Delft University of Technology

FACULTY MECHANICAL, MARITIME AND MATERIALS ENGINEERING
Department Marine and Transport Technology

Mekelweg 2
2628 CD Delft
the Netherlands
Phone +31 (0)15-2782889
Fax +31 (0)15-2781397
www.mtt.tudelft.nl

Specialization: Transport Engineering and Logistics

Report number: 2015.TEL.7925

Title: **Demonstratie Productielijn**

Author: G. M. Mul

Title (in Dutch) Demonstratie Productielijn

Assignment: Research assignment

Confidential: no

Initiator (university): Dr. ir. H.P.M. Veeke & Dr. ir. M. Wisse

Supervisor: Dr. ir. H.P.M. Veeke

Date: 13 Februari 2014

Table of Contents

Chapter 1

Introduction

Automation of a production line is an interesting topic for production facilities. It can lead to cost savings and higher production levels. At a production line robots can be used to replace human labor and grep products from a belt conveyor. This report will describe the simulation of a production line with three robots.

The robots are designed in Robot Operating System (ROS) which is a package in LINUX. This program enables to make a simulation of the virtual robot. When the simulation performs at the desired method, it can easily be replaced by real robot which works at the same commands.

When multiple robots are used to retrieve the products from the belt conveyor, assigning products to a specific robot might lead to better result. A logistic method can be used to find the optimal packing order of the products. To find this method the software Tool for Object oriented Modeling and Simulation (TOMAS) will be used. It can be considered as a 'toolbox' in the standard Delphi-environment. The program is object-oriented it is ideal for modeling and simulating discrete processes in production and transportation [Veeke and Ottjes, 1999]. In contradiction to ROS, TOMAS is working in a Windows environment.

Combining both programs into one model can theoretically lead to an ideal control of the production line. The challenge for this project is to combine the software packages as they are working on different environments. This means that the connection will experience some difficulties. Therefore the goal of this research assignment is to initialize a model which runs on a TOMAS distribution and assigns the ROS robots to execute tasks. The main focus will be on the TOMAS software to work at the desired performance. The robots in ROS will be modeled with basic elements to establish a connection and the simulation can be run.

Chapter 2

Setup of the model

2.1 Introduction

In this chapter the setup of the model will be introduced. It will exist of a belt conveyor with three robots which are able to grep the products. An overview of the setup shows a clear idea of the model and its input variables. The model will be running in TOMAS and ROS, this requires a connection between the programs for interaction of information. A TCP has been found as a useful tool to realize the connection. It operates with a Server and multiple Clients. In this chapter the setup of the system will be explained and how the connections are used.

2.2 Overview of the model and input parameters

The production line will exist of three robot arms which are positioned along a belt conveyor. Figure 2.1 shows an overview of the model.

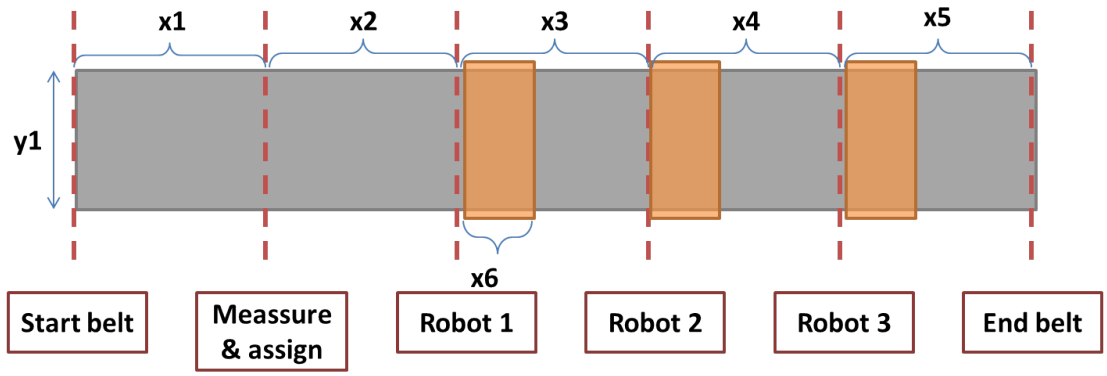

Figure 2.1: Overview of the model

In this figure the products will arrive at the start of the belt conveyor which has a width of y_1 [m]. By belt conveyor (running at speed V_{belt} [m/s]) the products transport distance x_1 [m] until they will be measured and assigned to a robot. Next they will transport to the robot over the right distance. More details of the robot are shown in figure 2.2.

