Technische Universität Dortmund
Department of Biochemical and Chemical Engineering
Chair of Process Dynamics and Operations
Prof. Dr. Sebastian Engell

DEVELOPMENT OF LOCAL POSITIONING SYSTEM FOR A PIPE-LESS PLANT

Automation & Robotics
Group Project SS18

Group Members:

Abdulrahman Abouelkhair(4511328)
Medhini Rajagopal Balamurugan(4511328)
Stefan Rottstegge(191455)
Stephan Vette(198907)

Supervisors:

Afaq Ahmad
Marina Rantanen-Modéer

Abstract

Summary. Note that the abstract heading is unnumbered, it should remain so. To remove heading numbering use:

```
\section*{}
```

Contents

List of Figures

List of Tables

1 Introduction

Add your name to the file name

2   Pipeless Plant

2.1   Existing setup

2.2   Problems with the Existing Setup

..
zb

- Fish eye

- Sunlight..

3 Selection Process

About the 4 techniques..

3.1 Triangulation[1]

Summary

Since our plant has a specified size in which the location of multiple objects has to be performed the method of Triangulation is one promising technic in which research was made. Triangulation was already a common principle of measurement in the 18th century and itÂ´s divided between active and passive triangulation. Passive triangulation is a geometrical method based on two measurement stations which positions are known exactly. At these two measurement points angels of the desired point in space are measured to compute the localization in the specified coordinate system (x, y, z) with trigonometrical formulas. With respect to the two measurement points which were already used in the 18th century nowadays two cameras are installed to perform a geographical method of 3D object-data estimation fig. 1 [1].
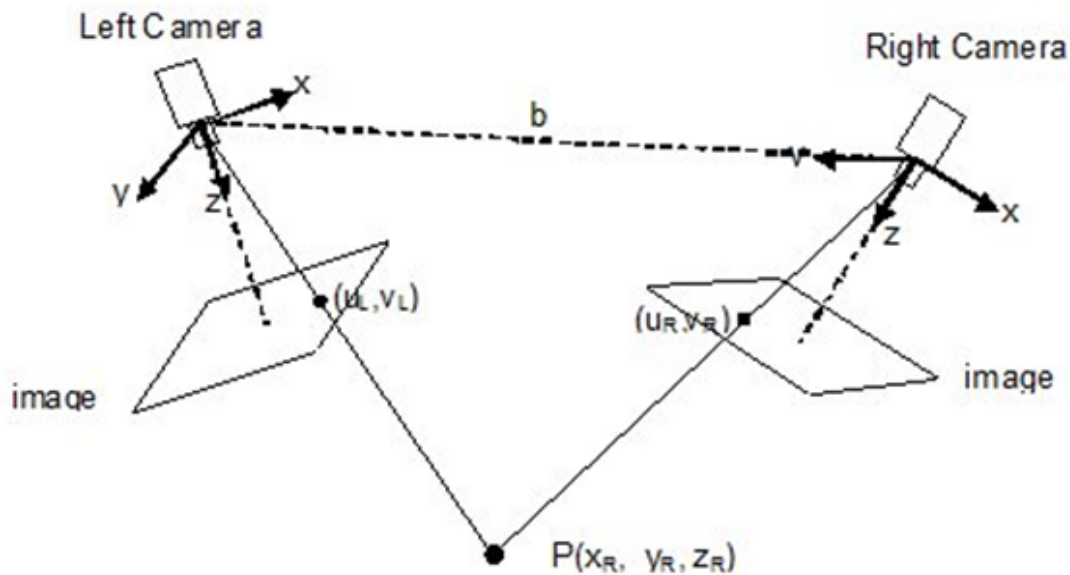
Figure 1: Passive Triangulation set-up with two cameras

To Solve the problem, it is necessary to know the parameters of the left and the right camera visualized in the figure. In theory the triangulation is trivial, since each and every point of the images of the respective cameras maps to a line in 3D space. If a pair of corresponding points, in the case of the pipes less plant it would be an AGV, is found the projection of a point x in

---

[1]Stefan

3D space can be computed. Active triangulation in comparison to passive triangulation needs one camera and at least one source of structured light (e.g. Laser). As the passive way here the geometrical location and orientation of the camera and light source is space need to be known. Two possible set-ups with either a laser point and a stripe as structured light are shown in fig. 2 [2].
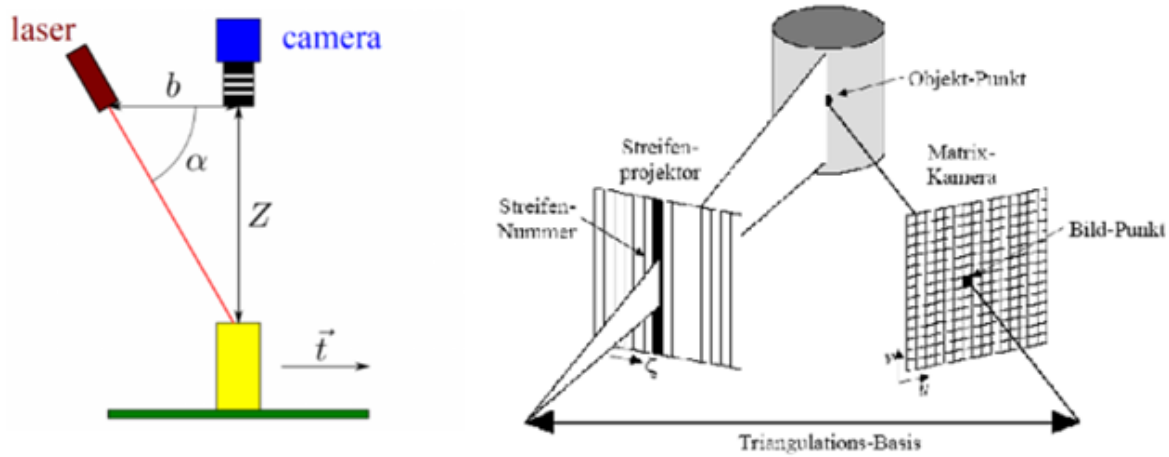
Figure 2: Active Triangulation

To solve the active triangulation problem, the structured light has to point on object which location is desired to estimate, in the case of the pipe less plant it would be the AGV. If this point is found on the 2D image of the camera, a triangulation with basic trigonometrical formulas which are using the properties and parameters of the camera and light source can be performed and the position of the AGV is estimated.

Implementation

One possible way to implement a solution for the passive triangulation is to attach 2 high resolution cameras with USB 3.0 for a fast data transmitting on two edges of the plant being not on each others opposite side as shown in fig. 3.
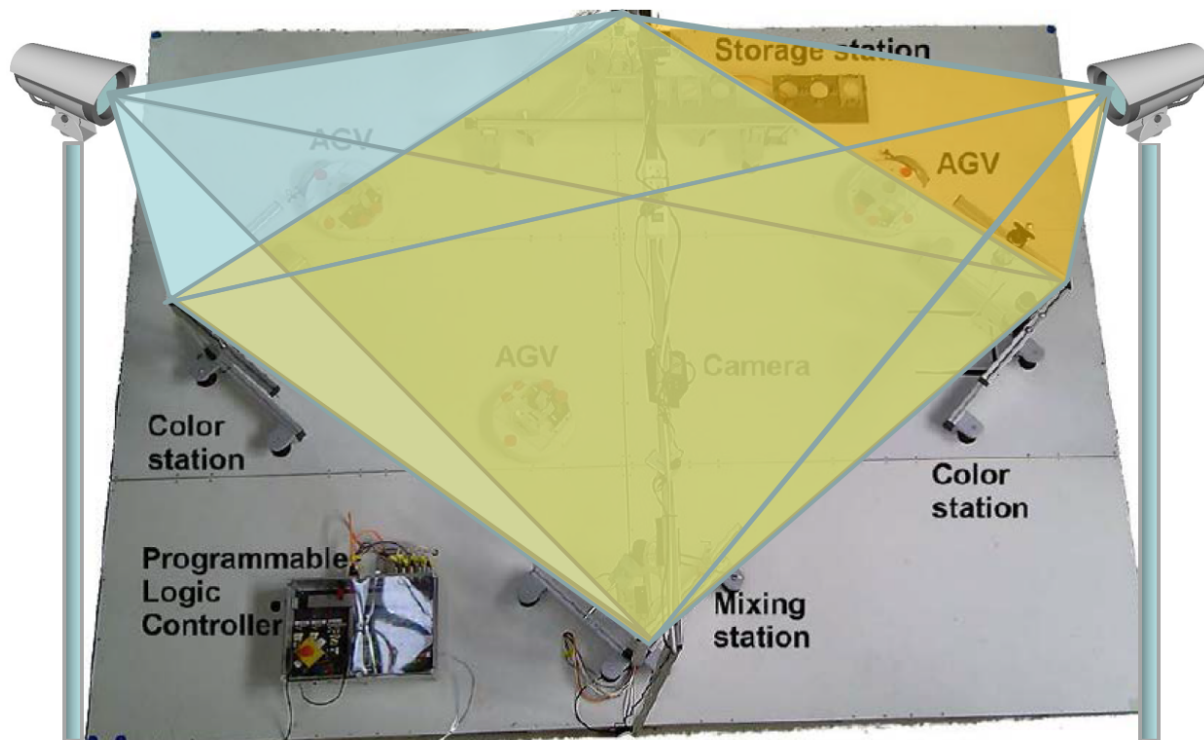
Figure 3: Impmentation of passive triangulation

The left and right camera are sequentially taking pictures which are transmitted to the plants computer where the image processing takes place.

Pro and con

Based on the made research, two tables with advantages and disadvantages of the two RFID systems are created.

Passive Triangulation	
Positive	**Negative**
Upgrade to USB 3.0 for faster data transmitting possible	Light dependent
Upgrade to a camera with higher resolution to reduce measurement error possible	New concept of orientation may be needed
No Fish-Eye-Lense problem	Limited range of observation
Low cost	

Table 1: Positive and Negative Points of Passive Triangulation

Active Triangulation	
Positive	**Negative**
Upgrade to USB 3.0 for faster data transmitting possible	New unknown laser technology is needed
Upgrade to a camera with higher resolution to reduce measurement error possible	High costs for several Laser (one per AGV)
Easy detection of laser points on camera image	Laser needs to move while AGVs are moving
	Limited range of observation
	Light dependent

Table 2: Positive and Negative Points of Active Triangulation

3.2   Pattern Recognition

Summary

Implementation

Pro and con

3.3   RFID[2]

**Summary**

One of the possible solutions to solve the challenging problem of indoor localization is the use of the Radio-frequency Identification (RFID) technology. The mainly areas of this technology is indeed still supply chains, transport, manufacturing, personnel access, animal tagging, toll collection reference 4, but has also become popular in localizing objects and persons. Where in the main applications only the identification has to be realized, also the strength of the signals is important to estimate the position of a certain object.
The main idea of those systems is that a reader detects a tag and reads its information. The technology can be divided into three main types: passive, semi-passive and active systems. A passive system, like it is been shown in fig. 4, consists of a reader, which is connected to an antenna and a computer and a passive tag.
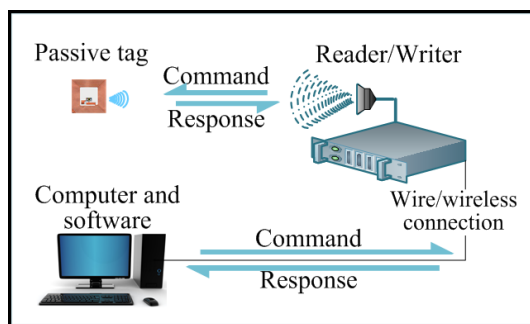
Figure 4: Passive RFID System
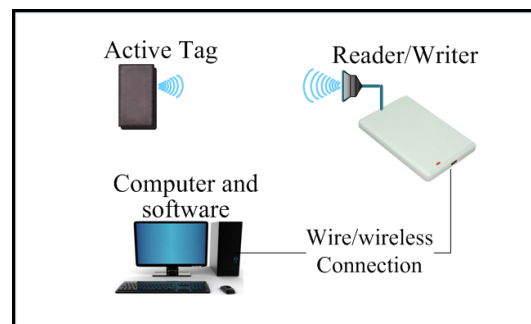
Figure 5: Active RFID System

The system is called passive, because the power supply will be realized by the radio signal of the reader. In case where the tag is in the reading range of the reader, the tags gets enough power to send predefined information (for example ID) back. The active system (see fig.5) in comparison has an active tag which has an own power supply. The semi-passive tag has a battery build in that the the tag has more power to communicate, but is not used to generate radio frequency signals.

Another classification of RFID systems is the frequency of the radio waves. It can reach from 0.135 MHz (Low Frequency) to 5875 MHz (Super High Frequency). The table 3 gives an overview about the systems related to reading ranges, reading rates and the ability to read near metal or water.

It can be seen that the passive systems in general have a smaller reading range then the active systems and has a bigger data rate. But it has also to be take into account, that passive tags a way cheaper then active tags.

---

[2]Stephan
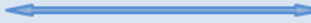
	LF	HF	UHF	SHF
FR (MHz)	< 0.135	3~28	433-435, 860-930	2400~2454 5725~5875
RR(P)	≤ 0.5 m	≤ 3 m	≤ 10 m	≤ 6 m
RR(A)	≤ 40 m	300 m	≤ 1 km	≤ 300 m
TRR	Slower	⟷		Faster
ARMW	Better	⟶		Worse

FR: Frequency Range
RRP: Typical Reading Range of Passive Tags
RRA: Typical Reading Range of Active Tags
TRR: Tag Reading Rate
ARMW: Ability to Read near Metal or Water

Table 3: Overview RFID systems

Implementation

There are mainly two different ways to realize a localization system of the AGVs in the pipeless plant. Based on the fact that the plant has a size of 3 by 4 meter, the tracking can be carried out by passive system in which a couple of passive tags on the floor can be used as landmarks. In this case the reader plus the antenna would be placed on the AGV and localize with the help of the detected tags. The other systems consists of three or four reader in each corner of the plant and an active tag on each AGV.

Pro and con

Based on the made research, two tables with advantages and disadvantages of the two RFID systems are created.

