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(4511328)
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

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 Trilateration[1]

Trilateration is a method to compute the intersecting 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 is to use the description of sphere[1].

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

where ($P_n = (x_n, y_n, z_n)$) is the center of the sphere. To use equation (1) for the 2D indoor localization on a floor, a few assumption can be made. 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).

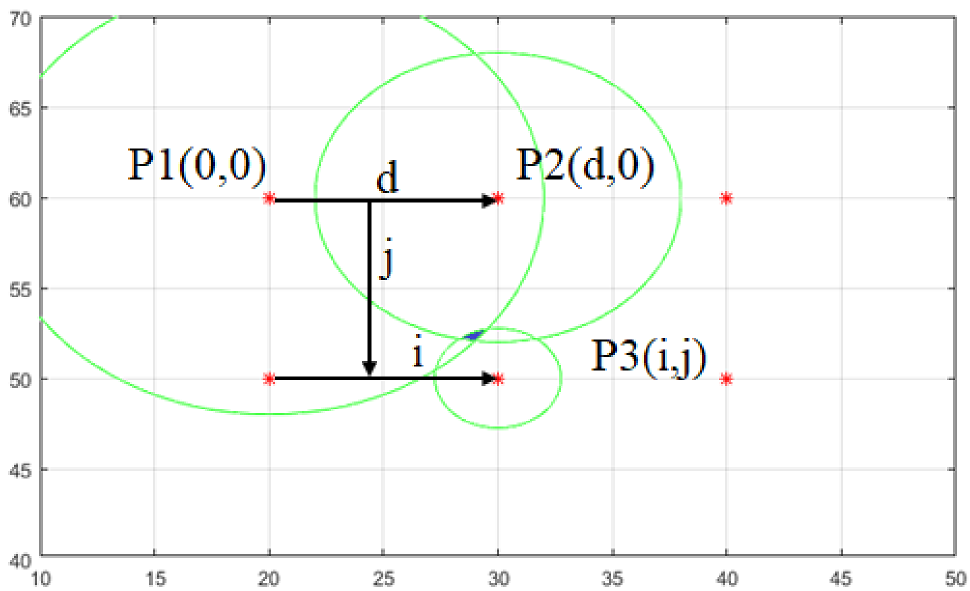


Figure 1: Overview Trilateration

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

[1]Stephan

way[1]:

$$d = |P_2 - P_1| \tag{2}$$
$$e_x = \frac{1}{d}(P_2 - P_1) \tag{3}$$
$$a_x = P_3 - P_1 \tag{4}$$
$$i = e_x \cdot a_x \tag{5}$$
$$a_y = (P_3 - P_1) - i * e_x \tag{6}$$
$$e_y = \frac{a_y}{|a_y|} \tag{7}$$
$$j = e_y \cdot a_x \tag{8}$$

After knowing the these values, the relative distance in x- and y-direction 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:

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

4.3 ...

5 Hardware[2]

5.1 RFID reader and antenna[3]

The RFID reader (see fig.2) is a HF Modul (frequency around 13.56 MHz) ...

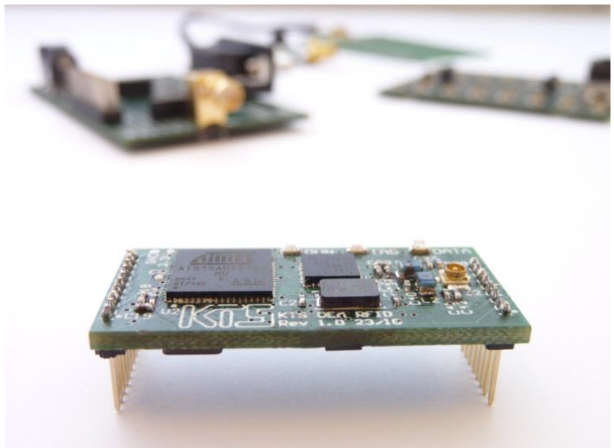


Figure 2: RFID reader KTS Systeme RFIDM1356-001

It is

5.2 RFID tag (Stephan ?)

5.3 Wifi modul (Abdul ?)

[2] Stephan and Abdul
[3] Stephan

6 Simulation[4]

These simulation was carried out to answer important design questions before the real implementation phase. After the decision for some suitable hardware, the 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 got these parameter (chapter 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 test leads 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 need 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 found tags in the created environment 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 of the antenna to the neighbouring tags. If a tags was closer than the maximal reading distance the emulator wrote the detected ID of the tags together with its RSSI into the text file.
The RSSI is 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 reference paper?!?! and mentioned a consistent increasing of the RSSI while the distance bewetten the tags and the antenna gets smaller.
During own measurements has been found out that this relation was inconsistent with this setup of components. 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. Fig. ?? presents the results of the measurements. It can be seen that there exists a blind spot at a distance of 5 cm where the RSSI drops to 0.

