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(4511328)

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

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.

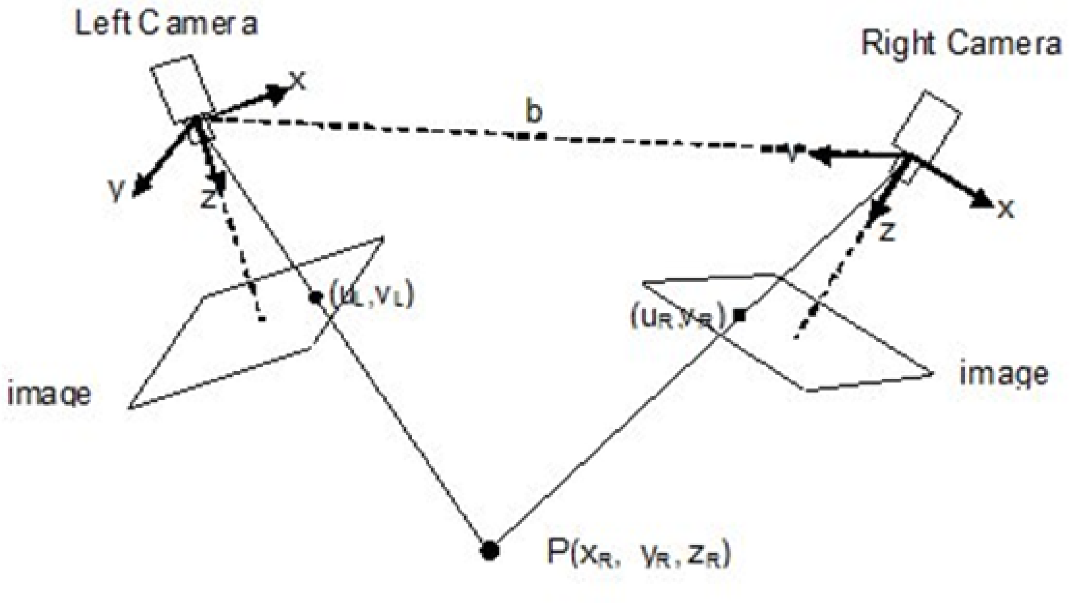


Figure 1: Passive Triangulation set-up with two cameras

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 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.

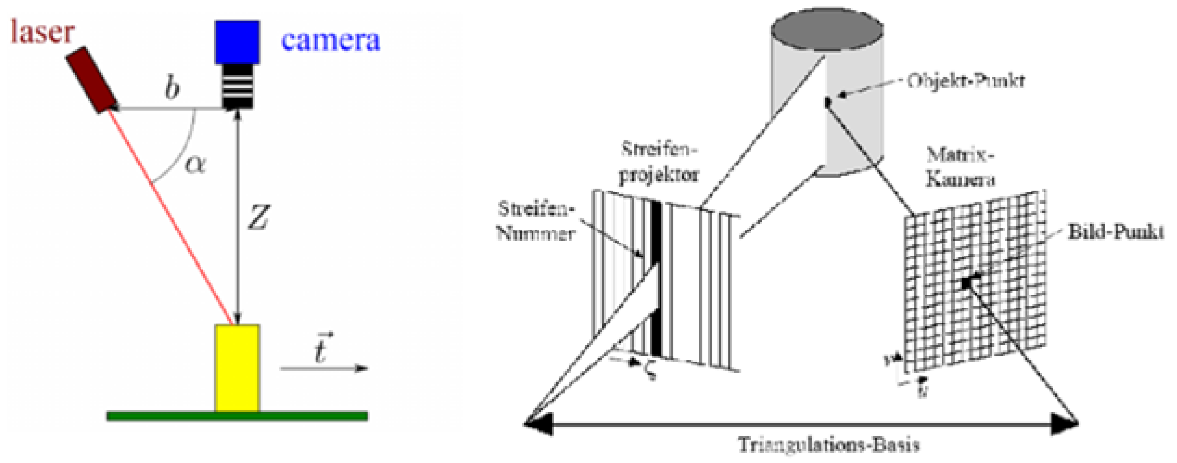


Figure 2: Active Triangulation

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. **??**.
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

3.2 Pattern Recognition

Summary

Implementation

Pro and con

..

