

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(4511328)
Stephan Vette(4511328)

Supervisors:

Afaq Ahmad
Marina Rantanen-Mod  er

Abstract

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

```
\section*{}
```

Contents

1	Introduction	5
2	Pipeless Plant	6
2.1	Existing setup	6
2.2	Problems with the Existing Setup	6
3	Selection Process	7
3.1	Triangulation	7
3.2	Pattern Recognition	7
3.3	RFID	7
3.4	Map-Based Localization	7
4	Theoretical Background	8
4.1	Radio Frequency Identification (Abdul)	8
4.2	Trilateration	8
4.3	9
5	Hardware	10
5.1	RFID reader and antenna	10
5.2	RFID tag ?	11
5.3	Wifi modul (Abdul ?)	11
5.4	HW setup?!? (Abdul ?)	11
6	Simulation	12
6.1	Emulator	12
6.2	RSSI Measurements with real hardware	13
6.3	Simulation with emulated data	14
6.4	Results	14
7	Implementation	15
7.1	Communication (Abdul and/or Stefan)	15
7.2	Initialization procedure (Stephan and Stefan)	15
7.2.1	Recording and filtering data (Stefan)	15
7.2.2	Analysing data (Stefan)	15
7.2.3	Selection of correct distance related to RSSI	15
7.2.4	Estimation of initial position and orientation	16
7.3	Test setup	18
7.4	Results	19
8	Conclusion	21
9	Future Work	22
10	References	23
11	Appendixes	24
11.1	Appendix A: Emulator RFID data (Matlab)	24

List of Figures

1	Overview Trilateration	8
2	RFID reader KTS Systeme RFIDM1356-001	10
3	RFID Antenna KTS Systeme PCBA1356_8	11
4	Relation between RSSI and distance antenna to tag	13
5	Flow Chart: Selection of correct distance and most proper IDs	15
6	Computing the center of the robot	17
7	Orientation of robot in absolute angle	17
8	testing setup for initialization procedure	18
9	Estimated position in x-direction	19
10	Estimated position in y-direction	20
11	Estimated orientation	20

List of Tables

1	Should be a caption	7
2	Relation between RSSI and distance antenna to tag (data)	13
3	Results Simulation	14
4	Possible shapes of pattern	16
5	Positions of the IDs in the test setup	19

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
- Implementation
- Pro and con

..

3.2 Pattern Recognition

- Summary
- Implementation
- Pro and con

..

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 1: Should be a caption

4 Theoretical Background

4.1 Radio Frequency Identification (Abdul)

4.2 Trilateration¹

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:

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

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.

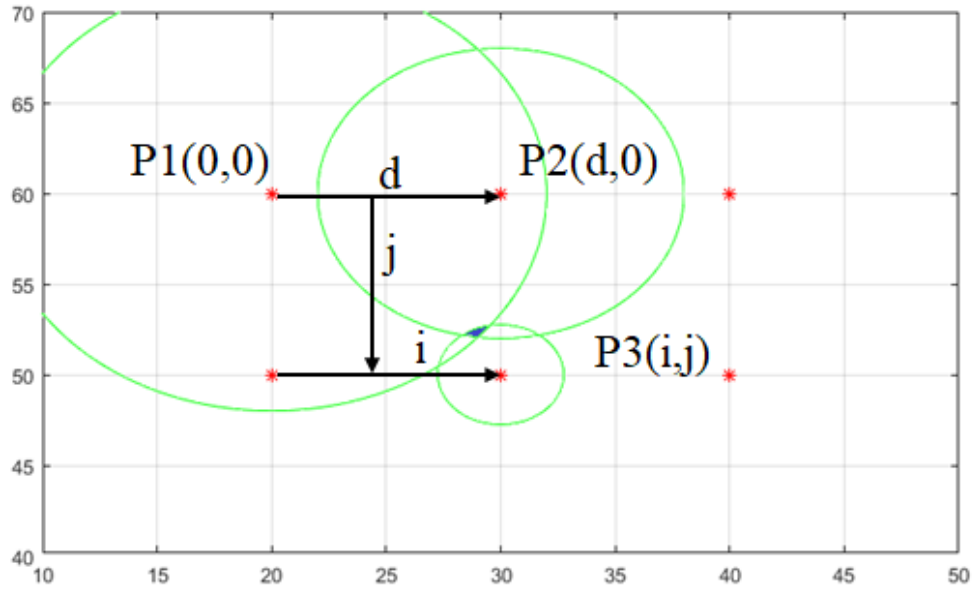


Figure 1: Overview Trilateration

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

¹Stephan

way[1]:

$$d = |P_2 - P_1| \quad (2)$$

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

$$a_x = P_3 - P_1 \quad (4)$$

$$i = e_x \cdot a_x \quad (5)$$

$$a_y = (P_3 - P_1) - i * e_x \quad (6)$$

$$e_y = \frac{a_y}{|a_y|} \quad (7)$$

$$j = e_y \cdot a_x \quad (8)$$

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:

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

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

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

$$P = P_1 + e_x * x_t + e_y * y_t \quad (11)$$

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 Hardware²

5.1 RFID reader and antenna³

The RFID reader from KTS Systeme (see fig.2) 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 50 Ω connection or via Pin Header U.FL. jack, which was used in this project.

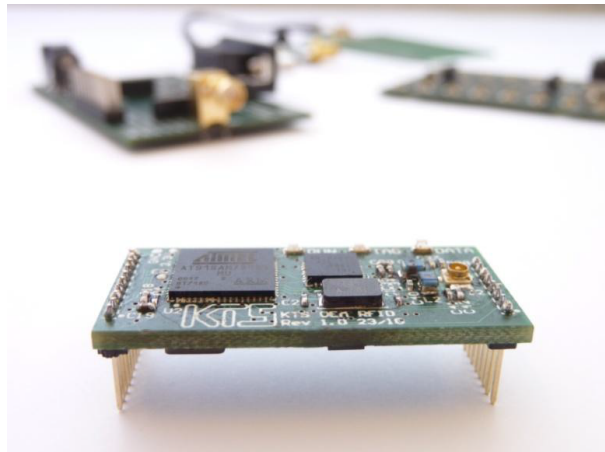


Figure 2: RFID reader KTS Systeme RFIDM1356-001

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

²Stephan and Abdul

³Stephan

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. 3) 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 3: 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 Simulation⁴

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 explained 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.

⁴Stephan

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. 2 and fig. 4 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 2: Relation between RSSI and distance antenna to tag (data)

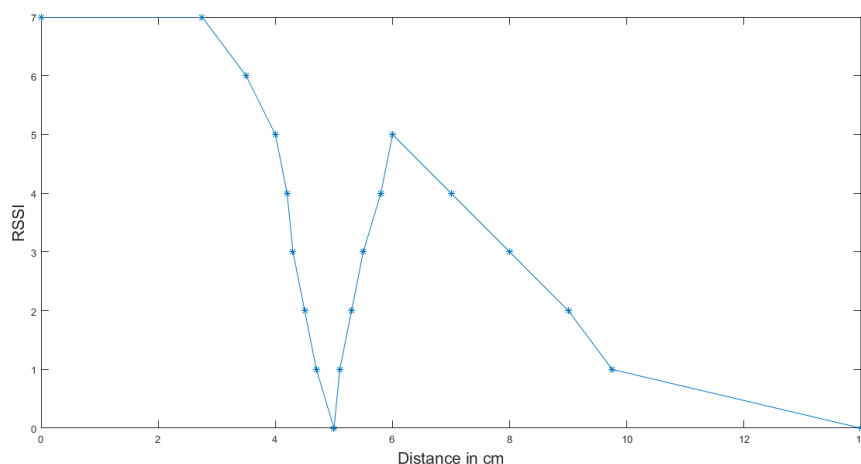


Figure 4: Relation between RSSI and distance antenna to tag

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 3: Results Simulation

As can be seen from tbl. 3, 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)

7.2.1 Recording and filtering data (Stefan)

7.2.2 Analysing data (Stefan)

7.2.3 Selection of correct distance related to RSSI footnoteStephan

In a first step the multiple occurring data points (see tbl.2) 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. 5:

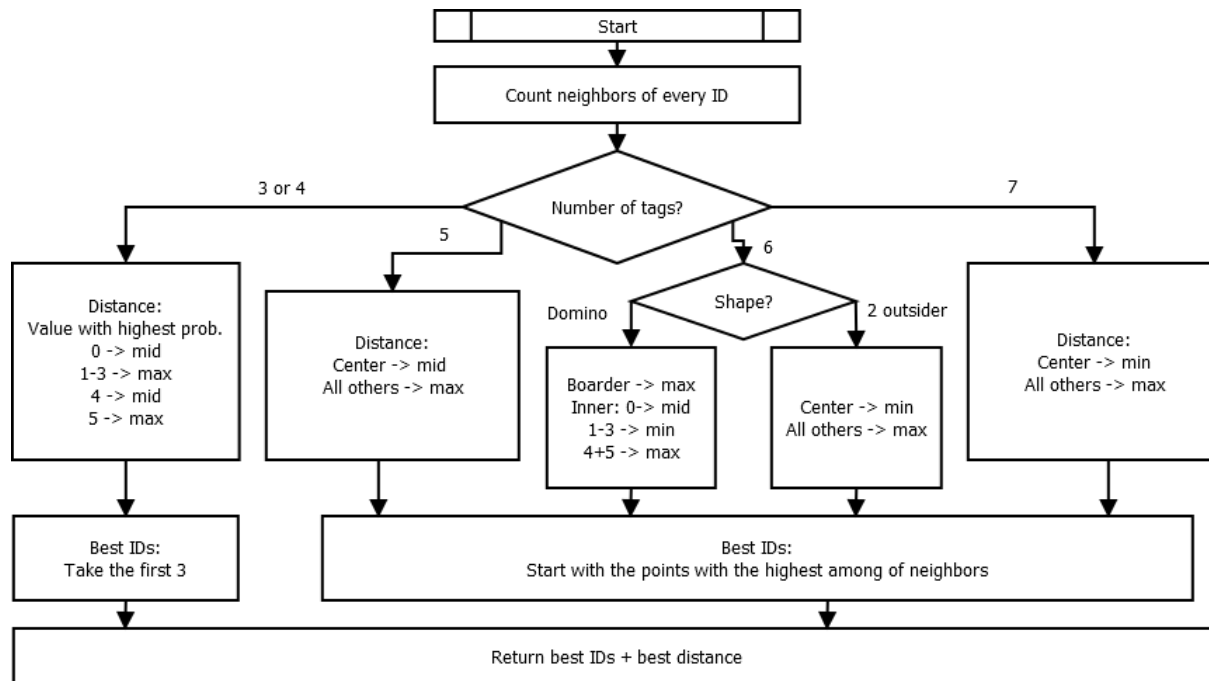


Figure 5: Flow Chart: Selection of correct distance and most proper IDs

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. 4.

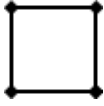
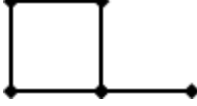

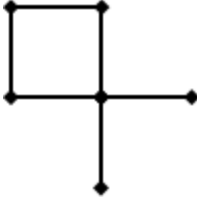
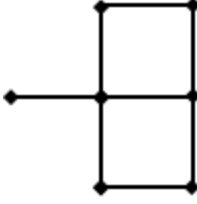
Number of detected tags	4	5	6 (Domino)	6 (2 alone)	7
Unique shapes					

Table 4: Possible shapes of pattern

Going back to the flow chart fig.5 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.

Afterwards the most suitable three IDs will be selected, in case where more than 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 sec.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

⁵Stephan

to estimate its position. For trilateration are three defined positions plus three radii necessary, which are available after the selection of the correct distance and proper IDs.

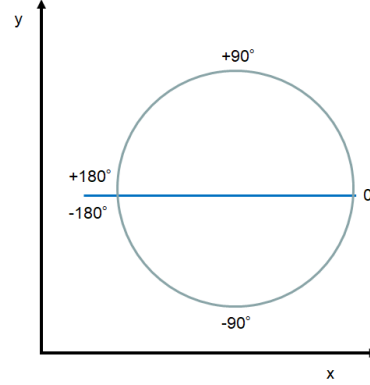
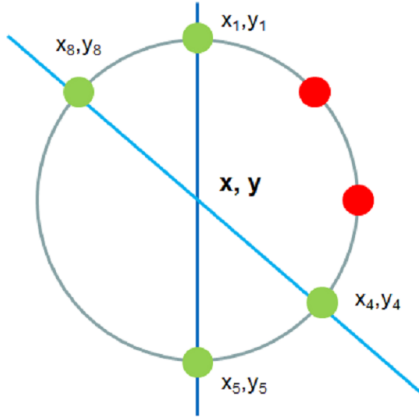


Figure 6: Computing the center of the robot Figure 7: Orientation of robot in absolute angle

As follows from the fig.6 shown above, the intersection point is found by computing two linear functions which go through 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:

$$x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \quad (12)$$

$$y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \quad (13)$$

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 $-180^\circ < \Theta \leq 180^\circ$ as shown in fig.7.