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)

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 very 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 first antenna 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. 3:

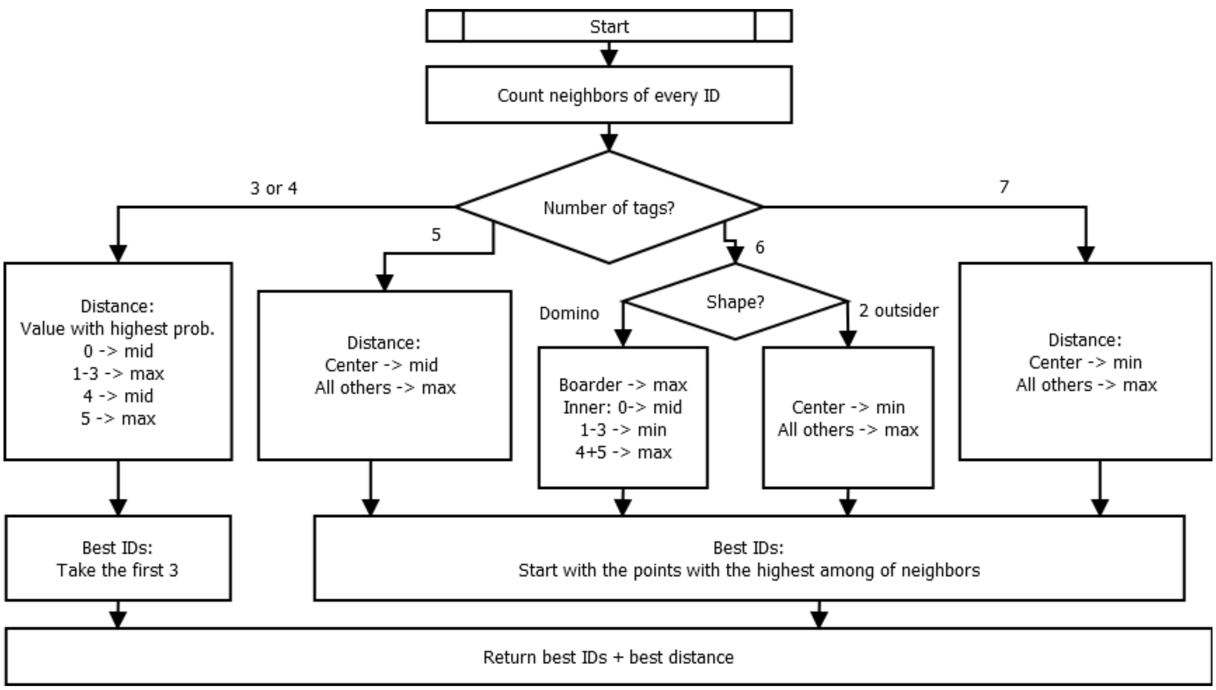


Figure 3: Flow Chart: Selection of correct distance and most proper IDs

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 several numbers (4-7) of detected tags possible. The different shapes can be found in the tbl. 4.

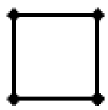
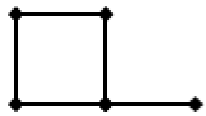
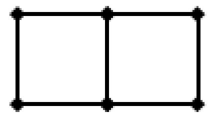
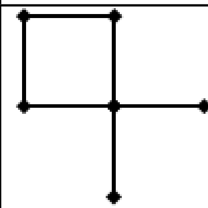
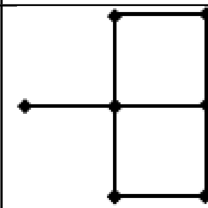
Number of detected tags	4	5	6 (Domino)	6 (2 alone)	7
Unique shapes					

Table 4: Possible shapes of pattern

Going back to the flow chart fig.3 the first step is to count the number of neighbours each tag has. With this information can the position of the tag in the pattern 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 clear, the selecting 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 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 tags, 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 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

[5] Stephan

to estimate its position. For trilatertion are three defined positions plus three radii necessary. Resulting from the last section, all these values are available.

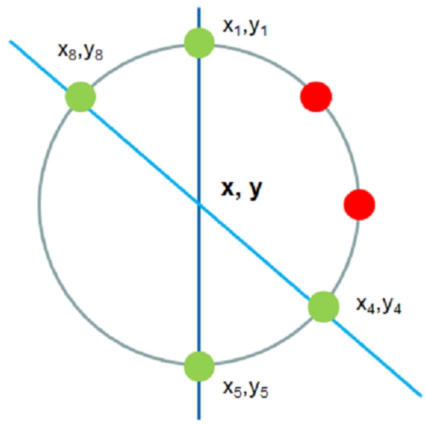


Figure 4: Computing the center of the robot

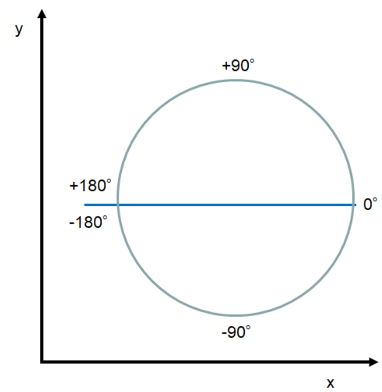


Figure 5: Orientation of robot in absolute angle

As follows from the fig.4 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:

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

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

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. The orientation of the robot between -180° and +180° (see also fig.5).

