Implementierung und Untersuchung einer Bluetooth Low Energy Datenübertragung zwischen einem Qualitätskontrollgerät und einem Webbrowser

Luca Brandt

s860463

Bacholorarbeit im Studiengang Elektrotechnik

Fachbereich VII
Beuth Hochschule für Technik Berlin
Deutschland

04.11.20

Betreuer: Prof. Dr. -Ing. Peter Gober

Inhaltsverzeichnis

Abbildungsverzeichnis

Quellcodeverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

APP Applikation

EZAG Eckert und Ziegler AG

BLE Bluetooth Low Energy

TLC Thin Layer Radiochromatograph

DC Dünnschichtchromatografie

DK Development Kit

UART Universal Asynchronous Receiver and Transmitter

TX Data Transmitter

RX Data Receiver

RTS Ready to Send

CTS Clear to Send

USB Universal Serial Bus

SDK Software Developmet Kit

SIG Bluetooth Special Interest Group

GAP Generic Access Profile

NFC Near-Field Communication

GATT Generic Attribute Profile

UUID Universally Unique Identifier

SES Segger Embbeded Studio

SVG Scalable Vector Graphics

JSON JavaScript Object Notation

1 Einleitung

Durch die Weiterentwicklung von Webbrowsern mit Bluetooth-Unterstützung eröffnen sich viele Möglichkeiten. Wofür bisher für jede Anwendung eine Applikation (APP) installiert werden musste, lässt sich jetzt mit Web Bluetooth und Bluetooth Low Energy (BLE) unter anderem ein Sensor direkt auf einer Webseite auslesen. Die Daten können anschließend auf Webseite weiterverarbeitet werden. Die Webseite ist nicht auf eine bestimmte Plattform angewiesen und kann über ein iPhone, einen Windows-PC oder ein Tablet aufgerufen werden, solange das Gerät über einen BLE-Chip verfügt. Auf diese Weise lassen sich alle möglichen Alltagsgeräte auf eine einfache Art in das *Internet of Things* einbinden.

1.1 Aufgabenstellung

Die Eckert und Ziegler Strahlen- und Medizintechnik AG (EZAG) stellt unter anderem Geräte zur Zusammenmischung von Substanzen zur Bildgebung her. Diese Substanzen sind radioaktiv und müssen dementsprechend in den Laboren der Krankenhäuser und der EZAG vor Verabreichung an Patienten überprüft werden. Dies geschieht mithilfe des Qualitätskontrollgeräts, namentlich dem Thin Layer Radiochromatograph (TLC). Diese TLC-Scanner sollen kabellos über eine Webseite bedienbar sein. Es soll sowohl möglich sein eine Messreihe auf der Webseite zu empfangen und darzustellen, als auch Parameter, zur Beeinflussung der Messung, auf der Webseite einzugeben.
Im Rahmen dieser Arbeit wird mit einer Simulation des TLCs gearbeitet. Stellt sich das Resultat für das Unternehmen als nutzbar heraus, wird nachfolgend das Bluetooth-Modul nRF52840 in das Gerät eingebaut. Auch wird recherchiert, wie sich ein Netzwerk von Geräten aufbauen lässt.
Um diese Vorlagen umzusetzen, wird das nRF52840 Development Kit (DK) programmiert. Dazu gehört sowohl die Bluetooth Seite, die Datenübertragung mit dem Simulationsprogramm des TLC-Scanners, als auch die Strukturierung der

Daten. Außerdem wird anschließend die Qualität der Verbindung untersucht. Anhand dieser Evaluierung kann das Unternehmen entscheiden für welche anderen Geräte eine BLE-Datenübertragung in Frage kommt.

Aufgaben:

- Einarbeitung in die Thematik
- Auswahl des BLE-Moduls
- Programmablauf festlegen
- Programmierung des nRF52840: UART
- Aufbereitung der Daten
- Programmierung des nRF52840: BLE
- Programmierung der Webseite
- Evaluation der Verbindung

1.2 Rahmenbedingungen

Die Aufgabenstellung wird im Rahmen einer Bachelorarbeit des Studiengangs Elektrotechnik der Beuth Hochschule für Technik in Zusammenarbeit mit dem Unternehmen Eckert und Ziegler Strahlen- und Medizintechnik AG bearbeitet. Der zeitliche Rahmen für die Erstellung der Arbeit und die Implementierung der Bluetooth Low Energy Datenübertragung beträgt drei Monate.

2 Grundlagen

Um die implementierte Datenübertragung zwischen TLC-Scanner und Webseite nachvollziehen zu können, muss die Funktionsweise und Funktion der verschiedenen technischen Bauteile verstanden werden. In den folgenden Unterkapiteln wird darauf eingegangen welche Funktion ein TLC-Scanner hat, wie das Bluetooth Low Energy Protokoll ganz allgemein aufgebaut ist und welche Funktion die Webseite einnimmt. Außerdem wird ausgeführt, wie die Programmierung des Bluetooth Moduls erfolgt und eine Methode vorgestellt, um die Qualität der Verbindung zu evaluieren.

2.1 TLC-Scanner

Der TLC-Scanner ist ein Qualitätskontrollgerät und prüft die Zusammensetzung von radioaktiven Substanzen.
Eine anderes Gerät, welches die EZAG herstellt, ist das Synthesesystem. Im Synthesesystem wird ein radioaktives Kontrastmittel für die Bildgebung eines Scans zusammengemischt. Das Kontrastmittel heftet sich im Körper an Krebszellen an, und macht diesen dadurch auf dem Scan sichtbar.
Der TLC-Scanner prüft vor jeder Verabreichung der Substanz an die Patienten, ob die Zusammensetzung dieser stimmt.

2.1.1 Anwendungsbereich

Die TLC-Scanner sind in Krankenhäusern anzufinden und werden von den dortigen Labormitarbeitern bedient. Vor jeder Verabreichung des Kontrastmittels an Patienten muss dieses geprüft werden, um die richtige Zusammensetzung garantieren zu können.

2.1.2 Funktionsweise der Dünnschichtchromatografie

Die TLC-Scanner basieren auf der Dünnschichtchromatografie (DC). Bei der DC wird das Wanderungsverhalten unterschiedlicher Moleküle untersucht. Mithilfe des Wanderungverhaltens erfolgt die Evaluierung der untersuchten Substanz.
Bei Chromatografien läuft eine mobile Phase an einer stationären Phase vorbei. Die Stoffe werden dabei von der mobilen Phase, bestehend aus einem Lösungsmittel, losgelöst, und bewegen sich, abhängig von ihrer Polarität, unterschiedlich stark mit der mobilen Phase mit.
Die zu untersuchende Substanz wird auf eine Folie - die stationäre Phase - aufgetragen. Diese Folie wird nachfolgend in ein geschlossenes Gefäß gestellt. Am Boden befindet sich die mobile Phase. Die mobile Phase ist ein Lösemittel, dessen genaue Zusammensetzung per *trial/error* gefunden wird. Wie weit die Moleküle wandern, wird, bei einer konventionellen DC, anschließend unter ultraviolettem Licht untersucht.
Zur Auswertung wird der Quotient aus der Strecke der mobilen Phase und der stationären Phase herangezogen. Diesen Wert nennt man den Rückhaltefaktor. [1]

2.1.3 Radio-TLC der EZAG

Der Name *radio-TLC Scanner* kommt daher, weil die Radioaktivität der Substanz untersucht wird. Bei der *radio-TLC* werden die Punkte nicht per UV-Strahlung untersucht. Stattdessen wird die Folie auf horizontaler Ebene abgefahren, und dabei die Gamma-Strahlung gemessen. Erwartet wird ein *Peak* der Gamma-Strahlung, gefolgt von einem etwas kleineren *Peak*.

[1] Dieses Kapitel ist sinngemäß der Quelle [4] entnommen.

Abbildung 1: TCL-Scanner der Eckert und Ziegler AG

Diese Daten werden visualisiert und ausgewertet. Dadurch kann die richtige Zusammensetzung der Substanz sichergestellt werden.

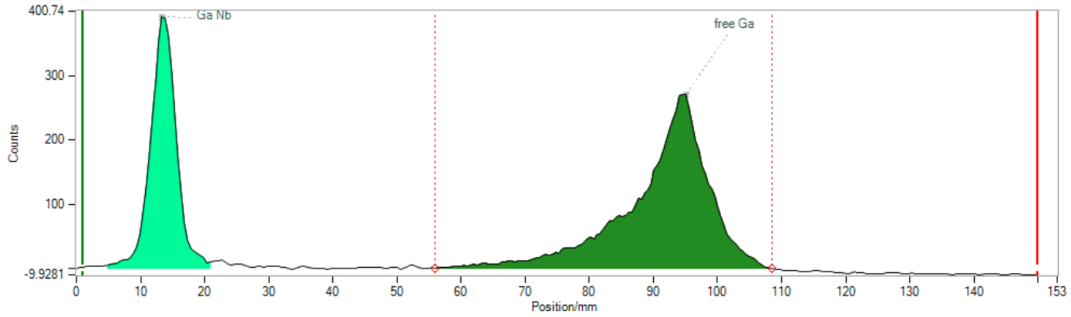

Abbildung 2: Messung mit TLC-Scanner

In der Abbildung 2 ist beispielshalber eine Messung dargestellt, welche mit dem TLC-Scanner von der Eckert und Ziegler AG durchgeführt wurde. Zu erkennen sind die beiden *Peaks*, 82 mm voneinander entfernt. Das freie Gallium wird von der mobilen Phase weiter mitgetragen als der Gallium Nanobody (Ga Nb)
Daraus können, wie in Gleichung 2.1 und 2.2 beispielhaft dargestellt, die

Rückhaltefaktoren berechnet werden.

$$R_{f,free} = \frac{S_{free}}{L} = \frac{95mm}{150mm} = 0.633 \tag{2.1}$$

$$R_{f,NB} = \frac{S_{NB}}{L} = \frac{13mm}{150mm} = 0.087 \tag{2.2}$$

Die Rückhaltefaktoren müssen je nach Messung in einem bestimmten Verhältnis zueinander stehen.

Durch Integration berechnet die Software die Flächen, welche die beiden *Peaks* einschießen. Der *Peak* des Ga Nb muss mindesten 98% der Fläche enthalten, ansonsten ist die Substanz zu stark verunreinigt.

Bei dieser Messung enthält dar Gallium Nanobody nur 35.04% der Fläche. Die Reinheit der untersuchten Substanz entspricht noch nicht den Anforderungen.

2.2 Bluetooth Low Energy

Ursprünglich wurde die Funktechnik BLE von Nokia, unter dem Namen *Wibree*, entwickelt. *Wibree* wurde von der Bluetooth Special Interest Group (SIG) übernommen, und in Bluetooth eingegliedert. Mit der Veröffentlichung der Bluetooth Kernspezifikation 4.0, wurde BLE schließlich veröffentlicht.
Verschiedenen Faktoren ist es zu verdanken, dass BLE mittlerweile ein sehr weit verbreitetes Protokoll zur Funkübertragung ist. Sehr früh wurde BLE in Smartphones und Tablets implementiert. Besonders von Apple wurde die Entwicklung vorangetrieben. Für Entwickler ist BLE besonders attraktiv wegen der geringen Anschaffungskosten, der einfacheren Umsetzung wegen und weil keine Lizenzen gekauft werden müssen. So kann jeder mit einer Idee und einem mit BLE ausgestatteten Mikrocontroller seine Funkübertragung selbst programmieren. Außerdem lässt sich ein BLE-Chip, dem Namen folgend, mit sehr kleinen Leistungen betreiben.
Bluetooth Classic ist durch die Einführung von BLE jedoch nicht veraltet. Bluetooth Classic ist besser geeignet, wenn es darum geht, viele Daten über einen längeren Zeitraum zu transferieren. Zum Musik-Streamingmit kabellosen Kopfhörern zum Beispiel, eignet es sich besser.
Seither wurde Bluetooth 4.1 und Bluetooth 4.2 veröffentlicht, welche neue Verbesserungen enthalten. Unter anderem ist damit bei BLE die Verbindung stabiler, die Leistungsaufnahme geringer und die Übertragungsgeschwindigkeit erhöht. BLE ist abwärtskompatibel, aber nicht direkt kompatibel mit Bluetooth Classic.
Geräte im *Dual Mode* können jedoch sowohl mit Bluetooth Classic, als auch mit BLE Daten austauschen.

2.2.1 Mögliche Arten des Datentransfers

In BLE können Daten ohne Verbindung, sowie auch mit Verbindung transferiert werden. Bei der verbindungslosen Übertragung spricht man *Broadcaster* und *Observer*. Ein *Broadcaster*, das kann zum Beispiel ein Temperatursensor sein, schickt

periodisch *Advertising Packets* aus. Diese *Advertising Packets* enthalten die Daten und können von einer Mehrzahl an *Observern* empfangen werden. Ein *Advertising Packet* enthält maximal 31 Bytes. Wird vom *Observer* eine *Scan Response* geschickt, kann vom *Broadcaster* ein weiteres *Packet*, mit nochmal 31 Bytes, geschickt werden.

