FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Local Positioning System for Quadcopters

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Andreas Brandstätter, B.Sc.
Matrikelnummer 0927824

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Mitwirkung: Univ.Ass. Dipl.-Ing. Bernhard Frömel, B.Sc.
Univ.Ass. Dipl.-Ing. Christian Hirsch, B.Sc.

Wien, 22. April 2019

Andreas Brandstätter   Radu Grosu

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Local Positioning System for Quadcopters

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Andreas Brandstätter, B.Sc.
Registration Number 0927824

to the Faculty of Informatics

at the TU Wien

Advisor:	Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Assistance:	Univ.Ass. Dipl.-Ing. Bernhard Frömel, B.Sc.
	Univ.Ass. Dipl.-Ing. Christian Hirsch, B.Sc.

Vienna, 22nd April, 2019

Andreas Brandstätter

Radu Grosu

Erklärung zur Verfassung der Arbeit

Andreas Brandstätter, B.Sc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 22. April 2019

Andreas Brandstätter

Danksagung

Ich möchte mich bei all jenen bedanken, die mich während der Ausarbeitung meiner Masterarbeit unterstützt haben.

Vor allem danke ich Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu für die Betreuung und Begutachtung dieser Arbeit. Für die zahlreichen Anregungen zu Verbesserungen und die konstruktive Kritik möchte Dipl.-Ing. Bernhard Frömel, B.Sc. und Dipl.-Ing. Christian Hirsch, B.Sc. danken. Mein besonderer Dank gilt dem Forschungsbereich Cyber-Physical Systems des Instituts für Computer Engineering der TU Wien für die Bereitstellung der Hardware für die praktische Implementierung dieser Arbeit.

Für die Möglichkeit der Teilnahme an der RobotChallenge 2017 in Peking, China danke ich Dr. Roland Stelzer und Karim Jafarmadar, B.Sc. von der Österreichischen Gesellschaft für innovative Computerwissenschaften (INNOC). Weiters danke ich der Freiwilligen Feuerwehr Sieghartskirchen für die Zurverfügungstellung von Räumlichkeiten für Testaufbauten.

Meinen Freunden möchte ich für die stetige Motivation - insbesondere in der finalen Phase der Masterarbeit - danken.

Abschließend bedanke ich mich besonders bei meinen Eltern, die mir durch ihre jederzeitige Unterstützung mein Studium ermöglicht haben.

Acknowledgements

My gratitude goes to all who have supported me throughout the work for this thesis.

With a special mention to Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu for supervising this thesis. For the numerous suggestions how to improve the work and constructive criticism I am grateful to Dipl.-Ing. Bernhard Frömel, B.Sc. and Dipl.-Ing. Christian Hirsch, B.Sc. A special gratitude goes to the Research Division Cyber-Physical Systems at the Institute of Computer Engineering at the TU Wien for providing the hardware to practically implement this work.

For the opportunity to take part at the RobotChallenge 2017 in Bejing, China I am grateful to Dr. Roland Stelzer and Karim Jafarmadar, B.Sc. from Austrian Society for Innovative Computer Sciences (INNOC). Furthermore I am grateful to the volunteer fire department of Sieghartskirchen for providing an indoor location as test environment.

A gratitude goes to my friends for their great motivation - especially during the final phase of the thesis.

Last but by no means least, I am very grateful to my parents who continuously supported me throughout my studies.

Kurzfassung

Der Bereich unbemannter Luftfahrzeuge (uLFZ), welche umgangssprachlich auch als Drohnen bezeichnet werden, entwickelte sich in jüngster Vergangenheit rasch weiter. Quadrokopter und andere Arten von uLFZ wurden bedeutend kleiner, leistungsstärker und kostengünstiger. Aufgrund der Verfügbarkeit in verschiedensten Qualitätsklassen von Unterhaltungselektronik bis hin zu professionellen, industriellen und der Forschung dienenden Anwendungen, werden diese uLFZ nicht nur im Außenbereich, sondern auch für eine steigende Anzahl an Anwendungen in Innenräumen eingesetzt.

uLFZ können entweder ausschließlich manuell gesteuert werden, oder technische Unterstützungssysteme, wie etwa die Aufzeichnung des Flugpfades oder Autopilot-Funktionen, bereitstellen. Für diese Arten der Navigation in Innenräumen sind präzise, rasch verfügbare und verlässliche Positionsdaten wesentlich. Diese können durch ein lokales Positionssystem ermittelt werden. In der vorliegenden Arbeit wird daher die folgende Problemstellung behandelt: Gegeben sei ein begrenzter dreidimensionaler Raum, in welchem die Position eines uLFZ mit einer definierten Genauigkeit festgestellt werden soll.

In dieser Arbeit wird ein solches Lokalisierungssystem auf Basis von Ultraschallsignalen implementiert. Auf dem uLFZ wird ein Sender platziert, welcher Ultraschallsignale aussendet. Im Raum, in welchem das uLFZ betrieben werden soll, werden mehrere ortfeste Empfänger platziert, welche die ausgesendeten Ultraschallsignale empfangen. Abhängig von der Ausbreitungsgeschwindigkeit des Signals und der Position des uLFZ wird das Signal von jedem Empfänger zu unterschiedlichen Zeitpunkten empfangen. Basierend auf den Unterschieden dieser Zeitpunkte und der bekannten Ausbreitungsgeschwindigkeit des Signals, welche im Fall von Ultraschallsignalen die Schallgeschwindigkeit ist, wird die Position des uLFZ berechnet.

Die durchgeführten Tests zeigen, dass mit dem implementierten System die Position eines uLFZ in Innenräumen erfolgreich festgestellt werden kann. Die Positionsbestimmung erfolgt mit einer Frequenz von $14Hz$ und liefert für statische Postitionen eine Standardabweichung von 1.9 bis $4.9cm$ in horizontaler und 5.8 bis $13.4cm$ in vertikaler Richtung. Für bewegte Objekte erhöht sich diese auf 6.0 bis $7.3cm$ in horizontaler Richtung. In den Tests konnte die Tauglichkeit des implementierten Systems für die Verwendung in einem Autopilot-System gezeigt werden. Der Quadrokopter kann damit selbständig einem Pfad, bestehend aus vordefinierten Wegpunkten, folgen.

Abstract

In recent times there has been a huge development in the field of Unmanned Aerial Vehicles(UAVs). Quadcopters and other types of UAVs have become smaller, more powerful and cheaper. Given the availability of UAVs in a wide range of quality classes from consumer market to professional, industrial, and research applications they are not only operated outdoors but also used for an increasing number of indoor applications.

UAVs can either be operated completely manually or employ assistive technologies like flight path recording or autopilot systems. For these types of indoor navigation it is essential to have accurate, timely and reliable location data which can be determined by a Local Positioning System (LPS). It is necessary to achieve a sufficient accuracy and reliability for such a LPS to safely move UAVs without crashing and hitting any obstacles. In this thesis we are dealing with the following problem: Given a bounded 3-dimensional space (indoor location), the position of an UAV should be determined within a certain precision in this space.

In this work we implement a LPS based on ultrasonic signals. On the UAV a sender is placed which transmits ultrasonic signals. At the indoor location, where the UAV is operated, there are fixed receivers placed which receive the ultrasonic signals. Depending on the propagation speed of the signal and the position of the UAV the signal is received at different time instants by each receiver. Based on the differences of these time instants and the known propagation speed for the signal, which is the speed of sound in the case of ultrasonic signals, the position of the UAV is calculated.

The tests which we carried out, show that the implemented LPS is capable of determining the position of an UAV at an indoor location. The position is determined with a rate of $14Hz$. For static positions the standard deviation is 1.9 to $4.9cm$ in horizontal direction and 5.8 to $13.4cm$ in vertical direction. For a moving object the standard deviation is 6.0 to $7.3cm$ in horizontal direction. Our tests demonstrate the fitness of the implemented LPS to be used in an autopilot setup for a quadcopter. The quadcopter is able to follow a path consisting of pre-defined waypoints.

Contents

CHAPTER 1

Introduction

In recent times there has been a huge development in the field of Unmanned Aerial Vehicles(UAVs). Quadcopters (helicopters that are propulsed by four rotors) and other types of UAVs have become smaller, more powerful and cheaper. They are available in a wide range of quality classes from consumer market to professional, industrial, and research applications.

1.1 Motivation

Given the availability of UAVs they are not only operated outdoors but also used for an increasing number of indoor applications. In this thesis we are looking at the problem of indoor localisation for UAVs.

1.1.1 Motivation and Problem Statement

For indoor navigation of UAVs it is essential to have accurate, timely and reliable location data. In contrast to Unmanned Ground Vehicles(UGVs) where it might be sufficient to have 2-dimensional data we need 3-dimensional location data for UAVs. It is necessary to achieve a sufficient accuracy and reliability for a Local Positioning System (LPS) to safely move UAVs or UGVs without crashing and hitting any obstacles. To catch this goal it is necessary to merge several sources of information and combine the position data in a reasonable way. One example for a navigational application is *guided learning* to teach human operators to pilot an UAV. Such an assistant defines a virtual 3-dimensional area where the UAV shall stay inside for all the time. In case of incorrect commands from a human operator the autopilot can take over and hold the current position to avoid crashes.

This motivation points out the demand for LPSs in the area of robotics. We therefore describe the problem of a LPS as follows: Given a bounded 3-dimensional space **V**, the position **M** of an UAV should be determined with a certain precision in this space **V** (see Figure 1.1). We can assume to have an indoor location (lecture hall, seminar room, gym, etc.) where fixed components of the LPS can be placed. Furthermore, we have an UAV that can be equipped with active or passive components of the LPS on board.

Let **V** denote the 3-dimensional space

$$\mathbf{V} = \left\{ \begin{pmatrix} x \\ y \\ z \end{pmatrix} : x, y, z \in \mathbb{R} \right\}$$

bounded by limits

$$x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}, z_{min} \leq z \leq z_{max}$$

then the position **M** of an UAV

$$\mathbf{M} = \begin{pmatrix} x_{UAV} \\ y_{UAV} \\ z_{UAV} \end{pmatrix}, \mathbf{M} \in \mathbf{V}$$

should be determined by the LPS.

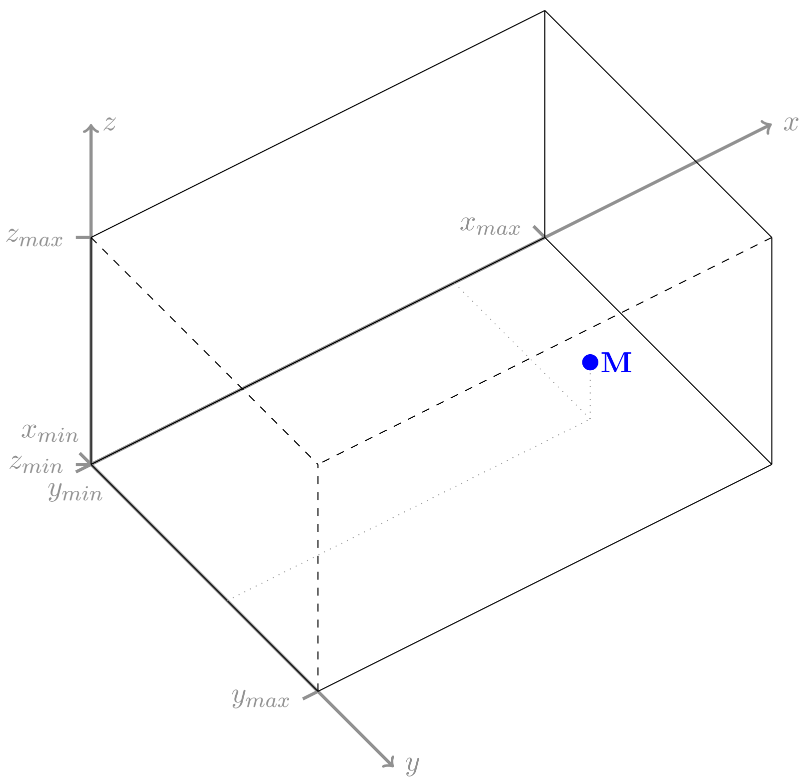


Figure 1.1: Given the 3-dimensional space **V** bounded by limits $x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}, z_{min} \leq z \leq z_{max}$, the position **M** of an UAV should be determined by the LPS.

1.1.2 Aim of the Work

The aim of this thesis is to compare existing indoor positioning systems. Based on this research, we design and implement a LPS which is able to detect the absolute position of an UAV in indoor environments. The determined position of an UAV should also take the relative position information, derived from on-board Inertial Measurement Units(IMUs), into account. This requires means of sensor fusion and appropriate transformation of position data with respect to their coordinate system.

In a first stage the system should record the trace of an UAV's flight. This will show a proof of concept of the developed system and can be used to check the accuracy and precision of the system. The second stage of this work should make it possible to set waypoints for the UAV to fly autonomously. More general the UAV should be able to follow a specified path or hold its position without any human interaction.

1.2 Methodology and Outline

First we are going to carry out a literature study of existing positioning systems. We will focus on LPSs but not limit our research in order to also incorporate methods currently not used in LPSs. Then we will specify the requirements for the design of our LPS. This includes accuracy and operating distance. Given these requirements we will analyse the types of systems which were identified in the literature study. These systems will be judged by their fitness to fulfil the specified requirements. Based on this analysis we will decide which system (or which combination of systems) we will use in this work.

Given the selected system we will formalize the mathematical model in order to carry out simulations in the next step. These simulations will help to check if the requirements can be satisfied by our specific design decisions for the implementation of the LPS. If there are any problems identified in this stage the design decisions should be revised and the simulations will be carried out again. Then, parts of the system are implemented in hard- and software and tests with these components are conducted to check against the simulated properties. In case of differences between simulation results and real world tests adjustments to the simulation and/or implementation are applied. Furthermore, simulations will be parametrized and refined by the real world test results.

Based on the updated simulation and the component-wise verified parts the whole LPS will be assembled. Simulations of the whole system will be carried out as well. The whole system will then be used to track a manually operated quadcopter. As soon as the system is verified to be reliable it will be used to implement an autopilot system for a quadcopter.

CHAPTER 2

State of the Art

First, we will give an overview of basic concepts how to determine the position of a mobile object. Then, we carry out a literature survey of LPSs. We characterize these LPSs by the type of the system and the used calculation method. We are also looking at the accuracy that these systems provide.

2.1 Basic Concepts

In this section we will describe the basic concepts how to determine the position of a mobile object. First, information about the mobile object is measured (e.g., time, angle, signal strength). This information is propagated by different types of transmission medium. With the measurements, the position can be inferred by different calculation methods.

2.1.1 Transmission Medium

Since every LPS needs to transmit information from fixed reference points to a mobile object or vice-versa a system can be categorized its transmission medium.

Radio Signals

For radio signal based systems there are either senders on the mobile object and receivers on fixed positions or the other way around. It may also be the case that both types are placed on the mobile object in case of bidirectional communication. Since mobile robots in the most cases require wireless communication it can be beneficial to use the same radio signal for information transmission and to determine the location.

Sound Signals

Another option is the usage of sound signals instead of radio signals. The principle of these systems is similar to those which use radio signals. The propagation speed of sound is significantly slower than the speed of radio signals. This makes it easier to use differences in propagation delays to calculate the position. While it is possible to use any frequency for the sound signal it is beneficial to use ultrasonic sound signals outside the range of audible frequencies for humans. Sound signals which are audible for humans can be distracting and a nuisance for people in the operating range of the system.

Optical Systems

A different approach is to use cameras. They can either be distributed in a room or placed on a mobile object. Additional active or passive markers placed on the flying object or within the operation area can help to improve the accuracy of such systems. The position is determined by image processing. For more details about optical system see Section 2.1.2.

2.1.2 Calculation Method

Using either radio, ultrasonic or optical signals it is possible to determine the location using various methods. These will be described in the following sections. In all of the following figures the mobile object is marked with m while P_i denote fixed receivers/senders with known location.

Time of Arrival (TOA)

The propagation speed v_{signal} for the used signal must be known. Furthermore there is a time synchronisation needed between the sender and receiver of the signal s.t. the propagation delay $\delta t = t_{receive} - t_{send}$ can be measured. From this delay the distance between sender and receiver $s = \delta t \cdot v_{signal}$ can be calculated. Any direction may be used for the signal: This means the sender can be placed on the mobile object and the receivers are placed within the operating area or the other way around.

The time synchronisation between the sender and receiver can be achieved using different methods:

- **Precise clocks**
  When the system is started the clocks of the mobile object and fixed receivers/senders are synchronized. The clock drift must be below a certain limit s.t. the accuracy of the system can be maintained for the specified operating time. An example for such precise clocks would be atomic clocks. E.g., frequency stability for *NAC1 Nano-Atomic-Clock* is $8 \cdot 10^{-12}$ [Pra+17].

- **Signals sent in both directions**
  Both sides (mobile object as well as fixed stations) are equipped with a sender and receiver. A signal is transmitted at time t_{send} by fixed stations and replied by the mobile object once it receives the signal at time $t_{receive} = t_{send} + \delta t$. At time $t_{replied} = t_{receive} + \delta t$ the signal is received at fixed receivers again. The delay $2 \cdot \delta t = t_{replied} - t_{send}$ can be measured.

- **Continuous clock synchronisation**
  The clocks of the mobile object and fixed receivers/senders are continuously synchronized by additional means. An additional radio frequency module can be used for this purpose.

For the 2-dimensional plane a measured timespan between one sender-receiver-pair determines a circle of known radius for possible positions of the mobile object. Using one more of those pairs the possible positions become the intersection of two circles which might be at most two possible locations. Using 3 pairs it is in theory possible to exactly determine the position of the object in the 2-dimensional plane. For 3-dimensional space one more sender-receiver-pair is needed. Figure 2.1 shows an example for Time of Arrival (TOA): The circles indicate the possible positions based on the measured propagation time for the respective sender-receiver-pair.

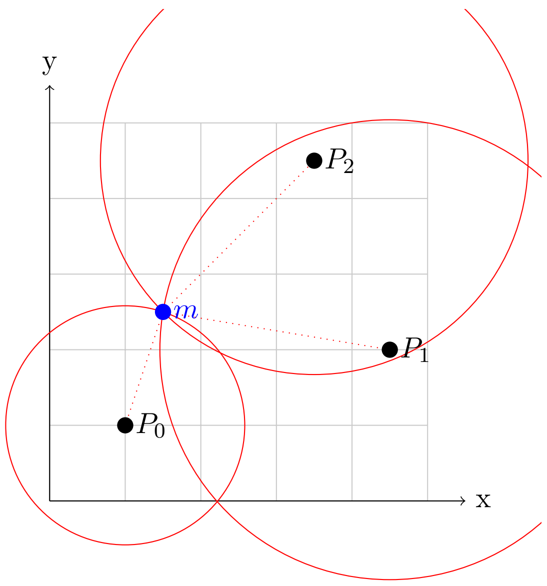


Figure 2.1: Example for TOA shown in 2-dimensional plane: Mobile object (m) and three reference points (P_0, P_1, P_2) with known location. Red circles indicate the possible positions based on the measured propagation time for the respective sender-receiver-pair ($m \leftrightarrow P_i$).

Time Difference of Arrival (TDOA)

This method is similar to TOA but there is no synchronisation with the mobile object needed. Again the direction of the signal can be in either direction, the sender can be placed on the mobile object and the receivers are placed within the operating area or the other way around. The fixed components must be synchronized, s.t. the time difference between the propagation delay for two sender-receiver-pairs can be measured. This means that t_{send} might be unknown, but $\delta t = t_{receive_i} - t_{receive_j}$ can be determined. From this difference in time and the known position of two fixed senders/receivers a hyperboloid with all possible positions of the mobile object can be calculated [Lee75]. For Time Difference of Arrival (TDOA) there are at least 3 fixed stations necessary for the 2-dimensional plane and 4 stations for the 3-dimensional space. Figure 2.2 shows an example for TDOA: The red hyperbolas (marked by $P_i \triangle P_j$) indicate the possible positions based on the measured time difference by two stations (P_i, P_j).

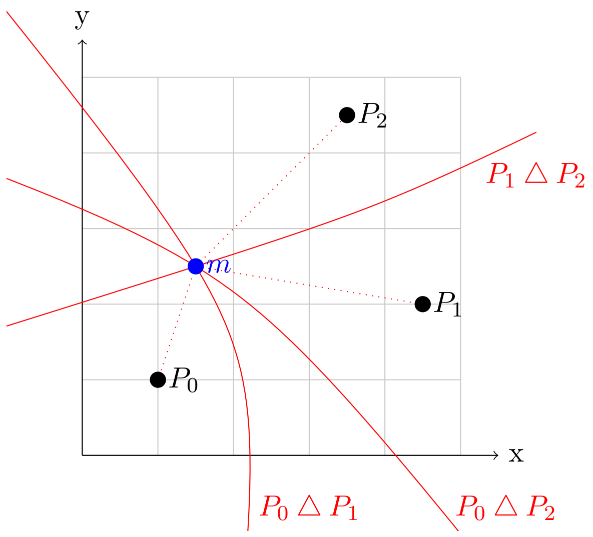


Figure 2.2: Example for TDOA shown in 2-dimensional plane: Mobile object (m) and three reference points (P_0, P_1, P_2) with known location. Red hyperbolas (marked by $P_i \triangle P_j$) indicate the possible positions based on the measured time difference by two stations (P_i, P_j).

Angle of Arrival (AOA)

To calculate the position using Angle of Arrival (AOA) the receivers must be able to determine the angle from where the signal was received. The position is then given by the intersection of two straight lines with these angles. The exact angle for the signal direction can be measured by rotating antennas as described by Gill and Hecht for aerial navigation [GH28]. Figure 2.3 shows an example for AOA with exact angle measurement: The red lines indicate the possible positions based on the measured angle for the signal.

Sector antenna arrays can be used for AOA estimation as described by Abdalla, Razavi-Ghods and Salous [ARS05]. The proposed antenna layout provides information of the signal direction within a certain angular interval. Figure 2.4 shows an example for AOA with sector antennas: Each reference point has marked 16 antenna sectors. The red areas indicate the possible positions based on the determined antenna sector.

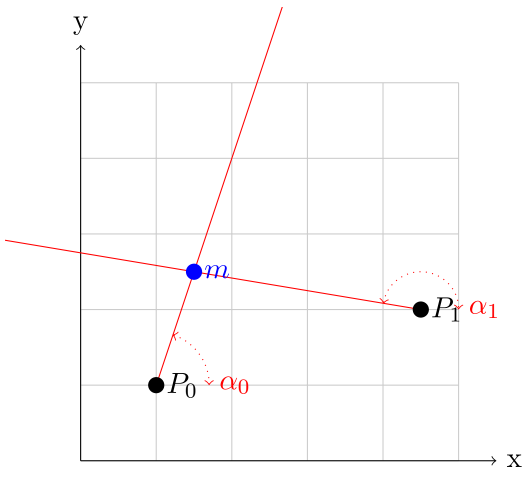


Figure 2.3: Example for AOA with exact angle measurement shown in 2-dimensional plane: Mobile object (m) and two reference points (P_0, P_1) with known location. Red lines indicate the possible positions based on the measured angles (α_0, α_1) for the signal.

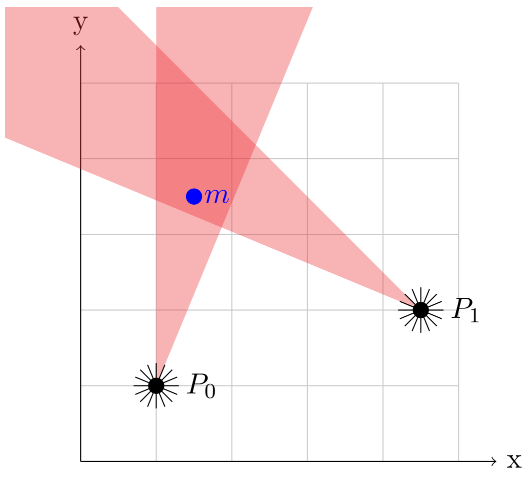


Figure 2.4: Example for AOA with sector antennas shown in 2-dimensional plane: Mobile object (m) and two reference points (P_0, P_1) with known location and 16 antenna sectors each. Red areas indicate the possible positions based on the determined antenna sector.

Received Signal Strength Indication (RSSI)

In free space propagation the received power level is inverse proportional to the squared distance of sender and receiver. If there is no disturbance (e.g., reflection or shading) the distance can be calculated if the transmitted and received power is known. If there are at least four receiver-transmitter pairs with known distance the position of the mobile object can be ultimately determined. Compared to free space propagation the accuracy and precision of the determined position might be significantly lower in real world environments. Figure 2.5 shows an example for Received Signal Strength Indication (RSSI) based localization: The red circles indicate the possible positions based on the measured signal power while the red regions indicate the signal level around the fixed receivers/senders.

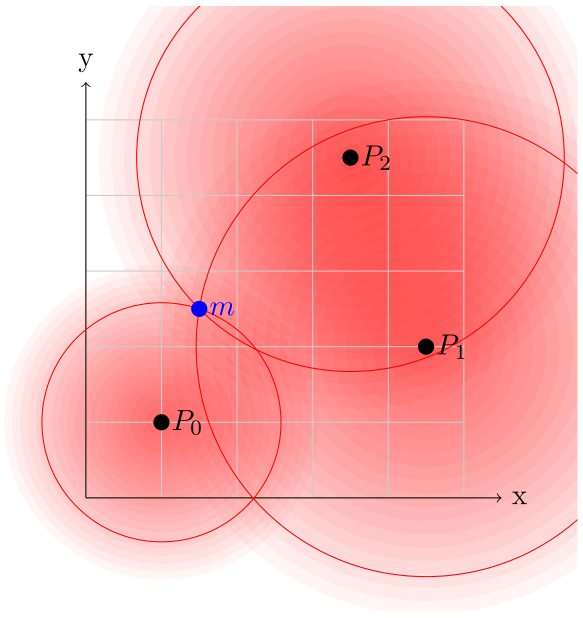


Figure 2.5: Example for RSSI based localization shown in 2-dimensional plane: Mobile object (m) and three reference points (P_0, P_1, P_2) with known location. Red circles indicate the possible positions based on the measured signal power. Red regions indicate the signal level around the reference points.

Location Fingerprinting

For RSSI based systems there might be problems if there are a lot of obstacles. In indoor locations this can be walls, furniture or other objects. Location fingerprinting provides a solution for this problem. Received power is not taken as direct measurement for the distance. Within the operation area a set of possible positions is defined and the respective signal power levels are recorded for all base stations at these positions. To determine the unknown position of a mobile object the current power levels for all stations are then matched against this pre-recorded reference levels. Location fingerprinting is not only applicable to signal power levels but also to other properties which can be recorded at all the possible positions. Figure 2.6 shows an example for location fingerprinting: The red dots indicate the reference positions with the recorded signal power for all base stations.

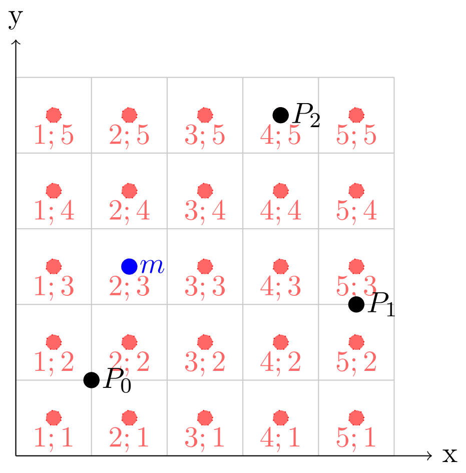


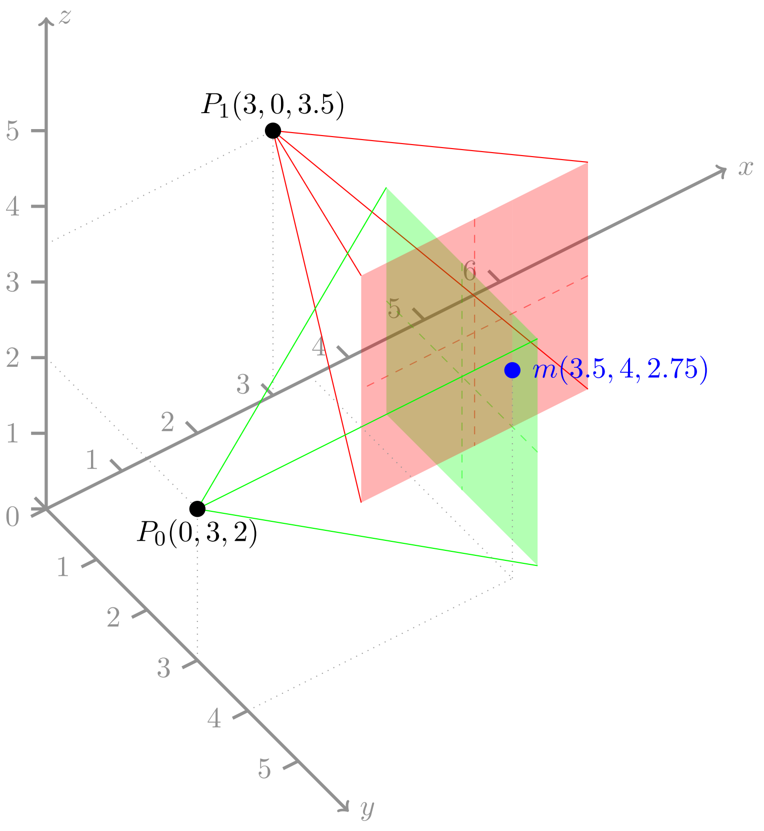
Figure 2.6: Example for location fingerprinting shown in 2-dimensional plane: Mobile object (m) and three reference points (P_0, P_1, P_2) with known location. Red dots indicate the reference positions with the recorded signal power for all base stations.

Fixed Camera Systems

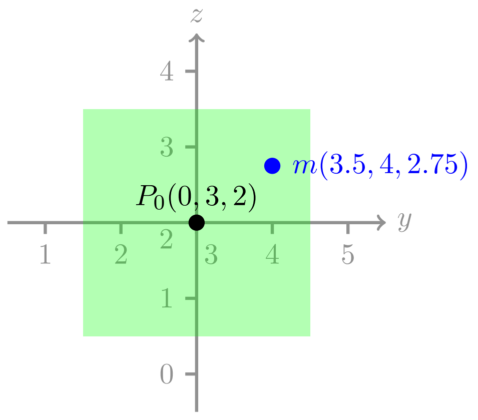
A different approach is the usage of fixed cameras which are distributed in space. Based on the images of two or more cameras the position of the mobile object is determined. Hannah [Han74] describes matching of a target area (this could be the mobile object for example) in a system of pictures from two cameras (stereo images). When the position of the mobile object is identified on the pictures the position within the three-dimensional space can be inferred. To enhance the system additional active or passive markers can be placed on the mobile object which make it easier identifying it on the camera pictures. Figure 2.7 shows an example of a fixed camera system: The red and green cones indicate the camera coverage. Red and green planes indicate the respective camera viewports.

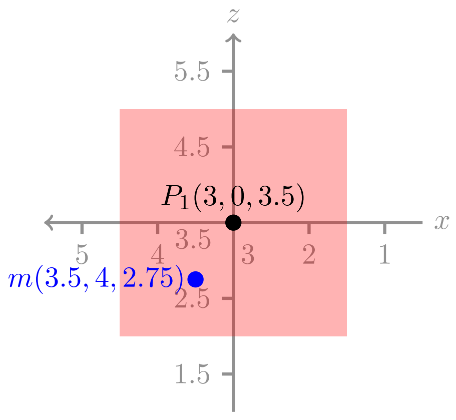
On-Board Camera Systems

It is also possible to mount a camera or several cameras on the mobile object. Image processing is used to calculate the position based on the images of the environment. For this method there might be active or passive optical markers distributed in space. These markers can be compared to reference images as described by Quam [Qua71] or by Ayache and Faugeras [AF86]. Given the known position of the markers and the identified location in the camera images the position of the mobile object can be inferred. Figure 2.8 shows an example of an on-board camera system: The blue cone indicates the camera coverage and the blue area indicates the camera viewport.



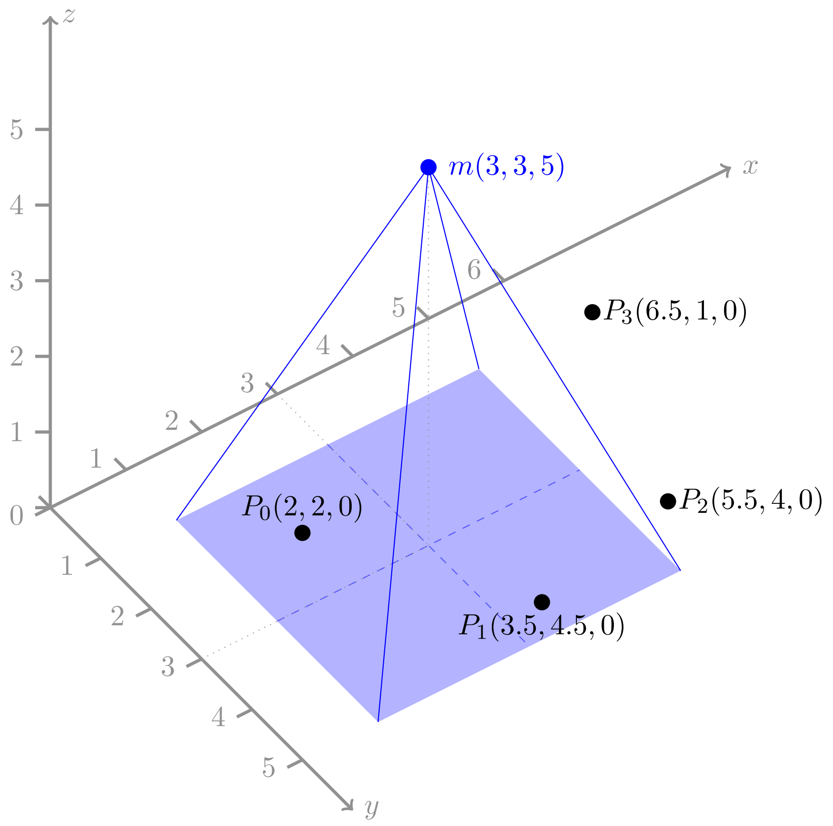
(a) Fixed camera system in 3-dimensional space.



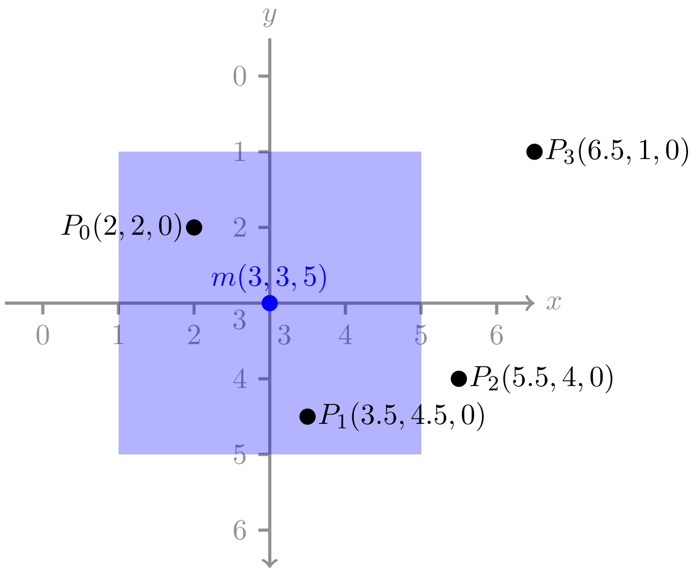


(b) Projection to the y-z-plane. **(c)** Projection to the x-z-plane.

Figure 2.7: Example for fixed camera system: Mobile object ($m(x, y, z)$) and two cameras ($P_0(x, y, z)$, $P_1(x, y, z)$) with known location. The red and green cones indicate the camera coverage. Red and green planes indicate the respective camera viewports.



(a) On-board camera system in 3-dimensional space.



(b) Projection to the y-z-plane.

Figure 2.8: Example for on-board camera system: Mobile object ($m(x, y, z)$) with downward facing camera and four fixed reference points ($P_0(x, y, z)$, $P_1(x, y, z)$, $P_2(x, y, z)$, $P_3(x, y, z)$) with known location. The blue cone indicates the camera coverage and the blue area indicates the camera viewport.

2.1.3 Relative Position Determination

IMUs and odometry are used in robotics to determine the relative position of mobile objects. These systems are not capable of providing an absolute position like LPS do. This means if the mobile object is placed at an unknown location m_0 these systems cannot determine the position without further information. Let us assume the mobile object is first placed at known location m_1 and then moved by vector $\vec{s}$ to the new location $m_2 = m_1 + \vec{s}$. Then these systems are able to determine the vector $\vec{s}$. Therefore the position m_2 can be inferred based on the previously known location m_1. IMUs and odometry can furthermore be used in combination with other systems to enhance precision and/or reliability of the overall position determination.

Inertial Measurement Unit (IMU)

An IMU is a system that measures or estimates the value of angular and linear velocities with the help of three-axis accelerometers, three-axis gyroscopes and three-axis magnetometers. After integrating the accelerations corrected by the angular speeds in the body-axis, three attitude angles are obtained [Mum+17]. In former days these devices mainly used gyrostats which were bulky and expensive instruments. The development of accelerometers and gyroscopes as Microelectromechanical systems(MEMSs) caused a breakthrough in IMU applications. The sensors are much cheaper and smaller than former sensors. Nowadays they can be found in nearly every smartphone and a lot of consumer devices.

