



Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Bachelorarbeit zum Thema:

Entwicklung eines drahtlosen Fußschalters zur Sammlung und Verarbeitung von Messdaten

Zur Erlangung des akademischen Grades Bachelor of Science

Vorgelegt von: Wolfram Barth
Matrikelnummer: 03708119
Studiengang: Informatik
Betreuer: Prof. Dr. Stefan Wallentowitz
Abgabedatum: 28. August 2023

Abstract

Die Digitalisierung von Hand- und Messwerkzeug in der Industrie führt dazu, dass eine steigende Menge an Daten zusammengeführt und verarbeitet werden muss. Für die Anbindung des Werkzeugs an einen Computer existieren verschiedene Möglichkeiten, die alle teils gravierende Nachteile besitzen. Zudem erfordern die unterschiedliche Anwendungsfälle für die Weiterverarbeitung und Speicherung die Daten der Messungen in verschiedenen Formaten. Dabei wird die folgende Forschungsfrage gestellt: Wie sehen die Komponenten eines drahtlosen Fußschalters aus, der die gängigen Anwendungsfälle der Anbindung von Messwerkzeug an einen Computer abdeckt? Es wird eine Anwendung entwickelt, sowohl auf einem drahtlosen Fußschalter, als auch einem USB-Dongle, die Daten von Messwerkzeugen zusammenführt und in verschiedenen konfigurierbaren Formaten dem Anwender zur Verfügung stellt. Diese Anwendung stellt einen neuen softwarebasierten Ansatz da, wie industrielles Messwerkzeug an einen Computer angebunden werden kann und die Messdaten verarbeitet werden können. Diese Anwendung wird von der Hoffmann Group, für deren Werkzeug exemplarisch die Entwicklung durchgeführt wurde, als Fußschalter und USB-Dongle in ihr Produktportfolio aufgenommen, während die Forschung die Ergebnisse dazu verwenden kann um weitere Konzepte zur Anbindung von Werkzeug an Computer zu entwickeln.

Inhaltsverzeichnis

Abbildungsverzeichnis	6
Tabellenverzeichnis	7
Quellcodeverzeichnis	8
Abkürzungsverzeichnis	9
1. Einleitung	10
1.1. Problemstellung	10
1.2. Zielsetzung	11
1.3. Struktur dieser Arbeit	11
2. Hintergrund	12
2.1. Bluetooth Low Energy	12
2.2. HCT-Plattform	12
2.3. HCT-Protokoll	14
3. Ausgangszustand	16
3.1. Dongle-App	16
3.2. Erweiterungen für Fußschalter	18
3.3. Überarbeitung der Dongle-App	19
3.3.1. Trennung der Projekte	19
3.3.2. Verbesserung Verbindungsaufbau	20
3.3.3. Optimierung Abfrage Messeinheit	21
4. Methodik	24
4.1. Konfiguration des Fußschalters	24
4.2. Einbindung der Werkzeuge	25
4.3. Fußschalter Hardware	26
5. Implementierung	27
5.1. Messmodi	27
5.1.1. USB-HID	28
5.1.2. BLE-HID	29
5.1.3. BLE-Windows-App	30
5.2. Überarbeitung des MSC	31
5.2.1. Evaluierung der Möglichkeiten zur Detektion	31
5.2.2. Manuelles Einlesen des Änderungszeitpunkts	32
5.2.3. Finale Lösung	33
5.3. Einbindung Messuhren	34
5.3.1. Unterscheidung der Geräte	34
5.3.2. Anpassung der Messdatenverarbeitung	35

5.3.3. Gruppenfunktion	36
5.4. Einbindung Hardware	37
5.4.1. Energie Management	37
5.4.2. Fußtaster Funktionalität	37
5.4.3. Inbetriebnahme LED	38
6. Ergebnisse	39
6.1. Evaluierung	39
6.1.1. Gruppenfunktion	39
6.1.2. Zeitmessungen HID	41
6.2. Diskussion	44
6.3. Stand Produktentwicklung	45
7. Fazit	46
7.1. Ausblick	46
A. Erfindungsmeldung	48
B. Hardwarezeichnung	54
C. HCT-Protocol-Description	55
Literatur	92

Abbildungsverzeichnis

1.	Schematische Zeichnung der Hoffmann connected tools (HCT)-Plattform . .	14
2.	Auszug aus dem virtuellen Speichermode für Messuhren und Messschieber	15
3.	Aufbau des Projekts im Ausgangszustand	18
4.	Zustandsdiagramm des Verbindungsaufbaus	21
5.	Messablauf	23
6.	Neue Abstraktionsschicht	28
7.	Durchführung von Messungen mit der Gruppenfunktion über CDC	40
8.	Durchführung von Messungen mit der Gruppenfunktion über HID	41
9.	Logging einer Ausgabe eines Messergebnis über USB-HID	42
10.	Mitschnitt der BLE Nachrichten eines BLE-HID Messergebnis von einer verbundenen Messuhr	43
11.	Mitschnitt der BLE Nachrichten der BLE-HID Messergebnisse von drei verbundenen Messuhr	44
12.	Diagramm der Connection Intervalle bei drei Messuhren und einer Verbindung zum Computer	45

