

EE2L21 EPO-4: Kitt - Final Report

June 18, 2015

Group B6

Martijn Berkers	4223438
Thomas Brasser	4139518
Richard de Jong	4238575
Jeroen van Uffelen	4232690

Abstract

In this report an approach to handle the final challenge given in the EPO-4 project is presented. This includes localization of the car with audio communication, collision detection is done using the data obtained from distance sensors, planning a route with the car and driving it. There are 3 parts of the final challenge, driving from point A to point B, driving from point A to point B with waypoint C, and driving from point A to point B with an obstacle in the route.

Introduction

This report describes the design process of an autonomous driving, wireless charging vehicle. The goal of the design is to make the car drive from point A to point B while avoiding obstacles. To achieve this several problems have to be overcome; wireless communication (Bluetooth connection), object sensing, mapping the location and **let the car navigate, devising a control system to steer and drive the car, using audio communication for localization of the car and combining these solutions into a system.**

To keep structure the report is divided into several sections. First, the specifications of the car will be determined.

The second section explores localization using audio communication, for this training sequences are discussed. The deconvolution of the measured signal and the reference signal **and** corresponding peak detection to calculate the Time-difference of arrival(TDOA) between microphones. The localization algorithm used to convert these TDOA's to a position, and the verification of this position.

The third section will discuss the route determination of the car, and the corresponding outputs to the wheels of the car.

The fourth section will discuss the system integration of all these parts and the loop where they are used in, and the GUI used for displaying the position of the car **and** obstacles.

Lastly a conclusion is constructed about the project and discussion about what could have **been** done different.

Specifications

The completed system consisting of the car and the software should be able to fulfill predefined tasks and is limited by some constraints. The EPO4 Manual[1] defines the final challenge as a system integration challenge featuring a demonstration to show the various capabilities of the system.

This demonstration consists of 4 succeeding challenges: In the first challenge the car should be able to drive from point 'A' to point 'B', the second challenge adds a waypoint 'C'. The third challenge adds an obstacle to avoid and the last challenge includes a source of interference.

Further constraints are the fully autonomous functioning of the system. Which means that neither the car or the controlling computer can be touched during the challenges. Also, at the start of the challenge, the supercaps should be charged wirelessly, and the challenge ends successfully when the car stops at its destination and the controlling computer plays a sound.

Contents

Introduction	2
Specifications	2
1 Localization using audio communication	5
1.1 Training sequence	5
1.2 Reference	6
1.3 Deconvolution	8
1.4 Peak detection	11
1.5 TDOA	13
1.6 Localization	13
1.7 Checking calculated position	15
1.8 Resulting Design	15
1.9 Testing	18
1.9.1 Field testing	19
1.10 Conclusion	19
2 Route planning	20
2.1 Introduction	20
2.2 Design	20
2.2.1 Specifications and Goals	20
2.2.2 Code	22
2.3 Testing	26
2.3.1 Trial and Error & Findings	27
2.4 Conclusion	27
3 Control Loop	28
3.1 Overview	28
3.2 VoltageMeasure	28
3.3 Drive	28
3.4 Sample	29
3.5 Mapping the car	29
4 The Final Challenge	31
4.1 Objective	31
4.2 Training sequence	31
4.3 Reference	31
4.4 What went wrong and how was it solved?	31
5 Conclusion and Discussion	35
5.1 Conclusion	35
5.2 Discussion	35
6 Planning and Teamwork	36
6.1 Planning	36
6.2 Teamwork	36

A	Appendix	39
A.1	Localization	39
A.1.1	Open_com.m	39
A.1.2	orientation.m	39
A.1.3	matched_f.m	39
A.1.4	matched_f.m	40
A.2	Planner	40
A.2.1	planner.m	40
A.2.2	turn.m	43
A.2.3	straight.m	43
A.3	Loop	43
A.3.1	control_loop.m	43
A.3.2	drive_car.m	48
A.3.3	states.m	48
A.3.4	status_update.m	48
A.4	Final Challenge	48

1 Localization using audio communication

The goal of this module is using audio beacons to accurately determine the position of the car. An audio beacon is placed on the car, the signal is received with five microphones at pre-determined positions. With this information the position of the car can be determined with the TDOA(Time difference of arrival). This time corresponds to the difference in arrival time between microphones. With microphone pairs the position can be determined. To determine when the "sound" is received a training sequence is used and transmitted using the audio beacon. This training sequence is received at all microphones at different times and **at** intensities, to filter the training sequence from the noise the channel response is used instead of the measured signal. For the estimation of the channel response a known training sequence **has to be** used. The deconvolution of the inverse of the training sequence and the measured signal **gives channel** response. There are several implementations for this because the inverse of the training sequence is not easily calculated. The implementation used is the frequency **equalization** The channel response will ideally have one peak at the point where the measured signal is the same as the training sequence. By comparing the position of the peaks of different microphones the difference of arrival in samples can be found; this is converted to time. In practice the measured signal is not clean, noise and reflections are also measured. To **chose** the right peak the bad ones need to be filtered. To improve the deconvolution properties of the measured signal, instead of the modelled training sequence the impulse response of the sequence is **measured, this** can be measured when the beacon is close to the microphone.

1.1 Training sequence

The training sequence that will be transmitted by the audio beacon is a binary sequence which uses "on-off keying". Besides a code word with a maximum of 64 bits, the training sequence is restricted by the parameters in table 1. The code word used is a random sequence of 32 bits, which is "1432AB5C" in HEX. The code is random because it has good convolution properties. The setup used has a sample frequency(F_s) of 48 kHz. That means that the maximum carrier frequency can be $\frac{1}{2} * F_s = 24$ kHz. To be on the safe side our Timer0 parameter is 2, which is 15 kHz.

To distinguish the carrier frequency from the code frequency, at least 5 waves are wanted in 1 bit. Because of this, the Timer1 parameter is 4, which corresponds to 3.0 kHz. The Timer3 parameter is 7, which is 8 Hz. This is because there has to be a silent period after the code has been send, to let reflections and the beacon die out. To clarify the parameters, in figure 1 an example of a generated beacon sequence can be found.

Timer index			0	1	2	3	4	5	6	7	8	9
Carrier Freq	(Timer0)	[kHz]	5	10	15	20	25	30	0	0	0	0
Code Freq	(Timer1)	[kHz]	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	0
Repeat Freq	(Timer3)	[Hz]	1	2	3	4	5	6	7	8	9	0

Table 1: Timers frequency configuration table

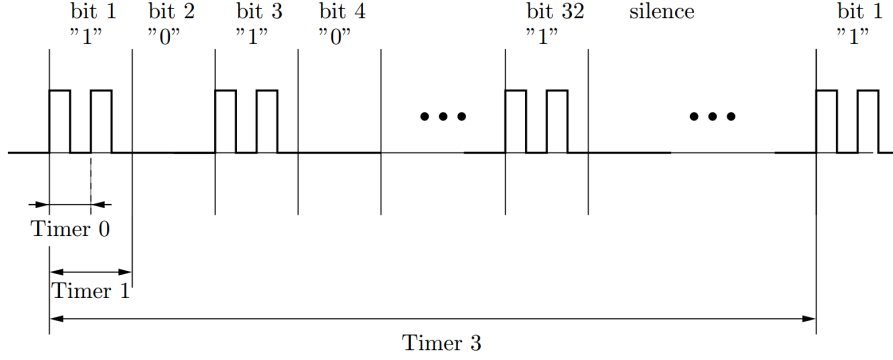


Figure 1: Example of pulses generated by the beacon

1.2 Reference

To have the best deconvolution properties an accurate reference signal has to be used. The modelled signal and the transmitted signal are not equal, because of this the deconvolution properties are not optimal. To get an accurate reference the signal that leaves the speaker needs to be measured. At close distances of around 5 cm, the measured signal will be the channel response $y[n] = h[n]$ where $y[n]$ is the measured signal in samples and $h[n]$ the channel response. Each time the training sequence is changed a new reference has to be measured. The reference used was one of the multichannel measurements, because these microphones are the same as the ones used in the final challenge. The signal was manually shortened to only contain the training sequence, the length of this is given by the settings of the training sequence, the settings used give it a length of 512 samples with a sample rate of 48kHz. Because it is not clear when the signal begins and ends some data points before the sequence and after have been included in the reference. A plot of the reference can be seen in figure 2

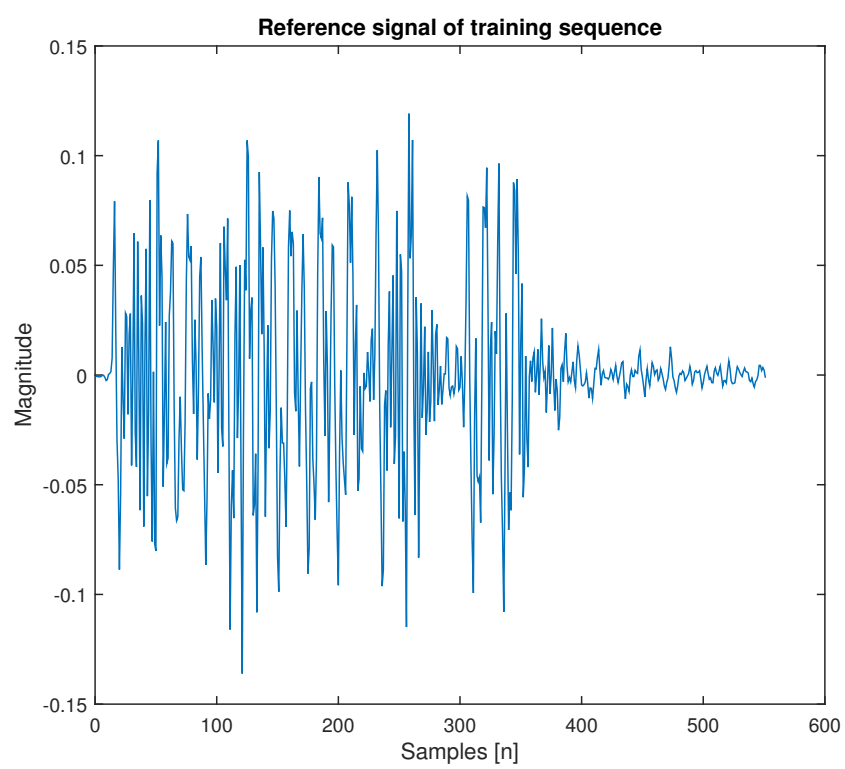


Figure 2: Refsignal

1.3 Deconvolution

The deconvolution of the received audio signal by taking the inverse of the reference signal is computationally complicated. There are three alternative methods which can be used for the deconvolution; the first is the calculation of the Toeplitz matrix, the second is using a matched filter and the third is using frequency equalization. The Toeplitz matrix has a size of the length of the measured signal times the length of the impulse response. The impulse response will be length of y minus the reference. Both of these for **our** usual measurements are around 18000. Taking the inverse of a matrix this size is not a viable option. Because there was still some doubt about which was better, the matched filter or the frequency equalization, it was decided to implement both. As the tests were done the method with the most reliable results **will** be the one to choose for the final challenge. The frequency equalization turned out to be the **best** option and is chosen for the final challenge

The deconvolution function has two inputs, the measured audio signal $y[n]$ and the reference measured training sequence $x[n]$. **f**unction will return two estimated channel responses $h[n]$, one for the matched filter and one for the frequency equalization. The matched filter will use the **filter** function in MATLAB, the filter coefficients are determined by the flipped training sequence $x[-n]$. Because this filtered data is not yet **yet** in the right scale, it is corrected with the magnitude of $x[n]$. The Matched Filter can be written as

$$\hat{h}[n] := y[n] * x[-n] = h[n] * (x[n] * x[-n]) = h[n] * r[n]$$

where $r[n]$ is the autocorrelation of the reference signal. The audio channel response will be smeared, because of the autocorrelation it is expected that there will always be a peak where the signals overlap. Because of this the channel response may be inaccurate but at least the peak will be pronounced and corresponds to the true channel. A plot of the typical impulse response estimated with a matched filter is found in 4. The code of the matched filter can be seen in appendix **??**

An alternative to the Matched Filter is to do frequency equalization. Because the frequency equalization had the best deconvolution properties this is used for the final challenge. $Y(\omega) = H(\omega)X(\omega)$ is derived from $y[n] = h[n] * x[n]$ and hence we can estimate $H(\omega) = Y(\omega)/X(\omega)$. To make this work using the FFT all the sequences need to be of equal length N, where N is the length of y. Since a pointwise division is done in frequency domain the performance will be low for frequencies where $X[k]$ is small. Therefore a threshold ϵ is used to determine which values of $H[k]$ will be set to zero before the Inverse Fast Fourier Transform is applied. The value of epsilon used is calculated in line 10 in Listing 1 after that the indices where X is smaller than ϵ are determined in lines 10-16 and the newly obtained channel estimate is calculated. The entire code can be found in appendix **??**

Listing 1: Calculating Epsilon

```
10 eps = 0.1*max(abs(X)); %Make epsilon depend on X
11 G = X; %make G the same length as X
12 ii = find(abs(X) <= eps);
13 G(ii) = 0;
14 ii = find(abs(X) > eps);
15 G(ii) = 1;
```

