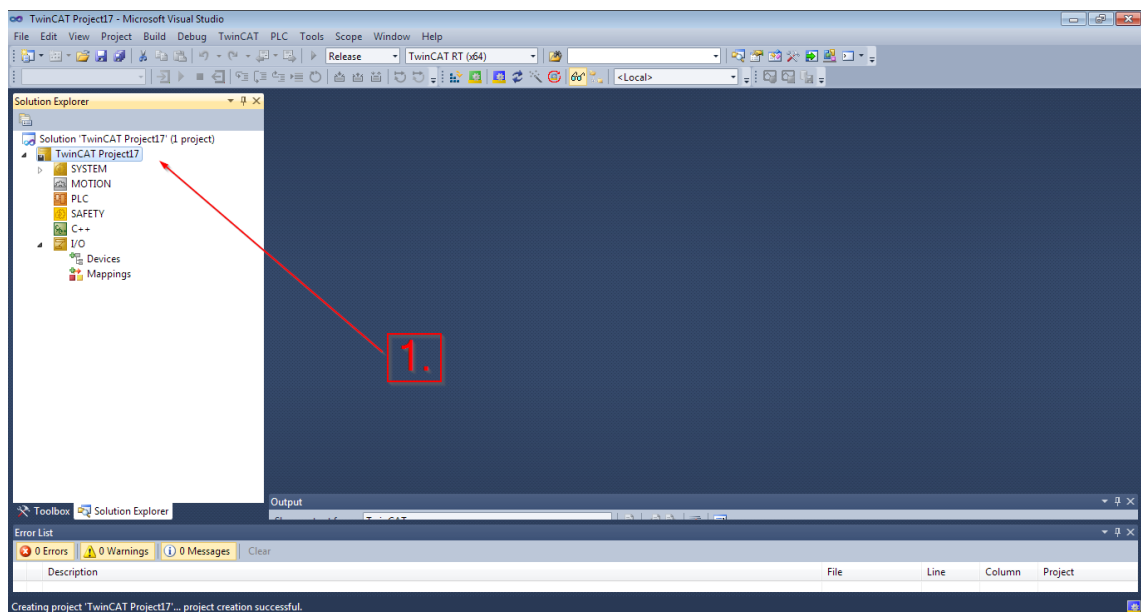
After clicking Enter and reseting simulation, every component from the library or local disk has a unique vcid (virtual components ID and if a user sets a vcid then it goes through all the components in eCat to looks for the component with the same vcid and grabs it. Now the Advanced Feeder creates the right color and product (picture 31).



Picture 31: Changing the component 1 and 2 search id for the eCat.

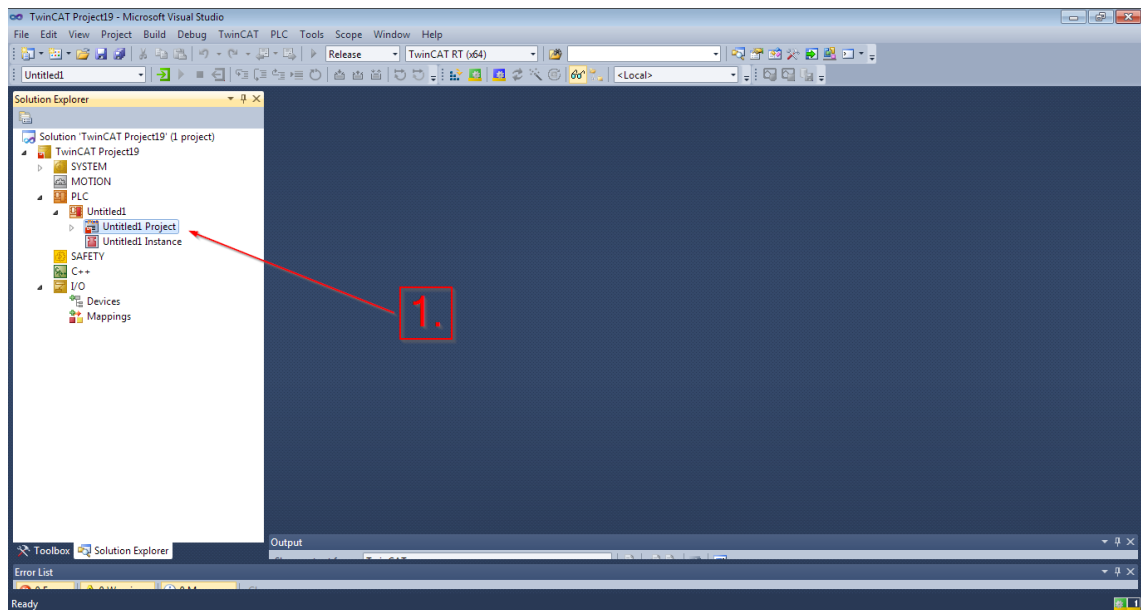
#### 4.2 TwinCAT logic simulation

In the picture 32, a new project is opened. When opening TwinCAT, you are greeted with an empty screen. Choose File from the left upper corner -> new -> project. Choose the only option that is TwinCAT XAE Project. After opening that the new project is available on the left side in the Solution Explorer (1.).



Picture 32: Opening new project.

Next part is opening a new PLC project (picture 33). By clicking right mouse button on the PLC it shows option for “Add new item” and “Add existing item”. Pressing “Add new item” opens a new window and after choosing “Standard PLC Project” it adds “Untitled1” and under that is “Untitled1 Project” (1.) within the PLC directory.



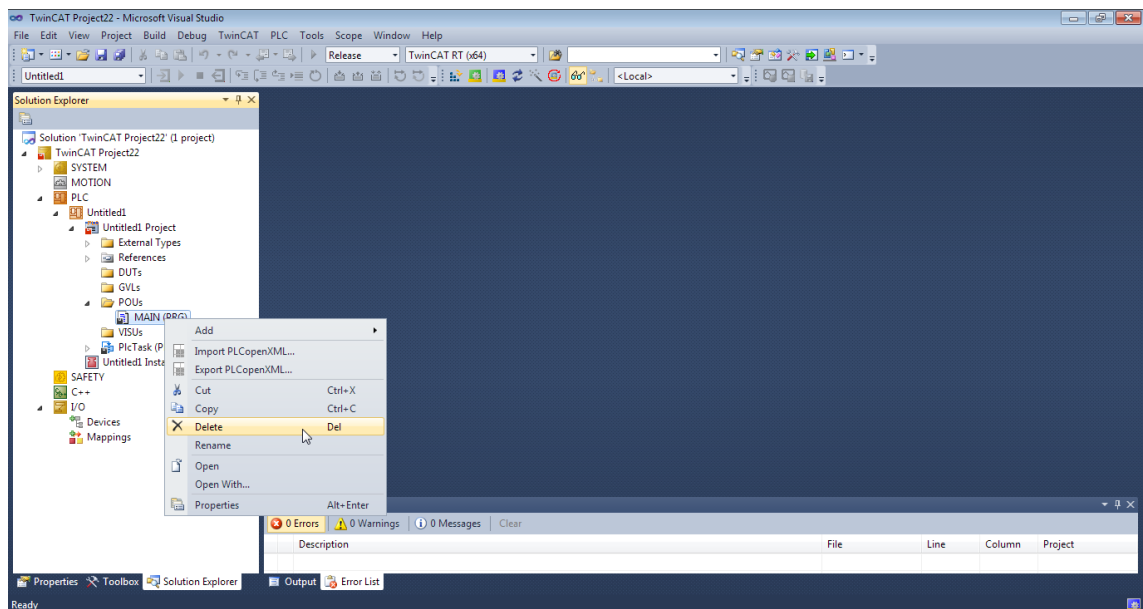
Picture 33: PLC program open.

From inside the project opens up choices to design 4 different simulations:

- DUT (Data Type Unit) = User can define own datatypes: structures, enumeration types and references can be created as Data Type Units in a DUT editor (Beckhoff 2016).
- GVL (Global Variable List) = GVLs are used to declare global variables. If GVL exists, the variables will be affect the whole project and not just a certain part (Beckhoff 2016).
- POU (Program Organization Unit) = Functions, function blocks and programs are POUs, which can be supplemented by actions. (Beckhoff 2016).

- VISU (Visualization) = For making the project more visualized like adding virtual buttons and gauges for showing information (Beckhoff 2016).

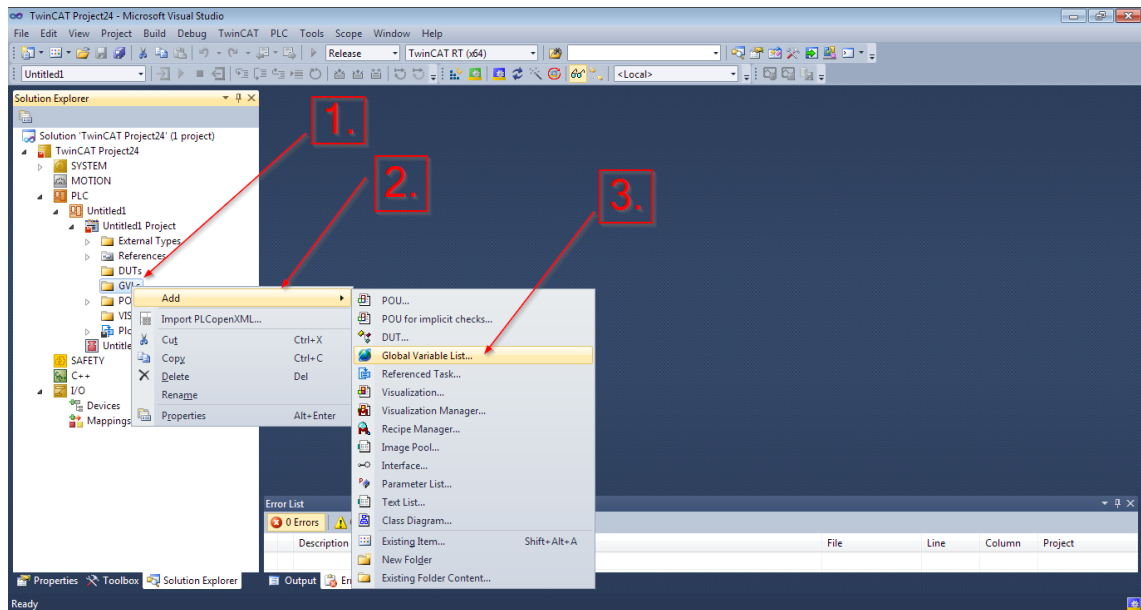
GVL and POU will be more essential for this project. First step is to make the PLC program that will be controlling the pusher in Visual Components. There will be three parts: MAIN (PRG), FB\_Cylinder and GVL. The MAIN (PRG) is the main program that makes the first and final command for PusherConveyor, FB\_Cylinder handles the feedback delay and cycle counter and GVL handles the connection between 3D Create and TwinCAT. In the picture 34, the MAIN (PRG), open the POU file and delete the existing MAIN (PRG).



Picture 34: Deleting old existing MAIN (PRG).