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 setup⁶

In order to verify the validity of the initialization procedure, we carried out experiments with the real HW. The beginning of these experiments was the reconstruction of one of the AGVs with the HW setup, like it was explained in sec.5.4. After we added all components to the AGV we realized the power supply for the wifi modul and the reader via a powerbank and the USB connection of then wifi modul. The plan was to replace this in the future with a direct connection to the battery of the AGV. Fig.8 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.

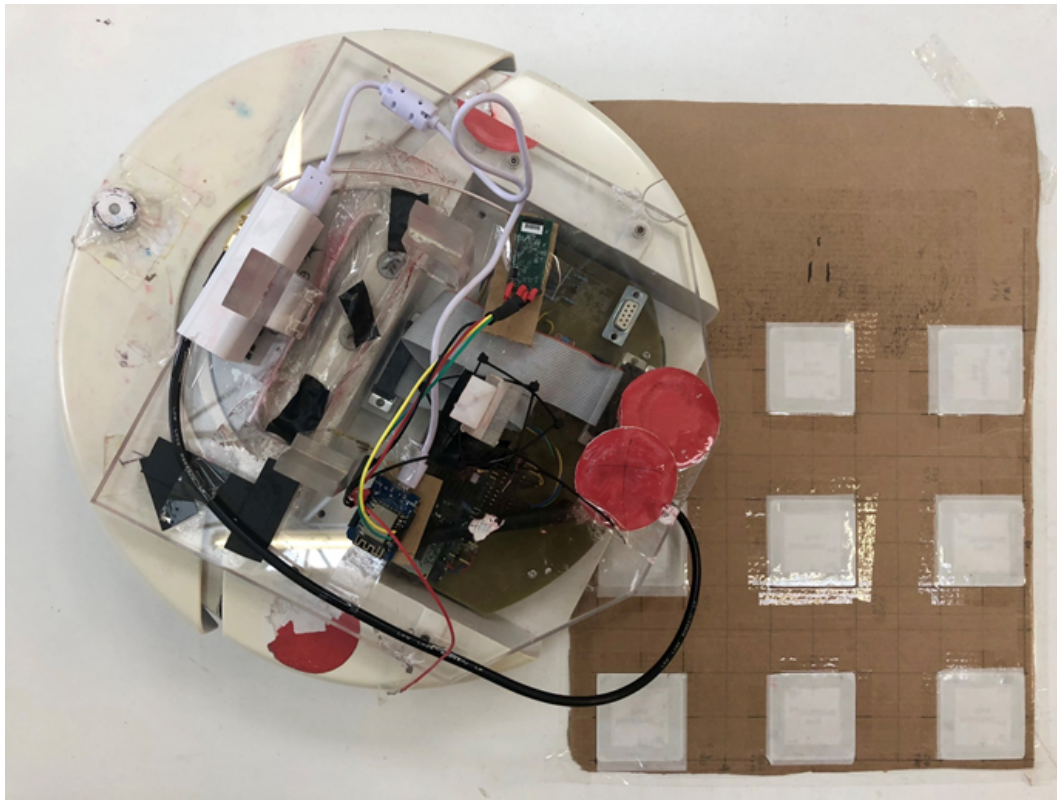


Figure 8: testing setup for initialization procedure

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.5.

The reason for the small setup was the fact that until the end of the project only 10 tags were

⁶Stephan

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 5: Positions of the IDs in the test setup

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

7.4 Results⁷

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 results of the position estimation after the procedure were read directly from the console. The initial position was 200 mm in x- and y-direction and a varying orientation (0°, 90°, 180° and -90°). Fig.9 and fig.10 illustrate the actual measurement results and the desired position in x- and y-direction.