Active RFID system	
Pro	**Con**
Light independent	Prototype more expansive (3 reader + avtive tags)
Space unlimited	Datarate is related to the amount of detected tags a the same time
Localization only has to be realized in a bigger area - medium accuracy	Anticollision need, cause more AGVs are used at the same time
Wired communication between reader and computer possible	Signal strength can be influenced by envirnonment (metal or water)
Simple algorithm (Trilateration)	

Table 4: Pro and cons of active RFID system

Passive RFID system	
Pro	**Con**
Light independent	Communication between AGV and computer has to be realized
Space unlimited	Data rate is related to the amount of detected tags a the same time
Localization only has to be realized between four tags (small area) - high accuracy	Anticollision need, cause more tags are detected at the same time
Simple algorithm (Trilateration)	
Prototype cheap (1 reader + passive tags)	

Table 5: Pro and cons passive RFID system

3.4   Map-Based Localization

Summary

Implementation

Pro and con

..

4 Theoretical Background

4.1 Radio Frequency Identification (Abdul)

4.2 Trilateration[3]

Trilateration is a method to compute the intersection point of three circles/spheres. For this, it is necessary to know the three center of the circles/spheres plus their corresponding radii. The basic idea to estimate the intersection point is to use the mathematical description of a sphere:

$$r^2 = (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 \tag{1}$$

where $(P_n = (x_n, y_n, z_n))$ is the center of the sphere [3]. A few assumption can be made to simplify (1) for the 2D indoor localization on a floor. First of all, the z-component of all spheres can be neglected. Another assumption is that we define the origin of the first circle as the center of the coordinate system, the second along the x-axis with an distance (d) and the third shifted in x- (i) and y-direction (j), which is illustrated in following fig.
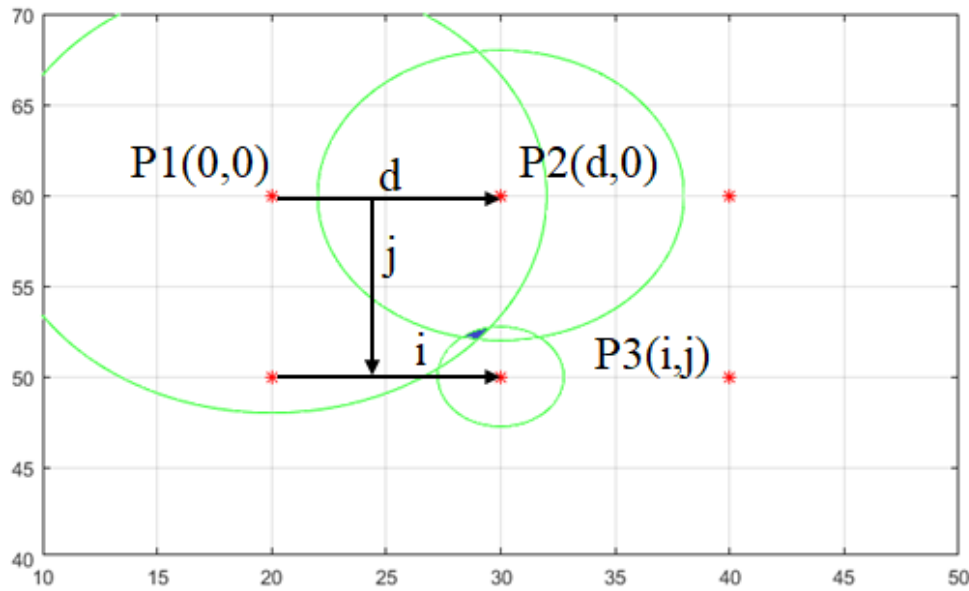

Figure 6: Overview Trilateration

With known positions of the center of the circles d, i and j can be computed in the following

---
[3]Stephan

way[3]:

$$d = |P_2 - P_1| \tag{2}$$

$$e_x = \frac{1}{d}(P_2 - P_1) \tag{3}$$

$$a_x = P_3 - P_1 \tag{4}$$

$$i = e_x \cdot a_x \tag{5}$$

$$a_y = (P_3 - P_1) - i * e_x \tag{6}$$

$$e_y = \frac{a_y}{|a_y|} \tag{7}$$

$$j = e_y \cdot a_x \tag{8}$$

It has to be notice that P_1, P_2 and P_3 are 2D vectors, which represents the x- and y-coordinate of the points.

After knowing these values, the relative distance from the origin of the coordinate system can be computed with the help of (1) and the center of the circles $P_1(0,0)$, $P_2(0,d)$ and $P_3(i,j)$ as follows:

$$x_t = \frac{r_1^2 - r_2^2 + d^2}{2 * d} \tag{9}$$

$$y_t = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2 * j} - i * \left(\frac{x_t}{j}\right) \tag{10}$$

The absolute position of the intersection point is computed in following way:

$$P = P_1 + e_x * x_t + e_y * y_t \tag{11}$$

It can be seen, that those equations are using the first two points plus radii to estimate the x-coordinate and first and third point plus the estimated x-coordinate to estimate the y-coordinate.

4.3 ...

5   Hardware[4]

5.1   RFID reader and antenna[5]

The RFID reader from KTS Systeme (see fig.7) is a HF Modul (frequency around 13.56 MHz). It contains a full-fledged microcontroller with a high-performance RFID transceiver IC. It has a 1.27 mm pitch pin-headers for THT mounting. The connection to an external antenna can be realized via a Single ended 50Ω connection or via Pin Header U.FL. jack, which was used in this project.
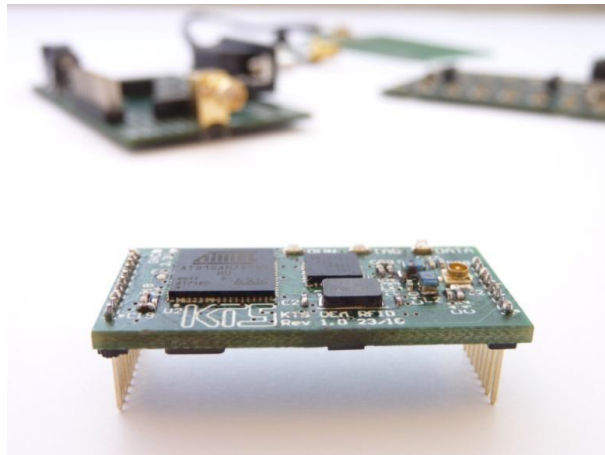

Figure 7: RFID reader KTS Systeme RFIDM1356-001

The communication to other devices is realized via a UART compatible serial interface via pin 6 (RX) and 7 (TX). The power supply is a 5 V DC connection via pin 1 (VCC) and pin 10 (GND). The reader is standardized to ISO 15693 and ISO14443A/B and has the overall dimensions 36 x 16 x 4 mm [LxWxH][4].
The reader has three LEDs:
- Green: Run - Lights when reader receives power
- Yellow: Tag - Lights when a tag is detected
- Red: Data - Lights when data transfer to or from a tag
To configure the reader, KTS Systeme provides also a software (Tag2Image) for free. The reader was configured to scan the environment in an automatic anti collision mode (AT+Scan=AC,RSSI). Anti collision means that multiple tags can be detected at the same time and is highly important in this project. The output of the scan is a continuous information of the Identification (ID) and the Received Signal Strength Indicator (RSSI) of the detected tags. For example means: SCAN:+UID=E00402000018313E,+RSSI=7/6 that the tag with

---

[4]Stephan and Abdul
[5]Stephan

the ID (in hex) E00402000018313E was detected with a RSSI of 7/6. For the RSSI is the first number the value for the main and the second for the auxiliary receiver channel. In this project only the first number of the RSSI was used. The RSSI is an integer value from 0...7 and gives an information about the distance between the antenna and the detected tag. 0 stands for the maximum reading range which was mentioned to be around 15 cm. A detailed relation was figured out experimental during the project and will be explained later in this report. An AT Command Reference Guide is also available on http://rfid.kts-systeme.de/downloads/.

The antenna (fig. 8) is a HF PCB Antenne (PCBA1356_8) also from the company KTS Systeme. It has a dimension of 80 x 80 mm. The connection to the reader is realized by a SMA jack and has a self-impedance of 50Ω. The antenna is designed for passive tags in a frequency range around 13.56 MHz and has a maximum power of 1W.

Figure 8: RFID Antenna KTS Systeme PCBA1356_8

The antenna and the reader are connected with a SMA to U.FL. adapter cable.

5.2 RFID tag ?

5.3 Wifi modul (Abdul ?)

5.4 HW setup?!? (Abdul ?)

6   Simulation[6]

The simulation was carried out to answer important design questions before the real implementation phase. Another idea was also to create artificial RFID reader data to test and simulate the algorithm, which will be explained in chapter 7.

To answer the design questions, the simulation has these parameter (Appendix 11.1 Line 1-50):
- the size of the simulation space
- distance between the tags
- distance between the first/last row/column of tags and the boarder of the simulation space
- diameter of the robot
- position of the antenna related to the origin of the robot
- the relation between RSSI and the distance antenna and tag
- initial start position and orientation
- difference between the measurement points of the initialization procedure
- optional: cycle time and speed of the robot (for another procedure)
- logging parameter (look of the logged text file)

Foregone tests lead to a distance between the tags of 10 cm. This was founded on the fact that in this case at least 4 tags are detected at the same time (maximum reading range of 14 cm). In this case are around 121 tags needed for every square meter, which turned out to be realistic number for a small plant size.

6.1   Emulator

To create artificial RFID reader data, the emulator was able to write all detected tags together with information about the measuring point into a text file. During the initialization procedure, which was the main focus in this project, the robot turns around 360° and makes measurements every 45°.

The emulator computed at each measurement point the distance from the center of the antenna to the neighbouring tags. If a tag was closer than the maximal reading distance, the emulator wrote the detected ID of the tag together with its RSSI into the text file.

The RSSI is, as expalined earlier, an integer value from 0...7. 0 defines in this case a distance from 14 to around 10 cm from the antenna to the tag. In the first version of the emulator the RSSI was on the basis of the information from a paper [5] and mentioned a consistent increasing of the RSSI while the distance between the tags and the antenna gets smaller.

During own measurements has been found out that this relation was inconsistent. Therefore the second version of the emulator was updated and creates more realistic data.

---

[6]Stephan

6.2 RSSI Measurements with real hardware

The relation of the RSSI is not just related to the distance between the antenna and the tag. It also depends on the orientation of the plain of both components. The tests with the real hardware was performed in a setup where the tags was placed on a floor and the antenna was parallel to the floor at a hight of 1.5 cm. The reason for this was the fact that the antenna should be placed directly under the robot. Tbl. 6 and fig. 9 present the results of the measurements.

RSSI (Received Signal Strength Indicator)	0/0	1/1	2/2	3/3	4/4	5/5	6/6	7/7
Maximal distance antenna to tag [cm]	14	9.8	9	8	7	6	3.5	2.8
Middle distance antenna to tag [cm]	5	5.1	5.3	5.5	5.8	4	-	-
Minimal distance antenna to tag [cm]	-	4.7	4.5	4.3	4.2	-	-	-

Table 6: Relation between RSSI and distance antenna to tag (data)
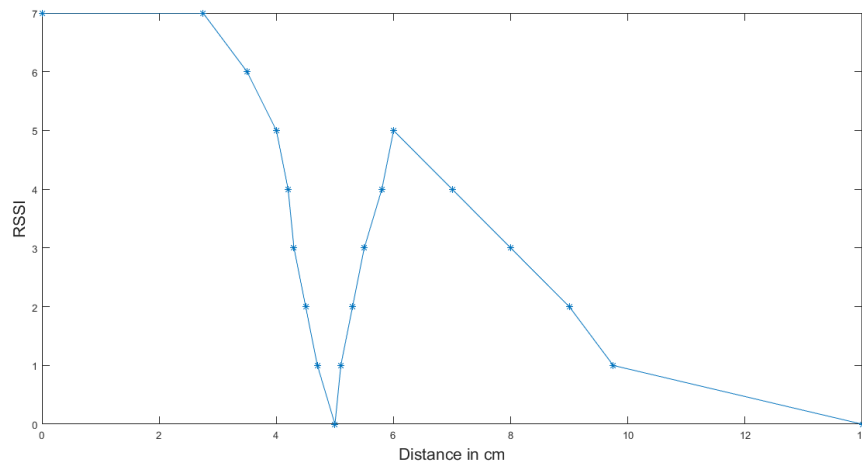
Figure 9: Relation between RSSI and distance antenna to tag

It can be seen that there exists a blind spot at a distance of 5 cm where the RSSI drops to 0. The consequence is now that it is not trivial to build up a relation from the RSSI back to the correct distance.

6.3 Simulation with emulated data

The idea of the final implementation is to estimate the initial position and orientation of the robot. A first version of an algorithm to solve this problem was created in matlab. The first part of these algorithm was the emulator which simulated the 360° turn and recorded the tag information. The second part was the solver which is also explained deeper in the chapter 7.
The first version of the solver which estimates the initial position and orientation based on the consistent RSSI data was quickly build up. After observing an inconsistent behaviour of the RSSI the simulation as well as the solver were updated.

6.4 Results

The application of the emulated data on the solver indicates the following results:

	Avg. accuracy position (x-, & y-direction) [mm]	Avg. Accuracy orientation [°]
Data mentioned in paper	2	<1
Own recorded data (blind spot)	10	20

Table 7: Results Simulation

As can be seen from tbl. 7, there is a sufficient good match between the estimated position and orientation of the robot for the consistent RSSI data. On the other hand results the inconsistent RSSI data in significant differences in the estimation of the position and orientation of the robot. The reason for this is the higher complexity of the algorithm to first estimate the correct distances related to RSSI values and then start to estimate the position based on those distances.
A small error in the estimation of the position of the antenna at the first measurement point leads also to a big error in the computed orientation of the robot.