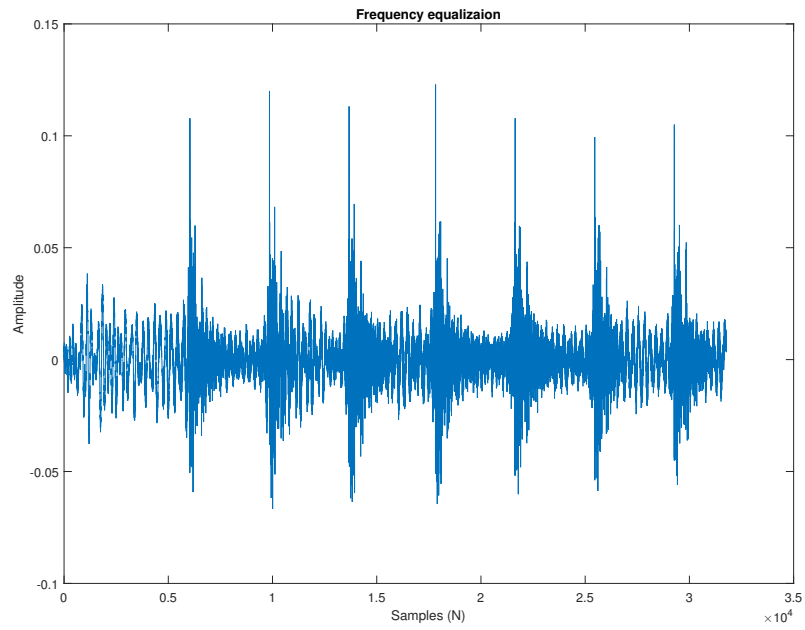



Figure 3: Typical impulse response of frequency equalization

16 $H = H.*G;$

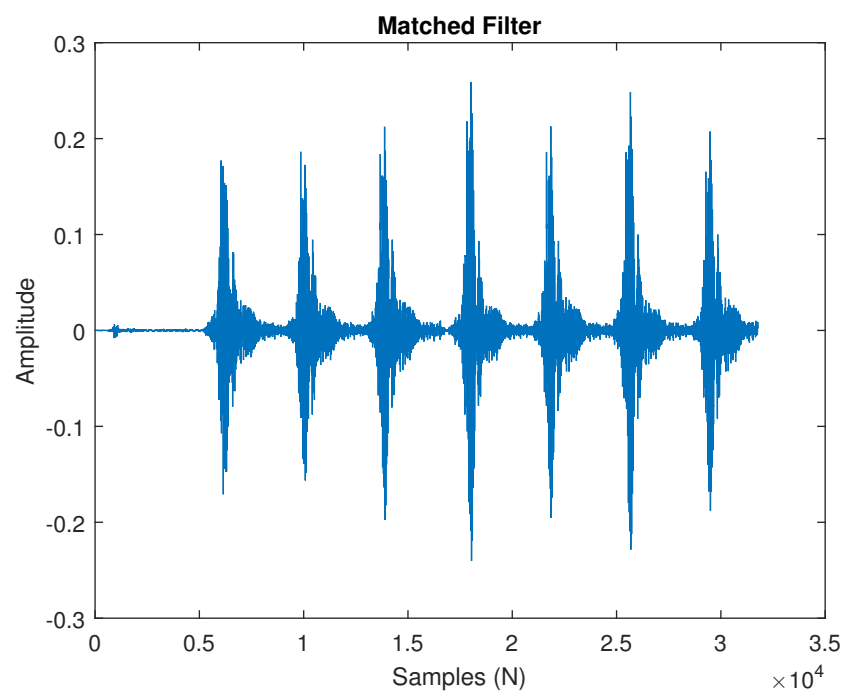


Figure 4: Typical impulse response of a matched filter

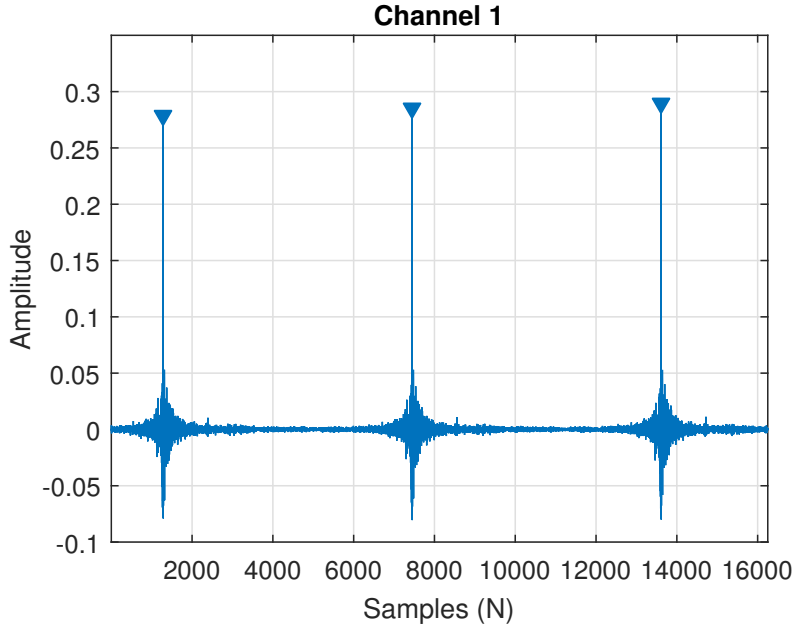


Figure 5: Peak detection channel 1

1.4 Peak detection

When the channels are estimated the position of the peaks of the estimated channels need to be determined. The peaks of the channels are determined using the `findpeaks` command in `matlab`. This function returns the height of the peaks and the location of the peaks based on the parameters given to the function `as` can be seen in Listing 2. The code used calculates the first peak of the signal after a 1200 samples silence period, which corresponds to the maximum delay that should appear in the signal, and a 6000 samples minimum peak distance range. The maximum delay found is across the diagonal of the field and therefore this delay in samples is calculated as $\sqrt{6^2 + 6^2}/340 * 48000 = 1.1979 * 10^3$ samples. Where $340m/s$ is the speed of sound and $48000Hz$ is the sample rate of the system. The minimum peak distance is based on the repetition timer of the beacon setting. An example of the calculated peaks of the first channel is shown in figure 5. The first peak found after the silence period of 1200 samples is used as reference for the other peaks. The entire code/function used to calculate the first peak is found in appendix ??

Listing 2: Peak position

```

6 Ts = 1200; %Search window
7
8 %Determine the location of the largest peak after a silence period
9 [peaks_f, locations_f] = findpeaks(h1(Ts:end), 'MinPeakDistance',
    6000);

```

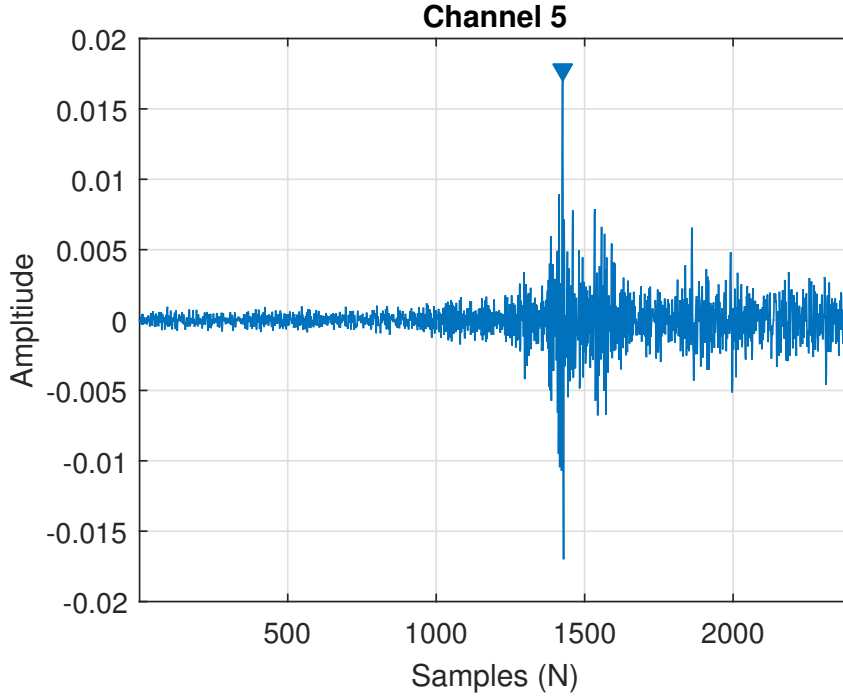


Figure 6: Peak detection channel 5

When the peak of the first channel is found this peak location is used as a reference for the other peak locations as can be seen in Listing 3 and is illustrated in figure 7. The position of peak 1 is the position of n_{max} in the figure. A search window is applied to the second channel ranging from ± 1200 samples from the position of the peak of the first channel. The location of the highest peak in this interval is returned and corresponds to the delayed or leading peak from the first channel. The resulting window is shown in figure 6 for channel 5 of the test measurement. The search window has a size of ± 1200 samples and the corresponding peak position is found. This same process is done for the remaining channels and all the peak positions are returned so the TDOA can be calculated.

Listing 3: Peak position channel 2

```

15 loc_peak_1 = loc_peak1;
16
17 [peaks_2_f, locations_2_f] = findpeaks(h2([loc_peak_1-Ts:loc_peak_1+
    Ts]), 'MinpeakDistance', 2*Ts-1);

```

1.5 TDOA

After the peak detection all the matching peak positions are known for each microphone. With these peak positions the differences between microphones can be determined. For use in the localization algorithm they will be returned in a row shape in. To convert the samples to cm it is divided by sample rate, the sample rate used is 48 kHz. With this time in seconds, the distance in meters can be recovered with the speed of sound. An estimate of 340 m s^{-1} is used for the speed of sound, this distance in meters multiplied will be the difference distance between microphones in cm. For the final challenge five microphones are present, the TDOA matrix will have the size of 5x5. The portion that calculates the TDOA is seen in listing 4. In table 1.5 the measurement used as a example in the peak detection subsection are used to display the tdoa matrix, the distance differences in cm are displayed.

Listing 4: TDOA calculation with microphone positions

```

27 %Calculate all the differences in distances
28 TDOA = zeros(5);
29 for x = 1:5
30     for y = 1:5
31         TDOA(y,x) = position_mic(x) - position_mic(y);
32     end
33 end
34 %Calculate the TDOA in centimeters
35 TDOA = TDOA ./ 48000;
36 TDOA = TDOA .* speed_of_sound*100;
37 %Make an array from the data for the function localize
38 TDOA_2 = [TDOA(2,1);TDOA(3,1);TDOA(4,1);TDOA(5,1);TDOA(3,2);TDOA
           (4,2);TDOA(5,2);TDOA(4,3);TDOA(5,3);TDOA(5,4)];

```

distance	mic 1	mic 2	mic 3	mic 4	mic 5
mic 1	0	388.1667	434.2083	187.0000	160.0833
mic 2	-388.1667	0	46.0417	-201.1667	-228.0833
mic 3	-434.2083	-46.0417	0	-247.2083	-274.1250
mic 4	-187.0000	201.1667	247.2083	0	-26.9167
mic 5	-160.0833	228.0833	274.1250	26.9167	0

Table 2: TDOA matrix with data of measurement of peak detection figure 5.

1.6 Localization

In EPO-4 manual, a solution given to localize the car with the TDOA data. This algorithm is partially implemented. The range difference r_{ij} consist of the difference d_i from microphone i to the car and the difference d_j from microphone j to the car. With the range difference, a hyperbolic curve can be drawn. When all the range difference are drawn, there will be one point where all lines will cross. That is the point where the beacon is. To get this point, a set of quadratic equations need to be solved. Luckily these equations can be transformed to a set of linear equations. This is done in the following way:

$$r_{ij} = d_i - d_j \Rightarrow (r_{ij} + d_j)^2 = d_i^2 \Leftrightarrow r_{ij}^2 + d_j^2 + 2r_{ij}d_j = d_i^2 \quad (1)$$

Inserting

$$d_i^2 = \|\mathbf{x} - \mathbf{x}_i\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_i^T \mathbf{x}$$

$$\text{and } d_j^2 = \|\mathbf{x} - \mathbf{x}_j\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{x}_j\|^2 - 2\mathbf{x}_j^T \mathbf{x}$$

into equation 1, $\|\mathbf{x}\|^2$ can be eliminated. Here is \mathbf{x} the position of the beacon and $\mathbf{x}_i, \mathbf{x}_j$ are positions of the microphones. The resulting equation in vector notation is

$$\begin{bmatrix} 2(\mathbf{x}_j - \mathbf{x}_i)^T & -2r_{ij} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ d_j \end{bmatrix} = r_{ij}^2 - \|\mathbf{x}_i\|^2 + \|\mathbf{x}_j\|^2$$

This can be done for all microphone pairs, which results in the following matrix of the form $\mathbf{A}\mathbf{y} = \mathbf{b}$

$$\begin{bmatrix} 2(\mathbf{x}_2 - \mathbf{x}_1)^T & -2r_{12} & & & \\ 2(\mathbf{x}_3 - \mathbf{x}_1)^T & & -2r_{13} & & \\ 2(\mathbf{x}_4 - \mathbf{x}_1)^T & & & -2r_{14} & \\ 2(\mathbf{x}_5 - \mathbf{x}_1)^T & & & & -2r_{15} \\ 2(\mathbf{x}_3 - \mathbf{x}_2)^T & & -2r_{23} & & \\ 2(\mathbf{x}_3 - \mathbf{x}_2)^T & & & -2r_{24} & \\ 2(\mathbf{x}_3 - \mathbf{x}_2)^T & & & & -2r_{25} \\ 2(\mathbf{x}_4 - \mathbf{x}_3)^T & & & -2r_{34} & \\ 2(\mathbf{x}_5 - \mathbf{x}_3)^T & & & & -2r_{35} \\ 2(\mathbf{x}_5 - \mathbf{x}_4)^T & & & & -2r_{45} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} r_{12}^2 - \|\mathbf{x}_1\|^2 + \|\mathbf{x}_2\|^2 \\ r_{13}^2 - \|\mathbf{x}_1\|^2 + \|\mathbf{x}_3\|^2 \\ r_{14}^2 - \|\mathbf{x}_1\|^2 + \|\mathbf{x}_4\|^2 \\ r_{15}^2 - \|\mathbf{x}_1\|^2 + \|\mathbf{x}_5\|^2 \\ r_{23}^2 - \|\mathbf{x}_2\|^2 + \|\mathbf{x}_3\|^2 \\ r_{24}^2 - \|\mathbf{x}_2\|^2 + \|\mathbf{x}_4\|^2 \\ r_{25}^2 - \|\mathbf{x}_2\|^2 + \|\mathbf{x}_5\|^2 \\ r_{34}^2 - \|\mathbf{x}_3\|^2 + \|\mathbf{x}_4\|^2 \\ r_{35}^2 - \|\mathbf{x}_3\|^2 + \|\mathbf{x}_5\|^2 \\ r_{45}^2 - \|\mathbf{x}_4\|^2 + \|\mathbf{x}_5\|^2 \end{bmatrix}$$

With this the location of the car can be calculated. This can be done by computing the pseudo-inverse of \mathbf{A} , i.e. $\mathbf{y} = \mathbf{A}^\dagger \mathbf{b}$.