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]

7.4 Results[7]

A couple of tests on the test setup were performed to compare the good results created with the simulated data with real measurements. The initial position was 200 mm in x- and y-direction and a varying orientation (0°, 90°, 180° and -90°). Fig.6 and fig.7 illustrate the actual measurement results and the desired position in x- and y-direction.

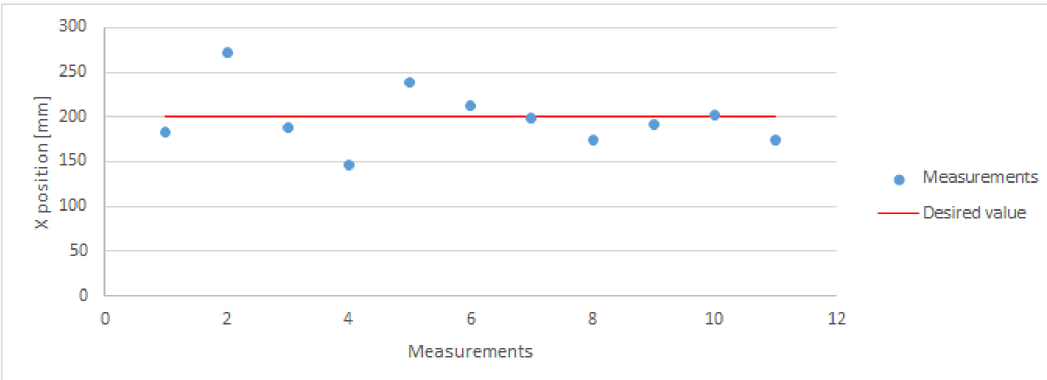


Figure 6: Estimated position in x-direction

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.

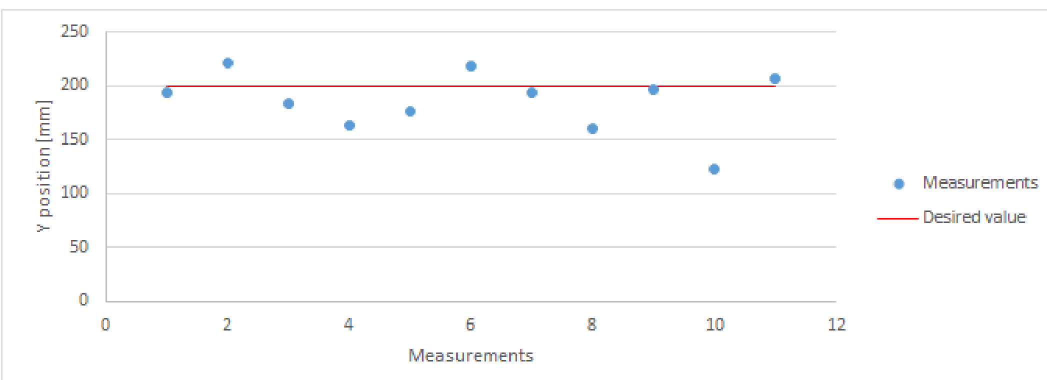


Figure 7: Estimated position in y-direction

The average of the absolute error of the estimation of the position in y-direction is with 23.3

[6]Stephan
[7]Stephan

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.

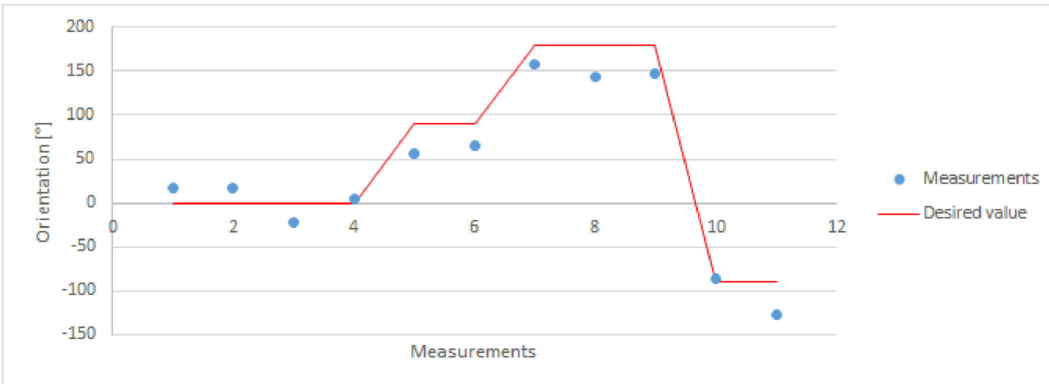


Figure 8: Estimated orientation

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.6,7 and 8).

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.

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]

```

```matlab
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')
```