7 Implementation

7.1 Communication (Abdul and/or Stefan)

7.2 Initialization procedure [7]

In the start-up phase, before running the pipe less plant with its AGVs the correct position and orientation of each and every vehicle is not known . Even though the controller is able to compute the position of the AGVs antenna in each point of time (t=0 included), several AGV positions in the plants operation space can be discribed by one single antenna position. In fig. 10 four possible AGV positions with one common antenna position are pointed out.
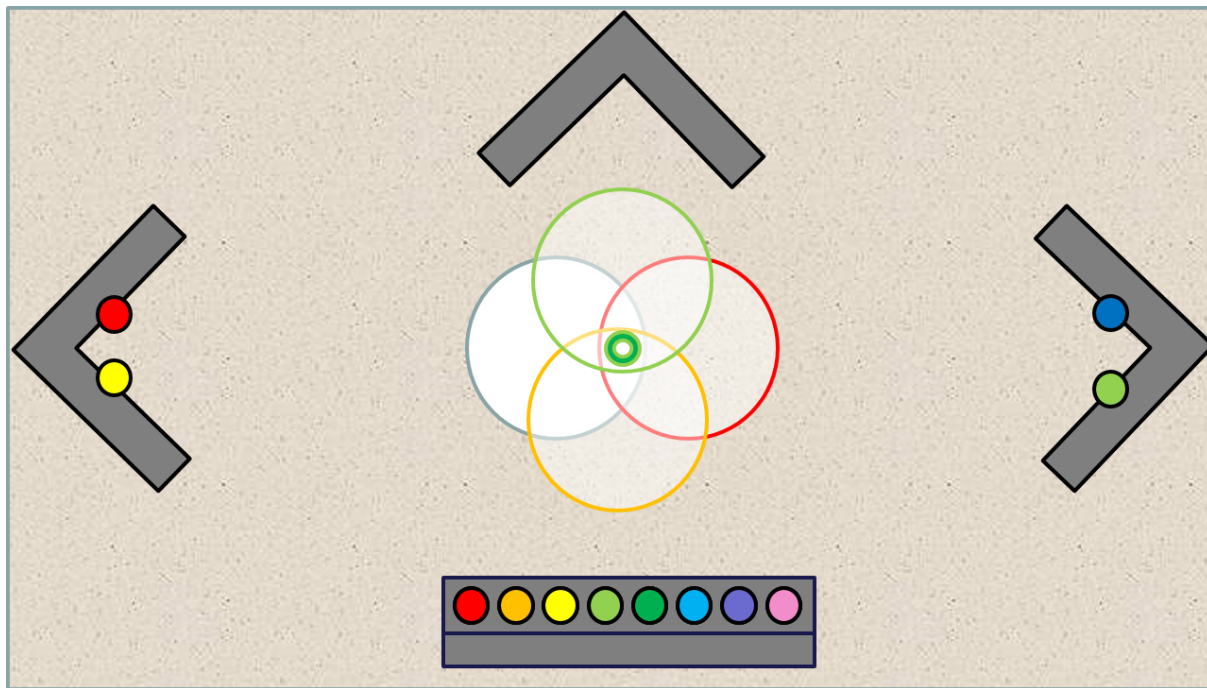

Figure 10: Different possible positions for one antenna position

Since this information is crucial for the plant, a procedure was set up to determine the starting positions of each and every AGV at time t=0. According to the fact, that the position and orientation of a single AGV is unknown at time t=0, some potential are taken into account. For instance, the plant contains several obstacles like the mixing stations, vessel storage, charging stations, plant edges and even other vehicles as represented in fig. 11.

---

[7]Stephan and Stefan
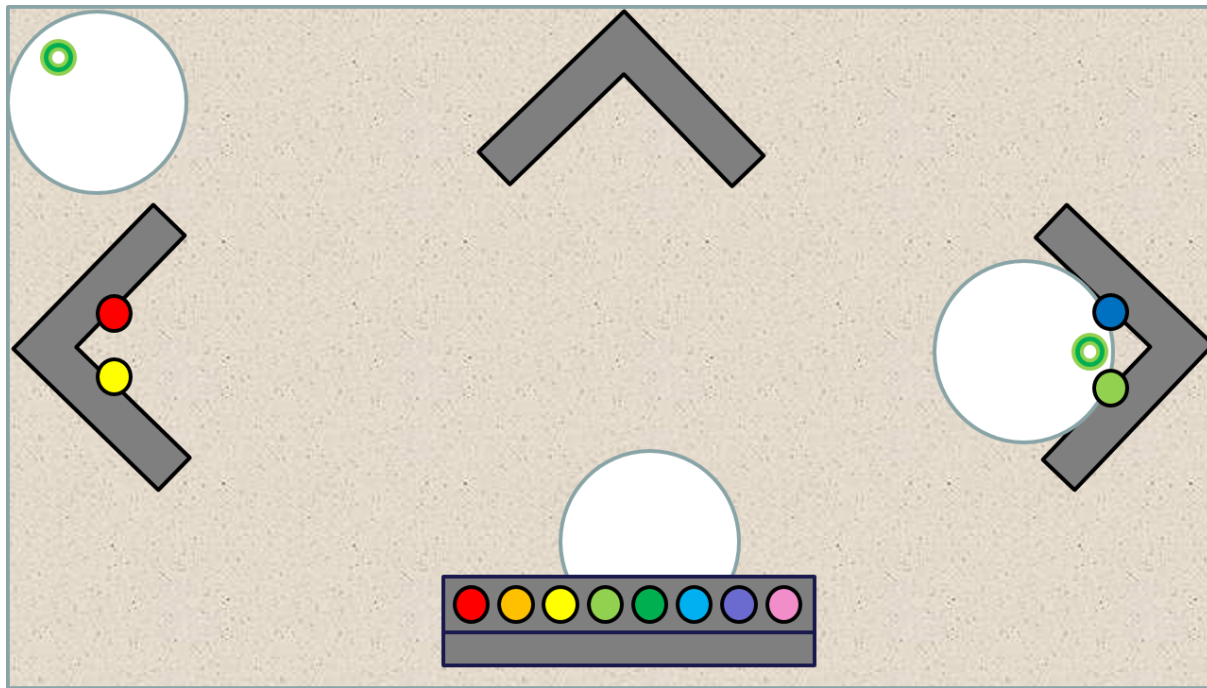
Figure 11: Possible hazards/obstacles

With respect to these potential hazards collisions during the initialization procedure have to be avoided. This is realized by taking advantage out of the AGVs ability to turn around its own axis without a change of the AGVs center point x and y position. This ability of the AGV leads the way that each and every robot performs an initialization turn of 360°in which measurements are taken every 45°to estimate the specific positions and orientations of the AGVs. Furthermore, the decision process of the antenna position under the robot is dominated by the fact that the position of the center point does not change during a turn around its own axis. During the 360°turn the intervals in which the measurments have to be taken need to be known by the controller. The determination of these measurement points can be computed in two different ways. On the one had the encoders of the AGV-wheels can be used to estimate the performed rotation. On the other hand, the time of a complete turn can be measured and used as a parameter in the procedure. In terms of simplicity the second option is used in the initialization procedure. fig. 12 represents a sequential flow chart which describes the movement and data processing during the initialization procedure.
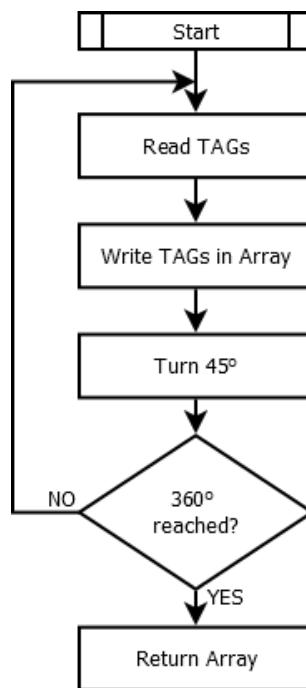
Figure 12: Flow Chart: Initial Procedure 360 degree turn

An initialization procedure for AGV No. 1 was created and is started in the GUI in the Test environment. In the first place an integer number is given to the field called Sleeptime. This integer number is interpreted as milliseconds and describes the time of rotation. Even though a time for a complete turn of 360°has been found at around 1125ms it has to be said that this time strongly depends on the battery charge of the AGV. After the desired turning time is given to the GUI the initialization is started by pushing the button Initialization, located over the input box 13.
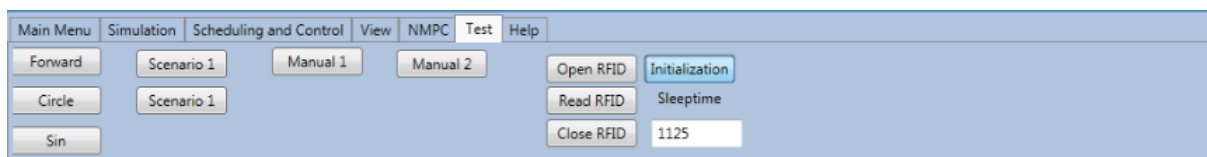
Figure 13: Test environment in GUI

In the second step after the procedure was started all the available IDs and their respective RSSI in the current reading range of the RFID-Reader are read. The reading is performed

in the Automatic Scan mode of the RFID reader. With the included timestamp for every measurement a delay of minimum 30ms between each and every TAG-Information was detected. With respect to this delay the antenna has to stop a specific period of time at each measuring point to deliver correct data of all the reachable TAGs. Experience has shown, that a measuring time between one and two seconds, in which every 100ms measurements are taken, the best results are delivered. In order to save the single TAG information of each and every measuring point an initially empty Array with 14 rows and 8 columns was created. The number of rows is derived by the fact that measurements are taken at every 45°.

$$Rows = 360/45 \tag{12}$$
$$Rows = 8 \tag{13}$$

The first seven entries of a row in the array is filled by the received TAG IDs and the last seven entries are filed by the respective RSSI.
The number of columns is derived by the fact that at each and every measurement point in the used test environment, information of maximal seven TAGs can be read.

$$Columns = max.no.ofTAGs * 2 \tag{14}$$
$$Columns = 7 * 2 \tag{15}$$
$$Columns = 14 \tag{16}$$

Once the received data is saved in its corresponding row, the AGV turns around 45°to place the antenna at the next measuring point. A AGV turn is realized by setting the velocity of the right and left wheel in different directions. During the turning sections the velocity is set to 100 mm/s or rather -100 mm/s. This procedure of reading information, writing information in the initialization array and turning 45°to the next measuring point is repeating itself until a 360°turn is performed. After a successful initialization turn the corresponding Array of measurement information can look like the example in table 8.

4	1	5	2	3			0	0	1	7	0		
5	3						2	3					
3	5						2	2					
9	8	6	5				1	1	1	2			
9	7	8	6	4	5		2	0	6	0	0	2	
4	7	5	8				0	2	3	3			
5	4	7	8	1			2	5	0	0	0		
2	4	1	5				0	2	2	0			

Table 8: Filled Array after 360 degree turn

7.2.1 Recording and filtering data [8]

To read the UID and RSSI of all the TAG laying in the reading range the RFID-Reader is set to its Automatic mode and its Anticollision is switched on. During this mode packages of strings with a length of 35 characters are reviewed by the plans computer. Even though these 36 character strings contain all the information of the TAG which is needed some effort has to be taken to seperate the useful parts which are prcessed in localization algorithm.

With exception of the information each and every string contains, the structure itselve is always the same. In the first five characters the substring "*SCAN:*" are found and deleted for the further process. The first important caracter is found in the sixth slot of the string. Here either a "*+*" or a "*-*" is written. With help of this sixth slot it is distriglished if rhe current reading is either a complete or incomplete one. In order to guarantee the correctness of the reciewed information the measurments are filtered by the "textit+" and the measurments in which a "-"' is included are ignored in the further processes. After the indicator for complete and in complete readings a introduction to the UID is indicated by "*UID=* " and cut out of the string. The next 16 caracters defines the unique identification of the specific TAG. As a last useless information which has to be cut out a string with the structure "*.RSSI=*" is found directly after the UID. As a result the sixteen caracter hexadecimal UID and its respective RSSI are seperated from the reviewed string. Since the ordered TAG UIDs differ each other just in the last tree numbers these numbers are transformed in a decimal number before UID and RSSI is used for further computations.

String Transformation	
Complete	**Incomplete**
SCAN:+UID=E00401503A5BD691,+RSSI=0/0	SCAN:-UID=E00401503A5BDAE4
UID=E00401503A5BD691,+RSSI=0/0	
E00401503A5BD691 0/0	
1681 0	

Table 9: String preperation

7.2.2 Analysing data[9]

In the next step of the Algorithm the previous described filled Array is analyzed. To estimate the position and orientation of the AGV the Array has to include two valid sets of each two valid measuring points. During this analyzation the single measurement point-sets are validated in terms of following restrictions:

1.: At the two valid measurement point each contains at least tree TAGs
2.: The two measurement points in one set needs to have a distance of 180°to each other.

---

[8]Stefan
[9]Stefan

In terms to get the adequate sets of measurment points the array is analyzed row by row. The stepwise workflow is vizialized in fig. 14.
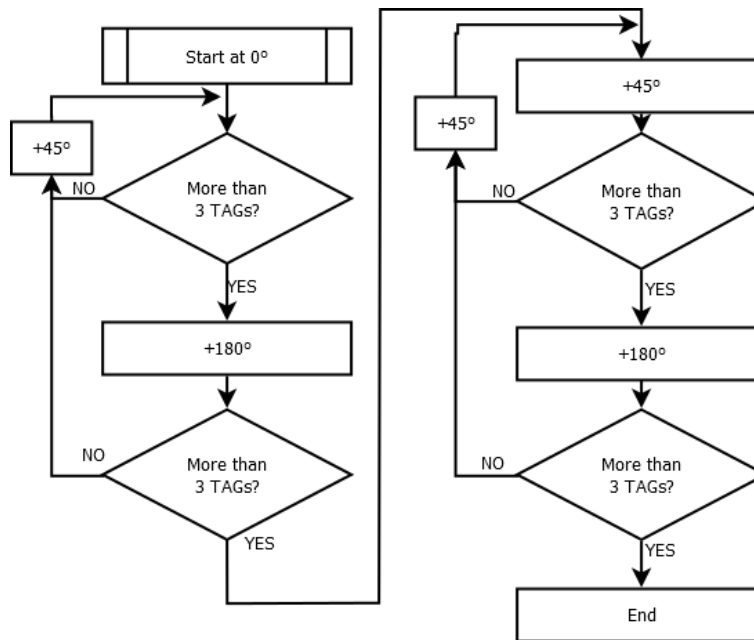

Figure 14: Flow Chart: Analizing Measurment Points

Initially the first row which represents the measurement at the point zero degree is checked in terms of the numbers of readable TAGs. If this specific number is higher or equal tree the transition is acknowledged as true and the same query will be performed at the measurement point with a distance of 180°to the former measurement point. If this next measurement point can be described as valid as well the first valid set of two measurement points is found. If, on the other hand, the number of readable TAGs are less than 3, which means that the triangulation algorithm cannot be performed, the current measurement point is ignored and the next point with a distance of 45°is evaluated. Each of this sets of two measurement points [degree] is saved as a 1x2 Array called Solution 1 and Solution 2 is used for the estimation of the position of the measurement points which is explained in the section 7.2.4 Estimation of initial position and orientation.

