St. Pölten University of Applied Sciences

/ medien & digitale technologien

/fh/// st.pölten

Hardware- und App-Entwicklung mit Bluetooth 5

Entwicklung eines konfigurier- und erweiterbaren Bluetooth 5 Beacons

Diplomarbeit

Ausgeführt zum Zweck der Erlangung des akademischen Grades
Dipl.-Ing. für technisch-wissenschaftliche Berufe

am Masterstudiengang Digitale Medientechnologien an der Fachhochschule St. Pölten, **Masterklasse Mobiles Internet**

von:

Werner Gundacker, BSc

dm161515

Betreuer/in und Erstbegutachter/in: Dr. Thomas Moser
Zweitbegutachter/in: Dipl.-Ing. Christian Jandl, BSc

St.Pölten, 10.09.2018

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Arbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.

- ich dieses Thema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter bzw. der Begutachterin beurteilten Arbeit überein.

....

St.Pölten, 10.09.2018 Unterschrift

Gender Erklärung

In dieser Diplomarbeit wird ausschließlich zum Zweck der besseren Lesbarkeit auf die geschlechtsspezifische Schreibweise verzichtet. Alle personenbezogenen Bezeichnungen, die sich sowohl auf männliche als auch auf weibliche Personen beziehen, werden ausschließlich in der männlichen Form angeführt. Zum Beispiel wird „Leser“ anstatt „LeserInnen“ oder „Leserinnen und Leser“ verwendet.

Dies soll jedoch keinesfalls eine Geschlechterdiskriminierung oder eine Verletzung des Gleichheitsgrundsatzes zum Ausdruck bringen.

Kurzfassung

Um Gegenstände innerhalb einer Produktion nachzuverfolgen (Asset Tracking) oder innerhalb von geschlossenen Räumen zu navigieren (Indoor Navigation), können die verschiedensten Technologien eingesetzt werden. Eine davon ist die Bluetooth Technologie in Verbindung mit Bluetooth Beacons. Da aktuelle Bluetooth 4 Beacons aber nur eine eingeschränkte Reichweite von maximal 50 Meter haben, werden in dieser Diplomarbeit die angeblichen Verbesserungen des neuen Bluetooth 5 Standards getestet und überprüft werden.

Dafür wird ein konfigurier- und erweiterbarer Bluetooth 5 Beacon mit einer zugehörigen mobilen Applikation für die Betriebssysteme Android und iOS entwickelt. Das einzigartige an dem Beacon sind seine beiden Signaleinheiten, er besitzt eine leuchtstarke optische LED-Kuppel und einen akustischen Signalgeber, welche bei der Ortung helfen und so das Auffinden von Paletten oder Gegenständen in großen Gebäuden vereinfachen und beschleunigen.

Wie sich im Laufe der Entwicklung herausstellte gibt es derzeit kein Smartphone, welches den Bluetooth 5 Standard vollständig unterstützt. Zudem sind derzeit auch nur wenige Bluetooth-Boards erhältlich, welche den neuen Standard vollständig unterstützen.

Durch die Tests, welche im Laufe dieser Diplomarbeit durchgeführt worden sind, konnte aber bewiesen werden, das Bluetooth 5 durch den neuen Long Range Modus eine deutlich größere Reichweite hat. Weiters wurde bestätigt, dass die Umsetzung eines konfigurier- und erweiterbaren Bluetooth 5 Beacons mit frei verfügbarer Hardware und Open-Source Software möglich ist. In Sachen Akkulaufzeit und Gesamtkosten kommt der entwickelte Prototyp aber nicht an kommerzielle Beacons heran.

Abstract

To track items within a production (asset tracking) or to navigate within closed spaces (indoor navigation), a variety of technologies can be used. One of these is Bluetooth in conjunction with Bluetooth beacons. Because current Bluetooth 4 beacons only have a limited range of a maximum of 50 meters, the alleged improvements of the new Bluetooth 5 standard will be tested and checked in this diploma thesis.

Therefore, a configurable and expandable Bluetooth 5 beacon with a corresponding mobile application for the two operating systems Android and iOS will be developed. The unique feature of the beacon are the two signal units. It has a bright optical LED dome and an acoustic signal generator, which will help to locate objects in large buildings easier and faster.

During the development, it turned out, that there are no smartphones on the market which fully supports the new Bluetooth 5 standard. Furthermore, there are only a few Bluetooth modules or boards available which fully support the new standard.

The higher range of the Bluetooth 5 Long Range mode has been verified through the tests, which were carried out in this diploma thesis. The creation of a Bluetooth beacon with freely available hardware and open source software was also confirmed, but in terms of battery life and total costs, the developed prototype is not comparable to commercial beacons.

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich im Rahmen dieser Diplomarbeit begleitet und unterstützt haben.

Besonders möchte ich mich bei Herrn Dr. Thomas Moser und Herrn Dipl.-Ing. Christian Jandl, BSc bedanken, die meine Arbeit durch ihre fachliche und persönliche Unterstützung begleitet haben und mir überhaupt erst die Möglichkeit gaben, diese Diplomarbeit unter ihrer Betreuung durchzuführen.

Für die Unterstützung beim 3D-Druck und bei der Auswahl einiger wichtiger Komponenten für den Prototyp möchte ich mich bei Herrn Dipl.-Ing. Christoph Braun, BSc recht herzlich bedanken.

Darüber hinaus möchte ich mich bei meiner Mutter Maria und bei meinen Geschwistern bedanken, die mir durch ihre Unterstützung immer wieder Kraft gegeben haben, um das komplette Studium und diese Diplomarbeit erfolgreich abzuschließen.

Ein weiterer Dank gilt meinen engsten Freunden, die mich immer wieder neu motiviert haben.

--- DANKE ---

Rappottenstein, 10.09.2018 Werner Gundacker, BSc

Inhaltsverzeichnis

1 Einleitung

Die Version 5 der bekannten Funktechnologie Bluetooth, welche Ende 2016 angekündigt wurde, ist nun der neue Standard. Das erste Smartphone mit einem Bluetooth 5 fähigen Modul ist das Samsung Galaxy S8, welches im zweiten Quartal 2017 vorgestellt wurde (Kolkmann, 2017).

Bluetooth 5 soll laut der Bluetooth SIG (Special Interest Group), der Interessengemeinschaft hinter der Entwicklung der Bluetooth-Technologie, einige wesentliche Verbesserungen mit sich bringen. Unter anderem soll sich die Geschwindigkeit verdoppeln (vgl. Bluetooth 4: max. 1Mbit/s, Bluetooth 5: bis zu 2Mbit/s) und die Reichweite vervierfachen (vgl. Bluetooth 4: max. 50 Meter, Bluetooth 5: bis zu 200 Meter) (Bluetooth SIG, Inc, 2017, S. 6). Zusätzlich ist es durch die 8-fache Kapazität erstmals möglich, zum Beispiel mit dem Samsung Galaxy S8, zwei Bluetooth-Lautsprecher gleichzeitig mit Ton zu versorgen (Gürayer, 2017). Es gibt noch mehr Neuerungen von Bluetooth 5, diese werden im ersten Teil der Diplomarbeit genauer erarbeitet und betrachtet.

Diese Neuerungen werden auch großen Einfluss auf das Internet der Dinge haben. Laut Gartner, soll es bis 2020 bereits über 20 Milliarden vernetzte Geräte im Internet der Dinge geben (Gartner, 2017). Deshalb ist es das Ziel dieser Arbeit, einige der neuen Möglichkeiten von Bluetooth 5 zu testen und diese dann in einem Prototyen umzusetzen. Bei diesem Prototyp handelt es sich um einen intelligenten Bluetooth 5 Beacon, welcher zuerst eigenständig verwendet wird. Bei einem Beacon handelt es sich grundsätzlich um einen kleinen Bluetooth-Sender, welcher in regelmäßigen Abständen Signale aussendet (Bhargava, 2017, S. 155). Ist der Beacon in der Reichweite eines Empfängers (Smartphone, Bluetooth Gateway, ...), kann dieser Empfänger die eindeutige Kennung des Beacons und die Sendeleistung auslesen und weiterverarbeiten.

Im Rahmen der Diplomarbeit wird kein „einfacher“ Beacon entwickelt, sondern ein intelligenteres System bestehend aus einer Bluetooth-Signaleinheit und einer mobilen Applikation. Die mobile Applikation zeigt alle Beacons in Reichweite an und mit einem Klick kann der Signalgeber eines beliebigen Beacons ausgelöst werden.

In einem parallel gestarteten Forschungsprojekt soll dieser Beacon dann im Zusammenspiel mit Bluetooth 5 Gateways und einem entsprechenden Backend für das Asset Tracking in einem Unternehmen getestet werden. Die Bluetooth Gateways überwachen ständig die von den Beacons ausgesendeten Signale und geben die Daten (unter anderem die eindeutige Kennung des Beacons und die Sendeleistung (RSSI - Received Signal Strength Indication)) an das Backend weiter, welches dann die Daten weiterverarbeitet.

Dieses Asset Tracking System ermöglicht die Wege der Produkte (jeweils mit einem Beacon gekoppelt) innerhalb einer Produktion nachzuverfolgen und verhindert somit ein „Verlorengehen“ eines Produktes bzw. sorgt für bessere Nachvollziehbarkeit von Produktionsprozessen. Zusätzlich soll es die Produktionsmitarbeiter dabei unterstützen ein gewünschtes Produkt einfacher und schneller innerhalb einer Lagerhalle zu finden. Für diesen Anwendungsfall wird die oben bereits erwähnte mobile Applikation verwendet. Das Thema der Diplomarbeit fällt somit auch unter den Begriff „Industrie 4.0“, unter diesem Marketingbegriff versteht man die sogenannte vierte industrielle Revolution. Die Vernetzung von Geräten oder das automatische Tracking von Gütern sind nur einige Themenbereiche die dort hineinfallen (Prof. Dr. Bendel, 2018).

1.1 Relevanz und Motivation

Bluetooth Beacons sind nicht neu, es gibt bereits eine Vielzahl davon, jedoch aktuell nur sehr wenige bis gar keine, welche Bluetooth 5 unterstützen und/oder über eine Signalreinrichtung verfügen. In großen Lagerhallen stößt Bluetooth 4 mit einer Reichweite von nur rund 10 Metern oder max. 50 Metern bei direkter Sichtverbindung bald an seine Grenzen. Zusätzlich enthält Bluetooth 5 einige Updates, die dazu beitragen mögliche Interferenzen mit anderen drahtlosen Technologien zu reduzieren, und es somit ermöglicht das mehrere Geräte nebeneinander gut funktionieren können (Bluetooth SIG, Inc., o. J.). Aus diesen Gründen ist es sinnvoll ein Asset Tracking System mit Bluetooth 5 aufzubauen und zu testen.

Die entwickelte Bluetooth-Signaleinheit soll aber nicht nur für das Asset Tracking verwendet werden können, sondern ist so konzipiert, dass sie auch für andere Anwendungsfälle wie zum Beispiel Indoor-Navigation verwendet werden kann. Es handelt sich somit um einen konfigurierbaren und erweiterbaren Bluetooth 5 Beacon, welcher mit frei verfügbarer Hardware und Open-Source Software umgesetzt wird und sich somit klar von vorhandenen kommerziellen Beacons unterscheidet.

1.2 Forschungsfrage und Hypothesen

Im Rahmen der Diplomarbeit sollen folgende Forschungsfragen beantwortet werden:

Frage 1:

Zu welchem Gesamtpreis in Zusammenhang mit sinnvollen Funktionen kann ein funktionierender Bluetooth 5 Beacon umgesetzt werden?

Hintergrund: Der Preis eines Bluetooth Beacons ist ein wichtiger Punkt, denn bei einem Asset Tracking System oder auch bei einem Indoor-Navigation-System werden eine Vielzahl von Beacons benötigt.

Beantwortung: Die Preise aller Komponenten des Prototyps werden zusammengerechnet, zusätzlich werden die Preise für mögliche zusätzliche Komponenten, welche für sinnvolle Erweiterungen (Funktionen) notwendig wären, recherchiert und eine Kostenübersicht erstellt. Diese wird anschließend mit Beacons, welche ähnliche Eigenschaften haben, verglichen.

Forschungsmethoden: Umsetzung des Prototyps, Literaturrecherche, Vergleich

Frage 2:

Welche Signalgeber sind in hellen großen Räumen am besten für die Positionsanzeige des Beacons geeignet? (Optisch, Akustisch, ...)

Hintergrund: Mit Hilfe des entwickelten Prototyps sollen Gegenstände innerhalb heller Lagerhallen leichter und schneller gefunden werden können. Zu diesem Zweck verfügt er über einen optischen und akustischen Signalgeber.

Beantwortung: Mit Hilfe eines Tests unter Laborbedingungen werden unterschiedliche Signalgeber auf ihre Erkennbarkeit getestet. Wie dieser Test genau abläuft, wird im Kapitel 4.3 beschrieben.

Forschungsmethode: Experiment mit Hilfe des erstellten Prototyps

Frage 3:

Welche Akkulaufzeit des Beacons ist realistisch möglich und in welchem Verhältnis stehen Akkukapazität und Akkulaufzeit in Beziehung auf den umgesetzten Prototypen?

Hintergrund: Eine hohe Akkulaufzeit ist für einen Bluetooth Beacon sehr wichtig, ein Laden oder Austauschen des Akkus soll nur in großen Zeitabständen erfolgen müssen.

Beantwortung: Der durchschnittliche Leistungsverbrauch des Prototyps wird in einem bestimmten Zeitraum und unter bestimmten Bedingungen ermittelt und anschließend die realistisch mögliche Akkulaufzeit berechnet.

Forschungsmethode: Experiment mit dem erstellten Prototyp

Frage 4:

Welche Entwicklungsoption (Web, Hybrid, Native, ...) ist für mobile Applikationen mit der Verwendung von Bluetooth am sinnvollsten?

Hintergrund: Eine mobile Applikation kann aktuell auf drei verschiedene Arten (Web, Hybrid, Native, ...) entwickelt werden. Für diese Diplomarbeit ist es aber wichtig, dass die Funktechnologie Bluetooth verwendet werden kann.

Beantwortung: Durch eine ausführliche Recherche der Entwicklungsoptionen und einem anschließenden Vergleich aller Vor- und Nachteile wird die sinnvollste Option ausgewählt.

Forschungsmethode: Literaturrecherche mit anschließender empirischer Evaluation

Frage 5:

Welchen Mehrwert bringt ein auf Bluetooth 5 basierendes Asset Tracking Systems unter Laborbedingungen?

Hintergrund: Der entwickelte Prototyp soll später in einem Asset Tracking System verwendet werden. Daher ist es sinnvoll den Mehrwert von Bluetooth 5 im Vergleich zu Bluetooth 4 zu ermitteln.

Beantwortung: Bei einem Labortest soll der mögliche Mehrwert (z.B.: erhöhte Reichweite, ...) ermittelt werden.

Forschungsmethode: Experiment mit Hilfe des erstellten Prototyps

Folgende aufgestellte Hypothesen sollen durch die Arbeit untersucht werden:

- Ein Bluetooth 5 Beacon bietet, im Vergleich zu Bluetooth 4 Beacons, eine höhere Reichweite.
- Es ist möglich einen konfigurierbaren und erweiterbaren Bluetooth 5 Beacon, mit frei verfügbarer Hardware und Open-Source Software umzusetzen.

1.3 Gliederung der Arbeit

Zu Beginn der Arbeit gibt eine State-of-the-Art Recherche Aufschluss über den derzeitigen Status quo im Bereich Asset Tracking, Indoor-Navigation, Bluetooth 5 und Bluetooth Beacons und zeigt relevante Forschungsprojekte im Bereich von Industrie 4.0 auf. Daraus werden die notwendigen Funktionen eines Bluetooth-Beacon-Systems abgeleitet. Zusätzlich werden in diesem Kapitel die möglichen Entwicklungsoptionen für mobile Applikationen in Bezug auf Bluetooth-Applikation recherchiert und anschließend die beste Option ausgewählt mit der der Prototyp umgesetzt wird.

Der zweite Teil der Arbeit beginnt mit der Anforderungsanalyse für den Prototyp des Bluetooth-Beacons-Systems. Diese enthält sowohl die notwendigen Funktionen des Beacons und der mobilen Applikation als auch die benötigte Hard- und Software. In Bezug auf die mobile Applikation werden Wireframes erstellt und beschrieben, sowie diese danach in fertige Mockups umgesetzt. Danach wird in diesem Kapitel die Umsetzung des Bluetooth-Beacon-Systems und die dafür notwendigen einzelnen Schritte beschrieben.

Im dritten Teil (Kapitel 4) wird das erstellte System einigen Tests unter Laborbedingungen unterzogen. Neben einem Reichweitentest und einem Akku-Laufzeittest wird mittels eines Versuches die Erkennbarkeit der Signalgeber getestet.

Im letzten Teil werden die gestellten Forschungsfragen beantwortet und die Hypothesen verifiziert. Zum Schluss werden noch weiterführende Fragen und Weiterentwicklungsmöglichkeiten angeführt.

1.4 Begriffe

Bluetooth Beacon

Bei einem Bluetooth Beacon handelt es sich grundsätzlich um einen kleinen Bluetooth-Sender, welcher in regelmäßigen Abständen Signale aussendet. Inhalt dieser Signale sind die eindeutige Kennung des Beacons und die Sendeleistung (RSSI).

Bluetooth Gateway

Bluetooth Gateways überwachen ständig die von Bluetooth-Beacons ausgesendete Signale und geben die Daten an ein Backend weiter, welches die Daten weiterverarbeitet. Es ist aber auch möglich das die Bluetooth Gateways selbst die Daten weiterverarbeiten und entsprechende Aktionen ausführen.

Asset Tracking

Ein Asset Tracking System ermöglicht es die Wege von Produkten (Assets) innerhalb einer Produktion nachzuverfolgen und somit ein „Verlorengehen“ eines Produktes zu verhindern. Es kann aber auch für eine bessere Nachvollziehbarkeit von Produktionsprozessen führen. Im Falle der Diplomarbeit soll es die Produktionsmitarbeiter zudem auch dabei unterstützen ein gewünschtes Produkt einfacher und schneller innerhalb einer Lagerhalle zu finden. Für diesen Anwendungsfall wird eine mobile Applikation verwendet.

2 Theoretischer Hintergrund

Dieses Kapitel enthält den aktuellen Stand der Technik im Bereich Asset Tracking und Indoor-Navigation. Aus diesen zwei Bereichen sind relevante Forschungsprojekte und fertige Systeme angeführt. Weiters beinhaltet dieses Kapitel nähere Beschreibungen des neuen Bluetooth 5 Standard sowie aktueller Bluetooth Beacons und die dafür verwendeten Protokolle. Am Ende des Kapitels sind die Entwicklungsmöglichkeiten von mobilen Applikationen in Bezug auf eine Bluetooth-Verbindung angeführt.

2.1 Industrie 4.0

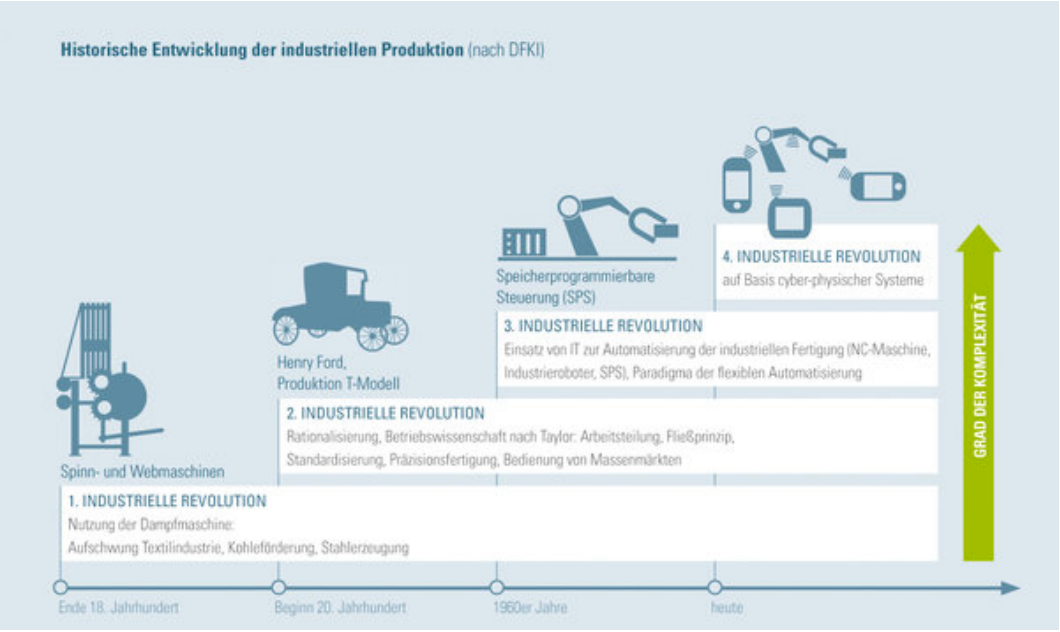

Abbildung 1: Die vier Stufen der industriellen Revolution (it´s owl, o. J.)

Der Begriff „Industrie 4.0“, unter dem auch die sogenannte vierte industrielle Revolution verstanden wird, ist innerhalb kurzer Zeit zu einem Schlagwort geworden. Die ersten drei industriellen Revolutionen (siehe Abbildung 1) wurden jedoch erst im Nachhinein so bezeichnet. Daher gibt es Gruppierungen die diese

Wahl des Begriffes kritisieren. Ihrer Meinung nach sind die aktuellen Entwicklungen nicht so essenziell um sie als vierte industrielle Revolution zu bezeichnen, sondern sehen es eher als einen weiteren Schritt der dritten industriellen Revolution. Dais (2017, S. 259) meint dazu, dass wir aktuell noch am Beginn stehen und es durchaus sein könnte, dass die aktuellen Entwicklungen im Nachhinein als vierte industrielle Revolution angesehen werden könnten.

In dem Paper „Design Principles for Industrie 4.0 Scenarios“ (Hermann, Pentek, & Otto, 2016, S. 3932 f.) werden vier Gestaltungsprinzipien (Vernetzung, Informationstransparenz, Technische Assistenz und Dezentrale Entscheidungen) für die Umsetzung von Industrie 4.0. beschrieben. Das erste Prinzip beschreibt die Vernetzung und die Kommunikation von Maschinen, Menschen, Sensoren und weiteren Bestandteilen. Ob es nun die vierte industrielle Revolution ist, oder auch nicht, unter dem Begriff Industrie 4.0 fällt demnach jedenfalls die Vernetzung von Geräten und auch das automatische Tracking von Wirtschaftsgütern. Genau in diese Themenbereiche fällt auch diese Arbeit, deshalb werden in den nächsten zwei Abschnitten die Begriffe Asset Tracking und Indoor-Navigation genauer erklärt.

2.1.1 Asset Tracking

Der Begriff Asset Tracking beschreibt einen sehr breiten Bereich der Positionsbestimmung, -aufzeichnung und -nachverfolgung unterschiedlicher Gegenstände. Darunter fallen zum Beispiel das Tracking von Fahrzeugen oder Personen per GPS (=Global Positioning System), das Nachverfolgen von Paketen/Produkten mittels eindeutigem aufgedruckten Code, das Aufzeichnen der Wege von Produkten per RFID-Tag und auch das Tracken von Produkten oder Materialien innerhalb einer Produktion mit Hilfe von Bluetooth Beacons.

Diese Diplomarbeit beschränkt sich auf Indoor-Asset-Tracking, das heißt großteils innerhalb von Gebäuden, sollen die Wege von Produkten (Assets) innerhalb einer Produktion nachverfolgt und die aktuelle Position abgerufen werden können. Somit soll ein „Verlorengehen“ eines Produktes oder ein zeitintensives Suchen benötigter Produkte verhindert werden. Da GPS, auf Grund der nicht vorhandenen Sichtverbindung zu den GPS-Satelliten, in geschlossenen Räumen nicht funktioniert, fällt diese Möglichkeit weg. Stattdessen können unter anderem die Technologien RFID (= Radio-Frequency Identification), WLAN (= Wireless Local Area Network), UWB (= Ultra-wideband) und Bluetooth verwendet werden. Es gibt aber auch Systeme die einfache aufgedruckte Codes (z.B. Barcodes oder QR-Codes) verwenden.

2.1.2 Indoor-Navigation

Mobile Applikationen für die Navigation auf öffentlichen Straßen sind weithin bekannt und werden sehr gerne verwendet. Aber wie kann die Navigation innerhalb von großen und oftmals mehrstöckigen Gebäuden ermöglicht werden? Wie im vorigen Kapitel bereits erwähnt, funktioniert GPS in geschlossenen Räumen gar nicht, in Räumen mit Fenstern ist es unter Umständen möglich das eine GPS Position bestimmt werden kann. Diese ist dann aber normalerweise nicht sehr genau, wodurch das Bestimmen des aktuellen Stockwerkes nur sehr schwer möglich ist.

Hier können zum Beispiel Bluetooth Beacons zum Einsatz kommen, welche an markanten Positionen innerhalb eines Gebäudes angebracht werden. Wird nun eine entsprechende mobile Applikation gestartet kann innerhalb eines Gebäudes navigiert werden. Durch die Auswahl eines bestimmten Zielorts, wird die Navigation gestartet. Ab nun werden alle möglichen Bluetooth Beacons in der Umgebung gescannt. Im Gegensatz zu Asset Tracking Systemen sind die Beacons, bei Indoor-Navigationssystemen an fixen Positionen angebracht. Deshalb ist eine Navigation innerhalb der geschlossenen Räume und über mehrere Stockwerke hinweg möglich. Es können aber auch andere Technologien wie WLAN, UWB oder RFID zum Einsatz kommen. UWB und RFID werden in Bezug auf Indoor-Navigation im Rahmen dieser Diplomarbeit nicht weiterbehandelt, da einem Benutzer dazu zusätzliche Geräte zur Verfügung stehen müssten. Bei der Verwendung von WLAN oder Bluetooth hingegen reicht ein herkömmliches Smartphone aus.

2.1.3 Unterschiede Asset Tracking – Indoor-Navigation

Für Asset Tracking Systeme und Indoor-Navigationssysteme kann dieselbe Technologie (z.B. Bluetooth, ...) verwendet werden. Bei beiden Systemen werden (Bluetooth) Beacons eingesetzt, welche immer ihre eindeutige Kennung und die Sendeleistung aussenden (vgl. Kapitel 1.4). Bei Asset Tracking Systemen, bewegen sich diese Beacons im Gebäude und werden von (Bluetooth) Gateways gescannt. Die empfangenen Daten werden am Gateway verarbeitet und anschließenden an einen Server übertragen, welcher die Daten weiterverarbeitet. Im Gegensatz dazu sind die Beacons bei der Indoor-Navigation an fixen Positionen innerhalb eines Gebäudes angebracht und werden von einem Smartphone gescannt. In Verbindung mit einem Server, welcher weiß wo im Gebäude sich die Beacons genau befinden, ist es möglich, dass ein Benutzer durch das Gebäude navigiert wird.

2.2 Mögliche Technologien

Für Asset-Tracking und Indoor-Navigation können wie oben bereits erwähnt verschiedenste Technologien verwendet werden. Folgend sind die bekanntesten Technologien mit ihren Funktionen sowie Vor- und Nachteilen aufgezeigt.

2.2.1 Aufgedruckte Codes

Zu diesen Codes zählen eindimensionale Barcodes, auch Strichcode genannt, und die zweidimensionalen QR-Codes (siehe Abbildung 2). Im Gegensatz zu Strichcodes, welche nur eine eindeutige Identifikationsnummer enthalten, können in QR-Codes zusätzliche Daten (z.B.: Produktname, Gewicht, ...) gespeichert werden.

Abbildung 2: Vergleich Barcode und QR-Code (Quora, 2017)

Ein weiterer Vorteil von QR-Codes ist, sie können zu einem Teil (max. 30%) verdeckt oder verschmutzt sein, und können trotzdem, zum Beispiel mit einer Smartphone-Kamera, aus jeder Blickrichtung eingelesen werden (Junnila, 2017).

Die Einfachheit der Codes liegt somit auf der Hand, es wird keinerlei Stromversorgung benötigt und es ist somit die günstigste Variante um Produkte zu tracken. Beim Einlesen eines Codes mit einem Smartphone wäre es möglich die GPS-Position des Gerätes zu verwenden um die Position des entsprechenden Produktes mitzuspeichern. Hier liegt aber, im geplanten Anwendungsfall, wieder das Problem bei der nicht vorhandenen GPS Verbindung innerhalb von Gebäuden. In geschlossenen Räumen wäre es durchaus möglich stationäre Scannern zu verwenden, um so die Produkte innerhalb der Produktion automatisch verfolgen zu können. Dadurch kann aber nicht die genaue Position bestimmt werden, sondern nur die Information, dass ein Produkt an einem bestimmten Ort zu einer bestimmten Zeit vorbeigekommen ist. Für ein Echtzeittracking und die genaue Positionsbestimmung sind diese aufgedruckten Codes somit nicht zu verwenden.

2.2.2 NFC – Near Field Communication

NFC basiert auf der Technologie von RFID (siehe 2.2.3) und verwendet die gleiche Frequenz wie High Frequency RFID (13,56 MHz). Der Unterschied zu RFID besteht in der geringeren Reichweite von nur maximal 10 cm. (Elektronik-Kompendium.de, o. J.-b). Der Preis von passiven NFC Tags liegt zwischen wenigen Cent und einigen Euros, je nach Ausführung und Speicherplatz.

NFC Tags können zum Beispiel ähnlich der zuvor erklärten QR-Codes verwendet werden, denn sie können ebenfalls von Smartphones, welche NFC-fähig sind, eingelesen werden. Im Gegensatz zu QR-Codes funktioniert das Scannen jedoch schneller und zudem muss der NFC-Tag auch nicht außen sichtbar an Produkten angebracht werden. Ein weiterer Vorteil ist, dass NFC-Tags flexibler sind, weil die Daten auf einem Tag immer wieder geändert werden können und mehr Speicherplatz als auf QR-Codes vorhanden ist (Junnila, 2017). Die Nachteile wiederum sind gleich wie bei QR-Codes, mit NFC ist ein Echtzeittracking und eine genaue Positionsbestimmung ohne weitere Technologien nicht umsetzbar.

2.2.3 RFID – Radio-Frequency-Identification

Bei RFID gibt es passive und aktive Tags, wobei die aktiven Tags wie der Name schon erraten lässt, eine Spannungsversorgung benötigen. Der Unterschied spiegelt sich in Preis und Größe wieder, aktive Tags sind deutlich teurer und größer als passive RFID-Tags.

Passive Tags funktionieren ähnlich den NFC Tags, sie werden mit Hilfe eines RFID-Lesegerätes ausgelesen, wobei die maximale Distanz, im Gegensatz zu NFC, bis zu 10 Meter betragen kann. Die Vor- und Nachteile sind aber mit denen von NFC vergleichbar. Wobei zu erwähnen ist, dass die Lesegeräte für RFID Tags wesentlich teurer sind und ein Smartphone dafür nicht verwendet werden kann.

Im Gegensatz dazu senden aktive RFID-Tags in regelmäßigen Abständen ein Signal aus, welches wiederum von Lesegeräten empfangen werden kann. Die Reichweite von aktiven Tags liegt bei mehreren hundert Metern, jedoch sollte diese auf unter 100 Meter begrenzt werden, da es ansonsten sehr auf Kosten der Batterielebensdauer geht.

Diese Technologie hat zwar seine Vor- und auch seine Nachteile, ist aber für diese Diplomarbeit nicht relevant, da es keine einfache Möglichkeit gibt die einzelnen Tags ohne zusätzliche Technologie oder Hardware mit einem Smartphone zu koppeln.

2.2.4 UWB – Ultra-Wide-Band

Ultra-Wide-Band oder auf Deutsch die Ultra-Breitband-Technologie setzt wie der Name schon sagt auf eine sehr große Bandbreite. Die UWB-Geräte haben dabei eine relative Bandbreite von mehr als 20% oder eine absolute Bandbreite von mehr als 500MHz. Die große Bandbreite erhöht die Zuverlässigkeit, da das Signal verschiedene Frequenzkomponenten enthält, dass wiederum die Wahrscheinlichkeit erhöht, dass mindestens einige von ihnen Hindernisse umgehen können. Zusätzlich wird durch eine große absolute Bandbreite die Entfernungsgenauigkeit verbessert. Darüber hinaus verringert das Verbreiten von Information über eine sehr große Bandbreite die spektrale Leistungsdichte, wodurch die Interferenz zu anderen Systemen reduziert wird (Gezici u. a., 2005, S. 70).

Zu den Vorteilen dieser Technologie zählen die hohe Genauigkeit, niedrigere Latenzzeiten und es gibt nahezu keine Störungen. Im Gegensatz dazu ist UWB aber sehr kostenintensiv und die Batterielebensdauer eines UWB-Tags ist kürzer als zum Beispiel die von Bluetooth LE Beacons. Weiters wird, wie auch bei einigen Technologien zuvor, zusätzliche Hardware benötigt um mit den UWB-Tags kommunizieren zu können. Die Verwendung eines Smartphones, wie bei Bluetooth Beacons, ist nicht möglich, weshalb diese Technologie auch nicht für den geplanten Prototyp verwendet werden kann.

2.2.5 WLAN – Wireless Local Area Network

Bei der Verwendung von WLAN gäbe es den großen Vorteil, dass in den meisten Gebäuden bereits eine WLAN Infrastruktur vorhanden ist. Ob diese jedoch auch gut verwendet werden kann, hängt vom jeweiligen Anwendungsfall ab. Ein weiterer Vorteil ist, dass man herkömmliche Smartphones ohne weitere Hardware verwenden kann. Für ein Indoor-Positionsbestimmungssystem können jedoch nur Android Smartphones verwendet werden, für iOS-Geräte ist eine Umsetzung schwierig. Apple stellt die notwendigen Programmierschnittstellen nicht öffentlich zur Verfügung, wodurch es nur schwer möglich ist, Apps für die Indoor-Navigation in Verbindung mit WLAN zu entwickeln (Luo u. a., 2017, S. 1).

Um Assets mit Hilfe von WLAN tracken zu können, benötigt man sogenannte WIFI-Tags. Die Firma Accuware bietet solche Tags mit einer Batterielebensdauer von ungefähr 2 Monaten an. Zu welchem Preis diese erhältlich sind, ist leider nicht bekannt. Weitere erwerblich erhältliche WIFI-Tags konnten im Rahmen der Recherche leider nicht gefunden werden, daher sind diese Daten mit Vorsicht zu genießen. Ein weiterer Nachteil, zusätzlich zur doch eher geringen Batterielebensdauer, kommt noch die eingeschränkte Genauigkeit die laut

einigen Recherchen zwischen fünf und fünfzig Metern betragen kann. Die Genauigkeit kann durch Veränderungen in der Umgebung (größere Anzahl an Personen, geschlossene oder offene Türen, Veränderung des Inventars) stark beeinflusst werden (infsoft GmbH, o. J.; Youn u. a., 2007).

2.2.6 Bluetooth

Bei der Verwendung von Bluetooth Beacons, wird wie bei WLAN auch der RSSI-Wert zur Positionsbestimmung herangezogen. Zusätzlich zu den Beacons werden hier aber auch Bluetooth-Gateways benötigt, welche die Signale empfangen und auswerten, oder an einen Server weiterleiten. Das heißt, bei der Errichtung eines Asset-Tracking Systems müssen zusätzliche Kosten in die Infrastruktur investiert werden, da normalerweise keine bestehenden Bluetooth Gateways verfügbar sind. Diese Kosten könnten jedoch durch die günstigeren Beacons wieder eingespart werden. Aktuelle Beacons, wie die von Kontakt.io, sind je nach Ausstattung und Aufbau zu einem Preis zwischen 20€ und 40€ erhältlich (Stand: Mai 2018).

Ein weiterer Vorteil der Beacons ist die lange Lebensdauer aufgrund des geringen Energieverbrauches. Je nach Ausführung und Signalleistung können die Beacons bis zu mehreren Jahren verwendet werden. Kontakt.io verspricht beim aktuellen Beacon Pro mit Bluetooth 5 und Standardeinstellungen eine Batterielebensdauer von mehr als 5 Jahren.

Wie bei der Verwendung von WLAN wird auch bei Bluetooth keine weitere Hardware benötigt, wenn man diese mit Smartphones verbinden möchte. Im Gegensatz dazu, können aber jegliche Betriebssysteme (auch iOS) verwendet werden. Die Genauigkeit wird von den meisten Anbietern auf ungefähr einen Meter eingegrenzt. Diese wird aber auch durch Veränderungen in der Umgebung und die Verwendung von anderen Funktechnologien (z.B.: WLAN) beeinflusst. Ein weiterer erwähnter Nachteil ist die eher geringere Reichweite, die aktuell bei der Verwendung von Bluetooth 4 Beacons bei maximal 50 Metern liegt.

Hier kommen die neuen Funktionen von Bluetooth 5 zum Einsatz, welche im Rahmen dieser Diplomarbeit in ein Asset Tracking System integriert werden sollen.

2.2.7 Zwischenfazit Technologien

In diesem Kapitel wurde nun auf die Vor- und Nachteile einiger Technologien eingegangen, welche für Asset Tracking und/oder auch für Indoor Navigation verwendet werden können. Bei Bluetooth und WLAN ist die Einbindung von

aktuellen Smartphones ohne zusätzliche Hardware ohne Probleme möglich. Zusätzlich unterstützt Bluetooth in Bezug auf Indoor-Navigation auch iOS Geräte, welche in der aktuellen Zeit natürlich auch beachtet werden müssen. Da das System auch für Indoor-Navigation, bei der zumeist Smartphones als Endgeräte zum Einsatz kommen, verwendet werden können soll, wurde der Autor in der Annahme, dass Bluetooth die richtige Technologie ist, bestärkt.

Einige Nachteile von Bluetooth 4, wie die eher geringe Reichweite und der Einfluss von anderen Funktechnologien, sollen laut Recherche durch die Verwendung von Bluetooth 5 vermindert werden. Mehr dazu in den Kapiteln 2.5.3 und 2.5.5.

