

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 (Stephan ?)	10
5.2	RFID tag (Stephan ?)	10
5.3	Wifi modul (Abdul ?)	10
6	Simulation	11
6.1	Emulator	11
6.2	RSSI Measurements with real hardware	12
6.3	Simulation with emulated data	12
6.4	Results	12
7	Implementation	14
7.1	Communication (Abdul and/or Stefan)	14
7.2	Initialization procedure (Stephan and Stefan)	14
7.2.1	Recording and filtering data (Stefan)	14
7.2.2	Analysing data (Stefan)	14
7.2.3	Selection of correct distance related to RSSI	14
7.2.4	Estimation of initial position and orientation	15
7.3	Test setup	17
7.4	Results	17
7.5	Improvements	17
8	Conclusion	18
9	Future Work	19
10	References	20
11	Appendixes	21
11.1	Appendix A: Emulator RFID data (Matlab)	21

List of Figures

1	Overview Trilateration	8
2	Flow Chart: Selection of correct distance and most proper IDs	14
3	Computing the center of the robot	16
4	Orientation of robot in absolute angle	16

List of Tables

1	Should be a caption	7
2	Relation between RSSI and distance antenna to tag (data)	12
3	Results Simulation	12
4	Possible shapes of pattern	15

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 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.²

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

where $(P_n = (x_n, y_n, z_n))$ is the center of the sphere. 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).

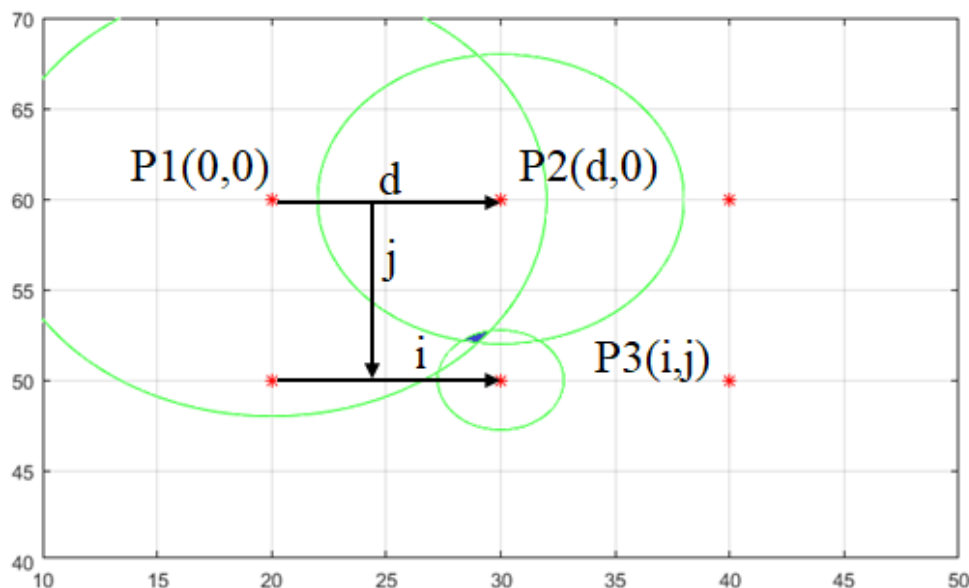


Figure 1: Overview Trilateration

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

¹Stephan

²Indoor Robot Positioning using an Enhanced Trilateration Algorithm

way: ³

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

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

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

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

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

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

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

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:

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

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

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

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

4.3 ...

³Indoor Robot Positioning using an Enhanced Trilateration Algorithm

5 Hardware⁴

5.1 RFID reader and antenna (Stephan ?)

5.2 RFID tag (Stephan ?)

5.3 Wifi modul (Abdul ?)

⁴Stephan and Abdul

6 Simulation⁵

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.