7.2.3 Selection of correct distance related to RSSI [10]

In a first step the multiple occurring data points (see tbl.6) are divided into three groups (max, middle and min) where max means the maximal possible distance related to one RSSI and so

---
[10]Stephan

on.

The measurements has shown that it is not trivial to define the correct distance related to most of the RSSI. The involved algorithm selects the correct distance out of the multiple possible solutions and is shown in fig. 15:
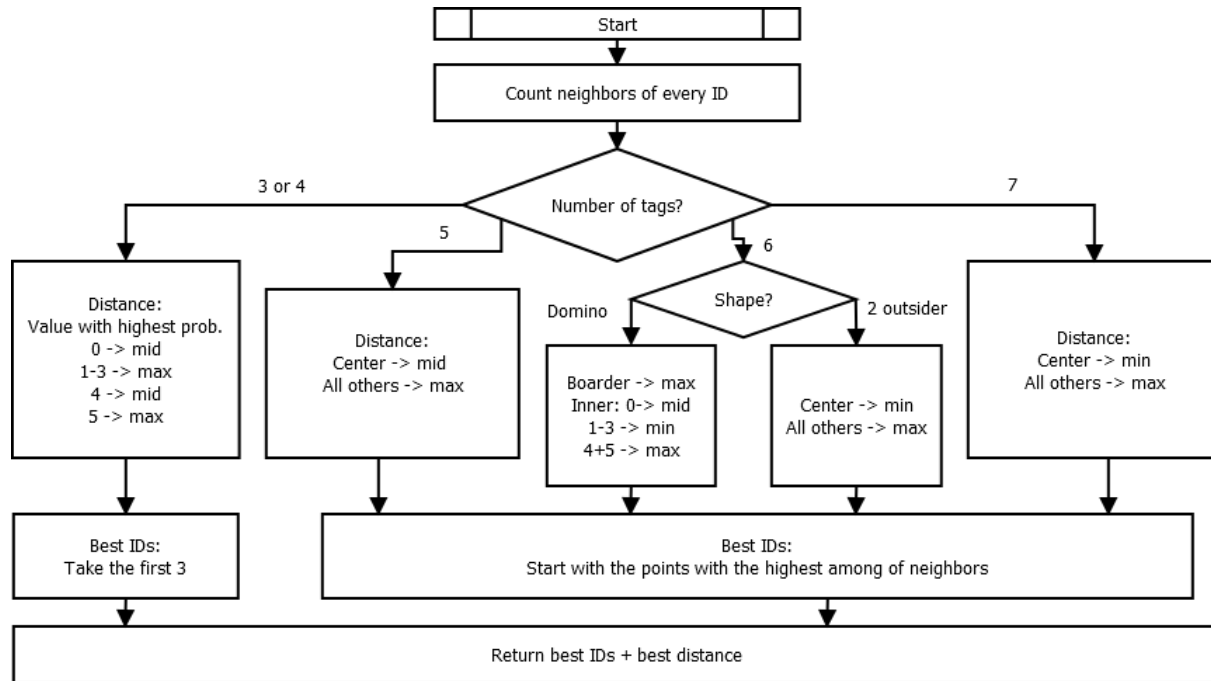

Figure 15: Flow Chart: Selection of correct distance and most proper IDs

To distinguish between the multiple possible solution for one RSSI, the algorithm defines the shape of the pattern of tags based on the number of tags at each measurement and the number of the neighbours each tag has. At each measurement point are in this scenario several numbers (4-7) of detected tags possible. The different shapes can be found in the tbl. 10.

Going back to the flow chart fig.15 the first step is to count the number of neighbours each tag has. With this information, the position of the tag in the pattern can be detected. For example is a tag with 3 neighbours in a pattern of 5 tags the center of this pattern.

After the number of tags at each measurement point and the position of each tag are defined, the selection of the correct distance will be performed based on the highest probability. To know the highest probabilities an analysis of measurements with emulated data has been done.

As an example are leading 4 detected tags to the fact that the position of the antenna should be very close to the center of this square. If in this case a RSSI of 4 is detected, the middle value (5.8 cm) will be taken.
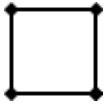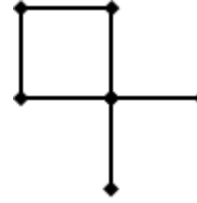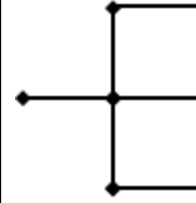
| Number of detected tags	4	5	6 (Domino)	6 (2 alone)	7
Unique shapes					

Table 10: Possible shapes of pattern

Afterwards the most suitable three IDs will be selected, in case where more then three are detected. The algorithm takes at first the ID with the highest amount of neighbours, because these tags are close to the position of the antenna and have probably a value of 6 or 7 and are uniquely defined. In the case where several tags with the same number of neighbours, the first ID (number increasing) will be taken.

The return of the function is an array (2x3) with the indices of the chosen IDs and the correct distance. The correct distance will be indicated by the number 0,1 and 2. 0 means the maximal, 1 the middle and 2 the minimum possible value related to one RSSI. For example leads

$$\begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 0 \end{bmatrix}$$

to the choice of the maximal value of the RSSI of the fourth detected ID and the minimum value of the RSSI of the third and the fifth ID in the recorded array at this measurement point.

7.2.4 Estimation of initial position and orientation [11]

As mentioned in chapter **??**, the main idea to estimate the initial position is to find the intersection point, which lies in the middle of the measurement points.

To compute this position, the algorithm uses trilateration at every suitable measurement point to estimate its position. For trilatertion are three defined positions plus three radii necessary, which are available after the selection of the correct distance and proper IDs.

As follows from the fig.16 shown above, the intersection point is found by computing two linear functions which go trough two corresponding points (blue lines). The center of the robot is then the intersection of those two linear functions and can be computed by the following equation:

$$x = \frac{(x_1 y_2 - y_1 x_2)(x_3 - x_4) - (x_1 - x_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{17}$$

$$y = \frac{(x_1 y_2 - y_1 x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{18}$$

---

[11]Stephan
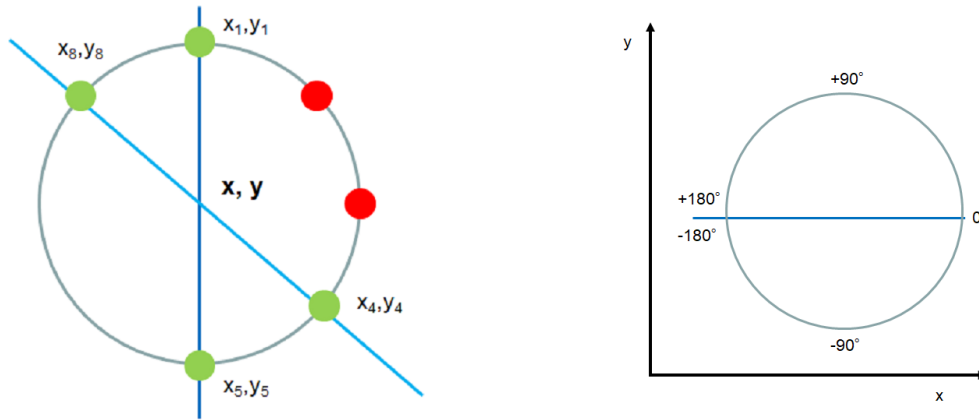
Figure 16: Computing the center of the robot Figure 17: Orientation of robot in absolute angle

Theoretical are all eight measuring points suitable points (at least four IDs found). But for the case that the real measurements differ from the theory, the algorithm just needs four suitable points.

After the initial position as well as the positions of 4 measurement points are known, the algorithm computes the orientation based on those information. The relative angle between the center and the first measurement point will be computed with the arctan2 function and leads to an orientation $-180° < \Theta \leq 180°$ as shown in fig.17.

To compute the absolute angle, the angle of the measurement point has to be subtracted and 180° has to be added. This is caused by the fact that the antenna is placed on the back of the robot and the absolute orientation should be the direction of the front. After this computation, the initial position and orientation of the robot are known.

7.3 Test setup[12]

In order to verify the validity of the initialization procedure, we carried out experiments with the components mentioned in chapter 5. The beginning of these experiments were the reconstruction of one of the AGVs with this HW setup. After we added all components to the AGV we realized the power supply via a powerbank and the USB connection of then wifi modul. The plan is to replace this in the future with a direct connection to the battery of the AGV. Fig.18 gives an overview of the test setup and shows also that for the prototype, the reader and the wifi modul was just stuck with Sellotape on the upper layer of the AGV.

The test platform was a field of 9 tags which were stuck on a piece of carton. The IDs and its positions are shown in tbl.11.

---

[12]Stephan
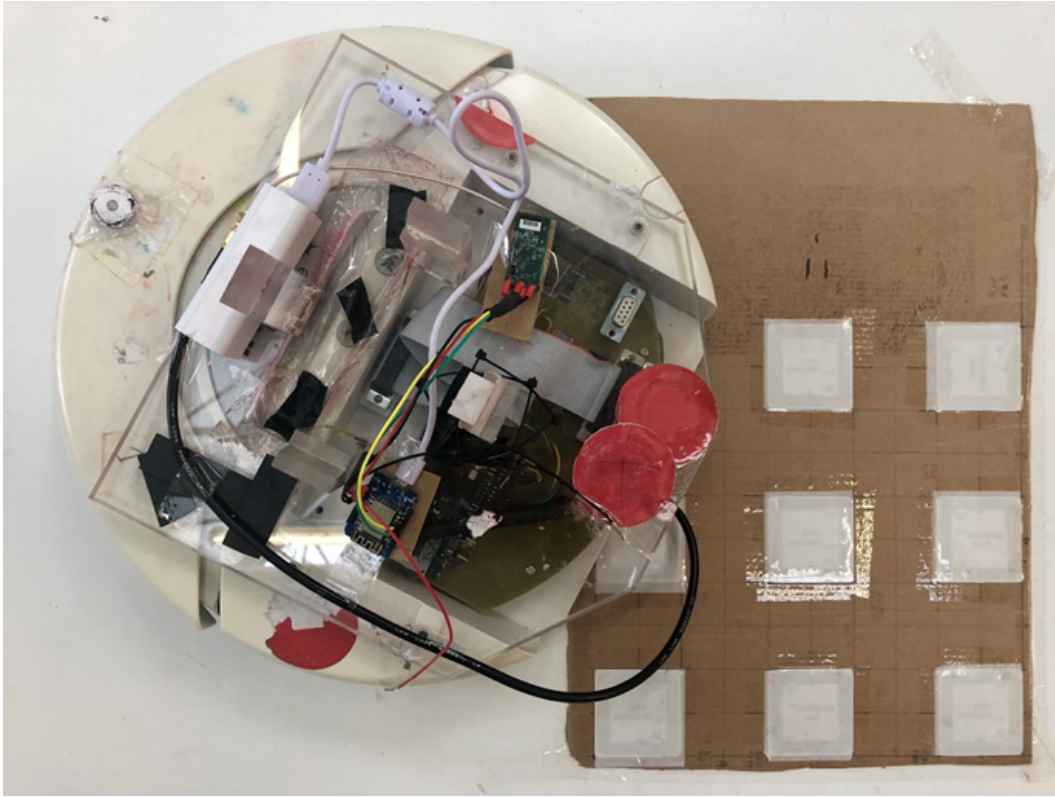
Figure 18: testing setup for initialization procedure

X-dir. [mm]	0	100	200	0	100	200	0	100	200
Y-dir. [mm]	0	0	0	100	100	100	200	200	200
ID tag [hex]	AE4	689	47A	586	785	ADC	BF4	691	78D
ID tag [dec]	2788	1673	1146	1414	1925	2780	3060	1681	1933

Table 11: Positions of the IDs in the test setup

The reason for the small setup was the fact that until the end of the project only 10 tags were available. One of the following steps should be to extend the platform with more tags.

The initialization procedure was started via the GUI. A time value was added in the GUI to perform the 45° turns. This number was around 1125 ms and is highly correlated to the battery status of the AGV.

7.4   Results[13]

A couple of tests on the test setup (previous section) were performed to compare the good results created with the simulated data with real measurements. The result of the position estimation was directly plotted in the console. The initial position was 200 mm in x- and y-direction and a varying orientation (0°, 90°, 180° and -90°). Fig.19 and fig.20 illustrate the actual measurement results and the desired position in x- and y-direction.
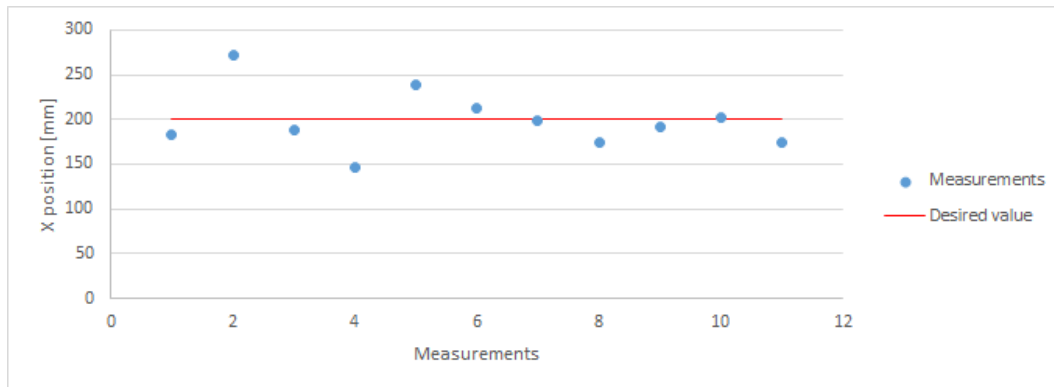
Figure 19: Estimated position in x-direction

The average of the absolute error of the position in x-direction was 24.5 mm. The minimum and maximum error were 2 mm and 72 mm.