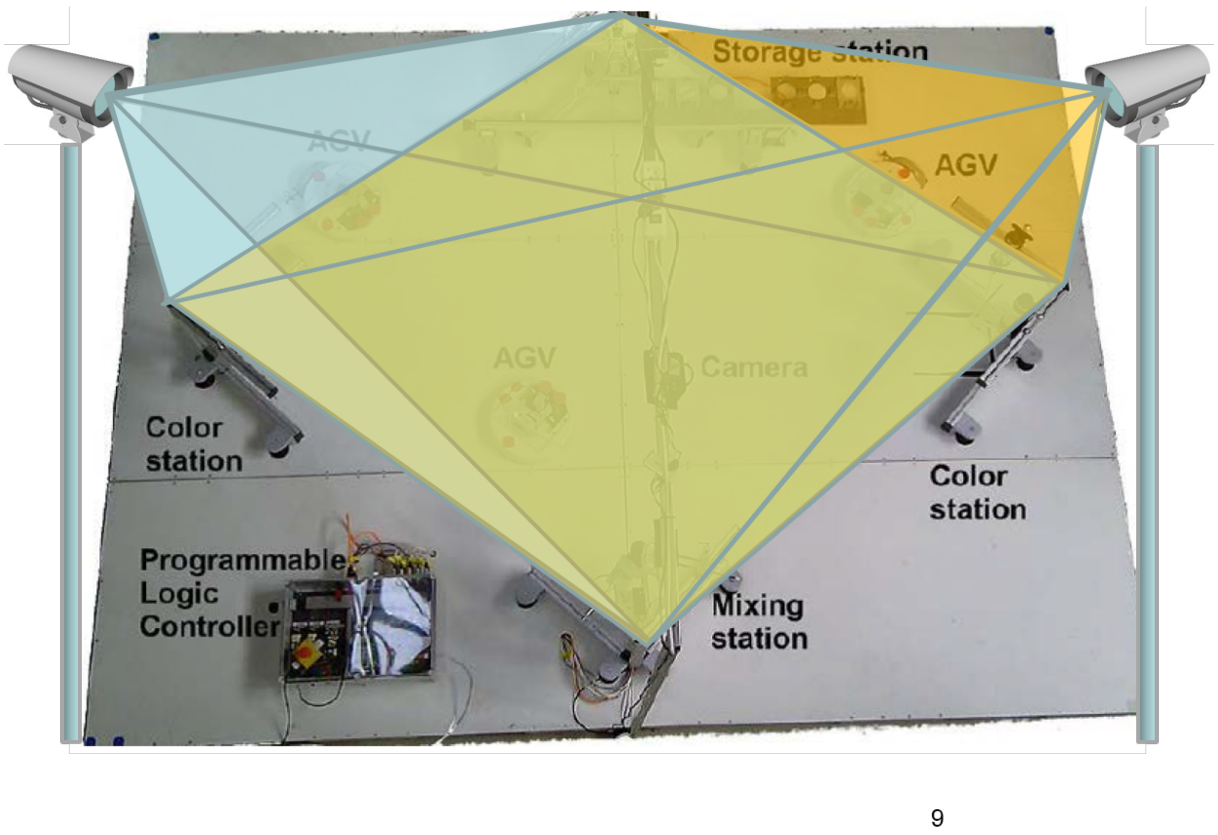


Figure 3: Impmentation of passive triangulation

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

3.3 RFID

Summary

Implementation

Pro and con

..

3.4 Map-Based Localization

Summary

Implementation

Pro and con

..
example:

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

Table 3: Should be a caption

4 Theoretical Background

4.1 Radio Frequency Identification (Abdul)

4.2 Trilateration[1]

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 [1]. 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.

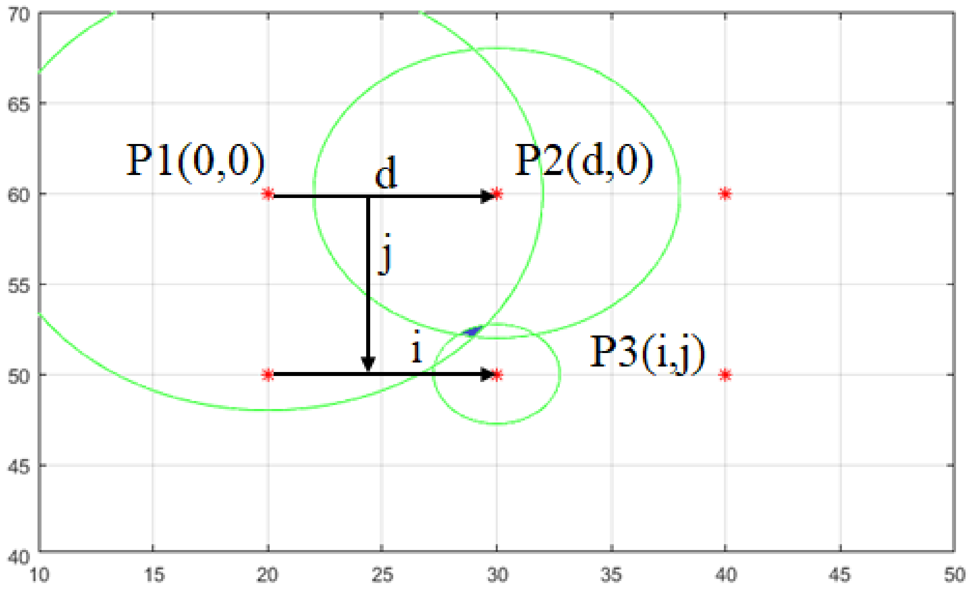


Figure 4: Overview Trilateration

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

[1]Stephan

way[1]:

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

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

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

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

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

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

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

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[2]

5.1 RFID reader and antenna[3]

The RFID reader from KTS Systeme (see fig.5) 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.

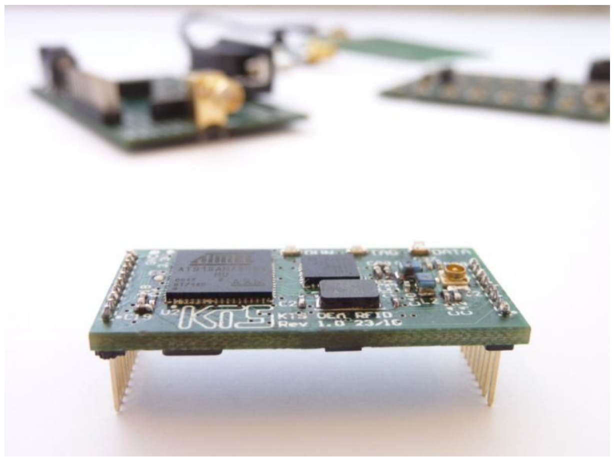


Figure 5: RFID reader KTS Systeme RFIDM1356-001

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][2].
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

[2]Stephan and Abdul
[3]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. 6) 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.

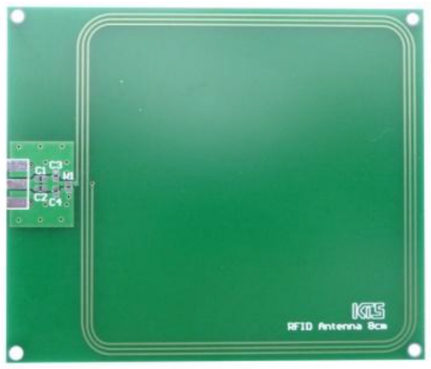


Figure 6: RFID Antenna KTS Systeme PCBA1356_8

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[4]

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 [3] 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.