To confirm that this algorithm works, a function is written to generate TDOA data. The function `make_tdoa` can be found in appendix A.1.4. This function calculates the range from all the microphone to the car. With this the difference between the microphone pairs is calculated. With this data the localization was tested and confirmed to work most of the times. When the given position is on the middle axis of the field, the deviation is maximum of 220 cm in a field of 600 cm \times 600 cm. To counter this, the least squares solution is used. Least squares is also not perfect, but the deviation is maximum 13 cm.

location [x, y]	pseudo inverse	LS
300, 300	300, 300	300, 300
300, 150	300, 175	300, 152
300, 0	300, 30	300, 3
300, 450	300, 381	300, 445
300, 600	300, 420	300, 589
150, 300	220, 300	154, 300
600, 300	380, 300	587, 300

Table 3: Example of deviations from original position

1.7 Checking calculated position

To check if the found location is correct, it will be checked for validity. This is done by using the previous (verified) location. If the car drives straight for a certain amount of time, the traveled distance cannot be larger than a known value. For instance, if the car drives for 1 second the travelled distance is about 0.75 m. If the found location is more than 1 m away, it cannot be correct and will be rejected. Also the cars orientation is kept and checked. For instance, if the car is known to be driving parallel to the positive x axis, and to get from the verified location to the found location, a sharp turn has to be taken, the found location cannot be correct and will be rejected. Because the location algorithm is not accurate in the middle of the playing field, every location found within a meter from the middle will be rejected. In figure 8 the rejection zone can be seen. When the car is in the middle it will drive as long as necessary to get out of the middle. When a location outside the 1 meter radius from the middle is found, this location is tested for validity. Also a location more then 1 meter outside the playing field is rejected. The matlab code for localization and checking of positions can be found in appendix ??.

1.8 Resulting Design

The combined design for determining the position of the car is implemented in the control loop. First the function TDOA in listing 5 is to record audio and pass this on to the `ch5_tdoa_final` function. As seen in this function audio will be recorded from 5 channels with a sample rate of 48 kHz, the time to record will be 0.25 s. This time is enough to guarantee that at least 2 samples will be within the recorded sample, one of these samples will be complete enough to use for the deconvolution.

Listing 5: function TDOA

```
1 function [ TDOA_data ] = TDOA
2 %TDOA runs the tdoa and returns a column with distances differences
3 Fs = 48000;
4 Trec = 0.25;
5 disp('Sampling');
6 measured_data = pa_wavrecord(3, 7, Fs*Trec, Fs, 'asio');
7 disp('TDOA calculating');
8 [TDOA_matrix, TDOA_data] = ch5_tdoa_final(measured_data);
9 end
```

Within `ch5_tdoa_final` the data will be split in data of each microphone, this data will be the input of the channel estimation. The channel estimation will return a estimated channel response, this response will be the input of the peak detection each microphone will receive their own channel response. The peak detection will pass the found position to a simple loop that creates the TDOA matrix. The function `ch5_tdoa_final` can be found in appendix ??. This matrix will be the input of the localization algorithm, this algorithm determines the position of the car in x,y and z coordinates. Only the x and y coordinates are of importance, these will be updated in the global variable `position`. In the localization function the determined position will be checked if it is correct. If it does not "pass" the TDOA will be measured again, after three failed attempts to determine the position the car will first move before attempting again.

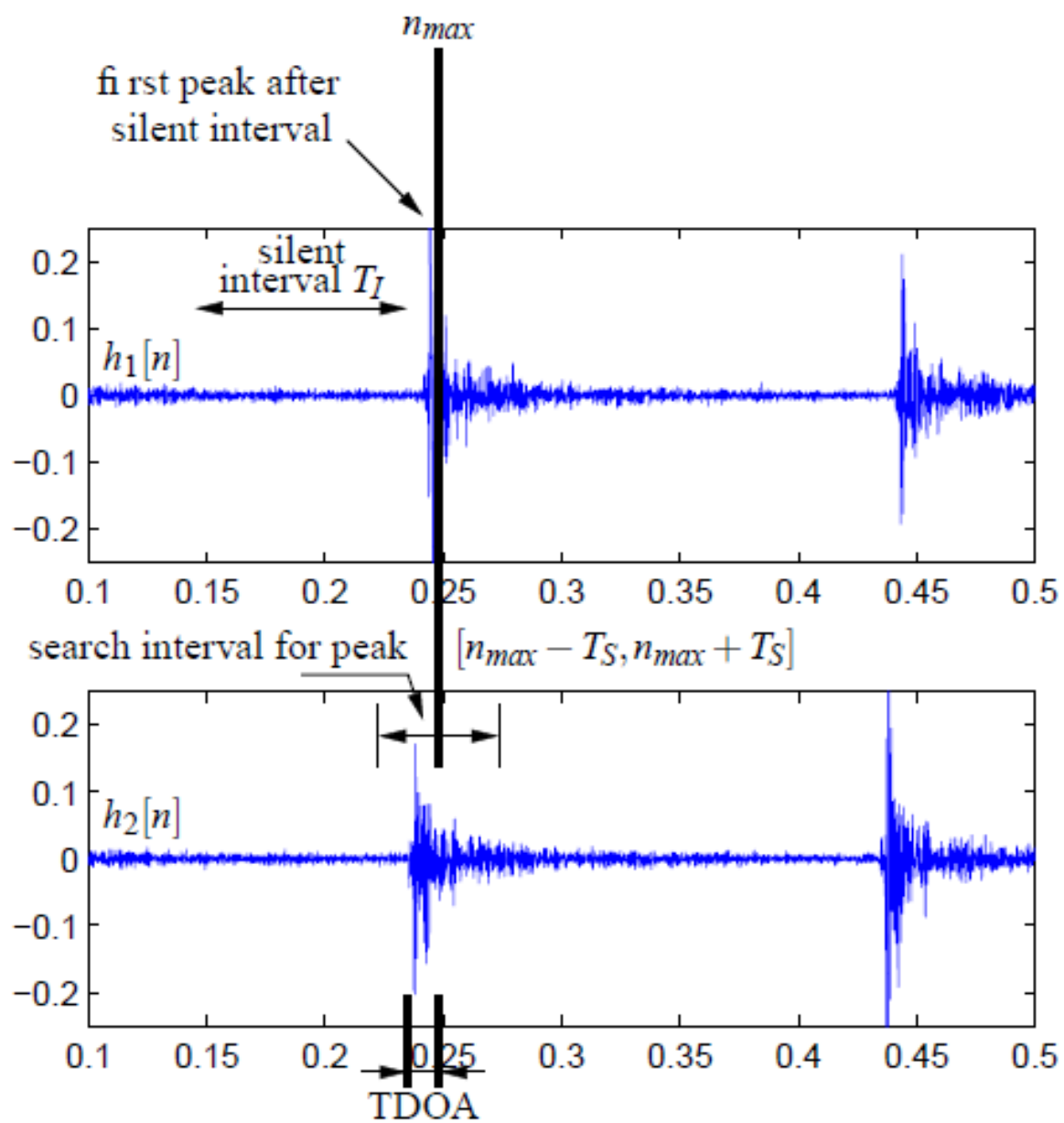


Figure 7: TDOA search window

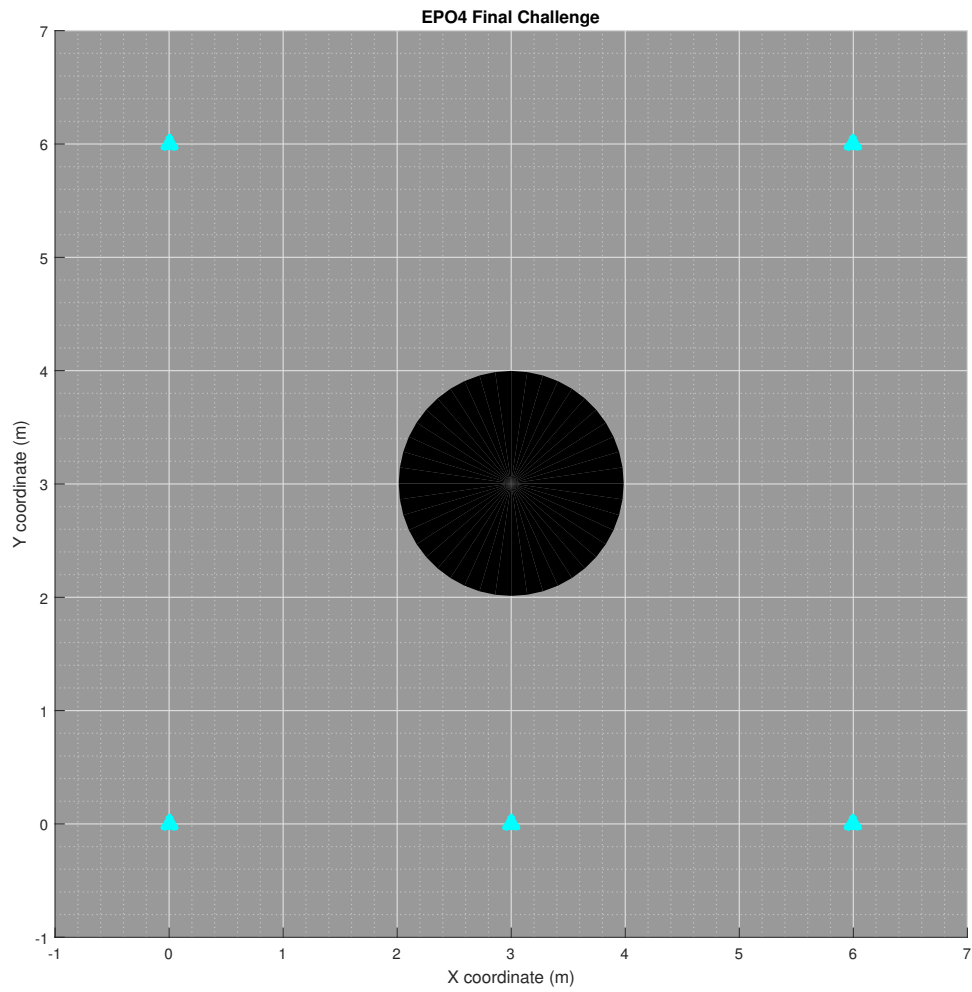


Figure 8: Zone of all locations that will not be put through planner cause of uncertainty in the localization, instead the car will drive a bit further for better localization results.

1.9 Testing

For the testing of the multichannel localization, a test setup was used. In table 4 the bounds of the field are the positions of the microphones one through four. With this field ten measurements are done, the calculated position of these measurements can be seen in table 5. The reference used by the deconvolution is a measurement at one of the microphones. Because of the small distance to the microphone this can be seen as the impulse response of the training sequence through the microphone.

Looking at the results in table 5 the localization is not accurate and does not always work. During the test it was discovered that the first localization algorithm used does not provide an accurate location when the beacon is close to the middle of the field. This is because two of the TDOA values are zero, this happens when microphones are at equal distance from the beacon. For the horizontal middle line the distance from microphone 1,4 and 2,3 to the beacon is the same. This makes it hard to calculate a correct x position, the y position will be calculated correctly. For the vertical middle line the microphone pairs 1,2 and 3,4 are the same, this prevents the localization to correctly calculate the y position.

Setting of the beacon parameters:	Nbits: 32 Timer0: 2 Timer1: 4 Timer3: 7
(x,y,z) locations of the microphones:	Mic 1: 0, 0, 30 Mic 2: 413, 0, 30 Mic 3: 413, 210, 30 Mic 4: 0, 210, 30 Mic 5: 173, 0, 77
Measurement 1:	Beacon at Mic 1
Measurement 2:	Beacon at Mic 2
Measurement 3:	Beacon at Mic 3
Measurement 4:	Beacon at Mic 4
Measurement 5:	Beacon at Mic 5
Measurement 6:	Location(x,y,z): (A) 102, 70, 26
Measurement 7:	Location(x,y,z): (B) 155, 105, 26
Measurement 8:	Location(x,y,z): (C) 216, 88, 26
Measurement 9:	Location(x,y,z): 203, 210, 26
Measurement 10:	Location(x,y,z): 355, 75, 26

Table 4: Measurement position of test setup.

measurement	1	2	3	4	5
x	-0.7041	250.8521	393.7024	3.8097	167.8806
y	-9.9271	394.9473	199.2680	201.5119	1.4720
z	-16.3226	1.1500e+03	19.9370	10.4180	229.1798
measurement	6	7	8	9	10
x	109.1353	216.1429	220.9459	204.5990	136.2869
y	73.1398	105.0000	75.3259	63.1943	176.8470
z	-10.6656	-594.9538	367.3060	-529.2243	-217.4669

Table 5: Localization results for tests.