The average of the absolute error of the estimation of the position in y-direction is with 23.3 mm, a minimum error of 3 mm and an maximum error of 77 mm very similar to the results from the estimation of the x-direction. The computation of the overall error of the position has an average derivation of 37.5 mm and a minimum and maximum error of 6.3 mm and 77 mm.

For the estimation of the orientation, the average of the absolute error was 23° with a minimum and a maximum value of 3.9° and 37.5°. The measurements also shows that an estimation of the position with a big error not necessarily leads to a big error in the estimation of the orientation (see measurement 4 in fig.19,20 and 21).

An extension of the results could also be an analyse of the estimated positions of the antenna

---

[13]Stephan
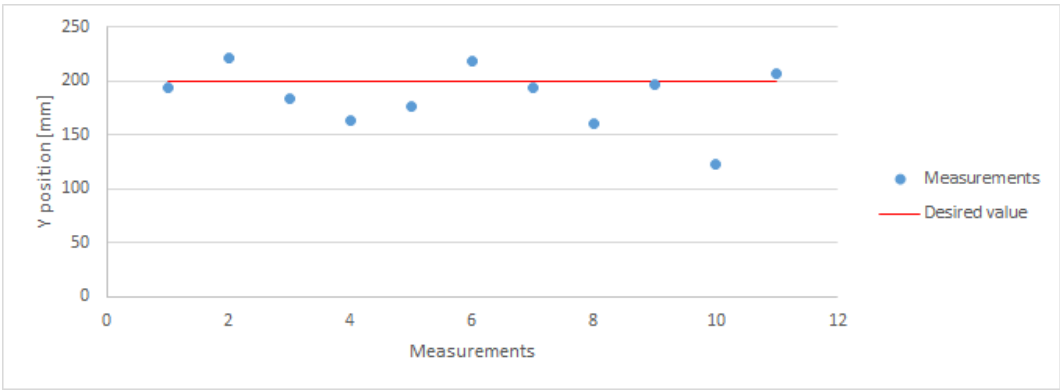
Figure 20: Estimated position in y-direction

Figure 21: Estimated orientation

at the measurement points. Those points were also plotted in the console.

8 Conclusion[14]

The developed localization solution was for the pipeless plant, a prototype of a chemical production plant which has a size of 3 by 4 meter. In this plant the vessel will be transported by AGVs from one station to another. In the actual setup only a camera, which is installed above the plant, was used to detected the AGVs and estimate their positions. The problem with this technology is the bad detection of the LED pattern from the AGVs during bright light conditions and also the space limitation. Another big disadvantage was the big computation effort which made the system also very slow. The main task of this project was to find an alternative tracking solution. During the project group phase differnt localization technologies were evaluated. With respect to the outcoming reseaches about Triangulation, Map-Based-Localizaion, Pattern recognizion and localizaion via Radio Frequency Identification the last RFID based localization of the AGV with passive tags as landmarks turned out to be the most promising among those four. With information of a similar project realized by the FH Dortmund a model to evaluate sample data and a localization algorithm was created in Matlab. This results of the simulation were promising and therefore used during the decision making process about the actual Hardware setup. With an demonstratino board with the size of 30 cm x 30 cm the initialization procedure algorithm was implemented in which the AGV performes and 360 degrees turn and estimates its position and its orientation based on measuremtns during this movement. With respect to this solutions it can be said that it is possible to assemble a reader on an AGV and detect passive TAGs with its antenna in a range of 14 cm. It also has been out that an inconsistent realation between the reviewed signal strength (RSSI) of the detected TAGs and the distance based on the RSSI is not generally trivial and was only solved in a rather simple and unriliable way during the project. Based on the results computed by the initializaion prcedure, it can be concluded that it is possible to estimate the position of the AGV with an average accuracy of aroung 2.5 cm and an estimation error of the orientation of around 23 degrees. Compared to the former localization set up this solutions, especially with respect to the orientation error, are not perfelty satisfying and just minimal requirements are fullfilled. The recieved data from the RFID reader have furthermore clearly shown that the anti-collision algorithm used by the reader leads to an unknown amount of time until each and every TAG in the detection area is identified. Summed up a model based demonstrator was realized which on the one hand does not improve the accuracy of the localizaion of the plant especially with respect to the orientation but on the other hand a promising technology for indoor localizaion with light independedcy, respectivelay cheap costs and highly scalability was found.

---

[14]Stephan and Stefan

9 Future Work[15]

After a proof-of-concept for an RFID based localizatin system has been build and a first demonstration set-up has been build the disadvantages and limitations of the prototype were evaluated. According to these results several points of improvement and extension were found and categorized into a hardware and a software section.

9.1 Hardware

- The AGVs are feed by an included 12V battery which provides the power for all included electronical deviced. This 12V power supply is available on board and is suggeted to be used. Currently the WiFi-Module and the RFID-Reader are fed by an external powerbank since a 5V power supply is needed. In terms of one zentralized power supply a 12 V to 5 V converter can be installed and connected to the new hardware.

- As a first set-up a demonstration area of 3 x 3 TAGs was developed. In this rather small area initializaion procedure was developed but a real time localization while a trajectory is followed by an AGV was not possible since the 30cm x 30cm was simply to small. For futrue research in terms of localizaion on a specified trajectory additional TAGs can be included to the area of operation. Since the RFID concept is highly scalable the only change that needs to be made in the algorithm is the insertion of the additional TAG into the lookup table.

- Currently the only Robot No. 1 is the only AGV which is equipped wich the RFID technolgoy. To run the plant wich multible AGVs the remaining robots needs to be upgraded.

9.2 Software

- During the Initalizatin procedure a 360 degres turn is performed. The degrees are devided by the change of degree over time. But it needs to be said that this movement is highly dependend on distrubances like changing battery charge and plant underground. For the future developers it is suggested to use the encoders of the robot weels as a determination of the orientation instead the parameter time.

- As an alternative localizatin technolgy was found several code lines in the VisualStudio can be deleted since the camera and image processing is simpls not used anymore. With a clean code an improvement of processing time will be accieved.

- As a last point it can be said that even though a localization with RFID is now possible the results are not 100 percent realiable and the accuracy especially with respect to the orientation is not satisfing so far. As an improvement the triangulation algorithm is to be optimized and or a second RFID-Antenna is to be added under the AGV to reduce measurment errors.

---

[15]Stefan

10   References

References

[1] Dr.-Ing. habil. Dipl.-Ing. Dipl.-Ing. Joerg Wollnack. Prinzip der dreidimensional messenden videometrischen messsysteme.

[2] Jeremie Houssineau, Daniel Clark, Spela Ivekovic, Chee Sing Lee and Jose Franco. A unified approach for multi-object triangulation, tracking and camera calibration.

[3] Pablo Cotera, Miguel Velazquez, David Cruz, Luis Medina, and Manuel Bandala. Indoor robot positioning using an enhanced trilateration algorithm. *International Journal of Advanced Robotic Systems*, 13(3):110, 2016.

[4] KTS Systeme. Rfid plug module rfidm1356, 2017.

[5] Christof Rohrig, Daniel Hess, and Frank Kunemund. Rfid-based localization of mobile robots rfid-based localization of mobile robots using the received signal strength indicator of detected tags.

11   Appendixes

11.1   Appendix A: Emulator RFID data (Matlab)

```matlab
1  %% ————————————————————————————————————————————————————————
2  % Description:   Emulator, which creates txt file like the reader
3  %                RSSI related to the real measurements
4  %                For the Initialization procedure, turn around 360°
5  % Date:          12.06.2018
6  % Created by:    Stephan Vette
7  % ————————————————————————————————————————————————————————
8  %% RFID signal emulator
9  clear all
10 clc
11 close all
12 % Initializing
13 l1 = 100;    % length of the plant, x   [cm]
14 l2 = l1;     % width of the plant, y    [cm]
15 d1 = 10;     % distance between tags     [cm]
16 d2 = 0;      % distance last tag <−> boarder [cm]
17 r1 = 14;     % radius of the reading range of every tag
18 r2 = [r1, 9.75, 9.0, 8.0, 7.0, 6.0, 5.8, 5.5, 5.3, 5.1, 5.0, 4.7, 4.5, 4.3, 4.2, 4.0,
         3.5, 2.75, 0]; % distances at certain RSSI
19 r4 = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 7]; % array with the
         different RSSI values
20
21 r3 = 33/2;  % radius of the robot
22 d3 = 10;     % distance between origin robot and origin antenna [cm]
23
24 angle1 = 45; % angle between the measurement points in the init procedure
25
26 gamma1 = deg2rad(22.5);  % Start orientation of robot [rad]
27 robStart = [22.5, 51.5]; % Start position of robot in x, y [cm]
28
29 robSpeed = 0.1;       % Speed robot [m/s]
30 cycleT = 100;         % Cycletime in [ms]
31
32 mode = 1;   % mode=1: tracking all available tags, which are nonzero
33             % mode=2: tracking only changes in the RSSI signals
34 mode_hex = 0;   % activate or deactivate hex ID
35
36 % For the name of the txt file
37 measuementeNumber = num2str(11); % Number of measurement
38 % Two possibilities for the content of the txt file
39 % 1. Without filtering. Exactly like the reader creates data
40 % text0 = '<\r>';
41 % text1 = 'OK';
42 % text2 = 'SCAN:+UID=';
43 % text3 = '+RSSI=';
44
45 % 2. Filtered data. Without unusable information.
46 text0 = ' ';
47 text1 = ' ';
48 text2 = ' ';
49 text3 = ' ';
50 %% Error check
51 if mod(l1/d1,1)~=0
52     error('Length of platform not dividable by distance between tags');
53 elseif mod(l2/d1,1)~=0
```

```matlab
54        error('Length of platform not dividable by distance between tags');
55   end
56
57   %% Computing position of antenna
58   numTagsX = (l1-2*d2)/d1 +1;
59   numTagsY = (l2-2*d2)/d1 +1;
60   numTags = numTagsX * numTagsY;
61   antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];
62
63   %% Display the setup, write important information into a seperate txt file
64   d1_str = num2str(d1);
65   l1_str = num2str(l1);
66   l2_str = num2str(l2);
67   numTags_str = num2str(numTags);
68
69   msg0 = ['Your plane is ',l1_str,'cm x ',l2_str,'cm.'];
70   msg1 = ['You chose a distance of ',d1_str, 'cm and need ', numTags_str,' Tags!'];
71   disp(msg0);
72   disp(msg1);
73   nameTxt = ['NumTags',measuementeNumber,'.txt'];
74   fileNumTags = fopen(nameTxt,'w');
75   fprintf(fileNumTags,'%6d\n',numTags);      % Write the number of tags in file
76   fprintf(fileNumTags,'%6d\n',l1);           % Write the size of the plant in file
77   fprintf(fileNumTags,'%6.4f\n',gamma1);        % Write the starting angle
78   fprintf(fileNumTags,'%6d\n',robStart(1));       % Write the starting pos
79   fprintf(fileNumTags,'%6d\n',robStart(2));       % Write the starting pos
80   fclose(fileNumTags);
81
82
83   %% Drawing environment
84   figure(1)
85   x1 = [0 l1 l1 0 0];
86   y1 = [0 0 l2 l2 0];
87   plot(x1, y1,'LineWidth',2)
88   xlim([-5 (l1+5)]);
89   ylim([-5 (l2+5)]);
90   hold on
91
92   % Position of the tags
93   ID = 1:numTags;
94   [Tagx,Tagy] = meshgrid(d2:d1:l1-d2,d2:d1:l2-d2);
95   plot(Tagx,Tagy,'r*')
96   % Circles
97   radiipl = ones(numTagsX,1)*r1;
98   for k=1:numTagsX
99       tempx = Tagx(1:end,k);
100      tempy = Tagy(1:end,k);
101      temppos = horzcat(tempx,tempy);
102      viscircles(temppos,radiipl,'Color','k','LineStyle',':','LineWidth',0.25);
103  end
104  robX = robStart(1);
105  robY = robStart(2);
106  plot(robX,robY,'bO','LineWidth',3);
107  plot(robX,robY,'r:');
108  viscircles([robX,robY],r3,'Color','k','LineWidth',0.25);
109  plot(antPos(1),antPos(2),'bs');
110  xlabel('Length platform in cm')
111  ylabel('Width platform in cm')
112  title({'Position and reading range of tags';'Start-, endpoint and path of the robot'});
```

```matlab
113  hold off
114  pause(1)
115
116  %% Animation and loggin
117  xUpdateAnt = antPos(1);
118  yUpdateAnt = antPos(2);
119  deltaR = deg2rad(angle1);          % A new measurement after every XX°
120  % Txt file name
121  name = ['Meas_StartingProc_like_reader_real_data',measuementeNumber,'.txt'];
122  fileID = fopen(name,'w');
123
124  % Data stored in variables
125  dataRSSI = zeros(8,numTags);
126  streamDataRSSI = zeros(1,numTags);
127  streamDataRSSIold = zeros(1,numTags);
128  timeStep = 1;    % current measurement step
129
130  % antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];
131  figure(2)
132  for l=0:360/angle1
133      deltaR_temp = deltaR * l;
134      xUpdateAnt = robStart(1) + d3 * cos(gamma1 + deltaR_temp);
135      yUpdateAnt = robStart(2) + d3 * sin(gamma1 + deltaR_temp);
136      plot(x1, y1,'LineWidth',2)
137      hold on
138      xlim([-5 (l1+5)]);
139      ylim([-5 (l2+5)]);
140      [Tagx,Tagy] = meshgrid(d2:d1:l1-d2,d2:d1:l2-d2);
141      plot(Tagx,Tagy,'r*')
142      plot(robX,robY,'bO','LineWidth',1);
143      plot(robX,robY,'r:');
144      plot(xUpdateAnt,yUpdateAnt,'bs');
145      xlim([-5 (l1+5)]);
146      ylim([-5 (l2+5)]);
147      viscircles([robX,robY],r3,'Color','b','LineWidth',0.5);
148          for k=1:numTagsX
149              tempx = Tagx(1:end,k);
150              tempy = Tagy(1:end,k);
151              temppos = horzcat(tempx,tempy);
152              viscircles(temppos,radiipl,'Color','k','LineStyle',':','LineWidth',0.25);
153          end
154      hold off
155
156      % Creating measurements
157      antPosnew=[xUpdateAnt,yUpdateAnt];
158      for m = 1:numTags                        % m = current number of tag
159          m_str = num2str(m);
160          tempTag=[Tagx(m),Tagy(m)];
161          tempD = pdist([antPosnew; tempTag] ,'euclidean');
162
163          % Display if tag is in range or not
164          if tempD > r1
165                  streamDataRSSI(m) = 0;
166                  if (streamDataRSSI(m) ~= streamDataRSSIold(m)) && mode == 2
167                      if mode_hex == 1
168                              fprintf(fileID ,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                                      text3,k(end),text0);
169                      elseif mode_hex == 0
170                              fprintf(fileID ,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
```

```matlab
                                                text0);
171                          end
172                            fprintf(' %d %d %1d\n',l*angle1,m,'0');
173                      end
174            elseif tempD <= r1
175                        % disp(['Label ',m_str,' in range !!!!!!!!!!!!!! ']);
176                        % Relation distance <-> RSSI
177                        k_temp = find(r2>=tempD);
178                        k = r4(k_temp);
179                        dataRSSI(timeStep,m) = k(end);
180                        streamDataRSSI(m) = k(end);
181                        if (streamDataRSSI(m) ~= streamDataRSSIold(m)) && mode == 2
182                            if mode_hex == 1
183                                fprintf(fileID,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                                        text3,k(end),text0);
184                            elseif mode_hex == 0
185                                fprintf(fileID,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
                                        text0);
186                            end
187                            fprintf(' %d %d %8d,\n',l*angle1,m,k(end));
188                        elseif mode == 1
189                            if mode_hex == 1
190                                fprintf(fileID,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                                        text3,k(end),text0);
191                            elseif mode_hex == 0
192                                fprintf(fileID,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
                                        text0);
193                            end
194                            fprintf(' %d %d %1d,\n',l*angle1,m,k(end));
195                        end
196                end
197        end
198        streamDataRSSIold = streamDataRSSI;
199        pause(cycleT/1000)
200        timeStep = timeStep + 1;
201  end
202  savefig('Figure2.fig');
203  fclose(fileID);
204
205  %%% Results
206  % figure(3)    % plot for the max value of every tag
207  % dataRSSInoT = reshape(max(dataRSSI),[numTagsX,numTagsY]);
208  % plot3(Tagx,Tagy,dataRSSInoT,'*');
209  % xlabel('Length platform in cm')
210  % ylabel('Width platform in cm')
211  % title('Max RSSI signal of every tag')
212
213  figure(4)    % plot of the RSSI signal which are non zero vs. time
214  dataRSSIsum = sum(dataRSSI);
215  IDclear = find(dataRSSIsum ~= 0);
216  IDstr = string(IDclear);
217  dataRSSIclear = dataRSSI;
218  dataRSSIclear( :, all( ~any( dataRSSI ), 1 ) ) = []; % and columns
219  plot(dataRSSIclear);
220  xlabel('Measurement points')
221  ylabel('RSSI')
222  ylim([0 360/angle1])
223  legend(IDstr,'FontSize',6);
224  title('RSSI Signal of every non zero tag')
```