[4]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. 4 and fig. 7 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 4: Relation between RSSI and distance antenna to tag (data)

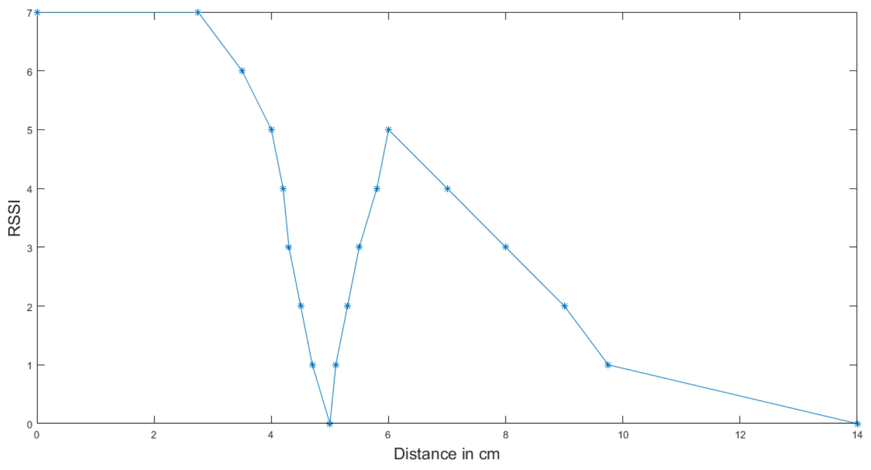


Figure 7: Relation between RSSI and distance antenna to tag

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 5: Results Simulation

As can be seen from tbl. 5, 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 (Stephan and Stefan)

In the start-up phase, before running the pipe less plant with its AGVs the controller initially does not know the correct position and orientation of each and every vehicle. Theoretically the controller is able to compute the position of the AGVs antenna in each point of time (t=0 included). But one initial antenna position can describe several AGV positions on the plants operation space. In fig. 8 four possible AGV positions with one common antenna position are pointed out.

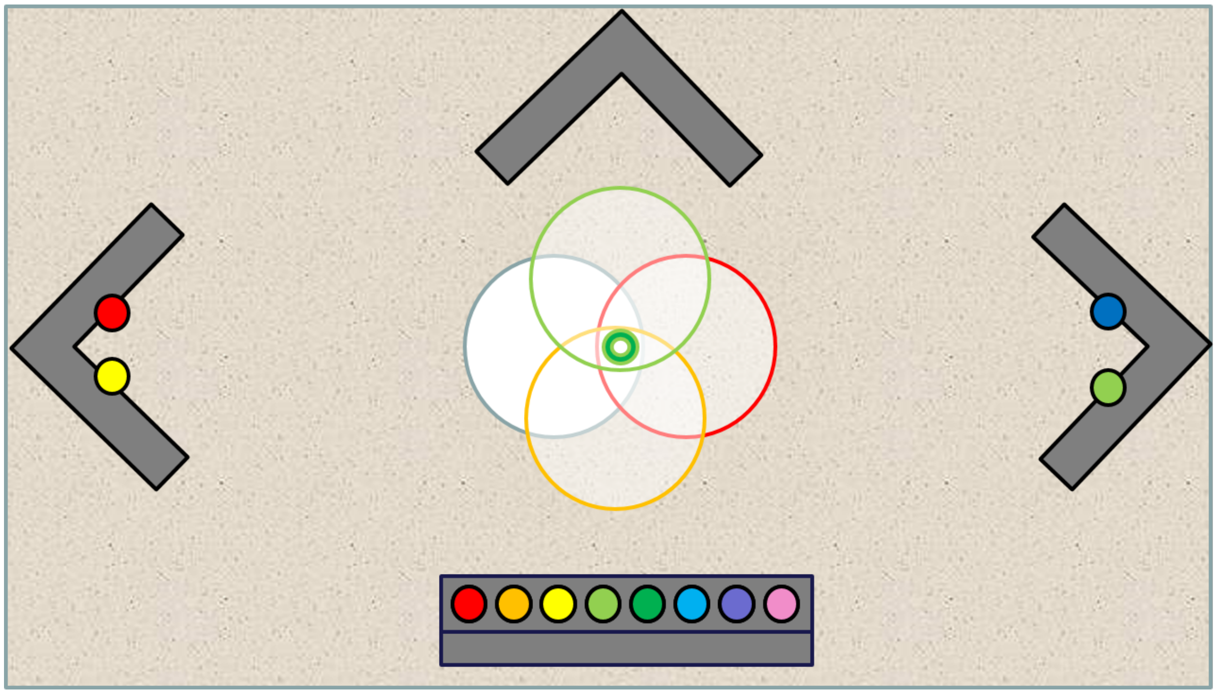


Figure 8: Different possible positions for one antenna position

Since this information is crucial for the plant, a procedure was set up to determine the starting positions of every AGV at t=0. According to the fact, that the position and orientation of a single AGV is unknown at t=0 some potential hazards have to be taken into account. For instance, the plant contains several obstacles like the mixing stations, vessel storage, charging stations, plant edges and even other vehicles shown in fig. 9.