The mathematical model of an IMU can be described as follows: An accelerometer is capable of directly measuring $\vec{a}(t)$ which gives by integration:

$$\vec{v}(t) = \int \vec{a}(t)\,\mathrm{d}t$$

Further integration results in:

$$\vec{s}(t) = \int \vec{v}(t)\,\mathrm{d}t = \iint \vec{a}(t)\,\mathrm{d}^2 t$$

Similarly, the gyroscope measures the angular velocity $\vec{\omega}(t)$ which gives by integration:

$$\vec{\phi}(t) = \int \vec{\omega}(t)\,\mathrm{d}t$$

Problems occur when IMUs are used over longer periods of time due to the ever increasing error for position and orientation. Let us assume that there is a small but constant error ϵ for the measurement of acceleration in one dimension:

$$a(t) = a_{actual}(t) + \epsilon$$

This leads to an accumulated error which is increasing quadratically over time:

$$s(t) = \iint a_{actual}(t) + \epsilon \, \mathrm{d}^2 t = s_{actual}(t) + \epsilon \cdot t^2$$

Therefore, IMUs alone are usually only used for short time navigation and the estimated position is regularly compared and adjusted by other means of position determination.

Odometry

Odometry can be used for nearly any ground-based robot. Given the initial position and summing up all movements over time it is possible to determine the current position. This is typically done by counting the revolutions of wheels or measuring other means of mechanical propulsion. Due to its easy application this method is in wide use for a lot of ground-based robots for already a long time [Hid+17]. Errors in the position determined by odometry may be caused by inaccurate measurements because of sliding wheels, uneven surfaces or incorrect wheel parameters [NAS71]. Without further sensors these errors cannot be detected and/or corrected. However, odometry is not applicable to flying robots because for such robots there is no mechanical propulsion with physical contact to the environment.

2.2 Literature Survey

In the following section we will list different implementations of LPS and highlight the specific properties of these systems. For the characterization we will look at the used transmission medium such as radio, sound or optical signals as well as the calculation method such as TOA, TDOA, AOA or RSSI.

2.2.1 Systems using Radio and Ultrasonic Signals

The field of robotics provides a wide range of different implementations of LPS. The following examples give an overview of systems using radio and ultrasonic signals.

Bjerknes et al. [Bje+07] describe a TOA localization system to determine the position of a mobile robot: In their work, they use 8 base stations to transmit ultrasonic signals and one mobile receiver on the robot. The transmitters send single bursts in a predefined sequence while the receiver on the robot is calculating the distance to each beacon based on the propagation delay. Time synchronisation between beacons and the mobile unit is done by a radio signal. Bjerknes et al. present 2D localization as well as 3D localization with a accuracy of ±1 cm for X- and Y-Axis and ±4 cm for Z-Axis.

Koenig, Schmidt and Hoene [KSH11] use TOA localization based on commonly known wireless networks like IEEE 802.11: They compare TOA and RSSI using off the shelf Wireless-LAN Hardware. The test setting consists of six Access-Points and a notebook which acts as mobile object. They use a sequence of User Datagram Protocol (UDP) packets which are sent over the IEEE 802.11 network in order to measure the time delay. Furthermore they describe the optional usage of a Software Defined Radio (SDR) which acts a a monitor node observing network traffic for a better accuracy of the time measurements. Possible locations are restricted to the 2-dimensional plane and results show a mean error of 1 meter.

Global Positioning System (GPS) is a very well established system which uses TDOA [MBP99]: Time-synchronized satellites send code-multiplexed signals on two frequencies. The signals are directly used to measure propagation delay on the one hand and also contain system meta data like positions of the satellites on the other hand. The system was initially built for non-civil use. Today receivers are available to everyone. The accuracy estimation of the system varies from millimeters to tens of meters depending on the receiver, needed time for a measurement and access to encrypted information. Typically GPS is only used in outdoor applications since in buildings the signal reception is either weak or not possible at all.

Pang and Trujillo [PT13] use TDOA with one omnidirectional transmitter and four receivers for indoor localization. The position for the mobile object is limited to the 2-dimensional space. Analogous signal processing is done by a rectifier circuit and digital signal processing is done with a low cost micro controller. Accuracy is not extensively discussed in the paper, however, it shows that it depends on the position within the measurement area and varies from approx. 8 cm to 25 cm to completely wrong results in the worst cases. The paper points out that the main reason for inaccuracies might be caused by problems detecting the edge of the transmitted signal precisely.

Oksar [Oks14] uses RSSI based localization in Bluetooth networks: Bluetooth transmitters are used as mobile objects and three Bluetooth receivers are used as reference points with known location. The receivers are off-the-shelf phones which have built-in means for RSSI measurement. The localization is simplified to the 2-dimensional plane and positions are restricted to a grid of 1 meter cell size. The presented method uses a number of sequential measurements which take a few seconds. The position is calculated based on Root-Mean-Square-Error metric.

Cheng, Wu and Zhang [CWZ11] use RSSI based localization in ZigBee networks: They describe a system using six fixed ZigBee nodes and one mobile node on a wheeled robot. There is also one TDOA node but the calculation is mainly based on RSSI. The position of the mobile robot is restricted to the 2-dimensional plane since the robot drives on wheels. Using filtered polynomial fitting they show an average localization accuracy of 0.5 meters.

Koenig, Schmidt and Hoene [KSH11] demonstrate location fingerprinting based on common known wireless network IEEE 802.11: They use location fingerprinting with RSSI and also TOA. With a test setting consisting of six Access-Points it is possible to determine the location of a notebook with mean error of 1.5 meters using RSSI fingerprinting. Possible locations are restricted to the 2-dimensional plane.

2.2.2 Fixed Camera Systems

The following examples give an overview of systems using fixed cameras which are distributed in space. Additional active or passive markers on a mobile object can enhance the accuracy and/or reliability of the system. The position of the mobile object is determined by image processing.

Szaloki et al. [Sza+13] apply the localization method using fixed cameras to the 3-dimensional space with a system called *Smart Mobile Eyes for Localization (SMEyeL)*. A robotic arm is used in this experiment to move a marker which is filmed by three cameras simultaneously. The results show a precision of approximately 1 mm for a distance of 1 to 2 meters.

The motion-capture system *Vicon MX* is used in the work of How et al. [How+08] in which a test environment for autonomous indoor vehicles is described. This system uses reflective markers on mobile objects and it is capable of measuring the position at a rate of 100 Hz. The accuracy for an object which does not move is stated to be less than 0.325 mm in x-dimension, and less than 0.199 mm in y-dimension, z-dimension is fixed to zero in this test. 18 cameras are used in a room with dimensions of 8 by 5 by 3 meters.

An example for a fixed camera system is demonstrated by the company *Festo* in a project called *eMotionButterflies* [Fes15]. The robots in this project are very lightweight with only $32g$ and equipped with beating wings with a wingspan of $50cm$ for propulsion. The shape and moving pattern of these robots is inherited from butterflies. Every flying robot is equipped with two active markers consisting of infrared LEDs. The system uses 10 infrared cameras with a frame rate of 160 Hz. Accuracy is not stated in the article but it can be assumed that it is significantly less than the size of the flying robots.

Jeong and Jung [JJ12] describe the localization of a quadcopter using a single camera for indoor application. The paper addresses the needed correction for radially distorted images by camera lenses. It further describes the usage of markers on the quadcopter. Accuracy for a region of approximately 3 by 3 meters is stated as 2 cm by the resolution of the used camera.

2.2.3 On-Board Camera Systems

It is also possible to mount one or several cameras on a mobile object. Image processing is used to calculate the position based on the images of the environment.

Wang et al. [Wan+12] present a self-localization system for mobile robots. A predefined map is used in this example to determine the location of a robot with one mobile video camera. A computer vision algorithm is used to perform feature extraction on the video feed. Experimental results are carried out and show an accuracy of 7 to 20 cm depending on the environment where the robot is located.

Hijikata, Terabayashi and Umeda [HTU09] propose a method using infrared LEDs. In contrast to feature extraction on the full video input it uses infrared LEDs as markers which are spread across the room. The camera is fitted with a infrared filter, s.t. only the LEDs are visible as light pixels in the video feed. This information is processed in order to calculate the position of the mobile robot. Tests in the paper show an accuracy of 5 to 15 cm depending on the test conditions.

2.2.4 Comparison

The described systems provide an accuracy ranging from less than a millimeter to several meters. Table 2.1 shows a comparison of the given examples. In this comparison the optical systems provide the best accuracy ranging from less than 0.4mm to 50cm. Ultrasonic based systems provide mid-range accuracy ranging from 1cm to 25cm. The last ranked systems by accuracy are radio based ones ranging from 0.5m to 10m.

System	Type of system	Calculation method	2D/3D	Accuracy
Bjerknes et al.	ultrasonic	TOA	2D	1cm
Bjerknes et al.	ultrasonic	TOA	3D	4cm
Koenig, Schmidt and Hoene	radio (IEEE 802.11)	TOA	2D	1m
GPS	radio	TDOA	3D	10mm[a] - 10m
Pang and Trujillo	ultrasonic	TDOA	2D	8cm - 25cm
Oksar	radio (Bluetooth)	RSSI	2D	1m
Cheng, Wu and Zhang	radio (ZigBee)	RSSI	2D	0.5m
Koenig, Schmidt and Hoene	radio (IEEE 802.11)	location fingerprinting	2D	1.5m
Szaloki et al.	optical (3 fixed cameras)	image processing	3D	1mm
How et al.	optical (18 fixed cameras)	image processing	2D	< 0.4mm
Festo	optical (10 fixed cameras)	image processing	2D	$\ll 50$cm
Jeong and Jung	optical (1 fixed camera)	image processing	3D	2cm
Wang et al.	optical (on-board camera)	image processing	2D	7cm - 20cm
Hijikata, Terabayashi and Umeda	optical (on-board camera)	image processing	2D	5cm - 15cm

Table 2.1: Comparison of different LPSs.

[a]Best accuracy of GPS depending upon access to encrypted signals for military use [MBP99].

CHAPTER 3

System design

In this chapter we are first introducing and later specifying the requirements which are needed for a LPS and our implementation. Then, we select an appropriate system based on the requirements, the described basic concepts and the literature survey. Then, we formulate the mathematical model and decide on the parameters for our concrete implementation.

3.1 Local Positioning System Requirements

First we need to specify which requirements are needed for a LPS that is capable of determining the position for an UAV in indoor locations. The following list gives an overview of these requirements:

- **Maximum range**
  We can assume to have an indoor location (e.g., lecture hall, seminar room, gym, etc.) to operate the UAV. The size of this indoor location therefore gives the required maximum range.
- **Maximum weight**
  For every UAV there is a maximum take-off weight and therefore also a maximum weight for payload. If there are parts of the LPS placed on the UAV these parts must not exceed the maximum payload weight. Furthermore, additional weight, which is added to the UAV, decreases flight time. Therefore, the parts of the LPS which are placed on the UAV should be as light as possible.
- **Maximum speed**
  The LPS must be functional even if the UAV is moving very fast. This implies that the maximum speed of the UAV is another requirement.

- **Accuracy**
  The difference of the measured position given by the LPS and the actual position must not exceed a certain limit. This defines the required accuracy.

- **Measurement frequency**
  For certain applications (e.g., autopilot systems) it is crucial to have position data which is updated with a high frequency. Therefore the measurement frequency is an important requirement.

- **Number of mobile objects**
  Depending on the application the LPS must be capable of determining the position of 1 up to n mobile objects.

In the following sections we will analyse two quadcopters as examples for UAVs. We will also describe a specific example where UAVs are operated in an indoor location. Based on these examples we will thereafter conclude the requirements for our implementation.

3.1.1 AscTec Pelican

The *AscTec Pelican* built by *Ascending Technologies* is a quadcopter flight system intended for research applications. The manufacturer lists the following areas as main applications for this quadcopter [Asc14]: Integration of custom sensors, indoor navigation with cameras and laser scanner as well as computer-vision applications.

Technical data [Asc14] for this quadcopter is listed in Table 3.1.

AscTec Pelican	
Dimensions	65.1 x 65.1 x 18.8 cm
Propeller size	10"
Motors	4 x 160W
Maximum thrust	36N
Maximum payload	650g
Maximum total weight	1650g
Maximum airspeed	16m/s
Maximum flight time	30mins (without payload)
Battery	6250mAh (LiPo)

Table 3.1: Technical data for quadcopter *AscTec Pelican* [Asc14].

Given the maximum thrust of $36N$ and net weight of $1000g$ we can calculate the maximum acceleration of this quadcopter.

$$a_{max} = \frac{F}{m} = \frac{36N}{1kg} = 36m/s^2$$

Taking the needed thrust for hovering (against gravity) into account we conclude the maximum acceleration upwards.

$$a_{up} = 36m/s^2 - 9,81m/s^2 \approx 26,2m/s^2$$

And maximum acceleration sidewards.

$$a_{side} = \sqrt{(36m/s^2)^2 - (9,81m/s^2)^2} \approx 34,6m/s^2$$

3.1.2 Parrot AR Drone 2.0

The *AR Drone 2.0* is built by *Parrot* for home and entertainment use. A Software Development Kit (SDK) [Par16] is provided for this quadcopter which makes it popular as low-cost research platform as well.

Technical data [Par15b] for this quadcopter is listed in Table 3.2.

AR Drone 2.0	
Dimensions	52 x 52 x 11 cm
Motors	4 x 14.5W
Total weight (internal frame)	380g
Total weight (external frame)	420g
Maximum flight time	12mins / 18mins
Battery	1000mAh / 1500mAh (LiPo)

Table 3.2: Technical data for quadcopter *AR Drone 2.0* [Par15b].

This quadcopter has officially no rating for a maximum payload capacity. But there is an official add-on component for GPS tracking that can be placed on the quadcopter. This module weights $31g$ [Par15a].

Unofficial information for maximum payload can be found in various online reports about custom modifications. A test video shows the quadcopter with payload of approx. $162g$ [ARD13].

3.1.3 Air Race competition

The *Air Race* competition is part of the international championship for self-made, autonomous, and mobile robots called *RobotChallenge* [INN15a].

For the *Air Race* competition an UAV follows a track as fast as possible within a predefined timespan. The track consists of two poles which are circled by the UAV in an 8-figure. Points are given for each complete round. If the UAV touches the ground or the safety net, it restarts a new attempt and the points are reset to zero. The track area is rectangular with dimensions $b_{track} \geq 5m$ and $l_{track} \geq 10m$. See Figure 3.1 for a drawing of the track.

In 2015 there was a participant at this competition who successfully managed to fly 50 rounds on this track within 10 minutes. This number of rounds included multiple starts but the time was taken as net flight time.

The approximate length of the track is calculated as follows.

$$\text{Pole distance:} \quad d_{pole} = 5m$$

$$\text{Circle radius:} \quad r_{circ} = 2m$$

$$\text{Diagonal length:} \quad s_{diag} = \sqrt{d_{pole}^2 + (2 \cdot r_{circ})^2} = \sqrt{(5m)^2 + (2 \cdot 2m)^2}m \approx 6,4m$$

$$\text{Half-circle length:} \quad s_{circ} = \frac{2 \cdot r_{circ} \cdot \pi}{2} = \frac{2 \cdot 2m \cdot \pi}{2} \approx 6,3m$$

$$\text{Total track length:} \quad s_{track} = 2 \cdot s_{diag} + 2 \cdot s_{circ} \approx 2 \cdot 6,4m + 2 \cdot 6,3m = 25,4m$$

The quadcopter of the aforementioned participant completed the track 50 times within 10 minutes which makes it possible to calculate the average speed.

$$v = \frac{s}{t} = \frac{50 \cdot 25,4m}{10 \cdot 60s} \approx 2,11m/s$$

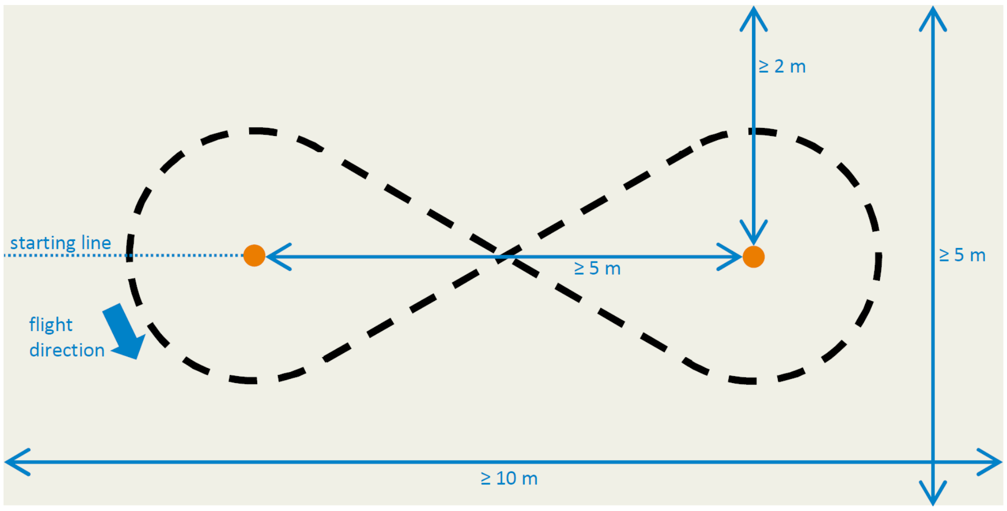


Figure 3.1: Track for *Air Race* competition: Two poles with a distance of at least 5 meters are circled by the robot in an 8-figure. [INN15b]

3.1.4 Specified Requirements

Based on the analysed quadcopters we specify the following requirements for our implementation:

- **Maximum range**
  As reference for an indoor location we choose the *Air Race* competition. Therefore the maximum distance is specified by: $s_{max} = max(b_{track}, l_{track}) = 10m$
- **Maximum weight**
  For the two analysed quadcopters the maximum payload is $650g$ and $31g$ (respectively $162g$). We specify the maximum weight for the parts of the LPS which are placed on the UAV with $m_{max} = 100g$. This ensures that both of the analysed quadcopters can be operated with the LPS.
- **Maximum speed**
  For the maximum speed of the quadcopter s.t. the LPS is still able to track the position we will take the reference of the aforementioned participant at *Air Race* competition. We will round this value up and take it as requirement for maximum speed: $v_{max} = 3m/s$.
- **Accuracy**
  The horizontal dimension of the two analysed quadcopters is $65.1cm$ and $52cm$. In order to achieve reasonable results for trace recording and autopilot applications we specify the accuracy of the LPS by one tenth of this dimension: $s_{accuracy} = 5cm$.
- **Measurement frequency**
  Based on the specified maximum speed and the specified accuracy we can calculate the minimum frequency needed to satisfy both of these requirements: $f_{min} = v_{max}/s_{accuracy} = \frac{3m/s}{5cm} = 60Hz$
- **Number of mobile objects**
  As pointed out in motivation we implement a LPS which is capable to determine the position of one UAV. The requirement for the number of mobile objects is therefore $n_{objects} = 1$.

A summary of all specified requirements is listed in Table 3.3.

Specified requirements	
Maximum range	$s_{max} = 10m$
Maximum weight	$m_{max} = 100g$
Maximum speed	$v_{max} = 3m/s$
Accuracy	$s_{accuracy} = 5cm$
Measurement frequency	$f_{min} = 60Hz$
Number of mobile objects	$n_{objects} = 1$

Table 3.3: Specified requirements for our LPS implementation.

3.2 System Selection

For our implementation of the LPS we need to select the transmission medium and calculation method as laid out in Section 2. We look at the described options and analyse if they are suitable for the specified requirements.

3.2.1 Transmission Medium

In Section 2.1.1 radio signals, sound signals, and optical systems are described as possible transmission medium for LPSs. Due to the computational complexity of optical systems (see examples for implementations in Section 2.2) we decide not to use an optical system.

To decide between radio and sound signals we look at the following calculation: Given the specified accuracy we calculate the time which the signal takes to cover this distance. The required accuracy is specified by $s_{accuracy} = 5cm$.

We calculate the timespan for sound signals with

$$t_{sound} = \frac{s_{accuracy}}{v_{sound}} = \frac{5cm}{340m/s} \approx 150\mu s.$$

Let us assume the system is running at $f_{clk} = 50Mhz$. Then we calculate the number of clock cycles within this timespan with

$$n_{cycles} = t_{sound} \cdot f_{clk} = 150\mu s \cdot 50Mhz = 7500.$$

We also calculate the timespan for radio signals with

$$t_{light} = \frac{s_{accuracy}}{v_{light}} = \frac{5cm}{3 \cdot 10^8 m/s} \approx 1.5 \cdot 10^{-4}\mu s.$$

Let us again assume the system is running at $f_{clk} = 50Mhz$. This gives far less then one clock cycle within this timespan with

$$n_{cycles} = t_{light} \cdot f_{clk} = 1.5 \cdot 10^{-4}\mu s \cdot 50Mhz = 7.5 \cdot 10^{-3} \ll 1.$$

Sound signals are therefore preferred over radio signals due to slower signal propagation and therefore easier signal processing. Furthermore we decide to use ultrasonic signals because they are outside the range of audible frequencies for humans.

3.2.2 Calculation Method

In Section 2.1.2 we describe various calculation methods such as TOA, TDOA, AOA, RSSI, location fingerprinting, fixed camera systems, and on-board camera systems. Since we use ultrasonic signals it remains to choose an appropriate calculation method.

For AOA based calculation we would need to measure the angle from where the signal is received. This could be achieved using rotating antennas or sector antennas (see Section 2.1.2). Rotating antennas would make the system mechanically complex and sector antennas would imply the usage of a lot of ultrasonic receivers. Therefore we decide not to use AOA.

It remains to decide between methods based on propagation delay (TOA, TDOA) and based on received signal power (RSSI, location fingerprinting). Methods based on received signal power are described to be very unreliable due to shadow fading and because the receiver cannot distinguish between signal strength in line-of-sight path and reflected path [Nai+12]. Therefore we decide to use a method based on propagation delay.

For TOA based systems there is a time synchronisation needed between the sender and receiver of the signal s.t. the propagation delay can be measured. We describe several methods (precise clocks, signals sent in both directions, and continuous clock synchronisation) for this task (see Section 2.1.2). All this methods require additional hardware to be placed on the mobile object. To reduce the weight on the UAV we therefore decide not to use TOA. As a result of all these considerations we decide to use TDOA for our implementation.

3.2.3 Fixed Senders vs. Mobile Sender

Depending on the number of mobile objects and the number of fixed reference points it can be beneficial to use mobile senders with fixed receivers or the other way round. We specified the number of mobile objects to be $n_{objects} = 1$. Let n_{ref} denote the number of fixed reference points. As laid out in Section 2.1.2 there are at least 4 reference points needed for TDOA in the 3-dimensional space. Therefore, it holds that $n_{ref} \geq 4$.

- **Fixed senders and mobile receiver**
  Using a ultrasonic signal with only one carrier frequency we need to use time-multiplexing for a number (n_{ref}) of multiple senders. This means that every transmitter needs a timeslot where it can send its signal. The duration of each time-slice for time-multiplexing is given by the signal delay for the specified maximum range ($t_{slice} = s_{max}/v_{sound}$). After all n_{ref} signals were received TDOA calculation can be carried out on the receiver. This makes it possible to determine the location every
  $$t_{interval} = t_{slice} \cdot n_{ref}.$$

For the specified value of $s_{max} = 10m$ this gives

$$t_{interval} = \frac{s_{max}}{v_{sound}} \cdot n_{ref} = \frac{10m}{340m/s} \cdot n_{ref} = \frac{n_{ref}}{34} s.$$

Based on $n_{ref} \geq 4$ this gives a lower bound of $t_{interval} \geq 0.12s$.

- **Mobile sender and fixed receivers**
  The single ($n_{objects} = 1$) mobile transmitter is sending its signal and all n_{ref} reference points receive the signal after the maximum propagation delay of $\delta t_{max} = s_{max}/v_{sound}$. Then TDOA calculation can be carried out. This makes it possible to determine the location every δt_{max}. For the specified value of $s_{max} = 10m$ this gives

$$\delta t_{max} = \frac{s_{max}}{v_{sound}} = \frac{10m}{340m/s} = \frac{1}{34} s \approx 0.03s.$$

For the specified number of mobile objects $n_{objects} = 1$ the method to use one mobile sender with fixed receivers is preferred. This makes it possible to determine the position with $t_{interval}/\delta t_{max} = n_{ref}$ times higher frequency. Since $n_{ref} \geq 4$ the frequency is at least by a factor of 4 higher. Using more fixed reference points results in even a larger factor.

3.3 Mathematical Model

In order to determine the position of the UAV a mathematical model is needed which makes it possible to get the UAV's unknown position by any measurable values. TDOA provides a model to get the position by measured differences of the signal propagation delay. For the following formulas we assume a 3-dimensional space and an euclidean coordinate system.

Let $\mathbf{M}$ denote the position of the UAV and $\mathbf{P_n}$ the position of base station n (where N stations exist).

$$\mathbf{M} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \forall 0 \leq n < N : \mathbf{P_n} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}$$

The distance between the UAV and a base station is given by the euclidean norm of the difference vector.

$$d_n = ||\mathbf{M} - \mathbf{P_n}|| = \sqrt{(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2}$$

Let v denote the speed of signal propagation and t_M the time when a signal was sent by the UAV. The signal is detected at base station n at time t_{P_n}.

$$t_{P_n} = t_M + \frac{d_n}{v}$$

The time differences r_{mn} can be measured with synchronized base stations.

$$r_{mn} = t_{P_m} - t_{P_n} = \frac{d_m - d_n}{v}; 1 \leq n < N, 0 \leq m < n$$

In 3-dimensional space there are 3 values which can be measured using 3 base stations: r_{01}, r_{02}, r_{12}.

$$r_{01} = \frac{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} - \sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2}}{v}$$
$$r_{02} = \frac{\sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2} - \sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2}}{v}$$
$$r_{12} = \frac{\sqrt{(x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2} - \sqrt{(x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2}}{v}$$

In general this would be sufficient to solve the three unknown variables x, y and z. But due to the square root terms there is no simple solution for this set of equation*s [BM02].

In the 3-dimensional space there are 6 values that can be measured using 4 base stations: $r_{01}, r_{02}, r_{12}, r_{03}, r_{13}, r_{23}$. Adding this additional station makes it much easier to solve the set of equations. The work of Bucher and Misra [BM02] presents an explicit solution for x, y and z.

3.4 Design Decisions

In the previous section we describe the selection of the transmission medium and calculation method. We use TDOA calculation based on ultrasonic signals. The UAV is equipped with a transmitter and the fixed reference points are equipped with receivers.

To actually implement the LPS we need to decide on further parameters of the system such as carrier frequency and signal waveform. We need to check for this parameters if the specified requirements (e.g., measurement frequency) can be satisfied.

3.4.1 Ultrasonic Carrier Frequency

In order to define the ultrasonic carrier frequency we carry out a survey of available ultrasonic transmitters and receivers on the market. It turns out that they have fixed resonance frequencies with low bandwidth. One example manufacturer of ultrasonic transmitters and receivers is *Pro-Wave Electronics Corporation* [Pro15]. *Pro-Wave Electronics Corporation* lists these center frequencies for its products: $25kHz$, $32.8kHz$, $40kHz$, $43kHz$, $48kHz$, $50kHz$, $80kHz$, $125kHz$, $200kHz$, $235kHz$, and $320kHz$. *SensComp Global Components* [Sen18] states that absorption varies with frequency. Attenuation due to absorption in air is higher if the frequency is higher. Therefore lower frequencies are better suited for long range propagation. Given the availability of components and for low attenuation by absorption we decide to use an ultrasonic carrier frequency of $f_c = 25kHz$.

3.4.2 Ultrasonic Signal Waveform

Some sensors, which use ultrasonic signals, send a packet of several ultrasonic impulses. This is often referred to as *burst* (see Figure 3.2). The figure shows the voltage on the ultrasonic transmitter while sending one ultrasonic burst. In this example the burst consists of 8 pulses of the respective carrier frequency. One example for such an sensor is *SRF05* [Dev15].

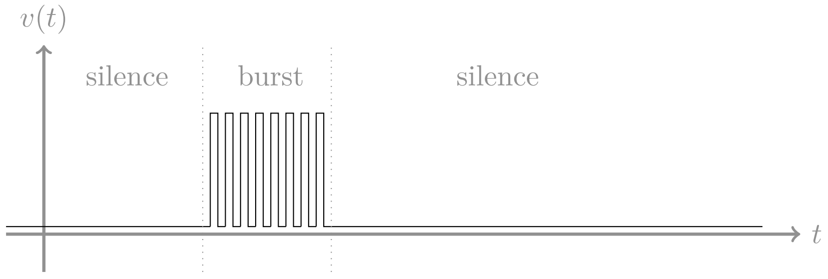


Figure 3.2: Voltage on the ultrasonic transmitter while sending one ultrasonic burst. The burst consists in this example of 8 pulses of the respective carrier frequency.

Using a burst as transmitted signal makes it difficult to determine the accurate propagation delay since there is no sharp slope for the received signal. Figure 3.3 shows the transmitted (yellow) and received (blue) signal for a single burst of 8 pulses. The shape of the received signal makes it difficult to measure a precise propagation delay. Furthermore there might be multiple echo paths from walls or other objects in the room (these can also be seen in Figure 3.3) which might get confused with the next burst of pulses. Pang and Trujillo [PT13] point out that they had issues using simple burst signals. The accuracy was therefore significantly reduced.

Continuous Signal: Gold Code

To achieve more precise measurement we do not use a single burst of pulses but a continuous (pseudo) random signal. The received signal is then correlated against a reference to determine the propagation delay. For the signal it is important that auto-correlation value is low to avoid incorrect matches. Gold [Gol67] describes sequences which satisfy the needed properties.

We carry out several tests on real hardware using such sequences. However, these tests show that the waveform of this sequences is not suitable for ultrasonic transmission. Ultrasonic transducers behave like oscillating circuits and need some time after bursts in order to decay. Figure 3.4b shows the received signal for a sequence of 32 bits gold code. The received signal keeps high amplitudes for short interruptions by single `0` bits. Figure 3.4a shows the transmitted signal for other 32 bits gold code. The sequence is amplitude modulated s.t. each bit is encoded by 8 pulses of the carrier frequency.

Continuous Signal: Improved Sequence

Based on the tests above we need to adapt the transmitted signal. Single `0` have to be avoided in any case because they were overshadowed by the slow decay of ultrasonic transducers. More tests show that also `00` sequences should be removed because of the same reason. Therefore the improved sequence is generated as a alternating sequence of one to four times the `1` symbol and three to seven times the `0` symbol. Furthermore the complete sequence needs low auto-correlation to avoid incorrect matches. This is solved by generating sequences and testing the correlation value until a good sequence is found. To further improve correlation quality the length of the received sequence is doubled from 32 symbols to 64 symbols. And the length of the complete reference signal is reduced to 372 symbols. Figure 3.5 shows an example of a received signal and the corresponding reference.

Relaxed Version: Burst Signal

As it turned out during hardware implementation the correlation of the continuous signal is very prone to variations in frequency. Therefore, we have to step back to a relaxed version of the ultrasonic signal. As an alternative we use burst signals similar to various examples in literature to get a working system.

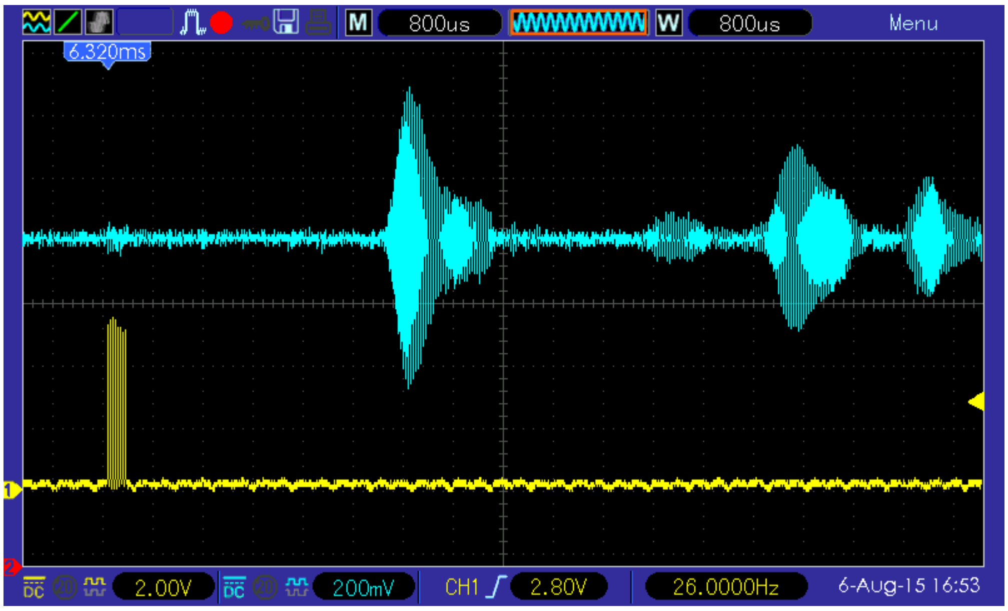
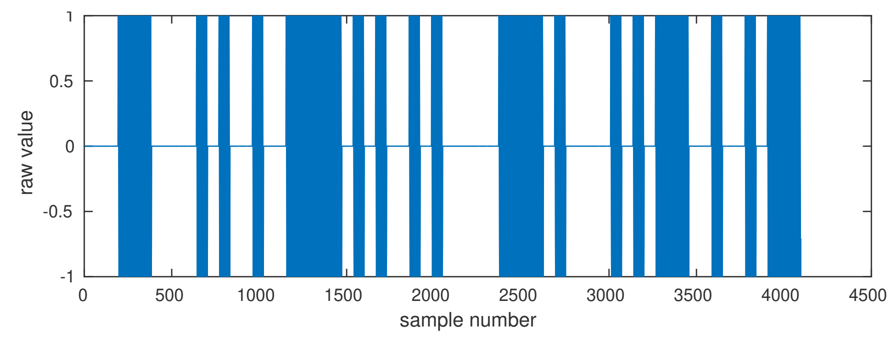
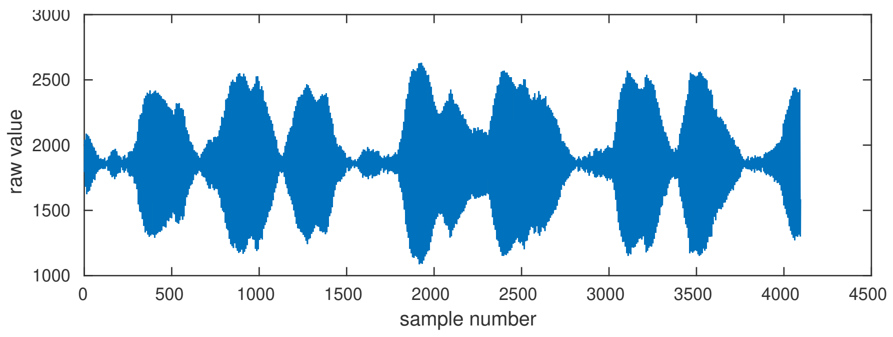


Figure 3.3: This oscillogram shows an ultrasonic signal test using one ultrasonic burst. Channel 1 (yellow) shows the voltage on the ultrasonic transmitter. Channel 2 (blue) shows the voltage on the ultrasonic receiver.

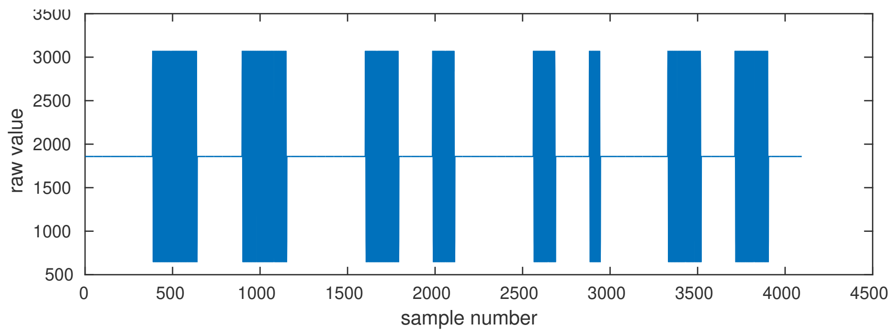


(a) Transmitted sequence of 32 bits gold code. The sequence is amplitude modulated s.t. each bit is encoded by 8 pulses of the carrier frequency.

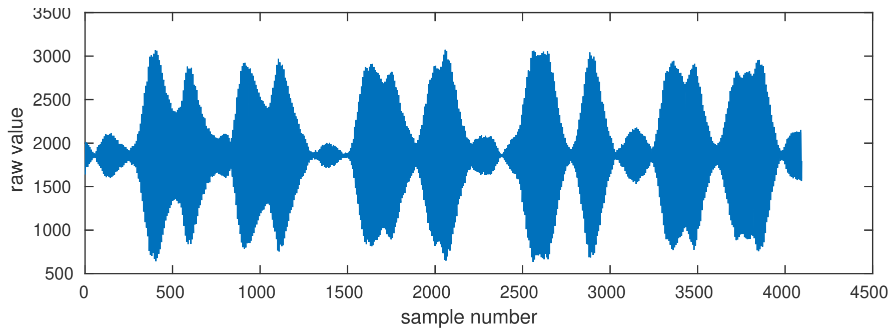


(b) Received sequence of 32 bits gold code. The received signal keeps high amplitudes for short interruptions by single `0` bits.

Figure 3.4: Tests transmitting and receiving ultrasonic signal using gold code sequences.



(a) Transmitted signal for 64 bits of the improved sequence (matched to the received signal below).