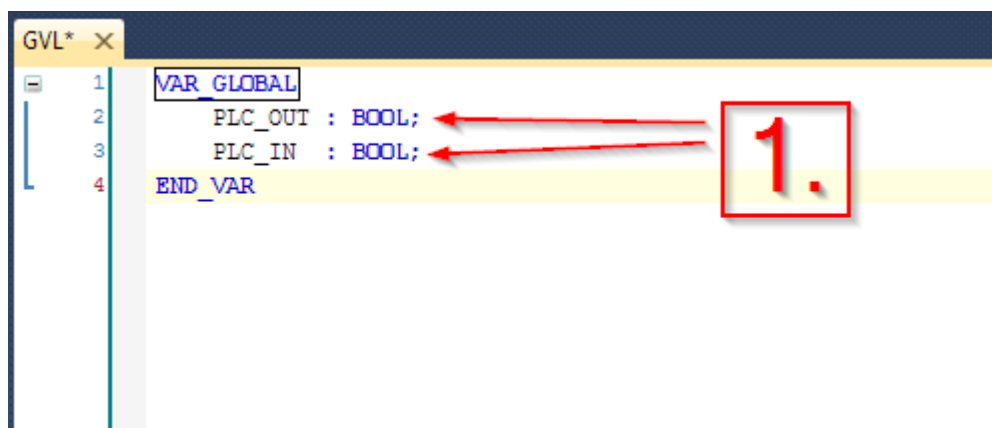
Next parts will be about adding variables to three different places: to GVL, to MAIN(PRG) and to FB\_Cylinder (function block). First job is to add GVL (Global Variables), which are visible throughout the whole PLC program and also their purpose is to connect TwinCAT to the Visual Components (more on that later). As shown in picture 35, right click the “GVLs” file (1.) under the “Untitled Project”, then choose “Add” (2.) and from the list “Global Variable List” (3.) and keep it named as “GVL”.





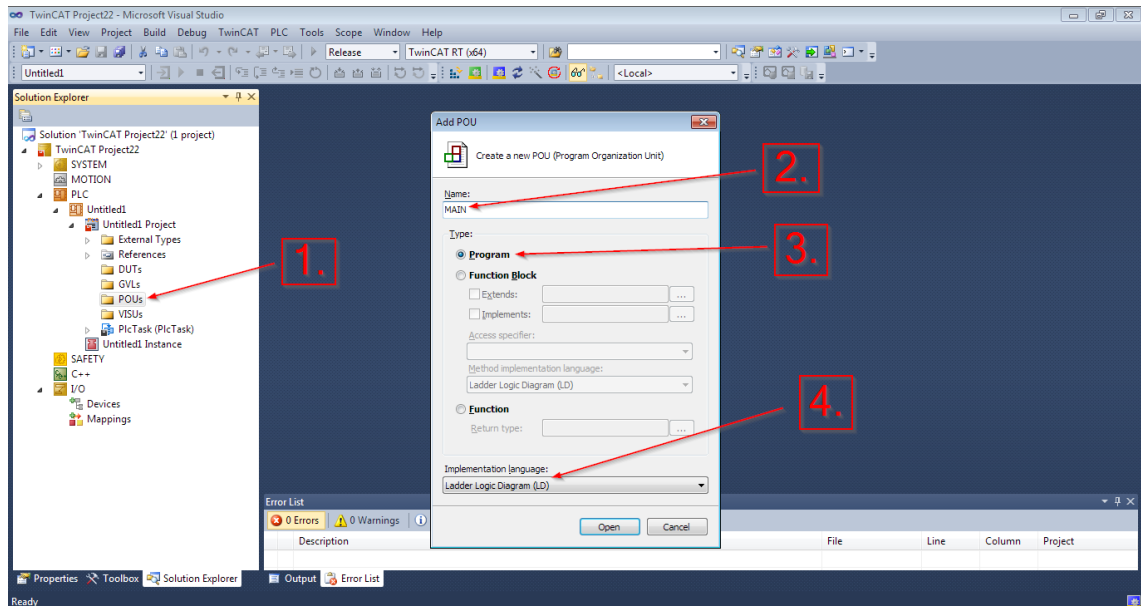
Picture 35: Adding Global Variable List.

Now that the GVL tab is open there will be two added variables: PLC out and PLC in. PLC\_IN signifies the cylinder is waiting request from the Visual Components and PLC\_OUT means that cylinder has pushed the defective product from the conveyor line (picture 36).



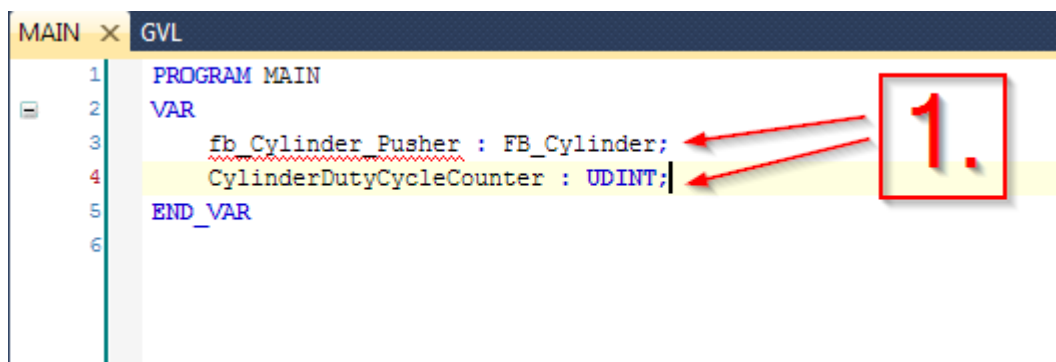
Picture 36: PLC\_out and PLC\_IN global variables added.

Picture 37 shows the next part: adding MAIN(PRG). Right click POU's (1.) and add a new "POU" called MAIN (2.) but in this case keep the type as a "Program" (3.) and change the implementation language to Function Block Diagram (4.)



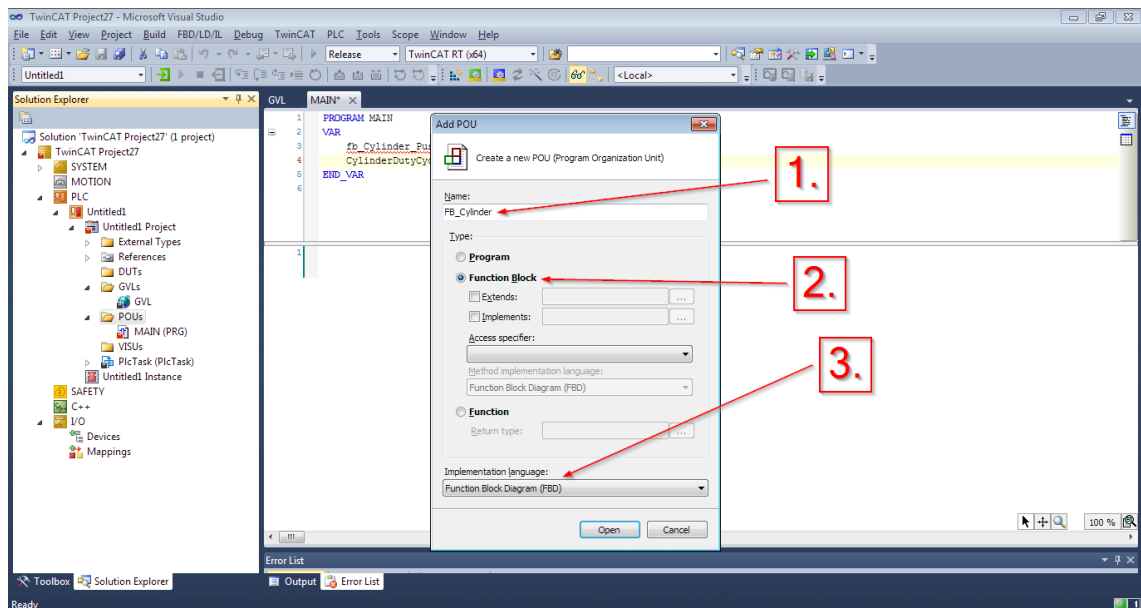
Picture 37: Adding MAIN (PRG) with LD.

Now that the MAIN (PRG) is open, the variables will be added to the upper part of TwinCAT. The red line on the first variable is not taken account at this time. In this case, "UDINT" is chosen, that there is enough space for the cycle counter (picture 38).



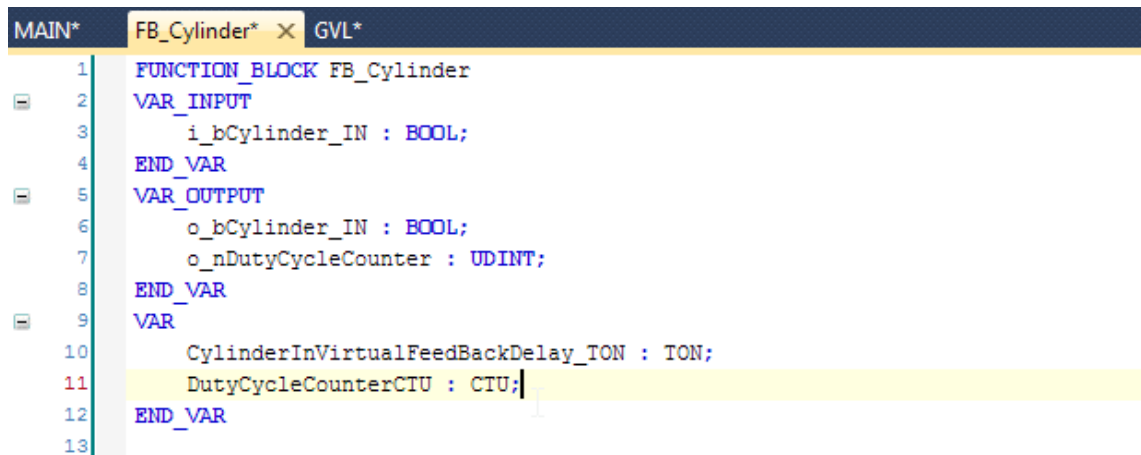
Picture 38: MAIN (PRG) variables added.

Next part is about adding variables to FB\_Cylinder which is a function block program. Same as adding the MAIN (PRG) but with a slight difference. Right click POU's and choose "add", which shows a new list with POU. Clicking that opens a new window where the name would be FB\_Cylinder (1.) but in this case it is a function block (2.) and not program as with MAIN (PRG). The implementation language is Function Block Diagram (3.) (picture 39).



Picture 39: Adding FB\_Cylinder function block.

Inserting variables for FB\_Cylinder shows how to write ST in more readable form. The user has to think how easy the ST is to read by another user, that is why there are certain rules. The variables for FB\_Cylinder function block are shown in picture 40:



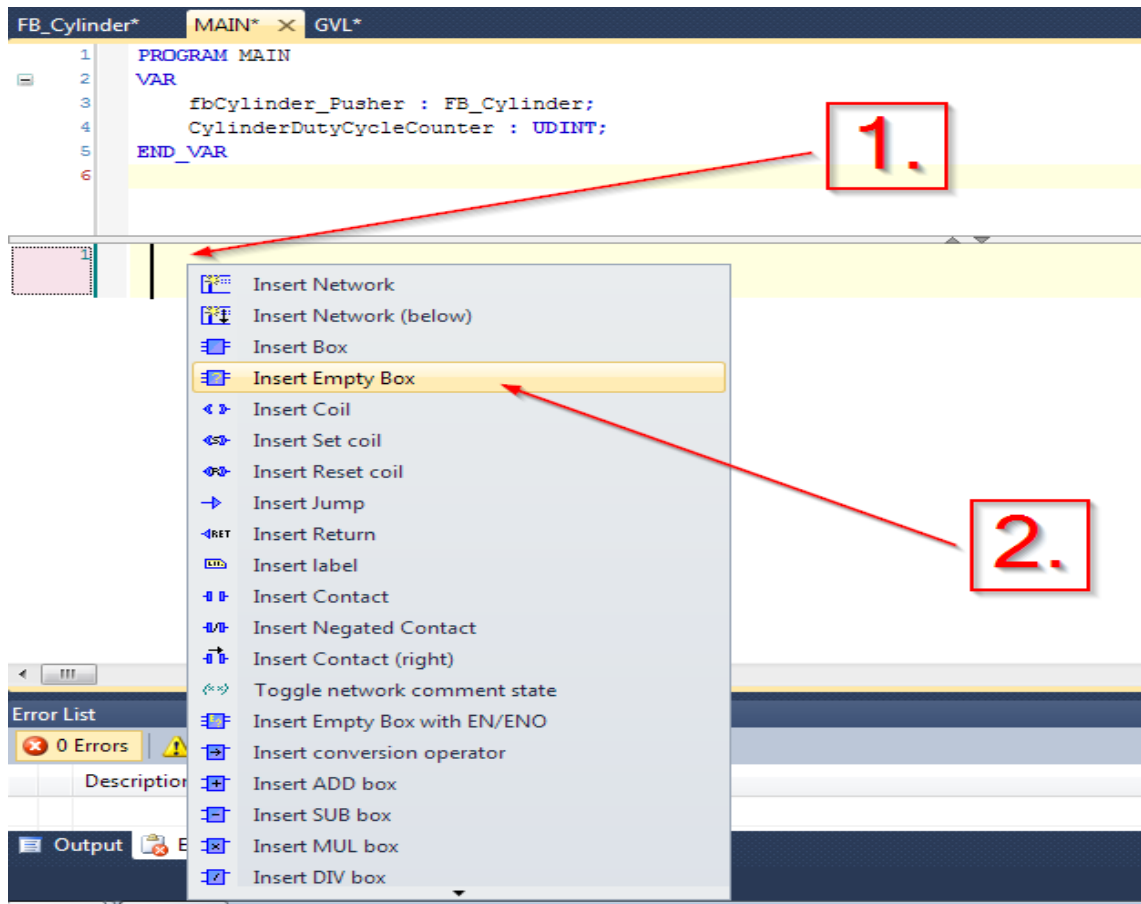
```

1  FUNCTION_BLOCK FB_Cylinder
2  VAR_INPUT
3      i_bCylinder_IN : BOOL;
4  END_VAR
5  VAR_OUTPUT
6      o_bCylinder_IN : BOOL;
7      o_nDutyCycleCounter : UDINT;
8  END_VAR
9  VAR
10     CylinderInVirtualFeedBackDelay_TON : TON;
11     DutyCycleCounterCTU : CTU;
12 END_VAR
13

```

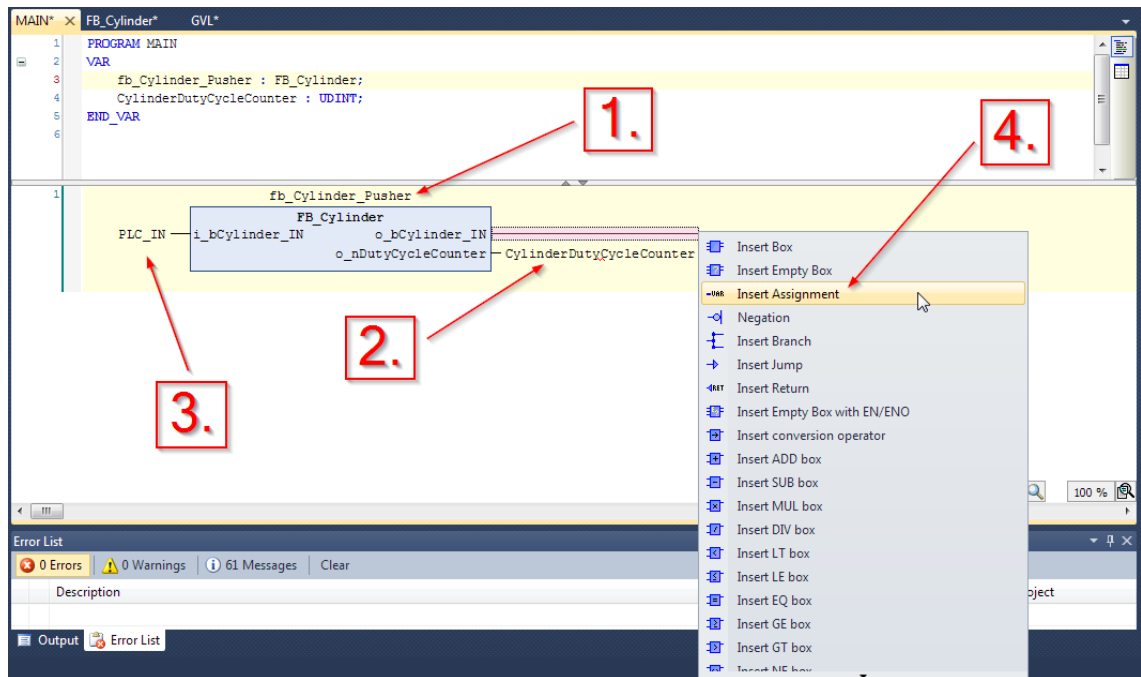
Picture 40: Variables for the FB\_Cylinder.

After adding variables for all three parts the next up is adding function blocks to Main (PRG) and FB\_Cylinder (picture 41). In the MAIN (PRG) the lower screen is for FBs, where has already been added one network . Right clicking (1.) the network line opens a list of choices that can be added to the network, choosing “Insert Empty Box” (2.) adds a function block that is modifiable.



Picture 41: Adding an empty function block.

Now that the function block has been added the variables (connections) can be added to the block (picture 42). Writing variable “fb\_Cylinder\_Pusher” (1.) on top of the empty box adds automatically “i\_bCylinder\_IN” to input and to the output of a block, o\_bCylinder\_IN. Also “o\_nDutyCycleCounter” is added automatically for the calculator. “CylinderCycleCycleCounter” variable will be written (2.) to the “o\_nDutyCycleCounter” output. “PLC\_IN” will be added (3.) to the input contact (left side) and for the PLC\_OUT an assignment has to be added, which is a place for a variable. Right clicking the end of the line of the output line opens up a list of what to add to the output. “Insert Assignment” adds an output box where to write “PLC\_OUT” (4.).

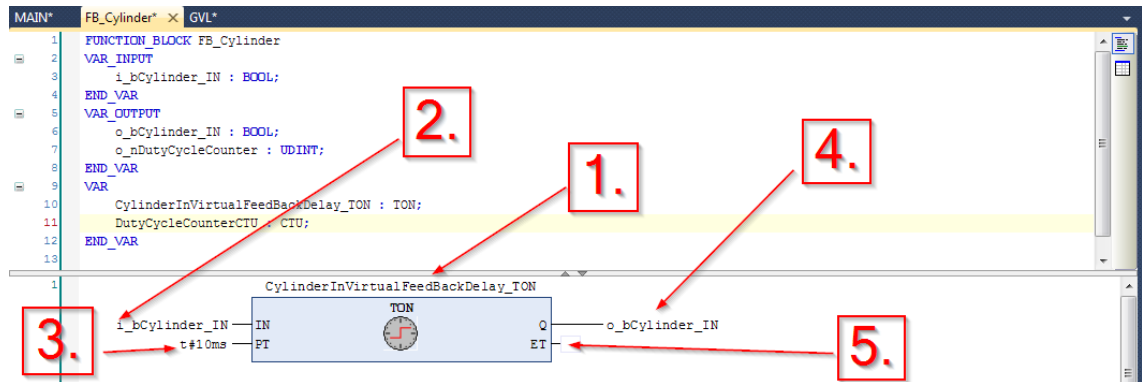


Picture 42: Adding the variables to the function block of the MAIN (PRG).

On the FB\_Cylinder a Count up (CTU)- and Timer on delay (TON) function blocks will be added. The CTU will count all the rejected products from the conveyor line and TON will be added for the delay that has to be usually taken into account in real life situations. Firstly a new network will be added because there are two different function blocks. There is a choice also to have many different function blocks within one network but in this project case there are two. Right clicking the network one line opens, like last time, the list of choices that can be added. This time “Insert Network Below” will be chosen. The first network will have the TON function block. Now add again an empty box on network one, like on the MAIN (PRG).

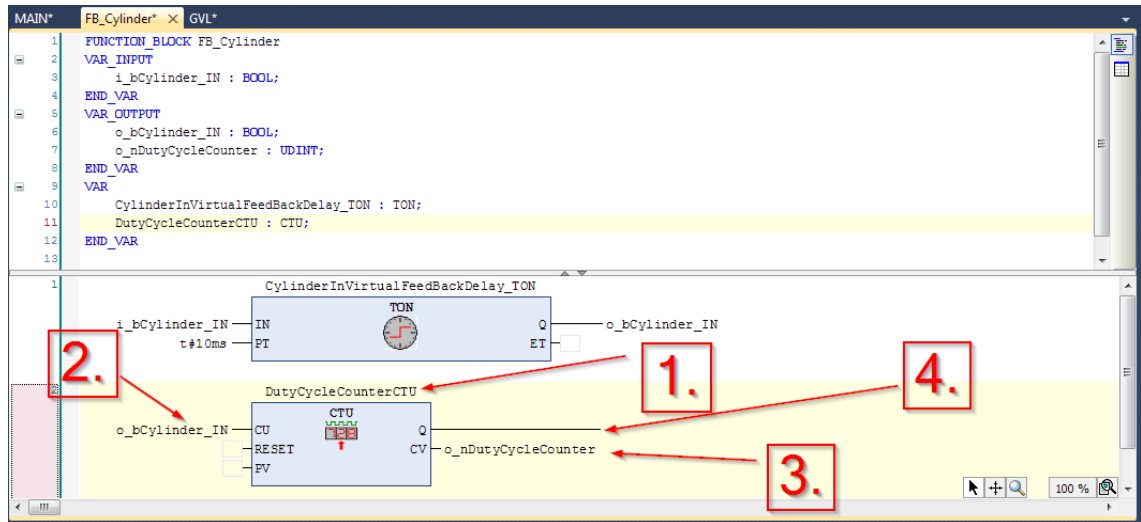
Adding (picture 43) “CylinderInVirtualFeedBackDelay\_TON” (1.) as the name of the block changes it automatically to TON function block, because on the variables it had already been decided to be TON. “i\_bCylinder\_IN” goes again to the input side (2.) of the TON block. Under that is PT, which stands for the time that the delay is. In this case “t#10ms” is added there (3.). On the output, is “q”, that is output of the timer and “ET” that is the current value of timer. ON the output “q”

there has to be again added “insert assignment” for the “o\_bCylinder\_IN” (4.), from the “ET” the question marks will be deleted (5.) because if they are left there TwinCAT responds that there should be something added.



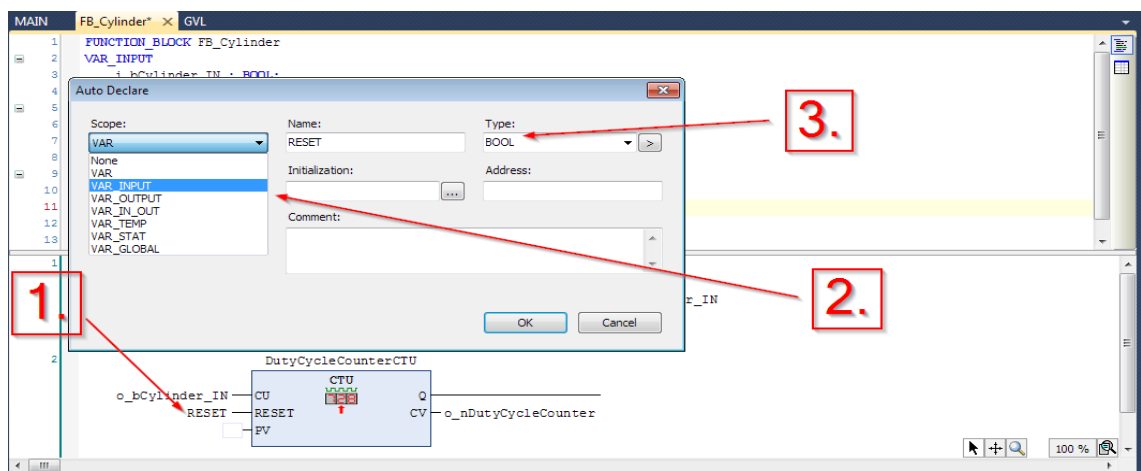
Picture 43: TON function block added with variables.