With respect to these potential hazards collisions during the initialization procedure have to be avoided. This was realized by taking advantage out of the AGVs ability to turn around

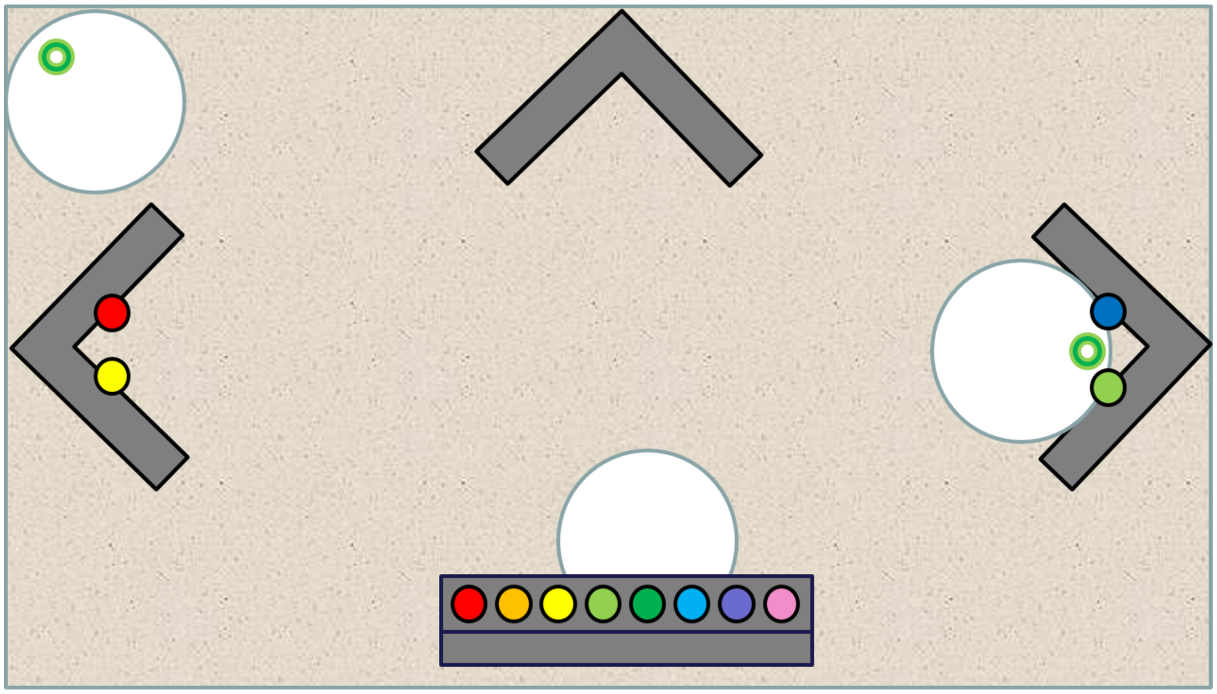


Figure 9: Possible hazards/obstacles

its own axis without changing the x,y position of the AGVÂ´s center point. This ability of the AGV lead the way that each and every robot performs an initialization turn of 360Â° in which measurements are taken every 45Â° to estimate the position and orientation of the AGVs. Additionally, the fact that the position of the center point does not change dominated the decision process of the antenna position under the robot. During the 360Â° turn the controller needs to know in which degree intervals measurements have to be taken. 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 was taken in initialization procedure. fig. 10 represents a sequential flow chart which describes the movement and data processing during the initialization procedure.

An initialization procedure for AGV No. 1 was created and can be 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 given in milliseconds and describes the time rotation. Even though a time for a complete turn of 360Â° 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

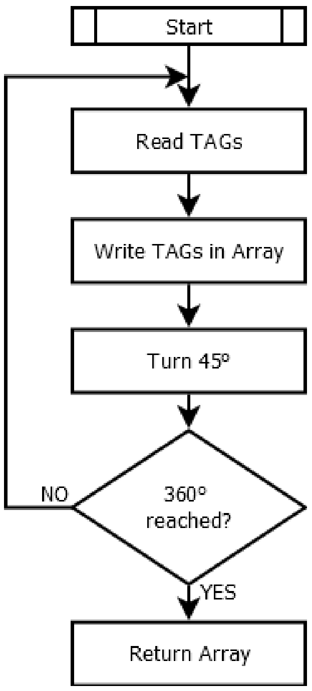


Figure 10: Flow Chart: Initial Procedure 360 degree turn

the input box.

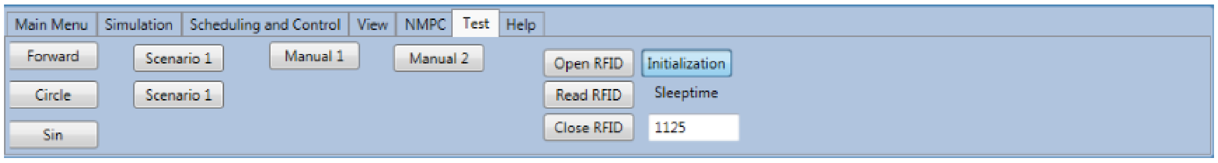


Figure 11: Test environment in GUI

In the second step after the procedure was started all the reviewed IDs and their respective RSSI from the RFID-Reader are read. The reading is performed with the Automatic Scan of the RFID reader. With the included timestamp for every measurement a measurement delay of minimum 30ms between each and every TAG 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, while 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 45Â°.

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

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 was saved in its corresponding row, the AGV turns around 45Â° 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 45Â° to the next measuring point is repeating itself until a 360Â° turn is performed. After a successful initialization turn the corresponding Array of measurement information can look like the example in table 6.

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 6: Filled Array after 360 degree turn

7.2.1 Recording and filtering data (Stefan)

To read the UID and RSSI from a TAG 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:* can be found and ignored for the further process. The first important caracter is found in the sixth slot of the string. Here either a + or - 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 recieved information the measurments are filtered by the + and the measurments where 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 last useless information which has to be cut out a string with the structure *.RSSI=* is found. As a result the 16 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.

.

7.2.2 Analysing data (Stefan)

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 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 at least tree TAGs should be read
2.: The two measurement points in one set needs to have a distance of 180Â° to each other.