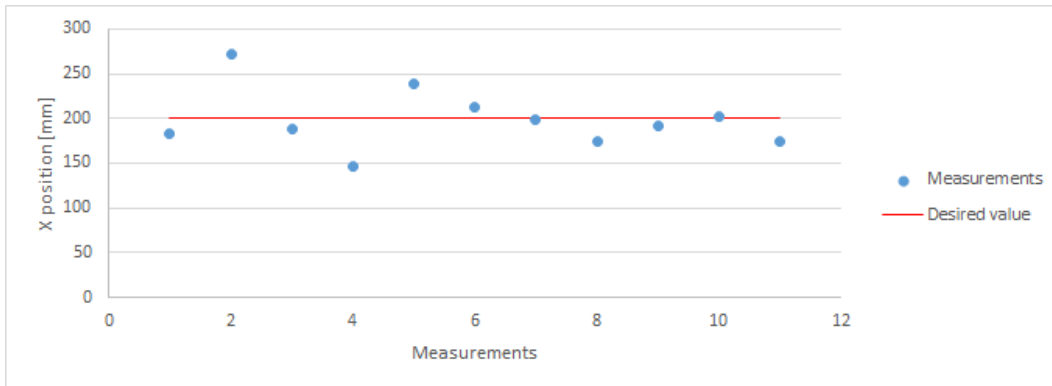


Figure 9: Estimated position in x-direction

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.

⁷Stephan

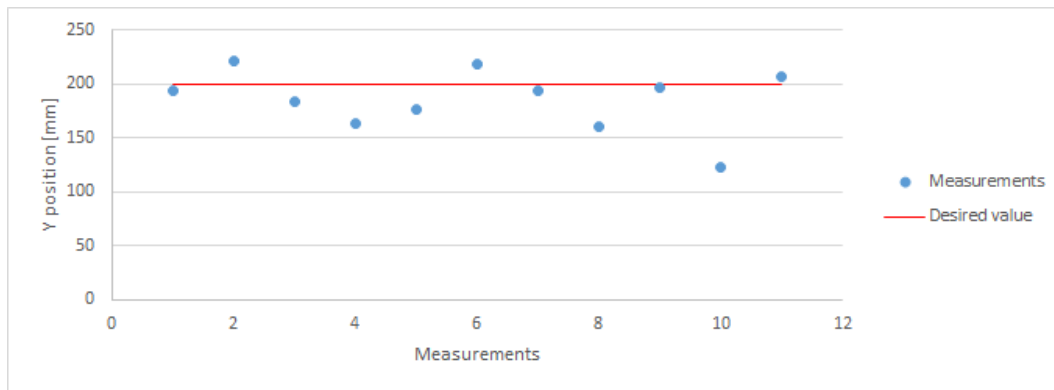


Figure 10: Estimated position in y-direction

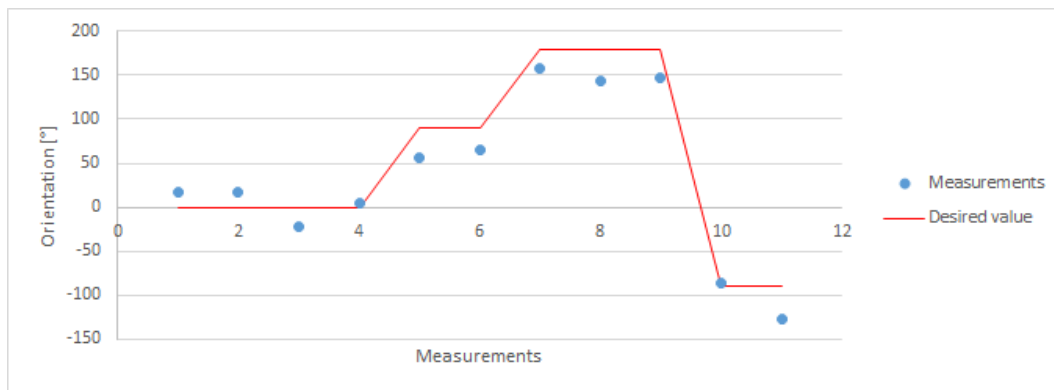


Figure 11: Estimated orientation

For the estimation of the average of the absolute error was 23° with a minimum and a maximum value of 3.9° and 37.5° during the measurements. 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.9,10 and 11).

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.

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)

```

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 gammal = 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 measuementNumber = 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

```



```

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(gammal), sin(gammal)];
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',measumeenteNumber,'.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',gammal); % 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'});

```

```

113 hold off
114 pause(1)
115
116 %% Animation and login
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',measumeenteNumber, '.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(gamma), sin(gamma)];
131 figure(2)
132 for l=0:360/angle1
133     deltaR_temp = deltaR * l;
134     xUpdateAnt = robStart(1) + d3 * cos(gamma + deltaR_temp);
135     yUpdateAnt = robStart(2) + d3 * sin(gamma + 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,radii1, '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),
169                     text3, k(end), text0);
170             elseif mode_hex == 0
171                 fprintf(fileID, '%d %s%d%s%d%s\n', l*angle1, text2, m, text3, k(end),

```

```

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

```