2.3 Aktuelle Forschungsprojekte bzw. kommerzielle Lösungen

Zu den im vorigen Kapitel genannten Technologien betreffend Asset-Tracking und Indoor Navigation gibt es natürlich aktuelle Forschungsprojekte und auch bereits einige kommerzielle Lösungen. Einige davon sind, mit den Unterschieden sowie dem Bezug zur Diplomarbeit, in diesem Kapitel angeführt.

2.3.1 Diplomarbeit – Indoor Positionierungssystem

Fahrngruber (2016) beschreibt in seiner Diplomarbeit mit dem Titel „Echtzeit Indoor Positionierungssystem mittels WPAN“ verschiedene Systeme und Technologien, welche für die Indoor Positionierung und auch die Indoor Navigation genutzt werden können. Dazu zählen unter anderem auch die bereits erwähnten Technologien UWB, Bluetooth, WLAN und RFID. Zusätzlich hat er aber auch noch Systeme mit Infrarot, Kameras, FM Radio, Schall und einigen anderen Technologien gegenübergestellt.

Im praktischen Teil der Diplomarbeit wurde ein Prototyp mit dem DWM1000 Modul von Decawave entwickelt, welcher zeigt, dass ein Echtzeit Indoor Positionierungssystem mit diesem Modul möglich ist. Das DWM1000 Modul basiert auf einem DW1000-Chip, bei dem es sich wiederum um einen Ultra Wide Band konformen Chip handelt. Zum Zeitpunkt der Diplomarbeit gab es noch keine Endprodukte mit diesem Chip am Markt. Pozyx (siehe nächstes Kapitel) hat aber genau diesen Chip bei seinem UWB System eingesetzt.

2.3.2 Pozyx – UWB (Kommerzielles System)

Pozyx[1] bieten ein UWB-System an, welches laut der Website eine zentimetergenaue Lokalisierung von Gegenständen ermöglichen soll. Grundsätzlich besteht das System aus Pozyx-Tags und Pozyx-Anchors. Die Anchors kann man mit dem Satelliten bei GPS vergleichen, sie sind an einer fixen Position montiert und empfangen die Signale der Tags. Die Tags sind, zum Beispiel im Falle eines Asset Tracking Systems, an einem Asset befestigt. Für die Lokalisierung eines Tags im 3D-Raum werden zumindest 4 Anchors benötigt.

Das Mindestpaket (Ready to Localize), bestehend aus einem Pozyx-Tag und 4 Pozyx-Anchors, um das System testen zu können, kostet aktuell € 599,- (Stand: Mai 2018). Ein Vergleich mit diesem System und den im Laufe dieser Arbeit entwickelten System wäre wünschenswert, aber aufgrund des hohen Anschaffungspreises leider im Rahmen dieser Diplomarbeit nicht umsetzbar.

2.3.3 infsoft – smart connected locations

Die Firma infsoft aus Deutschland bietet unter anderem Systeme für Indoor Navigation, Indoor Positionsbestimmung, Indoor Analytics und Indoor Tracking an. Sie verwenden dazu die Technologien WLAN, Bluetooth, UWB, RFID und Kamerasysteme. Auf der Hardwareseite verwenden sie, die von ihnen entwickelten, infsoft Locator Nodes und infsoft Locator Tags. Beide Komponenten sind modular aufgebaut und können mit den verschiedensten Bestandteilen, genau auf den Kundenwunsch abgestimmt, ausgestattet werden.

Für ein Asset Tracking System nutzen sie unter anderem aber auch „normale“ Bluetooth LE Beacons, wie zum Beispiel Beacons von kontakt.io (siehe 2.6.2.1). Sie unterscheiden dabei zwischen clientseitiger und serverseitiger Positionsbestimmung. Für die clientseitige Positionsbestimmung werden nur die Bluetooth Beacons und ein Smartphone mit der entsprechenden App benötigt. Die Positionsdaten werden dann direkt am Smartphone angezeigt. Bei der serverseitigen Positionsbestimmung werden die Signale der Beacons von, den zuvor erwähnten, Tags oder Nodes empfangen und die Daten an einen Server weitergeleitet, welcher diese dann entsprechend weiterverarbeitet.

Die Systeme von infsoft sind sehr vielseitig und umfangreich, deshalb verweist der Autor für weitere Informationen auf die Webseite https://www.infsoft.de/.

[1] Mehr Infos zu Pozyx: https://www.pozyx.io/

2.3.4 WLAN-basiertes Echtzeit-Tracking-System im Gesundheitsbereich

Youn u. a. (2007) beschreiben in ihrem Paper ein Echtzeit-Tracking-System von Gegenständen, wie zum Beispiel Infusionspumpen oder Rollstühlen, innerhalb einer Krankenhausumgebung. In ihrem Pilottest haben sie dafür die vier WLAN Access Points verwendet, welche bereits innerhalb der Testumgebung vorhanden waren. Zu Beginn wurden RSSI-Messungen an 430 vorher definierten Punkten durchgeführt. Dabei wurde an jedem Ort die Signalstärke von jedem der vier Access Points gemessen. Die gemessenen Werte wurden jeweils in Verbindung mit der Position im Raum in eine Datenbank eingetragen. Durch den Vergleich der RSSI-Werte eines WLAN-Tags mit den Einträgen in der Datenbank konnte die Position im Durchschnitt auf einen Meter genau bestimmt werden. Youn u. a. (2007) haben dann auch noch die Auswirkungen untersucht, wenn die Anzahl der Access Points oder deren Anordnung im Raum geändert wird. Je mehr Access Points verwendet wurden, umso geringer war die Ungenauigkeit bei der Positionsbestimmung. Bei dem Versuch stellten sie auch fest, dass mehr als 10 Access Points nicht sinnvoll sind, da sich die Ungenauigkeit nicht mehr signifikant änderte. Bei der Anordnung im Raum kamen sie zum Ergebnis, dass diese für ein gut funktionierendes System essentiell ist. Werden die Access Points an Positionen nah beieinander angebracht, steigt die durchschnittliche Fehlerdistanz erheblich an. Bringt man die Access Points hingegen gut verteilt im Raum an, funktioniert die Positionsbestimmung deutlich besser.

2.3.5 BeWhere – Knowing Counts

Die Firma BeWhere[2] bietet unter anderem BLE Beacons und BLE/WIFI Gateways für das Asset Tracking von allen möglichen Gegenständen (Werkzeuge, Paletten, Müllcontainer, Ausstattung eines Rettungsautos, ...) an. Weiters gibt es auch bereits Überlegungen für Beacons die mit NB-IoT oder LTE-M arbeiten, diese sind jedoch noch nicht bestellbar.

Der Beacon BTB-04 verwendet, laut dem technischen Datenblatt, den Bluetooth 4 Standard, soll aber trotzdem eine maximale Reichweite von bis zu 250 Metern erreichen. Zusätzlich verfügt er über einen 75dB-Buzzer, einer High-Intensity LED (Light Emitting Diode - Leuchtdiode) und drei Sensoren (Temperatur, Licht, Beschleunigung). Das Starterkit bestehend aus 8 Beacons und 2 Gateways kostet 499$, aufgrund des Preises ist ein Vergleich mit dem Prototyp, der im Rahmen dieser Diplomarbeit entwickelt wird, leider nicht möglich.

[2] http://bewhere.com/

2.3.6 Zwischenfazit

Im Laufe der Recherche von kommerziellen Lösungen oder aktueller Forschungsprojekte, wurde kein Produkt gefunden, welches dem im Rahmen dieser Diplomarbeit beschriebenen Prototyp entspricht. Die Verwendung von Bluetooth Beacons für das Tracken von Assets oder Personen, wird bereits in einigen Projekten genutzt. Jedoch gibt es keinen Beacon welcher Bluetooth 5 verwendet und zusätzlich über eine gute Signaleinrichtung verfügt. Diese beiden Funktionen gemeinsam, sollen es ermöglichen einen bestimmten Gegenstand noch schneller und aus größerer Entfernung (in Vergleich zu Bluetooth 4) zu finden.

2.4 Bluetooth Low Energy

2.4.1 Allgemeines zu Bluetooth

Mitte der 90er Jahre arbeiteten die Mobilfunkhersteller Ericsson und Nokia an einem Projekt, dessen Namen auf den Wikingerkönig Harald Gormsson auch bekannt als König Blauzahn zurückgeht. Aus Blauzahn entstand der Begriff Bluetooth. Ziel des Projektes war es eine Technik für einen Kurzstreckenfunk zwischen kleiner Geräte zu entwickeln („Bluetooth 1.0/1.1/1.2 (IEEE 802.15)“, o. J.).

Im Jahr 1998 gründeten 5 Firmen (Ericsson, IBM, Intel, Nokia und Toshiba) die Bluetooth Special Interest Group (SIG). Aktuell gehören mehr als 30000 Unternehmen dieser Interessensgemeinschaft, welche hinter der Bluetooth-Technologie steht, an („About Us | Bluetooth Technology Website“, o. J.). Der erste Bluetooth-Standard (v1.0) wurde im Jahr 1999 von der SIG verabschiedet. Zwischen der ersten und der aktuellen Version, liegen einige wichtige Updates und Zwischenversionen. Eine der größten Änderungen kam mit der Version 4.0, bei der es das erste Mal auch eine Bluetooth Low Energy (LE) Variante gab.

Wie Abbildung 3 zeigt, ist der Protokoll-Stack eines BLE Gerätes in drei Teile unterteilt: Controller, Host und Apps. Jeder dieser Teile ist wiederum in mehrere Schichten aufgeteilt, welche die erforderliche Funktionalität bereitstellen. Die unterste Schicht „Physical Layer“ wird in der neuen Bluetooth-Version 5 um zwei zusätzliche Varianten erweitert. Mehr dazu in Kapitel 2.5.

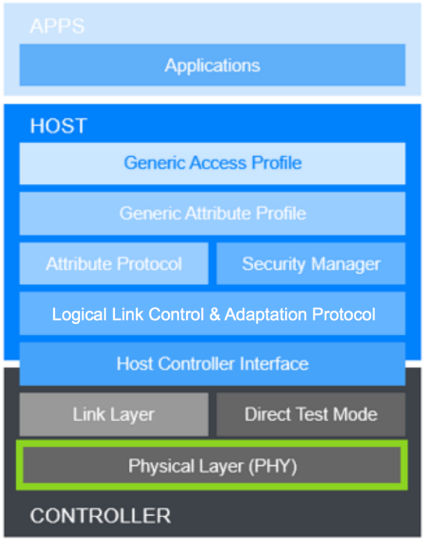

Abbildung 3: Bluetooth Low Energy Protokoll Stack (Woolley, 2017b)

Um den Verbindungsaufbau und das Advertising Konzept von Bluetooth etwas besser zu verstehen, werden die beiden Schichten „Generic Access Profile“ und „Generic Attribute Profile“ näher beschrieben. Die anderen Schichten sind für die weitere Bearbeitung der Diplomarbeit nicht notwendig und werden daher nicht weiter behandelt.

2.4.2 GAP – Generic Access Profile

Das Generic Access Profile (GAP) ist ein grundlegendes Bluetooth Profil und ist unter anderem zuständig für die Geräteerkennung, die Verbindungen und das Advertising. Im GAP wird definiert wie zwei Geräte miteinander interagieren können oder auch nicht.

GAP definiert viere verschiedene Rollen für Geräte: Observer, Broadcaster, Central und Peripheral (vgl. Kapitel 2.4.3). Die beiden wichtigsten Konzepte davon sind Central (Zentralgerät) und Peripheral (Peripheriegerät). Peripheriegeräte sind kleine, ressourcenarme Geräte mit geringem Stromverbrauch, die mit einem viel leistungsfähigeren Zentralgerät verbunden werden können. Peripheriegeräte sind zum Beispiel ein Brustgurt der die Herzfrequenz misst oder ein Bluetooth-Temperatursensor. Zentralgeräte hingegen sind in der Regel Smartphones, Tablets oder PCs, welche sich mit Peripheriegeräten verbinden können und über weitaus mehr Rechenleistung und Speicherkapazität verfügen (Adafruit, 2015a).

Damit ein Bluetooth-Gerät gefunden werden kann muss es sichtbar sein, dafür ist es notwendig, dass es sogenannte Advertising Pakete aussendet. In GAP sind

dafür der Advertising Payload und der Scan-Response Payload vorgesehen. Beide Payloads (= Nutzdaten) sind identisch und können bis zu 31 Bytes an Daten enthalten. Bei der Verwendung von Bluetooth 5 wird diese Kapazität um das Achtfache erhöht (vgl. 2.5.4). Der Advertising Payload ist obligatorisch und muss regelmäßig von einem Gerät ausgesendet werden um für andere Geräte (Centrals) sichtbar zu sein. Der Scan-Response Payload hingegen ist optional und kann von einem Central angefordert werden. Dadurch ist es möglich eine größere Datenmenge ohne Verbindungsaufbau zu versenden (Adafruit, 2015a).

Für eine verbindungslose Kommunikation (nur in eine Richtung) reicht das Generic Access Profile aus. Um aber eine verbindungsorientierte Verbindung (in beide Richtungen und mit größeren Datenmengen) zu ermöglichen wird das Generic Attribute Profile (= GATT) benötigt (siehe 2.4.4).

2.4.3 Vier verschiedene Rollen von Geräten - GAP

Bluetooth LE Geräte können auf zwei verschiedene Arten mit der Umgebung kommunizieren, entweder mit oder ohne einer bestehenden Verbindung. Für jede Art gibt es zwei verschiedene Rollen.

2.4.3.1 Verbindungslos – Broadcaster und Observer

Bei der verbindungslosen Übertragung werden Nachrichten an alle Geräte in Reichweite ausgesendet. Jedes kompatible Gerät innerhalb des Empfangsbereiches kann die Nachrichten empfangen und weiterverarbeiten. Es ist somit nur möglich Daten in eine Richtung an jedes beliebige Gerät zu senden, welches in der Lage ist, die übertragenen Daten zu empfangen.

Wie man in Abbildung 4 erkennen kann, gibt es zwei verschiedene Rollen, welche als Broadcaster und Observer bezeichnet werden. Zwischen diesen beiden Rollen wird keine Verbindung aufgebaut.

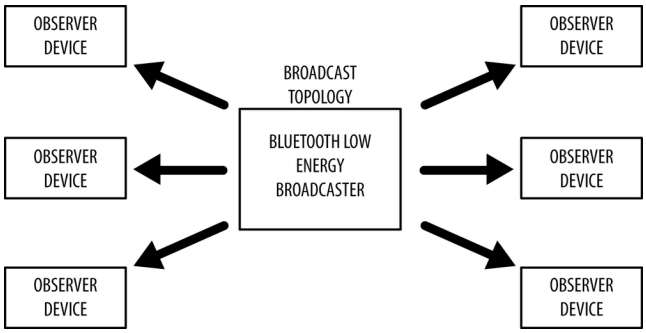

Abbildung 4: Verbindungslose Übertragung – Bluetooth (Davidson, 2014)

Broadcaster

Geräte die als Broadcaster arbeiten, senden in regelmäßigen Abständen Nachrichten aus. Diese Nachrichten werden als „Non-connectable“-Advertising Pakete bezeichnet und können bei Bluetooth 4 bis zu 31 Bytes an Daten enthalten. Im Falle von Bluetooth 5 wird die Kapazität dieser Pakete, wie bereits erwähnt, um das Achtfache erhöht (vgl. 2.5.4).

Observer

Observer sind Geräte die als Empfänger arbeiten, das heißt sie empfangen die Pakete von den Broadcastern und verarbeiten diese dann weiter. Sie senden aber selbst keine Nachrichten aus.

(Davidson, 2014)

2.4.3.2 Verbindungsorientiert – Central und Peripheral

Wenn Daten in beide Richtungen oder eine größere Datenmenge übertragen werden muss, ist eine Verbindung zwischen den Geräten notwendig. Eine Verbindung ist ein permanenter und periodischer Austausch von Datenpaketen zwischen zwei Geräten. Die Daten werden grundsätzlich nur von den beiden verbundenen Geräten gesendet und empfangen, es sei denn die Verbindung wird abgehört.

Wie man in Abbildung 5 erkennen kann, gibt es auch hier zwei verschiedene Rollen, welche als Central und Peripheral bezeichnet werden.

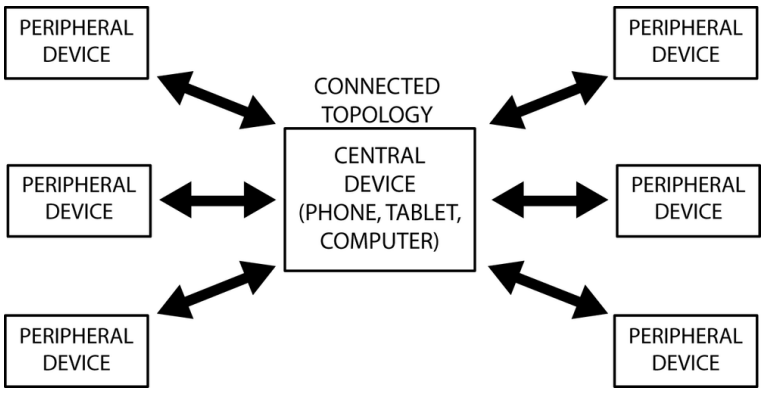

Abbildung 5: Verbindungsorientiere Übertragung – Bluetooth (Davidson, 2014)

Peripheral (Slave)

Pheriperal-Geräte senden, wie Broadcaster, in regelmäßigen Abständen Nachrichten aus. Im Gegensatz zu Broadcastern handelt es sich dabei aber um „Connectable“-Advertising Pakete und diese Geräte akzeptieren eingehende

Verbindungen. Sobald eine aktive Verbindung aufgebaut ist, arbeitet das Pheriperal als Slave und tauscht regelmäßig Daten mit dem verbundenen Central aus. Solange die Verbindung besteht, werden keine Advertising-Pakete mehr ausgesendet. Das bedeutet es kann von anderen Centrals nicht mehr gefunden werden.

Central (Master)

Ein Gerät in der Rolle Central, zum Beispiel ein Smartphone oder ein PC, sucht laufend nach „Connectable"-Advertising Paketen und initiiert nach Bedarf eine Verbindung mit einem Pheriperal. Ein Central Gerät ist immer Master, das heißt sobald eine Verbindung hergestellt ist, steuert es den zeitlichen Ablauf des Datenaustauschs.

(Davidson, 2014)

2.4.4 GATT – Generic Attribute Profile

GATT definiert die Art und Weise, wie zwei Bluetooth LE Geräte unter der Verwendung von Konzepten, die als Service und Charakteristiken bezeichnet werden, miteinander kommunizieren und Daten austauschen. GATT kommt erst dann ins Spiel, wenn eine dedizierte Verbindung zwischen zwei Geräten mit Hilfe von GAP hergestellt wurde. In den folgenden beiden Abschnitten sind die Begriffe Service und Charakteristik kurz erklärt.

Services

Services werden verwendet, um Daten in logische Objekte aufzuteilen, und enthalten bestimmte Datenblöcke, die als Charakteristiken bezeichnet werden. Ein Service kann ein oder mehrere Charakteristiken enthalten. Die einzelnen Services unterscheiden sich voneinander durch eine eindeutige UUID. Diese UUID kann entweder 16 Bit oder 128 Bit lang sein. 16 Bit UUIDs sind für Services reserviert die von der Bluetooth SIG angenommen worden sind. Für benutzerdefinierte und selbst erstellte Services werden 128 Bit UUIDs verwendet.

Charakteristiken

Die unterste Ebene einer GATT-Transaktion ist die Charakteristik. Diese enthält einen einzelnen Wert, wobei es sich dabei auch um ein Array von miteinander verwandten Werten handeln kann. Wie zum Beispiel die X-, Y- und Z-Werte eines 3-Achsen-Beschleunigungssensors. Ähnlich wie bei Services unterscheiden sich die einzelnen Charakteristiken wieder durch eine 16 Bit oder 128 Bit UUID.

Die Werte welche in einer Charakteristik gespeichert werden können ausgelesen und auch geändert werden. Jedoch ist das nicht für jede Charakteristik möglich, dafür gibt es verschiedene Optionen die für eine Charakteristik konfiguriert werden können. Die wichtigsten Optionen sind in Tabelle 1 aufgezählt. Es gibt noch zusätzliche Optionen, welche aber für diese Arbeit nicht relevant sind.

Tabelle 1: Mögliche Optionen einer GATT-Charakteristik

Bezeichnung	Beschreibung
WRITE	Wenn diese Option aktiviert ist, können GATT-Clients einen Wert an diese Charakteristik senden. Der GATT-Server sendet eine Bestätigung zurück, wenn alles geklappt hat.
WRITE WITHOUT RESPONSE	Diese Option ist ähnlich der „WRITE“-Option, jedoch bekommt der GATT-Client keine Bestätigung ob alles funktioniert hat.
READ	Wenn diese Option aktiviert ist, können GATT-Clients den Wert der Charakteristik auslesen.
NOTIFY	Wenn diese Option aktiviert ist, kann ein GATT-Client die Charakteristik abonnieren und erhält dann laufend mögliche Werteänderungen der Charakteristik.

GATT definiert zwei Rollen, welche die beiden miteinander verbundenen Geräte übernehmen können: Server und Client.

GATT-Server

Ein Peripheriegerät, wie zum Beispiel ein Brustgurt, arbeitet meist als GATT-Server und stellt verschiedene Services und Charakteristiken zur Verfügung.

GATT-Client

Der GATT-Client hingegen kann auf die Daten welche in den Charakteristiken gespeichert sind zugreifen. Alle Transaktionen werden vom GATT-Client gestartet, welcher dann eine Antwort vom GATT-Server empfängt.

(Adafruit, 2015b; Davidson, 2014)

2.5 Bluetooth 5.0

Bluetooth 5 soll nun laut SIG einige wesentliche Verbesserungen im Vergleich zu Bluetooth 4.2 mit sich bringen. Unter anderem soll sich die Geschwindigkeit verdoppeln und die Reichweite vervierfachen. Alle Details und weitere Verbesserungen werden im folgenden Kapitel beschrieben.

2.5.1 Grundlegendes

Auch bei Bluetooth 5 gibt es zwei verschiedene Versionen: Bluetooth Low Energy und Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR). Da Bluetooth Beacons aufgrund des geringeren Energieverbrauchs die Low Energy Variante benutzen, wird in den nächsten Kapiteln nur dieser näher behandelt. Die Änderungen von Bluetooth 5.0 haben aber auch einen Einfluss auf die BR/EDR Variante, welche aber hier nicht näher beschrieben werden.

Wie bereits in Punkt 2.4.1 erwähnt, besitzt Bluetooth LE, wie auch andere Protokolle einen Physical Layer. Diese unterste Schicht des Stacks wird bei Bluetooth als PHY bezeichnet. Bluetooth 5 fügt der Spezifikation, die in Bluetooth 4 verwendet wird, zwei neue PHY-Varianten hinzu. Jede PHY-Variante hat ihre eigenen besonderen Merkmale und wurde mit spezifischen Zielen entwickelt. Die drei PHYs wurden LE 1M, LE 2M und LE Coded benannt, damit sie in Spezifikationen leichter referenziert werden können.

LE 1M ist der Physical Layer, der in Bluetooth 4 verwendet wird. Er verwendet als Modulationsart Gaussian Frequency Shift Keying (GFSK) und hat eine Symbolrate von 1 Megasymbol pro Sekunde (MS/s).

2.5.2 Höhere Übertragungsgeschwindigkeit

Bluetooth SIG gibt an, dass sich die Übertragungsgeschwindigkeit von 1MBit/s auf 2MBit/s verdoppelt, ohne das sich auch dabei der Energieverbrauch erhöht (2017, S. 6). Jedoch trifft das nur zu, wenn der Physical Layer LE 2M verwendet wird. Dabei wird die Symbolrate von 1MS/s auf 2MS/s erhöht, wodurch ein höherer Datendurchsatz als bei Bluetooth 4 möglich ist. Laut Martin Woolley wird dadurch aber die Reichweite um einen Faktor von ungefähr 0,2 verringert (2017a, S. 15).

2.5.3 Größere Reichweite

Laut Angaben der Bluetooth SIG soll Bluetooth 5 die Reichweite erheblich vergrößern. Es soll dadurch erstmals möglich sein, ein ganzes Einfamilienhaus

mit Bluetooth zu durchdringen, ohne das die Verbindung abbricht (Bluetooth SIG, Inc, 2017, S. 6). Um eine größerer Reichweite zu erzielen wird der Physical Layer LE Coded verwendet, welcher wiederum mit zwei verschiedenen Codierungsschemas (S=2 oder S=8) verwendet werden kann. Der Unterschied liegt dabei bei der möglichen Reichweite und der möglichen Datenrate. LE Coded (S=2) verwendet eine Datenrate von 500kbit/s und erhöht die Reichweite um etwa das doppelte. LE Coded (S=8) hingegen hat nur eine Datenrate von 125 kbit/s, erhöht die Reichweite aber um einen Faktor von 4. Da heißt die vierfache Reichweite von Bluetooth 4 kann nur erreicht werden, wenn der LE Coded (S=8) PHY verwendet wird und eine Datenrate von 125kbit/s für den Anwendungsfall ausreichend ist.

Die ungefähre Bedeutung dieser Charakteristika in Metern wird aber nicht angegeben. Laut einer Recherche soll die maximal realistische Reichweite auf 200 Meter (vgl. Bluetooth 4: ca. 50 Meter) vergrößert werden, wobei das nur bei direkter Sichtverbindung möglich ist. In geschlossenen Räumen wird die Reichweite mit realistischen 40 Metern (vgl. Bluetooth 4: ca. 10 Meter) angegeben (Elektronik-Kompendium.de, o. J.-a). Laut Bluetooth SIG ist es aber auch mit Bluetooth 4 bereits möglich mehrere hundert Meter zu überbrücken, das hängt jedoch von vielen Faktoren ab und geht auf Kosten der Datenrate und dem Energieverbrauch. Mit Bluetooth 5.0 hat Espen Wium bereits eine Übertragungsreichweite von 1,6 km erreicht (Wium & Kervel, 2017). Wie groß die Reichweite des neuen Funkstandards ist, hängt somit also von einigen Faktoren, wie der Sendeleistung in Verbindung mit dem Energieverbrauch, der Datenrate und gerätespezifischen Daten ab. Jedoch verspricht Bluetooth 5.0 die vierfache Reichweite ohne die erforderliche Sendeleistung und somit den Energiebedarf zu erhöhen. Diese Eigenschaft bringt für Bluetooth Beacons einen großen Vorteil.

2.5.4 Höhere Übertragungskapazität

Laut Bluetooth SIG steigt die Nachrichtenkapazität von Broadcast- und Bluetooth-Advertising-Paketen um 800%. Somit sollen mehr Daten gesendet und empfangen werden können (Bluetooth SIG, Inc, 2017, S. 6). Bluetooth 4 hat bislang die Größe der Datenpakete auf 37 Byte begrenzt, wobei dabei 6 Byte für den Paket-Header vorgesehen waren. Mit dem neuen Standard wird die Grenze auf 255 Byte pro Datenpaket angehoben, das entspricht in etwa der achtfachen Übertragungskapazität. Das bedeutet, dass Beacons, welche im Grunde nur Broadcast-Pakete aussenden, jetzt mehr Daten schicken können.

Zusätzlich wurden die Advertising-Pakete bislang nur auf bis zu drei dezidierten Kanälen mit den Nummern 37, 38 und 39 übertragen. Insgesamt verfügt

Bluetooth jedoch über 40 Funkkanäle mit einer Bandbreite von je 2MHz. Bei Bluetooth 5 stehen nun auch die anderen 37 Kanäle (0 bis 36) zur Verfügung um Advertising-Pakete zu versenden. Das Advertising-Intervall wurde bei „non-connectable“-Paketen von 100ms auf 20ms reduziert. Das wiederum führt dazu, dass Beacons schneller erkannt werden und somit zeitnaher reagiert werden kann. Bluetooth 5 bietet somit die Grundlage für die Erstellung von Beacons der nächsten Generation, welche es ermöglichen, viel reichhaltigere und facettenreichere Daten zu senden als nur die ID oder eine URL. Zum Beispiel kann ein Beacon seine eindeutige Kennung, die Lufttemperatur, die Batterieladung und weitere Sensordaten auf einmal aussenden (Woolley, 2017a, S. 17–19).

2.5.5 Interoperabilität - Koexistenz

Bereits 96% aller ausgelieferten Tablets und Smartphones verfügen über Bluetooth. Daher gewinnt die Interoperabilität, also die Eigenschaft das mehrere Geräte und Technologien nebeneinander gut funktionieren, zunehmend an Bedeutung. Bei Bluetooth 5 wurden einige Änderungen vorgenommen, um Interferenzen mit anderen Geräte und den drahtlosen Technologien zu verbessern. Bluetooth verwendet das 2,4-GHz-ISM-Band, welches unmittelbar an die Bänder des Mobilfunknetzes angrenzt, welche unter anderem für LTE verwendet werden. Zwischen diesen beiden Systemen besteht ein Potential für Interferenzen. Bluetooth 5 führt ein System namens „Slot Availability Masks“ ein, mit dem Bluetooth die Verfügbarkeit seiner Zeitfenster anzeigen und optimal mit den benachbarten Mobilfunk-Bändern synchronisieren kann (Woolley, 2017a, S. 21).

2.5.6 Zusammenfassung – Bluetooth 5 Physical Layers

In der Tabelle 2 sind die verschiedenen physikalischen Layer von Bluetooth 5 mit der entsprechenden Datenrate und dem Reichweitenmultiplikator nochmals zusammengefasst.

Tabelle 2: Verschiedene Physical Layers von Bluetooth 5 Low Energy

PHY	***LE 1M***	***LE 2M***	***LE Coded (S=2)***	***LE Coded (S=8)***
Datenrate	1 Mbit/s	2 Mbit/s	500 kbit/s	125 kbit/s
Reichweiten Multiplikator	1	0,8	2	4

2.6 Bluetooth Beacons

2.6.1 Funktionsweiße und Standards

Bei einem Bluetooth Beacon handelt es sich grundsätzlich um einen kleinen Bluetooth-Sender, welcher in regelmäßigen Abständen Signale aussendet. Inhalte dieser Signale sind zumindest die eindeutige Kennung des Beacons und die Sendeleistung. Es können noch zusätzlich weitere Konfigurations- und Telemetriedaten gesendet werden. Diese Daten werden in einem bestimmten Format übertragen, welches in verschiedenen Protokollen festgelegt ist. Zu den bekanntesten zählen iBeacon von Apple und Eddystone von Google. Es ist aber auch durchaus möglich ein eigenes Protokoll für die Übertragung festzulegen (Fraunhofer ISST, o. J.). In den folgenden zwei Abschnitten werden die grundlegenden Bestandteile der beiden Protokolle beschrieben.

2.6.1.1 iBeacon Standard

Unter iBeacon versteht man den ersten entwickelten geschlossenen Standard für Bluetooth Beacons. Ein iBeacon versendet vier Teile an Information: UUID (= Unique Universal Identifier), Major ID, Minor ID und die Sendeleistung (TX Power).

- **UUID:** Bei der UUID handelt es sich um eine eindeutige Kennung, mit der ein Beacon zum Beispiel zu einem bestimmten Einkaufscenter zugeordnet werden kann.
- **Major ID und Minor ID:** Mit den beiden Werten Major und Minor, kann der Beacon genauer beschrieben werden. Zum Beispiel kann man dadurch festlegen das sich der Beacon in Geschäft Nummer 1 befindet und dort im ersten Gang. Mit allen drei Werten kann also ein Beacon eindeutig identifiziert werden. Es sollte somit nie zwei Beacons geben, welche überall die gleichen Werte senden .(Kontakt.io, o. J.)
- **Sendeleistung:** Die Sendeleistung beinhaltet einen Wert, der die Sendeleistung des Beacons in einem Abstand von einem Meter wiedergibt. Dieser Wert muss vom Hersteller eines Beacons konfiguriert werden (Blackstone, 2015).

2.6.1.2 Eddystone Standard

Eddystone ist ein von Google entwickeltes Bluetooth-Protokoll welches einige Arten von Paketen definiert, welche von Bluetooth Beacons ausgesendet werden können. Drei dieser Pakete sind unter anderem:

- **Eddystone-UID**
  Dieses Paket ist vergleichbar mit dem iBeacon Protokoll (UUID, Major & Minor), es enthält eine einzigartige statische Identifikationsnummer, bestehend aus den Teilen Namespace und Instanz.
- **Eddystone-URL**
  Ein Paket dieser Art, enthält eine komprimierte URL, also eine eindeutige Referenz auf eine Ressource im Web.
- **Eddystone-TLM**
  Im TLM Paket können verschiedenste Telemetriedaten übertragen werden. (Estimote, Inc., o. J.)

2.6.2 Funktionsübersicht aktueller Bluetooth 5 Beacons

In diesem Kapitel sind einige aktuelle Bluetooth 5 Beacons mit ihren Funktionen aufgezählt. Die folgenden Beacons sollen laut Hersteller bereits den neuen Bluetooth 5 Standard unterstützen und wurden deshalb als Vergleich ausgewählt.

2.6.2.1 Kontakt.io

Kontakt.io[3] ist ein Hersteller von Bluetooth Beacons und bietet zusätzliche Produkte und Dienstleistung an. Der Beacon Pro ist einer der ersten Bluetooth 5 fähigen Beacon am Markt. Deshalb werden hier kurz die wichtigsten Funktionen und technischen Daten zusammengefasst.

- Voller Support für iBeacon und Eddystone
- Sensoren: Beschleunigungsmesser und Umgebungslichtsensor
- Echtzeituhr
- LED Indikator
- NFC Kommunikation
- bis zu 80 Meter Reichweite
- Batterielebensdauer bis zu 60 Monate (2,5 Jahre)
- Akkukapazität: 3000mAh
- Verbauter Chip: Nordic Semiconductor nRF52832 (nur Bluetooth 5 Ready)
- Konfigurierbar: Sendeleistung, Sendeintervall und Sendezeiten
- Preis: 35 Dollar entspricht etwa 30,00 € (Stand: Juli 2018)

[3] Mehr Infos zu den Kontakt.io-Beacons: https://kontakt.io/

2.6.2.2 RuuviTag

Der RuuviTag[4] ist ein Open-Source Bluetooth Sensor Beacon der im Rahmen einer Kickstarter Kampagne mit mehr als 170.000 Dollar unterstützt wurde. Er soll laut Hersteller Website ebenfalls bereits Bluetooth 5 unterstützen, daher werden hier ebenfalls die wichtigsten technischen Daten und Funktionen kurz aufgezählt.

- Unterstützt iBeacon und Eddystone
- Sensoren: Beschleunigung, Luftfeuchtigkeit, Temperatur, Luftdruck,
- 2 LEDs und 2 Buttons
- NFC Kommunikation
- Keine genauen Angaben zur Reichweite
- Batterielebensdauer mehrere Jahre (abhängig von der Verwendung)
- Akkukapazität: 1000mAh
- Verbauter Chip: Nordic Semiconductor nRF52832 (nur Bluetooth 5 Ready)
- Preis: 23 Euro pro Stück (Stand: Juli 2018)

2.6.2.3 Estimote

Estimote[5] bietet wie Kontakt.io verschiedene Beacons sowie weitere Produkte und Dienstleistungen im Bereich Tracking und Lokalisierung an. Der Bluetooth Standard denn die Estimote Beacons verwenden, konnte vorerst nicht herausgefunden werden. Aufgrund der Tatsache, dass die Beacons Mesh-fähig sind, wurde angenommen, dass es sich um den neuen Bluetooth 5.0 Standard handelt. Nach einer ausführlicheren Recherche wurde jedoch festgestellt, dass bei den für den Bereich Asset Tracking relevante Beacon, genannt „Location Beacon“, nur der Chip nRF51822 von Nordic Semiconductor verbaut ist. Dieser verwendet noch den alten Bluetooth 4.2 LE Standard. Aufgrund der angegebenen, doch sehr hohen, Reichweite, wurden die Estimote Beacons trotzdem zu einem Vergleich herangezogen. Die wichtigsten technischen Daten und Funktionen werden hier kurz aufgezählt.

- Voller Support für iBeacon und Eddystone
- Sensoren: Bewegungssensor, Temperatur, Drucksensor, Magnetometer, und Umgebungslichtsensor
- Echtzeituhr

[4] Mehr Infos zum RuvviTag: https://tag.ruuvi.com/

[5] Mehr Infos zu den Estimote Beacons: https://estimote.com/

- GPIO (General Purpose Input/Output)
- NFC Kommunikation
- bis zu 200 Meter Reichweite
- Batterielebensdauer bis zu 5 Jahren
- Akkukapazität: 4000mAh
- Verbauter Chip: Nordic Semiconductor nRF51822 (nur Bluetooth 4.2 LE)
- Preis: 33 Dollar entspricht etwa 28,50 € (Stand: Juli 2018)

2.7 Entwicklungsoptionen mobiler Applikationen in Bezug auf Bluetooth

Im Rahmen der Diplomarbeit wird eine mobile Applikation entwickelt, welche es ermöglicht den, ebenfalls im Rahmen dieser Diplomarbeit entwickelten, Prototypen zu konfigurieren und zu steuern. Eine wichtige Anforderung ist, dass die Applikation sowohl auf Android als auch auf iOS Geräten einwandfrei funktioniert.

Im Grunde gibt es 3 verschiedene Arten von mobilen Applikation:

- Webapplikationen
- Hybride Applikationen
- Native Applikationen

In den folgenden 3 Unterkapiteln werden die grundlegenden Funktionen sowie die Vor- und Nachteile der Entwicklungsoptionen für mobile Applikationen in Bezug auf die Bluetooth Verbindung aufgezeigt und am Ende eine Auswahl getroffen.

2.7.1 Webapplikationen

Webapplikationen werden im Webbrowser des jeweiligen Endgerätes ausgeführt und sind somit mehr oder weniger plattformunabhängig. Die Entwicklung ist ähnlich der einer mobilen Website, die daher meist verwendeten Programmiersprachen sind: HTML5, CSS und JavaScript (Delía, Galdamez, Corbalan, Pesado, & Thomas, 2017, S. 653). Es gibt aber auch einige Frameworks, die bei der Programmierung helfen und speziell für mobile Webapplikationen entwickelt wurden (z.B.: Sencha, jQuery Mobile, ...).