The figure shows that the robot has a reaching length x_6 [m]. Combined with the belt width this will create the working area of the robot. Here it will operate with the speed V_{robot} [m/s]. In this model the acceleration of the robot is expected to be infinite which allows it to simulate the robot with a constant speed.

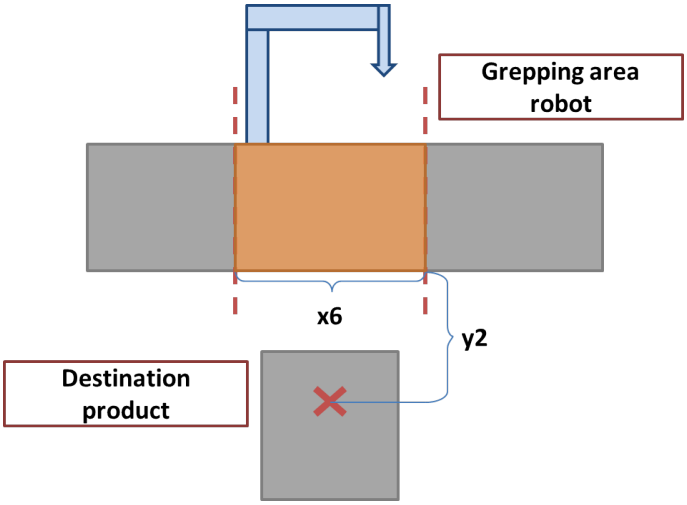

Figure 2.2: Setup of the robot

Next to the setup of the process there are also two input parameters about the circular products. The first describes the size of the products $D_{product}$ [m]. The second is the Inter Arrival Time (IAT) between the products in seconds.
Table 2.1 provides an overview of all the input variables of the model. These can be adjusted to obtain the desired setup.

Table 2.1: Input variables of the model

Variable	Unit	Description
x_1, x_2, x_3, x_4, x_5	[m]	Distances between sections on belt conveyor
x_6	[m]	Length of the reach of the robot arm
y_1	[m]	Width of the belt conveyor
y_2	[m]	Distance between belt conveyor and destination
V_{belt}	[m/s]	Constant speed of the belt conveyor
V_{robot}	[m/s]	Speed of the robot
$D_{product}$	[m]	Diameter of the products
IAT	[s]	Inter Arrival Time between products

2.3 Structure of the model

2.3.1 Connecting models with each other

The model will exist of two software programs, TOMAS and ROS. These have to connect with each other during the simulation. To set this connection a Transmission Control Protocol (TCP) will be followed. This is a connection-oriented, end-to-end reliable

protocol designed to fit into a layered hierarchy of protocols which support multi-network applications [Postel, 1981]. Since it is able to communicate in both Linux and Windows it is a useful tool for the simulation of the production line.

The hierarchy of TCP is build with a main Server which is able to control multiple Clients. The overall control simulates and orders all the incoming commands. In Delphi multiple sources are available to set up such a connection. Piette [2013] has developed a source which is integrated in TOMAS. This tool will be used to setup the Server and Clients as the main component of the model.

Next to the communication between TOMAS clients, also communication towards ROS is necessary. Research of Valk [2014] has used the Delphi component sgcWebSocketLib to establish communication between TOMAS and ROS.

2.3.2 TOMAS

The TOMAS model will be used to organize the distribution of the products. In this model the complete production line has to be modeled, id est it must exist of the belt conveyor to transport the products and of three separate robots to pick them up. An overall control will be available in the form of a TOMAS server. This displays the events taking place within the models. Further details of the model can be found in section 3.2 and 3.3.

2.3.3 ROS

The ROS model will display the movements of the robots. This will be a basic model to set the initial conditions for communication between ROS and TOMAS. It will exist of a server which is able to receive messages (commands from TOMAS) and of a client which exists of virtual robots (turtles) which are able to display the movements. Further details of this model will also be provided in section 3.4.

