



## TECHNISCHE DOKUMENTATION

# MEHRDIMENSIONALE RAUMERFASSUNG MITTELS ULTRASCHALL

31.03.2017

---

AUFTAGGEBER

FHNW, HOCHSCHULE FÜR TECHNIK

FACHBETREUUNG

RICHARD GUT, FHNW, INSTITUT FÜR MIKROELEKTRONIK  
MATTHIAS MEIER, FHNW, INSTITUT FÜR AUTOMATION

EXPERTE

DR. JÜRG STETTBACHER

TEAM

RAPHAEL MIJNSSEN, raphael.mijnssen@students.fhnw.ch  
JONAS PLÜSS, jonas.pluess@students.fhnw.ch

STUDIENGANG

ELEKTRO- UND INFORMATIONSTECHNIK

## Zusammenfassung

### Abstract

Ultrasound is commonly used by different technologies to collect information about certain objects. During this process, the echoes of sound waves transmitted by an ultrasonic transducer are measured. Signal processing is then used to gain the aforementioned information. If an array of ultrasonic transceivers is used (as opposed to just one), then the resulting sound beam will have a stronger directivity. By changing the signal phases of the excitation signals through a delay, the direction of the sound beam can be adjusted.

Since the ultrasonic transceivers used in air have a relatively low resonant frequency of about  $30 - 50\text{kHz}$ , it is possible to build such a phased array system cost-efficiently. For this project an ultrasonic phased array system containing a  $1 \times 8$  ultrasonic transceiver array has been developed. It scans the surrounding environment in a range of at least 5m with a depth resolution of about  $\pm 5\text{cm}$ . Thanks to a user interface, settings for the measurements can be specified. A range between  $\pm 30$  degrees can be scanned automatically. The results are presented as a heatmap-graph representing a picture of the surroundings and a line plot of the newest measured signal. Lastly, an algorithm finds the peaks in the scanned area and visualizes them in the heatmap-graph.

The system contains hardware in the form of a shield for the Arduino DUE and software. The firmware is written as a statemachine in C for the SAM3X8E microcontroller. The measured data is sent to a processing system via USB. This processing system is programmed in python and implemented as a web application. It is possible to run it on a Raspberry Pi. The user interface is accessible through a web browser.

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>2</b>
2.1 Grobkonzept . . . . .	2
2.1.1 Ultraschall Phased Array und Mikrocontroller . . . . .	3
2.1.2 Host . . . . .	4
2.1.3 Client . . . . .	4
2.2 Akustik . . . . .	5
2.2.1 Wellengleichung für akustische Schwingungen . . . . .	5
2.2.2 Die Schallgeschwindigkeit . . . . .	6
2.2.3 Die Dämpfung von Schall in Luft . . . . .	8
2.2.4 Reflexion und Transmission von Schall beim Auftreffen auf Grenzflächen .	8
2.2.5 Phased Array-Schallquellen . . . . .	9
2.2.6 Simulationen . . . . .	13
2.2.7 Amplitudenbelegung . . . . .	14
2.2.8 Phased Array-Sensoren . . . . .	15
2.3 Sampling . . . . .	16
2.3.1 Abtasttheorem . . . . .	16
2.3.2 Multiplexing . . . . .	17
2.4 Digitale Signalverarbeitung . . . . .	18
2.4.1 Zeitverschiebung im Frequenzbereich . . . . .	18
2.4.2 Upsampling . . . . .	18
2.4.3 Upsampling im Frequenzbereich . . . . .	19
2.4.4 Beamforming . . . . .	19
2.4.5 Enveloppe berechnen . . . . .	20
2.4.6 Downsampling . . . . .	20
<b>3 Hardware</b>	<b>21</b>
3.1 Überblick . . . . .	21
3.2 Ultraschall Phased Array . . . . .	22
3.3 Analogschaltung zum Phased Array . . . . .	24
3.3.1 Empfängerschaltung . . . . .	24
3.3.2 Senderschaltung . . . . .	25
3.4 ADC und Mikrocontroller . . . . .	25
3.5 PCB Design . . . . .	26

<b>4 Software</b>	<b>27</b>
4.1 Auswahl der Programmiersprachen und Schnittstellen . . . . .	28
4.2 Firmware auf dem Mikrocontroller . . . . .	30
4.2.1 Initialisierung und Hauptprogrammfluss – main.c . . . . .	30
4.2.2 Ultraschall Sender – transmitter.c . . . . .	31
4.2.3 Ultraschall Empfänger – receiver.c . . . . .	32
4.3 Hostseitige Software auf dem Desktop-Betriebssystem . . . . .	34
4.3.1 Hauptprogramm – server_flask.py . . . . .	35
4.3.2 Firmware Controller – firmware_controller.py . . . . .	35
4.3.3 Signalverarbeitung – dsp.py . . . . .	37
4.4 Clientseitige Software auf dem Desktop-Betriebssystem . . . . .	39
4.4.1 HTML und CSS . . . . .	40
4.4.2 Javascript – plot.js . . . . .	41
4.5 Entwicklungsumgebung . . . . .	42
4.5.1 Hardwareentwicklung . . . . .	42
4.5.2 Entwicklungsumgebung für die Firmware . . . . .	43
4.5.3 Installation auf dem Desktop Betriebssystem . . . . .	44
4.5.4 Bedienungsanleitung . . . . .	44
4.5.5 Lizenzen . . . . .	45
<b>5 Test</b>	<b>46</b>
5.1 Messung der sendeseitigen Richtcharakteristik . . . . .	46
5.2 Messung der sendeseitigen Richtcharakteristik mit Amplitudenbelegung . . . . .	48
5.3 Messung der empfangsseitigen Richtcharakteristik . . . . .	50
5.3.1 Messung der Empfangscharakteristik bei einem Sendewinkel von 0 Grad .	51
5.3.2 Messung der Empfangscharakteristik bei einem Sendewinkel von 10 Grad	53
5.3.3 Messung der Empfangscharakteristik bei einem Sendewinkel von 20 Grad	55
5.3.4 Messung der Empfangscharakteristik bei einem Sendewinkel von 30 Grad	57
5.4 Distanzmessungen . . . . .	59
5.5 Objekterkennung anhand ausgewählter Beispiele . . . . .	63
<b>6 Schlusswort</b>	<b>67</b>
<b>7 Bibliographie</b>	<b>69</b>
<b>A Anhang Schema</b>	<b>70</b>

<b>B Anhang Kosten</b>	<b>74</b>
<b>C Anhang Objekterkennung</b>	<b>75</b>
<b>D Anhang Lizenzen</b>	<b>79</b>
<b>E Anhang Aufgabenstellung</b>	<b>81</b>
<b>F Anhang Pflichtenheft</b>	<b>84</b>

## Tabellenverzeichnis

2.1	Schallgeschwindigkeit in verschiedenen Medien [1] . . . . .	7
2.2	Mögliche Abtastfrequenzen für die Unterabtastung . . . . .	17

## Abbildungsverzeichnis

2.1	Prinzip eines 1x8 Ultraschall Phased Array . . . . .	2
2.2	Blockschaltbild des Gesamtsystems . . . . .	3
2.3	Reflexion und Transmission von Schall beim Auftreffen auf Grenzflächen [2] . . . . .	9
2.4	Prinzip eines linearen Phased Arrays, exemplarisch für 5 Kanäle . . . . .	9
2.5	Richtcharakteristik für eine unterschiedliche Anzahl Elemente bei einem Abstand $d = \lambda/2$ und einem Sendewinkel von 20 Grad . . . . .	11
2.6	Richtcharakteristik für verschiedene Abstände bei $N = 8$ Elementen und einem Sendewinkel von 20 Grad . . . . .	11
2.7	Schärfe faktor für unterschiedliche Abstrahlwinkel, $N = 8$ . . . . .	12
2.8	Schärfe faktor für eine steigende Anzahl Elemente bei einem Abstrahlwinkel von 20 Grad . . . . .	12
2.9	Simulation für $N = 8$ , $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 0 Grad . . . . .	13
2.10	Simulation für $N = 8$ , $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 10 Grad . . . . .	13
2.11	Simulation für $N = 8$ , $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 20 Grad . . . . .	13
2.12	Richtcharakteristik für $N = 8$ , $d = 5.4\text{mm}$ und einem Sendewinkel von 20 Grad . .	15
2.13	Richtcharakteristik für $N = 8$ , $d = 5.4\text{mm}$ und einem Sendewinkel von 20 Grad . .	15
2.14	Signal nach der Unterabtastung mit $f_s = 32\text{kHz}$ . . . . .	18
2.15	Upsampling des unterabgetasteten Ultraschallsignals im Frequenzbereich . . . . .	19
2.16	Enveloppe eines Ultraschall-Echos. . . . .	20
3.1	Blockschaltbild des Gesamtsystems . . . . .	21
3.2	Fertig bestückter Print . . . . .	22
3.3	Richtcharakteristik mit $N = 8$ Elementen, im Abstand von $d = 0.635\lambda$ . . . . .	23
3.4	Schema der Operationsverstärkerschaltung . . . . .	25
4.1	Gesamtübersicht Software . . . . .	27
4.2	Hauptprogrammfluss der Software auf dem Mikrocontroller . . . . .	30
4.3	Übersicht Host . . . . .	34
4.4	Die Hostsoftware aus der Perspektive des Firmware Controllers . . . . .	36
4.5	Die grafische Benutzeroberfläche . . . . .	39

4.6 Struktur des Projektordners . . . . .	42
5.1 Messaufbau im reflexionsarmen Halbraum . . . . .	46
5.2 Richtcharakteristik des Sende-Arrays bei einem Winkel von $\theta = 0^\circ$ . . . . .	47
5.3 Richtcharakteristik des Sende-Arrays bei einem Winkel von $\theta = 10^\circ$ . . . . .	47
5.4 Richtcharakteristik des Sende-Arrays bei einem Winkel von $\theta = 20^\circ$ . . . . .	47
5.5 Richtcharakteristik des Sende-Arrays bei einem Winkel von $\theta = 30^\circ$ . . . . .	47
5.6 Richtcharakteristik des Sende-Arrays für eine Rechteckbelegung und $\theta = 10$ Grad	49
5.7 Richtcharakteristik des Sende-Arrays für eine cos-Belegung und $\theta = 10$ Grad . .	49
5.8 Richtcharakteristik des Sende-Arrays für eine $\cos^2$ -Belegung und $\theta = 10$ Grad . .	49
5.9 Richtcharakteristik des Sende-Arrays für eine Gaussbelegung und $\theta = 10$ Grad . .	49
5.10 Messaufbau zur empfangsseitigen Richtcharakteristik . . . . .	50
5.11 Maximum bei 2.00m und $0^\circ$ . . . . .	51
5.12 Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung	52
5.13 Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung	52
5.14 Richtcharakteristik des Empfangs-Arrays für eine $\cos^2$ -förmige Amplitudenbelegung	52
5.15 Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbele- gung . . . . .	52
5.16 Maximum bei 2.01m und $11^\circ$ . . . . .	53
5.17 Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung	54
5.18 Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung	54
5.19 Richtcharakteristik des Empfangs-Arrays für eine $\cos^2$ -förmige Amplitudenbelegung	54
5.20 Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbele- gung . . . . .	54
5.21 Maximum bei 2.01m und $21^\circ$ . . . . .	55
5.22 Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung	56
5.23 Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung	56
5.24 Richtcharakteristik des Empfangs-Arrays für eine $\cos^2$ -förmige Amplitudenbelegung	56
5.25 Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbele- gung . . . . .	56
5.26 Maximum bei 2.02m und $30^\circ$ . . . . .	57
5.27 Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung	58
5.28 Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung	58
5.29 Richtcharakteristik des Empfangs-Arrays für eine $\cos^2$ -förmige Amplitudenbelegung	58
5.30 Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbele- gung . . . . .	58

5.31 1m: Maximum bei 0.99m und $1^\circ$	59
5.32 2m: Maximum bei 2.00m und $0^\circ$	59
5.33 4m: Maximum bei 3.97m und $0^\circ$	60
5.34 6m: Maximum bei 5.96m und $1^\circ$	60
5.35 8m: Maximum bei 7.95m und $0^\circ$	61
5.36 10m: Maxima bei 9.91m und $1^\circ$ und bei 10.25m und $-1^\circ$	61
5.37 Messung für 10m mit skalierten Werten	62
5.38 Messung für 15m mit skalierten Werten	62
5.39 Schaumstoffstück bei 1.5m und $20^\circ$ , 20 Sendepulse	64
5.40 Schaumstoffstück bei 1.5m und $0^\circ$ , 20 Sendepulse	64
5.41 Person im reflexionsarmen Halbraum	65
5.42 2 Personen im Freien	65
5.43 Schaumsstoffstück parallel zum Array ausgerichtet	66
5.44 Mikrofonständer im reflexionsarmen Halbraum mit offener Türe	66
A.1 Schema	71
A.2 Top Copper (signal)	72
A.3 Inner Layer 1 (virtual ground 1.65V)	72
A.4 Inner Layer 2 (VCC 5.0V, VCC 3.3V)	73
A.5 Bottom Copper (ground)	73
C.1 Styroporplatte im reflexionsarmen Halbraum, Person, die sich davon weg bewegt	76
C.2 Styroporplatte im reflexionsarmen Halbraum, Sprung bei 0 Grad Sendewinkel	76
C.3 Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.4m seitlich vom Tuch	77
C.4 Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.3m hinter dem Tuch	77
C.5 Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.5m hinter dem Tuch	78
C.6 Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.1m hinter dem Tuch	78

## 1 Einleitung

Nicht grundlos ist die Fledermaus mit einem aussergewöhnlichen Sinn zur Orientierung im Raum ausgestattet. Durch die Erzeugung von hochfrequenten Schallwellen und der anschliessenden Auswertung der Echos kann sie Informationen über den Raum gewinnen. Dadurch wird die Fledermaus zum hervorragenden Jäger in der Dunkelheit. Dasselbe Verfahren machen sich unterschiedliche Technologien zunutze. In der Medizinaltechnologie, der zerstörungsfreien Materialprüfung oder beim Sonar werden so die relevanten Informationen gewonnen. Dabei werden Echos von Ultraschallwellen gemessen und geben Aufschluss über die zu untersuchenden Strukturen.

Wird statt eines einzelnen Ultraschallsenders ein Array aus mehreren Sendern verwendet, kann eine stärkere Richtwirkung des Schallkegels erzielt werden. Mithilfe von Zeitverzögerungen (Phasenverschiebungen) zwischen der Ansteuerung der einzelnen Sender kann die Richtung dieses Kegels in einen gewünschten Winkel gelenkt werden. Ein solches System wird als Ultraschall Phased Array bezeichnet.

Nicht nur in Flüssigkeiten oder Festkörpern, sondern auch in Luft kann ein solches Ultraschall Phased Array verwendet werden, um Informationen über die Umgebung zu gewinnen. Die für diesen Anwendungsfall benötigten Ultraschalltransceiver und deren typischerweise niedrige Eigenfrequenz im Bereich von 30 – 50kHz ermöglichen eine kostengünstige Umsetzung. Im Rahmen dieser Bachelor-Thesis wird ein solches Ultraschall Phased Array in Luft entwickelt, welches es ermöglicht, über den Winkel und die Distanz Objekte im Raum zu lokalisieren.

Im vorhergehenden Projekt 5 wurden theoretische Grundlagen als Basis für dieses Projekt erarbeitet und ein erster Prototyp angefertigt. Die Hardware des letzten Projektes besteht aus einem PCB-Shield für das Arduino DUE Board, welches über ein getrenntes 1x8 Sende- und 1x8 Empfangsarray verfügt. Gesteuert wird dieses vom SAM3X8E-Mikrocontroller auf dem Arduino DUE. Empfangene Ultraschallsignale werden über eine USB-Schnittstelle an ein weiterverarbeitendes Desktop-Betriebssystem gesendet.

Dieses System wird im Rahmen dieser Bachelor-Thesis weiterentwickelt. Um die verwendete Anzahl Ultraschalltransceiver zu verkleinern und damit die Kosten zu senken, wird die Elektronik so entwickelt, dass jeder Ultraschalltransceiver im Array sowohl als Sender als auch als Empfänger verwendet wird. Um die minimale Messdistanz zu verkleinern, können die einzelnen Ultraschalltransceiver nach dem Sendevorgang gedämpft werden. Nebenkeulen in der Richtcharakteristik können wahlweise durch verschiedene Amplitudenbelegungen unterdrückt werden. Die Firmware nimmt per USB-Schnittstelle Steuerbefehle entgegen und führt anhand dieser einen Messvorgang aus. Die Software für das Desktop-Betriebssystem ist als Webapplikation implementiert, welche hostseitig die Signalverarbeitung übernimmt und clientseitig ein User-Interface zur Verfügung stellt.

Über das User-Interface kann ein automatisierter Scan der Umgebung für einen gewünschten Winkelbereich zwischen  $\pm 30^\circ$  durchgeführt werden. Resultate werden parallel zum Scancvorgang als aktuelles Bild der Umgebung und als Linienplot der neusten Messung dargestellt. Ein Algorithmus durchsucht die eintreffenden Daten und berechnet daraus die Positionen von Maxima innerhalb des gescannten Winkelbereichs, welche den Positionen von reflektierenden Objekten im Raum entsprechen.

Die folgende technische Dokumentation beginnt mit dem Grobkonzept und den theoretischen Grundlagen, welche teilweise aus dem letzten Projekt übernommen wurden. Aufgeteilt in Hardware und Software wird daraufhin die detaillierte Umsetzung des Systems beschrieben. Abschliessend werden das Testkonzept und die erreichten Ergebnisse dokumentiert.

## 2 Grundlagen

Um einen Überblick über das Projekt zu verschaffen und den Einstieg ins Thema der Ultraschall Phased Arrays zu erleichtern, wird zuerst das Grobkonzept vorgestellt und danach werden die damit in Verbindung stehenden Grundlagen aus der Akustik und der Signalverarbeitung erklärt.

Damit eine Objekterkennung im Raum mittels Ultraschall möglich wird, muss ein schmaler, im Winkel verstellbarer Schallkegel erzeugt werden. Dafür sind mehrere Schallquellen nötig. Wenn der Schall auf ein Objekt auftrifft, wird ein Echo erzeugt. Aus einem solchen Echo können Informationen bezüglich Distanz und Winkel des Objekts im Raum gewonnen werden. Das Grundprinzip ist in Abbildung 2.1 dargestellt.

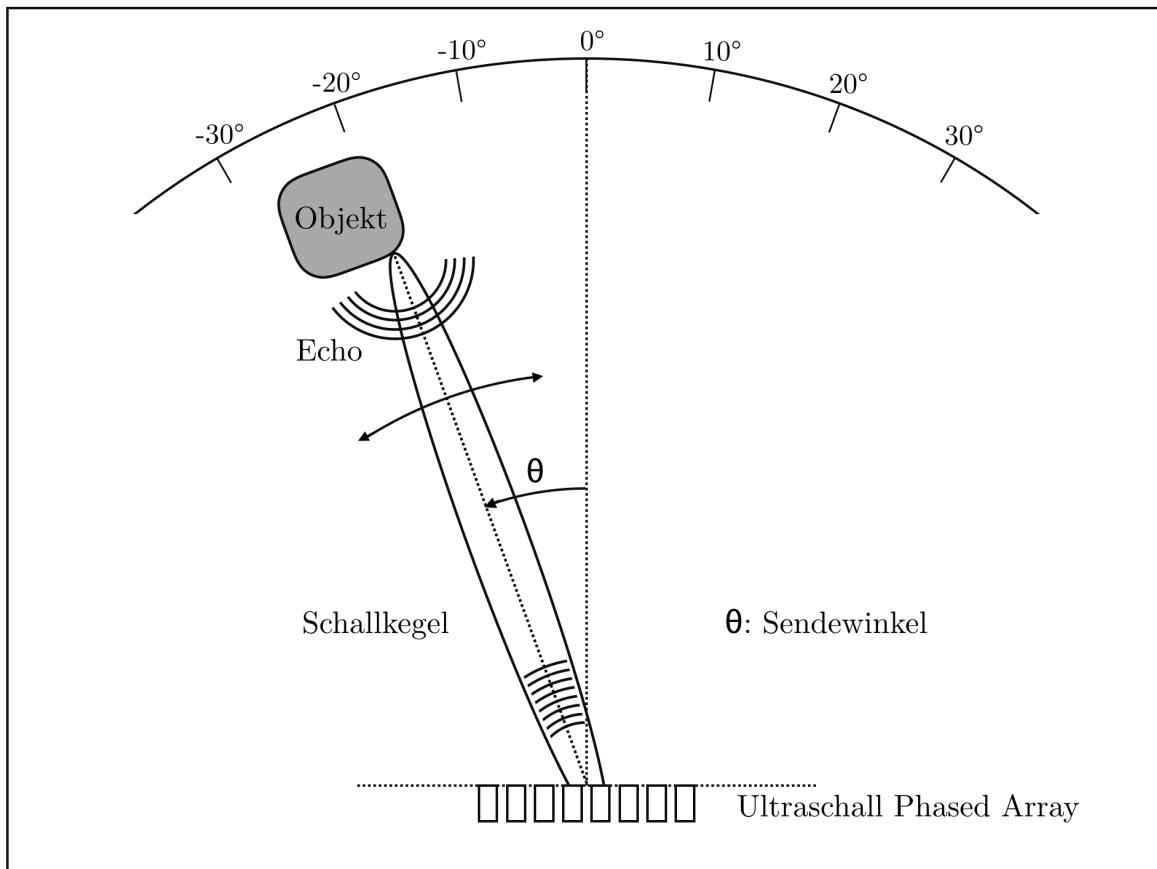


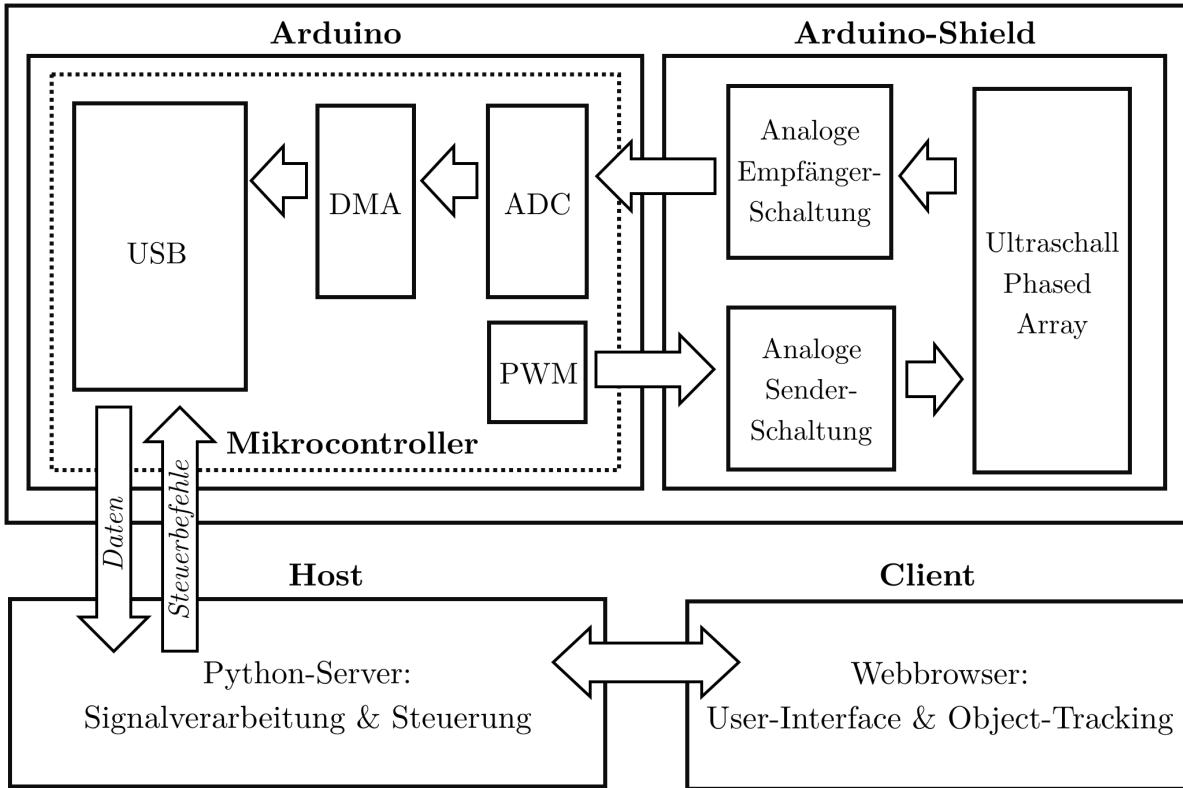
Abbildung 2.1: Prinzip eines 1x8 Ultraschall Phased Array

### 2.1 Grobkonzept

Das Ziel ist es, gerichtete Schallwellen im Raum zu erzeugen und aus deren Echos Informationen über Distanz und Winkel von Objekten im Raum zu gewinnen. Dafür wird ein System entwickelt, dass grob in drei verschiedene Komponenten gegliedert ist:

- das Ultraschall Phased Array inkl. zugehöriger Analogschaltung und deren Ansteuerung mittels Mikrocontroller
- ein Host-System, das die Signalverarbeitung übernimmt und als Server fungiert
- der Client, der als Schnittstelle zum Benutzer die Bedienung des Geräts ermöglicht und die gemessenen Resultate darstellt

In Abbildung 2.2 wird der grobe Aufbau des Gesamtsystems dargestellt.



**Abbildung 2.2:** Blockschaltbild des Gesamtsystems

### 2.1.1 Ultraschall Phased Array und Mikrocontroller

Das Phased Array wird mit Ultraschalltransceivern im Medium Luft bei einer Frequenz von 40kHz realisiert. Den Kern bilden die Analogschaltungen für den Empfänger und den Sender, sowie das 1x8 Array aus Ultraschalltransceivern. Jeder Ultraschalltransceiver kann einzeln in Schwingung gebracht, gedämpft oder als Sensor verwendet werden. Empfangene Signale werden dabei über eine Operationsverstärkerschaltung aufbereitet. Die Funktionsweise von Ultraschall Phased Arrays und deren Eigenschaften werden in den Kapiteln 2.2.5, 2.2.6, 2.2.8 und 2.2.7 genauer erläutert. Die Hardware ist als Shield für das Arduino DUE Board realisiert.

Die Software für den Mikrocontroller auf dem Arduino ist nicht mithilfe der Arduino IDE entwickelt, sondern mithilfe der arm-none-eabi-toolchain. Auf dem Mikrocontroller steuert eine in der Programmiersprache C umgesetzte Statemachine die Hardware. Dabei werden über die USB-Schnittstelle Aufträge mit Details zum gewünschten Sendevorgang entgegengenommen und ausgeführt. Danach werden die digitalisierten Rohdaten der empfangenen Echos wieder per USB zurückgesendet.

Der Mikrocontroller generiert während des Sendevorgangs Signale, die danach von der Senderschaltung verstärkt und von den Schallquellen ausgesendet werden. Die Empfängerschaltung wird dabei durch rechtzeitiges Entkoppeln geschützt.

Ein Auftrag an den Mikrocontroller enthält folgende Informationen:

- Sendewinkel
- Art der Amplitudenbelegung (siehe Kapitel 2.2.7)
- Anzahl Sendepulse
- Messzeit (Anzahl zu füllender Datenbuffer)

Die empfangenen 40kHz-Echos werden mit einer Abtastfrequenz von 32kHz unterabgetastet. Dies ist möglich, weil es sehr schmalbandige Signale sind. Details zur Unterabtastung sind im Kapitel 2.3.1 genauer erläutert.

### 2.1.2 Host

Der Host ist als Webserver in der Programmiersprache Python realisiert. Dieser kann von einem beliebigen pythonfähigen Desktop-Betriebssystem ausgeführt werden. Entwickelt wurde die Software auf einem GNU/Linux Desktop PC, wurde aber auf verschiedenen anderen Systemen (Microsoft Windows, Raspberry Pi) erfolgreich getestet. Er ist per USB mit dem Mikrocontroller verbunden. User-Eingaben nimmt er vom Client entgegen und führt entsprechende Befehle auf der Hardware aus. Empfangene Daten werden mithilfe der digitalen Signalverarbeitung aufbereitet und zur Darstellung an den Client weitergegeben. Damit ein gewünschter Winkelbereich automatisch gescannt werden kann, startet er auf Wunsch automatisiert die entsprechenden Aufträge über die USB-Schnittstelle und nimmt die Rohdaten entgegen. Als Webframework wird Flask verwendet.

Im Folgenden wird der Verlauf der Signalverarbeitung zusammengefasst. Die dabei verwendeten Methoden sind in den Kapiteln 2.3 und 2.4 detailliert beschrieben.

Die Rohdaten der 1x8 Kanäle werden als erstes in den Frequenzbereich transformiert. Dabei wird das Delay korrigiert, das durch das Multiplexing während der Analog-Digital-Wandlung entsteht. Eine weitere Zeitverschiebung muss durchgeführt werden, um die gewünschte empfangsseitige Richtwirkung zu erzielen. Die Abtastrate der 1x8 Kanäle wird daraufhin mittels Upsampling um den Faktor 4 erhöht. Um Rechenzeit zu sparen, wird auch das Upsampling im Frequenzbereich realisiert und die Daten danach in den Zeitbereich zurücktransformiert. Durch die Addition der acht Signale zu einem einzigen Signal wird die oben erwähnte Richtwirkung erzielt. Das so entstandene Signal enthält Informationen über die Position eines reflektierenden Objekts. Diese Informationen befinden sich in der Amplitude und der Zeitverschiebung von Wellenpaketen, die eine Trägerfrequenz von 40kHz aufweisen. Durch die Berechnung der Enveloppe wird diese Information hervorgehoben, während das Signal ins Basisband rückt. Die Enveloppe wird durch Betragsbildung des analytischen Signals berechnet, welches mithilfe der Hilbert-Transformation erzeugt wird. Da sich das Signal im Basisband befindet, kann die Datenmenge daraufhin mit Downsampling um den Faktor 4 reduziert werden. Dieses Signal wird schlussendlich zur Darstellung an den Client gesendet.

### 2.1.3 Client

Die Bedienung des Geräts durch den Benutzer erfolgt über das User-Interface, das als Webapplikation über einen Browser aufgerufen wird. Die dabei verwendeten Programmiersprachen sind Javascript, HTML und CSS. Im User-Interface wird durch den Benutzer der zu scannende Winkelbereich ausgewählt, die Schallgeschwindigkeit festgelegt und weitere Optionen eingestellt. Sobald diese Eingaben bestätigt sind, arbeitet das mit dem Server verbundene Ultraschall Phased Array-Gerät mit den aktualisierten Einstellungen im Hintergrund. Die Signale der empfangenen

Echos werden asynchron über Server-Sent-Events jeweils zum Zeitpunkt ihres Auftretens vom Client abgeholt und danach die graphischen Darstellungen aktualisiert.

Es werden zwei unterschiedliche Graphen erzeugt, welche beide zoom-/schiebbar sind und sich beim Eintreffen neuer Daten automatisch aktualisieren. Zur jeweiligen Position des Cursors werden zugehörige Daten wie Distanz oder Winkel mittels Tooltip dargestellt. Zum einen wird das aktuellste Messresultat als gewöhnlicher Linienplot dargestellt. Zum anderen wird Zeile für Zeile eine Heatmap generiert, welche die Signale farblich darstellt und so ein Bild der Umgebung erstellt, bzw. die Umgebung scannt.

Falls gewünscht, werden in Echtzeit eine variable Anzahl Maxima gesucht und auf der Heatmap mit zugehörigem Winkel und Distanz dargestellt. Dieses Maximum-Tracking ermöglicht die automatisierte Lokalisierung von Objekten im Raum.

## 2.2 Akustik

Das Interpretieren der empfangenen akustischen Echos ist die zentrale Aufgabe bei der Datenauswertung. Es müssen gerichtete Ultraschallwellen ausgesendet werden und anhand der empfangenen Echos Informationen über Distanz und Winkel gewonnen werden. Dabei bezeichnet Ultraschall nicht mehr hörbaren Schall, welcher einen Frequenzbereich von etwa 20kHz bis 10GHz umfasst. Schall mit höherer Frequenz wird als Hyperschall bezeichnet [2].

### 2.2.1 Wellengleichung für akustische Schwingungen

Ultraschall Phased Arrays machen sich die Wellencharakteristik von Schall zunutze. Im Folgenden wird deshalb die Herleitung der Ausbreitung von Schall im Raum erklärt. Schallwellen sind mechanische Schwingungen, die sich in Festkörpern, Gasen oder Flüssigkeiten ausbreiten. In Festkörpern können sich Schallwellen sowohl longitudinal als auch transversal zur Ausbreitungsrichtung bewegen im Gegensatz zu Fluiden, wo nur longitudinale Wellen vorkommen. Sie lassen sich mit verschiedenen Größen beschreiben, welche die Ausbreitung der Welle in Abhängigkeit der Zeit und des Orts definieren. Dafür werden oft die Druckschwankung  $p(r, t)$  um den Umgebungsdruck  $p_0$ , die Dichteschwankung  $\rho(r, t)$  um die Umgebungsichte  $\rho$  oder die Schallschnelle  $v(r, t)$  verwendet.

Die Herleitung der Wellengleichung, welche die Ausbreitung von Schall in einem Medium beschreibt, setzt kleine Schwankungen der oben erwähnten Größen im Vergleich mit  $p_0$ ,  $\rho$  und der Schallgeschwindigkeit voraus. Sie wird aus drei Grundgleichungen zusammengesetzt. Die folgende Herleitung beschränkt sich auf die Beschreibung des Schalls in  $x$ -Richtung im dreidimensionalen Raum.

Die Bewegungsgleichung (2.1) ergibt sich aus der Eulerschen Gleichung für die Rotation eines starren Körpers. Sie beschreibt die Beschleunigung eines kleinen Volumenelementes mit Dicke  $dx$  und den Seitenflächen  $dy \cdot dz$ . Eine senkrechte Druckausübung mit  $p(x)$  auf der einen Seitenfläche und  $-p(x+dx)$  auf der anderen kann weder ein Drehmoment noch eine Rotationsbeschleunigung auslösen. Deshalb wird das Volumenelement auf beiden Seitenflächen gleich beschleunigt [3]:

$$\begin{aligned} \rho \cdot dy \cdot dz \cdot dx \cdot \frac{\partial v}{\partial t} &= (p(x) - p(x+dx)) \cdot dy \cdot dz \\ \rho \cdot \frac{\partial v}{\partial t} &= -\frac{\partial p}{\partial x} \end{aligned} \tag{2.1}$$

Die Änderung des Drucks auf das Volumenelement  $V = \Delta x \cdot dy \cdot dz$  verursacht eine zeitliche Volumenänderung. Der Geschwindigkeitsgradient  $dv/dx$  beschreibt die örtliche Änderung der Verformungsgeschwindigkeit eines Körpers (in diesem Fall das Volumenelement  $V$ ) in  $x$ -Richtung.

Über die Kontinuitätsgleichung lässt sich dieser Geschwindigkeitsgradient mit der zeitlichen Volumenänderung verknüpfen. Dabei beschreiben  $v_1$  und  $v_2$  die Geschwindigkeiten der beiden Seitenflächen des Volumenelements  $V$  und  $S = dy \cdot dz$  deren Fläche. Gemäss [1] folgt:

$$\begin{aligned} \frac{\partial V}{\partial t} &= \frac{V(t_1 + \partial t) - V(t_1)}{\partial t} = \frac{S \cdot (\Delta x + (v_2 - v_1) \cdot \partial t) - S \cdot \Delta x}{\partial t} \\ &= \frac{S \cdot \Delta x + S \cdot ((v_1 + \frac{\partial v}{\partial x} \cdot \Delta x) - v_1) \cdot \partial t - S \cdot \Delta x}{\partial t} \\ \frac{\partial V}{\partial t} &= V \cdot \frac{\partial v}{\partial x} \end{aligned} \quad (2.2)$$

Das Kompressionsmodul  $K$  ist eine Materialeigenschaft und beschreibt die Volumenänderung  $dV$  bei einer Druckänderung  $dp$ . Es wird beschrieben durch:

$$\begin{aligned} \frac{\partial V}{\partial p} &= -\frac{V}{K} \\ \frac{\partial V}{\partial t} &= -\frac{V}{K} \cdot \frac{\partial p}{\partial t} \end{aligned} \quad (2.3)$$

Werden (2.2) und (2.3) gleichgesetzt, so entsteht ein Zusammenhang zwischen dem Geschwindigkeitsgradienten und der zeitlichen Druckveränderung. Wird dieser Zusammenhang nach der Zeit und die Bewegungsgleichung nach  $x$  abgeleitet, so können diese drei Grundgleichungen zur Wellengleichung 2.4 zusammengefasst werden [1].

$$\frac{\partial^2 p}{\partial t^2} = \frac{K}{\rho} \cdot \frac{\partial^2 p}{\partial x^2} \quad (2.4)$$

In dieser Differentialgleichung 2. Ordnung ist nur der Druck abhängig von Ort und Zeit. Sie beschreibt die Ausbreitung von akustischen Wellen. Lösungen dieser Wellengleichung sind alle Funktionen der allgemeinen Form [3]:

$$p(x, t) = f(x - c \cdot t) + g(x + c \cdot t) \quad (2.5)$$

Dabei beschreibt die Funktion  $f$  eine nach rechts laufende und die Funktion  $g$  eine nach links laufende Welle mit der Ausbreitungsgeschwindigkeit  $c$ . Mehrere solcher Wellen können sich überlagern, breiten sich jedoch unbeeinflusst von anderen Wellen aus.

Diese Herleitung funktioniert mathematisch etwas komplizierter auch für den dreidimensionalen Raum. Die Lösungen sind wiederum Wellen, die sich im Raum ausbreiten.

## 2.2.2 Die Schallgeschwindigkeit

Die Schallgeschwindigkeit  $c$  ergibt sich durch den Vergleich von (2.4) mit der Lösung der homogenen Wellengleichung, die im Wesentlichen gleich hergeleitet wird und dementsprechend der Wellengleichung für akustische Schwingungen sehr ähnlich ist [1]. Dies führt zur Beziehung:

$$c = \sqrt{\frac{K}{\rho}} \quad (2.6)$$

Medium	Dichte $\rho$ [ $\frac{\text{kg}}{\text{m}^3}$ ]	Schallgeschwindigkeit $c$ [ $\frac{\text{m}}{\text{s}}$ ]
Luft -20°C trocken	1,396	319
Luft 0°C trocken	1,293	331
Luft 20°C trocken	1,21	344
Luft 100°C trocken	0,947	387
Wasserstoff 0°C	0,090	1260
Wasserdampf 130°C	0,54	450
Wasser 0°C	1000	1400
Wasser 20°C	998	1480
Holz	600	4500
Stahl	7700	5050

Tabelle 2.1: Schallgeschwindigkeit in verschiedenen Medien [1]

Für ideale Gase kann das Kompressionsmodul in (2.6) durch den Isentropenexponent  $\kappa = c_p/c_v$ , der universellen Gaskonstante  $R = 8.31 \text{ J}/(\text{mol} \cdot \text{K})$ , der Molmasse des Gases  $M_m$  und der absoluten Temperatur  $T$  ausgedrückt werden. Damit ergibt sich für die Schallgeschwindigkeit in Gasen [3]:

$$c = \sqrt{\kappa \cdot \frac{R}{M_m} \cdot T} \quad (2.7)$$

In einem unendlich dünnen Stab können sich Kompressionen nur in einer Richtung ausbreiten. Anstelle des Kompressionsmoduls wird das Elastizitätsmodul  $E$  genutzt, um die Materialeigenschaft zu beschreiben. Aus (2.6) wird:

$$c = \sqrt{\frac{E}{\rho}} \quad (2.8)$$

Für allgemeine Festkörper gilt (2.8) nicht mehr. Denn es können sich auch Querspannungen ausbreiten, die sich seitlich zur Längswelle bewegen. Es folgen daraus zwei Schallgeschwindigkeiten für diese beiden Richtungen. Die Längswelle wird als Longitudinalwellen bezeichnet und die seitliche Welle als Transversalwelle. Sie bewegen sich mit folgender Geschwindigkeit:

$$\begin{aligned} c_L &= \sqrt{\frac{E \cdot (1 - \mu)}{\rho \cdot (1 + \mu) \cdot (1 - 2\mu)}} \\ c_T &= \sqrt{\frac{E}{2\rho \cdot (1 + \mu)}} \end{aligned} \quad (2.9)$$

In (2.9) beschreibt die Poissonszahl  $\mu$ , wie gut sich Querbewegungen ausbreiten können. Typische Werte liegen zwischen 0.2 und 0.5. Vergleicht man die Gleichungen in (2.8), so lässt sich erkennen, dass sich die Longitudinalwelle etwa doppelt so schnell ausbreitet wie die Transversalwelle [3].

Die Tabelle 2.1 enthält typische Werte für die Schallgeschwindigkeit. Für Festkörper wird die Geschwindigkeit der Longitudinalwellen angegeben.

Die Schallgeschwindigkeit in Luft wird sowohl bei steigender Temperatur als auch bei höherer Luftfeuchtigkeit grösser. Mit (2.7) kann die Schallgeschwindigkeit für beliebige Bedingungen berechnet werden. Die Luftfeuchtigkeit hat jedoch einen geringen Einfluss auf die Geschwindigkeit und ist vor allem für die Dämpfung wichtig, die im Folgenden erklärt wird.

### 2.2.3 Die Dämpfung von Schall in Luft

Schallwellen verlieren bei der Ausbreitung in einem Medium an Schallenergie. Diese wird zum einen wegen der Reibung direkt in Wärme umgesetzt, zum anderen geraten die Moleküle des Mediums in Bewegung, was nicht der Wellenübertragung dient. Dieser Energieverlust lässt sich mit dem Exponentialgesetz beschrieben, dabei bezeichnet  $I(r_0)$  die Schallintensität an einem bestimmten Ort  $r_0$  und  $\alpha$  [m/s] den Dämpfungskoeffizienten, der aufzeigt, wie stark Wellen bei der Ausbreitung in einem Medium an Energie verlieren [1].

$$p(r) = I(r_0) \cdot e^{-\alpha \cdot (r - r_0)} \quad (2.10)$$

Der Dämpfungskoeffizient  $\alpha$  ist keine Konstante und stark abhängig von weiteren Einflüssen. So wird der Dämpfungskoeffizient bei höheren Frequenzen grösser. Dies bedeutet, dass die Ein dringtiefe von Ultraschallwellen in ein Medium kleiner wird, je höher die Frequenz ist. In Luft ist die Feuchtigkeit ein weiterer Einflussfaktor auf den Dämpfungskoeffizienten.

Für tiefe Frequenzen von unter 10kHz nimmt die Dämpfung mit steigender Luftfeuchtigkeit tendenziell ab. Ab 100 kHz nimmt sie mit steigender Feuchtigkeit zu. [4]

### 2.2.4 Reflexion und Transmission von Schall beim Auftreffen auf Grenzflächen

Trifft eine Schallwelle auf ein anderes Medium auf, so wird ein Teil der Welle an der Grenzfläche reflektiert, der restliche Anteil dringt in das Medium ein. Für eine senkrecht einfallende Welle mit dem Schalldruck  $p_e$  gilt für die reflektierte Welle  $p_r$  und die transmittierte Welle  $p_d$  [2]:

$$\begin{aligned} r &= \frac{p_r}{p_e} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \\ p &= \frac{p_d}{p_e} = \frac{2Z_2}{Z_2 + Z_1} \end{aligned} \quad (2.11)$$

Dabei bezeichnet  $Z_m$  die Schallkennimpedanz  $Z_m = \rho \cdot c$  des Mediums  $m$ . Sie ist eine Materialeigenschaft und über die Dichte  $\rho$  sowohl von der Temperatur als auch vom Umgebungsdruck abhängig [1].

