Masterarbeit

FOLLOW-ME MIT ULTRA-WIDEBAND

ausgeführt am

CAMPUS 02 GRAZ

FACHHOCHSCHULE DER WIRTSCHAFT

Fachhochschul-Masterstudiengang

Automatisierungstechnik-Wirtschaft

von

Nico Grünwald, B.Sc.

51911255

betreut und begutachtet von

FH-Prof. Dipl.-Ing. Dieter Lutzmayr

Graz, im Mai 2024

.....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

KURZFASSUNG

Präzise Ortung, schnelle Datenübertragung und geringe Interferenz mit anderen drahtlosen Technologien würde in den kommenden Jahren zunehmend an Bedeutung gewinnen. Insbesondere im Bereich der Logistik sowie bei der sicheren und effizienten drahtlosen Übertragung größerer Datenmengen zwischen Geräten wird die Bedeutung von Indoor-Ortungssystemen weiter zunehmen.

Ultra-Wideband (UWB)-Technologie bietet eine vielversprechende Lösung für diese Anforderungen. UWB-Systeme zeichnen sich durch niedrigere Hardwarekosten, reduzierte Komplexität und einen geringeren Stromverbrauch im Vergleich zu anderen drahtlosen Übertragungsmethoden aus, wodurch sie sich in den kommenden Jahren weiter ausbreiten wird.

Diese Arbeit konzentriert sich auf die Implementierung eines Follow-Me-Systems mithilfe eines Modellfahrzeugs und eines UWB-Kits von Mobile Knowledge. Ziel ist es, den Aufwand und die Komplexität der Programmierung einer UWB-Anwendung zu demonstrieren sowie die Genauigkeit und Reichweite von UWB zu untersuchen und zu bewerten.

ABSTRACT

Precise location tracking, rapid data transmission, and minimal interference with other wireless technologies are becoming increasingly important in the coming years. Indoor positioning systems are set to play a crucial role in logistics and in the secure, wireless transmission of larger data packets across various sectors.

Ultra-Wideband (UWB) technology offers a promising solution for these needs. UWB systems are expected to grow in prevalence due to their lower hardware costs, reduced complexity, and lower power consumption compared to other wireless communication methods.

This work focuses on implementing a Follow-Me system using a model vehicle and a UWB kit from Mobile Knowledge. The goal is to demonstrate the effort and complexity involved in programming a UWB application, as well as to evaluate and research the accuracy and range of UWB technology.

INHALTSVERZEICHNIS

1 Einleitung ... 1
1.1 Ausgangssituation ... 1
1.2 Problemstellung ... 2
1.3 Zweck der Arbeit ... 2
2 UWB-Technologie ... 4
2.1 Allgemeines ... 4
2.2 Funktion von UWB ... 5
2.3 Technische Daten ... 6
2.3.1 Bandbreite ... 6
2.3.2 Impulse ... 6
2.3.3 Modulation ... 7
2.3.4 Positionierung und Ortung ... 7
2.3.5 Rauschunterdrückung ... 7
2.3.6 Niedrige Sendeleistung ... 7
2.3.7 Pulsformung ... 7
2.3.8 Regulierung ... 7
2.4 Varianten der Messung ... 8
2.4.1 Single Sided-Two Way Ranging (SS-TWR) ... 8
2.4.2 Double Sided-Two Way Ranging (DS-TWR) ... 9
2.4.3 Time Difference of Arrival (TDoA) ... 10
2.4.4 Angle of Arrival (AoA) ... 11
2.5 Vorteile von UWB ... 12
2.6 Anwendungsmöglichkeiten von UWB ... 12
3 Controller ... 14
3.1 Arduino MEGA ... 14
3.2 PX4 Drohnencontroller ... 16
3.3 Pixhawk 6X ... 17
4 UWB Kit ... 18
4.1 Allgemeines ... 18
4.2 Hardwareinhalt des MK UWB Kit ... 19
4.2.1 MK UWB Shield 2 ... 19
4.2.2 MK UWB 3D Antenna Board ... 21
4.2.3 UWB Tag SR040 ... 21
4.2.4 MK UWB SR040 Board ... 23
4.3 Softwareelemente des MK-UWB-Kit ... 24
4.3.1 MK UWB SDK ... 24
4.3.2 MK UWB SDK vorkompilierte Binärdateien ... 24
4.3.3 MK UWB Toolbox Android ... 25
4.3.4 MK UWB PC Shell-Anwendung ... 25

5 Software ... 26
5.1 NXP MCUXpresso IDE ... 26
5.2 Arduino IDE ... 28
5.3 Programmiersprachen ... 29
5.3.1 Python ... 29
5.3.2 C/C++ ... 30
5.3.2.1 C-Programmiersprache ... 30
5.3.2.2 C++-Programmiersprache ... 31
5.4 UART-Kommunikation ... 33
5.4.1 Allgemein ... 33
5.4.2 Synchronisation ... 33
5.4.2.1 Baudrate ... 34
5.4.2.2 Flow Control ... 34
5.4.3 Nachricht ... 34
5.4.3.1 Startbit ... 35
5.4.3.2 Stoppbit ... 35
5.4.3.3 Datenbit ... 35
5.4.3.4 Paritätsbit ... 36
5.5 TLV ... 37
6 Erste Schritte der Programmierung ... 39
6.1 Programminstallation ... 39
6.1.1 Python 3.9 ... 39
6.1.2 NXP MCUXpresso IDE ... 39
6.1.3 PyCharm ... 39
6.1.4 Arduino IDE ... 40
6.1.5 MK UWB PC Shell ... 40
6.2 Übungsprojekte ... 41
6.2.1 Hardwarevorbereitungen ... 41
6.2.2 Anwendungsfall Abstandswarnung ... 43
6.2.2.1 Allgemein ... 43
6.2.2.2 Android Applikation ... 44
6.2.2.3 PC Shell Applikation ... 45
6.2.3 Anwendungsfall Tracker ... 46
6.2.3.1 Allgemein ... 46
6.2.3.2 Android Applikation ... 47
6.2.3.3 PC Shell Applikation ... 48
6.3 Umbau des Modellautos ... 49
7 Programmierung ... 51
7.1 Auswahl Kommunikation und Mikrocontroller ... 51
7.2 Programmierung der Steuerung des Modellfahrzeuges ... 53
7.2.1 Bibliotheken und globale Variablen ... 53

7.2.2 Setup()-Funktion ... 55
7.2.3 Loop () -Funktion ... 55
7.3 Übertragung der Messdaten ... 57
7.3.1 REV01 des Programmes zur Übertragung ... 57
7.3.1.1 Globale Variablen ... 57
7.3.1.2 Setup()-Funktion ... 57
7.3.1.3 Loop()-Funktion ... 59
7.3.1.4 Ergebnis ... 60
7.3.2 REV02 des Programmes zur Übertragung ... 61
7.3.2.1 Globale Variablen und Setup()-Funktion ... 61
7.3.2.2 Loop()-Funktion ... 62
7.3.2.3 Ergebnis ... 63
7.3.3 Konfiguration der MK UWB Shield 2 ... 64
7.4 Zusammenführen der Messdaten und Steuerung ... 66
7.5 Programmierung des UWB-Kits ... 67
8 Messung ... 69
8.1 Beschreibung der Messungen ... 69
8.2 Ergebnisse der ersten Messung ... 69
8.3 Ergebnisse der zweiten Messung ... 70
9 ZUSAMMENFASSUNG UND AUSBLICK ... 72
9.1 Probleme ... 72
9.2 Zusammenfassung ... 72
Literaturverzeichnis ... 74
Abbildungsverzeichnis ... 76
Tabellenverzeichnis ... 79
Abkürzungsverzeichnis ... 80

1 EINLEITUNG

1.1 Ausgangssituation

Die FH Campus 02 hat das UWB-Kit von NXP erworben, zu sehen in Abbildung 1. Das Kit beinhaltet zwei UWB-Tags SR040. Dies sind autonome batteriebetriebene UWB-Geräte, die alle Arten von Tracking- und Lokalisierungsanwendungsfällen ermöglichen. Diese Tags sind besonders nützlich in Szenarien, in denen präzise Positionsbestimmung und Echtzeitverfolgung entscheidend sind. Zusätzlich umfasst das Kit einen MK UWB Shield 2, eine flexible Plattform zur Entwicklung von kundenspezifischen UWB-Systemen. Der MK UWB Shield 2 dient als zentrale Einheit, die Signale von den UWB-Tags empfängt und verarbeitet, um genaue Positionsdaten zu berechnen. Diese Plattform ist besonders anpassungsfähig und kann für eine Vielzahl von Anwendungen modifiziert werden.

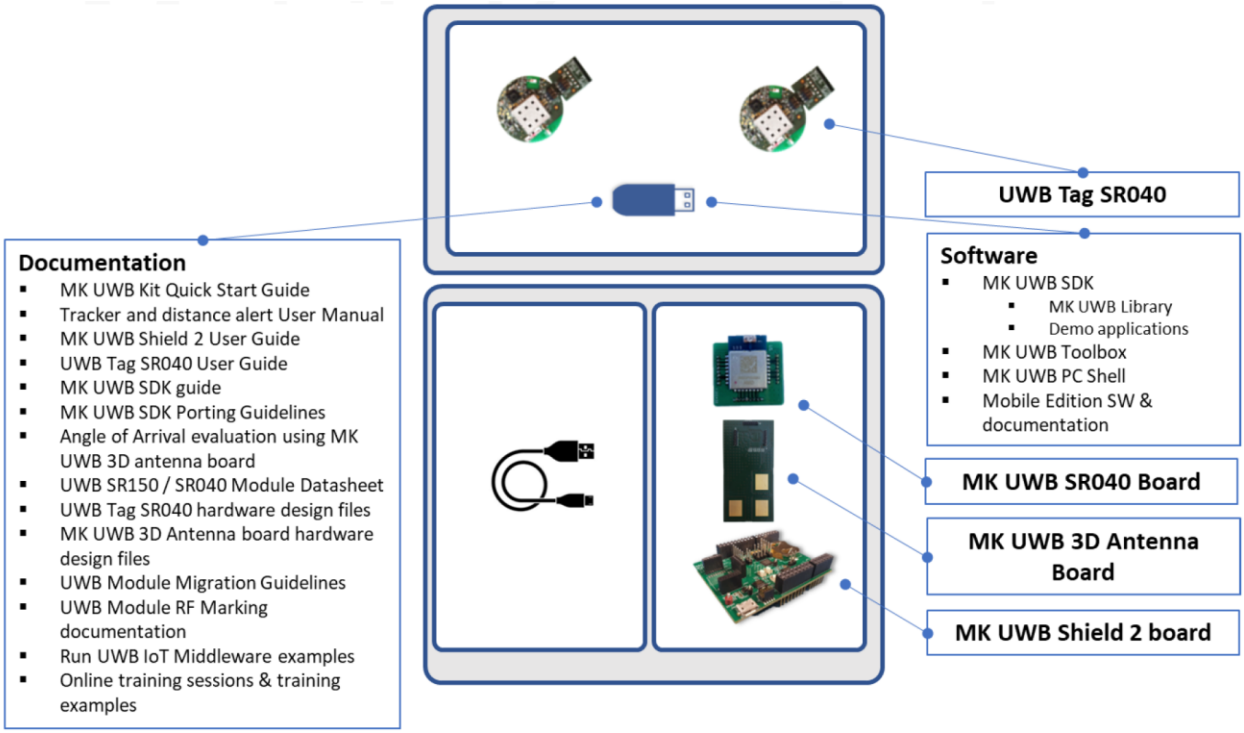


Abbildung 1: Das MK UWB Kit, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Weiters existiert an der FH ein NXP CUP model car, ein Modellauto, das mit Lenkung und Antrieb ausgestattet ist, jedoch ohne Steuerung. Dieses Modellauto bietet eine hervorragende Basis für praktische Experimente und Entwicklungsprojekte im Bereich autonomer Fahrzeuge und Robotik. Das Modellauto kann in Kombination mit dem UWB-Kit verwendet werden, um innovative Lokalisierungs- und Navigationslösungen zu entwickeln. Beispielsweise kann das MK UWB Shield 2 auf dem Modellauto installiert werden, zur Nutzung als stationäre Referenzstation, während die UWB-Tags an einer Person angebracht werden, um diese mit dem Modellauto zu verfolgen. Diese Kombination ermöglicht es, fortschrittliche Steuerungsalgorithmen zu testen sowie die Genauigkeit und Effizienz von UWB-basierten Trackingsystemen zu evaluieren.

1.2 Problemstellung

Im Zuge dieser Arbeit soll mittels UWB-Übertragung ein Follow-Me Modellfahrzeug realisiert werden. Dabei wird angestrebt, dass das Fahrzeug einem Tag bzw. einer Person in einem bestimmten Abstand folgt. Um dieses Ziel zu erreichen, müssen der UWB-Tag sowie das MK UWB Shield 2 programmiert und kalibriert werden, um präzise Abstands- und Winkelmessungen zu ermöglichen. Diese Messdaten werden dann an einen Mikrocontroller weitergegeben, der die Steuerung des Modellautos übernimmt.

Der Mikrocontroller verarbeitet die empfangenen Winkel- und Abstandsdatendaten und sendet entsprechende Steuerbefehle an das Modellfahrzeug. Dadurch kann das Fahrzeug kontinuierlich seine Position in Relation zu der sich bewegenden Person oder dem Tag anpassen und einen konstanten Abstand einhalten. Diese Follow-Me Funktion ist besonders nützlich in Anwendungsbereichen wie persönlicher Begleitung, autonomen Transportdiensten oder in industriellen Settings, in denen Roboter bestimmten Wegen oder Personen folgen müssen.

Für die Steuerung des Modellfahrzeugs stehen zwei Optionen zur Verfügung. Falls lieferbar, wird ein PX4 Drohnencontroller verwendet. Der PX4 ist bekannt für seine fortschrittlichen Steuerungsalgorithmen und seine Fähigkeit, präzise und stabile Flug- bzw. Fahrmanöver auszuführen. Er bietet eine Vielzahl von Funktionen und Schnittstellen, die für die Integration in das UWB-basierte Follow-Me System von Vorteil sind.

Falls der PX4 Drohnencontroller nicht verfügbar ist, wird ein Arduino Mikrocontroller als alternative Lösung eingesetzt. Arduino-Plattformen sind flexibel und weit verbreitet, mit einer großen Community und vielen verfügbaren Bibliotheken. Sie ermöglichen eine einfache und schnelle Implementierung der benötigten Steuerungsfunktionen und bieten genügend Rechenleistung und Schnittstellen, um die Anforderungen des Follow-Me Systems zu erfüllen.

1.3 Zweck der Arbeit

Geklärt werden soll, die Funktionsweise der UWB-Technologie. Kapitel 2 der Arbeit beschreibt die UWB-Technologie sowie deren Funktionen und Eigenschaften, um ein Basisverständnis für das Projekt zu schaffen. Dies umfasst die Grundlagen der UWB-Technologie, einschließlich der Signalübertragung und -empfang, der Bandbreite und Frequenznutzung sowie der Besonderheiten, die UWB von anderen drahtlosen Technologien unterscheidet. UWB nutzt sehr kurze Impulse mit einer breiten Frequenzbandbreite, um hochpräzise Positions- und Abstandsmessungen zu ermöglichen. Diese Technologie ist bekannt für ihre hohe Genauigkeit und geringe Störanfälligkeit.

Weiters soll untersucht werden, wie die Abstandsmessung und die Winkelmessung mit den NXP-Komponenten funktioniert. Dabei wird analysiert, wie die UWB-Tags und der UWB-Anchor zusammenarbeiten, um Entfernungen und Winkel zu berechnen. Die Methoden der Time-of-Flight (ToF) und Angle-of-Arrival (AoA) Messungen werden dabei besonders berücksichtigt. Diese Techniken ermöglichen es, die exakte Position eines UWB-Tags in Bezug auf den Anchor zu bestimmen, welches für die Realisierung des Follow-Me Modells essenziell ist.

Einerseits soll geklärt werden, welche Hardware zur Realisierung dieses Projektes benötigt wird. Dies umfasst die Auswahl geeigneter UWB-Tags und -Anchors, einen leistungsfähigen Mikrocontroller (wie den PX4 Drohnencontroller oder einen Arduino MEGA 2560), und möglicherweise zusätzliche Sensoren zur Unterstützung der Navigation. Auch die Energieversorgung und die mechanische Integration der Komponenten ins Modellfahrzeug müssen berücksichtigt werden.

Andererseits soll festgestellt werden, wie die Implementierung von UWB funktioniert und mit welchem Aufwand die Programmierung und Kalibrierung der NXP-Komponenten verbunden ist.

2 UWB-TECHNOLOGIE

2.1 Allgemeines

UWB steht für *Ultra-Wideband*, was auf Deutsch „Ultra-Breitband“ bedeutet. Es handelt sich um eine drahtlose Kommunikationstechnologie, die ein breites Frequenzband verwendet, um Daten über kurze Entfernungen mit hoher Störfestigkeit zu übertragen. UWB unterscheidet sich dadurch von traditionellen drahtlosen Technologien wie Wi-Fi oder Bluetooth, die auf schmalbandigen Frequenzbereichen arbeiten, wie in Abbildung 2 zu sehen.

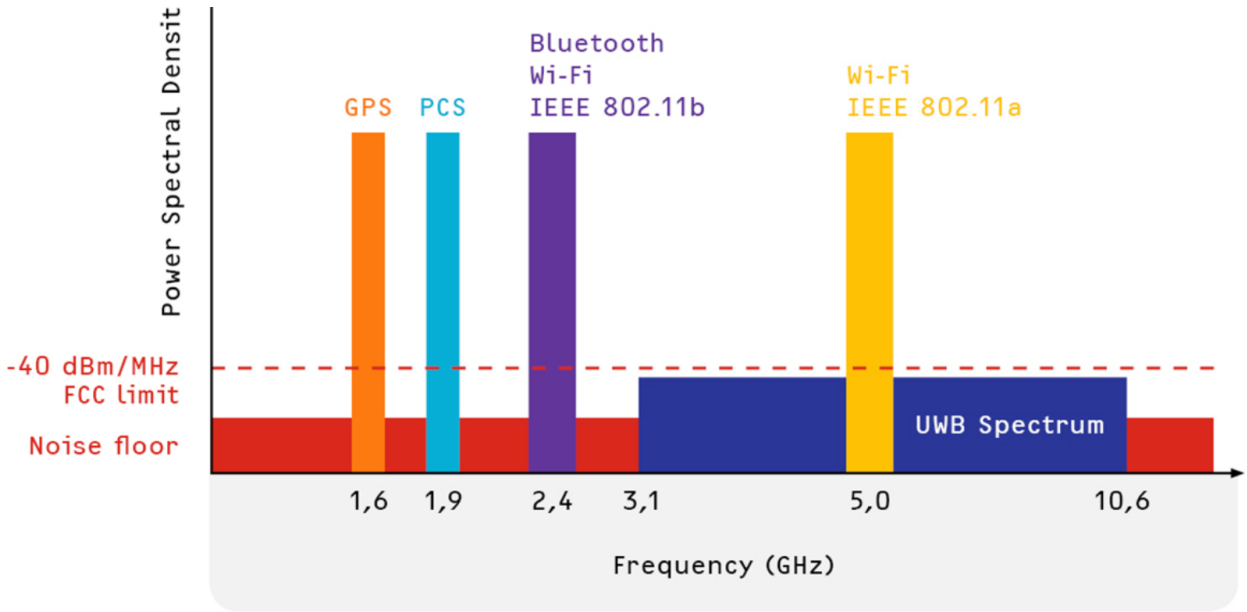


Abbildung 2: Frequenzbereiche verschiedener Übertragungsarten, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Die Funktionsweise von UWB basiert auf der Aussendung von kurzen Impulsen mit sehr geringer Sendeleistung über ein breites Frequenzspektrum. Dieses breite Frequenzspektrum ermöglicht es, Informationen auf verschiedene Arten zu übertragen. Die genaue Funktion von UWB wird in Abschnitt 2.2 erklärt. Zuvor wird kurz auf die Geschichte von UWB eingegangen:

Die Grundlage für UWB wurde bereits in den 1960er Jahren gelegt, als Forscher/-innen begannen, die Vorteile von sehr kurzen Funkimpulsen zu erkunden. Diese Impulse hatten eine sehr geringe Dauer und daher große Bandbreite und waren daher in der Lage, breite Frequenzbänder effektiv gleichzeitig zu nutzen.

In den 1970er Jahren wurde UWB-Technologie vermehrt für militärische Anwendungen genutzt. Sie wurde für die Ortung von Zielen und die Kommunikation in feindlichen Umgebungen eingesetzt, da sie schwer zu orten und zu stören war.

In den 1990er Jahren gewann UWB auch in zivilen Anwendungen zunehmend an Bedeutung. Die durch UWB ermöglichte genaue Ortung und Positionierung wurde in Bereichen wie Radar, drahtloser Kommunikation und drahtlosem USB (WUSB) genutzt.

In den 2000er Jahren wurden weltweit Bemühungen unternommen, UWB zu regulieren und Standards für den Einsatz in zivilen Geräten festzulegen. Dies ermöglichte die Entwicklung von UWB-Chipsätzen und-Geräten, die in Produkten wie Smartphones, Autos und drahtlosen Peripheriegeräten eingesetzt werden konnten.

UWB hat sich bis heute weiterentwickelt und wird mittlerweile in einer Vielzahl von Anwendungsfällen eingesetzt, darunter präzise Ortungsdienste (z.B. für Indoor-Navigation), die Steuerung von Smart-Home-Geräten und die drahtlose Übertragung von Daten mit hoher Geschwindigkeit. [1]

2.2 Funktion von UWB

Das UWB-Signal breitet sich in der Luft mit Lichtgeschwindigkeit aus. Wenn das Signal auf ein Hindernis trifft, wird es entweder absorbiert, gestreut oder gebeugt. Dabei breiten sich die Funkwellen auf mehreren Wegen in alle Richtungen aus, um ihr Ziel zu erreichen, wie in Abbildung 3 gezeigt. Dies bedeutet, dass der Empfänger, siehe Abbildung 3 rechts, dasselbe Signal mehrmals mit unterschiedlicher Intensität empfängt, bestehend aus einem direkten und mehreren reflektierten Anteilen. Genau das ist ein wichtiger Aspekt bei der Decodierung von Ultrabreitbandsignalen.

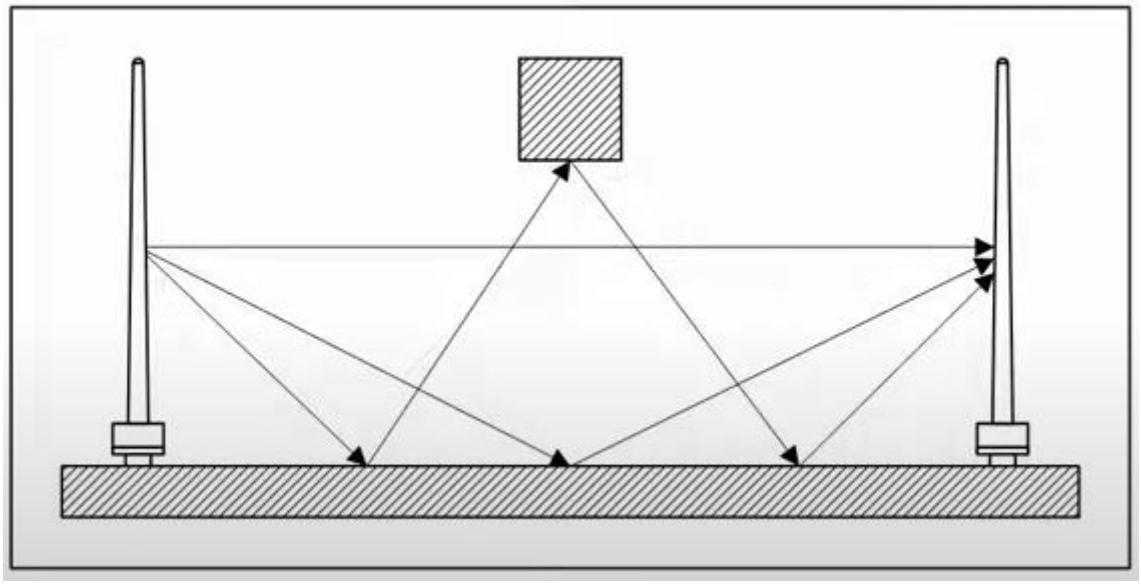


Abbildung 3: Ausbreitung der Funkwellen, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

In diesem Zusammenhang gibt es einige Konzepte, die hilfreich sind, um die Ausbreitung von Ultrabreitbandsignalen zu verstehen. Die Kanalimpulsantwort zeigt die über die Zeit empfangene Amplitude und reflektiert die Auswirkungen des Mehrwegeausbreitungsverhaltens des Signals. Unterschiedliche Zeitintervalle entstehen durch die verschiedenen Wege, die das Signal nehmen kann. Wenn der erste empfangene Impuls ein höheres Energieniveau aufweist als die nachfolgenden Impulse, deutet dies darauf hin, dass der direkte Weg der stärkste ist. Dies bedeutet, dass das Zielobjekt ohne

[1] FIRA (2023), Online-Quelle [01.04.2024]

signifikante Hindernisse auf dieser Seite liegt, wie in der Mitte von Abbildung 4 zu sehen. Wenn der Impuls mit einem höheren Energieniveau nicht der erste ist, bedeutet dies, dass er über eine Nicht-Sichtlinie empfangen wird, dies kann Aufgrund von Hindernissen oder Reflexionen passieren. Aufgrund der großen Anzahl von Reflexionen und Hindernissen kann es daher zu vielen parallelen Ausbreitungen kommen. Um die besten Ergebnisse der Entfernungsmessung zu erzielen, benötigt der Empfänger ausgefeilte Algorithmen, um die Entfernung zu berechnen.[2]

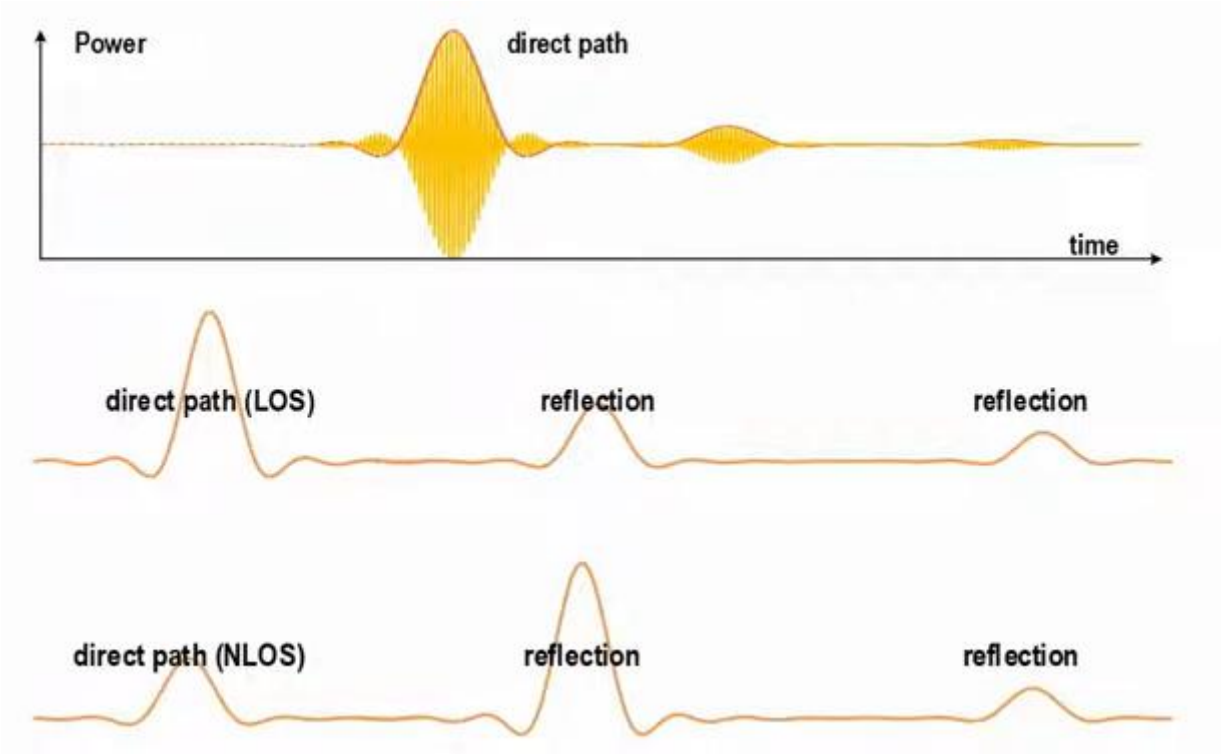


Abbildung 4: Empfangene Impulse mit Energieniveaus, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

2.3 Technische Daten

UWB ist eine drahtlose Kommunikationstechnologie, die aufgrund ihrer großen Bandbreite und kurzen Impulse einzigartige technische Eigenschaften aufweist.