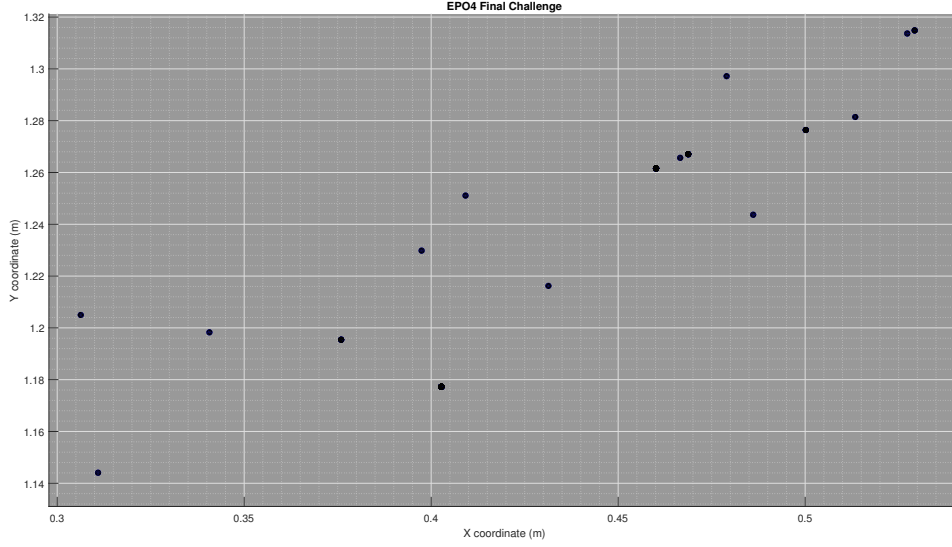


Figure 9: Positions detected by the code on the final test field

1.9.1 Field testing

The TDOA and localization code is also tested on the field that will be used for the final challenge. The car stopped at a position and the code ran several times. In figure 9 the results are visualized, the black dots represent the positions detected. The localization of the car is determined with an accuracy of ± 10 cm. This should be enough for the car to reach it's destination within 30 cm, which is required for the final challenge.

1.10 Conclusion

The localization of the car is done with a reasonable accuracy and it should be high enough for the car to reach it's goals. However with a sample rate of 48kHz the location of the car could be determined more precise. Sound travels around $1/(48 \text{ kHz}) \cdot 340 \text{ m s}^{-1} = 0.7 \text{ cm}$ per sample so it should be possible to determine the position of the car with a higher resolution than ± 10 cm. The accuracy of the position is reduced when the car approaches the positions where $x = 3\text{m}$ or $y = 3\text{m}$. This issue is now solved through filtering the positions and driving a bit further in case the car is near the mid lines ($x = 3\text{ m}$ or $y = 3\text{ m}$) of the field and no possible correct position is detected. It could be resolved **though** by using a different algorithm for the localization of the car.

2 Route planning

2.1 Introduction

The role of the route planner is to decide how the car should move given its objective. This can be a number of different things, for example, making a turn, driving a straight line or stopping all-together.

2.2 Design

The only variable that is really fixed is the **route** variable, containing the waypoints (if there are any) and the destination. All other variables and measurements are approximations and are prone to errors and uncertainties. That's why the design of the planner is made with compensation and self-correcting features in mind from the start.

This is the reason for the main design choice, namely that there is no fixed trajectory which is tracked. Instead, the **planner** is executed 'fresh' after each successful localization attempt by the Control Loop, which is explained in further detail in section 3. This means that the only thing that the planner needs to know in addition to the aforementioned **route** variable, is the current position and orientation of the car.

2.2.1 Specifications and Goals

The information available to the planner is summed up in table 6.

Name	Explanation
<code>position</code>	matrix containing all the known positions and orientations of the car, with the first entry being the start position and the last entry being the current position.
<code>car.voltage</code>	contains the current voltage of the car in Volts.
<code>static_positions.route</code>	contains the waypoints (if there are any) and the destination of the car in x and y coordinates.
<code>static_positions.point</code>	keeps track of which waypoint/destination the car is traveling to.
<code>car.steer_straight</code>	contains the steer setting for driving straight.

Table 6: Information and variables available to the planner.

The data and information that the planner should provide to the Control Loop is available in table 2.2.1.

Name	Explanation
<code>speed</code>	the cars needed speed setting including voltage drop compensation.
<code>car.d.theta</code>	difference in angle between the car and the next waypoint/destination.
<code>car.v_factor</code>	factor between 0 and 1 depending on the starting voltage and the current voltage of the car.
<code>steer</code>	steer setting of the car depending on whether to make a turn or not and which way.
<code>car.status</code>	status of the car, 1 when at a waypoint, 2 when at the destination and 0 elsewhere.
<code>car.did_turn</code>	toggles between 0 and 1 reflecting whether the last driven action of the car was a turn or not.
<code>time</code>	time in seconds the car has to move, this movement can be a turn, driving straight or standing still, depending on the other output variables.

Table 7: Information and variables returned to the Control Loop.

2.2.2 Code

The code of the Planner, which can be found in appendix A.2, consists of two parts. In the first part all the needed information is generated and in the second part a decision tree build out of if-statements figures out what needs to be done with that information.

speed setting adjustment

When driving with supercaps instead of a battery pack there is a significant voltage drop over **time**, to compensate for this the car will drive at different speed settings at different voltage levels. In addition to this, every car behaves differently, therefore the values in this code are determined by trial and error, which is more generally explained in paragraph 2.3.1.

Listing 6: Speed setting adjustment for different Voltages

```
8 if car.voltage >= 17 % speed compensation for voltage drop
9     speed = 158; % IMPORTANT, CALIBRATE FOR 1.3s for 1m
10     disp('Voltage level: normal')
11 elseif car.voltage >= 14 % adjust by hand accordingly
12     speed = 162;
13     disp('Voltage level: low')
14 else
15     speed = 165;
16     disp('Voltage level: drained')
17 end
```

generation of information

First, the number of fixed points is extracted from the **static_positions.route** variable. Then the angle that the car needs to turn and distance it has to travel is calculated using basic trigonometry and the Pythagorean theorem. Finally the voltage compensation factor is determined using the current car voltage.

Listing 7: Generation of needed information

```
21 [~, numberofpoints] = size(static_positions.route); % number of
    points (waypoints + destination)
22 d_x = static_positions.route(1, static_positions.
    point)-position(1, end); % delta x
23 d_y = static_positions.route(2, static_positions.
    point)-position(2, end); % delta y
24 desired_theta = atan2d(d_y, d_x); % desired orientation of the
    car
25 car.d_theta = desired_theta - position(3, end); % difference
    between car orientation and desired orientation
26 distance = sqrt(d_x^2 + d_y^2); % distance from car to
    destination/waypoint
27 test_data.d_theta = [test_data.d_theta, car.d_theta];
28 car.v_factor = 1; %max car voltage / car.voltage;
```

Decision tree consisting of nested if-statements.

This part of the code performs the necessary checks to determine what drive settings need to be generated. The first check that occurs is whether the car has arrived within 30 cm of its current destination, which can either be a waypoint or the final destination. When this is the case the `planner` outputs variables to the Control Loop to stand still and informs whether it's a waypoint or destination.

Listing 8: Check 1: Destination/Waypoint check

```
32 if distance <= 30
33     steer = car.steer_straight;
34     speed = 150; % standing still
35
36     if static_positions.point >= numberofpoints
37         time = 40;
38         disp('ARRIVED AT DESTINATION')
39         car.status = 2;
40     else
41         static_positions.point = static_positions.point + 1; % keep
42         track of where we're going
43         disp('ARRIVED AT A WAYPOINT')
44         time = 6;
45         car.status = 1;
46     end
47 else
```

When that check returns false the car needs to move, first the code checks whether the previous movement was a turn. This is done because the cars orientation needs to be calculated and this can only be done when a straight line has been driven from its previous to its current location. So if the previous movement was a turn, the car is ordered to drive a bit in a straight line so that the next time it performs this check it will not have made a turn.

Listing 9: Check 2: turn check

```
47     car.status = 0;
48     if car.did_turn == true
49         time = 0.6 / car.v_factor;
50         steer = car.steer_straight;
51         car.did_turn = false;
52         car.did_last_turn = true;
53         disp('Driving straight a bit to provide data for
54             orientation after the turn.')
55     else
```

The next part checks whether the car needs to make a turn, if it doesn't need to make a turn the code differentiates between 2 possible situations. Situation 1: the car is almost at its current destination. If that is the case, the function `straight`, found in appendix A.2.3, returns the time it takes to drive to that destination. Situation 2: the car has quite a long way to go before reaching its destination, so it drives a predefined amount of time in a straight line. It does this **because, due** to the aforementioned inaccuracies and uncertainties, it can occur that it is slightly off its course. Therefore this will all resolve itself by running the `planner` again after getting a bit closer to the goal destination.

Listing 10: Driving in a straight line towards the waypoint/destination.

```

57     if abs(car.d_theta) <= 5 % no turn needs to be made
58         disp('No turn!');
59         if distance <= 150 % the car is close to the waypoint /
           destination.
60             time = straight(distance);
61             steer = car.steer_straight;
62             disp('Almost there.')
63         else % distance is greater than 1.5m
64             time = 1 / car.v_factor; % return a 1s drive
65             steer = car.steer_straight; % straight
66             disp('Not there yet.')
67         end
68     else % turn is needed

```


If it has to make a turn it calculates the time it needs to do that using the `turn` function, found in appendix A.2.2, and provides the correct steer setting depending on whether the variable `car.d_theta` is positive or negative.

Listing 11: Making a turn.

```
55         disp('Lets turn')
56         car.did_turn = true; % make sure that we drive a bit
           in a straight line after a turn is made
57         if car.d_theta <= 0
58             if d_x <= 0 % fix when driving from right to left
59                 steer = 200;
60                 disp('to the left!')
61             else
62                 steer = 100;
63                 disp('to the right!')
64             end
65         else
66             if d_x <= 0 % fix for when driving from right to
           left
67                 steer = 100;
68                 disp('to the right')
69             else
70                 steer = 200;
71                 disp('to the left!')
72             end
73         end
74         time = turn(car.d_theta);
75     end
76 end
77 end
```

2.3 Testing

To implement the **turn** and **straight** functions mentioned in the previous code section, several tests were done to determine the time it takes to make a certain turn or drive in a straight line for a certain length. Because testing with supercaps was **unallowed**, testing was done with the batterypacks instead. This meant that no detailed effects of the voltagedrop on the timings could be found. To simplify things further, linear approximations were used, it was decided that as long as the timings were not causing overshoot, the final goal should be reached because of the compensating and self-correcting behaviour of the system.

To determine the linear approximation of our turn timings the **measurementdata** shown in table 8 was inter- and extrapolated. This resulted in the following code in the **turn** function (found in appendix A.2.2).

Listing 12: The **turn** function.

```
8 if d_theta > 5
9     time = (0.0142222222*d_theta + 0.5) / car.v_factor;
10 elseif d_theta < -5
11     % negative d_theta
12     time = (-0.0142222222*d_theta + 0.5) / car.v_factor;
13 else
14     time = 0;
15 end
```

Angle (d_theta)	time
-180	3.45 s
-90	2.05 s
0	0 s
90	2.11 s

Table 8: Measurement data for linear approximation of turn timings.

To determine the linear approximation of the timings for driving in a straight line we calibrated the speed setting to produce a 1.3s drive for covering 1 m. The resulting **straight** function can be found in appendix A.2.3.

To get an idea of the behaviour of the **planner** when included in the Control Loop, small messages were added to the code. These messages show up in the console while driving with updates on what decisions are made was chosen and in conjunction with saved test data, errors or unexpected behaviour was easy to spot. However as has become apparent while doing the final challenge, there was an error that slipped through, which will be mentioned in section ??.

2.3.1 Trial and Error & Findings

Between each run small changes to the variables were made to ensure a viable 'maximum car voltage' performance. These found settings would degrade when the voltage would start to drop but the code is able to compensate for that by rerunning itself until the car arrives at the destination. This means that in optimal state the car would make only one turn per waypoint and more turns when the voltage drops makes the turns less precise.

During both standalone and 'as part of the control loop' testing, unexpected behaviour was found. It turned out, for example, that when driving from the right to the left side, instead of the usual left-to-right route, the car would turn left when it needed to make a right turn and vice-versa. This has been fixed by first checking the driving direction before determining whether a turn should be to the right or to the left.

2.4 Conclusion

This implementation of route planning has proven to be useful due to its ability to recover from errors caused by invalid localization or other uncertainties. Unfortunately it failed the distance check as mentioned in listing 8 and 10 because of a conversion error during the final challenge, but other than that, the design and implementation proved to be simple and effective.

To improve on this design the linear approximations and guessed factors could be removed and replaced by measured curves for different voltage levels. Also it could be expanded further to include collision detection handling, which could include driving backwards and logging found obstacles.

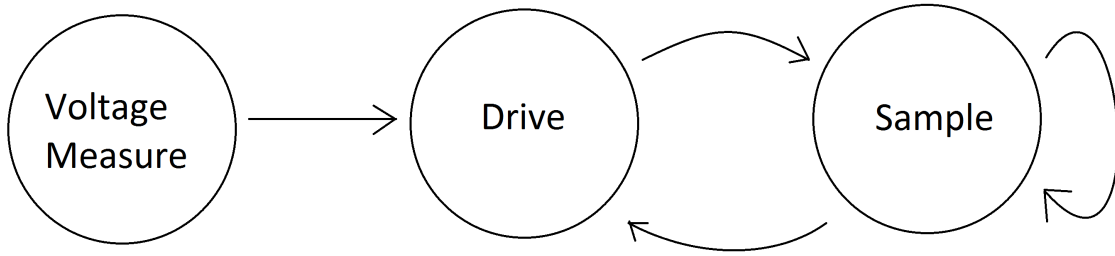


Figure 10: State machine implemented in the control loop

3 Control Loop

The control loop is the main function. All other functions are called from this control loop.