Durch verschiedene HTML5 - APIs ist es auch mit Webapplikationen möglich auf Gerätefunktionen zuzugreifen, dazu zählen unter anderem die Kamera, das

GPS-Modul oder die Lagesensoren eines Smartphones. Für den Zugriff auf das Bluetooth Modul steht die Web Bluetooth API zur Verfügung. In der Tabelle 3 werden die wichtigsten Gerätefunktionen, in Bezug auf die geplante mobile Applikation, und ihre Verwendbarkeit aufgezeigt. Die Daten der Tabelle stammen aus einem kurzen Test, mit Hilfe der Website https://whatwebcando.today/, welche alle HTML5 APIs anzeigt und ob diese mit dem aktuellen Browser und Gerät funktionieren. Der Test wurde jeweils auf einem iPhone X und einem Samsung Galaxy S5 Mini im Chrome Browser durchgeführt. Laut Statista wird der Chrome Browser auf mehr als 50 Prozent der mobilen Endgeräte verwendet, daher wurde dieser für den kurzen Test ausgewählt (StatCounter, 2018).

Tabelle 3: Zugriff auf Gerätefunktionen mittels HTML5 APIs im Chrome Browser - Android vs. iOS (Stand: 22.2.2017)

	Chrome auf Android (Samsung Galaxy S5 mini)	Chrome auf iOS (iPhone X)
Lokale Benachrichtigungen		
Push-Nachrichten		
Verlinkung auf Startscreen		
Offline Modus		
Kamera		
Standort Lokalisierung (GPS)		
Geräte Position (Drehung, ...)		
Bluetooth		
NFC		

Die wichtigste Funktion „Bluetooth“ steht leider nur auf Android Geräten zur Verfügung. Zusätzlich ist NFC weder auf Android noch auf iOS-Geräten verfügbar. Als einzige, für die mobile Applikation notwendigen Funktionen funktioniert nur die Standort Lokalisierung auf beiden Geräten. Auf die weiteren Gerätefunktionen sowie Vor- und Nachteile von Webapplikationen wird hier nicht weiter eingegangen, da der Zugriff auf das Bluetooth-Modul essentiell für die weitere Entwicklung wäre.

2.7.2 Hybride Applikationen

Für diese Entwicklungsoption gibt es viele verschiedensten Frameworks (z.B.: Ionic, React Native, Xamarin, ...). Alle haben den Vorteil, dass eine Applikation nur einmal in einer Programmiersprache programmiert werden muss und am Ende für mehrere Betriebssysteme bereitgestellt werden kann. Zwischen den Frameworks gibt es grundlegend nur wenige Unterschiede, zum Beispiel

unterstützt das Framework React Native nur die Betriebssysteme Android und iOS, Xamarin hingegen unterstützt zusätzlich auch noch das Betriebssystem Windows Phone. In Tabelle 4 werden verschiedene Frameworks miteinander verglichen.

Da es so viele verschiedene Frameworks gibt, wurde die Auswahl für den Vergleich auf vier Frameworks begrenzt. Der Autor hat bereits mit Ionic und testweise mit React Native gearbeitet, daher wurden diese Frameworks für den Vergleich ausgewählt. Zusätzlich wurden auch noch die Frameworks Xamarin und NativeScript ausgewählt, da diese wie die ersten beiden, sowohl Android als auch iOS unterstützen und den Zugriff auf Bluetooth ermöglichen.

Tabelle 4: Vergleich von Frameworks für die Entwicklung von Hybrid Apps

	Ionic	React Native	Xamarin	NativeScript
Programmiersprache	JavaScript, TypeScript	Javascript	C#	JavaScript, TypeScript
Android				
iOS				
Windows Phone				
Open Source	Ionic Pro möglich			
Native User Interface		eingeschränkt		
Zugriff auf Bluetooth				
Zugriff auf GPS				
Zugriff auf NFC				

Wie in der Tabelle 4 schön zu sehen, sind die Unterschiede zwischen den Frameworks in den relevanten Punkten nur sehr minimal. Daher ist die Auswahl eines Frameworks, mehr oder weniger Geschmackssache, wobei es aber immer auch auf die zu entwickelnde Applikation ankommt.

2.7.3 Native Applikationen

Die native Entwicklung ist die Aufwendigste aber die einzige die direkt auf alle Funktionen eines Smartphones zugreifen kann. Jede Applikation muss für jedes Betriebssystem eigens entwickelt werden, das bringt aber wiederum den Vorteil, dass die Applikationen normalerweise performanter sind als andere. Zumindest

ist es möglich mit einer nativen Entwicklung die größtmögliche Performance für eine Applikation zur Verfügung zu stellen.

Wie zu Beginn dieses Kapitels (siehe 2.7) bereits kurz erwähnt, soll die Applikation für Android und iOS entwickelt werden. Die dazu verwendeten Programmiersprachen sind Java und Swift. Da es sich um eine native Programmierung handelt, muss der Zugriff auf Gerätefunktionen nicht geklärt werden. Denn die native Programmierung ist die einzige Option, die einen optimalen Zugriff auf sämtliche Funktionen eines Smartphones ermöglicht.

2.7.4 Vergleich und Auswahl der Entwicklungsoptionen

In der Tabelle 5 werden die grundlegenden Punkte der drei Entwicklungsoptionen miteinander verglichen.

Tabelle 5: Vergleich von Web, Hybrider und Nativer Entwicklung von mobilen Applikationen

	Web	**Hybrid**	**Native**
Programmier-sprache	JavaScript, HTML5 und CSS3	JavaScript, HTML5 und CSS3	Swift, Java
Performance	Gering bis Mittel	Mittel	Schnell
UI/UX (User Interface / User Experience)	Designabhängig (auf allen Plattformen gleich)	Nativen Elementen nachempfunden (entsprechend der Plattform)	Nativ
Erhältlich in	Online (keine Installation)	App Stores	App Stores
Plattform-abhängigkeit	Keine	Gering	Hoch
Entwicklungszeit	Gering	Mittel	Hoch
Wartbarkeit	Leicht	Mittel	Aufwändig
Kosten	Gering	Mittel	Hoch
Zugriff auf Hardware	Sehr eingeschränkt	Leichte Einschränkungen (Plug-Ins)	Komplett
Verbindung	meist Online & beschränkt Offline	Online & Offline	Online & Offline
Zugriff auf Bluetooth	nur Android	Android und iOS	Android und iOS

Die grün hinterlegten Punkte in der Spalte Hybrid, entsprechen den wichtigsten Punkten, warum die Wahl auf eine Hybride Entwicklung gefallen sind. Eine Webapplikation fiel gleich zu Beginn weg, da es keine Möglichkeit gibt um auf das Bluetooth-Modul eines iOS Gerätes zuzugreifen. Da im Rahmen der Diplomarbeit nur ein Prototyp entwickelt werden soll und es nicht notwendig ist, dass die entwickelte Applikation extrem schnell ist, fiel die Entwicklung von zwei nativen Applikationen ebenfalls weg. Zusätzlich kommt der Autor eher aus der Webentwickler-Schiene und ist somit im Umgang mit JavaScript besser als in Java oder Swift.

Da es sich nun um eine hybride App handeln soll, muss auch noch die Auswahl eines Frameworks getroffen werden. Da es grundsätzlich keine großen Unterschiede zwischen den in Punkt 2.7.2 erwähnten Frameworks gibt, entschied sich der Autor für NativeScript. Kriterien für die Auswahl waren unter anderem die verwendete Programmiersprache JavaScript, das native User Interface und die Begebenheit das der Autor noch keine Applikation mit dem Framework umgesetzt hat.

3 Entwicklung des Prototyps

3.1 Geplantes System

Im Rahmen eines zeitgleich mit dieser Diplomarbeit gestarteten Forschungsprojekts wird ein Asset Tracking System für ein größeres Unternehmen in Niederösterreich umgesetzt. Dabei sollen Transportgestelle mit jeweils einem Beacon und einer Signaleinheit ausgestattet werden. Die Beacons sollen dann mit Hilfe von intelligenten Bluetooth-Gateways, welche im Rahmen des Forschungsprojektes entwickelt werden, getrackt und nachverfolgt werden können. Den Mitarbeiter des Unternehmens wird eine Mobile Applikation zur Verfügung gestellt, die es ermöglicht durch Eingabe einer Auftragsnummer die Signaleinheiten der entsprechenden Tragegestelle auszulösen oder den Standort der Tragegestelle auf einer Karte anzuzeigen. Damit soll eine gewisse Zeitersparnis für das sonst teilweise aufwendige Suchen erreicht werden. Durch das komplette System kommen noch weitere Funktionen hinzu, wie zum Beispiel das Nachverfolgen der Produkte im Produktionsprozess.

Im Rahmen dieser Diplomarbeit wird, wie schon in der Einleitung beschrieben, der Beacon mit integrierter Signaleinheit und eine Mobile Applikation zum Auslösen der Signaleinheit und Konfiguration der Beacons entwickelt.

3.2 Anforderungsanalyse Bluetooth-Beacon-System

3.2.1 Grundlegende Funktionen

Im Kapitel 2.6.2 werden einige aktuelle Beacons mit ihren Eigenschaften und Funktionen aufgezählt, einige davon soll auch der Prototyp erhalten. Nachfolgend befindet sich eine Zusammenfassung der Funktionen:

- Voller Support für iBeacon und Eddystone
- Sensoren: Bewegung/Beschleunigung, Temperatur, Luftdruck, Umgebungslicht und Luftfeuchtigkeit

- Echtzeituhr
- LEDs und Buttons, GPIO Anschluss
- NFC Kommunikation
- Konfigurierbar: Sendeleistung, Sendeintervall, Sendezeiten (Tag, Nacht, nur zu bestimmten Zeiten, ...)

Die möglichen Funktionen werden einzeln betrachtet und auf ihre Sinnhaftigkeit in Bezug auf das geplante System untersucht. Die Ergebnisse dieser Untersuchung sind nachfolgend aufgelistet.

3.2.2 Beacon Protokoll

Im Kapitel 2.6.1 wurden die beiden bekanntesten Beacon-Protokolle iBeacon und Eddystone bereits etwas näher betrachtet. Beide haben unterschiedliche Einstellungsmöglichkeiten und Inhalte. Jedoch ist das verwendete Beacon-Protokoll für die Anwendung im Forschungsprojekt grundsätzlich nicht wichtig. Zudem kann der Inhalt des Advertising-Pakets jederzeit im Nachhinein einfach geändert werden. Für das Forschungsprojekt ist es nur wichtig, dass die „eigenen“ Beacons erkannt und von anderen unterschieden werden können. Das ist mit dem iBeacon-Protokoll und der Verwendung einer eigenen UUID, sowie Major und Minor ID ohne Probleme möglich. Daher hat sich der Autor dazu entschlossen, für den ersten Prototyp das iBeacon-Protokoll zu verwenden und keine weitere Konfigurationsmöglichkeiten dafür zur Verfügung zu stellen.

Um zu testen ob der Beacon-Prototyp richtig konfiguriert ist und das iBeacon-Protokoll verwendet, wird das fertige System mit einer iBeacon-Scan Applikation getestet.

3.2.3 Sensoren

Die Sensoren für Temperatur, Luftdruck und Luftfeuchtigkeit haben für das Forschungsprojekt wenig Sinn, da diese nicht nützlich weiterverarbeitet oder für etwaige Einstellungsmöglichkeiten verwendet werden können. Es gäbe aber mit Sicherheit Anwendungsfälle bei denen diese Sensoren verwendet werden könnten, wie zum Beispiel bei der Lagerung von sensiblen Gegenständen oder Materialien. Dort könnten die Daten dieser Sensoren zur Überwachung des Lagerplatzes verwendet und bei unter- oder überschreiten von definierten Grenzwerten, bestimmte Aktionen oder Alarme ausgelöst werden. Da der Prototyp später erweitert und für die unterschiedlichsten Anwendungsfälle verwendet werden können soll, werden diese Sensoren zumindest als mögliche Erweiterungen vorgesehen.

Beschleunigungssensor

Wenn kein Echtzeittracking von Assets notwendig ist, könnte man den Beacon so konfigurieren, dass dieser nur Signale aussendet sobald er bewegt wird, wodurch der Energieverbrauch vermindert werden könnte. Zusätzlich könnte man die Daten des Beschleunigungssensors dazu verwenden, um wie bereits vorhin erwähnt, sensible Gegenstände oder Materialien zu überwachen. Ähnlich der vorhin genannten Sensoren, wäre es damit möglich Alarme oder ähnliches auszulösen sobald ein Gegenstand zu schnell bewegt oder in eine nicht wünschenswerte Schräglage gebracht wird.

Für das Forschungsprojekt soll der Beacon andauernd senden, deshalb wird ein Beschleunigungssensor eigentlich nicht benötigt. Da es aber sehr günstige Sensoren gibt und der Autor auch einen zur Verfügung hat, wird dieser bereits im ersten Prototyp integriert. Die Beschleunigungskräfte sowie die aktuelle Lage des Beacons werden aber nur in der Mobilen Applikation angezeigt und nicht weiterverarbeitet.

Umgebungslichtsensor

Ein Umgebungslichtsensor könnte dazu verwendet werden, um bei entsprechenden Lichtverhältnissen die Leistung und somit die Leuchtkraft des Signalgebers entsprechend anzupassen. Weiters könnten damit auch die Zeiten, in der der Beacon Signale aussendet, gesteuert werden. Dafür müssten aber weitere Tests durchgeführt werden, um zu testen ob der Einsatz eines Lichtsensors in Bezug auf den Energieverbrauch sinnvoll ist. Ein einfacher Lichtsensor wird im Prototyp integriert, aber vorerst werden die Daten davon nicht weiterverarbeitet, sondern nur in der mobilen Applikation angezeigt.

3.2.4 Echtzeituhr

Eine Echtzeituhr könnte dazu verwendet werden, um den Energiebedarf zu verringern und somit die Akkulaufzeit des Beacons zu erhöhen. Zum Beispiel könnte der Beacon so konfiguriert werden, dass er nur zu den Arbeitszeiten des entsprechenden Betriebes oder anderen bestimmten Zeiten Signale aussendet. Der Autor sieht diese Funktion als sehr sinnvoll an, deshalb wird im Rahmen der Diplomarbeit ein Prototyp mit einer Echtzeituhr ausgestattet. Die Daten werden in einem ersten Schritt aber nur in der Mobilen Applikation angezeigt und nicht weiterverarbeitet.

3.2.5 LEDs und Buttons

Ein Signalgeber ist von Anfang an für den Beacon vorgesehen, denn dieser ist für das Forschungsprojekt essentiell. Dieser soll es ermöglichen den Benutzer das Auffinden eines Assets zu erleichtern. Zusätzlich zum optischen Signalgeber, wird auch ein akustischer Signalgeber verwendet, um die Unterschiede in der Erkennbarkeit testen zu können.

Buttons sind für den ersten Prototyp nicht vorgesehen, da alle Einstellungsmöglichkeiten mit Hilfe der mobilen Applikation durchgeführt werden können. Eine Erweiterung um Buttons ist aber jederzeit möglich.

3.2.6 NFC Kommunikation

Es ist wichtig das der Beacon zu einem Asset zugeordnet werden kann, im Rahmen des Forschungsprojektes wäre das die entsprechende Auftragsnummer. Mit Hilfe einer App könnte der Benutzer einen Auftrag auswählen und anschließend den entsprechenden Beacon scannen, um die beiden miteinander zu koppeln. Dafür kann aber auch direkt die Bluetooth-Verbindung verwendet werden, daher ist eine NFC Kommunikation für den ersten Prototyp nicht vorgesehen.

3.2.7 Konfigurationsmöglichkeiten

Mit Hilfe der mobilen Applikation sollen einige Einstellungen konfiguriert werden können, dazu zählen die Sendeleistung und das Sendeintervall. Die Sendezeiten sollen in einem weiteren Schritt ebenfalls konfiguriert werden können, jedoch wird diese Möglichkeit im ersten Prototypen aus Zeitgründen nicht umgesetzt. Es soll weiters möglich sein, entweder den optischen oder den akustischen Signalgeber auszulösen.

3.2.8 Zusammenfassung der gewählten Funktionen

- iBeacon Protokoll
- Beschleunigungssensor
- Umgebungslichtsensor
- Echtzeituhr
- Optischer Signalgeber
- Akustischer Signalgeber
- Konfigurationsmöglichkeiten laut Kapitel 3.2.7

3.3 Verwendete Hardware

Als Grundbaustein für den Beacon wird ein Mikrocontroller mit einem Bluetooth Modul benötigt. Weiters wird für jede Funktion aus Kapitel 3.2.8 die entsprechende Hardware gesucht und der aktuelle Preis sowie die technischen Daten dokumentiert. Für die Spannungsversorgung wird ein Li-Ionen Akku verwendet, welcher mit Hilfe einer Ladeplatine per USB geladen werden kann.

3.3.1 Mikrocontroller und Bluetooth-Modul

Die Auswahl von Bluetooth 5 Modulen auf dem Markt ist aktuell nicht sehr groß. Dem Autor war es wichtig das die Module mit Hilfe der Arduino IDE programmiert werden können, da er damit bereits einige Erfahrungen machen konnte.

Der Hersteller RedBear hat im Rahmen eines Kickstarter Kampagne[6] die zwei Bluetooth Low Energy Boards Blend v2 und Nano v2 finanziert und anschließend zum Verkauf angeboten. Leider wurde die Produktion eingestellt und das Nano v2 Board ist nur mehr bis zum ersten Quartal 2019 erhältlich. Für die Diplomarbeit wurden aber bereits vor diese Bekanntgabe ein Blend v2 Board sowie vier Stück vom Nano v2 Board angeschafft, beide basieren auf dem nRF52832 SoC von Nordic Semiconductor. Es hat sich erst nach einer ausführlichen Recherche und ersten Tests herausgestellt, dass dieser Chip leider nur Bluetooth 5 Ready ist und die Long Range Erweiterung von Bluetooth 5 leider nicht unterstützt. Da sich aber im Laufe der Diplomarbeit auch herausgestellt hat, dass es noch kein Smartphone gibt, welches diesen Modus unterstützt, wurden die Boards trotzdem für den ersten Prototyp verwendet.

RedBear wird ein Teil der Firma Particle, welche ab September 2018 einige neue Entwicklerboards zur Verfügung stellt. Unter anderem eines Namens Xenon, welches auf dem nRF52840 SoC von Nordic Semiconductor basiert und dann alle Features von Bluetooth 5 unterstützen soll. Dieses könnte dann in einem weiteren Schritt das RedBear Nano v2 im Prototypen ersetzten.

[6] Mehr dazu: https://www.kickstarter.com/projects/redbearinc/bluetooth-5-ready-ble-module-nano-2-and-blend-2?lang=de

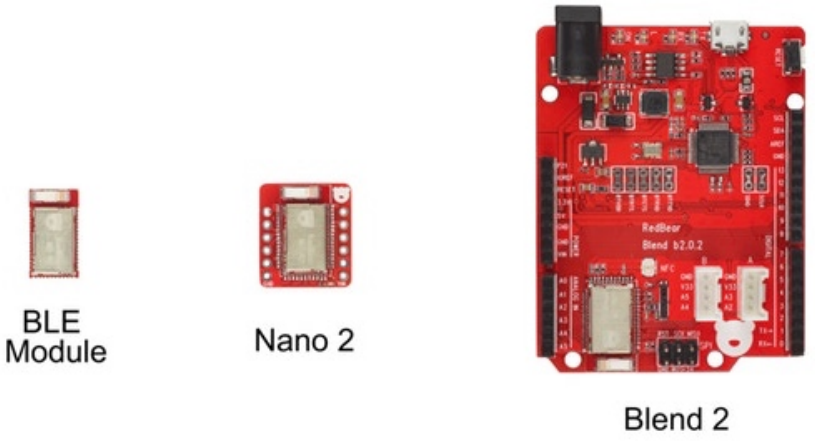

Abbildung 6: RedBear Bluetooth 5 Modul und Boards (Kickstarter, o. J.)

In Abbildung 6 sieht man das BLE Modul, welches das Herzstück der ebenfalls abgebildeten Boards Nano 2 und Blend 2 darstellt. Das BLE Modul beherbergt, wie vorhin bereits erwähnt, den nRF52832 SoC von Nordic Semiconductor. Der Chip ist Bluetooth 4.2 zertifiziert und laut Herstellerangaben Bluetooth 5 ready. Die weiteren wichtigsten technischen Daten des Chips sind nachfolgend aufgelistet.

Nordic Semiconductor nRF52832 SoC – Technische Daten

- Prozessor: 32-bit ARM Cortex-M4F (64MHz)
- Bluetooth: 4.2 zertifiziert und 5.0 Ready (nur BLE)
- Betriebsspannung: 0,7 bis 3,6 Volt
- Flash-Speicher: 512 kB
- Arbeitsspeicher (RAM): 64kB
- I/O Pins: gesamt 32, darunter:
    - eine UART-Schnittstelle (Universal Asynchronous Receiver Transmitter)
    - zwei I²C Schnittstellen (Inter-Integrated Circuit)
    - maximal 12 PWM Ausgänge (Pulsweitenmodulation)
    - maximal 8 ADC Eingänge (Analog Digital Konverter)
- Physikalische Abmessungen: 10mm x 18mm
- Integrierter NFC Tag

(RedBear, 2017/2018)

Weitere Details zu diesem Chip gibt es auf der Website von Nordic Semiconductor unter https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832.

3.3.1.1 RedBear Nano v2 Board

Beim Nano v2 Board wurden nicht alle verfügbaren Pins des nRF52832 Chips ausgeführt, weshalb dieser nur über eine beschränkte Anzahl an Digitalen und Analogen Ein-/Ausgängen verfügt. Diese sind aber für den ersten Prototyp ausreichend.

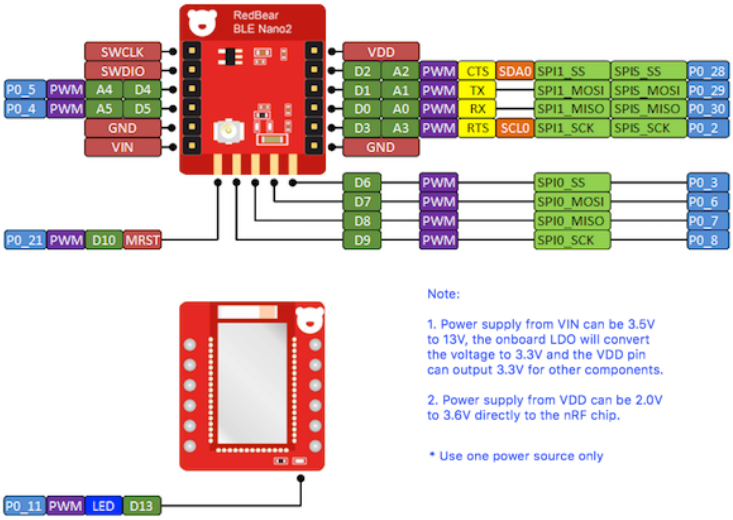

Abbildung 7: Aufbau und Pinbelegung vom RedBear Nano v2 (RedBear, 2017/2018)

In Abbildung 7 sind die verfügbaren Anschlüsse des RedBear Nano v2 Boards abgebildet. Direkt über die Pin Headers sind bei diesem Board somit, sechs analoge bzw. digitale Ein-/Ausgänge verfügbar, welche alle mit einem PWM-Signal angesteuert werden können. Weitere fünf digitale Ein-/Ausgänge sind mittels Lötpunkte verfügbar und an einem digitalen I/O-Pin (D13) ist eine Status-LED angeschlossen.

In der Abbildung steht auch ein wichtiger Hinweis, dieser gibt an, dass die Betriebsspannung zwischen 3,5 Volt und 13 Volt betragen darf, diese wird dann vom integrierten Spannungsregler auf die vom Chip benötigten 3,3 Volt umgewandelt.

Preis RedBear Nano v2

Das RedBear Nano v2 Board wurde um Euro 18,60 im Arduino Store erworben. Aktuell gibt es nur mehr einige Restbestände bei verschiedenen Anbietern.

Da das RedBear Nano v2 Board nicht über einen USB-Anschluss verfügt, wird zum Programmieren ein DAPLink Board benötigt. In Abbildung 8 sieht man das DAPLink v1.5 welches ebenfalls von RedBear hergestellt wird und in einem Set

erhältlich ist. Es kann direkt an einen USB-Anschluss eines PCs angeschlossen werden und unter anderem mit der Arduino IDE programmiert werden.

Abbildung 8: DAPLink Board zum Programmieren des RedBear Nano v2 Boards

3.3.1.2 RedBear Blend v2 Board

Will man alle Funktionen und GPIO-Pins des nRF52832 SoC nutzen, sollte man das RedBear Blend v2 Board in Betracht ziehen. Wie in Abbildung 9 zu sehen, ist diese Board ähnlich einem Arduino Uno aufgebaut und unterstützt somit auch viele Erweiterungsmodule für das Arduino Uno Board. Zusätzlich zu den Standard-Anschlüssen verfügt es noch über zwei Grove-Anschlüsse (Bezeichnung A und B in der Abbildung 9), welche in der späteren Umsetzung nützlich sein werden, da einige Sensoren/Aktoren direkt über dieses System angeschlossen werden können. Beim Grove System handelt es sich um ein modulares standardisiertes Steckverbindungssystem für Prototypen (Seeed Technology Co., o. J.-b).

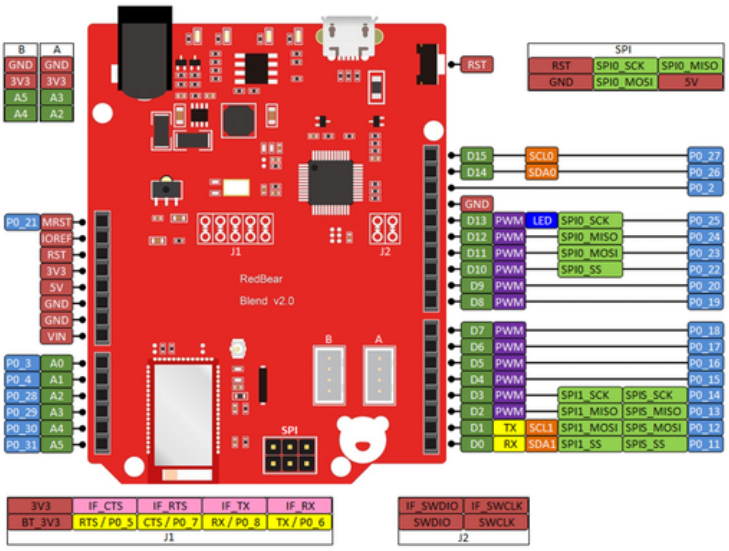

Abbildung 9: Aufbau und Pinbelegung vom RedBear Blend v2 (RedBear, 2017/2018)

Preis RedBear Blend v2

Das RedBear Blend v2 Board wurde um Euro 31,98 im Onlineshop von Mouser Electronics erworben. Aktuell gibt es nur mehr einige Restbestände bei verschiedenen Anbietern.

3.3.1.3 nRF52840 Development Kit

Wie vorhin bereits erwähnt gibt es neben dem nRF52832 SoC auch den nRF52840 SoC von Nordic Semiconductor. Dieser bietet laut Herstellerwebseite vollständige Hardware- und Softwareunterstützung für alle neuen Funktionen, die in Bluetooth 5 eingeführt wurden. Die wichtigsten technischen Daten sind nachfolgend aufgelistet.

Nordic Semiconductor nRF52840 SoC – Technische Daten

- Prozessor: 32-bit ARM Cortex-M4F
- Bluetooth: Bluetooth 5 (Unterstützt den „Long range“ sowie den „High throughput“ Modus
- Unterstützte Bluetooth-Datenraten: 2 Mb/s, 1Mb/s, 500 kb/s, 125 kb/s
- Betriebsspannung: 1,7 bis 5,5 Volt
- Flash-Speicher: 1MB
- Arbeitsspeicher (RAM): 256kB
- I/O Pins: gesamt 48, darunter:
    - zwei UART Schnittstellen
    - zwei I²C Schnittstellen
    - maximal 12 PWM Ausgänge
    - maximal 8 ADC Eingänge (Analog Digital Konverter)
- Integrierter NFC Tag
- Mehr Details: https://www.nordicsemi.com/eng/Products/nRF52840

Um nun alle Neuerungen von Bluetooth 5 testen zu können, gibt es aktuell von Nordic Semiconductor nur ein großes Development Kit mit dem nRF52840 SoC. Wie in Abbildung 10 zu sehen, ist dieses sehr umfangreich aufgebaut, um alle Funktionen und Möglichkeiten des Chips ausprobieren zu können.

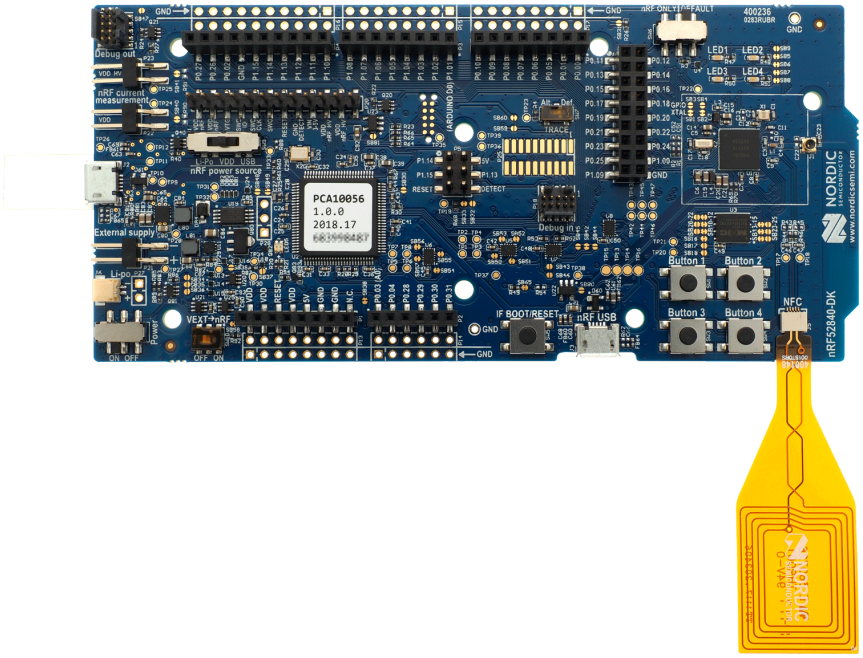

Abbildung 10: Developmentkit für den nRF52840 SoC (Nordic Semi, o. J.)

Um die erweiterte Reichweite von Bluetooth 5 testen zu können, wird ein eigener Test mit zwei dieser Boards durchgeführt. Mehr zu diesem Reichweitentest in Kapitel 4.1.

Die erwähnten Module und Development Kits basieren alle auf einen Chip vom Hersteller Nordic Semiconductor. Laut einer Recherche gibt es zwar auch Bluetooth 5 Chips vom Hersteller Texas Instruments (TI), diese wurden aber im Rahmen dieser Diplomarbeit nicht verwendet, da derzeit keine fertigen Module welche auf einem TI-Chip basieren erhältlich sind. Einzig ein großes Development Kit wäre aktuell erhältlich. Da für die fertigen Module, wie das verwendete RedBear Nano v2, mit Chips von Nordic Semi aber auch eine entsprechende Arduino Bibliothek zum Verfügung steht, wurden diese bei der Auswahl bevorzugt.

3.3.2 Sensoren

Weil es eine unüberschaubare Anzahl an Sensoren am Markt gibt, wird bei der Auswahl der benötigten Sensoren auf einige Details geachtet. Die Sensoren sollen mit einem Standard-Arduino kompatibel sein und sowohl mit 3,3 Volt als auch mit 5 Volt Versorgungsspannung funktionieren. Weiters wurde versucht Grove-Sensoren zu verwenden, weil die Hardware später auch im Unterricht an der FH verwendet werden soll und damit ein schnelles und einfaches Prototyping möglich ist. Wenn es möglich ist werden zudem Sensoren verwendet welche der

Autor bereits vor Beginn der Diplomarbeit zur Verfügung hatte und somit nicht extra angeschafft werden mussten. Für jeden Sensor wird ein eigener Arduino Sketch für den Arduino Uno Mikrocontroller programmiert um die Funktion der Sensoren zu testen. Weiters kann der dabei verwendete Programmcode im Weiteren direkt (nahezu 1:1) für den Prototyp verwendet werden.

3.3.2.1 Beschleunigungssensor GY-61

Abbildung 11: 3-Achsen Beschleunigungssensor GY-61 (Eckstein GmbH, o. J.-b)

Beim GY-61 handelt es sich um einen Beschleunigungsmesser mit dem ADXL335 Sensor, welcher nur einen geringen Stromverbrauch aufweist. Wie man in Abbildung 11 erkennen kann, ist dieser Sensor sehr einfach aufgebaut. Neben der Spannungsversorgung mit 3 bis 5 Volt über die Pins VCC und GND, verfügt er noch über drei analoge Ausgänge für die Beschleunigungswerte in alle drei Richtungen (X_OUT, Y_OUT und Z_OUT). Die Werte der drei Ausgänge können direkt an den analogen Eingängen eines Arduino-Boards eingelesen werden. Dadurch ist es sehr einfach die Daten weiterzuverarbeiten.

Technische Daten und Preis: GY-61

- Sensor Chip: ADXL335
- Betriebsspannung: 3 bis 5 Volt
- Betriebsstrom: 350µA
- Schnittstelle: Analogausgänge
- Betriebstemperatur: -40 ~ 85°C
- Abmessungen (L x W): 20,3mm x 15,7mm
- Messbare Größen
  - Beschleunigung in alle drei Richtungen
  - Aktuelle Drehung und Neigung im dreidimensionalen Raum
- Preis: aktuell um Euro 5,95 im Eckstein-Shop[7] erhältlich

[7] https://eckstein-shop.de/GY-61-ADXL335-3-Achsen-Beschleunigungssensor-Module-Magnetfeld-3-5V-fuer-Arduino

Arduino-Sketch

Der entwickelte Arduino Sketch liest die drei analogen Werte des Beschleunigungssensors ein und gibt sie im Serial Monitor aus. Zusätzlich werden die analogen Werte noch in die aktuellen Lagedaten des Sensors (Winkel in jede Richtung) umgerechnet. In Listing 1 sieht man, dass nur zwei Befehle notwendig sind um einen Wert, in dem Fall den x-Wert, einzulesen und auszugeben.

Listing 1: Einlesen und Ausgabe eines Werts vom Beschleunigungssensor

```
x = analogRead(xPin);
Serial.println(x);
```

Um die Beschleunigungswerte aus den analogen Werten zu berechnen, gibt es eine Bibliothek von Seeed Technology. Die Arduino-Bibliothek ist abrufbar unter https://github.com/Seeed-Studio/Accelerometer_ADXL335 und bietet zwei Beispiel Arduino Sketches. Der Erste dient zur Kalibration des Sensors und der Zweite misst die Beschleunigungskräfte in alle 3 Richtungen und gibt sie am Serial Monitor aus. In Listing 2 sieht man einen zusammengefassten Programmcode um die Beschleunigungskräfte auszulesen. Die Werte in alle drei Richtungen (x, y und z) werden in die Variablen ax, ay, und az als g-Kräfte (Vielfaches der Erdbeschleunigung) gespeichert.

Listing 2: Arduino-Sketch - Einlesen der Beschleunigungskräfte in g

```
#include "ADXL335.h"  //Einbinden der notwendigen Bibliothek
ADXL335 accelerometer;  //Definition der Variable
void setup(){
   accelerometer.begin();      //Initialisierung des Accerlerometers
}
void loop(){
   float ax,ay,az;    //Definition der Variablen für die Beschleunigung
   accelerometer.getAcceleration(&ax,&ay,&az);  //Auslesen der Werte
   delay(100);
}
```

3.3.2.2 Umgebungslichtsensor – Grove Light Sensor v1.2

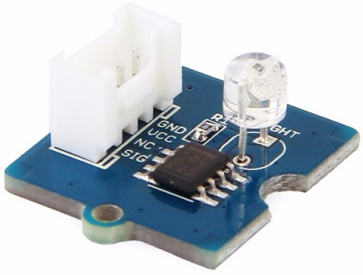

Abbildung 12: Grove Light Sensor v1.2 (Seeed Technology Co., o. J.-a)

Als Lichtsensor wird der „Grove Light Sensor v1.2“ (siehe Abbildung 12) verwendet, es handelt sich dabei um einen Sensor mit hoher Zuverlässigkeit und Sensibilität. Der Wert des analogen Ausgangs ändert sich entsprechend der Helligkeit, liefert jedoch nicht die genaue Lumenzahl (Seeed Technology Co., 2018b).

Technische Daten und Preis: Grove Light Sensor v1.2

- Photo-Widerstand und LM358 Operationsverstärker
- Betriebsspannung: 3 bis 5 Volt
- Betriebsstrom: 0,5 bis 3 mA
- Reaktionszeit: 20 bis 30 ms
- Spitzen-Wellenlänge: 540nm
- Preis: aktuell um Euro 2,90 im Eckstein-Shop[8] erhältlich

Arduino-Sketch

In Listing 3 sieht man den Arduino-Sketch der für das Einlesen des Umgebungslichtsensors notwendig ist. Am analogen Eingang erhält man Werte zwischen 0 und 750, diese werden in Prozente umgewandelt, dadurch erhält man danach Werte zwischen 0 (entspricht kompletter Dunkelheit) und 100 (entspricht Tageslicht).