2.3.1 Bandbreite

UWB nutzt breite Frequenzbänder (≈ 500 MHz), oft im Bereich von mehreren Gigahertz (GHz) der Mittenfrequenz. Die Frequenzbänder können zwischen 3 und 10.6 GHz liegen.

2.3.2 Impulse

UWB-Übertragungen erfolgen in Form von sehr kurzen Impulsen, die oft im Bereich von 1 bis 10 Nanosekunden liegen. Diese Impulse sind so kurz, dass sie in der Zeitdimension sehr schmal sind und zu einer breiten Bandbreite im Frequenzspektrum führen.

[2] Tsang/El-Gamal (2005), Online-Quelle [01.03.2024]

2.3.3 Modulation

UWB kann verschiedene Modulationsarten verwenden, um Daten zu übertragen. Dazu gehören Impulsmodulationstechniken wie Pulsamplitudenmodulation (PAM), Pulsdauermodulation (PDM) und Phasenmodulation (PM).

2.3.4 Positionierung und Ortung

UWB nutzt die Eigenschaft der Signallaufzeit, um präzise Ortung und Positionierung zu ermöglichen. Durch Messung der Laufzeit von UWB-Impulsen zwischen Sender und Empfänger können Entfernungen mit hoher Genauigkeit berechnet werden.

2.3.5 Rauschunterdrückung

Aufgrund der großen Bandbreite und kurzen Impulse ist UWB sehr robust gegenüber schmalbandiger Interferenzen und Rauschen. Dies ermöglicht eine zuverlässige Datenübertragung in Umgebungen mit vielen anderen Signalen.

2.3.6 Niedrige Sendeleistung

UWB verwendet in der Regel sehr geringe durchschnittliche Sendeleistungen, wodurch die elektromagnetischen Emissionen niedrig bleiben. Diese geringe durchschnittliche Leistung ist insbesondere für medizinische Anwendungen und Bereiche mit strengen EMV-Anforderungen von Vorteil. Es ist jedoch möglich, dass UWB-Systeme hohe Spitzenleistungen über sehr kurze Zeiträume nutzen. Obwohl die Impulse nur für einige Nanosekunden gesendet werden, können sie sehr hohe Spitzenleistungen aufweisen. Dies ermöglicht eine breite Frequenzabdeckung, während die durchschnittliche Leistung niedrig bleibt.

2.3.7 Pulsformung

UWB-Systeme verwenden oft Pulsformungstechniken, um die Form der gesendeten Impulse zu einzustellen. Dies ermöglicht eine effiziente Nutzung des Frequenzspektrums und trägt zur Reduzierung von Interferenzen bei.

2.3.8 Regulierung

Die Verwendung von UWB ist in vielen Ländern reguliert, um sicherzustellen, dass die Technologie nicht die Funkdienste anderer Frequenzbereiche stört. In den USA beispielsweise wird UWB von der FCC (Federal Communications Commission) reguliert. In Europa erfolgt die Regulierung durch die Europäische Kommission in Zusammenarbeit mit der European Conference of Postal and Telecommunications Administrations (CEPT).

Es ist wichtig zu beachten, dass UWB eine sich weiterentwickelnde Technologie ist und die technischen Details je nach Implementierung und Anwendung variieren können. Die genaue Modulation, Pulsformung und andere technische Aspekte können von Hersteller zu Hersteller unterschiedlich sein. [3]

2.4 Varianten der Messung

2.4.1 Single Sided-Two Way Ranging (SS-TWR)

SS-TWR ist eine Technik zur präzisen Messung der Entfernung zwischen einem Initiator und einem Responder unter Verwendung von Funkwellen. Die Flugzeit (Time of Flight, TOF) ist das Maß für die Zeit, die die Radiowellen benötigen, um eine Distanz in der Luft zurückzulegen. Der Initiator (A), wie in Abbildung 5 links zu sehen, sendet eine Anfrage an den Responder (B). Die Zeit, welche die Anfrage benötigt, um die Distanz DA_B zwischen A und B zurückzulegen und B zu erreichen, beträgt

$$ToF_{A_B} = \frac{D_{A_B}}{c} \qquad (2.1)$$

wobei c die Lichtgeschwindigkeit ist. Wegen der Verzögerung der Verarbeitung, die zwischen dem Empfang der Anfrage und dem Senden der Antwort entsteht, wird ein σ_{reply} hinzugefügt. Die Verzögerung, die die Antwort benötigt, um A zu erreichen, beträgt

$$ToF_{B_A} = \frac{D_{B_A}}{c} = ToF_{A_B} \qquad (2.2)$$

Der Initiator erkennt genau die Zeit zwischen dem Senden der Anfrage und dem Empfangen der Antwort:

$$\sigma_{loop} = 2\ x\ ToF_{A_B} + \sigma_{reply} \qquad (2.3)$$

TOFA_B ist die berechnete Zeit, die die Radiowellen benötigen, um die Distanz von A nach B zu überwinden. σ_{loop} repräsentiert die gesamte gemessene Zeit, die die Radiowellen zum Empfänger und wieder zurück sowie die Verarbeitungsverzögerung umfasst. σ_{reply} bezeichnet die Zeitverzögerung, die auftritt, während der Responder (B) die eingehende Anfrage verarbeitet und eine Antwort zurück an den Initiator (A) sendet.

$$ToF_{A_B} = \frac{\sigma_{loop} - \sigma_{reply}}{2} \qquad (2.4)$$

Dies ist eine einfache Technik, aber jeder Unterschied in der Taktfrequenz zwischen dem Initiator und dem Responder führt zu einem Fehler in der Zeitberechnung und somit in der Entfernung. [4]

[3] MobileKnowledge (2023), Online-Quelle [15.07.2023]

[4] MobileKnowledge (2023), Online-Quelle [15.07.2023]

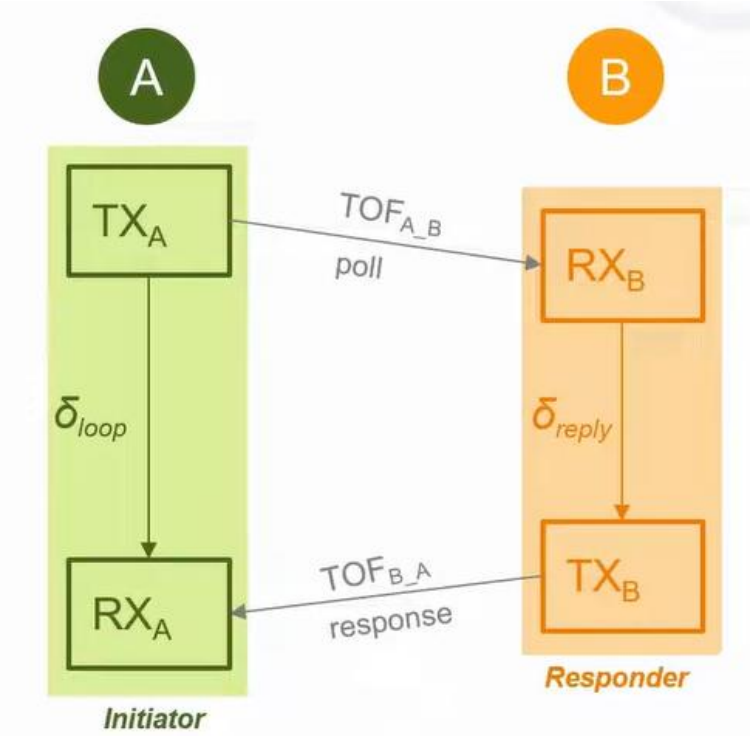


Abbildung 5: Single Sided-Two Way Ranging, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

2.4.2 Double Sided-Two Way Ranging (DS-TWR)

SS-TWR ist eine einfache Technik, aber jede Taktreferenzdifferenz zwischen Initiator und Responder führt zu einem TOF-Berechnungsfehler. Ungenaue Taktmessungen oder Synchronisationsfehler können zu erheblichen Abweichungen in der Entfernungsbestimmung führen, da die Berechnung direkt von den Zeitstempeln der Signale abhängt. Um dies zu vermeiden, kann das sogenannte Double Sided-Two Way Ranging (DS-TWR) verwendet werden. Dabei werden zwei SS-TWRs durchgeführt, wie in Abbildung 6 dargestellt. Jedes Gerät kann also die TOF berechnen. Der Taktunterschied zwischen den beiden Geräten wird so ausgeglichen, es wird jedoch dafür mehr Energie verbraucht. [5]

$$ToF_{A_B} = \frac{\sigma_{loopA} * \sigma_{loopB} - \sigma_{replyA} * \sigma_{replyB}}{\sigma_{loopA} + \sigma_{loopB} + \sigma_{replyA} + \sigma_{replyB}} \qquad (2.5)$$

[5] MobileKnowledge (2023), Online-Quelle [15.07.2023]

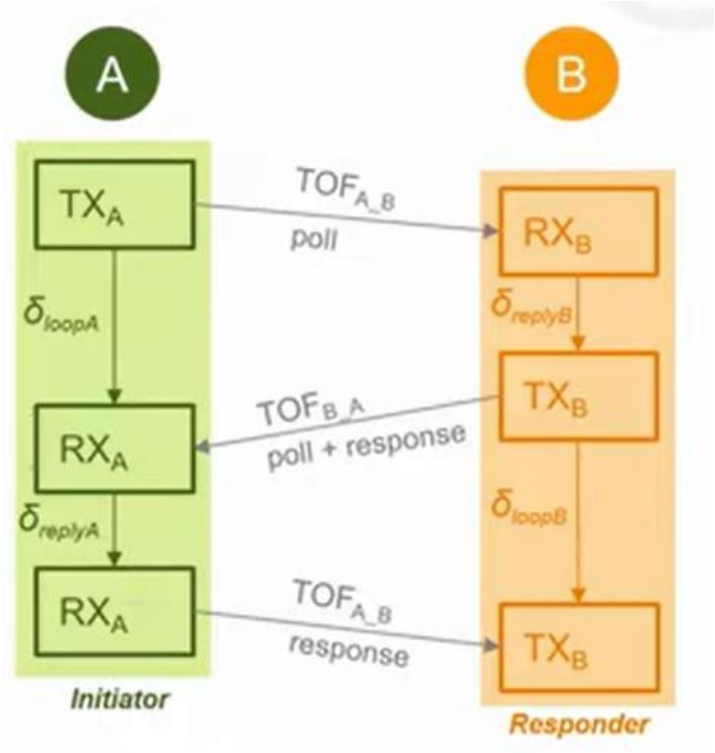


Abbildung 6: Double Sided-Two Way Ranging, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

2.4.3 Time Difference of Arrival (TDoA)

TDoA ist eine Methode zur Positionsbestimmung und Ortung, die auf den Zeitdifferenzen basiert, die Signale benötigen, um an verschiedenen Empfängern anzukommen. Diese Methode wird häufig in Kommunikations- und Navigationssystemen verwendet, einschließlich Radar, Funkortung und drahtlosen Sensornetzwerken.

Wie in Abbildung 7 dargestellt, sendet eine Quelle ein Signal aus, das von mehreren Empfängern empfangen wird. Jeder Empfänger misst die genaue Ankunftszeit des Signals. Aufgrund der unterschiedlichen Entfernungen zwischen der Quelle und den Empfängern kommt das Signal zu unterschiedlichen Zeiten an. Die Zeitdifferenzen Δt zwischen den Ankunftszeiten der Signale bei den verschiedenen Empfängern werden berechnet, wie in Abbildung 7 rechts zu sehen ist.

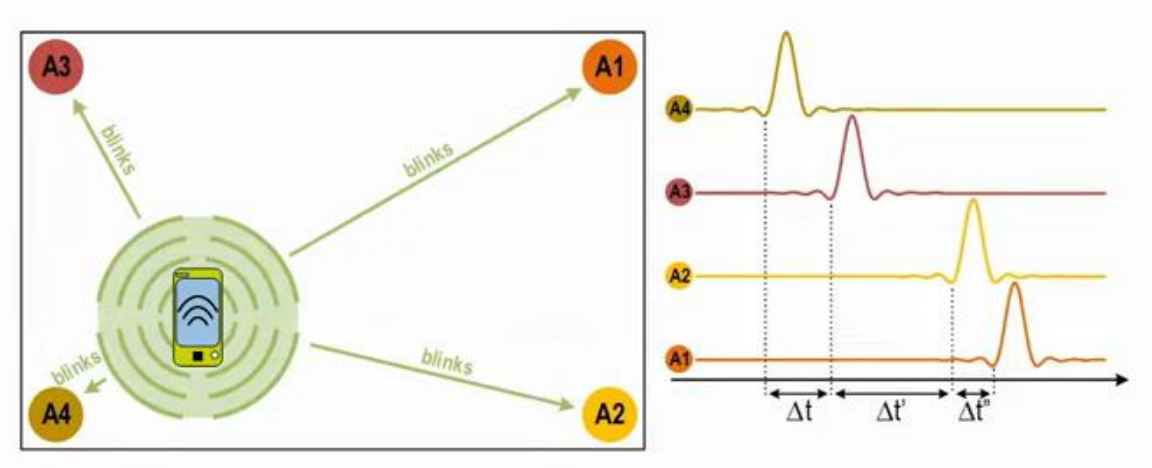


Abbildung 7: Time Difference of Arrival, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Für eine Lokalisierung ohne Mehrdeutigkeiten werden vier Anker benötigt. Grundsätzlich sind mindestens drei Anker erforderlich, um den Standort eines Objekts eindeutig zu bestimmen, es kann aber in bestimmten Fällen, zum Beispiel wenn die Anker zu nahe beieinander liegen, zu Mehrdeutigkeiten kommen. Der vierte Anker dient als zusätzlicher Referenzpunkt, um die Genauigkeit und Robustheit der Lokalisierung zu erhöhen. Alle Anker müssen synchronisiert sein. Dies führt im Vergleich zu TWR zu einer höheren Systemkomplexität und höheren Hardwarekosten. TDoA bietet aber dafür mehrere Vorteile. Einer der Hauptvorteile ist die hohe Genauigkeit, insbesondere wenn die Empfänger präzise synchronisiert sind und die Signallaufzeiten genau gemessen werden. Zudem ist das System skalierbar, da es durch das Hinzufügen weiterer Empfänger einfach erweitert werden kann, wodurch sowohl die Genauigkeit als auch die Reichweite erhöht werden. Ein weiterer Vorteil ist die Robustheit von TDoA gegenüber Interferenzen und Signalverzerrungen, da es auf Zeitdifferenzen und nicht auf absolute Zeiten angewiesen ist.[6]

2.4.4 Angle of Arrival (AoA)

Der Ankunftswinkel (AoA) basiert auf der PDoA-Methode (Phase Difference of Arrival). Die PDoA-Berechnung erfolgt während der Reaktion des TWR unter Verwendung der Phasendifferenz $\Delta\varphi$ des empfangen Signals zwischen zwei Antennen. Die Phasendifferenz $\Delta\varphi$ wird anhand des Trägersignals und nicht anhand der UWB-Impulshüllkurve berechnet. Der Abstand der zwei Empfangsantenne (d) sollte unter der Hälfte der Wellenlänge λ liegen. Der AoA Φ ist eine Berechnung aus den genannten Größen:

$$\Phi = \cos^{-1}\frac{\Delta\varphi * \lambda}{2\pi * d} \qquad (2.6)$$

Φ ist der Winkel des Eintreffens (AoA), $\Delta\varphi$ ist die Phasendifferenz, λ die Wellenlänge und d der Abstand der zwei Empfangsantenne.

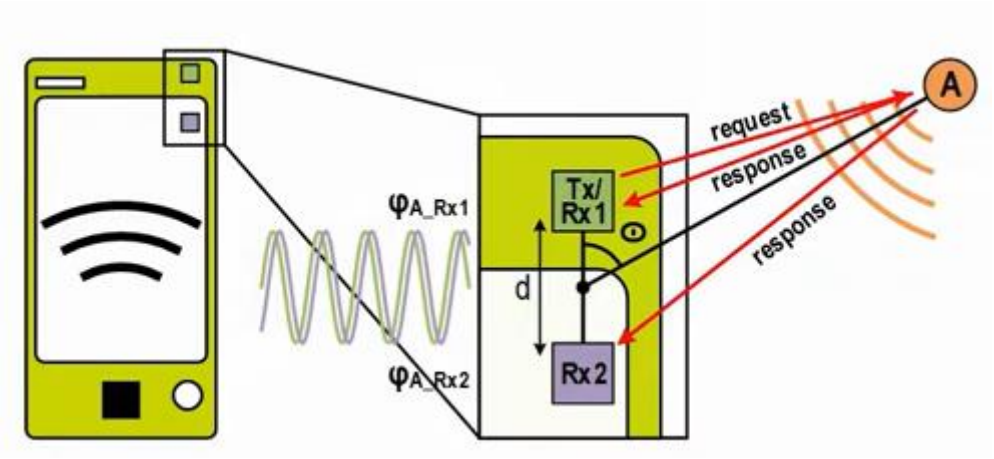


Abbildung 8: Angle of Arrival, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Die breite UWB-Frequenz macht die AoA-Schätzung genauer und aufgrund von mehreren Pfaden weniger anfällig für Fehler im Vergleich zu Schmalbandtechnologien wie Bluetooth oder Wi-Fi.

[6] MobileKnowledge (2023), Online-Quelle [15.07.2023]

2.5 Vorteile von UWB

Ultra-Wideband (UWB) bietet zahlreiche Vorteile, die es zu einer attraktiven Technologie machen. Erstens zeichnen sich UWB-Systeme durch geringe Hardwarekosten und Komplexität aus, da sie kostengünstig und stromsparend in Platinen implementiert werden können. Zweitens haben sie einen niedrigen Stromverbrauch, da der UWB-Basisbandbetrieb und die ultrakurzen Sendezeiten die Hardware-Komplexität und den Energiebedarf reduzieren.

Ein herausragender Vorteil von UWB ist die hohe Ortungsgenauigkeit. UWB-Geräte verwenden sehr kurze Zeitbereichsimpulse, die präzise Timings ermöglichen. In Kombination mit Flugzeitmessungen kann UWB eine Ortungsgenauigkeit bis auf den Zentimeter erreichen. Außerdem ermöglicht die sehr geringe Leistungsdichte des UWB-Signals eine Spektrumwiederverwendung. Die Eigenschaften des Signals erschweren die Unterscheidung vom Rauschen, wodurch eine bessere Koexistenz mit anderen Funktechnologien und eine erhöhte Sicherheit gegen Abhören erreicht werden.

Darüber hinaus zeichnet sich UWB durch hohe Robustheit aus. Die große Bandbreite des Signals sorgt für eine Mehrwegeauflösung und Frequenzvielfalt, wodurch das Signal widerstandsfähig gegen Mehrwegeausbreitung und Interferenzen wird. Zudem ermöglicht UWB hohe Datenraten im Nahbereich, die bis zu 500 Mbit/s über Entfernungen von bis zu 3 Metern erreichen können, und hebt sich somit von anderen Kommunikationstechnologien wie Bluetooth ab. [7]

2.6 Anwendungsmöglichkeiten von UWB

In Bezug auf Sicherheit und Zugangskontrolle kann UWB für sichere Zugangskontrollsysteme verwendet werden, um Türen zu öffnen oder zu verriegeln, wenn sich berechtigte oder unberechtigte Personen nähern. Darüber hinaus trägt UWB zur Verfolgung und Lokalisierung gestohlener Gegenstände bei, indem es genaue Positionsinformationen liefert.

In Smart Homes ermöglicht UWB eine präzise Standorterkennung von Personen und Geräten. Beispielsweise verwendet Apple UWB in seinen iPhones und AirTags, um verlorene Gegenstände wie Schlüssel oder Taschen zu lokalisieren. Die exakte Ortung in Innenräumen ermöglicht es, AirTags nicht nur zu finden, sondern sie auf wenige Zentimeter genau anzuzeigen. Ebenso könnte in einem Smart Home das Licht automatisch eingeschaltet oder die Klimaanlage angepasst werden, wenn sich eine Person in einem bestimmten Raum befindet. [8]

In der Logistik sorgt UWB für die Möglichkeit einer präzise Verfolgung von Waren in Lagern. Unternehmen wie DHL verwenden UWB, um den Standort von Paketen und Paletten in Echtzeit zu verfolgen. Diese genaue Verfolgung verbessert die Effizienz der Lagerhaltung und sorgt für schnellere Lieferprozesse, da die Waren sofort gefunden und versandt werden können.[9]

---

[7] Tsang/El-Gamal (2005), Online-Quelle [01.03.2024]

[8] APPLE (2024), Online-Quelle [04.08.2024]

[9] DHL (2024), Online-Quelle [01.08.2024]

In der Automobilindustrie findet UWB ebenfalls breite Anwendung. Volkswagen integriert UWB in seine Fahrzeuge, um schlüssellosen Zugang zu ermöglichen. Das Fahrzeug kann die Entfernung des Schlüssels oder Smartphones mit UWB-Technologie genau messen und öffnet automatisch, wenn sich der Benutzer dem Fahrzeug nähert. Diese Technologie bietet zusätzlich einen verbesserten Diebstahlschutz.[10]

Ein weiteres Beispiel ist der Einsatz von UWB in Drohnen. Hier ermöglicht die Technologie eine präzise Navigation in Umgebungen, in denen GPS nicht zuverlässig funktioniert, wie beispielsweise in Innenräumen oder urbanen Schluchten. Drohnenhersteller wie Skydio setzen UWB ein, um ihren Drohnen eine bessere Kollisionserkennung und eine sicherere Navigation in engen Räumen zu ermöglichen.

[10] NXP (2024), Online-Quelle [06.08.2024]

3 CONTROLLER

3.1 Arduino MEGA

Das Arduino Mega 2560, wie in Abbildung 9 dargestellt, ist ein vielseitiges Mikrocontroller-Board, das auf dem leistungsstarken ATmega2560-Chip basiert und sich ideal für komplexe Projekte eignet. Es bietet eine Vielzahl an Anschlussmöglichkeiten, mit insgesamt 54 digitalen Ein-/Ausgangspins, von denen 15 als PWM-Ausgänge genutzt werden können, womit die Steuerung von Motoren, LEDs und anderen Geräten ermöglicht werden. Zusätzlich verfügt es über 16 analoge Eingänge, die präzise Messungen von analogen Signalen ermöglichen.

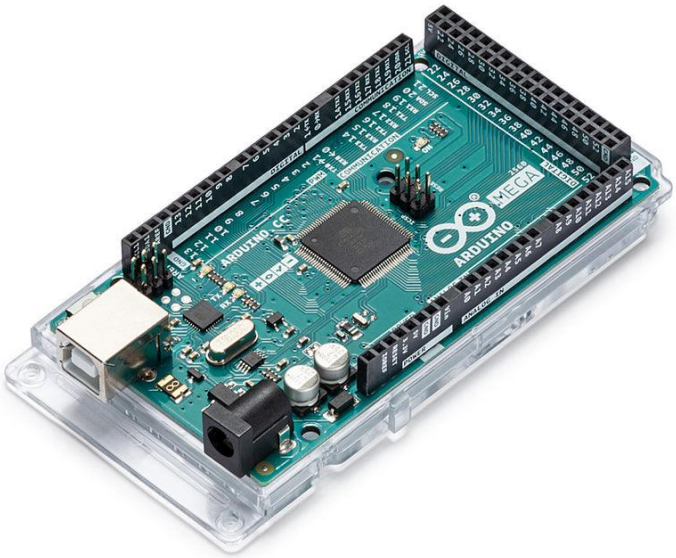


Abbildung 9: Arduino MEGA, Quelle: Arduino (2023), Online-Quelle [15.08.2023]

Das Board ist außerdem mit 4 UARTs ausgestattet, die serielle Kommunikation über mehrere Ports gleichzeitig ermöglichen und somit besonders nützlich für Projekte mit mehreren Kommunikationskanälen ist. Der 16-MHz-Quarzoszillator sorgt für eine stabile Taktfrequenz und zuverlässige Leistung. Ein USB-Anschluss ermöglicht eine einfache Programmierung und Stromversorgung über einen Computer. Alternativ kann das Board auch über eine externe Stromquelle wie einen AC-DC-Adapter oder eine Batterie betrieben werden, wodurch es in verschiedenen Umgebungen eingesetzt werden kann.

Ein ICSP-Header (In-Circuit Serial Programming Header) ermöglicht das Programmieren des Mikrocontrollers mit einem externen Programmiergerät. Der ICSP-Header ist eine spezielle Schnittstelle, die in Mikrocontrollern verwendet wird, um eine einfache Programmierung und das Debugging im eingebauten Zustand zu ermöglichen. Die Reset-Taste am Board erleichtert das Neustarten des Systems, ohne die Stromversorgung unterbrechen zu müssen. [11]

[11] Arduino (2023), Online-Quelle [15.08.2023]

Betriebsspannung	5 V
Eingangsspannung (empfohlen)	7–12 V
Eingangsspannung (Grenzwert)	6–20 V
Digitale I/O-Pins	54 (davon 15 mit PWM-Ausgang)
Analoge Eingangspins	16
Gleichstrom pro I/O-Pin	20 mA
Gleichstrom für 3,3-V-Pin	50 mA
Flash-Speicher	256 KB, davon 8 KB vom Bootloader verwendet
SRAM	8 KB
EEPROM	4 KB
Taktrate	16 MHz
LED_BUILTIN	13
Länge	101,52 mm
Breite	53,3 mm
Gewicht	37 g

Tabelle 1 Technische Daten des Arduino MEGA, Quelle: Arduino (2023), Online-Quelle [15.08.2023]

3.2 PX4 Drohnencontroller

Der PX4 Drohnencontroller ist ein vielseitiges und leistungsfähiges Open-Source-Autopilotsystem, das in einer breiten Palette von unbemannten Luftfahrzeugen (UAVs) eingesetzt wird. Ursprünglich als Forschungsprojekt entwickelt, hat sich der PX4, gemeinsam mit Ardupilot, aufgrund seiner Flexibilität und Anpassbarkeit zu einer Standardlösung in der Drohnenindustrie etabliert. Er ist sowohl für Hobbyanwender als auch für professionelle Entwickler und Unternehmen interessant, da er eine zuverlässige Plattform für Drohnensteuerung und autonome Flugmanöver bietet.

Der PX4 Drohnencontroller fungiert als zentrales Gehirn einer Drohne. Er verarbeitet Daten von verschiedenen Sensoren, wie Gyroskopen, Beschleunigungssensoren, Magnetometern und GPS-Modulen, um die Position und Haltung des Fahrzeugs zu bestimmen. Mit diesen Informationen kann der PX4 die Steuerung von Motoren und anderen Aktuatoren übernehmen, um eine stabile Fluglage und präzise Navigation zu gewährleisten.

Der Controller unterstützt unterschiedliche Flugmodi wie manuellen Betrieb, automatische Stabilisierung und autonome Missionen auf Basis von Wegpunkten. Dadurch ist der PX4 sowohl für Freizeitanwendungen als auch für komplexe kommerzielle oder industrielle Einsätze geeignet.

