

# Entwicklung und testen eines Ultraschall-Entfernungsmessers als Vorbereitung eines Produktentwurfes

---

## Projektarbeit

erstellt an der  
Fachschule für Technik des Carl-Severing-Berufskollegs  
für Metall- und Elektrotechnik der Stadt Bielefeld



Erstellt durch:

Eduard Meiser  
Omar Hachimi  
Stephan Dannat  
FET6A

in Zusammenarbeit mit der Fa. Tinkerforge  
betreut durch  
Herr Simon

Bielefeld, 16. März 2018

## Persönliche Erklärung

Hiermit bestätigen wir, dass die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internet-quellen) entnommen sind, wurden unter Angabe der Quellen kenntlich gemacht.

Bielefeld, \_\_\_\_\_

\_\_\_\_\_  
Eduard Meiser

\_\_\_\_\_  
Omar Hachimi

\_\_\_\_\_  
Stephan Dannat

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Lastenheft . . . . .	1
1.1.1	Über Tinkerforge . . . . .	1
1.1.2	Motivation . . . . .	1
1.1.3	Aufgabenbeschreibung . . . . .	1
1.2	Management des Projektes . . . . .	2
1.2.1	Trello . . . . .	2
1.2.2	Github . . . . .	2
1.3	Open Source CAD-Programm Paket zur Erstellung von elektronischen Leiterplatten	3
1.3.1	Schaltplan . . . . .	3
1.3.2	Platinenlayout . . . . .	3
<b>2</b>	<b>Vorbereitung</b>	<b>4</b>
2.1	Recherche der Funktionsweise . . . . .	4
2.1.1	Controller . . . . .	4
2.1.2	Sender . . . . .	4
2.1.3	Hochsetzsteller . . . . .	4
2.1.4	Ultraschallkapsel . . . . .	5
2.1.5	Filter . . . . .	5
2.1.6	Empfänger . . . . .	5
2.1.7	Besonderheit der Software . . . . .	5
2.2	Verwendete sowie Dimensionierung der Bauteile . . . . .	7
2.2.1	Controller . . . . .	7
2.2.2	Sender . . . . .	7
2.2.3	Hochsetzsteller . . . . .	8
2.2.4	Ultraschallkapsel . . . . .	8
2.2.5	Filter . . . . .	8
2.2.6	Empfänger . . . . .	8
2.3	Entwicklung der Software zum Betrieb des Prototypen . . . . .	9
2.3.1	Benötigte Kenntnisse . . . . .	9
2.3.2	Quellcodeentwurf . . . . .	9
<b>3</b>	<b>Messungen und Auswertung der Ergebnisse</b>	<b>12</b>
3.1	Prototyp 1 . . . . .	12
3.1.1	Senderkreis . . . . .	12
3.1.2	Empfängerkreis . . . . .	13
3.2	Prototyp 2 . . . . .	14
3.3	Ergebnisse der Forschung . . . . .	16
3.4	Fazit aus den Ergebnissen für den Auftraggeber . . . . .	17
<b>4</b>	<b>Reflektion über den Projektablauf</b>	<b>18</b>
<b>5</b>	<b>Anhänge</b>	<b>19</b>
5.1	Schaltpläne . . . . .	19
5.2	Platinenlayout . . . . .	19
5.3	Quellcode . . . . .	19

# 1 Einleitung

## 1.1 Lastenheft

### 1.1.1 Über Tinkerforge

Die Tinkerforge GmbH wurde Ende 2011 mit dem Ziel gegründet, die Handhabung eingebetteter Systeme zu vereinfachen. Das Tinkerforge Baukastensystem besteht aus aktuell fast 80 verschiedenen Modulen, die vom Anwender flexibel für die jeweilige Aufgabe kombiniert werden können. Zu den Modulen zählen diverse Sensor- Aktor- und Schnittstellenmodule, die alle über Hochsprachen wie C#, Python und Java gesteuert werden können. Tinkerforge unterstützt aktuell 17 verschiedene Programmiersprachen. Sowohl Hardware als auch die Software aller Module sind OpenSource. Die Stärke des Baukastensystems ist aus Anwendersicht die enorme Flexibilität, die Einfachheit und die Schnelligkeit mit der Projekte realisiert werden können. Es eignet sich daher besonders im Bereich Rapid Prototyping. Daher findet das Tinkerforge Baukastensystem Anwendung in vielen Forschungsinstituten, in diversen Entwicklungsabteilungen bekannter Automobilhersteller und Ingenieurbüros.

### 1.1.2 Motivation

Diese Technikerarbeit soll die Grundlage zur Entwicklung eines Entfernungssensors für das Baukastensystem bilden, der auf einer Ultraschall-Entfernungsmessung basiert. Das Baukastensystem verfügt aktuell über so einen Sensor. Bei diesem handelt es sich aber im wesentlichen um ein zugekauftes Modul, welches nicht die gewünschten Leistungen liefert. Daher soll an einem zu entwerfenden Prototypen Forschung betrieben werden, um eine eigene Lösung entwerfen zu können.

### 1.1.3 Aufgabenbeschreibung

Innerhalb dieser Arbeit soll der Entwurf eines Prototypen des Entfernungssensors und die damit verbundene Forschung durchgeführt werden. Dabei ist durch Recherche zu erarbeiten, welche Möglichkeiten zur Realisierung zur Verfügung stehen. Durch Messungen am Prototypen soll festgestellt werden, welche dieser Möglichkeiten funktional und finanziell realisierbar sind, um ein eigenes Produkt zu erstellen. Sollte im Anschluss noch die Möglichkeit bestehen, sind die Ergebnisse in ein serienreifes Modul umzusetzen.

Diverse Teilaufgaben sind zu erledigen:

- **Recherche**

Zu Anfang muss recherchiert werden, welche Möglichkeiten es gibt mittels Ultraschall eine Entfernung zu ermitteln und wie diese technisch umgesetzt werden können. Zusätzlich müssen die Techniker sich mit dem Tinkerforge Baukastensystem und seiner internen Funktionsweise vertraut machen.

- **Bauteilauswahl**

Abhängig von der gewählten technischen Umsetzung müssen geeignete Komponenten ausgewählt werden. Die Auswahl sollte auch unter dem Gesichtspunkten Preis, der Bauteilverfügbarkeit und der technischen Anforderungen erfolgen.

- **Schaltplanentwurf und Layouterstellung**

Von Tinkerforge wird das Open Source CAD Programm KiCad verwendet. Mit diesem Programm ist ein Schaltplan für den Prototypen und anschließend ein Leiterplattenlayout zu erstellen.

- **Leiterplattenbestückung**

Die erstellte Leiterplatte wird von Tinkerforge in Auftrag gegeben. Diese muss mit den gewählten Komponenten bestückt werden. Die Tinkerforge GmbH stellt dazu die notwendigen Werkzeuge bereit.

- **Einrichten und Einarbeitung in die Tinkerforge Toolchain**

Viele Softwarekomponenten werden von der Tinkerforge Toolchain automatisch generiert. Um diese Nutzen zu können muss ein Linux System eingerichtet werden. Anschließend muss sich mit der Funktionsweise des Generators und der Softwareversionsverwaltung "Git" vertraut gemacht werden.

- **Testsoftware und Forschung**

Um Messungen an der Hardware durchführen zu können gilt es Programmblöcke zu entwerfen, mit denen die einzelnen Funktionen der Baugruppen getestet werden können. So soll ermittelt werden, wie zum einen das Ultraschallsignal effektiv ausgegeben werden kann und wie sich die Signalamplitude auf die Reichweite auswirkt und zum anderen wie das zurückkommende Signal verarbeitet werden kann. Auch soll erarbeitet werden, wie gut das Signal unter verschiedenen Bedingungen verarbeitet werden kann und ob eine zuverlässige Verarbeitungsqualität ohne großen Aufwand realisierbar ist.

## 1.2 Management des Projektes

### 1.2.1 Trello

Zur zeitlichen Planung und Übersicht des Ablaufes wurde auf das Onlinetool Trello zurückgegriffen. Dieses ist ein kostenfreies, web basiertes Projektmanagementtool. Es ermöglicht den Gruppenmitgliedern gleichzeitig von verschiedenen Orten auf die Oberfläche zuzugreifen und Änderungen vorzunehmen. So kann ein Teilnehmer auch neue Termine mit Kennzeichnung der Fälligkeit für andere Gruppenmitglieder einfügen, oder bereits erledigte Aufgaben für alle abhaken. Auch können hier relevante Dokumente, die alle Gruppenmitglieder lesen sollen hochgeladen, und bei Bedarf noch kommentiert werden. Für die Dokumentation lässt sich an diesem System auch wunderbar abgleichen, zu welchen Zeitpunkten die einzelnen Aufgaben abgeschlossen wurden.