Tabellenverzeichnis

1.	Adressen Messergebnis und Messeinheit	34
2.	LED-Zustände	38

Quellcodeverzeichnis

Abkürzungsverzeichnis

HCT Hoffmann connected tools

BLE Bluetooth low energy

HID Human interface device

MSC Mass storage class

CSV Comma seperated values

CAQ Computer-aided quality assurance

FIFO First in first out

FAT File allocation Table

USB Universal Serial Bus

UART Universal Asynchronous Receiver Transmitter

LED light-emitting diode

IDE Integrated Development Environment

ASCII American Standard Code for Information Interchange

CDC Communication device class

HTML Hypertext Markup Language

GATT Generic Attribute profile

ADC Analog-to-digital converter

1. Einleitung

In diesem Kapitel wird zunächst vorgestellt, wie genau die Problemstellung aussieht und welche Zielsetzung sich für den Fußschalter daraus ergibt.

1.1. Problemstellung

Die Digitalisierung von Hand- und Messwerkzeug in der Industrie ist in vollem Gange. Der Hauptgrund dafür ist, neben der einfacheren und genaueren Bedienung, die Möglichkeit durchgeführte Arbeitsschritte auf einem Computer automatisiert zu protokollieren. Wurden früher Messergebnisse per Hand vom Werkzeug auf Papier übertragen, werden sie nun zuverlässig und fehlerfrei über Bluetooth an einen Computer übertragen. Das steigert die Effizienz und ermöglicht die Automatisierung der Qualitätskontrolle, sowie den Nachweis, dass Standards in der Fertigung eingehalten wurden, was in Branchen wie der Automobilindustrie oder der Luftfahrt von großer Bedeutung ist.

Diese Digitalisierung führt dazu, dass eine wachsende Anzahl von Geräten im Arbeitsumfeld der industriellen Fertigung ihre Messdaten zur Verarbeitung durch einen Computer zur Verfügung stellen. Dabei treten bei der Zusammenführung dieser Daten eine Reihe an praktischen Problemen auf. So muss zum Senden eines Messwerts an den Computer bei dem Messwerkzeug eine Taste gedrückt werden. Bei der Durchführung von hochpräzisen Messungen, die auf den hundertstel Millimeter genau sein müssen, verfälscht dieses Betätigen einer Taste auf dem Gerät jedoch bereits die Messung. Auch bei der Durchführung von möglichst zeitgleichen Messungen mit mehreren Messgeräten stellt das Drücken einer Taste zum Senden des Messwerts den Nutzer vor Probleme. Daher werden in der Industrie Fußschalter eingesetzt, die kabelgebunden sowohl an das Werkzeug als auch an den Computer, dieses Senden der Messung auslösen können. Durch die kabelgebundene Natur dieser Fußschalter ist jedoch deren Einsatzbereich reduziert, da es nicht gegeben ist, dass in den Fertigungshallen und Werkstätten, in denen die Fußschalter eingesetzt werden, sich ein Computer in nächster Nähe befindet.

Seit ihrer Einführung trifft die HCT-Windows-App, die einen alternativen softwarebasierten Lösungsansatz für dieses Problem darstellt, immer wieder auf großen Widerstand von IT-Abteilungen, da die App direkt in der Fertigung eingesetzt werden muss, wo meist höchste Sicherheitsrichtlinien gelten. Zur Integrierung der Windows-App in den Fertigungsprozess müssen daher langwierige interne Prozesse angestoßen werden, die die Einführung unattraktiv machen. Zudem stellt die Windows-App keine Funktionalität zur Verfügung um die Werkzeuge über Human interface device (HID) anzubinden, was jedoch eine beliebte und oft geforderte Weise der Anbindung ist.

Der Fußschalter stellt dabei einen Lösungsansatz dar, wie das Sammeln der Messdaten und Orchestrierung von mehreren in Gebrauch befindlichen Werkzeugen erleichtert und ermöglicht werden kann.

1.2. Zielsetzung

Der Fußschalter soll ohne einer Installation zusätzlicher Software die Messungen von Werkzeug erfassen und in verschiedenen Formaten zur Weiterverarbeitung durch einen Computer zur Verfügung stellen. Durch die Entwicklung eines Fußschalters der sich über Bluetooth mit dem Messwerkzeug verbindet, soll der Einsatzbereich gegenüber kabelgebundenen Fußschaltern deutlich vergrößert und der Einsatz dem Anwender erleichtert werden. Durch die Taste des Fußschalters können die Messgeräten zeitgenau und ohnen den Messwert zu verfälschen angesteuert werden. Es soll dabei eine Funktionalität geschaffen werden, die es ermöglicht Messwerkzeuge zu einer Gruppe zusammenzufassen und ihre Messergebnisse durch einen einzigen Tastendruck abzufragen. Dabei muss insbesondere darauf geachtet werden, dass die Messdaten bei ihrer Ausgabe wieder den Werkzeugen zuzuordnen sind.