In terms to get the adequate sets of measurment points the array is analyzed row by row. The stepwise workflow is vizialized in fig. 12.
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 equal as tree the transition is acknowledged as yes 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 was 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

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

Table 7: String preperation

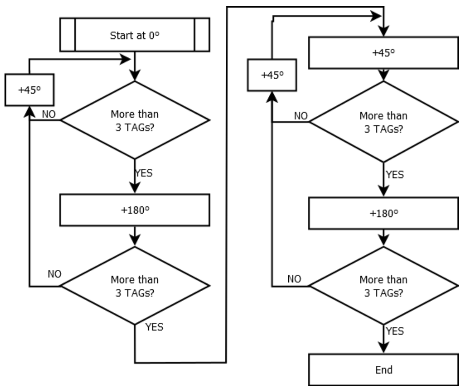


Figure 12: Flow Chart: Analizing Measurment Points

the next point with a distance of 45 degrees 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 footnoteStephan

In a first step the multiple occurring data points (see tbl.4) are divided into three groups (max, middle and min) where max means the maximal possible distance related to one RSSI and so 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. 13:

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. 8.

Going back to the flow chart fig.13 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.

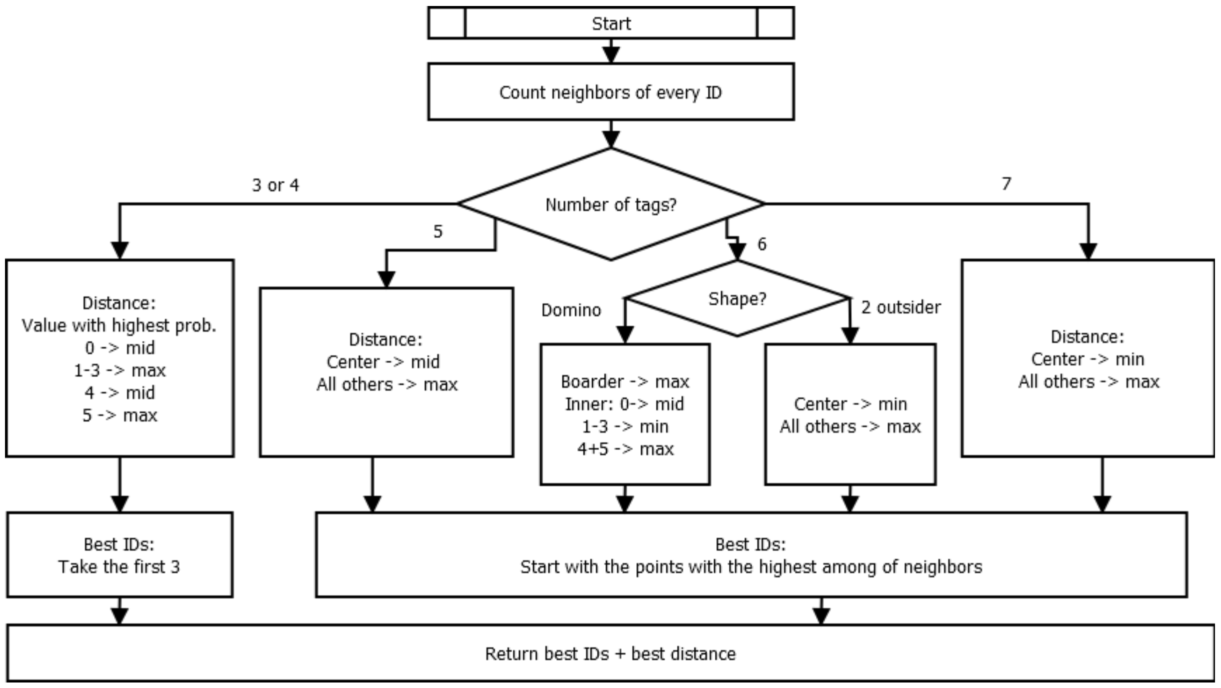


Figure 13: Flow Chart: Selection of correct distance and most proper IDs

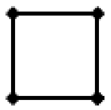
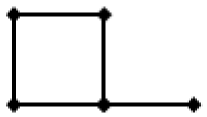
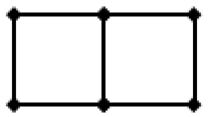
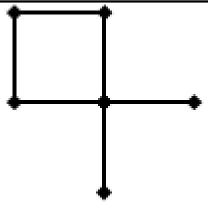
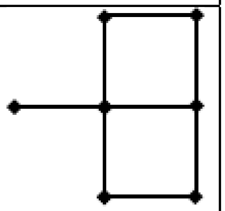
Number of detected tags	4	5	6 (Domino)	6 (2 alone)	7
Unique shapes					

Table 8: Possible shapes of pattern

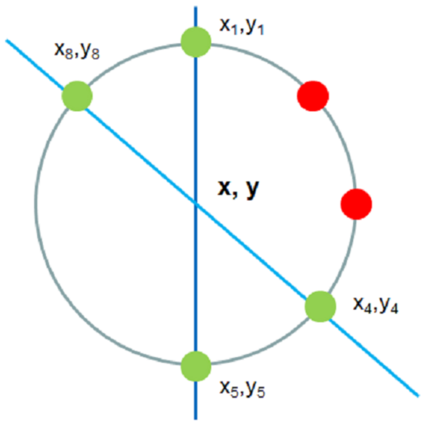
Afterwards the most suitable three IDs will be selected, in case where more then three are detected. The algorithm takes at first the ID with the highest amount of neighbours, because these tags are close to the position of the antenna and have probably a value of 6 or 7 and are uniquely defined. In 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 [5]

As mentioned in chapter 7.2, 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.