Falls der Einfallswinkel der Welle nicht senkrecht ist, verändern sich der Reflexions- und Transmissionskoeffizient in Funktion des Einfallswinkels. Dabei gilt für das Verhältnis zwischen dem Ein- und Austrittswinkel  $\alpha$  bzw.  $\beta$  in Abhängigkeit von der Schallgeschwindigkeiten  $c_1$  und  $c_2$ :

$$\frac{\sin(\alpha)}{\sin(\beta)} = \frac{c_1}{c_2} \quad (2.12)$$

In Abbildung 2.3 sind die drei möglichen Szenarien für die Übergänge zwischen gasförmigen Medien (G) und Festkörpern (F) dargestellt.

Dabei tritt beim Übergang in einen Festkörper sowohl eine Longitudinalwelle (L) als auch eine Transversalwelle (T) auf, die verschiedene Austrittswinkel aufweisen.

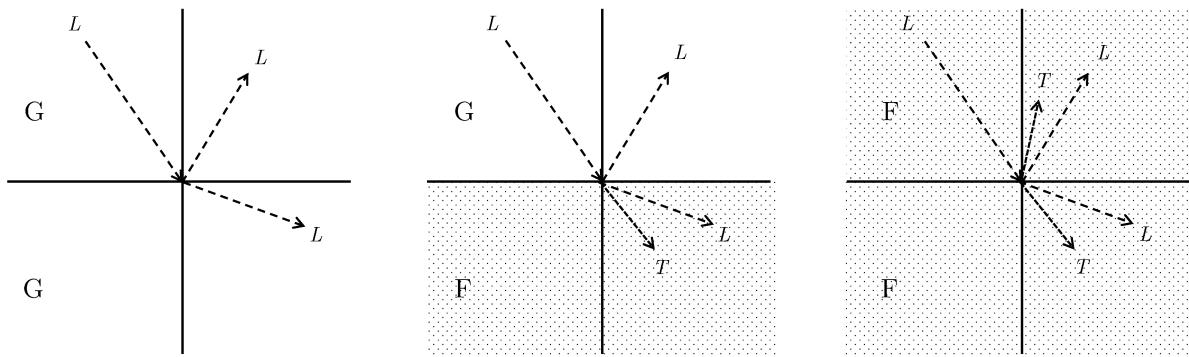


Abbildung 2.3: Reflexion und Transmission von Schall beim Auftreffen auf Grenzflächen [2]

### 2.2.5 Phased Array-Schallquellen

Anordnungen mehrerer Schallquellen nebeneinander werden als Array bezeichnet, dabei verändert sich die Richtcharakteristik des erzeugten Schalls. Es entsteht eine grosse Schallkeule, in welcher der Schalldruck am grössten ist, sowie mehrere Nebenkeulen. Falls die einzelnen Schallquellen phasenverzögert angesteuert werden, lässt sich der Hauptkegel durch die verzögerte Ansteuerung drehen. Das Prinzip wird in Abbildung 2.4 dargestellt.

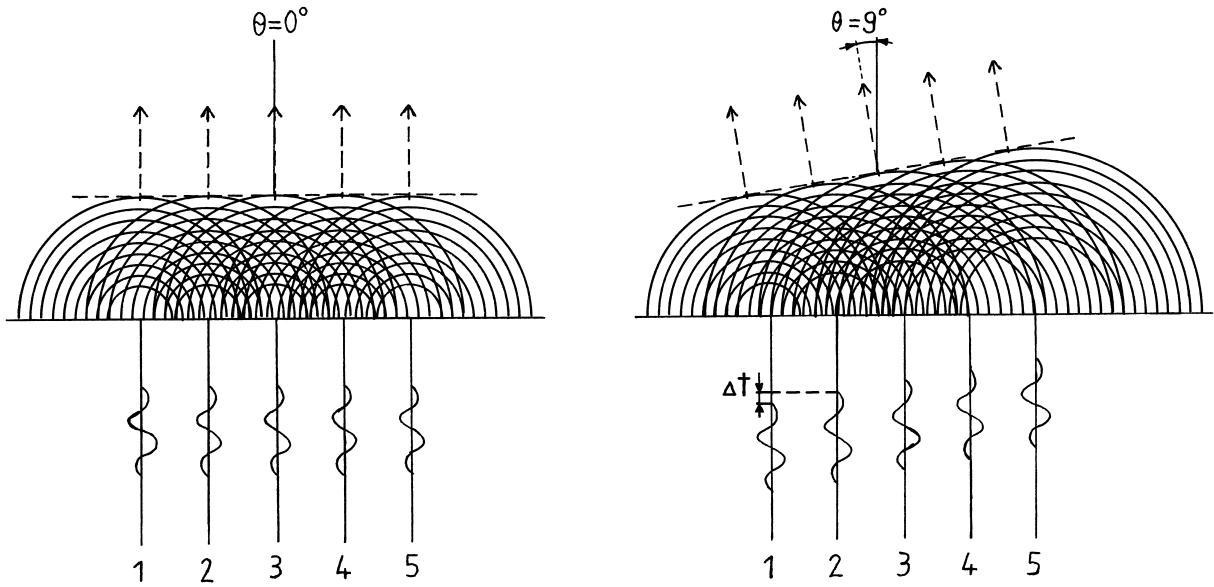


Abbildung 2.4: Prinzip eines linearen Phased Arrays, exemplarisch für 5 Kanäle

Am häufigsten wird unterschieden zwischen linearen Arrays, bei denen die Elemente in einer Reihe angeordnet sind (siehe Abbildung 2.4), und planaren Arrays, bei denen die Schallquellen in einer Ebene liegen.

Die nötige Zeitverzögerung zwischen den Kanälen, um die Richtwirkung um den Winkel  $\Theta$  zu erreichen, wird folgendermassen berechnet [5]:

$$\Delta\tau = \frac{d \cdot \sin(\Theta)}{c} \quad (2.13)$$

In (2.13) bezeichnet  $c$  die Schallgeschwindigkeit und  $d$  den Abstand zwischen den Elementen des Phased Arrays.

Der Schalldruck im Fernfeld eines linearen Phased Arrays ( $r \gg d$ ) in Abhängigkeit des Winkels, der Distanz und der Zeit lässt sich nach dem huygensschen Prinzip berechnen, dessen Herleitung in [6] zu finden ist. Für dieses Projekt interessant ist die Richtcharakteristik. Gemäss [5] ergibt sich nach einer Normierung auf den Schalldruck in Richtung des Sendewinkels folgende Beziehung zwischen dem Betrag des normierten Schalldrucks  $H(\varphi)$  und dem Winkel  $\varphi$ , in dem man zum Array steht:

$$H(\varphi) = \left| \frac{\sin\left(\frac{\pi \cdot d \cdot (\sin(\theta) - \sin(\varphi))}{\lambda} \cdot N\right)}{\sin\left(\frac{\pi \cdot d \cdot (\sin(\theta) - \sin(\varphi))}{\lambda}\right)} \right| \quad (2.14)$$

In (2.14) bezeichnet  $\theta$  den durch Phasenverzögerung eingestellten Sendewinkel,  $\lambda$  die Wellenlänge,  $d$  den Abstand zwischen den Sensoren und  $N$  die Anzahl Elemente im Array. Ausgewertet für eine steigende Anzahl Elemente  $N$  im Array und für verschiedene Abstände  $d$  zwischen den Schallquellen ergeben sich die Abbildungen 2.5 und 2.6.

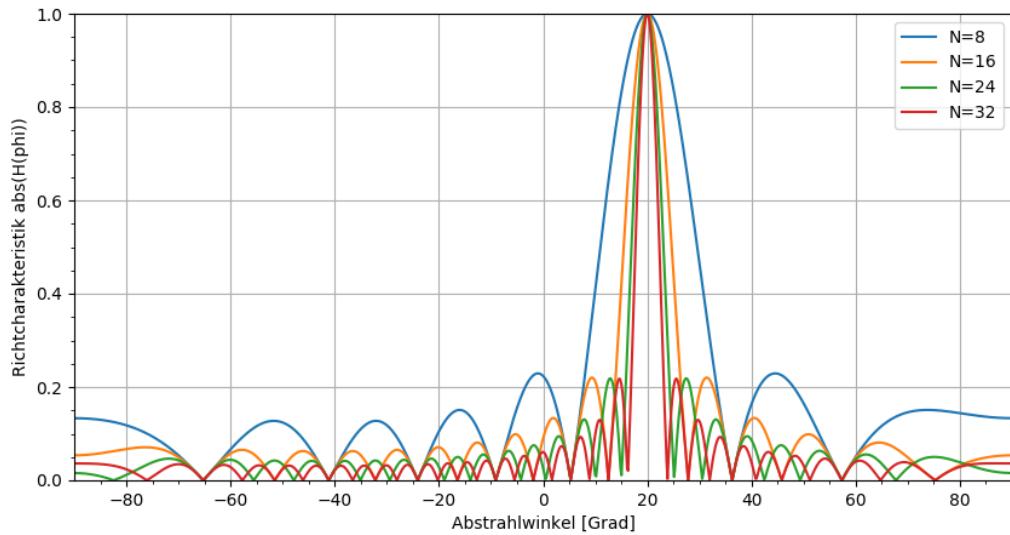
Wird die Anzahl Elemente  $N$  erhöht, so entsteht eine schmalere Hauptkeule und mehrere Seitenkeulen (siehe Abbildung 2.5). Ist der Abstand zwischen den Elementen grösser als die Wellenlänge  $\lambda$ , so entstehen viele Seitenkeulen, wovon einzelne gleich gross sind wie die Hauptkeule. Für Abstände kleiner als  $\lambda$  verschwinden diese grossen Nebenkeulen, jedoch nimmt die Breite der Hauptkeulen zu (siehe Abbildung 2.6).

Es kann zusätzlich ein Schärfefaktor  $q$  berechnet werden, der beschreibt, wie stark gerichtet die Hauptkeule ist [5]:

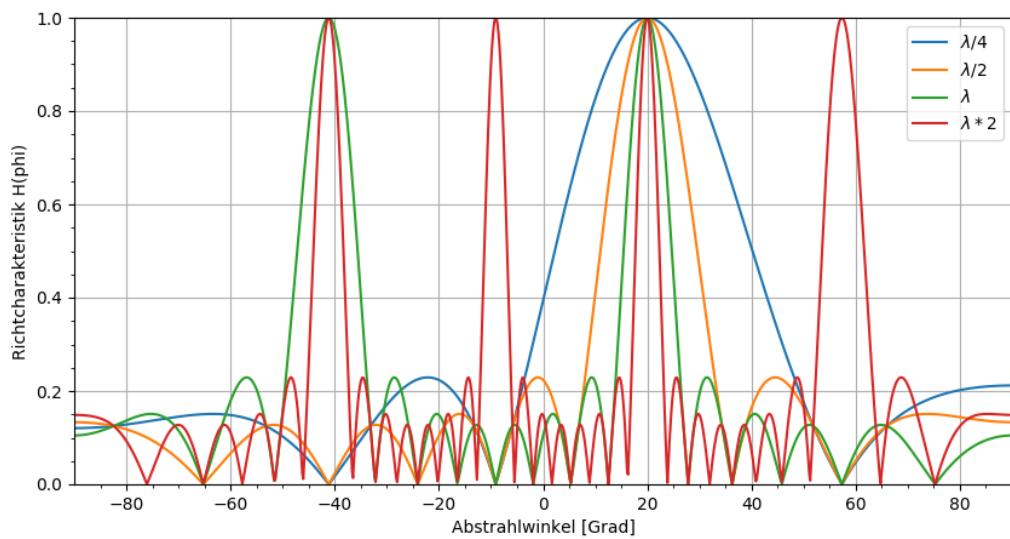
$$q = \frac{1}{\pi} \cdot \left[ \sin^{-1}\left(\sin(\theta) + \frac{\lambda}{N \cdot d}\right) - \sin^{-1}\left(\sin(\theta) - \frac{\lambda}{N \cdot d}\right) \right] \quad (2.15)$$

Ein kleineres  $q$  bedeutet eine steilere Hauptkeule. Ausgewertet für eine variierende Anzahl Elemente  $N$  und verschiedene Sendewinkel  $\theta$  sowie für die im Projekt verwendete Frequenz von 40kHz und einen Abstand von  $d = 5.4\text{mm}$  entstehen die Abbildungen 2.7 und 2.8.

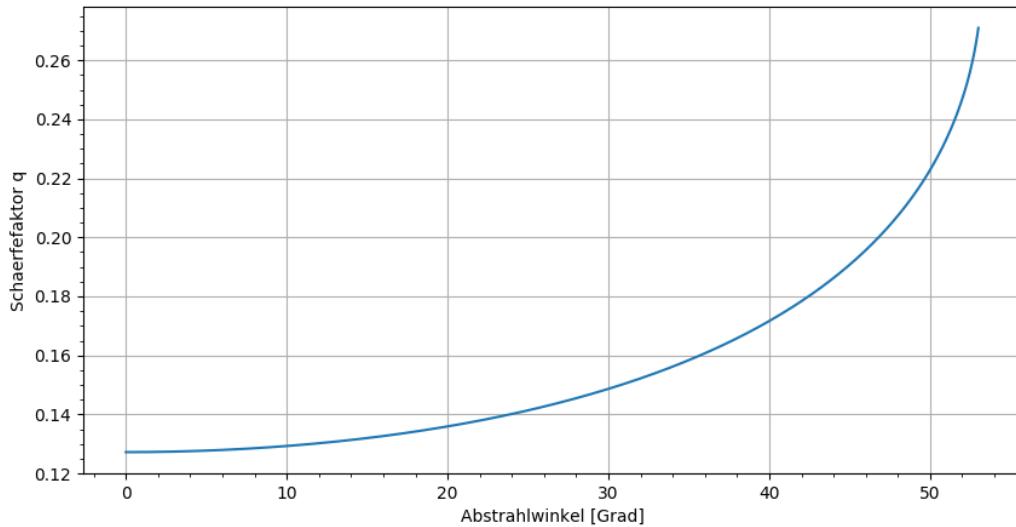
Je grösser der Sendewinkel ist, desto weniger steil wird die Hauptkeule. Mit einer grösseren Anzahl Elemente  $N$  kann  $q$  theoretisch unendlich klein werden. Den gleichen Effekt hätte eine kleinere Wellenlänge  $\lambda$  bei gleichbleibendem Abstand  $d$  oder umgekehrt eine Vergrösserung des Abstands. Dies ist jedoch nicht wünschenswert, weil dadurch grosse Nebenkeulen entstehen. Diese sind in Abbildung 2.6 dargestellt.



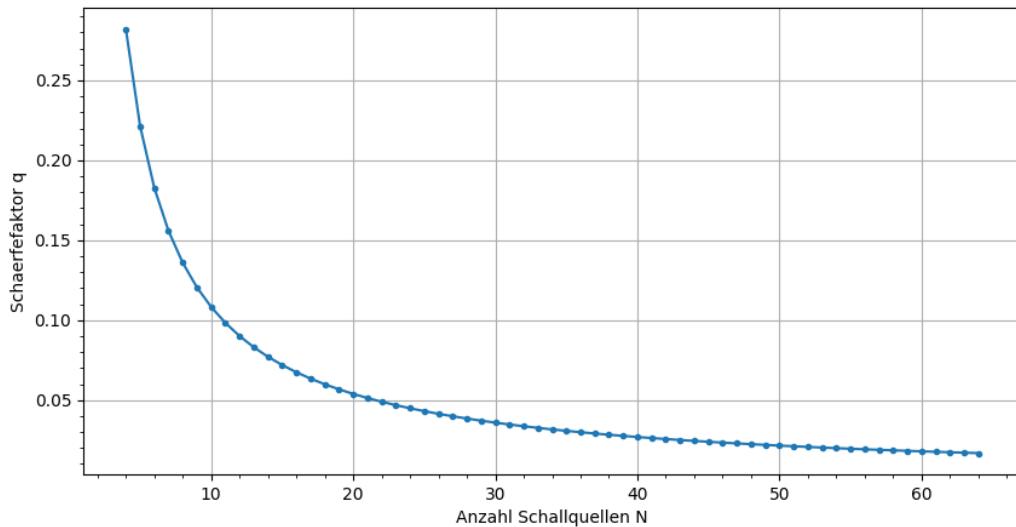
**Abbildung 2.5:** Richtcharakteristik für eine unterschiedliche Anzahl Elemente bei einem Abstand  $d = \lambda/2$  und einem Sendewinkel von 20 Grad



**Abbildung 2.6:** Richtcharakteristik für verschiedene Abstände bei  $N = 8$  Elementen und einem Sendewinkel von 20 Grad



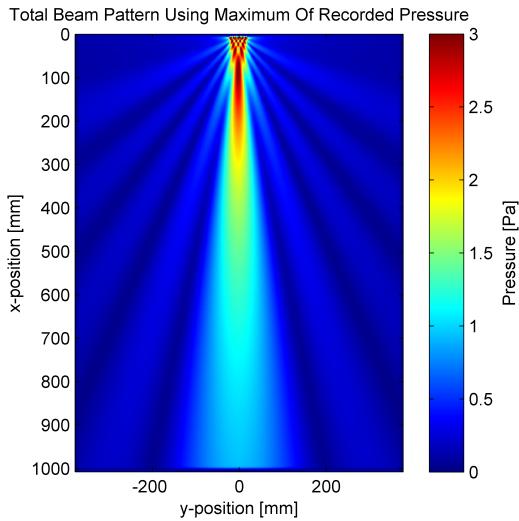
**Abbildung 2.7:** Schärfefaktor für unterschiedliche Abstrahlwinkel,  $N = 8$



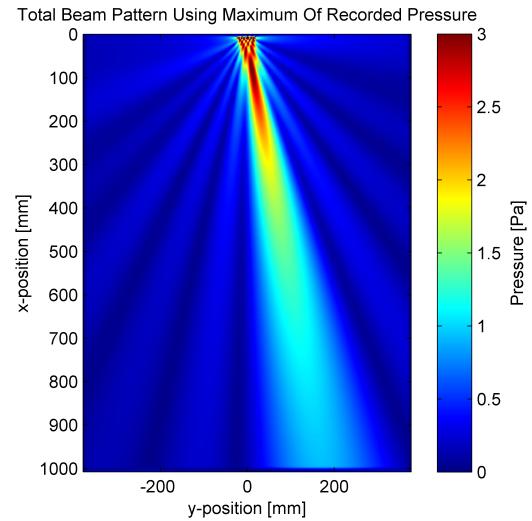
**Abbildung 2.8:** Schärfefaktor für eine steigende Anzahl Elemente bei einem Abstrahlwinkel von 20 Grad

## 2.2.6 Simulationen

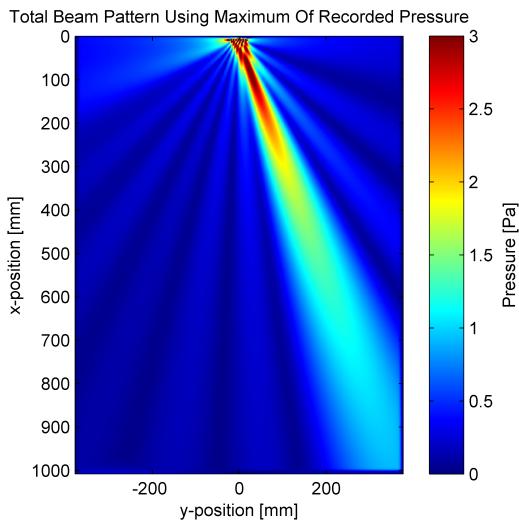
Die für das Projekt verwendeten Dimensionen (siehe Kapitel 3) werden mit der Opensource-Toolbox k-Wave simuliert. Dabei wird nun im Gegensatz zu den obigen Berechnungen auch das Nahfeld berücksichtigt. Als Schallquellen werden acht ideale Punktquellen verwendet, die ein Wellenpaket bestehend aus 25 Schwingungen aussenden. In den Abbildungen 2.9, 2.10 und 2.11 wird das Maximum des Schalldrucks auf 1m Distanz für verschiedene Abstrahlwinkel dargestellt.



**Abbildung 2.9:** Simulation für  $N = 8$ ,  $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 0 Grad



**Abbildung 2.10:** Simulation für  $N = 8$ ,  $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 10 Grad



**Abbildung 2.11:** Simulation für  $N = 8$ ,  $d = 5.4\text{mm}$ , bei einer Frequenz von 40kHz in Luft für einen Sendewinkel von 20 Grad

## 2.2.7 Amplitudenbelegung

Aus der Abbildung 2.5 im Kapitel 2.2.5 ist ersichtlich, dass durch die Erhöhung der Anzahl Elemente zwar die Hauptkeule und die Nebenkeulen schmäler werden und näher zusammenrücken, man sieht aber auch, dass sich die Intensität der Nebenkeulen nur gering verändert. Auch das Verhältnis zwischen dem Abstand der Elemente und der Wellenlänge haben, wie in Abbildung 2.6 im Kapitel 2.2.5 zu sehen ist, keinen grossen Einfluss auf die Intensität der Nebenkeulen. Will man die Nebenkeulen stärker dämpfen, kann dies durch eine Amplitudenbelegung der einzelnen Elemente erreicht werden. Dabei wird nicht jedes Element mit derselben Amplitude angesteuert, sondern von aussen nach innen symmetrisch mit unterschiedlichen Amplituden. Die Idee dazu stammt aus der Hochfrequenztechnik, wo Phased Array-Antennen zur Erzeugung einer starken Richtwirkung, zum Beispiel bei Datenübertragungs- oder Radarsystemen, eingesetzt werden. Die entsprechende Theorie stammt aus den Quellen [7] und [8].

Weisen alle Elemente dieselbe Amplitude auf, spricht man von einer Rechteckbelegung. Die resultierende Richtcharakteristik weist bei dieser Belegung eine  $\sin(x)/x$ -förmige Intensitätsverteilung auf. Dies legt die Vermutung nahe, dass die Amplitudenbelegung und die Intensitätsverteilung der Richtcharakteristik über die Fouriertransformation zusammenhängen. Diese Vermutung wird durch [7] bestätigt, wo sich auch gleich eine Anzahl üblicher Belegungsfunktionen findet, mit welchen die Nebenkeulen gut gedämpft werden können. In den Abbildungen 2.12 und 2.13 ist die Richtcharakteristik für vier verschiedene Amplitudenbelegungen dargestellt, im ersten Bild mit linearer Skalierung und normiert auf eins, in der zweiten Abbildung logarithmisch. Die Anzahl Elemente und der Abstand dazwischen entsprechen der in diesem Projekt entwickelten Hardware.

Da das Prinzip der Phased Array-Antennen auf konstruktiver und destruktiver Interferenz beruht, werden zur Berechnung die Anteile der einzelnen Strahler mit der Belegungsfunktion gewichtet und aufsummiert. Die Formel dazu lautet wie folgt [8]:

$$|H(\varphi)| = \left| \sum_{i=1}^N A_i \cdot e^{\frac{j2\pi d}{\lambda} \cdot (N-i) \cdot (\sin(\phi_0) - \sin(\phi))} \right| \quad (2.16)$$

Dabei ist  $N$  die Anzahl Elemente,  $d$  der Abstand zwischen den Elementen,  $\lambda$  die Wellenlänge,  $A_i$  der Gewichtungsfaktor der Amplitudenbelegung und  $\phi_0$  der eingestellte Sendewinkel. Für eine Rechteckbelegung ( $A_i = 1$ ) lässt sich diese Formel umformen zur Formel 2.14 aus dem Kapitel 2.2.5.

In den Abbildungen 2.12 und 2.13 ist gut zu sehen, dass eine bessere Nebenkeulenunterdrückung gleichzeitig in einer breiteren Hauptkeule resultiert. Dies könnte nun wiederum durch eine grössere Anzahl Elemente kompensiert werden. Die Rechteckbelegung weist zwar eine sehr schmale Hauptkeule auf, die ersten Nebenkeulen sind aber um lediglich etwas mehr als 12dB gedämpft. Mit einer  $\cos$ -förmigen Belegung kann die erste Hauptkeule bereits um ca. 18dB gedämpft werden, mit einer  $\cos^2$ -Belegung um etwa 26dB und mit einer Gaussbelegung um mehr als 30dB. Diese bessere Nebenkeulenunterdrückung erkauft man sich im Falle der Gaussbelegung mit einer ungefähr 1.5 Mal breiteren Hauptkeule als bei einer Rechteckbelegung, bezogen auf den 3dB-Öffnungswinkel.

Neben der Amplitudenbelegung hat auch die Richtcharakteristik der einzelnen Elemente einen Einfluss auf die Richtcharakteristik des ganzen Phased Arrays. Die resultierende Gesamtcharakteristik ist das Produkt aus der berechneten Richtcharakteristik des Arrays (mit Amplitudenbelegung) und der Richtcharakteristik eines einzelnen Elements.

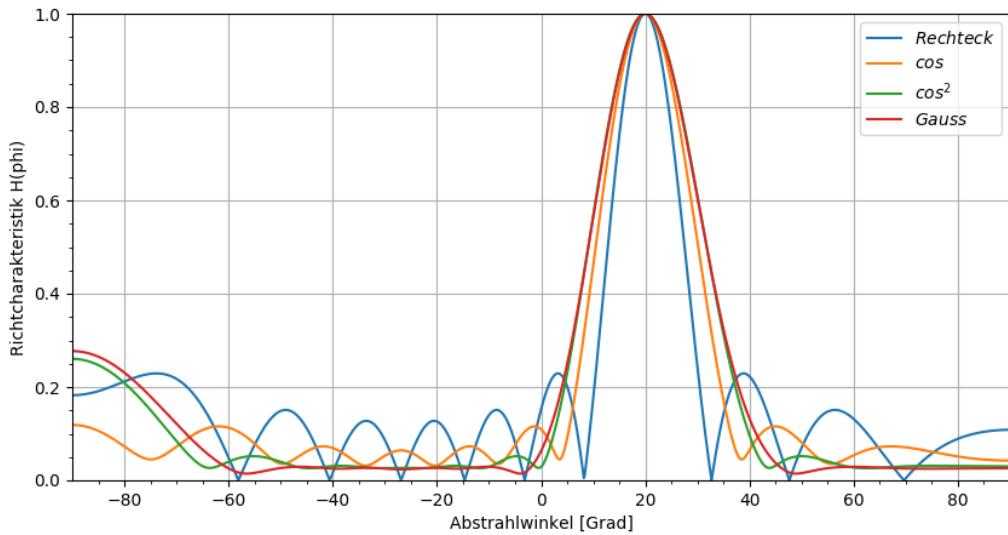


Abbildung 2.12: Richtcharakteristik für  $N = 8$ ,  $d = 5.4\text{mm}$  und einem Sendewinkel von 20 Grad

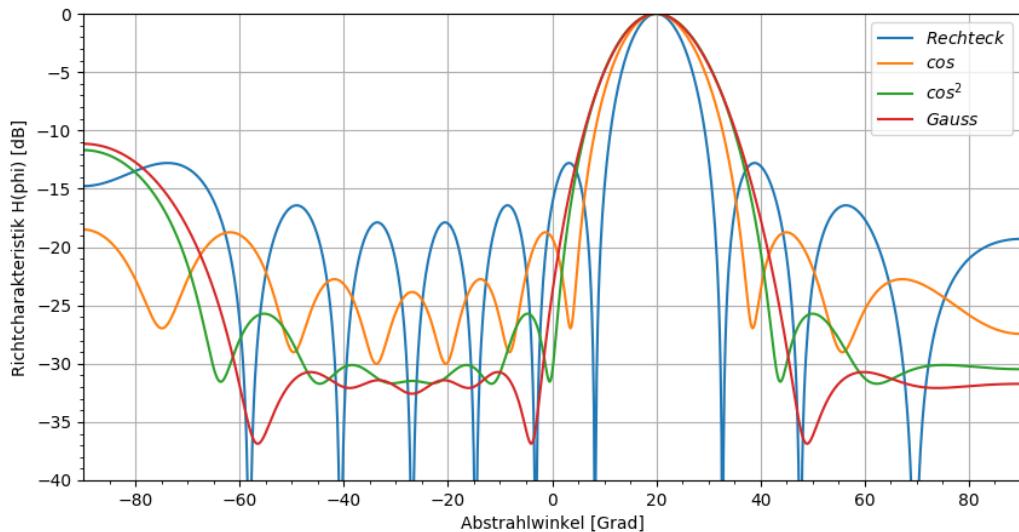


Abbildung 2.13: Richtcharakteristik für  $N = 8$ ,  $d = 5.4\text{mm}$  und einem Sendewinkel von 20 Grad

## 2.2.8 Phased Array-Sensoren

Die Theorie für Arrays aus Schallquellen ist praktisch identisch für Arrays aus Empfängern. Ein schräg auf das Array auftreffendes Echo erzeugt eine Phasenverschiebung zwischen den einzelnen Kanälen. Mittels Signalverarbeitung kann somit eine Richtwirkung des Empfängers erzielt werden. Dies wird im Kapitel 2.4.4 erklärt. Auch eine Nebenkeulenunterdrückung mithilfe der Amplitudenbelegung (siehe 2.2.7) ist möglich.

## 2.3 Sampling

Um analoge Signale mithilfe von Software verarbeiten zu können, müssen sie digitalisiert werden. Dazu wird das Signal abgetastet. Für die Abtastung sind einige Regeln einzuhalten, welche im Folgenden erläutert werden.

### 2.3.1 Abtasttheorem

Das Abtasttheorem nach Nyquist und Shannon sagt aus, dass ein Basisbandsignal mit mindestens dem doppelten seiner höchsten vorkommenden Frequenz abgetastet werden muss, da ansonsten Aliasing auftritt. Für Bandpasssignale gilt dies nicht. Man kann sie unter Einhaltung der Formeln 2.17 und 2.18 auch unterabtasten [9]. Die verwendeten Ultraschalltransceiver benutzen als Resonator Piezokristalle. Diese wirken wie sehr schmalbandige Filter, weshalb das Signal zwischen dem Sensor und dem Analog-Digital-Konverter nicht zusätzlich gefiltert werden muss. Wichtig ist dabei einzig, dass bei der Verstärkung des Signals kein zu starkes Rauschen eingekoppelt wird. Rauschen ist sehr breitbandig und enthält alle Frequenzen, daher hat es einen relativ grossen negativen Einfluss auf unterabgetastete Signale.

Um zu verstehen, wie die Unterabtastung funktioniert, muss man wissen, dass ein abgetastetes Signal ein periodisches Spektrum aufweist. Die periodische Wiederholung kommt daher, dass das Spektrum des Bandpasssignals an jedem Vielfachen der Abtastfrequenz gespiegelt wird. Diese einzelnen Teilspektren dürfen sich nicht überlagern, da sonst Information verloren geht. Um dies sicherzustellen, muss die  $(m - 1)$ -te Wiederholung der Abtastfrequenz  $f_s$  minus  $f_{gu}$  (untere Grenzfrequenz) kleiner sein als  $f_{gu}$  selber, wobei  $m$  eine natürliche Zahl ist. Zusätzlich muss die  $m$ -te Wiederholung der Abtastfrequenz  $f_s$  minus  $f_{go}$  (obere Grenzfrequenz) grösser als  $f_{go}$  sein [9]. Daraus lassen sich zwei Ungleichungen bilden, zusammenfassen und auflösen.

$$\frac{2 \cdot f_{go}}{m} \leq f_s \leq \frac{2 \cdot f_{gu}}{m - 1} , \quad m \in \mathbb{N} \quad (2.17)$$

Aus der Randbedingung  $\frac{2 \cdot f_{go}}{m} \leq \frac{2 \cdot f_{gu}}{m - 1}$  können durch Umstellen der Formel die Unter- und Obergrenze von  $m$  bestimmt werden.

$$1 \leq m \leq \frac{f_{go}}{f_{go} - f_{gu}} , \quad m \in \mathbb{N} \quad (2.18)$$

Aus der Formel 2.18 geht hervor, dass es umso mehr verschiedene Möglichkeiten der Unterabtastung gibt, je höher die obere Grenzfrequenz bei gleichbleibender Bandbreite ( $f_{go} - f_{gu}$ ) liegt, oder je kleiner die Bandbreite des abgetasteten Signals ist.

Um symmetrische Abstände zwischen den Teilspektren zu erreichen, kann die optimale Abtastfrequenz mit folgender Formel berechnet werden [9]:

$$f_s = 2 \cdot \frac{f_{go} + f_{gu}}{2m - 1} , \quad m \in \mathbb{N} \quad (2.19)$$

Die verwendeten Ultraschalltransceiver (siehe Kapitel 3) haben eine Mittenfrequenz von 40kHz und eine Bandbreite von ungefähr 2kHz. Somit ergibt sich nach Gleichung 2.18 für  $m$  folgender Wert:

$$1 \leq m \leq \frac{41\text{kHz}}{41\text{kHz} - 39\text{kHz}} = \frac{41\text{kHz}}{2\text{kHz}} = 20.5 , \quad m \in \mathbb{N}$$

$m$	$\frac{2 \cdot f_{go}}{m}$	$\leq f_{s_{\text{optimal}}} \leq$	$\frac{2 \cdot f_{gu}}{m-1}$
1	82.00000kHz	160.00000kHz	$\infty$
2	41.00000kHz	53.33333kHz	78.00000kHz
3	27.33333kHz	32.00000kHz	39.00000kHz
4	20.50000kHz	22.85714kHz	26.00000kHz
5	16.40000kHz	17.77778kHz	19.50000kHz
6	13.66667kHz	14.54545kHz	15.60000kHz
7	11.71429kHz	12.30769kHz	13.00000kHz
8	10.25000kHz	10.66667kHz	11.14286kHz
9	9.111111kHz	9.411765kHz	9.750000kHz
10	8.200000kHz	8.421053kHz	8.666667kHz
11	7.454545kHz	7.619048kHz	7.800000kHz
12	6.833333kHz	6.956522kHz	7.090909kHz
13	6.307692kHz	6.400000kHz	6.500000kHz
14	5.857143kHz	5.925926kHz	6.000000kHz
15	5.466667kHz	5.517241kHz	5.571429kHz
16	5.125000kHz	5.161290kHz	5.200000kHz
17	4.823529kHz	4.848485kHz	4.875000kHz
18	4.555556kHz	4.571429kHz	4.588235kHz
19	4.315789kHz	4.324324kHz	4.333333kHz
20	4.100000kHz	4.102564kHz	4.105263kHz

**Tabelle 2.2:** Mögliche Abtastfrequenzen für die Unterabtastung

In der Tabelle 2.2 sind in der zweiten und vierten Spalte alle möglichen Unterabtastfrequenzen nach Gleichung 2.17 aufgeführt, in der dritten Spalte sind die optimalen Werte für eine symmetrische Verteilung der Teilspektren nach Gleichung 2.19 aufgeführt.

In der Theorie sind all diese Unterabtastungen zwar möglich, in der Praxis jedoch nicht. Die tiefen Abtastfrequenzen für hohe  $m$  bräuchten extrem steile Filter, die sehr aufwendig zu implementieren sind.

### 2.3.2 Multiplexing

Der verwendete Analog-Digital-Konverter kann immer nur einen Kanal auf einmal konvertieren. Zwischen den einzelnen Konversionen findet dann jeweils ein Kanalwechsel statt. Dieses Verhalten wird als Multiplexing bezeichnet. Dadurch entsteht eine kleine zeitliche Verschiebung zwischen den einzelnen Abtastwerten der verschiedenen Kanäle. Optimalerweise sind diese Verschiebungen verschwindend klein gegenüber dem Abtastintervall  $1/f_s$ . Dann könnten sie nämlich in der nachfolgenden digitalen Signalverarbeitung vernachlässigt werden. Ist dies nicht der Fall, müssen alle Kanäle bis auf den ersten zeitlich um den Wert  $-(x - 1) \cdot \Delta_t$  verschoben werden, wobei  $x$  die Kanalnummer ist und  $\Delta_t$  der zeitliche Abstand zwischen zwei unmittelbar nacheinander abgetasteten Kanälen. Die Realisierung dieser zeitlichen Rückverschiebung ist im Kapitel 2.4.1 genauer beschrieben.

## 2.4 Digitale Signalverarbeitung

In diesem Teil des Fachberichtes werden für das Projekt relevante Algorithmen der digitalen Signalverarbeitung erläutert. Da nicht auf jedes Detail eingegangen werden kann, sei schon hier auf die Literatur [9] und [10] verwiesen.

### 2.4.1 Zeitverschiebung im Frequenzbereich

Eine Zeitverschiebung durch eine Verschiebung des Signals um eine ganze Anzahl Samples ist nicht möglich, falls man ein Signal um sehr kleine Zeiten ( $\Delta\tau < 1/f_s$ ) oder um Zeiten, welche kein ganzes Vielfaches der Samplingperiode sind, verschieben will. Durch eine Transformation des Signals in den Frequenzbereich und das anschliessende Anwenden einer frequenzabhängigen Phasendrehung kann eine beliebige Verzögerung im Zeitbereich realisiert werden. Die dazugehörige Beziehung ist:

$$x(t - \Delta\tau) \circledast X(s) \cdot e^{-s\tau} \quad (2.20)$$

### 2.4.2 Upsampling

Um aus einem unterabgetasteten Signal alle Informationen, insbesondere die Phaseninformation wieder herauszuholen, muss es interpoliert werden. Diesen Vorgang nennt man auch Upsampling.

Beim Signal des schlussendlich verwendeten Ultraschallsensors (siehe Kapitel 3) handelt es sich um ein schmalbandiges Signal bei 40kHz. Eine Abtastung mit  $f_s > 80\text{kHz}$  ist deshalb nicht nötig. Es wird deshalb eine Abtastfrequenz von  $f_s = 32\text{kHz}$  gewählt (siehe Kapitel 2.3.1 und Tabelle 2.2). Durch die Unterabtastung entstehen Kopien des zweiseitigen Spektrums bei  $\pm n \cdot f_s$ . Diese sind in Abbildung 2.14 zu sehen. Das empfangene Ultraschallsignal erscheint nach der Unterabtastung bei  $\pm 8\text{kHz}$ .

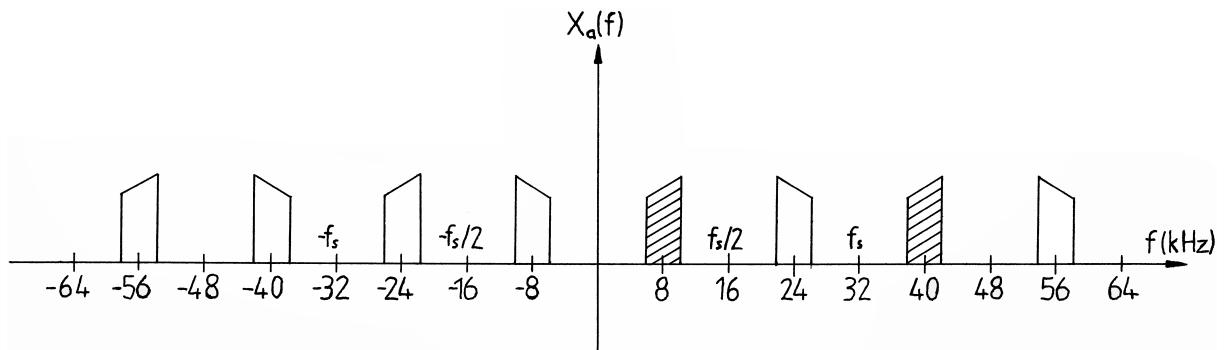


Abbildung 2.14: Signal nach der Unterabtastung mit  $f_s = 32\text{kHz}$

Erhöht man die Abtastrate nach dem Sampling um den Faktor  $R = 4$  auf eine Frequenz von  $f_s = 128\text{kHz}$ , kann das ursprüngliche Signal bei 40kHz wiederhergestellt werden. Damit ist die Phaseninformation des Signals wiederherstellbar, sofern bei der Unterabtastung kein Aliasing entsteht.

Das Upsampling im Zeitbereich ist im Prinzip ein Einfügen von Nullen zwischen den vorhandenen Sampling-Werten. Damit wird die Abtastrate für  $R$  Nullen zwischen den Werten um den Faktor  $R + 1$  erhöht.

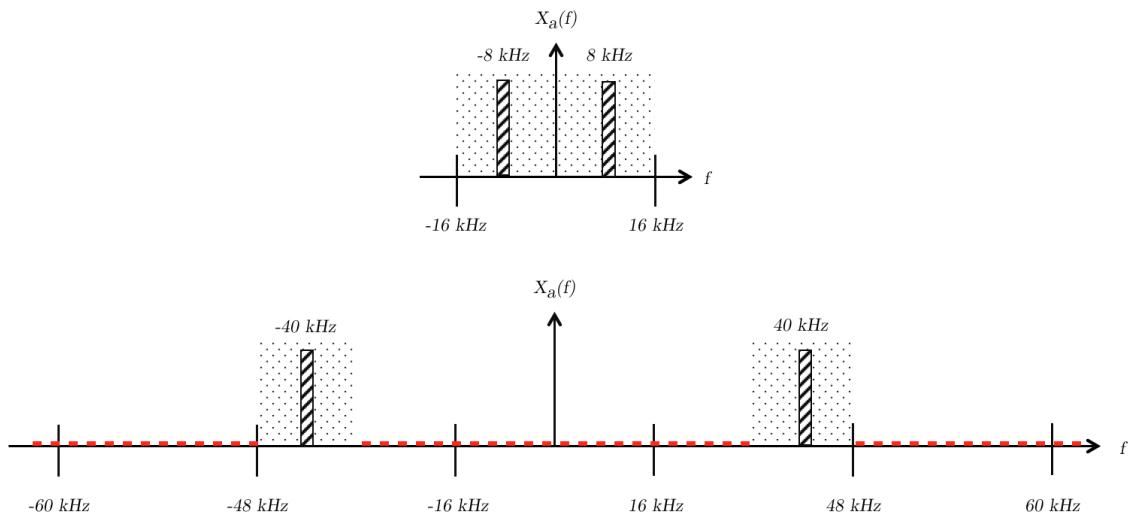
Nach dem Upsampling muss mittels geeignetem Filter diejenige Kopie des Signals herausgefiltert werden, die im ursprünglichen Frequenzbereich liegt (das Signal bei 40kHz). Wegen der schmalbandigen Signale der Ultraschallsensoren kann dafür ein relativ breitbandiges IIR-Bandpassfilter

verwendet werden, ohne dass Aliasing entsteht. Beim Filterdesign ist darauf zu achten, dass das Filter nicht schmalbandiger ist als das Signal, sonst würde dieses verfälscht. Jedoch müssen die Kopien bei  $40\text{kHz} \pm 16\text{kHz}$  trotzdem genügend gedämpft sein, so dass sie nicht in Erscheinung treten.

Da die Verarbeitung nicht in Echtzeit geschehen muss, kann mithilfe der Funktion `filtfilt()` das Signal zeitlich in beide Richtungen gefiltert und so ein akausales Filter ohne Phasenverzerrungen erzielt werden.

### 2.4.3 Upsampling im Frequenzbereich

Wie in Abbildung 2.14 zu sehen ist, erscheint das Ultraschallsignal vor dem Upsampling als Signal bei  $\pm 8\text{kHz}$ . Das ursprüngliche Signal hätte Komponenten bei  $\pm 40\text{kHz}$ . Wird vom unterabgetasteten Signal die FFT berechnet, so kann mit diesen Werten das Spektrum des Signals mit der erhöhten Abtastfrequenz zusammengesetzt werden. Dieser Vorgang ist in Abbildung 2.14 zu sehen. Das Spektrum des ursprünglichen Signals (siehe Abbildung 2.15 oben) wird durch Einfügen von Nullen an den richtigen Stellen (rot) zu einem um den Upsampling Faktor  $R = 4$  grösseren Frequenzvektor erweitert. Ein Ausschnitt des Frequenzvektors ist in Abbildung 2.15 unten zu sehen. Wird dieses Signal zurück in den Zeitbereich transformiert, so entsteht dadurch das Zeitsignal mit erhöhter Abtastfrequenz.