Aufgrund der Neuheit und Einzigartigkeit des Fußschalters müssen neue Konzepte zur Orchestrierung und Datenverarbeitung der mehreren verbundenen Werkzeuge geschaffen werden. So muss der Verbindungsaufbau zu den Werkzeugen durchgeführt und weitere Informationen vom Werkzeug abgefragt werden. Damit verbunden steht die Herausforderung, dass der Fußschalter die unterschiedlich aufgebauten Daten der verschiedenen Werkzeuge verarbeiten und für zukünftig entwickelte Werkzeuge erweiterbar bleiben muss. Es wird sich auf das HCT-Protokoll und HCT-Werkzeug der Firma Hoffmann beschränkt, während Produkte anderer Firmen die nicht kompatibel mit dem HCT-Protokoll sind von dem Lösungsansatz nicht umfasst werden.

Für diese Ziele werden im Laufe dieser Arbeit Lösungen erarbeitet, welche in Hinblick auf die verfügbare Hardware des Dongles und des Fußschalters umgesetzt werden. Dadurch wird für die Industrie ein Gerät geschaffen, welches die Datenverarbeitung von Messwerkzeugen signifikant erleichtert und bei einer Anbindung über HID diese erst ermöglicht. Die erarbeiteten Konzepte können für die Produkte und den damit verbundenen internen Protokollen von anderen Firmen abgewandelt oder erweitert werden, sowie Konzepte für generische Anbindung geschaffen werden.

1.3. Struktur dieser Arbeit

In Kapitel 2 werden Hintergrundinformationen gegeben, die für das Verständnis dieser Arbeit unerlässlich sind. Anschließend wird in Kapitel 3 der Ausgangszustand der Implementierung behandelt, da zu Beginn dieser Arbeit bereits ein großer Umfang an Code vorhanden ist, welcher abgegrenzt werden soll gegenüber den Erweiterungen, die für diese Arbeit durchgeführt wurden. Diese werden in Kapitel 4 methodisch und in Kapitel 5 praktisch vorgestellt. In Kapitel 6 werden dann die Ergebnisse der Implementierung diskutiert.

2. Hintergrund

Der Fußschalter soll in einer komplexen Umgebung aus Messwerkzeug und Computer agieren, weshalb in diesem Kapitel zum Verständnis wichtige Hintergrundinformationen gegeben werden sollen. Es wird Bluetooth low energy (BLE) vorgestellt, dass die Grundtechnologie für den Informationsaustausch zwischen Fußschalter und dem Messwerkzeug darstellt, sowie die HCT-Plattform und das HCT-Protokoll auf dem sie aufbaut.

2.1. Bluetooth Low Energy

Bluetooth Low Energy ist eine Funktechnologie, die in Hinsicht auf einen sehr geringen Energieverbrauch entwickelt wurde. Sie arbeitet im 2.4GHz unlicensed ISM frequency band. („Bluetooth Wireless Technology“, 2023) BLE findet hauptsächlich Verwendung darin, eine einfache “kabellose Verbindung zwischen einem und Audioendgeräten” herzustellen, sowie die “Anbindung von kabellosen Tastaturen und anderen Eingabegeräten an Notebooks, PCs und Smartphones” zu ermöglichen. (Sauter, 2022, S. 339)

Die Übertragung von gerätespezifischen Daten, wie die derzeit gemessenen Temperatur bei einem digitalen Thermometer, erfolgt dabei über das Generic Attribute profile (GATT). Es definiert Charakteristiken und ihre Attribute. Sie werden während des Verbindungsaufbaus übermittelt und ein Server kann sich auf Charakteristik anmelden, sodass er über eine Änderung der Attribute vom Client selbstständig notifiziert wird, was als Subscription bezeichnet wird. Andernfalls kann er direkt über einen Request die Werte einer Charakteristik anfordern. Mehrere Charakteristiken werden in einem Service zusammengefasst. (Bluetooth Core Specification, 2023, S. 1459) (Carles Gomez, 2012) Der Protokoll Stack von Bluetooth hält sich “lose an das 7 Schichten OSI Modell” (Sauter, 2022, S. 347).