Second function block for the 2nd network is CTU. It starts again with adding an empty box for the 2nd network. This time the name of the block is “DutyCycleCounterCTU”. There are three inputs to CTU function block: CU, RESET (to add a reset function, more on that later) and PV (a counter limit). At this moment “o\_bCylinder\_IN” will be added to the CU input and RESET and PV will be left empty. On the output side there is Q (counter reached the limit) and CV (current counter value). The Q will be left empty because there isn’t a limit and on CV “o\_nDutyCycleCounter” variable will be added. This sends the information to the MAIN (PRG) (picture 44).



Picture 44: Variables added to the CTU function block.

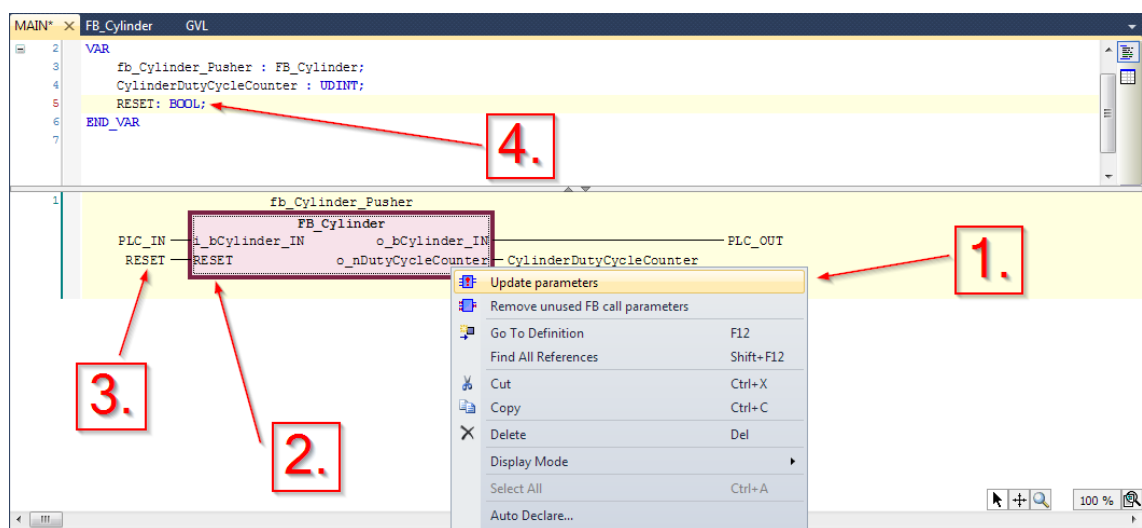
Lastly a reset function will be added (picture 45), so that the user can stop the counter to the rejected products and start calculating from zero again. This time the variable for reset will be added in another way. The user can actually right away write the variable into the block and decide what kind of type the variable is and it will be added to the variable list automatically. Writing “RESET” on the RESET input of the CTU function block and pressing Enter opens a window where the type is decided. From the new window under the “scope:” a “VAR\_INPUT” will be chosen, because this RESET will come as an input to the CTU block and the type will be kept as a “Bool”.



Picture 45: Adding RESET to the CTU function block.



So that the reset function works it has to be also added to the MAIN (PRG) (picture 46). As it is apparent there are no more input spots for the function block so it must be added. Now that the RESET has been added to the CTU and to the variables in the last chapter, it is possible to update the function block parameters. Right clicking the function block opens up a list and the top most option is “update parameters” (1.). This adds the RESET to the function block (2.). After adding the new input slot, writing RESET to it (3.) and this time keeping the “scope” as a “VAR” and keeping the type as a “BOOL”, the RESET variable is added (4.). Now the function is complete.



Picture 46: Adding RESET input and RESET variable to the MAIN (PRG).

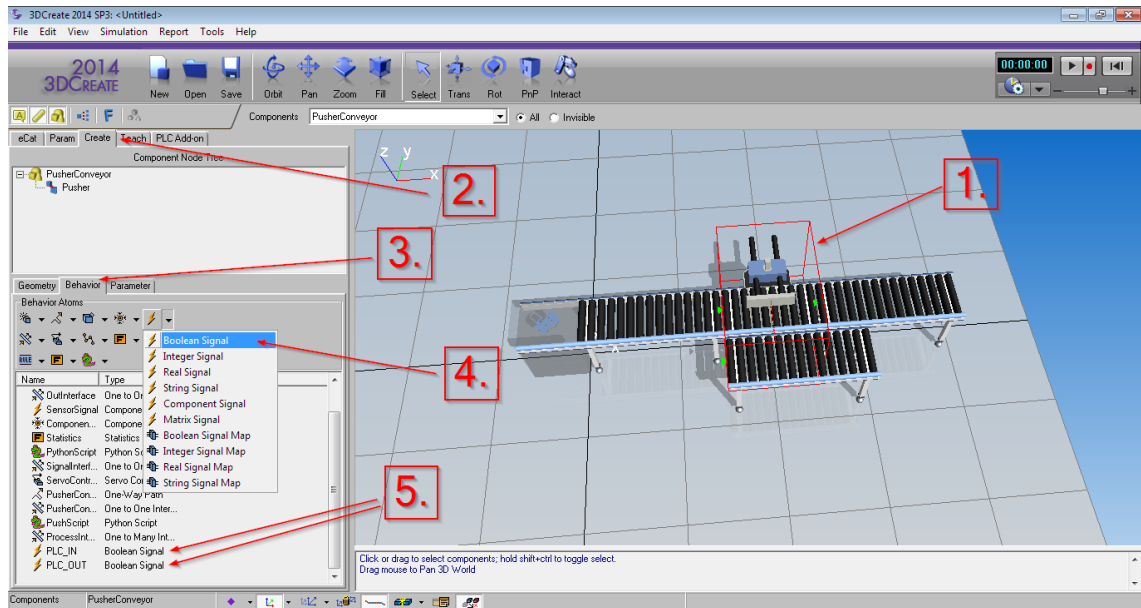
### 4.3 Connection between Visual Components and TwinCAT

Now that the automatic conveyor has been built and TwinCAT logics have been made, the connection between the two programs will be generated. A small change will be made in the PusherConveyor Python code.

#### 4.3.1 Adding PLC\_IN and PLC\_OUT signals

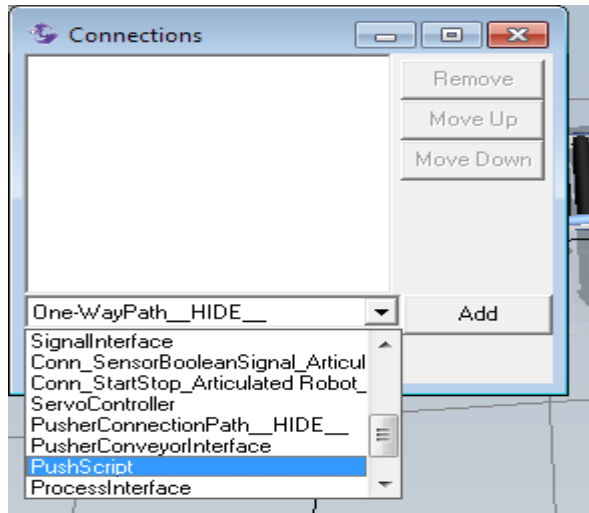
In this part, two signals will be added to PusherConveyor: PLC\_IN and PLC\_OUT (picture 47), these boolean signal transfers will make the contact with the Python

script that will be modified in the later chapter. After making PusherConveyor active (1.), the Pusher Signal adding can be found under the Create tab (2.) and from there Behavior tab (3.). From Behavior Atoms, they are the “lightning” icon. Inside the drop down list there is a “Boolean Signal” (4.). Two Boolean Signals will be added and they will be named in the pop up window PLC\_IN and PLC\_OUT (5.).



Picture 47: Adding PLC\_OUT and PLC\_IN signal.

Following the adding of the signals is making them connected to “PushScript” (picture 48). Double clicking the signals opens up a window that’s showing no connections. The “Automatic Reset” is automatically on, but in this case it will be “untabbed”, because the simulation doesn’t need a reset. Pressing the “...” button opens up drop down list which asks in where are the signals connected. Choosing “PushScript” and pressing “Add” moves it to the list of connections.

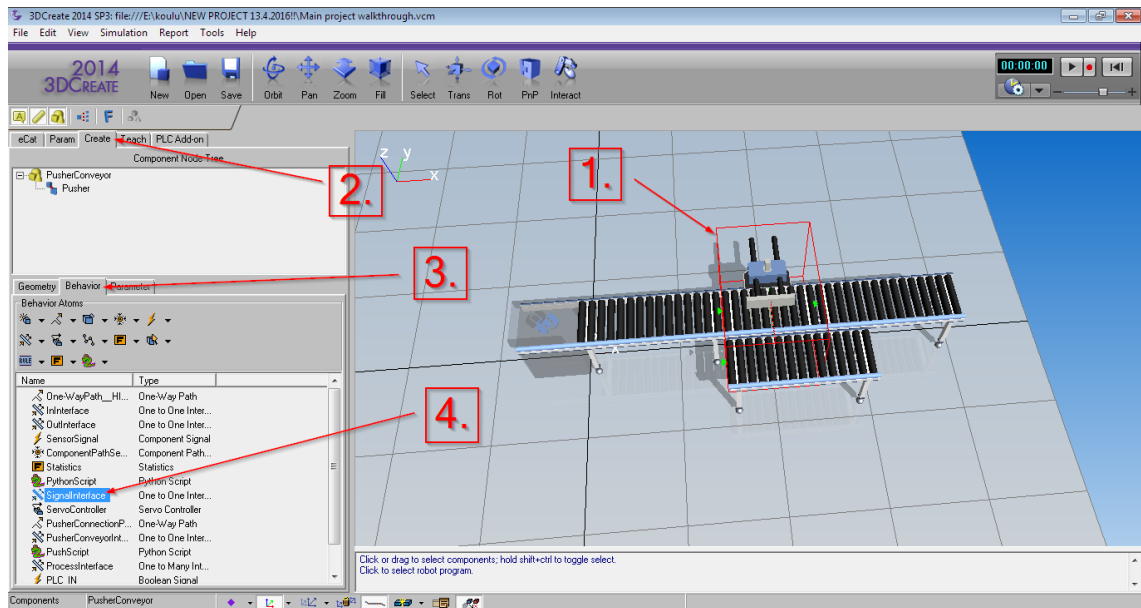


Picture 48: Connecting the two signals to the PushScript

Once this has been done to the both of the signals this chapter is done.