Listing 3: Arduino-Sketch - Einlesen des Umgebungslichtsensors

```
//Definition und Initialisierung der Variablen
const int lightSensorPin = A0;
int lightValue = 0, lightValuePercentage = 0;
void setup(){
   Serial.begin(9600);  //Initialisierung der seriellen Verbindung
}
void loop(){
   lightValue = analogRead(lightSensorPin); //Einlesen des Wertes
   //Umwandeln des Wertes in eine Prozentzahl
   lightValuePercentage = constrain(
                          map(lightValue, 0, 750, 0, 100),
                          0, 100);
  Serial.println(lightValuePercentage);    //Ausgabe des Wertes
  delay(100);
}
```

8 https://eckstein-shop.de/Seeed-Studio-Grove-Light-Sensor-v12-GL5528-Phototriode-540-nm-3-5-V

3.3.3 Echtzeituhr DS3231

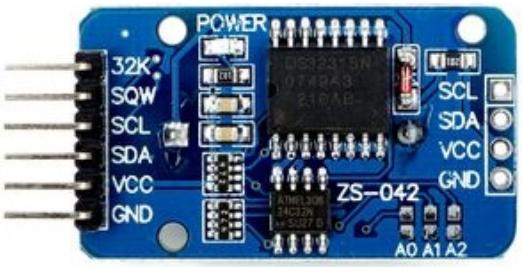

Abbildung 13: DS3231 RTC Modul – Echtzeituhr (Eckstein GmbH, o. J.-d)

Da dem Autor ein DS3231 Echtzeituhrmodul (siehe Abbildung 13) zur Verfügung stand, wurde diese in einem Prototyp verbaut. Dieses Modul muss an den I2C Bus eines Mikrocontrollers angeschlossen werden. Da das für den Prototyp verwendete RedBear Nano v2 Board diesen leider nicht ausgeführt hat, wurde ein weiterer Prototyp mit dem RedBear Blend v2 erstellt. Da die Daten des Echtzeituhrmoduls jedoch vorerst nicht weiterverarbeitet werden und nur in der mobilen Applikation angezeigt werden, war ein Prototyp auch ausreichend.

Beim verwendeten Modul DS3231 handelt es sich um eine kostengünstige und extrem genaue Echtzeituhr. Es arbeitet mit einer Versorgungsspannung von 2,3 bis 5,5 Volt und verfügt über einen zusätzlichen Akku, welcher das Modul immer mit der notwendigen elektrischen Energie versorgt. Im Vergleich zu anderen Modulen verfügt der DS3231 über einen integrierten Quarz (kein externer Quarz erforderlich) und einen Temperatursensor für die Kompensation von Temperaturschwankungen.

Technische Daten und Preis: DS3231

- Betriebsspannung: 3,3 bis 5,5 V
- RTC-Chip: DS3231
- Genauigkeit der Uhr (bei 0°C bis 40°C): 2ppm
- Betriebstemperatur: 0°C bis 70°C
- I2C-Bus-Schnittstelle
- Maximale Übertragungsgeschwindigkeit: 400 kHz (bei 5V)
- Durchgehende Stromversorgung durch eine LIR 2032 Batterie
- Abmessungen: ca. 38 x 22 x 15mm (Länge mit Stiftleiste ca. 44mm)
- Gewicht: ca. 8g
- Preis: aktuell um Euro 3,86 im Eckstein-Shop[9] erhältlich

[9] https://eckstein-shop.de/DS3231-RTC-Modul-LIR2032

Arduino Sketch

In Listing 4 sieht man den Arduino-Sketch der für das Einlesen der Uhrzeit notwendig ist, alle wichtigen Befehle sind direkt im Listing erklärt.

Da dieses Modul auch einen Temperatursensor besitzt, wird der Wert von diesem auch eingelesen und im Serial Monitor angezeigt. Die Temperatur wird normalerweise nur alle 64 Sekunden ausgelesen und in einem Register abgespeichert. Um immer die aktuelle Temperatur zu erhalten muss das Auslesen des Sensors mit der Funktion „forceConversion“ erzwungen werden.

Listing 4: Arduino-Sketch - Einlesen der Uhrzeit vom DS3231

```
//Einbinden der notwendigen Bibliotheken
#include <Wire.h>
#include <DS3231.h>
//Definition der Variablen
DS3231 clock;
RTCDateTime dt;
void setup(){
   Serial.begin(9600);  //Initialisierung der seriellen Verbindung
   clock.begin();       //Initialisierung des Echtzeituhrmoduls
   clock.setDateTime(__DATE__, __TIME__);
                        //Einstellen der aktuellen Uhrzeit
}
void loop(){
   dt = clock.getDateTime(); //Einlesen der aktuellen Uhrzeit
                          //Ausgabe der formatierten Uhrzeit
   Serial.println(clock.dateFormat("d-m-Y H:i:s", dt));

   clock.forceConversion();  //Erzwingen der Temperaturumwandlung
   Serial.println(clock.readTemperature()); //Ausgabe der Temperatur
   delay(1000);
}
```

Das Format der Uhrzeit und des Datums kann mit der Funktion „dateFormat“ jederzeit wie gewünscht angepasst werden. Zum Beispiel ist es möglich mit dem Befehl in Listing 5 den Unix-Zeitstempel auszugeben.

Listing 5: Funktion dateFormat - Unix Zeitstempel

```
clock.dateFormat("U", dt)
```

3.3.4 Lithium-Ionen-Akku

Bei der Energieversorgung werden handelsübliche Lithium-Ionen-Akkus mit einer Spannung von 3,7 Volt und einer elektrischen Ladung von 2500mAh verwendet. Der Schutz von Über- bzw. Unterspannung ist bei den verwendeten Akkus durch eine integrierte Schutzschaltung bereits gewährleistet. Somit kann der Akku direkt zur Versorgung des Prototyps verwendet werden. Im Vergleich zu den in Kapitel 2.6.2 betrachteten Beacons wurde eine durchschnittliche Akkukapazität gewählt. Es ist aber durchaus möglich einen Akku mit einer höheren Kapazität zu verbauen. Eine höhere Kapazität wirkt sich aber natürlich auf die physikalische Größe des Akkus aus, hier muss dann ein gewisser Kompromiss zwischen Akkulebensdauer und Kompaktheit gefunden werden.

Der Akku wird mit einem Standard 2-Pin JST-PH Steckverbinder (vgl. Abbildung 14) geliefert und kann somit direkt an das verwendete Ladegerät (siehe Punkt 3.3.5) angeschlossen werden.

Abbildung 14: Lithium-Ionen Akku mit 2500mAh und JST Stecker (Eckstein GmbH, o. J.-c)

Technische Daten und Preis:

- Gewicht: 49g
- Maße: 62mm x 51mm x 7,9mm
- Nennkapazität: 2500mAh
- Nennspannung: 3,7 Volt
- Integrierte Schutzschaltung gegen Über- bzw. Unterspannung
- Preis: aktuell um Euro 8,45 im Eckstein-Shop[10] erhältlich

[10] https://eckstein-shop.de/LiPo-Akku-Lithium-Ion-Polymer-Batterie-37V-2500mAh-JST-PH-Connector

3.3.5 Akku Ladegerät

Auch wenn die maximale Lebensdauer mit einer Akkuladung noch nicht bekannt ist, so ist es dem Autor trotzdem wichtig, dass der Beacon einfach aufgeladen werden kann. Dazu wird ein Ladegerät verwendet welches neben Lithium-Ionen (Li-Ion) auch Lithium-Polymer-Akkus (LiPo) laden kann.

Wie in Abbildung 15 zu sehen verfügt es über zwei JST-PH Buchsen und einem USB Mini-B Anschluss. Dadurch ist es möglich, dass sowohl die Batterie als auch die zu versorgende Schaltung während des Ladevorgangs direkt angeschlossen bleiben können. Durch den USB-Mini-B Anschluss kann zum Laden, jeder herkömmliche USB-Adapter (z.B.: von Smartphones, usw.) verwendet werden. Es ist nur ein Kabel mit eine USB-Mini Stecker notwendig.

Der Akku sowie das Ladegerät können durch die Steckverbinder auch sehr einfach ausgetauscht werden, beziehungsweiße überhaupt räumlich abgetrennt von der eigentlichen Schaltung verwendet werden.

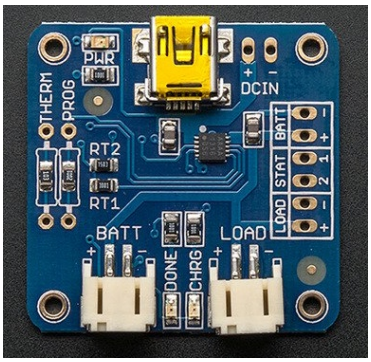

Abbildung 15: Adafruit LiIon/LiPoly Charger v1.2 (Adafruit, o. J.)

Das Board verfügt, wie in Abbildung 15 erkennbar, über 3 Status-LEDs – eine für die Stromversorgung (PWR), eine die anzeigt das gerade geladen wird (CHRG) und eine die leuchtet sobald der Ladevorgang abgeschlossen ist (DONE).

Technische Daten und Preis:

- Gewicht: 5,7g
- Maße: 33mm x 35mm x 7mm
- Standard-Ladestrom: 500mA (einstellbar von 100mA bis 1000mA)
- Zum Laden von 3,7 Volt oder 4,2 Volt Einzelzellen Li-Ion oder LiPo Akkus
- Preis: aktuell um Euro 14,88 im Eckstein-Shop[11] erhältlich

[11] https://eckstein-shop.de/Adafruit-USB-LiIon-LiPoly-charger

3.3.6 Signalgeber

Für das Forschungsprojekt ist ein Signalgeber ein wesentlicher Bestandteil des Beacon-Prototyps, daher wird im Rahmen der Diplomarbeit die Erkennbarkeit eines optischen und eines akustischen Signalgebers in einem Test eruiert.

3.3.6.1 Optischer Signalgeber: LED NeoPixel Ring

Da der Autor zusätzlich auch das Erkennen von unterschiedlichen Farben miteinander vergleichen will, wird als erster Schritt ein RGB LED Ring als optischer Signalgeber verwendet. Wie in Abbildung 16 erkennbar besteht der Ring aus 16 RGB LEDs, bei denen es sich um 5050 SMD LEDs mit einer durchschnittlichen Lichtstärke von 16 bis 22 Lumen handelt. Jeder sogenannte Pixel kann einzeln angesprochen und entsprechend gesteuert werden. Bei diesem LED Ring handelt es sich um einen NeoPixel Ring vom Hersteller Adafruit.

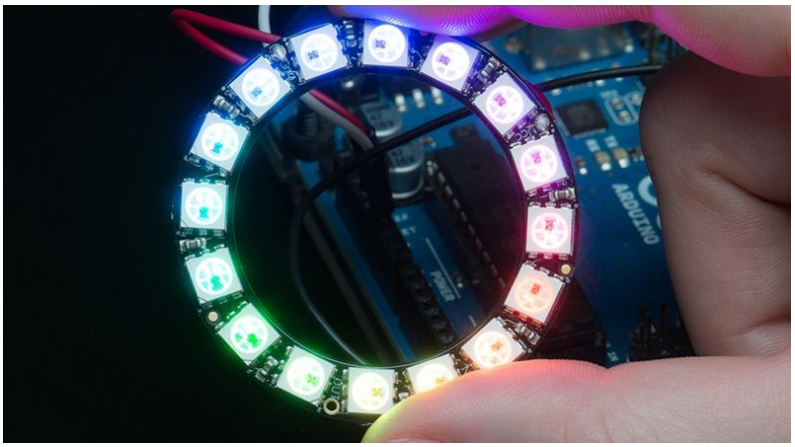

Abbildung 16: LED Ring (16 LEDs) von Adafruit (Eckstein GmbH, o. J.-a)

Technische Daten und Preis: 16 NeoPixel Ring

- Versorgungsspannung: 5V (weniger möglich, siehe nächsten Abschnitt)
- Integrierter Schaltkreis: WS2812B (Neue Version: SK6812)
- LED Chip Typ: 5050 SMD
- Betriebstemperatur: -40 bis +85°C
- Lichtstärke: 20 bis 22 Lumen
- Außendurchmesser: 72mm (Neue Version: 44,5mm)
- Preis: aktuell um Euro 11,84 im Eckstein-Shop[12] erhältlich

[12] https://eckstein-shop.de/Adafruit-NeoPixel-Ring-16-x-5050-RGB-LED-with-Integrated-Drivers

Die LED-Ringe sind in unterschiedlichen Größen mit einer unterschiedlichen Anzahl an Pixel erhältlich. NeoPixel sind auch in einigen weiteren Ausführungen erhältlich, wie zum Beispiel als LED-Streifen oder als einzelne LED-Pixel.

Wichtige Information bei der Verwendung von Adafruit NeoPixel

Der Hersteller Adafruit gibt auf seiner Webseite[13] viele nützliche Tipps für die Verwendung von NeoPixel. Einige davon sind im folgenden Abschnitt beschrieben, da diese für die Funktion des Prototyps sehr wichtig sind.

Kondensator parallel zur Spannungsversorgung

Um eine richtige Funktion des NeoPixel Rings zu gewährleisten, sollte ein Kondensator parallel zur Spannungsversorgung (zwischen Plus und Minus der Spannungsquelle) angeschlossen werden. Empfohlen wird ein Elektrolytkondensator mit einer Kapazität von 1000µF und einer Nennspannung von 6,3 Volt oder höher. Dieser dient dazu um plötzliche Änderungen in der Energieaufnahme zu puffern und eine durchgehende Versorgung aller Bauteile zu gewährleisten *(Adafruit, 2016).*

Widerstand am digitalen Ausgang des Mikrocontrollers

Wird der NeoPixel Ring über einen digitalen Ausgang eines Mikrocontrollers (z.B.: Arduino, ...) gesteuert, so sollte zwischen diesem und dem digitalen Eingang des NeoPixels ein Widerstand (300 bis 500 Ohm) platziert werden. Dieser Widerstand dient dazu, um Schwankungen im Signal zu vermeiden *(Adafruit, 2016).*

5 Volt oder 3,3 Volt Spannungsversorgung

Werden die NeoPixels mit 5 Volt versorgt, so ist auch ein 5 Volt Datensignal notwendig. Sollte ein Mikrocontroller genutzt werden welcher mit 3,3 Volt arbeitet, so muss ein Pegelumsetzer (Logic-Level-Shifter) verwendet werden. Das RedBar Nano v2 Board, welches für den Prototyp verwendet wird, arbeitet mit 3,3 Volt an den digitalen Ausgängen. Da aber der NeoPixel Ring mit einem Li-Ion Akku und somit mit 3,7 Volt versorgt wird, reicht ein 3,3 Volt Datensignal aus *(Adafruit, 2016).*

Gemeinsame Masse

Werden der Mikrocontroller und der NeoPixel Ring von zwei unterschiedlichen Quellen gespeist, so ist es unbedingt notwendig eine Masseverbindung zwischen den beiden herzustellen *(Adafruit, 2016).*

[13] https://learn.adafruit.com/adafruit-neopixel-uberguide

Testaufbau

Da die Verwendung und Programmierung des Neopixel Rings etwas komplexer ist, wird ein Testaufbau mit dem RedBear Nano v2 angefertigt und getestet. Die Abbildung 17 zeigt die Schaltung welche für den Test verwendet wird. Alle Tipps des Herstellers, wie der Kondensator und der Widerstand, werden befolgt und es wird ein 3,7 Volt Li-Ion Akku zur Spannungsversorgung verwendet.

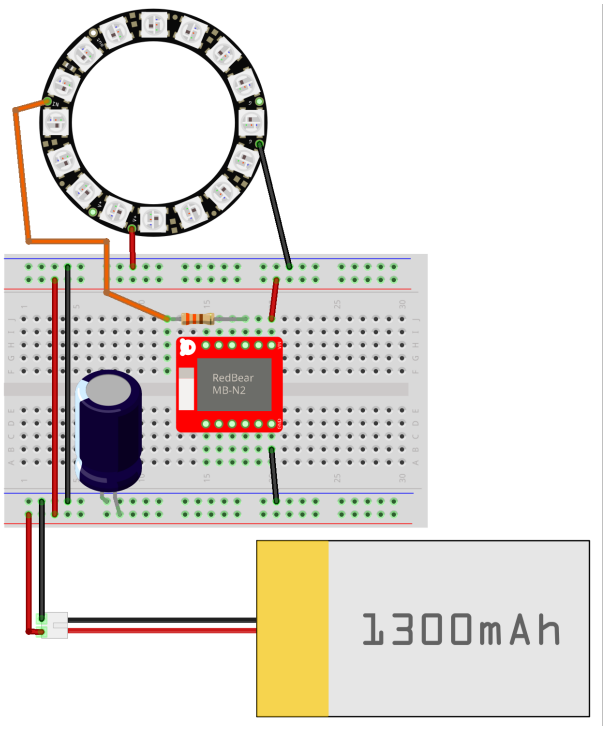

Abbildung 17: Testaufbau NeoPixel und RedBear Nano v2

Arduino Sketch

Um die Funktion des NeoPixel Rings ausführlich zu testen, wurden einige Lichtformen (Blitzlicht, Drehlicht, Farbwechsel, Dauerlicht, ...) und auch die verschiedensten Farben (Rot, Grün, Blau, Weiß, Violett, ...) ausprobiert. In Listing 6 sind die Programmzeilen angeführt, welche für ein einfaches blaues Dauerlicht notwendig sind.

Listing 6: Arduino Sketch Testaufbau RedBear Nano v2

```
#include <Adafruit_NeoPixel.h>     //Einbinden der NeoPixel Bibliothek
//Definition der NeoPixel Variable
Adafruit_NeoPixel strip =
                Adafruit_NeoPixel(16, 5, NEO_GRB + NEO_KHZ800);
void setup() {
   strip.begin();     //Initialisierung des NeoPixel Rings
   strip.setBrightness(255); //NeoPixel - Volle Leuchtkraft
```

```
   strip.show();      // NeoPixel Ring aktualisieren
}
void loop() {
   for(uint16_t i=0; i<=strip.numPixels(); i++) {
      //jedes einzelne Pixel wird in dieser Schleife auf Blau gestellt
      strip.setPixelColor(i, strip.Color(0, 0, 255));
   }
   strip.show(); //NeoPixel Ring aktualisieren
}
```

Zu Beginn muss die Anzahl und Art der verwendeten NeoPixel definiert werden und der digitale Ausgang am Mikrocontroller ausgewählt werden (vgl. Zeile 1 im Listing 7). Die Art der NeoPixel wird als Kombination folgender zweier Werte angegeben:

1. Verwendete Frequenz:
    a. 800kHz -> NEO_KHZ800
    b. 400kHz -> NEO_KHZ400
2. Verwendete Verkabelung:
    a. RGB -> NEO_GRB
    b. GRB -> NEO_RGB

Welche Werte man verwenden muss, findet man in der Beschreibung des entsprechenden NeoPixels. Der im Prototyp verwendete 16er-NeoPixel Ring verwendet 800kHz und die GRB-Verkabelung. Daher wird beim Typ der Neopixel der folgende Wert angegeben: NEO_GRB + NEO_KHZ800.

Listing 7: Funktionen für die Verwendung von NeoPixel

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(<Anzahl an Pixel>,
<Digitaler Pin>, <Typ der NeoPixel>);


strip.setPixelColor(<Pixel>, strip.color(<Rot>, <Blau>, <Grün>));
```

Mit der Funktion „setPixelColor“ kann jedem einzelnen LED-Baustein eine Farbe zugewiesen werden (vgl. Zeile 2 im Listing 7).

3.3.6.2 Akustischer Signalgeber: Piezo Lautsprecher

Als akustischer Signalgeber wird ein Piezo Lautsprecher mit dem Grove System verwendet (siehe Abbildung 18). Bei einer Resonanzfrequenz von 2300±300Hz erreicht der Lautsprecher eine Lautstärke von ungefähr 85dB. Laut Angaben benötigt er eine Spannung zwischen 4 und 8 Volt, nach den ersten Tests wurde aber festgestellt, dass der Lautsprecher auch mit den 3,7 Volt des verwendeten Akkus sehr gut funktioniert.

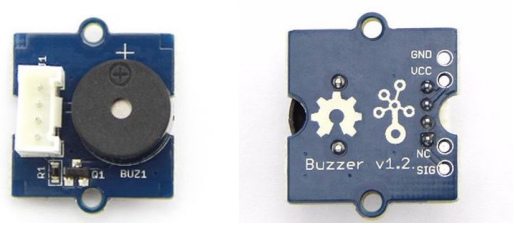

Abbildung 18: Grove Buzzer 85dB Audiosignal (Eckstein GmbH, o. J.-e)

Technische Daten und Preis: Grove Buzzer

- Gewicht: 7g
- Maße: 24mm x 20mm x 9,8mm
- Soundausgabe: 85dB
- Resonanzfrequenz: 2300±300Hz
- Betriebsspannung: 4 bis 8 Volt
- Preis: aktuell um Euro 1,90 im Eckstein-Shop[14] erhältlich

Arduino Sketch

Um den Buzzer zu testen, muss dieser auf den Pins VCC und GND mit Spannung versorgt werden. Schließt man dann noch den Pin SIG an einen digitalen Ausgang eines Mikrocontrollers an, reicht die Programmzeile in Listing 8 um einen Ton auszugeben. Es gibt noch weitere Möglichkeiten um mit Hilfe des Buzzers verschiedene Töne auszugeben, zum Beispiel die tone-Funktion[15] von Arduino. Auf diese wird aber nicht näher eingegangen, da die Ausgabe eines einfachen Tons für den Prototyp ausreichend ist.

Listing 8: Programmzeilen zum Testen des Grove Buzzers

```
digitalWrite( <Digitaler Pin>, HIGH);
```

14 https://eckstein-shop.de/Seeed-Studio-Grove-Buzzer-Sound-Output-85dB-Resonant-Frequency-2300a300Hz

15 https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

3.3.7 Kostenübersicht Prototyp

Um die erste Forschungsfrage „Zu welchem Gesamtpreis in Zusammenhang mit sinnvollen Funktionen kann ein funktionierender Bluetooth 5 Beacon umgesetzt werden?“ beantworten zu können wird eine Kostenübersicht über die Bestandteile des Prototyps erstellt. Weiters werden die Preise aller Bauteile aufgelistet um einen Überblick zu erhalten.

3.3.7.1 Einfacher Prototyp mit dem RedBear Nano v2 ohne Sensoren

Der Autor hat sich dazu entschieden aus Zeitgründen die drei Prototypen welche für einige Tests benötigt werden, nur einfach aufzubauen und auf zusätzliche Sensoren zu verzichten. Bei dieser Kostenübersicht (siehe Tabelle 6) fehlt noch das Gehäuse des Beacons, welches für die Diplomarbeit kostenlos an der Fachhochschule St. Pölten ausgedruckt werden konnte. Für einen besseren Vergleich wäre es aber natürlich sinnvoll auch diese Kosten miteinzubeziehen. Da aber keine konkreten Preise für den 3D Druck zur Verfügung stehen, wurden diese aber im ersten Schritt vernachlässigt.

Daher ergibt sich ein Gesamtpreis von Euro 57,67 für den im Rahmen der Diplomarbeit entwickelten Prototyp mit akustischem und optischem Signalgeber.

Tabelle 6: Kostenübersicht - Einfacher Prototyp mit RedBear Nano v2 ohne Sensoren

Bezeichnung	**Preis**
RedBear Nano v2	18,60 €
Akku	8,45 €
Ladegerät	14,88 €
LED Ring	11,84 €
Grove Buzzer	1,90 €
Kleinteile (Kondensator, Widerstand, Platine, Kabel, Stecker, ...)	ca. 2,00 €
Gesamt	**57,67 €**

3.3.7.2 Vollausgebauter Prototyp mit RedBear Blend v2 und allen Sensoren

Ein Prototyp wird laut den Angaben in Kapitel 3.2.8 mit allen Funktionen umgesetzt. Dieser wird aber nur zum Testen der mobilen Applikation und all ihren Funktionen verwendet. Aufgrund des höheren Preises vom RedBear Blend v2 Board und der drei zusätzlichen Sensoren ergibt sich ein etwas höherer

Gesamtpreis von Euro 83,76 (vgl. Tabelle 7). Auch bei dieser Berechnung fehlen, wie in Punkt 3.3.7.1, die Kosten für das Gehäuse.

Tabelle 7: Kostenübersicht - Vollausgebauter Prototyp mit RedBear Blend v2 und allen Sensoren

Bezeichnung	**Preis**
RedBear Blend v2	31,98 €
Beschleunigungssensor GY-61	5,95 €
Grove Light Sensor	2,90 €
Echtzeituhrmodul	3,86 €
Akku	8,45 €
Ladegerät	14,88 €
LED Ring	11,84 €
Grove Buzzer	1,90 €
Kleinteile (Kondensator, Widerstand, Platine, Kabel, Stecker, ...)	ca. 2,00 €
Gesamt	**83,76 €**

3.3.7.3 Kostenübersicht aller Bauteile

Tabelle 8: Kostenübersicht aller Bauteile

Bezeichnung	**Preis**
RedBear Nano v2	18,60 €
RedBear Blend v2	31,98 €
nRF52840 Development Kit	47,33 €
Beschleunigungssensor GY-61	5,95 €
Grove Light Sensor	2,90 €
Echtzeituhrmodul	3,86 €
Akku	8,45 €
Ladegerät	14,88 €
LED Ring	11,84 €
Grove Buzzer	1,90 €

In Tabelle 8 sind die aktuellen Preise (Stand: Juli 2018) aller in Kapitel 3.3 erwähnten Bauteile aufgelistet. Es fällt auf, dass die als sinnvoll angesehenen Sensoren im Vergleich zu den anderen Bauteilen günstig sind. Sollte ein Anwendungsfall keinen Signalgeber erfordern, könnte man den Gesamtpreis des Beacons somit etwas senken.

3.3.7.4 Vergleich mit kommerziellen Beacons

Tabelle 9: Vergleich Beacon Prototyp mit kommerziellen Beacons

	RedBear Nano Beacon	RedBear Blend Beacon	Kontakt.io Pro Beacon	RuuviTag	Estimote Beacon	BeWhere Beacon BTB-04
Voller Support für iBeacon und Eddystone	konfigu-rierbar	konfigu-rierbar	Ja	Ja	Ja	k.A.
Beschleunigungs-sensor	Nein	Ja	Ja	Ja	Ja	Ja
Temperatursensor	Nein	Ja	Nein	Ja	Ja	Ja
Luftdrucksensor	Nein	Nein	Nein	Ja	Ja	Nein
Umgebungslicht-sensor	Nein	Ja	Ja	Nein	Ja	Ja
Luftfeuchtigkeits-sensor	Nein	Nein	Nein	Ja	Nein	Nein
Echtzeituhr	Nein	Ja	Ja	Nein	Ja	Nein
LEDs	Ja, LED Ring	Ja, LED Ring	kleine Status-LED	ja, zwei	Nein	Ja
Buttons	Nein	Nein	Nein	ja, zwei	Nein	Nein
Akustischer Signalgeber	Ja	Ja	Nein	Nein	Nein	Ja
GPIO Anschluss	Nein*	Nein*	Nein	Nein	Ja	Nein
NFC Kommunikation	Nein*	Nein*	Ja	Nein	Ja	Nein
Bluetooth 5	ready	ready	ready	ready	Nein, BLE 4.2	Nein, BLE 4.0
Akkukapazität in mAh	2500	2500	3000	1000	4000	k.A.
Ladeanschluss	Ja	Ja	Nein	Nein	Nein	Nein
Preis pro Stück in Euro	57,75	83,76	30,00	23,00	28,50	50,00

* Nicht umgesetzt, kann aber jederzeit ohne zusätzliche Hardware hinzugefügt werden.

In Tabelle 9 sieht man eine Übersicht über alle Eigenschaften und den Preis der Beacon Prototypen im Vergleich zu den kommerziellen Beacons welche in

Kapitel 2.6.2 beschrieben werden. Zusätzlich wurde auch noch der Beacon von BeWhere zum Vergleich herangezogen (vgl. Kapitel 2.3.5).

Im Vergleich zu den kommerziellen Beacons, ist der entwickelte Prototyp, egal in welcher Ausbaustufe deutlich teurer. Jedoch verfügt keiner der kommerziellen Beacons über einen direkten Ladeanschluss, welcher bei beiden Prototypen mit fast 15 € doch sehr teuer ausfällt. Würde man auch noch auf den optischen Signalgeber (LED-Ring: ca. 12 €) verzichten, würde man zumindest beim RedBear Nano Prototyp annähernd an die Preise herankommen. Jedoch hat man dann noch keine Sensoren verbaut.

3.3.7.5 Fazit – Kosten

An die Preise von kommerziellen Beacons, welche industriell und serienmäßig gefertigt werden, heranzukommen ist natürlich schwierig. Der entwickelte Prototyp hat aber auch einige entscheidende Vorteile, wie den Ladeanschluss und den optischen Signalgeber, welche den höheren Preis verursachen. Zusätzlich kann der selbstentwickelte Beacon sehr leicht erweitert und sobald eines verfügbar ist mit einem vollständigen Bluetooth 5 Modul aufgebaut werden, ohne große Veränderungen durchführen zu müssen. Ein mögliches Modul ist das Xenon Develpment Kit von Particle, welches auf dem nRF52840 SoC von Nordic Semiconductor aufgebaut ist und alle Erweiterungen von Bluetooth 5 zur Verfügung stellt. Zum Zeitpunkt der Diplomarbeit war dieses jedoch noch nicht erhältlich.

Natürlich hat der Prototyp auch einige Nachteile, wie die fehlenden Sensoren. Zudem ist das entwickelte Gehäuse nicht spritzwassergeschützt und verfügt auch über keine entsprechende Zertifizierung im Gegensatz zu anderen Beacons. Beim Gehäuse ist aber auch die transparente Kuppel zu erwähnen, welche dem optischen Signalgeber eine gute Erkennbarkeit verschafft, was wiederum einen Vorteil des Prototyps darstellt.

Um die Kosten zu senken gäbe es aber auch einige Möglichkeiten. Zum Beispiel könnte man nach günstigerer Hardware suchen oder den Beacon ohne fertige Module entwickeln. Dafür wäre es allerdings notwendig eine eigene Platine mit allen Chips (Bluetooth, Sensoren, ...) und der Schaltung rundherum zu entwerfen. Das würde jedoch das derzeitige elektronische Know-how des Autors überschreiten. Ab einer gewissen Anzahl an Beacons könnte der Preis auch deutlich reduziert werden, da die Kosten für die Bauteile dann sinken.

3.4 Prototyp: Bluetooth Beacon plus LED

Um sich mit dem Thema Bluetooth auf der Hardwareseite vertraut zu machen, wird ein erster Protoyp mit dem RedBear Nano v2 Entwicklerboard entwickelt und aufgebaut. Der Prototyp verfügt über eine LED, welche mit Hilfe einer mobilen Applikation und Bluetooth ein- bzw. ausgeschalten werden kann. Zusätzlich soll der Beacon als iBeacon von einem Smartphone erkannt werden.

3.4.1 Hardware

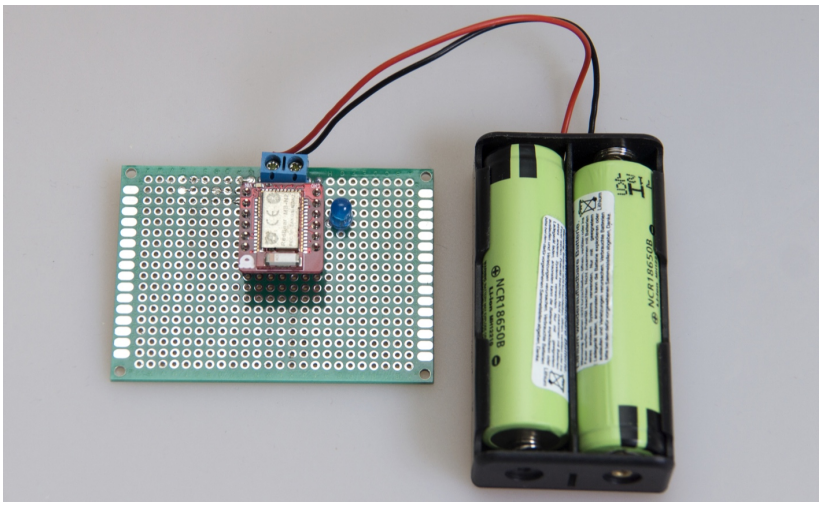

Abbildung 19: Erster Beacon + LED Prototyp

Auf der Abbildung 19 sieht man den fertigen Prototyp bestehend aus dem RedBear Nano v2 Board, einer blauen LED und 2 Stück 18650 Li-Ionen Akkus. Mehr Bauteile sind für den ersten funktionierenden Hardware-Prototyp nicht notwendig. Die LED ist am digitalen Pin 3 des Moduls angeschlossen, welcher als Ausgang definiert wird. Da das Bluetooth-Modul an den I/O-Ports mit 3,3 Volt arbeitet und eine blaue LED verwendet wird, kann die LED ohne weiteren Widerstand direkt an den digitalen Ausgang angeschlossen werden.

3.4.2 Programmierung

Die Programmierung des RedBear Nano v2 wird mit der Arduino IDE durchgeführt. Dafür muss zuerst das entsprechende Board Package installiert werden. Eine Anleitung für die diese Installation gibt es auf dem Github Repository[16] des Herstellers RedBear.

[16] https://github.com/redbear/nRF5x/blob/master/nRF52832/docs/Arduino_Board_Package_Installation_Guide.md

Nachdem das Board installiert ist, kann es in der Arduino IDE unter Tools->Board ausgewählt werden (BLE_Nano2). Es werden bereits einige Beispiel Sketches mitgeliefert, unter anderem ein einfacher iBeacon. Dieser Code wurde als Ausgangspunkt für den Prototyp verwendet. Folgend werden einige wichtige Code-Schnipsel erklärt.

Beacon Payload

Wie bereits in Kapitel 2.6.1.1 erklärt, besteht der iBeacon Standard grundsätzlich aus einer UUID, Major ID, Minor ID und der Sendeleistung. Alle diese Daten, abgesehen von der Sendeleistung, diese wird vom jeweiligen Empfänger gemessen, müssen definiert werden. In Listing 9 sieht man die Definition der drei Werte und einigen anderen Daten, welche als Beacon Payload bezeichnet werden.

Listing 9: Definition des Beacon Payload für den iBeacon Prototyp

```
const static uint8_t beaconPayload[] = {
   0x4C,0x00,                 // Company Identifier Code = Apple
   0x02,                      // Type = iBeacon
   0x15,                      // Following data length
   0x91,0x3f,0x44,0xa7,0xe1,0xf7,0x4b,0xd1,
   0xbe,0xbd,0xc3,0x52,0x85,0xeb,0x98,0x42, // Beacon UUID
   0x00,0x01,                 // Major
   0x01,0x00,                 // Minor
   0xC5                       // Measured Power
};
```

Der Beacon Payload wird als Byte-Array definiert, wobei alle Werte in HEX angegeben werden. Die ersten vier Bytes sind vordefinierte Werte des iBeacon-Protokolls (Hersteller und Typ) sowie ein Wert der angibt wie viele Bytes folgen (0x15 entspricht 21 Bytes). Danach kommt die UUID (16 Byte) sowie die Major ID und Minor ID (jeweils 2 Byte). Als letzter Wert ist die Signalstärke des Beacons in einem Meter Abstand angegeben, dieser wird zum Messen der ungefähren Entfernung benötigt. Die definierten Werte entsprechend somit folgenden Daten:

- Beacon UUID: 913f44a7-e1f7-4bd1-bebd-c35285eb9842
- Major ID: 1 (HEX: 0x0001)
- Minor ID: 256 (HEX: 0x0100)
- Measured Power: -59dBm (HEX: 0xC5)

In der Setup-Methode des Arduino-Sketches wird der Beacon Payload mit der Funktion „accumulateAdvertisingPayload“ (vgl. Listing 10) zum Advertising-Paket hinzugefügt.

Listing 10: Hinzufügen des Beacon-Payload zum Advertising-Payload

```
void setup(){
   ...
   ble.accumulateAdvertisingPayload(
      GapAdvertisingData::MANUFACTURER_SPECIFIC_DATA,
      beaconPayload, sizeof(beaconPayload));
   ...
}
```

Einstellen eines benutzerdefinierten Namens

Um beim Scannen nach Bluetooth-Geräten in einer entsprechenden Mobilen Applikation die Geräte unterscheiden zu können, ist es möglich dem Gerät einen Namen zu geben. In diesem Fall wird der Name „BlueDAT-0010“ dem Scan-Response Payload hinzugefügt, welcher von einem Smartphone angefordert werden kann. Die dafür notwendigen Programmzeilen sind in Listing 11 angeführt.

Listing 11: Gerätenamen zum Scan-Response Payload hinzufügen

```
const static char deviceName[] = "BlueDAT-0010";

void setup(){
   ...
   ble.accumulateScanResponse(
      GapAdvertisingData::COMPLETE_LOCAL_NAME,
      (uint8_t *)deviceName,
      sizeof(deviceName)
   );
   ...
}
```

Eigenes Service implementieren

Die am Beacon angeschlossene LED soll mit Hilfe einer App ein- bzw. ausgeschalten werden können. Dafür wird ein eigenes GATT-Service mit einer „Write without Response“-Charakteristik definiert, dadurch ist es möglich Daten an den Bluetooth Beacon zu senden. In Listing 12 sieht man zuerst die Definition der UUIDs für das Service und die Charakteristik. Danach werden sowohl die Charakteristik als auch das Service erstellt. Abschließend wird in der Setup-Methode noch das Service dem BLE-Objekt hinzugefügt.

Listing 12: Definiton des Service und der Charakteristik

```
//Initialisierung und Definition der UUIDs
static const char serviceUUID[] = "713D0000-503E-4C75-BA94-
3148F18D941E";
static const char charWriteUUID[] = "713D0003-503E-4C75-BA94-
3148F18D941E";


//Erstellung der WRITE_WITHOUT_RESPONSE Charakteristik
uint8_t writeValue[20] = {0};
GattCharacteristic char1(charWriteUUID, writeValue, 1,
sizeof(writeValue),
GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_WRITE_WITHOUT_RESPONSE);


 //Erstellung des Service und hinzufügen der Charakteristik
GattCharacteristic *chars[] = {&char1};
GattService service(serviceUUID, chars, sizeof(chars) /
sizeof(GattCharacteristic *));


void setup(){
   ...
   ble.addService(uartService);
   ...
}
```

Neustart des Advertising Prozesses

Es ist sehr wichtig das eine Funktion definiert wird, welche beim Trennen der Verbindung aufgerufen wird, damit der Beacon wieder mit dem Aussenden von Advertising Paketen beginnt. Ansonsten kann er nicht mehr von einem Smartphone gefunden werden. In Listing 13 ist die Funktion angeführt, welche das Advertising wieder startet, diese muss in der Setup-Methode initialisiert werden.

Listing 13: Definition der Callback-Funktion beim Trennen der Verbindung