2.3.4 Combining the models

The models are able to run separately but to combine the models a client has to be added. This client is able to communicate with both servers. It will receive commands for the robots in the form of coordinates. These will be transformed into the necessary command to ROS. The figure below shows the client "DataTransfer" as the communicator between the models.

The TOMAS model is only communicating towards ROS by sending sockets of information and not the other way around. But it has been found that the communication can take place in both directions [Valk, 2014]. This can be used when an extended ROS model will be made.

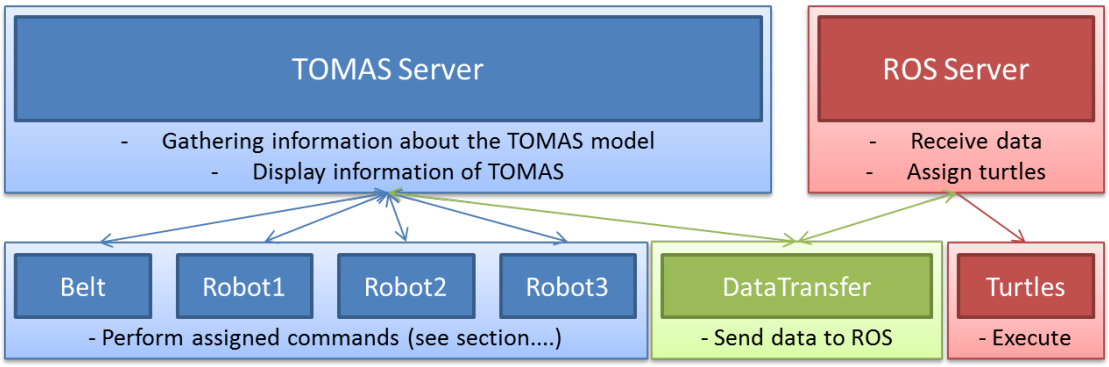

Figure 2.3: The setup of servers/clients of the model.

Chapter 3

Simulation of the production line

3.1 Introduction

The simulation of the production line will be done in separate models. TOMAS will process all the products and divide them to the robots. These steps will be done in the "BeltConveyorProject". The "RobotProject" is used to grep the products from the belt conveyor. The movements of the robots can be seen in the ROS model that shows a visualization.

The collaboration between the robots in TOMAS and ROS demand some adjustments before the data can be send. During this chapter more details will be provided about why these adjustments are required and how these are implemented in the model.

3.2 Belt conveyor in TOMAS

The belt conveyor is the basis of the TOMAS model. In this client the input data can be adjusted and the process initiates the products. The products are generated in the model by the procedure 'Product Generator'. They enter the belt conveyor and after distances x_1 their location can be determined by a sensor. At this point they will be eliminated from the "BeltConveyorProject" and assigned to a "RobotProject".
When the robots are occupied while the product passes, the robot will miss the product. The product will reenter the belt conveyor as a lost product. Here the product remains until it falls at the end of the production line.

3.2.1 Product generator

The products are generated in the client 'BeltProject' by the procedure Product Generator. An exponential distribution with an Inter Arrival Time (IAT) generates the products. A sample is taken to look for a suitable location to place the products. It can be possible that multiple products are generated near to each other. Therefore generator tries to avoid stapling the products by sampling for a new 'free' spot. With multiple products close to each other, it is possible that no sample can be done to find a suitable location. To avoid a never-ending loop, the simulation accepts the location of the product after 25 samples and a stapled product is generated. Even though the stapled products can not be processed in reality, the model will accept the products and processes them.

3.2.2 Dividing the products to a robot

When the products have transferred distance x_1 on the belt conveyor they arrive at the measurement zone. In reality, this zone will be able to measure the exact position of the products by sensors, for example cameras. This information will be used to assign the product to a robot. Different methods for this assignment can be considered. This basic model assigns the products in order of arrive to a robot (product 1,4,..,3n-2 to robot 1, 2,5,..,3n-1 to robot 2 and 3,6,..,3n to robot 3).

3.2.3 Lost products

Products that are missed by the robot, because it is occupied at the moment the product passes, will be accepted as lost products. The products remain on the belt conveyor until the end and 'fall of'. When optimization is pursued, it might be possible to reassign the products to another robot to get a second effort to handle the product. As this model focuses on the basic simulation, this optimization is neglected but advised to consider during implementation of a realistic production line.