Innerhalb einer Verbindung über Bluetooth werden den Geräten verschiedene Rollen zugeordnet. Das Gerät das den Verbindungsaufbau initiiert hat wird als das “Central” bezeichnet und verarbeitet die Daten des “Peripheral”. Das Peripheral ist dabei meist ein Gerät, welches streng limitiert durch seinem ihm zur Verfügung stehenden Ressourcen ist und stellt seinem Daten dem Central zur Verfügung. Wird ein digitales Thermometer mit einem Handy verbunden um die Temperatur abzulesen, ist somit das Thermometer das Peripheral und das Handy das Central. Neben den Rollen Central und Peripheral, gibt es weiterhin die Rollen “Broadcaster” und “Observer”, diese haben jedoch für diese Arbeit keine Bedeutung (Bluetooth Core Specification, 2023, S. 1246).

Bei den Werkzeugen der Hoffmann Group wurde sich für BLE als Verbindungsmedium entschieden, da den Hand- und Messwerkzeugen, aufgrund ihres Formafaktors und nicht kabelgebundenheit, nur begrenzte Batteriespeicherkapazitäten zu Verfügung stehen.

2.2. HCT-Plattform

Um die Digitalisierung der Messergebnisse dem Anwender so einfach wie möglich zu gestalten, setzt die Hoffmann Group mit der HCT-Plattform darauf, die Digitalisierung der durchgeführten Arbeitsschritte als einen festen Bestandteil in ihr Werkzeug zu integrieren. Diese sind zum Stand dieser Arbeit:

- Drehmomentschlüssel
- Messschieber bzw. Messuhren
- Drehmomentprüfgerät
- Bügelmessschrauben

Sie stellen dem Anwender die Daten der Messungen in verschiedener Weise zu Verfügung. Zum Einen können die Geräte als ein HID über BLE mit dem Computer verbunden werden. Sie simulieren dann eine über Bluetooth verbundene Tastatur über die, die Messergebnisse als Tastendrücke serialisiert werden. Das Messergebnis kann dann in einem Texteditor oder Excel aufgefangen werden. Des weiteren erzeugen die Drehmomentschlüssel und das Drehmomentprüfgerät eine Comma seperated values (CSV)-Datei, in der alle durchgeführten Messungen mit einer großen Anzahl an zusätzlichen Daten gespeichert werden. Wird das Gerät über Universal Serial Bus (USB) mit dem Computer verbunden, zeigt es sich als Mass storage class (MSC)-Device und die Datei kann per Drag-and-drop auf den Computer kopiert werden. Eine weitere Möglichkeit die durchgeführten Messungen zu digitalisieren, ist mithilfe der HCT-Windows-App. Diese erfordert zusätzlich zur frei verfügbaren Software einen speziellen Dongle der zum Verbinden der Geräte benötigt wird. Sie werden ebenfalls über BLE verbunden und sprechen über BLE das firmeneigene HCT-Protokoll. Die Windows-App bietet zahlreiche Möglichkeiten die Messdaten zu digitalisieren und den Produktionsprozess zu optimieren. Es können Schraubfälle in der App angelegt werden und mit Bildern hinterlegt werden. Die Seriennummer von Werkstücken kann automatisch mit dem dazugehörigen Messwert verlinkt werden und Computer-aided quality assurance (CAQ)-Software kann über einen virtuelle COM-Port angebunden werden. Die HCT-Windows-App unterstützt derzeit lediglich die Drehmomentschlüssel, jedoch ist die Einbindung der restlichen HCT-Geräte in Entwicklung. Der Fußschalter und der Dongle nehmen dabei eine ähnliche Position, jedoch mit geringerem Funktions- und Userinterfaceumfang, wie die Windows-App ein. Der letzte Baustein der HCT-Plattform ist die HCT-Mobile-App, sie ist ebenfalls frei erhältlich und erleichtert vorallem die Bedienung des Geräts, zum Beispiel bei Arbeitsschritten bei denen das Display des Werkzeugs für den Anwender nicht sichtbar ist.