```
void disconnectionCallBack(const Gap::DisconnectionCallbackParams_t
*params){
   ble.startAdvertising();
}
void setup(){
   ...
   ble.onDisconnection(disconnectionCallBack);
   ...
}
```

Verarbeiten der Daten von einem Central-Gerät

Die Daten welche von einem Central-Gerät, in dem Fall von einem Smartphone, gesendet werden, müssen natürlich auch verarbeitet werden. Dies passiert mit Hilfe der Callback-Funktion welche in der Setup-Methode mit dem Befehl

„onDataWritten“ definiert wird. In Listing 14 sind die dafür notwendigen Programmzeilen angeführt und mit entsprechenden Kommentaren versehen.

Listing 14: Definition der Funktion, welche die empfangenen Daten verarbeitet (gattServerWriteCallBack)

```
void gattServerWriteCallBack(const GattWriteCallbackParams *Handler) {
   uint8_t buf[20]; // Buffer-Variable
   uint16_t bytesRead = 20; //Anzahl an Bytes die max. gelesen werden
   if (Handler->handle == char1.getValueAttribute().getHandle()) {
      //Einlesen der Daten
      ble.readCharacteristicValue(
         char1.getValueAttribute().getHandle(),
         buf, &bytesRead);
      if (buf[0] == 0x01) {
         if (buf[1] == 0x01) digitalWrite(pinLED, HIGH); // LED ein
         else digitalWrite(pinLED, LOW); // LED aus
      }
   }
}
void setup(){
   ble.onDataWritten(gattServerWriteCallBack);
}
```

3.4.3 Funktionstest mit nRF Connect App

Um nun die Funktion des Prototyps zu testen, wird die App „nRF Connect“ auf einem Samsung Galaxy S8 verwendet. Die App ist sowohl für Android als auch für iOS Geräte im entsprechenden App Store[17] kostenlos erhältlich.

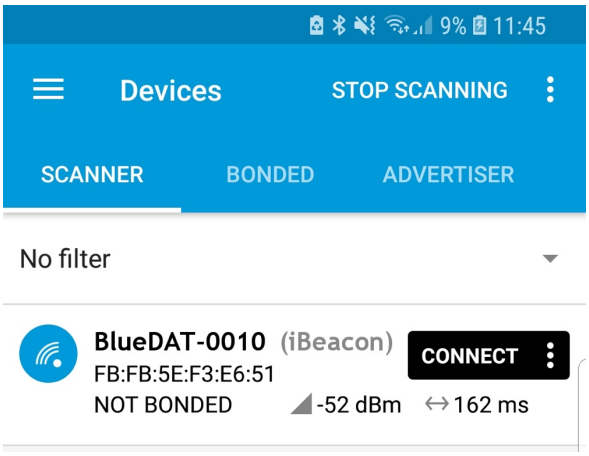

Abbildung 20: Anzeige der gefundenen Bluetooth-Geräte in der nRF Connect Applikation

[17] Download iOS: https://itunes.apple.com/de/app/nrf-connect/id1054362403
Download Android: https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

Beim Öffnen der App wird direkt nach allen verfügbaren Bluetooth-Geräten in der Umgebung gesucht und die gefundenen Geräte werden in einer Liste entsprechend angezeigt. Wie man in Abbildung 20 sieht, wird der Beacon als iBeacon richtig erkannt. Es wird auch das eingestellte Advertising-Intervall von 160ms (geringe Schwankung – 162ms) und der definierte Name (BlueDAT-0010) angezeigt.

Klickt man auf Connect, wird eine Verbindung mit dem Beacon aufgebaut und die vorhandenen Bluetooth-Services abgerufen und angezeigt. In Abbildung 21 sind die drei Services des Prototyps aufgelistet. Bei den ersten beiden (Generic Access – 0x1800, Generic Attribute – 0x1801) handelt es sich um Standard Services des Bluetooth Protokolls. Mit Hilfe des „Generic Access“ - Service kann zum Beispiel der Name des Gerätes ausgelesen oder geändert werden. Als dritter Punkt in der Liste wird, dass selbst erstellte Service mit der gewählten UUID angezeigt.

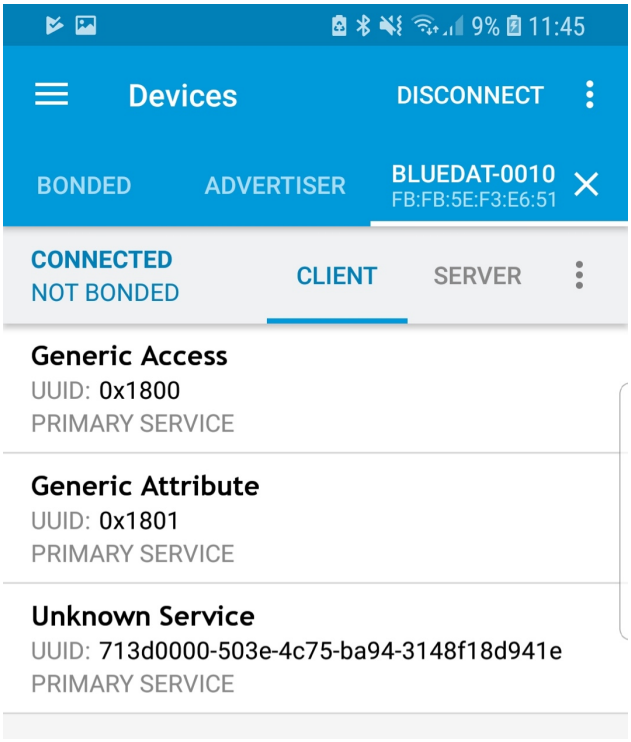

Abbildung 21: Anzeige der verfügbaren Bluetooth-Services vom Beacon-Prototyp in der nRF Connect Applikation

Wählt man ein Service aus, dann erscheinen alle dazugehörigen Charakteristiken. Im Falle des „Unknown Service“ ist das die definierte „WRITE NO RESPONSE“ Charakteristik, bei der die selbst gewählte UUID angezeigt wird (siehe Abbildung 22).

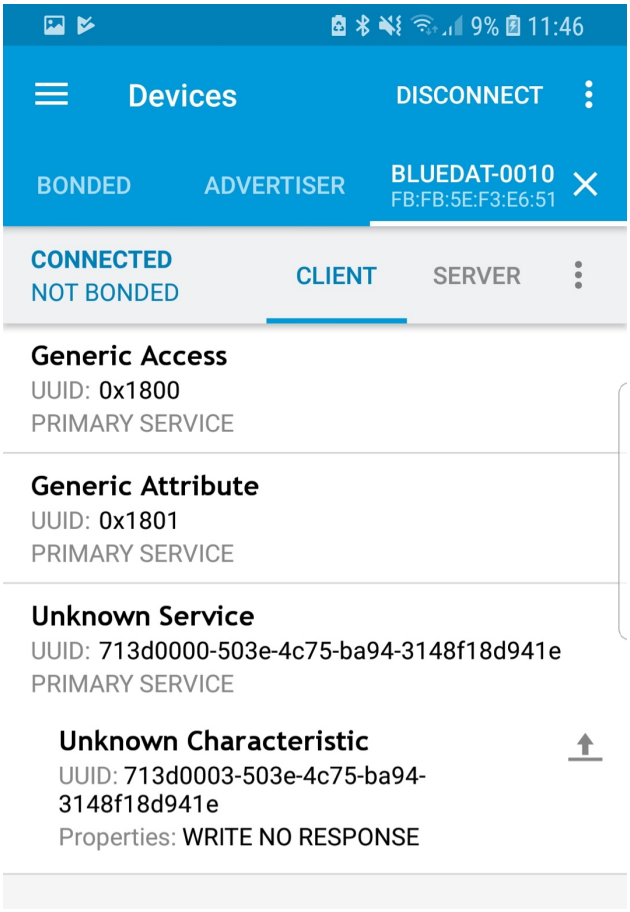

Abbildung 22: Anzeige der verfügbaren Charakteristiken vom selbst definierten Service in der nRF Connect Applikation

Klickt man auf den Pfeil nach oben (rechts neben der Charakteristik), öffnet sich ein Dialog-Fenster und man kann Daten an das Bluetooth-Modul senden. Um nun die LED am Prototyp einzuschalten muss man den HEX-Wert 0x0101 senden, um sie wieder auszuschalten den HEX-Wert 0x0100. Diese Werte wurden in der Programmierung so definiert, können aber nach Belieben angepasst werden. In Abbildung 23 sieht man das Dialogfenster mit dem vom Autor bereits zwischengespeicherten Werten welche nur noch ausgewählt werden müssen.

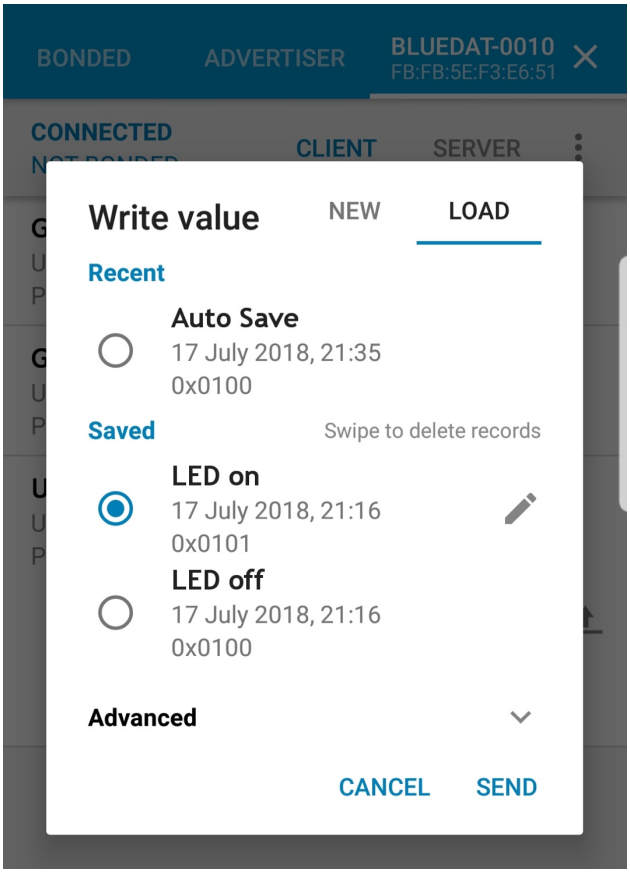

Abbildung 23: Dialogfenster "Wert senden" in der nRF Connect Applikation

Ist der Prototyp mit dem PC verbunden und der Serial Monitor der Arduino IDE geöffnet, so kann man die gesendeten Daten nachverfolgen. In Abbildung 24 sieht man die empfangenen Daten (LED ein bzw. LED aus) und den Neustart des Advertising-Prozesses sobald die Verbindung getrennt wurde.

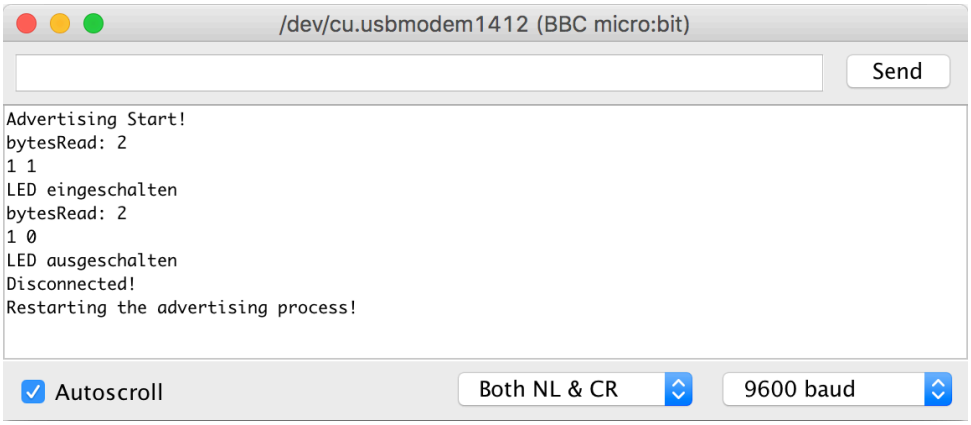

Abbildung 24: Ausgabe der Daten vom Beacon-Prototyp am Serial Monitor

3.5 Umsetzung BlueDAT Beacon

Nach den ersten Erfahrungen mit Bluetooth 5 und dem RedBear Nano v2 Board wird der BlueDAT Beacon umgesetzt. Der Begriff BlueDAT ist der Arbeitsname des parallel gestarteten Forschungsprojekts in der Forschungsgruppe Digital Technologies an der FH St.Pölten. BlueDAT ist die Abkürzung für Bluetooth Digital Asset Tracking und beschreibt das gesamte Projekt sehr gut.

In den folgenden Kapiteln sind die einzelnen Schritte (Aufbau der Schaltung, Programmierung des Beacons, Entwerfen und Drucken des Gehäuses, Fertigstellung) in der Entwicklung des BlueDAT Beacons beschrieben.

3.5.1 Aufbau der Schaltung

Aus den Schaltungen die in Kapitel 3.4 und in Kapitel 3.3.6.1 verwendet werden, wird eine vollständige Schaltung für den ersten BlueDAT-Prototyp entworfen. Zuerst werden Skizzen mit Hilfe des Programms Fritzing angefertigt um einen besseren Überblick über den gesamten Aufbau und der Verkabelung zu erhalten.

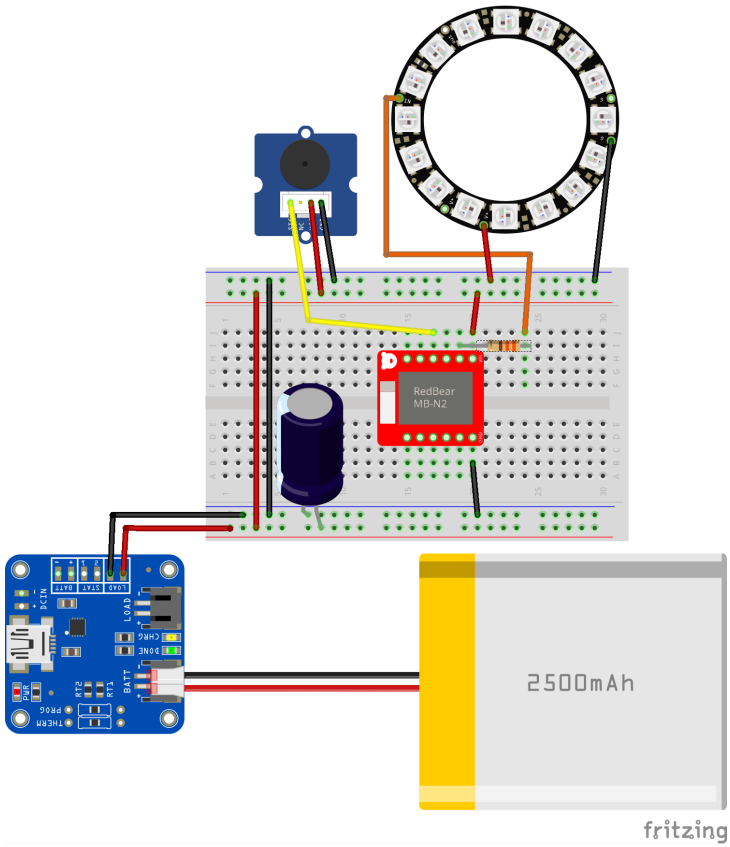

Abbildung 25: Fritzing Skizze des BlueDAT-Prototyps ohne Sensoren

Beim ersten Prototyp (siehe Abbildung 25) wurde auf jegliche Sensoren verzichtet, da diese im ersten Schritt nicht notwendig waren. Es wurde lediglich die für das Forschungsprojekt benötigten Signalgeber (optisch und akustisch) verbaut. Da der Prototyp aber erweiterbar sein soll, wurde in einem zweiten Schritt die Skizze um zwei Sensoren erweitert. In Abbildung 26 sieht man den veränderten Aufbau inklusive der Verkabelung eines Beschleunigungssensors und eines Umgebungslichtsensors.

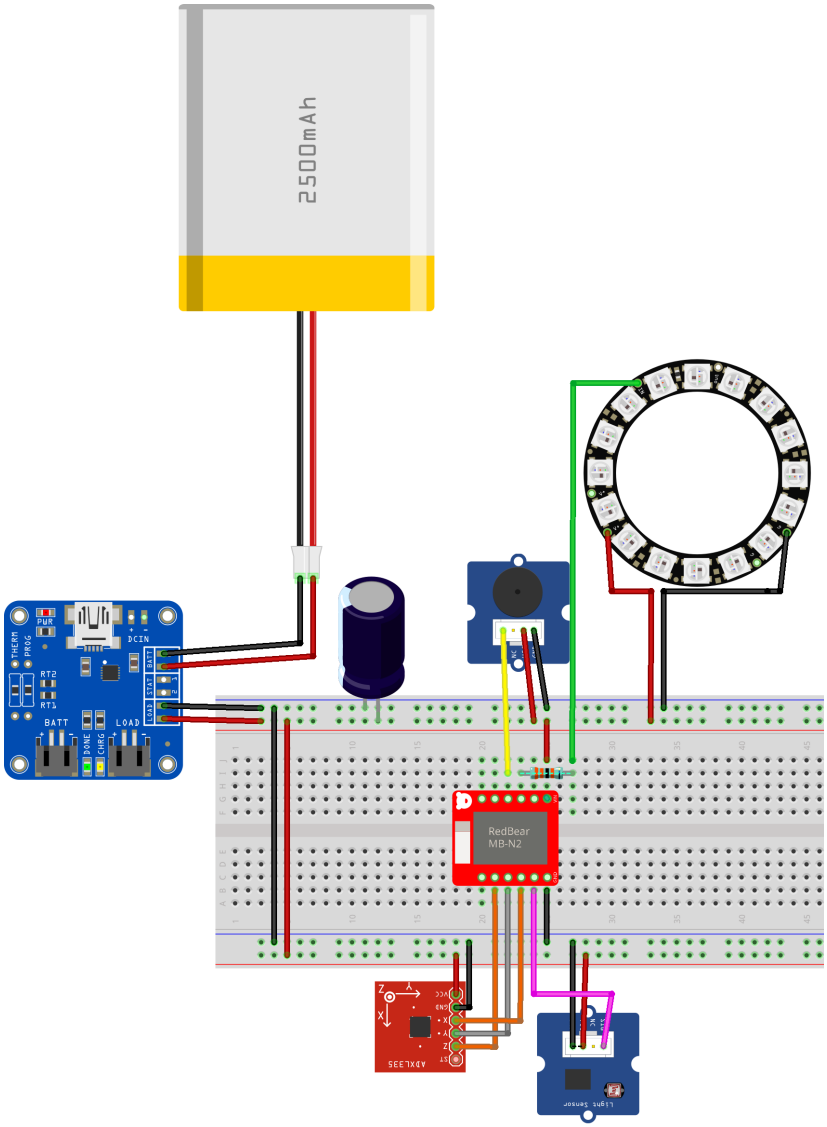

Abbildung 26: Fritzing Skizze des BlueDAT-Prototyps mit zwei Sensoren

Da in Punkt 3.3.3 eine Echtzeituhr als mögliche Hardware beschrieben wird, wurde eine weitere Fritzing Skizze angefertigt. Da die Echtzeituhr über den I2C-Bus an einen Mikrocontroller angeschlossen werden muss, welcher beim

RedBear Nano v2 nicht ausgeführt ist, wurde für diesen Aufbau das RedBear Blend v2 Board verwendet. Wie man in Abbildung 27 erkennen kann, wurde bei dieser Skizze der Akku sowie das Ladegerät nicht verwendet, da dieser Aufbau nur für den Test der mobilen Applikation verwendet wird. Die Energieversorgung wird dabei mit einem Micro USB-Kabel am RedBear Blend v2 durchgeführt.

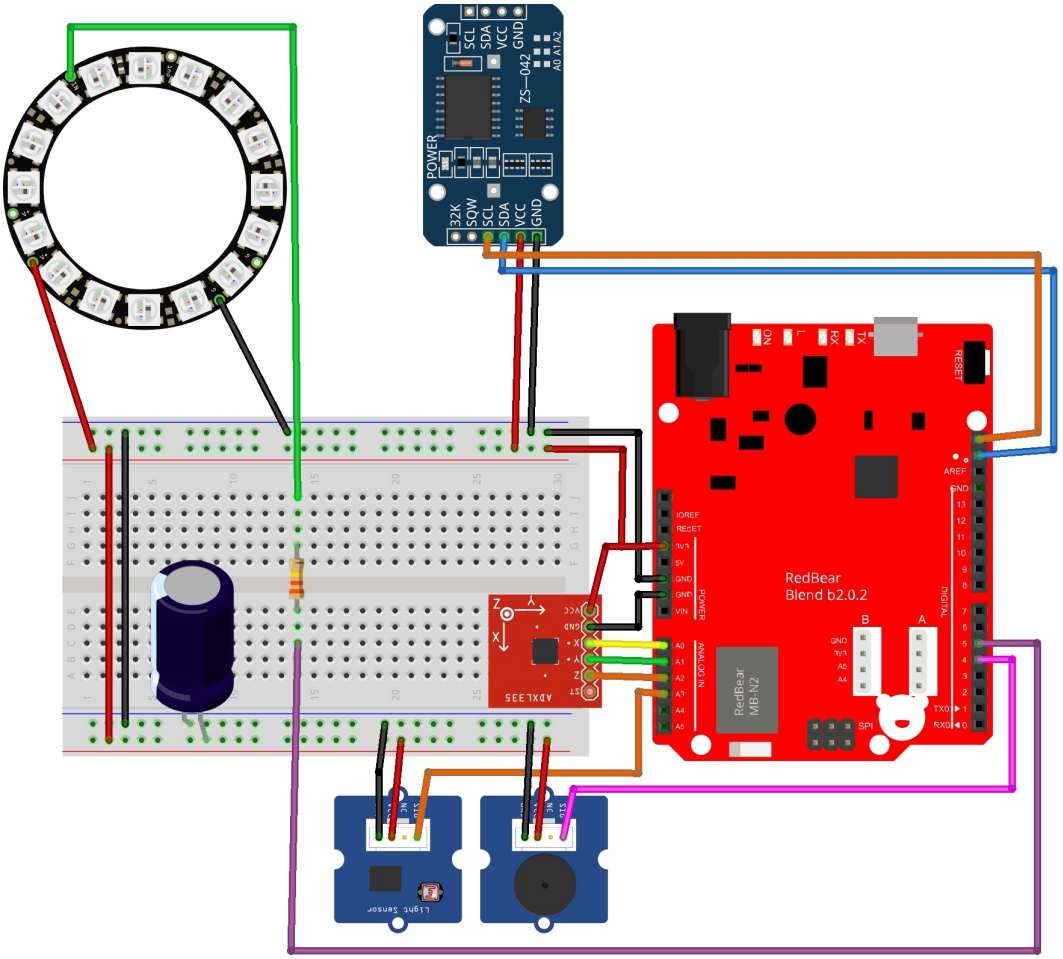

Abbildung 27: Fritzing Skizze des Aufbaus mit dem RedBear Blend v2 Board und einem Echtzeituhrmodul

Grundsätzlich werden alle Sensoren und Signalgeber mit 3,3 Volt über die Anschlüsse VCC und GND versorgt. Die verwendeten Pins sind für alle Sensoren und Mikrocontroller (MCU) gleich und es ergibt sich somit die folgende Pinbelegung (siehe Tabelle 10).

Tabelle 10: Pinbelegung des BlueDAT Prototypen

Pin am MCU	Pin am Bauteil	Bauteil
D4 (=Digitaler Pin 4)	SIG	Akustischer Signalgeber (Grove Buzzer)
D5	DIN	Optischer Signalgeber (NeoPixel Ring)
A0 (= Analoger Pin 0)	X_OUT	Beschleunigungssensor
A1	Y_OUT	Beschleunigungssensor
A2	Z_OUT	Beschleunigungssensor
A3	SIG	Umgebungslichtsensor

Die Definition und Initialisierung der digitalen und analogen Ein-/Ausgänge kann somit für alle Arduino-Sketches einheitlich durchgeführt werden.

Mit Hilfe der Skizze des BlueDAT-Prototyps mit zwei Sensoren (vgl. Abbildung 26) wird eine Platine entworfen. Die Platine soll so aufgebaut werden, dass so viele Einzelteile wie möglich leicht angeschlossen werden können, um so mögliche Änderungen oder Erweiterungen einfacher durchführen zu können. Daher werden für den Umgebungslichtsensor und den akustischen Signalgeber Grove Buchsen angebracht, da diese Bauteile bereits das 4-polige Grove System verwenden. Der optische Signalgeber wird auf das Grove System umgerüstet und auch über eine Grove Buchse angeschlossen. Da für den Beschleunigungssensor und das RedBear Nano v2 Board mehr als 4 Anschlüsse notwendig sind, werden dafür entsprechende Steckleisten montiert. Der Kondensator und der Widerstand, welche für den NeoPixel Ring benötigt werden, werden fix angelötet. Für die Spannungsversorgung wird eine 2-polige Leiterplattenklemme verwendet. Somit ist es möglich alle Sensoren und Signalgeber auszutauschen und den Mikrocontroller für eine etwaige Abänderung der Programmierung von der Platine zu entfernen.

Mit diesen Anforderungen wurde ein Entwurf einer Platine entwickelt und in zwei Änderungsdurchgängen optimiert. In Abbildung 28 sieht man den Entwurf der Lochrasterplatine mit den 3 Grove Buchsen (A, B und C), der Leiterplattenklemme (+ und -), den Steckleisten für Mikrocontroller (MCU) und Beschleunigungssensor (BS) sowie den fix verbauten elektronischen Bauteilen (Kondensator - C1 und den 330 Ohm Widerstand - R1).

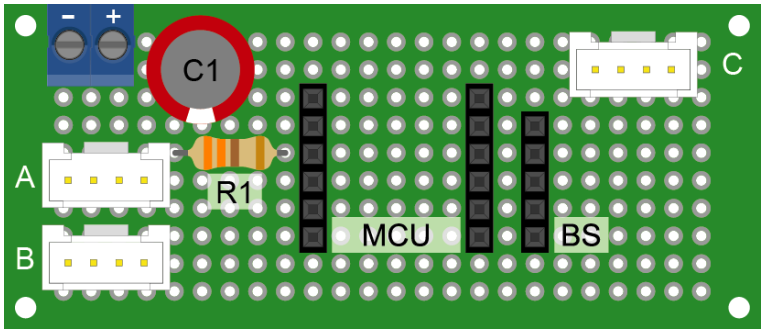

Abbildung 28: Entwurf der Lochrasterplatine mit den Grove Buchsen, den Steckerleisten und den Klemmen (Vorderseite)

In Abbildung 29 sieht man die Verkabelung auf der Rückseite der Lochrasterplatine. Die Punkte, an denen das entsprechende Kabel einen Kontakt mit einem Bauteil auf der Vorderseite hat, wurden mit einem farbigen Kreis gekennzeichnet. Die roten und schwarzen Verbindungen sind die Spannungsversorgung (+ und -). Die bunten Verbindungen sind die Daten-Anschlüsse für die Sensoren und die Signalgeber.

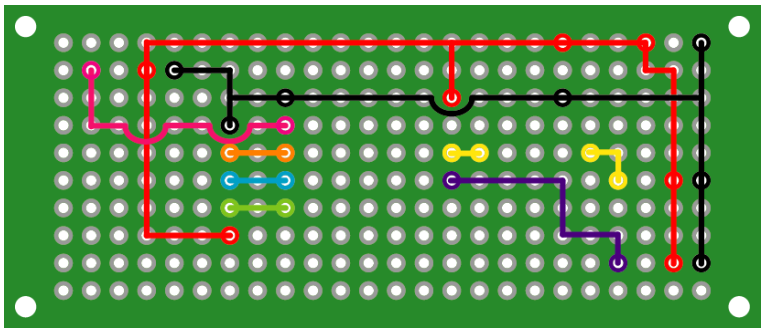

Abbildung 29: Entwurf der Lochrasterplatine – Verkabelung (Rückseite)

Zur Veranschaulichung der Verbindungen zu den Bauteilen wurde eine kombinierte Grafik angefertigt (vgl. Abbildung 30).

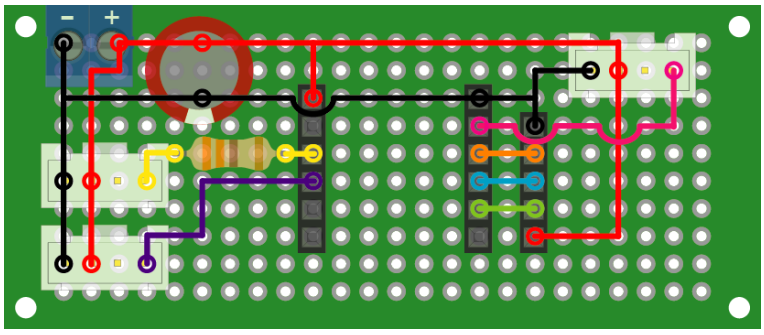

Abbildung 30: Entwurf der Lochrasterplatine – Bauteile mit Verkabelung (Vorderseite)

Mit Hilfe dieser Skizzen wurden insgesamt drei vollständige Lochrasterplatinen angefertigt. In Abbildung 31 sieht man die Vorder- und Rückseite einer fertig gelöteten Lochrasterplatine.

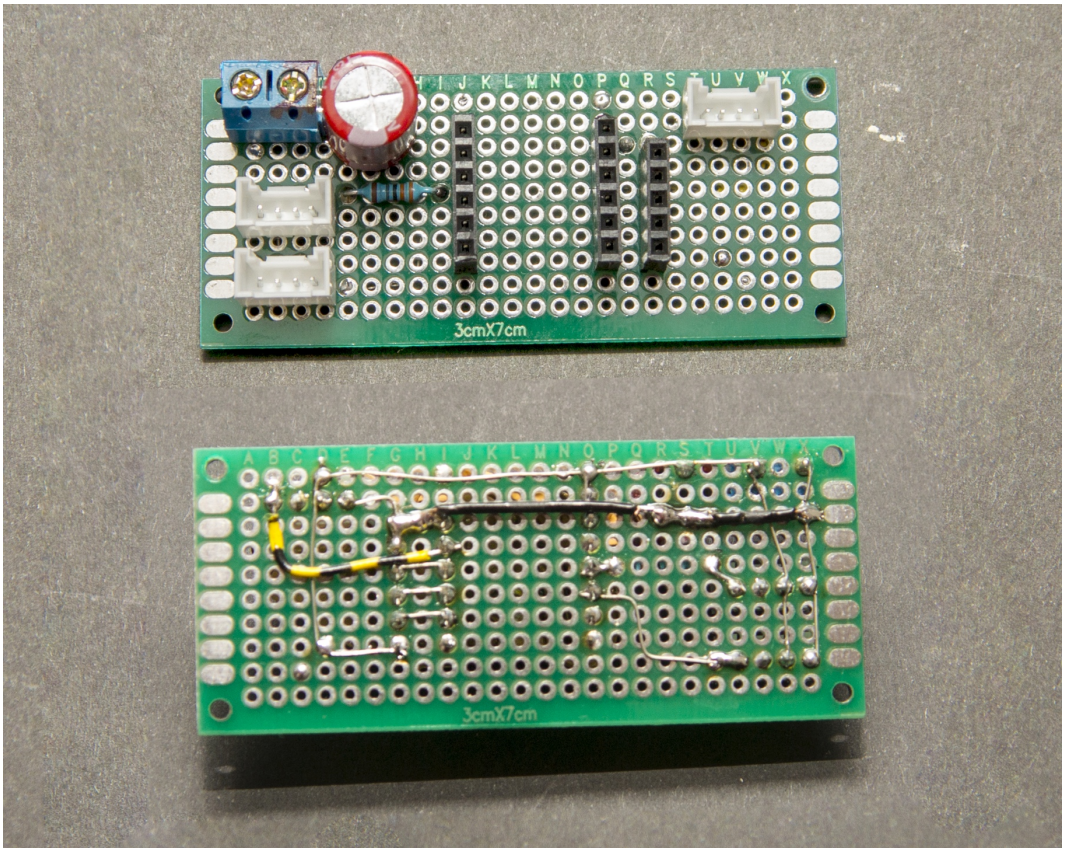

Abbildung 31: Fertig gelötete Lochrasterplatine des BlueDAT Beacons

3.5.2 Entwicklung und Herstellung des Gehäuses

Um den Prototypen auch ein entsprechendes Aussehen zu geben, wird ein 3D Gehäuse entwickelt. Das Gehäuse soll im Rahmen des Forschungsprojektes so designt werden, dass es in ein Transportgestell eingesteckt werden kann. Dieses Transportgestell besteht aus Eisen-Formrohren, welche Größe diese haben war aber zurzeit der Umsetzung leider nicht bekannt. Da die genaue Größe aber auch für den Prototyp nicht unbedingt notwendig ist, wurde angenommen, dass die Formrohre Innenmaße von 8 x 8 cm haben. Mit diesen Angaben und den Anforderungen des Beacons, wurde eine Skizze für das Gehäuse erstellt, welche anschließend in ein 3D Modell (siehe Abbildung 32) umgewandelt wurde. Das Gehäuse ist so konstruiert das es an die Maße des Formrohres in einem späteren Schritt genau angepasst werden kann und trotzdem noch genug Platz für die Hardware bleibt.

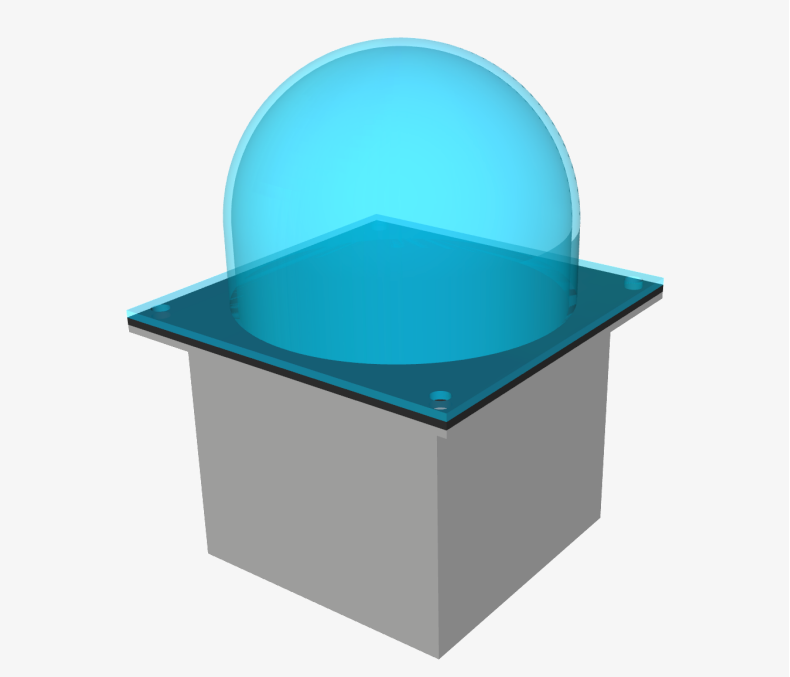

Abbildung 32: 3D-Entwurf des Gehäuses für den BlueDAT-Prototyp

Um der optischen Signaleinheit eine möglichst hohe Erkennbarkeit zu geben wurde eine Kuppel vorgesehen. Für diese wird im 3D-Druck ein transparentes und diffuses Material verwendet, welches die Lichtstrahlen bricht und somit für eine bessere Erkennbarkeit sorgt.

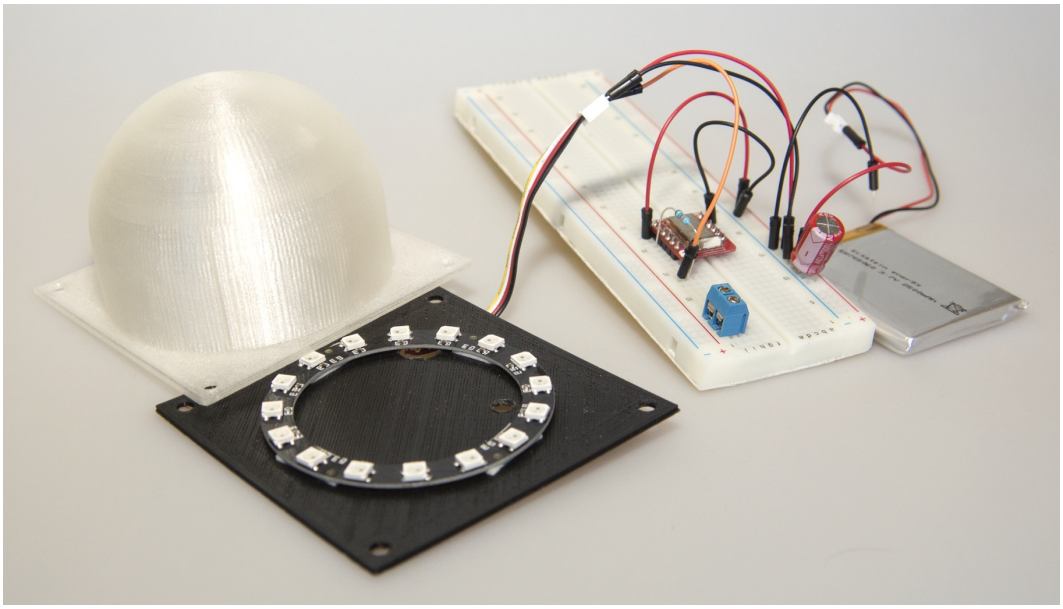

Abbildung 33: Kuppel und Mittelteil des Gehäuses als 3D-Druck mit Testaufbau und LED-Ring

Im ersten Schritt wurde nur die Kuppel und das Mittelstück mit Hilfe eines 3D Druckers angefertigt, um die Eigenschaften des diffusen 3D-Druck Materials für die Kuppel zu testen. Dafür wurde der Testaufbau für den NeoPixel Ring (laut Kapitel 3.3.6.1) verwendet. Auf Abbildung 33 sieht man die transparente Kuppel und das schwarze Mittelstück inklusive dem Testaufbau.

Abbildung 34: Transparente Kuppel in vier Farben beleuchtet (Rot, Grün, Blau und Weiß)

Durch die verschiedenen Lichtformen und Farben wurden die Eigenschaften des transparenten Materials getestet. In Abbildung 25 sieht man die Kuppel in vier verschiedenen Farben beleuchtet. Es sind durch die dreifärbigen LEDs natürlich die verschiedensten Farben mischbar, in diesem Test beschränkte man sich aber auf die Grundfarben Rot, Grün, Blau und Weiß.

Nachdem dieser Test zufriedenstellend ausgefallen ist, wurde das Gehäuse noch etwas angepasst und anschließend drei Stück inklusive dem Hauptteil angefertigt. Das fertige Gehäuse ist auf Abbildung 35 zu sehen.

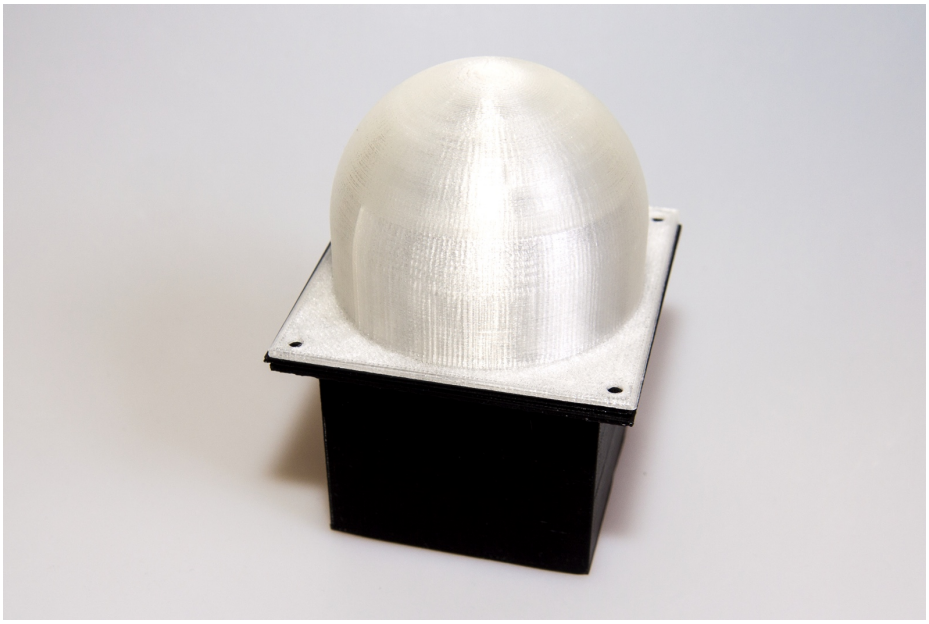

Abbildung 35: Eines der drei ausgedruckten Gehäuse für den Prototyp

3.5.3 Programmierung ohne Sensoren

Als Grundlage für die Programmierung wird der Arduino-Sketch verwendet, welcher bereits beim ersten Prototyp zum Einsatz gekommen ist (vgl. Kapitel 3.4). Dort war aber nur eine einseitige Kommunikation möglich, das heißt es konnten nur Daten vom Smartphone zum Beacon gesendet werden. Im Gegensatz dazu soll es nun auch möglich sein, dass der Beacon Daten an das verbunden Smartphone senden kann. Dafür muss eine zusätzliche Charakteristik zum bereits bestehenden Service hinzugefügt. Es handelt sich dabei um eine READ Charakteristik, welche es ermöglicht, dass ein Central-Gerät Daten vom Beacon anfordern kann.

Hinzufügen einer zweiten Charakteristik

In Listing 13 sieht man die beiden Programmzeilen die dafür hinzugefügt (Zeile 1) bzw. geändert (Zeile 2) werden müssen.

Listing 15: Hinzufügen der zusätzlichen READ-Charakteristik