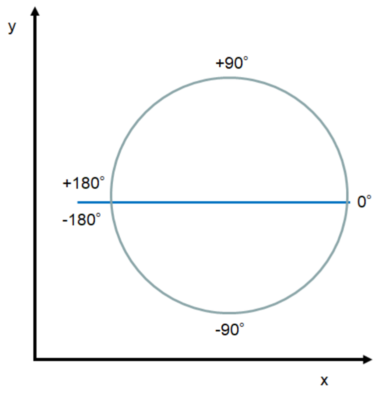


Figure 14: Computing the center of the robot Figure 15: Orientation of robot in absolute angle

As follows from the fig.14 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

[5]Stephan

the intersection of those two linear functions and can be computed by the following equation:

$$x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{17}$$

$$y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{18}$$

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.15.
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[6]

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.16 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.9.

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 9: 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.

[6]Stephan

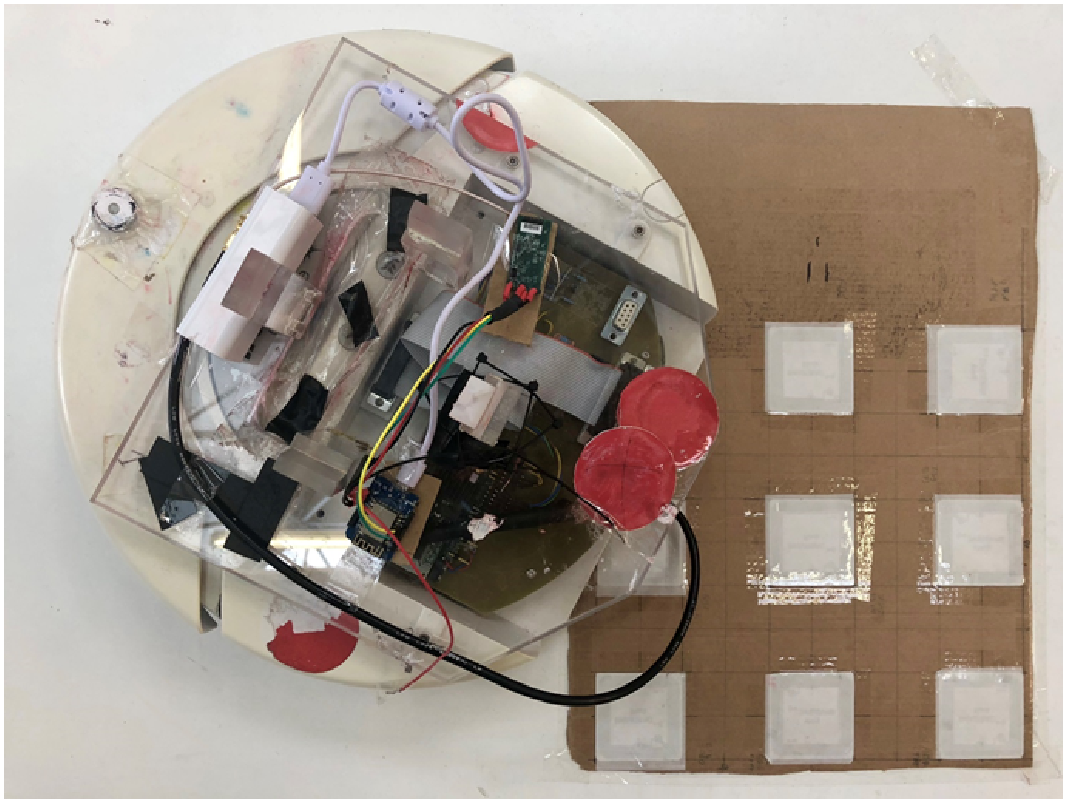


Figure 16: testing setup for initialization procedure

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 1150 ms and is highly correlated to the battery status of the AGV.

7.4 Results[7]

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.17 and fig.18 illustrate the actual measurement results and the desired position in x- and y-direction.

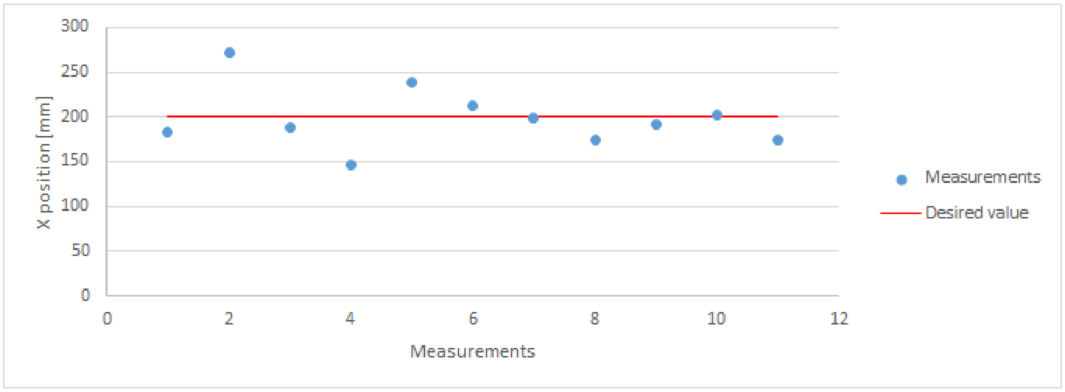


Figure 17: Estimated position in x-direction

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.17,18 and 19).
An extension of the results could also be an analyse of the estimated positions of the antenna at the measurement points. Those points were also plotted in the console.