(b) Received signal for 64 bits of the improved sequence. The sequence is generated alternating one to four times the `1` symbol and three to seven times the `0` symbol.

Figure 3.5: Tests transmitting and receiving ultrasonic signal using improved sequence.

3.4.3 ADC Conversion

To perform timing measurements we implement correlation on the received signal. This makes it possible to align the carrier frequency of the transmitted signal to the reference signal. We are using an ultrasonic carrier frequency of $f_c = 25kHz$. To satisfy the Nyquist-Shannon sampling theorem [Sha49] the sampling frequency must be at least the double of the bandwidth: $f_{sample} \geq 2 \cdot B$. This is in our case $f_{sample} \geq 2 \cdot f_c = 50kHz$.

First, we use the *National Semiconductor ADC128S022*, 8-Channel, 12-bit Analog-to-digital converter (ADC). The following calculation gives the actual sampling frequency: System clock $f_{sysclk} = 50MHz$. ADC clock: $f_{adcclk} = \frac{f_{sysclk}}{16} = 3.125MHz$.
One ADC-Sample needs 16 clock cycles: $f_{sample} = \frac{f_{adcclk}}{16} \approx 195kHz$.
Later, the *Linear Technology LTC2308* low noise 8-Channel, 12-bit ADC is used. This ADC is part of the FPGA-Boards which are used to implement the system. The revised calculation in Section 5.2.4 gives the actual sampling frequency which still satisfies the the Nyquist-Shannon sampling theorem.

Based on the initial modulation waveform (Continuous signal: Gold code) the following section calculates the maximum update frequency of the position measurement.
Given the ultrasonic carrier frequency $f_c = 25kHz$, the period length equals $t_c = \frac{1}{f_c} = \frac{1}{25kHz} = 40\mu s$. Using 8 pulses per bit gives the following for the modulated signal: $t_{bit} = 8 \cdot t_c = 8 \cdot 40\mu s = 0,32ms$.
Assuming a sample depth of $n_{sample} = 2048$, the correlation (and therefore the position determination) can be performed approx. 100 times per second: $f_{corr} = \frac{f_{sample}}{n_{sample}} = \frac{195kHz}{2048} \approx 95Hz$. This gives a time duration between each correlation of $t_{corr} = \frac{1}{f_{corr}} = \frac{1}{95Hz} = 10,5ms$ and results in the following number of transmitted bits (of the modulated waveform) per correlation: $n_{corr} = \frac{t_{corr}}{t_{bit}} = \frac{10,5ms}{0,32ms} \approx 32bit$.

3.4.4 Echo Elimination

In order to properly handle echo signals, which might be caused by reflections at walls or similar objects, the period length of the pseudo random signal is chosen long enough. The echo can travel twice the maximum distance as specified for the LPS $s_{echo} = 2 \cdot s_{max} = 20m$. For this distance the echo needs time $t_{echo} = \frac{s_{echo}}{v_{sound}} = \frac{20m}{340m/s} \approx 0.059s$. During this timespan $n_{echo} = \frac{t_{echo}}{t_{bit}} = \frac{0.059s}{0.32ms} \approx 184$ Bits are transmitted. Therefore the period of the pseudo random signal needs to be longer than 184 bits so that an echo within the specified maximum distance can be recognized as such.

CHAPTER 4

Simulation

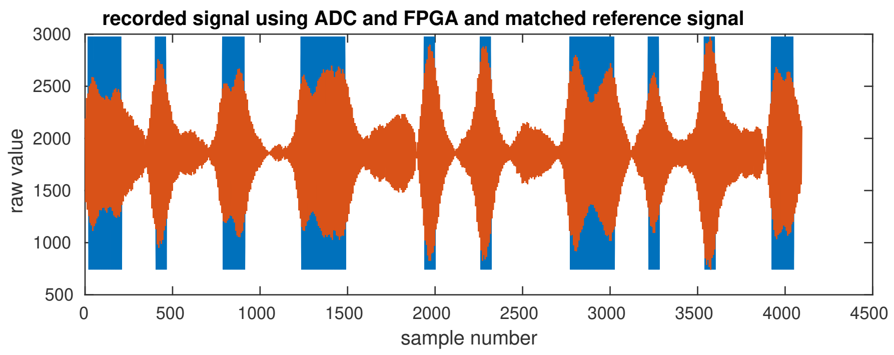
Given the selected system we carry out simulations for essential parts of the system using *Matlab* from *The MathWorks, Inc.* [The19]. These essential parts include the method to measure the time delay and the TDOA calculation algorithm. For the time delay measurement we first simulate the correlation of the received signal with a reference sequence and then the burst detection. For the TDOA calculation we first simulate the algorithm described by Bucher and Misra [BM02] and then a particle filter as presented by Gustafsson and Gunnarsson [GG03].

4.1 Correlation

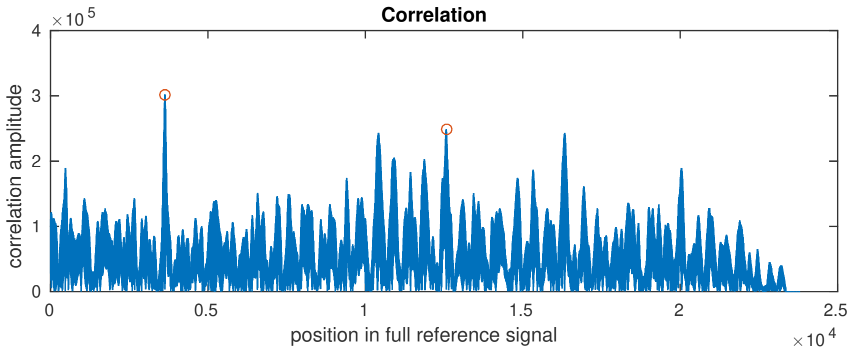
To execute the LPS calculation we first need to measure the difference in propagation delay for the transmitted signal. This signal consists of the ultrasonic carrier frequency which is modulated by the pseudo random sequence. To achieve maximum accuracy for time delay measurements we use the correlation of the received signal with the reference of the pseudo random sequence. This makes it possible to match the received signal to the reference in the time domain and further conclude the propagation delay.

For the following simulations we use actual data for the sampled signal which is recorded using the test circuit shown in Section 5.1.2.

Figure 4.1a shows the recorded signal in red and the matched part of the reference signal in blue. Figure 4.1b shows the correlation for the recorded signal with the reference signal. The maximum and the second highest value are marked with red circles. This simulation shows that it is possible to find the correct part of the reference signal by correlation.



(a) The recorded signal using ADC and FPGA is shown in red and the matched part of the reference signal is shown in blue.



(b) The correlation of the recorded signal with the reference signal is shown in blue. The maximum and the second highest value are marked with red circles.

Figure 4.1: Simulation of signal correlation.

4.1.1 Modified Correlation Function

Formula for correlation is given as follows [Yar10]:

$$R_{xh}(\tau) = x(\tau) \star h(\tau) = \int_{-\infty}^{\infty} x(t) \cdot h(t+\tau)\mathrm{d}t$$

For discrete periodic functions this leads to the formula for discrete circular correlation (for signal sig of N samples and reference ref with length of M):

$$R(m) = \sum_{n=0}^{N-1} sig(n) \cdot ref((m+n)\ mod\ M); 0 \leq m < M \tag{4.1}$$

Since multiplication is an expensive operation in FPGAs we replace it with the following function (which can be implemented using a case distinction):

$$f_{mult}(s,r) = \begin{cases} 9 & \text{if } s_{hi} < s \leq s_{max} \wedge r = 1 \\ 6 & \text{if } s_{hi} < s \leq s_{max} \wedge r = 0.8 \\ 0 & \text{if } s_{hi} < s \leq s_{max} \wedge r = 0 \\ -6 & \text{if } s_{hi} < s \leq s_{max} \wedge r = -0.8 \\ -9 & \text{if } s_{hi} < s \leq s_{max} \wedge r = -1 \\ 3 & \text{if } s_{avg} < s \leq s_{hi} \wedge r = 1 \\ 2 & \text{if } s_{avg} < s \leq s_{hi} \wedge r = 0.8 \\ 0 & \text{if } s_{avg} < s \leq s_{hi} \wedge r = 0 \\ -2 & \text{if } s_{avg} < s \leq s_{hi} \wedge r = -0.8 \\ -3 & \text{if } s_{avg} < s \leq s_{hi} \wedge r = -1 \\ -3 & \text{if } s_{lo} < s \leq s_{avg} \wedge r = 1 \\ -2 & \text{if } s_{lo} < s \leq s_{avg} \wedge r = 0.8 \\ 0 & \text{if } s_{lo} < s \leq s_{avg} \wedge r = 0 \\ 2 & \text{if } s_{lo} < s \leq s_{avg} \wedge r = -0.8 \\ 3 & \text{if } s_{lo} < s \leq s_{avg} \wedge r = -1 \\ -9 & \text{if } s_{min} \leq s \leq s_{lo} \wedge r = 1 \\ -6 & \text{if } s_{min} \leq s \leq s_{lo} \wedge r = 0.8 \\ 0 & \text{if } s_{min} \leq s \leq s_{lo} \wedge r = 0 \\ 6 & \text{if } s_{min} \leq s \leq s_{lo} \wedge r = -0.8 \\ 9 & \text{if } s_{min} \leq s \leq s_{lo} \wedge r = -1 \end{cases} \tag{4.2}$$

Where parameter s is the sampled signal and r the reference signal.

$$s_{max} = \max_{n=0}^{N-1} sig(n)$$

$$s_{min} = \min_{n=0}^{N-1} sig(n)$$

$$s_{avg} = \frac{1}{N} \sum_{n=0}^{N-1} sig(n)$$

$$s_{hi} = s_{avg} + \frac{s_{max} - s_{avg}}{2}$$

$$s_{lo} = s_{avg} - \frac{s_{avg} - s_{min}}{2}$$

Since the sampling frequency is 8 times the ultrasonic carrier frequency the reference signal can be stored as sequence of possible values out of the set $\{-1, -0.8, 0, 0.8, 1\}$ (see Figure 4.2). Using the alternative function f_{mult} (see Equation 4.2) instead of multiplication gives the following modified formula for correlation.

$$R^*(m) = \sum_{n=0}^{N-1} f_{mult}(sig(n), ref((m+n)\ mod\ M)); 0 \leq m < M \tag{4.3}$$

We perform simulations using this modified correlation to verify that the signal still matches the correct position of the full reference signal. Figure 4.3 shows a comparison of the original correlation function (Figure 4.3b) and the modified correlation function (Figure 4.3c).

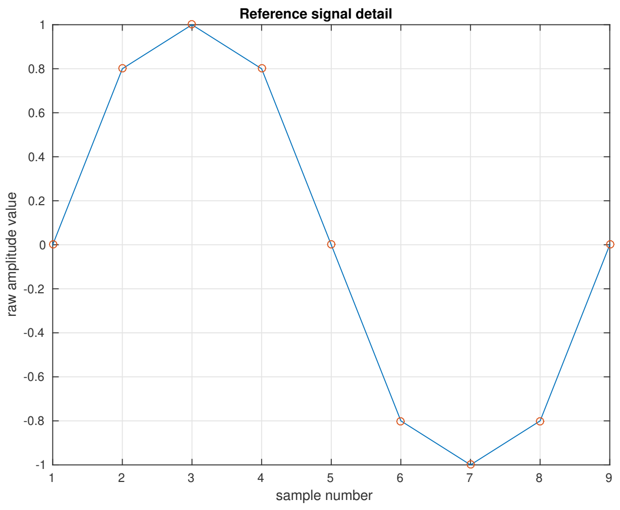
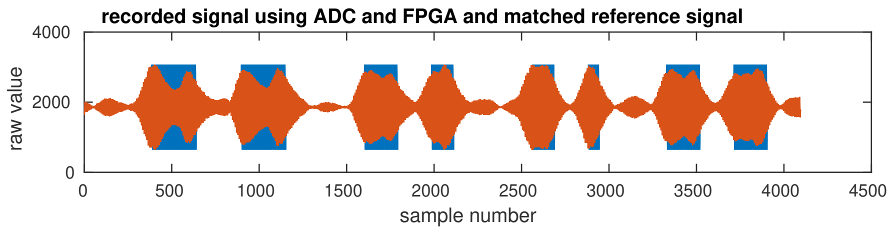
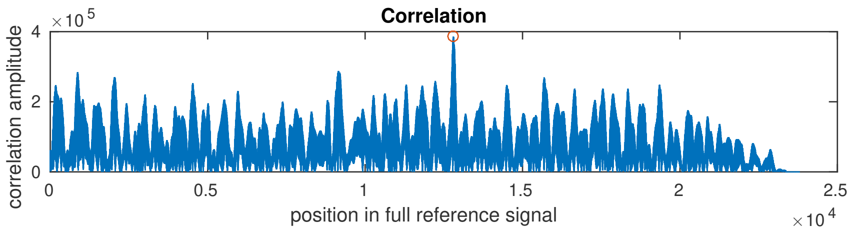


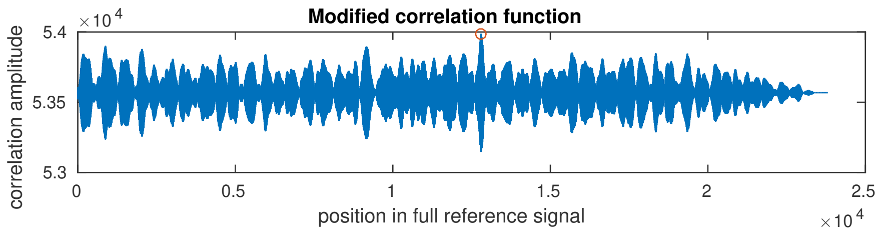
Figure 4.2: The reference signal is stored as sequence of possible values out of the set $\{-1, -0.8, 0, 0.8, 1\}$.



(a) The recorded signal using ADC and FPGA is shown in red and the matched part of the reference signal is shown in blue.



(b) The correlation of the recorded signal with the reference signal is shown in blue. For this computation the original correlation function in Equation 4.1 is used. The maximum is marked with a red circle.



(c) The correlation of the recorded signal with the reference signal is shown in blue. For this computation the modified correlation function in Equation 4.3 is used. The maximum is marked with a red circle.

Figure 4.3: Simulation of signal correlation using original and modified correlation function.

4.2 Burst Detection

We replace the continuous pseudo random ultrasonic signal by a burst signal. Therefore, the receiver needs to detect this burst signal instead of computing the correlation function for a given reference. The block diagram for the implemented burst detector is shown in Figure 4.4. The received signal is converted to a 12 bit value by the ADC. Then the signal is further processed in the FPGA. First, the signal is processed by a half-wave rectifier with integrated decay. The transfer function for this rectifier block is defined as:

$$y_{rect}(t) = \begin{cases} 0 & \text{if } t = 0 \\ x(t) & \text{if } x(t) > y_{rect}(t-1) \\ y_{rect}(t-1) - \delta_{decay} & \text{else} \end{cases}$$

Then, the output is processed by two low-pass filters with different cut-off frequencies. For the first filter the cut-off frequency is adjusted s.t. it gives a long-term average of the background noise. For the second filter the cut-off frequency is adjusted s.t. it retains the received burst signals but filters out disturbances with high frequency (e.g., glitches).

The two low-pass filters are realized as infinite impulse response filters:

$$y_{lp}(t) = \begin{cases} 0 & \text{if } t = 0 \\ y_{lp}(t-1) \cdot \frac{k_{lp}-1}{k_{lp}} + x(t) \cdot \frac{1}{k_{lp}} & \text{else} \end{cases}$$

$$k_{lp} \in \mathbb{N} \wedge k_{lp} \geq 2$$

The difference between the two low-pass filters is then compared against a fixed threshold. This output indicates a detected burst.

In order to determine the value for the fixed threshold and the filter parameters we carry out simulations. As input for the burst detector we use generated signals which are similar to those recorded from real hardware. For input signals in a range of $(-450, 450)$ the following values turn out as reasonable during the simulation: $\delta_{decay} = 1$, $k_{lp1} = 64$, $k_{lp2} = 256$ and $threshold = 50$.

The simulation of the burst detector shows that a burst can even be detected in case of a noisy signal. The test signals are modelled by noise which was recorded from real hardware with an amplitude modulated signal. This modulated signal represents the burst and consists of a constant value V_{noise} and a superimposed sine half-wave with peak-amplitude V_{burst}. Figure 4.5 shows an example with little noise ($\frac{V_{burst}}{V_{noise}} = 5$), whereas Figure 4.6 shows an example with a lot of noise ($\frac{V_{burst}}{V_{noise}} = 0.5$).

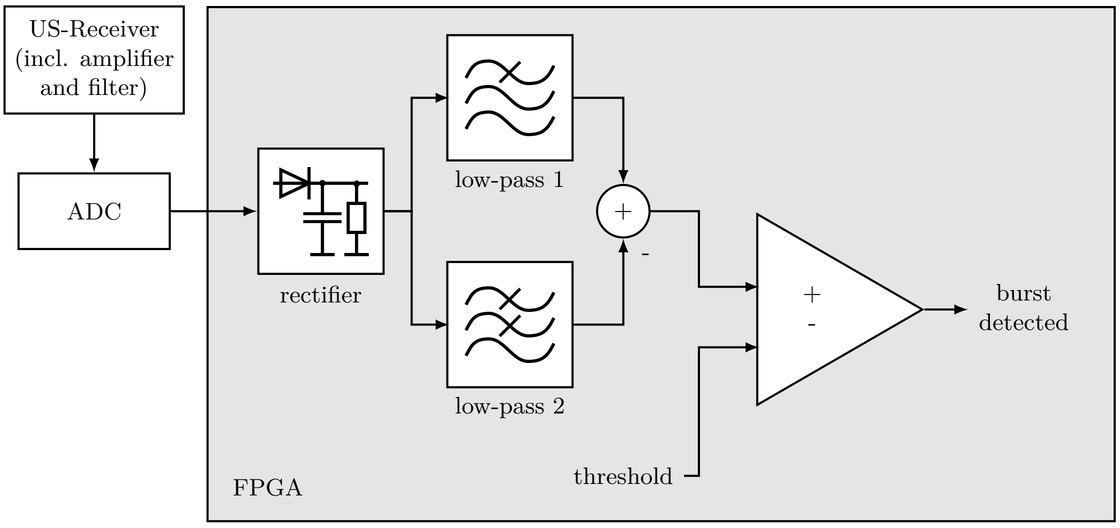
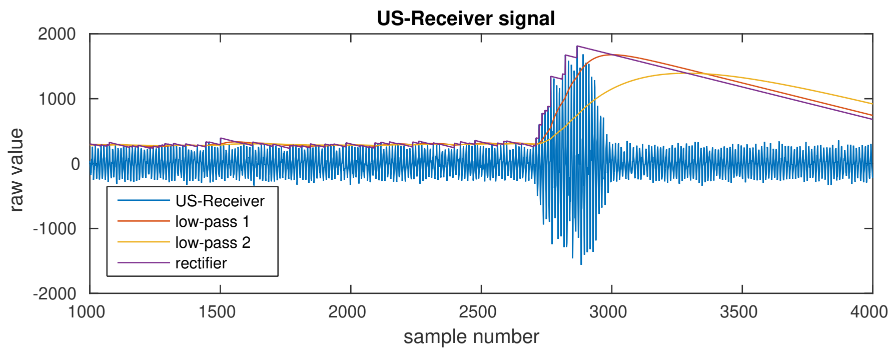
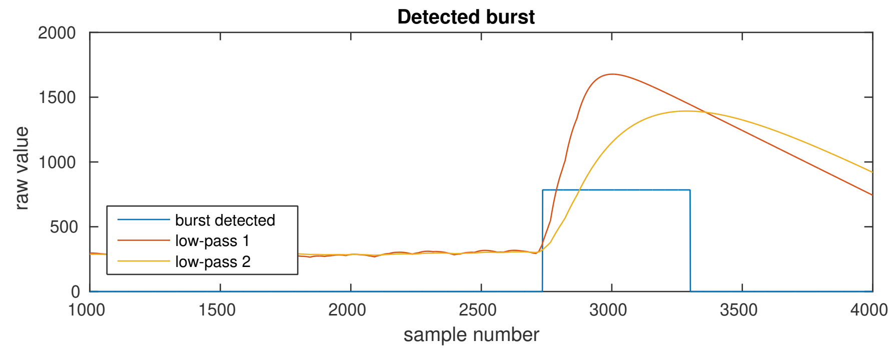


Figure 4.4: Burst detector block diagram: First, the signal is processed by a half-wave rectifier with integrated decay. Then, the output is processed by two low-pass filters with different cut-off frequencies. The difference between the two low-pass filters is then compared against a fixed threshold. This output indicates a detected burst.

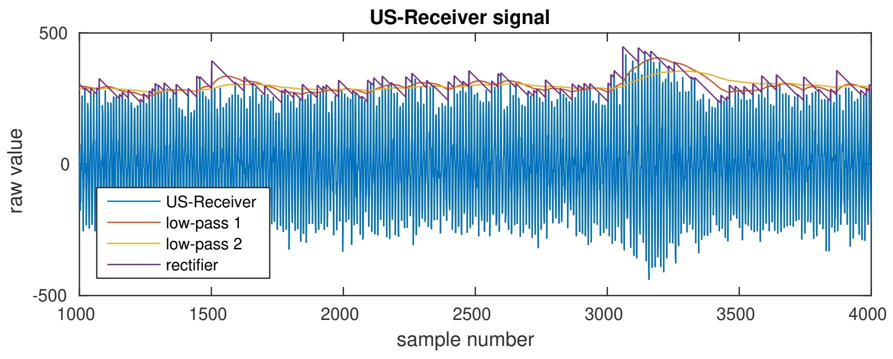


(a) The input signal for the burst detector is shown in blue. The output of the half-wave rectifier with integrated decay is shown in purple. The outputs of the two low-pass filters with different cut-off frequencies are shown in red and orange.

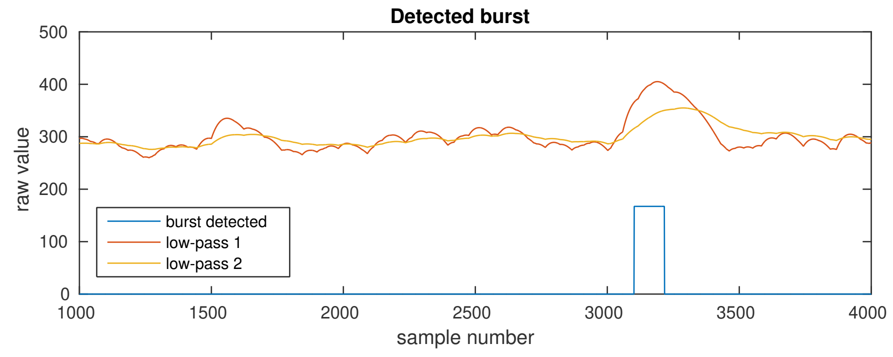


(b) The outputs of the two low-pass filters with different cut-off frequencies are shown in red and orange. This output of the detector indicating a burst is shown in blue.

Figure 4.5: Burst detector simulation with a strong burst and little noise ($\frac{V_{burst}}{V_{noise}} = 5$).



(a) The input signal for the burst detector is shown in blue. The output of the half-wave rectifier with integrated decay is shown in purple. The outputs of the two low-pass filters with different cut-off frequencies are shown in red and orange.



(b) The outputs of the two low-pass filters with different cut-off frequencies are shown in red and orange. This output of the detector indicating a burst is shown in blue.

Figure 4.6: Burst detector simulation with a weak burst and a lot of noise ($\frac{V_{burst}}{V_{noise}} = 0.5$).

4.3 TDOA Calculation and Accuracy

To estimate the implications of measurement errors we perform several simulations of the mathematical model of TDOA. For these simulations a 3-dimensional space is used as specified in the requirements. We use the track of the *Air Race* competition as reference (see Section 3.1.3). The available space for the location of the mobile object is therefore $0 \leq x \leq 10m$, $0 \leq y \leq 5m$ and $0 \leq z \leq 4m$. Figure 4.7 shows this space together with the positions of the base stations P_0 to P_4^*, whereas P_4^* is only used in simulations with 5 base stations. Example location of the UAV denoted by m is given at $(8, 3, 1)$.

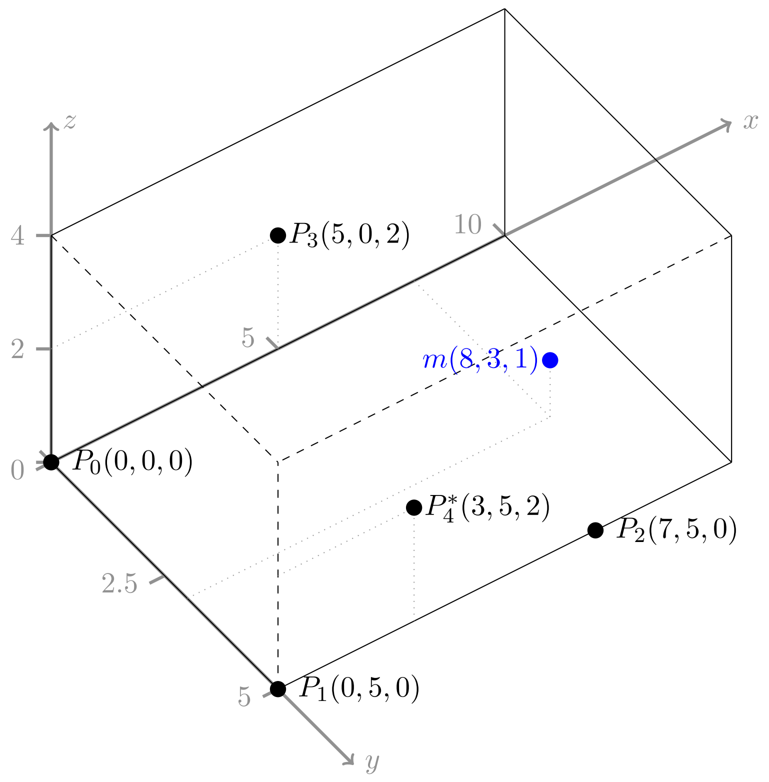


Figure 4.7: TDOA simulation setting: The available space for the location of the mobile object is given by $0 \leq x \leq 10m$, $0 \leq y \leq 5m$ and $0 \leq z \leq 4m$. P_0 to P_4^* indicate the positions of the base stations, whereas P_4^* is only used in simulations with 5 base stations. m denotes the example location of the UAV.

Bucher and Misra [BM02] present an algorithm for the TDOA calculation. We use this algorithm in our simulations. The implementation of this algorithm is attached in appendix (see Listing 8.1, Listing 8.2 and Listing 8.3).

The simulation is performed for randomly chosen positions within the bounding box. For $m_1 = (8, 3, 1)$ the algorithm works fine and yields the correct result: `8.0000 3.0000 1.0000`. But during the simulation it turns out that there are some points (e.g., $m_2 = (10, 2, 2.5)$) where the results are not correct. For m_2 the algorithm calculates the following two incorrect positions: `29.3009 4.2239 21.7354` and `17.6916 3.5073 11.7182`.

Detailed analysis of the algorithm turns out that there are some causes which might lead to incorrect results:

1. Given four base-stations (P_0, P_1, P_2, P_3) there are six measurements for difference in propagation delay (r_{01}, r_{02}, r_{03}, r_{12}, r_{13}, r_{23}). But the algorithm uses only four of those values. If the base-stations are given in a different order, the results are different.
2. The difference in propagation delay is used as absolute value ($|r_{mn}|$ instead of r_{mn}). This causes a loss of information because it makes a difference at which of two base-stations the signal is received first (i.e., if r_{mn} is positive or negative).
3. The calculation leads to two results. This is by design of the algorithm because it contains a root operation.

In order to improve the calculation method/algorithm, we perform the following tasks: The order of the 4 base-stations is permuted and calculation is performed 24 times. The results are checked against a predefined bounding box ($0 \leq x \leq 10m$, $0 \leq y \leq 5m$, $0 \leq z \leq 4m$). Differences in the propagation delay are calculated for the resulting values and checked against the given values. All the results are counted and the final result is voted by the number or occurrences.

The improvements help to solve the problem for the point $m_2 = (10, 2, 2.5)$, which results in: `10.0000 2.0000 2.5000`. But there are still points which cause problems: The resulting position for $m_3 = (8, 1, 3.5)$ is: `8.2501 0.9320 3.9780`. It turns out that for this result all differences in signal delays are equal to point m_3 (see Listing 8.4). Therefore, there is no way to distinguish these points based on given signal propagation difference measurements.

To further analyse this effect we perform a simulation which calculates the position for every point of a $0.5m$ grid spanning the complete bounding box. Results for this simulation are shown in Figure 4.8. Green circles indicate correctly calculated positions while blue circles indicate wrongly calculated positions. The base-stations are marked with red stars.

Bucher and Misra [BM02] describe in their work that there is the need for either a fifth base-station or some additional condition to get the correct position out of the two results which are given by the algorithm. In the case of GPS they describe the additional condition as checking if the value is in the horizon relative to earth. Therefore, we repeat the grid simulation with 5 base-stations. Calculation is done the same way as before: All time difference measurements are passed to the algorithm using all permutations of the base-stations. The results are shown in Figure 4.9. Green circles indicate correct positions. Base stations are marked with red stars. It turns out that there are no wrongly calculated positions any more.

To check the accuracy for various locations in case of errors for time delay measurements, we perform simulations with gaussian distributed errors. Locations for mobile objects were randomly chosen within the bounding box $m_1 = (8, 3, 1)$, $m_4 = (6, 4.5, 3)$, $m_5 = (0.1, 0.1, 0.1)$, $m_6 = (3.1, 1.5, 1.6)$ and $m_7 = (5, 2, 3)$. Figure 4.10 shows a scatter plot with the resulting positions with $\sigma = 5ms$ ($\sigma \approx 15 \cdot t_{bit}$). The circles indicate the calculated positions. Base stations are marked with red stars and reference positions are marked with blue stars. Deviation from the exact position varies for the different test locations. The scatter plot shows that the most of the calculated positions are correct and match the reference position within a certain accuracy. But in the light orange marked areas there are some purple colored circles which indicate positions that are wrongly calculated.

It turns out that an exact solution of the TDOA using the algorithm of Bucher and Misra [BM02] is not useful if the number of receivers is greater than four or five. Since we use additional receivers to mitigate problems with destructive interference a different algorithm or heuristic is presented for TDOA calculation in Section 4.4.

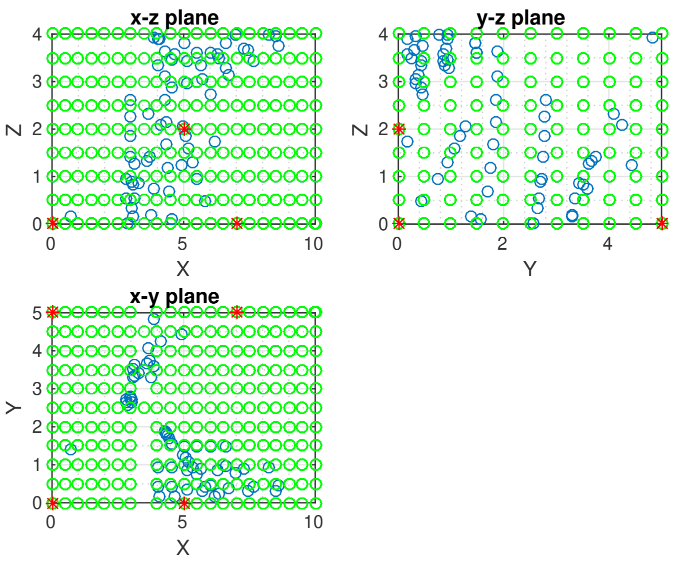


Figure 4.8: TDOA calculations for every point of a 0.5m grid spanning the complete bounding box using 4 base-stations. Green circles indicate correctly calculated positions. Blue circles indicate wrongly calculated positions. The base-stations are marked with red stars.

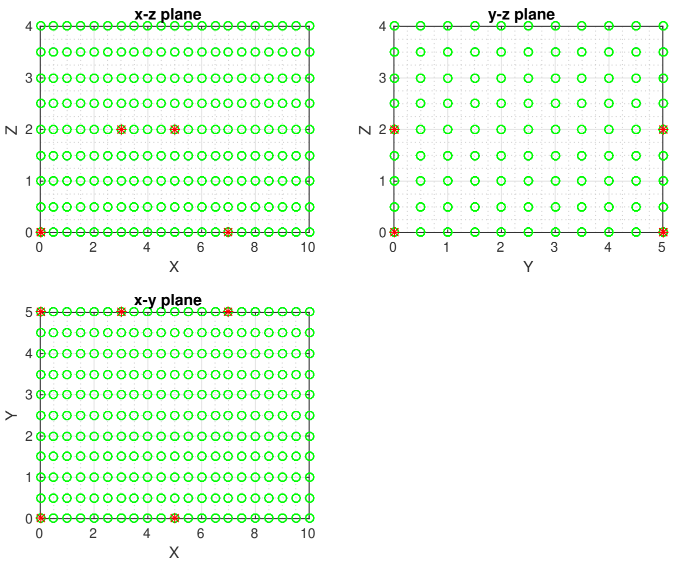


Figure 4.9: TDOA calculations for every point of a 0.5m grid spanning the complete bounding box using 5 base-stations. Green circles indicate correctly calculated positions. The base-stations are marked with red stars. There are no wrongly calculated positions.

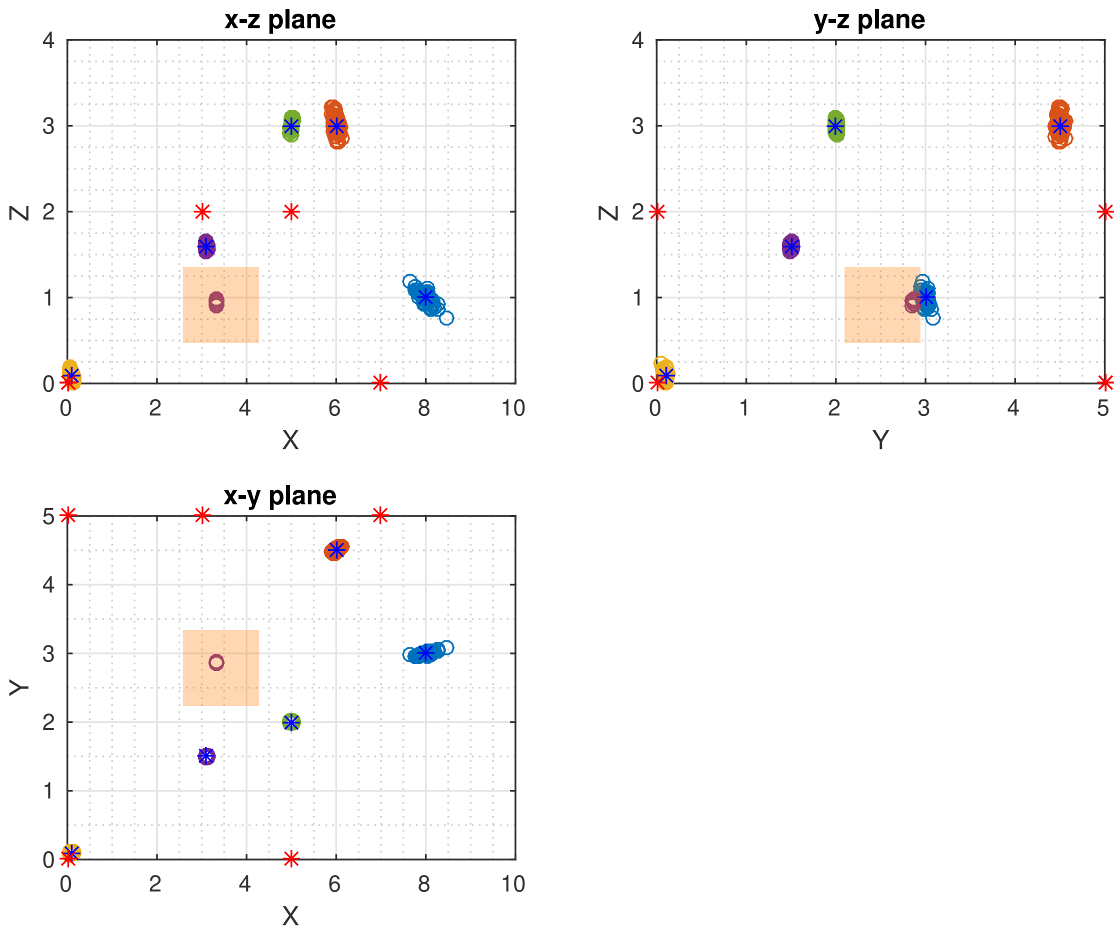


Figure 4.10: TDOA calculations with gaussian distributed errors ($\sigma = 5ms$) using 5 base-stations. The circles indicate the calculated positions. Base stations are marked with red stars. Reference positions are marked with blue stars. Orange marked areas indicate positions that are wrongly calculated.

4.4 Particle Filter

It turns out that an exact solution of the TDOA problem is not practical in our implementation for the following reasons:

1. **Overdetermined system**
   We use additional receivers to mitigate problems with destructive interference. Therefore, we have a much larger number of receivers in our test environment than it is required by the TDOA calculation algorithm presented by Bucher and Misra [BM02]. It would be required to take a subset of the measurements to apply this algorithm. But this contradicts the motivation for adding more receivers.

2. **Noisy values**
   There is an error $\epsilon_{\delta t}$ in the measurements of propagation delay because it is difficult to exactly detect the received burst signals. This error has an influence on the accuracy of the determined position. Depending on the geometrical arrangement of the base-stations and therefore possible glancing intersections (see Section 2.1.2) this may lead to very inaccurate results.