### 1.2.2 Github

Bei Github handelt es sich um einen webbasierten Online-Dienst, der Server für Software- Entwicklungsprojekte mit einer Versionsverwaltung bereitstellt. So können alle Daten nach einer Änderung im Programm wieder hochgeladen und mit einem Kommentar versehen werden. Sollte nach mehreren Änderungen ein Problem auftreten, kann einfach auf eine ältere Version zurück gegriffen und so der Fehler eingegrenzt werden. auch kann ein Projekt auf mehrere Abschnitte aufgeteilt werden, damit mehrere Personen unabhängig voneinander daran arbeiten können. Nach der Bearbeitung können die Programmteile wieder zusammengefügt werden. Dabei ist erkennbar, welche Änderungen von wem vorgenommen wurden. So können alle Vorgänge jederzeit verfolgt werden, um eine größtmögliche Übersicht zu gewährleisten. Durch das kommentieren der Änderungen kann die Nachvollziehbarkeit dieser ebenfalls deutlich gesteigert werden. Des weiteren ist diese Plattform gerade für Unternehmen wie Tinkerforge, die ihren Quellcode als Open-Source anbieten besonders reizvoll, da die Nutzer hier schnell an alle Dateien ran kommen.

## 1.3 Open Source CAD-Programm Paket zur Erstellung von elektronischen Leiterplatten

Das Open Source CAD-Programm KiCAD ist eine Anwendung zum Erstellen von Schaltplänen und elektronischen Leiterplatten. Hier lassen sich auf einfache Weise Schaltpläne erstellen und nach einer Prüfung auf fehlerfreie Verdrahtung zur vereinfachten Platinenerstellung nutzen.

### 1.3.1 Schaltplan

Beim Schaltplanentwurf gilt es auf gewisse Regeln zu achten, zu dem sollte die Übersichtlichkeit des Schaltplans nicht außer Acht gelassen werden. So sollten Beispielsweise bei Anschluss längerer Leitungen, Kondensatoren zum ausfiltern eingefangener Funksignale und anderer EMV-Belastungen, angebracht werden. Die Bauteile sollten möglichst auch so geführt und Platziert werden, wie sie im Layout auch liegen sollten, also gemeinsame Potenzial Bezugspunkte oder Ähnliches.

Im Schaltplan wurden die Schlüssel Komponente Empfänger, Sender, Hochsetzsteller und die Anschlüsse an der CPU getrennt und so Positioniert, dass die Best mögliche Übersicht dargestellt wird. Die Wichtigen Grenzpunkte wurden mit einem Null Ohm widerstand bestückt, um Etappen weise die Platine in Betrieb nehmen zu können. So konnten potentielle Gefahren vermieden werde.

### 1.3.2 Platinenlayout

Bei dem Entwurf eines Platinenlayouts gibt es viele Möglichkeiten ein Ergebniss zu erzielen. So können alle Bauteile so angeordnet werden, dass alle parallelen Bauteile sauber nebeneinander aufgereiht werden, und die in Reihe dazu liegenden Bauteile auch darunter angeordnet sind. So sähe die Platine zwar ähnlich eines Kontaktplanes aus, allerdings ist diese Variante aus Sicht der EMV nicht sonderlich empfehlenswert.

Auch können die Bauteile wie im Schaltplan in Gruppen zusammen gelegt werden und der Schaltplan auf der Platine 1:1 nachgebildet werden. Auch bei dieser Variante ergeben sich gelegentlich Probleme, was die Führung der Leitungen, und vor allem den Verlauf der Ströme angeht.

So sollte ein Augenmerk auf den Stromführenden Leitungen liegen. Je höher der Strom ist, desto breiter ist die Leitung auszulegen und sie sollte auch möglichst kurz gehalten werden um weniger EMV störungen zu erzeugen. Auch sollte die Rückführung (GND) günstigerweise als eigene Leiterschicht ausgeführt werden, um einen großen Leiterquerschnitt zu gewährleisten. So kann bei der Rückführung der Ströme auch das Risiko vermieden werden, durch die Bildung von größeren Schleifen Antennen zu erzeugen. Die GND Schicht sollte so wenig wie möglich unterbrochen werden, vorallem sind Unterbrechungen quer zur Stromflussrichtung zu vermeiden. Zusätzlich ist zu beachten, dass Kondensatoren meistens zur Verringerung von Störungen nahe digitalen Bauteilen angebracht werden sollten. Die optimale Platzierung ist direkt an den Pinns, so dass die Leiterbahn mit einem höheren Querschnitt auf den Kondensator geht, und dann mit leicht verringertem Querschnitt direkt auf die Pinne des IC-Bauteils verläuft.

## 2 Vorbereitung

### 2.1 Recherche der Funktionsweise

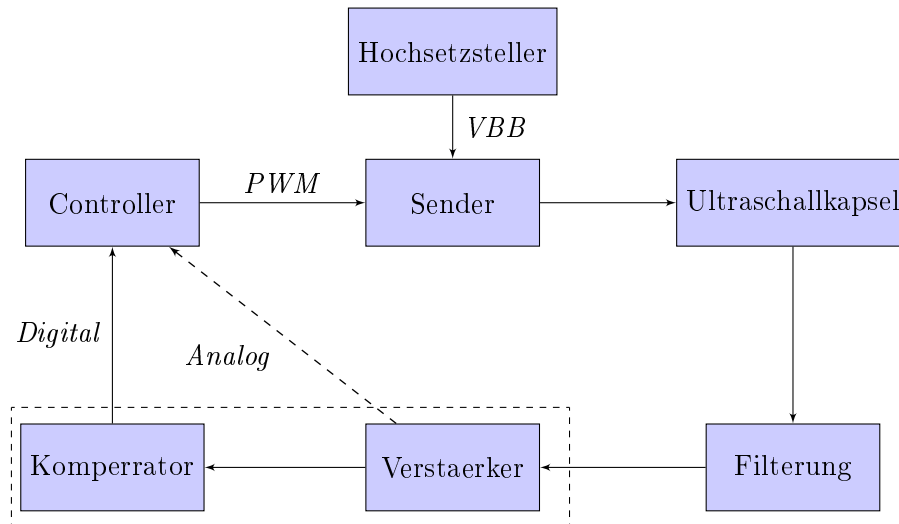


Abbildung 2.1: Blockschaltbild

Das Blockschaltbild, siehe Abbildung 2.1, zeigt eine vereinfachte Funktionsübersicht. Der Schaltplan befindet sich in den Anlagen.

#### 2.1.1 Controller

Der Controller soll eine stabile und variable PWM ausgeben können. Es werden für die Zeiterfassung und die PWM Ausgabe mehrere Timer benötigt die intern und von der Hardware gesteuert werden können. Ein Analog/Digital Wandler sollte vorhanden sein.

#### 2.1.2 Sender

Der Sender soll Schallwellen im Ultraschallbereich (40kHz) aussenden, die einen ausreichenden Schalldruck haben, um von Hindernissen, die bis zu mehrere Meter entfernt sind ein Echo erhalten zu können. Dafür muss die Lautsprecherkapsel mit einem sinusähnlichen Signal angesteuert werden, dessen Amplitude angemessen hoch ist, um den gewünschten Schalldruck zu erzeugen. Der Nachteil ist, dass der Mikrocontroller nur auf seine Spannungsebene von 3,3V begrenzt ist und durch höhere Spannungen zerstört würde. Um höhere Spannungen an der Ultraschallkapsel realisieren zu können, muss ein weiteres Bauteil dazwischen geschaltet werden. Dieses zusätzliche Bauteil dient als Trennung zwischen den 3,3V des Mikrocontrollers und der höheren Spannungsebene der Lautsprecherkapsel. Dabei wird darauf zu achten sein, dass dieser Schalter schnell genug arbeitet, um die 40kHz (Frequenz von Ultraschall) auch sauber schalten zu können.