⁵Stephan

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. 2:

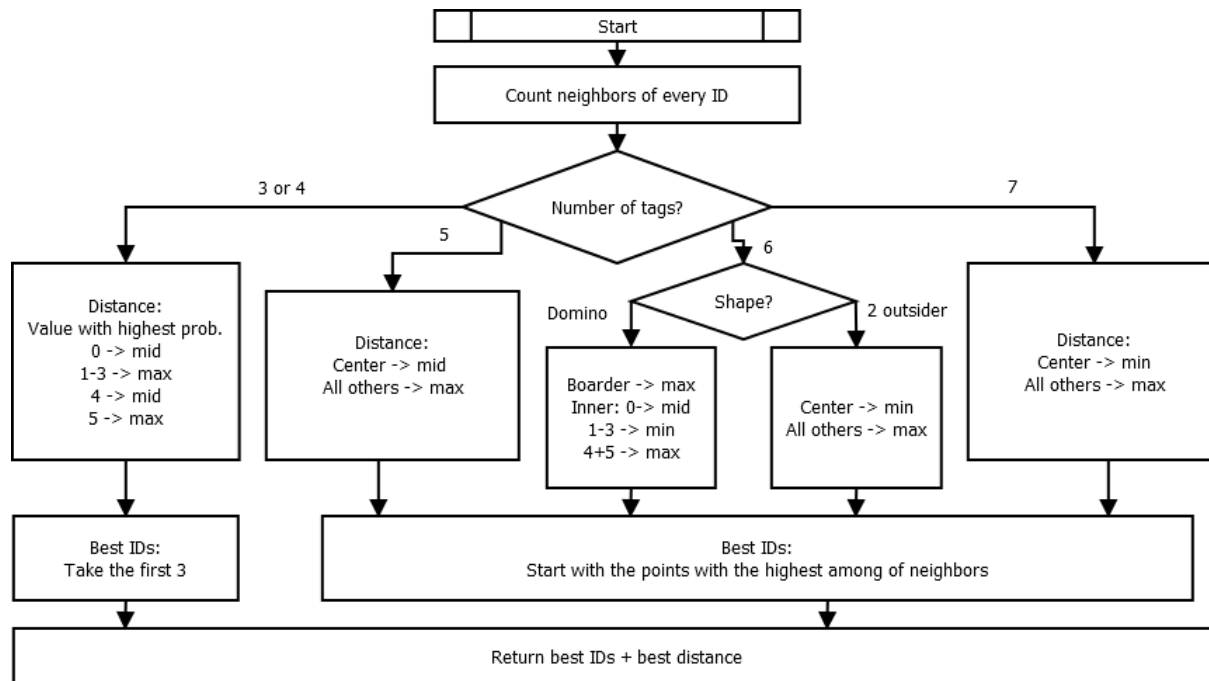


Figure 2: Flow Chart: Selection of correct distance and most proper IDs

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.

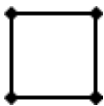
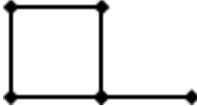
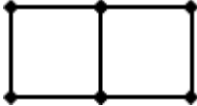
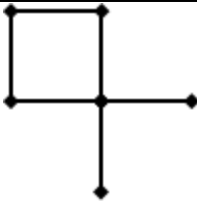
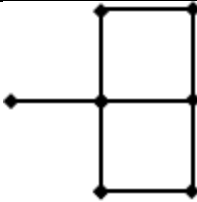
Number of detected tags	4	5	6 (Domino)	6 (2 alone)	7
Unique shapes					

Table 4: Possible shapes of pattern

Going back to the flow chart fig.2 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 number 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 ⁶

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

⁶Stephan

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

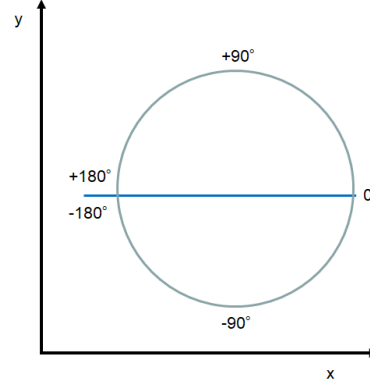
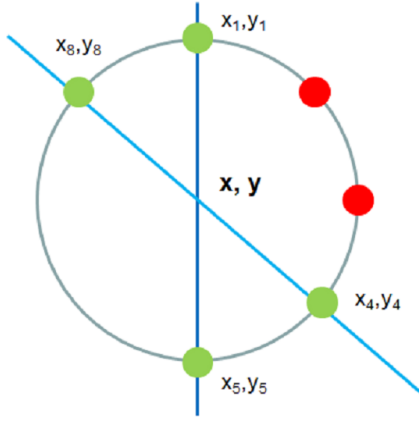


Figure 3: Computing the center of the robot Figure 4: Orientation of robot in absolute angle

As follows from the fig.3 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 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. Therefore is the orientation of the robot between -180° and $+180^\circ$ (see also fig.4).