11.2   Appendix B: Receiving data from reader via Wifi (C#)

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.IO;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using MULTIFORM_PCS.ControlModules.SchedulingModule;
using MULTIFORM_PCS.ControlModules.FeedForwadModule;
using MULTIFORM_PCS.ControlModules.RoutingModule.PathAndVelocityPlanning.
    DataTypes;
using MULTIFORM_PCS.ControlModules.CameraModule.CameraForm;
using MULTIFORM_PCS.ControlModules.CameraControl.CameraControlClass;
using System.Windows.Threading;
using System.Diagnostics; // Process
using System.Globalization;
using Emgu.CV.WPF;
using System.Threading.Tasks;
using System.Collections.Concurrent;

namespace MULTIFORM_PCS.ControlModules.RFID
{
    public class receive
    {
        public string[] availablearray=new string[1];
        public void connect()
        {
            try
            {
                Console.WriteLine("Connecting");
                TcpClient tcpClient = new TcpClient("192.168.0.100", 8883);
                if (tcpClient.Connected)
                {
                    Console.WriteLine("Connected to server");
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Connection Failed");
            }
        }

        public void reading(CancellationToken ct)
        {
            if (ct.IsCancellationRequested == true)
            {
                ct.ThrowIfCancellationRequested();
```

```
49                      }
50
51                      Console.WriteLine("Connecting");
52                      TcpClient tcpClient = new TcpClient("192.168.0.100", 8883);
53
54                      if (tcpClient.Connected)
55                      {
56                          Console.WriteLine("Connected to server");
57                      }
58
59                      using (StreamReader STR = new StreamReader(tcpClient.GetStream()))
60                      {
61                          string recieve;
62                          char[] trash = new char[16];
63                          char[] UID = new char[3];
64                          char[] RSSI = new char[3];
65                          long milliseconds, seconds, minutes;
66                          string UID_, RSSI_, RSSI___;
67                          string[] array = new string[1];
68
69                          List<string> RSSI__;
70                          int UID_DEC=0;
71                          int RSSI_int = 0;
72
73                              while ((recieve = STR.ReadLine()) != null && !ct.
74                                  IsCancellationRequested)
75                          {
76                              if (ct.IsCancellationRequested)
77                              {
78                                  try
79                                  {
80                                      ct.ThrowIfCancellationRequested();
81                                  }
82                                  catch (AggregateException e)
83                                  {
84                                  }
85                              }
86
87                              if (recieve.Contains("+"))
88                              {
89
90                              List<string> Worte = recieve.Split(new string[] { "OK",
91                                  "<\\r>", "\n", "", "SCAN:+UID=", "+RSSI=" },
92                                  StringSplitOptions.RemoveEmptyEntries).ToList();
93                              string Wort = string.Join("", Worte.ToArray());
94
95                              using (StringReader sr = new StringReader(Wort))
96                              {
                                    sr.Read(trash, 0, 13);
                                    sr.Read(UID, 0, 3);
                                    UID_ = new string(UID);
```

```
97                              sr.Read(trash, 0, 1);
98                              sr.Read(RSSI, 0, 1);
99                              RSSI_ = new string(RSSI);
100                             try
101                             {
102                                 UID_DEC = Int32.Parse(UID_, System.Globalization
                                        .NumberStyles.HexNumber);
103                             }
104                             catch (Exception e)
105                             {
106                             }
107                         }
108
109                         RSSI__ = RSSI_.Split(new string[] { "," },
                                StringSplitOptions.RemoveEmptyEntries).ToList();
110                         RSSI___ = string.Join("", RSSI__.ToArray());
111                         try
112                         {
113                             RSSI_int = Int32.Parse(RSSI___);
114                         }
115                         catch (Exception e)
116                         {
117                         }
118
119                         milliseconds = DateTimeOffset.Now.Millisecond;
120                         seconds = DateTimeOffset.Now.Second;
121                         minutes = DateTimeOffset.Now.Minute;
122                         array[0] = minutes + " " + seconds + " " + milliseconds
                                + " " + UID_DEC + " " + RSSI_int;
123                         //File.AppendAllText(AppDomain.CurrentDomain.
                                BaseDirectory + "\\pythonfiles\\python_1robot\\
                                RFID_Data.log", minutes + " " + seconds + " " +
                                milliseconds + "\t UID: " + UID_ + " RSSI: " +
                                RSSI___ + "\r");
124                         //File.AppendAllText(AppDomain.CurrentDomain.
                                BaseDirectory + "\\pythonfiles\\python_1robot\\
                                RFID_Data_original.log", hour + ":" + minutes + ":"
                                + seconds + ":" + milliseconds + "\t" + recieve + "\
                                r");
125                         //Console.WriteLine(minutes + " " + seconds + " " +
                                milliseconds + "\t" + " " + UID_ + "  " + RSSI___);
126                     }
127                         this.availablearray[0] = array[0];
128                 }
129             }
130         }
131
132
133
134     public void disconnect()
135     {
```

```
136                TcpClient tcpClient = new TcpClient();
137                tcpClient.Connect("192.168.0.100", 8883);
138                tcpClient.Close();
139
140            }
141    }
142    }
```

11.3   Appendix C: Initialization procedure (C#)

```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MULTIFORM_PCS.ControlModules.CameraModule.CameraForm;
using System.Threading;
using MULTIFORM_PCS.GUI;
using MULTIFORM_PCS.Gateway.ConnectionModule;
using MULTIFORM_PCS.ControlModules.RFID;
using System.Threading.Tasks;
using System.Collections;

namespace MULTIFORM_PCS.ControlModules.MPCModule
{
    public class Position
    {
        public int X = 0;
        public int Y = 0;
    }

    public class PositionD
    {
        public double X = 0;
        public double Y = 0;
    }

    class Init
    {
        public static void initialize(Int32 time)
        {
                int messungen = 100;
                //Gateway.ConnectionModule.ConnectionCTRLModule.getInstance().
                    setCTRLForRobot(0, 0.0, 100.0, 0.0, 8.0, 0.0, 0.0, 3.0);
                receive initial = new receive();    //Create a new instance of
                    class Receive
                var tokenSource = new CancellationTokenSource();
                var token = tokenSource.Token;
                Init compare = new Init();
                string[] rfid_signals = new string[messungen];
                int currentRobot = 0;
                int[] RobotAssingment = new int[] { 0, 1, 3 };
                Gateway.CTRLModule.getInstance().camCtrl.
                    processFrameAndUpdateGUI();
                RobotDiscription[] RobotArray = new RobotDiscription[] { Gateway
                    .CTRLModule.getInstance().camCtrl.RobotA, Gateway.CTRLModule
                    .getInstance().camCtrl.RobotB,
                Gateway.CTRLModule.getInstance().camCtrl.RobotC };
                double[][] velocity1 = new double[RobotArray.Length][];
                string[,] Signals = new string[messungen,8];
```

```
45                          velocity1[currentRobot] = new double[] { 0, 0 }; //Starts the
                                Robot
46                          Gateway.CTRLModule.getInstance().getRobotRemoteCTRL(
                                RobotAssingment[currentRobot]).forward(velocity1[
                                currentRobot], 0, 0, 0); //Sends velocity to Robot
47                          //Opening a new Task which works in the background to read data
                                from RFID Antenna
48                          Task t = Task.Factory.StartNew(() => initial.reading(token));
49                          Thread.Sleep(1000);
50                          for (int i = 0; i < 8; i++)        //9 Because of 8 measurements
                                every 45 degree
51                          {
52                              for (int j = 0; j < messungen; j++)     //in this for loop we
                                    find all the reachable TAGs
53                              {
54                                  Signals[j, i] = initial.availablearray[0];
55                                  Thread.Sleep(100);
56                              }
57                              velocity1[currentRobot] = new double[] { 100, -100 }; //
                                    Starts the Robot
58                              Gateway.CTRLModule.getInstance().getRobotRemoteCTRL(
                                    RobotAssingment[currentRobot]).forward(velocity1[
                                    currentRobot], 0, 0, 0); //Sends velocity to Robot
59                              Thread.Sleep(time); //time the robot needs for a 45 degree
                                    turn
60                              velocity1[currentRobot] = new double[] { 0, 0 };  //Stops
                                    the Robot
61                              Gateway.CTRLModule.getInstance().getRobotRemoteCTRL(
                                    RobotAssingment[currentRobot]).forward(velocity1[
                                    currentRobot], 0, 0, 0); //Sends velocity to Robot
62                          }
63
64                          tokenSource.Cancel();    //close the reading Thread
65                          try
66                          {
67                              Task.WaitAll(t);
68                          }
69                          catch (AggregateException e)
70                          {
71                          }
72                          finally
73                          {
74                              tokenSource.Dispose();
75                          }
76                              Console.WriteLine("END\r\r");
77
78                      Array[] Liste = new Array[8];    //List of arrays each array in the
                            array contains the data of a special position (45Â° , 90Â° ,...)
79                      string[,] Init_array = new string[8, 14];    //Array filled with
                            signal strengthes and ID of every degree position
80                      string temp_ID="begin", temp_RSSI;  //Substrings of Data
```

```csharp
81                 int counter;     //Counter for the row in the Init_Array
82                 Console.WriteLine("");
83
84                 for (int j = 0; j < 8; j++)
85                 {
86                     counter = 0;
87                     for (int i = 0; i < messungen; i++)
88                     {
89                         try
90                         {
91                             temp_ID = Signals[i, j].Substring(Signals[i, j].Length -
                                        6, 4);    //seperatiion of UID in the string
92                             if (temp_ID == "2788")
93                             {
94                                 temp_ID = "1";
95                             }
96                             if (temp_ID == "1414")
97                             {
98                                 temp_ID = "2";
99                             }
100                             if (temp_ID == "3060")
101                             {
102                                 temp_ID = "3";
103                             }
104                             if (temp_ID == "1673")
105                             {
106                                 temp_ID = "4";
107                             }
108                             if (temp_ID == "1925" || temp_ID == "1025")
109                             {
110                                 temp_ID = "5";
111                             }
112                             if (temp_ID == "1681")
113                             {
114                                 temp_ID = "6";
115                             }
116                             if (temp_ID == "1146")
117                             {
118                                 temp_ID = "7";
119                             }
120                             if (temp_ID == "2780")
121                             {
122                                 temp_ID = "8";
123                             }
124                             if (temp_ID == "1933")
125                             {
126                                 temp_ID = "9";
127                             }
128                         }
129                         catch (AggregateException e)
130                         {
```