3.1 Overview

To implement the control loop, a MATLAB `timer` is used. The timer object is very useful because it handles a few things which are hard to implement in a normal while loop. A timer has 4 main functions, which are `StartFcn`, `TimerFcn`, `StopFcn` and `ErrorFcn`. To initialize the timer, the control loop function is called. Here are `StartFcn`, `TimerFcn` etc. assigned. Also, 5 global variables are initialized. These variables are global so they don't have to be in the input and/or output of a function. For instance, one of these globals is to keep test data. From within every function this variable can be altered.

The `timer_startFcn` is the `StartFcn` of the timer. This is used to initialize the figure to map the car. This function is called just before the car starts driving. The `timer_loop` is the `TimerFcn` and is the main loop. In this function the car will drive and all the functions are called. A state machine is used to keep everything clear and structured. The 3 states are `VoltageMeasure`, `Drive` and `Sample`. These states will be explained in later. The `timer_stopFcn` is the `StopFcn` of the timer. It is used to clean when the function is done driving the car. Last but not least is the `timer_error`, which is the `ErrorFcn`. This function can handle every error thrown by one of the other functions. When an error is thrown, this function will send a command to the car to stop driving. This is so the car won't drive without control and drive against an object.

3.2 VoltageMeasure

For the final challenge the car has to be charged wireless, this is done with a air-core DC/DC converter. The super capacitors need to be charged to 17 V. Via the wireless communication the current voltage can be read. The voltage measured fluctuates heavily, because of this an outlier will count as 17 V. To make sure the super caps are charged to 17 V the measurement will count each time a value above 17 V is measured, if the count is five it will assume the car is charged. After the car is charged the next state will be `Drive`.

3.3 Drive

First thing that this state does is call the planner function. In the planner the parameters for the car are calculated. If the car is near its target, the planner will give a signal to the control

loop. When this signal is given, a sound will be played by the computer to indicate that it has arrived. When all the parameters are determined, a drive command has to be sent to the car to make it drive. A separate function is made to do this to be sure all the parameters are determined before transmitting it to the car. The function `drive_car.m`, in appendix ??, has as inputs a speed setting, steer setting and a time to drive. It executes the drive command for the time needed using a pause command to wait, before transmitting a roll out command which is a speed setting of 150 and a steer setting that corresponds to driving a straight line. After this is done, the next state will be **Sample**.

3.4 Sample

First the TDOA data will be determined. With this data, a location will be determined with the function `localize`. If the found location passes all tests, the next state will be **Drive**. But if the found location does not pass the tests, a fail code will be given. When the localization fails, the next state will be **Sample** and new TDOA data will be determined. This will be done until 3 tests fail. When 3 tests fail, the car will drive a little bit forward to find a better spot for a measurement. When the tests fail 9 times in a row, a different action is taken depending on the failed test. When the found location fails because it is too far away, after 9 times it is possible that it is actually correct. So this location will then be accepted. If the location fails because it is at the wrong angle, the car will drive straight until it finds the correct angle. If it does not pass the test within 10 tries, the position will be accepted and a new orientation will be determined.

3.5 Mapping the car

To visualize the location of KITT the matlab class `EPO4figure` from blackboard, appendix ??, is used to map the car location, mic positions, waypoint, destination and obstacle locations.

The class has 5 functions: `setMicLoc`, `setWayPoint`, `setDestination`, `setKITT` and `setObstacle` which all require a location in some form as input. A variable `start` is added to the function `setKITT` to prevent the figure to load the last position from a previous test. The last position of the car is shown as a blue dot and the past positions of the car are shown as a black dot, the mic positions are the light blue dots and the violet ones are the destination and waypoint positions as can be seen in figure 11.

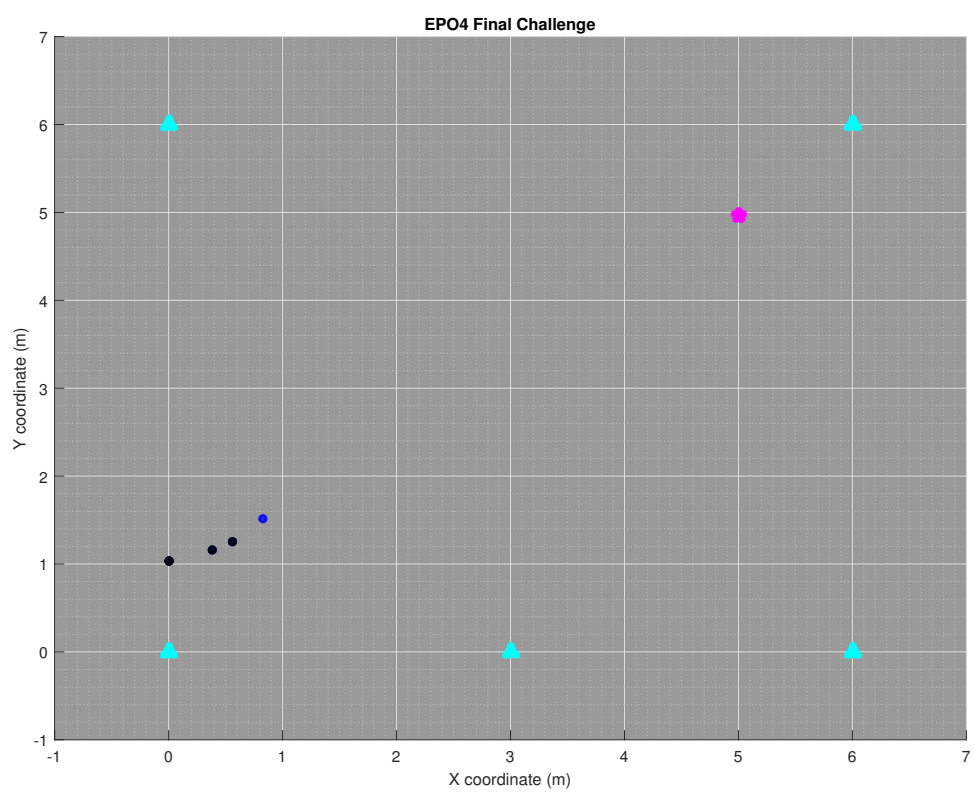


Figure 11: Mapping the locations

4 The Final Challenge

This section describes the performance of our system during the Final Challenge. The car didn't succeed in stopping at the destination and therefore it didn't meet our **expectations** however we are still satisfied with the **code** we created. Unfortunately some small mistakes were made and these will be explained below.

4.1 Objective

As already mentioned the first challenge was to drive from point A (0, 103) to point B (500,497).

4.2 Training sequence

The **training** sequence used for the final challenge is Timer1: 2, Timer2: 0, Timer: 9. These settings are used due to the limited availability of working cars.

4.3 Reference

A different reference signal was used for the final challenge. For completeness of the report this reference signal is shown in figure 12. To create this reference signal a measurement was done close (5cm) to microphone 1 and the desired pulse was cut from this data. The start point of the reference is the first point after the silence period of the signal. At the end a small silence part is included.

4.4 What went wrong and how was it solved?

In figure 13 the accepted car positions are shown. Near the destination, pictured by the pink dot, a strange turn can be observed. The car actually steers away from the destination. The test data showed that the sign of the calculated angle was inversed when the car **drives** from right to left. This bug was easily solved by checking the driving direction before issuing a steer setting command, as can be seen in listing 11.

In the final run of the final challenge the car actually reached the destination for the first **time**, the only problem was that it didn't pause there but continued driving. After inspection of the code it became clear that the distance was not calculated correctly and was off by a factor 100. This behaviour can be seen in figure 14. The car still tried to make a turn to approach the destination again but it hit a table on the outside of the field while turning and therefore we stopped the run. It is highly unlikely the cars position would have gotten within 0.3 cm accuracy of the goal and therefore it would have made turns around the destination until the batteries were empty.

In figure 14 the accepted positions are shown for the final run.

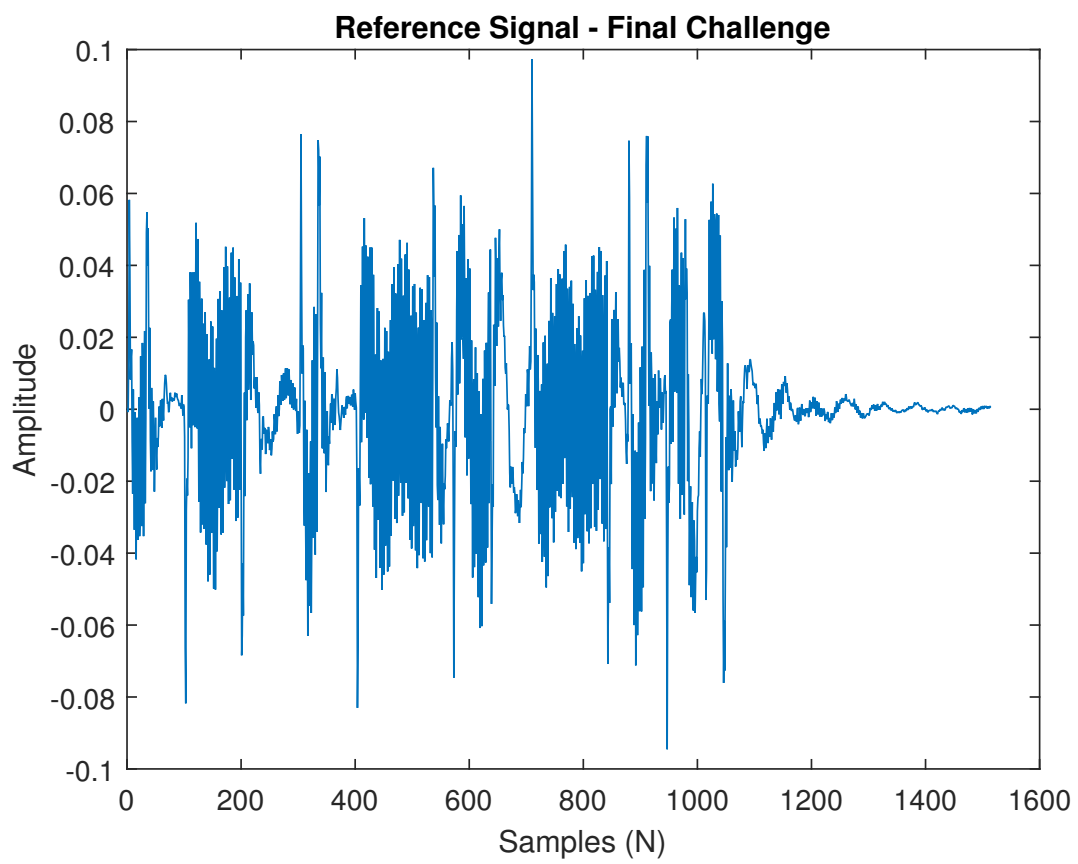


Figure 12: Reference signal used for the Final Challenge

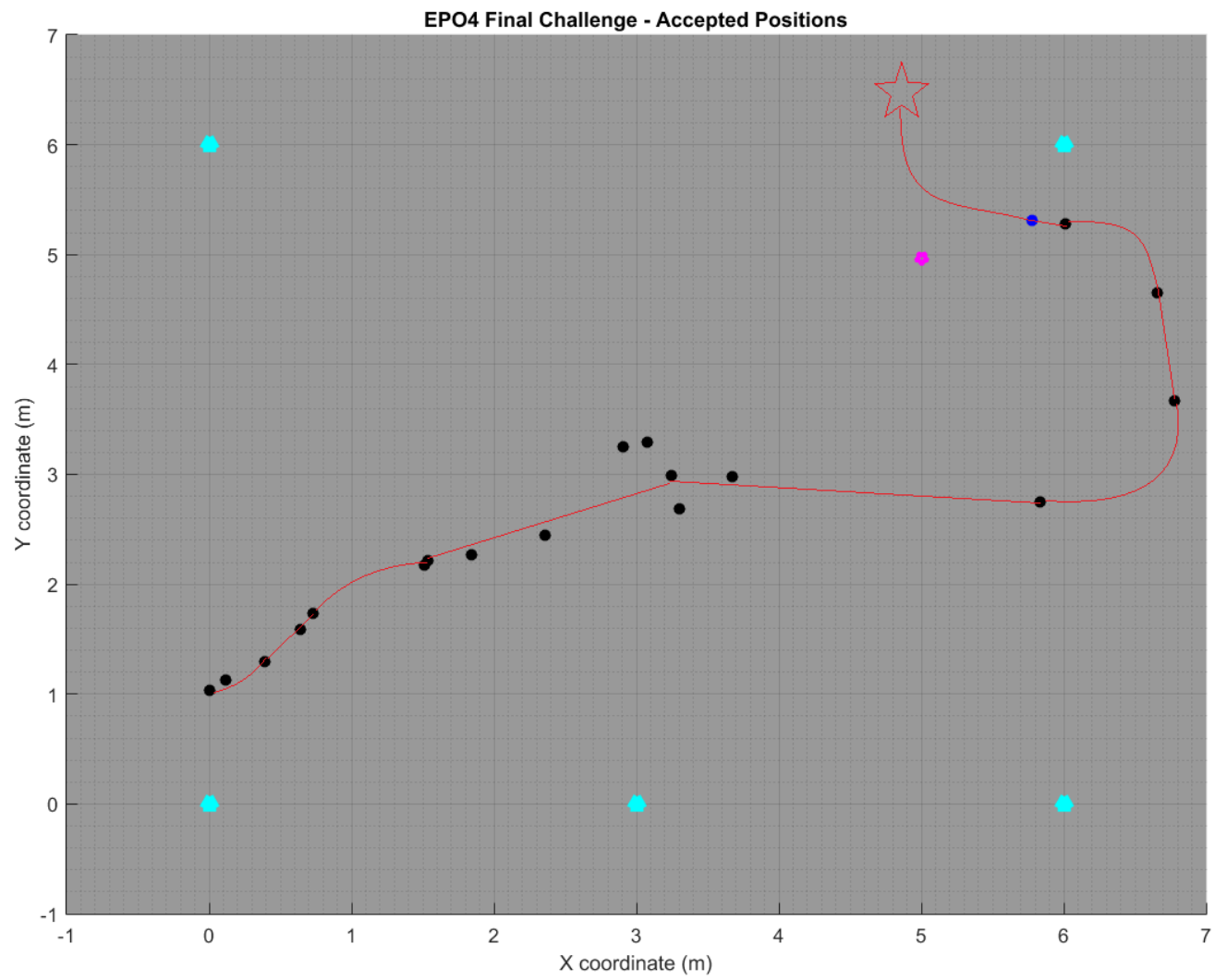


Figure 13: Map of the accepted car locations during a run, the red line is the observed route the car took