```
GattCharacteristic  char2(charReadUUID, readValue, 6,
sizeof(readValue),
GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_READ);
GattCharacteristic *chars[] = {&char1, &char2};
```

Die Charakteristik wird verwendet um die aktuellen Einstellungen des Beacons auszulesen und in der mobilen Applikation anzuzeigen. Für den Prototyp werden insgesamt 6 verschiedene Konfigurationsmöglichkeiten bereitgestellt:

- Art des optischen Signals
- Art des akustischen Signals
- Dauer des Signals
- Farbe des optischen Signals
- Sendeleistung des Bluetooth-Moduls
- Sendeintervall des Bluetooth-Moduls

Definition der Konfigurationsvariablen

In Listing 16 sieht man die Definition aller Werte sowie das Speichern in die Read-Variable, welche über die Charakteristik ausgelesen werden kann.

Listing 16: Einstellungen des Beacons welche vom Smartphone ausgelesen werden können

```
enum typesOfOpticalSignal typeOpticalSignal = DAUERLICHT;
enum typesOfAcousticSignal typeAcousticSignal = DAUERTON;
int intervalSignal = 7000;
enum colors colorOpticalSignal = WEISS;
enum txPower powerBeaconSignal = PLUS_4_DB;
```

```
int intervalBeaconSignal = 16; //von 10 bis 1000ms in 10 Schritten,
Wert mal 10


uint8_t readValue[20] = {
  (uint8_t) typeOpticalSignal,
  (uint8_t) typeAcousticSignal,
  (uint8_t) (intervalSignal/1000),
  (uint8_t) colorOpticalSignal,
  (uint8_t) powerBeaconSignal,
  (uint8_t) intervalBeaconSignal
};
```

Ändern sich die Werte muss die Charakteristik upgedatet werden, das funktioniert mit der Funktion „updateCharacteristicValue“ (vgl. Listing 17).

Listing 17: Updatefunktion für die Werte einer Charakteristik

```
ble.updateCharacteristicValue(char2.getValueAttribute().getHandle(),
readValue, 6, true);
```

Hinzufügen der Service UUID zum Advertising Payload

Um beim Scannen nach Bluetooth-Geräten in einer mobilen Applikation zwischen verschiedenen Geräten unterscheiden zu können gibt es einige Möglichkeiten. Es könnte zum Beispiel die physikalische Adresse oder Name des Bluetooth-Moduls verwendet werden, im Falles des Prototyps könnte auch die Beacon-UUID dazu genutzt werden. Jedoch ist es für die Verwendung der BlueDAT-Applikation notwendig das ein Bluetooth-Gerät das entsprechende Bluetooth-Service zur Verfügung stellt. Dazu muss die definierte UUID für das benutzerdefinierte Service dem Advertising Payload oder dem Scan-Response Payload hinzugefügt werden. Dann kann in der mobilen Applikation eindeutig unterschieden werden, ob es sich bei den gefundenen Geräten um einen BlueDAT Beacon handelt oder nicht. In Listing 18 sieht man die Funktion welche in der Setup-Methode hinzugefügt werden muss, damit die Service UUID mitgesendet wird.

Listing 18: Hinzufügen der UUID des Services zum Advertising Payload

```
ble.accumulateAdvertisingPayload(
   GapAdvertisingData::COMPLETE_LIST_128BIT_SERVICE_IDS,
   (const uint8_t *) reversedBaseUUID, sizeof(reversedBaseUUID));
```

Laut der Definition von Bluetooth 5 (vgl. Kapitel 2.5.4) soll es möglich sein 255 Byte an Daten im Advertising Payload zu senden. Jedoch wurde beim laufenden Testen des BlueDAT-Beacons mit der nRF Connect Applikation festgestellt das die UUID, sowie in Listing 18 hinzugefügt, nicht übertragen wird. Bei der Definition der Funktion „accumulateAdvertisingPayload“ in der verwendeten Bibliothek, fand der Autor dann den Hinweis das nur 31 Byte (wie bei Bluetooth

4) versendet werden können. Daher wurde die UUID im Scan-Response versendet. Dabei musst aber der Name um zwei Zeichen gekürzt werden, da auch hier die Kapazitätsgrenze erreicht war (vgl. Listing 19).

Listing 19: Erstellen des neuen Scan-Response Payload

```
const static char deviceName[] = "BlueDAT-10";

void setup(){
   ...
   ble.accumulateScanResponse(
      GapAdvertisingData::COMPLETE_LIST_128BIT_SERVICE_IDS,
      (const uint8_t *)reversedBaseUUID, sizeof(reversedBaseUUID)
   );
   ble.accumulateScanResponse(
      GapAdvertisingData::COMPLETE_LOCAL_NAME,
      (uint8_t *)deviceName,
      sizeof(deviceName)
   );
   ...
}
```

Ob das Problem bei der verwendeten Bibliothek liegt oder ob es mit einer anderen Programmierumgebung (z.B.: Segger Embedded Studio) vielleicht funktioniert, konnte aus Zeitgründen leider nicht mehr getestet werden. Man sieht aber sehr gut, dass eine höhere Kapazität des Advertising- bzw. auch des Scan-Response Payloads sehr sinnvoll wäre.

Die weiteren Programmzeilen des verwendeten Arduino-Sketches werden nicht weiter behandelt, da es sich um einfach zu verstehende Funktionen für die Steuerung des optischen oder akustischen Signalgebers, die Speicherung der Einstellungen und einige Hilfsfunktionen handelt.

3.5.4 Erweiterte Programmierung mit Sensoren

Zum Testen der mobilen Applikation wird, wie in Punkt 3.5.1 bereits erwähnt, ein weiterer Prototyp mit dem RedBear Blend v2 Board und drei Sensoren erstellt. Der Arduino-Sketch welcher im vorigen Kapitel beschrieben wird, kann 1:1 weiterverwendet werden. Deshalb werden in den folgenden Absätzen nur die wichtigsten Erweiterungen beschrieben.

Hinzufügen einer dritten Charakteristik

Damit die Werte der Sensoren in der mobilen Applikation regelmäßig aktualisiert werden, ist es notwendig eine dritte Bluetooth Charakteristik zu erstellen. Es wird eine NOTIFY-Charakteristik erstellt, welche von einem Smartphone „abonniert" werden kann, um Werteänderungen direkt zu erhalten.

In Listing 20 sieht man die beiden Programmzeilen die dafür hinzugefügt (Zeile 1) bzw. geändert (Zeile 2) werden müssen.

Listing 20: Hinzufügen der zusätzlichen NOTIFY-Charakteristik

```
GattCharacteristic  char3(charNotifyUUID,  notifyValue, 5,
sizeof(notifyValue),
GattCharacteristic::BLE_GATT_CHAR_PROPERTIES_NOTIFY);
GattCharacteristic *chars[] = {&char1, &char2, &char3};
```

Speichern der Sensorwerte in die Charakteristik

Die Werte der einzelnen Sensoren werden regelmäßig in Abständen von einer Sekunde ausgelesen und in die Charakteristik gespeichert. Wie das Auslesen der benötigten Werte funktioniert wurde bereits in den Punkten 3.3.2 und 3.3.3 für die beiden Sensoren und die Echtzeituhr erklärt.

Für das regelmäßige Senden der Werte wird ein „Ticker" benötigt, welcher die Funktion „periodicCallback" in dem Fall jede Sekunde ausführt (vgl. Listing 21). Durch die IF-Abfrage wird sichergestellt, dass die Werte der Charakteristik nur dann upgedatet werden, wenn der Beacon mit einem Central verbunden ist. Denn auch nur dann ist es notwendig, dass die aktuellen Werte vorhanden sind. In der Variable „notifyValue" sind alle Werte der Sensoren abgespeichert.

Listing 21: Regelmäßiges Update der NOTIFY-Charakteristik

```
Ticker tickerNotify;
void periodicCallback() {
   if (ble.getGapState().connected) {
      ble.updateCharacteristicValue(
         char3.getValueAttribute().getHandle(),
         notifyValue, sizeof(notifyValue));
   }
}

void setup() {
   ...
   tickerNotify.attach(periodicCallback, 1);
   ...
}
```

3.5.5 Fazit BlueDAT Beacon

3.5.5.1 BlueDAT Prototyp mit dem RedBear Nano v2

Beim Prototypen welcher für das Forschungsprojekt entwickelt und in dreifacher Ausführung angefertigt wurde, funktionieren alle Bauteile wie vorgesehen. Die Platine wurde so angefertigt, dass die in Kapitel 3.3.2 erwähnten Sensoren für die Beschleunigung und das Umgebungslicht direkt angeschlossen werden können. Zudem ist auf der Platine noch genügend Platz um ein oder zwei weitere Bauteile (Sensoren, Buttons, ...) zu verbauen. Das entwickelte Gehäuse erfüllt vorerst seinen Zweck und verleiht dem optischen Signalgeber eine sehr gute Erkennbarkeit.

3.5.5.2 Sensoren

Beim erweiterten Prototyp, welcher nur zum Testen der mobilen Applikation verwendet wird, ist zu erwähnen das die beiden Sensoren nicht so einfach verwendet werden können. Beim Beschleunigungssensor, welcher vorerst nur als Lagesensor eingesetzt wird, ist ein großer Nachteil die aufwendige Kalibrierung. Diese muss mit jedem Sensor einzeln und manuell durchgeführt werden, das kann dazu führen, dass die Werte nicht hundertprozentig genau sind. Auch der Umgebungslichtsensor liefert nur schwer zu interpretierbare Werte. Eine Unterscheidung zwischen Tag und Nacht ist aber relativ einfach möglich. Das Echtzeituhrmodul hingegen macht genau das was es soll. Es liefert die aktuelle Uhrzeit, das aktuelle Datum und auch den aktuellen Wochentag. Alle drei Werte können zur Konfiguration des Beacons verwendet werden. Jedoch kann dieses nicht mit dem RedBear Nano v2 Prototyp verwendet werden. Für die vorgesehenen Tests des BlueDAT Prototyps ist das allerdings vollkommen ausreichend.

3.5.5.3 Programmierung

Die Programmierung mit der Arduino IDE war bis auf einige kleine Schwierigkeiten relativ einfach. Jedoch könnte vermutlich mit dem Segger Embedded Studio und dem Nordic SDK (= Software Development Kit) mehr aus dem Bluetooth-Modul herausgeholt werden. Da die IDE dem Autor jedoch vollkommen unbekannt war und erste Programmierversuche scheiterten wurde auf eine weitere Verwendung verzichtet.

3.6 Umsetzung Mobile Applikation

3.6.1 Funktionsübersicht

Aus den Funktionen des Beacons (siehe Kapitel 3.2.8) sowie den Anforderungen für das geplante System, wurden die Details für die Mobile Applikation erarbeitet.

Funktionen für den Prototyp mit dem RedBear Nano v2:

- Anzeigen aller Beacons in Reichweite
- Direktes Auslösen des optischen oder akustischen Signalgebers
- Einstellen der Art des optischen Signalgebers (Dauerlicht, Blitzlicht, ...)
- Einstellen der Art des akustischen Signalgebers (Dauerton, Piepston, ...)
- Einstellen der Dauer des Signals der beiden Signalgeber
- Konfigurationsmöglichkeit für Sendeleistung und Sendeintervall

Erweiterte Funktionen für den Prototyp mit dem RedBear Blend v2:

- Einlesen und Anzeigen des Wertes vom Umgebungslichtsensor
- Einlesen und Anzeigen der aktuellen Uhrzeit des Beacons
- Konfigurationsmöglichkeit für die Sendezeiten

Eine weitere nützliche Funktion wäre das Anzeigen des Akkustands. Jedoch ist eine Akkumessung nicht so leicht umzusetzen, weshalb im ersten Prototyp darauf verzichtet wird.

3.6.2 Entwurf der Benutzeroberfläche

Die in Kapitel 3.6.1 aufgezählten Funktionen der App wurden in einfachen handgezeichneten Wireframes zusammengefasst. Somit hat man einen einfachen Überblick welche Screens die Mobile Applikation schlussendlich aufweisen soll. Anschließend wurden die Wireframes in fertige Screendesigns umgesetzt. Die fertige mobile Applikation soll dann etwa diesen Designs entsprechen. Um viele verschiedene Elemente des Hybrid-App Frameworks zu verwenden wird zu Beginn gleich der Splashscreen entsprechend gestaltet. In Abbildung 36 sieht man zwei Versionen des Splashscreens mit dem ebenfalls für den Prototyp entworfenen BlueDAT Logo.

Abbildung 36: Splashscreen der mobilen Applikation in weiß (links) und blau (rechts)

Die App zeigt direkt nach dem Starten, alle Beacons in Reichweite an (vgl. Abbildung 37 links). Neben jeden Eintrag gibt es, falls es sich um einen BlueDAT Beacon handelt, drei Buttons. Mit Hilfe dieser Buttons kann entweder die akustische bzw. die optische Signaleinheit des Beacons ausgelöst werden oder man wechselt in die Einstellungen des Beacons.

Abbildung 37: Übersichtsseite aller Beacons in Reichweite (links) sowie den Signalgeber-Screen (rechts)

Zum Auslösen des optischen Signalgebers muss der blaue Button und zum Auslösen des akustischen Signalgebers muss der orange Button gedrückt werden. Löst man einen Signalgeber aus, so öffnet sich ein neuer Screen mit einem Button zum Wiederholen des Signals und einem Button zum Trennen der Verbindung zum Beacon (siehe Abbildung 37 rechts). Standardmäßig wird das Signal des Beacons für 5 Sekunden ausgelöst. Klickt man auf den Wiederholen-Button kann man das Signal für weitere 5 Sekunden auslösen. Die Dauer des Signals kann in den Einstellungen angepasst werden.

Klickt man auf den grauen Button kommt man auf die Konfigurationsseite des entsprechenden Beacons. Hier werden verschiedene Daten (Adresse, Uhrzeit, Wochentag, ...) angezeigt und es ist möglich die folgenden Einstellungen vorzunehmen (vgl. Abbildung 38):

- Dauer des Signals
- Art des optischen Signalgebers: Dauerlicht, Blitzlicht oder Drehlicht
- Farbe des optischen Signalgebers: Rot, Grün, Blau oder Weiß
- Art des akustischen Signalgebers: Dauerton oder Piepston
- Sendeleistung des Beacons (-40dB bis +4dB)
- Sendeintervall des Beacons (1 Millisekunde bis 1 Sekunde)

In weiteren Schritten könnte diese Einstellungsübersicht um Funktionen erweitert werden. Angedacht sind die Steuerung des Bewegungs- und Lichtsensors sowie die Einstellung der Uhrzeiten an denen der Beacon aktiv ist. Diese Funktionen werden vorerst aber nicht umgesetzt.

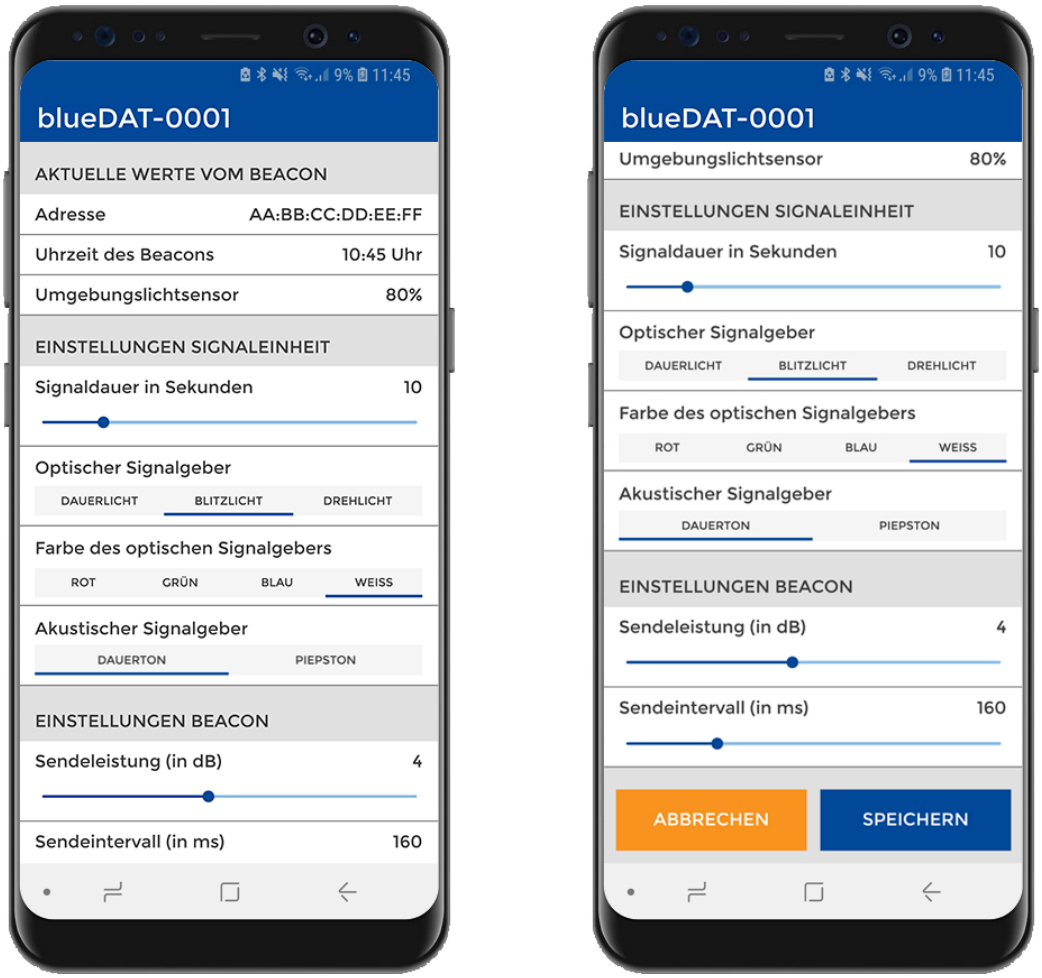

Abbildung 38: Beacon-Konfigurationsseite der mobilen Applikation

3.6.3 Programmierung

Für die Programmierung der Mobilen Applikation ist die Wahl auf das Hybrid Framework NativeScript gefallen (siehe Kapitel 2.7.4). Als Basis wurde die Demo-Applikation des NativeScript Bluetooth LE Plugins von Eddy Verbruggen[18] verwendet, welche zum Zeitpunkt des Programmierstarts noch in JavaScript erstellt war. Aktuell ist diese bereits in TypeScript umgewandelt worden.

Auf die Installation aller notwendiger Komponenten für das Framework wird hier nicht näher eingegangen, da es dafür eine ausführliche und gute Dokumentation auf der Website[19] von NativeScript gibt. Vom gesamten Programmcode werden nur einige wichtige Code-Schnipsel in Bezug auf die Bluetooth-Kommunikation näher erklärt. Der Rest sollte für Personen mit einem ähnlichen Know-how wie der Autor leicht verständlich sein.

Nur BlueDAT Prototypen anzeigen

Um die Bedienung der Applikation einfach zu halten, sollen in der Übersichtsliste nur BlueDAT Prototypen angezeigt werden. Bei der Funktion „bluetooth.startScanning“ ist es möglich Service UUIDs zu übergeben. Dann werden nur Geräte angezeigt, welche dieses Service zur Verfügung stellen. Dafür ist es notwendig das die Service UUID im Advertising oder Scan-Response Payload mitgesendet wird (vgl. Listing 18 auf Seite 77). Die UUID des benutzerdefinierten Service des BlueDAT Prototyp ist „713d0000-503e-4c75-ba94-3148f18d941e“ und wird der Funktion übergeben (siehe Listing 22). Alle weiteren Optionen und Funktionen sind in Listing 22 kommentiert.

Listing 22: startScanning-Funktion mit Übergabe der Service UUID

```
bluetooth.startScanning({
   serviceUUIDs: ["713d0000-503e-4c75-ba94-3148f18d941e"],
   seconds: 10, //Dauer des Scanvorgangs
   onDiscovered: function (peripheral) {
      /* Sobald ein Bluetooth-Gerät gefunden wurde, werden alle davon
         bekannten Daten in ein Array gespeichert */
      var periActual = observable.fromObject(peripheral);
      const index = observablePeripheralArray.indexOf(periActual);
         if(index == -1) observablePeripheralArray.push(periActual);
   }
}).then(function(){ console.log("Scanvorgang abgeschlossen"); },
   function (err) { console.log("Fehler beim Scanvorgang: " + err);
});
```

[18] https://github.com/EddyVerbruggen/nativescript-bluetooth

[19] https://docs.nativescript.org/

Scanvorgang regelmäßig durchführen

Wird die Funktion „bluetooth.startScanning“ einmal aufgerufen, scannt die Applikation für die Dauer des Scanvorgangs (vgl. Listing 22) nach Geräten und zeigt alle gefundenen BlueDAT-Geräte in einer Liste an. Bewegt sich in dieser Zeit jedoch ein BlueDAT-Beacon oder der Benutzer des Smartphones kann es sein, dass ein oder mehrere Geräte nicht mehr in Reichweite sind. Deshalb wird der Scanvorgang alle 12 Sekunden wiederholt, um die Liste so aktuell wie möglich zu halten. Dafür wird ein Intervall erstellt, welches den Scanvorgang neu startet (vgl. Listing 23). Die Zeiten für den Scanvorgang selbst und der Wiederholung des Scanvorgangs wurden vom Autor frei gewählt und können nach Belieben angepasst werden. Um die beste Einstellung zu finden, wäre es notwendig die Applikation einem User-Test zu unterziehen, welcher im Rahmen dieser Diplomarbeit aber zeitlich nicht möglich war.

Listing 23: Intervall für den Scanvorgang erstellen

```
scanInterval = setInterval(() => { startScanning(); }, 12000);
```

Auslösen des optischen bzw. akustischen Signalgebers

In der Programmierung des BlueDAT Beacons wurde ein GATT-Charakteristik mit der UUID „713d0003-503e-4c75-ba94-3148f18d941e“ angelegt und mit der Option „WRITE“ versehen. Weiters wurde festgelegt, dass wenn der Wert 0x01 oder der Wert 0x02 an die Charakteristik gesendet wird, entweder der optische oder der akustische Signalgeber für die eingestellte Signaldauer ausgelöst wird. In Listing 24 sieht man die dafür notwendige Funktion um den optischen Signalgeber auszulösen. Als Übergabeparameter werden die Geräteadresse des Bluetooth-Moduls, die UUID des Service, die UUID der entsprechenden Charakteristik und der zu sendende Wert benötigt.

Listing 24: Funktion für das Auslösen des optischen Signalgebers

```
function ledOn() {
  bluetooth.write({
    peripheralUUID: peri.UUID,  //Variable mit der Geräteadresse
    serviceUUID: "713d0000-503e-4c75-ba94-3148f18d941e",
    characteristicUUID: "713d0003-503e-4c75-ba94-3148f18d941e",
    value: "0x01"
  }).then(function (result) {
    console.log("Wert erfolgreich geschrieben");
   },
  function (errorMsg) {
    console.log("Fehler beim Schreiben des Wertes: " + errorMsg);
   });
}
```

Auslesen der Konfigurationsvariablen

Die Einstellungen des Beacons werden vereinfacht als Integer-Zahlen abgespeichert und können mit Hilfe der READ-Charakteristik mit der UUID 713D0002-503E-4C75-BA94-3148F18D941E ausgelesen werden. In Listing 25 ist die Funktion „bluetooth.read“ angeführt, welche als Übergabeparameter die Geräteadresse des Bluetooth-Moduls, die UUID des Service und die UUID der entsprechenden Charakteristik benötigt. Klappt das Lesen der Charakteristik, dann sind in der Variable „data“ alle Konfigurationsdaten gespeichert. Vergleiche dazu die Definition des Arrays in Listing 16 auf Seite 76.

Listing 25: Funktion zum Auslesen der Konfigurationsvariablen

```
function getSettingsFromBeacon(){
   bluetooth.read({
      peripheralUUID: peri.UUID,
      serviceUUID: "713d0000-503e-4c75-ba94-3148f18d941e",
      characteristicUUID: "713D0002-503E-4C75-BA94-3148F18D941E",
   }).then(function(result) {
      var data = new Uint8Array(result.value);
   }, function (err) {
         console.log("Fehler beim Lesen der Werte: " + err);
      }
   );
}
```

Speichern der Konfigurationsvariablen

Die Funktion ist ähnlich wie die beim Auslösen des Signalgebers (vgl. Listing 24), es ändert sich nur der Wert der geschrieben wird. Alle Werte werden nach dem Wert 0x04 als 1 Byte große HEX Zahl getrennt durch Beistriche an den die Variable „dataString“ angehängt. Durch den Anfangswert 0x04 wird definiert das es sich bei den nachfolgenden Werte um die Konfigurationsvariablen handelt. Damit wird sichergestellt, dass der BlueDAT Beacon die Daten richtig weiterverarbeitet. Wurden die Einstellungen erfolgreich gespeichert, wird die Funktion „disconnect“ aufgerufen und damit die Verbindung zum Beacon getrennt (vgl. Listing 26).

Listing 26: Funktion zum Speichern der Konfigurationsvariablen

```
function saveSettings() {
   ...
   var dataString = "0x04," + typeOpticalSignal + "," +
      typeAcousticSignal+ "," +intervalSignal+ "," +colorOpticalSignal+
      "," + powerBeaconSignal + "," + intervalBeaconSignal;
   bluetooth.write({
      peripheralUUID: peri.UUID,
      serviceUUID: "713d0000-503e-4c75-ba94-3148f18d941e",
```

```
      characteristicUUID: "713d0003-503e-4c75-ba94-3148f18d941e",
      value: dataString
   }).then(function (result) {
      console.log("Einstellungen erfolgreich gespeichert");
      disconnect();
   }, function (errorMsg) {
      console.log("Fehler beim Speicher: " + errorMsg);
   });
}
```

Auslesen der aktuellen Werte der Sensoren

Beim RedBear Blend v2 Prototyp wird eine zusätzliche NOTIFY-Charakteristik bereitgestellt, welche von der mobilen Applikation abonniert werden kann. Sobald die Charakteristik abonniert ist, werden die Werte der Sensoren regelmäßig ausgelesen. In Listing 27 ist die dafür zuständige Funktion „bluetooth.startNotifying“ inklusive der Callback-Funktion „onNotify“ angeführt. Sobald der BlueDAT Beacon den Wert der Charakteristik mit der UUID "713D0001-503E-4C75-BA94-3148F18D941E" ändert, bekommt die mobile Applikation eine Benachrichtigung und führt die Callback-Funktion aus. Ist alles erfolgreich, dann sind in der Variable „data“ alle Werte der Sensoren gespeichert und können weiterverarbeitet oder in der App angezeigt werden.

Listing 27: Funktion zum Abonnieren der Werteänderungen der Sensoren