Der größte Vorteil des PX4 Systems liegt in seiner Open-Source-Architektur. Die Software kann nach den spezifischen Anforderungen des Anwenders modifiziert und angepasst werden. Entwickler können neue Funktionen hinzufügen oder bestehende Flugalgorithmen optimieren. Die Open-Source-Community um PX4 trägt kontinuierlich zur Verbesserung des Systems bei, sodass regelmäßig neue Features und Updates bereitgestellt werden.

PX4 ist nicht auf eine bestimmte Art von Drohnen beschränkt, sondern unterstützt mehrere Plattformen, darunter Multirotor-Drohnen, Starrflügler und Hybrid-Drohnen. Dies macht es flexibel für unterschiedlichste Anwendungen.

PX4 wird in einer Vielzahl von Einsatzgebieten genutzt. In der kommerziellen Luftfahrt kommt es für Aufgaben wie Luftbildfotografie, Kartierung und Vermessung zum Einsatz. In der Landwirtschaft unterstützt PX4 Drohnen bei der Überwachung von Feldern und Ernten, während in der Industrie Drohnen zur Inspektion von Anlagen und Infrastruktur verwendet werden. Darüber hinaus wird PX4 in der Forschung eingesetzt, wo Entwickler/-innen neue Technologien und Algorithmen testen, um die Fähigkeiten von Drohnen zu erweitern. [12]

[12] PX4 (2023), Online-Quelle [15.08.2023]

3.3 Pixhawk 6X

Der Pixhawk 6X ist ein hochentwickelter Flugcontroller, der für die präzise Steuerung von unbemannten Luftfahrzeugen (UAVs) entwickelt wurde. Er gehört zur bekannten Pixhawk-Reihe und zeichnet sich durch hohe Zuverlässigkeit und moderne Hardware aus, die den Anforderungen komplexer Drohnenanwendungen gerecht werden. Entwickelt für verschiedene Flugplattformen wie Multirotoren, Starrflügler und VTOL-Drohnen, bietet er hohe Flexibilität.

Der Pixhawk 6X basiert auf einem leistungsstarken STM32 H7-Mikrocontroller. Dies bedeutet eine deutliche Verbesserung der Rechenkapazität im Vergleich zu Vorgängermodellen. Die erweiterte Rechenleistung erlaubt präzisere und stabilere Flugmanöver, selbst bei komplizierten Aufgaben oder in anspruchsvollen Umgebungen.

Sensorintegration spielt eine zentrale Rolle in der Funktionsweise des Controllers. Er verfügt über doppelte Gyroskope, Beschleunigungssensoren, Magnetometer, Barometer und GPS. Diese Sensoren liefern wichtige Echtzeitdaten zur Lage, Höhe und Bewegung des UAVs. Dank der Redundanz der Sensoren wird eine zusätzliche Sicherheitsstufe erreicht, die besonders bei kritischen Einsätzen wie Inspektions- oder Rettungsmissionen unerlässlich ist.

Für die Kommunikation mit externen Geräten bietet der Pixhawk 6X eine Vielzahl von Schnittstellen, darunter I²C, SPI, CAN, UART und PWM. Dies ermöglicht die Integration von weiteren Sensoren, Kameras, Kommunikationsmodulen oder anderen externen Geräten.

In der Industrie wird der Pixhawk 6X für die Inspektion von Anlagen und Infrastrukturen eingesetzt. Besonders bei der Überprüfung von schwer zugänglichen oder gefährlichen Bereichen wie Stromleitungen, Windturbinen oder Brücken kommt seine hohe Zuverlässigkeit zum Tragen. Durch seine robuste Architektur und präzise Steuerungsfähigkeiten eignet er sich ideal für den Einsatz in kritischen Umgebungen, in denen Sicherheit und Stabilität höchste Priorität haben. [13]

Abbildung 10: Pixhawk 6X Set, Quelle: PX4 (2023), Online-Quelle [15.08.2023]

[13] PX4 (2023), Online-Quelle [15.08.2023]

4 UWB KIT

4.1 Allgemeines

Das MK UWB Kit SR150/SR040 (im Folgenden als MK UWB Kit bezeichnet) beinhaltet eine Reihe von Hardware-, Software-Tools und Dokumentationen, wie in Abbildung 11 zu sehen, um schnell Prototypen von UWB-basierten Lösungen und Anwendungsfällen zu erstellen. Durch die Nutzung der erweiterten UWB-Funktionen und -Fähigkeiten der UWB SR150 und SR040 von NXP stellt das MK UWB Kit das gesamte Hardware- und Softwarematerial zur Verfügung, das zur Bewertung der UWB-Standort- und Positionierungsfunktionen, einschließlich genauer Entfernungsmessung und AoA, erforderlich ist.

Das MK UWB Kit enthält das MK UWB SDK, ein komplettes Softwareentwicklungspaket mit Bibliotheken, Demoanwendungen und Beispielcode zur schnellen Entwicklung von UWB-Anwendungen für NXP SR150- und SR040.

Darüber hinaus bietet das MK UWB Kit Zugriff auf praktisches Schulungsmaterial, das die Einführung vorhandener Anwendungsfälle erläutern und erleichtern soll. Durch die Verwendung des vorhandenen MK UWB SDK können eigene Anwendung entwickeln und auch vorhandene MK UWB Kit-Komponenten in die Lösungen integrieren werden. [14]

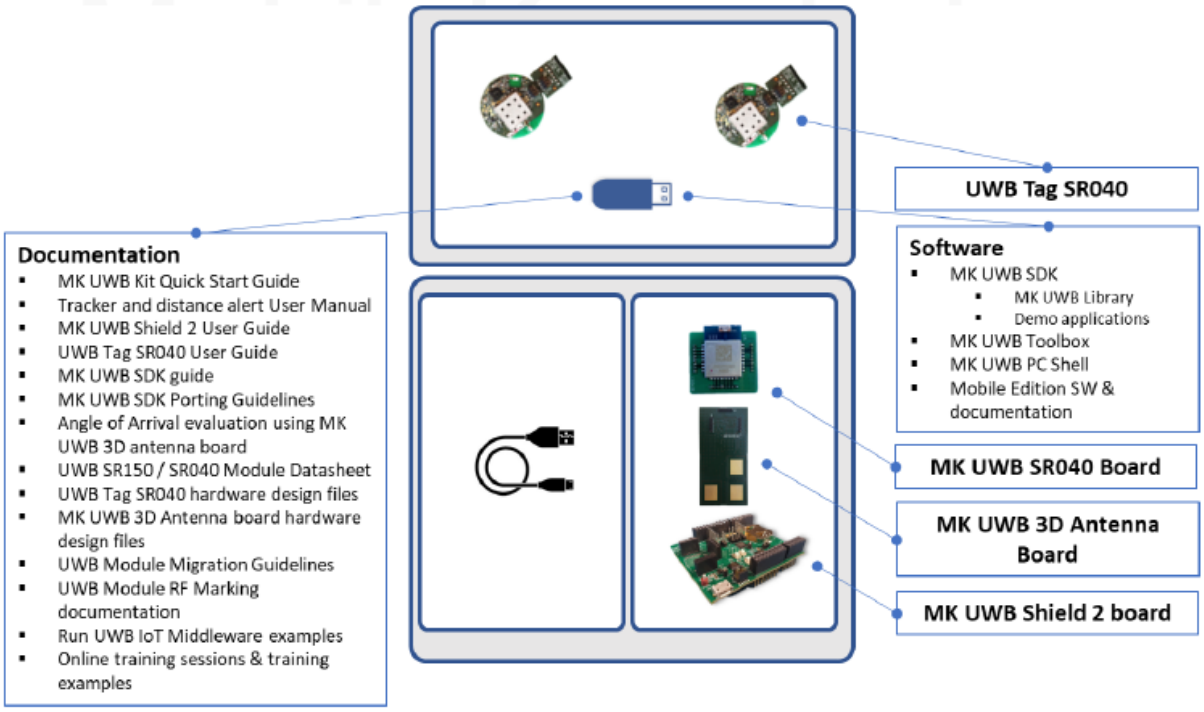


Abbildung 11: Inhalt des MK UWB Kit, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

[14] MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.2 Hardwareinhalt des MK UWB Kit

4.2.1 MK UWB Shield 2

Das MK UWB Shield 2 ist eine leistungsstarke, flexible Plattform für die schnelle Entwicklung und das Prototyping von Anwendungsfällen basierend auf UWB-Technologie. Das MK UWB Shield 2 ist die zentrale Hardwarekomponente des MK UWB Kits. Es kann mit anderen im UWB-Kit enthaltenen Hardwarekomponenten verbunden werden, um verschiedene Anwendungsfälle zu ermöglichen oder über die Erweiterungssteckplätze und Header mit benutzerdefinierten Hardwarekomponenten erweitert werden.

Abbildung 12: MK UWB Shield 2, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Die UWB-Funktionalitäten des MK UWB Shield 2 Boards werden durch ein UWB-Modul ermöglicht. Das UWB-Modul ist eine eigenständige Komponente, die sich in jedes Design integrieren lässt, da sie über den NXP Trimension SR150 verfügt und die notwendigen Schnittstellenpins zur Steuerung durch einen Host. Das Modul kann direkt als Bauteil mit dem MK UWB Shield 2 Board verlötet oder mit dem MK UWB Shield 2 verbunden werden.

Abbildung 13 zeigt die Hauptelemente des MK UWB Shield 2 Boards. Die verwendete Host-MCU ist der QN9090 von NXP, welcher über integrierte Bluetooth LE- und NFC-Funktionen verfügt. Die SPI-Signale des UWB-Moduls können entweder an den QN9090 oder an die Arduino-kompatiblen Stiftleisten weitergeleitet werden, um einer externen MCU-Karte die Steuerung des UWB-Moduls zu ermöglichen. [15]

[15] MobileKnowledge (2023), Online-Quelle [15.07.2023]

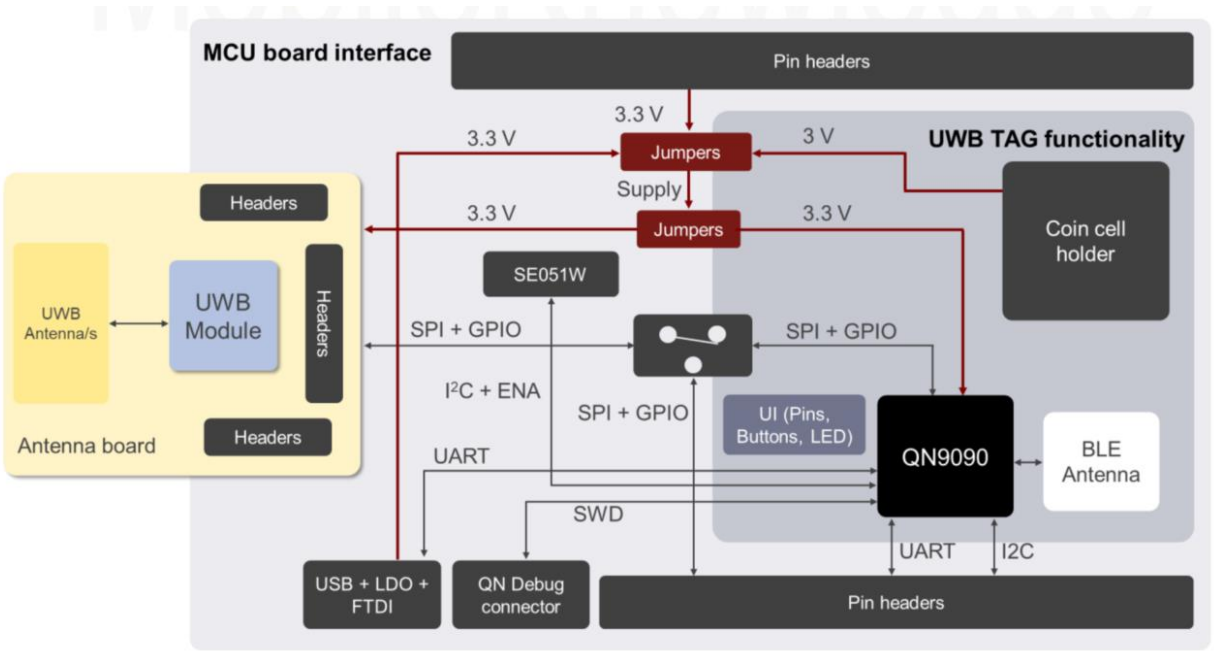


Abbildung 13: MK UWB Shield 2 Blockdiagramm, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Das MK UWB Shield 2 ermöglicht je nach angestrebtem Anwendungsfall unterschiedliche Entwicklungsszenarien. Die relevantesten Konfigurationen sind folgend angeführt:

- Stand Alone-Gerät

Das MK UWB Shield 2 UWB-Modul ist batteriebetrieben und kann als autonomes Gerät genutzt werden. Es ist möglich, das UWB-Modul entweder direkt auf die MK UWB Shield 2-Platine zu löten oder die MK UWB 3D-Antennenplatine mit gelötetem Trimension SR150-Modul oder die MK UWB SR040-Platine mit gelötetem Trimension SR040-Modul zu verwenden.

- Gerät mit mehreren Antennen

Das UWB-Modul SR150 kann über die im UWB- Kit enthaltene MK UWB 3D-Antennenplatine mit dem MK UWB Shield 2 verbunden werden. Das 3D-Antennenboard verfügt über drei Patch-Antennen, um 2D- und 3D-AoA1-Messungen mit dem Trimension SR150-Chip durchzuführen.

- UWB auf einem Gerät aktivieren

Das MK UWB Shield 2 kann mit einem Smartphone oder einem anderen Host verbunden werden, um die UWB-Funktionalität zu ermöglichen. Die Verbindung kann wahlweise per USB oder Bluetooth LE erfolgen. Das MK UWB Shield 2 UWB-Modul fungiert dann als HW-Erweiterung, um die UWB-Funktionalität in jedem Host-Gerät zu ermöglichen.

- Entwicklungsplattform mit externer MCU

Das MK UWB Shield 2 kann als Erweiterungsplatine für jedes Arduino-kompatible Entwicklungsboard verwendet werden, indem es über seine Erweiterungsstiftleisten angeschlossen wird. Dadurch kann der Benutzer andere Technologien als UWB kombinieren, um komplexere Anwendungsfälle zu erstellen. Zu den möglichen Technologien zur Integration gehören NFC, Bluetooth LE, Wi-Fi, LTE. [16]

[16] MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.2.2 MK UWB 3D Antenna Board

Das MK UWB 3D Antenna Board besitzt drei Patch-Antennen, die 2D- und 3D-AoA-Lesung ermöglichen. Die MK UWB 3D-Antennenplatine integriert das neueste Amotech-Modul mit Trimension SR150 IC. Es kann an das MK UWB Shield 2 angeschlossen werden. [17]

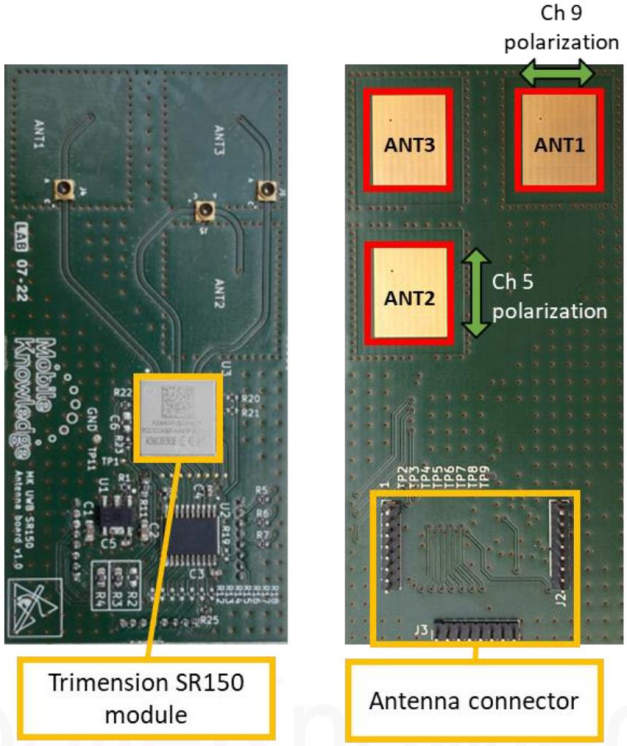


Abbildung 14: MK UWB 3D Antenna Board, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.2.3 UWB Tag SR040

Der UWB Tag SR040 ist ein autonomes, batteriebetriebenes UWB-Gerät, das über den UWB-Chipsatz Trimension SR040 von NXP und einen Bluetooth LE-fähigen Mikrocontroller (NXP QN9090) verfügt. Er beinhaltet alle notwendigen Komponenten und Schnittstellen, um alle Arten von Tracking- und Lokalisierungsanwendungsfällen mithilfe der UWB-Technologie zu ermöglichen.

[17] MobileKnowledge (2023), Online-Quelle [15.07.2023]

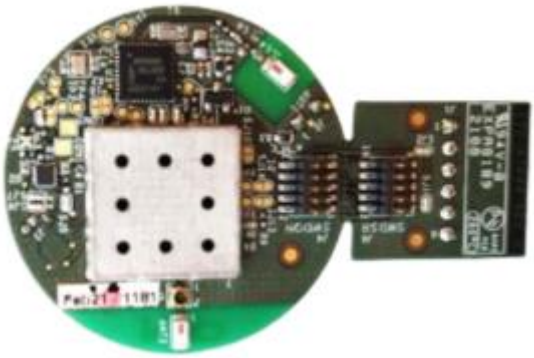


Abbildung 15: UWB Tag SR040, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Die High-Level-Architektur des UWB- Tags SR040 ist in Abbildung 16 dargestellt. Die wichtigsten Architekturblöcke sind:

- NXP QN9090 MCU

Ein leistungsstarker, basierend auf dem Arm Cortex M4, Wireless-Controller mit geringem Stromverbrauch und integrierter Bluetooth LE 5.0.

- Trimension SR040 UWB IC

Ein Hochleistungs-UWB-Chip, der speziell für den Einsatz in batteriebetriebenen IoT-Geräten wie UWB-Tags entwickelt wurde. Der UWB-Chip kommuniziert mit der Host-MCU über die SPI-Schnittstelle.

- Secure Element (SE)

Der UWB Tag SR040 beinhaltet ein EdgeLock SE050 SE für die sichere Speicherung und Verwaltung sensibler Daten und Anmeldeinformationen.

- Bluetooth LE RF und Antenne

Der UWB Tag SR040 verfügt über eine integrierte Bluetooth LE-Antenne und eine Radio Frequency-Verbindung zum Host-MCU, um den Bluetooth-Controller zu aktivieren.

- UWB-HF und -Antenne

Der Trimension SR040-Chip ist mit einer einzelnen UWB-Antenne verbunden, die für die Verwendung mit den Kanälen 5, 6, 8 und 9 optimiert ist.

- Bewegungssensor

Dies ist ein kompakter 3-Achsen-Beschleunigungsmesser für Anwendungen, die bei Bewegung ein Aufwachen mit geringem Stromverbrauch erfordern.

- Knopfzellenhalter

Der UWB Tag SR040 kann mit CR2032-Knopfzellenbatterien betrieben werden und enthält einen Halter zur Platzierung der Batterie. [18]

[18] MobileKnowledge (2023), Online-Quelle [15.07.2023]

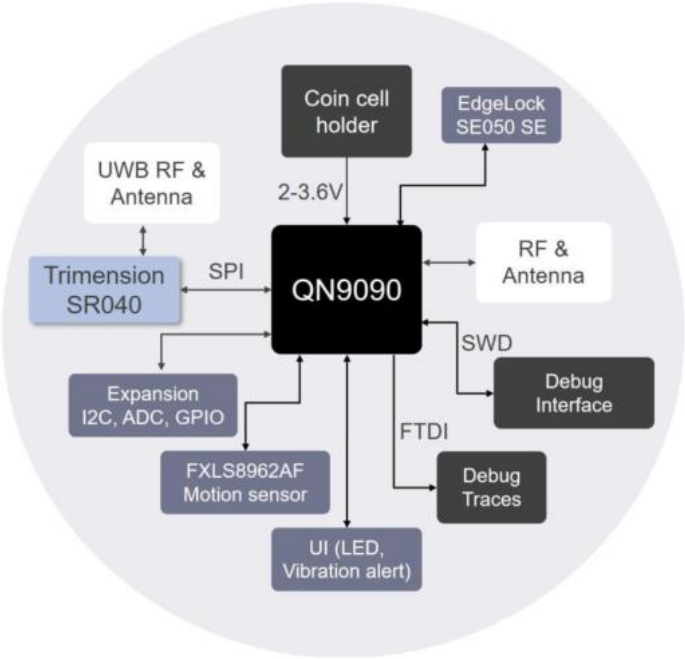


Abbildung 16: UWB Tag SR040 Blockdiagramm, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.2.4 MK UWB SR040 Board

Das MK UWB SR040 Board verfügt über ein Trimension SR040 Modul mit integrierter Antenne. Das Board kann an das MK UWB Shield 2 angeschlossen werden.[19]

Abbildung 17: MK UWB SR040 Board, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

[19] MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.3 Softwareelemente des MK-UWB-Kit

Das MK-UWB-Kit wird mit einem USB-Flash-Laufwerk mit Software- und Dokumentationsressourcen geliefert. Alle Softwareressourcen befinden sich im Softwareverzeichnis des USB-Flash-Laufwerks. Das folgendes Softwarematerial ist darin enthalten

4.3.1 MK UWB SDK

Dies ist ein Softwarepaket zur Vereinfachung der Entwicklung von UWB-Anwendungen auf Basis von NXP Trimension ICs. Es besteht aus:

MK UWB Library Lite – diese stellt eine API (Application Programming Interface) zur schnellen Implementierung von UWB-basierten Lösungen bereit, ohne sich auf Implementierungsdetails auf niedriger Ebene konzentrieren zu müssen. Die im MK UWB Kit enthaltene MK UWB Library Lite-Version stellt nur einige Funktionalitäten der vollständigen MK UWB SDK Library zur Verfügung.

Demos und Beispielen, welche die Fähigkeiten der UWB-Technologie und der MK UWB-Bibliothek demonstrieren.

4.3.2 MK UWB SDK vorkompilierte Binärdateien

Es werden vorkompilierte Binärdateien für den QN9090 für das MK UWB Shield 2 SR150, das MK UWB Shield 2 SR040 und den MK UWB Tag SR040 bereitgestellt. Des Weiteren werden vorkompilierte Binärdateien für den nRF52840 für das MK UWB Shield 2 SR150 und den MK UWB Tag SR040 bereitgestellt.[20]

[20] MobileKnowledge (2023), Online-Quelle [15.07.2023]

4.3.3 MK UWB Toolbox Android

Die Toolbox ist eine Android-Anwendung zur Unterstützung der Interaktion, der Datenanalyse und der Visualisierung für die im MK UWB SDK enthaltenen Beispielanwendungsfälle. Kompilierte APK und Quellcode sind enthalten. Die Anwendung ist auch im Google Play Store verfügbar.

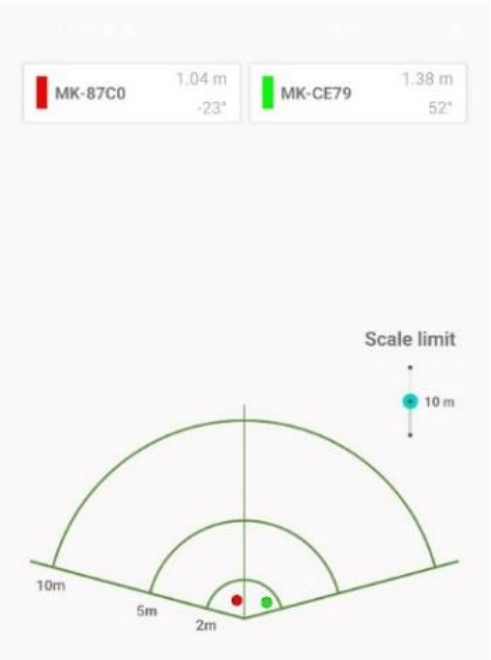


Abbildung 18: Beispiel der MK UWB Toolbox Android, Quelle: Eigene Darstellung.

4.3.4 MK UWB PC Shell-Anwendung

Dabei handelt es sich um eine PC-Anwendung zur Unterstützung der Interaktion, der Datenanalyse und der Visualisierung für die im MK UWB SDK enthaltenen Beispielanwendungsfälle. Der Quellcode ist enthalten. In Abbildung 19 ist ein Beispiel der MK UWB PC Shell bei Verwendung des Anwendungsfalls Abstandswarnung zu sehen:

```
PS C:\Users\nicog\Desktop\UWB_OnlineTraining\MK UWB PC Shell v1.1.0> python .\mk_uwb_pc_shell.py COM5
MK UWB PC Shell - v1.1.0

List of COMMANDS

start_application            Start Tracker or Distance alert application
grant_access                 Grant access to the UWB Lock to a given MAC address
exit                         Exit PC Shell
help                         PC Shell commands description

Command usage: cmd [arg]
  start_application tracker
  start_application distance_alert
  help
  exit
> start_application distance_alert
Sending: 0x1 0x0 0x2 0x10 0x3

Starting distance_alert application...

Distance Alert application has been started.

Looking for nearby devices... Please, press CTRL + C to stop distance alert application.

[2023-05-04 14:13:09] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 26 cm
[2023-05-04 14:13:09] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 27 cm
[2023-05-04 14:13:10] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 23 cm
```

Abbildung 19: Beispiel der MK UWB PC Shell-Anwendung, Quelle: Eigene Darstellung.

5 SOFTWARE

5.1 NXP MCUXpresso IDE

MCUXpresso ist eine integrierte Entwicklungsumgebung (Integrated Development Environment, IDE) für die Programmierung von Mikrocontrollern aus der Familie von NXP Semiconductors. NXP ist ein bekannter Hersteller von Halbleitern und Mikrocontrollern, der eine breite Palette von Produkten für verschiedene Anwendungen anbietet, beispielsweise für die Automobilindustrie und das IoT.

MCUXpresso bietet eine umfassende Suite von Tools und Ressourcen, die Entwicklern helfen, Anwendungen für NXP-Mikrocontroller zu entwickeln, zu debuggen und zu testen.

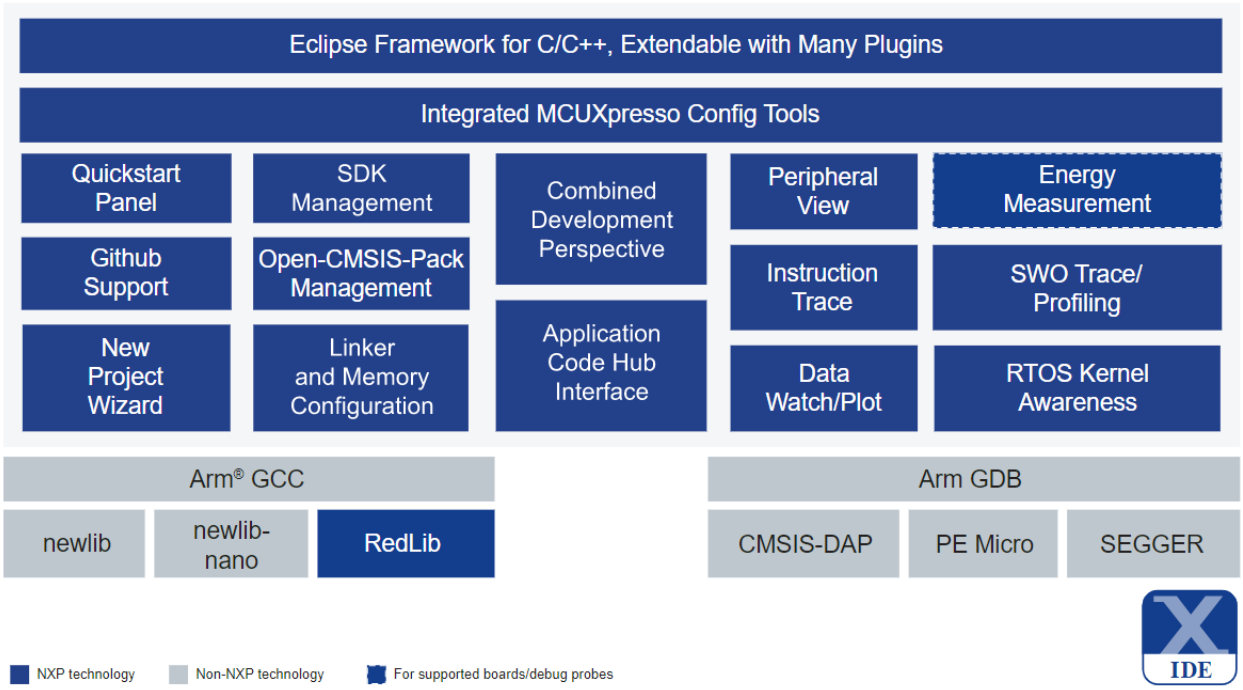


Abbildung 20: MCUXpresso IDE Block Diagramm, Quelle: NXP (2023), Online-Quelle [20.07.2023]

Hauptmerkmale von MCUXpresso:

- IDE

MCUXpresso bietet eine integrierte Entwicklungsumgebung mit Unterstützung für die Programmierung in C und C++. Die IDE bietet Funktionen wie Syntaxhervorhebung, Codevervollständigung und Projektmanagement.