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 setup⁷

7.4 Results⁸

7.5 Improvements

⁷Stephan

⁸Stephan

8 Conclusion

conclude..

9 Future Work

...

10 References

..

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 = 11; % 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,
19       4.5, 4.3, 4.2, 4.0, 3.5, 2.75, 0]; % distances at certain RSSI
20 r4 = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 7]; %
21     array with the different RSSI values
22
23
24 r3 = 33/2; % radius of the robot
25 d3 = 10; % distance between origin robot and origin antenna [cm]
26
27 angle1 = 45; % angle between the measurement points in the init
28     procedure
29
30 gammal = deg2rad(22.5); % Start orientation of robot [rad]
31 robStart = [22.5, 51.5]; % Start position of robot in x, y [cm]
32
33 robSpeed = 0.1; % Speed robot [m/s]
34 cycleT = 100; % Cycletime in [ms]
35
36

```

```
32 mode = 1;    % mode=1: tracking all available tags, which are nonzero
33             % mode=2: tracking only changes in the RSSI signals
34 mode_hex = 0;    % activate or deactivate hex ID
35
36 % For the name of the txt file
37 measuementeNumber = num2str(11); % Number of measurement
38 % Two possibilities for the content of the txt file
39 % 1. Without filtering. Exactly like the reader creates data
40 % text0 = '<\r>';
41 % text1 = 'OK';
42 % text2 = 'SCAN:+UID=';
43 % text3 = '+RSSI=';
44
45 % 2. Filtered data. Without unusable information.
46 text0 = ' ';
47 text1 = ' ';
48 text2 = ' ';
49 text3 = ' ';
50 %% Error check
51 if mod(11/d1,1)~=0
52     error('Length of platform not dividable by distance between tags'
53         );
54 elseif mod(12/d1,1)~=0
55     error('Length of platform not dividable by distance between tags'
56         );
57 end
58
59 %% Computing position of antenna
60 numTagsX = (11-2*d2)/d1 +1;
61 numTagsY = (12-2*d2)/d1 +1;
62 numTags = numTagsX * numTagsY;
63 antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];
64
65 %% Display the setup, write important information into a seperate txt
66 file
67 d1_str = num2str(d1);
68 l1_str = num2str(11);
69 l2_str = num2str(12);
70 numTags_str = num2str(numTags);
71
72 msg0 = [ 'Your plane is ',l1_str, 'cm x ',l2_str, 'cm. '];
```

```

70 msg1 = [ 'You chose a distance of ',d1_str, 'cm and need ',
           numTags_str, ' Tags! '];
71 disp(msg0);
72 disp(msg1);
73 nameTxt = [ 'NumTags',measuementeNumber, '.txt' ];
74 fileNumTags = fopen(nameTxt, 'w');
75 fprintf(fileNumTags, '%6d\n', numTags);    % Write the number of tags in
           file
76 fprintf(fileNumTags, '%6d\n', l1);          % Write the size of the plant
           in file
77 fprintf(fileNumTags, '%6.4f\n', gamma1);    % Write the starting
           angle
78 fprintf(fileNumTags, '%6d\n', robStart(1)); % Write the starting
           pos
79 fprintf(fileNumTags, '%6d\n', robStart(2)); % Write the starting
           pos
80 fclose(fileNumTags);
81
82
83 %% Drawing environment
84 figure(1)
85 x1 = [0 11 11 0 0];
86 y1 = [0 0 12 12 0];
87 plot(x1, y1, 'LineWidth', 2)
88 xlim([-5 (11+5)]);
89 ylim([-5 (12+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',measumentNumber,'
        .txt'];
122 fileID = fopen(name,'w');
123
124 % Data stored in variables
125 dataRSSI = zeros(8,numTags);
126 streamDataRSSI = zeros(1,numTags);
127 streamDataRSSIold = zeros(1,numTags);
128 timeStep = 1; % current measurement step
129
130 % antPos = robStart + d3 * [cos(gamma1), sin(gamma1)];
131 figure(2)
132 for l=0:360/angle1
133     deltaR_temp = deltaR * l;
134     xUpdateAnt = robStart(1) + d3 * cos(gamma1 + deltaR_temp);
135     yUpdateAnt = robStart(2) + d3 * sin(gamma1 + deltaR_temp);
136     plot(x1, y1,'LineWidth',2)
137     hold on
138     xlim([-5 (l1+5)]);
139     ylim([-5 (l2+5)]);
140     [Tagx,Tagy] = meshgrid(d2:d1:l1-d2,d2:d1:l2-d2);
141     plot(Tagx,Tagy,'r*')
142     plot(robX,robY,'bO','LineWidth',1);
143     plot(robX,robY,'r:');
144     plot(xUpdateAnt,yUpdateAnt,'bs');

```