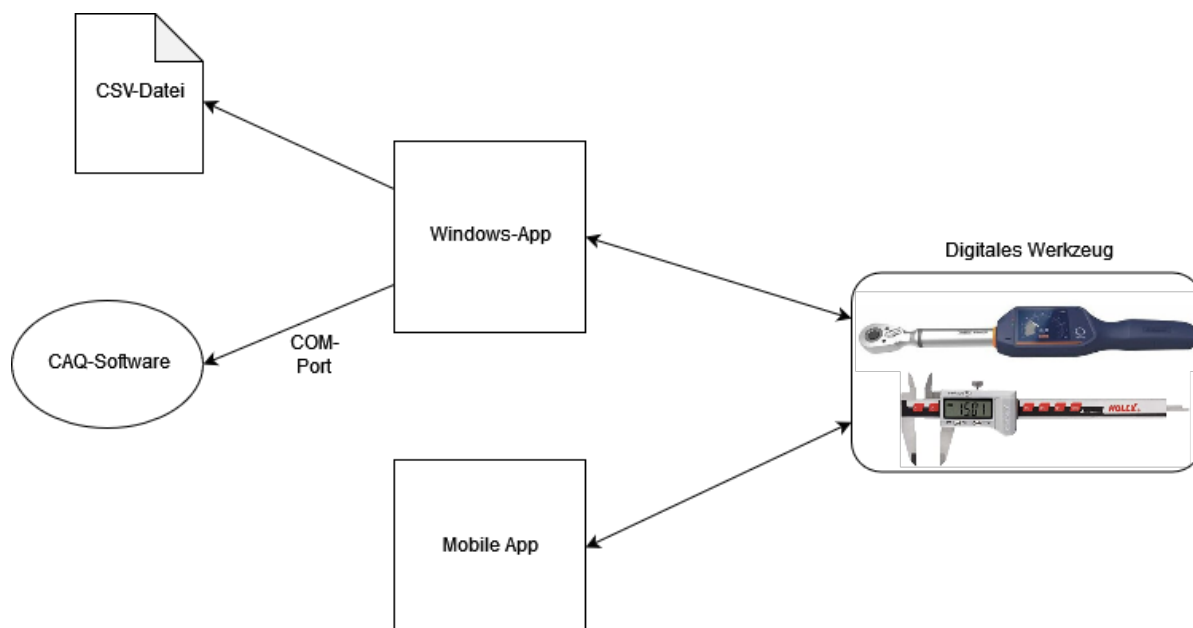


Abbildung 1: Schematische Zeichnung der HCT-Plattform

2.3. HCT-Protokoll

Der HCT-Plattform liegt das firmeneigene HCT-Protokoll zugrunde. Es stellt sicher, dass alle Geräte der HCT-Plattform stets kompatibel zu einander sind. Es ist ein binäres Protokoll, das entwickelt wurde, um über BLE gesprochen zu werden und stellt ein virtuelles Speichermodell der Geräte da. Dabei besitzen Werkzeuge unterschiedlicher Produktreihen und Hersteller jeweils verschiedene Speichermodelle. Über "READ" und "WRITE" Befehle auf die Speicheradressen kann dann der interne Zustand des Werkzeuges abgefragt und verändert werden. Siehe C. So können auch komplexe Operationen effizient über BLE durchgeführt werden. Der virtuelle Speicher ist dabei in Datenblöcken unterteilt, welche die Daten in logische Gruppen zusammenfassen. Die Datenblöcke stellen die Referenzadressen zu spezifischen Dateneinträgen dar. In Abbildung 2 ist ein Auszug aus dem Speichermodell der Messuhren bzw. Messschieber mit den erwähnten Datenblöcken und spezifischen Dateneinträgen zu sehen. Zudem stehen automatisierte Tools zur Verfügung, mit deren Hilfe das Framework, welche die Kommunikation über das HCT-Protokoll abstrahiert, um die Speichermodelle neu eingeführter Werkzeuge erweitert werden kann. Dieses Framework trägt den internen Namen "nrf_Base" und stellt die Anwendungsschicht im Bluetooth-Stack dar.

Protocol version 2.4.12 for Horex_DCDG

Name	Virtual Address	C Type	Bytes	Access	Value Restriction	Default	Remark
Device information	0x00000000			r			
Device class id	0x00000000	uint16	2		Generic (0) Torque wrench (1) Caliper (2) Dial gauge (3) Torque tester (4)		Classification of different Tools
Device sub type id	0x00000002	uint8	1		Garant basic (1) Horex basic (2) Garant plus (3) Horex plus (4) Hazett (5) Facom (6)	HorexBasic	Classification of different derivatives
Protocol type	0x00000003	uint8	1		0 <= x <= 2	2	Used to identify which type of protocol is used to communicate
Protocol version	0x00000004	uint8[3]	3			2, 4, 12	Major, Minor, Step: see Version, Used to check, what features are supported (necessary for updates); Step increased: bugfix (Software is fully compatible); // Minor increased: an backward compatible improvement; // Major increased: New features that may not be fully compatible
Device version	0x00000100			r			
Hardware version	0x00000100	uint8[3]	3				Major, Minor, Step
Software version	0x00000103	uint8[3]	3				Major, Minor, Step
Ble version	0x00000106	uint8[3]	3				Major, Minor, Step
Device status	0x00000200			r			
Battery level	0x00000200	uint8	1		0 <= x <= 100		Battery loading level %
Battery status	0x00000201	uint8	1		Battery charging (1) Ext power supply (2)		Bit0 = Battery Charging, Bit1 = USB-cable connected(Power)
Factory calibration status	0x00000202	uint16	2		Ok (1) Overload (2) Needs recalibration (4) Tool locked overload (8)		Signals if tool needs recalibration
Measurement counter	0x00000204	uint32	4				
Device access synchronisation	0x00000300			rw			
Ext config active	0x00000300	bool	1			False	Configuration of the tool from extern