**Abbildung 2.15:** Upsampling des unterabgetasteten Ultraschallsignals im Frequenzbereich

Der Vorteil dieser Variante ist, dass auf das Filtern des Signals mit erhöhter Abtastfrequenz im Zeitbereich verzichtet und dadurch Rechenleistung eingespart werden kann.

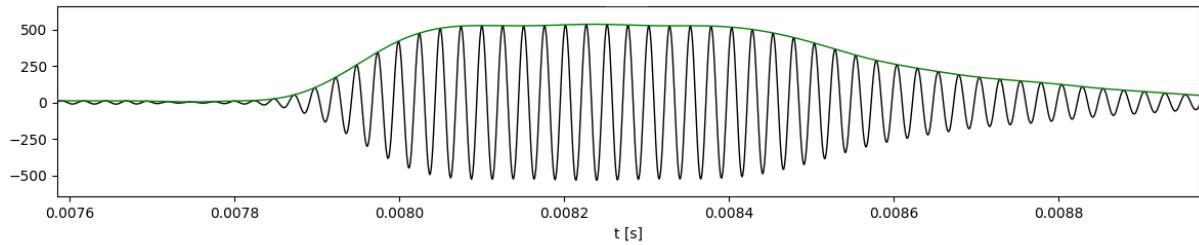
### 2.4.4 Beamforming

Werden die digitalisierten Signale der Ultraschallsensoren addiert, entsteht dadurch eine Richtwirkung des Empfängers mit Haupt- und Nebenkeulen: Durch die Addition der einzelnen Signale der Kanäle addieren sich diejenigen Anteile am stärksten, die in Phase sind. Dies entspricht Schall, der senkrecht auf das Array auftrifft. Um Schall aus einer bestimmten Richtung zu empfangen, werden die einzelnen Kanäle zeitverzögert, bevor sie aufaddiert werden. So werden die Anteile am grössten, die nach dieser Zeitverzögerung in Phase sind. Der dazugehörige Empfangswinkel lässt sich mit Formel (2.20) berechnen.

### 2.4.5 Enveloppe berechnen

Im negativen Anteil des komplex konjugierten Spektrums eines reellen Zeitsignals steckt keine weitere Information über das Signal, die nicht im positiven Spektrum vorhanden wäre. Theoretisch kann ein reelles Zeitsignal durch einen komplexen Anteil so erweitert werden, dass dessen Spektrum für negative Frequenzen verschwindet. Ein solches Signal bezeichnet man als analytisches Signal. Um das analytische Signal zu erzeugen, wird das reelle Zeitsignal  $x(t)$  um den imaginären Anteil  $j \cdot y(t)$  erweitert. Die Transformation, die aus  $x(t)$  das Signal  $y(t)$  erzeugt, wird als Hilbert-Transformation bezeichnet. Für eine detaillierte Beschreibung der Hilbert-Transformation wird auf [10] verwiesen.

Dieses komplexe Zeitsignal weist für jede Zeit einen Betrag und eine Phase auf. Der Verlauf der Amplitude stellt direkt die Enveloppe des Signals dar. Aus der Ableitung des Verlaufs der Phase kann der Verlauf der Momentanfrequenz berechnet werden. Der Verlauf des Betrags eines analytischen Signals von einem Ultraschallsignal ist in Abbildung 2.15 dargestellt.



**Abbildung 2.16:** Enveloppe eines Ultraschall-Echos.

### 2.4.6 Downsampling

Um die Datenmenge zu reduzieren, kann die Abtastfrequenz des Signals um den Faktor  $R$  verkleinert werden. Dafür wird nur jeder  $R$ -te Wert eines Signals berücksichtigt. Die Gesamtdauer des Signals bleibt unverändert. Die maximal mögliche Frequenz hat sich damit um den Faktor  $R$  verkleinert. Damit kein Aliasing entsteht, müssen im Normalfall störende Frequenzen mit einem Tiefpassfilter unterdrückt werden.

Das zu dezimierende Signal ist die Enveloppe des Ultraschallsignals. Diese ist in Abbildung 2.16 zu sehen. Weil es sich dabei um ein sehr niederfrequentes Signal handelt, ist kein Tiefpassfilter nötig.

### 3 Hardware

Im nachfolgenden Teil des Fachberichtes wird die entwickelte Hardware des Phased Arrays näher beschrieben. Das Ziel ist es, eine möglichst modulare und skalierbare Schaltung zu entwickeln. Als Hardwareumgebung für den Mikrocontroller dient ein schon bestehendes Evaluationsboard. Das dazugehörige Layout ist frei verfügbar und vom Preis her erschwinglich. Die in diesem Projekt entwickelte Hardware ist auf der Rückseite mit Pins ausgestattet, so dass sie als Modul auf das Evaluationsboard gesteckt werden kann. Auf der Vorderseite befindet sich ein Ultraschall Phased Array und die dazugehörige Analogschaltung. Die elektronischen Komponenten sind oberflächenmontiert.

#### 3.1 Überblick

Die Abbildung 3.1 zeigt das Gesamtsystem. Das Arduino Board stellt das Evaluationsboard dar, das Arduino Shield die in diesem Projekt entwickelte Hardware. Es ist zu sehen, dass das PWM-Modul des Mikrocontrollers über eine analoge Senderschaltung (Levelshifter) das Ultraschall Transceiver Array ansteuert und dieses wiederum die empfangenen Signale über die analoge Empfängerschaltung (Operationsverstärker und Analogschalter) auf den Analog-Digital-Konverter des Mikrocontrollers gibt. Die digitalisierten Daten werden von dort aus per DMA (direct memory access) über eine USB-Schnittstelle an das Desktop-Betriebssystem weitergeleitet.

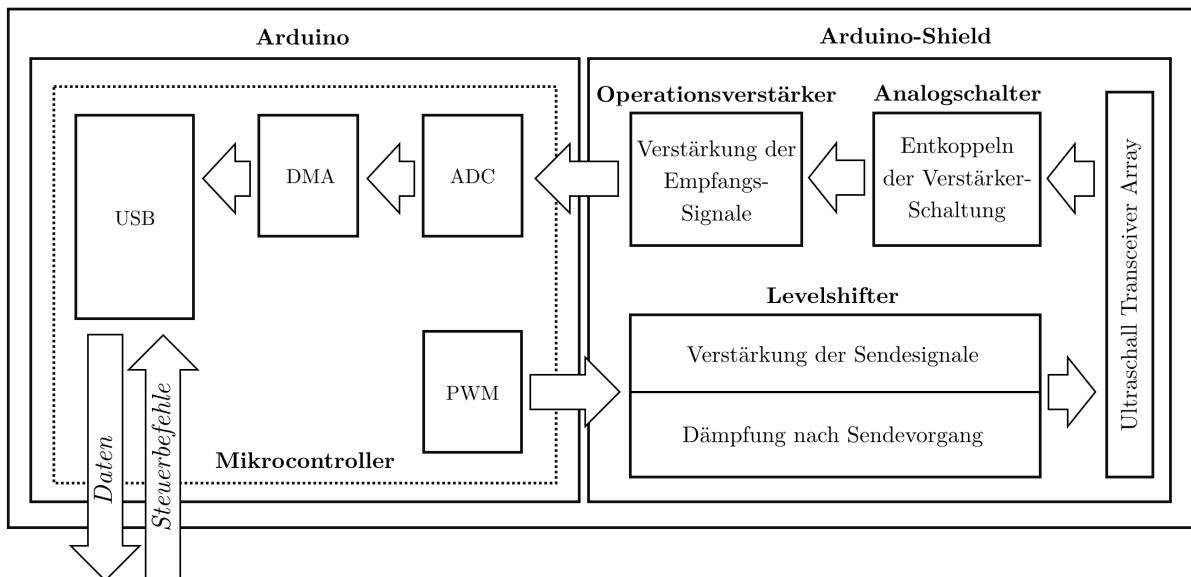


Abbildung 3.1: Blockschaltbild des Gesamtsystems

In Abbildung 3.2 ist die Hardware des Ultraschall Phased Arrays zu sehen, welche als Modul auf den Arduino Due gesteckt ist. Oben in der Mitte sind die acht Ultraschalltransceiver (1) zu sehen. Diese können sowohl zum Senden als auch zum Empfangen benutzt werden. Neben den Ultraschalltransceivern ist auf jeder Seite je ein Levelshifter (2) angebracht. Diese sind als Verstärker für das Senden zuständig. Außerdem werden sie zum Dämpfen der Ultraschalltransceiver nach dem Sendevorgang verwendet. Direkt unterhalb der Ultraschalltransceiver befinden sich zwei Analogschalter (3), welche die empfindliche Empfängerschaltung während des Sende-vorgangs von den Ultraschalltransceivern und somit von der Senderschaltung trennen. Unterhalb der Analogschalter befinden sich die acht Verstärkerschaltungen (4) des Empfängerseils. Direkt unter dem linken Levelshifter ist eine Spannungsfolgerschaltung (5) zu sehen, mit welcher die

Referenzspannung für die Empfängerschaltung erzeugt wird. Unten links sind die beiden Stütz-kondensatoren (6) für die Spannungsversorgungen von 3.3V und 5.0V zu sehen.

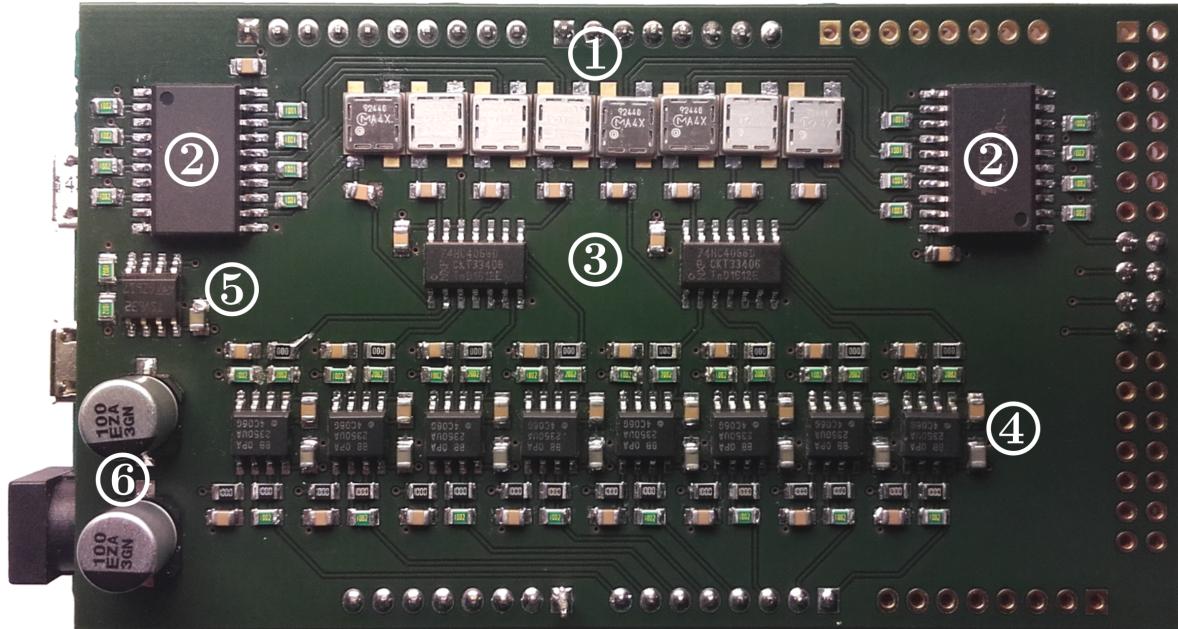


Abbildung 3.2: Fertig bestückter Print

### 3.2 Ultraschall Phased Array

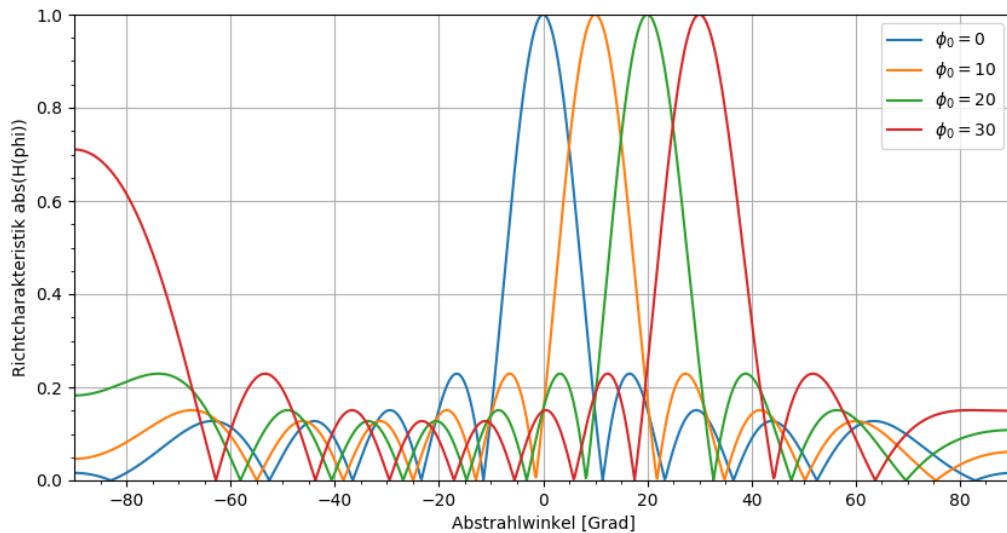
Das Phased Array besteht im Gegensatz zu demjenigen in Projekt 5 aus nur einer Reihe von 1x8 Ultraschalltransceivern, die gleichzeitig als Sender und Empfänger dienen. Da die Ultraschall-transceiver die teuersten Komponenten des Phased Arrays sind, können dadurch die Kosten deutlich gesenkt werden (siehe Anhang B). Mit je einer Reihe Sender- und einer Reihe Empfängermodulen besteht bei der Vorgängerschaltung der Vorteil, dass die beiden Teilschaltungen elektronisch entkoppelt sind, und somit die empfindliche Verstärkerschaltung des Empfängers vor den hohen Spannungen der Sendeschaltung geschützt ist. Jedoch ergeben sich starke akustische Kopplungen zwischen Sende- und Empfängermodulen. Dies ist genauso unerwünscht wie elektronische Koppelung, da dadurch der Ausschwingvorgang sehr lange dauert und die minimal messbare Distanz grösser wird.

Ursprünglich geplant war nur eine Reihe von fünf Ultraschalltransceivern. Aufgrund der Berechnungen im Kapitel 2.2.5, welche in der Abbildung 2.8 dargestellt sind, ist ersichtlich, dass bereits mit acht Schallquellen eine deutlich schmalere Hauptkeule in der Richtcharakteristik erzeugt werden kann.

Das Phased Array soll in Räumen eingesetzt werden. Daher werden Ultraschalltransceiver benötigt, die in Luft eine möglichst geringe Dämpfung aufweisen. Handelsübliche Ultraschallsender und Empfänger für diesen Anwendungsbereich arbeiten bei Frequenzen von ca. 30kHz – 50kHz. Diese liegen einerseits deutlich ausserhalb des hörbaren Bereichs, andererseits sind sie noch genügend tief, um eine breite Abstrahlcharakteristik zu erzeugen. Bei höheren Frequenzen nimmt die Richtwirkung der einzelnen Module aufgrund der grösseren Dimensionen im Verhältnis zur Wellenlänge tendenziell zu, was für ein Phased Array unerwünscht ist.

Wie aus dem Kapitel 2.2.5, Abbildung 2.6 hervorgeht, entstehen starke Nebenkeulen in der Richtcharakteristik, sobald der Abstand der Ultraschalltransceiver grösser als eine Wellenlänge

wird. Die meisten im Handel erhältlichen Ultraschalltransceiver kommen daher nicht infrage, da ihre Abmessungen zu gross sind. Für das Phased Array werden die kleinsten günstig erhältlichen Ultraschalltransceiver des Typs "Murata MA40H1S-R" verwendet, die im Abstand von 5.4mm montiert werden können und eine Mittenfrequenz von 40kHz aufweisen. Dies entspricht einer Wellenlänge von 8.5mm, womit der Abstand zwischen den Modulen  $d = 0.635\lambda$  beträgt. In Abbildung 3.3 ist die berechnete, theoretisch erreichbare Richtcharakteristik ohne Amplitudenbelegung für vier verschiedene Sendewinkel dargestellt.



**Abbildung 3.3:** Richtcharakteristik mit  $N = 8$  Elementen, im Abstand von  $d = 0.635\lambda$

### 3.3 Analogschaltung zum Phased Array

Für die Ansteuerung des Ultraschall Phased Arrays wird eine Analogschaltung benötigt, da einerseits die Sendesignale verstärkt werden müssen und andererseits die empfangenen Signale zu schwach sind, um sie ohne Verstärkung zu digitalisieren. Es werden also zwei unterschiedliche Verstärkerschaltungen benötigt, welche nachfolgend als Senderschaltung und Empfängerschaltung bezeichnet werden. Zur Empfängerschaltung gehört ausserdem noch die Teilschaltung, welche die Empfangsverstärker zu deren Schutz während des Sendens von den Ultraschalltransceivern trennt. Zur Sendeschaltung gehört zusätzlich eine Schaltung, mit der die Ultraschalltransceiver gedämpft werden können, um damit die Ausschwingzeit zu verkürzen. Dadurch reduziert sich auch die minimal messbare Distanz. Nachfolgend werden diese Schaltungen näher erläutert.

#### 3.3.1 Empfängerschaltung

Aufgabe der Empfängerschaltung ist es, die von den Ultraschallsensoren empfangenen Signale zu verstärken und an den Analog-Digital-Konverter des Mikrocontrollers weiterzuleiten. Ausserdem muss die Verstärkerschaltung aufgrund ihrer Empfindlichkeit während des Sendevorgangs von den Ultraschalltransceivern getrennt werden. Dafür werden zwei Analogschalter mit je vier Kanälen verwendet. Abbildung 3.4 zeigt die Empfängerschaltung und einen Teil der Senderschaltung für einen Kanal. Die anderen sieben Kanäle sind identisch aufgebaut. Der Analogschalter (U3A) kann nur Spannungen zwischen 0.0V und 5.0V schalten, da er keine negative Versorgungsspannung aufweist. Da nun aber der Ultraschalltransceiver (X1) als Bezugspotential 0.0V hat und während des Empfangens auch in den negativen Bereich schwingt, ist zwischen Ultraschalltransceiver und Analogschalter ein Kondensator (C15) zum Entkoppeln eingebaut. Die andere Seite des Analogschalters hängt über die Operationsverstärkerschaltung auf dem Potential der Referenzspannung V\_REF von 1.65V. Über das Signal REC\_EN (receiver enable) wird der Analogschalter direkt vom Mikrocontroller angesteuert.

Wie in der Abbildung 3.4 ebenfalls zu sehen ist, werden für die Verstärkung zwei kaskadierte Operationsverstärker verwendet. Wichtig bei der Auswahl der Operationsverstärker ist, dass diese bis zu einer Frequenz von 50kHz eine genügend hohe Verstärkung im Bereich von 100 bis 200 aufweisen. Da die Piezoelemente der Ultraschalltransceiver selber schon sehr schmalbandige Filter darstellen, ist eine weitere Bandpassfilterung vor der Unterabtastung unnötig. Es ist aber darauf zu achten, dass die Verstärkerschaltung möglichst rauscharm ist, weil sich bei einer Unterabtastung der Signale das Rauschen stark auf die Signalqualität auswirkt.

Die Referenzspannung V\_REF von 1.65V wird für alle acht Kanäle über eine Spannungsfolgerschaltung erzeugt und über ein RC-Tiefpassfilter (R18/C23 und R42/C39) an den nichtinvertierenden Eingang der Operationsverstärker gelegt. Das Tiefpassfilter dient dazu, allfällige hochfrequente Störungen direkt vor dem Operationsverstärker zu dämpfen. Die Grenzfrequenz für den ersten Operationsverstärker wurde um den Faktor 100 kleiner (bei ca. 150Hz) gewählt als für den zweiten, da bei jenem wesentlich kleinere und störanfällige Signale anliegen.

Die kompletten Empfänger- und Senderschaltungen kommen ohne negative Spannung aus. Die Operationsverstärker werden zwischen 0.0V und 3.3V betrieben, weshalb ein Single-Supply Rail-to-Rail Operationsverstärker eingesetzt wird. Der Operationsverstärker OPA2350UA weist ein genügend grosses Gain-bandwidth Product und eine ausreichende Slew-Rate auf, um bei mindestens 50kHz eine Verstärkung von über 200 zu erzielen, und erfüllt damit die nötigen Anforderungen.

Da eine Verstärkung um den Faktor 200 insgesamt zu klein ist, sind pro Ultraschallempfänger jeweils zwei Operationsverstärker kaskadiert. Die Verstärkung des ersten Operationsverstärkers

(U5A) hängt neben dem Rückkopplungswiderstand ( $R_{33} = 20\text{k}\Omega$ ) von der Impedanz des Piezoelements ( $Z_{X1} \approx 1000\Omega$ ), vom Schaltwiderstand des Analogschalters ( $R_{U3A} \approx 85\Omega$ ) und vom Vorwiderstand ( $R_{17} = 0\Omega$ ) ab. Die Verstärkung beträgt somit ungefähr 18.4, kann aber leicht variieren, weil weder der Schaltwiderstand des Analogschalters noch die Impedanz des Piezoelements ganz genau spezifiziert sind. Die Verstärkung des zweiten Operationsverstärkers beträgt  $R_{57}/R_{41} = 10\text{k}\Omega/100\Omega = 100$ . Insgesamt beträgt die Verstärkung also etwa 1840.

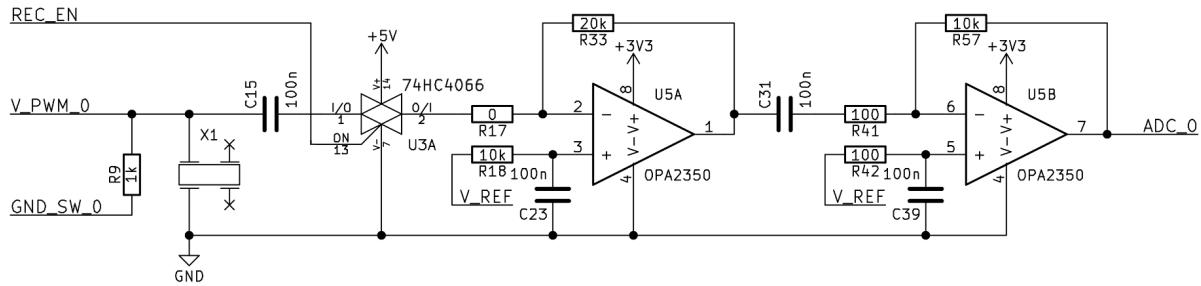


Abbildung 3.4: Schema der Operationsverstärkerschaltung

### 3.3.2 Senderschaltung

Die Senderschaltung hat die Aufgabe, den Piezokristall des Ultraschalltransceivers mit einem Rechtecksignal von 0.0 – 5.0V und einer Frequenz von 40kHz in Schwingung zu versetzen. Dadurch wird ein akustischer Ton derselben Frequenz erzeugt. Um dies zu realisieren, werden zwei Levelshifter des Typs 74ACT244 verwendet, welche die Ausgangsspannung des Mikrocontrollers von 3.3V auf 5.0V erhöhen. Die Levelshifter werden dazu vom Mikrocontroller mit einem pulsweitenmodulierten Signal von 40kHz angesteuert. Da während des Empfangsvorgangs die Levelshifter von den Ultraschalltransceivern getrennt werden müssen, um die Messung der empfangenen Signale nicht zu beeinflussen, müssen sie unmittelbar nach dem Senden auf "high impedance" geschaltet werden. Aus diesem Grund werden Tristate-Levelshifter verwendet. Auch diese Steuerung erfolgt direkt durch den Mikrocontroller. Von jedem Levelshifter werden nur je vier von acht verfügbaren Kanälen für die Verstärkung des Sendesignals verwendet. Die anderen vier Kanäle werden zum Dämpfen der Ultraschalltransceiver direkt nach dem Senden benutzt. Die Ultraschalltransceiver haben bei 40kHz eine Impedanz von ungefähr  $1\text{k}\Omega$ . Daher wird durch den Levelshifter pro Kanal ein Widerstand von  $1\text{k}\Omega$  parallel zum Ultraschalltransceiver gegen Ground geschaltet, um mittels Leistungsanpassung möglichst viel Energie aus dem Schwingkreis im Widerstand zu "verheizen". Die Funktionsweise dieser Schaltung wurde vor dem Redesign der Hardware mit einer Testschaltung an einem einzelnen Kanal mit Erfolg getestet. Abbildung 3.4 im Kapitel 3.3.1 zeigt den Kanal 0 der Sende- und Empfangsschaltung. Links im Bild sind die beiden vom Levelshifter kommenden tristate Signale "V\_PWM\_0" und "GND\_SW\_0" (ground switch) zu sehen. Das erste ist das PWM-Signal zum Anregen des Piezos, über das zweite wird der Widerstand zugeschaltet.

## 3.4 ADC und Mikrocontroller

Für die Ansteuerung eines  $1 \times N$  Arrays werden auf dem Mikrocontroller  $N$  ADC-Kanäle und  $N$  PWM-Kanäle benötigt. Für das in diesem Projekt entwickelte Phased Array wird  $N = 8$  gewählt. Gemäß den Simulationen (siehe Abbildungen 2.9, 2.10 und 2.11 im Kapitel 2.2.6) und den Berechnungen (siehe Abbildung 3.3 im Kapitel 3.2) kann damit eine genügend hohe Richtwirkung erzeugt werden. Auch die anfallende Datenmenge des Analog-Digital-Konverters kann problemlos per USB gesendet und auch auf einem leistungsschwachen Desktop PC verarbeitet werden.

Der ADC des Mikrocontrollers muss eine genügend hohe Abtastrate aufweisen, um die Ultraschallsignale zu empfangen. Bei einer Ultraschallfrequenz von maximal 50kHz und acht Kanälen (Multiplexing) ergibt sich laut dem Abtasttheorem nach Nyquist-Shannon eine Abtastfrequenz von minimal 0.8MHz. Durch Unterabtastung lässt sich aufgrund der schmalbandigen Signale diese Anforderung aber umgehen (siehe Kapitel 2.3.1). Um die anfallenden Datenmengen vom Mikrocontroller auf das weiterverarbeitende System (Desktop-Betriebssystem) zu übertragen, ist eine USB 2.0 Highspeed Schnittstelle mit Direct Memory Access (DMA) von Vorteil.

Ein verbreiteter Mikrocontroller, der all diese Anforderungen erfüllt, ist der SAM3X8E von Atmel. Er hat 16 ADC-Kanäle, 16 PWM-Kanäle, von denen aber nicht alle ohne weiteres verfügbar sind, sowie eine USB-Schnittstelle. Die Samplingfrequenz des ADC beträgt 1MHz (Multiplexing über alle 16 Kanäle). Dies ist ausreichend, wenn mit Unterabtastung gearbeitet wird. Aus schlaggebend bei der Unterabtastung ist dabei auch die Sample-and-Hold-Zeit, welche bei einer Abtastrate von 1MHz ausreicht. Ebenfalls weist der SAM3X8E einen DMA-Controller der einerseits mit dem ADC und andererseits mit der USB-Schnittstelle zusammen verwendet werden kann.

Für dieses Projekt wird ein Arduino DUE Board eingesetzt, da es den Mikrocontroller SAM3X8E enthält, welcher alle gestellten Anforderungen erfüllt. Das Arduino DUE Board hat desweiteren den Vorteil, dass es einfach erhältlich ist und das Projekt daher relativ einfach nachgebaut werden kann. Es muss dadurch keine Zeit mit dem Aufbau eines "eigenen" Mikrocontrollerboards aufgewendet werden.

### 3.5 PCB Design

Nachfolgend werden die wichtigsten Kriterien aufgezählt, nach denen die Leiterplatte für dieses Projekt entwickelt ist. Um Störungen möglichst gut vorzubeugen, wird ein vierlagiger Print entwickelt, was zwar teurer ist als ein zweilagiger, aber einige Vorteile bezüglich der elektromagnetischen Verträglichkeit hat. Die Bilder der einzelnen Printlayer und das komplette Schema befinden sich im Anhang A.

Der unterste Layer ist ein Groundlayer und weist keine Signalleitungen auf. Vom Arduino DUE stammende hochfrequente induktive Störungen werden dadurch bestmöglich abgeblockt.

Der nächste Layer enthält die beiden Spannungsversorgungen von 3.3V und 5.0V. Daneben verlaufen noch die digitalen Signalleitungen zum Einschalten der Analogschalter und der Levelshifter.

Der dritte Layer liegt auf dem Potential der Referenzspannung für die Operationsverstärkerschaltung des Empfängerteils. Daneben werden aussen am Rand die Signalleitungen zur Ansteuerung der Levelshifter geführt. Es ist darauf zu achten, dass keine Signalleitungen zwischen der Spannungsfolgerschaltung und den einzelnen Operationsverstärkern liegen. Die grossflächige Ausführung des Layers wirkt wie schon der Groundlayer abschirmend gegen hochfrequente induktive Kopplungen von unten.

Der oberste Layer enthält alle analogen Signalleitungen, welche sehr empfindlich sind. Alle nicht benutzten Flächen sind mit Ground ausgegossen. Bei der Führung der Signalleitungen ist darauf zu achten, dass zwischen den Leiterbahnen immer ein genügend grosser Abstand vorhanden ist, um Übersprechen zu verhindern. Für alle analogen Signale wird jeweils der kürzestmögliche Weg gewählt. Digitale Signale werden nur aussen am Rand des Layers, und niemals zu nahe bei analogen Signalen geführt.

## 4 Software

Im folgenden Teil des Fachberichtes wird der Aufbau der Software beschrieben. Es folgt zu Beginn eine Gesamtübersicht und ein Kommentar zur Auswahl der verwendeten Programmiersprachen und Schnittstellen. Danach werden die einzelnen Teilsysteme beschrieben.

### Übersicht

Die Software ermöglicht die Bedienung des Ultraschall Phased Array Systems über eine Webapplikation. Anhand der Benutzereingaben werden auf dem Arduino automatisch Messungen durchgeführt und die Resultate im User-Interface dargestellt.

Auf dem Arduino DUE wird eine in C programmierte Statemachine ausgeführt, diese übernimmt die Steuerung der Hardware. Abseits vom Arduino DUE ist die Software als Webapplikation realisiert, dabei wird grob zwischen host- und clientseitiger Software unterschieden. Der hostseitige Code ist in der Programmiersprache Python realisiert und wird von einem python-fähigen Desktop-Betriebssystem ausgeführt. Der clientseitige Code wird, falls gewünscht, auf einem anderen Computer/Betriebssystem ausgeführt. Das User-Interface wird über einen Webbrowser aufgerufen und ist in den Programmiersprachen HTML, CSS und Javascript umgesetzt. Näheres zur Installation und Kompatibilität ist in dem Kapitel 4.5 zu finden.

Die Dokumentation der Software wird aufgeteilt in drei verschiedene Teilsysteme. Abbildung 4.1 enthält eine Übersicht über die gesamte Software.

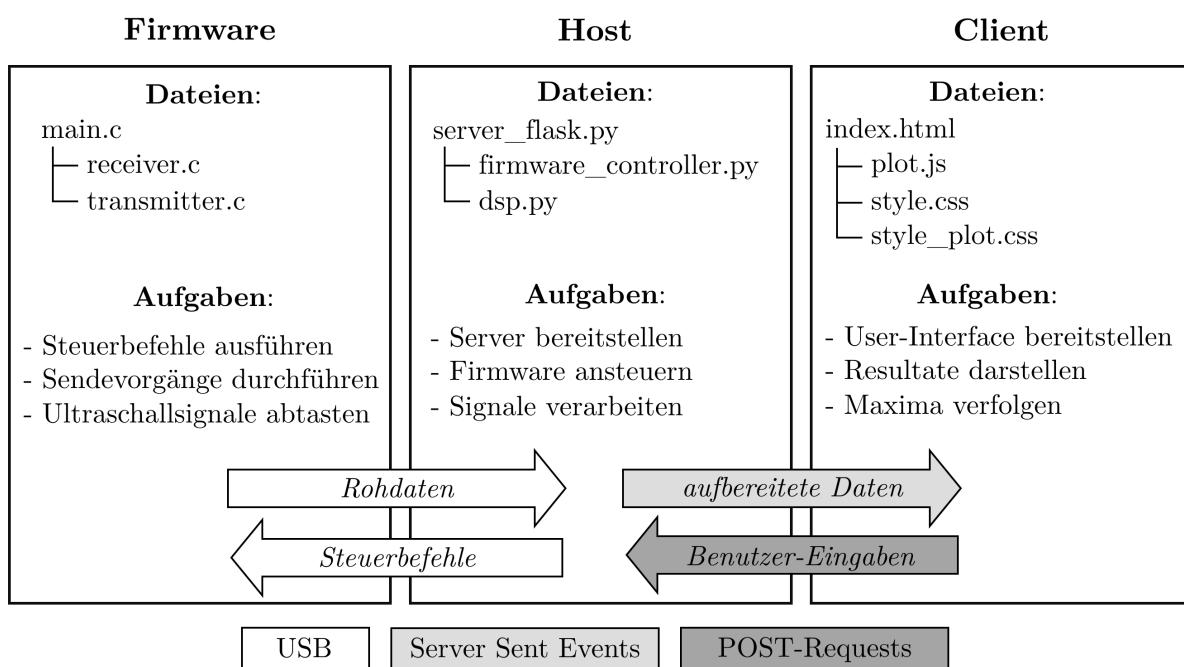


Abbildung 4.1: Gesamtübersicht Software

Über die USB-Schnittstelle werden die Rohdaten der acht Ultraschallsignale an den Host übergeben. Dieser berechnet daraus (als Hintergrund-Thread des Webservers) das darzustellende Signal. Server Sent Events sind eine HTML5-Technologie, welche es dem Client ermöglicht, asynchron Updates eines Servers zu erhalten. Somit kann der Webbrowser (Client) das darzustellende Signal zum jeweiligen Zeitpunkt seines Auftretens entgegennehmen und im User-Interface darstellen. Die Darstellung erfolgt als aktualisiertes Bild der Umgebung und als Linienplot.

Eingaben im User-Interface durch den Benutzer werden vom Server per POST-Request (auslesen eines Eingabeformulars auf der Webseite) vom Server übernommen. Diese Eingaben werden

vom Host (als Hintergrund-Thread des Webservers) in Steuerbefehle übersetzt, welche die genauen Spezifikationen des nächsten Messvorgangs enthalten. Diese Steuerbefehle werden per USB an den Mikrocontroller übergeben. Dieser führt einen neuen Messvorgang aus und sendet danach die neuen Rohdaten der acht Ultraschallsignale zurück.

#### 4.1 Auswahl der Programmiersprachen und Schnittstellen

Die Anforderungen an die Firmware auf dem Mikrocontroller erlauben dank der geringen Komplexität eine prozedurale Umsetzung in C. Ein Objektorientierter Ansatz in C++ ist nicht nötig.

Für die Umsetzung der Software auf dem Desktop-Betriebssystem (Signalverarbeitung und GUI) kommen zwei verschiedene Ansätze in Frage:

- Standalone
- Server / Client

Wird die gesamte Software als Standalone-Anwendung geschrieben, hat dies eine geringere Komplexität zur Folge. Schnittstellen innerhalb der Software sind einfacher zu realisieren, im Vergleich zu den systemübergreifenden Schnittstellen einer Server-Client Realisierung.

Der Ansatz einer Webapplikation hat die Vorteile, dass die Rechenleistung zwischen Host und Client aufgeteilt werden kann. Zudem wird das User-Interface plattformunabhängig. So kann der Server z.B. auf einem Raspberry Pi betrieben werden und der Client über das lokale Netzwerk mittels Laptop auf den Server zugreifen. Damit wird das ganze Phased Array System mobil. Mit einem geeigneten Webframework ist der Mehraufwand im Vergleich zu einer Lösung, welche nur auf einem System umgesetzt ist klein.

Folgende drei Programmiersprachen kommen für die Implementation der serverseitigen Software in Frage und werden nachfolgend gegeneinander abgewogen.

- C / C++
- Java
- Python

Eine sorgfältige Realisierung in C oder C++ ist bezüglich Performance kaum zu übertreffen. Im Vergleich zu den anderen Programmiersprachen ist C / C++ jedoch auch mit der aufwändigsten Realisierung verbunden. Die relativ lange Laufzeit des Schalls hat eine Wartezeit zwischen den Messungen zur Folge. Dadurch entsteht ein Zeitfenster, welches so gross ist, das eine derart optimierte Signalverarbeitung nicht nötig ist.

Eine Umsetzung in Java ist im Vergleich mit Python mit mehr Aufwand verbunden, da die Software für das Projekt 5 in Python realisiert wurde. Die zu klärende Frage ist, ob durch die Umsetzung der Signalverarbeitung in Python zusätzliche Wartezeiten zwischen den Messungen entstehen, welche mit Java verhindert werden könnten. Wartezeiten entstehen im Projekt 5 tatsächlich, sowohl durch die Darstellung, als auch durch die Signalverarbeitung. Nach ersten Versuchen zur Beschleunigung der Algorithmen kann die Signalverarbeitung mittels Upsampling im Frequenzbereich (siehe Kapitel 2.4.3) vervielfacht werden. Damit wird die Verarbeitung in Python schnell genug. Mithilfe der Berechnung der Enveloppe und anschliessendem Downsampling kann auch die Darstellung beschleunigt werden. Ein Server-Client Ansatz ermöglicht zusätzlich eine Auslagerung der zur Darstellung benötigten Rechenleistung auf ein anderes System.

Die bereits im Projekt 5 zur Signalverarbeitung in Python verwendeten Bibliotheken `numpy` und `scipy` sind sowohl effizient, wie auch praktisch in der Handhabung. Codefragmente zur USB-Kommunikation und zur Signalverarbeitung (Zeitverschiebung im Frequenzbereich) können vom P5 mit wenigen Änderungen übernommen werden. Zusätzlich ist Python beliebt bei

der serverseitigen Programmierung, weshalb diverse Webframeworks zur Verfügung stehen. Als Python-Webframework bietet sich Flask als gut dokumentierte Alternative zum verbreiteten Django an. Flask ist wesentlich einfacher in der Handhabung und ein einfaches Webapp kann ohne grossen Aufwand aufgesetzt werden. Server Sent Events ermöglichen es, dass der Client die darzustellenden Daten asynchron erhält und nicht ständig prüfen muss, ob neue Daten vorhanden sind.

## 4.2 Firmware auf dem Mikrocontroller

Die Firmware auf dem Mikrocontroller steuert das Ultraschall Phased Array auf der entwickelten Hardware. Der Sourcecode kann in drei Module aufgeteilt werden. Diese werden nachfolgend kurz beschrieben. Neben dem Modul "main.c", welches das Hauptprogramm beinhaltet, gibt es die Module "receiver.c" und "transmitter.c", wobei sich diese Namen auf das Aussenden und Empfangen der Ultraschallsignale beziehen.

### 4.2.1 Initialisierung und Hauptprogrammfluss – main.c

Abbildung 4.2 zeigt die Initialisierung und den Hauptprogrammfluss der Firmware auf dem Mikrocontroller. Die Initialisierung ist als Ablaufdiagramm und die Statemachine als Zustandsdiagramm dargestellt.

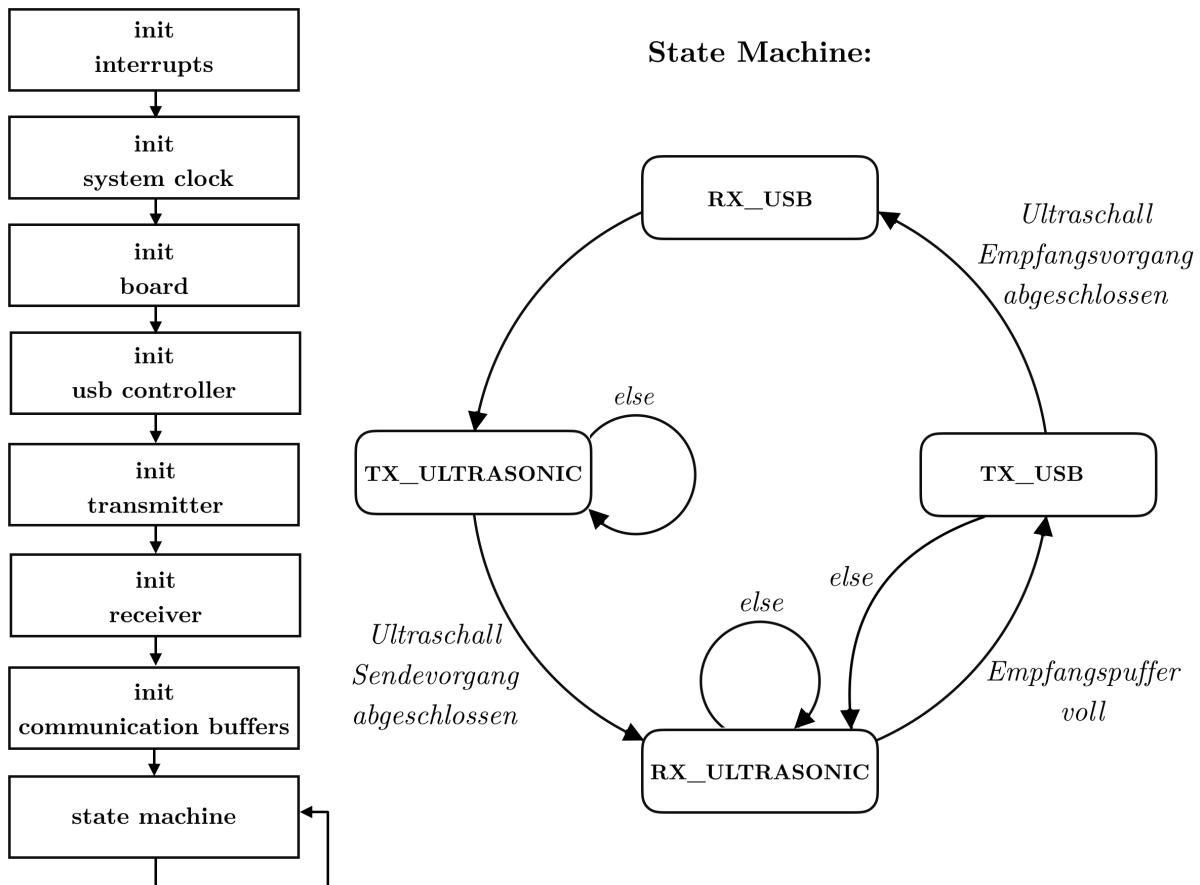


Abbildung 4.2: Hauptprogrammfluss der Software auf dem Mikrocontroller

In der Initialisierung werden als erstes alle Treiber und Module konfiguriert, anschliessend wird in einer Endlosschlaufe die Statemachine aufgerufen. Diese unterscheidet vier verschiedene Zustände, welche in folgender Auflistung näher beschrieben sind.

- **RX\_USB:** Dies ist der erste State nach dem Initialisierungsvorgang und als einziger ein blockierender State. In einer Endlosschlaufe wird gewartet, bis per USB ein Kontrollpuffer vom Host an die Firmware geschickt wurde. Nach Erhalt eines Puffers werden die darin enthaltenen Einstellungen extrahiert. Dies sind der aktuelle Sendewinkel, die gewünschte Amplitudenbelegung, die Anzahl Pulse mit welcher der Ultraschalltransceiver angeregt

wird und die Anzahl der Datenpuffer, welche während dem Empfangsvorgang zu füllen sind. Danach wird falls nötig das Ultraschall Sendemodul mit den neuen Einstellungen initialisiert und in den State TX\_ULTRASONIC gewechselt.