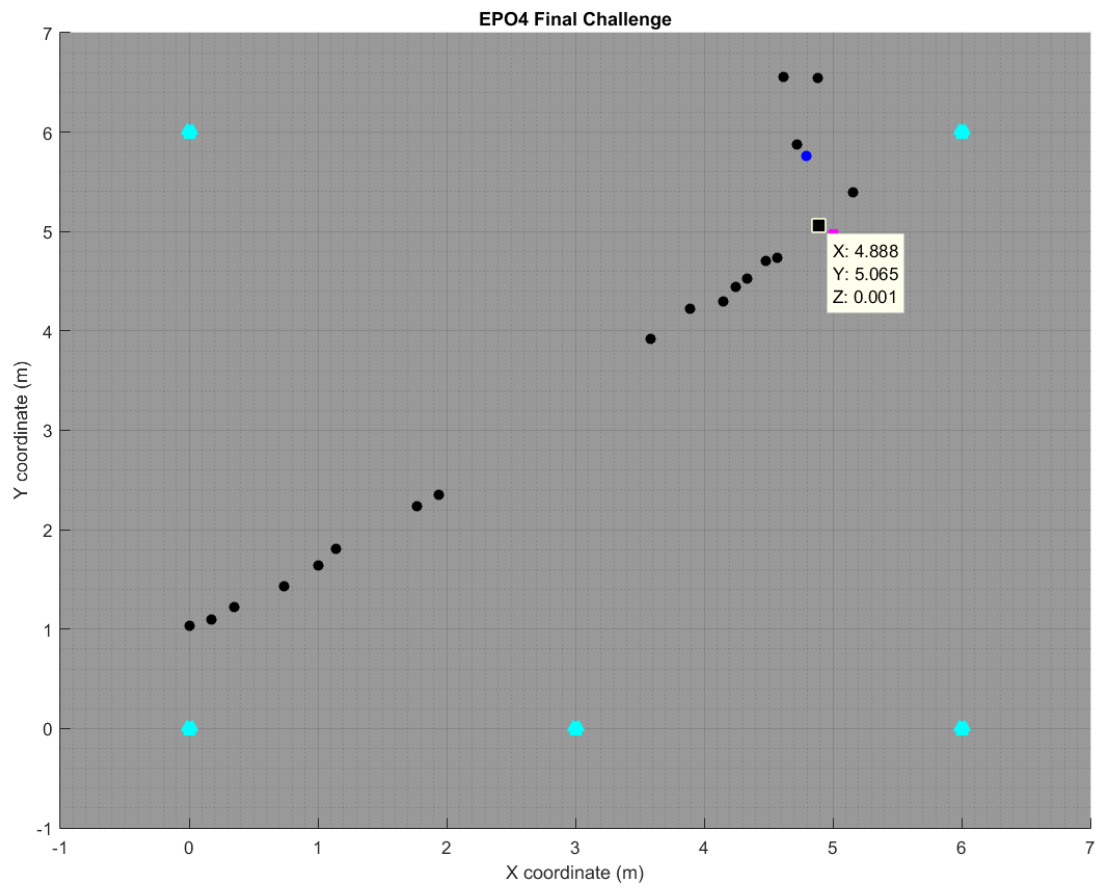


Figure 14: Map of the accepted car locations at the final run, the position selected is in stopping range of the destination

5 Conclusion and Discussion

5.1 Conclusion

In the end the control loop operated well. The TDOA and Localization algorithm were able to determine the position within ± 10 cm accuracy. The route planning worked aswell. We are convinced it will complete the first two challenges if given another chance.

5.2 Discussion

As mentioned before we didn't succeed in completing the objectives on the moment of truth, the final **challenge**, this was due to some small mistakes made in the code. These mistakes could have been detected before the challenge if the individual codes were tested more thoroughly. The localization algorithm used determines the location of the car with reasonable accuracy. When the car is in the middle of the field the location gets less precise for the mid lines ($x = 3\text{m}$ or $y = 3\text{m}$) of the field the location is determined with less accuracy as well. This could have been solved by using a more advanced algorithm for the location estimation. The obtained resolution on all positions with the current algorithm is high enough to complete the challenges and therefore we didn't elaborate this further. To get a higher precision with the current algorithm it would be an option to make a reference signal for every microphone because the microphones have their own slightly different characteristics this should result in a better channel estimation and therefore a better TDOA and better localization.

6 Planning and Teamwork

6.1 Planning

The planning of the final challenge is divided in four weeks. The first three lab day sessions are used for localization and multichannel measurements. the three lab day sessions after are meant for system integration, putting all parts together and writing functions to plan the route and control the car. The last few lab days are used for testing with the final setup. The final report deadline is planned for Friday 19 June noon, for this it was decided that two days before this deadline at 18:00 the individual works are finished and afterwards the written pieces will be corrected and if something is missing written. The final report was divided into parts and each individual will write the pieces they worked on, this can be reviewed on the next page.

6.2 Teamwork

Introductie		Martijn
	Abstract	Martijn
	Introduction	Martijn
Specs		Thomas
	Specs	Thomas
Planning/Teamwork		Martijn
	Planning	Martijn
	Teamwork	Martijn
Plaatsbepaling		Martijn
	Introduction	Martijn
	Training sequence	Jeroen
	Reference	Martijn
	Deconvolution	Martijn
	Peak Detection	Richard
	Localization	Jeroen
	Checking calculated position	Jeroen
	Testing	Richard
	Conclusion	Richard
Object detection?		Martijn
	Introduction	Martijn
	Design	Martijn
	Testing	Martijn
	Conclusion	Martijn
Route planning		Thomas
	Introduction	Thomas
	Design	Thomas
	Testing	Thomas
	Conclusion	Thomas
Loop		Jeroen
	Timer Function	Jeroen
	Determining Voltage	Martijn
	Driving	Richard
	Plotting	Richard
	Challenges	Jeroen
	Error	Jeroen
Conclusie		Richard
Discussie		Richard
Appendix		Thomas

References

- [1] A.-J. van der Veen, *EE2L21 Project EPO-4: Autonomous driving challenge*, 24th ed. TU Delft Faculty of Electrical Engineering, Mathematics and Computer Science, 2015. [Online]. Available: https://blackboard.tudelft.nl/bbcswebdav/pid-2422827-dt-content-rid-8391321_2/courses/34309-141504/epo4manual.24apr.pdf

A Appendix

A.1 Localization

A.1.1 Open_com.m

```
1 comport = '\\.\COM3'; %Bluetooth Module: 3215
2 EPOCommunications('close');
3 result = EPOCommunications('open', comport);
4 status = EPOCommunications('transmit', 'S');
5 EPOCommunications('transmit', 'A1');
```

A.1.2 orientation.m

```
1 %Start Orientation is known. Use this as 0
2 %For info on atan2d see
3 %http://nl.mathworks.com/help/matlab/ref/atan2d.html
4
5 function Orientation%(position)
6 %Function to keep track of the orientation of the car
7 global position
8
9     %Take positive y axis as 0 degree
10
11     x_diff = position(1, end) - position(1, end-1);
12     y_diff = position(2, end) - position(2, end-1);
13
14     angle = atan2d(y_diff, x_diff); % * 180 / pi;
15
16     position(3, end) = angle;
17
18 end
```

A.1.3 matched_f.m

```
1 function h = matched_f(x,y)
2 Ny = length(y);
3 Nx = length(x);
4 L = Ny-Nx +1;
5 xr = flipud(x);
6 h = filter(xr,1,y);
7 h = h(Nx:end);
8 alpha = x' * x;
9 h = h/alpha;
10 end
```

A.1.4 matched_f.m

```
1 function tdoa = make_tdoa(matrix_mics_position , car_location)
2 % Generates test TDOA data
3 % input: mic positions , some field position
4 % output: TDOA data
5     [row, col] = size(matrix_mics_position);
6     dimensions = length(car_location);
7     if(col ~= dimensions) % check for same dimensions
8         error('Mics location dimensions dont match with x-location
9             dimensions');
10    end
11    elements = (row * (row - 1))/2;
12    tdoa = zeros(elements , 1);
13    mic1 = 1;
14    mic2 = 2;
15    for i = 1:elements
16        tdoa(i, 1) = (norm(matrix_mics_position(mic1, 1:col) -
17            car_location) - norm(matrix_mics_position(mic2, 1:col) -
18            car_location));
19        mic2 = mic2 + 1;
20        if(mic2 > row)
21            mic1 = mic1 + 1;
22            mic2 = mic1 + 1;
23        end
24    end
25 end
```

A.2 Planner

A.2.1 planner.m

```
1 function [time,steer,speed] = planner
2 %input: (global)position(x,y,orientation),(global)next_position(x,y
3 )
4 global car;
5 global position;
6 global test_data;
7 global static_positions;
8 % compensation for voltage drop of supercaps
9 if car.voltage >= 17 % speed compensation for voltage drop
10     speed = 158; % IMPORTANT, CALIBRATE FOR 1.3s for 1m
11     disp('Voltage level: normal')
12 elseif car.voltage >= 14 % adjust by hand accordingly
13     speed = 162;
14     disp('Voltage level: low')
```



```

14 else
15     speed = 165;
16     disp('Voltage level: drained')
17 end
18 % end of compensation for voltage drop of supercaps
19
20 % begin computation of needed information
21 [~, numberofpoints] = size(static_positions.route); % number of
    points (waypoints + destination)
22 d_x = static_positions.route(1, static_positions.
    point) - position(1, end); % delta x
23 d_y = static_positions.route(2, static_positions.
    point) - position(2, end); % delta y
24 desired_theta = atan2d(d_y, d_x); % desired orientation of the
    car
25 car.d_theta = desired_theta - position(3, end); % difference
    between car orientation and desired orientation
26 distance = sqrt(d_x^2 + d_y^2); % distance from car to
    destination/waypoint
27 test_data.d_theta = [test_data.d_theta, car.d_theta];
28 car.v_factor = 1; % max car voltage / car.voltage;
29 % end of computation of needed information
30
31 % start if-based decision tree
32 if distance <= 30
33     steer = car.steer_straight;
34     speed = 150; % standing still
35
36     if static_positions.point >= numberofpoints
37         time = 40;
38         disp('ARRIVED AT DESTINATION')
39         car.status = 2;
40     else
41         static_positions.point = static_positions.point + 1; % keep
            track of where we're going
42         disp('ARRIVED AT A WAYPOINT')
43         time = 6;
44         car.status = 1;
45     end
46 else
47     car.status = 0;
48     if car.did_turn == true
49         time = 0.6 / car.v_factor;
50         steer = car.steer_straight;
51         car.did_turn = false;
52         car.did_last_turn = true;

```

```

53         disp('Driving straight a bit to provide data for
orientation after the turn.')
54     else
55         car.did_last_turn = false;
56         disp('Check whether a turn has to be made.')
57         if abs(car.d_theta) <= 5 % no turn needs to be made
58             disp('No turn!');
59             if distance <= 150 % the car is close to the waypoint /
destination.
60                 time = straight(distance);
61                 steer = car.steer_straight;
62                 disp('Almost there.')
63             else % distance is greater than 1.5m
64                 time = 1 / car.v_factor; % return a 1s drive
65                 steer = car.steer_straight; % straight
66                 disp('Not there yet.')
67             end
68         else % turn is needed
69             disp('Lets turn')
70             car.did_turn = true; % make sure that we drive a bit
in a straight line after a turn is made
71             if car.d_theta <= 0
72                 if d_x <= 0 % fix when driving from right to left
73                     steer = 200;
74                     disp('to the left!')
75                 else
76                     steer = 100;
77                     disp('to the right!')
78                 end
79             else
80                 if d_x <= 0 % fix for when driving from right to
left
81                     steer = 100;
82                     disp('to the right')
83                 else
84                     steer = 200;
85                     disp('to the left!')
86                 end
87             end
88             time = turn(car.d_theta);
89         end
90     end
91 end
92 % end of decision tree
93 end

```

A.2.2 turn.m

```
1 function [ time ] = turn(d_theta)
2 global car;
3 %TURN Summary of this function goes here
4 % returns time to make a certain turn of d_theta
5
6 % Moet nog getweaked / gecalibreerd worden
7 % positive d_theta
8 if d_theta > 5
9     time = (0.0142222222*d_theta +0.5) / car.v_factor;
10 elseif d_theta < -5
11     % negative d_theta
12     time = (-0.0142222222*d_theta +0.5) / car.v_factor;
13 else
14     time = 0;
15 end
16 end
```

A.2.3 straight.m

```
1 function [ time ] = straight(distance)
2 global car;
3 %TURN Summary of this function goes here
4 % returns time to drive a certain distance lower than 1.5m (for
   now)
5
6 % Moet nog getweaked / gecalibreerd worden
7 time = (2/1.5) * distance %car.v_factor;
8
9 end
```