- SDK (Software Development Kit)

Die Entwicklungsumgebung wird mit einem umfangreichen Software Development Kit geliefert, das eine Vielzahl von Softwarebibliotheken, Treibern und Beispielen für die Entwicklung von Anwendungen auf NXP-Mikrocontrollern bereitstellt. Diese SDKs sind speziell auf die Hardwarearchitektur der NXP-Mikrocontroller zugeschnitten und erleichtern dadurch die Entwicklung von Anwendungen.

- Debugger

MCUXpresso bietet einen leistungsstarken Debugger, der es Entwicklern ermöglicht, ihre Anwendungen zu debuggen und zu testen. Der Debugger unterstützt Funktionen wie Einzelschritt-Ausführung, Haltepunkte und Variablenüberwachung.

- Konfigurationstools

Konfigurationstools, mit denen Entwickler die Peripheriegeräte und Funktionen der NXP-Mikrocontroller konfigurieren können, sind im MCUXpresso enthalten. Diese Tools erleichtern die Konfiguration von GPIOs, UARTs, SPIs, I2C und anderen Peripheriegeräten.

- RTOS-Unterstützung

MCUXpresso unterstützt die Integration von Echtzeitbetriebssystemen (RTOS) wie FreeRTOS, um die Entwicklung komplexer eingebetteter Anwendungen zu erleichtern. [21]

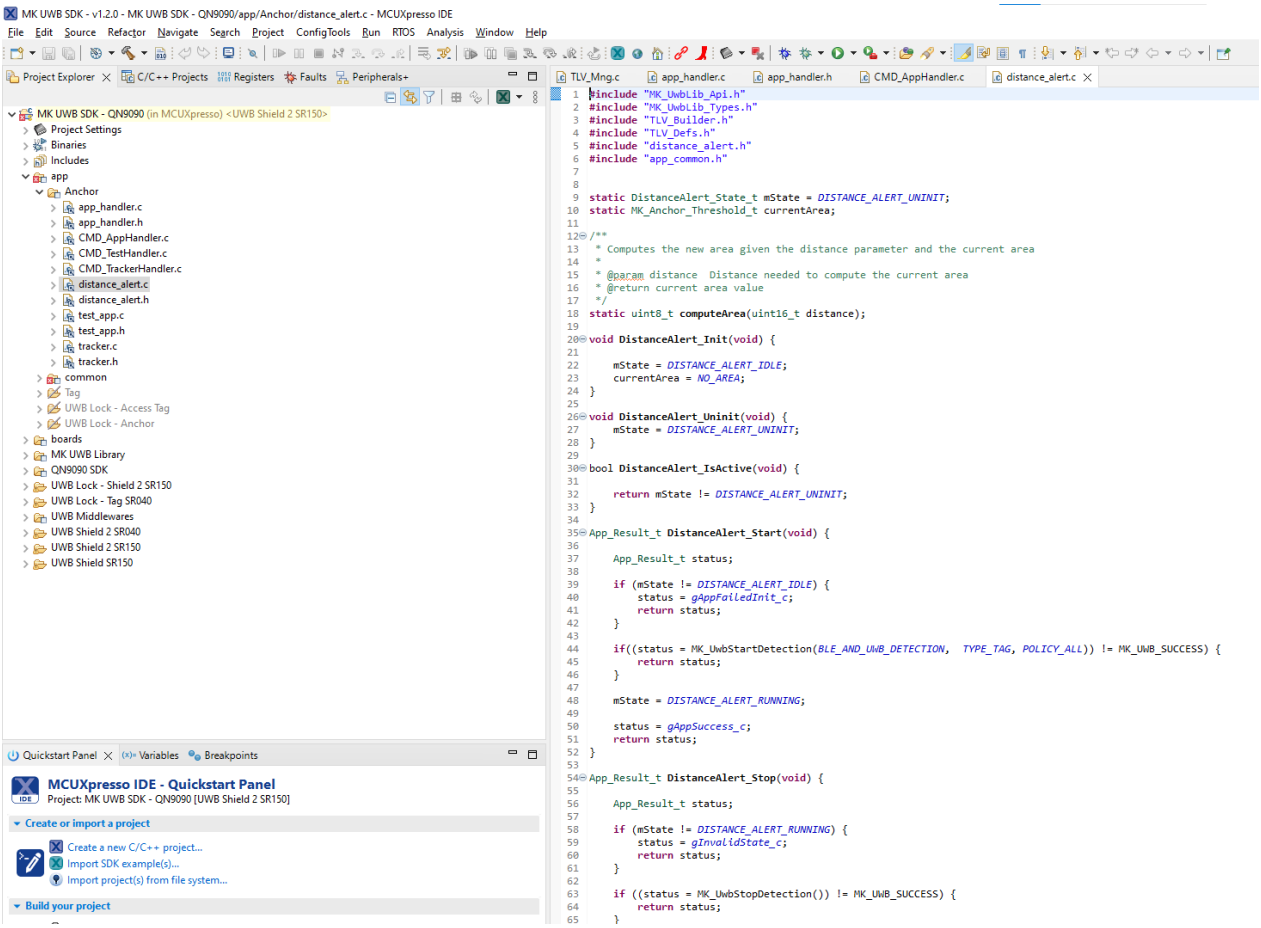


Abbildung 21: NXP MCUXpresso IDE, Quelle: Eigene Darstellung.

[21] NXP (2023), Online-Quelle [20.07.2023]

5.2 Arduino IDE

Die Arduino IDE ist eine Softwareanwendung, die speziell für die Programmierung von Arduino-Mikrocontrollern entwickelt wurde. Sie bietet eine benutzerfreundliche Umgebung, in der Entwickler Arduino-Code schreiben, kompilieren und auf Arduino-Boards hochladen können.

Die Arduino IDE bietet eine Vielzahl von Funktionen, um Entwicklern bei der Erstellung von Arduino-Code zu unterstützen:

Der integrierte Texteditor ermöglicht das Schreiben von Arduino-Code mit Funktionen wie Syntaxhervorhebung, automatischer Einrückung und Codevervollständigung, um die Entwicklung zu erleichtern.

Eine umfassende Sammlung von Code-Bibliotheken deckt häufig verwendete Funktionen und Hardware-Komponenten ab, wodurch die Integration von Sensoren, Aktuatoren und anderen elektronischen Bauteilen in Projekte deutlich vereinfacht wird.

Die IDE unterstützt eine Vielzahl von Arduino-Boards, einschließlich verschiedener Modelle und Varianten, und ermöglicht es Entwicklern, das gewünschte Board auszuwählen, bevor sie mit dem Codieren beginnen.

Durch Tools zur Kompilierung des Codes in ein ausführbares Format und zum Hochladen auf das ausgewählte Arduino-Board erleichtert die IDE den Entwicklungsprozess. Dies erfolgt in der Regel über eine USB-Verbindung.

Ein serieller Monitor ist in die IDE integriert, der es Entwicklern ermöglicht, die Kommunikation zwischen dem Arduino-Board und dem Computer zu überwachen. Dies ist besonders nützlich für die Fehlerbehebung und das Debuggen von Code. [22]

[22] Arduino (2023), Online-Quelle [01.09.2023]

Abbildung 22: Arduino IDE, Quelle: Arduino (2023), Online-Quelle [01.09.2023]

5.3 Programmiersprachen

5.3.1 Python

Python ist eine Programmiersprache, die von Guido van Rossum in den späten 1980er Jahren entwickelt und erstmals 1991 veröffentlicht wurde. Sie zeichnet sich durch eine klare und leserliche Syntax aus, die darauf abzielt, die Programmierung intuitiver und produktiver zu gestalten. Es ist eine vielseitige Sprache, die in verschiedenen Anwendungsgebieten eingesetzt werden kann, darunter Webentwicklung, Datenanalyse, maschinelles Lernen, wissenschaftliche Berechnungen, Automatisierung und Spielentwicklung.

Hier sind einige der wichtigsten Merkmale von Python:

Python legt großen Wert auf Lesbarkeit und eine klare Syntax, welches es zu einer bevorzugten Wahl für Anfänger und erfahrene Entwickler gleichermaßen macht.

Mit einer breiten Palette von Bibliotheken und Frameworks ist Python für verschiedene Anwendungen und Branchen geeignet, wodurch seine Vielseitigkeit unterstrichen wird.

Python-Code wird zur Laufzeit von einem Interpreter ausgeführt, was bedeutet, dass der Code Zeile für Zeile getestet und ausgeführt werden kann, ohne ihn vorher kompilieren zu müssen.

Python ist plattformunabhängig, sodass derselbe Code auf verschiedenen Betriebssystemen ohne Änderungen ausgeführt werden kann, womit die Portabilität erhöht wird.

Mit einer äußerst aktiven und engagierten Entwicklergemeinschaft wird Python kontinuierlich verbessert und bietet eine breite Palette von Bibliotheken und Frameworks.

Python unterstützt verschiedene Programmierparadigmen, darunter objektorientierte, funktionale und imperative Programmierung und bietet Entwicklern dadurch hohe Flexibilität.

Python wird mit einer umfangreichen Standardbibliothek geliefert, die zahlreiche nützliche Module und Funktionen umfasst. Dies erleichtert die Entwicklung von Anwendungen und steigert die Produktivität. [23]

5.3.2 C/C++

C und C++ sind zwei eng miteinander verwandte Programmiersprachen, die in der Softwareentwicklung weit verbreitet sind.

5.3.2.1 C-Programmiersprache

Die Programmiersprache C ist eine bedeutende Hochsprache, die in den frühen 1970er Jahren von Dennis Ritchie bei Bell Labs entwickelt wurde. C zeichnet sich durch die folgenden Merkmale aus:

C wurde entwickelt, um einfache und effiziente Programmierung zu fördern, indem grundlegende Konstrukte bereitgestellt werden, die es Entwicklern ermöglichen, direkt mit der Hardware zu arbeiten und einen optimierten Code zu schreiben.

Die Portabilität von C ermöglicht es, dass Programme auf verschiedenen Plattformen und Betriebssystemen ausgeführt werden können, da die Sprache hardwareunabhängig ist, was bedeutet, dass derselbe C-Code auf verschiedenen Computern kompiliert und ausgeführt werden kann.

Mit leistungsstarken Kontrollstrukturen wie Schleifen, bedingten Anweisungen und Funktionen ermöglicht C Entwicklern, komplexe Aufgaben zu lösen.

Die Funktionen und Modularität von C erlauben die Aufteilung von Programmen in einzelne Funktionen, wodurch der Code leichter wartbar und erweiterbar wird, ein entscheidender Aspekt der C-Programmierung. **Fehler! Textmarke nicht definiert.**

Die hohe Leistung von C-Code resultiert aus der Kontrolle, die Entwickler über die Speicher- und Ressourcennutzung besitzen, wodurch C sich besonders für die Programmierung von Betriebssystemen, Treibern und Anwendungen mit hohen Leistungsanforderungen eignet.

C wird in verschiedenen Anwendungsgebieten eingesetzt, einschließlich Betriebssystementwicklung, eingebetteter Programmierung, Spieleentwicklung und wissenschaftlicher Berechnungen, wodurch seine Vielseitigkeit deutlich wird.

Mit einer großen und aktiven Entwicklergemeinschaft, die Bibliotheken und Frameworks erstellt hat, die die C-Programmierung erleichtern und erweitern, profitiert C von einer starken Unterstützung durch die Community.

Obwohl C eine leistungsstarke und vielseitige Programmiersprache ist, erfordert sie von Entwicklern eine größere Sorgfalt bei der Verwaltung von Speicher und Ressourcen im Vergleich zu moderneren Sprachen. Trotzdem bleibt C eine wichtige Sprache in der Softwareentwicklung und wird in vielen kritischen Anwendungen eingesetzt.

[23] Steyer (2018), S. 2 ff.

5.3.2.2 C++-Programmiersprache

Die Programmiersprache C++ ist eine Weiterentwicklung der Programmiersprache C, die in den 1980er Jahren von Bjarne Stroustrup entwickelt wurde. C++ baut auf den Funktionen von C auf, erweitert diese jedoch um wichtige Konzepte der objektorientierten Programmierung. Hier sind einige wichtige Merkmale und Eigenschaften:

C++ ermöglicht objektorientierte Programmierung (OOP) durch die Verwendung von Klassen und Objekten, welche die Modellierung von Daten und die Organisation von Code in einer objektorientierten Struktur erleichtert. Diese Herangehensweise fördert die Wiederverwendbarkeit, Erweiterbarkeit und Strukturierung von Code.

Die Definition von Klassen in C++ dient als Baupläne für Objekte, und die Vererbung ermöglicht es, bestehende Klassen zu erweitern und wiederverwendbare Code-Teile zu erstellen.

Durch die Unterstützung von Polymorphismus können verschiedene Objekte einer gemeinsamen Oberklasse sich auf unterschiedliche Weise verhalten, womit die Erstellung von flexiblem Code erleichtert wird, der auf verschiedene Situationen reagieren kann.

C++ bietet Templates, mit denen generischer Code geschrieben werden kann. Durch Templates können Funktionen und Klassen erstellt werden, die mit verschiedenen Datentypen arbeiten können, ohne den Code für jeden Datentyp neu schreiben zu müssen.

Die Standard Template Library (STL) in C++ ist eine leistungsstarke Sammlung von vorgefertigten Datenstrukturen und Algorithmen. Sie enthält Container wie Vektoren, Listen und Maps sowie Sortier- und Suchalgorithmen.

Durch die direkte Kontrolle über Speicher und Ressourcen ermöglicht C++ die Entwicklung effizienter und leistungsstarker Anwendungen, insbesondere in Bereichen mit hohen Leistungsanforderungen.

C++ findet in verschiedenen Anwendungsbereichen Einsatz, darunter Spieleentwicklung, Systemprogrammierung, Echtzeitsoftware und wissenschaftliche Berechnungen.

Neben der objektorientierten Programmierung unterstützt C++ auch andere Programmierparadigmen wie die prozedurale und generische Programmierung, wodurch seine Vielseitigkeit und Anpassungsfähigkeit unterstrichen werden.

C++ ist eine leistungsstarke und vielseitige Programmiersprache, die in vielen Branchen weit verbreitet ist. Sie bietet Entwicklern die Flexibilität und Kontrolle, um komplexe Anwendungen zu erstellen, während sie gleichzeitig modernen Softwareentwicklungsprinzipien, wie OOP und generischer Programmierung, gerecht wird. [24]

Ein Beispiel des Codes ist in Abbildung 23 ersichtlich:

[24] Gerd Küveler (2017), S. 9 ff.

```
void loop()
{
  if(Serial1.available()>0)
  {

    int total_read=Serial1.readBytesUntil('\n',data_rx,32);
    data_rx[total_read]='\0';

    for (int i=0;i<total_read;i++)
    {
      Serial.print(data_rx[i],HEX); // Print value received as hex
      Serial.print(' ');
    }

    Serial.print('\n');

  }
```

Abbildung 23: C++ Beispielcode, Quelle: Eigene Darstellung.

5.4 UART-Kommunikation

5.4.1 Allgemein

Ein Universal Asynchronous Receiver Transmitter (UART) funktioniert durch die Übertragung von Daten als eine Reihe von Bits. Dazu gehören ein Startbit, Datenbits, ein optionales Paritätsbit und ein Stoppbit, welche im Abschnitt 5.4.3 noch genauer erklärt werden. Im Gegensatz zur parallelen Kommunikation, bei der mehrere Bits gleichzeitig übertragen werden, sendet UART Daten seriell, Bit für Bit, wie in Abbildung 24 zu erkennen. Wie der Name schon sagt, arbeitet das Protokoll asynchron, was bedeutet, dass es nicht auf ein gemeinsames Taktsignal angewiesen ist. Stattdessen werden vordefinierte Baudraten, unter Abschnitt 5.4.2.1 genauer erklärt, verwendet, um das Timing der Datenbits zu bestimmen.

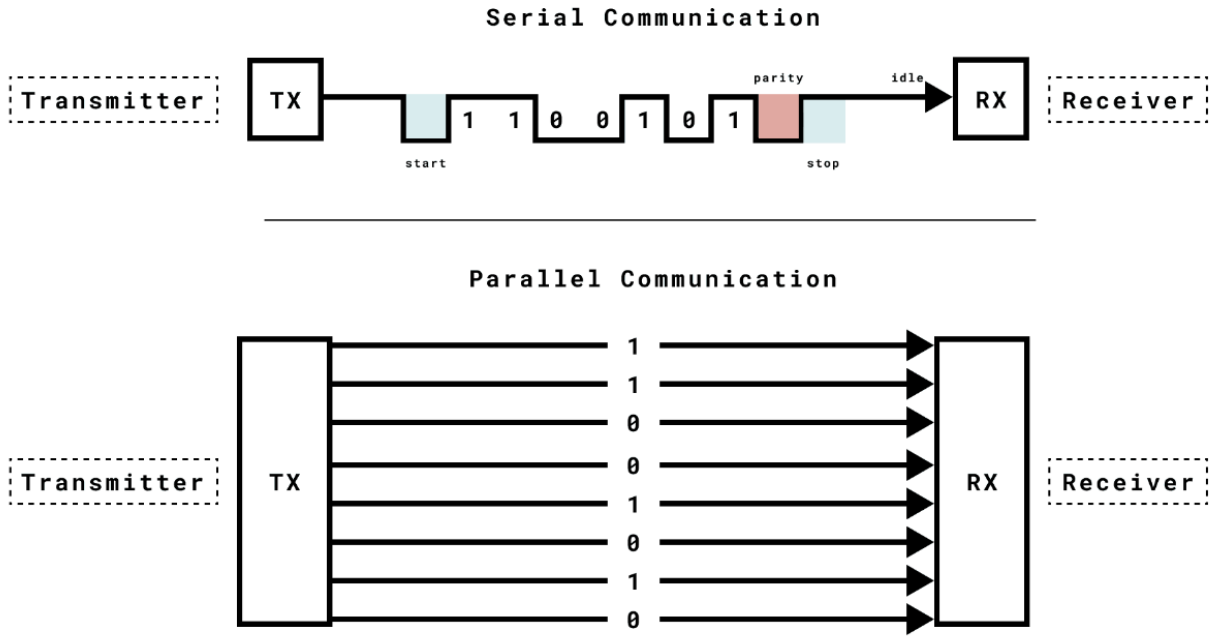


Abbildung 24: Unterschied serielle / parallele Kommunikation, Quelle: Arduino (2023), Online-Quelle [28.08.2023]

Zu den Schlüsselkomponenten von UART gehören Sender, Empfänger und Baudrate. Der Sender sammelt Daten von einer Quelle, formatiert sie in serielle Bits und sendet sie über einen TX-Pin (Transmitter). Der Empfänger erhält diese über einen RX-Pin (Receiver), verarbeitet eingehende serielle Daten und wandelt sie in parallele Daten für das Hostsystem um. Die Baudrate bestimmt die Geschwindigkeit der Datenübertragung. [25]

5.4.2 Synchronisation

Timing und Synchronisation sind entscheidende Aspekte der UART-Kommunikation. Im Gegensatz zu synchronen seriellen Kommunikationsprotokollen wie SPI und I2C arbeitet UART asynchron. Das bedeutet, UART nicht auf ein gemeinsames Taktsignal angewiesen ist, um die Datenübertragung zu

[25] Arduino (2023), Online-Quelle [28.08.2023]

koordinieren. Stattdessen werden vordefinierte Baudraten verwendet, um das Timing der Datenbits zu bestimmen.

5.4.2.1 Baudrate

Die Baudrate ist ein wesentlicher Parameter in der UART-Kommunikation und bestimmt die Geschwindigkeit, mit der Daten über den Kommunikationskanal übertragen werden. Sie wird in Bits pro Sekunde (bps) gemessen und gibt an, wie viele Zeichen pro Sekunde in der Kommunikationsverbindung übertragen werden. Ein Zeichen kann entweder ein einzelnes Bit oder eine Kombination von Bits sein. Für eine erfolgreiche Kommunikation müssen sowohl das sendende als auch das empfangende Gerät dieselbe Baudrate verwenden.

Die Bedeutung der Baudrate liegt in ihrem direkten Einfluss auf die Datenübertragungsgeschwindigkeit. Einerseits ermöglicht eine höhere Baudrate eine schnellere Datenübertragung, erfordert aber auch eine genauere Taktsynchronisation zwischen Sender und Empfänger. Andererseits kann eine niedrigere Baudrate für Anwendungen geeignet sein, bei denen die Synchronisation weniger wichtig ist, eine Kompatibilität mit älteren Systemen oder eine länger Übertragungsstrecke gefordert ist. Bei der Programmierung sind gängige Baudraten 9600, 115200, 4800 und 57600.

5.4.2.2 Flow Control

UART Flow Control ist eine Methode, mit der langsame und schnelle Geräte über UART miteinander kommunizieren können, ohne dass das Risiko eines Datenverlusts besteht. Ein typisches Beispiel ist die Kommunikation zwischen zwei Einheiten über UART. Ein Sender T sendet einen langen Bytestrom an einen Empfänger R. R ist ein langsameres Gerät als T und R kann nicht mithalten. Es muss entweder die Daten verarbeiten oder einige Puffer leeren, bevor es weiterhin Daten empfangen kann.

R muss T anweisen, für eine Weile mit der Übertragung aufzuhören. Hier kommt die Flusskontrolle ins Spiel. Die Flusskontrolle bietet zusätzliche Signalisierung, um den Sender darüber zu informieren, dass er die Übertragung stoppen (pausieren) oder starten (fortsetzen) soll.

Es gibt verschiedene Formen der Flusskontrolle. Beispielsweise verwendet die Hardware-Flusskontrolle zusätzliche Leitungen, wobei der Logikpegel auf diesen Leitungen definiert, ob der Sender weiterhin Daten senden oder aufhören soll. Bei der Software-Flusskontrolle werden Sonderzeichen über die normalen Datenleitungen gesendet, um die Übertragung zu starten oder zu stoppen. [26]

5.4.3 Nachricht

Bei der UART-Kommunikation ist jeder Datenrahmen durch Start- und Stoppbits gekapselt, wie in Abbildung 25 dargestellt. Diese Bits spielen eine entscheidende Rolle bei der Festlegung der Grenzen der Datenübertragung und der Sicherstellung der Synchronisierung zwischen Sender und Empfänger.

[26] Arduino (2023), Online-Quelle [28.08.2023]

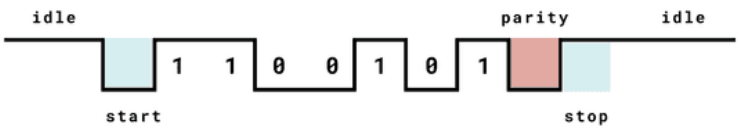


Abbildung 25: Aufbau UART Nachricht, Quelle: Arduino (2023), Online-Quelle [28.08.2023]

5.4.3.1 Startbit

Zu Beginn jedes UART-Rahmens wird ein einzelnes Startbit übertragen. Der Hauptzweck des Startbits besteht darin, den Beginn der Datenübertragung anzuzeigen und den Empfänger auf den Datenempfang vorzubereiten.

Das Startbit ist bei der UART-Kommunikation immer logisch niedrig (0). Das bedeutet, dass das Startbit als ein Spannungspegel übertragen wird, der niedriger ist als der logisch hohe Schwellenwert, normalerweise auf der Empfängerseite.

Wenn der Empfänger ein Startbit erkennt, weiß er, dass ein neuer Datenrahmen beginnt, und er bereitet sich auf den Empfang der eingehenden Bits vor.

5.4.3.2 Stoppbit

In jedem UART-Rahmen werden nach den Datenbits ein oder mehrere Stoppbits gesendet. Das/die Stoppbit(s) signalisieren das Ende des Datenbytes und dienen dazu, den Abschluss der Datenübertragung anzuzeigen. Die gebräuchlichste Konfiguration ist die Verwendung eines Stoppbits, aber in Situationen, in denen zusätzliche Zuverlässigkeit erforderlich ist, können auch zwei Stoppbits verwendet werden.

Die Polarität des/der Stoppbit(s) kann variieren, wobei einige Systeme ein hohes Stoppbit und andere ein niedriges Stoppbit verwenden, je nach der spezifischen UART-Konfiguration.

5.4.3.3 Datenbit

Datenbits sind ein grundlegender Bestandteil der UART-Kommunikation, da sie die eigentlichen zu übertragenden Informationen enthalten. Die Anzahl der Datenbits in einem UART-Frame kann variieren, eine gängige und weit verbreitete Konfiguration sind 8 Bits. UART unterstützt jedoch verschiedene Zeichengrößen, einschließlich 7-Bit- und 6-Bit-Konfigurationen, je nach den spezifischen Anwendungsanforderungen.

Die Zeichengröße bei der UART-Kommunikation wird durch die Anzahl der Datenbits innerhalb eines Rahmens definiert. Es ist wichtig, die richtige Zeichengröße zu wählen, die den Anforderungen der zu übertragenden Daten entspricht. Hier sind einige gängige Konfigurationen der Zeichengröße:

8-Bit: Dies ist die am weitesten verbreitete Datenlänge in der UART-Kommunikation. Sie ermöglicht die Übertragung eines Bytes an Daten, die eine Vielzahl von Werten darstellen können, einschließlich ASCII-Zeichen, numerische Werte und mehr.

7-Bit: In Fällen, in denen die Datengröße kleiner sein muss, wird die 7-Bit-Zeichengröße verwendet. Sie eignet sich für Anwendungen, die weniger Daten-Overhead benötigen, und kann 128 verschiedene Werte darstellen.

6-Bit: Für eine noch kompaktere Datendarstellung kann die 6-Bit-Zeichengröße verwendet werden. Diese Konfiguration ermöglicht die Darstellung von 64 verschiedenen Werten.

Die Datenbits stellen die Zeichen oder Daten in binärer Kodierung dar, wobei jedes Bit einer Zweierpotenz entspricht. Jedes Bit innerhalb des Datenbytes hat eine bestimmte Position und Gewichtung in der binären Darstellung. Diese Kodierung ermöglicht die Übertragung eines breiten Spektrums von Informationen und macht UART vielseitig einsetzbar für verschiedene Datentypen, von einfachen Textzeichen bis hin zu komplexen Binärdaten.