#### 4.3.2 Adding signal interface

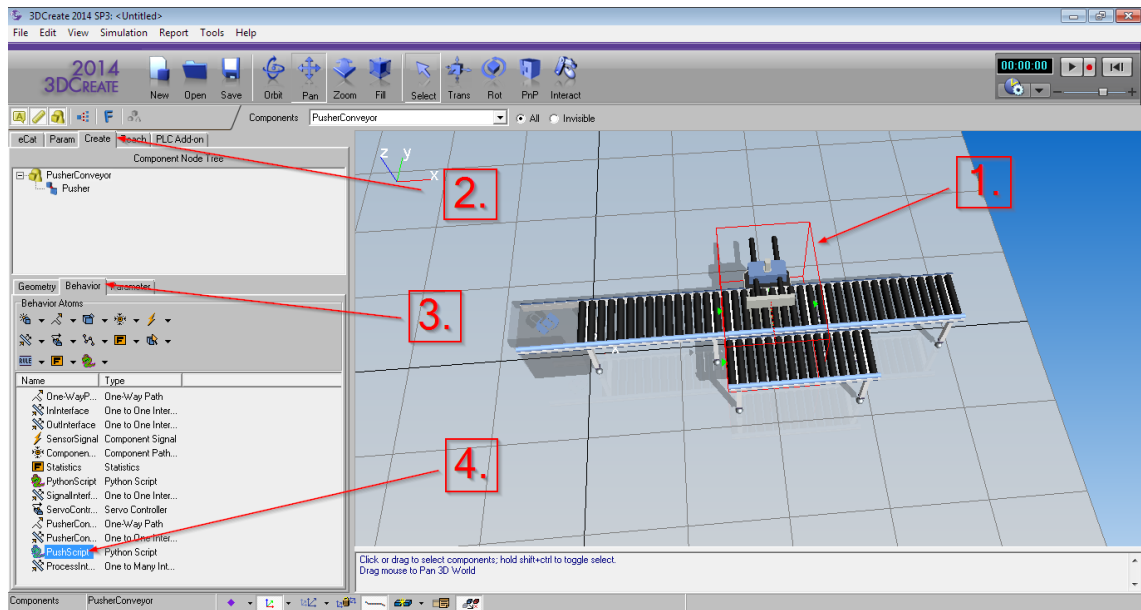
In this section, the signal interface will be assigned (picture 49). This means that the PLC\_IN and PLC\_OUT will be assigned as an input and output signals inside 3D Create. Making PusherConveyor active and going to the same place with the “PushScript”, Create tab → Behavior tab and from there SignalInterface.



Picture 49: Finding Signal Interface.

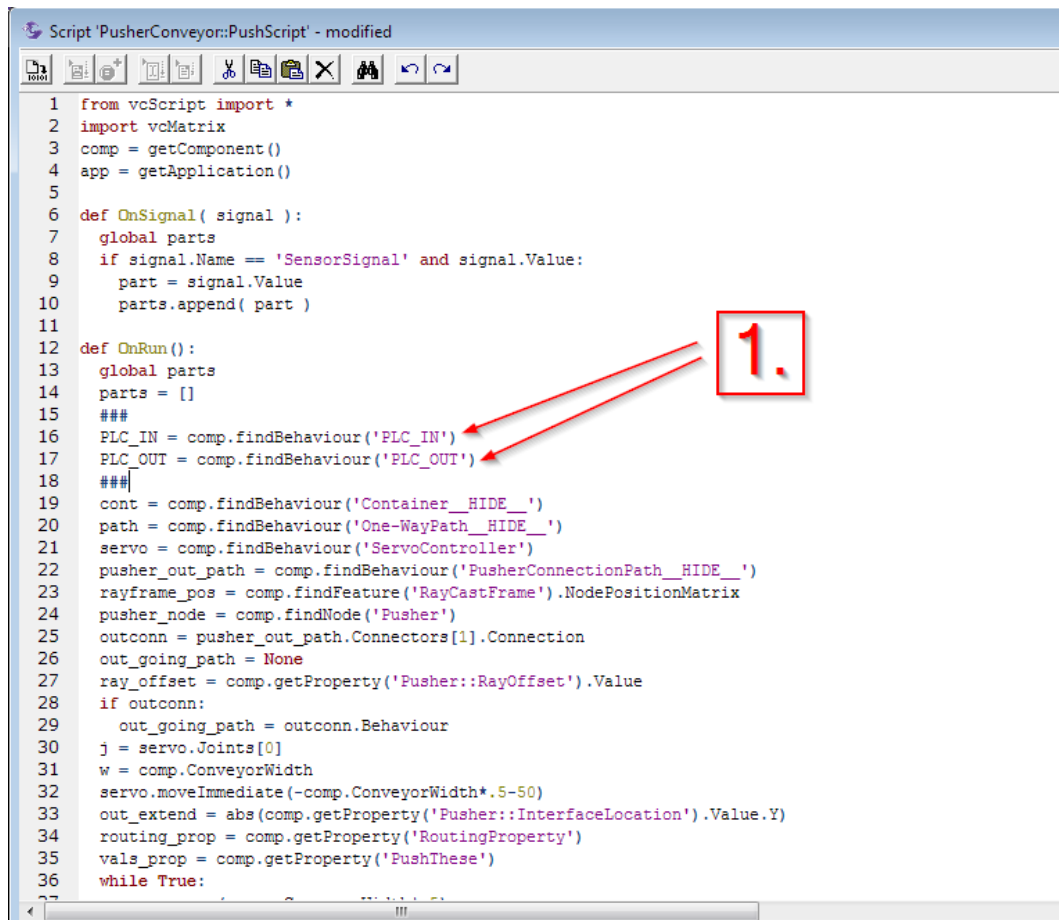
This opens a new window called “properties of SignalInterface” (picture 50). In this window, there is a “Sections” part with three dot button. It opens a window, which has “Interface Sections” on the left and “Section Fields” on the right. From the “Interface Sections”, the “PLC\_Section” will be renamed to just “PLC” (1.). From the PLCs Section Fields the input and output will be specified. When input is active (2.), from the “Current Field” the Signal will be paired with PLC\_OUT (3.). The output will be paired with PLC\_IN (4.).





Picture 51: Finding the PushScript of the PusherConveyor.

The PushScript (picture 52) looks intimidating initially when it opens, but there isn't that much that will be changed. First the variables for the PLC\_IN and PLC\_OUT are made, this will make the connection between the signals that were made in last chapter. The variables are added under the "Def OnRun" command line. The "#" marks are a comment so they don't affect the program.



```

1  from vcScript import *
2  import vcMatrix
3  comp = getComponent()
4  app = getApplication()
5
6  def OnSignal( signal ):
7      global parts
8      if signal.Name == 'SensorSignal' and signal.Value:
9          part = signal.Value
10         parts.append( part )
11
12  def OnRun():
13      global parts
14      parts = []
15      ###
16      PLC_IN = comp.findBehaviour('PLC_IN')
17      PLC_OUT = comp.findBehaviour('PLC_OUT')
18      ###
19      cont = comp.findBehaviour('Container_HIDE_')
20      path = comp.findBehaviour('One-WayPath_HIDE_')
21      servo = comp.findBehaviour('ServoController')
22      pusher_out_path = comp.findBehaviour('PusherConnectionPath_HIDE_')
23      rayframe_pos = comp.findFeature('RayCastFrame').NodePositionMatrix
24      pusher_node = comp.findNode('Pusher')
25      outconn = pusher_out_path.Connectors[1].Connection
26      out_going_path = None
27      ray_offset = comp.getProperty('Pusher::RayOffset').Value
28      if outconn:
29          out_going_path = outconn.Behaviour
30      j = servo.Joints[0]
31      w = comp.ConveyorWidth
32      servo.moveImmediate(-comp.ConveyorWidth*.5-50)
33      out_extend = abs(comp.getProperty('Pusher::InterfaceLocation').Value.Y)
34      routing_prop = comp.getProperty('RoutingProperty')
35      vals_prop = comp.getProperty('PushThese')
36      while True:

```

Picture 52: PLC\_IN and PLC\_OUT variables added to the Python code.


Now the main code part will be added (picture 53). Notice the right intendation after “if” command. The next code makes the next effects in this order:

- It waits for the signal.
- It receives the signal.
- It evaluates the signal.
- It triggers a signal.

```

38 path.Enabled = True
39 condition(lambda: parts)
40 part = parts.pop(0)
41 if part.getProperty(routing_prop.Value.strip()) and str(part.getProperty(routing_prop.Value.strip()).Value) :
42     ###
43     PLC_IN.signal(True)
44     triggerCondition(lambda: getTrigger() == PLC_OUT and PLC_OUT.Value == True)
45     PLC_IN.signal(False)
46     ###
47     part.update()
48     path.Enabled = False
49     pusher_node.update()
50     ray_pos = pusher_node.WorldPositionMatrix * rayframe_pos
51     mtx = vcMatrix.new(ray_pos)

```



Picture 53: Python program added.

It is important to make sure that the code is written the exact way. Last thing to do is a "compile code" button in the upper left corner, which updates the code.

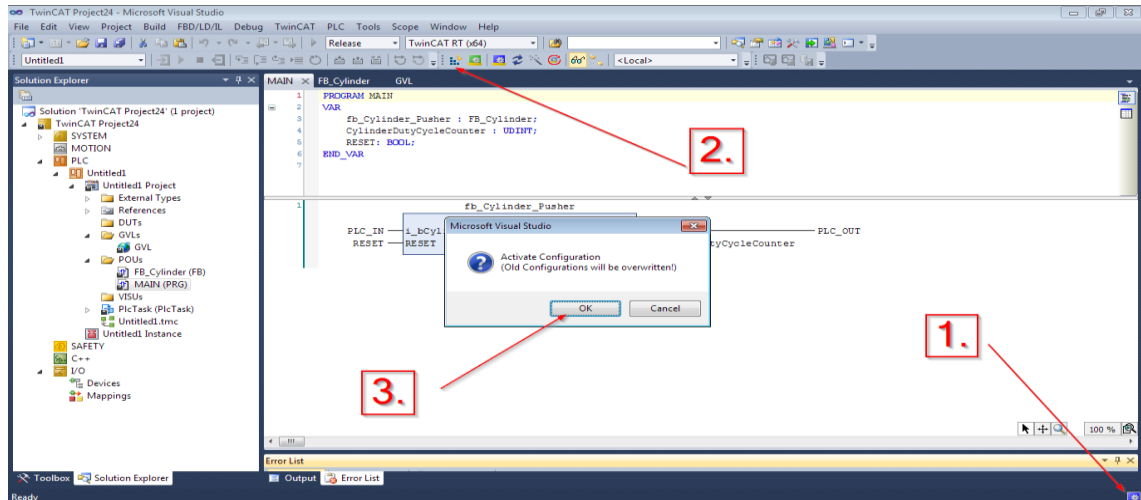
#### 4.4 Connection between Visual Components and TwinCAT

In this chapter the last part of the project is made. The connection will be made with the PLC add-on for the Visual Components 3D Create. Before everything else TwinCAT program the configuration has to be activated.

##### 4.4.1 Activating configuration

In this part (picture 54) the configuration will be activated. On the lower right corner, a gear icon can be seen which shows if the program is ready to go or if it is waiting (1.). The button for the activating the configuration can be found on top of the variable list in the tool bar (2.). Pressing it opens a new window, which asks if the user wants to activate the configuration. Pressing "OK" moves to the next step (3.).



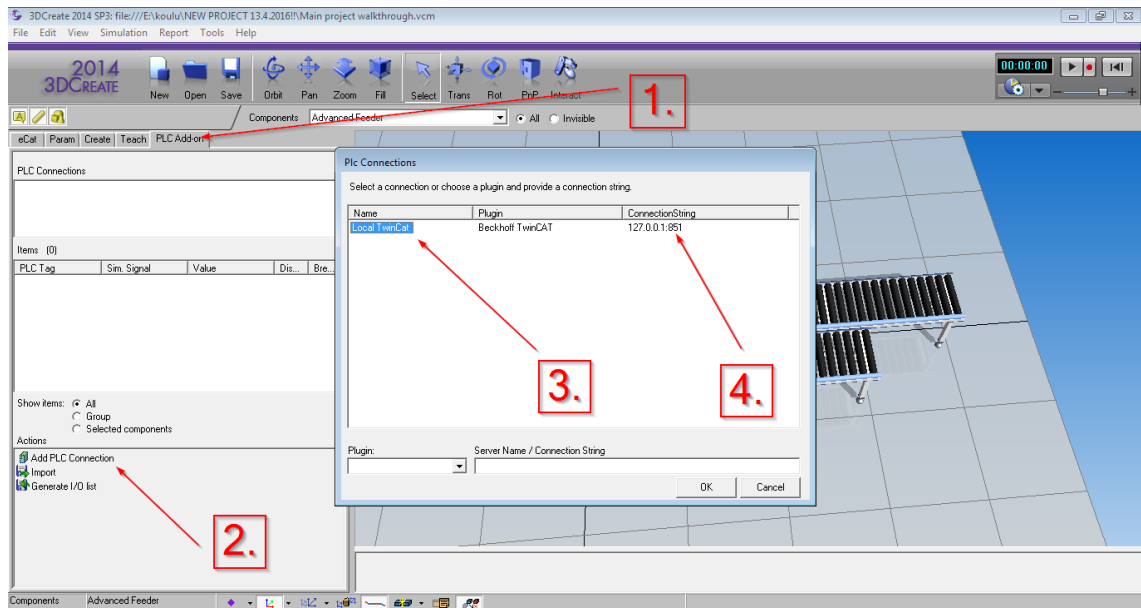


Picture 54: Activating configuration on the TwinCAT.