#### 2.1.3 Hochsetzsteller

Der Hochsetzsteller dient dazu, aus den 5V Versorgungsspannung eine (für die Versuche variable) höhere Spannung für den Sendebetrieb zu schaffen. So kann der Schalldruck der ausgegeben wird erhöht werden (größere Reichweite/größeres Rücksignal) siehe Kapitel: Messungen

Die Funktionsweise des Hochsetzstellers (Spannungspumpe/Aufwärtswandler/Aufwärtsregler) ist relativ simpel und findet in vielen Bereichen Anwendung. Grundsätzlich wird eine Induktivität

in Reihe mit einer Freilaufdiode vor einen Ladekondensator geschaltet. Dieser liegt parallel zum Ausgang. Zwischen der Spule und der Diode ist ein Schalter angeschlossen, der die Spule gegen Masse schaltet. So lädt sich die Spule bei Betätigung des Schalters auf (durch den Stromfluss entsteht ein Magnetfeld) und beim Öffnen steigt die Spannung am sekundären Ende der Spule, durch das zusammenbrechende Magnetfeld, an und lädt den Kondensator auf. Dieser Vorgang wird wiederholt, bis der Kondensator so weit aufgeladen ist, dass die Ausgangsspannung den gewünschten Wert hält. Dann wird die Schaltfrequenz auf das Mindestmaß verringert, um den Wert zu halten. Natürlich ist die mögliche Ausgangsspannung nicht unbegrenzt über das Schaltspiel regelbar, sondern ist auch von den Baugrößen der Bauteile abhängig.

#### 2.1.4 Ultraschallkapsel

Für den Sender (und Empfänger) wurden auf Piezomodulen basierende Ultraschallkapseln verwendet, weil Sie die Möglichkeit haben ein Signal auszugeben sowie zu empfangen und sind in Kleinbauform erhältlich. Wichtig sind bei der Überprüfung das nach schwingen der Kapsel, nach Ende des zu sendenden Signals, die Empfindlichkeit auf eingehende Störfrequenzen und die zu erreichende Reichweite mit verschiedenen Signalstärken. Es sollte auch überprüft werden welche Kapseln sich für ein stand-alone Betrieb am geeignetsten ist.

#### 2.1.5 Filter

Die Filterschaltung soll am Eingang des Empfängers mögliche Störfrequenzen herauszufiltern um eine Verfälschung der gemessenen Strecke zu vermeiden, da die Ultraschallkapsel nicht ausschließlich Ultraschallsignale aufnimmt.

#### 2.1.6 Empfänger

Im Empfangsbetrieb werden die zurückkommenden Schallsignale, die auf die Ultraschallkapsel treffen in Sinusförmige Spannungssignale umgewandelt. Die Amplitude dieser Signale hängt von dem Schalldruck der empfangenen Signale ab und ist deutlich niedriger als die Amplitude der gesendeten Signale. Diese Signale müssen anschließend verstärkt werden, um sie mit dem Mikrocontroller auswerten zu können. Zur Vereinfachung der Auswertung macht es auch Sinn, das eingehende analoge Signal in ein Digitales Signal umzuwandeln, dies kann einiges an Verarbeitungsaufwand einsparen. Um die Qualität der gemessenen Entfernung besser zu beurteilen sollte auch das Analogesignal ausgewertet und verglichen werden mit dem umgewandelten Digitalen Signal. Es dient unter anderem nämlich dazu wie die spätere Software auszusehen hat, denn die Auswertung eines Analogensignals erfordert mehr Programmieraufwand als die Auswertung eines Digitalensignals. Um ein Digitales Signal auszuwerten reicht es im Grunde genommen auf den ersten Impuls zu warten, die Wartezeit wird in die Entfernungsmessung mit einbezogen und somit ergibt sich die Entfernung. Bei der Analogen Auswertung muss mit dem gesendet mit dem empfangenen Signal verglichen und überprüft ob es sich um die 40kHz handelt oder nur eine Störung ist. Mit der Zeit, die zwischen dem gesendeten Signal und dem empfangenen Signal vergeht, muss letztendlich der Abstand zwischen dem Sensor und dem Hindernis berechnet werden.

#### 2.1.7 Besonderheit der Software

Um die Lesbarkeit des Programms zu fördern wurden APIs (Application Programming Interface) verwendet die schon das passende Register ansteuern anstatt per Hand aus dem Reference Manuel jedes einzelne Register mühevoll rauszusuchen. Das erlaubt Programme leichter zu warten und zu optimieren. Um .mode zu bearbeiten und nach der erfordernten Funktion zu konfigurieren reicht es die *xmc1-gpio.h* zu betrachten. Somit reicht es in das Feld .mode=gewünschte Funktion ein zu eintragen ,in diesem Beispiel ist es *XMC GPIO MODE NPUT PULL UP*. Siehe 2.2 : Die API für



ein Programmbeispiel anhand einer PULL-UP Initialisierung.

```
1 Code vom Infineon Prozessor XMC 1xxx48 mit der API:
2 void pin_in_pullup_init(XMC_GPIO_PORT_t *const port, const uint8_t pin)
3 {
4     const XMC_GPIO_CONFIG_t pin_in_config = {
5         .mode = XMC_GPIO_MODE_INPUT_PULL_UP, /**< Defines the direction
6         and characteristics of a pin */
7         .output_level = XMC_GPIO_OUTPUT_LEVEL_HIGH, /**< Defines output level of
8         a pin */
9         .input_hysteresis = XMC_GPIO_INPUT_HYSTERESIS_STANDARD /**< Defines input
10        pad hysteresis of a pin */
11     };
12     XMC_GPIO_Init(port, pin, &pin_in_config);
13 /*
14  * Initializes input / output mode settings like, pull up / pull down devices, push
15  * pull / open drain, and pad driver mode.
16  * Also configures alternate function outputs and clears hardware port control for
17  * selected \a port and \a pin .
18  * It configures hardware registers Pn_IOCR, Pn_OUT, Pn_OMR, Pn_PDISC and Pn_PDR.\n
19  */
20 }
```

**Abbildung 2.2:** Die API für ein Programmbeispiel anhand einer PULL-UP Initialisierung

In der 2.3 xmc1-gpio.c, veranschaulicht gut was hinter API passiert. Es erfordert viel Einarbeitung in den Prozessor um die einzelnen Funktionsaufrufe zu verstehen, somit erleichtert die API immens die Programmierarbeit und ist zeitgleich Anwenderfreundlich.

```

1 void XMC_GPIO_Init(XMC_GPIO_PORT_t *const port, const uint8_t pin, const
   XMC_GPIO_CONFIG_t *const config)
2 {
3     XMC_ASSERT("XMC_GPIO_Init: Invalid port", XMC_GPIO_CHECK_PORT(port));
4     XMC_ASSERT("XMC_GPIO_Init: Invalid mode", XMC_GPIO_IsModeValid(config->mode));
5     XMC_ASSERT("XMC_GPIO_Init: Invalid input hysteresis",
6         XMC_GPIO_CHECK_INPUT_HYSTERESIS(config->input_hysteresis));
7
8     /* Switch to input */
9     port->IOCR[pin >> 2U] &= ~(uint32_t)((uint32_t)PORT_IOCR_PC_Msk << (
10         PORT_IOCR_PC_Size * (pin & 0x3U)));
11
12     /* HW port control is disabled */
13     port->HWSEL &= ~(uint32_t)((uint32_t)PORT_HWSEL_Msk << ((uint32_t)pin << 1U));
14
15     /* Set input hysteresis */
16     port->PHCR[(uint32_t)pin >> 3U] &= ~(uint32_t)((uint32_t)PORT_PHCR_Msk << ((
17         uint32_t)PORT_PHCR_Size * ((uint32_t)pin & 0x7U)));
18     port->PHCR[(uint32_t)pin >> 3U] |= (uint32_t)config->input_hysteresis << ((
19         uint32_t)PORT_PHCR_Size * ((uint32_t)pin & 0x7U));
20
21     /* Enable digital input */
22     if (XMC_GPIO_CHECK_ANALOG_PORT(port))
23     {
24         port->PDISC &= ~(uint32_t)((uint32_t)0x1U << pin);
25     }
26
27     /* Set output level */
28     port->OMR = (uint32_t)config->output_level << pin;
29
30     /* Set mode */
31     port->IOCR[pin >> 2U] |= (uint32_t)config->mode << (PORT_IOCR_PC_Size * (pin & 0
32         x3U));
33 }

```

Abbildung 2.3: *xmc1\_gpio.c*

xmc1-gpio.c sowie xmc-gpio.h lassen sich in der XMC Library im Anhang finden.

## 2.2 Verwendete sowie Dimensionierung der Bauteile

### 2.2.1 Controller

Der Infineon XMC 1xxx48 gehört zu der Familie der ARM Cortex -M0 Prozessoren und ist ein 32-bit Industrial Microcontroller und wird mit 48MHz externer Clock betrieben. Die 48 im Namen des Prozessors steht für die Anzahl der Pins. Der interne Timer läuft mit 96Mhz. Unter anderem besitzt die CCU4 von dem Prozessor 2x4 16 bit Timer. Außerdem bietet der XMC einen 12 bit A/D Wandler, welcher für die Analogmessung eine viel genauere Auflösung bieten kann als ein 8 bit A/D Wandler. Die Betriebsspannung des Prozessors beträgt 3,3V. Die Auswahl des Controller wurde getroffen, weil der standardmäßig auch schon bei Tinkerforge eingesetzt wird und uns für den Prototypen vorgegeben wurde.