Die Genauigkeit der Datenübertragung bei der UART-Kommunikation hängt von der richtigen Konfiguration der Datenbits ab. Es ist wichtig, dass sich Sender und Empfänger über die Anzahl der Datenbits und deren Kodierung einig sind. Wenn die Konfiguration nicht korrekt ist, kann es zu Datenverfälschungen kommen. Wenn der Sender beispielsweise Daten als 8-Bit-Zeichen sendet, der Empfänger aber so konfiguriert ist, dass er 7-Bit-Zeichen erwartet, können die Daten falsch interpretiert werden und Fehler in den empfangenen Informationen verursachen.

5.4.3.4 Paritätsbit

Zusätzlich zu den Datenbits kann die UART-Kommunikation ein Paritätsbit als Teil des Datenrahmens enthalten. Die Parität ist ein Fehlerkontrollmechanismus, der helfen kann, Datenübertragungsfehler zu erkennen. Sie kann auf „ungerade“ oder „gerade“ eingestellt werden und stellt sicher, dass die Gesamtzahl, der auf eine logische 1 gesetzten Bits in einem Zeichen entweder gerade oder ungerade ist, je nach gewählter Paritätsart. Das Vorhandensein eines Paritätsbits ermöglicht es dem Empfänger, die Integrität der empfangenen Daten zu überprüfen. Wenn die Anzahl der "1"-Bits nicht mit der erwarteten Parität übereinstimmt, wird ein Fehler erkannt. [27]

[27] Arduino (2023), Online-Quelle [28.08.2023]

5.5 TLV

„TLV“ steht für „Type-Length-Value“ und bezeichnet ein Format, das in Abbildung 26 dargestellt ist. Es wird häufig in der Informatik verwendet, insbesondere bei der Datenübertragung und der Speicherung von strukturierten Daten. Das TLV-Format organisiert Daten in drei Teile:

- Type

Dieser Teil gibt den Typ der Daten an, die im TLV-Eintrag enthalten sind. Typen können unterschiedliche Bedeutungen haben und dienen dazu, die Art der enthaltenen Informationen zu kennzeichnen. Sie können als numerische Werte, Buchstaben oder eine Kombination aus beidem dargestellt werden. Zum Beispiel kann der Typ „Name“ bedeuten, dass die folgenden Daten den Namen einer Person enthalten, während der Typ „Alter“ bedeuten kann, dass die folgenden Daten das Alter einer Person angeben. Typen werden üblicherweise als eindeutige Codes oder Schlüssel dargestellt.

- Length

Dieser Teil gibt die Länge der nachfolgenden Daten im TLV-Eintrag an. Die Länge wird normalerweise in Bytes gemessen und gibt an, wie viele Bytes die „Value“ des TLV-Eintrags umfasst. Durch die Angabe der Länge können Empfänger wissen, wie viel Platz für die nachfolgenden Daten reserviert werden muss und wie viele Daten verarbeitet werden müssen.

- Value

Dieser Teil enthält die eigentlichen Daten, die im TLV-Format übertragen oder gespeichert werden sollen. Die Länge dieser Daten wird durch den vorherigen „Length“ -Wert angegeben. Der Wert kann verschiedene Formate haben, je nachdem, welche Art von Informationen er repräsentiert. Dies können Binärdaten, Textzeichenfolgen, numerische Werte oder sogar verschachtelte TLV-Strukturen sein.

```
# TLV_TYPES
TLV_APP_CMD = 0x01
TLV_TRACKER_CMD = 0x02
TLV_DISTANCE_ALERT_CMD = 0x03
TLV_UWB_LOCK_MSG = 0xAA
TLV_TEST_CMD = 0xFF
TLV_NTF = 0x0A
```

Abbildung 26: Beispiele für TLV-Definitionen, Quelle: Eigene Darstellung.

Das TLV-Format bietet eine strukturierte Möglichkeit, Daten zu übertragen oder zu speichern. Es ermöglicht eine flexible und effiziente Handhabung von Daten, da die Typen und Längen der Daten explizit angegeben sind.

Das TLV-Format wird häufig in verschiedenen Anwendungsbereichen und Branchen eingesetzt, darunter:

- Kommunikationsprotokolle

Viele Kommunikationsprotokolle verwenden TLV-ähnliche Strukturen, um Nachrichten und Datenpakete zu kodieren. Dies ermöglicht eine flexible Handhabung von Informationen und erleichtert die Erweiterung des Protokolls, da neue Typen einfach hinzugefügt werden können.

- Speicherformate

TLV-Strukturen werden oft verwendet, um Daten in Speichermedien wie Dateien oder Datenbanken zu speichern. Durch die Verwendung des TLV-Formats können Daten effizient organisiert und später leicht wiederhergestellt werden. [28]

[28] Asche (2016), S. 191 ff.

6 ERSTE SCHRITTE DER PROGRAMMIERUNG

6.1 Programminstallation

Um mit der Entwicklung auf dem MK UWB Hardware-Kit mit dem MK UWB SDK zu beginnen, werden einige Softwarekomponenten benötigt.

6.1.1 Python 3.9

Python 3.9 wird benötigt, um die MK-PC-Shell-Software auszuführen und zu ändern. Diese Software ermöglicht die Kommunikation mit der Kit-Hardware und die Ausführung von Befehlen.

Auf der Seite python.org kann das passende Installationsprogramm für das jeweilige Betriebssystem heruntergeladen werden. Folgend das Installationsprogramm starten und das Kontrollkästchen aktivieren, um Python 3.9 zur Pfadvariablen hinzuzufügen. Danach auf „Jetzt installieren“ klicken und warten, bis die Installation abgeschlossen ist.

6.1.2 NXP MCUXpresso IDE

MCUExpresso IDE wird benötigt, um Code zu kompilieren und die MK Ultra 11-Bibliothek an die gewünschten Anforderungen anzupassen.

Auf der Seite mcuxpresso.nxp.com, kann unter der Registerkarte „Software und Tools“,das passende Installationsprogramm für das jeweilige Betriebssystem heruntergeladen werden. Durch Nutzung der E-Mail-Adresse zur Anmeldung und Akzeptanz der Geschäftsbedingungen wird der Zugang gewährt. Nachfolgend kann das entsprechende Installationsprogramm für das Betriebssystem heruntergeladen und gestartet werden. Die auf den Setup-Bildschirmen präsentierten Anweisungen befolgen, wobei die Zustimmung zur Vereinbarung erforderlich ist, bevor auf "Weiter" geklickt wird. Die Standardoptionen sind für den Großteil der Benutzer ausreichend. Der Installationsvorgang beansprucht einige Minuten

Weiters müssen noch die MK UWB SDK und die QN9090 SDK importiert werden. Die ZIP-Datei der MK UWB SDK muss entpackt werden. In der MCUXpresso IDE auf „Datei“ > „Importieren“ > „Allgemeine vorhandene Projekte in den Arbeitsbereich“ klicken. Den entpackten Ordner suchen und auf „Fertig stellen“ klicken. Zum Importieren der QN9090 SDK ist es nötig die ZIP-Datei des QN9090 SDK mittels Drag & Drop in das Fenster „installierte SDK“ der MCUXpresso IDE zu ziehen.

6.1.3 PyCharm

Von Mobile Knowledge wird eine gebündelte Installation mit PyCharm angeboten. PyCharm ist eine leistungsstarke Entwicklungsumgebung für Python, die Funktionen wie Syntaxhervorhebung, Codevervollständigung und Debugging bietet, um die Entwicklung von Python-Code zu erleichtern.

Auf der Website jetbrains.com steht das Installationsprogramm für das entsprechende Betriebssystem zum Download bereit. Anschließend wird das Installationsprogramm gestartet, und es wird empfohlen, das Kontrollkästchen zu aktivieren, um PyCharm zur Pfadvariablen hinzuzufügen. Danach genügt ein Klick auf "Jetzt installieren", und es wird abgewartet, bis die Installation vollständig abgeschlossen ist.

6.1.4 Arduino IDE

Software von www.arduino.cc downloaden und auf dem PC installieren. Danach öffnet man den Softwareordner und startet das Programm mit der Datei arduino.exe. Zwei wichtige Einstellungen gibt es im Programm zu beachten. Es muss das richtige Board ausgewählt werden, das man anschließen möchte, wie in Abbildung 27 dargestellt. Es muss der richtige „Serial-Port“ ausgewählt werden, damit der PC weiß, an welchem USB-Anschluss das Board angeschlossen ist.

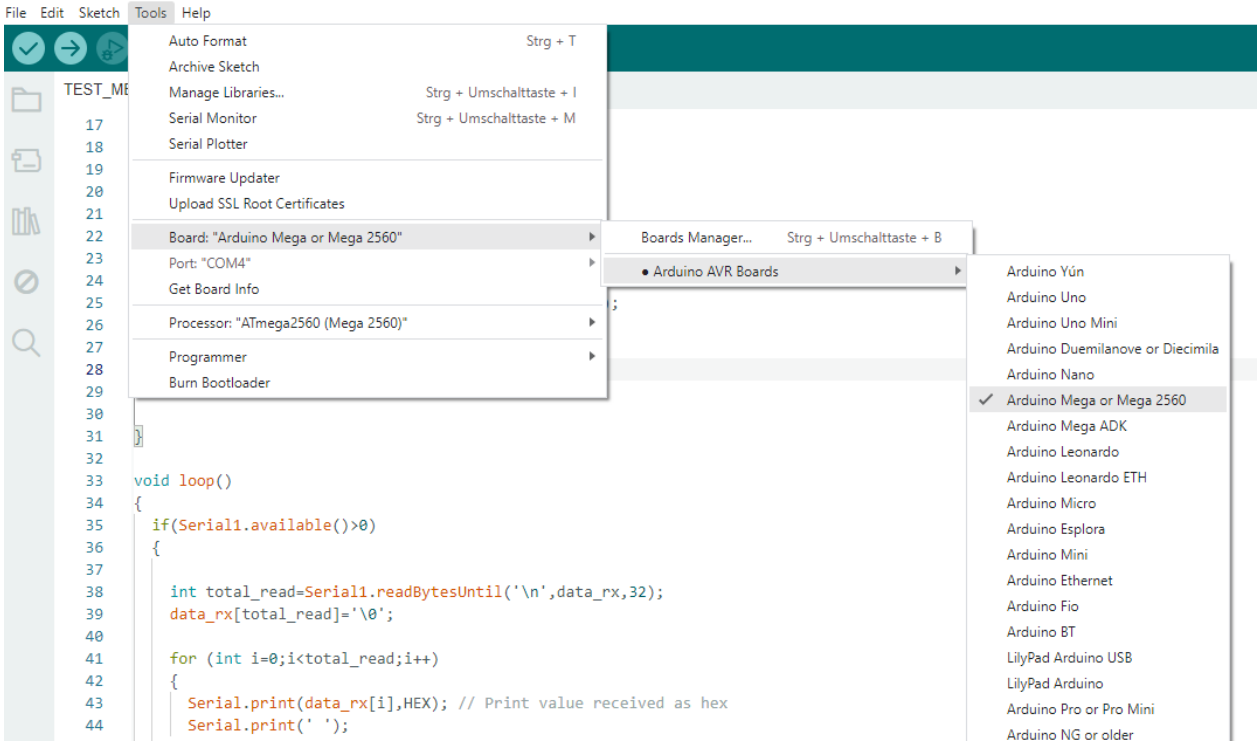


Abbildung 27: Board Auswahl in der Arduino DIE, Quelle: Eigene Darstellung.

6.1.5 MK UWB PC Shell

Die MK UWB PC Shell-Anwendung ist ein Python-CLI-Programm. Zur Ausführung muss Python 3.9 bereits auf dem Rechner installiert sein. Von der Seite ftdichip.com wird eine ZIP-Datei für die DX22-Treiber heruntergeladen und installiert. Dadurch kann der PC über einen seriellen USB-Anschluss mit dem MK UWB Shield 2 kommunizieren. Die Datei mit dem MK UWB PC Shell-Quellcode wird an einem gewünschten Ort am Rechner entpackt. Im entpackten Ordner wird, dann ein PowerShell-Fenster geöffnet.

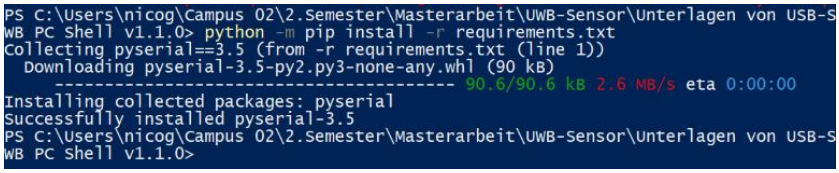


Abbildung 28: Python Installation für die MK UWB PC Shell, Quelle: Eigene Darstellung.

Danach ist in der PowerShell folgende Python- Installation nötig, wie in Abbildung 28 ersichtlich: -m pip install -r requirements.txt. Um die MK UWB PC Shell zu starten, muss vorher noch der richtige USB-Port

aus dem Gerätemanager abgelesen werden. Wie in Abbildung 29 zu sehen, muss folgender Python-Befehl gesendet werden, um die PC Shell zu starten: python mk_uwb_pc_shell.py <COM_port_number>

```
PS C:\Users\nicog\Campus 02\2.Semester\Masterarbeit\UWB-Sensor\Unterlagen von USB-S
WB PC Shell v1.1.0> python .\mk_uwb_pc_shell.py COM5
MK UWB PC Shell - v1.1.0

List of COMMANDS

start_application            Start Tracker or Distance alert application
grant_access                 Grant access to the UWB Lock to a given MAC address
exit                         Exit PC Shell
help                         PC Shell commands description

Command usage: cmd [arg]
  start_application tracker
  start_application distance_alert
  help
  exit
```

Abbildung 29: MK UWB PC Shell Startbildschirm, Quelle: Eigene Darstellung.

6.2 Übungsprojekte

6.2.1 Hardwarevorbereitungen

Um die Demos auszuführen, die in der MK UWB SDK enthalten sind, müssen einige Hardwarevorbereitungen getroffen werden:

Es muss das MK UWB 3D Antenna Board auf das MK UWB Shield 2 Board gesteckt werden. Wichtig ist hier, dass die zwei Boards gut verbunden sind, das heißt die Pins vollständig eingesteckt sind.

Um die Demos für die Android-Anwendungen auszuführen, muss auf dem Mobiltelefon die App MK UWB Toolbox installiert sein. Zuletzt muss das Mobiltelefon mit einem USB-Kabel mit dem MK UWB Shield 2 verbunden werden, wie in Abbildung 30 zu sehen.

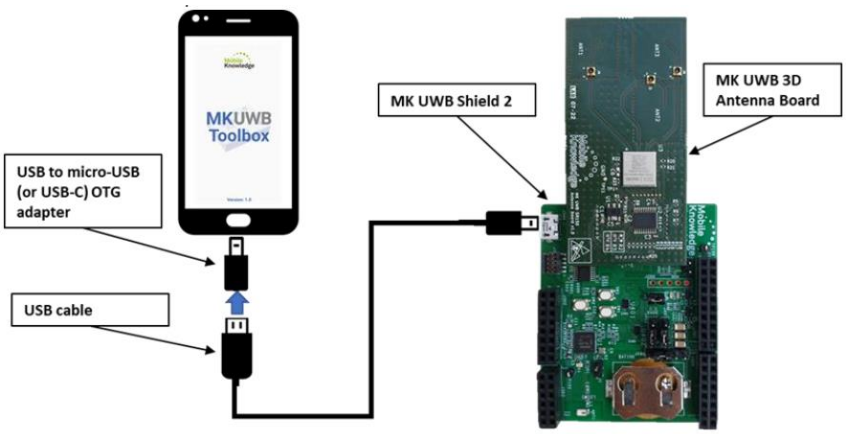


Abbildung 30: Verbindung MK UWB Shield 2 zu Mobiltelefon, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Um die Demos mithilfe der MK UWB PC Shell auszuführen, muss der PC mit einem USB-Kabel mit dem MK UWB Shield 2 verbunden werden (siehe Abbildung 31). Wenn das MK UWB Shield 2 ordnungsgemäß vom PC erkannt wird, erscheint im Geräte-Manager ein neuer COM-Port. Die Nummer dieses Ports ist für die Verwendung mit der MK UWB PC Shell erforderlich.

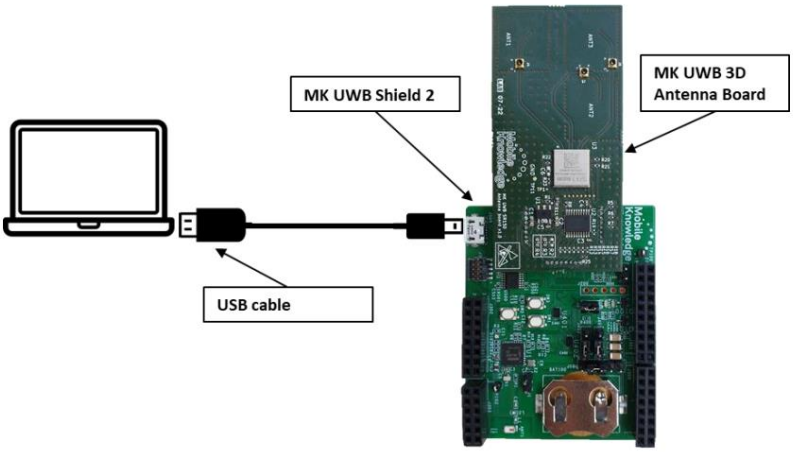


Abbildung 31: Verbindung MK UWB Shield 2 zu PC, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Wichtig ist, dass die Jumper des MK UWB Shield 2 gemäß Tabelle 2 konfiguriert sind.

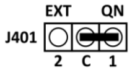
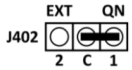
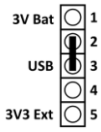
Position	Jumper	Beschreibung	Konfiguration
1	J2	QN9090 ein	
2	J401	Das UWB-Modul wird vom QN9090 aktiviert	
3	J402	Das UWB-Modul kommuniziert mit dem QN9090	
4	J100	Versorgung des Boards mit Strom über USB	

Tabelle 2 MK UWB Shield 2 Jumper Konfiguration, Quelle: Eigene Darstellung.

Die Jumper sind wie in Abbildung 32 dargestellt angeordnet:

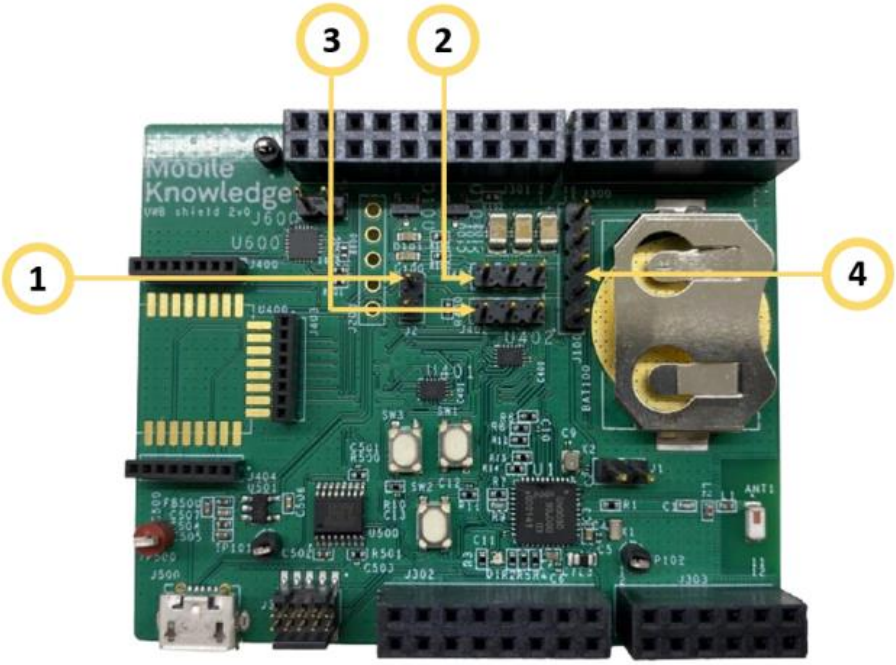


Abbildung 32: MK UWB Shield 2 Jumpers, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

6.2.2 Anwendungsfall Abstandswarnung

6.2.2.1 Allgemein

Mehrere Anwendungsfälle und Anwendungen basieren auf Entfernungsmessungen, um spezifische Aktionen auszulösen, beispielsweise um Warnungen anzuzeigen oder Benachrichtigungen an Benutzer zu senden. In Industrieumgebungen kann es beispielsweise erforderlich sein, dass ein gewisser Abstand zu gefährlichen Maschinen eingehalten oder bestimmte Bereiche gemieden werden. Oder bei der Zugangskontrolle könnte eine Tür so programmiert werden, dass sie je nach Nähe des (un)berechtigten Benutzers entriegelt oder verriegelt wird.

Aufgrund seiner präzisen Lokalisierungsfähigkeiten im Zentimeterbereich stellt UWB eine optimale Lösung dar, um Anwendungsfälle zu ermöglichen, die auf einer präzisen Näherungserkennung mit einem oder mehreren Entfernungsschwellenwerten basieren.

Abhängig von der Echtzeitposition der Tags vom Empfänger, die durch die UWB-Entfernung ermittelt wird, leuchtet eine LED in einer anderen Farbe in den Tags (zu sehen in Abb. 33):

Ist der Abstand des Tags vom Empfänger kleiner als 1 Meter, leuchtet die rote LED. Eine gelbe LED leuchtet am Tag, wenn der Abstand des Tags vom Empfänger größer als 1 Meter, aber kleiner als 2 Meter ist. Ist der Abstand des Tags vom Empfänger größer als 2 Meter, leuchtet eine grüne LED.

Es können maximal 5 Tags gleichzeitig getrackt werden. Werden mehr als 5 Tags erkannt, werden nur die ersten fünf für eine UWB-Anwendung berücksichtigt.

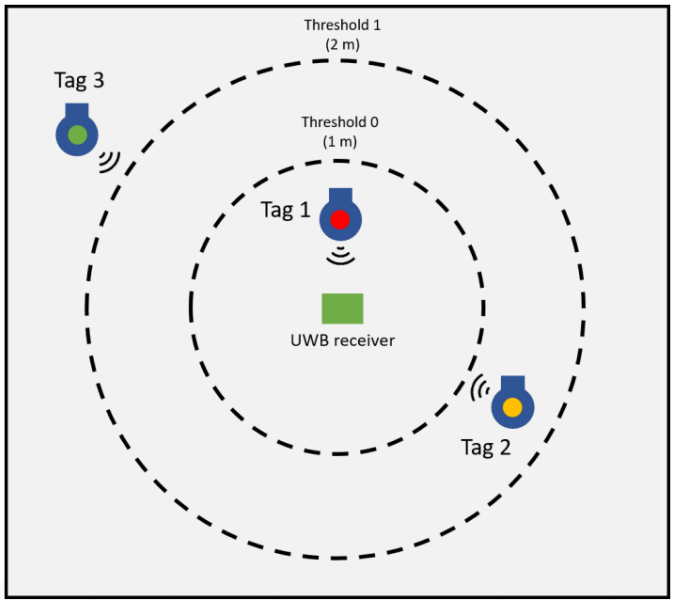


Abbildung 33: Abstandswarnung Beispiel, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

In einem realen Szenario kann der Anwendungsfall „Abstandswarnung“ angepasst werden, um komplexere Aktionen auf der Grundlage der vorhandenen Regeln auszulösen und die Anzahl der Schwellenwerte kann basierend auf den Anforderungen des Zielanwendungsfalls erhöht bzw. verringert werden.

6.2.2.2 Android Applikation

Um den Anwendungsfall für die Abstandswarnung auf dem Android-Mobiltelefon mithilfe der MK UWB Toolbox-Anwendung auszuführen, muss das Smartphone gemäß Abschnitt 6.2.1 mit dem MK UWB Shield 2-Board verbunden und die MK UWB Toolbox-Anwendung geöffnet sein. Danach muss im Auswahlbildschirm die Schaltfläche „Distance Alert“ ausgewählt werden.

Nach einigen Sekunden sollten die Tags auf dem Bildschirm zu sehen sein, wie in Abbildung 34 gezeigt. Werden die Tags vor dem MK UWB Shield 2-Board bewegt, kann man erkennen, wie in der Anwendung die Tags „von einer Schwelle zur andern wandern“ und gleichzeitig die LEDs auf den Tags, in Abhängigkeit der Entfernung, die Farbe ändern.

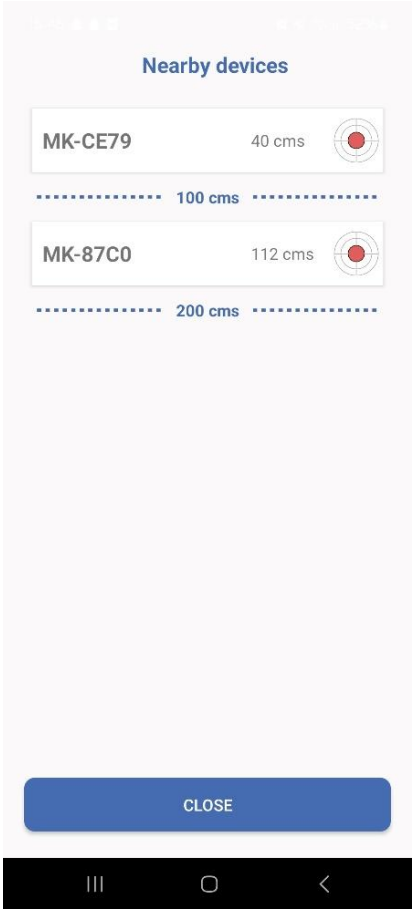


Abbildung 34: Tags werden basierend auf Schwellenwerten erkannt und in ihrem jeweiligen Bereich platziert, Quelle: Eigene Darstellung.

6.2.2.3 PC Shell Applikation

Um die Funktionalität für die Abstandswarnung auf dem PC mittels der MK UWB PC Shell-Anwendung auszuführen, ist es erforderlich, dass der PC gemäß Abschnitt 6.2.1 mit dem MK UWB Shield 2-Board verbunden ist und die MK UWB PC Shell-Anwendung gestartet wurde.

In der PC Shell muss dann Folgendes gesendet werden, um die Anwendung zu starten: python.\mk_uwb_pc_shell.py <COM port_number>. In der Eingabeaufforderung wird anschließend folgender Befehl zum Starten der Abstandswarnung genutzt: start_application distance_alert. Wenn der Anwendungsfall erfolgreich gestartet wurde und einige Tags in der Nähe sind, sollten die von den Tags gesendeten UWB-Bereichsbenachrichtigungen am Bildschirm erscheinen, wie in Abbildung 35 dargestellt. Jede Zeile meldet den Zeitstempel, die MAC-Adresse des Geräts, seinen Bluetooth LE-Kurznamen, die Schwelle, in die der Tag fällt, und die gemessene Entfernung.

```
PS C:\Users\nicog\Desktop\UWB_OnlineTraining\MK UWB PC Shell v1.1.0> python .\mk_uwb_pc_shell.py COM5
MK UWB PC Shell - v1.1.0

List of COMMANDS

start_application            Start Tracker or Distance alert application
grant_access                 Grant access to the UWB Lock to a given MAC address
exit                         Exit PC Shell
help                         PC Shell commands description

Command usage: cmd [arg]
  start_application tracker
  start_application distance_alert
  help
  exit
> start_application distance_alert
Sending: 0x1 0x0 0x2 0x10 0x3

Starting distance_alert application...

Distance Alert application has been started.

Looking for nearby devices... Please, press CTRL + C to stop distance alert application.

[2023-05-04 14:13:09] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 26 cm
[2023-05-04 14:13:09] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 27 cm
[2023-05-04 14:13:10] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 23 cm
[2023-05-04 14:13:10] 00:60:37:EE:CE:79 MK-CE79   AREA [0 m, 1 m] 32 cm
```

Abbildung 35: Empfangen von Entfernungsalarmbenachrichtigungen in der MK UWB PC Shell-Anwendung, Quelle: Eigene Darstellung.