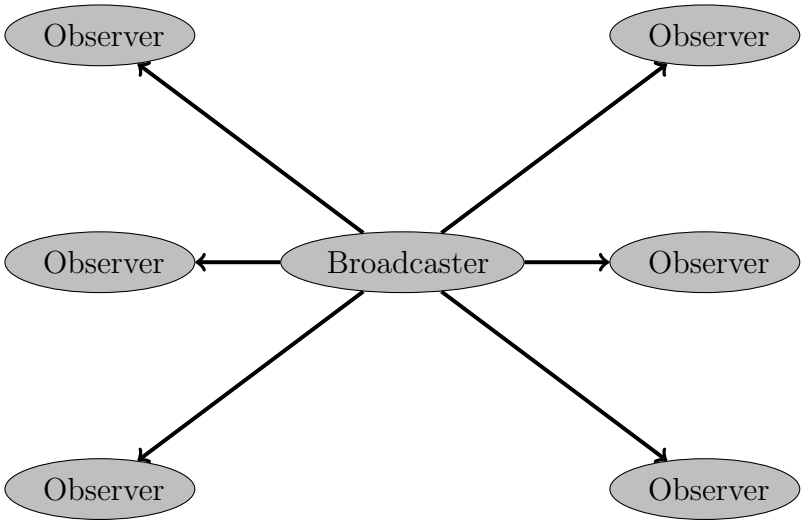

Abbildung 3: Broadcaster und Observer[2]

Diese Art der Datenübertragung hat Vor- als auch Nachteile. Ein großer Vorteil ist, dass die *Packets* von mehreren *Observer* erhalten werden können. Die Nachteile sind, dass 62 Bytes oftmals als Datenübertragung nicht ausreichen und die Daten nur in eine Richtung transferierbar sind. Sollen sensible Daten übertragen werden, ist diese Art der Übertragung auch nicht empfehlenswert, weil nicht gefiltert werden kann, welche *Observer* die Daten empfangen. Um diese Nachteile zu umgehen, kann eine Verbindung hergestellt werden.
Bei einer Datenübertragung mit Verbindung hängt die Bezeichnung von verschiedenen Faktoren ab. Grundsätzlich gibt es *Central/Master* und *Peripherial/Slave*. Der *Central* scannt die Umgebung und wartet auf *Advertising Packets* von *Slaves*. Findet der *Central* in einem *Advertising Packet* Daten, nach welchen er filtert,

[2]Diese Abbildung ist sinngemäß der Quelle [1] entnommen.

stellt er eine Verbindung her. Die hergestellte Verbindung ist fortan exklusiv und die Teilnehmer können Daten in beide Richtungen austauschen.

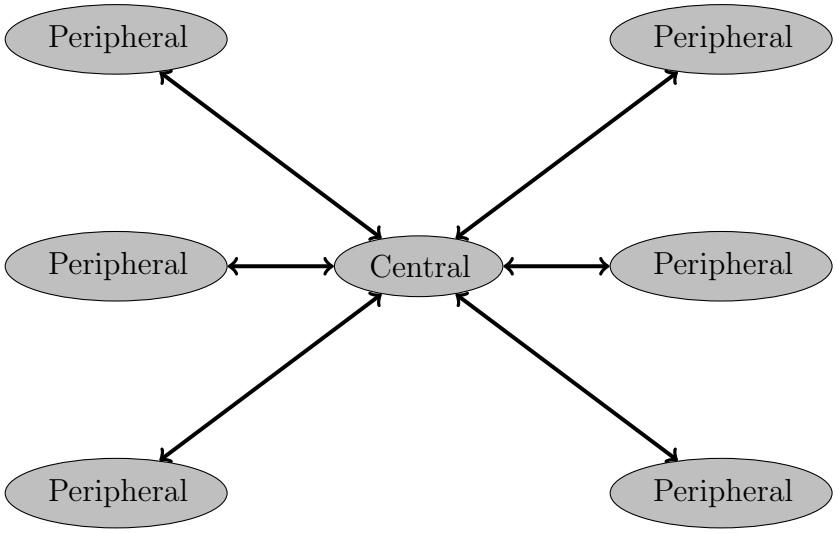

Abbildung 4: Central und Peripheral[3]

Ein *Peripheral* kann maximal in einer Verbindung sein, während ein *Central* mehrere Verbindungen simultan eingehen kann. Je nach Programmierung kann sich ein Gerät als *Center* oder *Peripheral* verhalten, oder sogar beides gleichzeitig.
Der *Central* ist meist ein komplexeres Gerät wie zum Beispiel ein Smartphone. Bei *Peripherals* handelt es sich oftmals um einfache, batteriebetriebene Geräte, welche dem *Central* Informationen liefern. [4]

2.2.2 Verwendung des Generic Access Profile

Im Generic Access Profile (GAP) ist festgelegt, wie Geräte vor und während einer Verbindung miteinander interagieren. Hier wird auch genau festgelegt, wie das *Advertising* vonstatten geht und wie gescannt werden soll.
Auch die *Advertising Data* gehören dazu. Darin enthalten ist unter anderem der

[3]Diese Abbildung ist sinngemäß der Quelle [1] entnommen.
[4]Dieses Kapitel ist sinngemäß der Quelle [1] entnommen.

Name des Gerätes, die *Flags* und optional ein encodierter *Service*. Die *Flags* enthalten Informationen zur Auffindbarkeit des Gerätes und dazu, inwieweit Bluetooth Classic/BLE unterstützt wird. Der *Service*, dies ist Ebene in der logischen Strukturierung der Daten, kann sich auch in der *Scan Response* befinden, muss jedoch vor der Verbindung bekannt sein. Ansonsten ist der *Master* nicht befugt darauf zuzugreifen.
Diese Daten sind in einer bestimmenden Anordnung vorzufinden. Das erste Byte gibt die Länge des folgenden Elementes an. Das nächste Byte spezifiziert, um welche Art von Daten es sich handelt. Dies kann zum Beispiel der vollständige oder abgekürzte Gerätename sein. Abschließend folgen die eigentlichen *Advertising Data*.
Ein Gerät welches die Umgebung scannt, kann dies entweder aktiv oder passiv tun. Bei dem aktiven Scannen sendet der *Scanner* eine *Scan Request* und erhält anschließend, wenn das andere Gerät dies unterstützt, eine *Scan Response*. Damit kann schneller eine Verbindung hergestellt werden.
Um zwischen zwei Geräten eine Verbindung herzustellen, müssen folgende Schritte durchgeführt werden:

- Ein Gerät muss *Avdertising Data* aussenden, während ein anderes Gerät in ausreichender Nähe scannt.
- Die *Advertising Data* müssen zeitlich mit dem Scannen überlappen.
- Der *Central* sendet eine *Connection Request*, welche vom *Peripheral* erhalten werden muss.

Um nach Geräten zu filtern, kann eine *White List* benutzt werden. Dies ist ein Array von Bluetooth Adressen. Damit kann die Anzahl an Geräten begrenzt werden, mit welchen eine Verbindung eingegangen wird. [5]

[5] Dieses Kapitel ist sinngemäß der Quelle [5] entnommen.

2.2.3 Strukturierung der Daten

Im Zusammenhang mit dem Generic Attribute Profile (GATT) spricht man von *Server* und *Client*. Der *Server* ist derjenige Teilnehmer, der die Daten anbietet. Der *Client* interagiert mit dem *Server*, um dessen Daten zu erhalten, diese sind als *Attributes* strukturiert. Der *Server* weist den *Attributes Handles* zu, über welche der *Client* darauf zugreifen kann. Den *Attributes* werden dann noch Erlaubnisse zugewiesen. Die *Attributes* können die Erlaubnisse *Writeable*, *Readable*, diese beiden, oder gar keine Erlaubnisse enthalten. Auch können für verschiedene *Attributes* verschiedene Sicherheitsverfahren festgelegt werden.
Mit den *Characteristics* können folgende Operationen durchgeführt werden:

- Commands
  Wird vom *Client* an den *Server* gesendet und benötigt keine Antwort.

- Requests
  Wird vom *Client* gesendet und benötigt eine Antwort des *Server*.

- Responses
  Dies ist die Antwort auf eine *Request*.

- Notifications
  Als Analogie dafür können Interrupts herangezogen werden. *Notifications* werden vom *Server* unaufgefordert gesendet. Ein Auslöser zum Senden von *Notifications* ist das Ändern eines Wertes.

- Indications
  Diese funktionieren wie *Notifications*, mit dem Unterschied, dass der *Client* den *Server* informiert, dass die Daten empfangen wurden.

- Confirmations
  Dies sind die Datenpackete, welche als Antwort auf *Indications* gesendet werden.

Das GATT ist hierarchisch aufgebaut. Die höchste Ebene ist das *Profile*, darin sind die die *Servics* zu finden. Innerhalb der *Services* sind die *Characteristics* angeordnet. In den *Characteristics* befinden sich die Werte, sowie die denen zuordneten Erlaubnisse, als auch eine - für Menschen lesbare - Beschreibung der Daten. Die *Services* und *Characterisitcs* sind über Universally Unique Identifier (UUID) erreichbar. Das sind entweder 128-Bit oder von der SIG festgelegte 16-Bit Zahlen.
Im Kapitel 5.3 GATT Struktur auf dem Server in Abbildung 9 ist das GATT für diese Anwendung grafisch dargestellt. [6]

2.2.4 Sicherheitsverfahren von BLE

Daten per Funk zu übertragen birgt gewisse Risiken. Diese werden von Herstellern von BLE-Chips und Programmieren möglichst minimiert. Mit der Veröffentlichung von Bluetooth 4.2, wurden die Sicherheitsmechanismen für BLE wesentlich überarbeitet. Die Risiken können in folgende Kategorien unterteilt werden:

- Passive Eavesdropping
  Dabei werden transferierte Daten zwischen den Geräten von einem Dritten abgehört.
- Active Eavesdropping
  Beim *active Eavesdropping* imitiert der Dritte die beiden Geräte. Die Geräte verbinden sich mit dem Dritten, und der Datenaustausch läuft über diesen ab. Auch Daten vom Dritten können in die Übertragung gelangen.
- Privacy and Identity Tracking
  Beim *Tracking* entnimmt der Dritte, mithilfe der Bluetooth-Adresse, den Aufenthaltsort der Geräte und kann dadurch auf die Bewegungsmuster der Person dieses Bluetooth-Gerätes schließen.

Fünf wichtige Konzepte der Sicherheitsverfahren sind *Pairing*, *Bonding Authen-*

[6] Dieses Kapitel ist sinngemäß der Quelle [2] entnommen.

tication, *Encryption* und *Message Integrity*. Beim *Pairing* werden zwischen den Geräten die notwendigen Schlüssel ausgetauscht, um die Verbindung zu verschlüsseln. Durch das *Bonding* werden die Informationen aus dem *Pairing*-Prozess auf den jeweiligen Geräten gespeichert. Auf diese Weise muss das *Pairing* nicht bei jeder neuen Verbindung dieser Geräte durchgeführt werden.
Bei der *Authentication* wird sichergestellt, dass die Geräte die gleichen *Keys*/Schlüssel benutzen. Die *Encryption* verschlüsselt die Daten. Die *Keys* werden zur Entschlüsselung benötigt. Bei der *Message Integrity* kann der Empfänger überprüfen, ob die Daten von dem gewünschten Gerät stammen oder nicht.
Um Schlüssel zwischen zwei Geräten auszutauschen, stehen vier Methoden zur Verfügung. Diese Methoden unterscheiden sich hinsichtlich ihres Sicherheitsgrades und der benötigten Hardware.

- Just Works
  Wie dem Namen zu entnehmen, ist bei dieser Methode keine speziellen Sicherheitsverfahren implementiert, die Geräte lassen sich demnach ohne Prüfung verbinden. Die Schlüssel werden ohne Überprüfung ausgetauscht. Just Works wird zum Beispiel bei Headsets eingesetzt.

- Numeric Comparison
  Bei dieser Methode wird auf beiden Geräten eine sechsstellige Nummer angezeigt. Die Benutzer vergleichen diese Nummern und bestätigen oder negieren deren Übereinstimmung. Dafür wird auf beiden Geräten eine Bildschirmausgabe und eine Art der Eingabe benötigt.

- Passkey Entry
  Hierbei wird auf einem Gerät eine Nummer angezeigt, welche auf dem anderen eingegeben wird. Dafür muss ein Gerät über eine Eingabe verfügen, das andere über eine Ausgabe.

- Out of Band
  Bei dieser Methode wird eine andere Art der kabellosen Datenübertragung

zum Austausch der Sicherheitsinformationen herangezogen. Wird zum Beispiel Near-Field Communication (NFC) benutzt, besteht der Schutz darin, dass sich die Geräte für die Dauer des Verbindungsaufbaus sehr nahe beieinander befinden müssen.

In einer aktiven Verbindung operieren die Geräte in einem festgelegten Sicherheitsmodus.