### 2.2.2 Sender

Um eine Trennung zwischen des CPU Kreises (3,3V) und des Hochsetzstellers (6V-20V) dessen Aufgabe ist eine höhere Spannung an der Ultraschallkapsel zu erzielen, dafür eignete sich das IC A5950 (Voll Brücke). Wie in Abbildung ?? zu sehen ist, kann die H-Brücke eine angeschlossene Last mit der vom Hochsetzstellers erzeugten Spannung versorgen, deren Frequenz über das Signal an der Anschluss Phase der CPU vorgegeben werden an Out1 und Out2 wird das Ausgangssignal abgegriffen. Für die genaue Beschaltung des ICs siehe Anhang Datenblatt "Schimatic A5950".

### 2.2.3 Hochsetzsteller

Tinkerforge nutzt schon eine Variante eines Hochsetzstellers auf ihren Platinen somit konnte der Aufbau und die Bauteilauswahl nur auf die Variable Ausgangsspannung erweitert werden. Die Standardschaltung, von Tinkerforge, wurde an dem Eingang der Feedbackspannung mit einem Potentiometer erweitert um die gewünschte Variablespeisung zubekommen.

$$R1 = R2 * \left( \frac{V_{out}}{1,23} - 1 \right) \Rightarrow V_{out} = \left( \frac{R1}{R2} + 1 \right) * 1,23$$

### 2.2.4 Ultraschallkapsel

Für den Prototypen wurden mehrere Kapseln verschiedener Hersteller bestellt. Dieses geschah um Unterschiede der verschiedenpreisigen Bauteile zu ermitteln und festzustellen, welches Preissegment die nötige Qualität für die vorliegende Anwendung erfüllt.

### 2.2.5 Filter

Die Filterschaltung wurde mit einem Hochpassfilter (CR Glied) bestückt bestehend aus C12 und R5 um unerwünschte Signalanteile mit Frequenzen, die unter 40kHz liegen, zu unterdrücken. Der Widerstand wurde nach der e24 Reihe ausgewählt. Die Kapazität des Kondensators C12 wurde an die Grenzfrequenz von 40 kHz und den Widerstand angepasst.

$$C12 = \frac{1}{2 * \pi * f_g * R} \Rightarrow \frac{1}{2 * \pi * 40kHz * 100K\Omega} \approx 40pF$$

Anhand der Berechnung wurde ein Kondensator mit 39pF genommen für den Hochpassfilter

### 2.2.6 Empfänger

Die Abbildung ?? zeigt die Empfängerschaltung. Durch diese Verschaltung von Operationsverstärkern(OPVs) wird das ankommende Sinusförmige Signal verstärkt und in ein digitales Signal umgewandelt. Für die Verstärkung der Amplitude so wie der Umwandlung des analogen Signals in ein Rechtecksignal mit 40 KHz standen zwei Operationsverstärker zur Auswahl, LT1112 mit einem Stückpreis von 4,80€ und den TLC272 mit einem Stückpreis von 0,88€ trotz der besseren Performance, wurde der TLC272 für die Prototypen ausgewählt um die preislich günstigen Möglichkeiten zu prüfen. Die Versorgungsspannung der OPV's von 3,3V wird durch den Kondensator C16 (EMV Störfilter) stabilisiert.

Für die Verstärkung der Amplitude ist der Operationsverstärker TLC272 U2B als nicht invertierender Verstärker geschaltet. Wenn eine Gleichspannung anliegt, wirkt der Kondensator (C10) in der Operationsverstärkerschaltung als Impedanzwandler, also mit einer Verstärkung von eins, geht nun die Eingangsfrequenz hoch, nimmt der Widerstand des Kondensators (C10) ab, somit beginnt der Operationsverstärker auch zu verstärken, und zwar mit zunehmender Verstärkung, bis irgendwann die Impedanz des Kondensators vernachlässigt werden kann und die Verstärkung nur noch durch das Verhältnis der Widerstände beeinflusst wird, R6 ist zudem notwendig um das Schwingen der Amplitude zu verhindern, somit kann die Verstärkung mit folgender Formel berechnet werden:

$$V_u = \frac{R6 + R8 + R12}{R6}$$

Für die Umwandlung des Analogen Signales in ein Digitales wurde der Operationsverstärker TLC272 U2C als Komparator geschaltet. Beim Auftreten von Differenzen zwischen den eingangs Signalen, wechselt der Ausgang des Komparators zwischen Low (0 Volt) auf High (3,3 Volt).

Die Referenz ( $U_{ref}$ ). Spannung wird durch den Spannungsteiler R9 und R8 bestimmt.

$$U_{ref} = \frac{U_{ges} * R9}{R8 + R9} \Rightarrow \frac{3,3V * 120K\Omega}{100K\Omega + 120K\Omega} = 1,8V$$

## 2.3 Entwicklung der Software zum Betrieb des Prototypen

### 2.3.1 Benötigte Kenntnisse

Die Variable  $x=0-3$  dient als Index. Um das Programm zu erstellen werden Kenntnisse z.B. für die Timer CCU4x sowie deren Slices CC40-43 benötigt. Die Funktionen der CCU4x wird in der Reference Manuel beschrieben, siehe dazu Anhang, um die Timer zu Initialisieren wird die XMC Lib benötigt die eine fertige API mit bringt, siehe Kapitel Besonderheiten der Software, so wird nur noch die API mit den richtigen Parametern beschrieben um die Timer lauffähig zu kriegen.

### Durchgeführte Berechnungen

Auch für die Programmierung waren diverse Berechnungen notwendig. So musste zur Erzeugung der Ultraschallimpulse ein Pulsweitenmoduliertes Rechteck signal geschaffen werden. Dafür wurde ein Timer der CCU4x auf einen Takt von 40kHz eingestellt. So musste bei einem Timertakt von 96MHz eine Periodendauer von 2400 Takten, siehe unten die Berechnung und ein Compare-Wert von 1200 Takten eingestellt werden. Im Zählvorgang des Timers wird der Ausgang nach Erreichen des Compare-Wertes auf 1, und nach Erreichen der Periodendauer wieder auf 0 gesetzt. Dadurch ergibt sich eine Periodendauer von 25us, was einer Frequenz von 40kHz entspricht.

Auch zur Erfassung der Zeit, die vergeht bis das Echo des Ultraschall-Impulses zurück kommt wird über einen Timer der CCU4x erfasst.

$$\text{Periodendauer} = \frac{96\text{MHz}}{40\text{kHz}} = 2400 \quad \text{Compare - Wert} = \frac{2400}{2} = 1200$$

### Einstellung im Programm

### 2.3.2 Quellcodeentwurf

Um Fehler in der Zeitnahme zu verringern sollten Zeitwerte, die ausgelesen werden sollen direkt am Anfang des Interrupts in Variablen gespeichert werden und nicht erst innerhalb anderer Anweisungen oder Schleifen, da das schon deutliche Abweichungen mit sich bringt.

**Programmstruktur:** Anstatt alles in der main.c an ,siehe Abbildung 2.4, Programmcode zu verfassen was bei sehr komplexen Programmen schnell zu unübersichtlichkeit führt hat das Auslagern den Vorteil das der Quellcode Logisch getrennt werden kann und so einer verschlankerung des Codes mit sich bringt. Somit stehen in der Main jetzt deutlich, welche Funktionen beim Starten initialisiert werden, und welche Unterprogramme regelmäßig aufgerufen werden. Auch vereinfacht diese Struktur gerade bei Prototypen das Testen der Funktion, so kann im Falle einer fehlerhaften Funktion einfach der Aufruf auskommentiert werden um zu testen, ob der Fehler wirklich von der Funktion herrührt. Dadurch müssen nicht etliche Zeilen Programmcode der Funktion auskommentiert werden, wodurch schnell Fehler entstehen könnten, durch übriggebliebene Zeichen, oder gar beim entfernen der Auskommentierung gelöschte Zeichen.

```

1 #include <stdio.h>
2 #include <stdbool.h>
3 #include "bricklib2/logging/logging.h"
4 #include "bricklib2/bootloader/bootloader.h"
5 #include "communication.h"
6 /****Eigene Include Dateien*****/
7 #include "configs/config.h"
8 #include "system_timer/system_timer.h"
9 #include "a16pt.h"
10 int main(void)
11 {
12     logging_init();
13     logd("Start Distance US V2 Bricklet/n/r"); //For the Debugmodus
14     communication_init(); //Function call
15     a16pt_init(); //Function call
16     while(true)
17     {
18         a16pt_tick(); //Function call
19         bootloader_tick(); //Function call
20         communication_tick(); //Function call
21     }
22 }
23 }