6.2.3 Anwendungsfall Tracker

6.2.3.1 Allgemein

Die Kenntnis der genauen Echtzeitposition eines Objektes relativ zu einem Ortungsgerät, einschließlich ihrer räumlichen 2D- oder 3D-Koordinaten werden in verschiedensten Anwendungen benötigt. Beispielsweise kann das Ortungsgerät verwendet werden, um einen Benutzer zu Objekten zu führen, wie etwa Produkten in einem Lagerregal. Eine einfache Darstellung des Tracker-Anwendungsfallbeispiels ist in Abbildung 36 dargestellt.

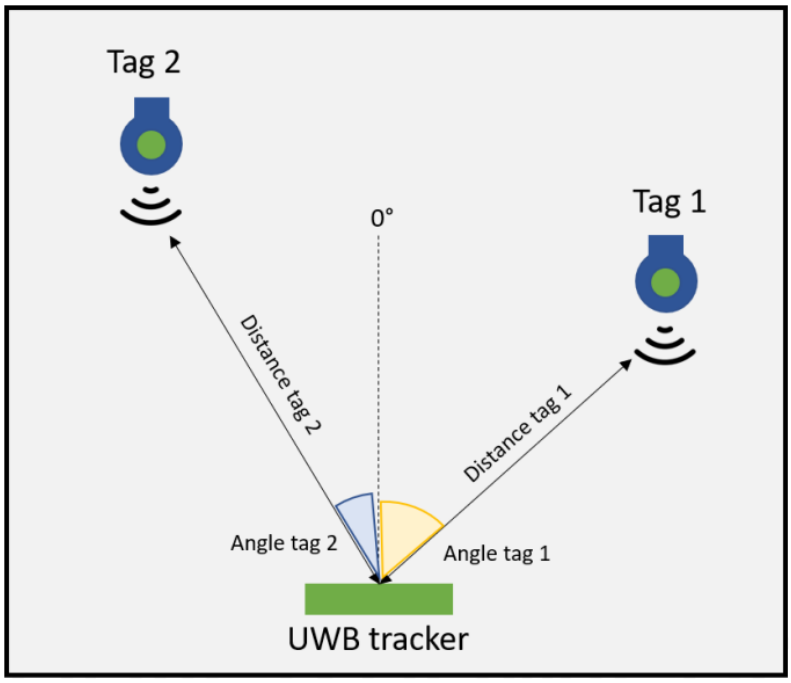


Abbildung 36: Tracker Beispiel, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]

Obwohl die für diese Arbeit genutzte Software nur die 2D-Positionierung ermöglicht, unterstützt die MK UWB 3D-Antennenplatine auch die 3D-Positionierung. Der AoA kann nur berechnet werden, wenn der Tracker über mindestens zwei Antennen für die 2D-Positionierung oder drei Antennen für die 3D-Positionierung verfügt.

6.2.3.2 Android Applikation

Um die Funktion zur Abstandswarnung auf einem Android-Mobiltelefon durch die MK UWB Toolbox-Anwendung zu nutzen, ist es notwendig, dass das Smartphone gemäß Abschnitt 6.2.1 mit dem MK UWB Shield 2-Board verbunden ist und die MK UWB Toolbox-Anwendung gestartet wurde. Anschließend sollte im Auswahlbildschirm die Option "Tracker" ausgewählt werden.

Am neuen Bildschirm erscheint ein +-Icon, mit dem nahegelegenen Tags entdeckt werden können. Es sollte beim Anklicken eine Liste mit Tags erscheinen, die verfolgt werden können. Die gewünschten Tags sind auszuwählen und werden mit der Schaltfläche „Tags hinzufügen“ getracket. Es können maximal 5 Tags ausgewählt werden. Nach einigen Sekunden erscheinen die Tags als Punkte im Radar wie in Abbildung 37 zu sehen ist.

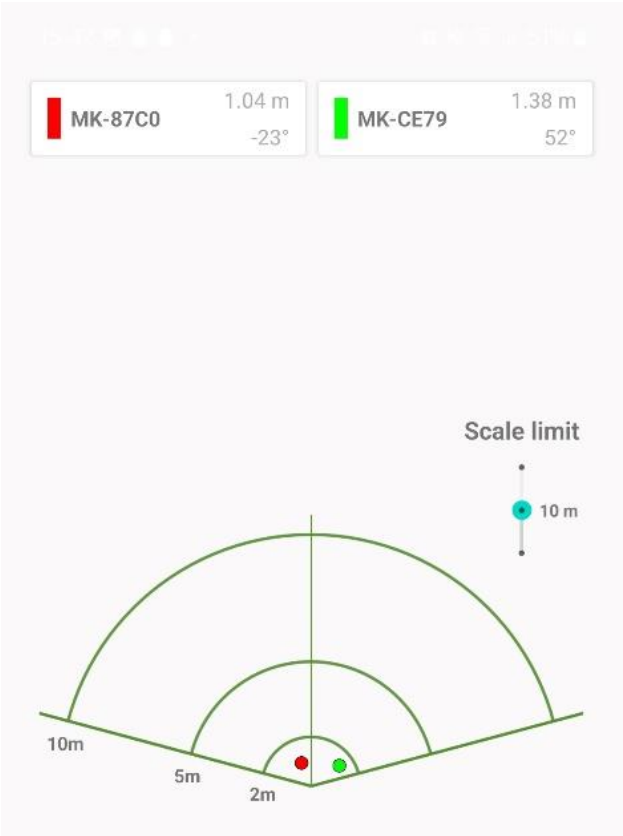


Abbildung 37: Tags in Radaransicht, Quelle: Eigene Darstellung.

Wird die Position der Tags vor dem MK UWB Shield 2-Board verändert, bewegen sich die Punkte basierend auf ihrer Entfernung und Winkelposition im Radar. Oben auf dem Bildschirm wird die Liste der verfolgten Tags sowie die gemessenen Entfernungen und Winkel angezeigt.

Die Genauigkeit der AoA-Messung kann durch Umgebungsbedingungen und die relative Position und Ausrichtung der verfolgten Tags und Antennen des MK UWB Shield 2 erheblich beeinträchtigt werden.

6.2.3.3 PC Shell Applikation

Um den Anwendungsfall für die Abstandswarnung auf dem PC mithilfe der MK UWB PC Shell-Anwendung auszuführen, muss der PC gemäß Abschnitt 6.2.1 mit dem MK UWB Shield 2-Board verbunden sein und die MK UWB PC Shell-Anwendung gestartet sein.

In der PC Shell muss dann folgendes gesendet werden: python.\mk_uwb_pc_shell.py <COM port_number> um die Anwendung zu starten. In der Eingabeaufforderung kann folgender Befehl zum Starten der Abstandswarnung gesendet werden: start_application tracker.

Das MK UWB Shield 2 beginnt über Bluetooth LE mit der Suche nach Geräten in der Nähe. Erkannte Tags werden mit ihrer MAC-Adresse und dem Bluetooth LE-Kurznamen in einer Liste aufgeführt, wie im oberen Bereich von Abbildung 38 dargestellt. Die Suche bricht nach einem gewissen Zeitlimit automatisch ab oder kann mit STRG-C beendet werden.

```
PS C:\Users\nicog\Desktop\UWB_OnlineTraining\MK UWB PC Shell v1.1.0> python .\mk_uwb_pc_shell.py COM6
MK UWB PC Shell - v1.1.0

List of COMMANDS

start_application             Start Tracker or Distance alert application
grant_access                  Grant access to the UWB Lock to a given MAC address
exit                          Exit PC Shell
help                          PC Shell commands description

Command usage: cmd [arg]
  start_application tracker
  start_application distance_alert
  help
  exit
> start_application tracker
Sending: 0x1 0x0 0x2 0x10 0x2

Starting tracker application...

Tracking application has been started. Scanned devices will appear in the list below.

Looking for devices to track...

Sending: 0x2 0x0 0x1 0x20
Sending: 0x2 0x0 0x1 0x21
00:60:37:39:87:C0 MK-87C0

Total: 1 device(s) found.
Use track <mac0> [mac1] [mac2] [mac3] [mac4] to track device(s) or press CTRL-C to stop the tracker application.
> track 00:60:37:39:87:C0 MK-87C0
ERROR: device with MAC MK-87C0 has not been scanned.
Tracking devices... Please press CTRL + C to stop tracking.

Sending: 0x2 0x0 0x7 0x22 0x0 0x60 0x37 0x39 0x87 0xc0
Start tracking 00:60:37:39:87:C0...
[2024-05-05 15:49:26] 00:60:37:39:87:C0 MK-87C0     53 cm  17°
[2024-05-05 15:49:27] 00:60:37:39:87:C0 MK-87C0     54 cm  23°
[2024-05-05 15:49:27] 00:60:37:39:87:C0 MK-87C0     51 cm  21°
[2024-05-05 15:49:28] 00:60:37:39:87:C0 MK-87C0     56 cm  15°
```

Abbildung 38: Tracker Anwendung in der MK UWB PC Shell, Quelle: Eigene Darstellung.

Um mit der Verfolgung eines oder mehrerer Tags zu beginnen, muss der Track-Befehl gefolgt von den MAC-Adressen der jeweiligen Tags, gesendet werden, wie in Abbildung 38 dargestellt. Es können bis zu 5 Geräte gleichzeitig verfolgt werden.

Wenn der Anwendungsfall erfolgreich gestartet wurde und einige Tags in der Nähe sind, sollten die von den Tags gesendeten UWB-Bereichsbenachrichtigungen am Bildschirm erscheinen, wie in Abbildung 38 dargestellt. Jede Zeile meldet den Zeitstempel, die MAC-Adresse des Geräts, seinen Bluetooth LE-Kurznamen, die gemessene Entfernung und die AoA.

Wichtig ist, dass MACs im richtigen Format mit Doppelpunkten eingefügt und durch ein Leerzeichen getrennt werden. Eine CSV-Datei mit Tracker-Bereichsbenachrichtigungen wird in <PC_Shell_path>/logs/tracker generiert. Für jede Trackersitzung wird eine neue Datei generiert.

6.3 Umbau des Modellautos

Abbildung 39: Originales Modellauto, Quelle: Eigene Darstellung.

Für die Verwendung des Modellautos, dessen Originalmodell in Abbildung 39 zu sehen ist, muss dieses zuerst umgebaut werden. Als erster Schritt wird die Gehäuseabdeckung entfernt, um die darunter liegenden Komponenten freizulegen. Zu den Komponenten zählen ein Motor, ein Lenkservo, ein Radioempfänger und ein ferngesteuerter elektronischer Drehzahlregler (Electronic Speed Controller, ESC). Zunächst müssen der originale ESC und der Radioempfänger entfernt werden. Dazu werden die Kabel, die den ESC mit dem Motor und dem Servo verbinden, abgesteckt und der ESC entfernt. Der ESC ist mit doppelseitigem Schaumstoffklebeband befestigt, daher muss dieser vorsichtig vom Rahmen gelöst werden.

Als nächstes werden die Heckflügel entfernt. Der Heckflügel muss aufbewahrt werde, da er später wieder benötigt wird. Der ursprüngliche Lenkservo, der werkseitig am Auto installiert ist, kann nicht mit der Fahrzeug-Management-Unit (FMU) verbunden werden. Er muss gegen einen ausgetauscht werden, der das traditionelle 3-Draht-RC-PWM (Radio-Controlled Pulse Width Modulation) verwendet. Um den Servo von der Basis des Autos zu entfernen, gibt es zwei Schrauben auf der Unterseite der Basis. Diese Schrauben müssen entfernt werden, damit der Servo gelöst werden kann.

Nachdem der Servo entfernt wurde, kann der Lenkarm des Servos abgenommen werden. Der Lenkarm wird am neuen Servo montiert. Dabei muss darauf geachtet werden, dass der Arm gerade nach oben zeigt und die Vorderräder geradeaus ausgerichtet sind. Dies ist wichtig, um eine präzise Steuerung des Modellfahrzeugs zu gewährleisten.

Im Anschluss kann der neue ESC anstelle des alten ESC eingebaut werden. Dabei ist darauf zu achten, genügend Platz zu lassen, um die Drähte leichter verstauen zu können. Die Drähte des ESC werden mit dem Motor verbunden. Sollte es passieren, dass die Drähte falsch zusammengesteckt werden, schadet dies dem Motor nicht; das Auto fährt nur in die entgegengesetzte Richtung. Die Drähte sind sauber zu

verstauen, und der ESC-Netzschalter sollte an einer zugänglichen Stelle angebracht werden. Der Netzschalter dient dazu, das Modellauto zu aktivieren, ohne dass es zufällig wegfährt.

Folgend wird der ESC am Power Distribution Board (PDB) angeschlossen. Das PDB verteilt die Energie gleichmäßig an alle elektronischen Komponenten des Fahrzeugs und stellt sicher, dass diese korrekt funktionieren. Zuletzt werden die Heckflügel wieder montiert.

Nach diesen Umbaumaßnahmen ist das Modellauto bereit für die Integration der UWB-Technologie. Die präzise Steuerung und die Anpassungen ermöglichen es dem Fahrzeug, den UWB-Tag zu verfolgen und die Follow-Me Funktion zu erfüllen. Die Kalibrierung und Programmierung der UWB-Komponenten sowie die Feinabstimmung der Fahrzeugsteuerung sind die nächsten Schritte, um das Projekt erfolgreich abzuschließen. Hierbei wird getestet, wie genau das Fahrzeug auf die Steuerbefehle reagiert und wie stabil die Verbindung zwischen den UWB-Geräten und der Fahrzeugsteuerung ist. Dies bildet die Grundlage für ein voll funktionsfähiges, autonomes Follow-Me Modellfahrzeug.

7 PROGRAMMIERUNG

Um die verschiedenen Programmierschritte zu vereinfachen und die Fehlersuche zu erleichtern, werden die verschiedenen Abschnitte des Projekts in separaten Unterprojekten programmiert und getestet. Dieser modulare Ansatz ermöglicht es, jedes Teilmodul unabhängig zu entwickeln und sicherzustellen, dass es korrekt funktioniert, bevor es in das Gesamtsystem integriert wird. Jedes Unterprojekt konzentrierte sich auf eine spezifische Funktionalität, wie die Ansteuerung der Servomotoren, die Kommunikation mit dem MK UWB Shield 2 oder die Implementierung der Steuerlogik.

Durch die Aufteilung des Projekts in kleinere, überschaubare Einheiten wird die Entwicklung effizienter gestaltet. Fehler können schneller lokalisiert und behoben werden, da die Komplexität in jedem einzelnen Unterprojekt geringer ist. Dies reduziert auch die Wahrscheinlichkeit, dass Änderungen an einem Teil des Codes unbeabsichtigte Auswirkungen auf andere Teile des Projekts haben. Jeder Abschnitt wird gründlich getestet, um sicherzustellen, dass er die gewünschten Anforderungen erfüllt und zuverlässig funktioniert.

7.1 Auswahl Kommunikation und Mikrocontroller

Die Ansteuerung des MK UWB Shield 2 kann mittels eines Arduino oder eines Raspberry Pi erfolgen. Aufgrund der Verfügbarkeit wird der Arduino gewählt. Der Arduino bietet eine einfache und flexible Entwicklungsumgebung, die besonders für schnelle Prototypen und Projekte im Bereich der Mikrocontroller-Programmierung geeignet ist. Die umfangreiche Bibliotheksunterstützung und die große Community machen den Arduino zu einer idealen Wahl für dieses Projekt.

Bei der Verwendung des Arduino zur Ansteuerung des MK UWB Shield 2 wird die serielle Kommunikation genutzt, um Befehle zu senden und Daten zu empfangen. Die serielle Schnittstelle des Arduino ermöglicht eine zuverlässige und effiziente Datenübertragung, die für die Konfiguration und den Betrieb des UWB Shields erforderlich ist.

Die Wahl des Arduino bietet zudem den Vorteil, dass zahlreiche zusätzliche Module und Sensoren leicht integriert werden können. Dies ermöglicht eine Erweiterung des Projekts um zusätzliche Funktionen, wie beispielsweise die Integration von GPS-Modulen, anderen Sensoren oder drahtlosen Kommunikationsmodulen.

Darüber hinaus ist die Programmierung des Arduino durch die benutzerfreundliche Arduino-IDE besonders zugänglich. Die IDE unterstützt die schnelle Entwicklung und das Testen von Code, wodurch die Implementierung und die Fehlersuche vereinfacht werden. Die Verfügbarkeit zahlreicher Beispiele und Tutorials beschleunigt den Lernprozess und die Entwicklung komplexer Anwendungen.

Des Weiteren ist die Ansteuerung und Einbindung der Servomotoren des Modellfahrzeugs einfacher. Der Arduino bietet eine intuitive und benutzerfreundliche Schnittstelle zur Steuerung von Servomotoren, die für Lenk- und Antriebsfunktionen in Modellfahrzeugen unerlässlich sind. Mit den vorhandenen Servo-Bibliotheken können Servomotoren präzise gesteuert werden, indem einfach der gewünschte Winkel oder die gewünschte Position angegeben wird.

Die Implementierung der Servo-Motorsteuerung auf dem Arduino ist besonders unkompliziert, da die Servo-Bibliothek (Servo.h) eine hohe Abstraktionsebene bietet. Dies ermöglicht es Entwicklern, Servomotoren mit minimalem Codeaufwand zu steuern. Der Arduino übernimmt die Erzeugung der notwendigen PWM-Signale (Pulsweitenmodulationssignale), um die Servos entsprechend den Eingaben zu positionieren. Dies reduziert die Komplexität der Hardwaresteuerung erheblich und erleichtert die Integration in das Gesamtsystem des Modellfahrzeugs.

Ein weiterer Vorteil der Verwendung des Arduino zur Steuerung der Servomotoren ist die Möglichkeit zur schnellen Anpassung und Feinabstimmung der Steuerungsparameter. Über die serielle Schnittstelle können Parameter in Echtzeit angepasst und optimiert werden, um die gewünschten Bewegungsabläufe und Reaktionen des Modellfahrzeugs zu erzielen. Dies ist besonders nützlich bei der Entwicklung und Erprobung von Fahralgorithmen und autonomen Fahrfunktionen.

Die Ansteuerung des MK UWB Shield 2 kann mittels SPI oder UART erfolgen. Bei der Verwendung von UART erfolgt die Kommunikation mit dem Mikrocontroller, wobei das MK UWB Shield die Signalverarbeitung und -weiterleitung selbst erledigt. Dies bedeutet, dass der Großteil der Datenverarbeitung und Steuerung direkt vom Shield übernommen wird, wodurch der Entwicklungsaufwand reduziert und die Implementierung vereinfacht wird. Durch die Verwendung von UART kann der Arduino über eine serielle Schnittstelle unkompliziert Befehle senden und Daten empfangen, ohne sich um die Details der UWB-Signalverarbeitung kümmern zu müssen.

Im Gegensatz dazu ermöglicht die Verwendung von SPI eine direkte Kommunikation mit dem UWB-Sensor. Diese Methode bietet möglicherweise eine höhere Datenrate und mehr Kontrolle über die UWB-Funktionalität, erfordert jedoch eine komplexere Implementierung und ein tieferes Verständnis der UWB-Sensorprotokolle. Die direkte Ansteuerung des Sensors über SPI kann zudem anspruchsvollere Hardware- und Softwarekonfigurationen erfordern, um eine zuverlässige und effiziente Kommunikation sicherzustellen.

Aufgrund der einfacheren Kommunikation und der Verfügbarkeit zahlreicher Beispiele und Bibliotheken wird die UART-Schnittstelle gewählt. UART ist weit verbreitet und gut dokumentiert, wodurch der Entwicklungsprozess erheblich erleichtert wird. Die serielle Kommunikation über UART erfordert lediglich zwei Pins (TX und RX) und kann problemlos in die bestehende Arduino-Umgebung integriert werden. Darüber hinaus bietet die UART-Schnittstelle eine robuste und zuverlässige Datenübertragung, die für viele Anwendungen ausreichend ist.

Aufgrund dieser Kriterien wird für das Projekt der Arduino MEGA ausgewählt.

7.2 Programmierung der Steuerung des Modellfahrzeuges

7.2.1 Bibliotheken und globale Variablen

```
1   //TEST PID ***GRÜNWALD***
2
3   #include <PIDLoop.h>
4   #include <Servo.h>
5
6   Servo servoMain; // Define our Servo
7   Servo motorMain; // Define our Servo
8
9   int int_Error[8]={0,1,4,8,0,-1,-4,-8};
10  int int_Command;
11  int int_PID_ERROR_left;
12  int int_PID_ERROR_right;
13  double angle = 90;
14  double speed =1500;
15
16  PIDLoop PID_CONTROLLER(5500, 0, 0, false);
```

Abbildung 40: Code für globalen Variablen und Bibliotheken, Quelle: Eigene Darstellung.

In Abbildung 40 ist der Code für die Deklarierung der globalen Variablen sowie die Einbindung der notwendigen Bibliotheken abgebildet. In Zeile 3 und 4 werden die benötigten Bibliotheken für das Projekt eingebunden.

Die Bibliothek Servo.h ist eine C/C++-Bibliothek, die verwendet wird, um Servos anzusteuern. Servos sind Motoren, die über ein integriertes Steuersystem verfügen, das es ermöglicht, den Motor auf eine bestimmte Position zu steuern. Diese Position kann präzise gesteuert werden und bleibt stabil, solange die Stromversorgung für den Motor aktiv ist. Die Servo.h-Bibliothek bietet Funktionen zum einfachen Ansteuern von Servomotoren mit einem Arduino oder einem kompatiblen Mikrocontroller. Sie bietet Funktionen zum Einstellen der Position des Servomotors, zur Festlegung der Bewegungsgeschwindigkeit und zur Festlegung von Grenzwerten für die Position.

Die Bibliothek PIDLoop.h ist eine C/C++-Bibliothek, welche die Implementierung eines PID-Reglers (Proportional-Integral-Differential-Regler) ermöglicht. Ein PID-Regler ist ein Regelalgorithmus, der zur Regelung von Prozessen verwendet wird, um ein gewünschtes Ziel zu erreichen. Er besteht aus drei Hauptteilen:

- Proportional (P)

Diese Komponente reagiert proportional zur aktuellen Fehlergröße. Sie bestimmt, wie stark der Regler auf die aktuelle Abweichung vom Sollwert reagieren soll.

- Integral (I)

Diese Komponente betrachtet die Summe der bisherigen Fehler über die Zeit. Sie korrigiert langsame, aber stetige Fehler, die vom Proportionalanteil nicht behoben werden können, und hilft, das System auf den Sollwert zu bringen.

- Derivative (D)

Diese Komponente betrachtet die Rate der Änderung des Fehlers über die Zeit. Sie wirkt dämpfend und hilft, die Geschwindigkeit der Annäherung an den Sollwert zu steuern, indem sie die Reaktion des Reglers auf schnelle Änderungen des Fehlers begrenzt.

Die PIDLoop.h-Bibliothek implementiert diese drei Komponenten des PID-Reglers und bietet Funktionen zum Einstellen der Reglerparameter sowie zum Aktualisieren des Reglers mit neuen Messwerten und zur Berechnung der Ausgabewerte des Reglers.

In den Zeilen 6 und 7 der Abbildung 40 werden zwei Servo-Objekte definiert: servoMain für die Lenkung und motorMain für die Geschwindigkeitssteuerung. In den Zeilen 9-14 werden verschiedene globale Variablen deklariert:

- int_Error

Array von Fehlerwerten, die für die PID-Regelung verwendet werden.

- int_Command

Befehlswert, der von der PID-Regelung berechnet wird.

- int_PID_ERROR_left / int_PID_ERROR_right

Fehlerwerte für die linke und rechte Seite des Fahrzeugs.

- angle

Winkel für die Lenkung des Servos, initialisiert auf 90 Grad.

- speed

Geschwindigkeit des Servomotors, initialisiert auf 1500.

In Zeile 16 wird ein PIDLoop-Objekt erstellt und mit den Reglerparametern (Proportional, Integral, Derivative) initialisiert. Der vierte Parameter, false, gibt an, ob der Regler invertiert ist.

7.2.2 Setup()-Funktion

```
17
18  void setup()
19  {
20    //start serial communication
21    Serial.begin(115200);
22
23    Serial.println("Starting FollowMe...");
24    Serial.println("");
25    servoMain.attach(13); // servo for steering on digital pin 13
26    motorMain.attach(12); // servo for speed on digital pin 12
27
28  }
29
```

Abbildung 41: Setup () -Funktion, Quelle: Eigene Darstellung.

Die Setup()-Funktion, ersichtlich in Abbildung 41, wird nur einmal beim Start des Programms ausgeführt. Ihre Hauptaufgabe besteht darin, die Initialisierungsvorgänge durchzuführen, die für das korrekte Funktionieren des Programms erforderlich sind.

Mit der Zeile 21 wird die serielle Kommunikation mit einer Baudrate von 115200 Baud gestartet. Die serielle Kommunikation ermöglicht es, Daten zwischen dem Arduino und einem Computer oder einem anderen Gerät auszutauschen. Hier wird eine Verbindung mit den Serial Monitor der Arduino IDE aufgebaut, zur Darstellung der Daten bzw. des Fortschrittes des Programmes wie in Zeile 23 und 24 ersichtlich, sowie für Debugging-Zwecke.

In den Zeilen 25 und 26 werden die Servomotoren an ihre entsprechenden Pins des Arduino-Boards angeschlossen. Hier wird angenommen, dass der Servo für die Lenkung an Pin 13 und der Servo für die Geschwindigkeitssteuerung an Pin 12 angeschlossen ist. Die attach()-Funktion weist dem Servo-Objekt die Pin-Nummer zu, an die der Servo angeschlossen ist. Nachdem die Servos an ihre Pins angeschlossen sind, können sie gesteuert werden, indem Werte an sie gesendet werden, um ihre Position oder Geschwindigkeit zu ändern.

7.2.3 Loop () -Funktion

```
30  void loop()
31  {
32
33    for (int i =0;i<9;i++)
34    {
35      // do PID calculations
36      Serial.print("Fehler ");
37      Serial.println(int_Error[i]);
38      Serial.println();
39      PID_CONTROLLER.update(int_Error[i]);
40      int_PID_ERROR_right=PID_CONRTOLLER.m_command;
41      int_PID_ERROR_left=-PID_CONRTOLLER.m_command;
```

Abbildung 42: Loop () -Funktion, Quelle: Eigene Darstellung.

Die loop () -Funktion, in Abbildung 42 dargestellt, ist eine Endlosschleife, die immer wieder ausgeführt wird. Innerhalb dieser Schleife wird eine weitere Schleife durchlaufen, die von 0 bis 8 (9 Durchläufe) zählt. Der PID-Regler wird mit verschiedenen Werten mit unterschiedlichen Abweichungen getestet. Für jeden Durchlauf der äußeren Schleife wird der PID-Regler mit einem anderen Wert aktualisiert, wie in Zeile 39 ersichtlich. int_Error[i] ist der Fehlerwert für den aktuellen Durchlauf, der aus dem Array int_Error genommen wird. Nachdem der PID-Regler aktualisiert wurde, werden die Befehlswerte für die linke und rechte Seite des Fahrzeugs berechnet. Diese Befehlswerte sind die Ausgabe des PID-Reglers und werden verwendet, um die Lenkung zu kontrollieren.

```
44      //Steering
45
46      if(int_PID_ERROR_left>1)
47      {
48        angle=90-(int_PID_ERROR_left*0.25);
49        Serial.print("Winkel links: "+ String(angle) +" von Fehlerwert" + String(int_Error[i]));
50        Serial.println();
51        servoMain.write(angle);
52      }
53
54      if(int_PID_ERROR_right>1)
55      {
56        angle=90+(int_PID_ERROR_right*0.1);
57        Serial.print("Winkel rechts: "+ String(angle) +" von Fehlerwert" + String(int_Error[i]));
58        Serial.println();
59        servoMain.write(angle);
60      }
61
62      if(int_PID_ERROR_right == 0 && int_PID_ERROR_right == 0)
63      {
64        angle=90;
65        Serial.print("Winkel gerade: "+ String(angle) +" von Fehlerwert" + String(int_Error[i]));
66        Serial.println();
67        servoMain.write(angle);
68      }
```

Abbildung 43: Loop () -Funktion für die Lenkung, Quelle: Eigene Darstellung.