After that the program asks if the user wants to create a temporary licence for the program, agreeing on that shows a five digit code that is typed to the free space and now the user has a free one-week licence for the program. This can be continued indefinitely. Continuing from that, the program enquires if the program should be restarted in run mode. After allowing that the program re-starts in to the run mode, which the user can deduce from the gear icon that is rolling around and changed to green. Now that TwinCAT is in Run mode, it has to be logged in. In this part, it asks if it should add “port 851”, this is the basic port that TwinCAT creates for the logics. TwinCAT is now operational.

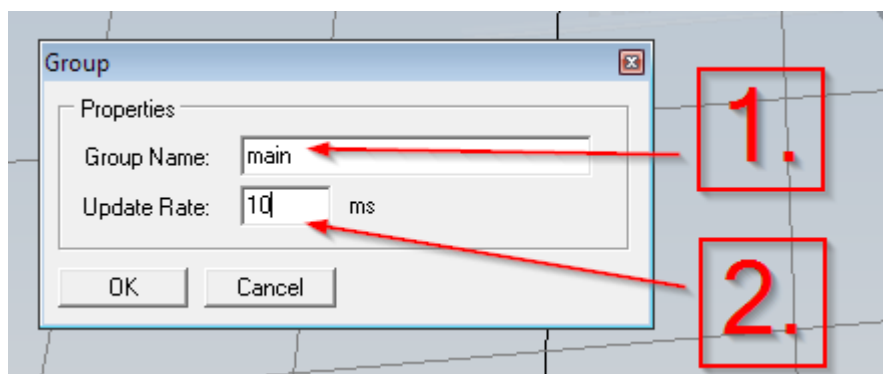
#### 4.4.2 Adding connection part inside 3D create

Now that TwinCAT is ready for information, the connection will be made with the PLC add-on in 3D Create (picture 55). The PLC add-on tab can be found on the main tabs last with eCat etc (1.). From there in the “Actions” is a “Add PLC Connection” icon (2.), double clicking that opens up a window where the connection is chosen. There could be many different logic programs shown in this window, but at this time it shows TwinCAT (3.). The connection string shows the port 851 that was created in TwinCAT (4.). Select the “Local TwinCAT” and press “OK”.



Picture 55: Adding TwinCAT connection to 3D Create.

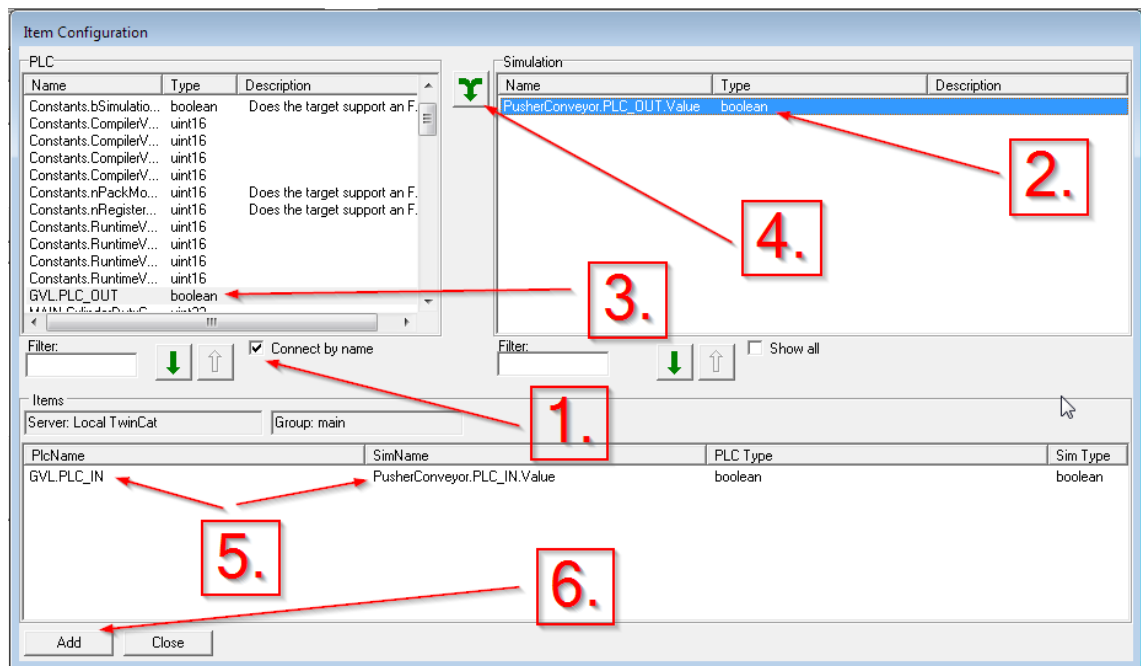
This opens a new pop up (picture 56), which asks the name of the group and update rate. The group will be named “main” and update rate will be changed to 10ms, because 1000ms is more optimal.



Picture 56: Adding main file in PLC add-on.

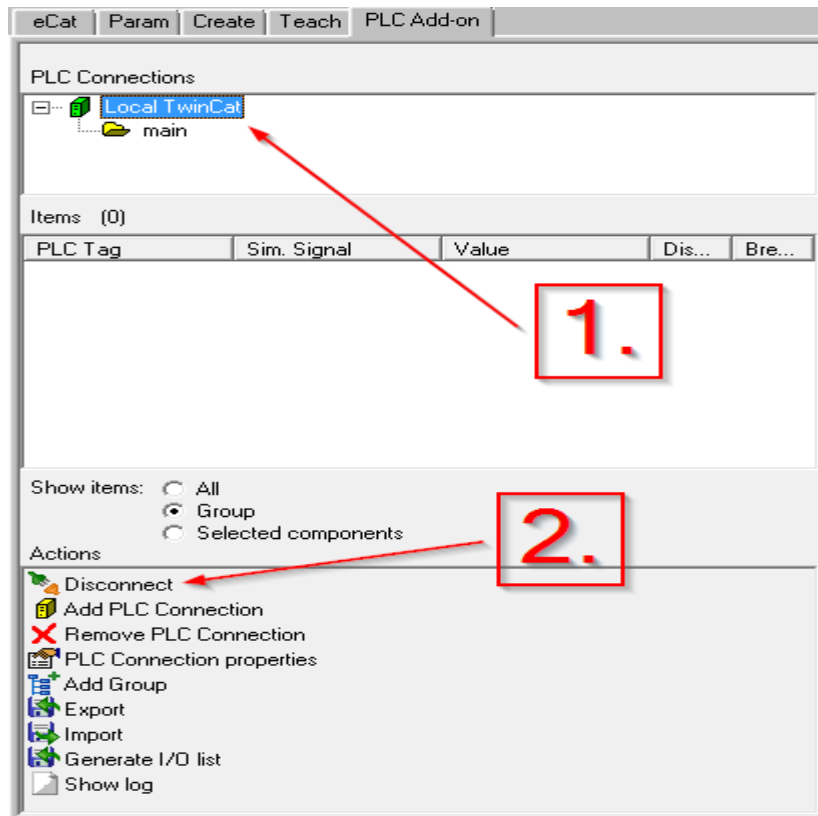
Now the folder for the connections is added and it asks to pair the signals between 3D Create and TwinCAT (picture 57). On the right are the signals that were added in the Signal Interface chapter and on the left are the connections from the PLC. To find the right connections easily “Connect by name” has to be filled (1.) and

PLC\_IN (2.) or PLC\_OUT is selected, it right away shows the right connection (3.). After this the “connect” button (4.) has to be pushed and it moves the paired signals to the lower screen (5.). Finally the “Add” button must be pushed or it doesn’t connect them (6.).



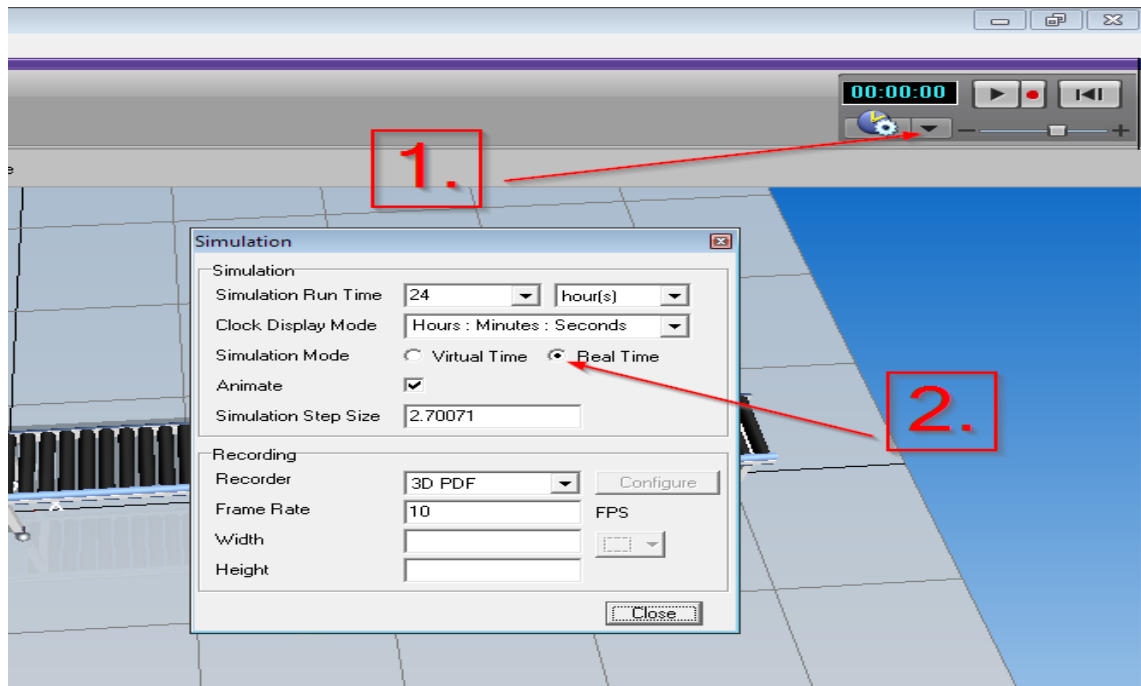
Picture 57: Pairing the 3D Create signals with the PLC signals.

The user can control the connection of the 3D Create with a simple button from the PLC add-on tab (picture 58). Notice that the “Local TwinCAT” has to be selected. In this case it automatically connects.



Picture 58: The connect button.

Finally, the simulation mode has to be changed from virtual time to real time (picture 59). It will be changed from the Play tools simulation speed settings (1.) on the upper right corner. From the new opened window virtual time → real time. It has to be on real time because the logics control the 3D Create simulation.



Picture 59: Changing simulation mode from virtual to real time.

## 4.5 Testing the project

Everything is now ready to test the automatic conveyor line, from the “Play tools” pressing play activates the conveyor line and the Advanced Feeder starts to create products randomly in 20%/80% ratio. The user can increase the speed for a faster or slower for the rejected products, but not too fast because 3D Create might not be fast enough to show the animation of the pusher. The Pusher Conveyor should remove the red products from the conveyor.