```

**Abbildung 2.4:** *main.c*

Um die Function call zu verstehen muss die Abbildung 2.5: a16pt.h näher betrachtet werden. In der der a16pt.h werden die Funktionen definiert die dann in der main.c aufgerufen werden und die Funktionsanweisungen stehen dafür in der a16pt.c.

```

1 #ifndef A16PT_H
2 #define A16PT_H
3 #include <stdint.h>
4 void a16pt_init(void); //Functional definition
5 void a16pt_tick(void); //Functional definition
6 uint16_t a16pt_get_distance(void); //Functional definition

```

**Abbildung 2.5:** *a16pt.h*

In der Abbildung 2.6 :a16pt.c ,ein Teilausschnitt des Codes, werden die für die Entfernungsmessung notwendigen Funktionen und die Interrupt anweisungen, in dem Fall die IRQ21, abgearbeitet außerdem werden die Timer Synchron abgeschaltet und aus experementellen gründen wurde ein weitere Impuls generiert um zu beobachten wie sich das nachschwingen verhält bei einer längeren Kurzschlusszeit an der Ultraschallkapsell.

```

1  /*****Interrupt_Funktionen*****/
2
3  void IRQ_Hdlr_21(void) // Compare Interrupt counter 10
4  {
5
6      // Disable IRQs so we can't be interrupted
7      __disable_irq();
8
9      // Set CCU trigger to low, otherwise ccu counter is restarted
10     XMC_SCU_SetCcuTriggerLow(XMC_SCU_CCU_TRIGGER_CCU41);
11
12     // Stop slice 2
13     XMC_CCU4_SLICE_StopClearTimer(CCU41_CC40);
14
15     // For slice 1 we wait until PWM is run through (to get exactly 10 pwm peaks on
16     // P4_4 and P4_6)
17     while(XMC_CCU4_SLICE_GetTimerValue(CCU41_CC42) > compare_1) {
18
19         __NOP();
20     }
21
22     //new pin configuration
23     const XMC_GPIO_CONFIG_t pin_out_config = {
24         .mode           = XMC_GPIO_MODE_OUTPUT_PUSH_PULL,
25         .output_level    = XMC_GPIO_OUTPUT_LEVEL_HIGH,
26     };
27
28     XMC_GPIO_Init(P4_6, &pin_out_config);
29     //Creat a high impulse
30     for(s=0; s<50; s++)
31     {
32         __NOP();
33     }
34     // Stop slice 0
35     XMC_CCU4_SLICE_StopClearTimer(CCU41_CC42);
36
37     //pin configuration back to the PWM-Mode
38     const XMC_GPIO_CONFIG_t gpio_out_config1 = {
39         .mode           = XMC_GPIO_MODE_OUTPUT_PUSH_PULL_ALT9,
40         .input_hysteresis = XMC_GPIO_INPUT_HYSTERESIS_STANDARD,
41         .output_level    = XMC_GPIO_OUTPUT_LEVEL_LOW,
42     };
43
44     XMC_GPIO_Init(P4_6, &gpio_out_config1);
45     // Enable IRQs again
46     __enable_irq();
47
48 }

```

Abbildung 2.6: *a16pt.c*

# 3 Messungen und Auswertung der Ergebnisse

Für die Messungen wurden Laufe des Projekts zwei Versionen an Prototyp-Platinen entworfen, an denen Messungen und Verbesserungen vorgenommen wurden.

## 3.1 Prototyp 1

Bei dem ersten Prototyp wurden die Sendereinheit und die Empfängereinheit auf getrennten Platinen aufgebaut. So bestand die Möglichkeit, den Senderkreis und den Empfängerkreis getrennt zu untersuchen, ohne dass sich elektrische Signale der beiden Schaltkreise überlagern konnten.

### 3.1.1 Senderkreis

Zu erst wurden Signale direkt an der CPU gemessen, um sicher zu stellen, dass die Einstellungen im Programm auch die gewünschten Ausgaben zur Folge haben, und keine Gefährdung der Bauteile entsteht. Um das Signal für die Entfernungsmessung zu generieren wurde der Mikrocontroller so programmiert, dass zehn Impulse mit einer Frequenz von 40kHz ausgegeben werden. Danach erfolgt eine Pause, um das zurückkehrende Signal abzuwarten und auszuwerten.

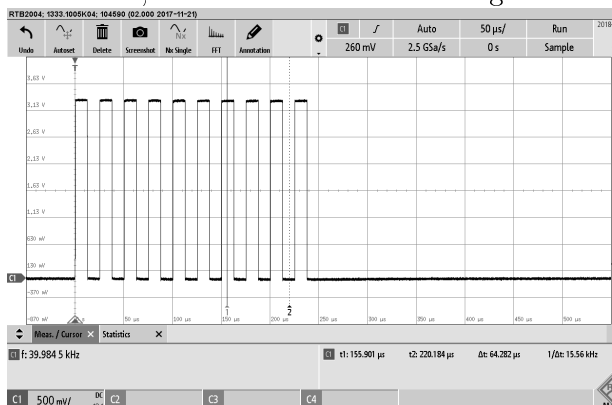


Abbildung 3.1: PWM-Burst auf 40kHz Basis an der CPU

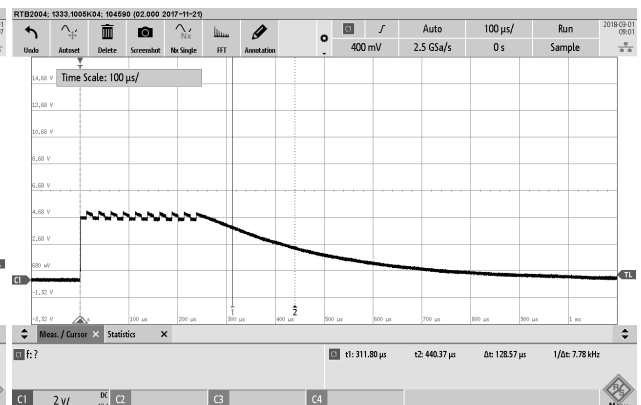
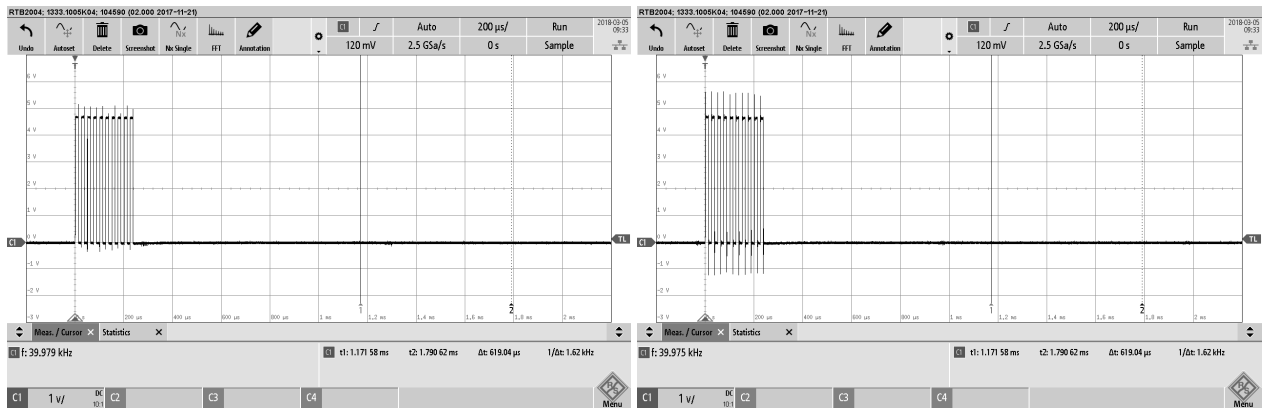


Abbildung 3.2: PWM Ausgabe über einen Hi-Side

In der Abbildung 3.1 ist zu sehen, dass der gewünschte Burst aus zehn Impulsen mit einer Periodendauer von jeweils 25µs vom Mikrocontroller generiert wurde. Diese Messung wurde auch vorgenommen, um zu überprüfen, wie sich das Signal durch die eingesetzten Bauteile verändert.