```
131                         Console.WriteLine("Array incomplete");
132                     }
133
134                     temp_RSSI = Signals[i, j].Substring(Signals[i, j].Length -
                             1, 1);    //seperation of RSSI in the string
135                     if (temp_ID != Init_array[j, 0] && temp_ID != Init_array[j,
                             1] && temp_ID != Init_array[j, 2] && temp_ID !=
                             Init_array[j, 3] && temp_ID != Init_array[j, 4] &&
                             temp_ID != Init_array[j, 5] && temp_ID != Init_array[j,
                             6] && temp_ID != Init_array[j, 7])  //check if the UID
                             already exists in the Init_array
136                     {
137                         //Filling Init_Array
138                         Init_array[j, counter] = temp_ID;
139                         Init_array[j, counter + 7] = temp_RSSI;
140                         counter++;
141                     }
142                 }
143             }
144
145             int rowLength = Init_array.GetLength(0);
146             int colLength = Init_array.GetLength(1);
147             string str;
148             string headline = "|" + "ID 1" + "|" + "ID 2" + "|" + "ID 3" + "
                     |" + "ID 4" + "|" + "ID 5" + "| " + "ID 6" + "|" + "ID 7" +
                     "|" + "ST 1" + "|" + "ST 2" + "|" + "ST 3" + "|" + "ST 4" +
                     "|" + "ST 5" + "|" + "ST 6" + "|" + "ST 7" + "|";
149             System.Console.WriteLine(headline);
150
151             for (int k = 0; k < rowLength; k++)
152             {
153                 str = "|" + Init_array[k, 0] + "   |" + Init_array[k, 1] + "
                         |" + Init_array[k, 2] + "   |" + Init_array[k, 3] + "
                         |" + Init_array[k, 4] + "   |" + Init_array[k, 5] + "
                         |" + Init_array[k, 6] + "   |" + Init_array[k, 7] + "
                         |" + Init_array[k, 8] + "   |" + Init_array[k, 9] + "
                         |" + Init_array[k, 10] + "   |" + Init_array[k, 11] +
                     "   |" + Init_array[k, 12] + "   |" + Init_array[k, 13]
                         + "   |";
154                 System.Console.WriteLine(str);
155             }
156
157             // Solver
158             // Different Positions
159             Position Starting = new Position();
160             Position Antenna1 = new Position();
161             Position Antenna2 = new Position();
162             Position Antenna3 = new Position();
163             Position Antenna4 = new Position();
164
165             //Initialization for Position estimation
```

```
166                    float m1 = 0.000f;
167                    float m2 = 0.000f;
168
169                    float RobStartx_fl = 0.000f;
170                    float RobStarty_fl = 0.000f;
171
172                    double angle;
173                    double angleTemp;
174
175                    int null_counter = 0;
176                    int[] check_row = new int[8];
177                    for (int m = 0; m < rowLength; m++)
178                    {
179                        null_counter = 0;
180                        for (int n = 0; n < 7; n++)
181                        {
182                            if (Init_array[m, n] == null)
183                            {
184                                null_counter++;
185                            }
186                        }
187                        check_row[m] = 7 - null_counter;    //Array of elements with
                                the number empty places of each init_array row
188                        System.Console.WriteLine("The number of elements at " + m *
                                45 + "Â° is: \t" + check_row[m]);
189                    }
190                    bool solution_found = false;    //true if initialization process
                            is solvable
191                    bool solution1_found = false;    //true if one possible point is
                            found
192                    bool solution2_found = false;    //true if two possible points
                            are found
193                    int count = 0;
194                    int[] solution1 = new int[2];    //Array with the both degree
                            numbers of solution 1
195                    int[] solution2 = new int[2];    //Array with the both degree
                            numbers of solution 2
196                    while (solution_found == false)
197                    {
198                        while (solution1_found == false)
199                        {
200                            if (check_row[count] >= 3)
201                            {
202                                if (check_row[count + 4] >= 3)
203                                {
204                                    solution1[0] = count;
205                                    solution1[1] = count + 4;
206                                    break;
207                                }
208                                if (count >= 2)    //if we reach the 180 degree
                                    there will be no solution for this
```

```
                                  initialization turn
209                            {
210                                  System.Console.WriteLine("NO SOLUTION FOUND!!!")
                                         ;
211                                  break;
212                            }
213                        }
214                    count++;
215                }
216            System.Console.WriteLine(count);
217            count = count + 1;
218            while (solution2_found == false)
219            {
220                if (check_row[count] >= 3)
221                {
222                    if (count >= 8)
223                    {
224                        Console.WriteLine("Out of Range Exception caused
                                 in Array: count");
225                    }
226                    if (check_row[count + 4] >= 3)
227                    {
228                        solution2[0] = count;
229                        solution2[1] = count + 4;
230                        solution_found = true;
231                        break;
232                    }
233                    if (count >= 3)    //if we reach the 180 degree
                            there will be no solution for this
                            initialization turn
234                    {
235                        System.Console.WriteLine("NO SOLUTION FOUND!!!")
                                 ;
236                        break;
237                    }
238                    else
239                    {
240                        //count = count - 1;
241                        break;
242                    }
243                }
244            }
245        System.Console.WriteLine("Solution No. 1 found at: " +
                solution1[0] * 45 + " degree -- " + solution1[1] * 45 +
                " degree");
246        System.Console.WriteLine("Solution No. 2 found at: " +
                solution2[0] * 45 + " degree -- " + solution2[1] * 45 +
                " degree");
247            }
248
249            // Providing the distance with the highest probability
```

```
250                    // Input:   # of tags, all IDs of the tags
251                    // Output:  best fitting IDs (e.g.[3 4 5] if 3rd, 4th and 5th
                           are best ones)
252                    //          the correct distance <-> RSSI signal (e.g.[2 1 3]
                           for middle, max and min)
253                    int[,] best_arr1 = new int[2, 3];
254                    int[,] best_arr2 = new int[2, 3];
255                    int[,] best_arr3 = new int[2, 3];
256                    int[,] best_arr4 = new int[2, 3];
257
258                    int[] temp_input1 = new int[7];
259                    int[] temp_input2 = new int[7];
260                    int[] temp_input3 = new int[7];
261                    int[] temp_input4 = new int[7];
262
263                    int[] temp_inputRSSI1 = new int[7];
264                    int[] temp_inputRSSI2 = new int[7];
265                    int[] temp_inputRSSI3 = new int[7];
266                    int[] temp_inputRSSI4 = new int[7];
267
268                    for (int i = 0; i < 8; i++)
269                    {
270                        for (int j = 0; j < 14; j++)
271                        {
272                            if (Init_array[i, j] == null)
273                            {
274                                Init_array[i, j] = "0";
275                            }
276                        }
277                    }
278
279                    for (int m = 0; m < 7; m++)
280                    {
281                        temp_input1[m] = Int32.Parse(Init_array[solution1[0], m]);
282                        temp_input2[m] = Int32.Parse(Init_array[solution1[1], m]);
283                        temp_input3[m] = Int32.Parse(Init_array[solution2[0], m]);
284                        temp_input4[m] = Int32.Parse(Init_array[solution2[1], m]);
285
286                        temp_inputRSSI1[m] = Int32.Parse(Init_array[solution1[0], m
                               + 7]);
287                        temp_inputRSSI2[m] = Int32.Parse(Init_array[solution1[1], m
                               + 7]);
288                        temp_inputRSSI3[m] = Int32.Parse(Init_array[solution2[0], m
                               + 7]);
289                        temp_inputRSSI4[m] = Int32.Parse(Init_array[solution2[1], m
                               + 7]);
290                    }
291
292                    best_arr1 = CorrectID_Distance(temp_input1, temp_inputRSSI1,
                           check_row[solution1[0]]);
293                    best_arr2 = CorrectID_Distance(temp_input2, temp_inputRSSI2,
```

```
                           check_row [ solution1 [1]]) ;
294                best_arr3 = CorrectID_Distance ( temp_input3 , temp_inputRSSI3 ,
                           check_row [ solution2 [0]]) ;
295                best_arr4 = CorrectID_Distance ( temp_input4 , temp_inputRSSI4 ,
                           check_row [ solution2 [1]]) ;
296
297                // Position of the antennae
298                Antenna1 = Trilateration ( IDtoPOS ( Int32 . Parse ( Init_array [
                           solution1 [0] , best_arr1 [0 , 0]]) ) , IDtoPOS ( Int32 . Parse (
                           Init_array [ solution1 [0] , best_arr1 [0 , 1]]) ) ,
299                                        IDtoPOS ( Int32 . Parse ( Init_array [ solution1
                                            [0] , best_arr1 [0 , 2]]) ) , Int32 . Parse
                                            ( Init_array [ solution1 [0] , best_arr1
                                            [0 , 0] + 7]) ,
300                                        Int32 . Parse ( Init_array [ solution1 [0] ,
                                            best_arr1 [0 , 1] + 7]) , Int32 . Parse (
                                            Init_array [ solution1 [0] , best_arr1
                                            [0 , 2] + 7]) ,
301                                        best_arr1 [1 , 0] , best_arr1 [1 , 1] ,
                                            best_arr1 [1 , 2]) ;
302
303                Antenna2 = Trilateration ( IDtoPOS ( Int32 . Parse ( Init_array [
                           solution1 [1] , best_arr2 [0 , 0]]) ) , IDtoPOS ( Int32 . Parse (
                           Init_array [ solution1 [1] , best_arr2 [0 , 1]]) ) ,
304                                        IDtoPOS ( Int32 . Parse ( Init_array [ solution1
                                            [1] , best_arr2 [0 , 2]]) ) , Int32 . Parse
                                            ( Init_array [ solution1 [1] , best_arr2
                                            [0 , 0] + 7]) ,
305                                        Int32 . Parse ( Init_array [ solution1 [1] ,
                                            best_arr2 [0 , 1] + 7]) , Int32 . Parse (
                                            Init_array [ solution1 [1] , best_arr2
                                            [0 , 2] + 7]) ,
306                                        best_arr2 [1 , 0] , best_arr2 [1 , 1] ,
                                            best_arr2 [1 , 2]) ;
307
308                Antenna3 = Trilateration ( IDtoPOS ( Int32 . Parse ( Init_array [
                           solution2 [0] , best_arr3 [0 , 0]]) ) , IDtoPOS ( Int32 . Parse (
                           Init_array [ solution2 [0] , best_arr3 [0 , 1]]) ) ,
309                                        IDtoPOS ( Int32 . Parse ( Init_array [ solution2
                                            [0] , best_arr3 [0 , 2]]) ) , Int32 . Parse
                                            ( Init_array [ solution2 [0] , best_arr3
                                            [0 , 0] + 7]) ,
310                                        Int32 . Parse ( Init_array [ solution2 [0] ,
                                            best_arr3 [0 , 1] + 7]) , Int32 . Parse (
                                            Init_array [ solution2 [0] , best_arr3
                                            [0 , 2] + 7]) ,
311                                        best_arr3 [1 , 0] , best_arr3 [1 , 1] ,
                                            best_arr3 [1 , 2]) ;
312
313                Antenna4 = Trilateration ( IDtoPOS ( Int32 . Parse ( Init_array [
                           solution2 [1] , best_arr4 [0 , 0]]) ) , IDtoPOS ( Int32 . Parse (
```

```
                             Init_array[solution2[1], best_arr4[0, 1]]]),
314                                     IDtoPOS(Int32.Parse(Init_array[solution2
                                         [1], best_arr4[0, 2]])), Int32.Parse
                                         (Init_array[solution2[1], best_arr4
                                         [0, 0] + 7]),
315                                     Int32.Parse(Init_array[solution2[1],
                                         best_arr4[0, 1] + 7]), Int32.Parse(
                                         Init_array[solution2[1], best_arr4
                                         [0, 2] + 7]),
316                                     best_arr4[1, 0], best_arr4[1, 1],
                                         best_arr4[1, 2]);
317
318          Console.WriteLine("1st Antenna " + Antenna1.X + " and " +
                 Antenna1.Y);
319          Console.WriteLine("2nd Antenna " + Antenna2.X + " and " +
                 Antenna2.Y);
320          Console.WriteLine("3rd Antenna " + Antenna3.X + " and " +
                 Antenna3.Y);
321          Console.WriteLine("4th Antenna " + Antenna4.X + " and " +
                 Antenna4.Y);
322
323          //Console.ReadKey();
324          // Alternative estimation of the centre of the robot + position
325          //m1 = ((float)Antenna2.Y - (float)Antenna1.Y) / ((float)
                 Antenna2.X - (float)Antenna1.X);
326          //m2 = ((float)Antenna4.Y - (float)Antenna3.Y) / ((float)
                 Antenna4.X - (float)Antenna3.X);
327          //RobStartx_fl = (1 / (m1 - m2)) * (m1 * (float)Antenna1.X - m2
                 * (float)Antenna3.X - (float)Antenna1.Y + (float)Antenna3.Y)
                 ;
328          //RobStarty_fl = m1 * (RobStartx_fl - (float)Antenna1.X) + (
                 float)Antenna1.Y;
329
330          //Starting.X = (int)RobStartx_fl;
331          //Starting.Y = (int)RobStarty_fl;
332
333          Starting.X = (((Antenna4.X-Antenna3.X)*(Antenna2.X*Antenna1.Y-
                 Antenna1.X*Antenna2.Y)-(Antenna2.X-Antenna1.X)*(Antenna4.X*
                 Antenna3.Y-Antenna3.X*Antenna4.Y)) /
334                             ((Antenna4.Y - Antenna3.Y) * (Antenna2.X -
                                 Antenna1.X) - (Antenna2.Y - Antenna1.Y)
                                 * (Antenna4.X - Antenna3.X)));
335          Starting.Y = (((Antenna1.Y - Antenna2.Y) * (Antenna4.X *
                 Antenna3.Y - Antenna3.X * Antenna4.Y) - (Antenna3.Y -
                 Antenna4.Y) * (Antenna2.X * Antenna1.Y - Antenna1.X *
                 Antenna2.Y)) /
336                             ((Antenna4.Y - Antenna3.Y) * (Antenna2.X -
                                 Antenna1.X) - (Antenna2.Y - Antenna1.Y)
                                 * (Antenna4.X - Antenna3.X)));
337
338          Console.WriteLine("Robotstarting Position at:" + Starting.X + "
```