Abbildung 2: Auszug aus dem virtuellen Speichermodell für Messuhren und Messschieber

3. Ausgangszustand

Zum Beginn dieser Arbeit wurde bereits in meiner vorangegangenen Tätigkeit bei der Hoffmann Group auf Basis des NRF52840-Dongle prototypisch eine Anwendung implementiert, die auf das Bluetooth Modul der Hoffmann Group aufbaut und zusammen mit diesem die Basis für die Anwendung des Fußschalters darstellt. Das Bluetooth Modul stellt dabei die Anwendungsschicht des BLE-Stack da und abstrahiert somit die BLE spezifischen Aufrufe zum Softdevice. Dieser Proof-of-Concept Prototyp sollte vorallem zeigen, dass es möglich ist eine Anwendung auf dem selben Chip wie das Bluetooth Modul und das Softdevice laufen zu lassen, die in der Lage ist einerseits ein Massenspeichermedium und einen virtuellen COM-Port zu öffnen und andererseits über BLE mehrere HCT-Werkzeuge verbindet. Als solche war sie ein voller Erfolg und schaffte die Bereitschaft des Projektmanagements die Ressourcen für die Implementierung des Fußschalters zu bewilligen.

In diesem Kapitel wird die Funktionalität und der Aufbau der Dongle-App, sowie die notwendigen Änderungen und Verbesserungen die durchgeführt werden sollen, vorgestellt. Damit soll auch die Implementierungen für den Fußschalter von den bestehenden abgegrenzt werden.

3.1. Dongle-App

Die Anwendung nutzt das Bluetooth Modul der Hoffmann Group. Es wird in allen HCT-Werkzeugen eingesetzt, wo es eine Vermittlerfunktion zwischen dem eigentlichen Gerätechip und dem Softdevice übernimmt, dabei kann es sowohl in peripheral und central Rolle agieren. Es führt den Verbindungsprozess zu Computern und HCT-Geräten durch. Als Softdevice wird das S140 von Nordic Semiconductor in der Version 7.2.0 verwendet.

Auf diesem Basisprojekt aufbauend befindet sich die USB-Dongle App. Im Gegensatz zu den HCT-Werkzeugen sitzt die gesamte Anwendung ebenfalls auf dem Chip des Softdevice und muss sich die Chipressourcen mit ihm teilen. Die Funktionalität, die in dem File `usb_dongle_app.c` gekapselt ist, ist dafür zuständig das Konfigurationsfile einzulesen und die zugehörigen Daten während der Laufzeit zu halten. Dabei handelt sich es um die zu verbindenden HCT-Gerät, welche in einer gleichnamigen Struktur mit Name, Seriennummer und einer Kanaluweisung gespeichert werden. Die Kanaluweisung wird für das MUX50 bzw. DMX16 Protokoll benötigt. Die Protokolle MUX50 bzw. DMX16 sind dabei proprietäre, American Standard Code for Information Interchange (ASCII)-basierte Protokolle und stellte jeweils nur eine leichte Variation des Anderen dar („HCT Windows App Nutzeranleitung“, 2022, s. 33). Sie werden benötigt um die Daten an CAQ-Software weiterzugeben, welche die Messdaten speichern und auf ihre Richtigkeit überprüfen kann. Werden die Geräte dann verbunden, muss weiterhin der Verbindungszustand, sowie das Connection-Handle gespeichert werden. Des weiteren sammelt es die Messdaten auf, die vom Central Device von den HCT-Geräten im Interrupt-Kontext empfangen werden und reiht die Daten, die von Interesse sind, in eine First in first out (FIFO) Nachrichtenqueue ein. Diese werden später aus dem Kontext der Main-Loop heraus, in das MUX50 bzw. DMX16 Protokoll umgewandelt und über den virtuellen COM-Port verschickt. Ebenfalls in der Funktionalität des `usb_dongle_app.c` Files und aus dem Kontext des Main-Loop heraus, werden die

Befehle des MUX50 Protokolls, welche über dem virtuellen COM-Port empfangen wurden, verarbeitet.

In dem File `usbd_msc_cdc_composite.c` werden USB spezifischen Aufrufe gekapselt. Hier wird das MSC und der virtuelle COM-Port (Communication device class (CDC)) initialisiert und konfiguriert. Der Code wurde nur mit kleinen Änderungen aus den Beispielen des NRF_SDKs übernommen („nRF5 SDK 17.0.2 USB Examples“, 2020), weswegen nicht weiter auf die Implementierung eingegangen wird.