### 4.5.1 Walkthrough of the PLC program

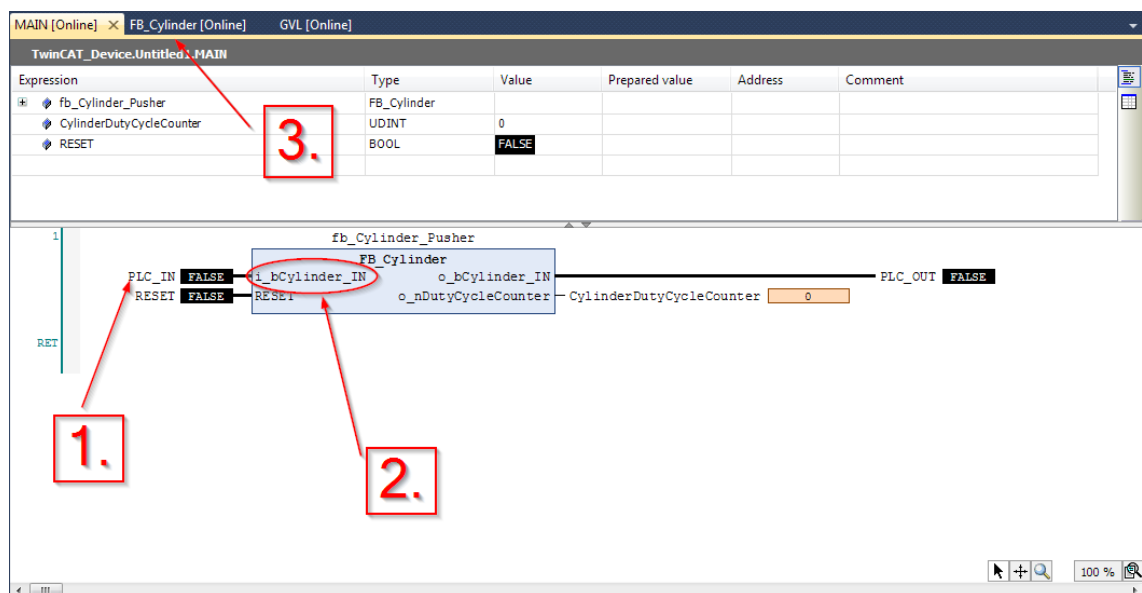
On TwinCAT side, the user can see the Boolean signals going from FALSE to TRUE every time PusherConveyor activates. On the GVL tab (picture 60) in TwinCAT, the PLC\_OUT and PLC\_IN will change to TRUE (1.) everytime a product has been pushed successfully out of the conveyor. Remember, that the PLC is so fast that even though both PLC\_IN and PLC\_OUT flicker TRUE at the same

time, actually after PLC\_IN gets the signal it goes through the whole program and lastly comes to the PLC\_OUT. GVLs show that the connection is working.

TwinCAT_Device.Untitled1.GVL				
Expression	Type	Value	Prepared value	Address
PLC_OUT	BOOL	FALSE		
PLC_IN	BOOL	FALSE		1.

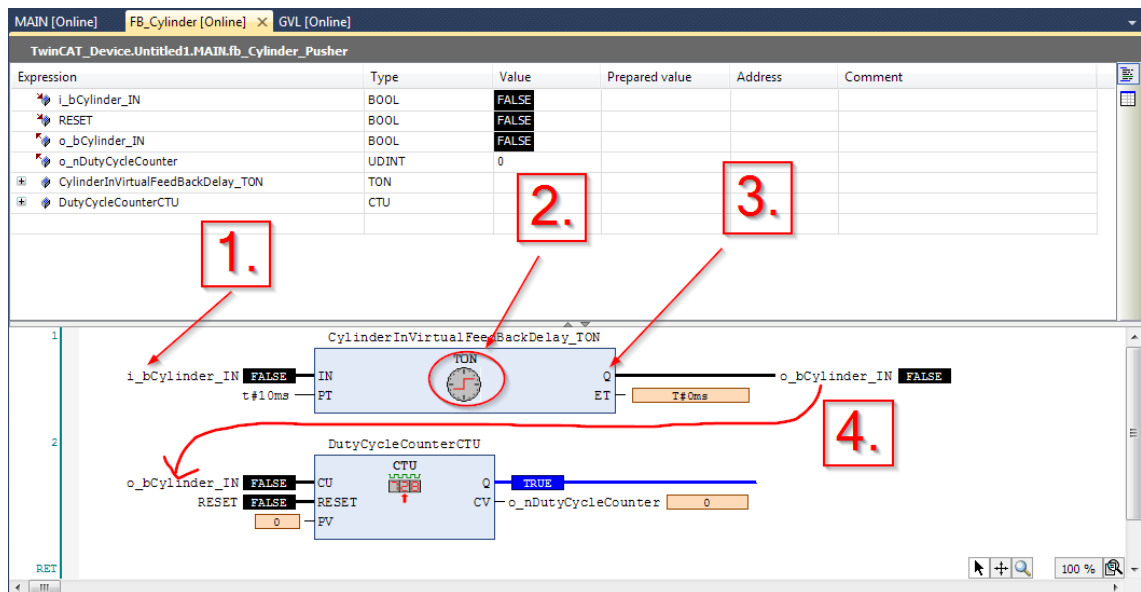
Picture 60: The Global Variables should flicker to TRUE.

Next will be explained how MAIN (PRG) and FB\_Cylinder share information and how the signal run through starts (picture 61). The PLC\_IN signal (from the GVL) (1.) comes to the input of the FB\_Cylinder in MAIN (PRG) (2.) and that will send the signal to i\_bCylinder\_IN at FB\_Cylinder (3.).



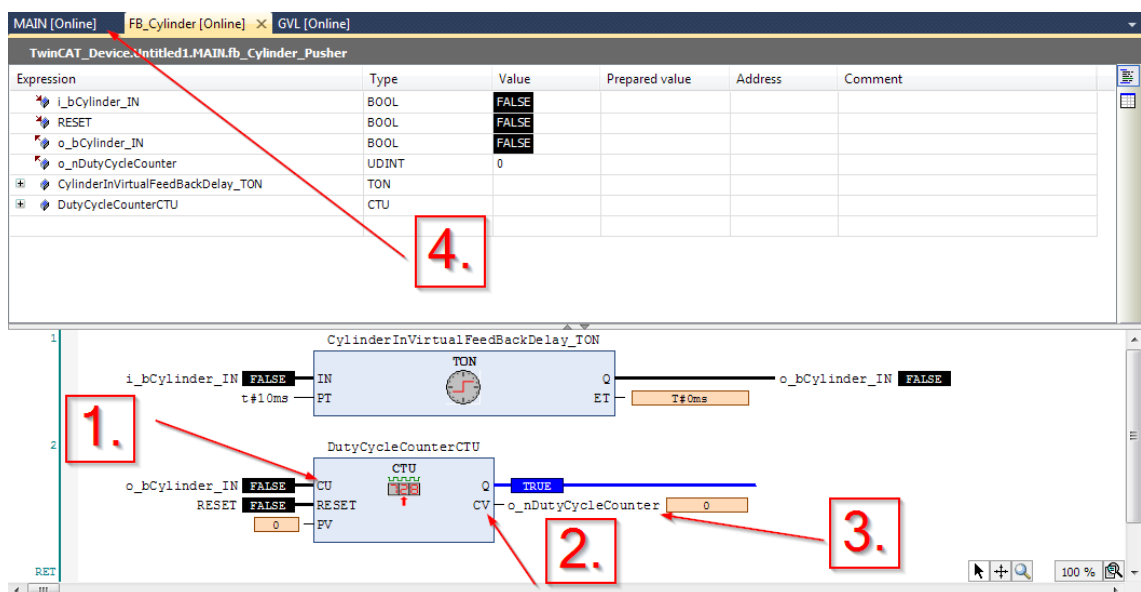
Picture 61: Signal run through in MAIN (PRG).

In the FB\_Cylinder side the “i\_bCylinder\_IN” signal comes to the input of the CylinderInVirtualFeedBackDelay\_TON function block (1.). After this, the Feedback delay will come on (10ms) (2.), then the signal continues to to “Q” output (3.) which sends the signal to o\_bCylinder\_IN (4.) in the CTU function block (picture 62). The o\_bCylinder will also send the signal to the MAIN (PRG) but first the signal will be going through DutyCycleCoounterCTU.



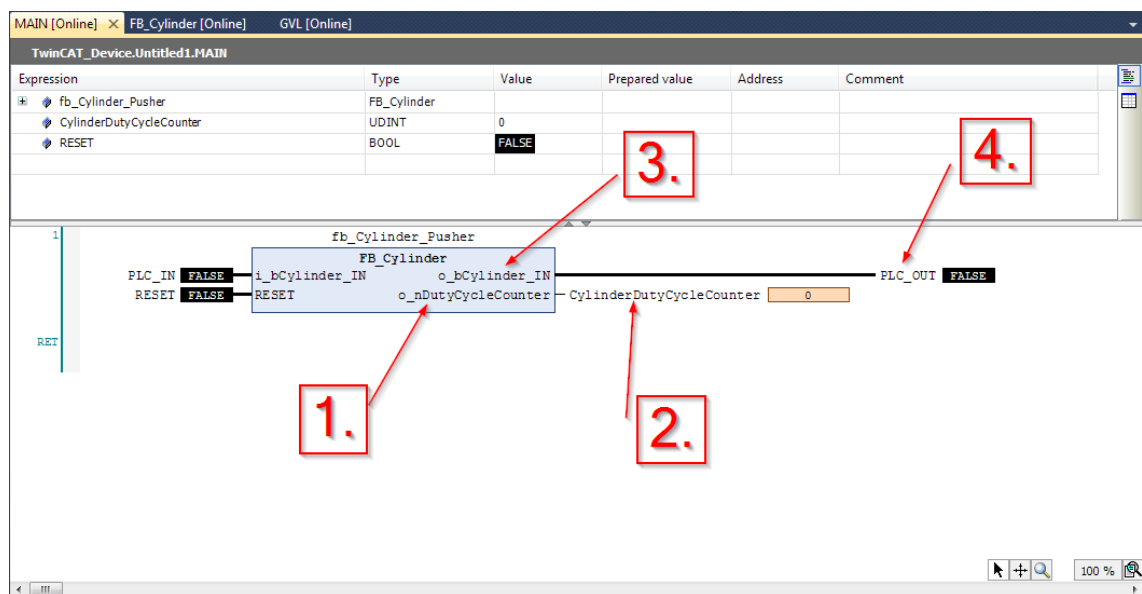
Picture 62: The run through of the signal in TON function block.

The next part shows the CTU function block signal run through (picture 63). On the DutyCycleCounterCTU function block, the signal comes in the CU (count up) input and it goes to the CV (current counter value) output. CV sends the signal to “o\_nDutyCycleCounter” on the MAIN (PRG) side.



Picture 63: DutyCycleCounterCTU function block run through.

Back to the MAIN (PRG) tab with the final signal output (picture 64). In this case two things will happen: the “o\_nDutyCycleCounter” signal (1.) adds one number up in the “CylinderDutyCycleCounter” output (2.) in the fb\_Cylinder\_Pusher and at the same time the output signal “o\_bCylinder\_IN” (3.), that was sent from “CylinderInVirtualFeedBackDelay\_TON”, goes to “PLC\_OUT” (4.) and makes the movement of PusherConveyor (picture 64).