3. **Outliers**
   To mitigate problems with destructive interference additional receivers are added. It can happen that some of the receivers do not detect the transmitted burst properly. The time delay measurements for these receivers should then be regarded as outliers and not be incorporated in the position determination.

Gustafsson and Gunnarsson [GG03] present a static particle filter used for finding the position of the mobile object given TDOA measurements. We subsequently implement and simulate such a filter. As addition to the original algorithm presented by Gustafsson and Gunnarsson we add an additional abort constant I for the case that $\mathbf{\hat{P}_i}$ does not converge. We also formalize the criteria of convergence for $\mathbf{\hat{P}_i}$ which is not stated by Gustafsson and Gunnarsson.

Algorithm 4.1 shows the steps which are necessary to compute the particle filter (for the mathematical model and variable names see Section 3.3).

Algorithm 4.1: Particle filter based on Gustafsson and Gunnarsson [GG03]

```
Input:
$K$: number of particles
$I$: maximum number of iteration steps
$N$: number of base stations
$\mathbf{P_n} = \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}, 0 \leq n < N$: positions of base stations
$x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}$: size of bounded space in x, y, and z-dimension
$\sigma_{jitter}$: jittering constant for particles
$r_{mn}$: measured TDOA values
$v_{sound}$: velocity of sound
$thr$: threshold for $span$ to check if estimated position has converged
Output: determined position $\mathbf{\hat{P}_i}$
1  for $k \leftarrow 0$ to $K$ do
2      $\mathbf{Q_{k,0}} = \begin{pmatrix} x_{k,0} \\ y_{k,0} \\ z_{k,0} \end{pmatrix} = \begin{pmatrix} rand_{uniform}(x_{min}, x_{max}) \\ rand_{uniform}(y_{min}, y_{max}) \\ rand_{uniform}(z_{min}, z_{max}) \end{pmatrix}$;
3  end
4  for $i \leftarrow 1$ to $I$ do
5      for $k \leftarrow 0$ to $K$ do
6          for $n \leftarrow 0$ to $N$ do
7              $d_{k,i,n} = ||\mathbf{Q_{k,i}} - \mathbf{P_n}||$ ;              // euclidean distance
8          end
9          for $n \leftarrow 0$ to $N$ do
10             for $m \leftarrow 0$ to $n$ do
11                 $r_{k,i,mn} = \frac{d_{k,i,m} - d_{k,i,n}}{v_{sound}}$ ;      // TDOA values for particles
12             end
13         end
14         $w_{k,i} = 1 / \sum_{1 \leq n < N, 0 \leq m < n} (r_{k,i,mn} - r_{mn})^2$ ;      // particle weights
15     end
16     $V_i = \sum_{0 \leq k < K} w_{k,i}$;
17     for $k \leftarrow 0$ to $K$ do
18         $v_{k,i} = \frac{w_{k,i}}{V_i}$ ;      // normalize weights s.t. $\sum_{0 \leq k < K} v_{k,i} = 1$
19     end
20     $\mathbf{\hat{P}_i} = \sum_{0 \leq k < K} v_{k,i} \cdot \mathbf{Q_{k,i}}$ ;              // estimated position
21     $span_{x,i} = \max_{0 \leq k < K} x_{k,i} - \min_{0 \leq k < K} x_{k,i}$;
22     $span_{y,i} = \max_{0 \leq k < K} y_{k,i} - \min_{0 \leq k < K} y_{k,i}$;
23     $span_{z,i} = \max_{0 \leq k < K} z_{k,i} - \min_{0 \leq k < K} z_{k,i}$;
24     $span_i = span_{x,i} \cdot span_{y,i} \cdot span_{z,i}$ ;      // quality metric for $\mathbf{\hat{P}_i}$
25 ... continue on next page ...
```

```
... continued from previous page ...
    if $span_i \leq thr$ then
        break for ;            // estimated position has converged
    end
    // resample particles for step $i+1$:
    for $k \leftarrow 0$ to $K$ do
        pick $\mathbf{Q_{k,i+1}}$ from $\mathbf{Q_{j,i}} : 0 \leq j < K$ where the probability to pick a particle is
        proportional to its weight $v_{j,i}$;
    end
    for $k \leftarrow 0$ to $K$ do
        $\mathbf{Q_{k,i+1}} += \frac{\sigma_{jitter}}{i^2} \cdot \begin{pmatrix} rand_{gaussian}(-1,1) \\ rand_{gaussian}(-1,1) \\ rand_{gaussian}(-1,1) \end{pmatrix}$;
    end
end
return $\hat{\mathbf{P}}_\mathbf{i}$
```

The listings of the simulations are attached in appendix (see Listing 8.5 and Listing 8.6). The simulation of this algorithm shows that the quality of position determination heavily depends on the noise of the time measurements. This behaviour is expected since even small errors in time measurements might have a large influence on the position because of the geometric properties of the TDOA equations (see Section 2.1.2).

Figure 4.11 and Figure 4.12 show examples of particle filter simulations. These examples use the same sensor locations as in real-word testing scenarios. All of the sensors are placed at the z-axis in the range from 20 to $90cm$. The particles after convergence are indicated by green circles. These are scattered around the determined position which is shown by a red circle. It turns out that for this sensor placement the deviation for the particles in z-dimension ($span_z$) is significantly larger than in x- and y-dimension ($span_x$ and $span_y$).

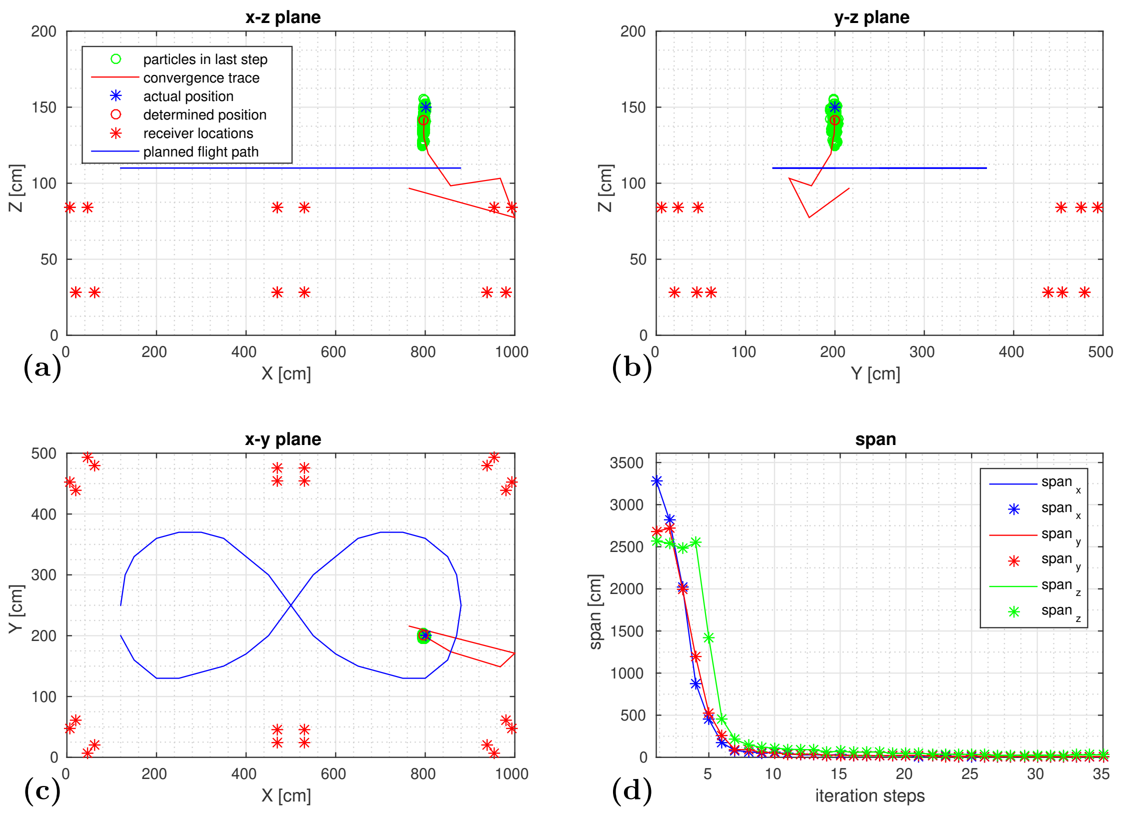


Figure 4.11: Particle filter simulation (simulated TDOA errors: $\sigma = 0.1ms$). Subfigures (a), (b), and (c) show the process of the calculation. The particles after convergence are indicated by green circles and the estimated positions during intermediate iteration steps of the calculation are indicated by a red line. The actual position of the mobile object is marked by a blue star and the determined position is shown by a red circle. The base stations are indicated by red stars, while an example for the planned flight path is shown by a blue line. In subfigure (d) the converge criteria $span_i$ is shown for all iteration steps.

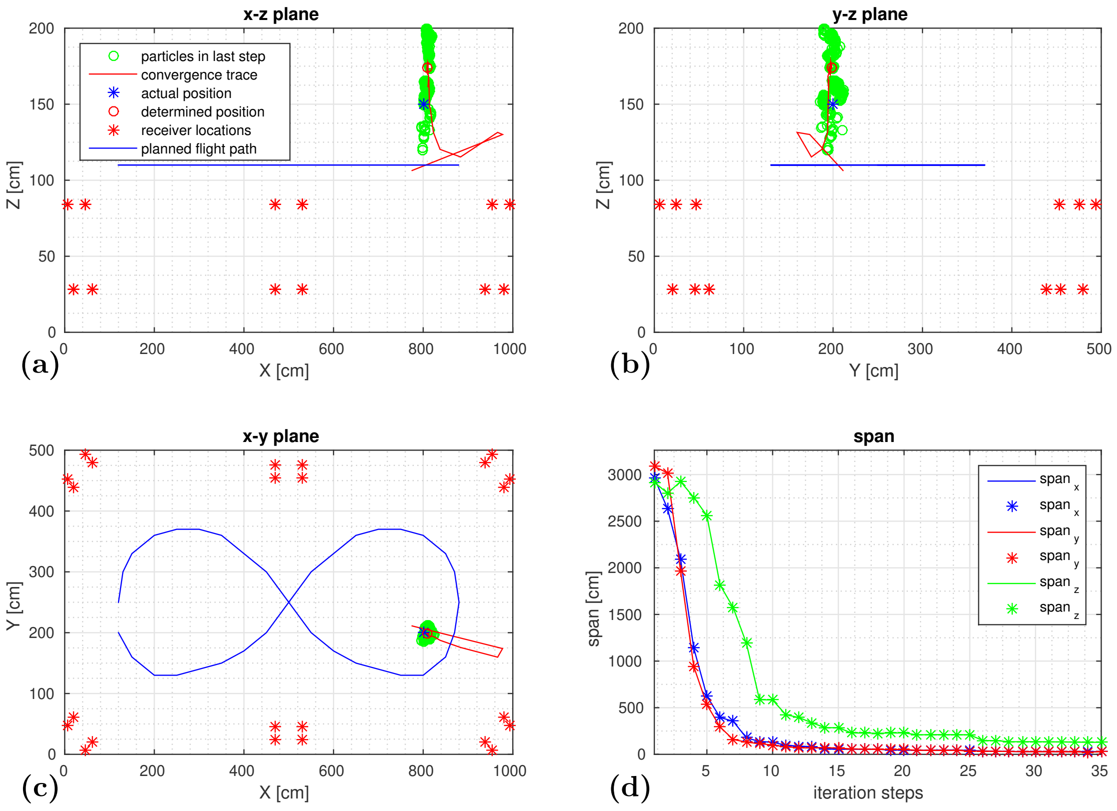


Figure 4.12: Particle filter simulation (simulated TDOA errors: $\sigma = 0.5ms$). Subfigures (a), (b), and (c) show the process of the calculation. The particles after convergence are indicated by green circles and the estimated positions during intermediate iteration steps of the calculation are indicated by a red line. The actual position of the mobile object is marked by a blue star and the determined position is shown by a red circle. The base stations are indicated by red stars, while an example for the planned flight path is shown by a blue line. In subfigure (d) the converge criteria $span_i$ is shown for all iteration steps.

CHAPTER 5

Implementation

This chapter shows the implementation which is done in hard- and software to demonstrate the functionality of the system and to verify the desired quality metrics. Figure 5.1 shows an block diagram as overview of the implementation.

5.1 Hardware Implementation

The first test circuit is built to examine the physical properties of the ultrasonic signal propagation. Therefore, it consists of a test signal generator with an ultrasonic transmitter and an ultrasonic receiver with amplifier and filter. Digital signal processing is performed using a FPGA together with an ADC for signal sampling. For later versions of the hardware the test circuits are adapted and improved.

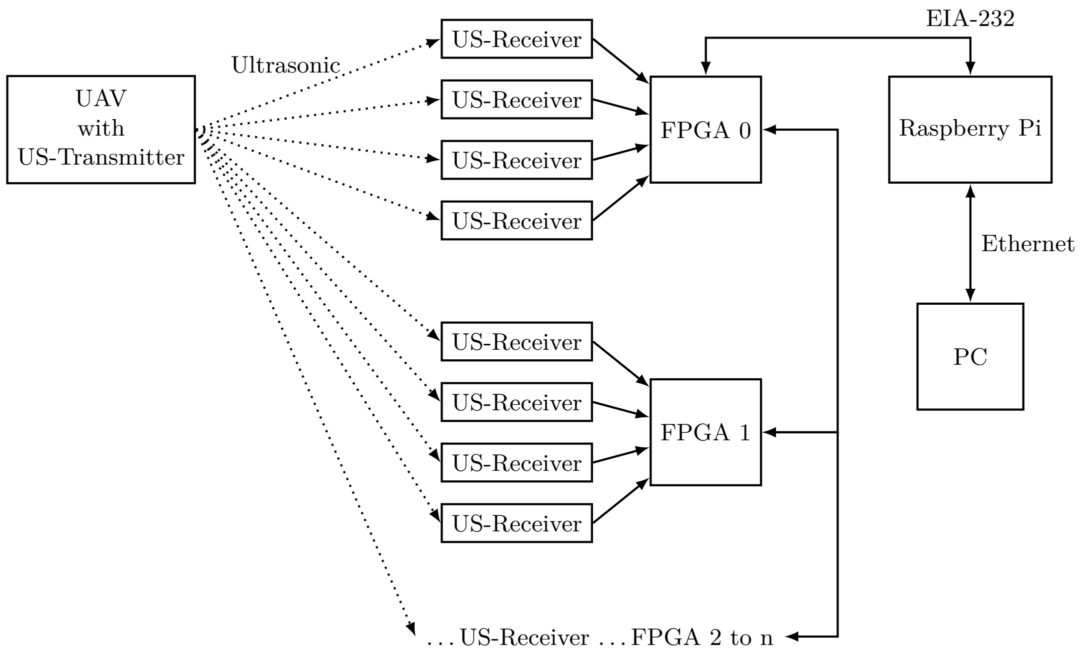


Figure 5.1: Block diagram as overview of implementation. A transmitter which is placed on the UAV sends ultrasonic signals. Every base-station is equipped with a FPGA-Board and four ultrasonic receivers in order to capture and process the signals. All FPGA-Boards are connected by a custom implemented network interface for time synchronisation and data exchange. A Raspberry Pi acts as gateway in order to connect a PC for further signal processing and TDOA calculation.

5.1.1 Transmitter Circuit

The transmitter circuit is built around an *Atmel ATtiny841* microcontroller. It is operated at $f_{uC} = 8MHz$ using the internal oscillator. The ultrasonic carrier frequency as well as the encoded signal is generated by the microcontroller. A timer is configured s.t. a pin toggles with an period length of $t_{USS} = 40.96\mu s$, which represents the $f_{USS} = 24.41kHz$ carrier frequency. This carrier can be modulated in an arbitrary way to generate the final test signal.

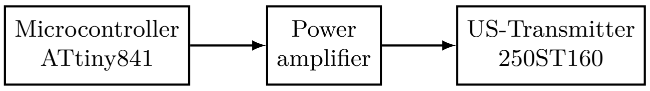


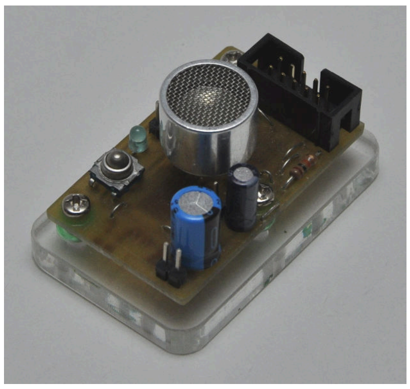
Figure 5.2: Transmitter circuit block diagram. It consists of a *Atmel ATtiny841* microcontroller which generates the test signal. This signal is fed into the power amplifier and then into the the ultrasonic transmitter.

Before the signal is fed into the ultrasonic transmitter *250ST160* it is amplified by a *BD6232* H-Bridge power amplifier to get $10V_{PP}$ maximum. A micro-USB-plug is used on the transmitter module to power the device by an USB-Power-Bank. Figure 5.3a shows a picture of the first version of the transmitter module. The corresponding schematic and test point documentation are included in appendix.

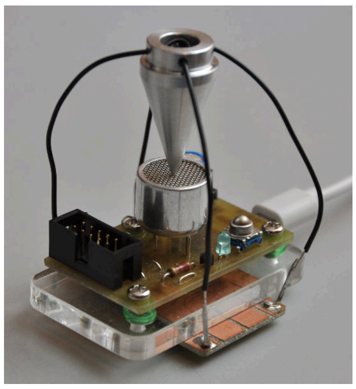
Since the beam angle of the ultrasonic transmitter is 85° for *250ST160* [Pro15] it is needed to incorporate additional means for a more omnidirectional emission of the signal. Bjerknes et al. [Bje+07] use a reflector cone to improve the directional pattern. Figure 5.3b shows the test module with an attached reflector cone. However, the cone did not lead to a omnidirectional emission but rather to a pattern which was limited to the form of a torus around the transmitter.

Revised Transmitter Circuits

We design a revised version of the transmitter circuit with four transmitters. A quartz is added to this version and used as clock source for the microcontroller. This quartz provides a higher accuracy for the operating frequency than the internal oscillator of the microcontroller. The ultrasonic transmitters are bent by 40° and facing outwards in four directions. This angle is chosen, s.t. the specified beam with 85° of the *250ST160* covers the half of a sphere. The circuit is built as stacked module s.t. the transmitters can be removed from the main PCB. This design is used because it is possible to add an extra PCB (e.g., additional power amplifier) in between. To transmit more power via the ultrasonic transmitters a higher amplitude is needed. Therefore, we add a voltage doubler *ADM660* to the improved design. While keeping the operating voltage at $VCC = 5V$ this makes it possible to operate the H-Bridge power amplifier at a voltage of $10V$ which results in $20V_{PP}$ at the ultrasonic transmitters. The H-Bridge is replaced by the power amplifier *B57928* (*TLE4202B* compatible version) which contains two independent power comparators.



(a) Transmitter module consisting of a microcontroller, H-Bridge power amplifier and ultrasonic transmitter.



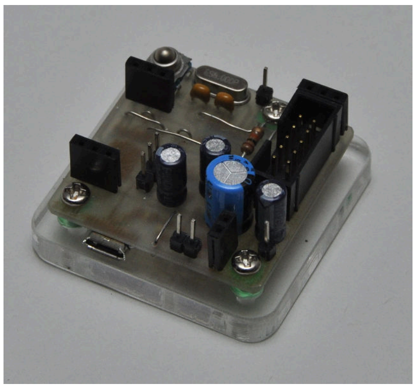
(b) Transmitter module with additional reflector cone attached on top of the ultrasonic transmitter.

Figure 5.3: First version of transmitter module (v1.0).

A picture of the second version of the transmitter module is shown in Figure 5.4a and Figure 5.4b. The corresponding schematic and test point documentation are included in appendix.

Using more than one ultrasonic transmitters results in the problem of interference. There are some angles where the distance of the transmitters is the half of the wavelength which causes destructive interference. This problem is mitigated by using four receivers for each base-station (see Figure 5.12) to minimize the risk of interference for all receivers.

To use the transmitter module on the UAV a thinner version is built with directly attached ultrasonic transmitters. The corresponding schematic and test point documentation are included in appendix. This version also allows stacking of another PCB to add four more transmitters (see Figure 5.5b).


(a) Transmitter module consisting of a microcontroller, a voltage doubler and H-Bridge power amplifier (without ultrasonic transmitters).

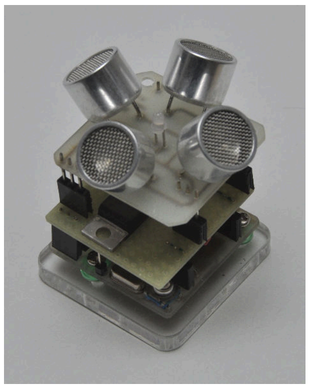
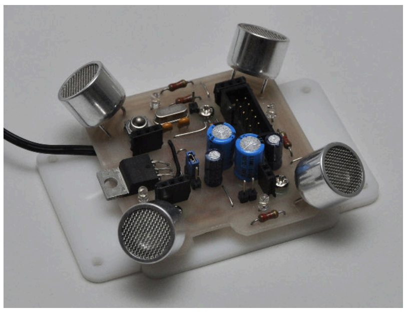

(b) Transmitter module with additional stacked PCBs. These PCBs contain an additional H-Bridge power amplifier and four ultrasonic transmitters.

Figure 5.4: Second version of transmitter module (v2.0).


(a) Transmitter module consisting of a microcontroller, a voltage doubler, H-Bridge power amplifier and four ultrasonic transmitters.

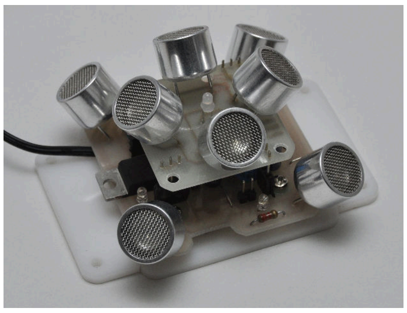

(b) Transmitter module with stacked PCB containing four additional ultrasonic transmitters.

Figure 5.5: Third version of transmitter module to be used on the UAV (v2.2).

5.1.2 Receiver Circuit

The receiver circuit consists of the ultrasonic receiver *250SR160*, two RC band-pass filters with amplification and resistors to adapt the signal level for the used ADC (see Figure 5.6). A picture of the first version of the receiver module is shown in Figure 5.7. The corresponding schematic and test point documentation are included in appendix. This receiver module is used to verify the intended behaviour of the amplifier and filter design.

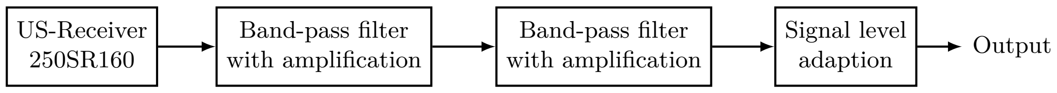

US-Receiver 250SR160
Band-pass filter with amplification
Band-pass filter with amplification
Signal level adaption
Output

Figure 5.6: Receiver circuit block diagram. It consists of the ultrasonic receiver, two RC band-pass filters with amplification and resistors to adapt the signal level on the output.

Amplifier and Filter

In order to sample the signal from ultrasonic receivers there is an amplifier needed since the signal level would otherwise be to too low for the ADC. Furthermore, the signal should be filtered s.t. the ultrasonic carrier frequency is amplified and other frequencies are suppressed. To design and simulate the amplifier and filter we use *LTspice IV* [Lin15]. It is implemented as two first-order active RC band-pass filters. The schematic of the filter is shown in Figure 5.8. Figure 5.9 shows its corresponding bode diagram. The filter is designed s.t. the center frequency equals $f_{filter} = 24kHz$. This is equal to the rated frequency for the used ultrasonic receivers.

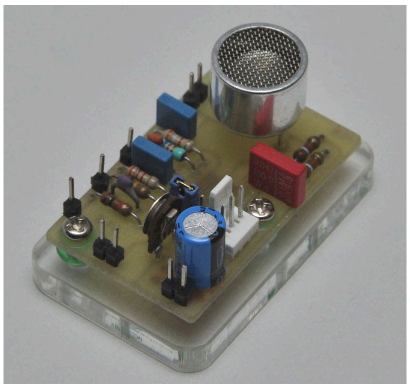


Figure 5.7: First version of receiver module (v1.0). It consists of the ultrasonic receiver, two RC band-pass filters with amplification and resistors to adapt the signal level on the output.

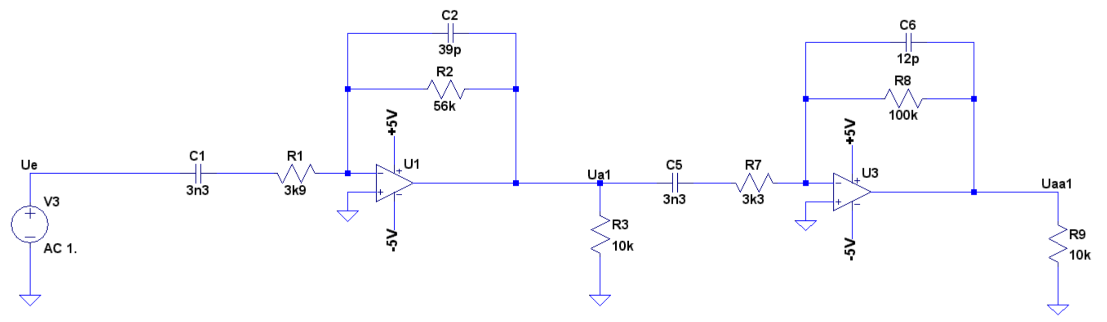


Figure 5.8: Schematic of amplifier and filter circuit. It is implemented as two first-order active RC band-pass filters.

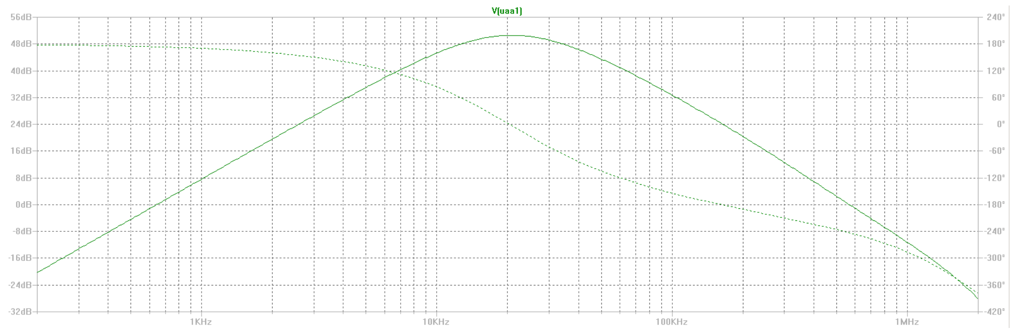


Figure 5.9: Bode-diagram of amplifier and filter circuit. The filter is designed s.t. the center frequency equals $f_{filter} = 24kHz$. Since it is implemented as two first-order active RC band-pass filters power decreases by 40 dB per decade.

Revised Receiver Circuit

For subsequent implementation we design a smaller version using only SMD components. The complete PCB is covered with a heat shrinking tube to protect against short-circuits and mechanical damages. Figure 5.10 shows two of the improved modules, one of them before assembling the heat shrinking tube. The corresponding schematic is included in appendix.

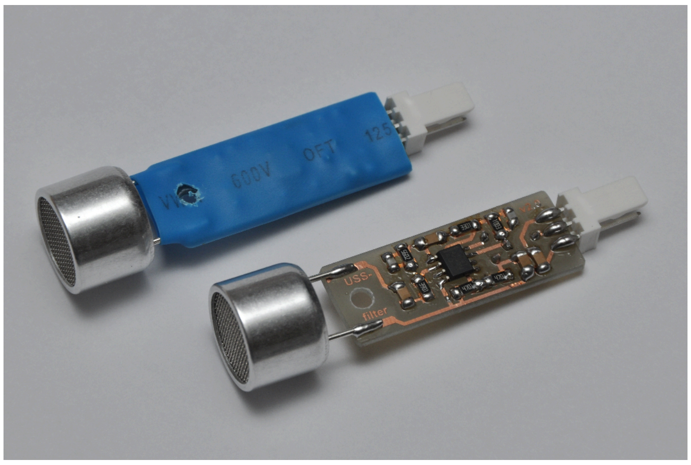
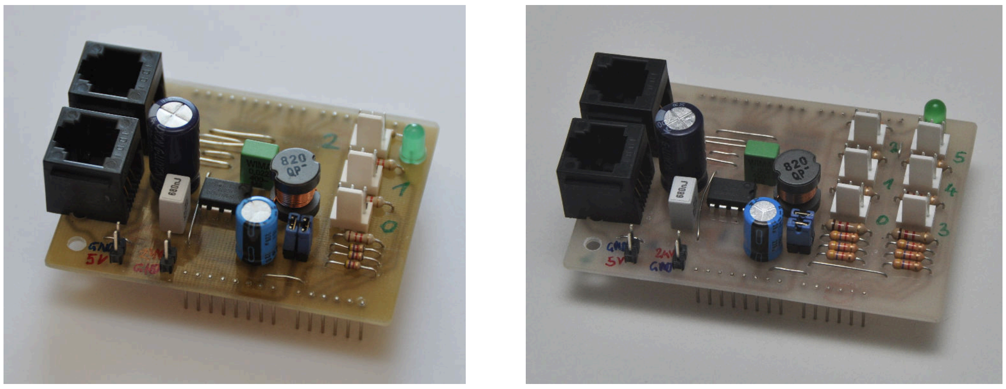


Figure 5.10: Second version of receiver module (v2.0). It consists of the ultrasonic receiver and two RC band-pass filters with amplification. The receiver modules are covered with a heat shrinking tube for protection. The right module is shown before assembling the heat shrinking tube s.t. the SMD components are visible.

5.1.3 FPGA and Adapter-Board

The *DE0-Nano-SoC Development and Education Board* is used to sample and process the signals. This FPGA-Board consists of an *Altera Cyclone® V SE 5CSEMA4U23C6N* FPGA and among other peripherals also a *Linear Technology LTC2308* low noise 8-Channel, 12-bit ADC. An Adapter-Board is built to power the FPGA-Boards and to connect them over a distance of at least $s_{max} = 10m$ (see Section 3.1.4). Since the *DE0-Nano-SoC Development and Education Board* contains pin headers on the top layer of the PCB it is easy to build a custom PCB which can be stacked onto the FPGA-Board.

To exchange data among the FPGAs we use a *EIA-485* interface. The *EIA-485* standard is chosen because its industry-proven reliability and resistance against interference. Two transceiver ICs *ISL83491* are used on each Adapter-Board in order to operate two completely independent network channels. One of them is solely used for time synchronization. The other one is used for data exchange between the FPGAs. All modules are connected to the same power supply. A step-down converter *LM2672N-5.0* on the Adapter-Board provides the $5V$ input for the FPGA-Board. This makes it possible to power the system with up to $24V$. As physical connection we use RJ45-jacks since cables with RJ45-plugs are cheap and widely available. The first version of the Adapter-Board allows the connection of up to three receivers to the analog channels of the ADC. A second improved version is built later to connect up to six receivers. A picture of the first version (v1.0) of the Adapter-Board is shown in Figure 5.11a and a picture of the second version (v1.1) is shown in Figure 5.11b. The corresponding schematics and test point documentation are included in appendix.



(a) First version (v1.0) of the Adapter-Board. It contains three jacks for analog channels of the ADC.

(b) Second version (v1.1) of the Adapter-Board. It contains six jacks for analog channels of the ADC.

Figure 5.11: Two different versions of the Adapter-Board. Both versions contain two RJ45-jacks and transceiver ICs for *EIA-485* connections as well as a step-down converter to provide $5V$ input for the FPGA-Board.

5.1.4 Mounting Platform

A mounting platform is built where the FPGA-Board is placed in the center and the four receivers on X-shaped booms. The mounting platform is built using mainly lasercut plywood. The booms are built using thin aluminium strips in order to reduce the weight of the whole platform. Figure 5.12 shows the dimensions of this mounting platform and the positions of the four receivers (red circles) relative to the reference point (grey circle). Figure 5.13 shows one of the mounting platforms.

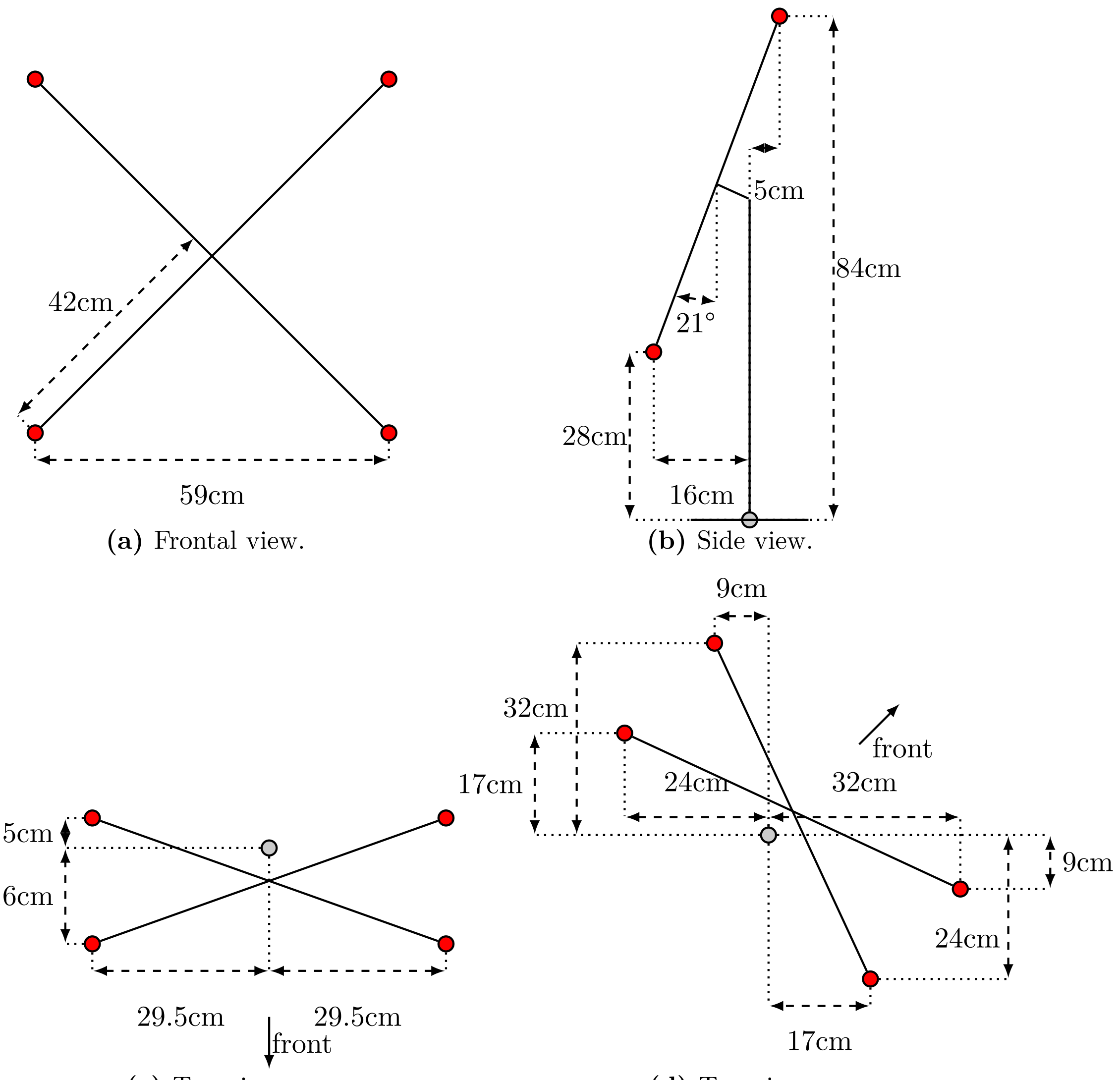


Figure 5.12: Drawings indicating the dimensions of the mounting platform for the FPGA-Board and four receivers on X-shaped booms. The positions of the four receivers are indicated by red circles and the the reference point is indicated by a grey circle. The mounting platform is tilted by 21° in drawing (b) and (c). The mounting platform is tilted by 21° and rotated by 45° in drawing (d).

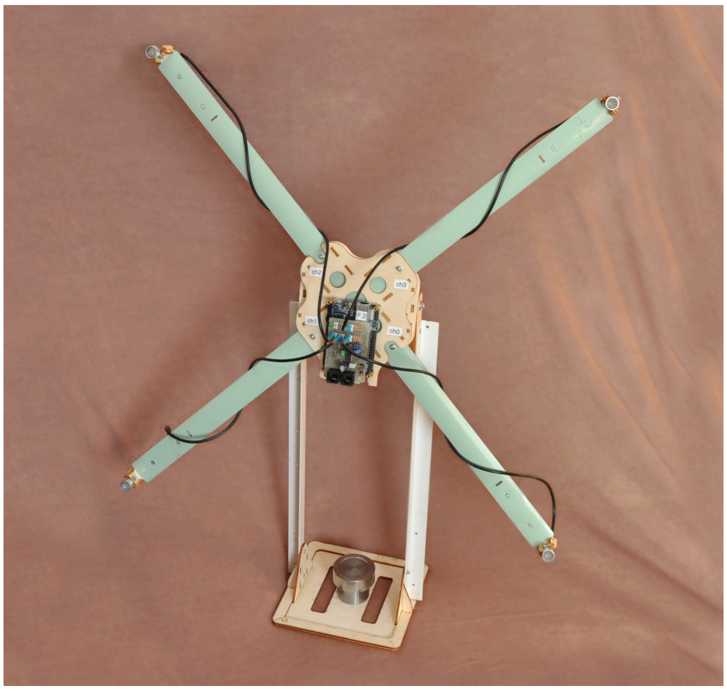


Figure 5.13: Picture of the mounting platform for the FPGA-Board and four receivers on X-shaped booms. The mounting platform is built using mainly plywood and the booms are built using thin aluminium strips.