- **TX\_ULTRASONIC:** Während diesem State werden Ultraschallsignale gesendet. Dies geschieht per PWM (Pulsweitenmodulation). Dieser State bleibt solange aktiv, bis die Funktion `transmitter_send(int angle)` aus dem Modul "transmitter.c" per Rückgabewert signalisiert, dass der Sendevorgang abgeschlossen ist. Danach wird in den State RX\_ULTRASONIC gewechselt.
- **RX\_ULTRASONIC:** In diesem State werden vom ADC die empfangenen und verstärkten Ultraschallsignale abgetastet. Dabei werden Empfangspuffer gefüllt. Sobald ein Empfangspuffer voll ist, wird über den Rückgabewert der Funktion `receiver_get_buffer(int *receiver_finished_flag)` ein Pointer auf diesen Puffer übergeben. Dadurch wird in den State TX\_USB gewechselt. Solange der Rückgabewert hingegen ein NULL-Pointer ist wird im bisherigen State verblieben.
- **TX\_USB:** In diesem State werden die Empfangspuffer mit den vom ADC abgetasteten Werten per USB an den Host gesendet. Dabei wird pro ADC-Kanal ein USB-Puffer der Länge 1024 Bytes (512 16-Bit-Werte) gesendet. Vom USB-Stack werden diese Puffer in kleineren Teilstücken von 512 Bytes übermittelt, was aber auf die Statemachine keinen Einfluss hat. Ein zusätzlicher Puffer mit Statusinformationen wird als letztes gesendet. Danach wird anhand des im State RX\_ULTRASONIC gesetzten `receiver_finished_flag` entschieden, ob der Empfangsvorgang vollständig abgeschlossen ist und in den State RX\_USB gewechselt werden kann, oder ob im State RX\_ULTRASONIC auf weitere Empfangspuffer gewartet werden muss.

#### 4.2.2 Ultraschall Sender – transmitter.c

Das Modul "transmitter.c" steuert die Senderschaltung des Ultraschall Phased Arrays. Dazu gehören die Levelshifter für die Verstärkung der Sendesignale und für das Dämpfen der Ultraschalltransceiver. Zusätzlich werden von diesem Modul auch die Analogschalter der Empfängerorschaltung gesteuert.

Die einzelnen Kanäle des Levelshifters werden mit einem PWM-Signal der Frequenz 40kHz angesteuert. Um eine Richtwirkung zu erzielen, werden die PWM-Kanäle gegeneinander phasenverschoben betrieben. Dazu werden sie zeitlich verschoben eingeschaltet. Die Erzeugung dieser zeitlichen Verschiebung ist nicht ganz einfach, da sehr kleine Verschiebungen schon grosse Winkelveränderungen verursachen. Eine interruptgesteuerte Realisierung ist so zum Beispiel nicht möglich, da die Interrupt-Latenzzeit im Mikrosekundenbereich liegt, während die kleinste gewünschte Phasenverschiebung lediglich bei  $\approx 250\text{ns}$  liegt. Aus diesem Grund wird auf eine relativ primitive Methode zurückgegriffen, bei welcher zwischen dem Einschalten der Kanäle eine winkelabhängige Anzahl Prozessortakte gewartet wird. Diese Wartezeit wird mithilfe von inline Assembler `nop` (no operation) Befehlen realisiert.

Der Ablauf des Sendevorgangs ist in einer Statemachine implementiert, welche zwischen den drei States START, RUNNING und ATTENTUATE unterscheidet. Zu Beginn des Sendevorgangs ist der State START aktiv. Sobald aus dem Hauptprogramm heraus die Funktion `transmitter_send(int angle)` zum ersten Mal aufgerufen wird, werden die PWM-Kanäle eingeschaltet und der aktuelle State auf RUNNING gesetzt.

Während dem Senden muss für jeden Kanal gezählt werden, wie viele Pulse schon gesendet wurden. Da dies nicht so zeitkritisch ist wie das Einschalten, ist eine Implementation mithilfe einer Interrupt Routine möglich. Darin werden beim Erreichen der Maximalzahl Pulse die einzelnen PWM-Kanäle gleich ausgeschaltet. Nach dem Ausschalten des letzten Kanals wird der Dämpfungsvorgang gestartet und der State dementsprechend auf ATTENTUATE gewechselt.

Der Dämpfungsvorgang ist über einen Timer gesteuert. Sobald dieser Timer einen Schwellwert überschreitet, wird der Dämpfungsvorgang beendet, der State auf **START** gesetzt und dem Hauptprogramm mit dem Rückgabewert der Funktion `transmitter_send(int angle)` mitgeteilt, dass der Sendevorgang beendet ist.

Neben dem Sendewinkel und der Anzahl zu sendender Ultraschallpulse kann auch die Amplitudenbelegung verändert werden. Eine Änderung der Spannungs- oder Stromamplitude an einem Ultraschalltransceiver hat eine Änderung der abgestrahlten Leistung zur Folge, welche quadratisch von der Amplitudenänderung abhängt. Da die Levelshifter in der Senderschaltung (siehe Kapitel 3.3.2) nicht einfach eine andere Spannung ausgeben können, wird die Sendeleistung mithilfe der Pulsbreite der PWM-Ansteuerung beeinflusst. Hier ist zu beachten, dass die abgestrahlte Leistung nicht quadratisch, sondern linear von der Pulsbreite abhängt. Die Werte für die Amplitudenbelegung müssen also im Quadrat mit der Pulsbreite multipliziert werden.

#### 4.2.3 Ultraschall Empfänger – `receiver.c`

Im Modul "receiver.c" werden die empfangenen analogen Ultraschallsignale digitalisiert. Dafür wird der interne Analog-Digital-Konverter (ADC) vom Mikrocontroller SAM3X8E eingesetzt. Für alle acht abzutastenden Kanäle steht lediglich ein einzelner ADC zur Verfügung. Dieser tastet demnach alle Kanäle in kurzen Abständen nacheinander ab. Dieses Verfahren nennt man Multiplexing und ist im Kapitel 2.3.2 näher beschrieben. Der gemessene zeitliche Abstand zwischen den Konversionen zweier benachbarter Kanäle beträgt  $2\mu\text{s}$ . Die daraus resultierende Phasenverschiebung bei einem 40kHz-Signal ist zu gross, um sie vernachlässigen zu können. Während dem Upsampling auf dem Hostsystem wird sie korrigiert (siehe Kapitel 2.4.1).

Der Abtastvorgang der analogen Daten auf dem ADC des Mikrocontrollers läuft grösstenteils hardwaregesteuert ab. Als Trigger wird ein Hardware Timer verwendet. Dadurch ist die Abtastfrequenz konstant und wird nicht durch Interrupts beeinflusst. Beim interruptbasierten Triggern des ADC wurden teilweise Verzögerungen von mehr als  $2\mu\text{s}$  gemessen, welche beim timerbasierten Triggern nicht auftreten.

Das Multiplexing findet mithilfe des "Sequencer" Modus statt. In diesem Modus werden sämtliche vorkonfigurierten ADC-Kanäle automatisch nacheinander konvertiert, sobald der ADC getriggert wurde. Die abgetasteten Werte werden ebenfalls hardwaregesteuert vom DMA-Controller (Direct Memory Access) hintereinander in einen zuvor festgelegten Puffer gespeichert.

Der Ablauf eines Empfangsvorgangs sieht wie folgt aus: Aus der Statemachine im Hauptprogramm heraus wird einmalig die Funktion `receiver_start(int bank_count)` aufgerufen. Diese Funktion startet den ADC Trigger Timer. Der Übergabeparameter `bank_count` legt fest, wie oft der Datenpuffer gefüllt werden soll bevor der Empfangsvorgang beendet wird. Die Statemachine im Hauptprogramm überprüft nun regelmässig über die Funktion `*receiver_get_buffer(int *stopped_flag)`, ob ein neuer Datenpuffer gefüllt ist. Ist dies der Fall, liefert die Funktion einen Pointer auf den gefüllten Puffer, ansonsten einen NULL-Pointer. Die Variable `stopped_flag` signalisiert den letzten Datenpuffer nachdem der Empfangsvorgang beendet wurde.

Insgesamt gibt es 4 Datenpuffer, welche alle gleich gross sind. Zwei davon werden als temporäre Puffer (`temp_buffer`) bezeichnet, zwei als Empfängerpuffer (`receiver_buffer`). Die zwei temporären Puffer werden vom DMA-Controller des ADC hardwaregesteuert beschrieben und in der ADC Interrupt Routine in die Empfängerpuffer umsortiert. Es wird jeweils ein Puffer beschrieben, während der andere ausgelesen wird, danach umgekehrt. Die beiden Empfängerpuffer werden in der ADC Interrupt Routine beschrieben und im Hauptprogramm vom USB-Controller ausgelesen. Auch hier werden die Puffer jeweils nach dem Schreiben/Lesen ausgetauscht.

Pro Kanal werden vom ADC per DMA je 512 Werte (16-Bit Integer) in die temporären Datenpuffer `temp_buffer` gespeichert, demzufolge sind alle Puffer  $512 \cdot 8 \cdot 2B = 8192B$  gross. Jedes Mal wenn der ADC vom Timer Trigger ausgelöst wird, schreibt er die Werte der 8 Kanäle der Reihe nach in den temporären Puffer. Bezeichnet man die Kanäle mit den Buchstaben a-h, sieht dieser Puffer demzufolge so aus:

$$\text{temp\_buffer}[0] = [a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0, a_1, b_1, \dots, a_{511}, b_{511}, c_{511}, d_{511}, e_{511}, f_{511}, g_{511}, h_{511}]$$

Sobald nun der erste Puffer voll ist, wird ein ADC Interrupt ausgelöst. In diesem Interrupt wird zunächst dem DMA-Controller des ADC der zweite temporäre Puffer übergeben. Danach wird der soeben gefüllte Puffer in einen der zwei Empfängerpuffer umsortiert. Nach dem Umsortieren stehen die Werte wie folgt in dem USB Empfängerpuffer:

$$\text{receiver\_buffer}[0] = [a_0, a_1, a_2, \dots, a_{510}, a_{511}, b_0, b_1, \dots, h_{510}, h_{511}]$$

Sobald ein Puffer umsortiert ist, wird dies dem Hauptprogramm über die oben schon erwähnte Funktion `*receiver_get_buffer(int *stopped_flag)` mitgeteilt. Im Hauptprogramm wird dann der Puffer dem USB-Controller übergeben und an den Host gesendet. Das Umsortieren der Daten hat den Zweck, dass auf dem Host die Daten schon nach Kanal getrennt ankommen, so dass dort keine Rechenleistung mit Umsortieren verschwendet wird.

### 4.3 Hostseitige Software auf dem Desktop-Betriebssystem

Das Pythonprogramm übernimmt als Hauptaufgabe die Vermittlung zwischen User-Interface (Client) und Hardware (Firmware). Es stellt dafür einen Webserver zur Verfügung, verarbeitet Benutzereingaben und bereitet Rohdaten mittels Signalverarbeitung auf. Der Webserver ist mithilfe des Webframeworks Flask implementiert. Eine Übersicht des Programms ist in Abbildung 4.3 dargestellt.

Das Hauptprogramm wird über das File "server\_flask.py" gestartet. Es generiert eine Webapplikation bestehend aus Server und Applikation. Bei einem Client-Zugriff auf den Webserver wird auf dem Host eine zur aufgerufenen HTTP-Adresse zugehörige Python-Funktion ausgeführt. So wird z.B. beim Aufruf der Adresse `http://address:5000/` die Funktion `index()` aufgerufen. Diese Funktion erzeugt dynamischen HTML-Code für das GUI, welcher daraufhin zur Ausführung an den Client (Webbrowser) übergeben wird. Näheres zum clientseitigen Code ist im Kapitel 4.4 zu finden. Die Applikation startet zudem den Firmware Controller, welcher als Instanz der Klasse `FirmwareController` in einem eigenen Thread ausgeführt wird. Dieser kommuniziert über USB mit der Hardware und verarbeitet die Rohdaten mithilfe statischer Methoden aus der Klasse `DSP`.

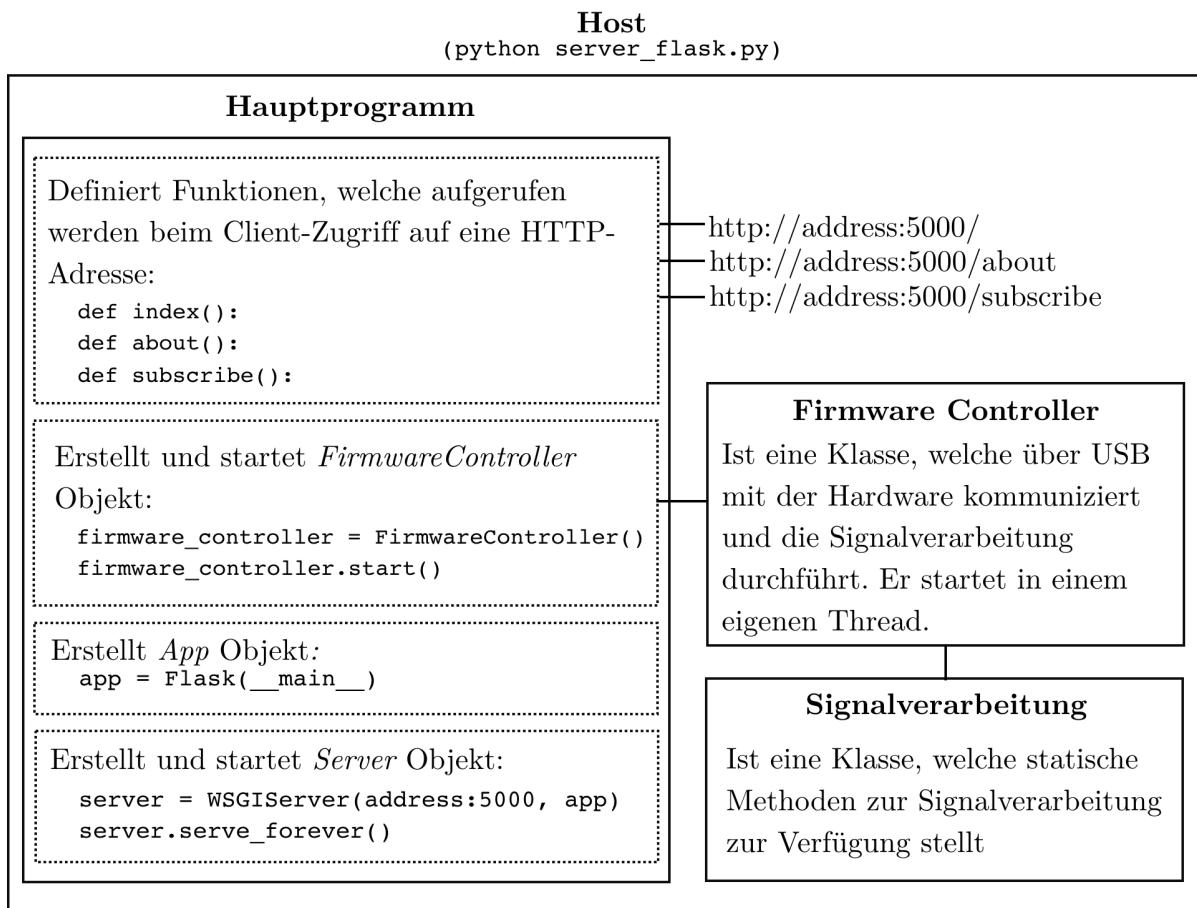


Abbildung 4.3: Übersicht Host

### 4.3.1 Hauptprogramm – `server_flask.py`

Flask ist ein BSD lizenziertes, kleines Webframework, welches es ermöglicht, mit wenigen Zeilen Code eine Webapplikation zu erstellen. Das Web Server Gateway Interface (WSGI) ist eine Schnittstellenspezifikation für die Kommunikation zwischen einer Python-Applikation und dem Server. Flask generiert WSGI-konforme Apps, welche folglich von WSGI-kompatiblen Servern ausgeführt werden können. Als Server wird der WSGI-Server der Bibliothek `Gevent` verwendet.

Wie in Abbildung 4.3 ersichtlich, wird beim Client-Zugriff auf eine HTTP-Adresse eine zugehörige Python Funktion ausgeführt. Es wird zwischen folgenden drei Funktionen unterschieden:

- `index()` : `http://adress:5000`
- `about()` : `http://adress:5000/about`
- `subscribe()` : `http://adress:5000/subscribe`

Die Funktion `index()` generiert den HTML-Code für das GUI auf der Basis eines HTML-Templates (siehe Kapitel 4.4.1). Es werden dafür die Eingabefelder des GUIs ausgelesen und die aktualisierten Benutzereingaben an den Firmware Controller übergeben. Da das Hauptprogramm über ein Firmware Controller Objekt verfügt, ist dies mithilfe der Funktion `firmware_controller.update_values(...)` möglich. Mit der Flask-Funktion `render_template(html_template, python_variables)` wird der HTML Code generiert. Dieser Funktion werden als Python-Variablen Eingabefelder (HTML kompatibel) mit den aktualisierten Werten mitgegeben.

Die Funktion `about()` generiert statischen HTML-Code mit Informationen über das Projekt.

Über die Funktion `subscribe()` abonniert der Client die asynchronen Updates, welche jeweils das darzustellende Signal mit dazugehörigen Informationen enthalten. Dafür werden Server Sent Event Objekte mit den neuen Daten aus dem Firmware Controller erstellt und an den Client übertragen.

Damit diese asynchrone Schnittstelle möglich wird, verfügt die Applikation über ein Interface Objekt. Dieses aktualisiert sich über eine Callback-Funktion, welche vom Firmware Controller aufgerufen wird, sobald neue Daten vorhanden sind. Das Hauptprogramm hat über die Funktion `interface.get_data()` Zugriff auf die darzustellenden Ultraschallsignale.

Flask erstellt aus dem Source-Code eine WSGI-konforme Applikation. Mit dieser wird daraufhin der WSGI-Server erstellt, welcher schlussendlich gestartet wird (siehe Abbildung 4.3).

### 4.3.2 Firmware Controller – `firmware_controller.py`

Der Firmware Controller vermittelt zwischen der Flask-Applikation und der Firmware auf dem Arduino. Er erstellt dabei aus Benutzereingaben, welche in der Flask-Applikation gespeichert sind, Steuerbefehle für den Arduino. Umgekehrt bearbeitet er vom Arduino kommende Rohdaten, sodass sie zur Darstellung bereit sind und übergibt diese der Applikation. Eine Übersicht der hostseitigen Software aus der Perspektive des Firmware Controllers ist in Abbildung 4.4 zu sehen.

Das File "firmware\_controller.py" enthält eine Klassendeklaration, die vom Hauptprogramm (Flask-Applikation) genau einmal instanziert wird. Über dieses `FirmwareController` Objekt kommuniziert die Flask-Applikation mit dem Arduino (siehe Abbildung 4.3).

Für den Firmware Controller wird ein eigener Thread gestartet, damit er seine Aufgaben parallel zur Flask-Applikation ausführen kann. Seine Hauptaufgabe führt der Firmware Controller dabei über die Funktion `main_loop()` aus, welche in einer Endlosschleife ausgeführt wird ("Ablauf" in der Abbildung 4.4). Der Sendewinkel ist als globale Variable `angle` im Firmware Controller

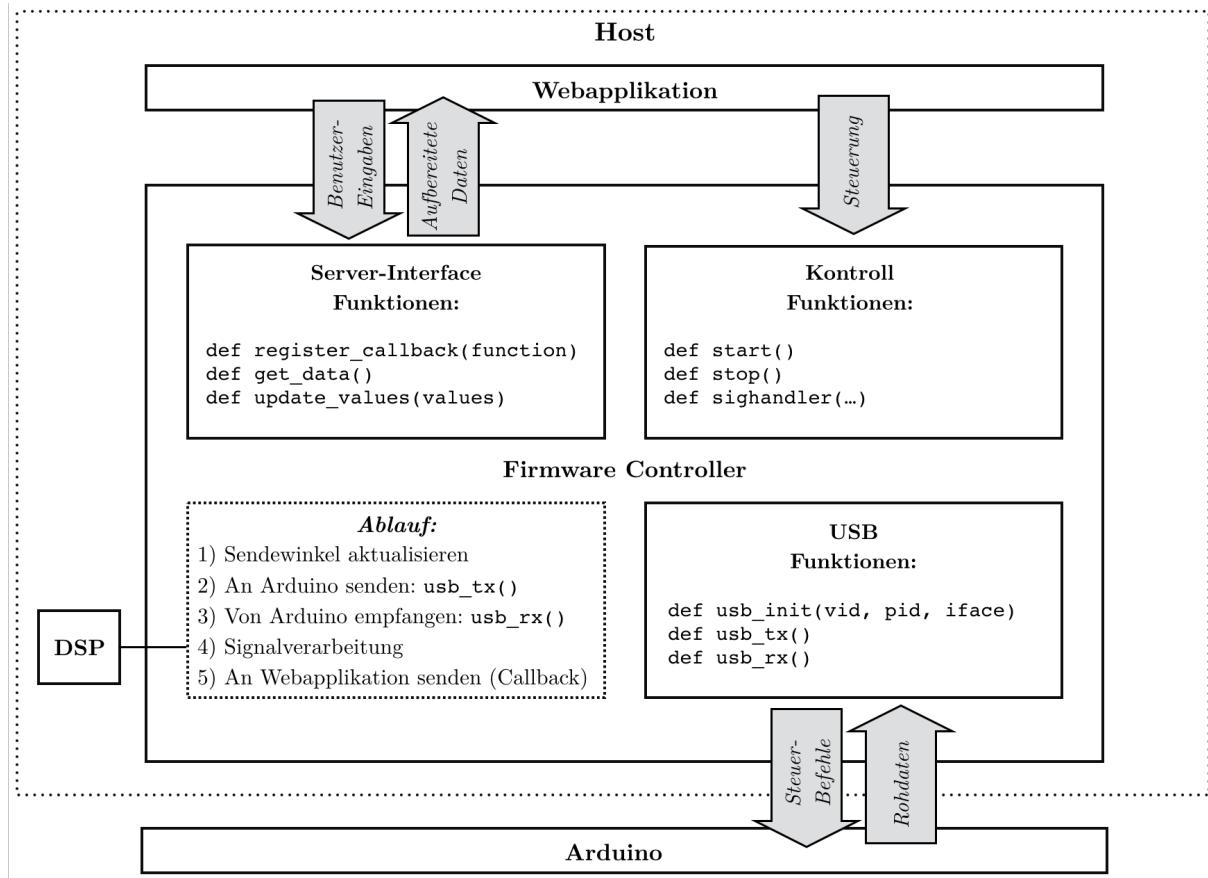


Abbildung 4.4: Die Hostsoftware aus der Perspektive des Firmware Controllers

gespeichert. Vor jedem neuen Zyklus der Endlosschleife wird dieser Sendewinkel inkrementiert oder auf den Startwinkel zurückgesetzt, sodass der im GUI eingestellte Winkelbereich durchlaufen wird. Danach wird der neue Messvorgang für diesen Winkel ausgelöst und die entsprechenden Rohdaten der Ultraschallsignale entgegengenommen. Das darzustellende Signal wird mithilfe der Methoden aus der DSP Klasse (siehe 4.3.3) aus diesen Rohdaten berechnet. Schlussendlich wird das Signal (und weitere Informationen) mithilfe einer Callback-Funktion an die Flask-Applikation übergeben.

Die restlichen Funktionen im Firmware Controller sind anhand ihrer Zuständigkeit grob in drei Bereiche unterteilt (siehe Abbildung 4.4):

- Kontroll-Funktionen
- USB-Funktionen
- Server-Interface-Funktionen

Die *Kontroll-Funktionen* sind für das Starten und Beenden des Firmware Controllers verantwortlich. Mit der Funktion `firmware_controller.start()` wird das USB-Interface initialisiert, ein eigener Thread gestartet und Keyboardinterrupt-Signale abgefangen. Danach beginnt der Controller im neuen Thread seine Hauptaufgabe (`main_loop()`). Wenn der Server aus der Shell mit Ctrl-C beendet wird, wird der Firmware Controller mit der Funktion `stop()` beendet. Diese Funktion beendet den Thread des Firmware Controllers.

Für die Kommunikation mit dem Arduino sind *USB-Funktionen* implementiert. Nachdem das USB-Interface mithilfe der Funktion `usb_init(vid, pid, iface)` initialisiert ist, kann mit

der Funktion `usb_tx()` ein Messvorgang auf der Firmware (Arduino) gestartet werden. Der Firmware Controller hat die aktuellsten Eingaben aus dem User-Interface und den aktuellen Sendewinkel gespeichert. Mit diesen Variablen ist der auszuführende Messvorgang spezifiziert. Da innerhalb der Funktion `usb_tx()` direkt auf diese Variablen zugegriffen wird, benötigt die Funktion keine Übergabewerte. Mit der Funktion `usb_rx()` werden die Rohdaten der gemessenen Ultraschallsignale entgegengenommen. Die im GUI ausgewählte Anzahl Puffer bestimmt dabei die Grösse (Signallänge) der empfangenen USB-Daten.

Die *Server-Interface-Funktionen* sind für die Kommunikation zwischen der Flask-Applikation und dem Firmware Controller zuständig. Die Flask-Applikation übergibt die aktuellen Benutzereingaben mithilfe der Funktion `firmware_controller.update_values(...)` dem Firmware Controller. Über die Funktion `firmware_controller.register_callback(callback_function)` speichert die Flask-Applikation eine Funktion im Firmware Controller, welche jeweils sofort nach der Signalverarbeitung ausgeführt wird. Bei dieser von der Flask-Applikation übergebenen Funktion handelt es sich um `firmware_controller.get_data()`. Damit werden die asynchronen Updates zwischen Client und Host ausgelöst (siehe Kapitel 4.3.1).

### 4.3.3 Signalverarbeitung – `dsp.py`

Echos von Objekten im Umfeld des Phased Arrays kommen mit zeitlichen Verzögerungen bei den acht Ultraschall Empfängern an. Durch die Auswertung dieser Verzögerungen können Informationen über Distanz und Winkel dieser Objekte berechnet werden. Aus den Rohdaten der acht Signale wird ein einzelnes Signal berechnet, welches anzeigt, aus welcher Distanz Echos aus einem bestimmten Winkel stammen. Dieses Signal wird im User-Interface dargestellt.

Im Kapitel 2.4 ist die dazugehörige Theorie zusammengefasst. Die Rohdaten durchlaufen der Reihe nach folgende Verarbeitungsschritte:

- 1: Upsampling
- 2: Beamforming
- 3: Enveloppe berechnen
- 4: Downsampling

Die Klasse `DSP` stellt die dafür nötigen Signalverarbeitungs-Methoden als statische Funktionen zur Verfügung. Diese Klasse ist im File "dsp.py" deklariert und wird vom Firmware Controller importiert. Es werden für die Berechnungen Funktionen aus den open source Bibliotheken `numpy` und `scipy` verwendet. Die Signalverarbeitung ist unterteilt in folgende drei Funktionen:

- `get_aperture(aperture, max_n)`
- `upsamp_beam_form(y, phi_rad, aperture, sonic_speed, upsample_factor)`
- `downsamp_env(y, downsample_factor)`

Die Funktion `get_aperture(aperture, max_n)` berechnet die im Kapitel 2.2.7 beschriebenen Gewichtungsfaktoren der Amplitudenbelegung für `max_n` Elemente. Der Übergabeparameter `aperture` wird als Integer erwartet. Es kann zwischen Rechteckbelegung (0), cos-Belegung (1),  $\cos^2$ -Belegung (2) und Gaussbelegung (3) ausgewählt werden. Als Resultat wird eine Python-Liste mit den berechneten Werten zurückgegeben.

Die Funktion `upsamp_beam_form(y, phi_rad, aperture, sonic_speed, upsample_factor)` erhöht die Abtastrate der unterabgetasteten Rohdaten um den Faktor `upsample_factor` (siehe Kapitel 2.4.2). Falls gewünscht, werden die Signale mit einer Amplitudenbelegung versehen (siehe Kapitel 2.2.7). Schlussendlich werden die Signale mittels Beamforming (siehe Kapitel 2.4.4) zu einem Signal addiert, welches eine empfangsseitige Richtwirkung in die Senderichtung

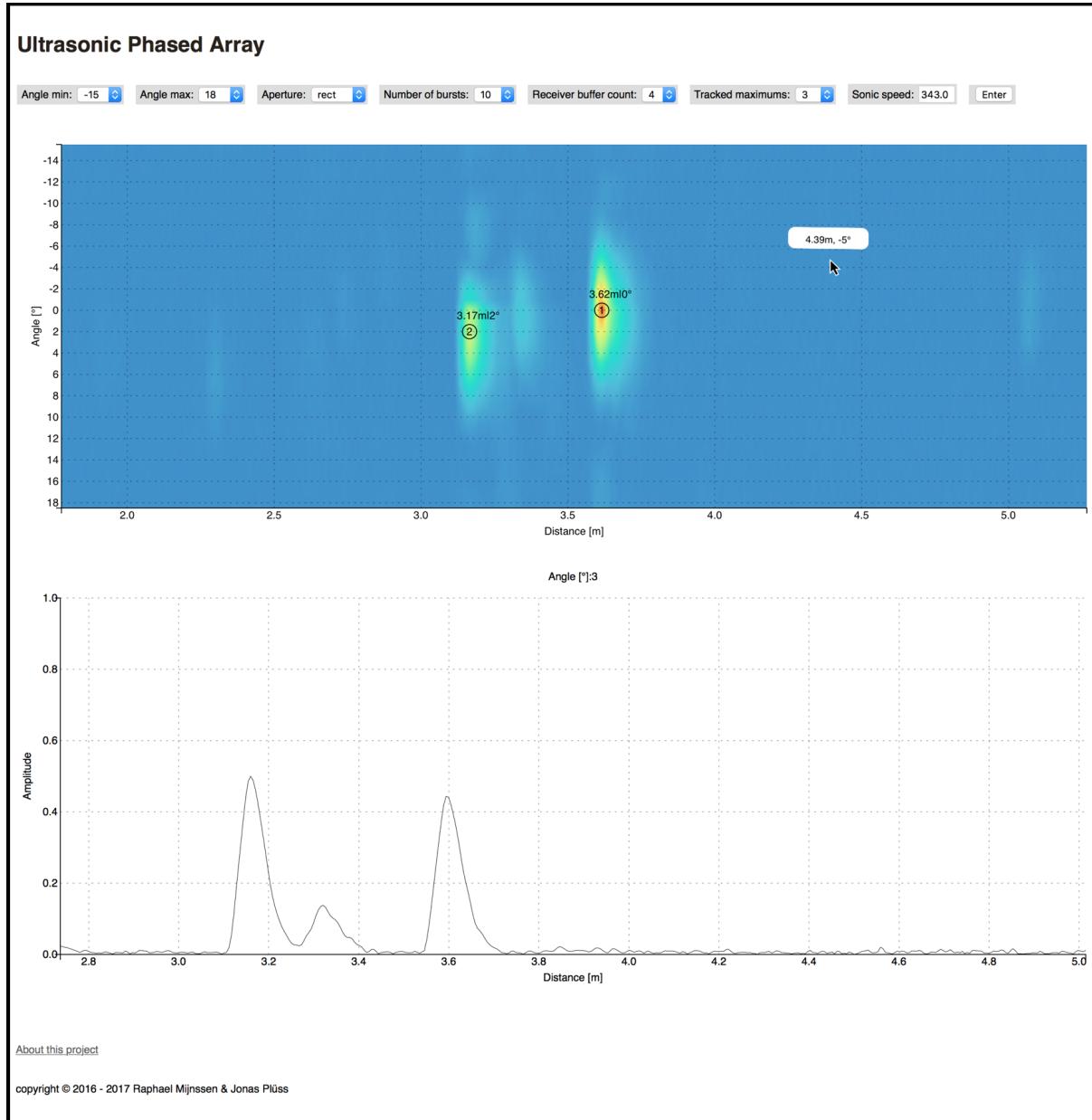
aufweist. Die Funktion übernimmt als Eingabeparameter die acht unterabgetasteten Signale der Ultraschalltransceiver  $y$  als  $1 \times 8$  Python-Liste. Zusätzlich wird der Sendewinkel in Radian, die gewünschte Apertur, die Schallgeschwindigkeit und der Upsampling-Faktor übergeben.

Nachfolgend wird der Ablauf dieser Funktion noch etwas genauer beschrieben. Falls eine Amplitudenbelegung gewünscht ist, werden die acht Ultraschallsignale als erstes mit Gewichtungsfaktoren multipliziert, welche mit der Funktion `get_aperture(aperture, max_n)` berechnet werden. Diese unterabgetasteten Signale werden danach mittels `numpy FFT` in den Frequenzbereich transformiert und eine Zeitverschiebung im Frequenzbereich (siehe Kapitel 2.4.1) durchgeführt. Es wird die unerwünschte Zeitverschiebung durch das Multiplexing korrigiert (siehe Kapitel 2.3.2) und die nötige Zeitverschiebung für die empfangsseitige Richtwirkung durchgeführt (siehe Kapitel 2.4.4). Aus jedem dieser acht komplexen Frequenzvektoren wird daraufhin ein um den Faktor `upsample_factor` grösser Frequenzvektor zusammengesetzt (siehe Kapitel 2.4.3). Diese Frequenzvektoren werden mithilfe der `numpy IFFT` in den Zeitbereich zurücktransformiert und zum gewünschten Signal aufaddiert.

Die Funktion `downsamp_env(y, downsample_factor)` berechnet die Enveloppe des Signals  $y$  und reduziert danach die Abtastrate des Signals um den Faktor `downsample_factor` (siehe Kapitel 2.4.6). Das Signal  $y$  wird als Array erwartet und das resultierende Signal auch wieder als Array zurückgegeben. Mithilfe der Funktion `hilbert(y)` aus der `scipy` Bibliothek wird das analytische Signal berechnet (siehe Kapitel 2.4.5). Der Betrag dieses komplexen Zeitsignals entspricht der Enveloppe. Da dieses Basisbandsignal keine hohen Frequenzen enthält, kann auf ein Tiefpassfilter vor der Unterabtastung verzichtet werden. Aus jedem `downsample_factor`-ten Wert des Enveloppen-Signals wird das gewünschte Signal zusammengesetzt.

## 4.4 Clientseitige Software auf dem Desktop-Betriebssystem

Die clientseitige Software ist das Bindeglied zwischen dem Benutzer und dem Host. Über eine graphische Benutzeroberfläche (GUI) werden Einstellungen verändert, Messvorgänge ausgelöst und die Messresultate präsentiert. Das GUI wird über einen Webbrower aufgerufen und ist in den Programmiersprachen HTML, CSS und Javascript geschrieben. In Abbildung 4.5 ist ein Screenshot des GUIs zu sehen.



**Abbildung 4.5:** Die grafische Benutzeroberfläche

Der Benutzer hat über die HTTP Seite `http://server_address:5000/` Zugriff auf die grafische Benutzeroberfläche. Sobald die Seite geladen ist, laufen die Messungen mit anschliessender Darstellung der Resultate automatisch. Die Darstellungen aktualisieren sich dabei nach jedem Messvorgang. Beide Graphen sind zoom- und schiebbar. Wird der Mauszeiger über die Graphen bewegt, werden mithilfe von Tooltips Informationen zur Position im Graph eingebendet (Winkel und Distanz im oberen Graph in Abbildung 4.5).

Die Webseite kann grob in 3 Bereiche unterteilt werden:

- Eingaben
- Heatmap-Graph
- Liniensplot

Im ersten Bereich sind die Eingabeformulare zu sehen. Der Benutzer kann dort folgende Einstellungen verändern:

- Minimaler Winkel und maximaler Winkel: Der Bereich dazwischen wird "gescannt" (Maximal  $\pm 30^\circ$ )
- Aperture: Auswahl der Amplitudenbelegung (Rechteck, cos,  $\cos^2$  oder Gauss)
- Number of Bursts: Anzahl Sendepulse (Gerade Zahl zwischen 0 und 80)
- Receiver Buffer Count: Anzahl Daten Puffer (1,2,4 oder 8 / bestimmt die Messdauer)
- Tracked Maxima: Anzahl Maxima welche verfolgt/gespeichert werden (0 - 15)
- Sonic Speed: Schallgeschwindigkeit

Im Heatmap-Graph (erster Graph in Abbildung 4.5) ist ein "Bild" der Umgebung zu sehen. Die y-Achse ist der ausgewählte Winkelbereich und die x-Achse die Distanz. Farblich codiert wird die Stärke von Echos aus dem jeweiligen Winkel aufgetragen. Es handelt sich dabei um das im Kapitel 4.3.3 beschriebene Signal. Die zeitliche Information dieses Signals (x-Achse) entspricht der zurückgelegten Distanz des Schalls. Für jede neue Messung wird jeweils eine Zeile im Bild an der richtigen Stelle überschrieben. Die Scanrichtung verläuft vom minimalen Winkel zum maximalen Winkel. Falls das "Maximum Tracking" aktiviert ist, werden gefundene Maxima mit dazugehörigem Winkel und Distanz angezeigt (4.5). Diese sind der Grösse nach sortiert. Im obigen Bild entspricht das grosse Maximum der Decke in einem hohen Raum und das kleinere Maximum einer Runden Lampe, welche an der Decke befestigt ist.

Im Liniensplot (zweiter Graph in Abbildung 4.5) ist der Verlauf des zum aktuellen Sendewinkel zugehörigen Signals dargestellt. Der aktuelle Winkel ist oberhalb dieses Plots aufgetragen. Wird gewünscht, dass nur in einem Winkel gesendet wird, so können beide Winkeleingaben im GUI auf den selben Winkel gestellt werden. Die Amplituden-Angabe ist normiert auf einen gemessenen Maximalwert. Da es sich nicht um ein Schalldruck-Messgerät handelt, wird auf eine Referenzmessung verzichtet.

#### 4.4.1 HTML und CSS

Die Webseite, welche das GUI zur Verfügung stellt, ist in HTML geschrieben. Es werden die Eingabeformulare (oben in Abbildung 4.5) erzeugt und Javascript-Code ausgeführt, welcher die Graphen erzeugt. Der HTML-Source-Code wird von der Flask-Applikation zur Laufzeit des Programms dynamisch erzeugt und an den Client übergeben (4.3.1). Beim Source-Code, der vom Flask-Server an den Client übergeben wird, handelt es sich nicht um reinen HTML-Code. Es handelt sich um ein in der `Jinja2`-Sprache geschriebenes HTML-Template, in welchem auf Variablen der Flask-Applikation zugegriffen wird.

Der CSS-Code ist im Gegensatz zum HTML-Code statisch. In den zwei Files "style.css" und "style\_plot.css" sind die Darstellungsoptionen für die Webseite und die Plots gespeichert.

Sobald die Benutzereingaben bestätigt sind, werden diese per HTTP POST-Request an den Host übertragen. Danach wird der Javascript Code ausgeführt.

#### 4.4.2 Javascript – plot.js

Im File "plot.js" sind Javascript-Funktionen deklariert. Diese erstellen die Graphen, updaten diese mit neuen Daten und stellen das SSE-Interface zum asynchronen Empfangen neuer Daten zur Verfügung. Die Plots werden mithilfe der Bibliothek `d3` erstellt. Diese Bibliothek vereinfacht den Umgang mit Vektorelementen im Browser und hilft dabei, dynamische und interaktive Plots zu erstellen.

Zusätzlich werden die Positionen der Maxima im ausgewählten Winkelbereich mit einer Javascript-Funktion berechnet. Konsequenterweise müsste dieses Maximum-Tracking auf dem Server stattfinden, mit der restlichen Signalverarbeitung. Durch eine Berechnung im Client kann jedoch der Server entlastet werden. Somit kann z.B. ein Raspberry Pi als Host eingesetzt werden.

Aus dem HTML-File werden die Javascript Funktionen folgendermassen aufgerufen:

```

1 tracking_obj = start_maximum_tracking( numb_of_maximums );
2
3 timeseries_graph = make_timeseries_plot( '#time_series_plot' );
4 heatmap_graph = make_heatmap_plot( '#heatmap_plot' , maximum_tracking_obj );
5
6 sse_listener( heatmap_graph.update_plot , timeseries_graph.update_plot );

```

Die Funktion `start_maximum_tracking(n)` speichert Informationen zu den `n` grössten Peaks im gescannten Winkelbereich im Objekt `tracking_obj`. Die Aktualisierung dieser Maxima findet beim Eintreffen neuer Daten statt. Es wird dabei nur das neuste Signal untersucht. Als erstes werden Peaks gesucht und diese gespeichert. Es wird der Euklidische Abstand dieser Peaks zu den bereits gespeicherten Maxima berechnet (Distanz zu den entsprechenden Punkten im Heatmap-Graph). Schlussendlich wird anhand dieses Abstands entschieden, welche dieser neuen Peaks gespeichert werden sollen, sodass keine Maxima direkt nebeneinander liegen.

Die Funktionen zum Erzeugen der Plots erwarten als Übergabeparameter die ID des HTML-Div-Containers, in welchem der Plot gezeichnet werden soll. Der Funktion zum Erzeugen der Heatmap kann ein weiterer Übergabeparameter mitgegeben werden. Wird als zweites Element das `tracking_obj` übergeben, so werden die darin gespeicherten Maxima im Plot dargestellt. Als Rückgabeparameter geben die Funktionen jeweils ein Objekt zurück, über welches eine Funktion `update_plot(new_data)` aufgerufen werden kann. Damit können die Plots aktualisiert werden.

Der Heatmap-Plot wird aus Teilbildern zusammengesetzt, die in ein HTML5-Canvas-Element gezeichnet werden. Für jedes neue Signal wird eine Zeile im Bild ersetzt. Jedes Pixel des Bildes entspricht dabei einem Abtastwert aus einem Signal. Dieses Bild wird als unsichtbares HTML5-Canvas gespeichert. Je nach Zoom-/Schiebeeinstellung wird ein skalierter Ausschnitt davon auf ein sichtbares HTML5-Canvas-Element gezeichnet.

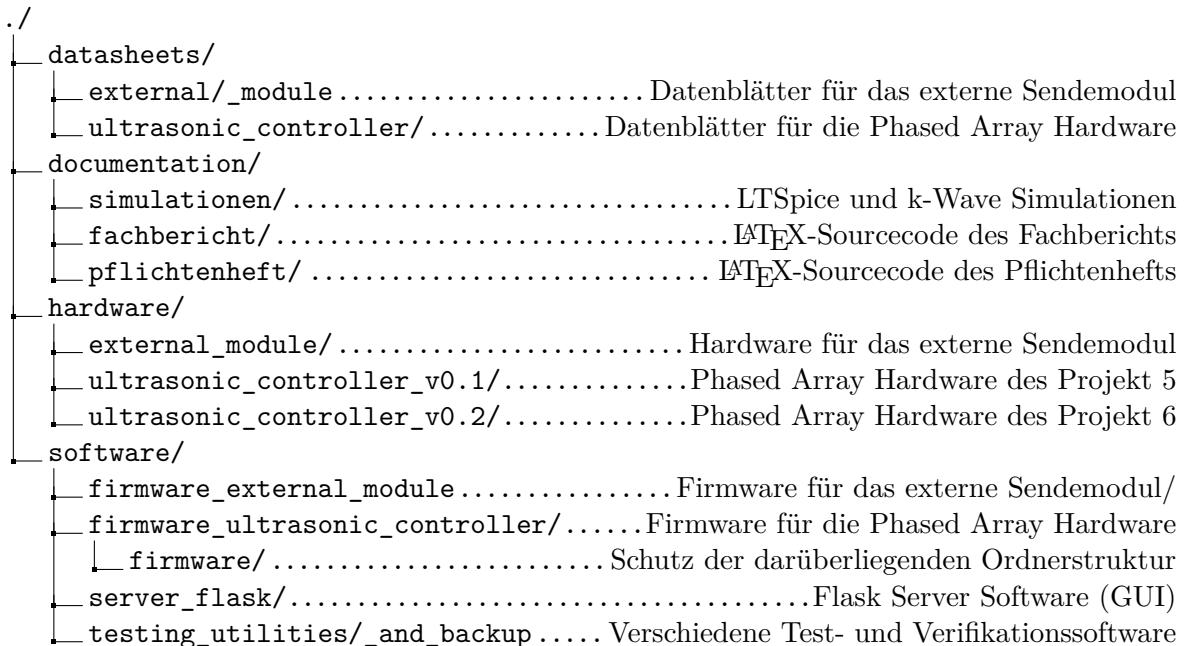
Die Funktion `sse_listener(functions)` ist ein Eventlistener. Sie ruft alle ihr übergebenen Funktionen auf, sobald ein neuer Server Sent Event (siehe Kapitel 4.3.1) eingetroffen ist. Dabei übergibt sie diesen Funktionen die neuen Daten. Ihr werden die beiden `update_plot` Funktionen übergeben.