3.3 Robots in TOMAS

The robot receives the products from the belt conveyor and determines how they will processed. Therefore the pickup coordinates have to be determined, this can be done by combining the path of the product and the path of the robots. An iterative procedure will be used to find the exact coordinate of the pickup place which will be explained during this section.

3.3.1 Path of the robot

In ROS a virtual robot will be created. A turtlesim is used as it is a basic model which can be easily adopted for simplistic simulation. The only disadvantage of the turtle is that it is only able to move in a curve. Translation towards the sides is impossible for this simulation. This means that the tasks that are assigned to the robot have to be adjusted to the necessary commands. The turtle requires two input values for its path.

- θ: The curve of the path in radials
- L: The length of the path that has to be followed

The basic model in TOMAS uses x and y-coordinates, which means that they have to be transformed. Figure 3.1 shows an overview of the path that has to be followed by the robots. This will be followed by the necessary steps to determine the θ and L for the model.

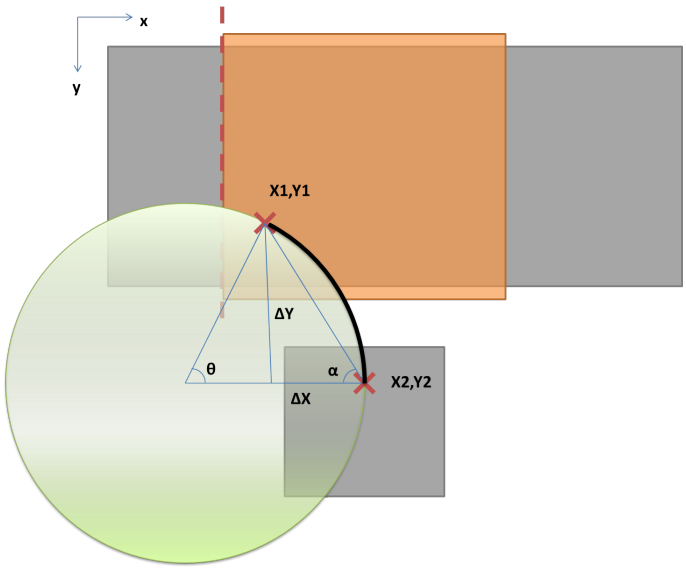

Figure 3.1: Schematic view of the path of the robots

The distance that has to be covered by the robot must first be expressed in coordinates of ΔX and ΔY.

$$\Delta X = X2 - X1 \tag{3.1}$$

$$\Delta Y = Y2 - Y1 \tag{3.2}$$

These coordinates are used to determine the corner α from origin to destination. As a isosceles triangle is created the curve of the path θ can be determined.

$$\alpha = \arctan(\frac{\Delta Y}{\Delta X}) \tag{3.3}$$

$$\theta = \pi - 2 \times \alpha \tag{3.4}$$

To retrieve the length of the path L, first the radius of the circle has to be determined using the sinus rule. The arc length will follow from equation 3.6.

$$r = \frac{\Delta Y}{\sin(\theta)} \tag{3.5}$$

$$L = r \times \theta \tag{3.6}$$

When using the circle to determine the path of the robot, a limitation to the input variables will follow. The circle can only be drawn when the vertical displacement is larger then the horizontal displacement. This means that the restriction gained by this procedure is:

$$y_2 \geq 0.5x_6$$

3.3.2 Determine the grepping position of the product

When the products are assigned to the robot, the pickup place must be determined. To do this, the TOMAS model will look at the first product which will arrive at the robots grepping area. The product can be positioned at multiple areas when it is processed by the robot:

- The product is in front of the grepping area
- The product is in the grepping area
- The product has passed the grepping area

For the three methods the same iterative process will be used to determine the pickup place of the product. The model compares the transfer times of the product and the robot to a position. When the product arrives earlier at the pickup place (then the robot), the product can't be handled. The model will shift the pickup place slightly further to determine whether the robot will be earlier at the next position. Eventually a position will be found where the robot will arrive earlier then the product, this will be the pickup place of this product, see figure 3.2 for an illustrative overview of this procedure.

The next step is to determine whether this place is within the grepping area of the robot.