Da es keine Unterstützung seitens der NRF Bibliothek für den internen Flash als Speichermedium für das MSC gibt („Nordic Devzone Forum Case ID 241111“, 2019), musste der Treiber `block_device_ram.c` („nRF5 SDK v17.0.2 Block device RAM“, 2020) angepasst werden. Er wurde als `block_dev_fStorage.c` in das Projekt gezogen und ruft die Funktionen des Files `fStorage.c` auf, welches die Schreibbefehle im Interrupt-Kontext ebenfalls in eine FIFO Nachrichtenqueue einreicht. Das ist nötig, da für die korrekte Ausführung auf Interrupts des Flash Memory gewartet werden muss, welche nur im Kontext der Main-Loop korrekt empfangen werden. Zudem wird dort Flash Memory spezifische Logik abstrahiert. So muss eine Flash Page erst `erased` werden, bevor sie geschrieben werden kann und die Block Logik des MSC wird auf die Flash Page Logik des Speichers übertragen.

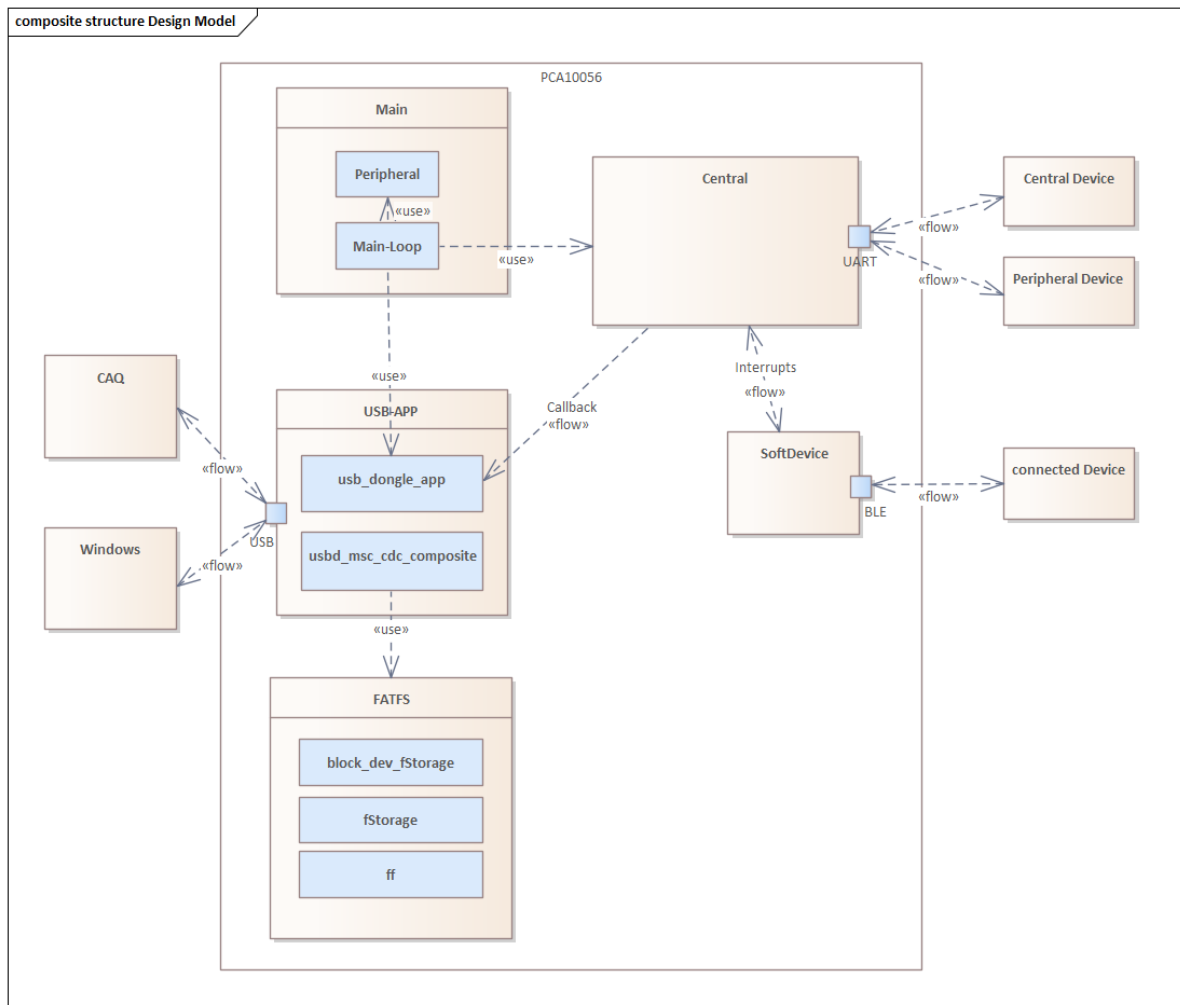


Abbildung 3: Aufbau des Projekts im Ausgangszustand

3.2. Erweiterungen für Fußschalter

Für den Fußschalter muss die bestehende Software wie folgt erweitert werden. Die Peripherie des Fußschalter muss eingebunden werden. Diese umfasst den Schalter der durch den Anwender betätigt werden kann, eine LED-Leuchte und den Akku des Fußschalters. In Hinblick auf den Akku muss eine Energiemanagement geschaffen werden, um dessen Kapazität zu schonen. Während die Dongle-App die Daten ausschließlich über den virtuellen COM-Port zur Verfügung stellt, soll im Fußschalter die Art der Ausgabe konfigurierbar sein. Des Weiteren müssen die Schreibbefehle des MSC optimiert werden, da sie nur sehr langsam und fehlerbehaftet abgearbeitet werden. Die Änderungen an der Konfiguration der Anwendung wird außerdem erst übernommen, wenn der Dongle abgezogen und wieder angesteckt wird, also die Anwendung neugestartet wird. Beim Fußschalter sollen Änderungen an den Konfigurationsfiles detektiert werden und die Anwendung programmatisch neugestartet

werden, auch weil durch den Akku die Anwendung durch den Nutzer nicht direkt neugestartet werden kann. Es müssen außerdem die Messuhren bzw. Messschieber eingebunden werden. Zusätzlich soll die Anwendung weiterhin auch als Dongle erhältlich gemacht werden und die Implementierung muss in Hinblick auf diese Hardwarunterschiede durchgeführt werden.

3.3. Überarbeitung der Dongle-App

Der Prototyp der Dongle-App zeigte, dass die genannten Funktionalitäten technisch möglich sind. Für den Fußschalter sollen sie evaluiert und wenn nötig verbessert werden. Zudem sollen die verschiedenen Projekte, die sich durch die Implementierung des Fußschalters ergeben, getrennt werden.

3.3.1. Trennung der Projekte