5.2 FPGA Implementation

All time-critical parts of the system are implemented in the FPGAs. This includes the interface to the ADC, signal processing such as correlation and/or burst detection as well as real-time communication of the detected time instants. Further calculation such as determination of the position based on the measured time instants could also be performed on the FPGA in the future but is done on a different platform for now.

5.2.1 Time Synchronization

For the implementation of our LPS we use one FPGA-Board for each base station. In order to exchange time measurements a synchronized timebase is needed. Therefore, a counter is used to provide timestamps which can be exchanged among the components of the system. This counter is derived from the system clock and counts with ticks of

$$t_{syscnt} = \frac{1}{f_{sysclk}/64} = \frac{1}{50MHz/64} = \frac{1}{781,25kHz} = 1.28\mu s.$$

It is chosen so that the clock counts faster than the sample time of the ADC (which is $t_{sample} = 5.12\mu s$). This ensures that there is no additional inaccuracy caused by the resolution of time measurement.

The counter is implemented as 40 Bit number which makes it possible to count up to $n_{max} = 2^{40} - 1 \approx 1,1 \cdot 10^{12}$. This results in a maximum operating time (limited by a counter overrun) of $t_{operation} = n_{max} \cdot t_{syscnt} \approx 390h \approx 16$ days.

Time Synchronization Module

The counter needs to be synchronized among all base stations. Therefore a distinct cable connection is used on which the master FPGA transmits the current timestamp approx. 100 times per second and all other FPGAs use this signal to synchronize the local counter.

Figure 5.14 shows the principle of this time synchronization module. A quartz is connected on every FPGA-Board which works at nominal frequency of $f_{sysclk} = 50MHz$. This frequency is fed into the prescaler which divides the clock by factor 64. The prescaler feeds the local counter which counts with nominal ticks of $t_{syscnt} = 1.28\mu s$. The output of local counter `syscnt` is the timebase which can be used by all other modules within the FPGA. This value of the counter is transmitted by the master. On all other FPGAs a receiver is triggered on the first edge of the data packet and compares the local counter value to the received value. If the difference exceeds a threshold a reset of all other modules in the FPGA is triggered and the local counter is set to the received value. Otherwise, the difference is used to derive the deviation between the oszillators and the needed adjustment for the local counter. On the slave FPGAs the clock division factor can be modified (depending on +/− input) to incorporate these adjustments. Like in the master FPGA the output of the local counter `syscnt` is the timebase for all other modules within the FPGA.

The local counter value is not directly influenced by the synchronization but counting speed is adjusted (using the `+/-` input of the prescaler). This design ensures that the values for `syscnt` are always in order. There are no duplicated or missing values as long as no reset happens.

Counter Transmission Signal

The value of the counter is transmitted by the master using Non-Return-to-Zero coding with $81.92\mu s$ per bit. Since the correct transmission of the counter value is very critical (incorrect values could trigger a reset of the FPGAs) a 16 Bit CRC [Sta10] is used. Every transmitted packet consists of 50 bits (see Table 5.1). For encoding the counter value (40 bit counter) 27 bits are sufficient since the 13 least significant bits are zero for every transmission. The first bit is zero in every packet in order to generate the first edge which is used by the receivers to trigger the packet. Figure 5.15 shows a packet (blue signal) together with a debug signal (yellow signal). The debug signal is an internal trigger that starts the transmission of the packet.

Test of Synchronization

The FPGA operates with a system clock of $f_{sysclk} = 50MHz$. To get an estimation for the expected deviation for the clock frequencies for two FPGA-Boards we examine the typical properties for quartzes. Manufacturer *Abracon* lists a wide range of different quartzes on its website [Abr16]. The quartz with the highest tolerance is specified with a stability of $\pm 100ppm$ and a tolerance of $\pm 50ppm$. To be an the save side we assume a deviation of $\pm 1000ppm$ which results in $f_{min} = 50MHz \cdot 0.999 = 49.95MHz$ and $f_{max} = 50MHz \cdot 1.001 = 50.05MHz$. These frequencies are tested on a single FPGA using two PLLs to generate the specified frequency (see Figure 5.16).

- **Case f_{min}:**
  Desired frequency is $f_{min} = 49.95MHz$. The configured PLLs result in frequencies $f_{PLL1} = 50.2MHz$ and $f_{PLL2} = 49.949MHz$. Which gives an time offset (measured with oscilloscope) of $\delta t_{min_1} = +19\mu s$ to $\delta t_{min_2} = +21\mu s$. This time offset would result in an error of $\delta s_{min_1} = v_{sound} \cdot \delta t_{min_1} = 340m/s \cdot 19\mu s \approx 0.6cm$ to $\delta s_{min_2} = v_{sound} \cdot \delta t_{min_2} = 340m/s \cdot 21\mu s \approx 0.7cm$ for TDOA measurements. Requirements for accuracy (see Section 3.1.4) are satisfied since $\delta s_{min_1} < s_{accuracy}$ and $\delta s_{min_2} < s_{accuracy}$.

- **Case f_{max}:**
  Desired frequency is $f_{max} = 50.05MHz$. The configured PLLs result in frequencies $f_{PLL1} = 49,8MHz$ and $f_{PLL2} = 50.049MHz$. Which gives an time offset (measured with oscilloscope) of $\delta t_{max} = -83\mu s$. This time offset would result in an error of $\delta s_{max} = v_{sound} \cdot \delta t_{max} = 340m/s \cdot 83\mu s \approx 2.8cm$ for TDOA measurements. Requirements for accuracy (see Section 3.1.4) are satisfied since $\delta s_{max} < s_{accuracy}$.

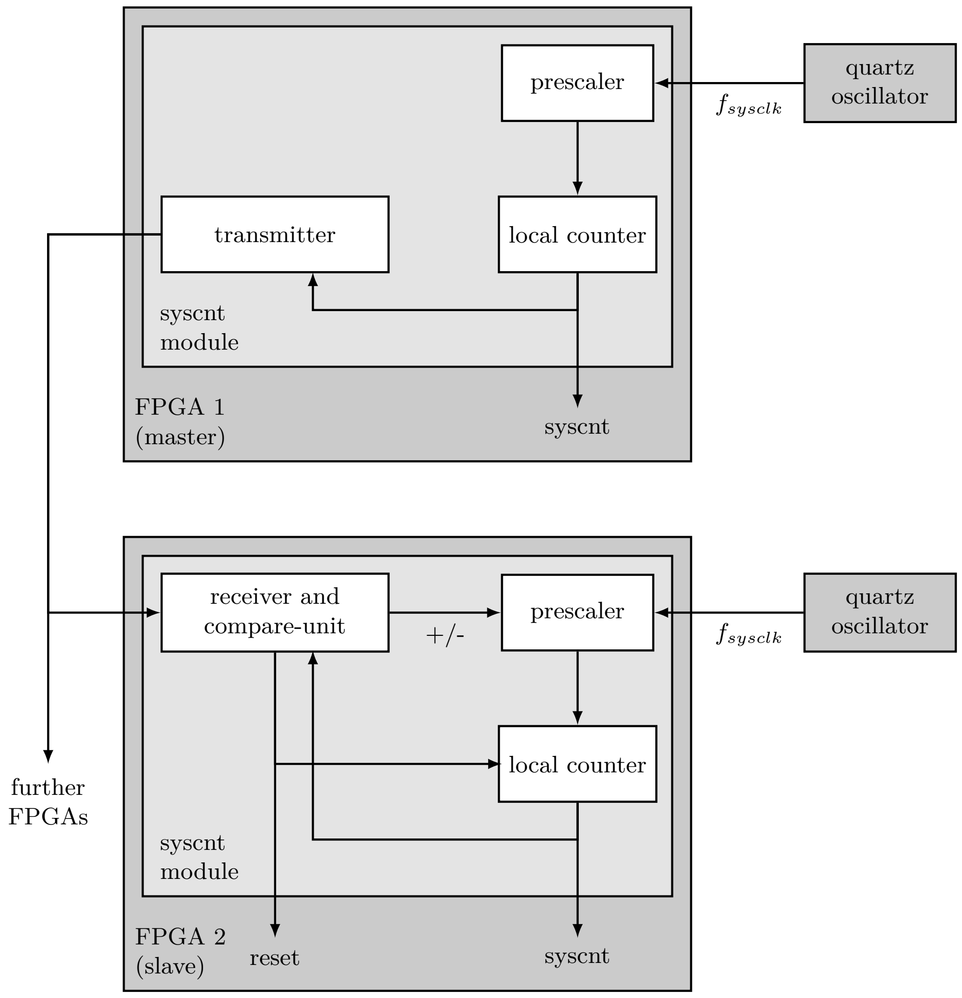


Figure 5.14: Time synchronization module principle: A quartz is connected on every FPGA-Board. The frequency of this quartz is fed into the prescaler. The prescaler drives the local counter `syscnt` which is the timebase used by all other modules within the FPGA. The value of the counter is transmitted by the master. All other FPGAs compare the local counter value to the received value and adjust the prescaler (+/− input) or execute a reset in case the difference exceeds a threshold.

Bit number	Content	Note
0	constant 0	used to generate the first edge
1	counter value bit 39	most significant bit of counter value
2	counter value bit 38	
...	...	
37	counter value bit 13	
38	CRC bit 15	
39	CRC bit 14	
...	...	
49	CRC bit 0	

Table 5.1: Packet for transmitting system counter consisting of a constant first bit in order to generate the first edge, the counter value and a CRC.

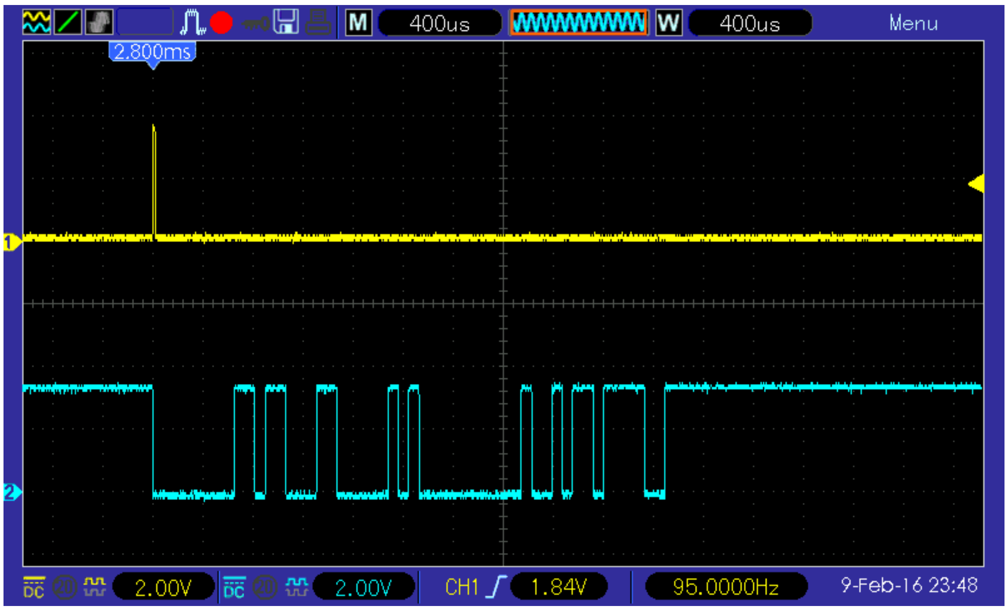


Figure 5.15: Example of a transmitted packet for time synchronization. The blue signal shows the packet which is encoded using Non-Return-to-Zero. The yellow signal shows a debug signal which is an internal trigger that starts the transmission of the packet.

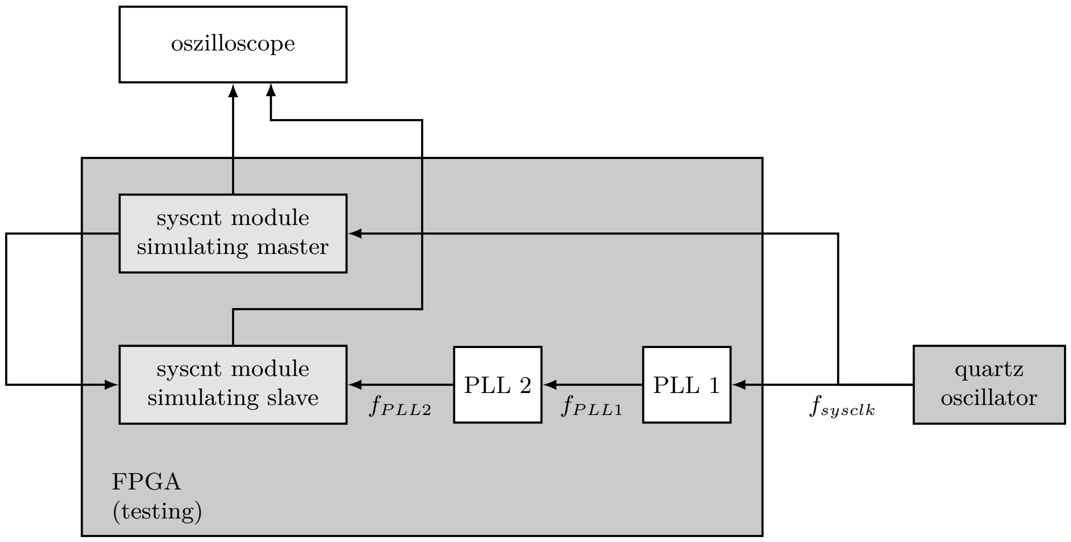


Figure 5.16: Time synchronization test principle. The module simulating the master is directly fed by the quartz oscillator and the module simulating the slave is fed by a modified frequency. This modified frequency is generated by two consecutive PLLs. A debug signal of both syscnt modules is connected to the oscilloscope s.t. the time offset can be measured.

5.2.2 User-defined FPGA Network

Based on a *EIA-485* physical link we develop a protocol for data exchange between the FPGAs. It is capable of handling up to 16 devices which may send up to 256 bits of data per transmission cycle. Bus arbitration is done by a time-division approach which is based on the synchronized timebase.

Based on the 40 Bit system counter `syscnt` the timebase is sliced into 16 slots with 512 ticks of `syscnt` each. This makes it easy to determine the current sending device by extracting bits 12 to 9 of the 40 bit counter. The duration for each bit using Non-Return-to-Zero coding on the bus is $t_{bit} = \frac{1}{f_{sysclk}/64} = \frac{1}{50MHz/64} = \frac{1}{781{,}25kHz} = 1,28\mu s$ (which is the same as t_{sysclk}). Each timeslot is capable of 512 bits but not the whole timeslot is used for transmission. Every transmitted packet consists of 256 data bits and a 16 bit CRC [Sta10] to ensure data integrity in case of errors on the physical channel (details see Table 5.2).

For the whole network the bitrate can be calculated as follows:

$$\frac{1}{t_{bit}} \cdot \frac{256}{512} = \frac{1}{1,28\mu s} \cdot \frac{1}{2} = \frac{781,25kbit/s}{2} \approx 390kbit/s.$$

This results in $\frac{390kbit/s}{16} \approx 24kbit/s$ for each FPGA.

Network module

To perform network communication a distinct module is implemented in the FPGA which deals with bus arbitration based on system counter `syscnt`, transmission timing, CRC generation and error detection. Figure 5.17 shows an overview of this network module. Each FPGA sends data which is represented as internal signals as well as button inputs in the respective timeslot. This data is mainly calculated by the signal processing unit and contains, for example, the instant of received ultrasonic signals.

An auxiliary sending unit is available to forward data from a PC to the FPGA network. This unit is only enabled at exactly one FPGA-Board using an external jumper input. The auxiliary sending unit uses the fixed timeslot with ID `1111` (= 15). Data from the PC contains configuration parameters and various control information.

The receiver unit decodes all packages from the bus, checks the CRC and processes the payload. All data, except for transmitter ID `1111` (= 15), is stored in an internal RAM. This means that data from all senders is available at all FPGAs at any time for further processing. Data which was forwarded from the PC to the network is further parsed to filter out the configuration parameters and control information for the respective FPGA and then provided to other modules using internal signals.

Bit number	Content	Note
0	constant `0`	used to generate the first edge
1	data bit 256	sender FPGA ID bit 3 = most significant bit of packet
2	data bit 255	sender FPGA ID bit 2
3	data bit 254	sender FPGA ID bit 1
4	data bit 253	sender FPGA ID bit 0
5	data bit 252	begin of other payload
6	data bit 251	
...	...	
256	data bit 0	
257	CRC bit 15	
258	CRC bit 14	
...	...	
272	CRC bit 0	
273	constant `1`	set bus into idle state
274	constant `1`	set bus into idle state

Table 5.2: Packet for network transmission consisting of a constant first bit in order to generate the first edge, the FPGA ID of the sender, payload, a CRC and constant trailer bits to set bus into idle state.

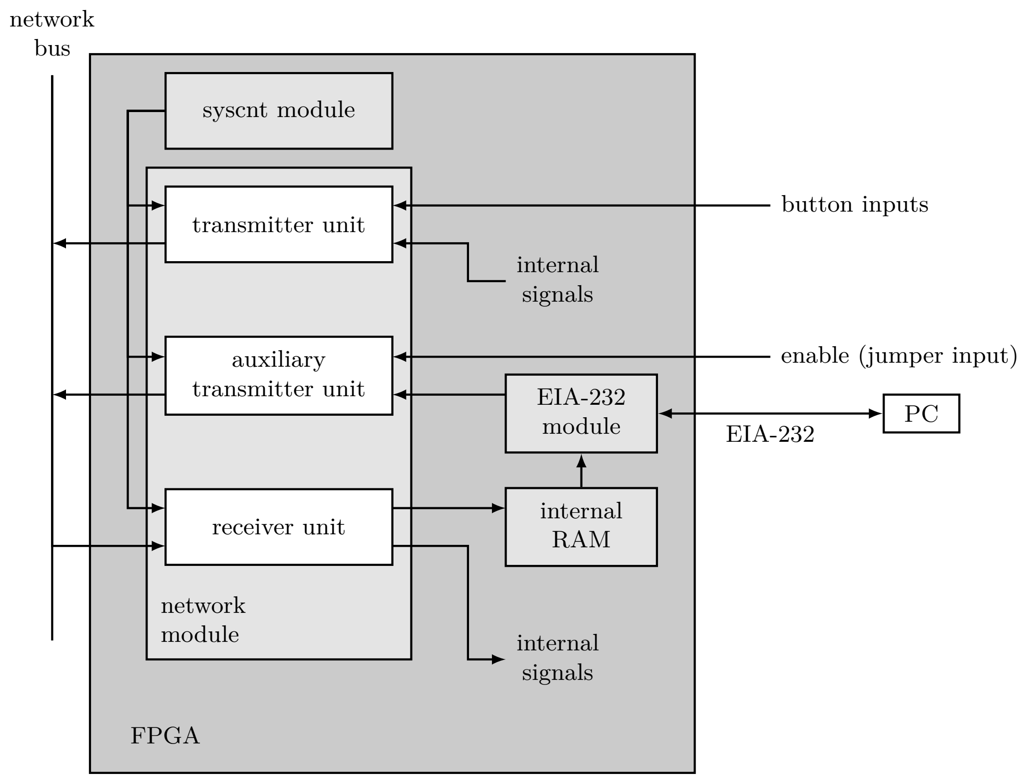


Figure 5.17: Overview of the network module. The transmitter unit deals with bus arbitration (based on system counter `syscnt`) and sends data which is provided as internal signals. An auxiliary sending unit is available to forward data from a PC to the FPGA network. The receiver unit decodes packages from the bus and stores the data depending on the type of the package in an internal RAM or provides it as internal signal.

5.2.3 EIA-232 Interface

To exchange data between a FPGA-Board and a PC a EIA-232 module is implemented. Low level processing of the EIA-232 signals is done using the IP-Core *rs232_with_buffer_and_wb* by Tobias Jeppe [Jep13]. This IP-Core is published on the *OpenCores*-webpage under the GNU Lesser General Public License (LGPL). We implement a custom module which utilizes this IP-Core and transmits data according to the following packet scheme.

The EIA-232 port is operated with 8 data bits, odd parity, 1 stop bit and 115200 baud. The most significant bit is used to distinguish control information and actual data. Every packet starts with a status byte which has set the most significant bit to `1`. A fixed number of bytes follow (see Table 5.3) and the last byte represents a constant trailer. If debug information is transmitted the package is initiated with a different start byte. Then 2 bytes of debug information (see Table 5.4) follow and the last byte also represents a constant trailer. The configuration data which is sent from the PC to the FPGA is transmitted in a similar way (see Table 5.5).

Byte number	Content (in binary format)	Note
0	`1000 SSSS`	`S` = status
1	`0000 FFFF`	`F` = FPGA ID n
2	`0000 0000`	reserved for future use
3	`0000 0000`	...
4	`0000 0001`	
5	`0000 0000`	
6	`0000 0000`	
7	`0000 0010`	
8	`000P PPPP`	`P` = payload from FPGA n
9	`0PPP PPPP`	...
10	`0PPP PPPP`	
11	`0PPP PPPP`	
12	`0PPP PPPP`	
13	`0PPP PPPP`	
14	`1111 1110`	constant trailer

Table 5.3: Packet for EIA-232 transmission to PC consisting of status byte, the FPGA ID of the original sender, payload and constant trailer to indicate the end of the packet.

Byte number	Content (in binary format)	Note
0	`1101 SSSS`	`S` = status
1	`0DDD DDDD`	`D` = debug payload
2	`0DDD DDDD`	...
3	`1111 1110`	constant trailer

Table 5.4: Packet for EIA-232 transmission of debug information to PC consisting of status byte, debug payload and constant trailer to indicate the end of the packet.

Byte number	Content (in binary format)	Note
0	`1000 FFFF`	`F` = FPGA number n
1	`0CCC CCCC`	`C` = configuration for FPGA n
2	`0CCC CCCC`	...
...	...	
36	`0CCC CCCC`	
14	`1111 1111`	constant trailer

Table 5.5: Packet for EIA-232 transmission from PC consisting of the FPGA ID of the target, configuration data for this FPGA and constant trailer to indicate the end of the packet.

5.2.4 ADC Interface

The ADC Interface module polls the ADC via the Serial Peripheral Interface (SPI) connection and provides the values of all n ADC-channels as internal signals for further processing. It is strictly driven by a time-schedule which depends on an internal counter which is derived from the global clock. Using this time-schedule the channel configuration is sent and sampling data is read continuously according to the protocol specified by the ADC manufacturer.

A readout of one channel needs $n_{1ch} = 2^7 = 128$ clock cycles. The clock of the SPI is operated at $f_{adcclkSPI} = \frac{f_{sysclk}}{2} = 25MHz$. Reading four channels results in $n_{4ch} = 4 \cdot n_{1ch} = 512$ clock cycles per conversion. This gives a sampling frequency of $f_{sample4ch} = \frac{f_{sysclk}}{n_{4ch}} \approx 97.7kHz$.

Using only two channels would result in $n_{2ch} = 2 \cdot n_{1ch} = 256$ clock cycles per conversion and therefore a sampling frequency of $f_{sample2ch} = \frac{f_{sysclk}}{n_{2ch}} \approx 195.3kHz$ can be reached.

5.2.5 Signal Processing

High speed signal processing is entirely implemented in the FPGA. First we implement the correlation to detect the modulated signal. After changing to the relaxed signal (see Section 3.4.2) we also implement the burst detector in the FPGA.

Correlation

For each channel of the ADC the read value is stored in an internal dual-port RAM. This RAM provides the storage for the necessary values for the correlation. The correlation itself is partially parallelized to speed up the calculation. Figure 5.18 shows the block diagram of the correlation module. The correlation unit (*corr_unit*) is started by a trigger signal (*start*) and provides the maximum (position and value) of the correlation function as a result (*match_pos*, *match_val*). Internally the parallelization is done by submodules (*G_proc*) which are present n_{par} times. Given the following function (see Section 4.1.1):

$$R^*(m) = \sum_{n=0}^{N-1} f_{mult}(sig(n), ref((m+n)\ mod\ M)); 0 \leq m < M,$$

the n_{par} instances of *G_proc* implement the following function:

$$\sum_{n=0}^{N-1} f_{mult}(sig(n), ref((m+n+i)\ mod\ M)); 0 \leq i < n_{par}.$$

The function $f_{mult}(s, r)$ is implemented in the module *ref_add*, and the function $ref(k)$ is implemented in the module *ref*.

Calculation is performed by the state machine which is shown in Figure 5.19. In state *do corr.* the actual calculation is performed. The values $sig(n)$ are read from the RAM (by iteration of *ram_index*) and n_{par} values for $R^*(m)$ are generated. Afterwards in state *do calc.* the maximum of these values for $R^*(m)$ is calculated (by iteration of *calc_index*). The whole process is repeated (edge from *done calc.* to *start corr.*) and the next n_{par} values for $R^*(m)$ are generated. If $current\_pos = \top$ then all values for $R^*(m) : 0 \leq m < M$ were processed, the final results are updated (as output signals of the module in state *done*) and a new trigger is awaited (transition to state *initial*).

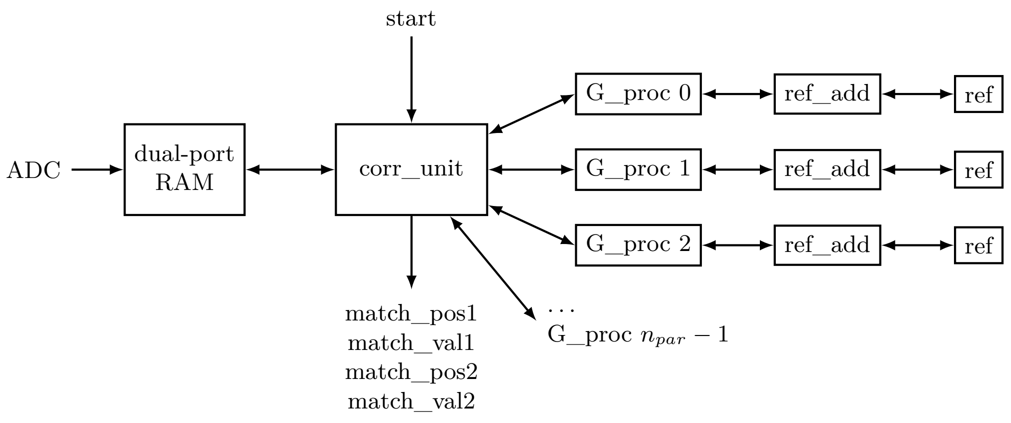


Figure 5.18: Correlation module block diagram

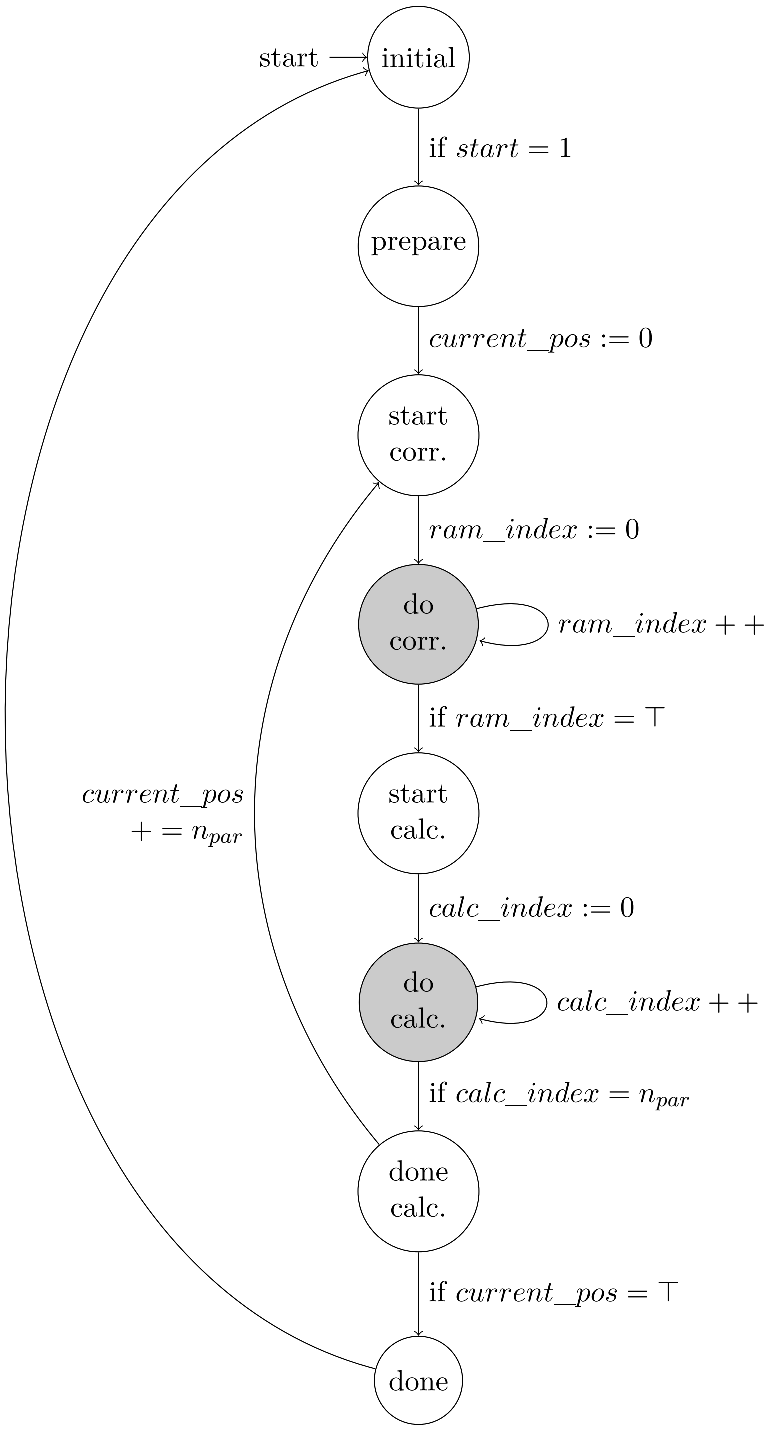


Figure 5.19: Correlation module state machine

Burst Detection

For the relaxed signal the burst detector is implemented in the FPGA analogous to the simulated filter (see Section 4.2). The block diagram is also the same as shown in Figure 4.4. The half-wave rectifier with integrated decay and low-pass filters are implemented as digital infinite impulse response filters.

The transfer function of the half-wave rectifier with integrated decay is given as follows:

$$y_{rect}[n](x[n], y_{rect}[n-1]) = \begin{cases} 0 & \text{if } n = 0 \\ x[n] & \text{if } x[n] > y_{rect}[n-1] \\ y_{rect}[n-1] - 1 & \text{else} \end{cases}$$

The transfer function of the two low-pass filters is given as follows:

$$y_{lp}[n](x[n], y_{lp}[n-1]) = \begin{cases} 0 & \text{if } n = 0 \\ y_{lp}[n-1] \cdot \frac{2^{K_{lp}-1}}{2^{K_{lp}}} + x[n] \cdot \frac{1}{2^{K_{lp}}} & \text{else} \end{cases}$$

$$K_{lp} \in \mathbb{N} \wedge K_{lp} \geq 1$$

The filters are designed s.t. there are only multiplications and divisions by factors of two ($2^{K_{lp}}$) since these become shift operations in binary representation.

5.3 Software Implementation

All measured time instants are transmitted to a PC for further processing and TDOA calculation. This could also be performed on the FPGA in the future but is done on a different platform for now.

5.3.1 EIA-232 - Ethernet Gateway

The single board computer *Raspberry Pi 1 Mod. B+* [Ras14] is used as EIA-232 - Ethernet Gateway. It is operated by *Raspbian* (a Linux operating system based on *Debian*) [Ras18]. The only task of this single board computer is to act as gateway from EIA-232 to Ethernet. One of the FPGA-boards is connected to the single board computer via a 3V3 EIA-232 link on the GPIO pins of the Raspberry Pi. Any computer which is connected via Ethernet can then attach via software to this serial port (e.g., by using TCP forwarding).

5.3.2 PC Software with User Interface

The software for the PC is implemented in C/C++ using *ncurses* [Fre18] for the text user interface. It parses the data from the FPGA which contains the time instants of the received signals. These time measurements are then periodically processed by the particle filter (see Section 4.4) to determine the location of the mobile object. The particle filter is implemented using multiple threads s.t. it utilizes the optimal number of

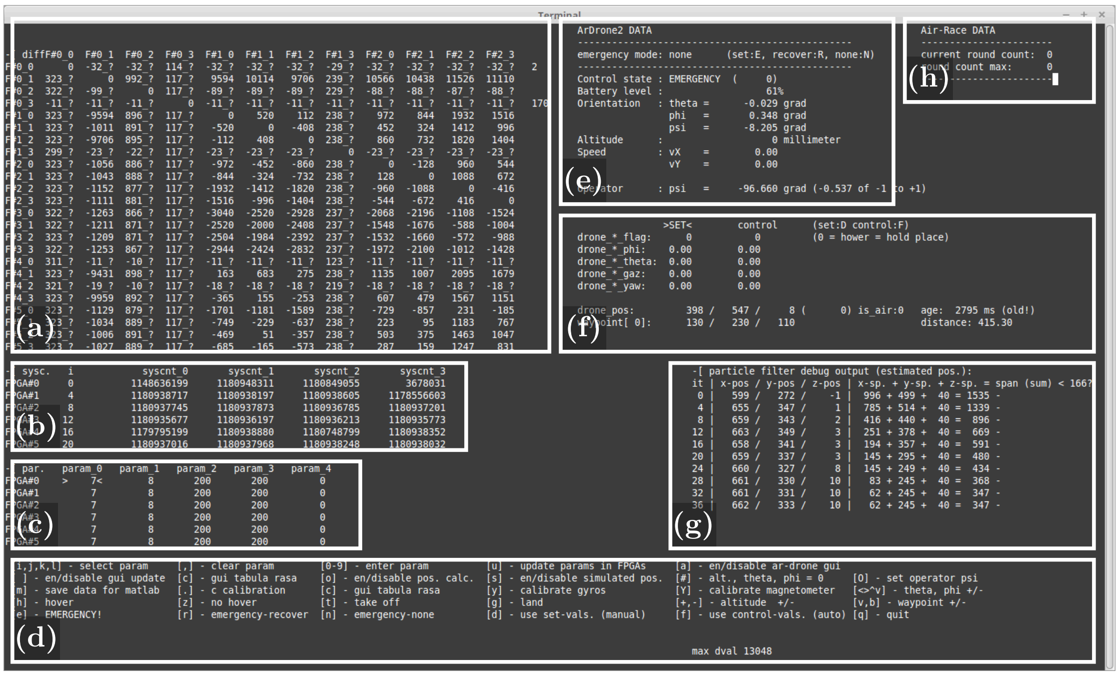
CPU cores. On the used PC (*Sun Fire X4140*) the software is running on 12 cores and calculates positions with up to $14Hz$. If the position is outside of the bounding area in any dimension it is discarded. It is also discarded if the value for *span* is above a certain level. To reduce the risk of wrongly calculated values the final position is given by a *three point running median* for every dimension. This means the median of the last three points is regarded as correct position.

The further tasks of the PC software include input and output for control parameters as well as logging functionalities in order to visualise and analyse a trace later. In order to directly test the usefulness of the localization system for an autopilot system the interface to a quadcopter control system is added to the PC software.

Text User Interface

The text user interface shows the current position of the mobile object, current parameters which were set in the FPGAs, control information of the quadcopter as well as all relevant debug information. Figure 5.20 shows a screenshot of the user interface.