[7]Stephan

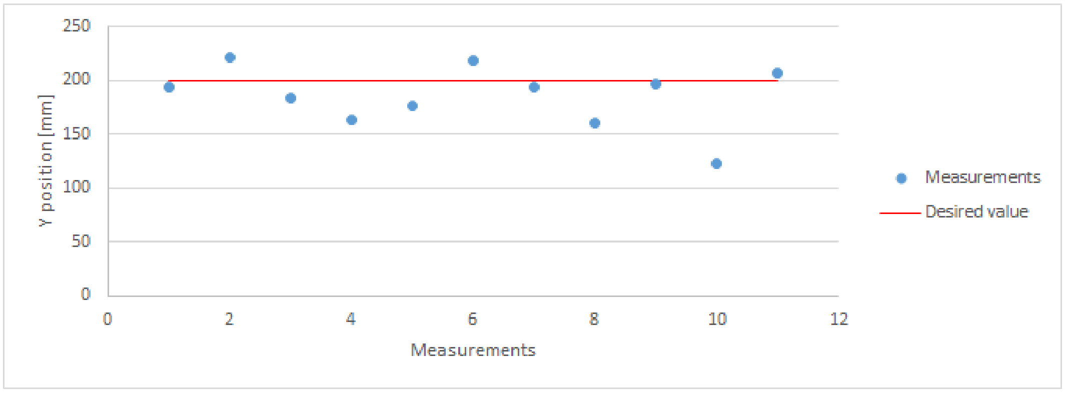


Figure 18: Estimated position in y-direction

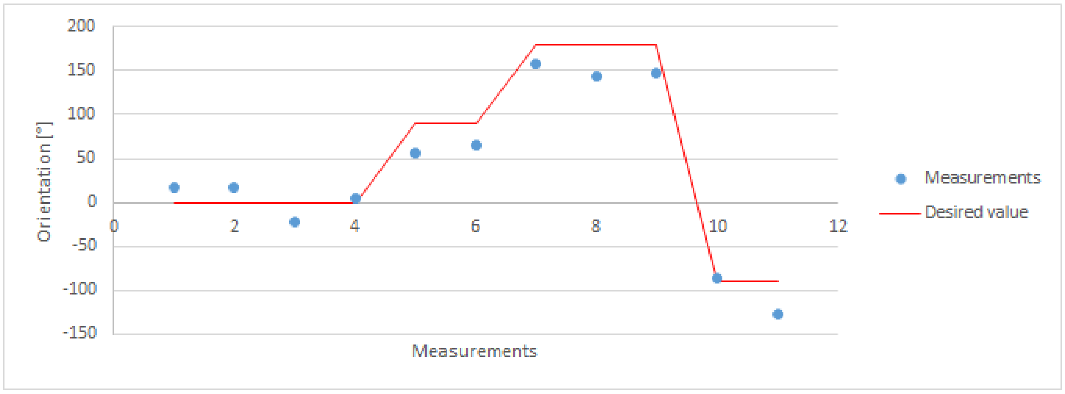


Figure 19: Estimated orientation

8 Conclusion

9 Future Work

...

10 References

References

[1] 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.