Die Abbildung 3.2 zeigt, wie das Ausgangssignal nach einer Hi-Side aussieht. So wird zwar im Takt des PWM-Signals geschaltet, allerdings fehlt es an einem Gegenpool, um das Potential in den Schaltphasen wieder auf Null zu ziehen. Dadurch bleibt die Spannung während des Schaltens immer auf einem erhöhten Pegel und sinkt erst nach Ende des PWM-Signals langsam ab. Dadurch kann natürlich keine vernünftige Ausgabe am Lautsprecher erzeugt werden, denn ohne deutliche Potentialunterschiede kann dieser auch nicht in Schwingungen versetzt werden. Der ausgegebene Schalldruck würde maximal für kürzeste Entfernungsmessungen reichen, wenn überhaupt und dann würde das zurückkommende Signal noch von der abklingenden Spannung des Hi-Side überlagert. Somit ist dieser Aufbau nicht operabel.

Um die Spannung nicht nur auf einen Hi-Pegel, sondern auch auf einen LOW-Pegel schalten zu können wurde danach auf eine Halbbrücke gewechselt. Mit dieser lässt sich der Ausgang, über zwei durch das PWM-Signal gesteuerte MOSFETs, sauber auf Hi- oder LOW-Pegel schalten. Mit der verwendeten Halbbrücke ergab sich die Abbildung 3.3



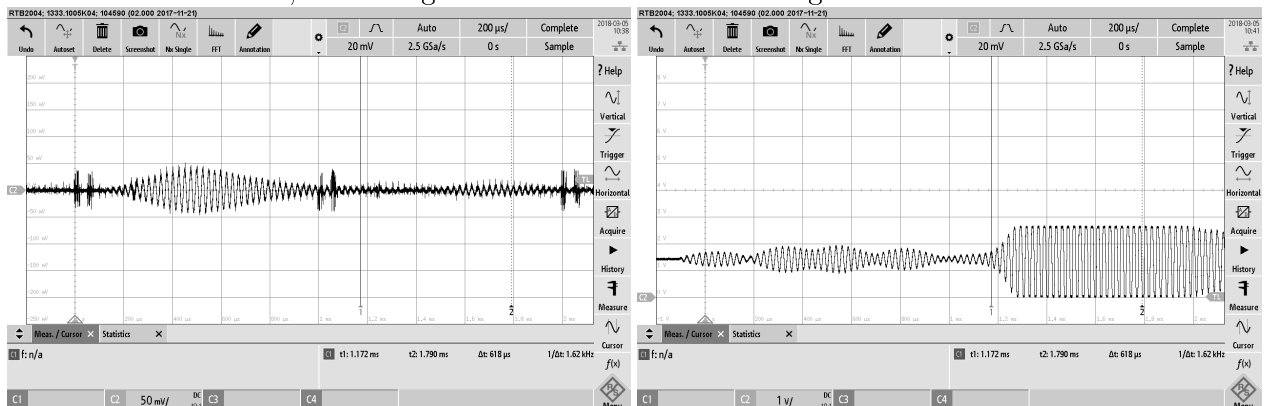
**Abbildung 3.3:** *PWM Ausgabe über eine Halbbrücke*

**Abbildung 3.4:** *Ausgabe der PWM an der Ultraschallkapsel*

Es zeigt sich, dass das Signal nach der Erweiterung auf eine Halbbrücke wieder wie das von der CPU ausgegebene PWM-Signal 3.1 aussieht, nur dass die Amplitude wie geplant höher ausfällt. Somit kann die Höhe der Amplitude über die Spannungspumpe variiert werden um die Stärke des ausgegebenen Signals zu verändern, ohne die CPU durch die höhere Spannung zu beschädigen. Wie in der Abbildung 3.4 zu entnehmen ist, entstehen durch die angeschlossene Ultraschallkapsel höhere Spannungsimpulse im Einschaltmoment.

### 3.1.2 Empfängerkreis

Die Platinen des Sender- und Empfängerkreises wurden gemeinsam auf einer Halterung montiert, so dass die Ultraschallkapseln zum senden und empfangen der Signale nebeneinander befestigt werden konnten. Ziel war es, durch verschieben eines Hindernisses die Signaländerungen an den Platinen beobachten zu können, ohne die gesamten Messaufbauten bewegen zu müssen.



**Abbildung 3.5:** *Signal Empfang*

**Abbildung 3.6:** *Signal nach Verstärkung*

Die Abbildung 3.5 zeigt das Signal, das direkt am Empfänger zu messen war. Hier sind verschiedene vorerst nicht zuordenbare Signale zu sehen. Allein aus diesem Bild lässt sich aber keine Aussage zu den Signalen machen. Fest steht nur, dass ebenfalls Signale die nicht der gewünschten Frequenz entsprechen, vom Empfänger aufgenommen werden. Dies gilt es natürlich schnellst möglich auszu-merzen, um unerwünschte Störungen zu vermeiden. Die Abbildungen 3.6 und 3.7 zeigen den Verlauf des Signals nach der Filterung und Verstärkung in zwei verschiedenen Zeitaufösungen. Dabei entspricht 3.6 den ersten drei Kästchen von 3.7 und dient um darzustellen, dass die Verstärkung eine maximale Aussteuerung von 3,3V nicht überschreitet.



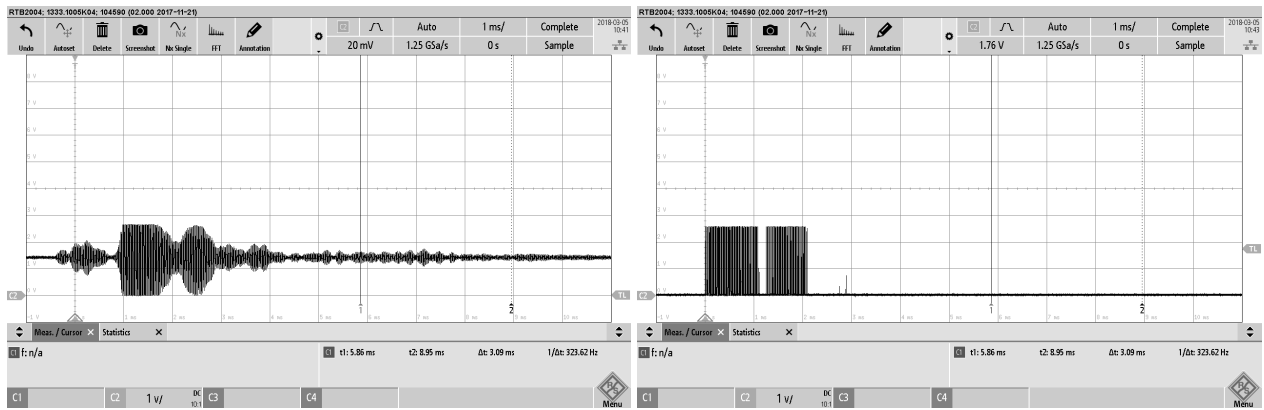


Abbildung 3.7: Signal nach Verstärkung2

Abbildung 3.8: Signal nach Komparator

Nach dem das Signal den Komparator passiert hat, ergibt sich das Bild wie in Abbildung 3.8 zu sehen ist. Bei einem Vergleich mit dem Signal nach der Verstärkung 3.7 wird sichtbar, dass der Komparator nur Signale, die über seinem Schwellwert liegen, durchschaltet. Die Aufteilung in zwei Signalblöcke in den Abbildungen kommt daher, dass der erste Block das Signal der Sender-Kapsel ist, das direkt beim Senden seitlich auf die Empfänger-Kapsel abgestrahlt wurde. Der zweite Block ist bereits das Echo, das vom 20cm entfernten Hindernis zurückgeworfen wurde.

## 3.2 Prototyp 2

Bei der zweiten Prototyp-Version wurden der Sender- und der Empfängerkreis auf einer Platine aufgebaut und es wurde nur noch eine Ultraschallkapsel für beide Anwendungen vorgesehen. Um bei diesem Aufbau, einen fehlerfreien Betrieb der verwendeten voll gesteuerten Halbbrücke sicherzustellen, wurden durch den Mikrocontroller zwei getrennte PWM-Signale generiert, die wie in Abbildung 3.9 zu sehen ist, durch Lücken getrennt sind. So ist sichergestellt, dass auch trotz Verzögerungen im Schaltbetrieb der Halbleiter, keine Kurzschlüsse entstehen können.