Figure 5.20a shows a matrix with the differences between time instants when a burst signal was detected by FPGA n_F on receiver n_R. The columns and rows contain the labels `F`$\#n_F\_n_R$. Figure 5.20b shows a matrix with the time instants when a burst signal was detected by each FPGA and receiver. These are raw values of the global system counter `syscnt`. In Figure 5.20c the table shows all the parameters which can be set in the FPGAs. These parameters change the properties of the burst detection (see Section 5.2.5). In Figure 5.20d the list shows possible keyboard commands with a short description. Figure 5.20e shows the current status of the quadcopter including control state, battery level, orientation, altitude, and speed. Figure 5.20f shows the values for the flight controller including set points for orientation and altitude as well as the current position of the mobile object and the current target waypoint. In Figure 5.20g the list shows the debug output of the performed steps while determining the position using the particle filter. These entries include the position and the span-value for every nth step. Figure 5.20g shows the counter of already completed 8-figures which is incremented at every transition of the last waypoint.



```
-[ diffF#0_0  F#0_1  F#0_2  F#0_3  F#1_0  F#1_1  F#1_2  F#1_3  F#2_0  F#2_1  F#2_2  F#2_3
F#0_0       0  -32_?  -32_?  114_?  -32_?  -32_?  -32_?  -29_?  -32_?  -32_?  -32_?  -32_?   2
F#0_1  323_?      0  992_?  117_?   9594  10114   9706  239_?  10566  10438  11526  11110
F#0_2  322_?  -99_?      0  117_?  -89_?  -89_?  -89_?  229_?  -88_?  -88_?  -87_?  -88_?
F#0_3  -11_?  -11_?  -11_?      0  -11_?  -11_?  -11_?  -11_?  -11_?  -11_?  -11_?  -11_?   170
F#1_0  323_?  -9594  896_?  117_?      0    520    112  238_?    972    844   1932   1516
F#1_1  323_?  -1011  891_?  117_?   -520      0   -408  238_?    452    324   1412    996
F#1_2  323_?  -9706  895_?  117_?   -112    408      0  238_?    860    732   1820   1404
F#1_3  299_?  -23_?  -22_?  117_?  -23_?  -23_?  -23_?      0  -23_?  -23_?  -23_?  -23_?
F#2_0  323_?  -1056  886_?  117_?   -972   -452   -860  238_?      0   -128    960    544
F#2_1  323_?  -1043  888_?  117_?   -844   -324   -732  238_?    128      0   1088    672
F#2_2  323_?  -1152  877_?  117_?  -1932  -1412  -1820  238_?   -960  -1088      0   -416
F#2_3  323_?  -1111  881_?  117_?  -1516   -996  -1404  238_?   -544   -672    416      0
F#3_0  322_?  -1263  866_?  117_?  -3040  -2520  -2928  237_?  -2068  -2196  -1108  -1524
F#3_1  322_?  -1211  871_?  117_?  -2520  -2000  -2408  237_?  -1548  -1676   -588  -1004
F#3_2  323_?  -1209  871_?  117_?  -2504  -1984  -2392  237_?  -1532  -1660   -572   -988
F#3_3  322_?  -1253  867_?  117_?  -2944  -2424  -2832  237_?  -1972  -2100  -1012  -1428
F#4_0  311_?  -11_?  -10_?  117_?  -11_?  -11_?  -11_?  123_?  -11_?  -11_?  -11_?  -11_?
F#4_1  323_?  -9431  898_?  117_?    163    683    275  238_?   1135   1007   2095   1679
F#4_2  321_?  -19_?  -10_?  117_?  -18_?  -18_?  -18_?  219_?  -18_?  -18_?  -18_?  -18_?
F#4_3  323_?  -9959  892_?  117_?   -365    155   -253  238_?    607    479   1567   1151
F#5_0  323_?  -1129  879_?  117_?  -1701  -1181  -1589  238_?   -729   -857    231   -185
F#5_1  323_?  -1034  889_?  117_?   -749   -229   -637  238_?    223     95   1183    767
F#5_2  323_?  -1006  891_?  117_?   -469     51   -357  238_?    503    375   1463   1047
F#5_3  323_?  -1027  889_?  117_?   -685   -165   -573  238_?    287    159   1247    831
```

```
ArDrone2 DATA
-----------------------------------------------
emergency mode: none      (set:E, recover:R, none:N)
-----------------------------------------------
Control state : EMERGENCY  (     0)
Battery level :                 61%
Orientation   : theta =      -0.029 grad
                phi   =       0.348 grad
                psi   =      -8.205 grad
Altitude      :                   0 millimeter
Speed         : vX    =        0.00
                vY    =        0.00

Operator      : psi   =     -96.660 grad (-0.537 of -1 to +1)
```

```
Air-Race DATA
----------------------
current round count:  0
round count max:      0
----------------------
```

```
                >SET<        control      (set:D control:F)
drone_*_flag:      0            0         (0 = hower = hold place)
drone_*_phi:    0.00         0.00
drone_*_theta:  0.00         0.00
drone_*_gaz:    0.00         0.00
drone_*_yaw:    0.00         0.00

drone_pos:         398 /   547 /     8 (       0) is_air:0   age:  2795 ms (old!)
waypoint[ 0]:      130 /   230 /   110                       distance: 415.30
```

```
-[ sysc.   i          syscnt_0     syscnt_1     syscnt_2     syscnt_3
FPGA#0     0        1148636199   1180948311   1180849055      3678031
FPGA#1     4        1180938717   1180938197   1180938605   1178556603
FPGA#2     8        1180937745   1180937873   1180936785   1180937201
FPGA#3    12        1180935677   1180936197   1180936213   1180935773
FPGA#4    16        1179795199   1180938880   1180748799   1180938352
FPGA#5    20        1180937016   1180937968   1180938248   1180938032
```

```
-[ par.   param_0   param_1   param_2   param_3   param_4
FPGA#0    >    7<         8       200       200         0
FPGA#1         7          8       200       200         0
FPGA#2         7          8       200       200         0
FPGA#3         7          8       200       200         0
FPGA#4         7          8       200       200         0
FPGA#5         7          8       200       200         0
```

```
-[ particle filter debug output (estimated pos.):
it | x-pos / y-pos / z-pos | x-sp. + y-sp. + z-sp. = span (sum) < 166?
 0 |   599 /   272 /    -1 |  996 + 499 +  40 = 1535 -
 4 |   655 /   347 /     1 |  785 + 514 +  40 = 1339 -
 8 |   659 /   343 /     2 |  416 + 440 +  40 =  896 -
12 |   663 /   349 /     3 |  251 + 378 +  40 =  669 -
16 |   658 /   341 /     3 |  194 + 357 +  40 =  591 -
20 |   659 /   337 /     3 |  145 + 295 +  40 =  480 -
24 |   660 /   327 /     8 |  145 + 249 +  40 =  434 -
28 |   661 /   330 /    10 |   83 + 245 +  40 =  368 -
32 |   661 /   331 /    10 |   62 + 245 +  40 =  347 -
36 |   662 /   333 /    10 |   62 + 245 +  40 =  347 -
```

```
[i,j,k,l] - select param     [,] - clear param          [0-9] - enter param          [u] - update params in FPGAs     [a] - en/disable ar-drone gui
[ ] - en/disable gui update  [c] - gui tabula rasa      [o] - en/disable pos. calc.  [s] - en/disable simulated pos.  [#] - alt., theta, phi = 0     [O] - set operator psi
[m] - save data for matlab   [.] - c calibration        [c] - gui tabula rasa        [y] - calibrate gyros            [Y] - calibrate magnetometer   [<>^v] - theta, phi +/-
[h] - hover                  [z] - no hover             [t] - take off               [g] - land                       [+,-] - altitude  +/-          [v,b] - waypoint +/-
[e] - EMERGENCY!             [r] - emergency-recover    [n] - emergency-none         [d] - use set-vals. (manual)     [f] - use control-vals. (auto) [q] - quit

                                                                                                             max dval 13048
```

Figure 5.20: The PC software text user interface shows the current position of the mobile object, current parameters which were set in the FPGAs, control information of the quadcopter as well as all relevant debug information.

CHAPTER 6

Validation

The implemented parts of the system are tested as individual components to check their functionality. Using the component-wise verified parts the whole LPS is assembled and tested in two stages: First it is used to determine static positions of the mobile object. As soon as the system is verified to be reliable it is then used to implement an autopilot system for an UAV.

6.1 Test of Components

During implementation the different components, such as transmitter modules, receiver modules and parts of the FPGA implementation, are tested individually. Based on these component tests further adjustments (e.g., revised hardware modules) are made to ensure the correct functionality of each component.

6.1.1 Transmitter Module

The functionality of the transmitter module is tested by measuring the output directly at the ultrasonic transmitter (see Figure 6.1). First the correct frequency ($f_{USS} = 24kHz$) and amplitude ($V_{USS} = 20V_{PP}$) is verified (see Figure 6.2). The transmitter module is capable of generating an arbitrary modulated waveform. The oscillogram in Figure 6.3 shows the test of such a waveform modulation. In this case a modified gold code is shown (see Section 3.4.2). Other waveforms such as the later used burst signals are verified also by inspection using a oscilloscope (see Figure 6.4).

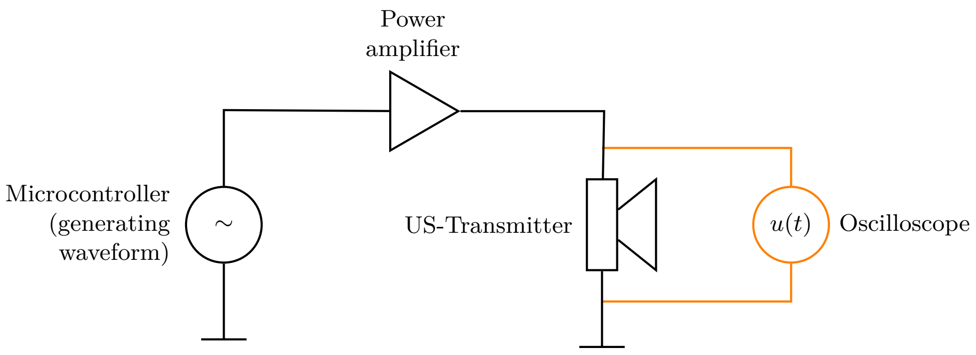
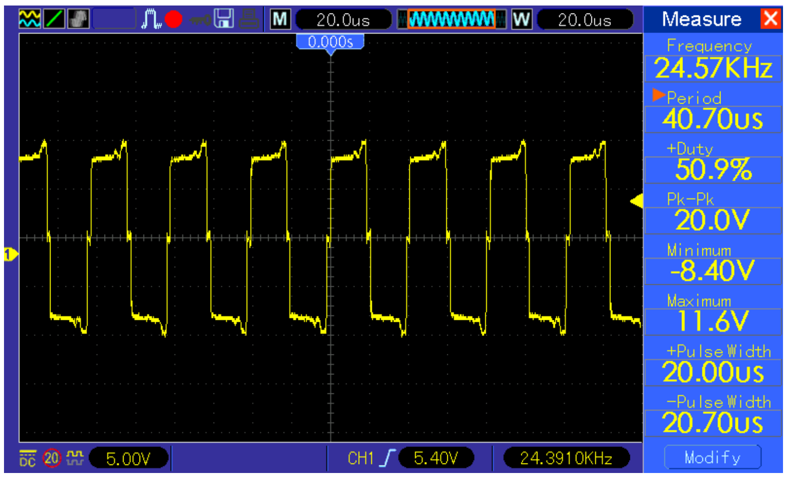


Figure 6.1: Block diagram of transmitter module test. The functionality of the transmitter module is tested by measuring the output directly at the ultrasonic transmitter.



Frequency 24.57KHz
Period 40.70us
+Duty 50.9%
Pk-Pk 20.0V
Minimum -8.40V
Maximum 11.6V
+Pulse Width 20.00us
-Pulse Width 20.70us

Figure 6.2: Test of transmitter module: The oscillogram shows the carrier frequency of $24.57kHz$ and the amplitude of $20V_{PP}$.

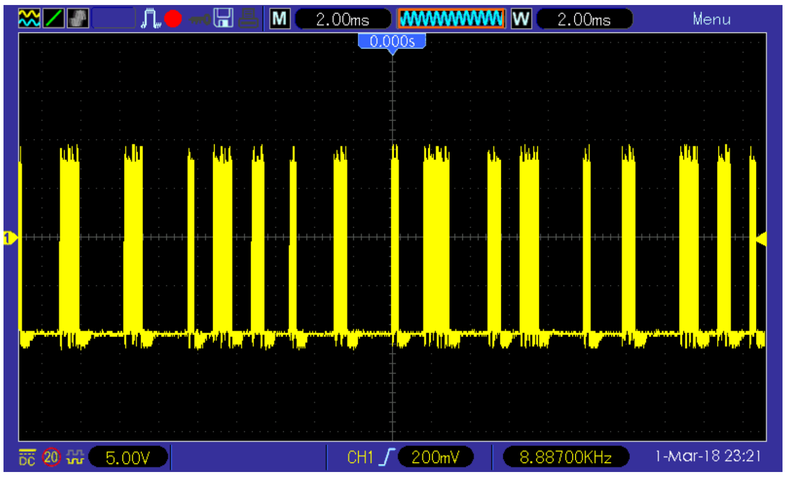
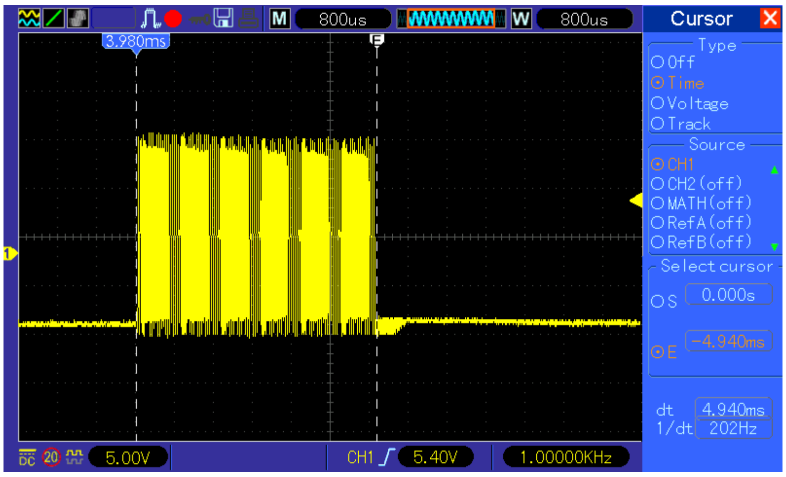


Figure 6.3: Test of transmitter module: The oscillogram shows the carrier frequency which is modulated (turned on and off) by a modified gold code sequence.



Cursor

Type
Off
[✓ Time]
Voltage
Track

Source
[✓ CH1]
CH2 (off)
MATH (off)
RefA (off)
RefB (off)

Select cursor
S 0.000s
[✓ E] -4.940ms

dt 4.940ms
1/dt 202Hz

Figure 6.4: Test of transmitter module: The oscillogram shows a burst signal where the carrier frequency is switched on for the duration of $4.94ms$.

6.1.2 Receiver Module

The functionality of the receiver module is tested by measuring the output of the receiver module and also the signal which is transmitted by the ultrasonic transmitter (see Figure 6.5). A basic test of the receiver module is shown in Figure 6.6. This oscillogram shows the transmitted signal on channel 1 (yellow) and the received signal on channel 2 (blue). In this test the amplitude is high and the burst signal can be identified clearly because the receiver is placed only one meter away from the transmitter and is directly aimed.

To verify the directional receiving pattern a transmitter is placed on a static position and configured to continuously transmit the ultrasonic carrier without any modulation. A receiver module is placed approximately four meters away on a mounting platform with an offset in the plane by 30 centimeter from the center of rotation and manually rotated while the received power at $f_{USS} = 24kHz$ is measured using the FFT function of the oscilloscope. Figure 6.7a shows the recorded values for this test, while Figure 6.7b represents the reference from the manufacturer datasheet [Pro15]. The difference between these two graphs is caused by the offset in the horizontal plane when the mounting platform is rotated. Despite from this the directional receiving pattern of the receiver module is successfully verified with the reference pattern of the ultrasonic transducer.

Similar to the directional pattern the signal strength as function of the distance from the transmitter to the receiver is verified. Every $0.5m$ a point is measured. Figure 6.8 shows the result for this measurement. If the distance between the transmitter and the receiver is less than three meters the receiver is fully saturated and therefore the amplitude is independent from the distance in this range. Because of spherical divergence for any signal there is a theoretical intensity drop of $6dB$ per distance doubled [Sen18]. Furthermore, there is atmospheric attenuation which depends on the frequency of the signal, the temperature and humidity. For a frequency of $25kHz$, a temperature of $24°C$ and a relative humidity of 50% this is $0.75dB/m$ [ISO93]. Adding these effects results in a loss of $6dB + 3 \cdot 0.75dB = 8.25dB$ for a change in the distance from $3m$ to $6m$. However, the measurements indicate a higher loss of $9.5dB$.

Figure 6.9 shows another test where the time delay of ultrasonic signals is shown. Two receivers are placed at different locations ($\mathbf{P_1}$ and $\mathbf{P_2}$) with known distance to the transmitter at location $\mathbf{M}$. The difference $\delta d = d_1 - d_2$ is known (where d_n denotes the distance from the transmitter to the respective receiver $d_n = ||\mathbf{M} - \mathbf{P_n}||$). In this test scenario the distance is $\delta d = 50cm$. The signal is received by each of the two receivers and the time delay can be measured: $\delta t = 1.4ms$. Given the time delay and the speed of sound the distance can be calculated: $s = \delta t \cdot v_{sound} = 1.4ms \cdot 340m/s \approx 47.6cm$. The deviation of s and δd is caused by imprecise manual measurement of the timing at the oscillogram.

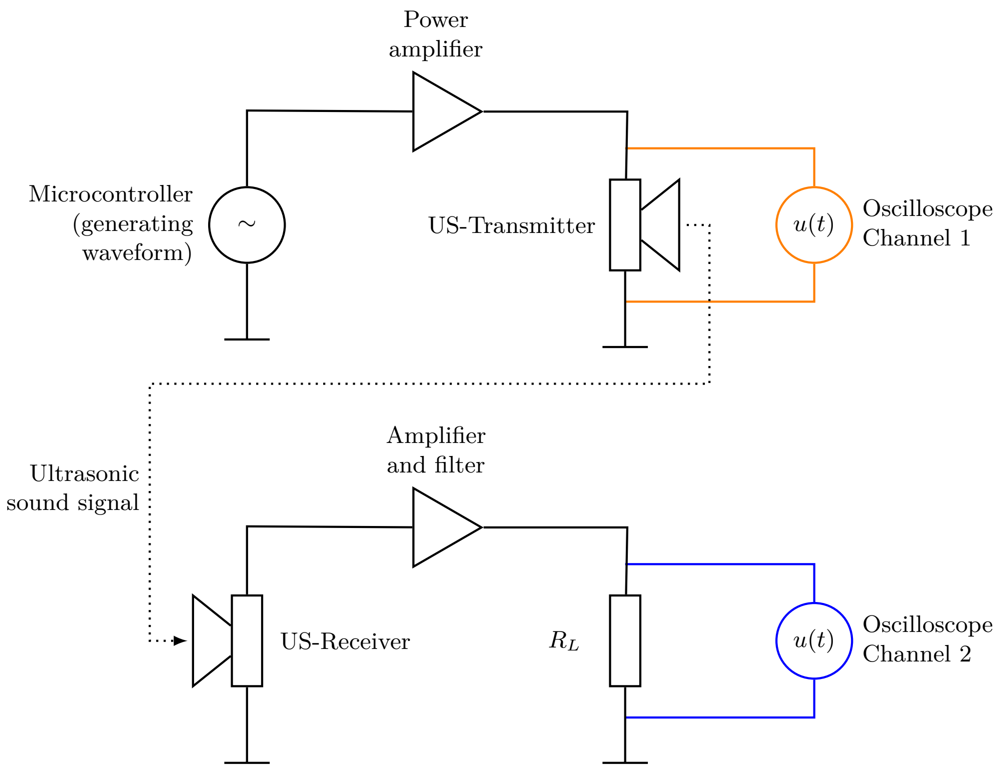
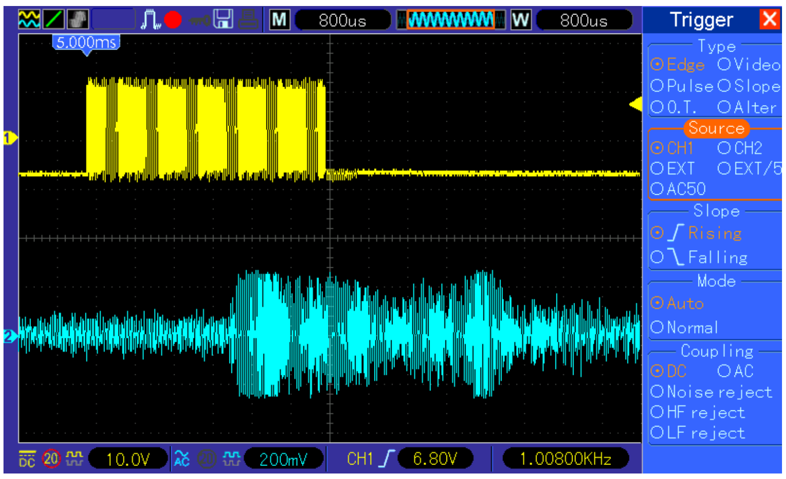


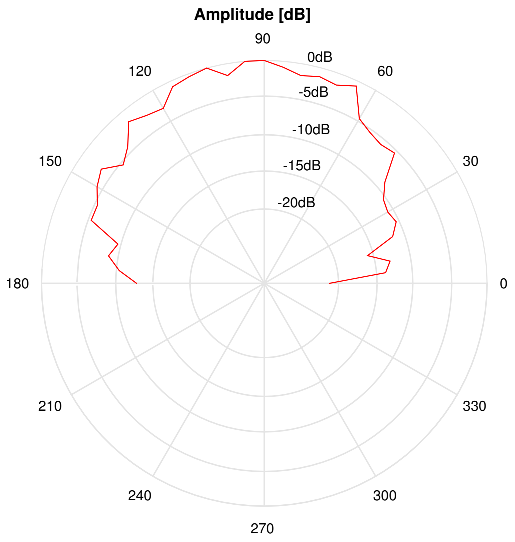
Figure 6.5: Block diagram of receiver module test. The functionality of the receiver module is tested by measuring the output of the receiver module and also the signal which is transmitted by the ultrasonic transmitter.



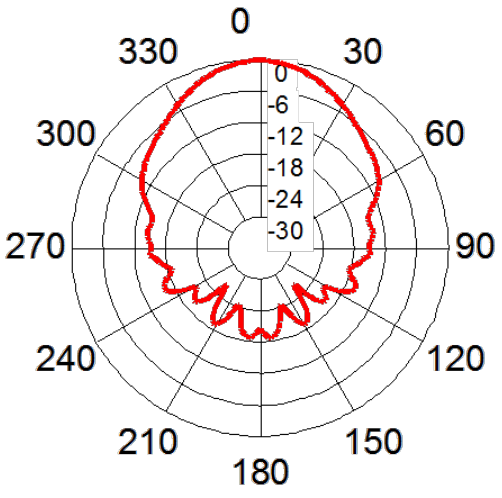
Type: [✓ Edge] Video Pulse Slope O.T. Alter
Source: [✓ CH1] CH2 EXT EXT/5 AC50
Slope: [✓ Rising] Falling
Mode: [✓ Auto] Normal
Coupling: [✓ DC] AC Noise reject HF reject LF reject

DC 20 10.0V AC 200mV CH1 6.80V 1.00800KHz

Figure 6.6: Test of receiver module: The oscillogram shows the transmitted signal on channel 1 (yellow) and the received signal on channel 2 (blue).



(a) Measured values for receiver module.



(b) Reference for the used ultrasonic receiver from manufacturer datasheet [Pro15].

Figure 6.7: Test of directional receiving pattern for the receiver module: Amplitude as function of orientation angle. The receiver module (v2.0) is placed approximately four meters from the transmitter and manually rotated while the received power at $f_{USS} = 24kHz$ is measured using the FFT function of the oscilloscope.

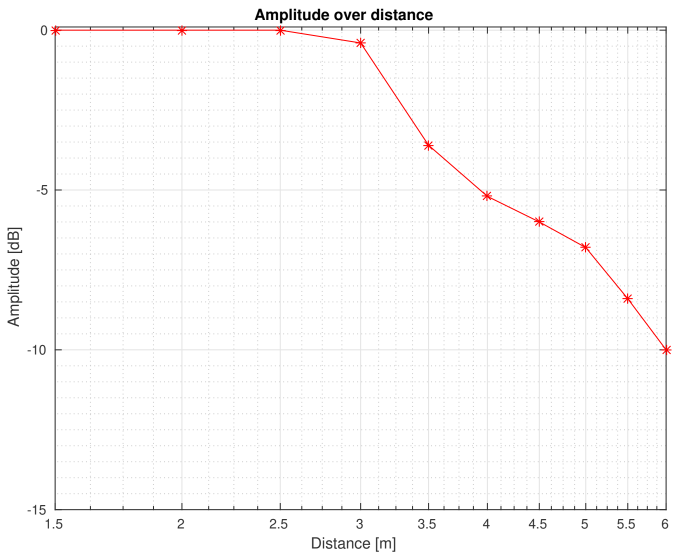


Figure 6.8: Test of the signal strength as function of the distance from transmitter to receiver. A receiver module is placed at different distance from the transmitter while the received power at $f_{USS} = 24kHz$ is measured using the FFT function of the oscilloscope.

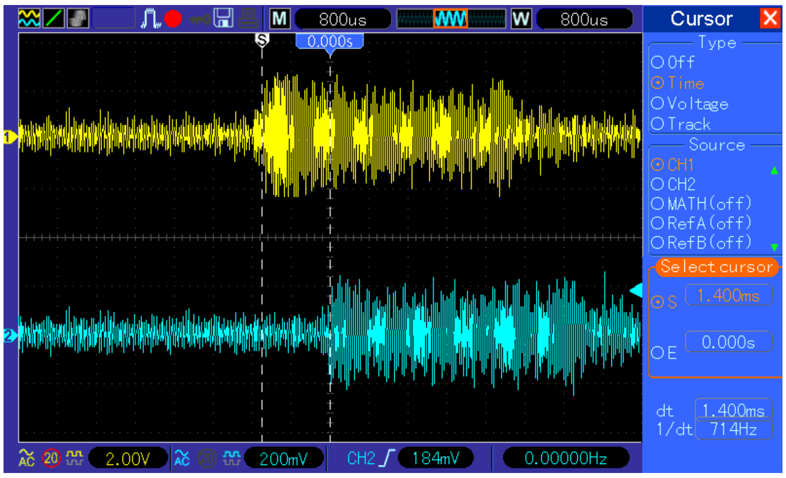


Figure 6.9: Test of receiver module: The oscillogram shows the received signal by two receiver modules on channel 1 (yellow) and channel 2 (blue). The two modules are located at a different distance from the transmitter and therefore the signal delay is different by $1.4ms$.

6.2 Static Positions

Static position tests are carried out with six FPGA-Boards, with four receivers each, in an rectangular area with $x_{max} = 6m$ and $y_{max} = 3m$. The transmitter is placed on a tripod at various locations within this area. Figure 6.10 shows the traces for eight different locations of the mobile object. The light blue, red, magenta and green lines indicates the traces of the mobile object. The blue stars indicate the actual position of the mobile object which was measured manually while the red stars indicate the positions of the receivers.

The deviation of the determined position to the actual position in x- and y-dimension is significantly smaller than in z-dimension. This is caused by the placement of the receivers. The used mounting platform is standing on the floor and therefore all receivers are below $90cm$ on the z-axis. This behaviour is consistent with the simulation.

In Table 6.1 the measured and calculated values are given for m_0^* to m_7^* (units for all values are centimeters). The actual positions (x_{act}, y_{act}, z_{act}) with median (x_{med}, y_{med}, z_{med}) and standard deviation (σ_x, σ_y, σ_z) are given. At each position n_{dp} values were determined by the LPS. The deviations between the actual positions and the median of the determined positions (x_{dev}, y_{dev}, z_{dev}) as well as the absolute value of the deviation ($|x_{dev}|$, $|y_{dev}|$, $|z_{dev}|$) are given. For some columns the table contains the mean (avg), minimum (min), maximum (max), and median (med). The deviation is calculated by $x_{dev} = x_{med} - x_{act}$, $y_{dev} = y_{med} - y_{act}$, and $z_{dev} = z_{med} - z_{act}$.

These tests show that depending on the position the standard deviation for static objects in x- and y-dimension is 1.9 to $4.9cm$. Standard deviation for z-dimension is 5.8 to $13.4cm$. Figure 6.11 shows a histogram for the measurements in x-, y- and z-dimension for the static position m_4^*. This example shows a standard deviation of $\sigma_x = 3.3cm$, $\sigma_y = 4.9cm$ and $\sigma_z = 12.0cm$. In these tests the absolute deviation between the actual positions and the median of the determined positions vary in x- and y-dimension from 0.0 to $7.6cm$. For z-dimension this deviation is 2.3 to $18.2cm$.

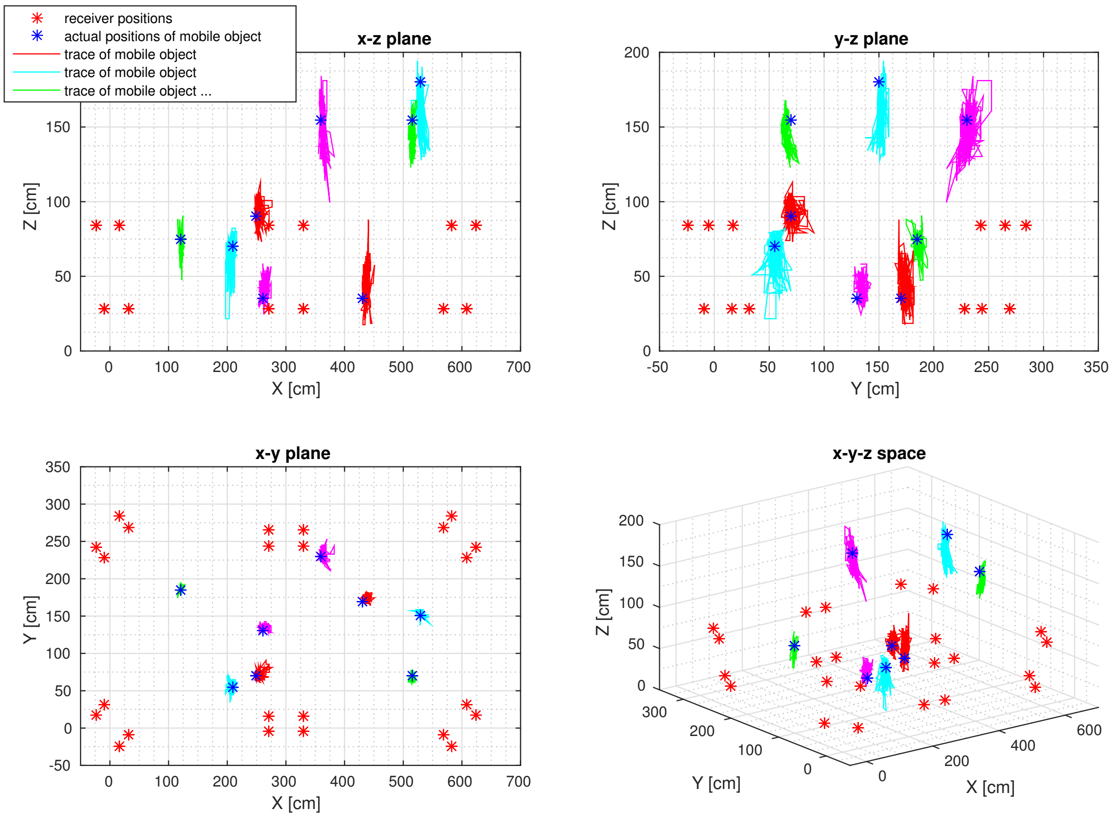


Figure 6.10: Trace of eight measurements for static positions while the transmitter is mounted on a tripod. The light blue, red, magenta and green lines indicates the traces of the mobile object (which is located on static positions). The blue stars indicate the actual position of the mobile object while the red stars indicate the positions of the receivers.

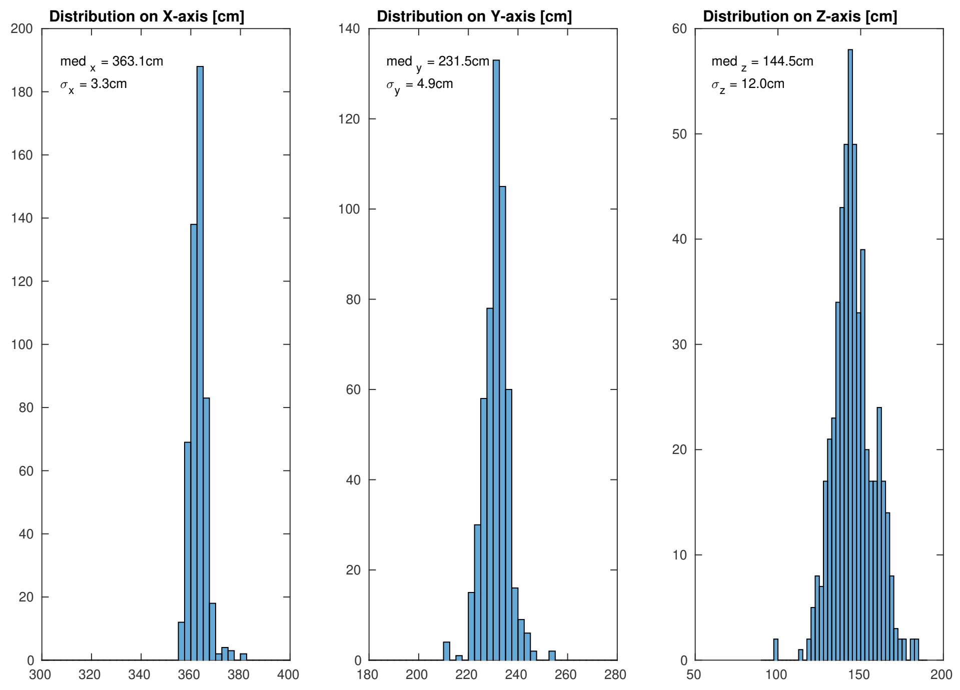


Figure 6.11: Histogram for the measurements in x-, y- and z-dimension for the static position m_4^*. These measurements show a standard deviation of $\sigma_x = 3.3cm$, $\sigma_y = 4.9cm$ and $\sigma_z = 12.0cm$.

Position	x_{act}	y_{act}	z_{act}	x_{med}	y_{med}	z_{med}	σ_x	σ_y	σ_z	n_{dp}	x_{dev}	y_{dev}	z_{dev}	$\|x_{dev}\|$	$\|y_{dev}\|$	$\|z_{dev}\|$
m_0^*	250	70	90	253.6	69.6	92.7	4.2	3.8	5.8	552	3.6	-0.4	2.7	3.6	0.4	2.7
m_1^*	530	150	180	529.5	152.5	161.8	4.6	2.8	13.4	247	-0.5	2.5	-18.2	0.5	2.5	18.2
m_2^*	120	185	75	121.4	185.8	72.7	1.9	2.8	5.9	311	1.4	0.8	-2.3	1.4	0.8	2.3
m_3^*	210	55	70	207.8	55.0	66.9	3.2	3.9	9.3	461	-2.2	0.0	-3.1	2.2	0.0	3.1
m_4^*	360	230	155	363.0	231.5	144.5	3.3	4.9	12.0	519	3.0	1.5	-10.5	3.0	1.5	10.5
m_5^*	430	170	35	437.6	173.4	44.1	3.6	3.2	10.8	354	7.6	3.4	9.1	7.6	3.4	9.1
m_6^*	260	130	35	264.3	134.9	43.6	4.5	2.7	6.2	317	4.3	4.9	8.6	4.3	4.9	8.6
m_7^*	515	70	155	515.9	66.3	140.1	2.7	2.7	8.0	331	0.9	-3.7	-14.9	0.9	3.7	14.9
avg							3.5	3.4	8.9		2.3	1.1	-3.6	2.9	2.2	8.7
min							1.9	2.7	5.8					0.5	0.0	2.3
max							4.6	4.9	13.4					7.6	4.9	18.2
med							3.5	3.0	8.7					2.6	2.0	8.9

Table 6.1: Measurements for static positions where the transmitter is mounted on a tripod. Units for all values are centimeters. Actual positions (x_{act}, y_{act}, z_{act}) with median (x_{med}, y_{med}, z_{med}) and standard deviation (σ_x, σ_y, σ_z) for n_{dp} determined positions. Deviations between the actual positions and the median of the determined positions (x_{dev}, y_{dev}, z_{dev}) and absolute value of the deviation ($|x_{dev}|$, $|y_{dev}|$, $|z_{dev}|$).

6.3 Mobile Object

After testing static positions, we perform further tests with an actually moving object. For this purpose a model railway is used and the transmitter is placed on a waggon. Figure 6.12 shows the setup. The tracks of the railway are measured manually in order to get a reference for the actual positions. The outer dimension of the track is $l_{track} = 220cm$ and $b_{track} = 220cm$. The oval shape of the track gives a length of $s_{track} = 2 \cdot (l_{track} - b_{track}) + b_{track} \cdot \pi = 5.35m$.

For the first test the model railway is operated with low speed. The duration per round is $t_{low} = 28s$. This gives the speed $v_{low} = \frac{s_{track}}{t_{low}} = \frac{5.35m}{28s} \approx 0.2m/s$ for the mobile object. Figure 6.13 shows the result for the test with the given speed. The blue line indicates the trace of the mobile object and the red line indicates the tracks as reference. The red stars indicate the positions of the receivers. For another test the model railway is operated with the maximum speed. The duration per round is $t_{high} = 7.4s$. This gives the speed $v_{high} = \frac{s_{track}}{t_{high}} = \frac{5.35m}{7.4s} \approx 0.75m/s$ for the mobile object. Figure 6.14 shows the result for the test with the higher speed.

Histograms of the deviation between the determined position and the reference track in x- and y-dimension are given in Figure 6.15 and Figure 6.16 respectively. The results show a standard deviation of 6.0 to $7.3cm$ for moving objects. Due to the large number of measurements at various locations the median of the deviation between the determined position and the reference track results in values from 0.0 to $0.3cm$. The values for z-axis are omitted since the calculation of z-values is modified in these tests and is therefore not representative to characterize the accuracy of the system in z-dimension. The test with low speed and with the higher speed do not lead to significant differences in standard deviation. The differences are comparable to the results for static positions where the standard deviation varies at different locations.

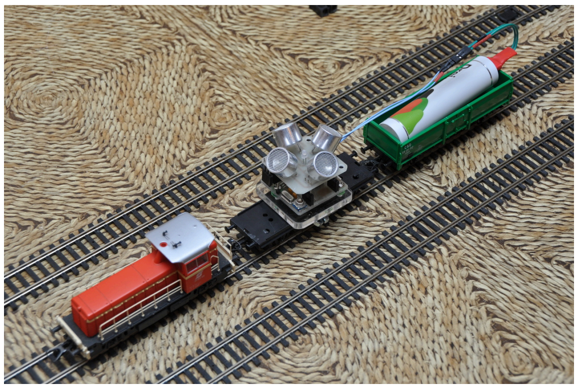


Figure 6.12: Transmitter module v2.0 is placed on a model railway waggon for testing a dynamic scenario with known positions.