## 4.5 Entwicklungsumgebung

Im folgenden Kapitel ist beschrieben, welche Software für die Entwicklung des Projektes verwendet wird. Anhand dieser Beschreibung soll es möglich sein, eine eigene Entwicklungsumgebung aufzusetzen, in welcher das Projekt weiterentwickelt werden kann.

Als Entwicklungsbetriebssystem wird für das ganze Projekt GNU/Linux Debian in verschiedenen Versionen verwendet. Die älteste getestete Version ist Debian 8.0. Sämtliche nachfolgenden Beschreibungen beziehen sich auf die Verwendung einer vergleichbaren Installation. Grundsätzlich wurde aber während des gesamten Projektverlaufs grosser Wert darauf gelegt, plattformunabhängig zu entwickeln. Es ist auch möglich, das Projekt auf einem Microsoft Windows- oder einem Mac-System zum Laufen zu bringen. Die grösste Stolperfalle unter Windows/MacOS ist dabei eine fehlende Installation der USB-Bibliothek `libusb`.

Die folgende Grafik 4.6 zeigt die Struktur des Projektordners.



**Abbildung 4.6:** Struktur des Projektordners

Viele Vorgänge in dem Projektordner wurden mithilfe von Makefiles automatisiert. So können zum Beispiel im Dokumentationsordner der Fachbericht und das Pflichtenheft per `make` Befehl generiert werden, und auch die Firmware wird per Makefile kompiliert und auf dem Mikrocontroller des Arduino DUE geladen. Die Ordnerstruktur sollte soweit selbsterklärend sein, bis auf ein kleines Detail. Der Ordner `./software/firmware_ultrasonic_controller` enthält einen weiteren Ordner `firmware`, der den darüberliegenden Ordner vor Veränderungen durch das Makefile schützt. Dieses generiert während dem Kompilervorgang eine Ordnerstruktur mit Binärdateien im Ordner `./software/firmware_ultrasonic_controller`.

### 4.5.1 Hardwareentwicklung

Das Schema und der PCB für die Hardware sind mithilfe des opensource Programms Kicad entwickelt. Dieses PCB-Design-Tool steht unter der GPL und wird unter anderem vom CERN mitentwickelt.

Kicad ist in seiner Grundinstallation mit relativ grossen Bauteilbibliotheken ausgestattet, so dass nur von wenigen Komponenten Schema und Footprint gezeichnet werden müssen. Außerdem können damit ohne grossen Aufwand auch mehrlagige PCB's entwickelt werden, wovon in diesem Projekt mit einem vierlagigen Print Gebrauch gemacht wurde.

#### 4.5.2 Entwicklungsumgebung für die Firmware

Wie schon im Kapitel 3 erläutert, wird als Hardwareumgebung für den Mikrocontroller das Arduino DUE Board verwendet. Es besitzt einen 32-Bit Mikrocontroller des Typs SAM3X8E. Naheliegend wäre als Softwareumgebung dementsprechend die Arduino IDE. Darauf wird aber bewusst verzichtet, da die Arduino IDE einerseits hauptsächlich für 8-Bit Mikrocontroller entwickelt wurde und daher nicht alle Funktionalitäten des SAM3X8E unterstützt und andererseits einigen unnötigen Ballast mit sich bringt, den man nur schwer entfernen kann. Auch ein Herauslösen einzelner Module aus den Arduino Bibliotheken ist aufgrund ihrer stark verstrickten Abhängigkeiten sehr aufwändig.

Stattdessen wird das Atmel Software Framework (ASF) verwendet, welches sehr viele Treiber für den benutzten Mikrocontroller beinhaltet. Leider umfasst das ASF insgesamt mehr als 1.5GB an Daten, wovon nur ein Bruchteil für das Projekt überhaupt gebraucht werden. Mithilfe des Atmel Studios wurden einmalig die für das Projekt benötigten Bibliotheken aus dem ASF extrahiert und separat gespeichert, wodurch sich die Sourcecodegrösse der ASF auf 5.9MB reduzieren liess. Dadurch wird einerseits die Pflege des ganzen Projektcodes und andererseits auch die Fehlersuche beim Debuggen des eigenen Sourcecodes vereinfacht.

Zum Kompilieren des Sourcecodes greift das Atmel Software Framework im Hintergrund auf den gcc-arm-none-eabi zurück, welcher auf der GNU Compiler Collection (GCC) aufbaut. Darin sind alle Werkzeuge enthalten, welche zum Kompilieren und Linken von C-Code für den SAM3X8E verwendet werden. Um das kompilierte Programm auf den Mikrocontroller zu laden, wird das Programm "Bossac" (bossa-cli) verwendet. Der ganze Kompilier- und Ladevorgang wird über ein Makefile aus dem Atmel Software Framework gesteuert, welches für unsere Zwecke angepasst wurde.

Nachfolgend befindet sich eine kurze Auflistung aller Programme, welche für die Entwicklung der Firmware benötigt werden:

- gnu make v4.0
- gcc-arm-none-eabi v4.8.4
- bossac (bossa-cli) v1.3a
- Doxygen (optional zum Dokumentieren des Sourcecodes)

Die Versionsnummern geben jeweils die älteste erfolgreich getestete Version der einzelnen Programme an. Es ist durchaus möglich, dass auch noch ältere Versionen funktionieren, dies wurde aber nicht verifiziert.

### 4.5.3 Installation auf dem Desktop Betriebssystem

Um die Hardware ansteuern zu können, braucht es ein Desktop Betriebssystem, auf welchem Server und Client ausgeführt werden. Diese sind in Python, HTML, CSS und Javascript geschrieben. Auf dem System, an welchem die Hardware per USB angeschlossen ist und auf dem der Flask Server laufen soll, muss eine funktionierende Pythonumgebung mit folgenden Paketen installiert werden:

- numpy v1.8.2
- scipy v0.14.0
- pyusb v1.0.0
- flask v0.10.1
- wtforms v2.0.1
- gevent v1.0.1

Die Versionen beziehen sich auch hier auf die niedrigste getestete Version. Ausser dem Paket "pyusb" können auf dem verwendeten Debian 8 System alle Pakete über die offiziellen Quellen installiert werden. Bei "pyusb" muss die Installation von Hand ausgeführt werden, da sich in den offiziellen Quellen eine zu alte Version befindet. Am einfachsten installiert man sich dazu das Programm "pip" (python-pip) und installiert das Paket per Befehl `pip install --upgrade pyusb` auf dem System. Auf einer Debian 9 Installation tritt dieses Problem nicht mehr auf.

Um auf den Server zuzugreifen, verwendet man am besten den Browser "Firefox", da der JavaScript Code hauptsächlich auf diesem Browser entwickelt und getestet wurde. Im Browser muss Javascript eingeschaltet werden.

### 4.5.4 Bedienungsanleitung

Im Folgenden wird beschrieben, wie das Ultraschall Phased Array-System in Betrieb genommen wird. Es wird empfohlen dafür ein Linux-Betriebssystem zu verwenden, der Server wurde jedoch auch unter Windows und Mac OS X erfolgreich getestet.

Ist ein Arduino DUE mit der aktuellen Firmware vorhanden, so muss nur noch der Server (Host) in Betrieb genommen, und mit einem Webbrowser darauf zugegriffen werden. Dafür muss das Desktop-Betriebssystem nach der Beschreibung im Kapitel 4.5.3 eingerichtet sein.

Der Arduino DUE muss über den "native USB" Port mit dem System verbunden sein. Danach kann der Server mit folgendem Befehl gestartet werden (alle relativen Pfadangaben beziehen sich auf den Projektordner):

```
python ./software/server_flask/server_flask.py
```

In der Shell wird nach jeder Messung mit der Ausgabe: `ServerFlask: callback function` signalisiert, dass neue Daten vorhanden sind.

Über einen Firefox-Webbrowser wird das User-Interface mit folgender Adresse aufgerufen:

```
http://localhost:5000/
```

Wird von einem anderen System aus über ein Netzwerk auf den Server zugegriffen, so muss `localhost` mit der Adresse des Servers ersetzt werden. Sobald die Daten vom Webbrowser abgeholt werden, wird dies in der Shell mit der Ausgabe: `ServerFlask: event` bestätigt.

Falls sich die Firmware nicht auf dem Arduino DUE befindet, muss die im Kapitel 4.5.2 beschriebene Entwicklungsumgebung eingerichtet werden. Der Arduino DUE muss anschliessend zurückgesetzt werden (Reset-Button). Um die Firmware zu installieren, wechselt man seinen

Arbeitsordner zu: `./software/firmware_ultrasonic_controller/firmware/`. Mit folgendem Befehl wird die Firmware auf dem Arduino DUE installiert:

```
make install
```

Dafür sind Rootrechte nötig.

#### 4.5.5 Lizenzen

Die in diesem Projekt entwickelte Soft- und Hardware werden unter den Bedingungen der GNU GPL v3 (GNU General Public License) freigegeben. Das Atmel Software Framework, aus dem der USB Stack und weitere Treiber für ADC, Timer, PWM, usw. verwendet werden, steht unter einer Lizenz von Atmel, welche im Anhang D zu finden ist.

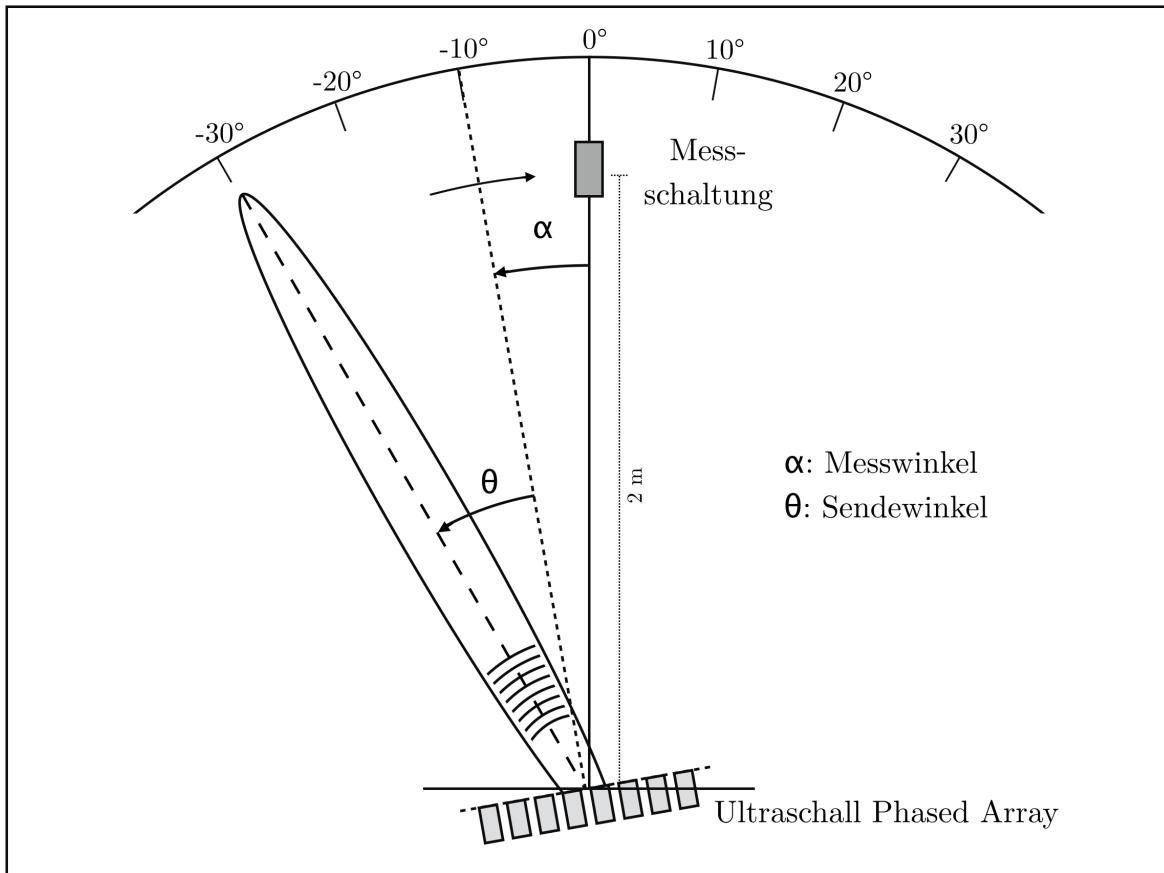
## 5 Test

Das Ultraschall Phased Array wird bezüglich der Sende-Charakteristik, Empfangs-Charakteristik, Distanzauflösung und der maximalen Distanz getestet. Dabei wird jeweils der Messaufbau beschrieben und die Resultate dargestellt und kommentiert. Abschliessend sind ausgewählte Bilder von Umgebungsscans dargestellt und beschrieben. Die Anforderungen sind im Pflichtenheft (siehe F) im Abschnitt "Ziele" zu finden.

### 5.1 Messung der sendeseitigen Richtcharakteristik

Die Richtcharakteristik des Sende-Arrays wird im reflexionsarmen Halbraum gemessen. Das Ultraschall Phased Array ist dabei auf einem Stativ befestigt und auf das Senden von acht Sendepulsen in einem bestimmten Winkel  $\theta$  eingestellt. In einer Distanz von 2m befindet sich die Messschaltung. Sie liegt auf gleicher Höhe wie das Phased Array und ist auf dieses ausgerichtet. Die Messschaltung besteht dabei aus einem extern aufgebauten Kanal der Empfängerschaltung (siehe Kapitel 3.3.1). Das Ultraschall Phased Array-Gerät wird Grad für Grad im Bereich von  $\alpha = \pm 45^\circ$  um die seine vertikale Achse gedreht. Die maximalen Spannungen der Wellenpakete, die an der Messschaltung auftreten, werden jeweils gemessen.

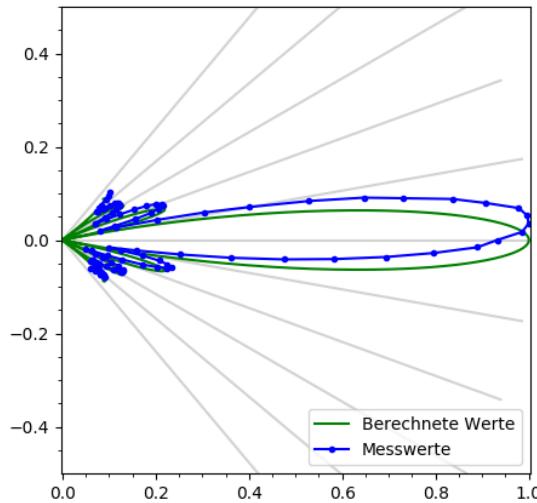
Der Messaufbau ist in Abbildung 5.1 zu sehen.



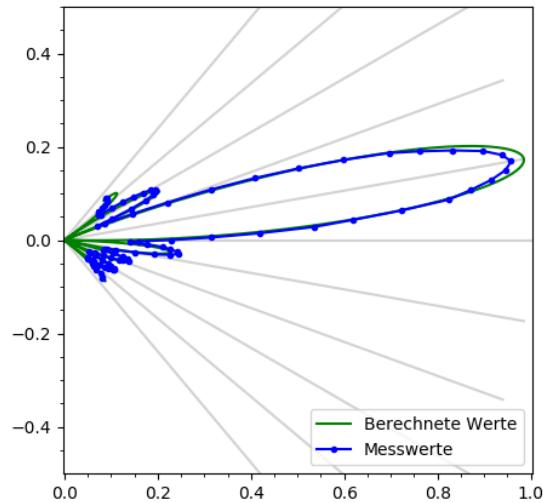
**Abbildung 5.1:** Messaufbau im reflexionsarmen Halbraum

Die Messreihe für einen fest eingestellten Sendewinkel  $\theta$  wird mit berechneten Werten verglichen (Formel 2.14). Die Messwerte sind dabei auf den maximalen Wert der Messreihen normiert (bei  $\theta = 0^\circ$  und  $\alpha = 0^\circ$ ).

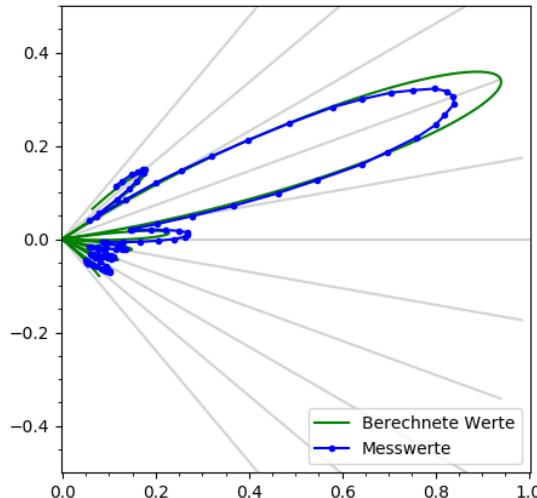
Die Ergebnisse sind in den Abbildungen 5.2, 5.3, 5.4 und 5.5 dargestellt.



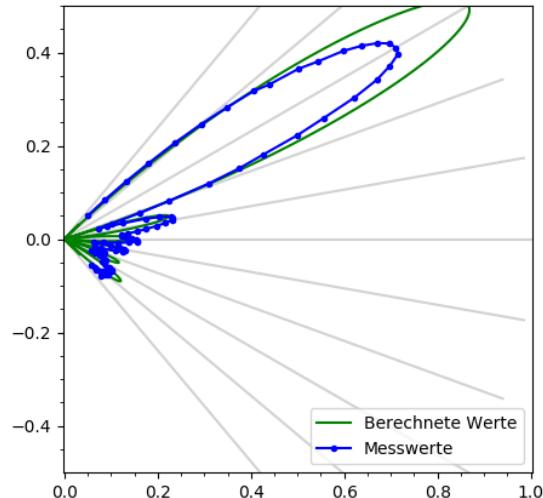
**Abbildung 5.2:** Richtcharakteristik des Sende-  
Arrays bei einem Winkel von  $\theta = 0^\circ$



**Abbildung 5.3:** Richtcharakteristik des Sende-  
Arrays bei einem Winkel von  $\theta = 10^\circ$



**Abbildung 5.4:** Richtcharakteristik des Sende-  
Arrays bei einem Winkel von  $\theta = 20^\circ$



**Abbildung 5.5:** Richtcharakteristik des Sende-  
Arrays bei einem Winkel von  $\theta = 30^\circ$

Die Messresultate stimmen insgesamt gut mit den berechneten Werten überein. Die Abweichungen werden im Folgenden erklärt:

Da ein Rauschpegel vorhanden ist, sinken die Messungen nur auf einen gewissen minimalen Pegel.

In Abbildung 5.2 ist eine Abweichung von ca.  $3^\circ$  zu sehen. Diese ist auf die minimale Verzögerungszeit zwischen dem Aktivieren der einzelnen PWM-Kanäle auf dem Mikrocontroller zurückzuführen. Diese Abweichung tritt im Bereich zwischen  $\theta = \pm 2^\circ$  auf. In den Heatmap-

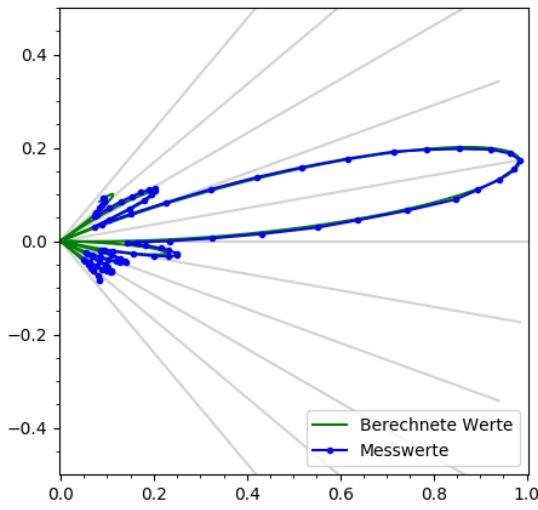
Graphen im GUI äussert sich das Problem als Sprung um  $0^\circ$ , dies ist in Abbildung 5.40 zu sehen.

Da es sich bei den verwendeten Ultraschalltransceiver nicht um ideale Punktquellen handelt, nimmt der Schalldruck für grössere Sendewinkel ab. Dieses nicht-ideale Verhalten ist in Abbildung 5.5 gut zu sehen. Die Hauptkeule ist kleiner, die erste rechte Nebenkeule jedoch grösser als erwartet.

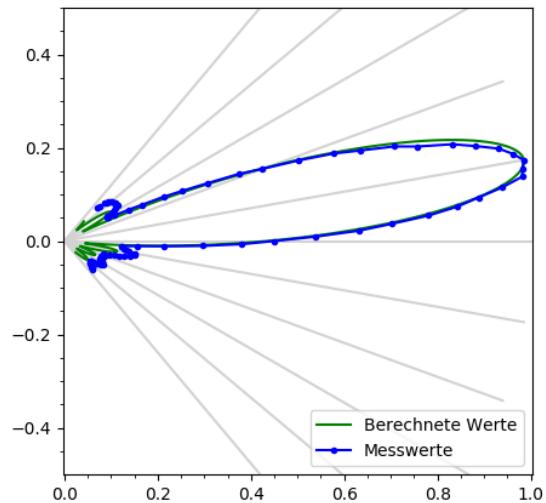
## 5.2 Messung der sendeseitigen Richtcharakteristik mit Amplitudenbelegung

Mit dem gleichen Messaufbau aus Kapitel 5.1 wird die vom Mikrocontroller erzeugte Amplitudenbelegung (siehe letzter Abschnitt im Kapitel 4.2.2) gemessen. Dabei wird jeweils auf das Maximum der Messreihe normiert. Die theoretisch erreichbare Richtcharakteristik mit Amplitudenbelegung wurde mithilfe der Formel (2.16) aus dem Kapitel 2.2.7 berechnet.

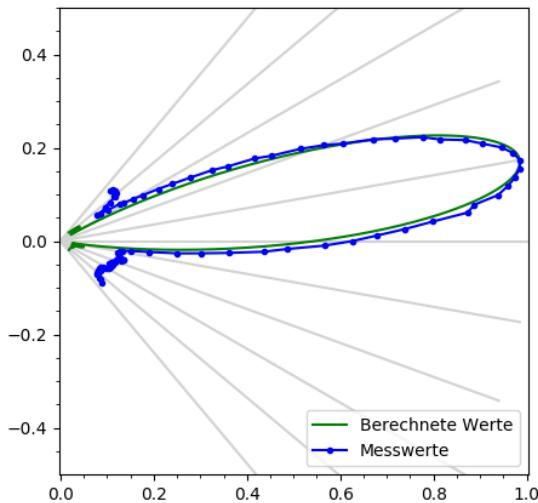
Die gemessenen Hauptkeulen mit Amplitudenbelegung in den Abbildungen 5.7, 5.8 und 5.9 sind breiter als die Hauptkeule in Abbildung 5.6 mit Rechteckbelegung, was der Theorie entspricht. Da die Messungen auf das Maximum normiert sind, die Sendeleistung jedoch insgesamt verkleinert wird, hebt sich der Rauschpegel. Die Nebenkeulen sind zwar gedämpft, jedoch nicht so stark wie berechnet. Dies könnte auf mechanische Kopplung zwischen den Sendern und/oder kleinere mechanische Ungenauigkeiten beim Auflöten der Ultraschalltransceiver zurückzuführen sein.



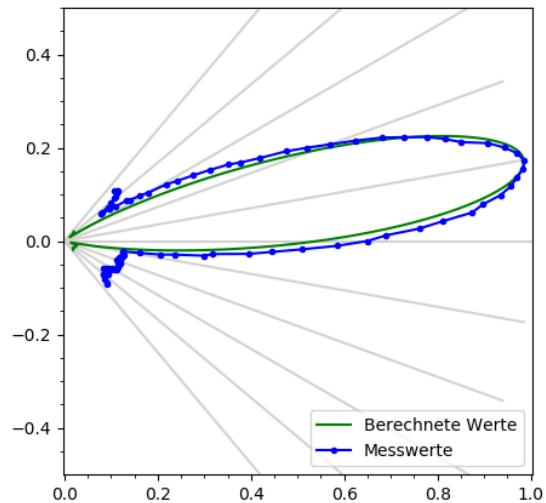
**Abbildung 5.6:** Richtcharakteristik des Sendearrays für eine Rechteckbelegung und  $\theta = 10$  Grad



**Abbildung 5.7:** Richtcharakteristik des Sendearrays für eine cos-Belegung und  $\theta = 10$  Grad



**Abbildung 5.8:** Richtcharakteristik des Sendearrays für eine  $\cos^2$ -Belegung und  $\theta = 10$  Grad



**Abbildung 5.9:** Richtcharakteristik des Sendearrays für eine Gaussbelegung und  $\theta = 10$  Grad

### 5.3 Messung der empfangsseitigen Richtcharakteristik

Werden die acht empfangenen Signale vor der Addition zeitlich korrekt verschoben, entsteht auch empfängerseitig eine Richtcharakteristik (siehe Kapitel 2.4.4). Für den Messaufbau wird das Ultraschall Phased Array auf einem Stativ in 2m Distanz vor einer gut reflektierenden Oberfläche (Wand) aufgestellt. Bodenreflexionen werden mit einem Schaumstoffstück verhindert.

Es wird daraufhin das Gerät auf einem Stativ so ausgerichtet, dass der Schall eines Sendevorgangs mit dem Sendewinkel  $\theta$  senkrecht auf der Wand auftrifft. Danach trifft der reflektierte Schall wieder mit dem gleichen Winkel auf das Array auf. Der Messaufbau ist in Abbildung 5.10 zu sehen. Nach einem Sendevorgang in diesem Winkel  $\theta$  werden die acht empfangenen Signale gespeichert. Um Störungen zu vermeiden werden die ersten Werte dieser Signale abgeschnitten. Danach werden die Signale für jeden empfangsseitigen Winkel zwischen  $\pm 90^\circ$  aufaddiert. Dies geschieht mithilfe der Funktion `upsamp_beam_form(...)` (siehe Kapitel 4.3.3). Aus den daraus resultierenden Wellenpaketen wird jeweils das Maximum ausgelesen. Werden diese Maxima zum zugehörigen Winkel aufgetragen, entsteht so die empfangsseitige Richtcharakteristik. Diese Messung wird nach korrekter Positionierung des Phased Arrays über das Python-Skript "testing.py" durchgeführt, dabei werden die Graphen automatisch erzeugt. Mit `python ./software/server_flask/testing.py --help` kann eine Bedienungsanleitung dieser Funktion aufgerufen werden. Die Messwerte sind jeweils dabei auf den maximalen Wert der Messreihe normiert.

Die empfangsseitige Amplitudenbelegung (siehe Kapitel 2.2.7) wird durch entsprechendes Gewichten der Signale vor der Addition erzielt.

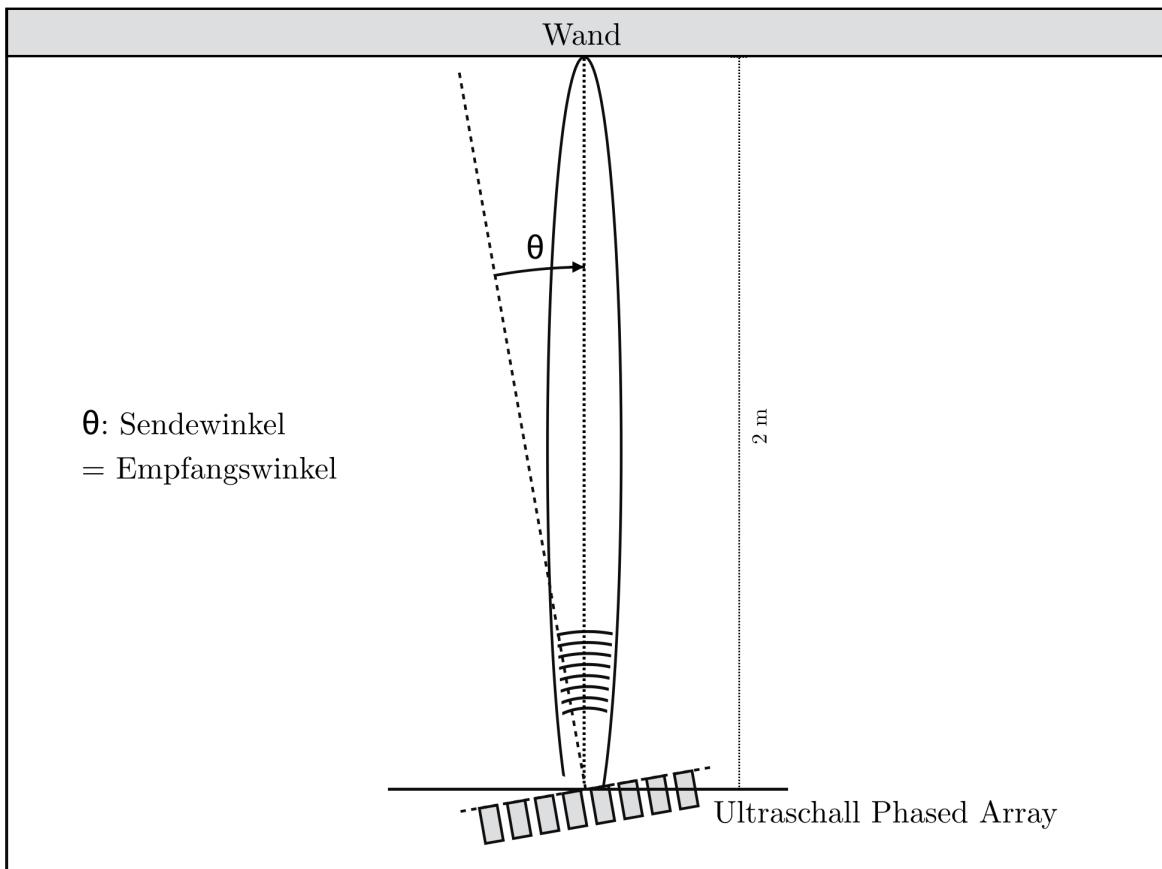
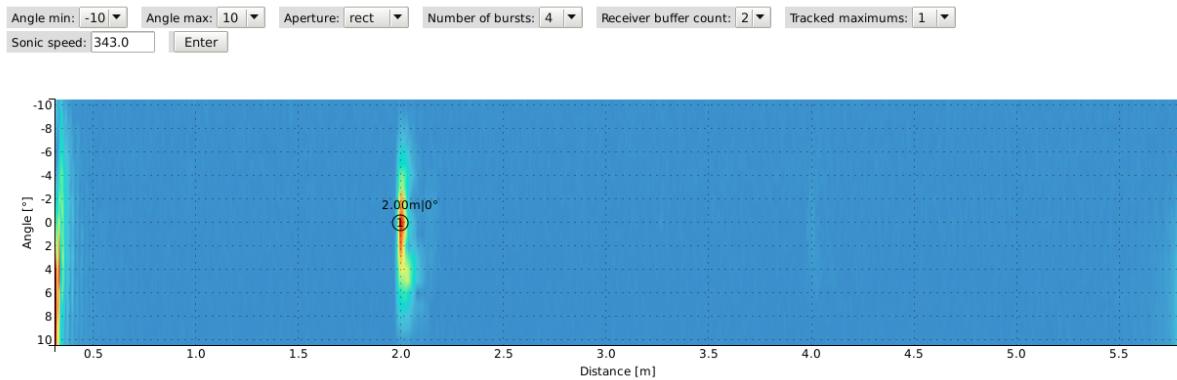


Abbildung 5.10: Messaufbau zur empfangsseitigen Richtcharakteristik

### 5.3.1 Messung der Empfangscharakteristik bei einem Sendewinkel von 0 Grad

Das Gerät wird mithilfe des Heatmap-Graphen im GUI in einer Distanz von 2m für einen Sendewinkel von  $\theta = 0^\circ$  senkrecht auf eine Wand ausgerichtet. Ein Umgebungsscan der Messsituation ist dabei in Abbildung 5.11 zu sehen. Es werden für die Messung vier Sendepulse ausgesandt.



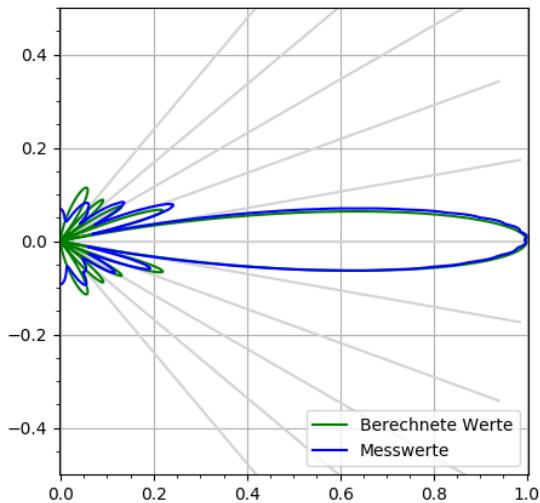
**Abbildung 5.11:** Maximum bei 2.00m und  $0^\circ$

Es entstehen dabei die Abbildungen 5.12, 5.13, 5.14 und 5.15 mit den entsprechenden Amplitudenbelegungen.

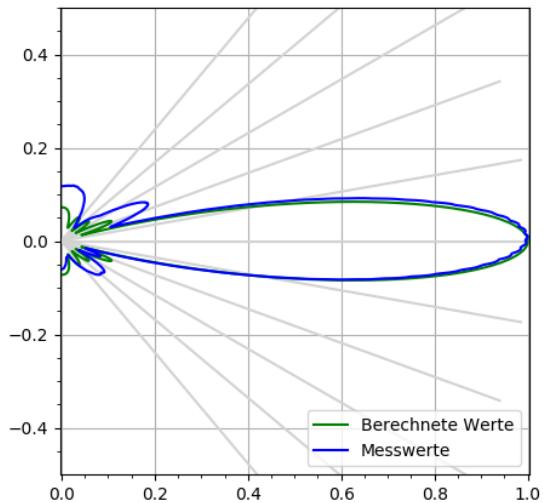
Die empfangsseitige Richtcharakteristik stimmt dabei insgesamt gut mit der Theorie überein. Für die Hauptkeulen ist die Übereinstimmung sogar sehr gut. Es fällt jedoch auf, dass für die Belegungen erneut grössere Nebenkeulen vorhanden sind (siehe Kapitel 5.2), als theoretisch berechnet. Mögliche Gründe sind wieder mechanische Kopplung und/oder kleinere mechanische Ungenauigkeiten beim Auflöten.

Bei den ersten Nebenkeulen ist erkennbar, dass die Linke jeweils grösser ist als die Rechte. Dieses asymmetrische Verhalten ist ausgeprägter bei den Messungen mit Amplitudenbelegung (Abbildungen 5.13, 5.14 und 5.15).

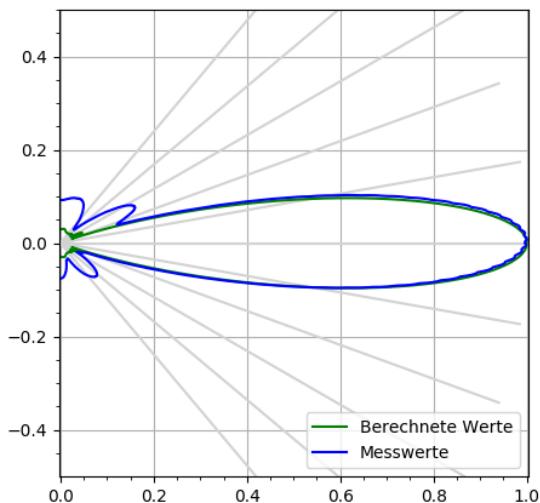
Desweiteren ist zu erkennen, dass das Ultraschall Phased Array zwar nicht mit genau  $0^\circ$  senden kann (siehe Kapitel 5.1), jedoch mit  $0^\circ$  empfangen kann. Im Gegensatz zur Verzögerung beim Einschalten der PWM-Kanäle kann der Fehler, der durch das ADC-Multiplexing entsteht (siehe Kapitel 2.3.2) im Nachhinein korrigiert werden.



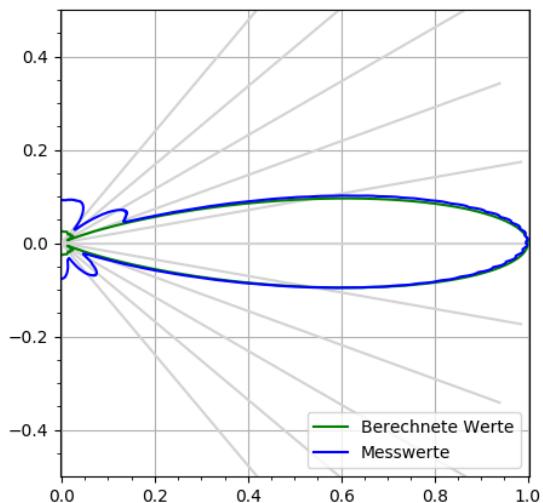
**Abbildung 5.12:** Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung



**Abbildung 5.13:** Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung



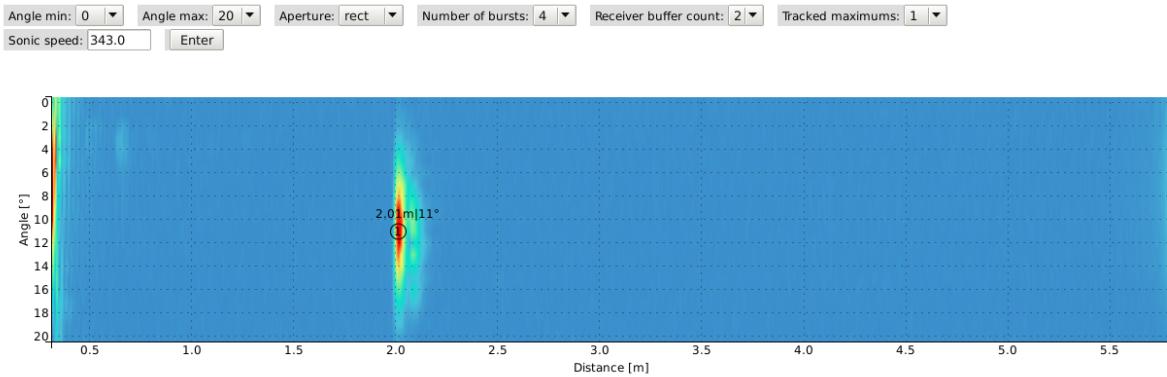
**Abbildung 5.14:** Richtcharakteristik des Empfangs-Arrays für eine  $\cos^2$ -förmige Amplitudenbelegung



**Abbildung 5.15:** Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbelegung

### 5.3.2 Messung der Empfangscharakteristik bei einem Sendewinkel von 10 Grad

Mithilfe des Heatmap-Graphen im GUI wird das Gerät in 2m Distanz für einen Sendewinkel von  $\theta = 10^\circ$  senkrecht auf eine Wand ausgerichtet. Die Messsituation ist dabei als Umgebungsscan in Abbildung 5.11 dargestellt. Zur Erzeugung des Sendesignals werden vier Sendepulse ausgesandt.

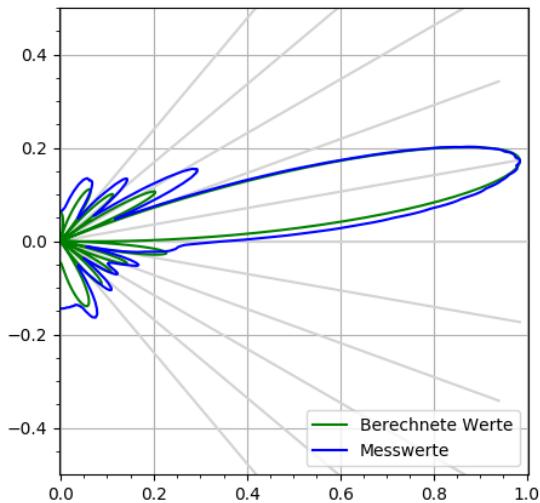


**Abbildung 5.16:** Maximum bei 2.01m und 11°

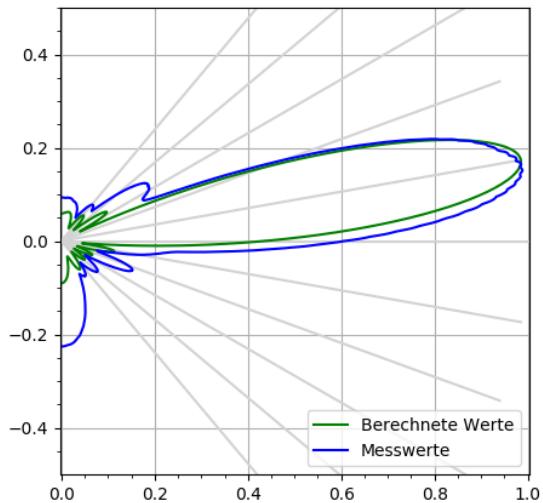
Dabei entstehen die Abbildungen 5.17, 5.18, 5.19 und 5.20 mit den entsprechenden Amplitudenbelegungen.

Theorie und Messungen stimmen insgesamt wieder gut überein. Die Hauptkeulen sind dabei etwas breiter als berechnet und weisen einen Winkeloffset von ein paar Grad in negativer Richtung auf.

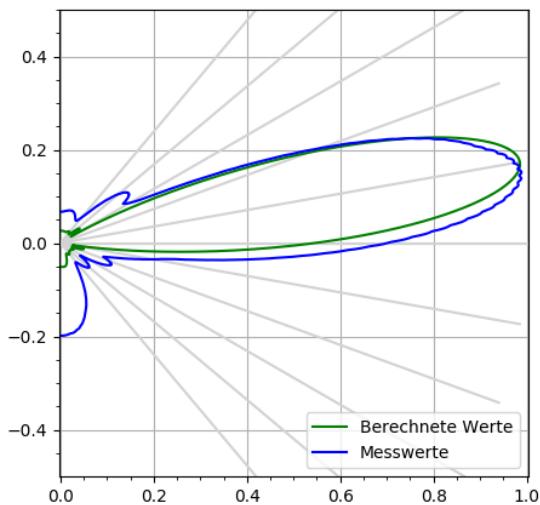
Wieder treten etwas grössere Nebenkeulen auf, als theoretisch berechnet. In Abbildung 5.19 ist erneut erkennbar, wie die erste Nebenkeule links etwas grösser ist als auf der rechten Seite.



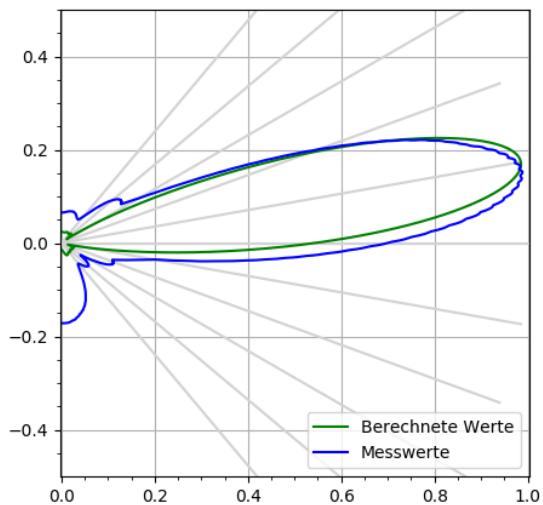
**Abbildung 5.17:** Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung



**Abbildung 5.18:** Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung



**Abbildung 5.19:** Richtcharakteristik des Empfangs-Arrays für eine  $\cos^2$ -förmige Amplitudenbelegung



**Abbildung 5.20:** Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbelegung

### 5.3.3 Messung der Empfangscharakteristik bei einem Sendewinkel von 20 Grad

Über den Heatmap-Graph im GUI wird das Gerät in 2m Distanz für einen Sendewinkel von  $\theta = 20^\circ$  senkrecht auf eine Wand ausgerichtet. Ein Umgebungsscan der Messsituation ist in Abbildung 5.11 dargestellt. Für die Messung werden vier Sendepulse ausgesandt.