A.3 Loop

A.3.1 control_loop.m

```
1 function t = control_loop() %Timer functie met een acceleratie tijd
   en een remtijd
2
3 global position
4 global voltage
5 global car % need this don't delete
6 global static_positions
7 global test_data
8
9 fail_factor = 0;
```

```

10 drive_counter = 0;
11 middle_counter = 0;
12
13 test_data.pass = 0;
14 test_data.TDOA = [0;0;0;0;0;0;0;0;0;0;0];
15 test_data.measured = zeros(1,12000,5);
16 test_data.dtheta = 0;
17 test_data.cartime = [0 0 0];
18 test_data.pos_tdoa = [0;0;0];
19 static_positions.origin = [0;103;0]; %start position
20 static_positions.destination = [500;497;0];
21 static_positions.waypoint = [0;0;0];
22 %static_positions.waypoint = [193;400;0];
23 static_positions.point = 1; % need this in planner
24 [sound, Fs] = audioread('ObjectiveComplete.mp3');
25 player = audioplayer(sound,Fs);
26 if static_positions.waypoint == [0;0;0]
27     static_positions.route = [static_positions.destination];
28 else
29     static_positions.route = [static_positions.waypoint,
30         static_positions.destination];
31 end
32 static_positions.mic_positions = [0 0 40; 600 0 40; 600 600 40; 0
33     600 40; 300 0 46];
34
35 car.did_turn = false;
36 car.did_last_turn = false;
37 car.steer_straight = 150;
38 position = static_positions.origin; %Postion in centimeters
39
40 state = States.VoltageMeasure;
41 t = timer;
42 t.TimerFcn = @timer_loop;
43 t.StartFcn = @timer_startFcn;
44 t.StopFcn = @timer_stopFcn;
45 t.ErrorFcn = @(~,~)timer_error;
46 t.StartDelay = 0;
47 t.Period = 0.01;
48 t.ExecutionMode = 'fixedRate';
49
50 function timer_startFcn(timerObj, timerEvent)
51     disp('Started - Load figure');
52     EPO4figure; %Load the figure
53     EPO4figure.setMicLoc(static_positions.mic_positions/100) %
54         Update Mic Positions

```

```

53     EPO4figure.setDestination(static_positions.destination/100)
54     ;
55     EPO4figure.setKIT([position(1,end)/100 position(2,end)
56                       /100], 1); %Update car position
57     if static_positions.waypoint == [0;0;0]
58         disp('No waypoint');
59     else
60         EPO4figure.setWayPoint(static_positions.waypoint/100);
61     end;
62     EPOCommunications('transmit','A0');
63 end
64
65 function timer_loop(timerObj, timerEvent)
66     switch(state)
67         case States.VoltageMeasure
68             disp('Measuring Voltage')
69             measure_voltage;
70             if(voltage.done == true)
71                 state = States.Drive;
72             end
73         case States.Drive %Example, states and flow can be
74             altered.
75             disp('Driving straight');
76             fail_factor = 0;
77             drive_counter = 0;
78             %Status request
79             status_update;
80             %Planner
81             [car.time, car.steer, car.speed] = planner;
82             test_data.cartime = [test_data.cartime, car.time];
83             if(car.status == 1)
84                 %Car is at waypoint
85                 play(player)
86                 disp('WAYPOINT')
87             elseif(car.status == 2)
88                 %Car is at destination
89                 play(player)
90                 disp('At destination!')
91                 EPOCommunications('transmit','A1');
92                 stop(timerObj);
93             end
94
95             drive_car(car.speed, car.steer, car.time);
96             state = States.Sample;
97
98         case States.Sample

```

```

97     disp('Sampling after straight');
98     %Sample
99     %TDOA
100    TDOA_data = TDOA;
101    disp('TDOA afgerond');
102    test_data.TDOA = [test_data.TDOA ,TDOA_data];
103    %Localize
104
105    pass = localize_5ch(TDOA_data, ((100/1.3) * car.
        time) + (15 * drive_counter) + (50 *
        middle_counter));
106    %pass = 1;
107    if(pass ~= 8)
108        middle_counter = 0;
109    end
110    test_data.pass = [test_data.pass; pass];
111
112    if(pass == 1)
113        EPO4figure.setKITT([ position(1,end)/100
            position(2,end)/100], 0); %Update car
            position
114        state = States.Drive;
115        Orientation; %function without nonglobal
            arguments
116    else
117        state = States.Sample;
118
119        fail_factor = fail_factor + 1;
120        if(pass == 8)
121            %Too close to middle, drive straight.\
122            fail_factor = 0;
123            middle_counter = middle_counter + 1;
124            drive_car(car.speed, car.steer_straight,
                0.7);
125        end
126        if(fail_factor >= 3)
127
128            fail_factor = 0;
129            if(drive_counter >= 3)
130                switch(pass)
131                    case 2
132                        disp('Case 2');
133                        %This will hopefully never
                            happen. In
134                        %this case just throw an error.
                            Maybe