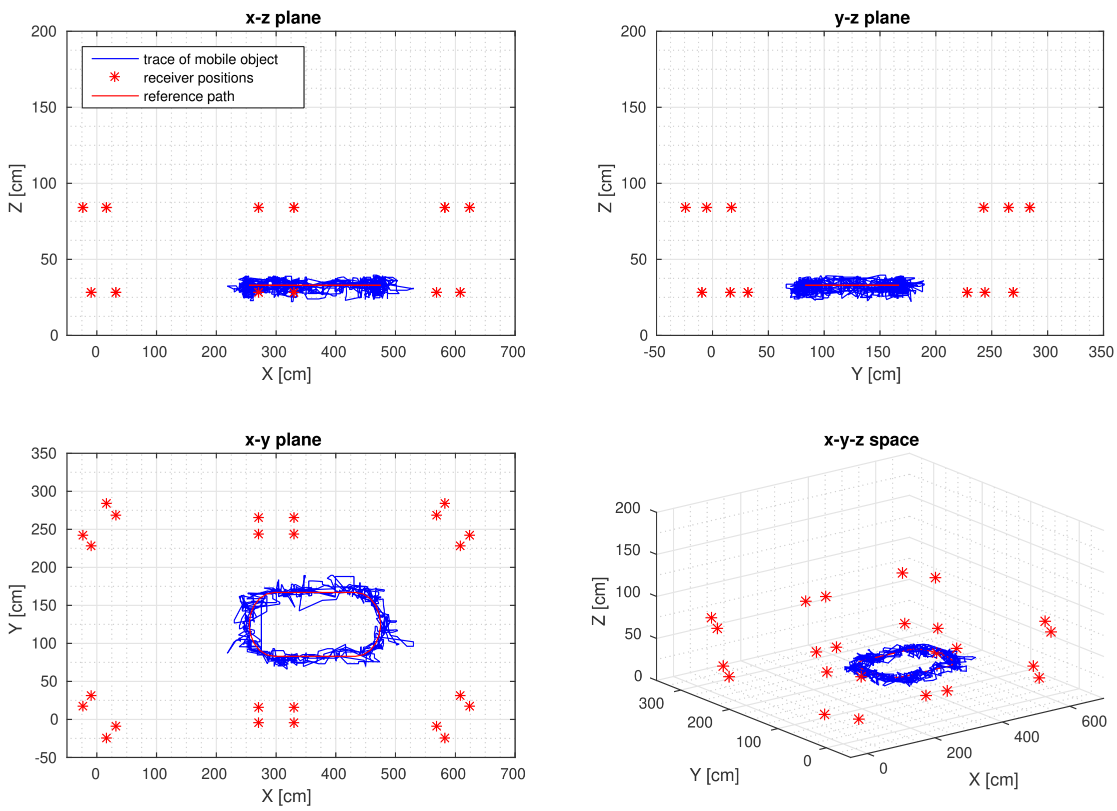


Figure 6.13: Trace of model railway on a oval shaped tracks operated with $v_{low} = 0.2m/s$. The blue lines indicates the trace of the mobile object and the red line indicates the tracks as reference. The red stars indicate the positions of the receivers.

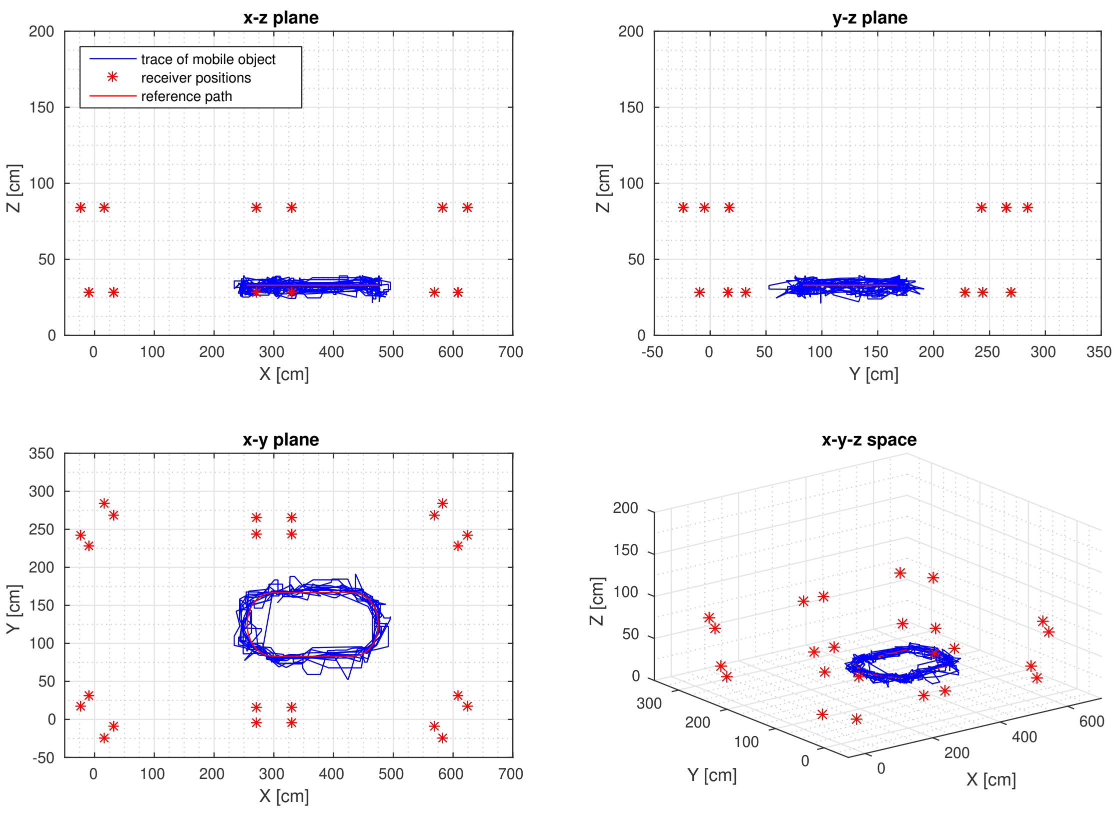


Figure 6.14: Trace of model railway on a oval shaped tracks operated with $v_{high} = 0.75m/s$. The blue lines indicates the trace of the mobile object and the red line indicates the tracks as reference. The red stars indicate the positions of the receivers.

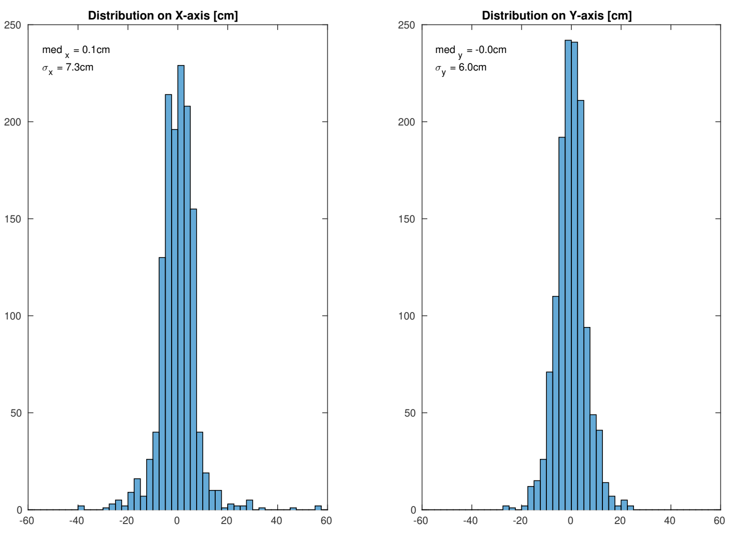


Figure 6.15: Histogram for the measurements in x and y-dimension for the mobile object moving with $v_{low} = 0.2m/s$. These measurements show a standard deviation of $\sigma_x = 7.3cm$ and $\sigma_y = 6.0cm$.

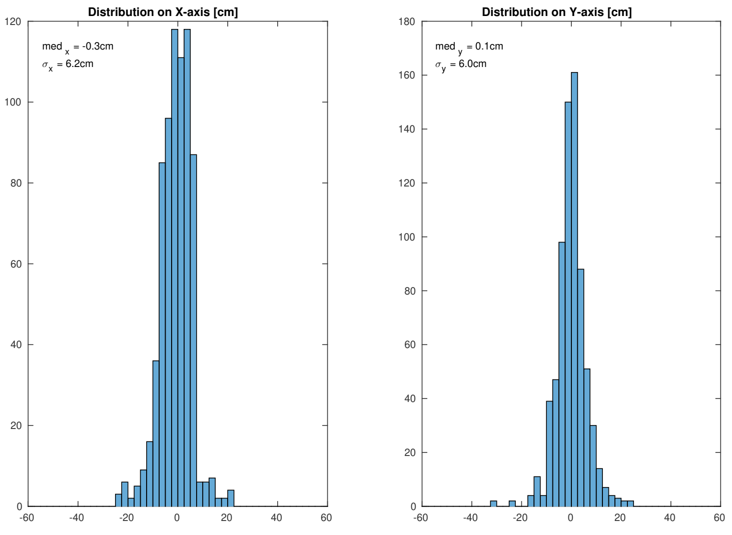


Figure 6.16: Histogram for the measurements in x and y-dimension for the mobile object moving with $v_{high} = 0.75m/s$. These measurements show a standard deviation of $\sigma_x = 6.2cm$ and $\sigma_y = 6.0cm$.

6.4 Quadcopter Autopilot

After successfully carrying out static and mobile tests, we move on to a dynamic setup. The transmitter is mounted on the quadcopter in order to control it. This transforms our system from simple measurements to a control loop. An overview of the quadcopter autopilot system is shown in Figure 6.17.

For this tests the *AR Drone 2.0* quadcopter is used and the transmitter module is placed in the center on the bottom of the UAV. Figure 6.18 shows this setup. The *Raspberry Pi*, which acts as EIA-232 - Ethernet Gateway, is equipped with a Wireless LAN interface using a USB Wireless LAN adapter. This is necessary since the used quadcopter is controlled via Wireless LAN. All autopilot tests are carried out in an area with $x_{max} = 10m$ and $y_{max} = 5m$. Figure 6.19 shows the test setup used for the dynamic tests. As reference flightpath the 8-figure of *Air Race*-Competition is used (see Section 3.1.3).

An UAV control system is implemented using the *Parrot AR Drone SDK* [Par16]. This makes it possible to control the *AR Drone 2.0* over Wireless LAN. For this purpose the SDK provides commands to control the quadcopter, e.g., commands for takeoff and landing, emergency shutdown and orientation which implies movements.

The autopilot is implemented by a flight path which consists of a set of waypoints. Given the current position of the UAV (determined by the local positioning system) and the next waypoint (WP) as target a proportional controller is used to determine the needed orientation of the quadcopter:

$$\theta_{control} = (x_{WP} - x_{UAV}) \cdot K_x$$

$$\phi_{control} = (y_{WP} - y_{UAV}) \cdot K_y$$

$$F_{control} = (z_{WP} - z_{UAV}) \cdot K_z$$

The values $\theta_{control}$, $\phi_{control}$ and $F_{control}$ are the control parameters which are transmitted to the quadcopter via Wireless LAN. If the distance projected to the plane δs is less than a certain threshold the waypoint is considered as reached and the next waypoint is selected as target. The distance δs is computed as:

$$\delta s = \sqrt{(x_{WP} - x_{UAV})^2 + (y_{WP} - y_{UAV})^2}$$

Figure 6.20 shows the partial trace of such a test. This example shows that the UAV stays inside the the test area for several repetitions of the complete 8-figure. There are some deviations of the UAV's position and the reference flight path. These deviations might be caused on the one hand by the simple proportional controller and on the other hand by delays because of the Wireless LAN link.

A video of a flight where the UAV autonomously performs several repetitions of the complete 8-figure is available at:
`https://q.lanthan.at/ff8db3a3bfaf182f683c29d071c78ea9`

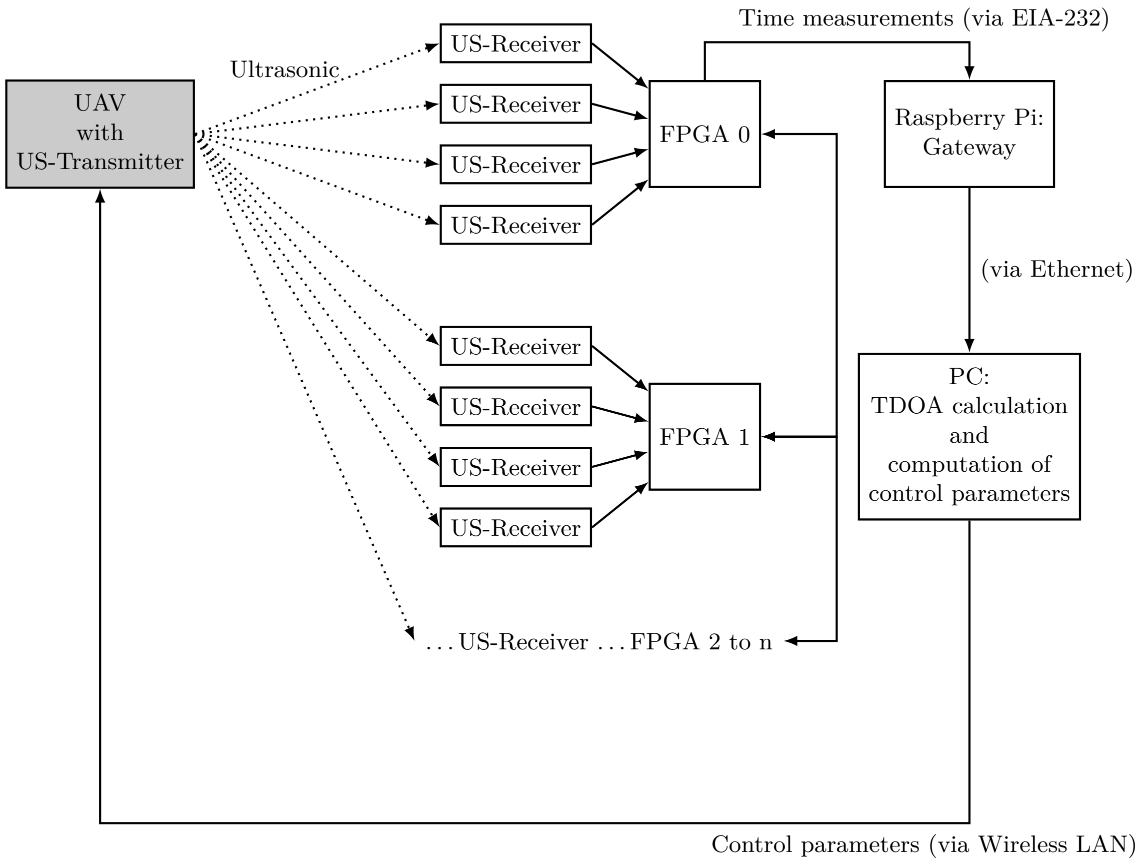
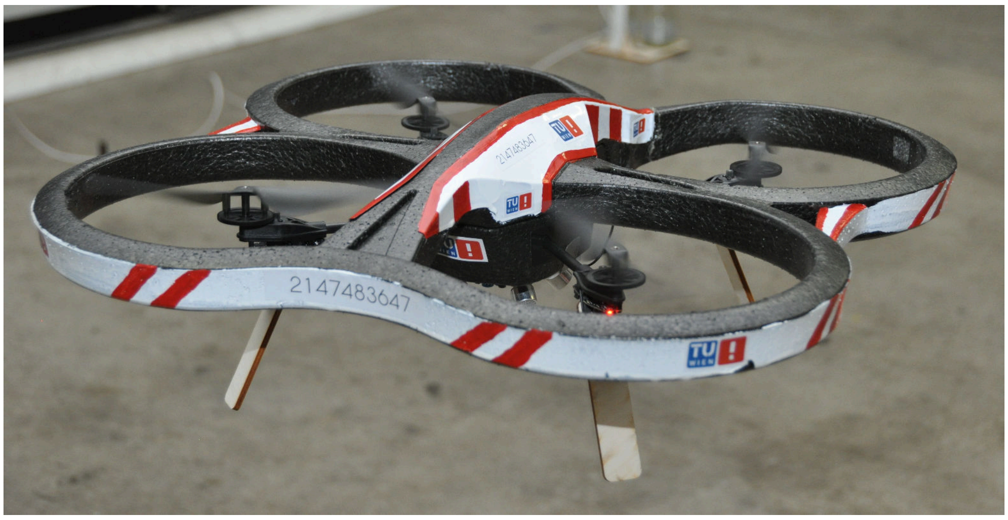
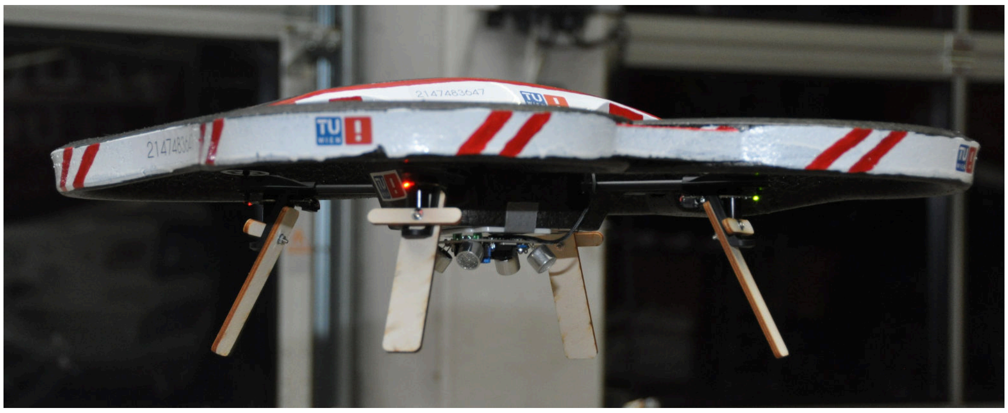


Figure 6.17: Block diagram of the implemented LPS with dynamic test setup including UAV autopilot. A transmitter which is placed on the UAV sends ultrasonic signals. Every base-station is equipped with a FPGA-Board and four ultrasonic receivers in order to capture and process the signals. The time measurements are sent through the Raspberry Pi gateway to the PC. The PC performs TDOA calculation and computation of the control parameters. These control parameters are sent via Wireless LAN to the UAV.



(a) Top side of the UAV which is used for the dynamic tests.



(b) Bottom side of the UAV which is used for the dynamic tests. The transmitter module v2.2 is placed in the center on the bottom of the UAV.

Figure 6.18: Quadcopter *AR Drone 2.0* which is used for the dynamic tests.

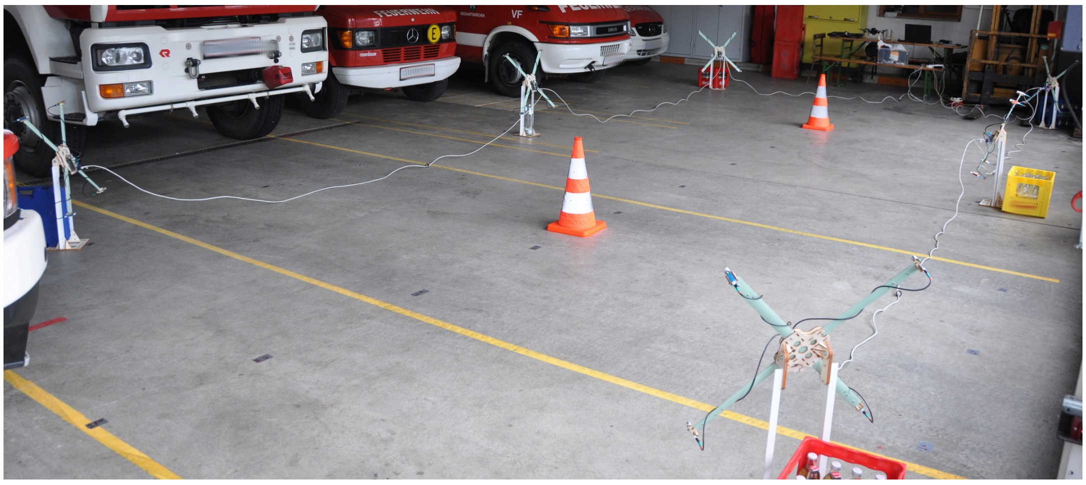


Figure 6.19: Autopilot test area setup with a dimension of $x_{max} = 10m$ and $y_{max} = 5m$. The four of the base stations are placed on the corners of the test area and two on the border at $x = 5m$. The centers of the 8-figure (reference flightpath) is marked by orange traffic cones.

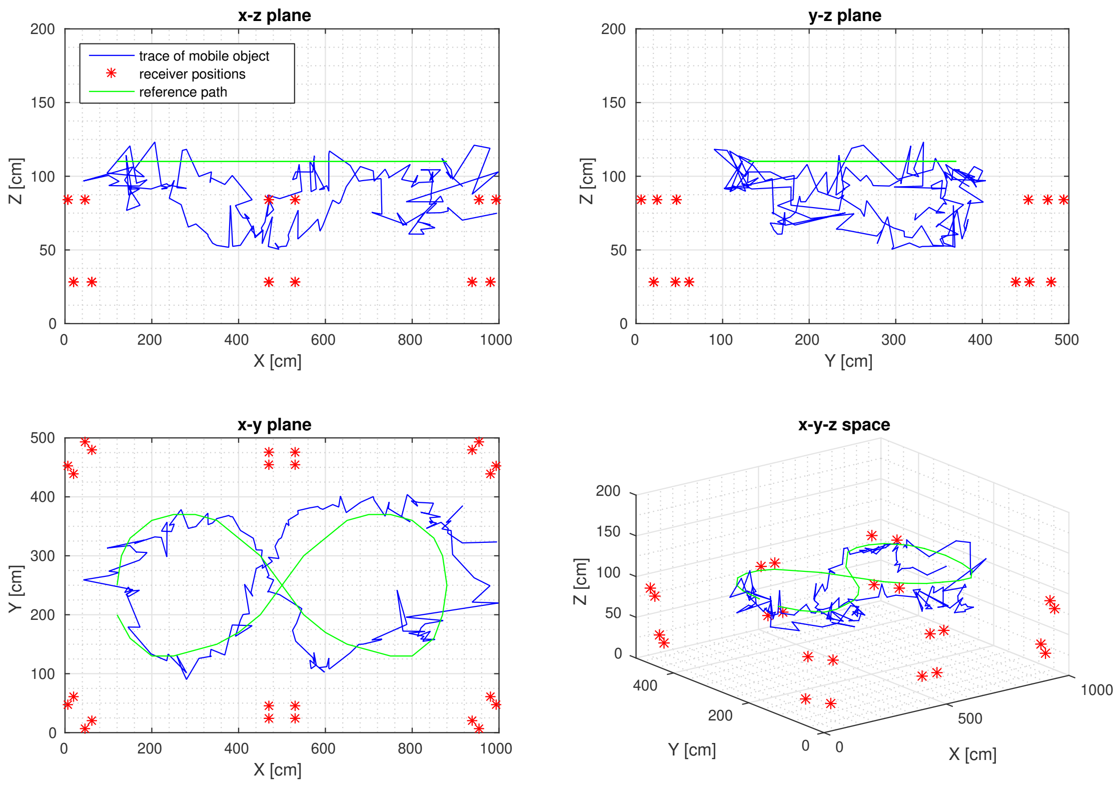
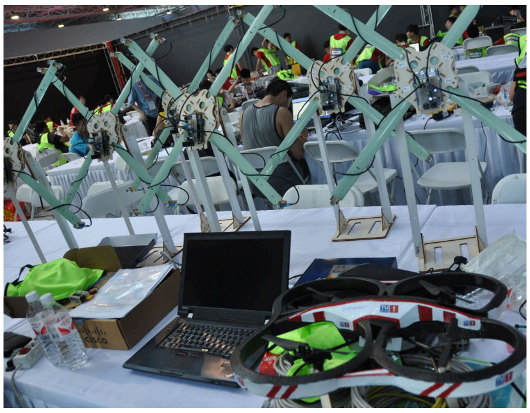


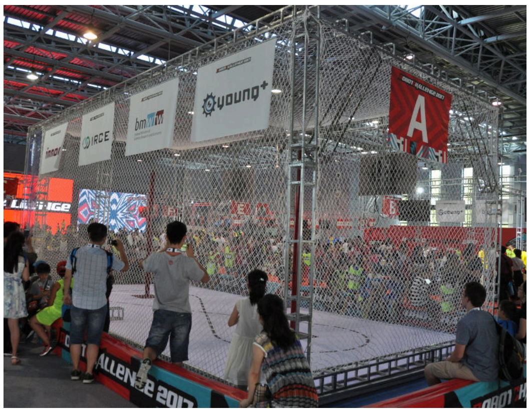
Figure 6.20: Partial trace of flight path of autopilot test. The red stars indicate the positions of the receivers and the blue line indicates the trace of the mobile object. The reference flightpath (8-figure) is indicated by a green line.

6.4.1 RobotChallenge China

The system was planned to be used in the *Air Race* competition at the RobotChallenge 2017 in Bejing, China [You17]. For this purpose all parts of the mounting platform (see Section 6.2) are built modular so that they can be disassembled into small parts. The whole hardware was packed into a suitcase and transported to China.

Unfortunately, more than 25 different Wireless LAN networks were present. Due to this fact it was not possible to transmit any control information in time via the wireless LAN link to the quadcopter. The typical delay for control-messages was about several seconds which made it impossible to operate the quadcopter in this environment and therefore it was not possible to successfully take part in the competition.


(a) Hardware used at the competition.



(b) Competition area with safety net.

Figure 6.21: *Air Race* competition at the RobotChallenge 2017 in Bejing, China. a) shows the hardware that was used at the competition. The mounting platforms for the for the FPGA-Boards and four receivers on X-shaped booms each in the back, quadcopter *AR Drone 2.0* with transmitter module in the front. b) shows the competition area with safety net. The reference flight path (8-figure) is marked by a black dashed line on the ground.

CHAPTER 7

Conclusion and Outlook

In this work we successfully implemented a LPS to be used for an UAV in an indoor location. We are able to demonstrate the fitness of the implemented system to be used in an autopilot setup for a quadcopter. With this autopilot setup the quadcopter is able to follow a path consisting of pre-defined waypoints. The successful operation is demonstrated in a video where the quadcopter autonomously performs several repetitions of an 8-figure.

7.1 Solved Difficulties

The following difficulties had to be solved in order to implement a functional LPS. This was on the one hand a major design decision which had to be revised. On the other hand there were problems for which we describe methods of mitigation or alternative implementations.

- **Ultrasonic Signal**
  Initially we planned to use a continuous ultrasonic signal (see Section 3.4.2) in order to measure the time delay as precise as possible. Simulations looked very promising to measure the time delay using correlation of the received signal with the reference waveform (see Section 4.1.1). However, during hardware implementation it turned out that the correlation of the continuous signal is very prone to variations in frequency. This forced us to implement the system using a relaxed version of the ultrasonic signal (see Section 3.4.2). Due to this relaxed ultrasonic signal the accuracy in time measurements is limited and therefore the accuracy of the whole system is limited as well. The specified accuracy of $s_{accuracy} = 5cm$ could therefore not be reached.

- **Beam Pattern and Destructive Interference**
  The angular beam pattern of the used ultrasonic transmitters is specified with 85°. Since the transmitter is placed on the UAV and the receivers are located in all directions around the UAV there is the need for a more omnidirectional beam pattern of the whole transmitter module. This was achieved using four (or eight) ultrasonic transmitters on the transmitter module. Using more than one ultrasonic transmitters resulted in the problem of destructive interference. This problem was mitigated by using four receivers for each base-station. With this modification the risk of interference for all receivers was minimised.

- **TDOA Calculation Algorithm**
  The algorithm for TDOA calculation described by Bucher and Misra [BM02] did not result in correct results using only four base stations. Furthermore, it turned out that an exact solution of the TDOA problem is not practical in our implementation for the following reasons: As a result of the mitigation strategy to the previous problem the number of received signals was multiplied by four which results in an overdetermined system. Outliers were problematic as well. These were values of receivers which did not recognize the correct signal and there led to wrongly calculated time delay measurements. We therefore changed our calculation method to a particle filter and subsequently implemented such a filter.

7.2 Outlook and Further Work

We identified several aspects which can be improved in the implemented LPS in order to achieve a better accuracy, higher measurement rates, or a more reliable system. The following aspects should be considered in further work to enhance the system.

- **State Estimation**
  Given the current position, the current speed, and the physical properties of the UAV, the position for the next timestep can be estimated. This estimated position can then be incorporated within the TDOA calculation resulting in, speeding up the calculation and therefore leading to a higher measurement rate.

- **Correlation of Received Signal**
  Since there have been problems with the initially planned waveform a burst signal was used as alternative (see Section 3.4.2). If the problems with variations in frequency can be sorted out by using additional means of signal processing the correlation of the received signal with a reference signal can be used to measure the time delay. This may improve the accuracy of the time delay measurements and therefore also the accuracy of the system.

- **Calculation on FPGA**
  For the current implementation of the system the actual TDOA calculation is performed on a PC. The TDOA calculation could be executed on the FPGA. If the implementation of the TDOA calculation is optimized for parallel execution on the FPGA this may speed up the calculation and therefore lead to a higher measurement rate.

- **Identification of Outliers**
  Since we added additional receivers to mitigate problems with destructive interference (see Section 4.4) there might be outliers for time delay measurements. Using the current implementation all measurements are processed by the particle filter. If the outliers can be identified beforehand and omitted from the particle filter it may speed up the convergence for the estimated position and also increase the accuracy of the determined position.

- **Quality Metric for Time Measurements**
  The outlier identification could be further improved to a quality metric for each time delay measurement. The particle filter could then also incorporate the quality of the measurements. This would give more weight to measurements which are identified as very certain. Whereas more uncertain measurements contribute to the calculation with less weight. The quality metric to determine how much certitude is given by a measurement could be inferred from the value of the maximum correlation.

- **Optimized Particle Filter**
  The particle filter was implemented as contingency plan since the originally planned algorithm turned out to be not usable (see Section 4.4). The filter parameters of the particle filter were not extensively optimized yet. This could be done to further improve the position determination.

- **Code Multiplexing**
  If the problem with correlation can be sorted out, a continuous (pseudo) random signal can be used as initially planned. This would make it possible to extend the system for more mobile objects which can be in operation at the same time by using orthogonal codes (which have low cross-correlation to each other).

CHAPTER 8

Appendix

This appendix includes schematics and test point documentation of the implemented hardware modules. It also contains listings of relevant parts of the simulations.

- **Transmitter circuit**
  Figure 8.1 shows the schematic of transmitter circuit v2.2. Figure 8.4 shows the test point documentation for transmitter circuit v2.2.
- **Receiver circuit**
  Figure 8.2 shows the schematic of transmitter circuit v2.0. Figure 8.5 shows the test point documentation for receiver circuit v1.0.
- **FPGA Adapter-Board**
  Figure 8.3 shows the schematic of FPGA Adapter-Board v1.1. Figure 8.6 shows the test point documentation for FPGA Adapter-Board v1.1.
- **TDOA calculation simulation**
  Listing 8.1 shows the TDOA algorithm as presented by Bucher and Misra [BM02]. Listing 8.2 shows the simulation for differences in propagation delays given the known position of the mobile object. Listing 8.3 shows the simulation of TDOA calculation for a pre-defined location of the mobile object. Listing 8.4 shows the simulation of TDOA calculation resulting in a wrongly calculated position.
- **Particle filter simulation**
  Listing 8.5 shows the algorithm for one step of the particle filter. Listing 8.6 shows the script for particle filter simulation.