```
function getActualDataFromBeacon(){
   bluetooth.startNotifying({
      peripheralUUID: peri.UUID,
      serviceUUID: "713d0000-503e-4c75-ba94-3148f18d941e",
      characteristicUUID: "713D0001-503E-4C75-BA94-3148F18D941E",
      onNotify: function (result) {
         var data = new Uint8Array(result.value);
      }
  }).then(function() {
      console.log("Charakteristik abonniert");
  });
}
```

In den letzten Absätzen wurden einige wichtige Programmteile beschrieben die für die mobile Applikation essentiell sind. Betrachtet man die einzelnen Code-Schnipsel genauer stellt man fest, dass es noch keine echte Fehlerbehandlung gibt. Mögliche Fehler werden zwar in der Konsole ausgegeben, aber dem Benutzer der App nicht angezeigt und auch nicht entsprechend darauf reagiert. Für die Tests im Rahmen dieser Diplomarbeit ist das ausreichend, sollte aber die App veröffentlicht oder von anderen Personen benutzt werden, muss eine Fehlerbehandlung integriert werden.

3.6.4 Verwendete Smartphones

Im Rahmen der Diplomarbeit wurde ein Apple iPhone X und ein Samsung Galaxy S8 für die Programmierung der Mobilen Applikation und die Tests des Prototyps verwendet. Laut den ersten Recherchen sollten beide Geräte bereits über ein Bluetooth 5 fähiges Modul verfügen, und somit die Neuerungen unterstützen.

Nach einer ausführlicheren Recherche wurde jedoch festgestellt, dass es sich bei den Modulen, wie auch bei den Mikrocontroller Modulen, nur um Bluetooth 5 Ready Module handelt. Diese unterstützen den neuen Standard nur eingeschränkt, unter anderem wird der Long Range Modus auf keinen der beiden Geräte unterstützt.

Test Bluetooth 5 Long Range Funktionalität

Um die Funktionalität der Smartphones in Bezug auf den Long Range Modus zu testen wurde vom Autor ein kurzer Test durchgeführt. Ein Nordic nRF52840 Development Kit wurde als einfaches Peripheriegerät konfiguriert, welches den physikalischen Layer „LE Coded“ und somit den Long Range Modus verwendet. Weiters wurde das Board so programmiert, das der physikalische Layer mit einem einfachen Button-Klick von LE Coded auf LE 1M geändert werden kann.

Auf den beiden Smartphones wurde die App nRF Connect installiert und nach Bluetooth-Geräten in der Nähe gesucht. Bei der Verwendung des PHY LE Coded erkannte keines der Smartphones das Board. Sobald der PHY LE 1M verwendet wurde, erkannten beide Geräte sehr schnell das Board und zeigten alle definierten Werte an.

Somit wurde die Vermutung bestätigt, dass es sich bei den verbauten Bluetooth-Modulen der Smartphones um Bluetooth 5 Module handelt, welche den Long Range Modus nicht unterstützen.

Anzeige der möglichen PHYs

Bei der Verwendung der nRF Connect App am Samsung Galaxy S8 wurde festgestellt, dass bei einem Verbindungsaufbau der bevorzugte physikalische Layer ausgewählt werden kann (siehe Abbildung 39). Wie man sieht, ist der PHY LE Coded ausgegraut, was wiederum ein weiteres Indiz dafür ist, dass der Long Range Modus nicht unterstützt wird.

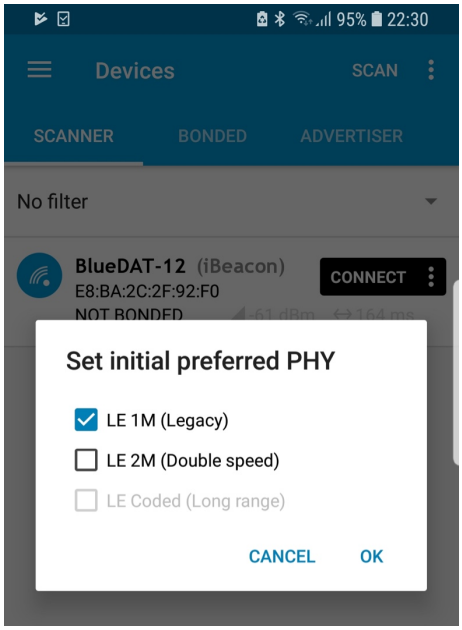

Abbildung 39: Auswahl des bevorzugten physikalischen Layers in der nRF Connect App

Eignung der Smartphones für die Programmierung

Bei der Programmierung der mobilen Applikation wurde diese laufend auf einem Smartphone installiert und getestet. Der Autor wollte eigentlich das iPhone X dazu verwenden, da es bisher beim Programmieren mit dem Ionic-Framework (auch ein Hybrid-App Framework) immer sehr gut funktioniert hat. Leider wurde festgestellt, dass bei der Verwendung des NativeScript-Framework genau das Gegenteil der Fall war. Normalerweise ist es möglich mit dem Befehl `tns run ios` die Applikation auf einem angeschlossenen iOS-Gerät zu installieren und zu debuggen. Wird die App mit diesem Befehl installiert und es wird der Programmcode geändert, dann aktualisiert sich die App innerhalb weniger Sekunden automatisch. Bei der Verwendung des iPhone X hat das leider nicht geklappt, da immer Fehler aufgetreten sind, welche auch nach einer ausführlichen Recherche leider nicht gelöst werden konnten. Um die Applikation am iPhone testen zu können, musste diese händisch mit Hilfe von Xcode installiert werden.

Daher wurde zum Programmieren und regelmäßigen Debuggen nur das Samsung Galaxy S8 verwendet. Bei diesem hat der Befehl `tns run android` einwandfrei funktioniert und es sind keinerlei Probleme aufgetreten.

3.6.5 Fertige Applikation

In den folgenden Abbildungen sind die einzelnen Screens, welche in Kapitel 3.6.2 beschrieben wurden, sowie die dazugehörigen Screenshots der fertigen mobilen Applikation von beiden verwendeten Smartphones abgebildet.

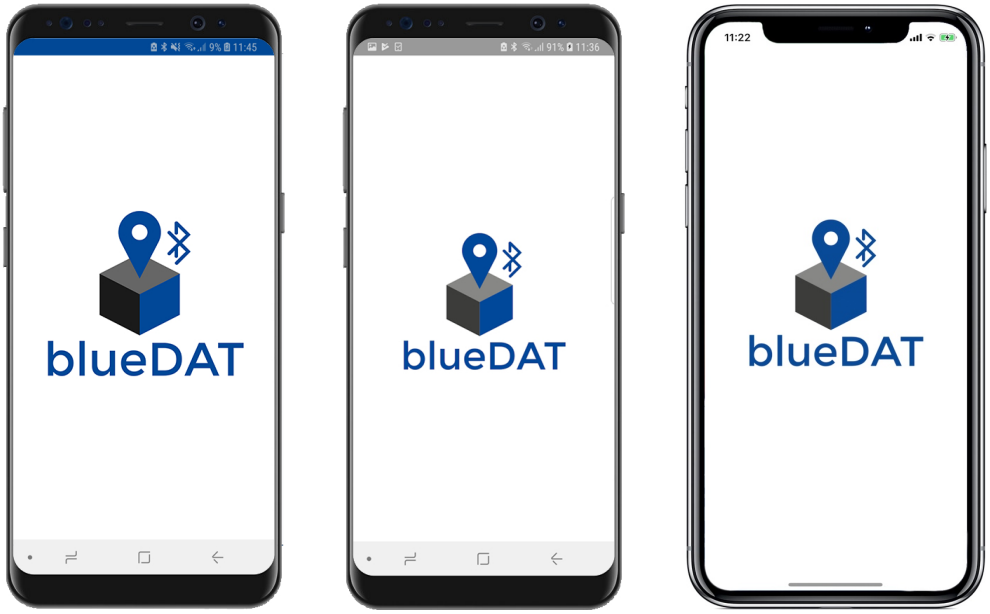

Abbildung 40: Splashscreen der mobilen Applikation als Entwurf (links), auf Android (Mitte) und auf iOS (rechts)

Die Splashscreens entsprechen auf beiden Plattformen, bis auf kleine Unterschiede, dem Entwurf (vgl. Abbildung 40).

Auch die Übersichtslisten und die Signalgeber-Screens entsprechen, bis auf kleine Abweichungen und der unterschiedlichen Darstellung auf den Smartphones, dem Entwurf (vgl. Abbildung 41 und Abbildung 42).

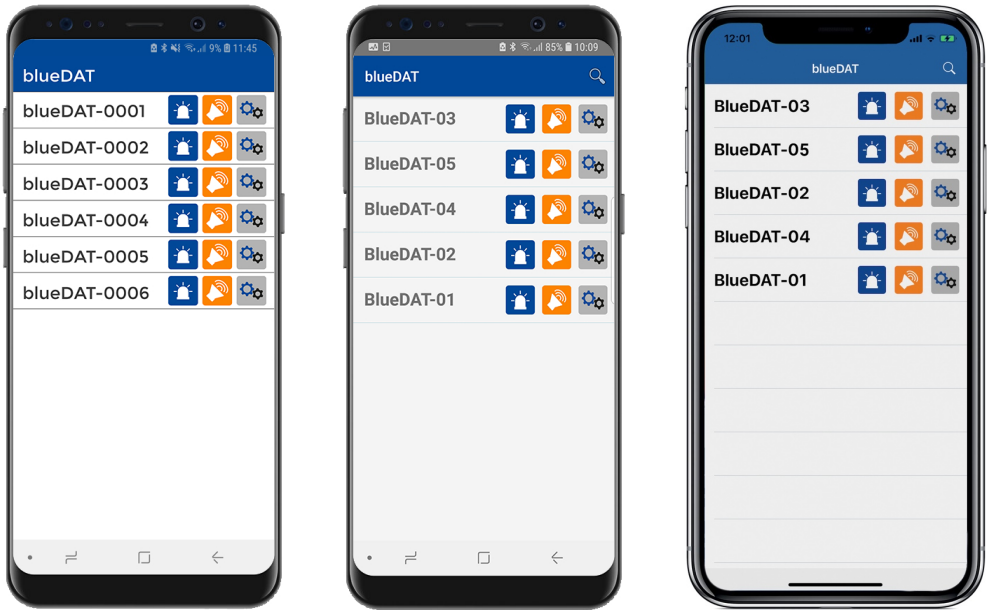

Abbildung 41: Übersichtsliste aller Beacons in Reichweite als Entwurf (links), auf Android (Mitte) und auf iOS (rechts)

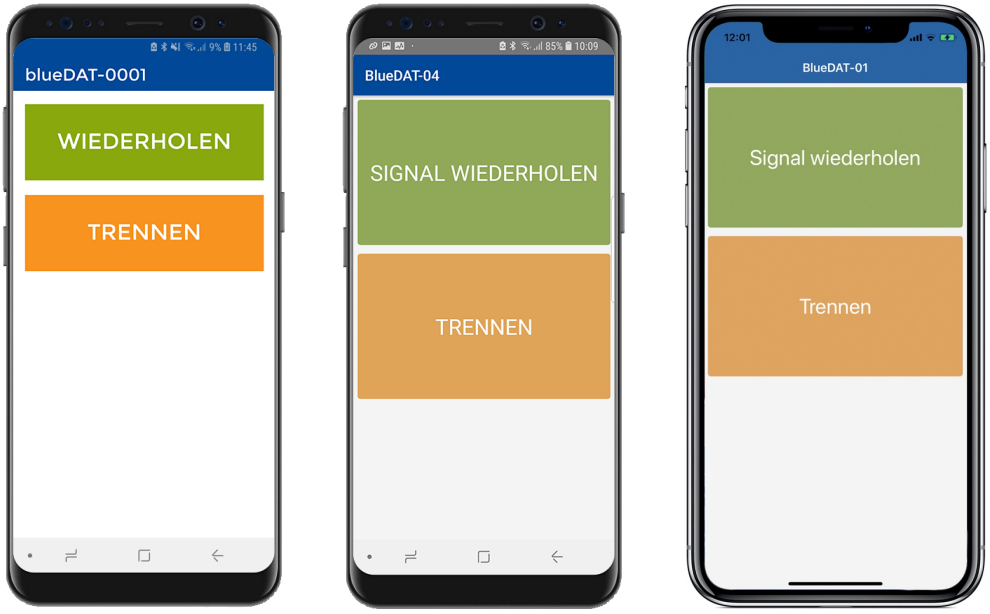

Abbildung 42: Signalgeber-Screen als Entwurf (links), auf Android (Mitte) und auf iOS (rechts)

Bei der Konfigurationsseite wurden in der Actionbar (Leiste am oberen Rand), zwei zusätzliche Buttons, im Vergleich zum Entwurf hinzugefügt (siehe Abbildung 43. Bis auf kleine Unterschiede, wie die Darstellung der Geräte-Adresse und der Auswahlbuttons, ist die mobile Applikation auf beiden Geräten identisch.

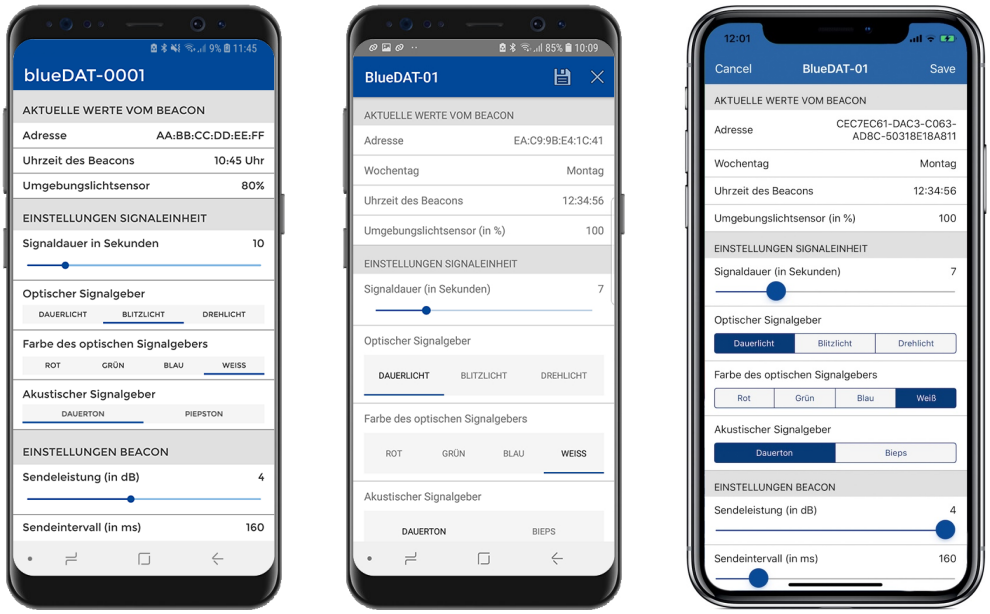

Abbildung 43: Beacon-Konfigurationsseite Teil 1 als Entwurf (links), auf Android (Mitte) und auf iOS (rechts)

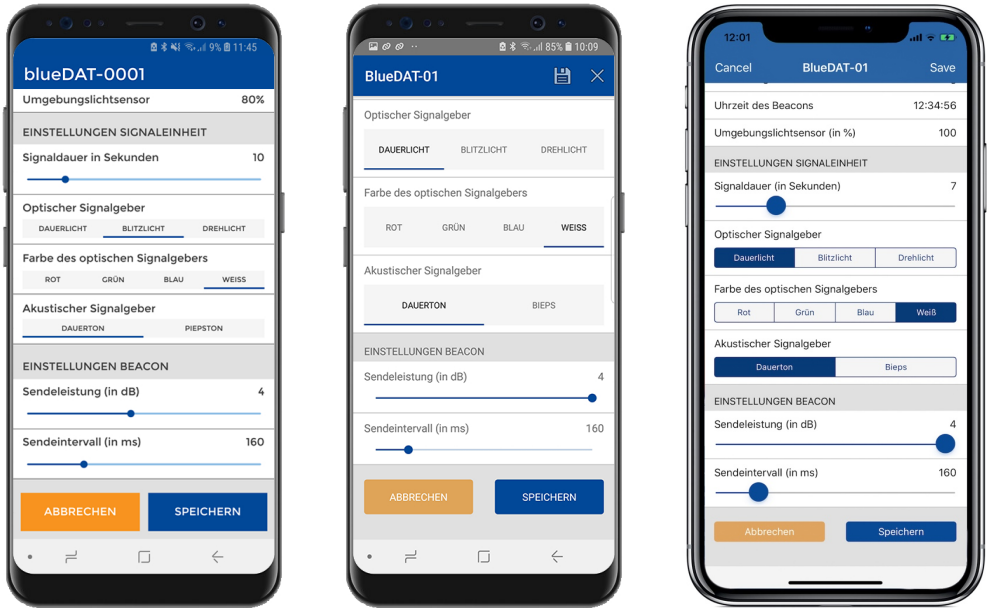

Abbildung 44: Beacon-Konfigurationsseite Teil 2 als Entwurf (links), auf Android (Mitte) und auf iOS (rechts)

Die Unterschiede zwischen den beiden Plattformen iOS und Android sind nur geringfügig und auf die unterschiedliche Darstellung laut den entsprechenden User Interface Guidelines zurückzuführen. Die farblichen Unterschiede beziehungsweise die unterschiedlichen Größen der einzelnen Elemente könnten noch angepasst werden. Grundsätzlich konnte die mobile Applikation aber wie erwartet umgesetzt werden.

3.6.6 Fazit – Mobile Applikation mit NativeScript

Das Programmieren der Hybriden Applikation mit dem NativeScript Framework hat, bis auf kleinere Schwierigkeiten beim Debuggen und Installieren, sehr gut funktioniert. Die aktuelle Applikation wurde noch in JavaScript programmiert. Der Umbau auf TypeScript wäre aufgrund der Kompatibilität zu den neuen NativeScript Versionen und Plug-Ins sinnvoll und für eine Weiterentwicklung notwendig. Auch die stetig wachsende Dokumentation und die vielen Tutorials werden hauptsächlich für TypeScript erstellt. Da die aktuelle Applikation aber den Zweck für die notwendigen Tests erfüllt und keine Weiterentwicklung mehr vorgesehen ist, wurde der Umbau nicht mehr durchgeführt.

3.7 Funktionstest - Mobile Applikation und BlueDAT Beacon

Nachdem die Applikation und der Beacon fertiggestellt waren, wurden alle festgelegten Funktionen der mobilen Applikation (vgl. Kapitel 3.6.1) in Verbindung mit dem BlueDAT Beacon auf beiden verwendeten Smartphones getestet. In Tabelle 11 sind alle getesteten Funktion mit den entsprechenden Ergebnissen aufgelistet.

Jede einzelne Funktion wurde vom Autor getestet und durch die optische bzw. akustische Wahrnehmung überprüft. Nur beim Überprüfen ob die Sendeleistung und das Sendeintervall richtig geändert wurden, verwendete der Autor die nRF Connect App.

Tabelle 11: Auflistung aller Funktionen der mobilen Applikation mit dem entsprechenden Testergebnis

Funktion	Samsung Galaxy S8	Apple iPhone X
Alle Beacons werden angezeigt	Ja	Ja
Optischer Signalgeber - Dauerlicht	Ja	Ja
Optischer Signalgeber - Blitzlicht	Ja	Ja
Optischer Signalgeber - Drehlicht	Ja	Ja
Optischer Signalgeber - Rot	Ja	Ja
Optischer Signalgeber - Blau	Ja	Ja
Optischer Signalgeber - Grün	Ja	Ja
Optischer Signalgeber - Weiß	Ja	Ja
Akustischer Signalgeber - Dauerton	Ja	Ja
Akustischer Signalgeber - Piepston	Ja	Ja
Signaldauer ändern	Ja	teilweise
Signalleistung ändern	Ja	teilweise
Sendeintervall ändern	Ja	teilweise
Wert des Umgebungslichtsensors wird angezeigt	Ja *	Ja *
Aktuelle Uhrzeit wird angezeigt	Ja *	Ja *

* Diese Werte wurden nur in Verbindung mit dem Prototyp, welcher das RedBear Blend v2 Board verwendet, getestet, da auch nur dieser die Werte zur Verfügung stellt.

Am Samsung Galaxy S8 konnten alle Funktionen erfolgreich getestet werden. Beim Apple iPhone X wurde leider festgestellt, dass die Einstellung der Signaldauer, der Signalleistung und dem Sendeintervall mit den Schiebereglern leider nicht hundertprozentig funktioniert. Der Schieberegler liefert Werte mit Kommastellen und nicht wie benötigt ganzzahlige Werte. Es handelt sich dabei aber nur um einen kleinen Fehler, welcher aber in der Programmierung nicht mehr gelöst werden konnte.

4 Tests unter Laborbedingungen

4.1 Bluetooth – Reichweitentest Smartphone

Dieser Test soll einen Teil der Forschungsfrage „Welchen Mehrwert bringt ein auf Bluetooth 5 basierendes Asset Tracking Systems unter Laborbedingungen?“ beantworten und die aufgestellten Hypothese „Ein Bluetooth 5 Beacon bietet, im Vergleich zu Bluetooth 4 Beacons, eine höhere Reichweite.“ verifizieren. Um die Aussagekraft dieses Tests zu erhöhen werden unterschiedliche Bluetooth Module und Smartphones verwendet.

Dabei wurden folgende allgemeinen Testparameter gewählt:

- Die Leistung der einzelnen Module wird durchgehend auf 0dbm eingestellt.
- Das Advertising Intervall wird auf jedem Modul in dem Bereich von 160ms eingestellt.
- Es wird, wenn möglich, das iBeacon Protokoll verwendet.
- Die Module sind so konfiguriert das man keine Verbindung mit ihnen aufbauen kann, sie senden lediglich das Advertising Paket aus. Funktionieren also wie ein herkömmlicher Beacon.

4.1.1 Verwendete Bluetooth Module und Smartphones

Für den Versuch werden insgesamt fünf verschiedene Bluetooth Module bzw. fertige Beacons und zwei verschiedene Smartphones verwendet.

Bluetooth 4 Low Energy Module

Für diesen Test steht leider kein Bluetooth 4 Beacon zur Verfügung, deshalb werden zwei Bluetooth 4 LE Module als Beacon konfiguriert. Bei den Modulen handelt es sich um ein HM-10 Modul basierend auf dem SoC (System-on-a-Chip) CC2540 von Texas Instruments und einem Bluefruit BLE Modul basierend auf dem SoC nRF51822 von Nordic Semiconductor.

Bluetooth 5 Low Energy Module bzw. Beacon

Für den Test steht ein Kontakt.io Pro Beacon zur Verfügung, welcher laut Hersteller „Bluetooth 5 Ready“ ist. Als zweites und drittes Modul wird jeweils ein RedBear Nano v2 Modul und ein nRF52840 Development Kit als Beacon konfiguriert. Der Kontakt.io Beacon sowie das RedBear Modul sind auf dem SoC nRF52832 von Nordic Semiconductor aufgebaut. Das nRF52840 Development Kit verwendet, wie der Name schon sagt, den nRF52840 SoC, ebenfalls von Nordic Semiconductor.

Verwendete Smartphones

Für diesen Test werden ebenfalls die bereits in Punkt 3.6.4 erwähnten Smartphones, ein Apple iPhone X und ein Samsung Galaxy S8 verwendet. Beide verfügen über ein Bluetooth Modul welches Bluetooth 5 Ready ist.

Verwendete Applikation

Für den Test wird auf jedem Smartphone, die im Laufe der Diplomarbeit bereits verwendete Applikation „nRF Connect“ installiert und benutzt.

Download iOS: https://itunes.apple.com/de/app/nrf-connect/id1054362403

Download Android:
https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

4.1.2 Konfiguration der einzelnen Module

Auf die Konfiguration der einzelnen Module wird nicht näher eingegangen, es werden nur nützliche Links und eine kurze Hilfestellung bereitgestellt.

Bluetooth 4 Modul - HM-10

Das HM-10 muss mit AT Commands programmiert werden.

Beschreibung:
http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth/bluetooth40_en.pdf

Bluetooth 4 Modul - Bluefruit BLE

Für dieses Modul wird eine gute Beschreibung mit einigen Tutorials direkt vom Hersteller Adafruit bereitgestellt. Es ist auch ein Tutorial für einen Beacon vorhanden, welches leicht angepasst verwendet wurde.

Webseite: https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/introduction

Bluetooth 5 Beacon – Kontakt.io Beacon Pro

Der Kontakt.io Beacon kann direkt mit der App von Kontakt.io konfiguriert werden.

Android: https://play.google.com/store/apps/details?id=com.kontakt.app
iOS: https://itunes.apple.com/us/app/kontakt-io-administration-app/id1067320511

Bluetooth 5 Beacon – RedBear Nano v2

Installiert man das Board wie in Punkt 3.4.2 in der Arduino IDE, dann wird ein Beispiel Sketch für einen iBeacon mitgeliefert.

Bluetooth 5 Beacon – nRF52840 Development Kit

Für dieses Board wurde ein Beispiel-Programm vom Nordic SDK verwendet, welches mit Hilfe des Segger Embedded Studios auf das Board gespielt wurde.

Nordik SDK: https://www.nordicsemi.com/eng/Products/nRF52840
Segger Embedded Studio: https://www.segger.com/products/development-tools/embedded-studio/

4.1.3 Versuchsaufbau - Version 1

Alle Bluetooth Module bzw. Beacons sind an einem fixen Ort im Freien positioniert. Die Smartphones werden in 5 Meter Schritten von dem Ort entfernt und mit Hilfe der nRF-Connect-App wird nach verfügbaren Bluetooth-Geräten gescannt. Ist ein Bluetooth-Modul in Reichweite wird es entsprechend in eine Tabelle eingetragen. Die Entfernung von Bluetooth-Modulen und Smartphones wird solange erhöht bis keine Module mehr in Reichweite sind. Die Tests werden jeweils einzeln durchgeführt - das heißt, nur ein Modul und ein Smartphone sind gleichzeitig aktiv - um mögliche Störungen unter den Geräten zu vermeiden. Als Versuchsort wird ein freies Gelände ohne Hindernisse und WLAN-Netzwerken gewählt und der Test wird vom Autor durchgeführt. Die endgültige Entfernung zwischen Smartphone und Modul wird mit Hilfe von GPS ermittelt.

4.1.4 Ergebnisse - Version 1

Der Versuch laut 4.1.3, wurde nach den ersten zwei Bluetooth Modulen im Test mit dem Samsung Galaxy S8 abgebrochen. Grund dafür war, dass zwar bereits mit dem HM 10 (Bluetooth 4) eine Reichweite von mehr als 150 Metern erreicht werden konnte, jedoch man nicht genau wusste ob es noch möglich war eine Verbindung mit dem Beacon aufzubauen; dies ist in Bezug auf das Forschungsprojekt jedoch essentiell. Zudem wurde der Versuch HM-10 und

Galaxy S8, zweimal durchgeführt, wobei beim ersten Test mehr als 150 Meter und beim zweiten Test nur rund 80 Meter erreicht werden konnten, obwohl der Versuchsaufbau nicht verändert wurde.

4.1.5 Veränderter Versuchsaufbau – Version 2

Im Vergleich zum Versuchsaufbau in Punkt 4.1.3, wurden nur zwei Veränderungen vorgenommen:

1. Die Konfiguration der Bluetooth-Module wird so verändert, dass es möglich ist, eine Verbindung mit dem Modul aufzubauen.
2. Der Testvorgang wird dahingehend abgeändert, dass der Autor ein Smartphone mit einem Modul verbindet und sich danach schrittweise vom Modul entfernt, bis die Verbindung abgebrochen wird. Danach wird ein neuer Verbindungsaufbau versucht, ist dieser möglich, so wird die Entfernung gemessen und abgespeichert. Ist kein Verbindungsaufbau möglich, dann wird die Entfernung solange verringert, bis einer möglich ist und dann erst die Entfernung gemessen und abgespeichert. Die Entfernung wird auch bei diesem Aufbau mittels GPS gemessen, da eine andere Art der Messung zu aufwendig gewesen wäre.

4.1.6 Ergebnisse - Version 2

Wie in Tabelle 12 ersichtlich ist die Reichweite der Bluetooth Module nahezu identisch, es gibt nur minimale Unterschiede zwischen Bluetooth 4 und Bluetooth 5 Modulen. Die erreichten Entfernungen von ungefähr 70 bis 100 Metern, sind für beide Bluetooth Versionen realistisch, da es sich um eine direkte Sichtverbindung ohne jegliche Störfaktoren handelte. Für die Unterschiede (ca. 10 Meter mehr beim iPhone) zwischen den beiden Smartphones, sieht der Autor die verbauten Bluetooth-Chips oder den Aufbau der Gehäuse verantwortlich.

Tabelle 12: Ergebnisse des Bluetooth Reichweitentest mit zwei Smartphones

	Samsung Galaxy S8	Apple iPhone X
HM-10	70 Meter	80 Meter
Bluefruit BLE	70 Meter	80 Meter
Kontakt.io Pro Beacon	10 Meter	10 Meter
RedBear Nano v2	90 Meter	100 Meter
nRF52840 DK	90 Meter	100 Meter

Was auffällt ist, dass mit dem Kontakt.io Pro Beacon nur eine Entfernung von rund 10 Meter erreicht werden konnte. Der Beacon ist jedoch so konfiguriert,

dass er zwei unterschiedliche Bluetooth Signale aussendet (vgl. Abbildung 45). Eines bei dem ein Verbindungsaufbau möglich ist und eines bei dem kein Verbindungsaufbau möglich ist, dafür werden aber die Beacon-Daten mitgesendet. Aufgrund dieser Tatsache geht der Autor davon aus, dass der Beacon zwar eine eingeschränkte Reichweite bei einer bestehenden Verbindung hat, dass jedoch nur ein Problem der Konfiguration ist. Die Werte des Kontakt.io Pro Beacons werden daher im Vergleich vernachlässigt.

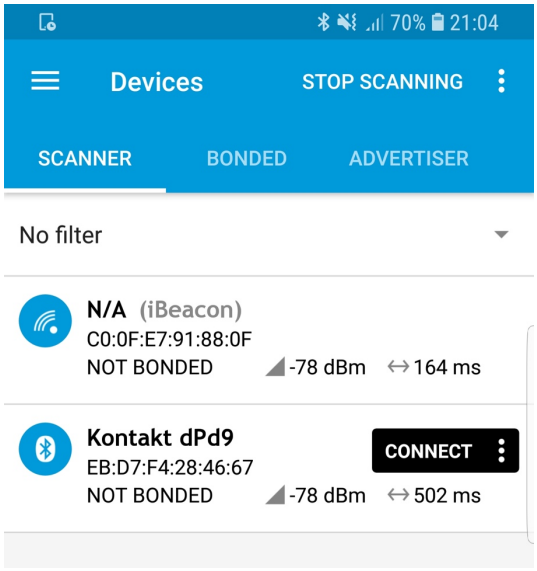

Abbildung 45: Anzeige der beiden Einträge für den Kontakt.io Beacon in der nRF Connect Applikation

4.1.7 Fazit Reichweitentest

Da alle verwendeten Modulen und Smartphones den Bluetooth 5 Long Range Modus nicht unterstützten, sind die nahezu identischen Ergebnisse nachvollziehbar. Es gibt zwar kleine Reichweitenunterschiede zwischen den Modulen, welche aber auch durch den unterschiedlichen Aufbau der Module hervorgerufen werden können.

Um die Reichweite von Bluetooth 5 nun wirklich testen zu können wird ein weiterer Test durchgeführt, siehe dazu Kapitel 4.2

4.2 Bluetooth – Reichweitentest nRF52840DK

Wie in Punkt 4.1.7 beschrieben, verfügt keines der beiden Smartphones über vollständige Bluetooth 5 Chips. Deshalb wird ein weiterer Reichweitentest mit zwei Stück der nRF52840 Development Kits durchgeführt. Diese bauen, wie der Name schon sagt, auf dem nRF52840 SoC auf, welcher auch den Long Range Modus von Bluetooth 5 unterstützt.

4.2.1 Konfiguration der Module

Für diesen Test wird ein Beispiel Code[20] von Nordic Playground verwendet, welcher nur geringfügig angepasst wird. Die Änderungen sind nachfolgend aufgelistet:

1. Connection Intervall auf 2 Sekunden verkürzen
2. Advertisting Intervall auf 160ms einstellen
3. LE Coded als Standardwert einstellen
4. Leistung auf 0dBm einstellen

Es wird jeweils ein Modul als Central und das andere als Peripheral konfiguriert. Die Programmierung wird mit dem Segger Embedded Studio durchgeführt, wobei die dafür notwendigen Schritte nicht beschrieben werden.

Erklärung LEDs der Module

Die 4 LEDs auf jedem Board, zeigen die aktuellen Einstellungen und den aktuellen Status der Verbindung an. In Tabelle 13 und Tabelle 14 sind die möglichen LED-Stati und die Bedeutung der beiden Boards aufgelistet.

Tabelle 13: Bedeutung der LEDs am Peripheral Board

LEDs	Status	Bedeutung
LED 1	ein	Gewählter PHY: LE Coded
	langsam blinken	Gewählter PHY: LE 1M
LED 2	ein	Sendeleistung: 0 dBm
	langsam blinken	Sendeleistung: 8 dBm
LED 3	schnell blinken	„Non-Connectable“ Advertising Pakete
LED 4	schnell blinken	„Connectable“ Advertising Pakete
	ein	Verbindung hergestellt

[20] https://github.com/NordicPlayground/nRF52-ble-long-range-demo

Tabelle 14: Bedeutung der LEDs am Central Board

LEDs	Status	Bedeutung
LED 1	ein	Gewählter PHY: LE Coded
	langsam blinken	Gewählter PHY: LE 1M
LED 2	ein	Sendeleistung: 0 dBm
	langsam blinken	Sendeleistung: 8 dBm
LED 3	ein	Scannen und Versuchen eine Verbindung herzustellen
	langsam blinken	Scannen
LED 4	ein	Verbindung hergestellt
	langsam blinken	Peripheral in Reichweite aber erlaubt keine Verbindung

Erklärung Buttons der Module

Mit den 4 Buttons ist es möglich die Einstellungen der Boards zu verändern. In Tabelle 15 und

Tabelle 16 sind die Buttons mit der zugehörigen Funktion für jedes Board (Peripheral und Central) aufgelistet.

Tabelle 15: Funktion der Buttons am Peripheral Board

Buttons	Funktion
Button 1	Umschalten zwischen LE Coded und LE 1M
Button 2	Umschalten zwischen 0 dBm und 8dBm
Button 3	Umschalten zwischen „Non-Connectable“ und „Connectable“ Advertising
Button 4	Nicht verwendet

Tabelle 16: Funktion der Buttons am Central Board

Buttons	Funktion
Button 1	Umschalten zwischen LE Coded und LE 1M
Button 2	Umschalten zwischen 0 dBm und 8dBm
Button 3	Umschalten zwischen „Scannen“ und „Scannen und Versuchen eine Verbindung herzustellen“
Button 4	Nicht verwendet

4.2.2 Durchführung

Für den Versuch wird das Peripheral Board an einem fixen Ort im Freien positioniert. Die Verbindung zwischen den beiden Modulen wird automatisch aufgebaut und der aktuelle Verbindungsstatus mit den LEDs laut Tabelle 13 und Tabelle 14 angezeigt. Nachdem eine Verbindung hergestellt wurde, entfernt sich der Autor in 5 Meter Schritten mit dem Central Board von dem Standpunkt des Peripheral Boards. Die Entfernung zwischen den Boards wird solange erhöht bis die Verbindung abgebrochen wird (LED 4 am Central Board erlischt). Danach wird die Entfernung mit Hilfe von GPS ermittelt und abgespeichert.

Der Test wird einmal mit dem PHY 1M und einmal mit dem PHY LE Coded durchgeführt, wobei die Einstellungen dafür direkt mit den Buttons (jeweils Button 1) durchgeführt werden (vgl. Tabelle 15 und Tabelle 16).

4.2.3 Ergebnisse und Fazit

Tabelle 17: Ergebnisse des Bluetooth Reichweitentest mit dem nRF52840 DK

Physical Layer	LE 1M	LE Coded
Maximale Reichweite	100 Meter	280 Meter

Wie in Tabelle 17 ersichtlich konnten bei diesem Test eine Entfernung von 100 Meter (LE 1M) und 280 Meter (LE Coded) bei einer bestehenden Bluetooth-Verbindung erreicht werden.

Im Vergleich zum Test mit den Smartphones ist auch hier zu erkennen, dass der Standard Bluetooth 5 LE 1M Modus keine Reichweitenverbesserung mit sich bringt. Jedoch wird mit dem neuen LE Coded Modus die Reichweite nahezu verdreifacht. Die in Kapitel 2.5.3 angegebene Vervierfachung der Reichweite konnte zwar nicht ganz erreicht werden, aber es wurde mit diesem Test bestätigt, dass mit der Verwendung von Bluetooth 5 eine deutlich größere Reichweite möglich ist.

4.3 Test der Signaleinheit

Mit diesem Versuch soll die Erkennbarkeit des optischen sowie des akustischen Signalgebers unter Tageslicht-Bedingungen aus zwei verschiedenen Entfernungen getestet werden. Der Autor wollte zuerst den Test unter drei verschiedenen Lichtbedingungen (Tageslicht, Dunkelheit, große helle Lagerhalle) durchführen. Da aber leider keine große helle Lagerhalle zur Verfügung stand und angenommen wurde das bei Tageslicht die Erkennbarkeit deutlich schwieriger ist, als bei Dunkelheit, wurde nur ein Test bei Tageslicht durchgeführt. Auf Abbildung 46 sieht man die Testumgebung für den Erkennbarkeitstest. Die drei BlueDAT Beacons sind mit den orangen Kreisen markiert.

Abbildung 46: Testumgebung des Erkennbarkeitstest mit den drei BlueDAT Beacons

Wie man auf dem Foto erkennen kann, schien am Tag des Tests die Sonne, was dazu führte das die optische Signaleinheit schwieriger zu erkennen war.

4.3.1 Versuchsaufbau und Testbeschreibung

Für diesen Versuch wurde der programmierte Arduino Code (vgl. Kapitel 3.5.3) sowie auch die Mobile Applikation leicht verändert. Mit der neuen Applikation ist es möglich die Art und Farbe des optischen Signalgebers sowie die Art des akustischen Signalgebers vor dem Auslösen einzustellen. Zudem wird das entsprechende Signal erst ausgelöst, wenn eine aktive Bluetooth-Verbindung besteht und ein Button geklickt wird. In der Abbildung 47 sind die angepassten Screens der mobilen Applikation mit den Einstellungsmöglichkeiten und dem Button zum Auslösen des Signals abgebildet.

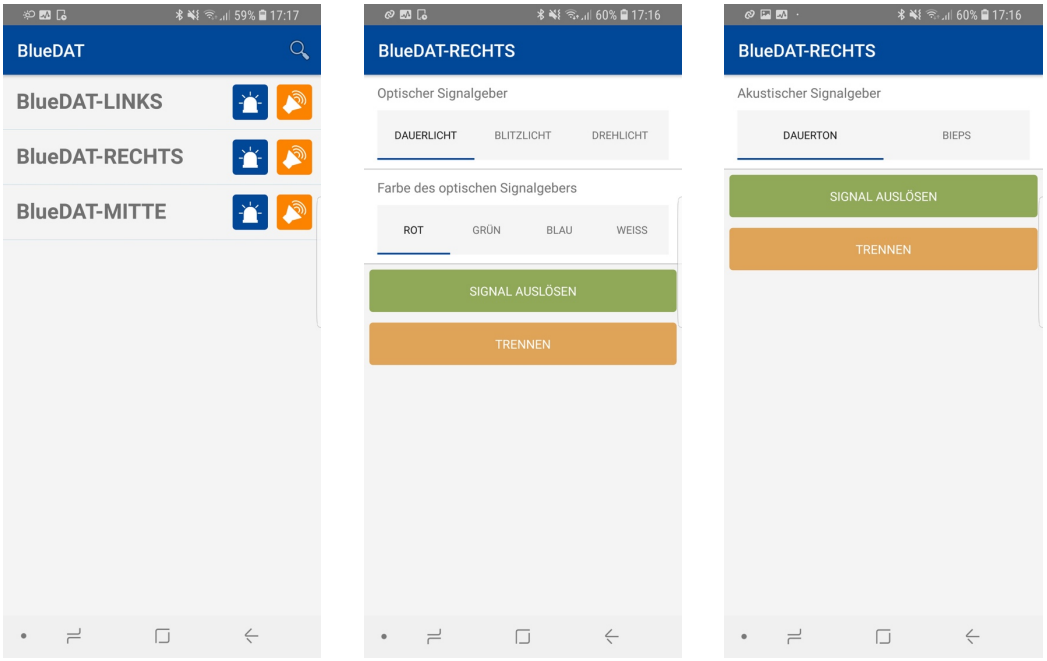

Abbildung 47: Veränderte Screens der mobilen Applikation

Die drei entwickelten BlueDAT Prototypen (vgl. Kapitel 3.5) werden mit einem seitlichen Abstand von 3 Meter in einer Linie aufgestellt. Der Proband nimmt in 20 Metern Abstand davon seine Position ein und wartet auf das Kommando des Testleiters. Dieser wiederum stellt in der Mobilen Applikation eine Sequenz (vgl. Tabelle 19) ein und gibt das Startkommando „3, 2, 1, Go!“. Gleichzeitig mit „Go“ löst er das entsprechende Signal aus und startet eine Stoppuhr. Der Proband versucht nun so schnell wie möglich den Namen des Beacons zu nennen, dessen Signal ausgelöst wurde. Die Beacons werden als LINKS, MITTE und RECHTS bezeichnet. Sobald der Proband einen Namen gesagt hat, stoppt der Testleiter die Stoppuhr und trägt die benötigte Zeit in die zugehörige Zeile der Auswertungstabelle ein. Zusätzlich wird in die Tabelle ebenfalls eingetragen, ob der richtige Beacon genannt wurde.

Der Test wird für jede Sequenz (vgl. Tabelle 19) aus 20 Metern und 40 Metern Entfernung durchgeführt. Alle Sequenzen werden bei jedem Probanden gleich durchlaufen, wobei die maximale Dauer einer Sequenz 10 Sekunden beträgt. Insgesamt wird der Versuch mit 5 verschiedenen Testpersonen durchgeführt, wobei der Autor dabei jedes Mal die Testleitung übernimmt.

Vor dem Test werden der Name, das Alter und ob der Proband Brillenträger ist oder nicht notiert. Nach dem Test werden jedem Probanden noch die folgenden abschließenden Fragen gestellt:

1. Wie ist das allgemeine Empfinden zu diesem Erkennbarkeitstest?
2. Welcher Signalgeber und welche Signalart ist gefühlsmäßig für Sie am besten erkennbar?

4.3.2 Testsequenzen

Für den Erkennbarkeitstest wurden die Arten und Farben der Signalgeber auf die folgenden Möglichkeiten eingeschränkt (siehe Tabelle 18):

Tabelle 18: Mögliche Arten/Farben der Signalgeber für den Erkennbarkeitstest

Optischer Signalgeber		**Akustischer Signalgeber**
Arten	**Farben**	**Arten**
Dauerlicht	Rot	Dauerton
Blitzlicht	Blau	Piepston
Drehlicht	Grün	
	Weiß	

Aus den Arten und Farben (laut Tabelle 18) wurden insgesamt 27 verschiedene zufällige Sequenzen erstellt. Diese sind in Tabelle 19 auf der nächsten Seite aufgelistet.

Tabelle 19: Sequenzen für den Erkennbarkeitstest

Sequenz	Beacon	Signalgeber	Signalart	Farbe
1	Links	LED	Drehlicht	Grün
2	Rechts	SOUND	Dauerton	
3	Mitte	LED	Drehlicht	Weiß
4	Links	LED	Dauerlicht	Weiß
5	Rechts	LED	Drehlicht	Rot
6	Links	LED	Drehlicht	Blau
7	Rechts	LED	Blitzlicht	Grün
8	Links	LED	Dauerlicht	Rot
9	Mitte	LED	Dauerlicht	Blau
10	Mitte	SOUND	Piepston	
11	Mitte	LED	Drehlicht	Grün
12	Rechts	SOUND	Piepston	
13	Rechts	LED	Drehlicht	Weiß
14	Links	SOUND	Dauerton	
15	Rechts	LED	Dauerlicht	Grün
16	Mitte	LED	Dauerlicht	Weiß
17	Links	LED	Blitzlicht	Weiß
18	Mitte	LED	Blitzlicht	Rot
19	Rechts	LED	Drehlicht	Blau
20	Rechts	LED	Dauerlicht	Rot
21	Links	LED	Dauerlicht	Blau
22	Mitte	LED	Blitzlicht	Grün
23	Rechts	LED	Blitzlicht	Weiß
24	Links	SOUND	Piepston	
25	Links	LED	Blitzlicht	Rot
26	Mitte	LED	Blitzlicht	Blau
27	Mitte	SOUND	Dauerton	

4.3.3 Auswertung der Testergebnisse - Zeiten