- LE Security Mode 1

  Level 1: No securtiy

  Level 2: Unauthenticated encryption

  Level 3: Authenticated encryption

- LE Security Mode 2

  Level 1: Unauthenticated data signing

  Level 2: Authenticated data signing

Modus 1 legt fest, inwiefern von Verschlüsselung Gebrauch gemacht wird. In Modus 2 sind die Levels des *Message Integrity* festgelegt.
Jede Verbindung beginnt auf Modus 1, Level 1 und wird dann den Anforderungen entsprechend verändert.
Anzumerken ist, dass einige dieser Verfahren vor der Spezifikation BLE 4.2 nicht zur Verfügung stehen. [7]

[7] Dieses Kapitel ist sinngemäß der Quelle [5] entnommen.

2.2.5 Bluetooth Mesh

Die Eckert und Ziegler AG (EZAG) wird in Zukunft ihre Geräte vermehrt mit Funktechnologie ausstatten. Um dafür den Grundstein zu legen, wird in dieser Arbeit die Funktionsweise von Bluetooth Mesh recherchiert. Damit ist es möglich, ein Netzwerk aus verschiedenen Geräten der EZAG und den Computern und Smartphones der Labormitarbeiter zu erschaffen.
Mit den bisher geschilderten Arten des Datentransfers in Kapitel 2.2.1 Mögliche Arten des Datenverkehrs ist es nur möglich Daten per *One-to-One* und *One-to-Many* zu senden und empfangen. Mit der Einführung von Bluetooth Mesh in 2017, welches auf BLE aufbaut, sind auch *Many-to-Many*-Verbindungen möglich.
Die Teilnehmer eines Netzwerks werden Knoten genannt.

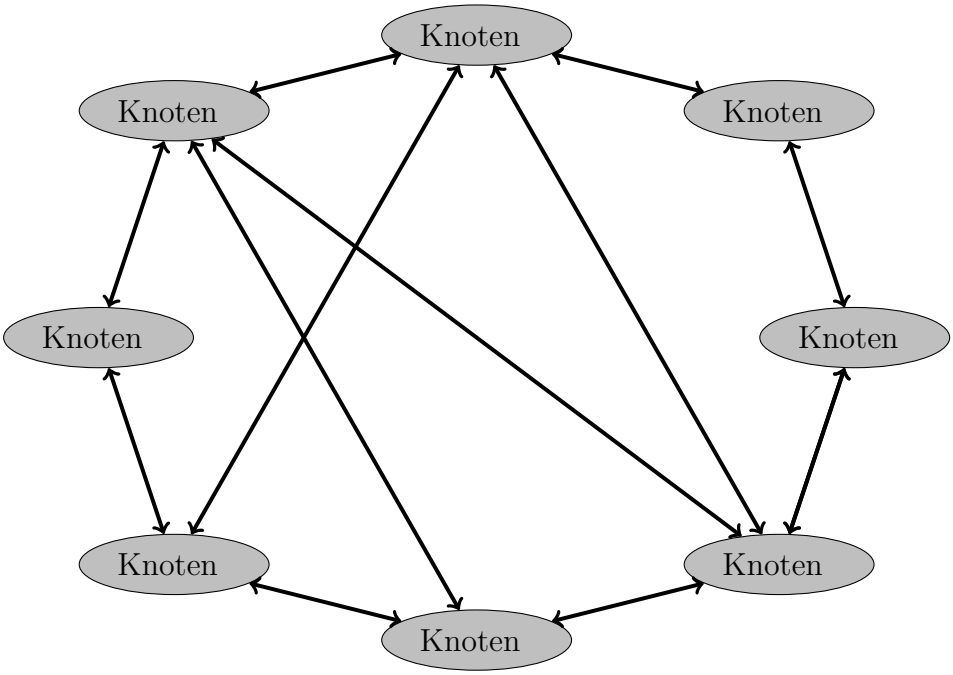

Abbildung 5: Topologie eines Netzwerks

An einem Netzwerk können bis zu 32.000 Geräte beteiligt sein. Daten eines Knotens, bestimmt für einen bestimmten anderen Knoten, können über andere Knoten weitergeleitet werden. Das Weiterleiten wird *Hops* genannt. Dadurch müssen zwei

Knoten, welche Daten austauschen, nicht in direkter Verbindung stehen. Dies vergrößert die maximale Entfernung für den Datentransfer erheblich.
Es gibt vier Arten von Knoten:

- Relais-Knoten
  Diese leiten Nachrichten per *Hops* an das nächste Gerät weiter. Die maximale Anzahl an *Hops* beträgt 127.
- Low-Power-Knoten
  Knoten welche besonders stromsparend arbeiten müssen.
- Freund-Knoten
  Diese Knoten haben ausreichend Energie zur Verfügung und unterstützen den Low-Power-Knoten. Freund-Knoten können Nachrichten speichern und an den Low-Power-Knoten weiterleiten, wenn diese angefordert werden.
- Proxy-Knoten
  Diese Art von Knoten stellen eine GATT-Schnittstelle zu BLE-Geräten her, welche keinen Mesh-Anschluss haben.

In Mesh Netzwerken wird mit *Messages* gearbeitet. Dabei gibt es verschiedene Arten von *Messages*:

- Get Message
  Diese erfragen andere Geräte nach deren Status.
- Set Message
  Dies ist eine Nachricht, welche den Wert eines Status ändert.
- Status Message
  Diese *Messages* teilten anderen Geräten den momentanen Status mit. Dies erfolgt entweder von diesem Gerät aus, als Antwort auf eine *Get Message* oder als Antwort zur Bestätigung einer *Set Message*.

Messages funktionieren nach dem *Publishing*- und *Subscribing*-Prinzip. Das Sen-

den von Daten an ein Gerät wird *Publishing* genannt. Beim *Subscribing* werden die Geräte mit einer Adresse konfiguriert. Diese Geräte können danach über diese Adresse erreicht werden.
Das Senden der Daten wird *Flooding* genannt. Dabei wird die Nachricht an alle Geräte geschickt und von den jeweiligen Gerät weitergesendet, bis der vorgesehene Empfänger die Nachricht bekommt. Bei Bluetooth Mesh kommen einige Methoden zum Einsatz, um die Nachrichten auf eine gezieltere Art und Weise zum Empfänger zu bringen.
Fällt ein einzelnes Gerät aus, die Netzwerkdichte aber groß genug ist, erreicht die Nachricht dennoch ihr Ziel. Diese Eigenschaft wird *Self-Healing* genannt und macht Bluetooth Mesh zu einer sehr zuverlässigen Methode der Datenübertragung.
Der *Provisioning Process* legt fest, wie Geräte dem Mesh hinzugefügt werden können. Ein Gerät, welches ein neues Gerät zum Mesh hinzufügen soll, wird *Provisioner* genannt. Erhält dieses Gerät ein Datenpaket, und dieses Gerät steht nicht auf der *Black List*, sendet der *Provisioner* eine Einladung zurück. Das neue Gerät antwortet darauf mit einem Datenpaket, welches Informationen zu dem Gerät enthält. Bei Bluetooth Mesh wird jedes Gerät nur mit durchgeführten Sicherheitsverfahren in das Mesh aufgenommen. Deshalb werden anschließend die *Keys* ausgetauscht und eine Authentifizierung vorgenommen. Zuletzt wird dem neuen Gerät eine Adresse zugewiesen. Daraufhin ist das neue Gerät Teil des Netzwerkes.
Wird ein Gerät aus dem Netzwerk entfernt, ist es sinnvoll dieses auf die vorher erwähnte *Black List* zu setzten. Denn die Sicherheitsinformationen können noch auf dem Gerät vorhanden sein. Mithilfe der *Black List* kann das Gerät nicht in das Netzwerk eindringen. Werden die *Keys* erneuert, werden Geräte auf der *Black List* nicht miteinbezogen. Die neuen *Keys* werden von verschiedenen Geräten zu unterschiedlichen Zeitpunkten empfangen. In dieser Übergangszeit sind die sowohl neuen, als auch die alten *Keys* gültig. [8]

[8] Dieses Kapitel ist sinngemäß der Quelle [7] entnommen.

2.3 Verwendung und Funktionsweise des UART

Um mit einem Computer kommunizieren zu können, kann ein *Universal Asynchronous Receiver and Transmitter (UART)* verwendet werden. Damit ist eine Möglichkeit der Ein- und Ausgabe per digitaler, serieller Schnittstelle geschaffen. Der UART wurde bereits 1962 eingeführt und ist in bestimmten Konstellationen noch immer gut geeignet. Zum Beispiel für die Kommunikation zwischen PC und Mikrocomputer ist der UART heute noch Standard.

2.3.1 Funktionsweise

Die Daten, die transferiert werden, sind in einer bestimmten Struktur angeordnet. Diese Struktur besteht aus einem Start-Bit, fünf bis neun Daten-Bits, einem optional *Parity-Bit* und einem Stop-Bit. Die Bits in dieser Struktur entsprechen einem Datenpaket mit der maximalen Nutzgröße von neun Bits. Wird das *Parity-Bit* benutzt, reduziert sich die maximale Größe der Daten-Bits auf acht Bits. Das *Parity-Bit* dient der Überprüfung der übertragenen Daten. Dabei werden die Zustände der Bits der empfangenen Daten untersucht. Für eine gerade Anzahl an logischen Einsen wird als *Parity-Bit* eine Null geschickt, ansonsten eine Eins. Vom Empfänger werden anschließend die Daten-Bits und das *Parity-Bit* verglichen, um im Fehlerfall darauf reagieren zu können.
Dabei werden Spannungswerte von -3V bis -15V als logische Null gewertet, Werte von 3V bis 15V als Eins.
Der Wert des Start-Bits entspricht im Normalzustand einer logischen Eins. Um den Transfer eines Packets anzukündigen, wird das Start-Bit für eine bestimmte Zeit von Eins auf Null gesetzt. Der UART-Empfänger bemerkt die Veränderung, und beginnt die gesendeten Daten mit der passenden Frequenz zu lesen.
Das Ende des Packets wird mithilfe des Stop-Bits angekündigt. Das Stop-Bit wird dafür für eine bestimmte Zeitspanne auf Null gesetzt.
Die Geschwindigkeit, mit welcher die Daten übertragen werden, wird Baudrate

(Bd) genannt.

$$1Bd = \frac{1}{s} \tag{2.3}$$

Dies entspricht einem Symbol pro Sekunde. In der Praxis sind Werte von $Bd = 115\,200\,\frac{1}{\mathrm{s}}$ oder $Bd = 9600\,\frac{1}{\mathrm{s}}$ üblich. Die Baudrate muss auf beiden teilnehmenden Geräten eingestellt werden, und darf maximal 10% Abweichung betragen. Ansonsten funktioniert die Datenübertragung nicht. [9]

2.3.2 Verbindung der teilnehmenden Geräten

Einer der Vorteile der Datenübertragung per UART ist, dass der Datentransfer mit nur zwei Leitungen realisiert werden kann.

Die Datenleitungen werden über Kreuz angeschlossen. Die Data Transmitter

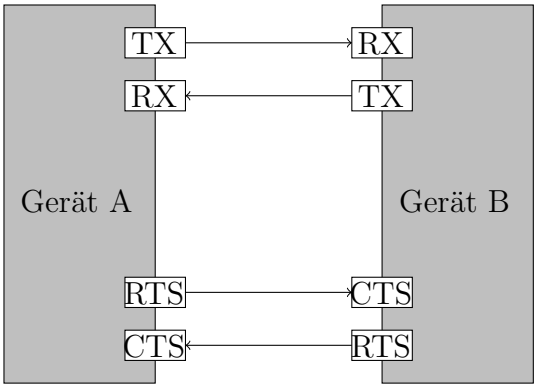

Abbildung 6: Verbindungen UART

(TX)-Leitung des Gerätes A wird mit dem Data Receiver (RX) des Gerätes B verbunden. Die RX-Leitung des Gerätes A wird dementsprechend mit dem TX-Anschluss des Gerätes B verbunden. Auf diese Weise ist die Leitung zum Senden des Gerätes A, mit dem Empfänger des Gerätes B verbunden. Die Verkabelung der RX-Leitung funktioniert analog.

Zusätzlich lässt sich mit den Leitungen Clear to Send (CTS) und Ready to Send (RTS) das *Flow Control* implementieren. Dies dient der erweiterten Sicherheit der

[9] Dieses Kapitel ist sinngemäß der Quelle [6] entnommen.

Datenübertragung. Dadurch kann ein Gerät übermitteln, ob und wann es Daten empfangen oder senden kann. Die Anschlüsse werden auch dafür über Kreuz verbunden. [10]

2.3.3 UART zu USB

Um eine serielle Schnittstelle von Computer oder TLC-Scanner zu dem BLE-Modul zu implementieren, wird eine Übersetzung von UART zu Universal Serial Bus (USB) benötigt. Dies wird mit einem UART zu USB-Adapter erreicht. Dafür werden auf der UART-Seite die Datenleitungen TX und RX, das *Flow Control* mit CTS und RTS und die Spannungsversorgung verbunden. Auf der USB-Seite wird der Mikro-USB-Anschluss angesteckt.
Die Auswahl fiel auf das PmodUSBUART von Digilent. Auf dem Board wird der LCL-Anschluss mit Spannung versorgt, weil das verbundene Board, das BLE-Modul, extern mit Spannung versorgt wird.