8.1 Schematics

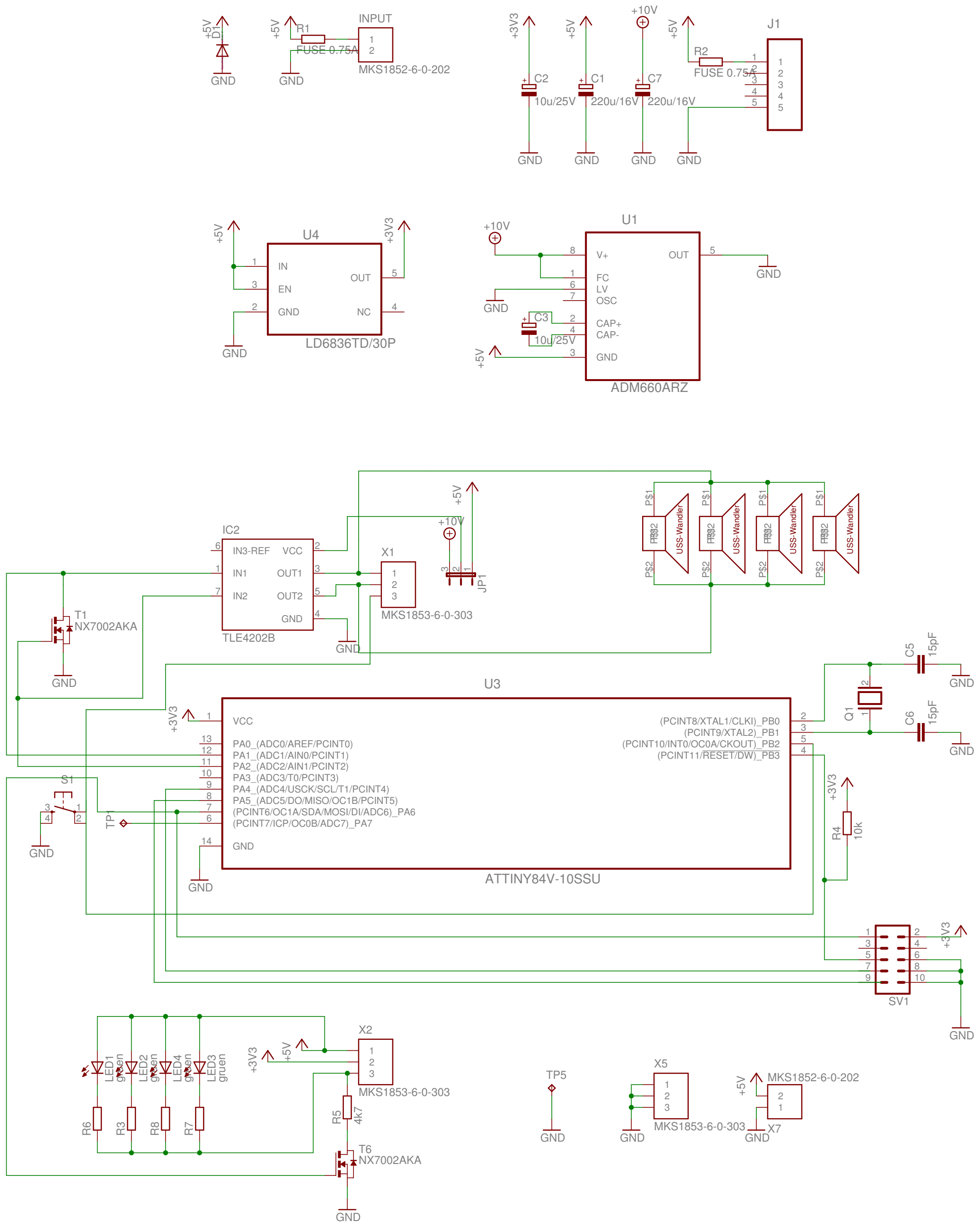


Figure 8.1: Schematic: Transmitter circuit v2.2

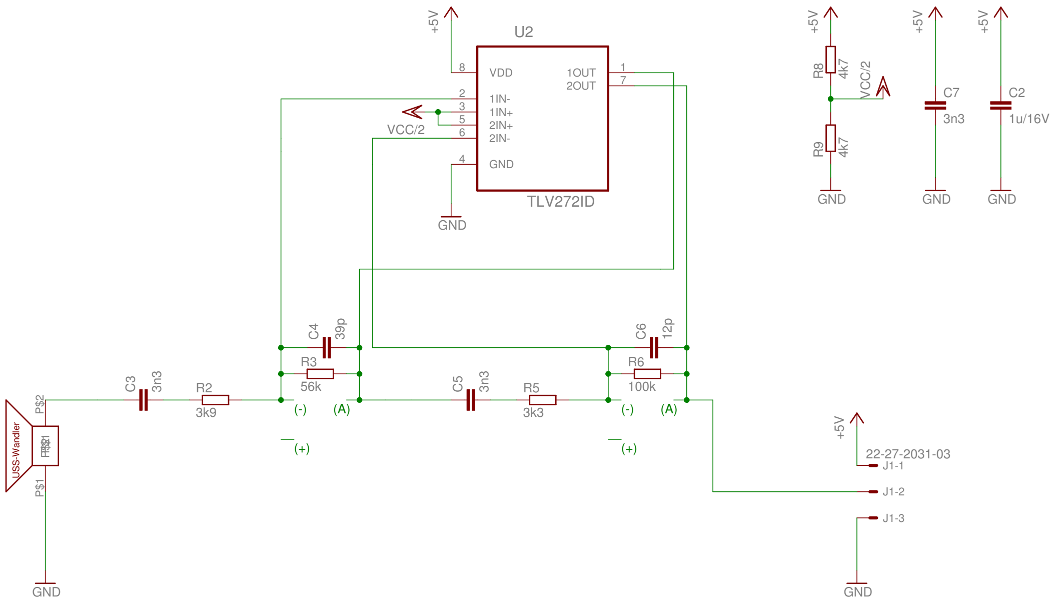


Figure 8.2: Schematic: Receiver circuit v2.0

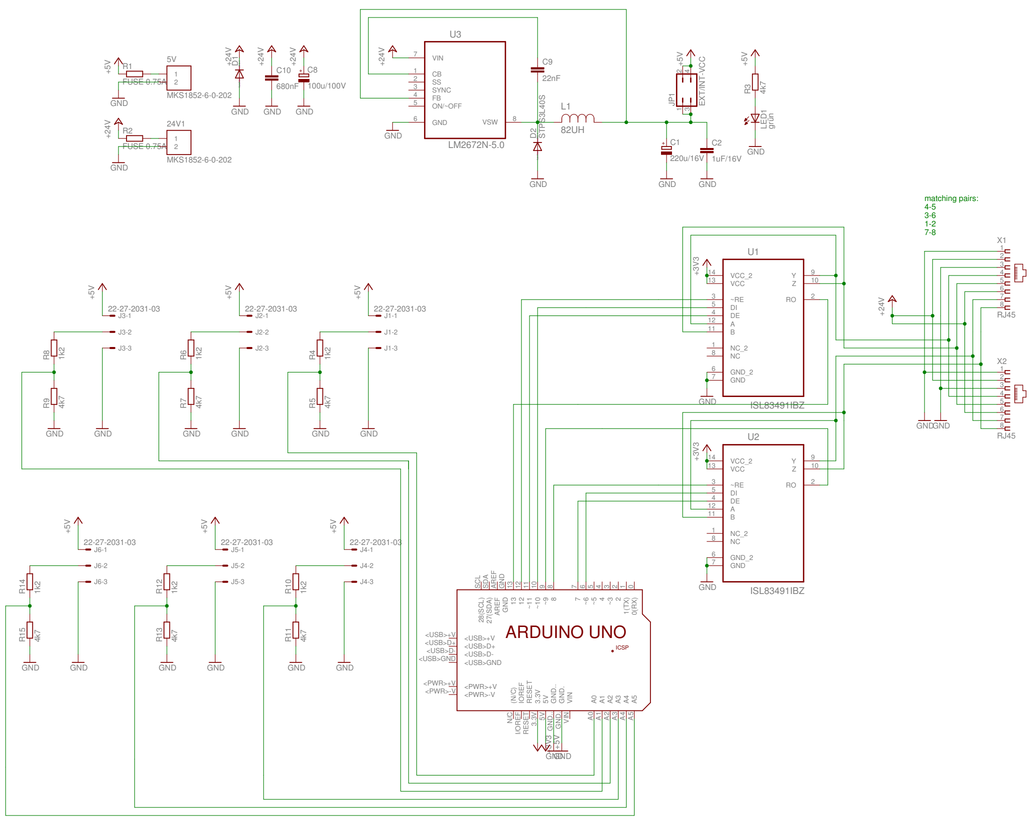


Figure 8.3: Schematic: FPGA Adapter-Board v1.1

8.2 Test Point Documentation

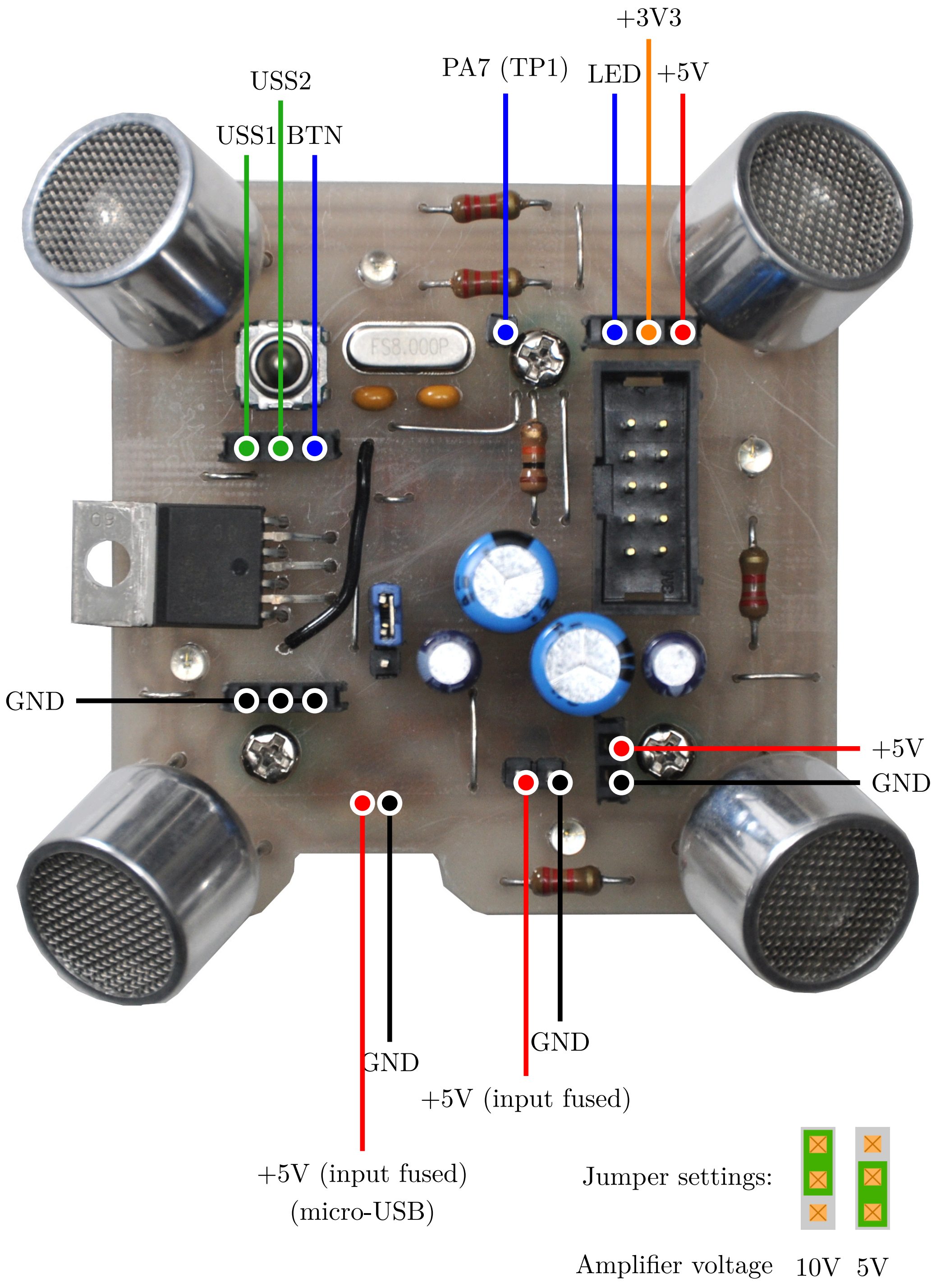
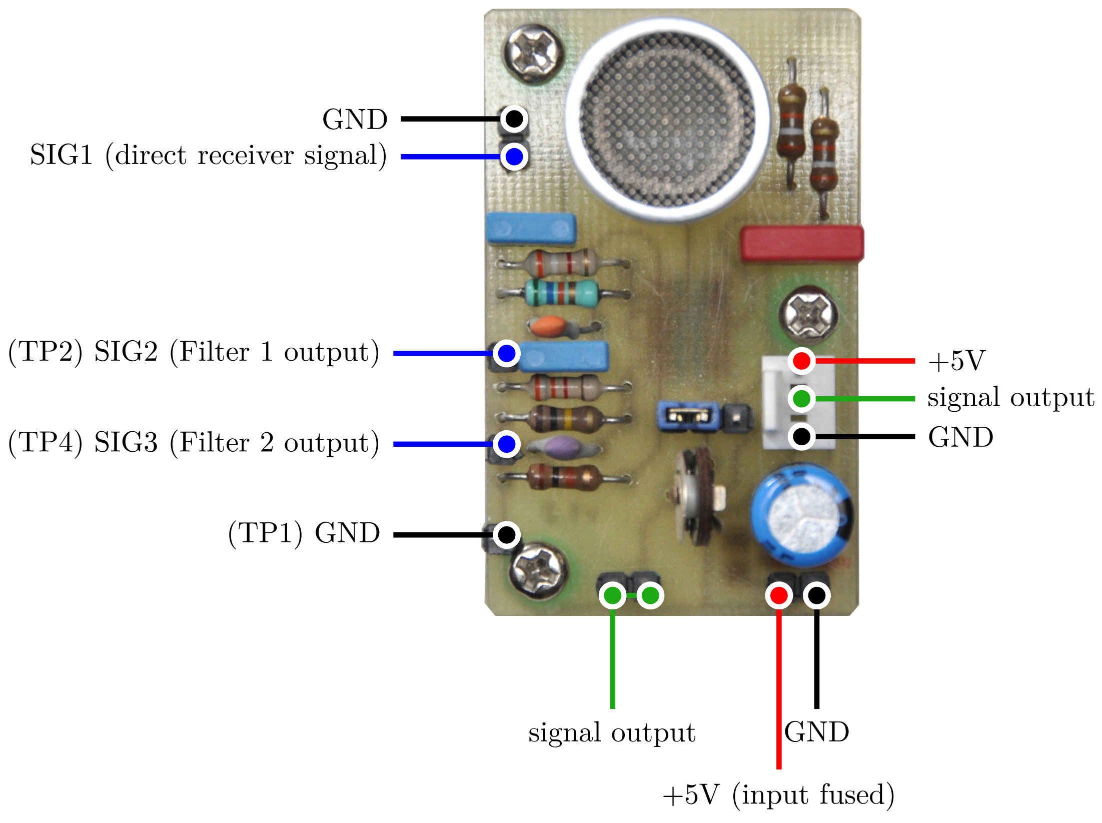


Figure 8.4: Test point documentation: Transmitter circuit v2.2


GND
SIG1 (direct receiver signal)
(TP2) SIG2 (Filter 1 output)
(TP4) SIG3 (Filter 2 output)
(TP1) GND
+5V
signal output
GND
signal output
GND
+5V (input fused)

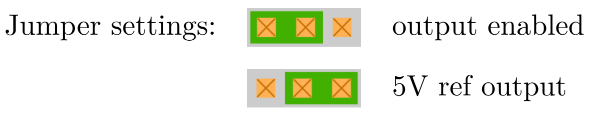
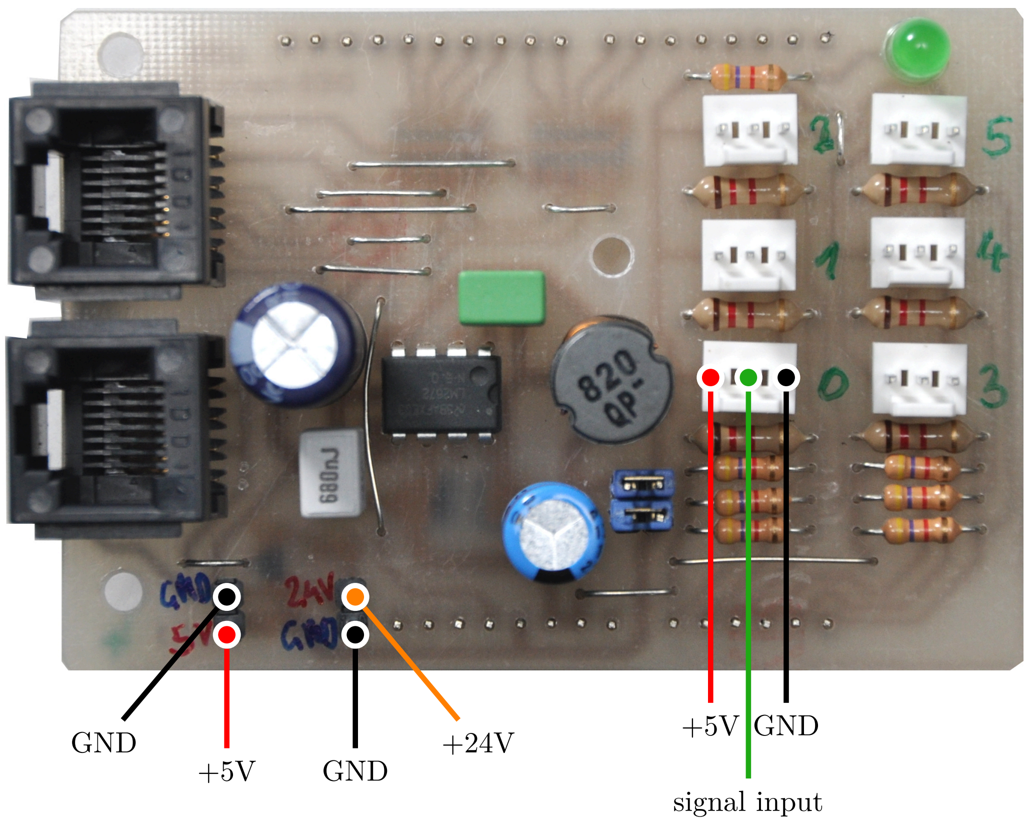
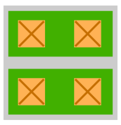


Figure 8.5: Test point documentation: Receiver circuit v1.0



Jumper settings:


power via RJ45 enabled

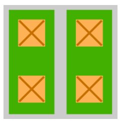

external power source

Figure 8.6: Test point documentation: Adapter-Board v1.1

8.3 Listings

```
function [ p1, p2 ] = tdoa_solve( pi, pj, pk, pl, rvals )
%determine x, y, and z of mobile object by difference of signal propagation
%delays

% unit for time: 1 s
% unit for distance: 1 m
% unit for speed: 1 m/s
v = 340;

xi=pi(1); yi=pi(2); zi=pi(3); % position for base station 'i'
xj=pj(1); yj=pj(2); zj=pj(3); % position for base station 'j'
xk=pk(1); yk=pk(2); zk=pk(3); % position for base station 'k'
xl=pl(1); yl=pl(2); zl=pl(3); % position for base station 'l'

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% calculation is taken from the following paper:
%
% Ralph Bucher and D. Misra. "A Synthesizable VHDL Model of the Exact
% Solution for Three-dimensional Hyperbolic Positioning System". In: VLSI
% Design 15.2 (202), pp. 507-520. doi: 10.1080/1065514021000012129.

xji=xj-xi;
xki=xk-xi;
xjk=xj-xk;
xlk=xl-xk;
xik=xi-xk;
yji=yj-yi;
yki=yk-yi;
yjk=yj-yk;
ylk=yl-yk;
yik=yi-yk;
zji=zj-zi;
zki=zk-zi;
zik=zi-zk;
zjk=zj-zk;
zlk=zl-zk;

rij = v*(rvals(1));    % = abs((v*(ti-tj)));
rik = v*(rvals(2));    % = abs((v*(ti-tk)));
rkj = v*(rvals(3));    % = abs((v*(tk-tj)));
rkl = v*(rvals(4));    % = abs((v*(tk-tl)));
ril = v*(rvals(5));    % = abs((v*(ti-tl)));
rjl = v*(rvals(6));    % = abs((v*(tj-tl)));

s9 =rik*xji-rij*xki;
s10=rij*yki-rik*yji;
s11=rik*zji-rij*zki;
s12=(rik*(rij*rij + xi*xi - xj*xj + yi*yi - yj*yj + zi*zi - zj*zj) -rij*(rik*rik + ⤸
    xi*xi - xk*xk + yi*yi - yk*yk + zi*zi - zk*zk))/2;
s13=rkl*xjk-rkj*xlk;
s14=rkj*ylk-rkl*yjk;
s15=rkl*zjk-rkj*zlk;
s16=(rkl*(rkj*rkj + xk*xk - xj*xj + yk*yk - yj*yj + zk*zk - zj*zj) -rkj*(rkl*rkl + ⤸
    xk*xk - xl*xl + yk*yk - yl*yl + zk*zk - zl*zl))/2;

a= s9/s10;
b=s11/s10;
```

```
c=s12/s10;
d=s13/s14;
e=s15/s14;
f=s16/s14;
g=(e-b)/(a-d);
h=(f-c)/(a-d);
i=(a*g)+b;
j=(a*h)+c;
k=rik*rik+xi*xi-xk*xk+yi*yi-yk*yk+zi*zi-zk*zk+2*h*xki+2*j*yki;
l=2*(g*xki+i*yki+zki);
m=4*rik*rik*(g*g+i*i+1)-l*l;
n=8*rik*rik*(g*(xi-h)+i*(yi-j)+zi)+2*l*k;
o=4*rik*rik*((xi-h)*(xi-h)+(yi-j)*(yi-j)+zi*zi)-k*k;
s28=n/(2*m);
s29=(o/m);
s30=(s28*s28)-s29;
root=sqrt(s30);

z1=s28+root;
z2=s28-root;
x1=g*z1+h;
x2=g*z2+h;
y1=a*x1+b*z1+c;
y2=a*x2+b*z2+c;

p1 = [ x1 y1 z1 ];
p2 = [ x2 y2 z2 ];

end
```

Listing 8.1: TDOA calculation function

```
function [ rvals ] = tdoa_simulate_rvals( pi, pj, pk, pl, m )
%simulate difference of signal propagation delays by known position of
%mobile object

% unit for time: 1 s
% unit for distance: 1 m
% unit for speed: 1 m/s
v = 340;
eps = 0.0;

xm=m(1); ym=m(2); zm=m(3); % position for mobile object
xi=pi(1); yi=pi(2); zi=pi(3); % position for base station 'i'
xj=pj(1); yj=pj(2); zj=pj(3); % position for base station 'j'
xk=pk(1); yk=pk(2); zk=pk(3); % position for base station 'k'
xl=pl(1); yl=pl(2); zl=pl(3); % position for base station 'l'

% signal propagation delays (add eps to avoid r_mn which are zero)
ti=sqrt((xm - xi)^2 + (ym - yi)^2 + (zm - zi)^2 + 1*eps)/v;
tj=sqrt((xm - xj)^2 + (ym - yj)^2 + (zm - zj)^2 + 2*eps)/v;
tk=sqrt((xm - xk)^2 + (ym - yk)^2 + (zm - zk)^2 + 3*eps)/v;
tl=sqrt((xm - xl)^2 + (ym - yl)^2 + (zm - zl)^2 + 4*eps)/v;

rij=(ti-tj);
rik=(ti-tk);
rkj=(tk-tj);
rkl=(tk-tl);
ril=(ti-tl);
```

```
rjl=(tj-tl);

rvals = [rij rik rkj rkl ril rjl];

end
```

Listing 8.2: Calculate signal propagation timings

```
% simulate TDOA calculation

m =  [ 8 3 1 ]; % position for mobile object 'm' in format [x y z], unit: 1m
p0 = [ 0 0 0 ]; % position for base station 'P_0' in format [x y z], unit: 1m
p1 = [ 0 5 0 ]; % position for base station 'P_1' in format [x y z], unit: 1m
p2 = [ 7 5 0 ]; % position for base station 'P_2' in format [x y z], unit: 1m
p3 = [ 5 0 2 ]; % position for base station 'P_3' in format [x y z], unit: 1m

rvals = tdoa_simulate_rvals( p0, p1, p2, p3, m );
[ m1, m2 ] = tdoa_solve( p0, p1, p2, p3, rvals );
position_of_mobile_object = m1
```

Listing 8.3: Script for TDOA simulation

```
% simulate TDOA calculation

m =  [ 8 1 3.5 ]; % position for mobile object 'm' in format [x y z], unit: 1m
p0 = [ 0 0 0 ]; % position for base station 'P_0' in format [x y z], unit: 1m
p1 = [ 0 5 0 ]; % position for base station 'P_1' in format [x y z], unit: 1m
p2 = [ 7 5 0 ]; % position for base station 'P_2' in format [x y z], unit: 1m
p3 = [ 5 0 2 ]; % position for base station 'P_3' in format [x y z], unit: 1m

rvals = tdoa_simulate_rvals( p0, p1, p2, p3, m ) % calculate TDOA timings for m
[ m1, m2 ] = tdoa_solve( p0, p1, p2, p3, rvals );
m1 % possible location m1
m2 % possible location m2

rvals1 = tdoa_simulate_rvals( p0, p1, p2, p3, m1 ) % calculate TDOA timings for m1
rvals2 = tdoa_simulate_rvals( p0, p1, p2, p3, m2 ) % calculate TDOA timings for m2

rvals1 - rvals2 % difference for timings is less than 1.0e-17.
```

Listing 8.4: Script for TDOA simulation

```
function [ particles, estimate ] = particle_solve_one_step( N, particles, p, dvals, ⤸
    sigma, max_range, v_sound )
%One step of particle solver for TDOA

% distance and weight calculation
w = zeros(N,1);
for i = 1:N % iterate over all particles
    d_particle = tdoa_sim_d_vals( p, particles(i,:), 0 ); % calculate TDOA values ⤸
        for current particle

    diff = dvals - d_particle;
    diff2 = diff .* diff;

    w(i) = 1/sum(sum(diff2)); % weight = likelyhood for particle to be chosen
end;
w = w / sum(w);

 % current position estimate based on weighted particle positions
weighted_x = w .* particles(:,1);
weighted_y = w .* particles(:,2);
weighted_z = w .* particles(:,3);
estimate = [ sum(weighted_x), sum(weighted_y), sum(weighted_z)]; % estimated current ⤸
    position

w_sums = zeros(N,1);
ssum = 0;
for i = 1:N % iterate over all particles
    ssum = ssum + w(i);
    w_sums(i) = ssum; % sum of weights (up to index i), needed for resampling (see ⤸
        below)
end;

% resampling
particles_new = particles;
for i = 1:N % iterate over all particles
    rrand = rand();
    for j = 1:N % iterate over all particles
        if w_sums(j) >= rrand
            particles_new(i,:) = particles(j,:) + ([ randn() randn() randn() ] * ⤸
                sigma);
            break;
        end;
    end;
end;
particles = particles_new;

end
```

Listing 8.5: Particle filter (one step)

```
% simulate TDOA calculation with error in one time difference measurement

%unitM = 1; % unit: 1m
unitM = 100; % unit: 1cm

%unitT = 1; % unit: 1s
unitT = (5.12*(1/(1000 * 1000))); % 1.28us

v_sound = (340 * unitM) * unitT;

p = csvread('data_pos_rec.csv');
p_count = size(p,1);
way = csvread('data_pos_waypoints.csv');
way_count = size(way,1);
dvals_read = csvread('data_dvals.csv');

max_range = [ 10 5 2 ] * unitM;      % maximum area for mobile object in format [x y z]
m = [ 1 2 0.3 ] * unitM;      % position for mobile object 'm' in format [x y z]
m = [ 8 2 1.5 ] * unitM;      % position for mobile object 'm' in format [x y z]

%sigma = 0.001 / unitT; % 1ms
```

```
%sigma = 0.0005 / unitT; % 0.5ms
sigma = 0.0001 / unitT; % 0.1ms
N = 300; % number of particles
dvals = tdoa_sim_d_vals( p, m, sigma ); % get distance values for simulated position
%dvals = dvals_read(1:p_count,1:p_count); % use actual measurements

particles = [ rand(N, 1) * max_range(1), rand(N, 1) * max_range(2), rand(N, 1) * ⤸
    max_range(3) ]; % randomize N particles

max_iter = 35; % limit the number of iterations for finding a position
sigma_iter = 5 * unitM; % jittering constant

estimates = [0 0 0];
spans = 0;
spans_x = 0;
spans_y = 0;
spans_z = 0;

for i = (1:max_iter)
    [ particles, estimate ] = particle_solve_one_step( N, particles, p, dvals, ⤸
        sigma_iter/i^2, max_range, v_sound);
    estimates(i,:) = estimate;

    span_x = max(particles(:,1)) - min(particles(:,1));
    span_y = max(particles(:,2)) - min(particles(:,2));
    span_z = max(particles(:,3)) - min(particles(:,3));

    span = span_x * span_y * span_z;
    spans(i) = span;
    spans_x(i) = span_x;
    spans_y(i) = span_y;
    spans_z(i) = span_z;
    [ (span / unitM^3) i ]
    if span < (0.1 * unitM)^3 % less than 10 cm in every dimension
        break;
    end;
end

particle_plot_est( N, max_range, particles, p, way, m, estimates, spans, spans_x, ⤸
    spans_y, spans_z);

d_val_max_range = sqrt(max_range(1)*max_range(1) + max_range(2)*max_range(2) + ⤸
    max_range(3)*max_range(3))/ v_sound * 0.9
```

Listing 8.6: Script for particle filter simulation

List of Figures

List of Tables

Acronyms

ADC Analog-to-digital converter.

AOA Angle of Arrival.

GPS Global Positioning System.

IMU Inertial Measurement Unit.

LGPL GNU Lesser General Public License.

LPS Local Positioning System.

MEMS Microelectromechanical system.

RSSI Received Signal Strength Indication.

SDK Software Development Kit.

SDR Software Defined Radio.

SPI Serial Peripheral Interface.

TDOA Time Difference of Arrival.

TOA Time of Arrival.

UAV Unmanned Aerial Vehicle.

UDP User Datagram Protocol.

UGV Unmanned Ground Vehicle.

uLFZ unbemanntes Luftfahrzeug.

Bibliography

[Abr16] Abracon LLC. *Catalog: Quartz Crystals.* [Online; last accessed on 2016-02-09]. 2016. URL: http://abracon.com/products.php.

[AF86] Nicholas Ayache and Oliver Faugeras. "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.1 (Jan. 1986), pp. 44–54. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767751.

[ARD13] ARDroneShow.com. *AR Drone 2.0 Ultraflight Upgrade Propeller Mod - Max Payload & Flight Tests.* [Online; last accessed on 2015-07-30]. 2013. URL: https://www.youtube.com/watch?v=z9zJQNpBC6Q.

[ARS05] Mahmoud Abdalla, Nima Razavi-Ghods, and Sana Salous. "Performance evaluation of sector antenna array for angle of arrival estimation". In: *11th International Symposium on Antenna Technology and Applied Electromagnetics.* June 2005, pp. 1–4. DOI: 10.1109/ANTEM.2005.7852012.

[Asc14] Ascending Technologies GmbH. *Technical specification: AscTec Pelican.* [Online; last accessed on 2015-07-30]. 2014. URL: http://wiki.asctec.de/display/AR/AscTec+Pelican.

[Bje+07] Jan D. Bjerknes, Wenguo Liu, Alan F. T. Winfield, and Chris Melhuish. *Low Cost Ultrasonic Positioning System for Mobile Robots.* [Online; last accessed on 2015-02-26]. 2007. URL: http://www.tankeogteknikk.no/images/stories/position/article.pdf.

[BM02] Ralph Bucher and Durga Misra. "A Synthesizable VHDL Model of the Exact Solution for Three-dimensional Hyperbolic Positioning System". In: *VLSI Design* 15.2 (202), pp. 507–520. DOI: 10.1080/1065514021000012129.

[CWZ11] Long Cheng, Cheng-Dong Wu, and Yun-Zhou Zhang. "Indoor robot localization based on wireless sensor networks". In: *IEEE Transactions on Consumer Electronics* 57.3 (Aug. 2011), pp. 1099–1104. ISSN: 0098-3063. DOI: 10.1109/TCE.2011.6018861.

[Dev15] Devantech, Ltd. *Technical specification: SRF05 - Ultra-Sonic Ranger.* [Online; last accessed on 2015-08-06]. 2015. URL: http://www.robot-electronics.co.uk/htm/srf05tech.htm.

[Fes15] Festo AG & Co. KG. *Brochure: eMotionButterflies – Ultralight flying objects with collective behaviour.* [Online; last accessed on 2015-04-03]. 2015. URL: http://www.festo.com/net/SupportPortal/Files/367913/Festo_eMotionButterflies_en.pdf.

[Fre18] Free Software Foundation, Inc. *Announcing ncurses 6.1.* [Online; last accessed on 2018-03-10]. 2018. URL: https://www.gnu.org/software/ncurses/.

[GG03] Fredrik Gustafsson and Fredrik Gunnarsson. "Positioning using time-difference of arrival measurements". In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on.* Vol. 6. Apr. 2003, pages. DOI: 10.1109/ICASSP.2003.1201741.

[GH28] Gill and Hecht. "Rotating-loop radio transmitters, and their application to direction-finding and navigation". In: *Journal of the Institution of Electrical Engineers* 66.375 (Mar. 1928), pp. 241–255. DOI: 10.1049/jiee-1.1928.0017.

[Gol67] Robert Gold. "Optimal binary sequences for spread spectrum multiplexing (Corresp.)" In: *IEEE Transactions on Information Theory* 13.4 (Oct. 1967), pp. 619–621. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1054048.

[Han74] Marsha J. Hannah. "Computer Matching of Areas in Stereo Images." [Online; last accessed on 2019-02-17]. PhD thesis. Stanford, CA, USA, 1974. URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/786720.pdf.

[Hid+17] Javier Hidalgo-Carrió, Daniel Hennes, Jakob Schwendner, and Frank Kirchner. "Gaussian process estimation of odometry errors for localization and mapping". In: *2017 IEEE International Conference on Robotics and Automation (ICRA).* May 2017, pp. 5696–5701. DOI: 10.1109/ICRA.2017.7989670.

[How+08] Johnathan How, Behihke Bethke, Adrian Frank, Daniel Dale, and John Vian. "Real-time indoor autonomous vehicle test environment". In: *Control Systems, IEEE* 28.2 (Apr. 2008), pp. 51–64. ISSN: 1066-033X. DOI: 10.1109/MCS.2007.914691.

[HTU09] Shunsuke Hijikata, Kenji Terabayashi, and Kazunori Umeda. "A simple indoor self-localization system using infrared LEDs". In: *2009 Sixth International Conference on Networked Sensing Systems (INSS).* June 2009, pp. 1–7. DOI: 10.1109/INSS.2009.5409955.

[INN15a] INNOC - Österreichische Gesellschaft für innovative Computerwissenschaften. *RobotChallenge.* [Online; last accessed on 2015-07-29]. 2015. URL: http://www.robotchallenge.org/.

[INN15b] INNOC - Österreichische Gesellschaft für innovative Computerwissenschaften. *RobotChallenge - Air Race Rules.* [Online; last accessed on 2015-07-29]. 2015. URL: http://www.robotchallenge.org/fileadmin/user_upload/_temp_/RobotChallenge/Reglement/RC-AirRace.pdf.

[ISO93] ISO – International Organization for Standardization. *Acoustics - Attenuation of sound during propagation outdoors*. Standard ISO 9613-1. Geneva, Switzerland: International Organization for Standardization, 1993. URL: `https://www.iso.org/standard/17426.html`.

[Jep13] Tobias Jeppe. *RS232 with buffer and wb*. [Online; last accessed on 2018-02-28]. 2013. URL: `https://opencores.org/project,rs232_with_buffer_and_wb`.

[JJ12] Seungho Jeong and Seul Jung. "Vision-based localization of a quad-rotor system". In: *9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. Nov. 2012, pp. 636–638. DOI: `10.1109/URAI.2012.6463105`.

[KSH11] Stefan Koenig, Mark T. Schmidt, and Christian Hoene. "Multipath mitigation for indoor localization based on IEEE 802.11 time-of-flight measurements". In: *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2011, pp. 1–8. DOI: `10.1109/WoWMoM.2011.5986392`.

[Lee75] Harry B. Lee. "A Novel Procedure for Assessing the Accuracy of Hyperbolic Multilateration Systems". In: *IEEE Transactions on Aerospace and Electronic Systems* AES-11.1 (Jan. 1975), pp. 2–15. ISSN: 0018-9251. DOI: `10.1109/TAES.1975.308023`.

[Lin15] Linear Technology. *LTspice IV*. [Online; last accessed on 2015-08-06]. 2015. URL: `http://www.linear.com/designtools/software/#LTspice`.

[MBP99] Pratap Misra, Brian P. Burke, and Michael M. Pratt. "GPS performance in navigation". In: *Proceedings of the IEEE* 87.1 (Jan. 1999), pp. 65–85. ISSN: 0018-9219. DOI: `10.1109/5.736342`.

[Mum+17] Naeem Mumtaz, Sana Arif, Nimra Qadeer, and Zeashan H. Khan. "Development of a low cost wireless IMU using MEMS sensors for pedestrian navigation". In: *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*. Mar. 2017, pp. 310–315. DOI: `10.1109/C-CODE.2017.7918948`.

[Nai+12] Gauri A. Naik, Madhavi P. Khedekar, Mahalakshmy Krishnamoorthy, Sayali D. Patil, and Rupali N. Deshmukh. "Comparison of RSSI techniques in Wireless Indoor Geolocation". In: *2012 National Conference on Computing and Communication Systems*. Nov. 2012, pp. 1–5. DOI: `10.1109/NCCCS.2012.6413008`.

[NAS71] NASA – National Aeronautics and Space Administration. *Lunar Roving Vehicle Operations Handbook*. LS006-002-2H. [Online; last accessed on 2019-02-07]. National Aeronautics and Space Administration. 1971. URL: `https://history.nasa.gov/alsj/LRV_OpsNAS8-25145Pt2.pdf`.

[Oks14] Irfan Oksar. “A Bluetooth signal strength based indoor localization method”. In: *2014 International Conference on Systems, Signals and Image Processing (IWSSIP)*. May 2014, pp. 251–254.

[Par15a] Parrot SA. *Technical specification: AR.Drone Flight Recorder.* [Online; last accessed on 2015-07-30]. 2015. URL: `http://ardrone2.parrot.com/apps/flight-recorder/`.

[Par15b] Parrot SA. *Technical specification: Parrot AR.Drone 2.0.* [Online; last accessed on 2018-12-03]. 2015. URL: `https://www.parrot.com/us/drones/parrot-ardrone-20-elite-edition#technicals`.

[Par16] Parrot SA. *Parrot Developers.* [Online; last accessed on 2018-02-28]. 2016. URL: `http://developer.parrot.com/`.

[Pra+17] Shemi Prazot, Avinoam Stern, Weiss Israel, S. Shlomo Marmor, Chagay Levi, Israel Yevilevich, Uriel Arad, Rony Man, and Benny Levi. “The medium and long term stability of the NAC atomic clock”. In: *2017 Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium (EFTF/IFCS)*. July 2017, pp. 162–168. DOI: `10.1109/FCS.2017.8088834`.

[Pro15] Pro-Wave Electronics Corporation. *Datasheet: Air Ultrasonic Ceramic Transducers.* 250ST/R160. [Online; last accessed on 2019-03-17]. Pro-Wave Electronics Corporation. Oct. 2015. URL: `http://www.prowave.com.tw/pdf/txall.pdf`.

[PT13] Shuo Pang and Ricardo Trujillo. “Indoor localization using ultrasonic time difference of arrival”. In: *2013 Proceedings of IEEE Southeastcon.* Apr. 2013, pp. 1–6. DOI: `10.1109/SECON.2013.6567445`.

[Qua71] Lynn H. Quam. “Computer Comparison of Pictures.” [Online; last accessed on 2019-02-17]. PhD thesis. Stanford, CA, USA, 1971. URL: `https://apps.dtic.mil/dtic/tr/fulltext/u2/785172.pdf`.

[Ras14] Raspberry Pi Foundation. *Technical specification: Raspberry Pi 1 Model B+.* [Online; last accessed on 2018-12-18]. 2014. URL: `https://www.raspberrypi.org/products/raspberry-pi-1-model-b-plus/`.

[Ras18] Raspberry Pi Foundation. *Welcome to Raspbian.* [Online; last accessed on 2018-12-21]. 2018. URL: `https://www.raspbian.org/`.

[Sen18] SensComp Global Components. *Application Note: Using Piezo Sensors.* Ultrasonic Ceramic Transducers. [Online; last accessed on 2019-02-07]. SensComp Global Components. Oct. 2018. URL: `http://www.senscomp.com/pdfs/using-piezo-sensors.pdf`.

[Sha49] Claude E. Shannon. “Communication in the Presence of Noise”. In: *Proceedings of the Institute of Radio Engineers* 37.1 (Jan. 1949), pp. 10–21. ISSN: 0096-8390. DOI: `10.1109/JRPROC.1949.232969`.

[Sta10] Evgeni Stavinov. *A Practical Parallel CRC Generation Method*. [Online; last accessed on 2018-01-28]. 2010. URL: `http://outputlogic.com/my-stuff/circuit-cellar-january-2010-crc.pdf`.

[Sza+13] Dávid Szalóki, Norbert Koszó, Kristóf Csorba, and Gábor Tevesz. "Marker localization with a multi-camera system". In: *2013 International Conference on System Science and Engineering (ICSSE)*. July 2013, pp. 135–139. DOI: `10.1109/ICSSE.2013.6614647`.

[The19] The MathWorks, Inc. *Matlab*. [Online; last accessed on 2019-02-07]. 2019. URL: `https://www.mathworks.com/products/matlab.html`.

[Wan+12] Ke Wang, Guanglei Huo, Lijun Zhao, Ruifeng Li, and Wei Wang. "A mobile robot self-localization approach based on unidirectional vision". In: *2012 International Conference on Mechatronics and Automation (ICMA)*. Aug. 2012, pp. 1966–1971. DOI: `10.1109/ICMA.2012.6285123`.

[Yar10] Rao Yarlagadda. *Analog and Digital Signals and Systems*. Springer US, 2010. ISBN: 978-1-4419-0034-0. DOI: `10.1007/978-1-4419-0034-0`.

[You17] Young+. *Welcome to RobotChallenge*. [Online; last accessed on 2018-03-16]. 2017. URL: `http://robotchallenge.org.cn/`.