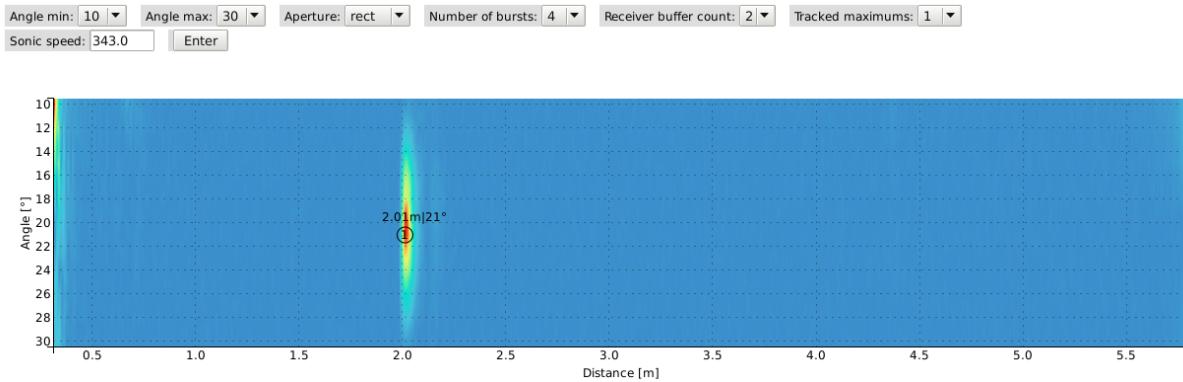
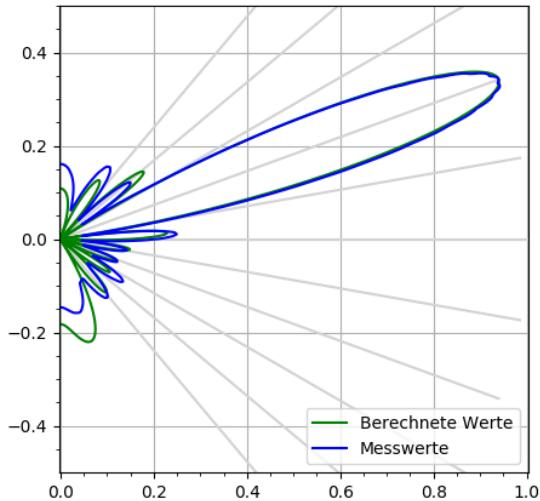


Abbildung 5.21: Maximum bei 2.01m und  $21^\circ$

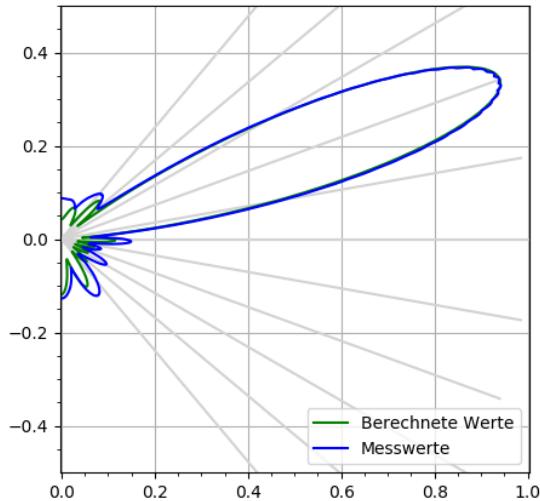
Das Ergebnis wird in den Abbildungen 5.22, 5.23, 5.24 und 5.25 mit den entsprechenden Amplitudenbelegungen dargestellt.

Erneut stimmen Theorie und Messungen gut überein. Die Hauptkeule ist nahezu identisch zum berechneten Verlauf.

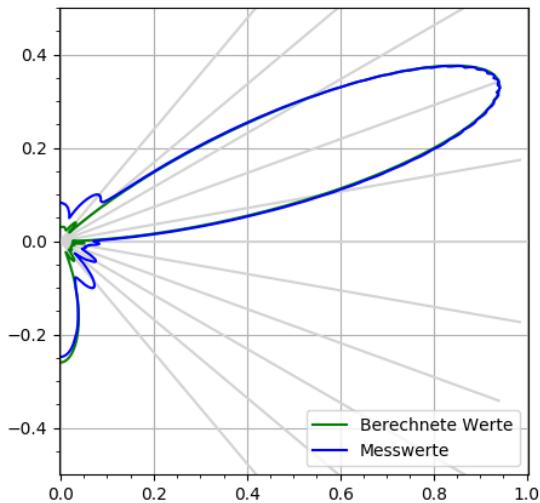
Wie bei den vorherigen Messreihen treten für die Belegungen grössere Nebenkeulen auf als berechnet. In den Abbildungen 5.24 und 5.25 ist dies gut erkennbar. Wie bereits erwähnt, könnten die grösseren Nebenkeulen auf mechanische Kopplung und/oder mechanische Ungenauigkeiten beim Auflöten zurückzuführen sein. Die Asymmetrie zwischen den links- und rechtsseitigen Nebenkeulen ist jedoch weniger ausgeprägt als in den vorherigen Messreihen.



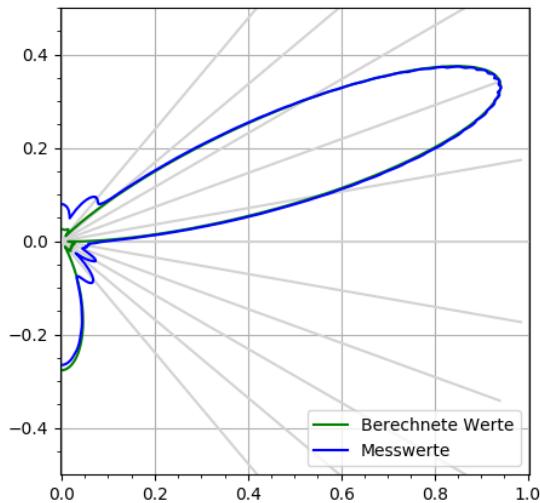
**Abbildung 5.22:** Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung



**Abbildung 5.23:** Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung



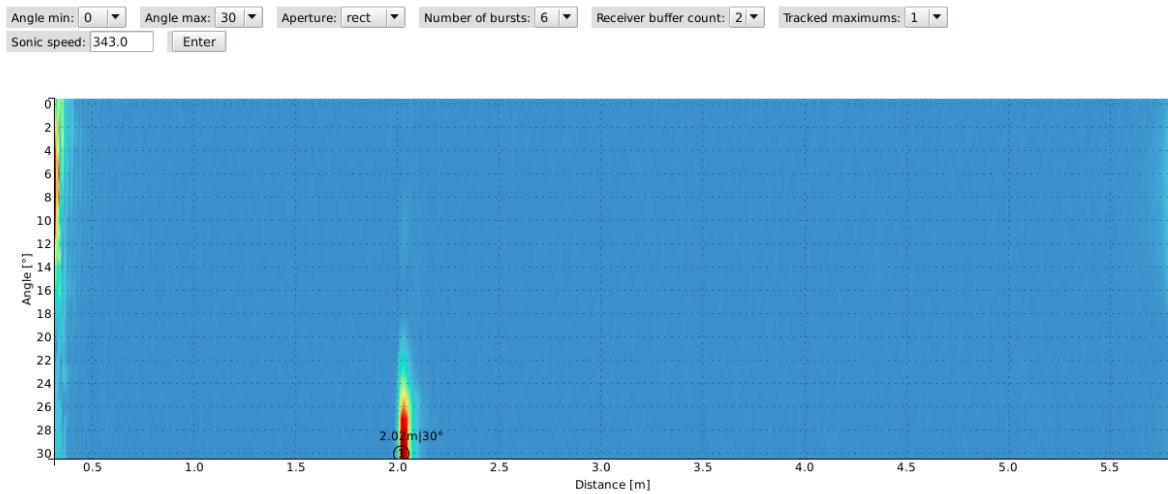
**Abbildung 5.24:** Richtcharakteristik des Empfangs-Arrays für eine  $\cos^2$ -förmige Amplitudenbelegung



**Abbildung 5.25:** Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbelegung

### 5.3.4 Messung der Empfangscharakteristik bei einem Sendewinkel von 30 Grad

Mithilfe des Heatmap-Graphs im GUI wird das Gerät in 2m Distanz für einen Sendewinkel von  $\theta = 30^\circ$  senkrecht auf eine Wand ausgerichtet. Die Messsituation ist dabei als Umgebungsscan in Abbildung 5.11 zu sehen. Das Sendesignal wird mit sechs Sendepulsen erzeugt.



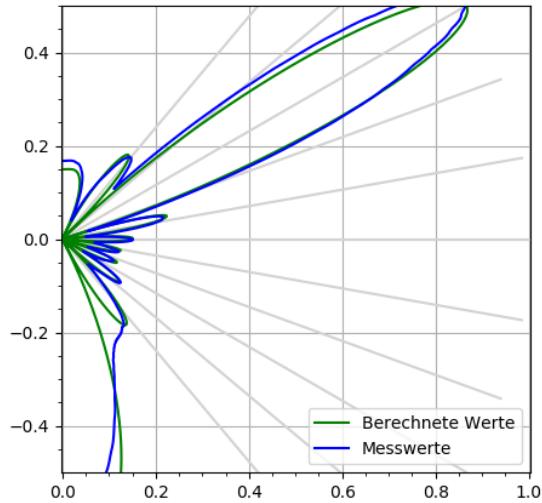
**Abbildung 5.26:** Maximum bei 2.02m und 30°

Als Ergebnis entstehen dabei die Abbildungen 5.27, 5.28, 5.29 und 5.30 mit den entsprechenden Amplitudenbelegungen.

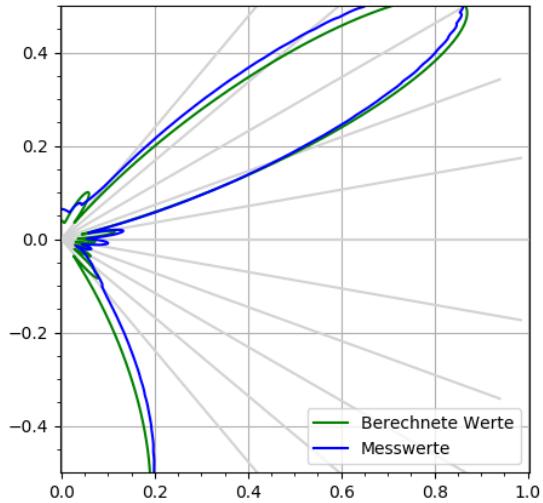
Theorie und Messungen stimmen insgesamt gut überein. Die Hauptkeule ist etwas breiter als berechnet und weist einen Winkeloffset von ein paar Grad in die positive Richtung auf.

Es fällt auf, dass im Gegensatz zu den vorherigen Messreihen für die Belegungen kaum grössere Nebenkeulen entstehen, als berechnet wurde. Dies ist in den Abbildungen 5.19 und 5.20 gut erkennbar.

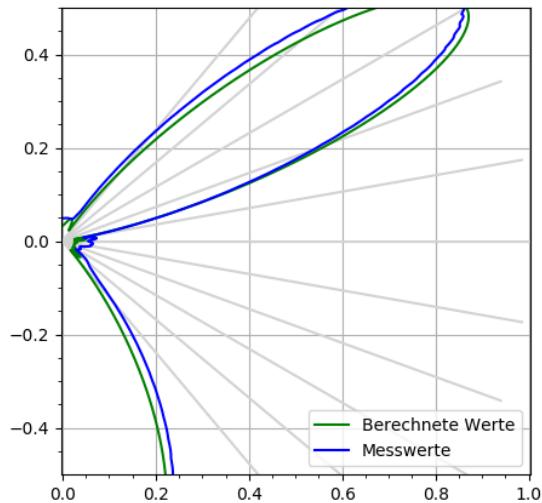
Interessant ist, wie in Abbildung 5.27 die grosse Nebenkeule rechts aussen kleiner, in den Abbildungen 5.28, 5.29 und 5.30 jedoch grösser ist als berechnet.



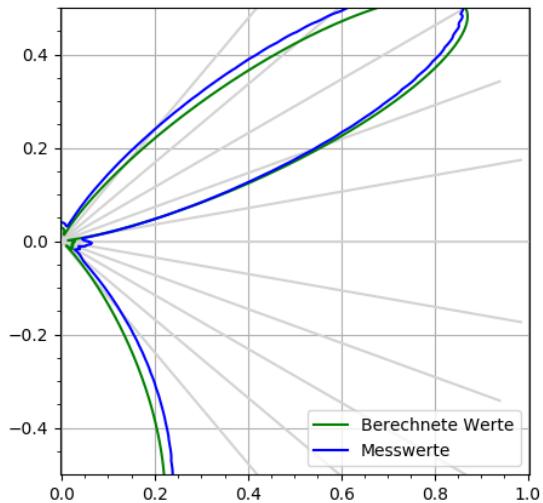
**Abbildung 5.27:** Richtcharakteristik des Empfangs-Arrays für eine rechteckige Amplitudenbelegung



**Abbildung 5.28:** Richtcharakteristik des Empfangs-Arrays für eine cos-förmige Amplitudenbelegung



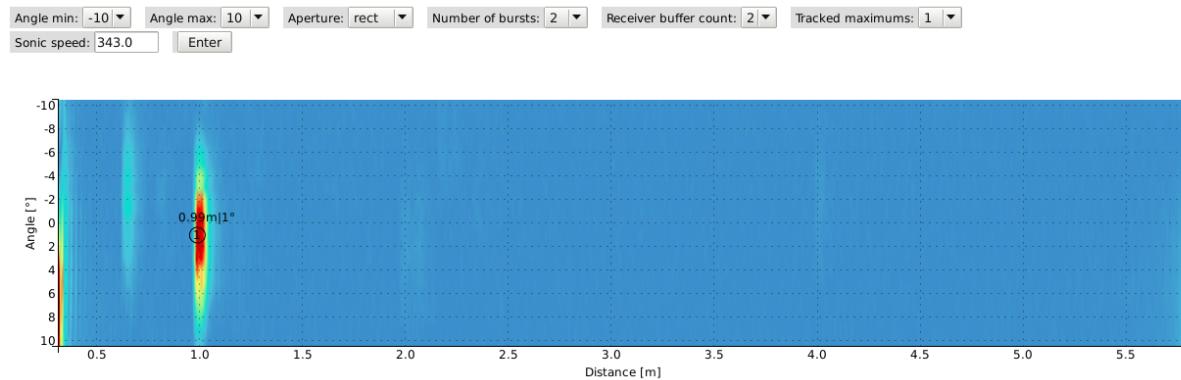
**Abbildung 5.29:** Richtcharakteristik des Empfangs-Arrays für eine  $\cos^2$ -förmige Amplitudenbelegung



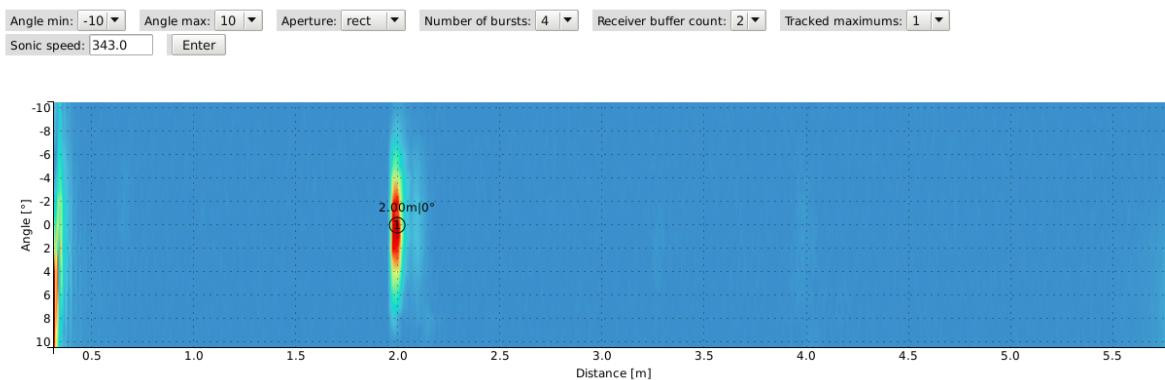
**Abbildung 5.30:** Richtcharakteristik des Empfangs-Arrays für eine gaussförmige Amplitudenbelegung

## 5.4 Distanzmessungen

Für die Messung der Distanzauflösung wird das Gerät frontal vor einer glatten Wand aufgestellt und die Distanz vom Ultraschall Phased Array zur Wand gemessen. Daraufhin wird dieser Referenzwert verglichen mit dem Distanzwert aus dem GUI, welcher mithilfe der Maximum-Tracking Funktion abgelesen wird. Die dadurch entstehen Umgebungscans sind jeweils dargestellt. Störende Boden- und Deckenreflexionen werden mit einem Schaumstoffstück verhindert. Für grössere Distanzen wird aufgrund der Dämpfung die Anzahl Sendepulse erhöht. Für jede Messung ist die Anzahl Pulse vermerkt.



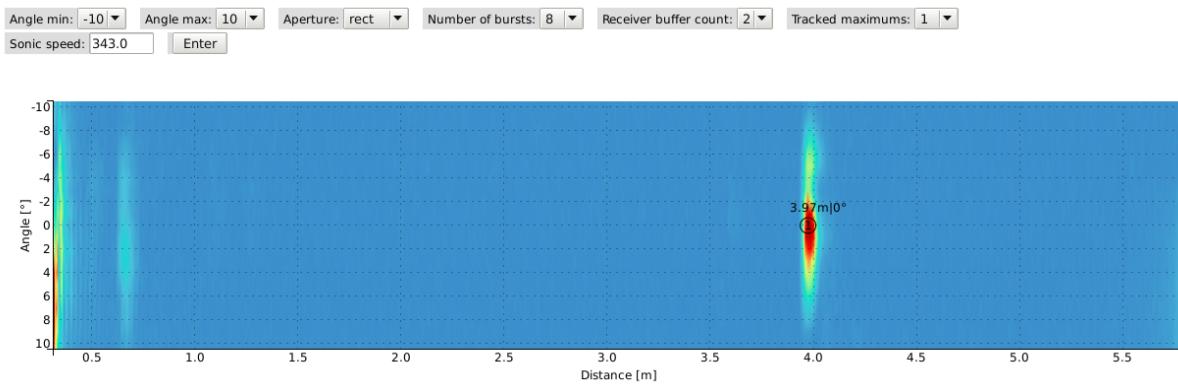
**Abbildung 5.31:** 1m: Maximum bei 0.99m und 1°



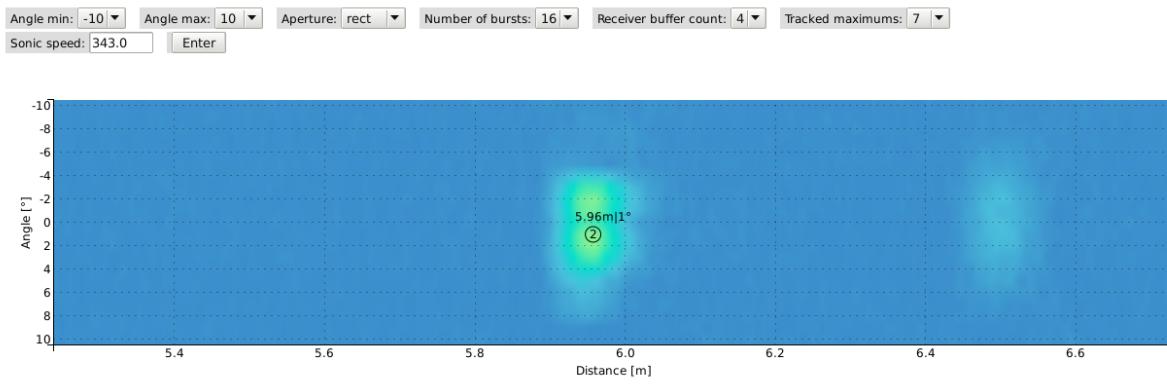
**Abbildung 5.32:** 2m: Maximum bei 2.00m und 0°

Für eine Distanz von 1.00m wird die Messung mit zwei Sendepulsen durchgeführt. Die Messabweichung zwischen dem gemessenen Referenzwert und dem im GUI abgelesenen Wert beträgt –1cm.

Bei einer Distanz von 2.00m wird die Messung mit vier Sendepulsen durchgeführt. Die Maximum-Tracking Funktion im GUI zeigt dabei genau 2.00m an.



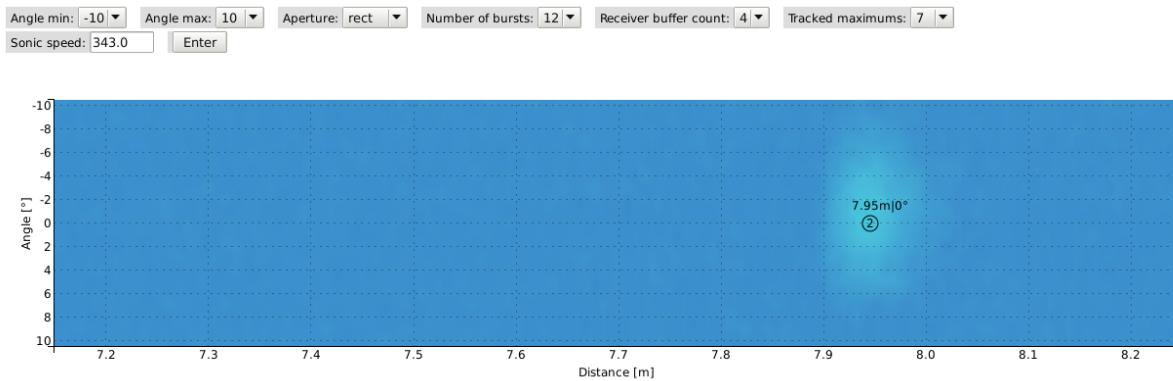
**Abbildung 5.33:** 4m: Maximum bei 3.97m und 0°



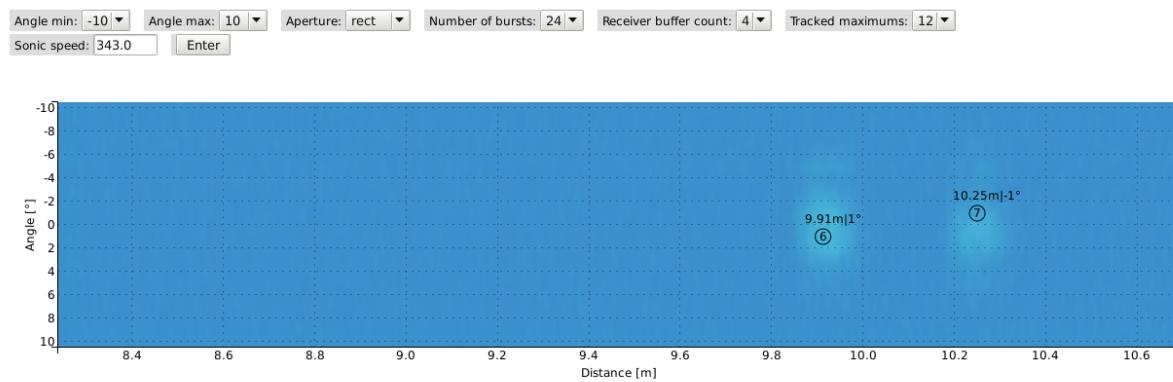
**Abbildung 5.34:** 6m: Maximum bei 5.96m und 1°

Bei der Messung für den Referenzwert von 4.00m wird die Anzahl Sendepulse auf acht erhöht. Die Messabweichung beträgt dabei -3cm.

Für eine Distanz von 6.00m ist trotz einer Erhöhung der Anzahl Sendepulse auf 16 ein Abfall in der gemessenen Amplitude des Echos festzustellen. Die Messabweichung zwischen dem von der Maximum-Tracking Funktion gefunden Wert und der Referenzmessung beträgt -4cm. Ein zweites, störendes Maximum ist in der Abbildung 5.34 bei einer Distanz von ungefähr 6.5m zu erkennen.



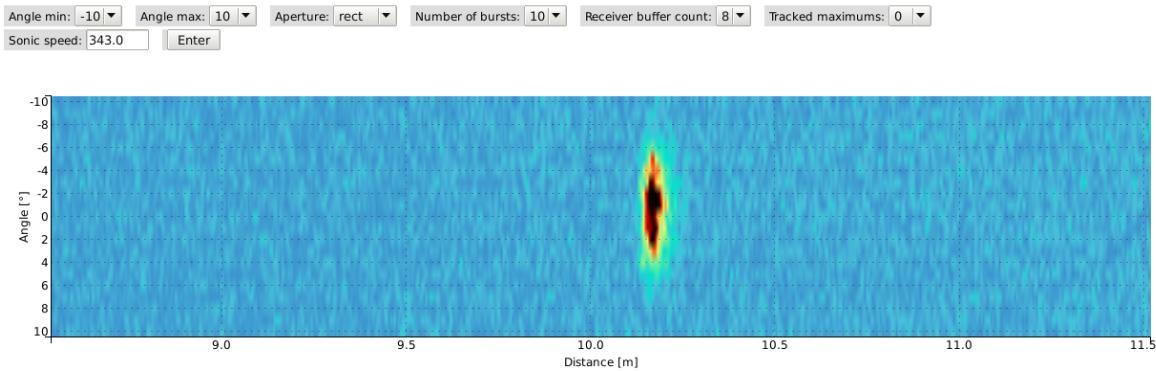
**Abbildung 5.35:** 8m: Maximum bei 7.95m und  $0^\circ$



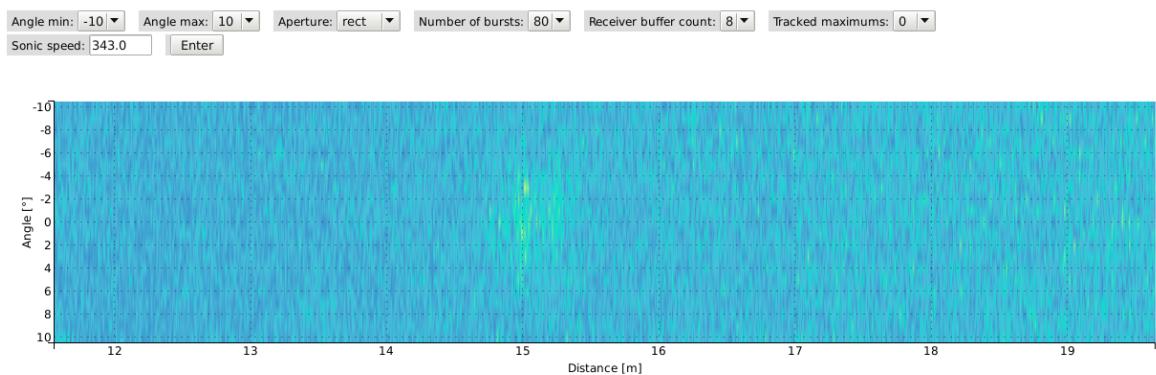
**Abbildung 5.36:** 10m: Maxima bei 9.91m und  $1^\circ$  und bei 10.25m und  $-1^\circ$

Für grosse Distanzen werden vermehrt störende Echos von anderen Objekten im Raum empfangen. Die Amplitude des zu messenden Echos wird bei einer Distanz 8.00m von der Maximum-Tracking Funktion nur noch als zweitgrößter Peak erkannt. Die Messabweichung beträgt dabei  $-5\text{cm}$ . Es ist zu erwähnen, dass das Ultraschall Phased Array für derart grosse Distanzen genau auf die Wand ausgerichtet sein muss, ansonsten ist das von der Wand reflektierte Echo zu schwach.

Ab einer Distanz von 8m wird die Distanzmessung problematisch. So sind im Bereich von  $\pm 30\text{cm}$  um die gemessene Referenzdistanz von 10.00m zwei Maxima vorhanden und es ist nicht klar welches von der Wand stammt.



**Abbildung 5.37:** Messung für 10m mit skalierten Werten



**Abbildung 5.38:** Messung für 15m mit skalierten Werten

Werden die Werte des dargestellten Signals linear mit der Distanz skaliert, so werden Werte stärker erhöht, die aufgrund der Dämpfung abgeschwächt sind. Dies ist zwar mathematisch nicht korrekt und nur ein erster Versuch zur Weiterentwicklung des Projektes (siehe Weiterentwicklung im Kapitel 6), der positive Effekt auf die Sichtbarkeit von weit entfernten Objekten soll hier aber trotzdem kurz aufgezeigt werden. Dieses skalierte Signal ist in Abbildung 5.37 dargestellt.

Aufgrund des mit der Skalierung angehobenen Rauschpegels geht das Echo ab einer Distanz von 15m fast im Rauschen unter. Dies ist in Abbildung 5.38 zu sehen.

## 5.5 Objekterkennung anhand ausgewählter Beispiele

Im Folgenden sind ausgewählte Beispiele von Umgebungsscans aufgeführt und kommentiert. Weitere Bilder von Umgebungsscans sind im Anhang C zu finden.

Das zur Dämpfung unerwünschter Reflexionen verwendete Schaumstoffstück (für die Messungen im Kapitel 5.3 und 5.4) wird als Testobjekt im reflexionsarmen Halbraum vermessen. Das Schaumstoffstück ist 40cm breit und wird in einer Entfernung von 1.5m vor dem Array aufgestellt. Bei einem Winkel von  $20^\circ$  und frontaler Ausrichtung auf das Array entsteht mit 20 Sendepulsen die Abbildung 5.39. Der vom Schaumstoffstück eingenommene Winkel von ca.  $15^\circ$  ist auf dem Bild erstaunlich gut erkennbar. Die Mehrfachechos in der Nähe des Arrays stammen von einem Metallhebel am Stativ.

Wird das Schaumstoffstück in einem Winkel von  $0^\circ$  frontal vor dem Array aufgestellt, entsteht das Bild 5.40. Der Sprung um  $0^\circ$  ist auf die minimale Verzögerung beim Einschalten der PWM-Kanäle zurückzuführen (ebenfalls zu sehen in Abbildung 5.2).

Für die Abbildung 5.41 hat sich eine Person im reflexionsarmen Halbraum vor das Array gestellt und für die Abbildung 5.42 sind es zwei Personen im Freien.

Die Messung mit dem Schaumstoffstück in einem Winkel von  $20^\circ$  (Abbildung 5.39) wird wiederholt, jedoch wird dieses nicht dem Array zugewandt, sondern parallel zum Array aufgestellt. Dadurch entsteht die Abbildung 5.43. Es ist eine Problematik bei der Objekterkennung mittels gerichtetem Schallkegel zu sehen. Schall von Objekten, welche nicht dem Array zugewandt sind, wird nicht zwingend in die Richtung des Arrays zurückgeworfen (siehe Kapitel 2.2.4).

Zum Abschluss ist ein Bild mit einem Mikrophonstativ als Test-Objekt zu sehen. Es handelt sich dabei um eine Metallstange mit ca. 3cm Durchmesser im Abstand von 2.00m und in einem Winkel von  $20^\circ$ . Die Messung wird mit 20 Sendepulsen durchgeführt. Dabei entsteht die Abbildung 5.44. Im Hintergrund ist die offene Tür zum reflexionsarmen Halbraum erkennbar.

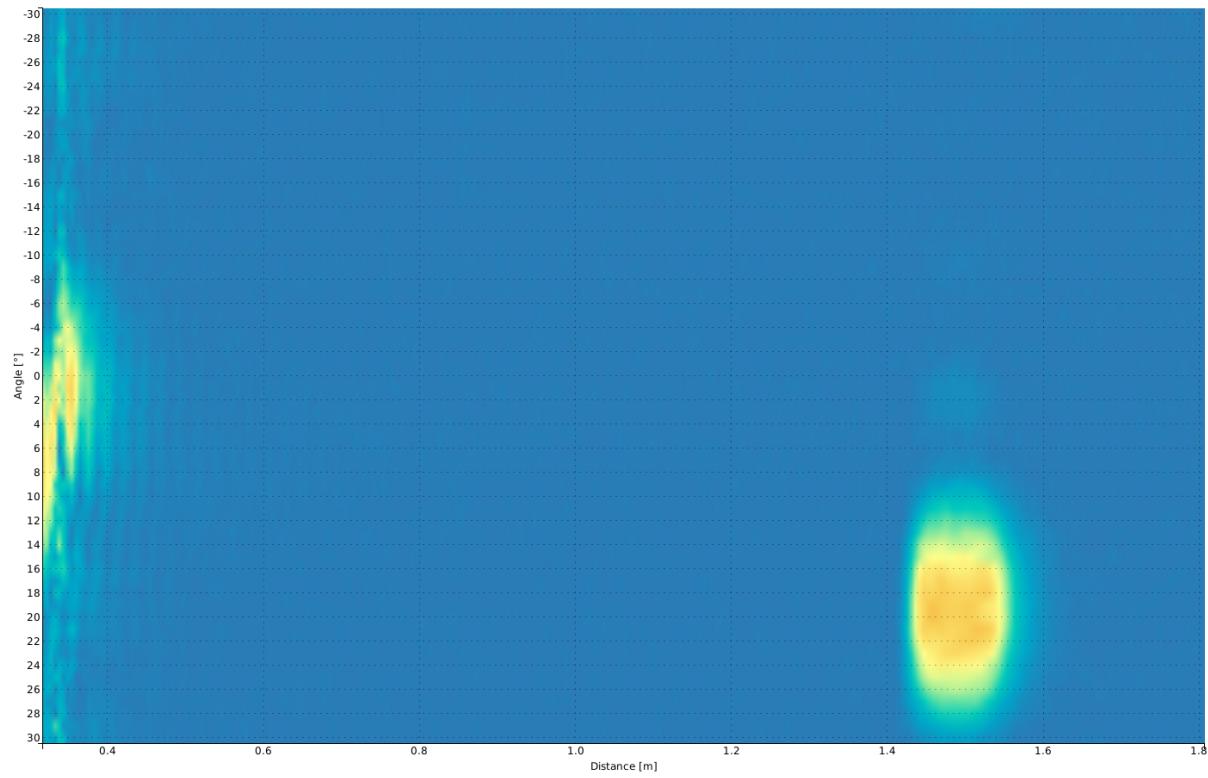


Abbildung 5.39: Schaumstoffstück bei 1.5m und 20°, 20 Sendepulse

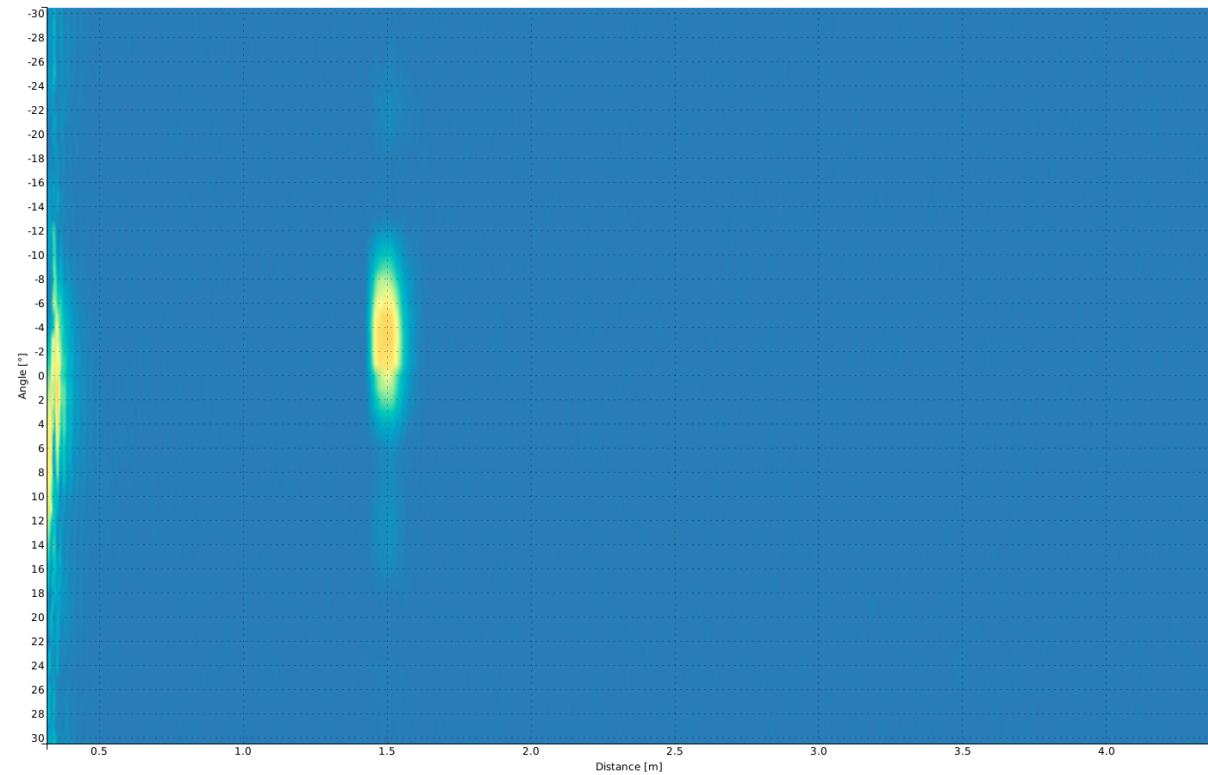


Abbildung 5.40: Schaumstoffstück bei 1.5m und 0°, 20 Sendepulse

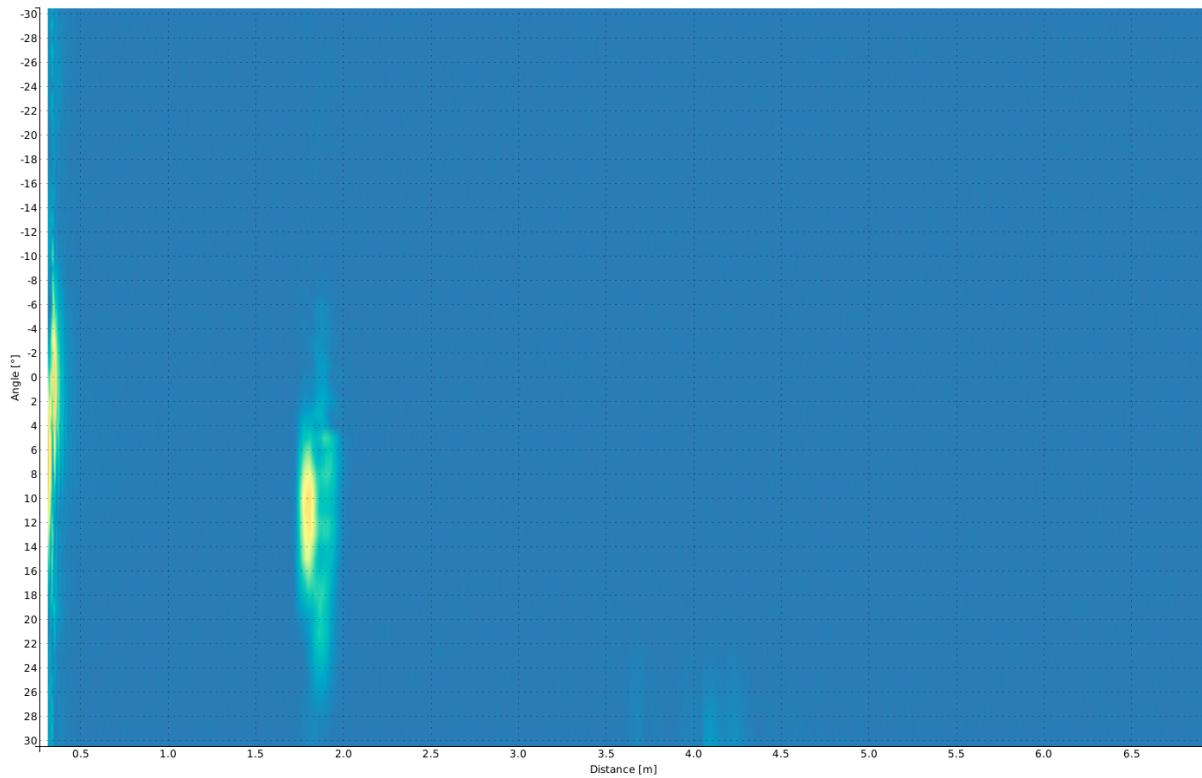


Abbildung 5.41: Person im reflexionsarmen Halbraum

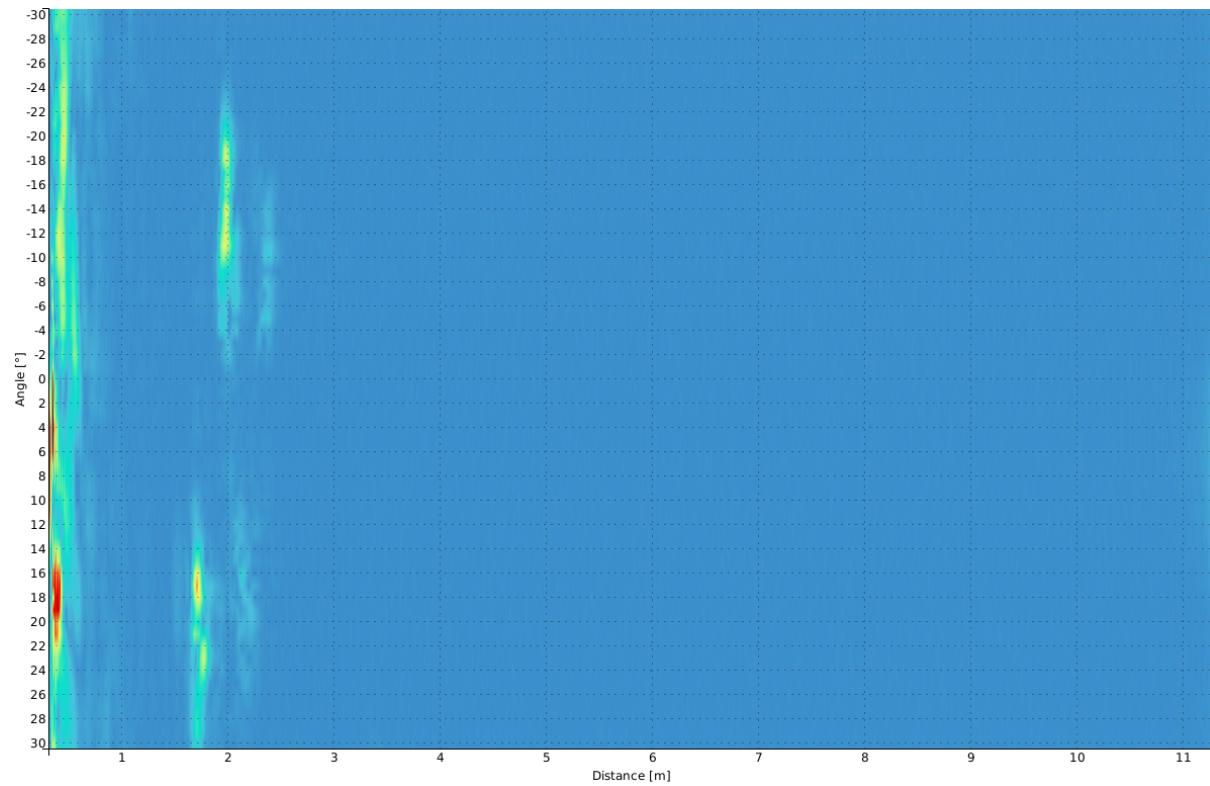
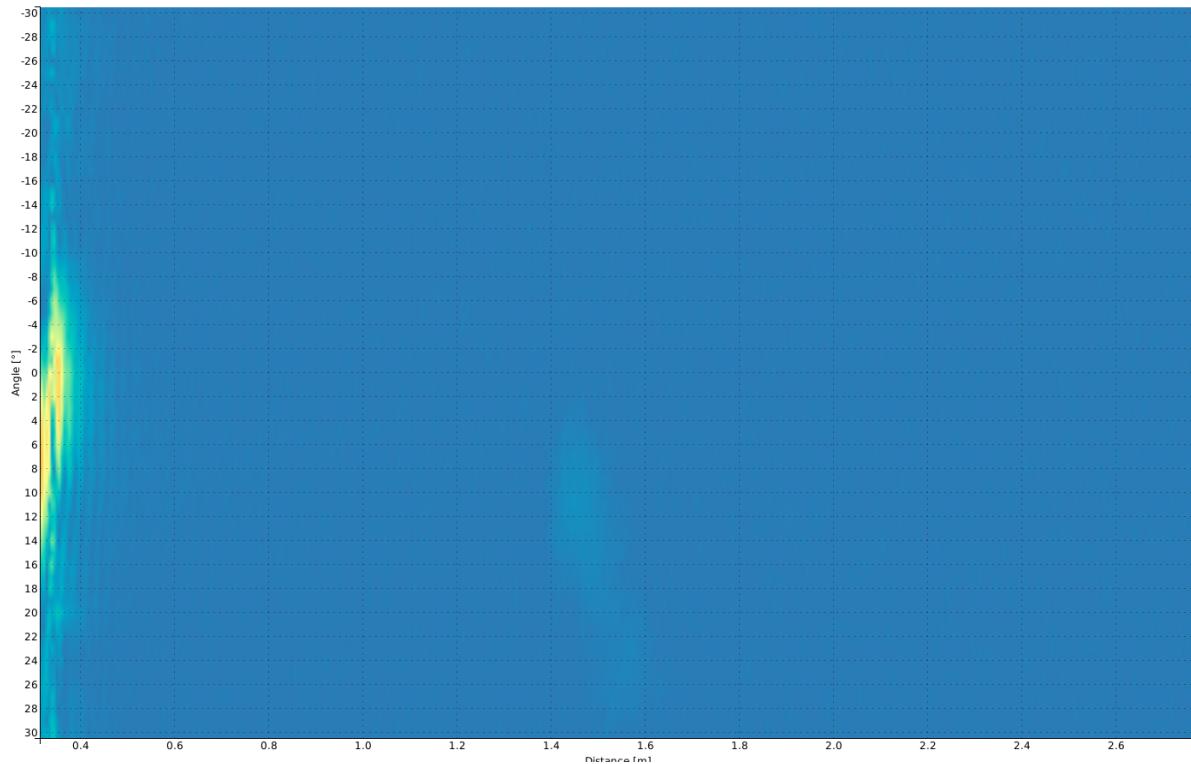
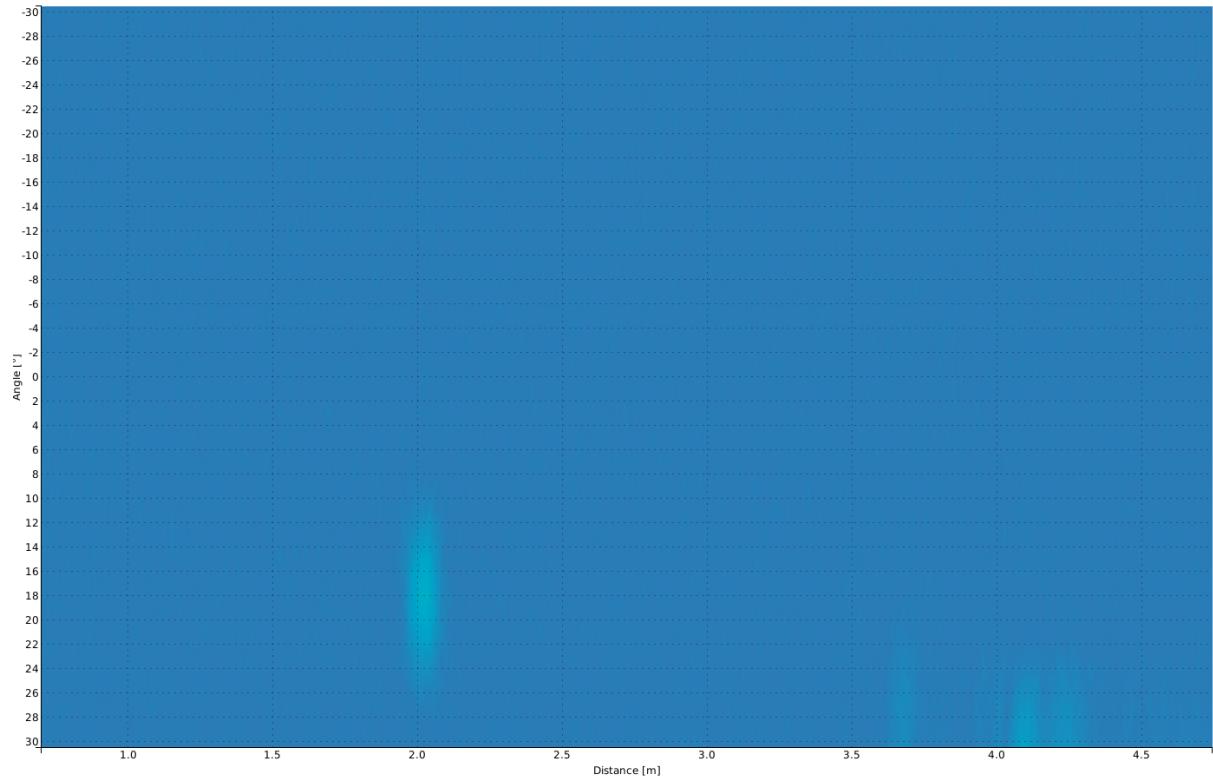