2.4 Auswahl und Programmierung des BLE-Moduls

Bei der Auswahl des BLE-Chips wurde besonders auf folgende Kriterien geachtet:

- Preis
  Um die Geräte möglichst preiswert verkaufen zu können, werden die Produktionskosten minimiert.
- Größe
  Das Modul wird in ein bereits existierendes Gerät eingebaut. Der Platz darin ist begrenzt.
- Verwendbarkeit in der Zukunft
  Das verwendete Modul sollte in Zukunft ausbaubar sein. Das heißt, es soll

[10] Dieses Kapitel ist sinngemäß der Quelle [6] entnommen.

Bluetooth 5 und Bluetooth Mesh fähig sein.

- Reputation
  Ein weiterer Faktor bei der Auswahl war die Reputation des Unternehmens. Bei der Auswahl wird auf ein Unternehmen gesetzt, von welchem ein solides Produkt erwartet werden kann.

Die Auswahl fiel auf den nRF52840 des Unternehmens Nordic Semiconductor. Der Chip kann mit dem Devolopment Kit nRF52840 entwickelt werden. Das fertige Programm kann anschließend entweder auf dem nRF52840 Dongle verwendet werden, oder auf einer eigens hergestellten Platine der EZAG eingesetzt werden.
Zur Entwicklung des Programms wird das nRF52840 Development Kit (DK) von Nordic Semiconductor verwenden. Dies ist per mikro-USB Verbindung programmierbar. Der Vorteil des DK ist, dass darauf *debugging* möglich ist. Das fertige Programm kann anschließend auf dem nRF52840 Dongle verwendet werden. Die Programmierung erfolgt über das vom Hersteller empfohlene Segger Embbeded Studio (SES).
Die Programmierung des Dongles erfolgt etwas anders. Wird in SES *built* angewählt, produziert SES eine Datei im hexadeximalformat. Diese Datei und das *Softdevice* werden dann über die App *nRF Connect: Programmer* per USB-Verbindung auf den Dongle geladen.
Das Programm für das Bluetooth-Modul basiert auf dem nRF5 Software Developmet Kit (SDK) v17.0.2 von Nordic Semiconductor. Verwendet wird das *Blinky Example* in der pca10056 Version. Darin sind zum Beispiel Dinge wie das *Advertising* und die Definitionen für das Board bereits vorhanden. Das *Blinky Example* wird den Anforderung des Datentransfers entsprechend verändert und ergänzt.

2.4.1 Softdevice von Nordic Semiconductor

Das *Softdevice* ist ein Stück Programmcode, welches die Bluetoothkommunikation übernimmt. Es kann von der Webseite von Nordic Semiconductor heruntergeladen

werden. Für den nRF52840 wird das *SoftDevice* 140 verwendet.

2.5 Webseite

Am weitesten fortgeschritten ist die Entwicklung von Web Bluetooth auf Google Chrome, verglichen mit den anderen Browsern. Die Verwendung dieses Browser wird für das Aufrufen dieser Webseite empfohlen, denn andere Browser unterstützen bestimmte Funktionen, die verwendet werden, noch nicht.
Die Webseite, welche für diese Arbeit programmiert wurde, wird vom Benutzer aufgerufen und bedient. Die Seite besteht aus einem *Interface* und einem Datenaustausch, welcher im Hintergrund stattfindet.
In der Konstellation mit dem Dongle, nimmt die Webseite die Funktion des *Master/Client* ein.

2.5.1 Grafische Gestaltung der Webseite

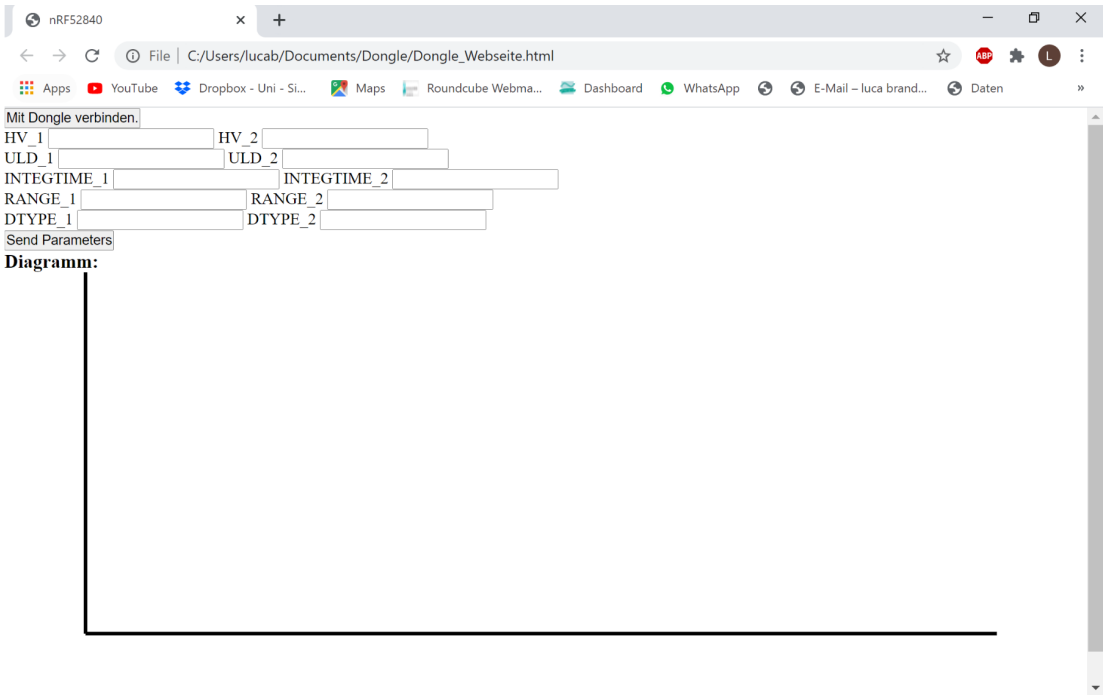

Abbildung 7: Interface Webseite

Die Gestaltung der Webseite ist, weil es sich um einen Prototyp handelt, schlicht gehalten. Im Vordergrund steht die Funktionalität.
Die Webseite wurde für diese Abbildung mit einem Laptop aufgerufen.
Oben links ist der Knopf *"Mit Dongle verbinden"* zu sehen. Über diesen Knopf wird eine Liste der Bluetooth-Geräte, die am *Advertisen* sind, aufgerufen. Darunter sind die Eingabefelder für den Benutzer zu sehen. Über diese Feder ist zum Beispiel die Hochspannung 1 einstellbar. Die Werte werden anschließend mit *"Send Parameters"* gesendet.
Unterhalb davon wird eine Fläche für die Darstellung der Messwerte freigehalten.

2.5.2 Aufrufen der Webseite

Um die Webseite aufrufen zu können, muss die Datei in erreichbarem Speicher untergebracht werden. Dazu kommen verschiedene Varianten in Frage:

- Lokale Speicherung
  Die Datei wird auf dem Computer, auf welchem die Webseite aufgerufen werden soll, gespeichert. Dies ist in der Praxis, mit mehreren Computern, nicht empfehlenswert. Wird die Webseite zum Beispiel anders programmiert, muss die Datei überall neu gespeichert werden.

- Web Hosting
  Die Datei kann auf einem offiziellen Server gespeichert werden. Der Nachteil ist, dass dies teuer werden kann. Besonders wenn die Webseite in Zukunft in verschiedenen Ländern auf der ganzen Welt aufgerufen wird.

- Github Server
  Die Datei der Webseite kann kostenlos, und auf eine einfache Weise, auf Github gespeichert werden.

Die Option Github ist für diese Anwendung die am Besten geeignete.

2.6 Up-and-Down Methode

Für die Untersuchung der Bluetooth-Abbruchentfernung wird später in dieser Arbeit die Up-and-Down Methode benutzt. Damit kann die Entfernung, bei welcher die Verbindung zu 50% abbricht, abgeschätzt werden. Der Vorteil der Up-and-Down Methode ist, dass die Versuchsergebnisse trotz relativ kleiner Anzahl an Versuchen statistisch gut abgesichert ist.
Bei der Up-and-Down Methode wird zuerst eine Entfernung zwischen den Geräten gewählt, welche bestimmt nicht zu einem Verbindungsabbruch führt. Danach wird die Entfernung um eine vorher festgelegte Entfernung vergrößert. Diese Entfernung sollte zwischen 3% bis 10% der erwarteten Abbruchentfernung liegen. Bricht die Verbindung nach Erhöhung der Entfernung ab, wird die zuletzt gehaltene Entfernung wiederhergestellt. Wird diese Entfernung gehalten, wird die Entfernung wieder um die Differenzstrecke vergrößert, ansonsten um die Differenzstrecke verkleinert. Dieser Vorgang wird für eine festgelegte Anzahl an Entfernungen wiederholt. Die Laufvariable i wird bei dem ersten Verbindungsabbruch auf Eins gesetzt, und fortan für jede neue Entfernung inkrementiert. Die festgelegte Anzahl an Schritten ist auf die Laufvariable bezogen.
Nach der Messung wird ausgewertet, ob das Event Verbindungsabbruch oder Nichtverbindungsabbruch häufiger aufgetreten ist. Das seltenere Event tritt k-mal auf, das häufigere q-mal. Der Nullte Schritt wird dem weniger häufig Auftretenden Event zugewiesen.
Je höher die Anzahl des Vorgangs und je kleiner die festgelegte Entfernungsdifferenz, desto genauer das Ergebnis.

Δs festgelegte Entfernungsänderung
s_{00} Erster Abstand der garantiert zu keinem Abbruch führt
s_{l1} Entfernung bei erstem Verbindungsabbruch
s_{ln} n-te Entfernung

n Anzahl Messungen
i Laufvariable ab erstem Verbindungsabbruch
s_{d50} Entfernung, die zu 50% zum Verbindungsabbruch führt
k Anzahl Verbindungsabbrüche
q Anzahl Nicht-Abbrüche
s_A Standartabweichung
l Schritte

Zuerst werden die Koeffizienten A und B berechnet. Die Koeffizienten gewichten die Häufigkeit des Auftretens der verschiedenen Entfernungen.

$$A = \sum_{i=0}^{j} i * k_i \tag{2.4}$$

$$B = \sum_{i=0}^{j} i^2 * k_i \tag{2.5}$$

Aus den Messwerten und den Koeffizienten kann anschließend die Entfernung berechnet werden, bei welcher die Verbindung zu 50% Sicherheit abbricht.

$$s_{d50} = s_0 + \Delta s \left(\frac{A}{k} \pm \frac{1}{2} \right) \tag{2.6}$$

In der Klammer wird 0.5 abgezogen, wenn der Verbindungsabbruch das seltenere Event ist und addiert, wenn der Nicht-Abbruch das seltenere Event ist.
Abschließend wird noch die Standardabweichung berechnet.

$$s_A = 1.62 * \Delta s \left(\frac{kB - A^2}{k^2} + 0.029 \right) \tag{2.7}$$

3 Systemdesign

Zuerst einen kurzen Überblick wie die Teilnehmer miteinander interagieren. Dadurch kann das Vorgehen im späteren Verlauf dieser Arbeit besser nachvollzogen werden.

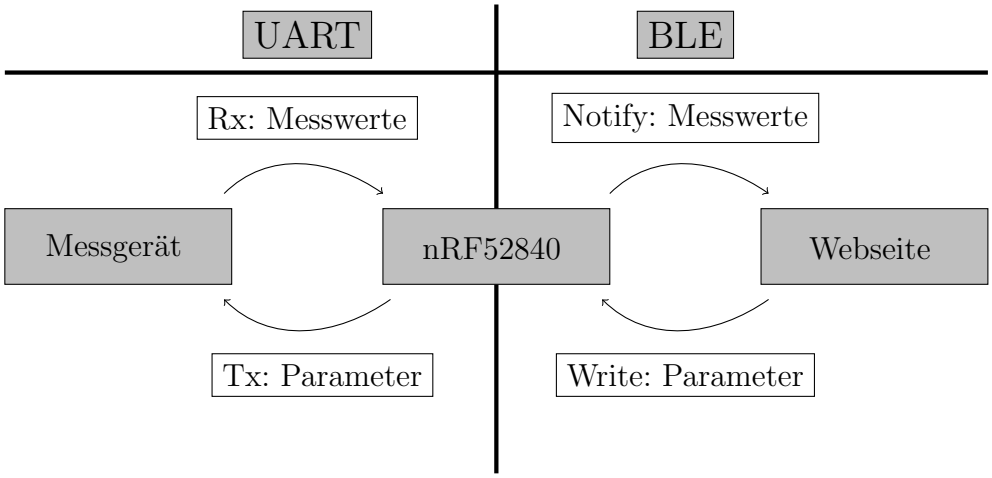

Abbildung 8: Systemdesign

Die Bezeichnungen der Leitungen des UART sind vom nRF52840 aus festgelegt, die BLE-Verbindungen vom *Client* - der Webseite - aus.
Wie in Abbildung 8 dargestellt, ist der nRF52840 per serieller Schnittstelle mit dem TLC-Scanner, oder der TLC-Simulation, verbunden. Das heißt, das BLE-Modul ist über den USB zu UART-Adapter in das Messgerät eingebaut. Der TLC-Scanner sendet per UART die Messwerte an das BLE-Modul und erhält die Parameter, welche vom Benutzer eingegeben werden auf der RX-Leitung.
Vom BLE-Modul aus werden die Daten per BLE transferiert. Auf einem Webbrowser wird die Webseite aufgerufen und eine Verbindung mit dem nRF52840 hergestellt. Anschließend ist die Verbindung von TLC-Scanner und Webseite vollendet, der Datentransfer kann beginnen. Der *Server* schickt, wenn sich die Messwerte aus dem TLC-Scanner ändern, per *Notifications* die neuen Werte an die Webseite. Per *Write* werden von der Webseite die Parameter an den nRF52840 gesendet.