Mit drei if-Bedingungen wird, wie in Abbildung 43 dargestellt, aufgrund der Variablen int_PID_ERROR_left und int_PID_ERROR_right entschieden, ob nach links oder rechts gelenkt wird. Ist die Variable int_PID_ERROR_left größer 1, so wird die erste if ausgewählt und nach links gelenkt. Ist die Variable int_PID_ERROR_right größer 1, so wird die zweite if ausgewählt und nach rechts gelenkt. Sind beide Variablen, also die Abweichungen der Winkel kleiner 1, wird die dritte if verwendet und das Modellfahrzeug fährt gerade aus.

In den if wird aufgrund der Abweichung der Variable der benötigte Winkel ausgerechnet. Um zu große Sprünge im Winkel zu verhindern, werden diese mit einem Faktor multipliziert. Nachdem der Lenkwinkel berechnet wurde, wird dieser Winkel an den Servo gesendet, um die Lenkung des Fahrzeugs entsprechend anzupassen.

7.3 Übertragung der Messdaten

7.3.1 REV01 des Programmes zur Übertragung

7.3.1.1 Globale Variablen

```
1
2   //Global Variablen
3   String str_recievedDATA; // for incoming serial data
4   String str_sendDATA; // for sending serial data
5
6   char buffer[32];
7   int bytesRead=0;
8
```

Abbildung 44: Globale Variablen, Quelle: Eigene Darstellung.

In den Zeilen 3 und 4 der Abbildung 44, werden die zwei String-Variablen str_recievedDATA und str_sendDATA deklariert. Diese werden für die Kommunikation mit dem MK UWB Shield 2 benötigt, um die gesendeten Daten zu speichern und um Daten zu senden. Die Variable buffer ist ein Array mit der Größe 32 und dem Datentyp char, die als Puffer für die empfangenen Daten dient. Die Variable bytesRead, wird verwendet, um die Anzahl der empfangenen Bytes zu speichern.

7.3.1.2 Setup()-Funktion

```
9
10  void setup()
11  {
12    Serial.begin(115200)); //Initialize the Serial monitor for debugging
13    Serial1.begin(115200); //initialize serial1 at a 11520 baud rate for communication with UWB-Board
14
15
16    //Configuration for UWB-Board
17
18    Serial1.write(1); //sent in Bytes:0x01
19    Serial1.write(0);//sent in Bytes:0x00
20    Serial1.write(10);//sent in Bytes:0x02
21    Serial1.write(10000);//sent in Bytes:0x10
22    Serial1.write(1);//sent in Bytes:0x03
23    Serial.print("Configuration finished\n");
24
25  }
26
```

Abbildung 45: Setup () -Funktion, Quelle: Eigene Darstellung.

Abbildung 45 zeigt die Setup()-Funktion. In den Zeilen 12 und 13 werden zwei serielle Kommunikationen gestartet. Serial.begin(115200) startet die serielle Kommunikation mit dem Computer mit einer Baudrate von 115200. Diese serielle Schnittstelle wird hauptsächlich für Debugging-Zwecke verwendet. Sie ermöglicht es, den Fortschritt des Codes in Echtzeit zu überwachen, Fehlermeldungen zu analysieren und die Ausgabe verschiedener Variablen und Systemzustände darzustellen. Diese Informationen sind entscheidend, um den Entwicklungsprozess zu beschleunigen und sicherzustellen, dass das System wie erwartet funktioniert.

Serial1.begin(115200) startet die serielle Kommunikation mit einer Baudrate von 115200 für die Kommunikation mit dem MK UWB Shield 2. Diese Schnittstelle ist speziell für die Interaktion mit dem MK UWB Shield 2 vorgesehen, das zur präzisen Positionserfassung und -kommunikation verwendet wird. Um die Kommunikation zwischen dem MK UWB Shield 2 und dem Arduino MEGA zu ermöglichen, muss die serielle Übertragung über Serial1 richtig verdrahtet werden, wie in Abbildung 46 dargestellt.

Beim Arduino MEGA werden die Pins D19 (RX1) und D18 (TX1) für Serial1 verwendet. Um sie zur Kommunikation mit einem externen seriellen Gerät zu verwenden, muss der TX-Pin (Transmitter) des Arduino mit dem RX-Pin (Receiver) des externen Geräts und der RX-Pin des Arduino mit dem TX-Pin des externen Geräts verbunden werden. Diese Kreuzverbindung stellt sicher, dass das gesendete Signal des einen Geräts vom anderen empfangen wird und umgekehrt.

Beim Anschluss über ein USB-Kabel sind die Adernfarben zu beachten: Die grüne Ader stellt die TX-Verbindung dar und sollte daher mit Pin 19 (RX1) des Arduino verbunden werden. Die weiße Ader ist die RX-Verbindung und gehört entsprechend auf Pin 18 (TX1) des Arduino. Die rote Ader dient der Spannungsversorgung und sollte an den 3,3 V-Ausgang des Arduino angeschlossen werden, da das MK UWB Shield 2 mit 3,3 V betrieben wird. Die schwarze Ader ist für die Masse (GND) und stellt die gemeinsame Rückleitung für die elektrische Schaltung dar.

Korrekte Verdrahtung und Konfiguration sind essenziell, um eine zuverlässige und stabile Datenübertragung sicherzustellen. Falsche Verbindungen können nicht nur zu Kommunikationsfehlern führen, sondern auch die angeschlossenen Komponenten beschädigen. Daher ist es wichtig, die Verdrahtung sorgfältig nach den Spezifikationen vorzunehmen und zu überprüfen.

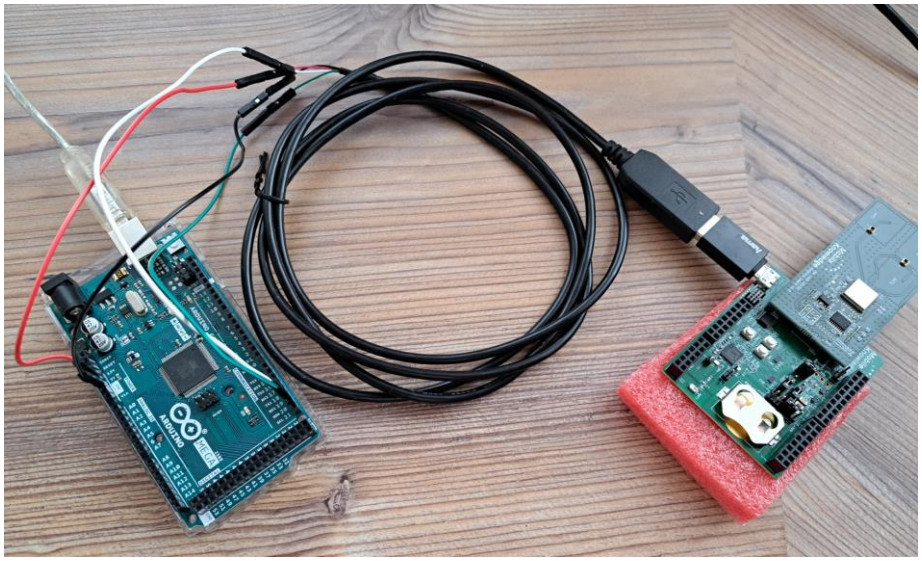


Abbildung 46: Verdrahtung des Arduino mit dem MK UWB Shield 2, Quelle: Eigene Darstellung.

In den Zeilen 18 bis 23, von Abbildung 45, werden über die serielle Verbindung Serial1 die Konfigurationsbytes an das MK UWB Shield 2 gesendet. Die Befehle des MU UWB Shield 2 sind als TLV abgespeichert und müssen daher als Bytes gesendet werden. Allerding sind die Befehle im hexadezimalen Format geschrieben und müssen für die Kommunikation in Bytes umgewandelt werden. Die TLV hat folgende Bedeutung:

Befehl: 0x01

Länge: 0x00 | 0x02 (zeigt an, dass die Nachrichtenlänge 2 Byte beträgt)

App-Start: 0x10

Entfernungsalarm: 0x03

Nach dem Senden der Konfiguration wird eine Nachricht auf dem seriellen Monitor ausgegeben, dass die Konfiguration abgeschlossen ist.

7.3.1.3 Loop()-Funktion

```
27  void loop()
28  {
29
30    while (Serial1.available() > 0)
31    {
32      Serial.print("Start recieving Data \n");
33      int bytesRead=Serial1.readBytes(buffer,32);
34      //char receivedChar = Serial1.read();
35      if (bytesRead>0)
36      {
37        Serial.print("I received: ");
38        Serial.print(bytesRead);
39        for (int i = 0; i < sizeof(buffer); i++)
40        {
41          Serial.write(buffer[i]);
42        }
43        for (int i = 0; i < sizeof(buffer); i++)
44        {
45          Serial.println(buffer[i]);
46        }
47        Serial.println();
48      }
```

Abbildung 47: Loop()-Funktion, Quelle: Eigene Darstellung.

In Abbildung 47 wird die Loop()-Funktion dargestellt. In Zeile 30 wird überprüft, ob Daten im seriellen Puffer von Serial1 verfügbar sind. Sind Daten verfügbar wird die Schleife ausgeführt. Zeile 32 gibt auf dem seriellen Monitor eine Nachricht aus, dass Daten empfangen werden. Danach werden bis zu 32 Bytes von Serial1 eingelesen und in der Variable buffer gespeichert. Die Anzahl der tatsächlich gelesenen Bytes wird in bytesRead gespeichert. Im nächsten Schritt wird mit bytesRead überprüft, ob Daten gelesen wurden. Als nächster Schritt wird die Anzahl der empfangenen Bytes an den seriellen Monitor ausgegeben. Mit der for-Schleife in Zeile 39 werden die empfangenen Bytes auf dem seriellen Monitor geschrieben. Mit der zweiten for-Schleife in Zeile 43 werden die empfangenen Bytes als Zeichen (ASCII-Werte) auf dem seriellen Monitor geschrieben. In Zeile 48 wird eine neue Zeile hinzugefügt, um die Ausgabe zu trennen und die Lesbarkeit zu verbessern.

7.3.1.4 Ergebnis

Wie in Abbildung 48 ersichtlich, wird eine Antwort von dem MK UWB Shield 2 mittels der seriellen Kommunikation erhalten. Die erhaltenen Daten werden beim seriellen Monitor jedoch als unleserliche Zeichen dargestellt und konnten im Programm nicht verwendet werden. Dies deutet darauf hin, dass die Daten möglicherweise in einem binären oder speziellen Format vorliegen, das für die direkte Anzeige im seriellen Monitor nicht geeignet ist. Um die Daten interpretieren und im Programm verwenden zu können, ist es notwendig, sie in ein lesbares Format zu konvertieren.

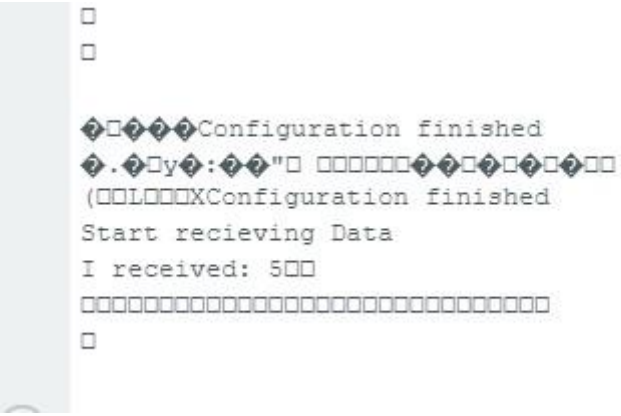


Abbildung 48: Darstellung der Antwort im seriellen Monitor, Quelle: Eigene Darstellung.

Ein möglicher Ansatz wäre, die empfangenen Bytes als ASCII-Zeichen oder Hexadezimalwerte darzustellen. Dies kann durch eine geeignete Umwandlung der empfangenen Daten im Programm erfolgen. Es könnte auch hilfreich sein, die Dokumentation des MK UWB Shield 2 zu überprüfen, um herauszufinden, in welchem Format die Daten gesendet werden. Oftmals bieten solche Geräte eine Spezifikation, die das Datenprotokoll beschreibt, einschließlich der Struktur und Bedeutung der einzelnen Datenfelder.

7.3.2 REV02 des Programmes zur Übertragung

7.3.2.1 Globale Variablen und Setup()-Funktion

```
1  //Global Variablen
2  String str_recievedDATA; // for incoming serial data
3  String str_sendDATA; // for sending serial data
4
5  char buffer[32];// Definiert einen Puffer mit einer Größe von 32 Bytes für Empfangenedaten
6  int bytesRead=0;// Zur Überprüfung ob Daten empfangen wurden
7  uint8_t data_rx[32]; // modify array size as needed
8  uint8_t start=0;
9
10 void setup()
11 {
12   Serial.begin(115200); //Initialize the Serial monitor for debugging
13   Serial1.begin(115200); //initialize serial1 at a 11520
14   //baud rate for communication with UWB-Board
15
16   //Configuration for UWB-Board
17
18   Serial1.write(1); //sent in Bytes:0x01
19   Serial1.write(0);//sent in Bytes:0x00
20   Serial1.write(10);//sent in Bytes:0x02
21   Serial1.write(10000);//sent in Bytes:0x10
22   Serial1.write(10);//sent in Bytes:0x02
23   Serial.print("Sending: 0x01 0x00 0x02 0x10 0x02\n");
24   Serial.print("Starting tracker Application...\n");
25 }
```

Abbildung 49: Globale Variablen und Setup()-Funktion, Quelle: Eigene Darstellung.

Die globalen Variablen sind ähnlich wie im Kapitel 7.3.1 konfiguriert. Zusätzlich ist die Variable data_rx hinzugefügt, ein uint8_t-Array mit einer Größe von 32 Bytes. Der Datentyp uint8_t steht für „unsigned 8-bit integer“. Es handelt sich um einen ganzzahligen Datentyp, der keinen Vorzeichen-Bit hat und daher nur positive Werte darstellen kann, die im Bereich von 0 bis 255 liegen. Dieser Datentyp wird häufig bei der seriellen Schnittstelle verwendet, da hier die Kontrolle über die Größe der Daten entscheidend ist. Die Übertragung mittels UART ist ebenfalls auf 8 Bit begrenzt, weshalb uint8_t die geeignete Wahl ist.

In der Setup()-Funktion werden, wie zuvor, zwei serielle Kommunikationsschnittstellen initialisiert. Die erste Schnittstelle wird für die Kommunikation mit dem seriellen Monitor verwendet, um Debugging-Informationen anzuzeigen und den Entwicklungsprozess zu erleichtern. Die zweite Schnittstelle wird für die Kommunikation mit dem MK UWB Shield 2 verwendet. Diese Schnittstelle ermöglicht den Austausch von Daten zwischen dem Mikrocontroller und dem UWB-Board.

Zusätzlich werden in der Setup()-Funktion Konfigurationsdaten im TLV-Format an das MK UWB Shield 2 gesendet. Diese Konfigurationsdaten dienen dazu, das UWB-Board richtig einzustellen und sicherzustellen, dass es ordnungsgemäß funktioniert. Das Senden von Konfigurationsdaten im TLV-Format ermöglicht eine strukturierte und flexible Kommunikation, bei der jeder Datensatz aus einem Typ, einer Länge und einem Wert besteht. Dies erleichtert die Interpretation der Daten und gewährleistet, dass beide Geräte korrekt miteinander kommunizieren.

7.3.2.2 Loop()-Funktion

```
33  void loop()
34  {
35    if(Serial1.available()>0)
36    {
37
38      int total_read=Serial1.readBytesUntil('\n',data_rx,32);
39      data_rx[total_read]='\0';
40
41      for (int i=0;i<total_read;i++)
42      {
43        Serial.print(data_rx[i],HEX); // Print value received as hex
44        Serial.print(' ');
45      }
46
47      Serial.print('\n');
48
49    }
```

Abbildung 50: Loop()-Funktion, Quelle: Eigene Darstellung.

In Abbildung 50 wird die Loop()-Funktion dargestellt. In Zeile 35 wird überprüft, ob Daten im seriellen Puffer von Serial1 verfügbar sind. Sind Daten verfügbar wird die Schleife ausgeführt. In Zeile 38 werden die Daten von der seriellen Schnittstelle Serial1 bis zum Zeilenumbruch-Zeichen ('\n') oder bis zu einer maximalen Anzahl von 32 Bytes eingelesen und im Array data_rx gespeichert. Die Anzahl der tatsächlich gelesenen Bytes wird in der Variable total_read gespeichert. Die Funktion readBytesUntil wird häufig verwendet, um Daten von der seriellen Schnittstelle zu lesen, insbesondere wenn die Daten durch ein bestimmtes Zeichen, wie etwa einen Zeilenumbruch, getrennt sind. Sie bietet eine einfache Möglichkeit, eine definierte Anzahl von Bytes oder bis zu einem Terminierungszeichen zu lesen und die Daten in einem Puffer zu speichern.

Die Zeile 39 setzt ein Nullzeichen ('\0') am Ende der gelesenen Daten im Array data_rx, um den empfangenen Daten-String ordnungsgemäß zu terminieren. Dies ist wichtig, um sicherzustellen, dass die Daten als gültiger C-String behandelt, werden können. Ein C-String ist eine Folge von Zeichen, die mit einem Nullzeichen endet. Dieses Nullzeichen signalisiert das Ende des Strings und ermöglicht es Funktionen der C-Standardbibliothek sowie anderer String-Verarbeitungsfunktionen, die Länge des Strings zu erkennen und korrekt mit dem String zu arbeiten. Durch die Null-Terminierung wird sichergestellt, dass Funktionen, die auf Strings operieren, nicht über das Ende des Arrays hinaus lesen. Ohne die Null-Terminierung würden diese Funktionen über das Ende des Strings hinauslesen, wodurch undefiniertem Verhalten auftreten kann, wie möglicherweise Speicherbereiche zu überschreiben oder ungewollt auf vertrauliche Daten zuzugreifen. Da in einem null-terminierten String das Nullzeichen als Ende des Strings dient, kann der String eine beliebige Anzahl von Zeichen enthalten, solange er mit einem Nullzeichen endet. Dies macht es einfach, die Länge des Strings zu bestimmen und ihn korrekt zu verarbeiten.

In der for-Schleife werden die erhaltenen Daten ausgegeben. Diese Schleife durchläuft jedes Byte im data_rx-Array bis zur Anzahl der tatsächlich gelesenen Bytes. In Zeile 43 werden die empfangenen Bytes

des Arrays an den seriellen Monitor ausgegeben. Die Funktion Serial.print sendet Daten in Textform über die serielle Verbindung zum Computer, wo sie auf dem Monitor angezeigt werden können. Sie kann verschiedene Datentypen und Formate ausgeben. Der erste Parameter ist der Wert oder die Variable, die gedruckt werden soll. Der optionale zweite Parameter gibt das Zahlenformat an. Die möglichen Werte sind:

- DEC (dezimal)
- HEX (hexadezimal)
- OCT (oktal)
- BIN (binär)

In diesem Anwendungsfall werden die Bytes als Hexadezimalzahlen ausgegeben. Die nächste Zeile fügt ein Leerzeichen zwischen den ausgegebenen Hexadezimalwerten ein, um die Lesbarkeit zu verbessern.

7.3.2.3 Ergebnis

```
Sending: 0x01 0x00 0x02 0x10 0x02
Starting tracker Application...
1 0
10
Sending: 0x01 0x00 0x02 0x10 0x02
Starting tracker Application...
1 0
10 E
```

Abbildung 51: Darstellung der Antwort im seriellen Monitor, Quelle: Eigene Darstellung.

Wie in Abbildung 51 ersichtlich, wurde eine Antwort von dem MK UWB Shield 2 mittels der seriellen Kommunikation empfangen. Die empfangenen Bytes werden daraufhin in Hexadezimalzahlen umgewandelt und ausgegeben. Dieser Prozess ermöglicht eine klarere und präzisere Darstellung der Daten, wodurch die Analyse und Fehlerbehebung erleichtert werden. Die Hexadezimaldarstellung bietet den Vorteil, dass sie eine kompakte und leicht interpretierbare Form der Datenrepräsentation ist, insbesondere in der Embedded- und Hardware-Entwicklung.

Des Weiteren werden die TLV ebenfalls in Hexadezimalzahlen angegeben, wodurch die Umschlüsselung der Befehle leichter erfolgen kann. Durch die Angabe der TLV in Hexadezimalformat wird eine effizientere und genauere Kommunikation zwischen den verschiedenen Systemkomponenten gewährleistet. Die erhaltenen Daten können somit direkt zur Weiterverarbeitung oder zur Auswertung in nachfolgenden Schritten genutzt werden und tragen zur Optimierung der Systemperformance bei.

7.3.3 Konfiguration der MK UWB Shield 2

```
50
51  if(start<1)
52  {
53    Serial1.write(10); //sent in Bytes:0x02
54    Serial1.write(0);//sent in Bytes:0x00
55    Serial1.write(1);//sent in Bytes:0x01
56    Serial1.write(100000);//sent in Bytes:0x20
57    Serial.print("Sending: 0x02 0x00 0x01 0x20\n");
58
59    Serial1.write(10); //sent in Bytes:0x02
60    Serial1.write(0);//sent in Bytes:0x00
61    Serial1.write(1);//sent in Bytes:0x01
62    Serial1.write(100001);//sent in Bytes:0x21
63    Serial.print("Sending: 0x02 0x00 0x01 0x21\n");
64    start=2;
65  }
66
```

Abbildung 52: Konfiguration der MK UWB Shield 2, Quelle: Eigene Darstellung.

Wie in Abbildung 52 in den Zeilen 53 bis 63 zu sehen ist, erfolgt die Konfiguration des MK UWB Shield 2 mittels TLV. Die ersten TLV sind für das Starten des MK UWB Shield 2 sowie der Tracker Applikation. Um die Applikation betreiben zu können, müssen die UWB-Tags in der Applikation hinzugefügt werden. Mit den folgenden TLV werden die Tags hinzugefügt:

Befehl: 0x01

Länge: 0x00 | 0x01 (zeigt an, dass die Nachrichtenlänge 1 Byte beträgt)

App-Start: 0x10

Start Scan: 0x20

Dieser initiale Satz von TLV-Kommandos sorgt dafür, dass das MK UWB Shield 2 und die zugehörige Tracker-Applikation in den Betriebsmodus versetzt werden. Dies ist der erste Schritt, um die Umgebung auf vorhandene UWB-Tags zu scannen und sie in das System zu integrieren.

Um den Scan-Vorgang zu initiieren, wird ein weiterer Satz von TLV gesendet:

Befehl: 0x01

Länge: 0x00 | 0x01 (zeigt an, dass die Nachrichtenlänge 1 Byte beträgt)

App-Start: 0x10

Stopp Scan: 0x21

Bei diesen TLV wird nach Tags in der Nähe gesucht. Die Antwort sollte die MAC-Adressen und den Bluetooth LE-Kurznamen der gefundenen Tags enthalten. Dies ermöglicht eine präzise Identifizierung und Verwaltung der Tags innerhalb der Applikation. Die Suche nach den Tags erfolgt durch das

Aussenden eines speziellen Scan-Befehls, der darauf ausgelegt ist, die Anwesenheit und die Details der UWB-Tags in der Umgebung zu erfassen.

Sobald die Tags identifiziert sind, können sie zur weiteren Verfolgung und Datenverarbeitung in das System integriert werden. Die genaue Kenntnis der MAC-Adressen und der Bluetooth LE-Kurznamen ist entscheidend für die Zuordnung und Verwaltung der einzelnen Tags. Die durch den Scan-Prozess gewonnenen Informationen bilden die Grundlage für die weitere Nutzung und Optimierung der Applikation, wie zum Beispiel die genaue Positionierung und Verfolgung der Tags in Echtzeit.

```
Sending: 0x002 0x10 0x02
Starting tracker Application...
1 0
Sending: 0x02 0x00 0x01 0x20
Sending: 0x02 0x00 0x01 0x21
F0 FE
```

Abbildung 53: Antwort des MK UWB Shield 2, Quelle: Eigene Darstellung.

Die Antworten des MK UWB Shield 2, wie in Abbildung 53 ersichtlich, sind im Rahmen des Projekts dieser Arbeit stets unterschiedlich und inkonsistent. Auch durch wiederholtes Senden der gleichen Anfragen kann kein Muster in den erhaltenen Antworten identifiziert werden. Die empfangenen Nachrichten lassen sich zudem nicht anhand der TLV-Tabelle übersetzen. Dies bedeutet, dass die Struktur und der Inhalt der Antworten nicht den erwarteten Standards entsprechen und die Interpretation sowie Analyse erheblich erschweren.

Aufgrund dieser Unregelmäßigkeiten ist es nicht möglich, den genauen Bedeutungsinhalt der empfangenen Antworten zu ermitteln. Dieses Verhalten führt zu erheblichen Schwierigkeiten in der Fehlerdiagnose und Systemintegration. Ohne ein klar verständliches Muster oder eine zuverlässige Übersetzungsmethode für die empfangenen Daten kann keine effektive Kommunikation zwischen dem MK UWB Shield 2 und den anderen Systemkomponenten etabliert werden.

7.4 Zusammenführen der Messdaten und Steuerung

Es werden die Projekte von Abschnitt 7.2 und 7.3 zusammengefügt. Zunächst werden alle benötigten Bibliotheken, PIDLoop.h und Servo.h, eingebunden und Variablen deklariert. Die Variablen werden zum Speichern der erhaltenen Daten des MK UWB Shield 2 und zur Berechnung des PID-Reglers verwendet. Des Weiteren werden die Servomotoren deklariert und der PID-Regler initialisiert.

In der Setup()-Funktion werden die zwei seriellen Kommunikationen für den seriellen Monitor und das MK UWB Shield 2, mit einer Baudrate von 115200, initialisiert, die Pins für die Servos definiert und die benötigte Initialisierung des MK UWB Shield 2 gesendet.

```
69
70  void loop()
71  {
72
73    if(Serial1.available()>0)
74    {
75      Serial.print("recieving Data\n");
76      UWB_read();
77      Serial.print("adjustment\n");
78      car();
79
80    }
81
```

Abbildung 54: Loop()-Funktion, Quelle: Eigene Darstellung.

In der Loop()-Funktion, abgebildet in Abbildung 54, wird kontinuierlich überprüft, ob Daten vom MK UWB Shield 2 empfangen werden und es werden zwei verschiedene Funktion ausgeführt.

Zuerst wird die Funktion UWB_read ausgeführt. In der Funktion werden die empfangen Daten des MK UWB Shield 2 gelesen und in einer Variable gespeichert, wie in Abschnitt 7.3.2 erklärt. Diese Daten dienen als Grundlage für die nachfolgenden Berechnungen und Anpassungen. Die UWB_read-Funktion verwendet die serielle Schnittstelle, um die Daten vom UWB-Board zu empfangen und stellt sicher, dass die Daten korrekt und vollständig in das data_rx Array eingelesen werden. Anschließend werden die empfangenen Daten hexadezimal auf dem seriellen Monitor ausgegeben, um die Debugging-Möglichkeiten zu verbessern und sicherzustellen, dass die Kommunikation mit dem UWB-Board ordnungsgemäß funktioniert.

Nach der Funktion UWB_read wird die Funktion car ausgeführt. In dieser Funktion wird der PID-Regler mit dem aktuellen Fehlerwert aktualisiert. Der Fehler wird dabei als Differenz zwischen einem festgelegten Referenzwert und dem aktuell gemessenen Wert definiert. Der PID-Regler berechnet basierend auf diesem Fehler die notwendigen Anpassungen, um die Abweichung zu minimieren. Diese Anpassungen betreffen die Steuerwinkel der Servos, die für die Lenkung des Fahrzeugs verantwortlich sind. Diese Vorgehensweise und die Funktionsweise des PID-Reglers sind detailliert in Abschnitt 7.2 beschrieben.