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

[3] 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)

```
%% ----------------------------------------------------------------------------
% Description:  Emulator, which creates txt file like the reader
%               RSSI related to the real measurements
%               For the Initialization procedure, turn around 360°
% Date:         12.06.2018
% Created by:   Stephan Vette
% ----------------------------------------------------------------------------
%% RFID signal emulator
clear all
clc
close all
% Initializing
l1 = 100;   % length of the plant, x   [cm]
l2 = l1;   % width of the plant, y    [cm]
d1 = 10;    % distance between tags      [cm]
d2 = 0;     % distance last tag <-> boarder [cm]
r1 = 14;    % radius of the reading range of every tag
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
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

r3 = 33/2;  % radius of the robot
d3 = 10;    % distance between origin robot and origin antenna [cm]

angle1 = 45; % angle between the measurement points in the init procedure

gamma1 = deg2rad(22.5);  % Start orientation of robot [rad]
robStart = [22.5, 51.5]; % Start position of robot in x, y [cm]

robSpeed = 0.1;       % Speed robot [m/s]
cycleT = 100;        % Cycletime in [ms]

mode = 1;   % mode=1: tracking all available tags, which are nonzero
            % mode=2: tracking only changes in the RSSI signals
mode_hex = 0;   % activate or deactivate hex ID

% For the name of the txt file
measuementeNumber = num2str(11); % Number of measurement
% Two possibilities for the content of the txt file
% 1. Without filtering. Exactly like the reader creates data
% text0 = '<\r>';
% text1 = 'OK';
% text2 = 'SCAN:+UID=';
% text3 = '+RSSI=';

% 2. Filtered data. Without unusable information.
text0 = ' ';
text1 = ' ';
text2 = ' ';
text3 = ' ';
%% Error check
if mod(l1/d1,1)~=0
    error('Length of platform not dividable by distance between tags');
elseif mod(l2/d1,1)~=0
```

```
    error('Length of platform not dividable by distance between tags');
end

%% Computing position of antenna
numTagsX = (l1-2*d2)/d1 +1;
numTagsY = (l2-2*d2)/d1 +1;
numTags = numTagsX * numTagsY;
antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];

%% Display the setup, write important information into a seperate txt file
d1_str = num2str(d1);
l1_str = num2str(l1);
l2_str = num2str(l2);
numTags_str = num2str(numTags);

msg0 = ['Your plane is ',l1_str,'cm x ',l2_str,'cm.'];
msg1 = ['You chose a distance of ',d1_str, 'cm and need ', numTags_str,' Tags!'];
disp(msg0);
disp(msg1);
nameTxt = ['NumTags',measuementeNumber,'.txt'];
fileNumTags = fopen(nameTxt,'w');
fprintf(fileNumTags,'%6d\n',numTags);   % Write the number of tags in file
fprintf(fileNumTags,'%6d\n',l1);        % Write the size of the plant in file
fprintf(fileNumTags,'%6.4f\n',gamma1);      % Write the starting angle
fprintf(fileNumTags,'%6d\n',robStart(1));      % Write the starting pos
fprintf(fileNumTags,'%6d\n',robStart(2));      % Write the starting pos
fclose(fileNumTags);


%% Drawing environment
figure(1)
x1 = [0 l1 l1 0 0];
y1 = [0 0 l2 l2 0];
plot(x1, y1,'LineWidth',2)
xlim([-5 (l1+5)]);
ylim([-5 (l2+5)]);
hold on

% Position of the tags
ID = 1:numTags;
[Tagx,Tagy] = meshgrid(d2:d1:l1-d2,d2:d1:l2-d2);
plot(Tagx,Tagy,'r*')
% Circles
radiipl = ones(numTagsX,1)*r1;
for k=1:numTagsX
    tempx = Tagx(1:end,k);
    tempy = Tagy(1:end,k);
    temppos = horzcat(tempx,tempy);
    viscircles(temppos,radiipl,'Color','k','LineStyle',':','LineWidth',0.25);
end
robX = robStart(1);
robY = robStart(2);
plot(robX,robY,'bO','LineWidth',3);
plot(robX,robY,'r:');
viscircles([robX,robY],r3,'Color','k','LineWidth',0.25);
plot(antPos(1),antPos(2),'bs');
xlabel('Length platform in cm')
ylabel('Width platform in cm')
title({'Position and reading range of tags';'Start-, endpoint and path of the robot'});
```

```
hold off
pause(1)

%% Animation and loggin
xUpdateAnt = antPos(1);
yUpdateAnt = antPos(2);
deltaR = deg2rad(angle1);       % A new measurement after every XX°
% Txt file name
name = ['Meas_StartingProc_like_reader_real_data',measuementeNumber,'.txt'];
fileID = fopen(name,'w');

% Data stored in variables
dataRSSI = zeros(8,numTags);
streamDataRSSI = zeros(1,numTags);
streamDataRSSIold = zeros(1,numTags);
timeStep = 1;   % current measurement step

% antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];
figure(2)
for l=0:360/angle1
    deltaR_temp = deltaR * l;
    xUpdateAnt = robStart(1) + d3 * cos(gamma1 + deltaR_temp);
    yUpdateAnt = robStart(2) + d3 * sin(gamma1 + deltaR_temp);
    plot(x1, y1,'LineWidth',2)
    hold on
    xlim([-5 (l1+5)]);
    ylim([-5 (l2+5)]);
    [Tagx,Tagy] = meshgrid(d2:d1:l1-d2,d2:d1:l2-d2);
    plot(Tagx,Tagy,'r*')
    plot(robX,robY,'bO','LineWidth',1);
    plot(robX,robY,'r:');
    plot(xUpdateAnt,yUpdateAnt,'bs');
    xlim([-5 (l1+5)]);
    ylim([-5 (l2+5)]);
    viscircles([robX,robY],r3,'Color','b','LineWidth',0.5);
        for k=1:numTagsX
            tempx = Tagx(1:end,k);
            tempy = Tagy(1:end,k);
            temppos = horzcat(tempx,tempy);
            viscircles(temppos,radiipl,'Color','k','LineStyle',':','LineWidth',0.25);
        end
    hold off

    % Creating measurements
    antPosnew=[xUpdateAnt,yUpdateAnt];
    for m = 1:numTags                    % m = current number of tag
        m_str = num2str(m);
        tempTag=[Tagx(m),Tagy(m)];
        tempD = pdist([antPosnew; tempTag] ,'euclidean');

        % Display if tag is in range or not
        if tempD > r1
                streamDataRSSI(m) = 0;
                if (streamDataRSSI(m) ~= streamDataRSSIold(m)) && mode == 2
                    if mode_hex == 1
                         fprintf(fileID,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                             text3,k(end),text0);
                    elseif mode_hex == 0
                         fprintf(fileID,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
                            text0);
                    end
                     fprintf(' %d %d %1d\n',l*angle1,m,'0');
                end
        elseif tempD <= r1
                 % disp(['Label ',m_str,' in range!!!!!!!!!!!!']);
                 % Relation distance <-> RSSI
                 k_temp = find(r2>=tempD);
                 k = r4(k_temp);
                 dataRSSI(timeStep,m) = k(end);
                 streamDataRSSI(m) = k(end);
                 if (streamDataRSSI(m) ~= streamDataRSSIold(m)) && mode == 2
                     if mode_hex == 1
                         fprintf(fileID,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                            text3,k(end),text0);
                     elseif mode_hex == 0
                         fprintf(fileID,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
                            text0);
                     end
                     fprintf(' %d %d %8d,\n',l*angle1,m,k(end));
                 elseif mode == 1
                     if mode_hex == 1
                         fprintf(fileID,'%d %s%s%s%d%s\n',l*angle1,text2,dec2hex(m, 16),
                            text3,k(end),text0);
                     elseif mode_hex == 0
                         fprintf(fileID,'%d %s%d%s%d%s\n',l*angle1,text2,m,text3,k(end),
                            text0);
                     end
                     fprintf(' %d %d %1d,\n',l*angle1,m,k(end));
                end
        end
    end
    streamDataRSSIold = streamDataRSSI;
    pause(cycleT/1000)
    timeStep = timeStep + 1;
end
savefig('Figure2.fig');
fclose(fileID);

%% Results
% figure(3)   % plot for the max value of every tag
% dataRSSInoT = reshape(max(dataRSSI),[numTagsX,numTagsY]);
% plot3(Tagx,Tagy,dataRSSInoT,'*');
% xlabel('Length platform in cm')
% ylabel('Width platform in cm')
% title('Max RSSI signal of every tag')

figure(4)   % plot of the RSSI signal which are non zero vs. time
dataRSSIsum = sum(dataRSSI);
IDclear = find(dataRSSIsum ~= 0);
IDstr = string(IDclear);
dataRSSIclear = dataRSSI;
dataRSSIclear( :, all( ~any( dataRSSI ), 1 ) ) = []; % and columns
plot(dataRSSIclear);
xlabel('Measurement points')
ylabel('RSSI')
ylim([0 360/angle1])
legend(IDstr,'FontSize',6);
title('RSSI Signal of every non zero tag')
```