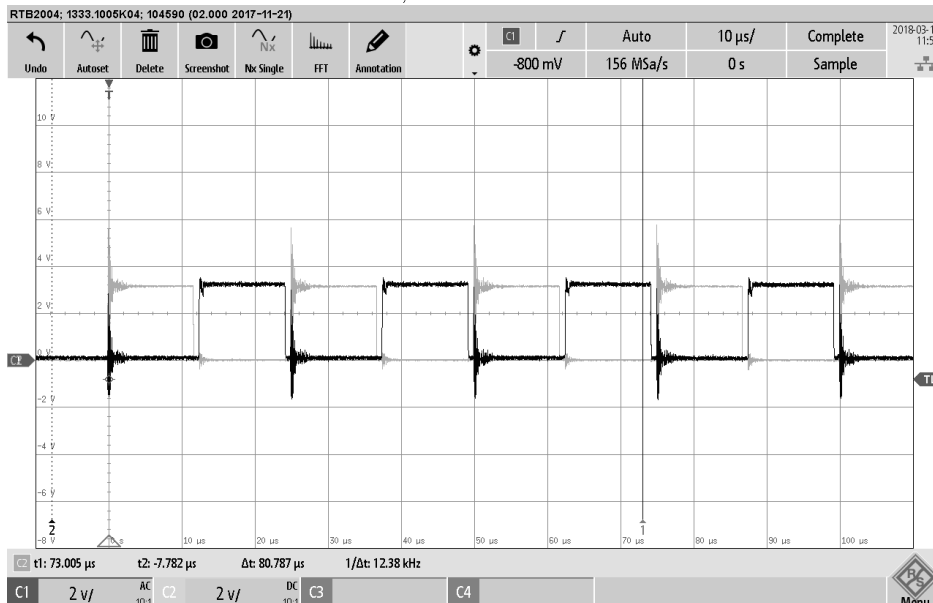


Abbildung 3.9: Verlauf der zwei generierten PWMs für den Betrieb der voll gesteuerten Halbbrücke

Nachdem dieser Betrieb sichergestellt war, wurden Messungen am Verstärker (obere Linie), und am Komparator (untere Linie) vorgenommen. Dabei wurde die Verstärkung so eingestellt, dass unerwünschte Störungen gerade so nicht vom Komparator weitergegeben wurden. Die Spannung für

den Sendebetrieb wurde für die Versuche zwischen 5V und 20V variiert, um betrachten zu können, wie sich das auf die Reichweite und Genauigkeit der Messungen auswirkt. Als Hindernis wurde bei allen Versuchen eine glatte Holzplatte der Maße 40x60cm verwendet und in einem Abstand von ein bis fünf Metern von der Ultraschallkapsel aufgestellt. In den Abbildungen 3.10 bis 3.13 sind die Ergebnisse einer Messreihe mit einer Spannung von 5V für den Sendebetrieb dargestellt. Die Ansicht wurde so eingestellt, dass zwei Sendepulse zu sehen sind. Dadurch wird deutlicher, welches die Sendepulse sind, und welches die von der Entfernung abhängigen Echos sind.

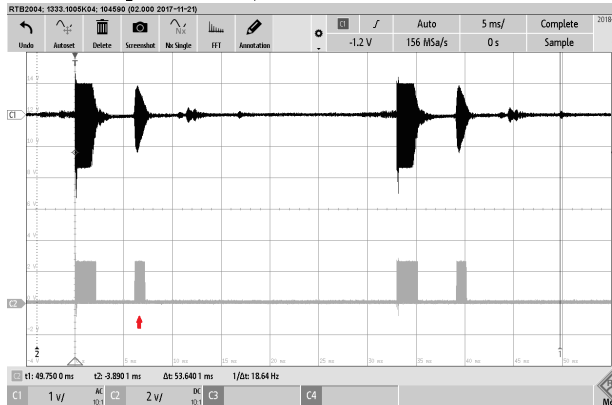


Abbildung 3.10: Signalverlauf bei 5V auf 1m Abstand

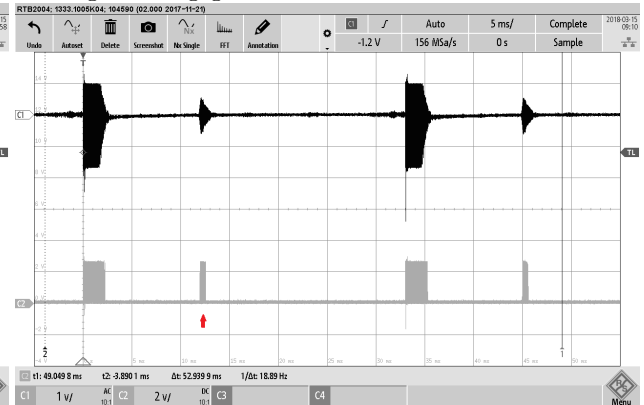


Abbildung 3.11: Signalverlauf bei 5V auf 2m Abstand

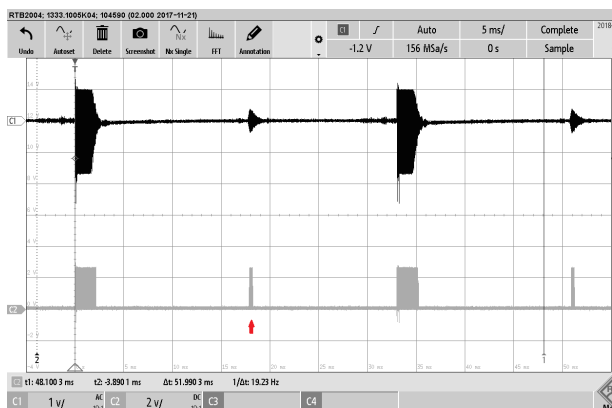


Abbildung 3.12: Signalverlauf bei 5V auf 3m Abstand

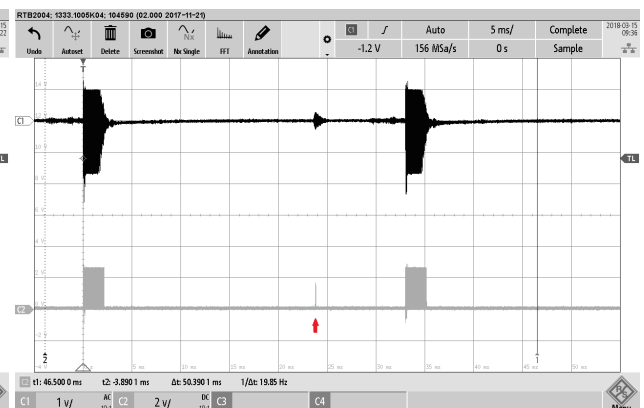


Abbildung 3.13: Signalverlauf bei 5V auf 4m Abstand

Bei den Abbildungen ist zu sehen, dass das Echo-Signal mit zunehmender Entfernung immer schwächer wird. Bei einer Entfernung von fünf Metern (Abbildung 3.13) wird das Echo-Signal so schwach, dass die Signalstärke nach dem Komparator nicht mehr für eine eindeutige Auswertung über den Mikrokontroller ausreicht. Nachfolgend sind die Abbildungen einer Messreihe mit verschiedenen Spannungseinstellungen für den Sendebetrieb zu sehen. Anhand dieser Messreihe soll dargestellt werden, welchen Einfluss die eingestellte Spannung im Sendebetrieb auf die Reichweite des Ultraschallsignals hat. Für die Darstellung wurden die Messungen bei 5 Meter Abstand ausgewählt.