```
                              mm, " + Starting.Y + "mm");
339
340                 // Computing the orientation of the Robot
341                 //angle = (Math.Atan2(y, x)) * (180 / Math.PI);
342                 angleTemp = (Math.Atan2((Antenna1.Y - Starting.Y), (Antenna1.X -
                        Starting.X))) * (180 / Math.PI);
343                 Console.WriteLine("Angle temp " + angleTemp);
344                 angle = angleTemp - (double)(solution1[0]*45.0);  // in deg
345                 Console.WriteLine("Angle wrong direction " + angle);
346                 if (angle <= 0.0)
347                 {
348                     angle = angle + 180;
349                 }
350                 else
351                 {
352                     angle = angle - 180;
353                 }
354
355             Console.WriteLine("Robotangle: " + angle + "Â°");
356
357         }
358
359     // Procedure and function
360     // Method to compute the norm of a vector
361     public static double Norm(PositionD p) // get the norm of a vector
362     {
363         return (Math.Pow(Math.Pow(p.X, 2) + Math.Pow(p.Y, 2), 0.5));
364     }
365
366     //Methode to compute the position based on the ID in [cm], Output in [mm
            ]
367     public static Position Trilateration(Position point1, Position point2,
            Position point3, int r1t, int r2t, int r3t, int bestr1, int bestr2,
            int bestr3)
368     {
369         //double[] dist = new double[] { 10.5, 10.0, 9.5, 9.0, 8.0, 6.0,
                5.0, 4.0 }; // FH paper
370         double[,] dist = new double[3, 8] { { 14, 9.75, 9.0, 8.0, 7.0, 6.0,
                3.5, 2.75 },
371                                             { 5.0, 5.1, 5.3, 5.5, 5.8, 4.0,
                                                3.5, 2.75 },
372                                             { 5.0, 4.7, 4.5, 4.3, 4.2, 4.0,
                                                3.5, 2.75} }; //
                                                Approximation of our
                                                measurements
373
374         Position resultPose = new Position();
375         PositionD ex = new PositionD();
376         PositionD ey = new PositionD();
377         PositionD aux = new PositionD();
378         PositionD auy = new PositionD();
```

```
379                 PositionD aux2 = new PositionD();
380                 double r1;
381                 double r2;
382                 double r3;
383                 r1 = dist[bestr1, r1t];
384                 r2 = dist[bestr2, r2t];
385                 r3 = dist[bestr3, r3t];
386
387                 // For testing purpose
388                 //Console.WriteLine("1st radius " + r1);
389                 //Console.WriteLine("2nd radius " + r2);
390                 //Console.WriteLine("3rd radius " + r3);
391
392                 //Console.WriteLine("1st point " + point1.X + " " + point1.Y);
393                 //Console.WriteLine("2nd point " + point2.X + " " + point2.Y);
394                 //Console.WriteLine("3rd point " + point3.X + " " + point3.Y);
395
396                 //unit vector in a direction from point1 to point 2
397                 double p2p1Distance = Math.Pow(Math.Pow(point2.X - point1.X, 2) +
                        Math.Pow(point2.Y - point1.Y, 2), 0.5);
398                 ex.X = (point2.X - point1.X) / p2p1Distance;
399                 ex.Y = (point2.Y - point1.Y) / p2p1Distance;
400                 aux.X = point3.X - point1.X;
401                 aux.Y = point3.Y - point1.Y;
402                 //signed magnitude of the x component
403                 double i = ex.X * aux.X + ex.Y * aux.Y;
404                 //the unit vector in the y direction.
405                 aux2.X = point3.X - point1.X - i * ex.X;
406                 aux2.Y = point3.Y - point1.Y - i * ex.Y;
407                 ey.X = aux2.X / Norm(aux2);
408                 ey.Y = aux2.Y / Norm(aux2);
409                 //the signed magnitude of the y component
410                 double j = ey.X * aux.X + ey.Y * aux.Y;
411                 //coordinates
412                 double x = (Math.Pow(r1, 2) - Math.Pow(r2, 2) + Math.Pow(
                        p2p1Distance, 2)) / (2 * p2p1Distance);
413                 double y = (Math.Pow(r1, 2) - Math.Pow(r3, 2) + Math.Pow(i, 2) +
                        Math.Pow(j, 2)) / (2 * j) - i * (x / j);
414                 //result coordinates
415                 double finalX = 10 * (point1.X + x * ex.X + y * ey.X);
416                 double finalY = 10 * (point1.Y + x * ex.Y + y * ey.Y);
417                 resultPose.X = (int)(finalX);
418                 resultPose.Y = (int)(finalY);
419
420                 return resultPose;
421             }
422
423             // Methode to compute the position based on the ID in [mm]
424             public static Position IDtoPOS(int ID)
425             {
426                 Position FinalPos = new Position();
```

```csharp
427                 int[] posx = new int[9] { 10, 10, 10, 20, 20, 20, 30, 30, 30 };
428                 int[] posy = new int[9] { 10, 20, 30, 10, 20, 30, 10, 20, 30 };
429                 // For a 3x3 testing field
430                 FinalPos.X = posx[ID - 1];
431                 FinalPos.Y = posy[ID - 1];
432                 return FinalPos;
433             }
434
435             // Find neighbours of the IDs
436             public static int[] FindNeig(int[] arrID, int numTags)
437             {
438                 // Init
439                 int[] neighbours = new int[numTags];
440                 int[] tempNeig = new int[4];
441
442                 // Find the number of neighbours
443                 for (int m = 0; m < numTags; m++)
444                 {
445                     // Init
446                     neighbours[m] = 0;
447
448                     // Take actual ID and compute the possible neighbours
449                     tempNeig[0] = arrID[m] - 11;
450                     tempNeig[1] = arrID[m] - 1;
451                     tempNeig[2] = arrID[m] + 1;
452                     tempNeig[3] = arrID[m] + 11;
453
454                     for (int v = 0; v < 4; v++)
455                     {
456                         foreach (int tempinput in arrID)
457                         {
458                             if (tempinput == tempNeig[v])
459                             {
460                                 neighbours[m] += 1;
461                             }
462                         }
463                     }
464                 }
465                 return neighbours;
466             }
467
468             // Methode to compute the best IDs and correct distances
469             public static int[,] CorrectID_Distance(int[] arr, int[] arrRSSI, int
                    numTags)
470             {
471                 // Inputs
472                 /*   arr = Array of all IDs
473                  *   arrRSSI = Array of all RSSI
474                     numTags = Int with the num of tags found
475                 */
476                 // Init
```

```
477                 int [,] best = new int [2, 3];
478                 int [] dist = new int [numTags];              // Array which contain the
                        best distance (max(0),middle(1),min(2))
479                 int [] neighbours = new int [numTags];
480                 int i = 0;
481                 int p = 4;
482
483                 // Compute the neighbours
484                 neighbours = FindNeig (arr, numTags);
485
486                 // Switch case for the different possible shapes
487                 switch (numTags)
488                 {
489                     case 3:
490                         Console.WriteLine ("3 Tags ----------");
491                         for (int l = 0; l < neighbours.GetLength (0); l++)
492                         {
493                             if (neighbours [l] == 0)      // Detect outlier and
                                    boarder
494                             {
495                                 dist [l] = 1;              // stay max
496                             }
497                             else if (neighbours [l] <= 3)      // Detect outlier and
                                    boarder
498                             {
499                                 dist [l] = 0;              // stay max
500                             }
501                             else if (neighbours [l] == 4)      // Detect the inner,
                                    change it to min/middle
502                             {
503                                 dist [l] = 1;              // change it to middle
504                             }
505                             else if (neighbours [l] > 4)       // Detect the inner,
                                    change it to min/middle
506                             {
507                                 dist [l] = 0;              // change it to middle
508                             }
509                             // all other numbers are at the boarder
510                         }
511                         // Select the best 3 readings
512                         i = 0;
513                         p = 4;
514                         while (i < 3)                      // Start for the first ID
515                         {
516                             for (int h = 0; h < neighbours.GetLength (0); h++)    //
                                    looks for a fitting
517                             {
518                                 if (neighbours [h] == p && i < 3)      // hit must be
                                        same value and less then 3 hits
519                                 {
520                                     best [0, i] = h;                      // index of the
```

```
                                        best ID
521                             best[1, i] = dist[h];              // info about
                                    max, mid and min of this ID
522                             i += 1;
523                         }
524                         else if (i >= 3)
525                         {
526                             break;
527                         }
528                     }
529                     p -= 1;
530                 }
531             break;
532         /*
               -----------------------------------------------------------------------------
               */
533         case 4:
534             Console.WriteLine("4 Tags ----------");
535             for (int l = 0; l < neighbours.GetLength(0); l++)
536             {
537                 if (neighbours[l] == 0)      // Detect outlier and
                        boarder
538                 {
539                     dist[l] = 1;             // stay max
540                 }
541                 else if (neighbours[l] <= 3)     // Detect outlier and
                        boarder
542                 {
543                     dist[l] = 0;             // stay max
544                 }
545                 else if (neighbours[l] == 4)     // Detect the inner,
                        change it to min/middle
546                 {
547                     dist[l] = 1;             // change it to middle
548                 }
549                 else if (neighbours[l] > 4)     // Detect the inner,
                        change it to min/middle
550                 {
551                     dist[l] = 0;             // change it to middle
552                 }
553                 // all other numbers are at the boarder
554             }
555             // Select the best 3 readings
556             i = 0;
557             p = 4;
558             while (i < 3)                       // Start for the first ID
559             {
560                 for (int h = 0; h < neighbours.GetLength(0); h++)   //
                        looks for a fitting
561                 {
562                     if (neighbours[h] == p && i < 3)    // hit must be
```

```
                                            same value and less then 3 hits
563                                 {
564                                     best[0, i] = h;                    // index of the
                                                best ID
565                                     best[1, i] = dist[h];              // info about
                                                max, mid and min of this ID
566                                     i += 1;
567                                 }
568                                 else if (i >= 3)
569                                 {
570                                     break;
571                                 }
572                             }
573                             p -= 1;
574                         }
575
576                     break;
577                 /*
                    ----------------------------------------------------------------------------
                    */
578                 case 5:
579                     Console.WriteLine("5 Tags ----------");
580                     for (int l = 0; l < neighbours.GetLength(0); l++)
581                     {
582                         if (neighbours[l] <= 2)     // Detect outlier and
                                boarder
583                         {
584                             dist[l] = 0;             // stay max
585                         }
586                         else if (neighbours[l] == 3)    // Detect the inner,
                                change it to min/middle
587                         {
588                             dist[l] = 1;             // change it to middle
589                         }
590                         // all other numbers are at the boarder
591                     }
592                     // Select the best 3 readings
593                     i = 0;
594                     p = 4;
595                     while (i < 3)                         // Start for the first ID
596                     {
597                         for (int h = 0; h < neighbours.GetLength(0); h++)    //
                                looks for a fitting
598                         {
599                             if (neighbours[h] == p && i < 3)    // hit must be
                                    same value and less then 3 hits
600                             {
601                                 best[0, i] = h;                      // index of the
                                            best ID
602                                 best[1, i] = dist[h];                // info about
                                            max, mid and min of this ID
```

```
603                                    i += 1;
604                                }
605                                else if (i >= 3)
606                                {
607                                    break;
608                                }
609                            }
610                            p -= 1;
611                        }
612                        break;

614                /*
                    ----------------------------------------------------------------------------------
                    */
615                case 6:
616                    Console.WriteLine("6 Tags ----------");
617                    switch (neighbours.Sum())
618                    {
619                        case 12: // Shape with 2 outliers
620                            Console.WriteLine("2 Outliers");
621                            for (int l = 0; l < neighbours.GetLength(0); l++)
622                            {
623                                if (neighbours[l] <= 2)     // Detect outlier
                                    and boarder
624                                {
625                                    dist[l] = 0;             // stay max
626                                }
627                                else if (neighbours[l] == 4)    // Detect the
                                    inner, change it to min/middle
628                                {
629                                    if (arrRSSI[l] <= 3)
630                                    {
631                                        dist[l] = 2;          // change it to
                                            min
632                                    }
633                                    else if (arrRSSI[l] > 3)
634                                    {
635                                        dist[l] = 1;          // change it to
                                            middle
636                                    }
637                                }
638                                // all other numbers are at the boarder
639                            }
640                            break;
641                        case 14: // Shape like a domino
642                            Console.WriteLine("Domino");
643                            for (int l = 0; l < neighbours.GetLength(0); l++)
644                            {
645                                if (neighbours[l] <= 2)     // Detect outlier
                                    and boarder
646                                {
```

```
647                                    dist[l] = 0;                // stay max
648                                }
649                                else if (neighbours[l] == 3)    // Detect the
                                       centre, change it to min
650                                {
651                                    dist[l] = 2;                // change it to min
652                                }
653                                // all other numbers are at the boarder
654                            }
655                        break;
656                    default:
657                        Console.WriteLine("Default case");
658                        break;
659                }
660                // Select the best 3 readings
661                i = 0;
662                p = 4;
663                while (i < 3)                        // Start for the first ID
664                {
665                    for (int h = 0; h < neighbours.GetLength(0); h++)    //
                           looks for a fitting
666                    {
667                        if (neighbours[h] == p && i < 3)    // hit must be
                               same value and less then 3 hits
668                        {
669                            best[0, i] = h;                        // index of the
                                   best ID
670                            best[1, i] = dist[h];            // info about
                                   max, mid and min of this ID
671                            i += 1;
672                        }
673                        else if (i >= 3)
674                        {
675                            break;
676                        }
677                    }
678                    p -= 1;
679                }
680
681            break;
682            /*
                   ------------------------------------------------------------------------
                   */
683            case 7:
684                Console.WriteLine("7 Tags ----------");
685                for (int l = 0; l < neighbours.GetLength(0); l++)
686                {
687                    if (neighbours[l] <= 3)      // Detect outlier and
                           boarder
688                    {
689                        dist[l] = 0;              // stay max
```

```
690                         }
691                         else if (neighbours[l] == 4)        // Detect the centre,
                                  change it to min
692                         {
693                             dist[l] = 2;                    // change it to min
694                         }
695                         // all other numbers are at the boarder
696                     }
697                     // Select the best 3 readings
698                     i = 0;
699                     p = 4;
700                     while (i < 3)                           // Start for the first ID
701                     {
702                         for (int h = 0; h < neighbours.GetLength(0); h++)    //
                                  looks for a fitting
703                         {
704                             if (neighbours[h] == p && i < 3)    // hit must be
                                      same value and less then 3 hits
705                             {
706                                 best[0, i] = h;                     // index of the
                                          best ID
707                                 best[1, i] = dist[h];           // info about
                                          max, mid and min of this ID
708                                 i += 1;
709                             }
710                             else if (i >= 3)
711                             {
712                                 break;
713                             }
714                         }
715                         p -= 1;
716                     }
717                     break;
718                 default:
719                     Console.WriteLine("Default case");
720                     break;
721             }
722             foreach (int ee in best)
723             {
724                 Console.WriteLine(ee);
725             }
726             return best;
727         }
728     }
729 }
```