```

```

135         %later drive backwards and
136         reset position?
137         error('Car outside microphone
138         range');
139     case 3
140         disp('Case 3');
141         drive_counter = 100;
142     case 4
143         drive_counter = drive_counter +
144         1;
145         drive_car(car.speed, car.
146         steer_straight, 0.2);
147         if(drive_counter > 10)
148             car.did_last_turn = true;
149             localize_5ch(TDOA_data,
150             1000);
151             car.did_last_turn = false;
152         end
153     case 5
154         disp('Case 5');
155
156     case 6
157         disp('Case 6');
158         localize_5ch(TDOA_data, 1000);
159         state = States.Drive;
160         Orientation; %function without
161         nonglobal arguments
162     case 7
163         disp('Case 7');
164         localize_5ch(TDOA_data, 1000);
165         state = States.Drive;
166         Orientation; %function without
167         nonglobal arguments
168
169     end
170 else
171     drive_counter = drive_counter + 1;
172     drive_car(car.speed, car.steer_straight
173     , 0.2);
174
175 end
176
177 end
178
179 end
180
181 end
182
183 end
184
185 end
186
187 end
188
189 end
190
191 end
192
193 end
194
195 end
196
197 end
198
199 end
200
201 end
202
203 end
204
205 end
206
207 end
208
209 end
210
211 end
212
213 end
214
215 end
216
217 end
218
219 end
220
221 end
222
223 end
224
225 end
226
227 end
228
229 end
230
231 end
232
233 end
234
235 end
236
237 end
238
239 end
240
241 end
242
243 end
244
245 end
246
247 end
248
249 end
250
251 end
252
253 end
254
255 end
256
257 end
258
259 end
260
261 end
262
263 end
264
265 end
266
267 end
268
269 end
270
271 end
272
273 end
274
275 end
276
277 end
278
279 end
280
281 end
282
283 end
284
285 end
286
287 end
288
289 end
290
291 end
292
293 end
294
295 end
296
297 end
298
299 end
300
301 end
302
303 end
304
305 end
306
307 end
308
309 end
310
311 end
312
313 end
314
315 end
316
317 end
318
319 end
320
321 end
322
323 end
324
325 end
326
327 end
328
329 end
330
331 end
332
333 end
334
335 end
336
337 end
338
339 end
340
341 end
342
343 end
344
345 end
346
347 end
348
349 end
350
351 end
352
353 end
354
355 end
356
357 end
358
359 end
360
361 end
362
363 end
364
365 end
366
367 end
368
369 end
370
371 end
372
373 end
374
375 end
376
377 end
378
379 end
380
381 end
382
383 end
384
385 end
386
387 end
388
389 end
390
391 end
392
393 end
394
395 end
396
397 end
398
399 end
400
401 end
402
403 end
404
405 end
406
407 end
408
409 end
410
411 end
412
413 end
414
415 end
416
417 end
418
419 end
420
421 end
422
423 end
424
425 end
426
427 end
428
429 end
430
431 end
432
433 end
434
435 end
436
437 end
438
439 end
440
441 end
442
443 end
444
445 end
446
447 end
448
449 end
450
451 end
452
453 end
454
455 end
456
457 end
458
459 end
460
461 end
462
463 end
464
465 end
466
467 end
468
469 end
470
471 end
472
473 end
474
475 end
476
477 end
478
479 end
480
481 end
482
483 end
484
485 end
486
487 end
488
489 end
490
491 end
492
493 end
494
495 end
496
497 end
498
499 end
500
501 end
502
503 end
504
505 end
506
507 end
508
509 end
510
511 end
512
513 end
514
515 end
516
517 end
518
519 end
520
521 end
522
523 end
524
525 end
526
527 end
528
529 end
530
531 end
532
533 end
534
535 end
536
537 end
538
539 end
540
541 end
542
543 end
544
545 end
546
547 end
548
549 end
550
551 end
552
553 end
554
555 end
556
557 end
558
559 end
560
561 end
562
563 end
564
565 end
566
567 end
568
569 end
570
571 end
572
573 end
574
575 end
576
577 end
578
579 end
580
581 end
582
583 end
584
585 end
586
587 end
588
589 end
590
591 end
592
593 end
594
595 end
596
597 end
598
599 end
600
601 end
602
603 end
604
605 end
606
607 end
608
609 end
610
611 end
612
613 end
614
615 end
616
617 end
618
619 end
620
621 end
622
623 end
624
625 end
626
627 end
628
629 end
630
631 end
632
633 end
634
635 end
636
637 end
638
639 end
640
641 end
642
643 end
644
645 end
646
647 end
648
649 end
650
651 end
652
653 end
654
655 end
656
657 end
658
659 end
660
661 end
662
663 end
664
665 end
666
667 end
668
669 end
670
671 end
672
673 end
674
675 end
676
677 end
678
679 end
680
681 end
682
683 end
684
685 end
686
687 end
688
689 end
690
691 end
692
693 end
694
695 end
696
697 end
698
699 end
700
701 end
702
703 end
704
705 end
706
707 end
708
709 end
710
711 end
712
713 end
714
715 end
716
717 end
718
719 end
720
721 end
722
723 end
724
725 end
726
727 end
728
729 end
730
731 end
732
733 end
734
735 end
736
737 end
738
739 end
740
741 end
742
743 end
744
745 end
746
747 end
748
749 end
750
751 end
752
753 end
754
755 end
756
757 end
758
759 end
760
761 end
762
763 end
764
765 end
766
767 end
768
769 end
770
771 end
772
773 end
774
775 end
776
777 end
778
779 end
780
781 end
782
783 end
784
785 end
786
787 end
788
789 end
790
791 end
792
793 end
794
795 end
796
797 end
798
799 end
800
801 end
802
803 end
804
805 end
806
807 end
808
809 end
810
811 end
812
813 end
814
815 end
816
817 end
818
819 end
820
821 end
822
823 end
824
825 end
826
827 end
828
829 end
830
831 end
832
833 end
834
835 end
836
837 end
838
839 end
840
841 end
842
843 end
844
845 end
846
847 end
848
849 end
850
851 end
852
853 end
854
855 end
856
857 end
858
859 end
860
861 end
862
863 end
864
865 end
866
867 end
868
869 end
870
871 end
872
873 end
874
875 end
876
877 end
878
879 end
880
881 end
882
883 end
884
885 end
886
887 end
888
889 end
890
891 end
892
893 end
894
895 end
896
897 end
898
899 end
900
901 end
902
903 end
904
905 end
906
907 end
908
909 end
910
911 end
912
913 end
914
915 end
916
917 end
918
919 end
920
921 end
922
923 end
924
925 end
926
927 end
928
929 end
930
931 end
932
933 end
934
935 end
936
937 end
938
939 end
940
941 end
942
943 end
944
945 end
946
947 end
948
949 end
950
951 end
952
953 end
954
955 end
956
957 end
958
959 end
960
961 end
962
963 end
964
965 end
966
967 end
968
969 end
970
971 end
972
973 end
974
975 end
976
977 end
978
979 end
980
981 end
982
983 end
984
985 end
986
987 end
988
989 end
990
991 end
992
993 end
994
995 end
996
997 end
998
999 end
1000
1001 end
1002
1003 end
1004
1005 end
1006
1007 end
1008
1009 end
1010
1011 end
1012
1013 end
1014
1015 end
1016
1017 end
1018
1019 end
1020
1021 end
1022
1023 end
1024
1025 end
1026
1027 end
1028
1029 end
1030
1031 end
1032
1033 end
1034
1035 end
1036
1037 end
1038
1039 end
1040
1041 end
1042
1043 end
1044
1045 end
1046
1047 end
1048
1049 end
1050
1051 end
1052
1053 end
1054
1055 end
1056
1057 end
1058
1059 end
1060
1061 end
1062
1063 end
1064
1065 end
1066
1067 end
1068
1069 end
1070
1071 end
1072
1073 end
1074
1075 end
1076
1077 end
1078
1079 end
1080
1081 end
1082
1083 end
1084
1085 end
1086
1087 end
1088
1089 end
1090
1091 end
1092
1093 end
1094
1095 end
1096
1097 end
1098
1099 end
1100
1101 end
1102
1103 end
1104
1105 end
1106
1107 end
1108
1109 end
1110
1111 end
1112
1113 end
1114
1115 end
1116
1117 end
1118
1119 end
1120
1121 end
1122
1123 end
1124
1125 end
1126
1127 end
1128
1129 end
1130
1131 end
1132
1133 end
1134
1135 end
1136
1137 end
1138
1139 end
1140
1141 end
1142
1143 end
1144
1145 end
1146
1147 end
1148
1149 end
1150
1151 end
1152
1153 end
1154
1155 end
1156
1157 end
1158
1159 end
1160
1161 end
1162
1163 end
1164
1165 end
1166
1167 end
1168
1169 end
1170
1171 end
1172
1173 end
1174
1175 end
1176
1177 end
1178
1179 end
1180
1181 end
1182
1183 end
1184
1185 end
1186
1187 end
1188
1189 end
1190
1191 end
1192
1193 end
1194
1195 end
1196
1197 end
1198
1199 end
1200
1201 end
1202
1203 end
1204
1205 end
1206
1207 end
1208
1209 end
1210
1211 end
1212
1213 end
1214
1215 end
1216
1217 end
1218
1219 end
1220
1221 end
1222
1223 end
1224
1225 end
1226
1227 end
1228
1229 end
1230
1231 end
1232
1233 end
1234
1235 end
1236
1237 end
1238
1239 end
1240
1241 end
1242
1243 end
1244
1245 end
1246
1247 end
1248
1249 end
1250
1251 end
1252
1253 end
1254
1255 end
1256
1257 end
1258
1259 end
1260
1261 end
1262
1263 end
1264
1265 end
1266
1267 end
1268
1269 end
1270
1271 end
1272
1273 end
1274
1275 end
1276
1277 end
1278
1279 end
1280
1281 end
1282
1283 end
1284
1285 end
1286
1287 end
1288
1289 end
1290
1291 end
1292
1293 end
1294
1295 end
1296
1297 end
1298
1299 end
1300
1301 end
1302
1303 end
1304
1305 end
1306
1307 end
1308
1309 end
1310
1311 end
1312
1313 end
1314
1315 end
1316
1317 end
1318
1319 end
1320
1321 end
1322
1323 end
1324
1325 end
1326
1327 end
1328
1329 end
1330
1331 end
1332
1333 end
1334
1335 end
1336
1337 end
1338
1339 end
1340
1341 end
1342
1343 end
1344
1345 end
1346
1347 end
1348
1349 end
1350
1351 end
1352
1353 end
1354
1355 end
1356
1357 end
1358
1359 end
1360
1361 end
1362
1363 end
1364
1365 end
1366
1367 end
1368
1369 end
1370
1371 end
1372
1373 end
1374
1375 end
1376
1377 end
1378
1379 end
1380
1381 end
1382
1383 end
1384
1385 end
1386
1387 end
1388
1389 end
1390
1391 end
1392
1393 end
1394
1395 end
1396
1397 end
1398
1399 end
1400
1401 end
1402
1403 end
1404
1405 end
1406
1407 end
1408
1409 end
1410
1411 end
1412
1413 end
1414
1415 end
1416
1417 end
1418
1419 end
1420
1421 end
1422
1423 end
1424
1425 end
1426
1427 end
1428
1429 end
1430
1431 end
1432
1433 end
1434
1435 end
1436
1437 end
1438
1439 end
1440
1441 end
1442
1443 end
1444
1445 end
1446
1447 end
1448
1449 end
1450
1451 end
1452
1453 end
1454
1455 end
1456
1457 end
1458
1459 end
1460
1461 end
1462
1463 end
1464
1465 end
1466
1467 end
1468
1469 end
1470
1471 end
1472
1473 end
1474
1475 end
1476
1477 end
1478
1479 end
1480
1481 end
1482
1483 end
1484
1485 end
1486
1487 end
1488
1489 end
1490
1491 end
1492
1493 end
1494
1495 end
1496
1497 end
1498
1499 end
1500
1501 end
1502
1503 end
1504
1505 end
1506
1507 end
1508
1509 end
1510
1511 end
1512
1513 end
1514
1515 end
1516
1517 end
1518
1519 end
1520
1521 end
1522
1523 end
1524
1525 end
1526
1527 end
1528
1529 end
1530
1531 end
1532
1533 end
1534
1535 end
1536
1537 end
1538
1539 end
1540
1541 end
1542
1543 end
1544
1545 end
1546
1547 end
1548
1549 end
1550
1551 end
1552
1553 end
1554
1555 end
1556
1557 end
1558
1559 end
1560
1561 end
1562
1563 end
1564
1565 end
1566
1567 end
1568
1569 end
1570
1571 end
1572
1573 end
1574
1575 end
1576
1577 end
1578
1579 end
1580
1581 end
1582
1583 end
1584
1585 end
1586
1587 end
1588
1589 end
1590
1591 end
1592
1593 end
1594
1595 end
1596
1597 end
1598
1599 end
1600
1601 end
1602
1603 end
1604
1605 end
1606
1607 end
1608
1609 end
1610
1611 end
1612
1613 end
1614
1615 end
1616
1617 end
1618
1619 end
1620
1621 end
1622
1623 end
1624
1625 end
1626
1627 end
1628
1629 end
1630
1631 end
1632
1633 end
1634
1635 end
1636
1637 end
1638
1639 end
1640
1641 end
1642
1643 end
1644
1645 end
1646
1647 end
1648
1649 end
1650
1651 end
1652
1653 end
1654
1655 end
1656
1657 end
1658
1659 end
1660
1661 end
1662
1663 end
1664
1665 end
1666
1667 end
1668
1669 end
1670
1671 end
1672
1673 end
1674
1675 end
1676
1677 end
1678
1679 end
1680
1681 end
1682
1683 end
1684
1685 end
1686
1687 end
1688
1689 end
1690
1691 end
1692
1693 end
1694
1695 end
1696
1697 end
1698
1699 end
1700
1701 end
1702
1703 end
1704
1705 end
1706
1707 end
1708
1709 end
1710
1711 end
1712
1713 end
1714
1715 end
1716
1717 end
1718
1719 end
1720
1721 end
1722
1723 end
1724
1725 end
1726
1727 end
1728
1729 end
1730
1731 end
1732
1733 end
1734
1735 end
1736
1737 end
1738
1739 end
1740
1741 end
1742
1743 end
1744
1745 end
1746
1747 end
1748
1749 end
1750
1751 end
1752
1753 end
1754
1755 end
1756
1757 end
1758
1759 end
1760
1761 end
1762
1763 end
1764
1765 end
1766
1767 end
1768
1769 end
1770
1771 end
1772
1773 end
1774
1775 end
1776
1777 end
1778
1779 end
1780
1781 end
1782
1783 end
1784
1785 end
1786
1787 end
1788
1789 end
1790
1791 end
1792
1793 end
1794
1795 end
1796
1797 end
1798
1799 end
1800
1801 end
1802
1803 end
1804
1805 end
1806
1807 end
1808
1809 end
1810
1811 end
1812
1813 end
1814
1815 end
1816
1817 end
1818
1819 end
1820
1821 end
1822
1823 end
1824
1825 end
1826
1827 end
1828
1829 end
1830
1831 end
1832
1833 end
1834
1835 end
1836
1837 end
1838
1839 end
1840
1841 end
1842
1843 end
1844
1845 end
1846
1847 end
1848
1849 end
1850
1851 end
1852
1853 end
1854
1855 end
1856
1857 end
1858
1859 end
1860
1861 end
1862
1863 end
1864
1865 end
1866
1867 end
1868
1869 end
1870
1871 end
1872
1873 end
1874
1875 end
1876
1877 end
1878
1879 end
1880
1881 end
1882
1883 end
1884
1885 end
1886
1887 end
1888
1889 end
1890
1891 end
1892
1893 end
1894
1895 end
1896
1897 end
1898
1899 end
1900
1901 end
1902
1903 end
1904
1905 end
1906
1907 end
1908
1909 end
1910
1911 end
1912
1913 end
1914
1915 end
1916
1917 end
1918
1919 end
1920
1921 end
1922
1923 end
1924
1925 end
1926
1927 end
1928
1929 end
1930
1931 end
1932
1933 end
1934
1935 end
1936
1937 end
1938
1939 end
1940
1941 end
1942
1943 end
1944
1945 end
1946
1947 end
1948
1949 end
1950
1951 end
1952
1953 end
1954
1955 end
1956
1957 end
1958
1959 end
1960
1961 end
1962
1963 end
1964
1965 end
1966
1967 end
1968
1969 end
1970
1971 end
1972
1973 end
1974
1975 end
1976
1977 end
1978
1979 end
1980
1981 end
1982
1983 end
1984
1985 end
1986
1987 end
1988
1989 end
1990
1991 end
1992
1993 end
1994
1995 end
1996
1997 end
1998
1999 end
2000
2001 end
2002
2003 end
2004
2005 end
2006
2007 end
2008
2009 end
2010
2011 end
2012
2013 end
2014
2015 end
2016
2017 end
2018
2019 end
2020
2021 end
2022
2023 end
2024
2025 end
2026
2027 end
2028
2029 end
2030
2031 end
2032
2033 end
2034
2035 end
2036
2037 end
2038
2039 end
2040
2041 end
2042
2043 end
2044
2045 end
2046
2047 end
2048
2049 end
2050
2051 end
2052
2053 end
2054
2055 end
2056
2057 end
2058
2059 end
2060
2061 end
2062
2063 end
2064
2065 end
2066
2067 end
2068
2069 end
2070
2071 end
2072
2073 end
2074
2075 end
2076
2077 end
2078
2079 end
2080
2081 end
2082
2083 end
2084
2085 end
2086
2087 end
2088
2089 end
2090
2091 end
2092
2093 end
2094
2095 end
2096
2097 end
2098
2099 end
2100
2101 end
2102
2103 end
2104
2105 end
2106
2107 end
2108
2109 end
2110
2111 end
2112
2113 end
2114
2115 end
2116
2117 end
2118
2119 end
2120
2121 end
2122
2123 end
2124
2125 end
2126
2127 end
2128
2129 end
2130
2131 end
2132
2133 end
2134
2135 end
2136
2137 end
2138
2139 end
2140
2141 end
2142
2143 end
2144
2145 end
2146
2147 end
2148
2149 end
2150
2151 end
2152
2153 end
2154
2155 end
2156
2157 end
2158
2159 end
2160
2161 end
2162
2163 end
2164
2165 end
2166
2167 end
2168
2169 end
2170
2171 end
2172
2173 end
2174
2175 end
2176
2177 end
2178
2179 end
2180
2181 end
2182
2183 end
2184
2185 end
2186
2187 end
2188
2189 end
2190
2191 end
2192
2193 end
2194
2195 end
2196
2197 end
2198
2199 end
2200
2201 end
2202
2203 end
2204
2205 end
2206
2207 end
2208
2209 end
2210
2211 end
2212
2213 end
2214
2215 end
2216
2217 end
2218
2219 end
2220
2221 end
2222
2223 end
2224
2225 end
2226
2227 end
2228
2229 end
2230
2231 end
2232
2233 end
2234
2235 end
2236
2237 end
2238
2239 end
2240
2241 end
2242
2243 end
2244
2245 end
2246
2247 end
2248
2249 end
2250
2251 end
2252
2253 end
2254
2255 end
2256
2257 end
2258
2259 end
2260
2261 end
2262
2263 end
2264
2265 end
2266
2267 end
2268
2269 end
2270
2271 end
2272
2273 end
2274
2275 end
2276
2277 end
2278
2279 end
2280
2281 end
2282
2283 end
2284
2285 end
2286
2287 end
2288
2289 end
2290
2291 end
2292
2293 end
2294
2295 end
2296
2297 end
2298
2299 end
2300
2301 end
2302
2303 end
2304
2305 end
2306
2307 end
2308
2309 end
2310
2311 end
2312
2313 end
2314
2315 end
2316
2317 end
2318
2319 end
2320
2321 end
2322
2323 end
2324
2325 end
2326
2327 end
2328
2329 end
2330
2331 end
2332
2333 end
2334
2335 end
2336
2337 end
2338
2339 end
2340
2341 end
2342
2343 end
2344
2345 end
2346
2347 end
2348
2349 end
2350
2351 end
2352
2353 end
2354
2355 end
2356
2357 end
2358
2359 end
2360
2361 end
2362
2363 end
2364
2365 end
2366
2367 end
2368
2369 end
2370
2371 end
2372
2373 end
2374
2375 end
2376
2377 end
2378
2379 end
2380
2381 end
2382
2383 end
2384
2385 end
2386
2387 end
2388
2389 end
2390
2391 end
2392
2393 end
2394
2395 end
2396
2397 end
2398
2399 end
2400
2401 end
2402
2403 end
2404
2405 end
2406
2407 end
2408
2409 end
2410
2411 end
2412
2413 end
2414
2415 end
2416
2417 end
2418
2419 end
2420
2421 end
2422
2423 end
2424
2425 end
2426
2427 end
2428
2429 end
2430
2431 end
2432
2433 end
2434
2435 end
2436
2437 end
2438
2439 end
2440
2441 end
2442
2443 end
2444
2445 end
2446
2447 end
2448
2449 end
2450
2451 end
2452
2453 end
2454
2455 end
2456
2457 end
2458
2459 end
2460
2461 end
2462
2463 end
2464
2465 end
2466
2467 end
2468
2469 end
2470
2471 end
2472
2473 end
2474
2475 end
2476
2477 end
2478
2479 end
2480
2481 end
2482
2483 end
2484
2485 end
2486
2487 end
2488
2489 end
2490
2491 end
2492
2493 end
2494
2495 end
249
```

```

174     function timer_stopFcn(timerObj, timerEvent)
175         disp('Timer is gestopt');
176         delete(timerObj);
177     end
178     function timer_error
179         disp('Error');
180         drive(150, 150);
181         EPOCommunications('transmit', 'A1');
182     end
183 end

```

A.3.2 drive_car.m

A.3.3 states.m

```

1 classdef States
2     enumeration
3         VoltageMeasure, Drive, Sample
4     end
5 end

```

A.3.4 status_update.m

```

1 function status_update
2 global car;
3 inc_data = EPOCommunications('transmit', 'S');
4 raw = strsplit(inc_data, {'D', 'U', 'A', 'udio ', '\n', ' '});
5 data = str2double(raw);
6 car.LeftSensor = data(4);
7 car.RightSensor = data(5);
8 car.voltage = data(6) / 1000;
9 car.AudioStatus = data(7);
10 end

```

A.4 Final Challenge