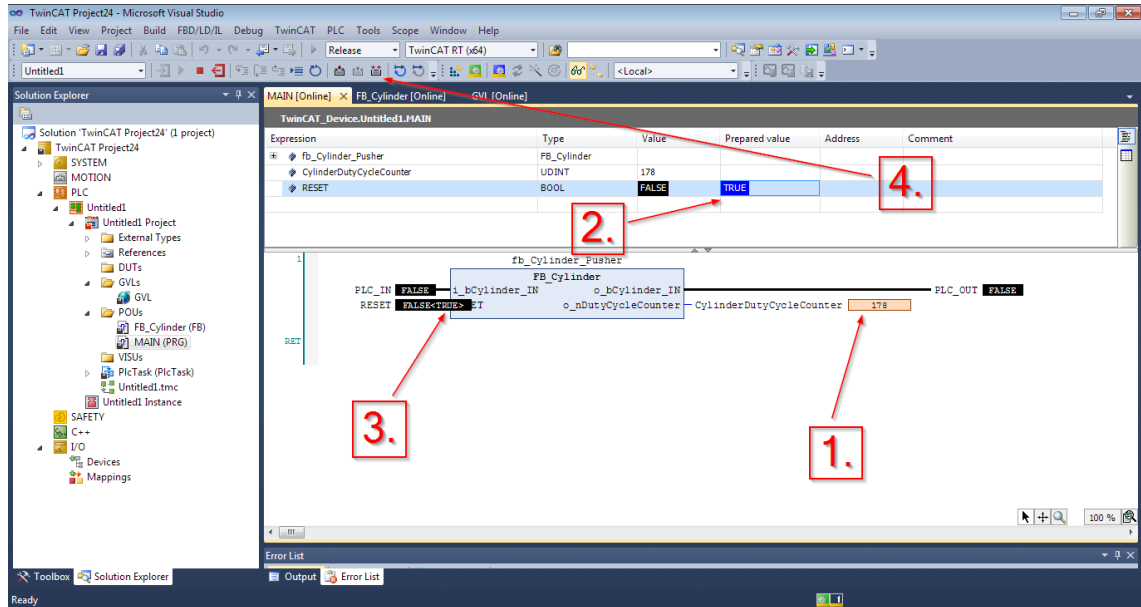
Picture 64: The final outcome.

#### 4.5.2 The RESET function

The RESET function will be explained in the picture 65. RESET function is used, when the user wants to start the counter again from zero. When TwinCAT is logged in and the 3D Create conveyor is working, the counter is counting up. To make the RESET work the “FALSE” value has to be changed to “TRUE”. By clicking the “prepared value” section of the RESET variable, the user can change the value. Notice that on the function block beside the RESET signal “FALSE” is now <TRUE>. This means, that when “write values” button ( ) has been pushed it will change to “TRUE”. After the “prepared value” has been changed, the value has to be written to all online applications. Pressing “write values” button changes the RESET from “FALSE” to “TRUE” and it makes the counter zero. To make the



program again calculate the rejected products the RESET has to be made to "FALSE". Change the "prepared value" to "FALSE" and write values and it starts to work again.



Picture 65: The RESET signal changes to "TRUE".

## 5 RESULTS AND REFLECTIONS

The goal of the thesis was to make a simple simulation that uses Visual Components 3D Create and Beckhoff TwinCAT 3 logic program with added gamification methods for adding motivation and enjoyment in learning.

The initial goal was to make a more complex simulation in both 3D Create and TwinCAT 3. The goal was not realized because of time limitations and also because of the needed expertise for 3D Create. The plan was to have a robot, that would have moved the rejected products to another conveyor line that would take them to a garbage bin. There would have been also more different kind of products coming and there would have been a different garbage bin for every product. So it would have separate the products. It would have needed better knowledge on Python programming language and RSL programming.

Most challenging part of the thesis was to learn to use the 3D Create and TwinCAT, because of not having knowledge of both simulation programs. For the Visual Components, there were good tutorial videos on the official community page. This community helped always if there were questions with the project planning. A huge thank you for Jamal Muhammed on the Visual Components customer service for help with planning what is possible with 3D Create. On TwinCAT side was quite more difficult there wasn't tutorial videos how to use logic programs. There was only one user in Youtube making guide videos that were also saying that there isn't available a good self learning guide. However once again the customer service at Beckhoff was excellent. A big gratitude for Aapo Vuoristo at Beckhoff, he helped and guided me through TwinCAT 3 and aided with the planning of the logic program.

It was interesting to think how to add gamification methods for the project. The gamification term wasn't known to me, but after reading about it I could see gamification everywhere around me. The original plan was also to make badges for the thesis project, but not having knowledge of using image design software and time constraint made implementation impossible.

To make sure that the project it is possible to make also for new students, I had a test group go through the making of the simulation. The result was a success, the test group could make the main project and I was overlooking the project, explaining how the gamification would be implemented in different parts. I did get a great feedback on the gamification part, the test group did say that it would really motivate them in school or at work, with having the points, levels and badges.

## SOURCE MATERIAL

Banks, J. 1998. Handbook of simulation. USA: John Wiley & Sons Inc.

Beckhoff, 2016. Data Type Units.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tc3\\_plc\\_intro/18014398645430411.html&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/18014398645430411.html&id=) Retrieved: 20.4.2016.

Beckhoff, 2016. Global Variable Lists.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tc3\\_plc\\_intro/18014398645438091.html&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tc3_plc_intro/18014398645438091.html&id=) Retrieved: 20.4.2016.

Beckhoff, 2016. Program Organization Unit.

[http://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl\\_setup.htm&id=](http://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl_setup.htm&id=) Retrieved: 20.4.2016.

Beckhoff , 2016. Visualization.

[http://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcvisu\\_target\\_set.htm&id=](http://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcvisu_target_set.htm&id=) Retrieved: 20.4.2016.

Beckhoff, 2016. Function Block Diagram.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl\\_Languages%20FBD.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl_Languages%20FBD.htm&id=) Retrieved: 3.5.2016.

Beckhoff, 2016. Ladder Logic Diagram.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl\\_languages%20ld.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl_languages%20ld.htm&id=) Retrieved: 3.5.2016.

Beckhoff, 2016. Different data types.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl\\_plc\\_data\\_types\\_overview.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/tcplcctrl_plc_data_types_overview.htm&id=) Retrieved: 4.5.2016.

Bolton, W. 2015. Programmable logic controllers. Sixth Edition. USA, Massachusetts:Newnes.

[https://books.google.fi/books?hl=fi&lr=&id=sDqnBQAAQBAJ&oi=fnd&pg=PP1&dq=programmable+logic+controller&ots=-9go0m4may&sig=CWfpY-jaA1oKFkBE3lyAth9mii\\_g&redir\\_esc=y#v=onepage&q=programmable%20logic%20controller&f=true](https://books.google.fi/books?hl=fi&lr=&id=sDqnBQAAQBAJ&oi=fnd&pg=PP1&dq=programmable+logic+controller&ots=-9go0m4may&sig=CWfpY-jaA1oKFkBE3lyAth9mii_g&redir_esc=y#v=onepage&q=programmable%20logic%20controller&f=true) Retrieved: 25.4.2016.

Duggan, K. & Shoup, K. 2013. Business Gamification For Dummies. Hoboken, New Jersey: John Wiley & Sons, Inc.

Example of gamification for medicine students.

<http://www.anatomyarcade.com/games/WAB/WAB.html> Retrieved 3.3.2016.

Hevehagen, t. Hirschfeld, R. Tracht, R. Function blocks introduction.

<http://www.functionblocks.org/Introduction.html> Retrieved: 13.5.2016.

Järvilehto, L. 2014. Hauskan oppimisen vallankumous. 2. Painos Jyväskylä:Bookwell Oy.

Khan Academy, 2016. Basic Knowledge.

<https://www.khanacademy.org/about> Retrieved: 12.1.2016.

Krokkfors, L. Kangas, M. Kopisto, K. 2014. Oppiminen pelissä: pelit, pelillisyyys ja leikillisyyys opetuksessa. Vantaa: Hansaprint Oy.

Koehler, K. 2013. Unsigned and signed integers.

<http://kias.dyndns.org/comath/13.html> Retrieved: 4.5.2016.

Lobdell, M. 2011. Study Less Study Smart.

<https://www.youtube.com/watch?v=IIU-zDU6aQ0> Retrieved 10.2.2016.

Paharia, R. 2013. Loyalty 3.0: How BIG DATA and GAMIFICATION Are Revolutionizing Customer and Employee Engagement. USA: McGraw-Hill Education.

Pelling, N. 2015. Panel about gamification.

<https://www.youtube.com/watch?v=Y7bHyHR9ysQ> Retrieved 3.2.2016.

Peter. 2015. Structured Text tutorial to expand your PLC programming skills.

<http://www.plcademy.com/structured-text-tutorial/#what-is> Retrieved: 6.5.2016.

Physicaltherapyweb. Fitocracy fitness app review.

<http://physicaltherapyweb.com/fitocracy-fitness-app-review/> Retrieved: 6.4.2016.

2016. Python Overview.

[http://www.tutorialspoint.com/python/python\\_overview.html](http://www.tutorialspoint.com/python/python_overview.html)

Retrieved: 15.4.2016.

Recycle Bank

<http://www.recyclebank.com/corporation-info> Retrieved: 3.3.2016.

Rouse, M. 2005. Definition of data type. Service oriented architecture. <http://searchsoa.techtarget.com/> Retrieved: 4.5.2016.

Schell, J. Gamifications future. Retrieved 10.3.2016.

<https://www.youtube.com/watch?v=9NzFCfZMBkU>

Visual Components, 2004) Robot Sequence Language.

<http://download.visualcomponents.net/elib/3.1/eCat/EquipmentLibrary/6%20Support/Quickstart/Quickstart.pdf> Retrieved: 25.4.2016.

Värjä, P. & Mikkola, J. 1995. Ohjelmoitavat logiikat. Kouvola:Nettopaino Oy.

Ängeslevä, S. 2014. Level up:Työruutinit peliksi. Helsinki:Talentum.

## PICTURE SOURCES

Peter. 2015. Structured text example.

<http://www.plcacademy.com/structured-text-tutorial/#what-is> Retrieved:4.5.2016.

Beckhoff. 2016. Timed on-delay function block.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibstandard/html/tcplclibstandard\\_ton.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibstandard/html/tcplclibstandard_ton.htm&id=) Retrieved: 13.5.2016.

Beckhoff. 2016. Count up function block.

[http://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibstandard/html/tcplclibstandard\\_ctu.htm&id=](http://infosys.beckhoff.com/english.php?content=../content/1033/tcplclibstandard/html/tcplclibstandard_ctu.htm&id=) Retrieved: 13.5.2016.

Beckhoff. 2016. Function Block Diagram Example.

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl\\_Languages%20FBD.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl_Languages%20FBD.htm&id=) Retrieved 13.5.2016.

Example photo of memorization in Memrise.

[www.memrise.com](http://www.memrise.com) Retrieved 13.01.2016.

Example picture of Function Block Diagram (FBD)

[https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl\\_Languages%20FBD.htm&id=](https://infosys.beckhoff.com/english.php?content=../content/1033/tcplc-control/html/TcPlcCtrl_Languages%20FBD.htm&id=) Retrieved: 3.5.2016.

Differences between Python language and Java language.

<https://pythonconquerstheuniverse.wordpress.com/2009/10/03/python-java-a-side-by-side-comparison/> retrieved: 20.4.2016.

Example set of Badges.

<http://www.webmarketing-com.com/2014/04/24/27263-4-exemples-gamification-dune-communaute> Retrieved: 15.4.2016.

Pacman videogame photo.

<http://gamefabrique.com/games/pac-man/> Retrieved: 12.1.2016.

Pong videogame photo.

<http://www.bbc.com/news/technology-33005297> Retrieved: 12.1.2016.

Example points bar for the gamification points.

<http://teso.curseforge.com/teso-addons/slightly-improved-experience-bar/images/4-slightly-improved-experience-bar/> Retrieved: 15.4.2016.