- When the pickup place is in front or within the grepping area, the robot will position itself at the pickup place, wait for the product to arrive and handle it towards the destination.
- When the pickup place is behind of the grepping area, the products will be accepted as a loss and remain on the belt conveyor. Here it will stay until the end of the belt. In a more detailed model it might be possible to add the option to reassign the product to an other robot to reduce the number of losses.

These steps will only lead to a solution for the pickup place when the robot arm moves at a higher speed then the belt conveyor. This is a restriction for the input values, but it is assumed that this condition will always be hold at a production line.

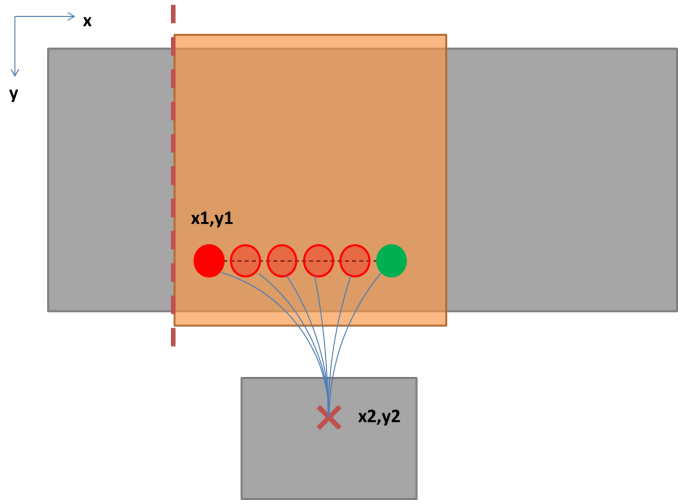

Figure 3.2: The procedure to determine the pickup place of the product

$$V_{robot} > V_{belt}$$

As mentioned it is also possible that the products are positioned before the grepping area. The iterative process will then begin with a pickup place at the start of the grepping area. This will enable the option that the robot arrives earlier at the pickup place, at this position it will hold until the product arrives to be handled.

When a product is already behind the grepping area it will result in a pickup place even further behind the grepping area. Therefore the product will automatically be accepted as a lost product by the iterative process.

3.4 Visualization of the robot in ROS

The ROS model is fairly simplistic and is used to show the communication between the models. The tutorial model of a turtlesim will be used to display all movements. As mentioned, the turtles are only able to move with a twist (by arc length L and rotation θ).

The tasks that has to be executed by the turtles are received from the TOMAS model, by the client "DataTransferProject". With a more detailed ROS model, the commands can be adjusted to obtain a straight trajectory of the robot arm. The detailed model might also be able to extend the communication between ROS and TOMAS. For example the robot of ROS can send a message towards TOMAS when it is occupied or free. This can be used to stop or resume processes in the TOMAS model. The communication between the models has already been shown by Valk [2014] in "How to communicate between ROS and TOMAS".

3.5 Real-time simulation

The ROS model is working in real-time mode. Therefore it is necessary that TOMAS also does this. A function is embedded in the TOMAS Server to enable the real-time mode. At this way the models run at the same speed and are able to simulate the production line.

It has been found that the TOMAS Server suffers some lack at real-time mode. This will cause serious problems when it will be used in a real production line since the reality has to be modeled perfectly. Otherwise the model sends the robots to late to a specific position and errors will occur. This serious problem has to be eliminated in further research before any implementation can be done in a production line.

Chapter 4

Running the model

4.1 Introduction

In this chapter the steps will be provided to start the simulation. First the ROS model has to be ready to receive commands and position the turtles correctly. This will be followed by starting the TOMAS model which starts the complete simulation. An overview of the resulting ROS and TOMAS Server will be shown in section 4.3.

4.2 Starting the model

Before the model can be started, the connection settings have to be checked. During this project the virtualbox of Oracle has been used to run the Linux software. A guideline to change the connection settings found for example on Youtube [Rahman, 2014].

When all setting are correct the model can be started. Multiple steps have to be followed to connect all clients to the servers. The first step is to start the ROS-model.