4 Implementierung der seriellen Schnittstelle

Um Daten zwischen dem Qualitätskontrollgerät und dem BLE-Modul transferieren zu können, wird eine serielle Schnittstelle implementiert. Dafür wird ein UART verwendet.

4.1 Initialisierung UART

Zuerst wird der UART initialisiert. Den Parametern für die Datenübertragung werden darin die Werte zugewiesen. Dies ist der erste - und bisher einzige - UART

Listing 1: Initialisierung UART (rm_uart.c)

```
21 nrf_drv_uart_t app_uart_inst = NRF_DRV_UART_INSTANCE(0);
22
23 void rm_uart_init()
24 {
25     nrf_drv_uart_config_t cfg = NRF_DRV_UART_DEFAULT_CONFIG;
26
27     cfg.pselrxd                = NRF_GPIO_PIN_MAP(0,31);
28     cfg.pseltxd                = NRF_GPIO_PIN_MAP(0,30);
29     cfg.pselcts                = NRF_GPIO_PIN_MAP(0,4);
30     cfg.pselrts                = NRF_GPIO_PIN_MAP(0,3);
31     cfg.baudrate               =          \
          UART_BAUDRATE_BAUDRATE_Baud115200;
32     cfg.parity                 = false;
33     cfg.use_easy_dma           = false;
34
35     ret_code_t err_code;
36     err_code = nrf_drv_uart_init(&app_uart_inst, &cfg, 0);
37     APP_ERROR_CHECK(err_code);
38 }
```

in Betrieb, daher die nullte Instanz. Als Sender wird, wie in Listing 1 dargestellt, der Pin (0,30), und als Empfänger Pin (0,31) festgelegt. Für das *Flow Control* werden die Pins drei und vier benutzt. Die Baudrate beträgt $115\,200\,\frac{1}{\mathrm{s}}$, auf das Parity-Bit und Easy-DMA wird verzichtet. Die restlichen Parameter sind auf die Standardeinstellungen gesetzt.

Der Funktion nrf_drv_uart_init() wird die Instanz, die Konfiguration und eine Null übergeben. Die Null setzt den UART auf *Blocking Mode*, anstatt mit *Handles* zu arbeiten.

4.2 Funktionsweise Datenaustausch

Nach der Initialisierung werden auf dem nRF52840 eine Funktion zum Senden der Daten, und eine zum Empfangen implementiert.
Die Funktion, um Daten zum Qualitätskontrollgerät zu schicken, wird aufgerufen, wenn der nRF52840 neue Daten über BLE erhält. Der Funktion rm_uart_put() werden die Parameter von der Webseite übergeben.

Listing 2: UART: Daten senden (rm_uart.c)

```
51 void rm_uart_put(char_x char_1)
52 {
53     ret_code_t err_code;
54     tx_buffer[0] = char_1.HV;
55
56     err_code = nrf_drv_uart_tx(&app_uart_inst, tx_buffer, 1);
57     APP_ERROR_CHECK(err_code);
58 }
```

Beispielhaft wird hier die Hochspannung für den TLC-Scanner transferiert.
Um Daten vom TLC-Scanner zu empfangen, wird permanent die rm_uart_get()-Funktion aufgerufen. Da die Daten im *Blocking Mode* empfangen werden, kehrt die Funktion erst beim Erhalten neuer Daten zurück.

Listing 3: UART: Daten empfangen (rm_uart.c)

```
61 uint8_t rm_uart_get(void)
62 {
63     ret_code_t err_code;
64     rx_buffer[0] = 0;
65
66     err_code = nrf_drv_uart_rx(&app_uart_inst,rx_buffer,1);
67     APP_ERROR_CHECK(err_code);
68
69     return *rx_buffer;
70 }
```

Die erhaltenen Daten werden über die main()-Funktion weitergeleitet. Von dort aus werden die Daten der Funktion übergeben, welche den *Client* per *Notifications* die neuen Daten sendet.
Der Rückgabewert der beiden Funktionen ist die Art des Fehlers und wird in err_code gespeichert. Mit APP_ERROR_CHECK() wird überprüft, ob ein Fehler aufgetreten ist.

5 Implementierung des Bluetooth Dienstes

In diesem Kapitel wird ausgeführt, wie das Bluetooth Modul - der nRF52840 - auf der Bluetooth-Seite programmiert ist, und wie die Webseite aufgebaut ist, um die gewünschte Funktion umzusetzen. Das beinhaltet, welche Rollen die Teilnehmer einnehmen, das Aufbauen einer Verbindung und die Strukturierung der Daten auf dem Server.

5.1 Rollen der Teilnehmer

Um Daten in beide Richtungen schicken zu können, muss, nach Kapitel 2.2.1 Mögliche Arten des Datenverkehrs, eine Verbindung hergestellt werden.
Das BLE-Modul nimmt beim Verbindungsaufbau die Rolle des *Peripherals* ein, die Webseite die des *Central*.

5.2 Verbindungsaufbau

Eine Verbindung entsteht, wenn die Webseite, welche als *Central* agiert, in einem *Advertising Packet* Daten erhält, welche seinem Filter entsprechen. *Client* und *Server* müssen also übereinstimmen.

5.2.1 Server-Seite

Auf dem nRF52840 ist in encodierter Form das *Advertising Packet* und die *Scan Response* festgelegt.
Im *Advertising Packet* befindet sich der Gerätename und die *Flags*. Um Platz zu sparen ist der Gerätename auf *"DON"* festgelegt, und die *Flags* setzen ein *Byte*, um das Gerät auf *General Discovery Mode* zu konfigurieren. Sobald der Dongle mit Spannung versorgt wird, sendet er periodisch sein *Advertising Packet* aus und ein *Central* - die Webseite - kann sich damit verbinden. Das heißt, dass das Gerät von allen *Scannern* für eine unbegrenzte Zeit auffindbar ist.

Möchte sich ein *Central* mit dem Dongle verbinden, wird die *Scan Response* angefordert. In der *Scan Response* befindet sich dann die *UUID* des *Service*, auf welchen später zugegriffen wird.
Wird eine Verbindung hergestellt, beendet das BLE-Modul das *Advertising*.

5.2.2 Client-Seite

Auf der Webseite wird auf Knopfdruck vom Benutzer nach Bluetooth Low Energy Geräten gescannt, welche am *Advertisen* sind. Um andere Geräte zu ignorieren, wird nach dem Namen des nRF52840, *"DON"*, gefiltert. Dadurch wird nicht eine lange Liste aller Geräte angezeigt die am *Advertisen* sind und der nRF52840 muss gesucht werden, sondern dieser ist die einige Option. Die Deklaration der

Listing 4: Scanning nach Dongle (JavaScript)

```
126 async function bind_bt() {
127  try {
128   console.log("Requesting Dongle...");
129   const device = await navigator.bluetooth.requestDevice({
130     filters: [{ name: 'DON' }],
131     optionalServices:
            ['00000001-0000-0000-0000-000000000000']
132     })
133     console.log("Connected with: " + device.name);
```

Funktion enthält ein *async*. Dadurch wird alles innerhalb der Funktion asynchron ausgeführt. Wird zum Beispiel mit *await* ein Gerät angefordert, wartet das Programm an dieser Stelle, bis ein Gerät gefunden wird.
Die bind_bt() Funktion wird aufgerufen, wenn vom Benutzer auf der Webseite die Option *"Mit Dongle verbinden"* gewählt wird.
Wird anschließend auf der Webseite vom Benutzer der nRF52840 ausgewählt, wird eine exklusive Verbindung zwischen der Webseite und dem nRF52840 hergestellt. Unter *optionalServices* wird die *UUID* des Service angegeben. Die *Service-UUID* besteht aus praktischen Gründen an achter Stelle aus einer Eins und ansonsten

aus Nullen. In der *Scan Response* ist diese *UUID* enthalten, und daher kann die Webseite darauf zugreifen.
Unter dem *device*-Objekt wird das verbundene Gerät, bei erfolgreicher *requestDevice*-Methode, gespeichert.
Um Fehler abzufangen, ist die Funktion von *try* und *catch* eingeschlossen. Im Fehlerfall wird der Fehler ausgegeben, und es kann entsprechend gehandelt werden.

5.3 GATT Struktur auf dem Server

Auf dem nRF52840 muss die GATT Struktur festgelegt werden. Um die Funktion des Messgerätes möglichst effektiv umzusetzen, wird ein *Serivce* mit zwei *Characteristics* implementiert.

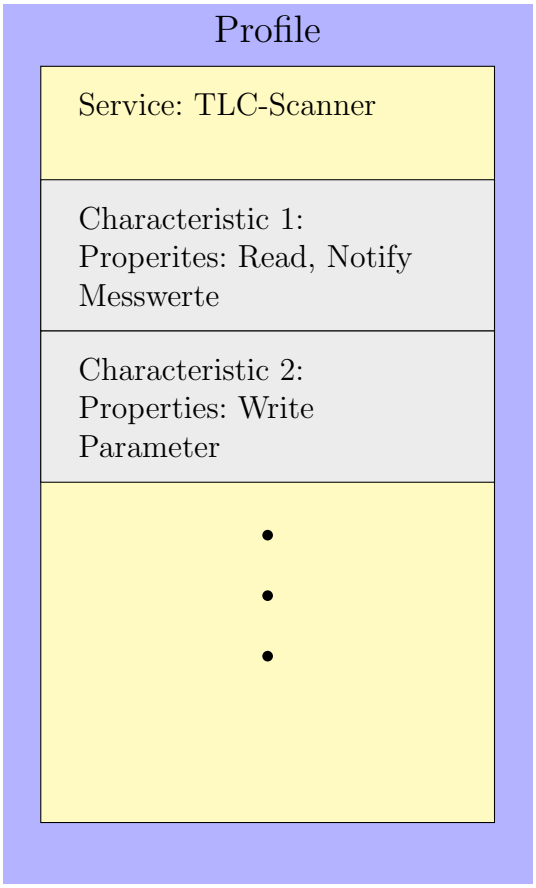

Abbildung 9: Implementierte GATT Struktur auf dem Server[11]

Wie in der Abbildung 9 zu sehen ist, befindet sich der *Service: TLC-Scanner* innerhalb des *Profile* und innerhalb davon die beiden *Characteristics*.
In der *Characteristic 1* befinden sich die Messwerte. Diese werden in das GATT geschrieben, und mit *Notify* wird der *Client* benachrichtigt, wenn sich die Werte

[11]Diese Abbildung ist sinngemäß der Quelle [3] entnommen.

ändern. Diese werden auf diese Art auf der Webseite permanent aktualisiert.
Die *Characteristic 2* hat das *Property Write*. Dies erlaubt, dass Daten vom *Client* in das GATT geschrieben werden können. Der nRF52840 bekommt so die Parameter von der Website, um die Messung nach Belieben zu beeinflussen.
Die drei Punkte sollen versinnbildlichen, dass in Zukunft darauf aufgebaut werden kann. Sollten irgendwann mehr Daten transferiert werden, oder der ganze *Service* für eine andere Applikation verwendet werden, lässt sich dies ohne großen Aufwand realisieren. Dem *Service* können auch neue *Characteristics* hinzugefügt werden.

5.3.1 GATT Service: Rate Meter

Zuerst wird ein neuer Service vorbereitet. Dafür wird eine 128-Bit Nummer benötigt. Diese kann man sich selbst auswählen, oder von der Webseite *www.uuidgenerator.net* generieren lassen. In der Variable wird die Nummer nach *Little Engine* gespeichert. Dies ist die *Base UUID*. Für die folgenden *UUIDs* können dann nur 16-Bit Nummern gewählt werden und in die *Base* an fünfter bis neunter Stelle eingefügt werden.
Anschließend wird die Funktion aufgerufen. Wobei die base_uuid die 128-Bit Num-

Listing 5: Hinzufügen der *UUID* zu *BLE Stack* (ble_lbs.c)

```
145  sd_ble_uuid_vs_add(&base_uuid, &p_lbs->uuid_type);
```

mer ist, und p_lbs ein Zeiger auf einen *Struct Member*, welcher den Typ der *UUID* angibt. In diesem Fall gibt der Typ an, dass es sich um eine 128-Bit Nummer handelt.
Anschließend wird der *Service* ins GATT geschrieben. Als erster Parameter wird der Funktion übergeben, dass es sich um einen *Primary Service* handelt. Ein *Secondery Service* wird von einem *Primary Service* aufgerufen. Dies ist hier nicht der Fall ist, handelt es sich um einen *Primary Service*. An zweiter Stelle wird die

Listing 6: Hinzufügen des *Service* zu *GATT* (ble_lbs.c)

```
151 sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY, \
                &ble_uuid, &p_lbs->service_handle);
```

Adresse von einem *Struct* übergeben, und enthält die 16-Bit Nummer des *Service* und den Typ der UUID. An dritter Stelle wird vom *BLE Stack* ein *Handle* für diesen *Serivce* bereitgestellt.