Abbildung 5.42: 2 Personen im Freien



**Abbildung 5.43:** Schaumstoffstück parallel zum Array ausgerichtet



**Abbildung 5.44:** Mikrofonständer im reflexionsarmen Halbraum mit offener Tür

## 6 Schlusswort

### Ergebnisse

Im Rahmen dieses Projektes wurde ein Ultraschall Phased Array von der Idee her bis zum funktionierenden System entwickelt. Es erfüllt sämtliche Anforderungen aus dem Pflichtenheft (siehe Anhang F). Die maximale Messdistanz, bei der die Distanzabweichung weniger als  $\pm 10\text{cm}$  beträgt, ist gemäss den Messungen in Kapitel 5.4 grösser als 5m. Damit sind die Anforderungen bezüglich der maximalen Distanz und der Tiefenauflösung erfüllt.

Die Verkürzung der Abklingzeit und die Reduktion des akustischen Übersprechens wurden durch ein Redesign der Hardware aus dem Projekt 5 erreicht. Dabei wurde die Schaltung so abgeändert, dass zum Senden und Empfangen dieselben Ultraschalltransceiver verwendet werden können. Die Firmware wurde einerseits an die neu entwickelte Hardware angepasst und andererseits so abgeändert, dass sie Steuerbefehle vom Host entgegennimmt, auf deren Basis entsprechende Messvorgänge ausführt werden. Nebenkeulen können optional mithilfe einer Amplitudenbelegung unterdrückt werden, was die Messungen im Kapitel 5.2 aufzeigen. Die Software abseits vom Arduino DUE ist als plattformunabhängige Webapplikation implementiert. Über ein GUI wird das Phased Array System bedient und die Messresultate werden fortlaufend dargestellt. Die Signalverarbeitung ist in der Programmiersprache Python implementiert und schnell genug, sodass keine Verzögerungen entstehen.

### Schwierigkeiten

Objekte mit stark reflektierender Oberfläche, welche nicht auf das Array gerichtet sind, werfen kaum Schall in die Richtung des Arrays zurück. Sie können deshalb schlecht detektiert werden (siehe Abbildung 5.43).

Die vom Mikrocontroller verursachten Verzögerungen beim Anschalten der PWM-Kanäle verhindern das Senden mit einem Winkel von unter  $2^\circ$ .

Die Anzahl verfügbarer ADC- und PWM-Kanäle ist vom Arduino DUE limitiert (siehe Kapitel 3.4). Mehr als acht PWM- und zwölf ADC-Kanäle sind nicht möglich.

### Weiterentwicklung

Ein externes Sendemodul zur Positionsbestimmung (siehe Wunschziele im Anhang F) wurde parallel zum Projekt entwickelt. Ein Hardwareprototyp inkl. zugehöriger Software ist fertiggestellt. Die nötigen Veränderungen, damit das Sendemodul mit der aktuellen Software zusammenarbeiten kann, sind grösstenteils umgesetzt. Im GUI könnten die Positionsdaten bereits dargestellt werden. Leider hat es das externe Sendemodul aus zeitlichen Gründen nicht in diesen Bericht geschafft.

Wenn die Sendepulse nicht mit einem Mikrocontroller, sondern mit einem FPGA generiert würden, könnte die Verzögerung beim Anschalten der PWM-Kanäle umgangen werden. Ein Senden mit  $0^\circ$  wäre dann möglich.

Für die dreidimensionale Steuerung des Schallkegels sind mehr ADC- und PWM-Kanäle nötig. Eine Portierung der Software (Firmware) auf ein leistungsfähigeres System wäre dann nötig.

Die mechanische Kopplung zwischen den Modulen könnte weiter untersucht werden (z.B. durch Fitzen eines theoretischen Modells an die Messungen der Richtcharakteristik).

Signalwerte von entfernten Echos könnten invers zum Verlauf der Dämpfung (siehe Kapitel 2.2.3) skaliert werden. In einem Versuch wurde dies, wenn auch mathematisch nicht korrekt, bereits ausprobiert (siehe Abbildungen 5.37 und 5.38).

Eine Darstellung der Differenz zwischen den Scanvorgängen würde interessante Informationen zu Veränderungen in einer Umgebung aufzeigen. Spannend wäre zudem auch eine Betrachtung der Ableitungen des darzustellenden Signals.

## Danksagung

In erster Linie möchten wir uns bei unseren Projektbetreuern Herrn Matthias Meier und Herrn Prof. Dr. Richard Gut für Ihre grossartige Unterstützung während der vergangenen zwei Projekten bedanken. Mit ihren guten Ideen, der konstruktiven Kritik, dem Fachwissen und der unermüdlichen Hilfe wenn irgendwo Probleme auftraten haben sie massgeblich zum erfolgreichen und zufriedenstellenden Abschluss der Bachelorarbeit beigetragen.

Auch bei Frau Anita Gertiser möchten wir uns für das ausdauernde Korrekturlesen und die Unterstützung beim Strukturieren des Fachberichtes ganz herzlich bedanken.

Ebenfalls bei Herrn Christoph Biel möchten wir uns bedanken. Material- und PCB-Bestellungen wurden von ihm jeweils innert rekordverdächtig kurzer Zeit bearbeitet. Auch bei allen anderen Anliegen bezüglich Messgeräten und Bauteilen hat er uns stets ausgeholfen und mit seiner positiven Einstellung zu einem guten Arbeitsklima beigetragen.

Besonders möchten wir uns auch bei Herrn Christoph Mijnssen für die interessierte Teilnahme bedanken. Mit seinem Fachwissen in Gebieten der Akustik hat er uns stets unterstützt und auf neue Ideen gebracht. Auch spontane Bauteilbestellungen konnten wir dank ihm jederzeit erledigen. Ebenfalls möchten wir uns für das geduldige Gegenlesen des Fachberichtes bedanken.

Zudem möchten wir uns bei Herrn Prof. Dr. Urs Bopp und bei Herrn Peter Hug bedanken für die freundliche Erlaubnis, den reflexionsarmen Halbraum an der FHNW für Messungen benutzen zu können.

## 7 Bibliographie

### Literatur

- [1] R. M. Ekbert Hering and M. Stohrer, *Physik für Ingenieure*. Berlin Heidelberg: Springer, 2008.
- [2] F. Kohlrausch, *Praktische Physik Band 1*. Stuttgart: B. G. Teubner, 1996.
- [3] H. Looser, *Physiklabor W12 Ultraschall*. Brugg AG: FHNW, 2015.
- [4] F. Kohlrausch, *Praktische Physik Band 3 Tabellen und Diagramme*. Stuttgart: B. G. Teubner, 1986.
- [5] S.-C. Wooh and Y. Shi, *Optimization of ultrasonic phased arrays*. Cambridge, Massachusetts, United States: Massachusetts Institute of Technology, Department of Civil and Environmental Engineering, 2008.
- [6] E. Skudrzyk, *Die Grundlagen der Akustik*. Wien: Springer, 1954.
- [7] J. D. Kraus and K. R. Carver, *Electromagnetics – Second Edition*. Tokyo: Kosaido Printing Co, 1981.
- [8] H. J. Visser, *Array and Phased Array Antenna Basics*. West Sussex: Wiley, 2005.
- [9] I. Rennert and B. Bundschuh, *Signale und Systeme*. München: Carl Hanser Verlag, 2013.
- [10] M. Meyer, *Grundlagen der Informationstechnik*. Wiesbaden: Vieweg, 2012.

## A Anhang Schema

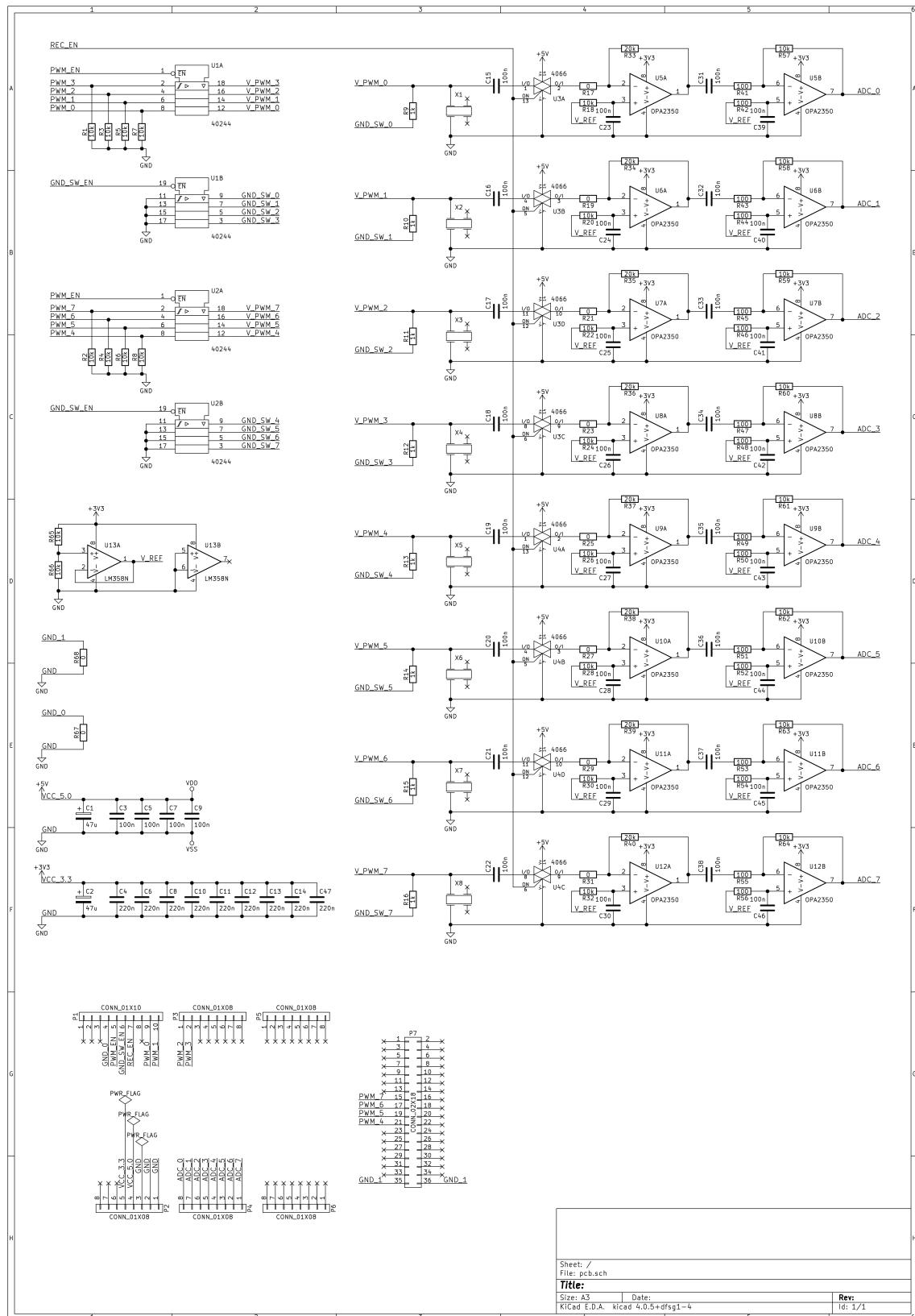


Abbildung A.1: Schema

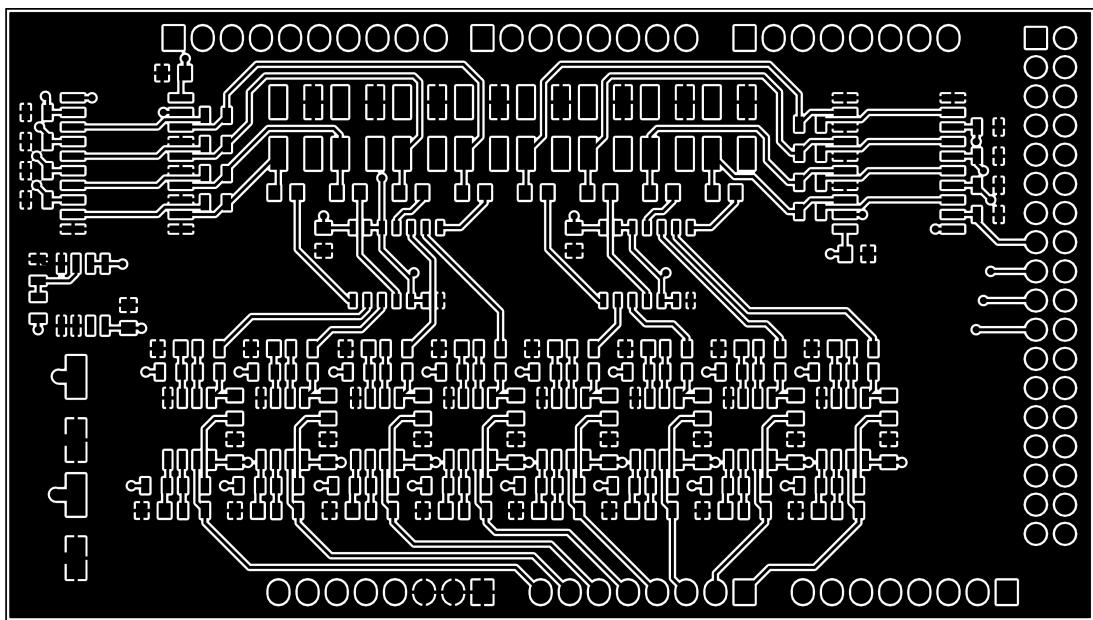


Abbildung A.2: Top Copper (signal)

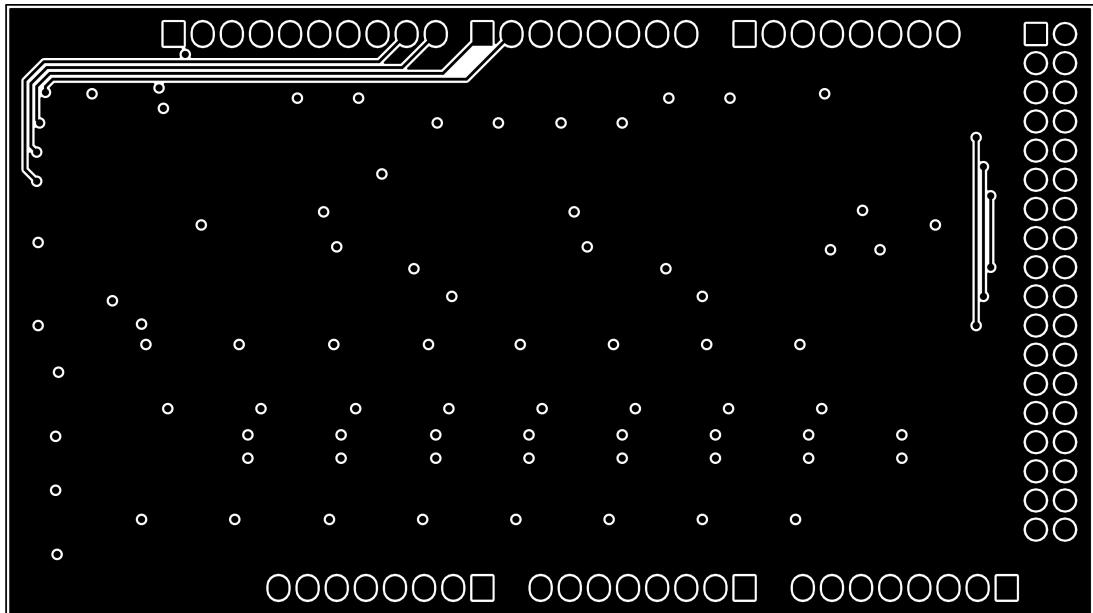


Abbildung A.3: Inner Layer 1 (virtual ground 1.65V)

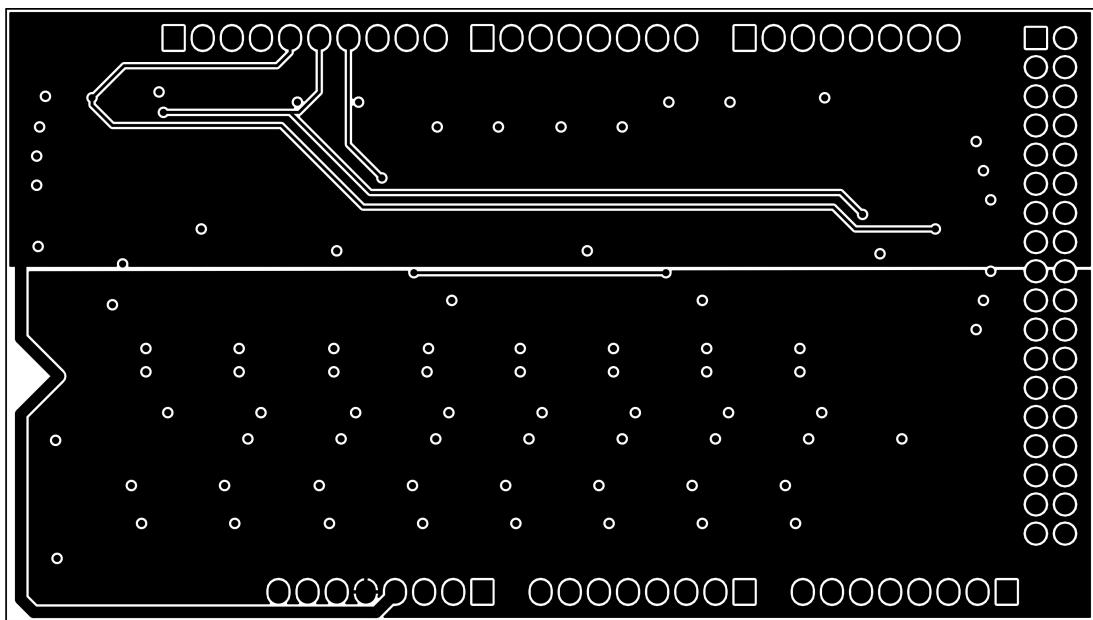


Abbildung A.4: Inner Layer 2 (VCC 5.0V, VCC 3.3V)

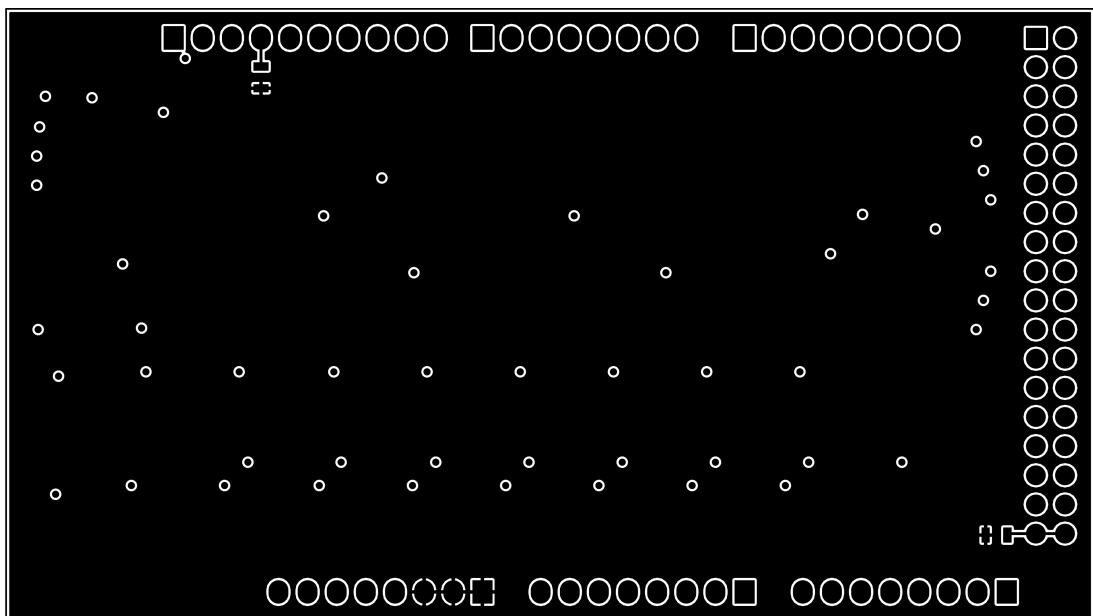


Abbildung A.5: Bottom Copper (ground)

## B Anhang Kosten

Nachfolgend findet sich die Auflistung der Bauteilkosten für ein Gerät, exklusive der Herstellungskosten für den Hardware Print. Die Kosten für vier Prints beliefen sich auf ca. 195 Euro.

Bauteil	Bestellnummer Mouser	Stk.	Stückpreis [CHF]	Preis [CHF]
Pin Header	538-22-28-4163	6	0.724	4.34
Kondensator 0.1 $\mu$ F	81-GRM40X104K50L	36	0.073	2.63
Kondensator 0.22 $\mu$ F	81-GRM40X7R224K050AL	9	0.255	2.30
Widerstand 0 $\Omega$	667-ERJ-U060R00V	10	0.096	0.96
Widerstand 100 $\Omega$	667-ERJ-U06F1000V	16	0.110	1.76
Widerstand 1k $\Omega$	667-ERJ-U06F1001V	8	0.275	2.20
Widerstand 10k $\Omega$	667-ERJ-U06F1002V	26	0.110	2.86
Widerstand 20k $\Omega$	667-ERJ-U06J203V	8	1.050	8.40
Levelshifter	512-74ACT244SCX	2	0.642	1.28
Anologschalter	595-SN74HC4066DR	2	0.437	0.87
OpAmp LM358	926-LM358MX/NOPB	1	0.774	0.77
OpAmp OPA2350	595-OPA2350UA/2K5	9	4.310	34.48
Ultraschalltransceiver	81-MA40H1S	8	7.780	62.24
<b>Total</b>				124.82

## C Anhang Objekterkennung

Nachfolgend finden sich noch einige aufgenommene Bilder.

Die Abbildung C.1 zeigt eine parallel zum Phased Array stehende Styroporplatte mit einer Breite von 0.5m. Die Mitte der Platte befindet sich in einem Abstand von von 2m und unter einem Winkel von  $20^\circ$  zum Array. Obwohl die Schallwellen nicht senkrecht auf die Platte auftreffen, wird die Platte in 2m erkannt. Der Strich von links oben nach rechts unten ist eine Person, welche sich nach wiederholtem befestigen der Platte aus dem Bild herausbewegt. Für die Aufnahme wurde mit 20 Pulsen gesendet.

In Abbildung C.2 ist dieselbe Styroporplatte von 0.5m Breite in einem Abstand von 2m einem Sendewinkel von  $0^\circ$  zu sehen. Dabei ist um  $0^\circ$  herum ein Sprung zu sehen. Dies ist auf das verzögerte Einschalten dieser PWM-Kanäle zurückzuführen. Auch für dieses Bild wurde mit 20 Pulsen gesendet.

Die nächsten vier Abbildungen C.3, C.4, C.5 und C.6 zeigen alle denselben Aufbau. Quer durch den reflexionsarmen Halbraum wurde ein Tuch gespannt. Dahinter befindet sich eine Person an verschiedenen Positionen. Bei 1.5m, 1.75m und 2.0m sieht man das Tuch. Da es Falten wirft, sieht man es nicht durchgehend. In Abbildung C.3 befindet sich die Person noch knapp nicht hinter dem Tuch, daher ist ein sehr starkes Echo zu sehen. In den weiteren Bildern sieht man die Person hinter dem Tuch an verschiedenen Positionen.

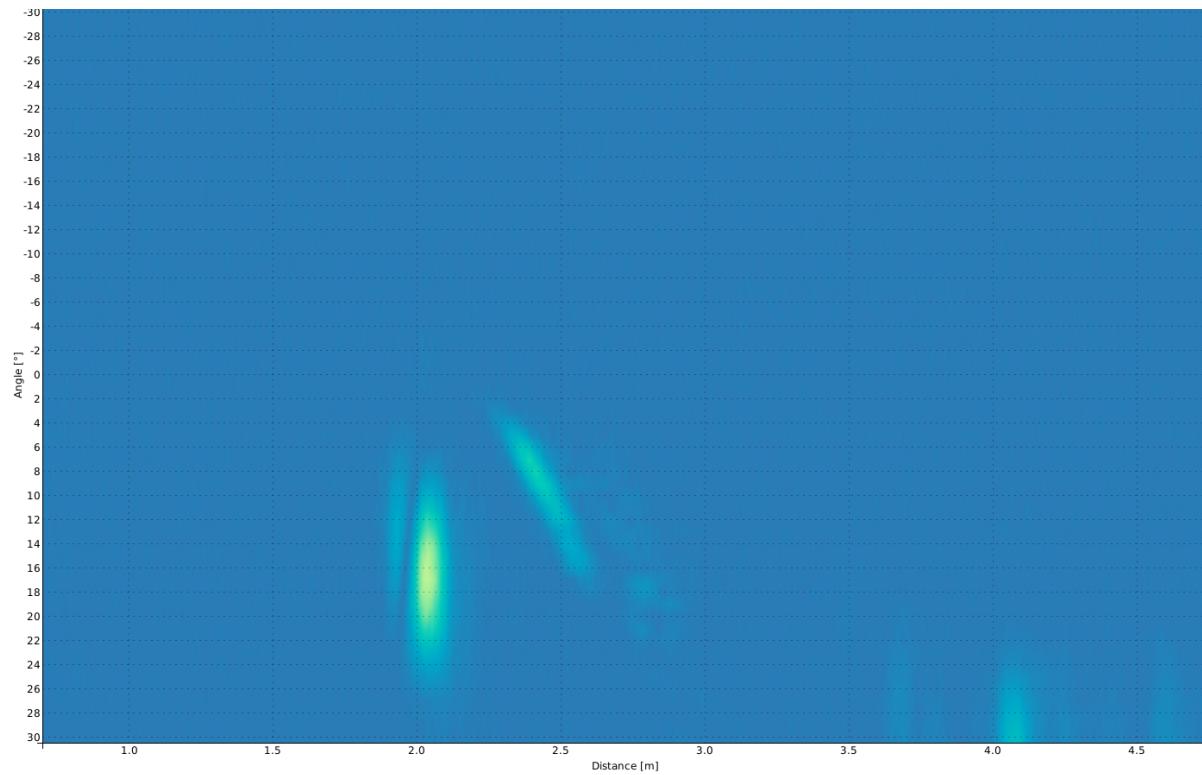


Abbildung C.1: Styroporplatte im reflexionsarmen Halbraum, Person, die sich davon weg bewegt

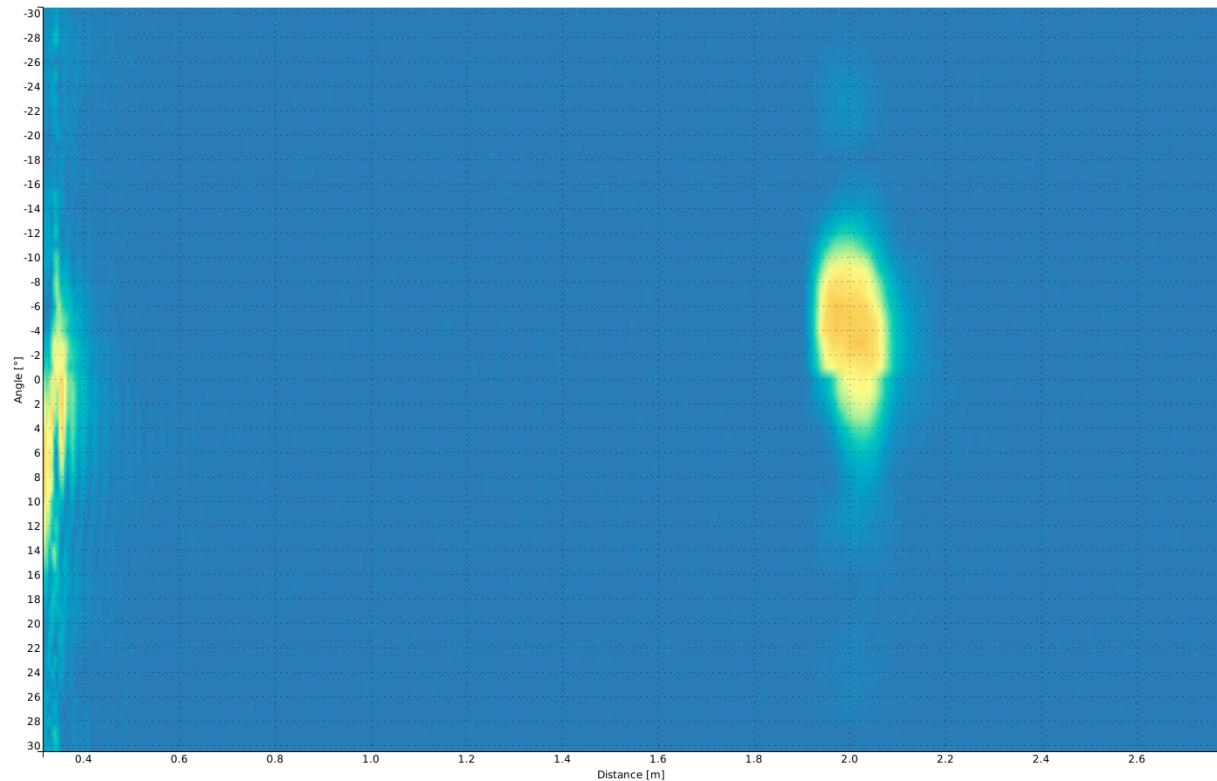
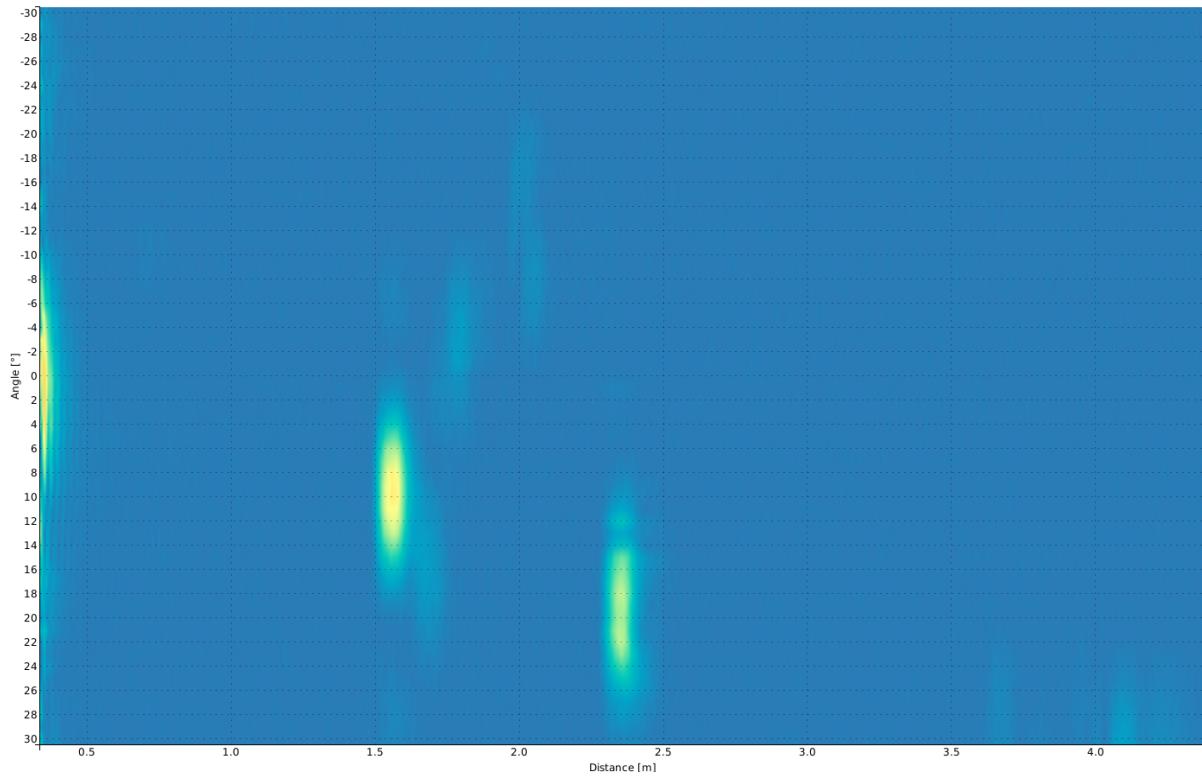
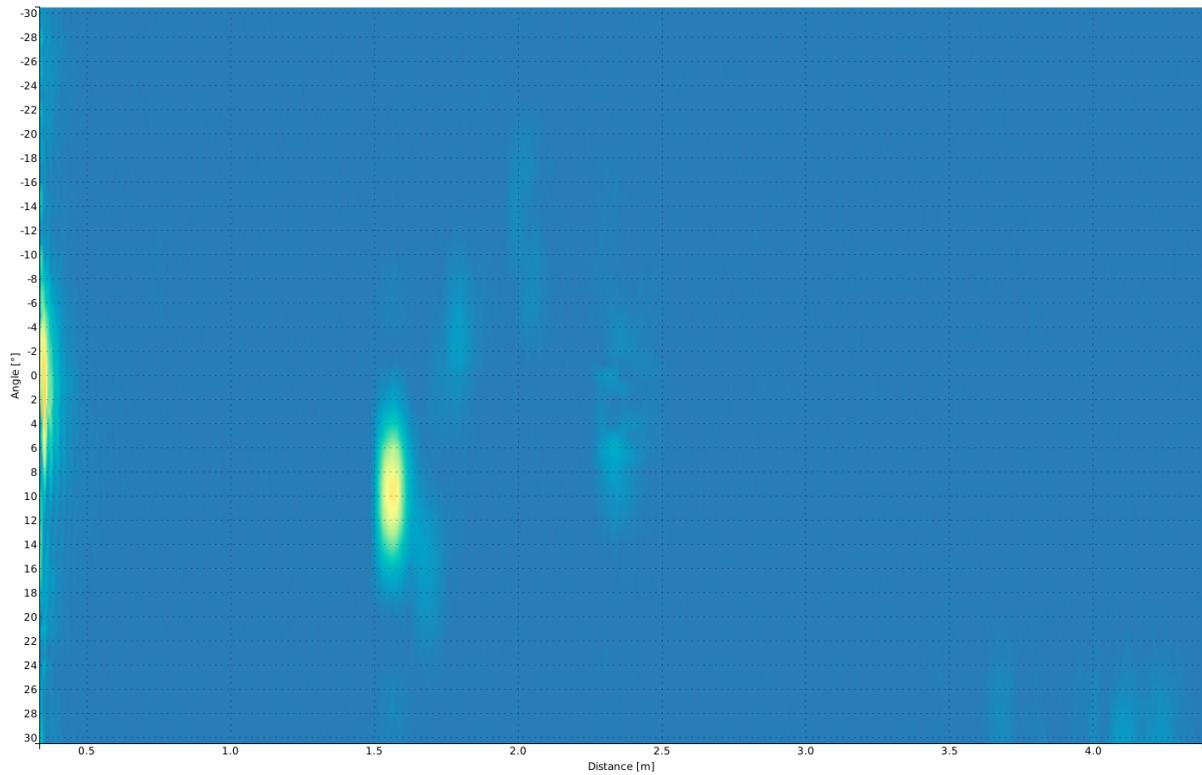


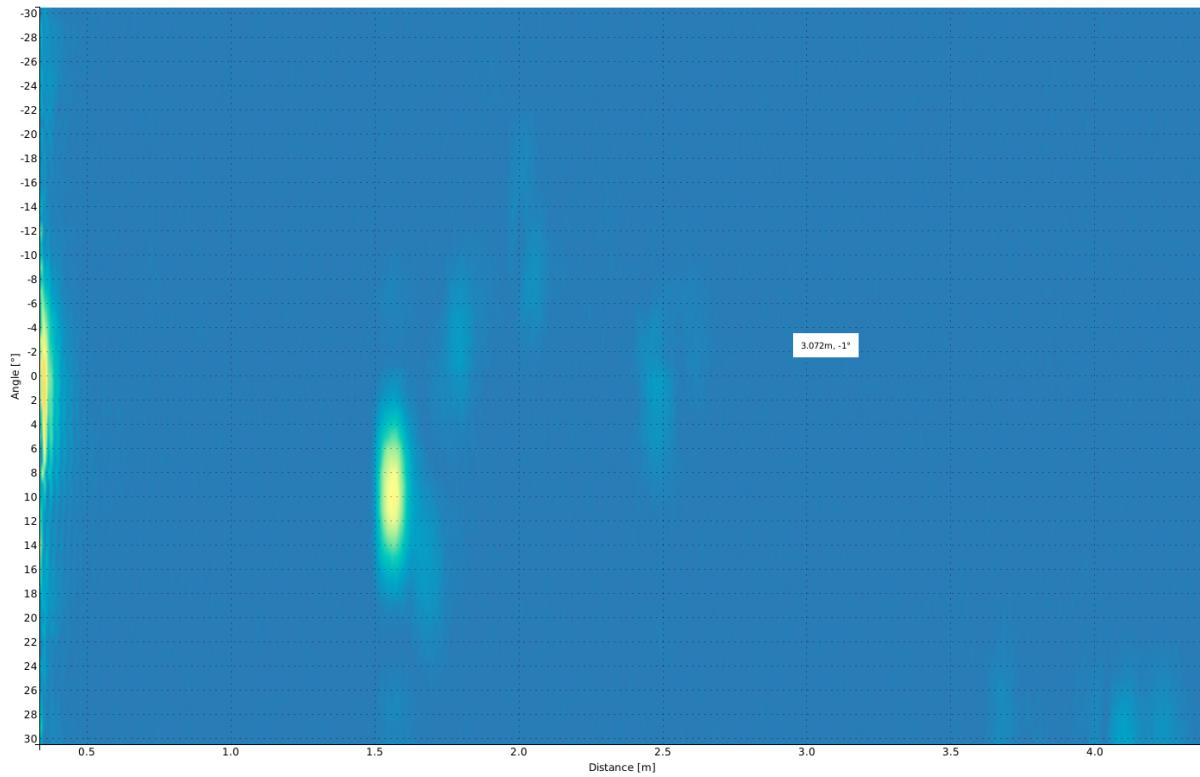
Abbildung C.2: Styroporplatte im reflexionsarmen Halbraum, Sprung bei 0 Grad Sendewinkel



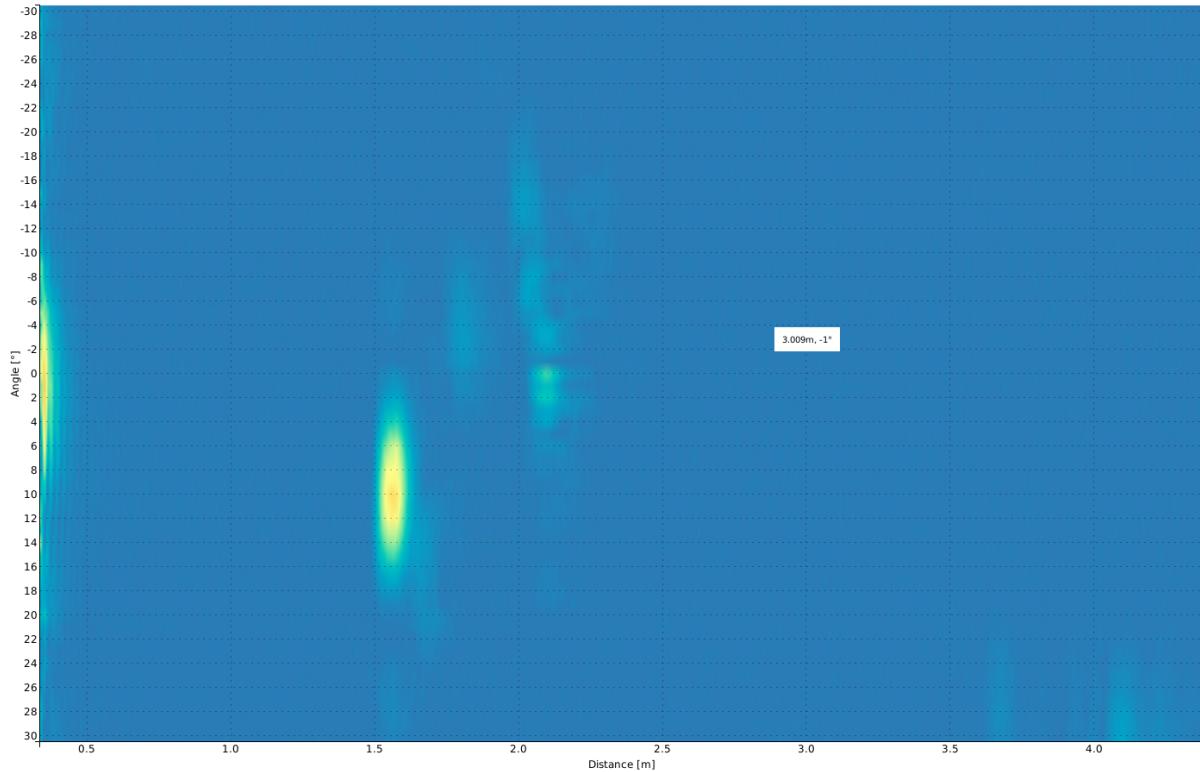
**Abbildung C.3:** Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.4m seitlich vom Tuch



**Abbildung C.4:** Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.3m hinter dem Tuch



**Abbildung C.5:** Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.5m hinter dem Tuch



**Abbildung C.6:** Aufgespanntes Tuch im reflexionsarmen Halbraum, Person steht bei 2.1m hinter dem Tuch

## D Anhang Lizenzen

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of Atmel may not be used to endorse or promote products derived from this software without specific prior written permission.
4. This software may only be redistributed and used in connection with an Atmel microcontroller product.