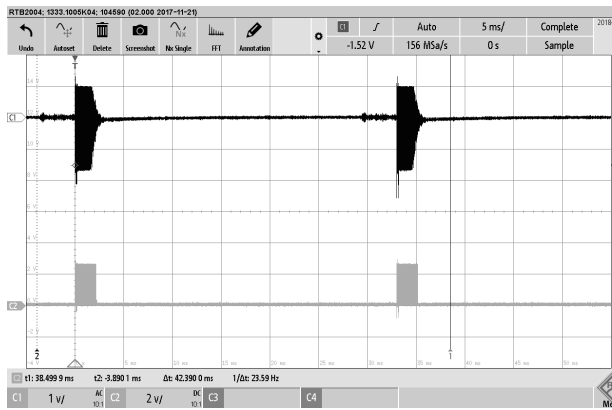


Abbildung 3.14: Signalverlauf bei 5V auf 5m Abstand

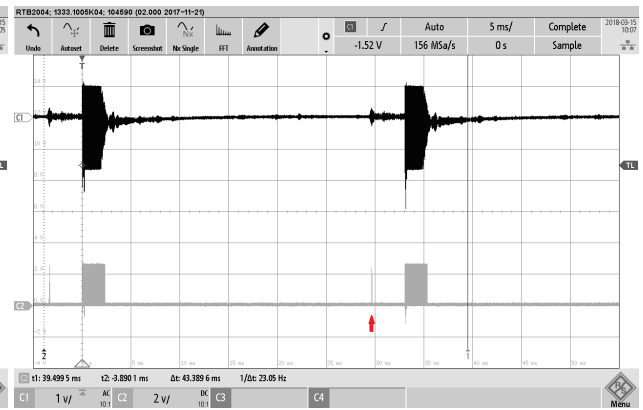


Abbildung 3.15: Signalverlauf bei 10V auf 5m Abstand

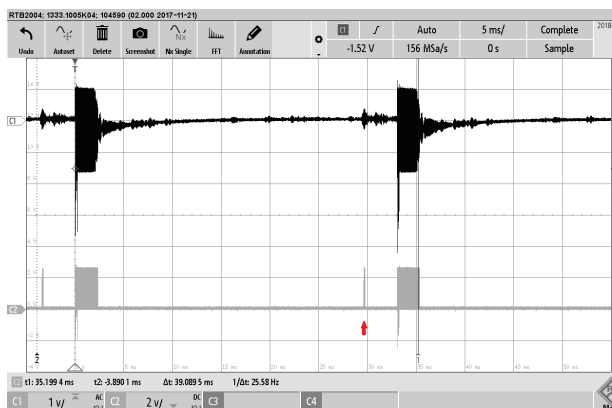


Abbildung 3.16: Signalverlauf bei 15V auf 5m Abstand

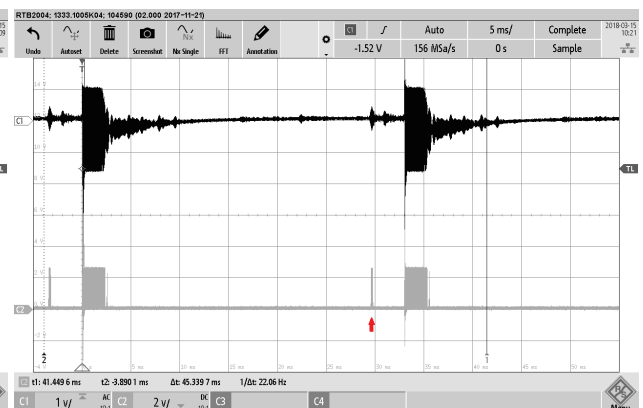


Abbildung 3.17: Signalverlauf bei 20V auf 5m Abstand

Bei Vergleich der Abbildungen 3.14 und 3.15 mit den Abbildungen 3.16 und 3.17 ist zu sehen, dass bei einem Abstand von 5 Metern erst bei einer Sendespannung von über 10V am Komparator ein über den Mikrokontroller auswertbares Signal vorhanden ist.

### 3.3 Ergebnisse der Forschung

Aus den Versuchen und Messungen lassen sich mehrere Aussagen treffen.

Als erstes, eine Ultraschall-Entfernungsmessung ist mit wenigen Bauteilen, sowohl als zwei Kapsel Variante, als auch als ein Kapsel Variante ist durchführbar. Bei der ein Kapsel Variante ist darauf zu achten, dass der Verstärker eine ausreichende Spannungsfestigkeit besitzt, um nicht durch das Sendersignal zerstört zu werden. Auch ist wichtig, dass bei der Erzeugung des PW-Modulierten Ausgangssignals die MOSFETs so angesteuert werden, dass das MOSFET, welches die High-Impulse schaltet, genug Zeit zum abschalten hat, bevor das MOSFET, das das LOW-Signal schaltet, einschaltet, und umgekehrt um Kurzschlüsse an dieser Stelle zu vermeiden.

Eine Auswertung des Echo-Signals ist prinzipiell sowohl Digital, als auch Analog möglich. Die Digitale Auswertung läuft recht simpel ab, prinzipiell muss nur die Zeit erfasst werden, die zwischen dem Senden des PWM-Signals und dem Empfangen des Echo-Signals vergeht. Die errechnete Strecke ist zu halbieren, da die vergangene Zeit sowohl den Hin-, als auch den Rückweg beinhaltet. Bei der analogen Auswertung besteht zwar die Möglichkeit über einen Frequenzvergleich auch Signale mit noch kleinerer Echo-Amplitude zu erkennen und auszuwerten, allerdings beinhaltet dieses Vorgehen einen deutlich höheren Programmieraufwand.

Bei der Berechnung der Zeiten muss berücksichtigt werden, dass das eingehende Echo-Signal die Ultraschallkapsel langsam in Schwingungen versetzt, die ersten eintreffenden Schwingungen eines PWM-Signals erzeugen also kleinere Spannungssignale als die darauf folgenden. Außerdem schwingt

die Ultraschallkapsel auch nach Ende des eingehenden Signals noch etwas nach, was zur Folge hat, dass das analoge Abbild des gesendeten PWM-Signals leicht versetzt und etwas verlängert wirkt. All diese Faktoren müssen für eine genauere Berechnung der Strecke, die das Signal zurück gelegt hat, berücksichtigt werden. Allein durch die Tatsache, dass das empfangene Signal an der Ultraschallkapsel derart verändert wird, macht eine absolut exakte Messung nicht möglich. Eine Beschränkung des Fehlers auf einzelne Zentimeter dürfte aber durchaus realisierbar sein. Die Problematik ist nun, dass die Zusammenschaltung von dem Empfänger nun mit dem Sender was zur Folge hat, dass ein Ausgang vom A5950 an Masse angeschlossen ist und somit ein Kurzschluss erzeugt wird, was beim Wechsel von der positiven zur negativen Amplitude geschieht. Um eine bessere Aussage zu treffen, wurde eine H-Side Schaltung aufgebaut, siehe dazu das Schaltbild x.X, das IC A5950 wurde entfernt. Die H-Side konnte nur den Ultraschallsensor nicht gegen Masse schalten, was zur Folge hatte, dass wir ein nachschwingen am Oszilloskop messen konnten, siehe „Kapitel Messung bild x.x“. Um die Schwierigkeit zu lösen, benötigten wir eine Halbbrücke mit einer separaten Ansteuerung ohne eine interne Logik. Somit wurde eine Halb-Brücke aufgebaut, was die H-Side Schaltung ersetzt. Die Ansteuerungslogik wurde vom Prozessor XMC 1xxx48 bewerkstelligt.

### **3.4 Fazit aus den Ergebnissen für den Auftraggeber**

## 4 Reflektion über den Projektablauf

Nach Erhalt der Aufgabenbeschreibung war das Bild, das man sich von den bevorstehenden Aufgaben machte, doch sehr anders, als die Arbeiten später aussahen. So war mit der Programmierung, auf Grund des doch deutlich komplexeren Mikrocontrollers, ein enormer Lernaufwandt verbunden. Die Recherche der Hardwarebauteile und das Erstellen der Schaltpläne und Platinen hingegen verlief ähnlich den Erwartungen.

Eddy: Die Einarbeitung in die Tinkerforge verwendeten Programme, KiCad und GitHub, erforderten genausoviel Aufmerksamkeit wie die Programmierung trotz der Einarbeitungsphasen war stest konstant ein Lernerfolg zu verbuchen und konnten bis zum ende des Projektes uns neues Wissen sowie Fähigkeiten erwerben.

# **5 Anhänge**

## **5.1 Schaltpläne**

## **5.2 Platinenlayout**

## **5.3 Quellcode**

Hier könnte Ihr Quellcode (inklusive Werbung) stehen  
weitere Abbildungen

# Abbildungsverzeichnis

2.1	Blockschaltbild . . . . .	4
2.2	Die API für ein Programmbeispiel anhand einer PULL-UP Initialisierung . . . . .	6
2.3	xmc1_gpio.c . . . . .	7
2.4	main.c . . . . .	10
2.5	a16pt.h . . . . .	10
2.6	a16pt.c . . . . .	11
3.1	PWM-Burst auf 40kHz Basis an der CPU . . . . .	12
3.2	PWM Ausgabe über einen Hi-Side . . . . .	12
3.3	PWM Ausgabe über eine Halbbrücke . . . . .	13
3.4	Ausgabe der PWM an der Ultraschallkapsel . . . . .	13
3.5	Signal Empfang . . . . .	13
3.6	Signal nach Verstärkung . . . . .	13
3.7	Signal nach Verstärkung2 . . . . .	14
3.8	Signal nach Komparator . . . . .	14
3.9	Verlauf der zwei generierten PWMs für den Betrieb der voll gesteuerte Halbbrücke . . . . .	14
3.10	Signalverlauf bei 5V auf 1m Abstand . . . . .	15
3.11	Signalverlauf bei 5V auf 2m Abstand . . . . .	15
3.12	Signalverlauf bei 5V auf 3m Abstand . . . . .	15
3.13	Signalverlauf bei 5V auf 4m Abstand . . . . .	15
3.14	Signalverlauf bei 5V auf 5m Abstand . . . . .	16
3.15	Signalverlauf bei 10V auf 5m Abstand . . . . .	16
3.16	Signalverlauf bei 15V auf 5m Abstand . . . . .	16
3.17	Signalverlauf bei 20V auf 5m Abstand . . . . .	16

# Tabellenverzeichnis