1. Start a terminal and run "roscore".
2. Open a new terminal and run "rosrun turtlesim turtlesim_node".
3. Open a new terminal and run the following steps to create and position the turtles:
   (a) "rosservice call kill turtle1"
   (b) "rosservice call spawn 2 7 0 "turtle1" "
   (c) "rosservice call spawn 5 7 0 "turtle2" "
   (d) "rosservice call spawn 8 7 0 "turtle3" "
   (e) "rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist −− '[0, 0, 0]' '[0, 0, 1.5708]' "
   (f) "rostopic pub -1 /turtle2/cmd_vel geometry_msgs/Twist −− '[0, 0, 0]' '[0, 0, 1.5708]' "
   (g) "rostopic pub -1 /turtle3/cmd_vel geometry_msgs/Twist −− '[0, 0, 0]' '[0, 0, 1.5708]' "
4. Run the command "roslaunch rosbridge_server rosbridge_websocket.launch"

At this moment the ROS-model is up and running, waiting for tasks from TOMAS. Therefore the next step is to start this model by the following procedure:

5. Open the executable "TomasServerProject"

6. Open and start the TOMAS executables "BeltProject", "Robot1Project", "Robot2Project", "Robot3Project" and "DataTransferProject"

7. At this moment the complete model is able to start running by opening starting the "TomasServerProject"

4.3 Running the model

The model will start and the products will be displayed on the TOMAS Server. Figure 4.1 shows an image of the Server at the start of the simulation. All clients have been connected to the server and receive a "ClientName". This has enabled to start the simulation. The first product has been assigned to robot 1 and two others are in front of the measuring area. The Key Performance Indicators are the number of products processed and missed. Others can eventually be added when the model has been expended. The time display shows that at this moment the model is running at a ratio of 1,00. This means that simulation is running at the same speed as real-time, which is necessary for the simulation in ROS.