5.3.2 GATT Charakteristiken

Anschließend können dem *Service* die *Characteristics* hinzugefügt werden. In Listing 7 ist das Hinzufügen einer *Characteristic* und das Festlegen der Parameter dargestellt.

Listing 7: Hinzufügen der *Characteristic* zum *GATT* (ble_lbs.c)

```
174 ble_add_char_params_t char1_params;
175
176 memset(&char1_params, 0, sizeof(char1_params));
177 char1_params.uuid              = LBS_UUID_CHAR_1;
178 char1_params.uuid_type         = p_lbs->uuid_type;
179 char1_params.init_len          = sizeof(uint8_t);
180 char1_params.max_len           = sizeof(uint8_t);
181 char1_params.char_props.read   = 1;
182 char1_params.char_props.notify = 1;
183 char1_params.read_access       = SEC_OPEN;
184 char1_params.cccd_write_access = SEC_OPEN;
185 characteristic_add(p_lbs->service_handle, \
186                    &char1_params, \
187                    &p_lbs->chars1_handle);
```

Mithilfe des Datentyps ble_add_char_params_t werden die Parameter für diese *Characteristic* festgelegt. Diese *Characteristic* soll zum *Client* schreiben, und ihn über neue Daten informieren. Es werden also die Parameter *read* und *notify* aktiviert. Dies ist die *Characteristic 1*, zum Senden der Messwerte. Alle anderen *Struct Member*, welche hier keinen Wert zugewiesen bekommen, sind per *memset* auf Null gesetzt.

5.3.3 Events vom Softdevice

Ändert sich der Zustand von der Bluetooth Seite her, wird vom *Softdevice* ein Event aufgerufen. Der Inhalt dieses Events wird per *switch case* überprüft. Am in-

Listing 8: Verschiedene Events vom *Softdevice* (ble_lbs.c)

```
57 switch (p_ble_evt->header.evt_id)
58 {
59     case BLE_GAP_EVT_CONNECTED:
60         on_connect(p_lbs, p_ble_evt);
61         break;
62     case BLE_GAP_EVT_DISCONNECTED:
63         on_disconnected(p_lbs, p_ble_evt);
64         uint8_t bye = 255;
65         rm_uart_put(bye);
66         break;
67     case BLE_GATTS_EVT_WRITE:
68          write(p_lbs, p_ble_evt);
69          break;
70     default:
71          break;
72 }
```

teressantesten ist der BLE_GATTS_EVT_WRITE Event. Aufgerufen wird dieser Event durch das Senden von Daten auf der Webseite.
Bei diesem Event wird die Funktion write() aufgerufen, und dabei p_lbs und p_ble_evt übergeben. In der write() Funktion werden daraufhin die Daten aus der *Characteristic 2* in die char_1 *Struct Member* gespeichert. Die *High Voltage* (HV)

Listing 9: Speicherung der Daten aus Event (ble_lbs.c)

```
25 char_x char_1;
26 ble_gatts_evt_write_t const * p_evt_write =     \
27                                         &p_ble_evt->     \
28                                         evt.gatts_evt.params.\
                                               write;
29
30 char_1.HV      = p_evt_write->data[0];
31 char_1.ULD     = p_evt_write->data[1];
```

zum Beispiel ist die Hochspannung, welche auf der Webseite eingegeben werden kann.

Auf das BLE_GAP_EVT_DISCONNECTED Event wird im Unterkapitel Implementierte Sicherheitsvorkehrungen 5.5 eingegangen.

5.4 Zugriff auf GATT von Client

Nach dem Verbindungsaufbau wird sequenziell auf eine Ebene nach der Anderen zugegriffen. Erst wird das GATT verbunden, anschließend wird, per dessen *UUID*, auf den *Service* zugegriffen, und zuletzt auf die *Characterisitic 1*.

In der *Characterisitic 1* wird auf *Notifies* reagiert. Sobald sich die Daten vom *Server* ändern, werden die Werte aktualisiert.

Um Daten senden zu können, wird auf Knopfdruck auf der Webseite *"Send Parameters"* die Funktion submit() aufgerufen. Daraufhin wird auf die *Characteristic 2* zugegriffen, und die Daten hineingeschrieben.

5.5 Implementierte Sicherheitsvorkehrungen

In Anbetracht davon, wo die BLE-Verbindung in der Praxis stattfinden wird und welche Daten transferiert werden, werden keine speziellen Sicherheitsvorkehrungen implementiert. In den Laboren der EZAG und den Labortrakten der Krankenhäusern sind nur Mitarbeiter zugangsberechtigt, daher ist nicht mit Angriffen auf die Geräte zu rechnen.
Die Verbindung wird deswegen nicht verschlüsselt, und das DK agiert auf dem Level 1 des Sicherheitsmodus 1.
Eine Sicherheitsvorkehrung wurde jedoch schon auf sehr einfache Art umgesetzt. Wird der TLC-Scanner per Bluetooth bedient, besteht das Risiko, dass die Verbindung abbricht, und der Benutzer dies nicht realisiert.
Dafür wird (wie in Listing 8, Zeile 62) dargestellt, das Event benutzt, welches aufgerufen wird, wenn die Verbindung nicht mehr besteht. Bei diesem Event wird die rm_uart_put-Funktion aufgerufen, und eine spezifische Zahl gesendet, die im Normalbetrieb nicht vorkommt. Auf der Simulation, oder später dem TLC-Scanner, kann diese Zahl ausgewertet werden. Als Auswertung kommt zum Beispiel eine Meldung auf dem Bildschirm in Frage, um die Labormitarbeiter davon in Kenntnis zu setzen.

6 Datenübertragung zwischen der TLC-Simulation und der Webseite

In diesem Kapitel wird der Datenfluss zwischen der TLC-Simulation und der Webseite beschrieben. Dafür wird darauf eingegangen, wie die Daten eines TLC-Scanner simuliert werden, und die Art der Übertragung zum nRF52840. Außerdem wird ausgeführt, wie die Daten über BLE transferiert werden, und wie die Daten auf der Webseite empfangen und ausgewertet werden.
Mit der Datenübertragung von der Simulation bis zur Webseite wird der ganze Umfang dieser Arbeit dargelegt. Dafür wird ein MacBook und ein Acer Computer verwendet.

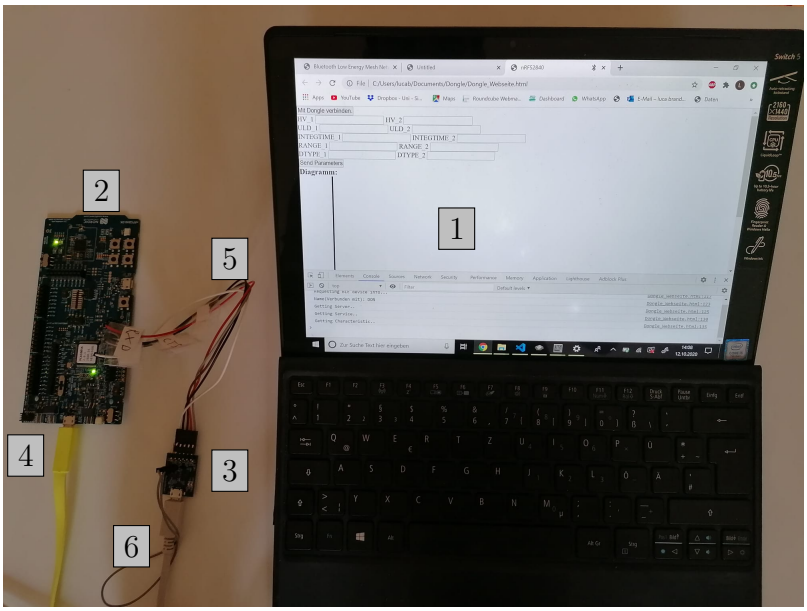

Abbildung 10: Aufbau mit allen Komponenten

- 1: Webseite auf Acer-PC
- 2: nRF52840 DK
- 3: PmodUSBUART
- 4: Spannungsversorgung Board
- 5: UART zu nRF52840
- 6: USB zu MacBook

6 DATENÜBERTRAGUNG ZWISCHEN DER TLC-SIMULATION UND DER WEBSEITE

In Abbildung 10 ist der Aufbau mit allen beteiligten Komponenten dargestellt. Angefangen mit dem DK, welches von dem MacBook mit Spannung versorgt wird. Gleichzeitig ist das BLE-Modul über UART mit dem MacBook verbunden. Dazwischen befindet sich, als UART zu USB-Adapter, das PmodUSBUART.
Auf dem anderen Computer ist die Webseite zu sehen. Die Daten werden über Bluetooth Low Energy zwischen dem MacBook und dem Acer-Computer transferiert.

6.1 Daten aus der TLC-Simulation

Aus dem radio TLC-Scanner werden Daten erwartet, wie in Abbildung 2 dargestellt. Die Daten entsprechen bei einer grafischen Darstellung zwei *peaks*. Bei der Simulation ist es nicht das Ziel diese *Peaks* möglichst genau nachzubilden, sondern das Prinzip zu verdeutlichen.
Dafür werden mit *Python Shell* die Werte einer Sinushalbwelle generiert, gefolgt von einer invertierten Sinushalbwelle. Zwischen den Sinushalbwellen werden Nullen gesendet, um sich auf diese Weise dem originalen Verlauf anzunähern.

Listing 10: Simulation des TLC-Scanners (Python Shell)

```
import serial
import math
import json
s = serial.Serial()
s.port='/dev/tty.usbserial-A904CYI2'
s.baudrate=115200
s.timeout=5
s.open()

HV=1
while(HV != 0):
    s.write("[".encode("utf-8"))
    i=0
    while(i<=math.pi*2):
      if(i<math.pi):
            x1=round(math.sin(i)*50,2)
            a1=json.dumps(x1)
            s.write(a1.encode("utf-8"))
            s.write(", ".encode("utf-8"))
      elif(3 < i < 3.3):
           while(k<math.pi):
                a3 = json.dumps(0)
                s.write(a3.encode("utf-8"))
                s.write(", ".encode("utf-8"))
                k += 0.1
       else:
            x2=math.sin(i+(math.pi*2))*HV
            a2=json.dumps(round(x2,2))
            s.write(a2.encode("utf-8"))
            if((i+0.1)<(math.pi*2)):
                s.write(", ".encode("utf-8"))
        i+=0.1
    s.write("]".encode("utf-8"))
    x = json.loads(s.readline())
    HV=float(x["par1"])
```

Dafür wird ein serieller Port mit einer Baudrate von $115\,200\,\frac{1}{\mathrm{s}}$ und einem *timeout* von 5 Sekunden geöffnet.

Die Werte werden einzeln, verpackt in JavaScript Object Notation (JSON), gesen-

det. Zu Beginn und zu Ende jeder simulierten Messreihe wird eine eckige Klammer angefügt, um die Werte auf der Webseite als Vektor interpretieren zu können.
Um das Empfangen der Daten von der Webseite zu simulieren, wird in *Python Shell* mit der Eingabe der HV interagiert. Die Amplitude der zweiten Sinushalbwelle ist auf der Webseite einstellbar. Dadurch ist die prozentuale Flächenverteilung der beiden *Peaks* variabel.
Der Datenaustausch wird durch die Eingabe einer Null für die Hochspannung auf der Webseite terminiert.

6.2 Senden der Daten über BLE

Sobald das BLE-Modul vom UART Daten erhält, wird eine Funktion aufgerufen, um die Daten an die Webseite weiterzuleiten.
Der Datenverkehr von dem nRF52840 zu der Webseite wird mit *Notifies* realisiert. Die Webseite wird benachrichtigt, wenn neue Daten ankommen. Dafür werden

Listing 11: Sendung der Daten über BLE (ble_lbs.c)

```
208 ble_gatts_hvx_params_t params;
209 uint16_t len = sizeof(uint8_t);
210
211 memset(&params, 0, sizeof(params));
212 params.type   = BLE_GATT_HVX_NOTIFICATION;
213 params.handle = p_lbs->char_1_handle.value_handle;
214 params.p_data = &data_von_uart;
215 params.p_len  = &len;
216
217 return sd_ble_gatts_hvx(conn_handle, &params);
```

die Parameter festgelegt. Dazu gehören die Größe des verwendeten Datentyps der zu sendenden Daten, die eigentlichen Daten, das *Handle* und es wird die Art der Übertragung zugewiesen. Anschließend wird die Funktion aufgerufen, um die Parameter an das *SoftDevice* zu übergeben.

6.3 Datenverkehr auf der Webseite

Auf der Webseite werden die Werte der Simulation empfangen und dargestellt. Über die Eingabefelder auf der Webseite werden Parameter zur Beeinflussung der Simulation gesendet.