7.5 Programmierung des UWB-Kits

Das MK UWB Kit wird mit einem Übungsprojekt für die Anwendungsfälle Abstandswarnung und Tracker ausgeliert. Für die vorliegende Arbeit wurde der Anwendungsfall Tracker herangezogen. In diesem Projekt werden die notwendigen Komponenten für den Betrieb des MK UWB Shield 2 initialisiert. So wird die Kommunikation für den Datenaustausch über UART definiert. Es sind nur kleine Anpassungen in den wichtigsten Abschnitten des Codes nötig.

```
43
44 App_Result_t Tracker_StartBleScan() {
45
46     App_Result_t status = gAppSuccess_c;
47
48     if (mState != TRACKER_IDLE) {
49         status = gInvalidState_c;
50         return status;
51     }
52
53     status = MK_UwbStartDetection(BLE_ONLY_DETECTION, TYPE_TAG, POLICY_ALL);
54
55     if (status == MK_UWB_SUCCESS) {
56         mState = TRACKER_SCAN;
57         status = gAppSuccess_c;
58     }
59
60     return status;
61 }
62
63 App_Result_t Tracker_StopBleScan() {
64
65     App_Result_t status = gAppSuccess_c;
66
67     if (mState != TRACKER_SCAN) {
68         status = gInvalidState_c;
69         return status;
70     }
71
72     status = MK_UwbStopDetection();
73
74     if (status == MK_UWB_SUCCESS) {
75         mState = TRACKER_IDLE;
76         status = gAppSuccess_c;
77     }
78
79     return status;
80 }
81
```

Abbildung 55: Code für das MK UWB Shield 2, Quelle: Eigene Darstellung.

In der MK UWB SDK befindet sich unter app→Anchor die Datei tracker.c, in welcher der wichtigste Code für den Anwendungsfall Tracker ist. Der Code ist als Zustandsmaschine aufgebaut. Wie in der Abbildung 55 zu sehen, startet der Code mit der Funktion Track_StratBleScan. Der Funktion werden drei Argumente übergeben. Das erste Argument gibt an, dass Geräte nur über Bluetooth zu erkennen sind: das zweite Argument, dass Benachrichtigungen von Tags zu erhalten sind: das dritte Argument gibt an, dass von allen verfügbaren Tags eine Antwort zubekommen ist.

```
82⊖ App_Result_t Tracker_TrackDevice(MK_BleMac_t addr) {
83
84      App_Result_t status = gAppSuccess_c;
85
86      if ((mState != TRACKER_IDLE && numConnectedDevices == 0) ||
87              (mState != TRACKER_TRACK && numConnectedDevices > 0)) {
88          status = gInvalidState_c;
89          return status;
90      }
91
92      status = MK_UwbWhitelistAdd(addr);
93      if (status != MK_UWB_SUCCESS) {
94          return status;
95      }
96      numConnectedDevices++;
97
98      if(numConnectedDevices == 1) {
99          status = MK_UwbStartDetection(BLE_AND_UWB_DETECTION, TYPE_TAG, POLICY_WHITELIST);
100         if(status != MK_UWB_SUCCESS){
101             return status;
102         }
103     }
104
105     mState = TRACKER_TRACK;
106     status = gAppSuccess_c;
107     return status;
108 }
109
110⊖ App_Result_t Tracker_UntrackDevice(MK_BleMac_t addr) {
111
112     App_Result_t status = gAppSuccess_c;
113
114     if (mState != TRACKER_TRACK) {
115         status = gInvalidState_c;
116         return status;
117     }
118
119     status = MK_UwbWhitelistRemove(addr);
120     if (status != MK_UWB_SUCCESS) {
121         return status;
122     }
123     numConnectedDevices--;
124
125     if (numConnectedDevices == 0) {
126         status = MK_UwbStopDetection();
127         if (status != MK_UWB_SUCCESS) {
128             return status;
129         } else {
130             mState = TRACKER_IDLE;
131         }
132     }
133
134     status = gAppSuccess_c;
135     return status;
136 }
137
```

Abbildung 56: Code für das MK UWB Shield 2, Quelle: Eigene Darstellung.

Anschließend fügt die Funktion Tracker_TrackDevice, wie in Abbildung 56, das ausgewählte Gerät zur Whitelist hinzu und beginnt mit der Erkennung, sondern dies noch nicht geschehen ist. Mit dem Argument BLE_AND_UWB_DETECTION wird sofort mit der Ortung der getrackten Geräte begonnen. Die Funktion Tracker_UntrackDevice führt genau das Gegenteil aus, sie entfernt das angegebene Gerät aus der Whitelist und stoppt die Erkennung, wenn keine weiteren Geräte angeschlossen sind.

8 MESSUNG

8.1 Beschreibung der Messungen

Im Rahmen dieser Arbeit werden zwei Messungen mit dem UWB-Kit durchgeführt, um dessen Funktionalität und Genauigkeit zu bewerten. Das MK UWB Shield 2 kommt dabei in Kombination mit zwei Tags zum Einsatz. Die Entscheidung, zwei Tags zu verwenden, wird bewusst getroffen, um die Messwerte besser verifizieren zu können und mögliche Abweichungen oder Fehler frühzeitig zu identifizieren.

In der ersten Messung werden Punkte entlang einer geraden Linie über eine Distanz von 8 Metern erfasst. Diese Messung dient dazu, zu untersuchen, ob die Genauigkeit der Tags mit zunehmender Entfernung abnimmt. Insbesondere soll festgestellt werden, ob es eine signifikante Verschlechterung der Messgenauigkeit gibt, je weiter sich die Tags vom UWB-Sensor entfernen.

Die zweite Messung konzentriert sich auf die Ermittlung der Genauigkeit der 2D-Positionierung. Hierbei wurden mehrere Punkte erfasst, die nicht auf einer geraden Linie lagen, sondern unterschiedlich im Raum verteilt waren. Ziel dieser Messung ist es, die Präzision des UWB-Systems bei der Bestimmung der Position in zwei Dimensionen zu bewerten und herauszufinden, wie zuverlässig das System komplexere Bewegungsmuster und räumliche Verteilungen abbilden kann.

Durch die Kombination dieser beiden Messungen soll ein umfassendes Bild von der Leistungsfähigkeit des UWB-Kits gewonnen werden. Die Ergebnisse dieser Experimente liefern wichtige Erkenntnisse darüber, wie präzise und verlässlich das UWB-System in verschiedenen Szenarien arbeitet, und sie bieten wertvolle Hinweise auf mögliche Verbesserungs- oder Anpassungspotenziale.

8.2 Ergebnisse der ersten Messung

In Abbildung 57 sind die Ergebnisse der ersten Messung detailliert dargestellt. Die Grafik zeigt zwei Linien, die jeweils die Messdaten eines der beiden verwendeten Tags repräsentieren. Sie veranschaulicht den Verlauf der Messung, indem sie die Veränderung der Entfernung über die Zeit hinweg aufzeigt.

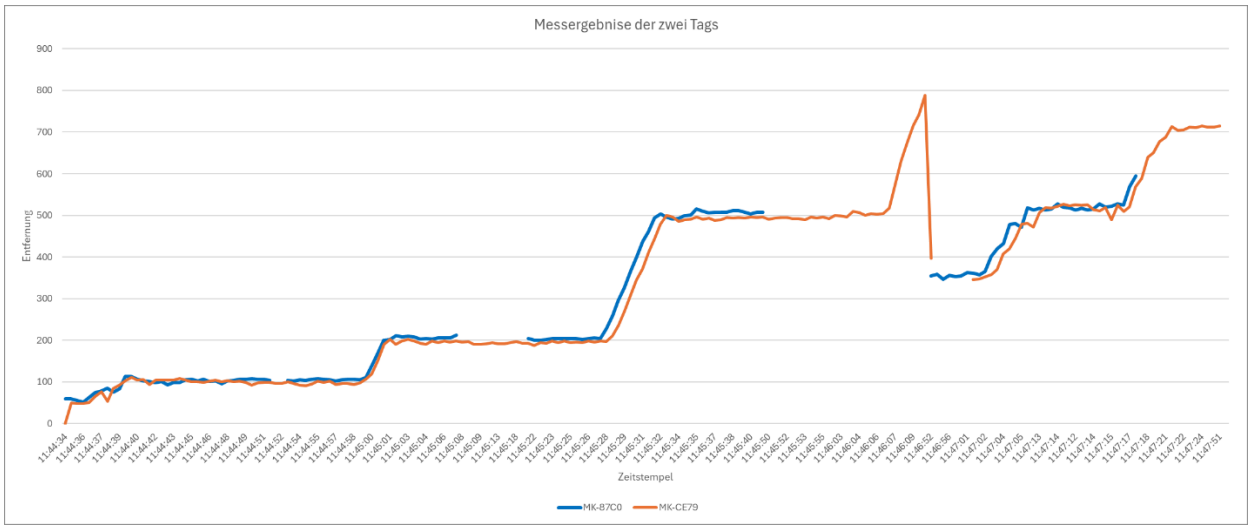


Abbildung 57: Messergebnisse; Quelle: Eigene Darstellung.

Ein bemerkenswertes Merkmal der Grafik ist die Nähe der beiden Linien zueinander und die insgesamt hohe Präzision der Messungen. Die Messergebnisse der beiden Tags bleiben über den gesamten Messbereich hinweg eng beieinander und weisen auf die Zuverlässigkeit und Genauigkeit des UWB-Systems hin. Selbst bei einer maximalen Entfernung von etwa 8 Metern zeigen die Tags eine beeindruckende Genauigkeit und liefern nahezu identische Messwerte. Die maximale Abweichung zwischen den beiden Tags beträgt lediglich ± 2 cm und unterstreicht die hohe Genauigkeit der Positionierung. Diese geringe Abweichung ergibt sich aus der Tatsache, dass die Tags bei der Messung nebeneinandergelegt wurden.

Die Unterbrechungen in den Linienverläufen der Grafik deuten darauf hin, dass das MK UWB Shield 2 zu bestimmten Zeitpunkten die Verbindung zu einem der Tags verloren hat. Diese Unterbrechungen könnten auf temporäre Störungen oder Schwächen in der Signalübertragung hinweisen. Solche Verbindungsverluste sind von Bedeutung, da sie die Zuverlässigkeit des Systems beeinflussen können. Diese potenziellen Schwächen sollten weiter untersucht werden, um herauszufinden, ob sie auf spezifische Umgebungsbedingungen, technische Einschränkungen oder andere Faktoren zurückzuführen sind. Eine genauere Analyse dieser Unterbrechungen könnte wertvolle Informationen liefern, um das System in Zukunft zu optimieren und eine noch höhere Zuverlässigkeit zu gewährleisten. Im Zuge dieser Messung konnte keine Gemeinsamkeit der Unterbrechungen der Kommunikation herausgefunden werden.

8.3 Ergebnisse der zweiten Messung

In Abbildung 58 werden die Ergebnisse der zweiten Messung umfassend dargestellt. Die Grafik visualisiert die 2D-Positionen der beiden verwendeten Tags, indem sie die aufgezeichneten Punkte für jedes Tag zeigt. Diese Darstellung erlaubt es, den Verlauf der Messungen in Bezug auf die räumliche Position der Tags detailliert nachzuvollziehen.

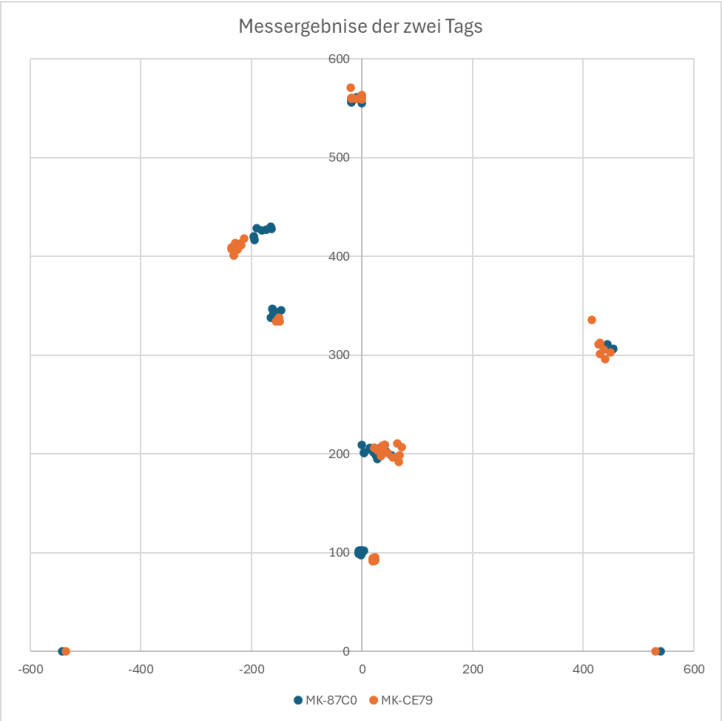


Abbildung 58: Messergebnisse; Quelle: Eigene Darstellung.

In der Grafik ist deutlich erkennbar, dass die erfassten Punkte in einem sehr engen Bereich liegen und weist so auf eine äußerst präzise Messung hin. Alle aufgezeichneten Punkte befinden sich innerhalb eines Kreises mit einem Durchmesser von maximal 10 cm. Die leichten Unterschiede zwischen den beiden Ergebnissen der Tags können möglicherweise darauf zurückgeführt werden, dass die Tags während der Messungen nebeneinander positioniert waren.

Trotz der hohen Dichte der Messpunkte stellte die Auswertung jedoch eine Herausforderung dar, da es unerklärliche Ausreißer gibt. Obwohl die Tags stabil auf dem Boden platziert werden, zeigten einzelne Messwerte Abweichungen von über einem Meter. Diese Ausreißer treten sporadisch auf, obwohl die Messwerte unmittelbar davor und danach konsistent bleiben. Die Ursachen für diese plötzlichen Abweichungen kann nicht vollständig geklärt werden. Es liegt nahe, dass es sich um eine vorübergehende Störung in der Signalübertragung handelt.

Ein weiteres interessantes Ergebnis der Messung ist die Beobachtung, dass sich die Messwerte veränderten, wenn sich ein Objekt, wie etwa eine Person, zwischen dem UWB-Shield und dem Tag bewegte. Diese Bewegung führt zu Störungen und beeinflusst die Messwerte signifikant. Es ist jedoch bemerkenswert, dass im Gegensatz zur ersten Messung keine Unterbrechungen in der Aufzeichnung festgestellt werden und dies auf eine stabilere Signalverbindung während dieser Messung hinweist.

9 ZUSAMMENFASSUNG UND AUSBLICK

Die vorliegende Masterarbeit untersuchte die Anwendung der UWB-Technologie zur präzisen Positionsbestimmung und Steuerung eines autonomen Modellfahrzeugs. Das Projekt konnte aus mehreren Gründen nicht fertiggestellt werden:

9.1 Probleme

Der Code für die Steuerung des Modellfahrzeugs und die Berechnung der Messwerte wurde zwar fertiggestellt, jedoch traten Probleme bei der Kommunikation mit dem MK UWB Shield 2 auf. Obwohl eine Verbindung hergestellt werden konnte, war es nicht möglich, das Board vollständig zu initialisieren und Daten auszutauschen. Um die Messdaten zu erhalten, muss der Code zur Initialisierung des MK UWB Shield 2 fertiggestellt werden. Darüber hinaus müssen die empfangenen Daten aufbereitet und in Variablen für den PID-Regler gespeichert werden.

Ein weiteres Problem trat mit dem Support von Mobile Knowledge auf. Die gestellten Fragen wurden teilweise hilfreich beantwortet, jedoch war die Zeitspanne bis zum Erhalten der Antworten groß. Da viele technische Details und spezifische Konfigurationsprobleme nur mit Hilfe des Supports gelöst werden konnten, war es schwierig, ohne die benötigten Informationen weiterzuarbeiten. Diese Verzögerungen führten dazu, dass in den Phasen, in denen Antworten ausstanden, die Entwicklung und Programmierung ins Stocken geriet.

9.2 Zusammenfassung

Eine mögliche Verbesserung des Gesamtprojektes wäre, die Verwendung von globalen Variablen zu minimieren. Es wäre effizienter, die einzelnen Funktionen so anzupassen, dass die Daten an die Funktionen übergeben und die verarbeiteten Daten zurückgegeben werden. Dies würde nicht nur die Lesbarkeit und Wartbarkeit des Codes verbessern, sondern auch die Zuverlässigkeit und Fehlersuche erleichtern.

Zusätzlich sollte eine Halterung für den Arduino und das MK UWB Shield 2 am Modellfahrzeug erstellt werden, um diese Komponenten sicher zu fixieren. Diese Halterung muss robust und leicht zugänglich sein, um sowohl den Betrieb als auch die Wartung zu erleichtern. Eine ordnungsgemäße Befestigung würde sicherstellen, dass die Elektronik während der Fahrt stabil bleibt und Störungen durch Vibrationen oder Erschütterungen minimiert werden.

Die bereitgestellten Beispiele für Abstandswarnung und Tracking sind äußerst hilfreich und können als wertvolle Vorlagen genutzt werden. Diese Beispiele demonstrieren die grundlegenden Funktionen der UWB-Technologie und erleichtern die Implementierung und Anpassung an spezifische Anwendungsfälle.

Die MK UWB PC Shell stellt ebenfalls eine ausgezeichnete Ressource dar. Sie bietet Beispielcodes für die Ansteuerung des MK UWB Shield 2, die detailliert erläutern, wie die Kommunikation zwischen dem Computer und dem UWB Shield eingerichtet und genutzt werden kann. Diese Beispiele sind besonders nützlich, um ein grundlegendes Verständnis der UWB-Programmierung zu entwickeln und die ersten Schritte bei der Implementierung zu erleichtern.

Trotz der Verfügbarkeit dieser umfassenden Beispiele und Vorlagen konnte im Rahmen dieses Projekts jedoch keine stabile Verbindung zum MK UWB Shield 2 hergestellt werden.

Obwohl im Zuge des Projekts die Kommunikation zum MK UWB Shield 2 nicht funktioniert hat, bietet die Technologie trotzdem Vorteile, die in zukünftigen Projekten genutzt werden können. UWB ist bekannt für seine hohe Präzision in der Abstandsmessung und seine Fähigkeit, auch in komplexen und signalreichen Umgebungen genaue Positionsdaten zu liefern. Dies macht UWB ideal für Anwendungen wie das Follow-Me Modellfahrzeug, da es eine genaue und zuverlässige Verfolgung des Tags ermöglicht. Die breite Bandbreite von UWB-Signalen sorgt zudem für eine geringe Störanfälligkeit gegenüber anderen Funktechnologien und somit die Zuverlässigkeit des Systems erhöht.

LITERATURVERZEICHNIS

Gedruckte Werke (3)

Asche, Rüdiger (2016): *Embedded Controller*, Springer Vieweg, Deutschland

Gerd Küveler, Dietrich (2017): *C/C++ für Studium und Beruf*, Springer Vieweg, Deutschland

Steyer, Ralph (2018): *Programmierung in Python*, Springer Vieweg, Deutschland

Online-Quellen (15)

MobileKnowledge (2023): *MK UWB Shield 2 User Guide v1.4*
https://www.themobileknowledge.com/ [Stand: 15.07.2023]

MobileKnowledge (2023): *Kit 150_040 - Quick Start Guide*
https://www.themobileknowledge.com/ [Stand: 15.07.2023]

MobileKnowledge (2023): *Tracker_distance alert UM v1.3*
https://www.themobileknowledge.com/ [Stand: 15.07.2023]

MobileKnowledge (2023): *UWB Tag SR040 User Guide v1.4*
https://www.themobileknowledge.com/ [Stand: 15.07.2023]

PX4 (2023): *Pixhawk 6X*
https://docs.px4.io/main/en/flight_controller/pixhawk6x.html [Stand: 15.08.2023]

Arduino (2023): *Arduino Mega 2560 Rev3*
https://store.arduino.cc/products/arduino-mega-2560-rev3 [Stand: 15.08.2023]

PX4 (2023): *PX4 Software Overview*
https://px4.io/software/software-overview/ [Stand: 15.08.2023]

NXP (2023): *MCUXpresso IDE - User Guide*
https://www.nxp.com/design/design-center/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE [Stand: 20.07.2023]

Arduino (2023): *Universal Asynchronous Receiver-Transmitter (UART)*
https://docs.arduino.cc/learn/communication/uart/ [Stand: 28.08.2023]

Arduino (2023): *Getting Started with Arduino IDE 2*
https://docs.arduino.cc/software/ide-v2/tutorials/getting-started-ide-v2/ [Stand: 01.09.2023]

FIRA (2023): *The History of Ultra-Wideband (UWB)*
https://www.firaconsortium.org/resource-hub/blog/the-history-of-ultrawideband-uwb [Stand: 01.04.2024]

DHL (2024): *Next-Generation Wireless Logistic*
https://www.dhl.com/us-en/home/insights-and-innovation/thought-leadership/trend-reports/next-generation-wireless-logistics.html [Stand: 01.08.2024]

APPLE (2024): *Nearby Interaction with UWB*
https://developer.apple.com/nearby-interaction/ [Stand: 04.08.2024]

NXP (2024): *VW with NXP Show the Power of UWB for Future Cars* https://www.nxp.com/video/vw-with-nxp-show-the-power-of-uwb-for-future-cars:NXP_VW_UWB [Stand: 06.08.2024]

Tsang, T.K.K.; El-Gamal, M.N. (2005): *Ultra-wideband (UWB) communications systems: an overview* https://ieeexplore.ieee.org/abstract/document/1496688 [Stand: 01.03.2024]

ABBILDUNGSVERZEICHNIS

Abbildung 1: Das MK UWB Kit, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 1

Abbildung 2: Frequenzbereiche verschiedener Übertragungsarten, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 4

Abbildung 3: Ausbreitung der Funkwellen, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 5

Abbildung 4: Empfangene Impulse mit Energieniveaus, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 6

Abbildung 5: Single Sided-Two Way Ranging, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 9

Abbildung 6: Double Sided-Two Way Ranging, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 10

Abbildung 7: Time Difference of Arrival, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023].. 10

Abbildung 8: Angle of Arrival, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 11

Abbildung 9: Arduino MEGA, Quelle: Arduino (2023), Online-Quelle [15.08.2023]..... 14

Abbildung 10: Pixhawk 6X Set, Quelle: PX4 (2023), Online-Quelle [15.08.2023] 17

Abbildung 11: Inhalt des MK UWB Kit, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023].... 18

Abbildung 12: MK UWB Shield 2, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 19

Abbildung 13: MK UWB Shield 2 Blockdiagramm, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 20

Abbildung 14: MK UWB 3D Antenna Board, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 21

Abbildung 15: UWB Tag SR040, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 22

Abbildung 16: UWB Tag SR040 Blockdiagramm, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023]..... 23

Abbildung 17: MK UWB SR040 Board, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] ... 23

Abbildung 18: Beispiel der MK UWB Toolbox Android, Quelle: Eigene Darstellung. 25

Abbildung 19: Beispiel der MK UWB PC Shell-Anwendung, Quelle: Eigene Darstellung. 25

Abbildung 20: MCUXpresso IDE Block Diagramm, Quelle: NXP (2023), Online-Quelle [20.07.2023]..... 26

Abbildung 21: NXP MCUXpresso IDE, Quelle: Eigene Darstellung. 27

Abbildung 22: Arduino IDE, Quelle: Arduino (2023), Online-Quelle [01.09.2023] 29

Abbildung 23: C++ Beispielcode, Quelle: Eigene Darstellung. 32

Abbildung 24: Unterschied serielle / parallele Kommunikation, Quelle: Arduino (2023), Online-Quelle [28.08.2023]..... 33

Abbildung 25: Aufbau UART Nachricht, Quelle: Arduino (2023), Online-Quelle [28.08.2023] 35

Abbildung 26: Beispiele für TLV-Definitionen, Quelle: Eigene Darstellung. 37

Abbildung 27: Board Auswahl in der Arduino DIE, Quelle: Eigene Darstellung. 40

Abbildung 28: Python Installation für die MK UWB PC Shell, Quelle: Eigene Darstellung. 40

Abbildung 29: MK UWB PC Shell Startbildschirm, Quelle: Eigene Darstellung. 41

Abbildung 30: Verbindung MK UWB Shield 2 zu Mobiltelefon, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 41

Abbildung 31: Verbindung MK UWB Shield 2 zu PC, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 42

Abbildung 32: MK UWB Shield 2 Jumpers, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 43

Abbildung 33: Abstandswarnung Beispiel, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 44

Abbildung 34: Tags werden basierend auf Schwellenwerten erkannt und in ihrem jeweiligen Bereich platziert, Quelle: Eigene Darstellung. 45

Abbildung 35: Empfangen von Entfernungsalarmbenachrichtigungen in der MK UWB PC Shell-Anwendung, Quelle: Eigene Darstellung. 46

Abbildung 36: Tracker Beispiel, Quelle: MobileKnowledge (2023), Online-Quelle [15.07.2023] 46

Abbildung 37: Tags in Radaransicht, Quelle: Eigene Darstellung. 47

Abbildung 38: Tracker Anwendung in der MK UWB PC Shell, Quelle: Eigene Darstellung. 48

Abbildung 39: Originales Modellauto, Quelle: Eigene Darstellung. 49

Abbildung 40: Code für globalen Variablen und Bibliotheken, Quelle: Eigene Darstellung. 53

Abbildung 41: Setup () -Funktion, Quelle: Eigene Darstellung. 55

Abbildung 42: Loop () -Funktion, Quelle: Eigene Darstellung. 55

Abbildung 43: Loop () -Funktion für die Lenkung, Quelle: Eigene Darstellung. 56

Abbildung 44: Globale Variablen, Quelle: Eigene Darstellung. 57

Abbildung 45: Setup () -Funktion, Quelle: Eigene Darstellung. 57

Abbildung 46: Verdrahtung des Arduino mit dem MK UWB Shield 2, Quelle: Eigene Darstellung. 58

Abbildung 47: Loop()-Funktion, Quelle: Eigene Darstellung. 59

Abbildung 48: Darstellung der Antwort im seriellen Monitor, Quelle: Eigene Darstellung. 60

Abbildung 49: Globale Variablen und Setup()-Funktion, Quelle: Eigene Darstellung. 61

Abbildung 50: Loop()-Funktion, Quelle: Eigene Darstellung. 62

Abbildung 51: Darstellung der Antwort im seriellen Monitor, Quelle: Eigene Darstellung. 63

Abbildung 52: Konfiguration der MK UWB Shield 2, Quelle: Eigene Darstellung. 64

Abbildung 53: Antwort des MK UWB Shield 2, Quelle: Eigene Darstellung. 65

Abbildung 54: Loop()-Funktion, Quelle: Eigene Darstellung. 66

Abbildung 55: Code für das MK UWB Shield 2, Quelle: Eigene Darstellung. 67

Abbildung 56: Code für das MK UWB Shield 2, Quelle: Eigene Darstellung. 68

Abbildung 57: Messergebnisse; Quelle: Eigene Darstellung. 69

Abbildung 58: Messergebnisse; Quelle: Eigene Darstellung. 71

TABELLENVERZEICHNIS

Tabelle 1 Technische Daten des Arduino MEGA, Quelle: Arduino (2023), Online-Quelle [15.08.2023] ... 15

Tabelle 2 MK UWB Shield 2 Jumper Konfiguration, Quelle: Eigene Darstellung. 42

ABKÜRZUNGSVERZEICHNIS

UWB	Ultra-Wideband
FCC	Federal Communications Commission
TOF	Time of Flight
SS-TWR	Single Sided-Two-Way Ranging
DS-TWR	Double Sided-Two-Way Ranging
TDoA	Time Difference of Arrival
AoA	Angle of Arrival
PDoA	Phase Difference of Arrival
IoT	Internet of Things
AR	Augmented Reality
IDE	Integrated Development Environment
UART	Universal Asynchronous Receiver Transmitter
TLV	Type-Length-Value