```

145     xlim([-5 (l1+5)]);
146     ylim([-5 (l2+5)]);
147     viscircles([robX,robY],r3,'Color','b','LineWidth',0.5);
148     for k=1:numTagsX
149         tempX = Tagx(1:end,k);
150         tempY = Tagy(1:end,k);
151         tempPos = horzcat(tempX,tempY);
152         viscircles(tempPos,radiopl,'Color','k','LineStyle',':',',
            'LineWidth',0.25);
153     end
154     hold off
155
156     % Creating measurements
157     antPosnew=[xUpdateAnt,yUpdateAnt];
158     for m = 1:numTags % m = current number of tag
159         m_str = num2str(m);
160         tempTag=[Tagx(m),Tagy(m)];
161         tempD = pdist([antPosnew; tempTag], 'euclidean');
162
163         % Display if tag is in range or not
164         if tempD > r1
165             streamDataRSSI(m) = 0;
166             if (streamDataRSSI(m) ~= streamDataRSSIold(m)) && mode
                == 2
167                 if mode_hex == 1
168                     fprintf(fileID, '%d %s%s%s%d%s\n', l*angle1,
                        text2, dec2hex(m, 16), text3, k(end), text0);
169                 elseif mode_hex == 0
170                     fprintf(fileID, '%d %s%d%s%d%s\n', l*angle1,
                        text2, m, text3, k(end), text0);
171                 end
172                 fprintf(' %d %d %ld\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,
184                 text2, dec2hex(m, 16), text3, k(end), text0);
185         elseif mode_hex == 0
186             fprintf(fileID, '%d %s%d%s%d%s\n', l*angle1,
187                 text2, m, text3, k(end), text0);
188         end
189         fprintf(' %d %d %8d,\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,
193                 text2, dec2hex(m, 16), text3, k(end), text0);
194         elseif mode_hex == 0
195             fprintf(fileID, '%d %s%d%s%d%s\n', l*angle1,
196                 text2, m, text3, k(end), text0);
197         end
198         fprintf(' %d %d %1d,\n', l*angle1, m, k(end));
199     end
200 end
201 end
202 streamDataRSSIold = streamDataRSSI;
203 pause(cycleT/1000)
204 timeStep = timeStep + 1;
205 end
206 savefig('Figure2.fig');
207 fclose(fileID);
208
209 %% Results
210 % figure(3) % plot for the max value of every tag
211 % dataRSSIInoT = reshape(max(dataRSSI), [numTagsX, numTagsY]);
212 % plot3(Tagx, Tagy, dataRSSIInoT, '*');
213 % xlabel('Length platform in cm')
214 % ylabel('Width platform in cm')
215 % title('Max RSSI signal of every tag')
216
217 figure(4) % plot of the RSSI signal which are non zero vs. time
218 dataRSSIsum = sum(dataRSSI);
219 IDclear = find(dataRSSIsum ~= 0);
220 IDstr = string(IDclear);
221 dataRSSIclear = dataRSSI;
222 dataRSSIclear(:, all(~any(dataRSSI), 1)) = []; % and columns
223 plot(dataRSSIclear);

```

```
220 xlabel('Measurement points')
221 ylabel('RSSI')
222 ylim([0 360/angle1])
223 legend(IDstr,'FontSize',6);
224 title('RSSI Signal of every non zero tag')
```