6.3.1 Empfangen der Werte

Auf der Webseite werden nach dem Zugreifen auf die entsprechende *Characteristic* die *Notifications* aktiviert. Wird ein Zeichen empfangen, wird dieses auf der

Listing 12: Empfangen der Werte auf der Webseite (JavaScript)

```
151 window.notifications = await myChar.startNotifications();
152 window.notifications.addEventListener('
        characteristicvaluechanged', (e) => {
153
154     let d = document.getElementById("out");
155     let dec = new TextDecoder();
156     v = dec.decode(e.target.value);
157
158     if (v.startsWith('[')) {
159         vector = [];
160         d.innerHTML = v;
161     }
162     else if (v.endsWith(']')) {
163         d.innerHTML += v;
164         vector = JSON.parse(d.innerHTML);
165         drawSVG(vector);
166     }
167     else {
168         d.innerHTML += v;
169     }
170 });
```

Webseite dargestellt. Dabei wird überprüft, ob es sich um eine eckige Klammer handelt. Handelt es sich bei dem Zeichen um eine eckige Klammer gegen rechts, werden die weiteren Zeichen in die Variable *vector* gespeichert. Dadurch wird der Vektor wieder hergestellt. Folgt die Klammer gegen links wird die Messreihe als

vollständig betrachtet und die Funktion *drawSVG* aufgerufen.

6.3.2 Darstellung der Daten

Die Werte werden mithilfe der Scalable Vector Graphics (SVG) in eine vorbereitete Fläche, mit Abszisse und Ordinate, auf der Webseite dargestellt. Um die Grafik ordentlich zu gestalten, wird der Verlauf auf zwei verschiedene Arten gezeichnet:

- Balkendiagramm
  Um das Balkendiagramm zu erzeugen, wird der *Cursor* der Ordinate entlang inkrementiert und der jeweilige Wert in y-Richtung invertiert abgetragen. (Grüner Verlauf)

- Graf
  Hierfür wird das *Polyline*-Element benutzt. Die Werte werden auf der Fläche eingetragen und verbunden. (Roter Graf)

In Abbildung 11 ist die Webseite in Betrieb dargestellt, mit Diagramm und den empfangenen Werten. Aufgerufen wurde die Webseite für diese Demonstration mit einem Notebook.

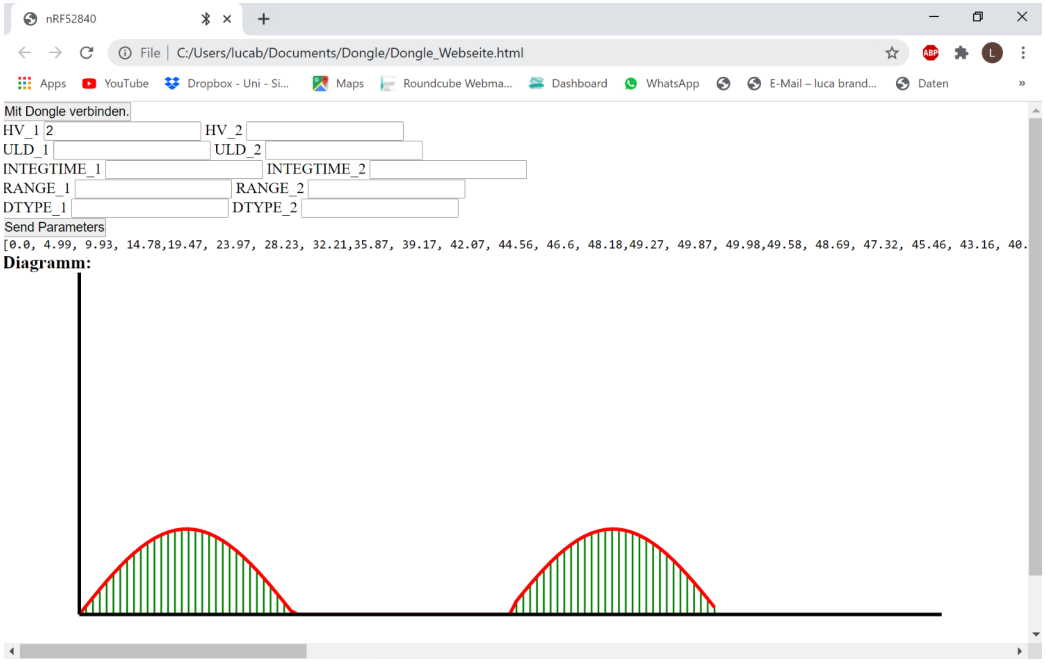

Abbildung 11: Webseite in Betrieb

Die Eingabefelder für die Parameter sind vorbereitet, werden jedoch, abgesehen von der Hochspannung 1, für die Simulation nicht benutzt.

6.3.3 Eingabe der Parameter

Wird ein Wert auf der Webseite eingeben, und anschließend *"Send Parameters"* gedrückt, wird die submit()-Funktion aufgerufen. Darin werden die Elemente mit der *document.getElementById*-Methode in Variablen gespeichert. Daraufhin wird auf die *Characteristic 2*, welche die Daten empfängt, zugegriffen. Die Parameter werden in JSON verpackt und in die *Characteristic* geschrieben.

7 Evaluation der Bluetooth-Verbindung

Der Datenverkehr zwischen dem Qualitätskontrollgerät und der Webseite ist somit realisiert. Um das Programm nutzen zu können, muss noch die Qualität der BLE-Verbindung untersucht werden.
In der Praxis spielen dafür verschiedene Faktoren eine Rolle. Es wird versucht, diese Faktoren möglichst umfassend und praxisnahe nachzubilden. Die folgenden Messungen sind jedoch nicht genau so in der Praxis vorzufinden. Diese Messungen können herangezogen werden, um einen Anhaltspunkt zur Einschätzung der Verbindung zu bekommen.
Anhand der folgenden Untersuchungen kann die EZAG entscheiden, für welche anderen Geräte in der Zukunft eine BLE-Funkverbindung zum Einsatz kommen könnte.
Für die folgenden Versuche wird das nRF52840 DK und ein Switch 5 Laptop verwendet.

7.1 Zuverlässigkeit des Verbindungsaufbaus

Der erste Aspekt, um die Zuverlässigkeit der BLE-Verbindung zu bestimmen, ist der Verbindungsaufbau.
Dafür wird 100 Mal versucht die Webseite mit dem DK zu verbinden, um die Anzahl der erfolgreichen Versuchen mit den Nicht-Erfolgreichen zu vergleichen. Der Abstand zwischen den zu verbindenden Geräten beträgt, inspiriert durch die räumlichen Gegebenheiten der Labore der EZAG, 3 Meter.

Tabelle 1: Wahrscheinlichkeit des Verbindungsaufbaus

Anzahl Verbindungsaufbauten	100
Erfolgreiche Verbindungsaufbauten	99
Nicht-Erfolgreiche Verbindungsaufbauten	1

Wie der Tabelle 1 zu entnehmen ist, funktionierte der Verbindungsbaubau in dieser Versuchsreihe zu 99%. Der eine nicht-erfolgreiche Verbindungsversuch resultierte in einem *Unknown GATT Error*, dessen Grund nicht ermittelt werden konnte. Bei einer Aktualisierung der Webseite und im nächsten Anlauf des Verbindens funktionierte der Verbindungsaufbau wieder fehlerlos.
Um genauere und allgemeingültige Aussagen über die Verbindungsfähigkeit der Geräte zu treffen, wäre jedoch eine größere Datenmengen nötig.
Der nächste Faktor zur Beurteilung des Verbindungsaufbaus ist die benötigte Dauer für das Herstellen einer Verbindung. Wird für das Herstellen einer Verbindung viel Zeit benötigt, mindert dies den Komfort einer Datenübertragung per Funk für den Benutzer erheblich. Aus diesem Grund wird untersucht, wie groß der empirische Mittelwert für den Aufbau einer Verbindung, bei einer Menge von 20 Verbindungsaufbauten, ist. Es wird nur eine gewissen Menge an Verbindungsaufbauten untersucht, daher wird der empirische Mittelwert herangezogen.
Dafür wird der Mittelwert der Verbindungsaufbauten ($\mathrm{VA_{MW}}$), wie in Formel 7.1 dargestellt, per Summe der Verbindungsaufbauten geteilt durch die Menge der benötigten Zeit der Verbindungsaufbauten berechnet. Die Anzahl der nicht erfolgreichen Verbindungsaufbauten wird mitgezählt, jedoch nicht in die Statistik eingerechnet.
Die Zeit wird von dem Moment an gestoppt, wenn auf der Webseite das DK angewählt ist und *Pair* gedrückt wird. Angehalten wird die Zeit, wenn auf der Konsole der Webseite die Nachricht *"Mit Gerät verbunden"* erscheint.

$$VA_{MW} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{20}\sum_{i=1}^{20} x_i = 0.3s \tag{7.1}$$

In dieser Versuchsreihe gab es keine Nicht-Erfolgreiche Verbindungsaufbauten. Jeder der 20 Verbindungsaufbauten wurde ausgewertet.
Der empirische Mittelwert beträgt bei einer Menge von 20 Verbindungsaufbauten 0.3 Sekunden. Die gemessenen Zeitwerte sind so kurz, dass die Genauigkeit der

Messung nicht sehr hoch ist. Es lässt sich jedoch sagen, dass dieser Wert, subjektiv betrachtet, im Bereich des Zumutbaren für die Labormitarbeiter ist, und sollte niemanden davon abhalten, diese Art der Kommunikation aus diesem Grund nicht nutzen zu wollen.

7.2 Sicherheit der Verbindung

Da keine Sicherheitsverfahren implementiert wurden, ist die Verbindung sowohl anfällig für aktives und passives *Eavesdropping*, sowie für *Privacy-* und *Idendity Tracking*.
Auch könnten sich zu diesem Zeitpunkt unbefugte Personen mit dem TLC-Scanner verbinden und mit diesem interagieren. Jedenfalls solange das BLE-Modul noch in keiner Verbindung ist.
In Kapitel 9 Ausblick werden Sicherheitsmaßnahmen empfohlen, welche für diese Anwendung sinnvoll wären, um eine sichere Datenübertragung zu können.

7.3 Entfernung für Datentransfer

Für die Anwendung in der Praxis ist die Entfernung, bei welcher der Datenverkehr funktioniert, relevant. Dafür wird der Abstand der zwei Geräte vergrößert, und währenddessen die Signalstärke in Dezibel aufgezeichnet. Dieser Versuch findet auf freiem Feld statt. Das heißt, zwischen den beiden Geräten befinden sich keine Hindernisse, welche die Funkverbindung stören könnten.

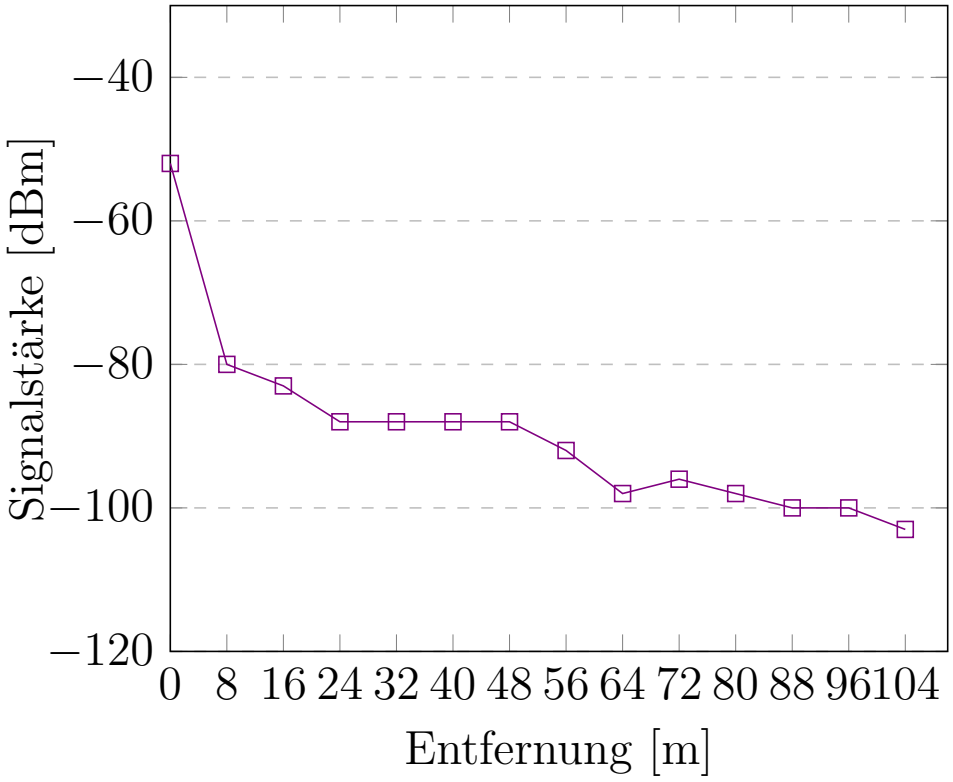

Abbildung 12: Signalstärke in Abhängigkeit der Entfernung ohne Hindernisse

Der Messung ist zu entnehmen, dass bis zu 8 Meter die Signalstärke sehr gut ist. Zwischen 8 Meter und 56 Meter ist die Signalstärke noch akzeptabel. Danach lässt die Signalstärke bis 104 Meter nach. Abgebrochen ist das Signal bei 105 Meter.
Die Genauigkeit der Messung wird durch viele äußere Faktoren beeinflusst. In der Umgebung der Messung gab es andere Bluetooth- und WLAN-Signale und vielleicht sogar Satelliten-Strahlung oder Hochspannungsleitungen. Selbst das Tageslicht, welches elektromagnetische Strahlung unter anderem im 2.4 GHz-Bereich ausstrahlt, kann die Messung beeinflussen.
Diese Messwerte lassen jedoch darauf schließen, dass die Funkverbindung selbst bei relativ großen Entfernungen funktionieren kann.