In Tabelle 20 sind alle ermittelten Zeiten und Fehler (F) pro Proband, Sequenz und Entfernung aufgelistet. Wie man erkennen kann, traten Fehler nur bei den Sequenzen 2, 10, 12, 14, 24 und 27 auf (Orange markierte Zeilen). Sucht man in der Tabelle 19 die entsprechende Signaleinheit, so stellt man fest, dass es sich

dabei immer um den akustischen Signalgeber handelt. Die Fehler sind auf die schlechte Ortung des akustischen Signals zurückzuführen. Der geringe Abstand zwischen den Beacons macht es schwierig den genauen Ausgangspunkt des Signals zu erkennen.

Tabelle 20: Gesamtübersicht der erhobenen Daten

	Proband 1				Proband 2				Proband 3				Proband 4				Proband 5			
Seq.	**20m**	**F**	**40m**	**F**	**20m**	**F**	**40m**	**F**	**20m**	**F**	**40m**	**F**	**20m**	**F**	**40m**	**F**	**20m**	**F**	**40m**	**F**
1	2,72		3,75		2,80		2,47		1,12		3,10		1,34		1,70		2,36		1,02	
2	3,31		4,25	x	2,42		3,24		3,32	x	2,65	x	2,58		2,70	x	10,35	x	7,88	
3	2,33		1,09		2,82		1,06		0,79		0,95		1,15		0,92		1,06		1,10	
4	1,58		2,39		1,66		1,37		1,01		1,65		1,11		0,94		0,82		1,66	
5	2,77		2,11		1,85		3,98		1,76		1,12		1,75		1,45		1,08		1,36	
6	3,30		2,10		2,22		1,76		1,23		1,03		1,18		1,10		0,98		1,16	
7	1,53		1,33		1,32		1,33		1,36		1,53		1,41		1,35		1,45		1,29	
8	2,25		1,97		2,68		1,29		1,11		1,67		0,97		1,15		0,85		1,53	
9	1,26		1,07		1,10		0,96		0,90		1,14		1,04		0,96		1,04		1,41	
10	2,75		2,59	x	2,71		4,05		3,16	x	5,72	x	2,80		3,96	x	8,39		4,26	
11	1,29		1,03		1,03		1,07		0,96		1,06		1,12		1,05		1,12		1,02	
12	4,00	x	4,07	x	2,49		6,22		5,56		7,07		2,04		3,44		3,13	x	3,85	
13	1,40		3,20		2,80		1,52		0,96		1,04		1,31		1,19		2,09		1,25	
14	4,96		2,73		1,32		7,30	x	4,54		4,23		2,34	x	2,24	x	4,28		2,95	
15	2,58		1,08		0,89		0,97		0,87		0,99		1,03		1,00		1,41		0,84	
16	1,10		1,22		1,07		0,96		0,88		0,99		0,83		0,84		1,23		1,11	
17	1,69		1,58		1,22		1,17		1,34		1,23		1,53		1,36		1,45		1,77	
18	1,83		1,59		1,48		1,05		1,80		1,08		1,52		1,13		1,30		1,23	
19	1,23		2,01		0,93		0,94		0,97		1,12		1,11		1,21		1,91		1,17	
20	3,56		1,40		0,75		1,79		0,90		0,92		1,11		0,95		1,40		1,02	
21	1,82		1,42		1,05		0,92		0,91		0,91		1,33		1,14		1,05		1,18	
22	1,32		1,58		1,01		1,21		1,25		1,25		1,31		1,55		1,70		1,42	
23	1,24		1,32		1,18		1,32		1,22		1,16		1,27		1,53		1,52		1,56	
24	2,12		2,92	x	1,85		8,94	x	2,44		4,00		2,88		5,87		4,23		4,95	
25	1,90		2,53		1,59		1,29		1,36		1,18		1,48		1,39		1,70		1,65	
26	1,23		1,42		1,20		1,66		1,18		1,11		1,53		1,39		1,42		1,35	
27	2,66		3,12		1,42		5,50	x	3,78	x	3,72	x	1,74		3,02		11,05	x	3,18	
MIN	1,10		1,03		0,75		0,92		0,79		0,91		0,83		0,84		0,82		0,84	
MAX	4,96		4,25		2,82		8,94		5,56		7,07		2,88		5,87		11,05		7,88	
Ø	2,21		2,11		1,66		2,42		1,73		1,99		1,51		1,72		2,61		2,01	

Zusätzlich werden in Tabelle 20 die kürzeste (MIN), längste (MAX) und die durchschnittlich (Ø) benötigte Zeit pro Proband und Entfernung berechnet. In Abbildung 48 sieht man diese Werte in einem Diagramm zusammengefasst.

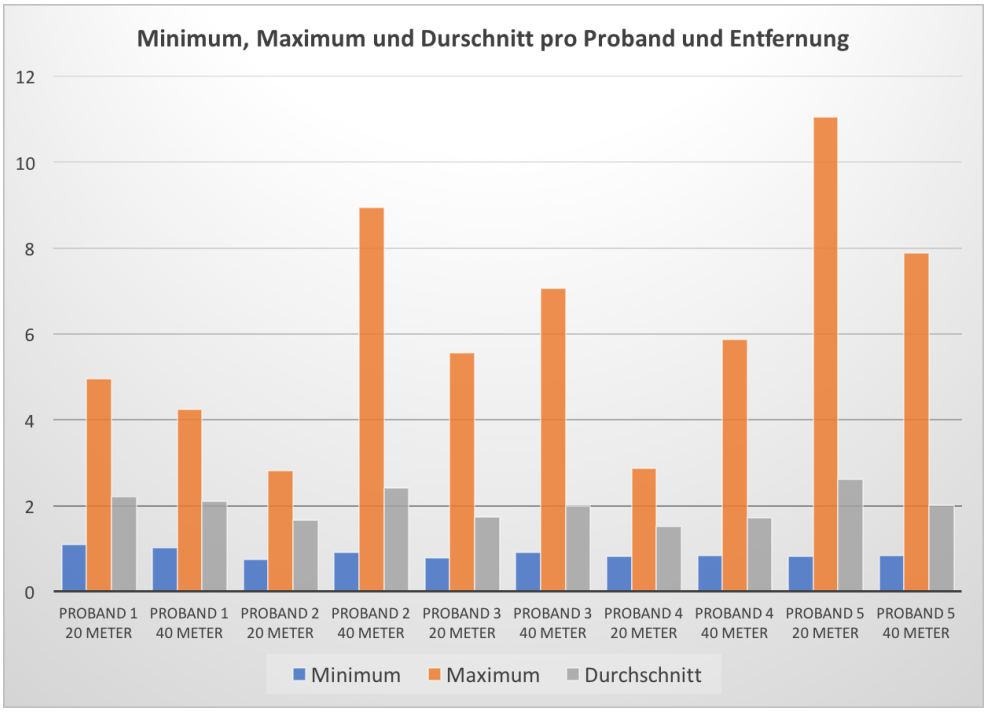

Abbildung 48: Minimale, maximale und durchschnittliche Zeit pro Proband und Entfernung

Wie man sieht sind die Unterschiede bei der kürzesten und der durchschnittlichen Zeit bei allen Probanden und bei beiden Entfernungen eher gering. Bei der maximalen Zeit sieht man einige Ausreißer, welche auf die schlechte Ortung des akustischen Signals zurückzuführen sind. Weiters ist zu erkennen, dass die Erhöhung der Entfernung (von 20 auf 40 Meter) zu den Beacons keine großen Veränderungen mit sich gebracht hat.

In Abbildung 49 sind die kürzeste (Minimum), längste (Maximum) und die durchschnittlich (Durchschnitt) benötigte Zeit pro Sequenz als Durchschnitt über alle Probanden und Entfernungen dargestellt. Die größten Unterschiede gibt es zwischen den beiden Arten von Signalgeber – optisch (LED) und akustisch (SOUND). Bei den akustischen Signalen ist die durchschnittliche Dauer bis zum Erkennen wesentlich länger als bei optischen Signalen. Zwischen den Arten und Farben des optischen Signalgebers gibt es keine nennenswerten Unterschiede, außer, dass bei der Verwendung des Drehlichtes teilweise eine etwas längere Zeit benötigt wurde.

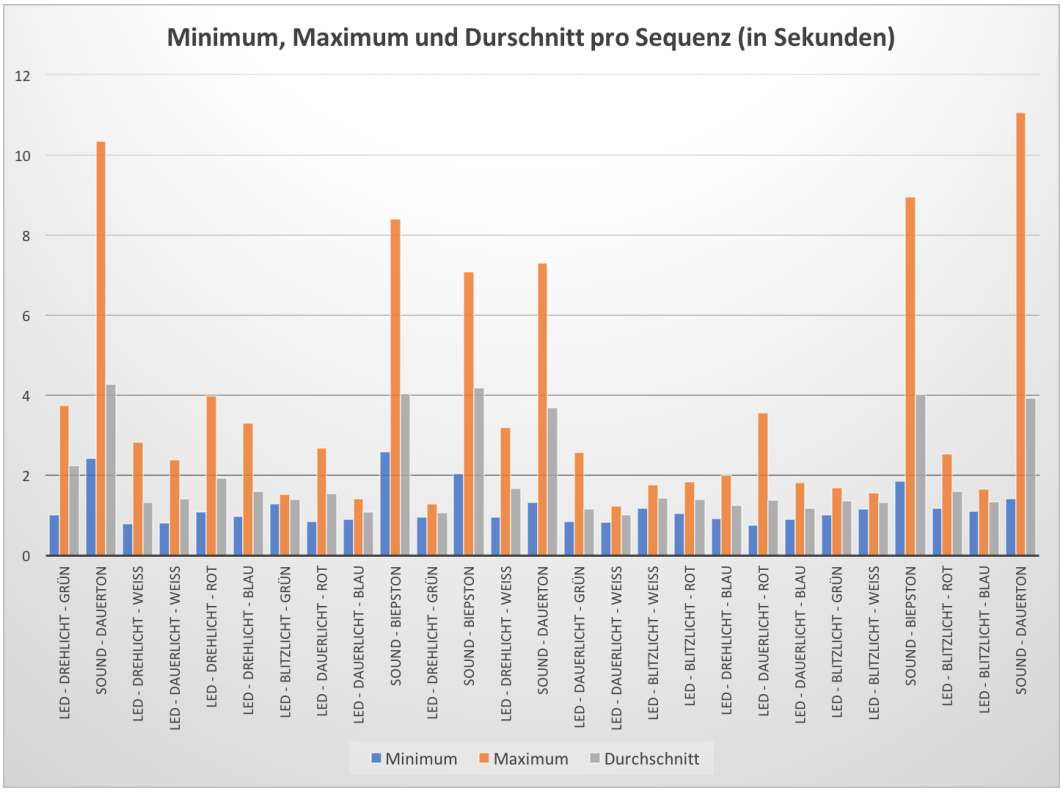

Abbildung 49: Minimale, maximale und durchschnittliche Zeit pro Sequenz

Auf Abbildung 50 sind die durchschnittliche Zeiten bis zum Erkennen des Signals für die unterschiedlichen Signalarten dargestellt.

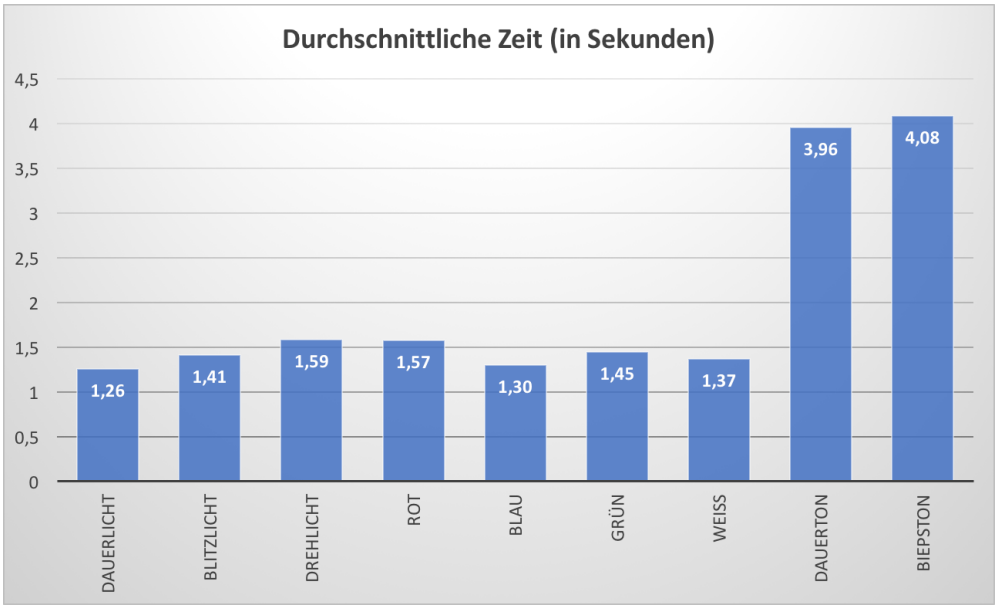

Abbildung 50: Durchschnittliche Zeit bis zum Erkennen des Signals (in Sekunden) für die unterschiedlichen Signalarten

Das Diagramm in Abbildung 50 bestätigt den zuvor erwähnten Unterschied zwischen optischen und akustischen Signalgeber. Die Dauer bis zum Erkennen des Signals ist bei der Verwendung des akustischen Signalgebers um etwa das dreifache höher als beim optischen Signalgeber. Die Unterschiede zwischen den vier möglichen Farben sowie denn drei möglichen Arten des optischen Signalgebers sind im Gegensatz dazu eher gering.

4.3.4 Auswertung der persönlichen Daten und Fragen an die Probanden

In Tabelle 21 sind alle gesammelten persönlichen Daten der Probanden aufgelistet. Die Namen der Probanden werden aus Datenschutzgründen nicht angegeben. Der Autor hat sich für Personen im Alter von 20 bis 40 Jahren und zwei weibliche und drei männliche entschieden.

Tabelle 21: Übersicht der gesammelten persönlichen Daten der Probanden

	Geschlecht	Alter	Brille
Proband 1	weiblich	35	Ja
Proband 2	männlich	25	Nein
Proband 3	weiblich	29	Nein
Proband 4	männlich	21	Nein
Proband 5	männlich	37	Ja

Zusammenfassung der Allgemeine Fragen nach dem Test

1. **Wie ist das allgemeine Empfinden zu diesem Erkennbarkeitstest?**

Grundsätzlich waren alle Probanden mit der Erkennbarkeit der Beacons zufrieden. Es kam der Vorschlag, beide Signalgeber miteinander zu verbinden bzw. den optischen Signalgeber zum genauen Bestimmen der Position und den akustischen Signalgeber zum ungefähren Bestimmen der Position zu verwenden. Das heißt, wenn man weiter entfernt vom Beacon ist, soll das akustisches Signal ausgelöst werden und wenn man sich in einer gewissen Nähe zum Beacon befindet, soll das optische Signal ausgelöst werden.

Das akustische Signal ist bei beiden Entfernungen nur sehr schwierig zu orten, daher auch die recht langen Zeiten bis zum Erkennen des Signals.

2. **Welcher Signalgeber und welche Signalart ist gefühlsmäßig für Sie am besten erkennbar?**

Nach den Aussagen der fünf Probanden sind gefühlsmäßig die Signale „Grün-Blitzlicht“ und „Grün-Dauerlicht“ am besten zu Erkennen. Das Signal „Drehlicht“ wird als zu wenig hell empfunden und ist deshalb tendenziell eher schlechter zu erkennen.

4.3.5 Kurzes Fazit

Zusammenfassend ist zu sagen, dass es keine wesentlichen Unterschiede zwischen den Farben und Arten des optischen Signals gibt. Der Unterschied zwischen optischen und akustischen Signal wurde aber deutlich bewiesen. Durch die Aussagen und Meinungen der Probanden konnten einige gute Ideen gesammelt werden, welche für das parallel gestartete Forschungsprojekt mit Sicherheit von Vorteil sein können.

4.4 Akkulaufzeit berechnen

Die realistisch mögliche Akkulaufzeit des Beacon Prototyp soll durch eine Berechnung ermittelt werden. Dazu wurde zuerst der Stromverbrauch von jedem der drei erstellten Prototypen in den unterschiedlichen Betriebsmodi mit Hilfe eines Multimeters gemessen. Aus den gesamten Messergebnissen wurde der durchschnittliche Stromverbrauch pro Betriebsmodi berechnet. Die Messergebnisse sind in Tabelle 22 angeführt.

Tabelle 22: Durchschnittlicher Stromverbrauch des Prototyps in den verschiedenen Modi

Betriebsmodi	Stromverbrauch
MCU (Standby)	9mA
MCU + LED Ring Rot	180mA
MCU + LED Ring Grün	130mA
MCU + LED Ring Blau	125mA
MCU + LED Ring Weiß	740mA
MCU + Dauerton	30mA

Der verwendete Akku hat eine Kapazität von 2500mAh, daraus ergibt sich im Standby-Modus eine maximale Laufzeit von nur rund 12,5 Tagen. Mit dieser doch sehr kurzen Laufzeit hat der Autor nicht gerechnet und wollte diese durch

einen Langzeittest realistisch überprüfen. Leider war es aus zeitlichen Gründen nicht mehr möglichen diesen Test durchzuführen, daher wurde ein kürzerer Test mit dem optischen Signalgeber getätigt. Dabei wurde der Prototyp so umprogrammiert, dass der LED Ring durchgehend in Weiß leuchtet. Bei diesem Test konnte eine Laufzeit von rund 4 Stunden erreicht werden. Die berechnete Akkulaufzeit im Betriebsmodi „MCU + LED Ring Weiß“ von ungefähr 3,5 Stunden stimmt somit nahezu überein.

Fazit Akkulaufzeit

Die sehr kurze Akkulaufzeit ist ein großer Nachteil des Prototyps im Vergleich zu anderen kommerziellen Beacons. Sollte der BlueDAT Beacon weiterverwendet werden, ist eine Verbesserung der Laufzeit unbedingt notwendig.

5 Fazit

Das Ziel der Arbeit war es, einen konfigurier- und erweiterbaren Bluetooth 5 Beacon sowie eine mobile Applikation für Android und iOS zur Steuerung und Konfiguration des Beacons zu entwickeln. Dafür wurde der entsprechende theoretische Hintergrund in Bezug auf Bluetooth 5, aktuelle Beacons und Forschungsprojekte für Asset Tracking bzw. Indoor Navigation und die möglichen Entwicklungsoptionen für die mobile Applikation recherchiert.

Der entwickelte BlueDAT Beacon wurde mit Hilfe der entwickelten BlueDAT Applikation einigen Tests unterzogen, um die zu Beginn gestellten Forschungsfragen zu beantworten und die aufgestellten Hypothesen zu verifizieren.

5.1 Forschungsfragen und Hypothesen

Frage 1: Zu welchem Gesamtpreis in Zusammenhang mit sinnvollen Funktionen kann ein funktionierender Bluetooth 5 Beacon umgesetzt werden?

Die kleinste Ausbaustufe des Bluetooth 5 Beacons mit dem RedBear Nano v2 Board und einen akustischen sowie einen optischen Signalgeber kann zu einem Gesamtpreis von circa Euro 60,00 umgesetzt werden. Bei diesem Preis ist das 3D Gehäuse jedoch noch nicht mit einberechnet. Im Vergleich zu kommerziellen Beacons ist das zwar deutlich teurer, jedoch verfügt der Prototyp auch über einige einzigartige Komponenten wie die leuchtstarke LED Kuppel. Eine genaue Kostenaufstellung ist in Kapitel 3.3.7 angeführt.

Frage 2: Welche Signalgeber sind in hellen großen Räumen am besten für die Positionsanzeige des Beacons geeignet? (Optisch, Akustisch, ...)

Durch den Erkennbarkeitstest mit insgesamt 5 Probanden kam man zu den folgenden Erkenntnissen in Bezug auf den Signalgeber:

- Für die genaue Ortung sollte ein optischer Signalgeber verwendet werden.

- Für die ungefähre Ortung aus einer größeren Entfernung, macht es Sinn ein akustisches Signal zu verwenden.
- Die unterschiedlichen Farben oder Arten des optischen Signalgebers haben nur einen geringen Einfluss auf die Erkennbarkeit.
- Laut den Aussagen der Probanden ist die Farbe Grün am besten zu erkennen.

Zusammenfassend ist zu sagen, dass auf diese Frage keine eindeutige Antwort gegeben werden kann. Der beste Signalgeber sollte für jeden Anwendungsfall explizit getestet und danach ausgewählt werden.

Frage 3: Welche Akkulaufzeit des Beacons ist realistisch möglich und in welchem Verhältnis stehen Akkukapazität und Akkulaufzeit in Beziehung auf den umgesetzten Prototypen?

Die realistische Akkulaufzeit des entwickelten Prototyps ist mit nur rund 12 Tagen sehr gering. Würde man die Akkukapazität (aktuell 2500mAh) erhöhen, würde natürlich auch die Laufzeit entsprechend ansteigen. Allerdings wird man bei dem aktuellen Stromverbrauch des Beacons nie eine anständige Laufzeit von zumindest einem Jahr erreichen, ohne dabei die Dimension des Akkus sowie des Beacons gravierend zu vergrößern. Um die Akkulaufzeit etwas zu erhöhen, würde es vermutlich ausreichen alle verwendeten Bauteile sowie auch den Programmcode zu optimieren. Will man aber eine Akkulaufzeit von mehreren Jahren erreichen, dann wird es notwendig sein den Beacon ohne fertiges MCU-Board aufzubauen. Dafür wäre es notwendig eine Platine nur mit den notwendigen und optimierten Bauteilen/Chips zu entwickeln.

Frage 4: Welche Entwicklungsoption (Web, Hybrid, Native, ...) ist für mobile Applikationen mit der Verwendung von Bluetooth am sinnvollsten?

Für die Entwicklung der mobilen Applikation im Rahmen der Diplomarbeit wurde das Hybrid Framework NativeScript gewählt. Für den Test des Beacons war die damit umgesetzte Applikation ausreichend und funktionierte auch auf beiden benutzen Smartphones ohne große Schwierigkeiten. Das lag auch daran, dass sowohl der entwickelte Prototyp als auch die verwendeten Smartphones den Bluetooth Long Range Modus nicht unterstützten. Denn mit dem Hybrid Framework ist es aktuell nicht möglich den gewünschten Physikalischen Layer auszuwählen und somit den Long Range Modus von Bluetooth 5 zu verwenden.

Vorausgesetzt es steht ein Smartphone mit einem vollständigen Bluetooth 5 Modul zur Verfügung, sollte man eher auf eine native Programmierung setzen und für jedes Betriebssystem eine optimierte Applikation erstellen. Denn nur

dann ist es mit Sicherheit möglich, dass die kompletten Funktionen des Smartphones ausgenutzt werden können.

Frage 5: Welchen Mehrwert bringt ein auf Bluetooth 5 basierendes Asset Tracking Systems unter Laborbedingungen?

Durch die durchgeführten Tests kann in Bezug auf den Mehrwert von Bluetooth 5 leider nur die folgende Aussage getätigt werden: „Bei der Verwendung von vollständigen Bluetooth 5 Modulen und dem Long Range Modus kann eine höhere Reichweite als bei Bluetooth 4 erreicht werden."

Der Mehrwert eines Asset Tracking Systems mit intelligenten Bluetooth 5 Beacons im Vergleich zum „normalen" Suchen von Gegenständen, konnte leider nicht ermittelt werden, da der dafür vorgesehene Laborversuch aus zeitlichen Gründen nicht mehr durchgeführt werden konnte.

Ist man beim Asset Tracking System auf ein Smartphone angewiesen, dann ist der Mehrwert im Vergleich zu Bluetooth 4 Systemen, aktuell nicht sehr viel größer, da es, wie bereits des Öfteren erwähnt, derzeit noch kein Smartphone mit einem Bluetooth Modul gibt welches den LE Coded Modus unterstützt.

Verifikation der Hypothesen

Die folgenden beiden aufgestellten Hypothesen wurden durch die Arbeit untersucht und können aufgrund der erlangten Erkenntnisse bestätigt werden:

- Ein Bluetooth 5 Beacon bietet, im Vergleich zu Bluetooth 4 Beacons, eine höhere Reichweite.
- Es ist möglich einen konfigurierbaren und erweiterbaren Bluetooth 5 Beacon, mit frei verfügbarer Hardware und Open-Source Software umzusetzen.

Hinweis zur Verifikation der Hypothese 1:

Nur bei der Verwendung des Long Range Modus von Bluetooth 5. Jedoch gibt es aktuell weder Smartphones noch Beacons die diesen Modus unterstützten Im Rahmen der Diplomarbeit konnte einzig mit zwei nRF52840DK Boards eine höhere Entfernung überbrückt werden.

Hinweis zur Verifikation der Hypothese 2:

Aktuell konnte nur ein Bluetooth 5 Ready Beacon hergestellt werden, der jedoch in Sachen Akkulaufzeit und Preis nicht mit kommerziellen Beacons vergleichbar ist.

5.2 Weiterführendendes

Optimierung des Beacon

- Da der entwickelte Prototyp eine geringe Akkulaufzeit und einen hohen Herstellungspreis aufweist, wäre es sinnvoll den Beacon weiter zu optimieren oder komplett neu aufzubauen.
- Das 3D Gehäuse erfüllt zwar zum jetzigen Zeitpunkt seinen Zweck, sollte aber ebenfalls optimiert werden. Zum Beispiel könnte man es spritzwassergeschützt oder überhaupt wasserdicht aufbauen.
- Für die Erstellung von mehreren Beacons wäre ein automatisches Konfigurierungsskript wünschenswert, welches beim Anstecken an einem PC oder auch direkt über die Bluetooth-Schnittstelle den Beacon entsprechend programmiert.

Weiterentwicklung des gesamten Systems

Das System Beacon und Mobile Applikation ist derzeit nur für die Tests entwickelt worden. In Zusammenhang mit dem parallel gestarteten Forschungsprojekt müssen beide Bestandteile und das Zusammenspiel noch weiterentwickelt werden. Wie in Kapitel 3.1 bereits erwähnt, soll ein Mitarbeiter durch die Auswahl einer Auftragsnummer die Signaleinheiten der zugehörigen Transportgestelle auslösen können. Dafür ist es notwendig das eine Beacon ID zu einem Auftrag oder auch zu einem Asset zugeordnet werden kann und diese Zuordnung zentral gespeichert wird. Dafür hat der Autor die folgende zwei Ideen gesammelt:

- Ein Mitarbeiter scannt mit einer Mobile Applikation den Barcode eines Assets und wählt danach den zugehörigen Bluetooth Beacon aus einer Liste aus.
- Der Beacon wird um einen NFC Reader erweitert. Das Asset verfügt über einen NFC-Tag, welcher vom Beacon gescannt wird und die Zuordnung automatisch abspeichert.

Weiters sollen die Beacons nicht nur durch eine Mobile Applikation, sondern auch über die Bluetooth Gateways ausgelöst werden können.

Weiterführende Tests

Im parallel gestarteten Forschungsprojekt soll mit dem entwickelten Bluetooth 5 Beacon ein Asset Tracking System umgesetzt werden. Da es dafür verschiedene Technologien gibt (vgl. Kapitel 2.2), würde sich ein Vergleich mit dem in Punkt 2.3.2 erwähnten Pozyx System anbieten. Der Autor wollte das System bereits im

Laufe der Diplomarbeit testen, das wurde jedoch durch den hohen Anschaffungspreis verhindert.

Sobald es ein Smartphone gibt, welches den Bluetooth 5 Long Range Modus unterstützt, wäre es sinnvoll den Reichweitentest mit diesem zu wiederholen. Als Grundmodul für den Beacon könnte dann zum Beispiel das neue Xenon[21] Modul von Particle verwendet werden.

[21] https://www.particle.io/mesh/buy/xenon

Literaturverzeichnis

About Us | Bluetooth Technology Website. (o. J.). Abgerufen 22. Mai 2018, von https://www.bluetooth.com/about-us

Adafruit. (2015a, Mai 4). GAP | Introduction to Bluetooth Low Energy | Adafruit Learning System. Abgerufen 8. August 2018, von https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap

Adafruit. (2015b, Mai 4). GATT | Introduction to Bluetooth Low Energy | Adafruit Learning System. Abgerufen 9. August 2018, von https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt

Adafruit. (2016, September 13). Best Practices | Adafruit NeoPixel Überguide | Adafruit Learning System. Abgerufen 24. Juli 2018, von https://learn.adafruit.com/adafruit-neopixel-uberguide/best-practices

Adafruit. (o. J.). USB LiIon/LiPoly charger. Abgerufen 10. Juli 2018, von https://www.adafruit.com/product/259

Bhargava, M. (2017). *IoT Projects with Bluetooth Low Energy*. Packt Publishing Ltd.

Blackstone, A. (2015, März 25). Understanding the different types of BLE Beacons | Mbed. Abgerufen 9. März 2018, von https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/

Bluetooth 1.0/1.1/1.2 (IEEE 802.15). (o. J.). Abgerufen 22. Mai 2018, von https://www.elektronik-kompendium.de/sites/kom/0803301.htm

Bluetooth SIG, Inc. (2017). rethinking the future / Bluetooth 5, beacon technology and the Internet of Things. Abgerufen von https://www.bluetooth.com/~/media/files/marketing/bluetooth5-brochure-web-144ppi.ashx?la=en

Bluetooth SIG, Inc. (o. J.). Bluetooth 5 | Bluetooth Technology Website. Abgerufen 13. Februar 2018, von https://www.bluetooth.com/specifications/bluetooth-core-specification/bluetooth5

Dais, S. (2017). Industrie 4.0 – Anstoß, Vision, Vorgehen. In *Handbuch Industrie 4.0 Bd.4* (S. 261–277). Springer Vieweg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-53254-6_14

Davidson, K. T., Carles Cufí, Akiba, Robert. (2014). *Getting Started with*

Bluetooth Low Energy. Abgerufen von http://shop.oreilly.com/product/0636920033011.do

Delía, L., Galdamez, N., Corbalan, L., Pesado, P., & Thomas, P. (2017). Approaches to mobile application development: Comparative performance analysis. In *2017 Computing Conference* (S. 652–659). https://doi.org/10.1109/SAI.2017.8252165

Eckstein GmbH. (o. J.-a). Adafruit NeoPixel Ring - 16 x 5050 RGB LED with Integrated Drivers - Raspberry Pi, Arduino, Robote, Motor etc. beim Eckstein Komponente, 11,84 €. Abgerufen 17. Juli 2018, von https://eckstein-shop.de/Adafruit-NeoPixel-Ring-16-x-5050-RGB-LED-with-Integrated-Drivers

Eckstein GmbH. (o. J.-b). GY-61 ADXL335 3-Achsen Beschleunigungssensor Module Magnetfeld 3-5V f. Abgerufen 10. Juli 2018, von https://eckstein-shop.de/GY-61-ADXL335-3-Achsen-Beschleunigungssensor-Module-Magnetfeld-3-5V-fuer-Arduino

Eckstein GmbH. (o. J.-c). LiPo Akku Lithium-Ion Polymer Batterie 3,7V 2500mAh JST-PH Connector,. Abgerufen 10. Juli 2018, von https://eckstein-shop.de/LiPo-Akku-Lithium-Ion-Polymer-Batterie-37V-2500mAh-JST-PH-Connector

Eckstein GmbH. (o. J.-d). RTC DS3231 AT24C32 Memory Real Time Clock IIC Modul - Raspberry Pi, Arduino, Robote, Motor etc. beim Eckstein Komponente, 3,86 €. Abgerufen 10. Juli 2018, von https://eckstein-shop.de/DS3231-RTC-Modul-LIR2032

Eckstein GmbH. (o. J.-e). Seeed Studio Grove - Buzzer Sound Output >=85dB Resonant Frequency. Abgerufen 10. Juli 2018, von https://eckstein-shop.de/Seeed-Studio-Grove-Buzzer-Sound-Output-85dB-Resonant-Frequency-2300a300Hz

Elektronik-Kompendium.de. (o. J.-a). Bluetooth 5. Abgerufen 13. Juni 2018, von https://www.elektronik-kompendium.de/sites/kom/2107121.htm

Elektronik-Kompendium.de. (o. J.-b). NFC - Near Field Communication. Abgerufen 20. Februar 2018, von https://www.elektronik-kompendium.de/sites/kom/1107181.htm

Estimote, Inc. (o. J.). Intro to Eddystone - Estimote Developer. Abgerufen 9. März 2018, von https://developer.estimote.com/eddystone/

Fahrngruber, N. (2016, Oktober 9). Echtzeit Indoor Positionierungssystem mittels WPAN.

Fraunhofer ISST. (o. J.). Funktionsweise von Beacons. Abgerufen von http://www.zbb.de/projekte/handel-

kompetent/?no_cache=1&cid=3384&did=2040&sechash=2bf24981

Gartner. (2017, Februar). Prognose zur Anzahl der vernetzten Geräte im Internet der Dinge (IoT) weltweit in den Jahren 2016 bis 2020 (in Millionen Einheiten). Abgerufen 1. Februar 2018, von https://ezproxy.fhstp.ac.at:2081/statistik/daten/studie/537093/umfrage/anzahl-der-vernetzten-geraete-im-internet-der-dinge-iot-weltweit/

Gezici, S., Zhi Tian, Giannakis, G. B., Kobayashi, H., Molisch, A. F., Poor, H. V., & Sahinoglu, Z. (2005). Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE Signal Processing Magazine*, *22*(4), 70–84. https://doi.org/10.1109/MSP.2005.1458289

Gürayer, K. (2017, März 30). Bluetooth 5: Galaxy S8 bringt Musik auf zwei Headsets gleichzeitig. Abgerufen 13. Februar 2018, von http://www.giga.de/smartphones/samsung-galaxy-s8-plus/news/bluetooth-5-galaxy-s8-bringt-musik-auf-zwei-headsets-gleichzeitig/

Hermann, M., Pentek, T., & Otto, B. (2016). Design Principles for Industrie 4.0 Scenarios. In *2016 49th Hawaii International Conference on System Sciences (HICSS)* (S. 3928–3937). https://doi.org/10.1109/HICSS.2016.488

infsoft GmbH. (o. J.). Indoor Positionsbestimmung und Indoor Navigation mit WLAN. Abgerufen 23. Mai 2018, von https://www.infsoft.de/technologie/sensorik/wlan

it´s owl. (o. J.). Evolution statt Revolution. Abgerufen 10. Juli 2018, von https://www.its-owl.de/industrie-40/evolution-statt-revolution/

Junnila, A. (2017, Juni 21). 5+1 Asset Tracking Technologies: Which One to Use for Your Business? Abgerufen 20. Februar 2018, von https://trackinno.com/2017/06/21/asset-tracking-technologies/

Kickstarter. (o. J.). Bluetooth 5 Ready: BLE Module, Nano 2 & Blend 2. Abgerufen 10. Juli 2018, von https://www.kickstarter.com/projects/redbearinc/bluetooth-5-ready-ble-module-nano-2-and-blend-2

Kolkmann, T. (2017, September 21). Bluetooth: Geschwindigkeit, Reichweite & grundlegende Infos zur Funktechnik. Abgerufen 13. Februar 2018, von http://www.giga.de/extra/bluetooth/

Kontakt.io. (o. J.). iBeacon vs Eddystone. Abgerufen 8. März 2018, von https://kontakt.io/beacon-basics/ibeacon-and-eddystone/

Luo, C., Cheng, L., Chan, M. C., Gu, Y., Li, J., & Ming, Z. (2017). Pallas: Self-Bootstrapping Fine-Grained Passive Indoor Localization Using WiFi Monitors.

IEEE Transactions on Mobile Computing, *16*(2), 466–481. https://doi.org/10.1109/TMC.2016.2550452

Nordic Semi. (o. J.). Nordic Semiconductor Infocenter. Abgerufen 11. Juli 2018, von http://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52%2Fdita%2Fnrf52%2Fdevelopment%2Fnrf52840_pdk%2Fkit_hw_content.html

Prof. Dr. Bendel, O. (2018, Februar 19). Definition » Industrie 4.0 «. Abgerufen 13. Februar 2018, von http://wirtschaftslexikon.gabler.de/Definition/industrie-4-0.html

Quora. (2017, August 17). What's a QR code? - Quora. Abgerufen 10. Juli 2018, von https://www.quora.com/Whats-a-QR-code

RedBear. (2018). *nRF5x: nRF51822 and nRF52832 based baords, e.g. BLE Nano, RBL_nRF51822, Nano 2 and Blend 2*. C, RedBear. Abgerufen von https://github.com/redbear/nRF5x (Original work published 2017)

Seeed Technology Co. (o. J.-a). Grove - Light Sensor. Abgerufen 10. Juli 2018, von http://wiki.seeedstudio.com/Grove-Light_Sensor/

Seeed Technology Co. (o. J.-b). Grove System. Abgerufen 7. Juli 2018, von http://wiki.seeedstudio.com/Grove_System/

StatCounter. (2018, Februar). Marktanteile der führenden mobilen Browser an der Internetnutzung mit Mobiltelefonen weltweit von Januar 2009 bis Januar 2018. In Statista - Das Statistik-Portal. Abgerufen 22. Februar 2018, von https://de.statista.com/statistik/daten/studie/13120/umfrage/marktanteile-mobiler-browser-bei-der-internetnutzung-weltweit-seit-2009/

Wium, E., & Kervel, F. (2017, Jänner 20). Long Range with CC2640R2F. Abgerufen 13. Juni 2018, von https://training.ti.com/long-range-cc2640r2f

Woolley, M. (2017a). Bluetooth 5 | Go Faster. Go Further.

Woolley, M. (2017b, Februar 13). Exploring Bluetooth 5 - Going the Distance | Bluetooth Technology Website. Abgerufen 10. Juli 2018, von http://blog.bluetooth.com/exploring-bluetooth-5-going-the-distance

Youn, J. H., Ali, H., Sharif, H., Deogun, J., Uher, J., & Hinrichs, S. H. (2007). WLAN-Based Real-Time Asset Tracking System in Healthcare Environments. In *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)* (S. 71–71). https://doi.org/10.1109/WIMOB.2007.4390865

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Abkürzungsverzeichnis

ADC	*Analog Digital Converter*, *Analog Digital Converter*
BR	*Basic Rate*
BS	*Beschleunigungssensor*
EDR	*Enhanced Data Rate*
GAP	*Generic Access Profile*
GATT	Generic Attribute Profile
GFSK	*Gaussian Frequency Shift Keying*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
I²C	*Inter-Integrated Circuit*, *Inter-Integrated Circuit*
ID	*Identifier*
IDE	*Integrierte Entwicklungsumgebung*
LE	*Low Energy*
LED	*Light Emitting Diode - Leuchtdiode*
Li-Ion	*Lithium-Ionen*
LiPo	*Lithium-Polymer*
MCU	*Mikrocontroller*
NFC	*Near Field Communication*
PHY	*Physical Layer*
PWM	*Pulsweitenmodulation*, *Pulsweitenmodulation*
RFID	*Radio-Frequency Identification*
RSSI	Received Signal Strength Indication
SDK	*Software Development Kit*

SIG	*Special Interest Group*
SoC	*System-on-a-Chip*
TI	*Texas Instruments*
UART	*Universal Asynchronous Receiver Transmitter*
UUID	*Unique Universal Identifier*
UWB	*Ultra-wideband*
WLAN	*Wireless Local Area Network*