THIS SOFTWARE IS PROVIDED BY ATMEL ÄS ISÄND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Die Sinngemäße deutsche Übersetzung dieses Lizenztextes lautet wie folgt:

Weitergabe und Verwendung in Sourcecode oder kompilierter Form, mit oder ohne Veränderungen, sind erlaubt solange folgende Bedingungen eingehalten werden:

1. Weitergegebener Sourcecode muss das obenstehende Copyright, diese Auflistung von Vereinbarungen, sowie den nachfolgenden Hinweis enthalten.
2. Weitergegebene Software in kompilierter Form muss das obenstehende Copyright, diese Auflistung von Vereinbarungen, sowie den nachfolgenden Hinweis in der Dokumentation und/oder anderen mit der Software verteilten Bestandteilen des Produktes enthalten.
3. Ohne explizite schriftliche Erlaubnis darf der Name von Atmel nicht verwendet werden, um das von dieser Software abgeleitete Produkt zu bewerben.
4. Diese Software darf nur weitergegeben und benutzt werden in Verbindung mit einem Atmel Mikrocontroller Produkt.

DIESE SOFTWARE WIRD VON ATMEL ZUR VERFÜGUNG GESTELLT OHNE JEGLICHE DIREKTE ODER IMPLIZIERTE GARANTIE AUF FUNKTIONSFÄHIGKEIT FÜR EINEN BESTIMMTEN ZWECK, DIE FUNKTIONSFÄHIGKEIT ALLGEMEIN, SOWIE JEGLICHER RECHTSVERLETZUNG, JEDOCH NICHT DARAUF BESCHRÄNKT. IN KEINEM FALL IST ATMEL HAFTBAR FÜR JEGLICHEN DIREKTEN, INDIREKTEN, UNBEABSICHTIGTEN, SPEZIELLEN, EXEMPLARISCHEN ODER FOLGESCHADEN (INKLUSIVE, ABER NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZGÜTERN ODER DIENSTLEISTUNGEN; NUTZUNGSAUSFALL, DATENVERLUST ODER PROFITVERLUST; ODER BETRIEBSAUSFALL) WIE AUCH IMMER VERURSACHT UND UNABHÄNGIG VON DER RECHTSGRUNDLAGE DER HAFTUNG, OB VERTRAGLICH, IN

VERSCHULDUNGSUNABHÄNGIGER HAFTUNG, ODER AUS UNERLAUBTER HANDLUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER ANDERE), WELCHE AUF JEGLICHE ART DURCH DEN GEBRAUCH DER SOFTWARE AUFTRITT, SOGAR WENN AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE.

## E Anhang Aufgabenstellung

## Aufgabenstellung P6/Thesisarbeit HS16

### Mehrdimensionale Raumerfassung mittels Ultraschall

Auftraggeber:

FHNW, Hochschule für Technik

Betreuende Dozenten:

FHNW, Institut für Mikroelektronik, Richard Gut, [richard.gut@fhnw.ch](mailto:richard.gut@fhnw.ch)

FHNW, Institut für Automation, Matthias Meier, [matthias.meier@fhnw.ch](mailto:matthias.meier@fhnw.ch)

Studenten:

Jonas Plüss, [jonas.pluess@students.fhnw.ch](mailto:jonas.pluess@students.fhnw.ch)

Raphael Mijnssen, [raphael.mijnssen@students.fhnw.ch](mailto:raphael.mijnssen@students.fhnw.ch)

Ausgangslage:

Im Medizinalbereich sind 2D Ultraschall-Messgeräte weit verbreitet. Mit derartigen Geräten lässt sich Weichteil-Gewebe im Vergleich zu anderen Verfahren recht unkompliziert und ohne schädliche Strahlung untersuchen. Grundlage dieser interessanten Technik ist ein Ultraschall Phased-Array System im Megahertzbereich, da nur so eine genügend hohe Tiefenauflösung im Festkörperbereich erreicht werden kann.

Leider sind solche Geräte teuer und deshalb für den „Privatgebrauch“ unerschwinglich. Im Rahmen von Studierendenarbeiten soll deshalb ein kostengünstiges Ultraschall-Messgerät entwickelt werden. Um den Gesamt-Komplexitätsgrad in der ersten Phase in Grenzen zu halten, soll sich das erste Gerät auf Anwendungen im Luftbereich und damit auf Frequenzen im nahen Ultraschallbereich begrenzen.

Im Rahmen einer vorangehenden P5 Projektarbeit wurden die theoretischen Grundlagen der Phased-Array Technik erarbeitet und ein Frontend-Prototyp entwickelt.

Ziel der Arbeit:

Ziel dieses Projektes ist die Entwicklung eines funktionierenden Low-Cost Ultraschall 2D oder 3D Messsystems im Luftbereich für Distanzen im Meterbereich sowie Tiefenauflösung im cm-Bereich.

Pflichtenheft, Projektvereinbarung:

Nach Projektstart soll innert ca. 1 Monat eine Projektvereinbarung inkl. technischem Pflichtenheft erarbeitet werden beinhaltend:

- Lösungskonzept und Spezifikation des zu erstellenden Systems
- Formulierung von Arbeitspaketen (typisch 5-15) und Meilensteinen
- Zeitplan in Form eines Gantt-Diagrammes.

Projektmanagement, Kommunikation, Abgabetermine, Bewertung:

Das Projekt soll von einem schlanken, ergebnisorientierten Projektmanagement begleitet werden.

Die betreuenden Dozenten sollen periodisch (mind. alle 3 Wochen) über den Stand der Arbeiten sowie allfälliger Abweichungen zum Pflichtenheft und Projektplan informiert werden.

Es finden mindestens folgende Meetings statt: Kickoffmeeting, Besprechung Pflichtenheft/Projektvereinbarung, Zwischenpräsentation sowie Schlusspräsentation. Bei Bedarf können mehr Meetings durchgeführt werden.

Betreffend Fachbericht, Ausstellung der Diplomarbeit (inkl. Erstellen eines Posters und einer Webseite), Verteidigung und Bewertung gelten die Vorgaben und Richtlinien der FHNW, Hochschule für Technik.

Termine:

Es gelten die offiziellen Termine der FHNW, Hochschule für Technik, Bachelor-Thesis Variante A/Start HS:

- Das Projekt startet der KW 38 / 2016 und umfasst 360 Stunden für (je)den Probanden.
- Der Abgabetermin der Arbeit (Fachbericht) ist am Freitag 24.3.2017 (KW12).
- Die Verteidigung (d.h. Präsentation und Diskussion der Thesis vor Auftraggeber, Betreuer und externem Experten) findet voraussichtlich in der KW17 statt (der genaue Termin wird noch bekannt gegeben).

**F Anhang Pflichtenheft**

## PFLICHTENHEFT

# MEHRDIMENSIONALE RAUMERFASSUNG MITTELS ULTRASCHALL

06.10.2016

---

AUFTAGGEBER

FHNW, HOCHSCHULE FÜR TECHNIK

BETREUUNG

RICHARD GUT, FHNW, INSTITUT FÜR MIKROELEKTRONIK  
MATTHIAS MEIER, FHNW, INSTITUT FÜR AUTOMATION

TEAM

RAPHAEL MIJNSSEN, raphael.mijnssen@students.fhnw.ch  
JONAS PLÜSS, jonas.pluess@students.fhnw.ch

STUDIENGANG

ELEKTRO- UND INFORMATIONSTECHNIK

---

### **Inhaltsverzeichnis**

<b>1 Ausgangslage</b>	<b>1</b>
<b>2 Ziele</b>	<b>2</b>
<b>3 Lösungskonzept</b>	<b>3</b>
<b>4 Testkonzept</b>	<b>6</b>
<b>5 Projektplan</b>	<b>8</b>

## 1 Ausgangslage

Nicht ohne Grund hat die Evolution die Fledermaus mit einem aussergewöhnlichen Sinn zur Orientierung im Raum ausgestattet. Die Möglichkeit, anhand der Echos von hochfrequenten akustischen Schallwellen Informationen über den Raum und mögliche Hindernissen zu gewinnen, ist eine spannende Alternative gegenüber dem rein optischen Sehsinn. So ist die Fledermaus z.B. nicht auf grosse Mengen Licht angewiesen. Auch aus der heutigen Medizinaltechnologie und der Zerstörungsfreien Materialprüfung (engl. non-destructive testing) ist Ultraschall nicht mehr wegzudenken. Die Möglichkeit mit relativ billigen Ressourcen Informationen über den Raum gewinnen zu können oder das Innere von Strukturen und Objekten zu untersuchen, ohne diese zu beschädigen wird dabei ausgenutzt. Es werden dafür je nach Art und Position der zu untersuchenden Struktur unterschiedliche Reflektionen der Ultraschallwellen gemessen. Die zu diesem Zweck erzeugten Schallwellen werden in der Regel mittels piezoelektrischem Effekt erzeugt. Wenn anstelle eines einzelnen Senders ein ganzes Array von Sendern verwendet wird, kann die Richtwirkung des resultierenden Sendekegels verbessert werden, sowie mittels elektrischer Zeitverzögerungen (Phasenverschiebung) bei der Ansteuerung die Richtung des Kegels geändert werden.

Umsetzungen dieser Technologien sind teuer und die Produkte in der Regel kommerziell, daher finden sich kaum öffentlich zugängige Projekte.

Ziel ist die Umsetzung eines Ultraschall Phased Array Systems, welches eindimensional in Luft aufgebaut wird und mit welchem die Objekterkennung im Raum ermöglicht wird. Es soll dabei sowohl Hardware, als auch Software entwickelt werden, welche unter GPL/Creative-Commons Lizenz steht und möglichst simpel, bzw. preiswert realisiert ist. Als Basis dient das vorausgehende Projekt, wobei mittels funktionierendem Prototyp die Machbarkeit geprüft wurde und theoretische Grundlagen erarbeitet wurden. Dieser Prototyp wird durch ein Hardware Redesign bezüglich der Kosten, der Nebenkeulenunterdrückung und der minimalen messbaren Distanz optimiert. Die Signalverarbeitung der empfangenen Daten wird in einer dafür geeigneten Programmiersprache umgesetzt und dem GUI übergeben, welches möglichst Plattformunabhängig sein sollte. Die Visualisierung der Daten und die Steuerung der Hardware erfolgen über das GUI.

## 2 Ziele

Das Ziel des Projektes ist es, ein 1x8 Ultraschall Phased Array System zu entwickeln, welches der Objekterkennung in Luft dient. Im vorhergehenden Projekt wurde ein Prototyp gebaut, mit welchem bereits die Machbarkeit eines solchen Systems untersucht wurde. Dieser Prototyp wird bezüglich Unterdrückung der Nebenkeulen, Kosten und Auflösung für kleine Distanzen optimiert und dafür einem Hardware Redesign unterzogen. Die Signalverarbeitung der empfangenen Echos wird in Software auf einem Linux Rechner in einer dafür geeigneten Programmiersprache implementiert und die Daten mit einem plattformunabhängigen GUI visualisiert. Zusätzlich wird die Steuerung der Hardware auch über dieses GUI erfolgen.

### Sollziele

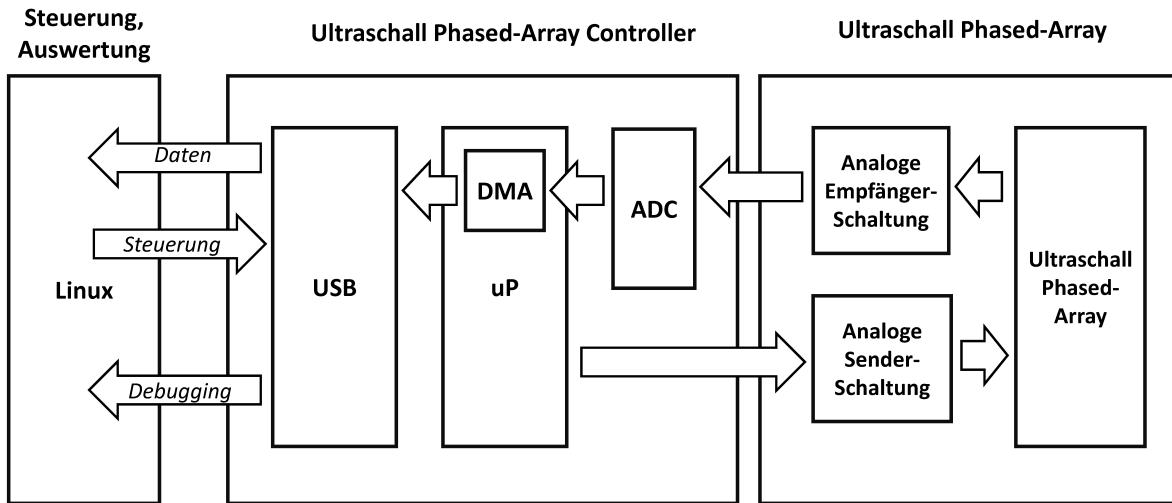
- Maximale Messdistanz: 5 – 10m
- Minimale Tiefenauflösung:  $\pm 10\text{cm}$
- Verwenden des gleichen Piezoelementes beim Senden und Empfangen
- Verkürzung der Abklingzeit der Piezoelemente
- Redesign der Hardware
- Verbesserung der Richtwirkung durch Amplitudenbelegung
- Anpassung der Firmware an Hardware Redesign
- Anpassung der Firmware für eine Steuerung per GUI
- Evaluation der verwendeten Programmiersprachen/Schnittstellen
- Implementation der Signalverarbeitung in der evaluierten Programmiersprache
- Implementation der Visualisierung der Daten
- Erstellen eines GUI
- Messen der Richtcharakteristik im Schalltoten Raum (siehe Testkonzept)
- Messen der Performance (Objekterkennung) im freien Feld (siehe Testkonzept)

### Wunschziele

- Externes Sendemodul für Positionsbestimmung entwickeln
- Anpassung der Firmware/Software für Zusammenarbeit mit externem Sendemodul
- Darstellung der Positionsdaten des externen Sendemoduls

### 3 Lösungskonzept

Im nachfolgenden Kapitel wird erläutert, wie die bestehenden Probleme in der Hardware gelöst werden sollen und wie die Weiterentwicklung der Software aussehen soll. Das Lösungskonzept für das in den Wunschzielen aufgeführte externe Sendemodul wird separat betrachtet. Die grundlegende Struktur des Systems ist in der Abbildung 3.1 ersichtlich.



**Abbildung 3.1:** Gesamtübersicht des Systems

### Hardware

Die im letzten Projekt entwickelte Hardware funktioniert zwar, weist aber einige kleine Schwächen auf, welche behoben werden sollen. Zwischen den Ultraschallsendern und -empfängern findet ein relativ starkes Übersprechen statt, was zusammen mit der langen Abklingzeit der Module dafür verantwortlich ist, dass jeweils zu Beginn des Empfangens noch keine Echos detektiert werden können. Diese zeitliche Verzögerung vergrößert die minimal erkennbare Distanz des Phased Arrays. In zwei Schritten soll dieser Umstand verbessert werden:

- Aufbau mit nur einem Sender-/Empfängermodul, um Übersprechen zu verhindern.
- Verkürzung der Abklingzeit durch Dämpfen des Ausschwingens der Ultraschallmodule.

Um Gewissheit zu haben, dass die Verbesserungen auch funktionieren, werden die Konzepte zuerst auf einer Versuchsschaltung mit einem Ultraschallmodul aufgebaut und erst bei erfolgreichen Tests im Phased Array implementiert.

### Versuche 1 & 2 – Evaluation für das Hardware Redesign

Auf einer Versuchsplatinen wird ein Schaltkreis mit nur einem Sender-/Empfängermodul aufgebaut. Dabei sind folgende Punkte wichtig:

- Die beiden Vorspannungen an den Operationsverstärkern der Empfängerschaltung werden nicht über einzelne Spannungsteiler sondern über eine gemeinsame aktive Spannungsquelle erzeugt.
- Die Empfängerschaltung soll während dem Sendevorgang abkoppelbar sein.
- Über einen Steuerpin soll ein  $1\text{k}\Omega$ -Widerstand parallel zum Ultraschallmodul geschaltet werden können, um die Abklingzeit zu verkürzen.

Die Abkoppelung der Empfängerschaltung und das Zuschalten des Widerstandes soll mithilfe eines MOSFETs oder einer PIN-Diode geschehen. Die Vorspannung am Operationsverstärker wird mithilfe einer Spannungsfolgerschaltung (Impedanzwandler) mit einem weiteren Operationsverstärker erzeugt.

### **Redesign der Hardware**

Aufbauend auf den Erkenntnissen aus den Versuchen 1 & 2 wird der Hardware Redesign gemacht. Für alle Vorspannungen an den Operationsverstärkern soll dieselbe aktive Spannungsquelle verwendet werden. Um schnelle Spannungsschwankungen zu verhindern, wird vor jedem Operationsverstärker ein Blockkondensator eingesetzt. Ansonsten wird die Beschaltung aus dem letzten Projekt übernommen. Das Umschalten der Ultraschallmodule zwischen Sender- und Empfängerschaltung soll für alle Module über einen einzelnen Pin gesteuert werden können, wie auch das Zuschalten des parallelen Widerstandes, falls sich dieser Ansatz gegen das Phasenverschobene Ansteuern (siehe Lösungskonzept Software) durchsetzt.

### **Software – Firmware**

Verursacht durch das Hardware Redesign müssen auch an der Firmware einige Änderungen gemacht werden. Zusätzlich wird eine Softwareschnittstelle eingebaut, über die das Phased Array gesteuert werden kann.

### **Anpassung der Firmware an Hardware Redesign**

Um die Ultraschallmodule zwischen Sender- und Empfangsschaltung umzuschalten, sowie um die Widerstände zum Verkürzen der Abklingzeit dazuzuschalten müssen zusätzliche Pins per Software angesteuert werden.

Ein zweiter Ansatz zum Verkürzen der Abklingzeit ist das um  $180^\circ$  Phasenverschobene Ansteuern der Sendemodule am Ende des Sendevorgangs. Es wird versucht, dies in der Firmware zu implementieren. Aufgrund des problematischen PWM-Timings im Atmel Software Framework (ASF) kann es aber sein, dass dieser Versuch scheitert. Dann bliebe nur noch das Dämpfen mithilfe eines parallel zum Sendemodul geschalteten Widerstandes.

### **Anpassung der Firmware für Amplitudenbelegung**

Um eine Nebenkeulenunterdrückung am Senderarray zu erreichen, kann die Amplitude der einzelnen Sendemodule mit einer Fensterfunktion belegt werden. Da die Anregung der Sendemodule mit einer PWM geschieht, kann die Pulsbreite entsprechend der gewünschten Sendeleistung angepasst werden.

### **Anpassung der Firmware für Steuerung über GUI**

Um das Ultraschall Phased Array über das GUI steuern zu können, muss die Firmware ebenfalls angepasst werden. Dazu wird wie auch zum Senden der Daten vom Phased Array zum PC die USB-Schnittstelle verwendet. Folgende Parameter sollen per GUI auf dem Phased Array Modul verändert werden können:

- Sendewinkel
- Anzahl der Empfangsbuffer und somit die Empfangszeit
- Amplitudenbelegung

## **Software – Signalverarbeitung / Visualisierung**

Die Software, welche die Daten des Phased Arrays auswertet, wird grob unterteilt in Signaverarbeitung und GUI. Eine erste rudimentäre Visualierung der Daten wird in Python geschrieben, um einen groben Überblick über die empfangen Daten zu erzielen. Danach werden die optimalen Programmiersachen und Schnittstellen für die Umsetzung evaluiert.

### **Signalverarbeitung**

Die Signalverarbeitung läuft auf einem Linuxrechner. Die Interpolation soll dabei nicht mittels Upsampling und Filterung im Zeitbereich, sondern im Bildbereich mithilfe der FFT realisiert werden. So kann Rechenzeit eingespart werden. Daraufhin soll die Enveloppe des Signals gebildet werden. Diese wird danach zusammen mit dem Winkel dem GUI übergeben.

### **GUI**

Das GUI soll möglichst plattformunabhängig sein, evtl. Browserbasiert mit Javascript oder als Java Applet. Als Schnittstelle zwischen GUI und Signalverarbeitung kann womöglich TCP/IP oder Unix Sockets verwendet werden. Die empfangenen Enveloppen werden jeweils für jeden Winkel aufgetragen. Dabei wird die Distanz (Zeit) als Strecke und die Signalamplitude als Farbe dargestellt.

### **Externes Sendemodul**

Optional soll ein externes Sendemodul aufgebaut werden, das vom Phased Array Modul angepingt werden kann und darauf aktiv ein kurzes Ultraschallsignal sendet. Das Phased Array empfängt dieses Signal und berechnet aus der Phasenverschiebung den Eintrittswinkel und aus der Signallaufzeit die Distanz zum externen Modul.

### **Pingsignal**

Damit die Signallaufzeit berechnet werden kann, muss das externe Sendemodul vom Phased Array getriggert werden. Dazu gibt es mehrere Lösungsansätze. Es könnte ein Ultraschallsignal, ein Funksignal oder ein Lichtsignal (Infrarot) verwendet werden. Beim triggern mittels Ultraschall entsteht einerseits die doppelte Signallaufzeit, andererseits könnten Reflexionen vom Phased Array falsch interpretiert werden. Die anderen beiden Lösungsansätze sind deshalb vorzuziehen.

### **Mikrocontroller**

Da nur ein Pingsignal empfangen und daraufhin ein Ultraschallsignal ausgesendet werden muss und keinerlei Signalverarbeitung auf dem externen Sendemodul stattfindet, werden an den Mikrocontroller keine hohen Anforderungen gestellt. Ein einfacher Atmega oder Attiny, welcher über einen Hardwareinterruptpin und einen PWM-Ausgang verfügt, sollte ausreichen. Für erste Tests wird dazu ein Arduino Nano verwendet.

### **Ultraschallmodul**

Da nur ein Sendemodul benötigt wird, kann dazu eigentlich jedes Ultraschallmodul mit 40kHz Resonanzfrequenz verwendet werden. Je höher dabei der Schalldruckpegel ist, desto weitere Distanzen können ausgemessen werden. Dabei ist aber darauf zu achten, dass die Empfänger des Phased Arrays nicht übersteuert werden.

## 4 Testkonzept

Im folgenden Kapitel wird das Testkonzept vorgestellt. Für eine zeitliche Übersicht sei dafür auf Abbildung 4.1 verwiesen.

### Verifikation der Versuche 1 & 2 – Evaluation für das Hardware Redesign

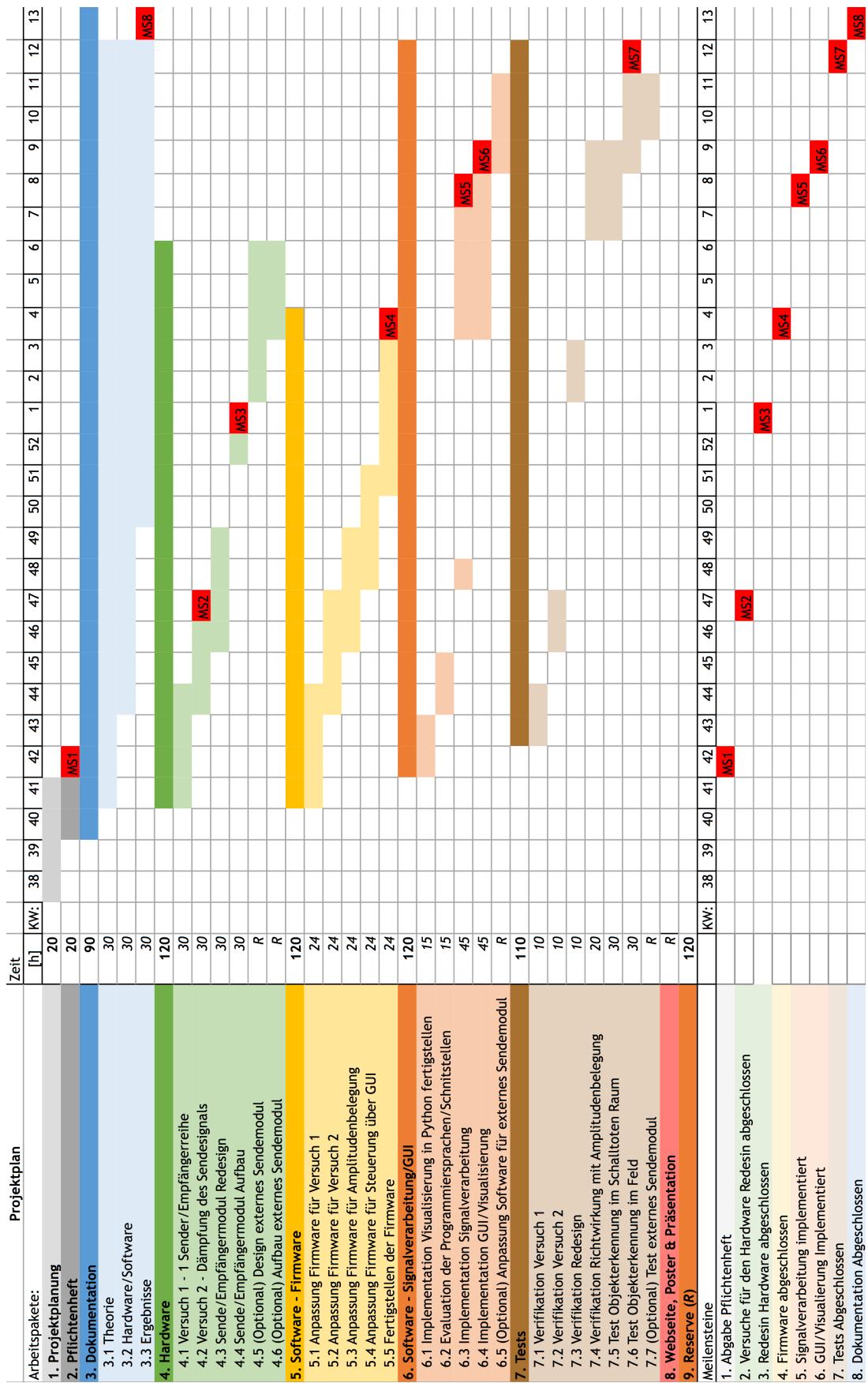
Es wird eine Testschaltung aufbaut, wobei nur ein Piezoelement für Sende- und Empfangsschaltung verwendet wird. Diese Schaltung wird mittels angepasster Firmware auf Funktionalität geprüft. Als zweiter Schritt wird die Dämpfung der Sensoren im Vergleich zum vorherigen Prototyp ohne Dämpfung verglichen. Es soll so festgestellt werden, wie klein der minimal messbare Abstand zum Sender ist. Dafür wird die Zeit gemessen, nach welcher die Amplitude des Signal am Ausgang der Verstärkerschaltung auf -26dB gedämpft ist im Vergleich zu den Aussteuerbaren 3.3 V.

### Tests nach Redesign der Hardware

Die fertiggestellte Hardware wird bezüglich Richtwirkung, Winkelauflösung und Distanz getestet. Dafür wird im schalltoten Raum der Schalldruck in Abhängigkeit des Winkels und der Distanz vermessen. Die Wirkung der Amplitudenbelegung auf die Unterdrückung der Nebenkeulen wird dabei auch vermessen.

### Tests nach fertigstellung der Software

Nach der Fertigstellung der Software wird das Gesamtsystem getestet. Dafür werden Testobjekte zuerst im schalltoten Raum und danach im freien Feld vermessen. Die Resultate werden direkt im GUI dargestellt. Falls das externe Sendemodul (optionales Ziel) entwickelt worden ist, wird für dieses die minimale Ortsauflösung und die maximal messbare Distanz ermittelt. Testmessungen sollen daraufhin im GUI dargestellt werden.



## 5 Projektplan

In diesem Kapitel wird das Zeitmanagement dieses Projektes behandelt. Der Projektplan gibt einen Überblick über die durchzuführenden Arbeiten, deren Zeitliche Einteilung und die Meilensteine welche zu erreichen sind. In einem Weiteren Abschnitt werden die einzelnen Arbeitspakete genauer beschrieben.

### Arbeitspakete

Im Folgenden werden die im Zeitplan (Abbildung 4.1) aufgelisteten Arbeitspakete beschrieben.

#### 1. Projektplanung

Während der Projektplanung werden die Ergebnisse des letzten Projektes ausgewertet und das Lastenheft für das neue Projekt mit dem Auftraggeber besprochen. Die für dieses Projekt relevanten Dokumentvorlagen werden erstellt. Es wird ein Zeitplan ausgearbeitet, in welchem ersichtlich ist, wer zu welcher Zeit wieviele Stunden am Projekt arbeitet. Parallel dazu wird ein Zeitplan in Form eines Gantt-Diagrammes erstellt, in welchem alle Arbeitspakete, die Meilensteine sowie die dazu benötigte Zeit ersichtlich sind.

#### 2. Pflichtenheft

Das Pflichtenheft beschreibt konkret, was die Ziele dieses Projekts sind und wie sie umgesetzt werden. Es soll am Schluss des Projektes als Referenz zum Vergleich mit den Resultaten zur Verfügung stehen. Zusätzlich dient es der Orientierung und Planung während des Projektverlaufs. Die während der Projektplanung erstellten Zeitpläne sind Bestandteil des Pflichtenheftes.

#### 3. Dokumentation

Am Ende des Projektes wird ein Fachbericht abgegeben, welcher während der Projektdurchführung ausgearbeitet wird. Er beinhaltet die theoretischen Grundlagen der Akustik und der benötigten Signalverarbeitung sowie die Dokumentation der Hardware, der Software und der Ergebnisse. Das Ziel des Fachberichtes ist es, dass getroffene Entscheidungen nachvollzogen werden können, die Funktionsweise der Hard-/Software verständlich ist und die Tests reproduziert werden können.

##### 3.1 Theorie

Die theoretischen Grundlagen werden aufbauend auf dem im letzten Projekt erarbeiteten Wissen erweitert. Hauptsächlich wird der Teil der Signalverarbeitung verfeinert, sowie die Theorie zur Aperturbelegung bei Phased Array's zur Unterdrückung der Nebenkeulen erarbeitet.

##### 3.2 Hardware / Software

Aufbauend auf der Dokumentation aus dem letzten Projekt werden die Hardware und die Software im Top-Down Prinzip beschrieben. Für eine Fachperson soll es möglich sein, getroffene Entscheidungen nachzuvollziehen, die Hardware nachzubauen und die Software sowie die Entwicklungsumgebung in Betrieb zu nehmen.

##### 3.3 Ergebnisse

Die während dem Projekt gemachten Tests und Messungen werden in den Ergebnissen dokumentiert und sollen reproduzierbar sein. In diesem Teil der Dokumentation werden ebenfalls Empfehlungen zur Weiterentwicklung des Projektes gemacht, welche aus zeitlichen Gründen nicht realisiert werden konnten.

## 4. Hardware

Da zwischen den Ultraschallsendern und -empfängern ein relativ starkes Übersprechen stattfindet, welches die minimale Empfangsdistanz einschränkt, wird die Hardware einem Redesign unterzogen. Dabei können ev. auch noch zusätzliche Funktionalitäten eingebaut werden.

### 4.1 Versuch 1 – Sender/Empfängerreihe

Als erstes wird ein Aufbau getestet, bei dem zum Senden und zum Empfangen dasselbe Ultraschallmodul verwendet wird. Dies verhindert ein Übersprechen zwischen den Sende- und Empfangsmodulen und spart Kosten. Dazu wird kein ganzes Array aufgebaut, sondern nur ein einzelnes Modul getestet.

### 4.2 Versuch 2 – Dämpfung des Sendesignals

Wenn das Ultraschallmodul nach dem Senden um  $180^\circ$  Phasenverschoben angesteuert wird, sollte es relativ schnell gedämpft werden können. Sollte dies aufgrund des Mikrocontrollers nicht möglich sein, wird nach dem Senden für kurze Zeit ein Widerstand parallel zum Sendemodul geschaltet, welcher dieses dämpft.

### 4.3 Sende-/Empfängermodul Redesign

Den Ergebnissen der Versuche 1 und 2 entsprechend wird die komplette Hardware einem Redesign unterzogen, wobei die in den Versuchen getesteten Verbesserungen implementiert werden.

### 4.4 Sende-/Empfängermodul Aufbau

Nach der Anfertigung des PCB werden die Komponenten darauf verbaut. Es sollen mindestens zwei, besser drei Module aufgebaut werden.

### 4.5 (Optional) Design externes Sendemodul

Eine zusätzliche Möglichkeit des Phased Arrays auf dem Empfängermodul ist die zweidimensionale Ortung einer externen Ultraschallquelle. Mithilfe von zwei Empfängermodulen (eines horizontal und eines vertikal angeordnet) könnte sogar eine dreidimensionale Ortung stattfinden. In diesem Arbeitspaket soll ein externes Sendemodul entwickelt werden, welches vom Sende-/Empfängermodul angepingt werden kann und danach ein Signal aussendet welches vom Empfängerarray aufgenommen wird.

### 4.6 (Optional) Aufbau externes Sendemodul

Nach dem Entwickeln des externen Sendemoduls wird es aufgebaut und getestet.

## 5. Software – Firmware

Das Redesign der Hardware erfordert auch einige Änderungen an der Firmware auf dem Mikrocontroller. Diese wird schrittweise immer dann angepasst, wenn die entsprechenden Änderungen an der Hardware erfolgreich abgeschlossen wurden.

### 5.1 Anpassung der Firmware für Versuch 1

Die Firmware muss so angepasst werden, dass sie die Ultraschallmodule im richtigen Moment zwischen der Sendelektronik und der Empfängerelektronik hin- und her schalten kann.

### 5.2 Anpassung der Firmware für Versuch 2

Die Firmware muss so angepasst werden, dass das PWM-Signal, welches die Sender steuert am Ende des Sendevorgangs für kurze Zeit um  $180^\circ$  Phasenverschoben sendet. Dadurch kann das lange Ausschwingen des Ultraschallmodules verkürzt werden. Sollte dieser Lösungsansatz aus Timinggründen des Mikrocontrollers scheitern, wird die Firmware so umprogrammiert, dass sie nach dem Sendevorgang einen Widerstand parallel zum Sendemodul schaltet, welcher dessen Ausschwingen maximal dämpft.

### 5.3 Anpassung der Firmware für Amplitudenbelegung

Um am Senderarray die ausgesendeten Nebenkeulen zu unterdrücken, kann die Amplitude mit einer Fensterfunktion belegt werden. Dazu muss die Firmware so angepasst werden, dass jedes Ultraschallmodul mit einer anderen Leistung angesteuert wird. Am einfachsten erreicht man dies, indem man den Duty-Cycle des PWM-Signals für jeden Sender anpasst. Optimal wäre es, wenn die Amplitudenbelegung zur Laufzeit geändert werden könnte.

### 5.4 Anpassung der Firmware für Steuerung über GUI

Damit während dem Betrieb des Moduls Parameter wie der Sende-/Empfangswinkel, die Amplitudenbelegung, die maximale Empfangszeit, usw. geändert werden können, muss eine Schnittstelle implementiert werden, sodass das Modul auf Steuerbefehle reagieren kann. Da die Daten vom Modul zum PC per USB übertragen werden, soll auch dazu die USB-Verbindung benutzt werden.

### 5.5 Fertigstellen der Firmware

In dieser Phase werden nur noch kleine Anpassungen gemacht und der gesamte Sourcecode einem Review unterzogen.

## 6. Software - Signalverarbeitung/GUI

Die Software auf dem PC, welche die vom Phased Array aufgenommenen Ultraschallsignale verarbeitet soll in dieser Arbeitspaketgruppe weiterentwickelt und fertiggestellt werden. Dabei wird grob zwischen der Signalverarbeitung und der Visualisierung (GUI) unterschieden.

### 6.1 Implementation Visualisierung in Python fertigstellen

In diesem Arbeitspaket wird die im letzten Projekt entwickelte Software weiterentwickelt, so dass die empfangenen Signale besser visualisiert werden. Momentan ist es damit noch relativ schwierig nachzuvollziehen, was das Phased Array "sieht". Dies soll durch eine zweidimensionale Ansicht des Empfangsbereich mit farblicher Auflösung der Signalamplitude verbessert werden.

### 6.2 Evaluation der Programmiersprachen/Schnittstellen

Im nächsten Schritt wird evaluiert, in welcher Programmiersprache die Software schlussendlich implementiert werden soll, und welche Schnittstellen dabei genutzt werden sollen. Die Idee dahinter ist, dass die Signalverarbeitung in einer schnellen und effizienten Programmiersprache implementiert wird, die Visualisierung dagegen in einer Programmiersprache, mit der möglichst Plattformunabhängige GUI-Entwicklung ohne grossen Aufwand möglich ist.

### 6.3 Implementation Signalverarbeitung

Der Signalverarbeitungsteil wird in der im letzten Arbeitspaket definierten Programmiersprache implementiert.

### 6.4 Implementation GUI/Visualisierung

Das GUI wird programmiert, so dass die verarbeiteten Signale dargestellt werden können und das Phased Array Modul angesteuert werden kann.

### 6.5 (Optional) Anpassung Software für externes Sendemodul

In diesem Arbeitspaket wird die Software (Signalverarbeitung und GUI) so angepasst, dass eine Verwendung mit dem externen Sendemodul möglich ist. Das Externe Modul soll ebenfalls über das GUI steuerbar sein.

## 7. Tests

Alle Tests werden im Fachbericht dokumentiert, so dass sie nachvollziehbar, sowie auch reproduzierbar sind.

## **7.1 Verifikation Versuch 1**

In diesem Test wird verifiziert, ob anstatt je einem separaten Sender/Empfänger auch ein einzelnes Modul verwendet werden kann.

## **7.2 Verifikation Versuch 2**

Im Test zum Versuch 2 wird verifiziert, ob das verkehrphasige Ansteuern der Ultraschallmodule wie erwartet funktioniert, oder ob eine Dämpfung des Ausschwingvorganges mithilfe eines Parallelwiderstandes besser funktioniert.

## **7.3 Verifikation Redesign**

Nach abgeschlossenem Redesign und Aufbau der Hardware wird verifiziert, ob sich das Array wie erwartet verhält. Dazu wird die Richtcharakteristik in Sende- sowie in Empfangsrichtung ausgewertet.

## **7.4 Verifikation Richtwirkung mit Amplitudenbelegung**

Wenn die Hardware wie gewünscht funktioniert, wird in diesem Test verifiziert, ob mithilfe der Amplitudenbelegung tatsächlich die gewünschte Nebenkeulenunterdrückung erreicht wird. Auch der Einfluss unterschiedlicher Belegungen auf die breite der Hauptkeule soll untersucht werden.

## **7.5 Test Objekterkennung im Schalltoten Raum**

Im Schalltoten Raum wird an verschiedenen Objekten getestet, wie gut sie sich mit dem Phased Array erkennen lassen. Dabei soll hauptsächlich untersucht werden, wie hoch die Winkelauflösung und wie präzise die Distanzmessung funktioniert. Zusätzlich wird auch an verschiedenen Materialien getestet, wie gross der Einfluss des Eintrittswinkels des Ultraschallstrahls ist. Damit soll die Frage geklärt werden, ab welcher "Rauheit" geneigte Oberflächen erkennbar sind.

## **7.6 Test Objekterkennung im Feld**

In diesem Test wird die Einsatzfähigkeit in realer Umgebung getestet, in der auch unerwünschte Reflexionen und Echos existieren. Dabei soll auch die maximale Reichweite ermittelt werden.

## **7.7 (Optional) Test externes Sendemodul**

In diesem Arbeitspaket wird das externe Sendemodul getestet. Es soll verifiziert werden, wie hoch die Winkelauflösung ist und wie präzise die Distanzmessung funktioniert.

## **8. Webseite, Poster & Präsentation**

Dieses Arbeitspaket wird erst nach der Abgabe des Projektes bearbeitet. Es wird nach den Richtlinien der FHNW eine Webseite, ein Poster und eine Präsentation erstellt, um das Projekt vorzustellen.

## **9. Reserve**

Die Reservezeit wird wenn möglich für die Umsetzung der mit "optional" gekennzeichneten Arbeitspakete genutzt, in welchen ein externes Sendemodul angefertigt wird.