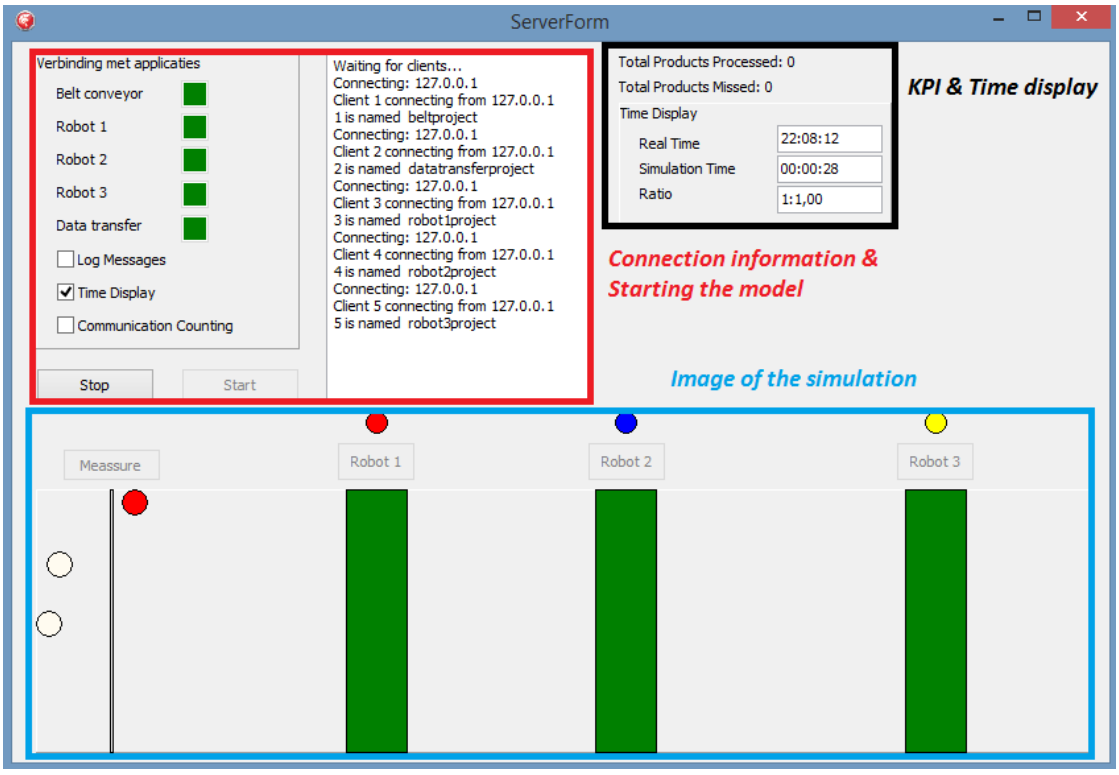

Figure 4.1: The Tomas Server at the start of the simulation

Figure 4.1 is an image of the beginning of the simulation. As mentioned it shows that the model is running at a ratio of 1,00. This means that at this moment the simulation time has been running at the same speed as reality. For a real life application of the production line it is critical that this ratio remains at 1,00. Unfortunately, when the simulation runs, the ratio decreases and shows fluctuations. In extended research, this problem has to be solved before implementation can be realized.

Next to this the ROS simulation has also started. As can be seen in the TOMAS Server, the first product has been assigned to robot 1. This means that the pickup place of this product can determine and the robot will be moved to the location. Figure 4.2 shows that the left turtle (robot 1) has moved to the assigned location and waits for the product to arrive.

Figure 4.2: The ROS model at the start of the simulation

When assigning products to the other robots, these will also start moving. This shows that it is possible to run a simulation in ROS which drives on commands received from TOMAS. The next important step for the production line is to improve the ROS model and simulate a realistic robot.

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

This research has shown that a simulation running in TOMAS and ROS is possible. A basic model has been build to simulate a production line existing of a belt conveyor with three robot arms. Using a webbridge with a TCP connection, sockets of information can be send from TOMAS to ROS. This information can be used to move the robot.

Figure 5.1 shows an image of the simulation running. The robots (turtles) move up and down to grep the assigned product in ROS. Next to this, in TOMAS the simulation of the products on the belt conveyor can be seen.

(a) TOMAS Server in progress **(b)** ROS Server in progress

Figure 5.1: The simulation in progress.

5.2 Recommendations

The model that has been build is fairly basic and exist of the necessary commands to simulate the production line. Improvement is advised as there are multiple indication that in reality this model will encounter some problems. In this section multiple recommendations will be provided to improve the model.

5.2.1 Improve the ROS model

This research has made an effort to set up a connection between ROS and TOMAS. The capabilities to create a ROS model with characteristics of a real robot were not available.

Therefore the turtlesimulation has been used, as this is a fairly simplistic model to use. But multiple problems occurred while using this model.

One error is clearly shown in figure 5.2. The turtles use global coordinates for the path they follow. When the turtle executes the commands it can return at a slightly different location. Using this for a longer time period, will lead to a bigger error, which must be avoided. It can for example be solved by using commands that work on relative coordinates and look at the position of the complete system.

Figure 5.2: The turtles make small errors in path

With an improved ROS model, better communication can be set between ROS and TOMAS. At this moment the ROS model only receives commands from the "Data-TransferProject" but an extended model communication can go in both directions. The advantage of this would be that the ROS is able to inform TOMAS about the occupancy of the robots.

These points are critical for a realistic simulation, therefore the next step must be to create a realistic ROS model. When this is done, other improvements can be done before implementation of the production line.

5.2.2 Improving the TOMAS model

The TOMAS model has been focused on the communication with ROS. This has been set and commands can send with help of a TCP connection. With this being realized, the model itself can be further improved.

In the TOMAS Server a function has been embedded that slows down the simulation to real time, unfortunately this works not yet properly. When the model runs, it shows sensitivity and fluctuations occur. To implement this model in reality, modifications have to be done to increase the simulation speed.

Bibliography

F. Piette. Overbyte ICS - Delphi, 2013. URL http://www.overbyte.be/frame_index.html. Date assessed: 10-2-2015.

J. Postel. RFC 793 - Transmission Control Protocol, 1981. URL http://www.ietf.org/rfc/rfc793.txt. Date assessed: 9-2-2015.

M. Rahman. How to add host only adapter in virtualbox in linux or windows, 2014. URL https://www.youtube.com/watch?v=AWYVXb19wzU. Date assessed: 8-2-2015.

S. Valk. How to communicate between ROS and Tomas. Technical report, TU Delft, Transport Engineering and Logistics, 2014.

H.P.M. Veeke and J.A. Ottjes. Problem oriented modelling and simulation. *Summer Computer Simulation Conference*, 1999. ISBN: 1-56555-173-7.