7.3.1 Ermittlung der mittleren Abbruchentfernung

Außerdem wird mit der *Up-and-Down Methode* der Abstand ermittelt, bei welchem die Verbindung zu einer Wahrscheinlichkeit von 50% abbricht.
Dafür wird nicht die Signalstärke benutzt, sondern es werden Daten übertragen. Werden auf einem der Geräte nicht mehr alle Daten empfangen, wird dies als Verbindungsabbruch gewertet.
Die Differenzstrecke (Δs), wird auf vier Meter festgelegt. Der Versuch wird im Freien, ohne Hindernisse für die Datenübertragung zwischen den Geräten, durchgeführt.

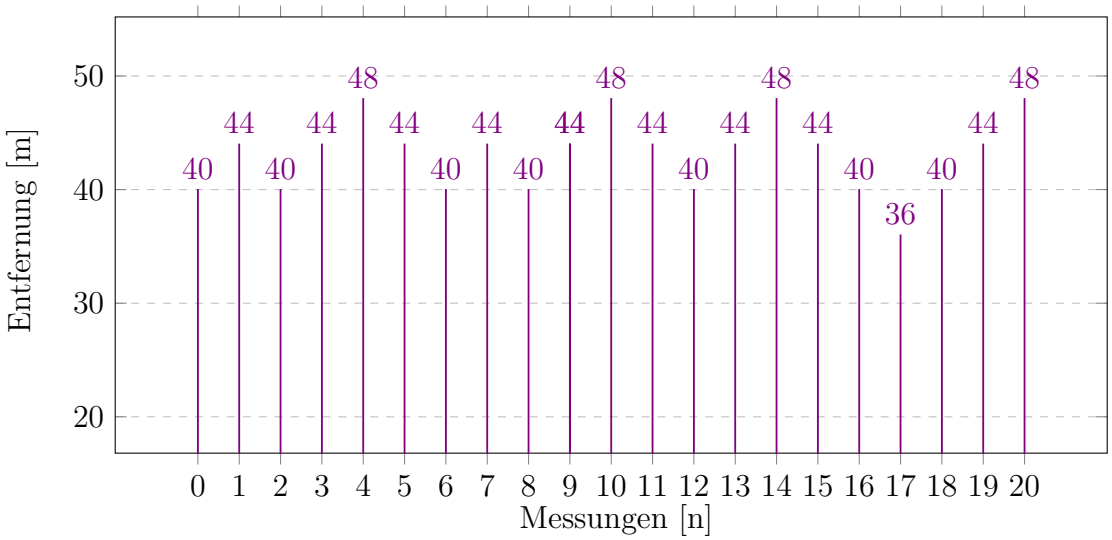

Abbildung 13: Messwerte für die mittlere Abbruchentfernung

Der erste Verbindungsabbruch erfolgt nach der ersten Erhöhung der Entfernung. Die Laufvaribale i wird Eins gesetzt.
Anschließend werden den Schritten die Events und entsprechenden Entfernungen zugeordnet. Da die Events Verbindungsabbruch und Nicht-Verbindungsabbruch gleich oft auftreten, werden k und q frei zugeordnet. k sind die Verbindungsabbrüche und q die Events bei gehaltener Entfernung.

Tabelle 2: Auswertung Up-and-Down Methode

l	k_i	q_i	$Entfernung$ [m]
2	4	0	48
1	5	4	44
0	1	6	40
-	0	1	36
	k=10	q=10	n=k+q=20

Damit werden die Koeffizienten A und B berechnet.

$$A = \sum_{i=0}^{j} i * k_i = 2 * 4 + 1 * 5 = 13 \tag{7.2}$$

$$B = \sum_{i=0}^{j} i^2 * k_i = 2^2 * 4 + 1^2 * 5 = 21 \tag{7.3}$$

Aus den Messwerten und den Koeffizienten kann anschließend die Standartabweichung und die Entfernung berechnet werden, bei welcher die Verbindung zu 50% abbricht. Die Entfernung vor dem ersten Verbindungsabbruch beträgt 40 Meter (s_0).

$$s_{d50} = s_0 + \Delta s \left(\frac{A}{k} \pm \frac{1}{2} \right) = 40m + 4m * \left(\frac{13}{10} - \frac{1}{2} \right) = 43.2m \tag{7.4}$$

$$s_A = 1.62 * \Delta s \left(\frac{kB - A^2}{k^2} + 0.029 \right) = 2.84m \tag{7.5}$$

Die Entfernung, bei der zu 50% nicht mehr alle Daten ankommen, beträgt 43.2 Meter, mit einer Standartabweichung von 2.84 Meter.

Die Genauigkeit unterliegt, wie im vorherigen Versuch, äußeren Einflüssen. Die mittlere Verbindungsentfernung kann unter anderen Umständen unterschiedlich groß sein.

7.4 Datenübertragung mit Hindernissen zwischen Server und Client

In der Praxis findet die Datenübertragung nicht ausschließlich ohne Hindernisse statt. Darum werden mit zwei Versuchen die realen Gegebenheiten simuliert. Anhand der Schlussfolgerungen der jeweiligen Versuche können anschließend Empfehlungen für die Anwendbarkeit gegeben werden.

7.4.1 Hindernis 1 zwischen den Geräten

In diesem Kapitel wird untersucht, wie Zuverlässig die Verbindung ist, wenn sich zwischen den Geräten das Hindernis 1 - eine Wand - befindet, mit einer offenen Türe nebenan. Begibt sich zum Beispiel ein Labormitarbeiter von dem Raum mit dem TLC-Scanner in einen anderen Raum, ist es gut zu Wissen, ob die Verbindung standhalten wird. Bei der Wand für diesen Versuch handelt es sich um eine gewöhnliche Backsteinwand mit einer Dicke von 11.5cm.

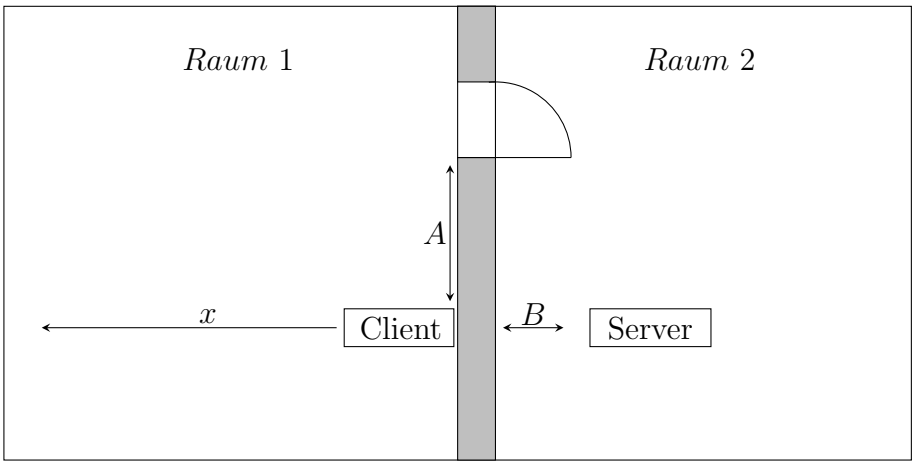

Abbildung 14: Versuchsaufbau mit dem Hindernis 1

Die Entfernung wird dabei in x-Richtung in Ein-Meter-Schritten vergrößert. Dafür wird der Acer Laptop auf horizontaler Ebene verschoben, die Vertikale bleibt konstant. Der Abstand A beträgt dabei drei Meter, Abstand B einen Meter.
Die maximale Distanz von x beträgt 5m, aufgrund der räumlichen Gegebenheiten

von Raum 1. Der Abstand in x-Richtung wurde bis zu den maximalen 5 Metern vergrößert. Dabei wurde die Datenübertragung nicht beeinflusst. Der Datentransfer in dieser Anordnung funktionierte einwandfrei.

7.4.2 Hindernis 2 zwischen den Geräten

Bei dem Hindernis 2 handelt es sich um eine Glasscheibe, welche in den Laboren der EZAG eingebaut sind. Um die Verbindung mit einer Glasscheibe zwischen den Geräten zu untersuchen, werden die Geräte so angeordnet, dass der einzige Weg durch die Glasscheibe führt. In diesem Versuch gibt es keine offene Türe. Das nRF52840 befindet sich einen Meter von der Scheibe entfernt. Der Abstand des *Centrals* wird für diesen Versuch vergrößert.
Die maximale Entfernung zwischen dem Hindernis 2 und dem *Central* beträgt im Labor 5 Meter.
Die Datenübertragung funktionierte wieder bis zu der maximalen Entfernung zwischen Glasscheibe und *Central*.

8 Ergebnis

Im Rahmen dieser Bachelorarbeit wurde ein funktionierender Prototyp zur Datenübertragung zwischen zwei Geräten geschaffen. Es wurde unter anderem gezeigt, dass der TLC-Scanner auch über eine Webseite bedienbar ist.
Mit dem nRF52840 wurde ein BLE-Modul ausgewählt, welches für diese Anwendung den Anforderungen entspricht. Die UART-Verbindung, sowie die *Client*- und *Server*-Seite wurden programmiert, und können von der EZAG für weitere Projekte benutzt werden.
Außerdem wurde die Qualität der Verbindung zwischen den Geräten auf verschiedene Arten untersucht. Die Evaluation der Verbindung ist sehr positiv ausgefallen. Durch die große Reichweite und Stabilität der Datenübertragung wird eine Datenübertragung per Bluetooth Low Energy für weitere Geräte der EZAG in Betracht gezogen.
Die Grundlagen, um ein Netzwerk mit Geräten der EZAG und den Bedienapparaten zu erstellen, wurden im Kapitel *Bluetooth Mesh* erarbeitet.
Die Sicherheitsverfahren von BLE sind erörtert worden, und können für weitere Projekte herangezogen werden.

9 Ausblick

Da der zeitliche Rahmen der Arbeit begrenzt ist, wurden die nicht-essenziellen Aspekte teilweise nicht zu Ende erarbeitet.
Um den Datentransfer per BLE in verkaufsfertigen Produkten einsetzten zu können, müssen noch einige Aufgaben weitergeführt werden.
Zu Beginn kann das Interface der Webseite verschönert werden. Dies trägt nicht zur Funktion bei, erhöht aber die Bedienfreundlichkeit. Um die Webseite aufrufen zu können, ohne diese lokal speichern zu müssen, muss diese auch noch auf einem Github Account abgespeichert werden.
Bei dieser Arbeit liegt der Schwerpunkt auf der BLE-Verbindung. Die UART-Schnittstelle wurde einfach gehalten, und kann noch verbessert werden, indem die Funktion um Daten zu empfangen per *Handler* aufgerufen wird.
Was den BLE-Teil betrifft, können die Sicherheitsverfahren ausgebaut werden. Um die *Keys* für die Sicherheitsinformationen auszutauschen, kommt beispielsweise die *Out of Band*-Methode per NFC in Frage. Wird, um eine Verbindung herstellen zu können, eine kurze Distanz zwischen den Geräten benötigt, ist es auch ausgeschlossen, dass unbefugte Personen außerhalb des Labors eine Verbindung mit dem BLE-Modul herstellen.

Literatur

[1] Townsend, Kevin et al: *Getting Started with Bluetooth Low Energy: Tools and Techniques for low-power Networking.* O'Reilly: 2014

[2] Afaneh, Mohammad: *Intro To Bluetooth Low Energy: The fastest way to learn BLE.* Indiana: NovelBits, 2018

[3] Bluetooth Special Interest Group: *Bluetooth Core Specification Version v4.2.* 2014 [Online]

[4] Sherma, Joseph und Fried, Bernard: *Handbook of Thin-Layer Chromatography.* New York: Marcel Dekker, 2003

[5] Gupta, Naresh: *Inside Bluetooth Low Energy* 2nd ed. London: Artech House, 2016

[6] Campell, Scott: *Basics of UART Communication.* https://www.circuitbasics.com/basics-uart-communication/ Stand: 30.09.20

[7] Bluetooth Special Interest Group (Woolley, Martin und Schmidt, Sarah): *Bluetooth Mesh Networking: An Introduction for Developers* Bluetooth SIG, 2017: https://www.bluetooth.com/wp-content/uploads/2019/03/Mesh-Technology-Overview.pdf Stand:10.10.20