Durch die Entwicklung des Fußschalter entstehen drei verschiedene Projekte, deren Code sich jeweils stark überschneidet. Dabei müssen Updates und Änderungen für eines der Projekte mit einem möglichst geringen Aufwand auch in die anderen übernommen werden. Insbesondere die Updates für das Framework der Hoffmann Group zur Abstraktion der BLE-Schnittstelle `nrf_Base` müssen, um die fehlerfreie Kommunikation des Fußschalters zu den HCT-Werkzeugen sicherzustellen, in die Projekte des Dongles und des Fußschalters übernommen werden. Es wird in allen HCT-fähigen Produkten eingesetzt und stetig weiterentwickelt und darf unter keinen Umständen den Code der anderen Projekte nicht beeinhalteten. Auch muss berücksichtigt werden, dass die Projekte jeweils unterschiedliche Peripherie benötigen. Folgende Peripherie darf dabei nur in den jeweiligen Projekten initialisiert werden:

- `nrf_Base`: Universal Asynchronous Receiver Transmitter (UART)
- Dongle: Ein-Farben LED, USB
- HCT-FootSwitch: Fußtaster, Akku Power Management, Drei-Farben LED, USB

Um diese Anforderungen zu erfüllen kann einerseits für alle drei Projekte eine eigene Codebasis geschaffen werden, die sich jeweils stark überschneiden und gesondert gepflegt werden müssen oder die selbe Codebasis für alle Projekte benutzt werden. Die erste Möglichkeit hat den Vorteil, dass sich die Entwicklung einfacher gestaltet, da auf diese Abhängigkeiten keine Rücksicht genommen werden muss. Zudem kann der Code stärker auf den Anwendungsfall optimiert werden, jedoch macht der enorme Arbeitsaufwand die gesonderten Codebasen zu unterhalten und auf dem neuesten Stand zu halten, diese Möglichkeit unpraktikabel. Des weiteren wurde aus softwarearchitektonischen Gründen die Anwendung bereits gekapselt, wodurch die programmatische Abtrennung der Teile keinen außerordentlichen Entwicklungsaufwand erfordert.

Stattdessen muss aus der Main-Routine nur zwei Funktionsaufrufe mit Compilerschaltern abgetrennt werden, um den Code des Fußschalter bzw. Dongles vom `nrf_Base` Projekt zu trennen. Zum einen der Initialisierungsaufwurf und ein App-process Aufruf, da der Main-Loop als Dispatcher für die gesamte Anwendung fungiert. Im Central müssen die Datencallbacks,

sowie im Connection State Callback die Aufrufe an die Dongle-App abgetrennt werden. Die Funktionalität der UART stellte sich als wenig gekapselt heraus und musste an zahlreiche Stellen kleinteilig aus dem Fußschalterprojekt abgetrennt werden.

Das ambitioniertes Ziel war dabei, dass nach der Einführung der Compilerschalter das Binärfile des nrf_Base Projekts identisch zu der vorangegangenen Version ohne Fußschalter sein sollte. Aus noch nicht geklärten Umständen ist das jedoch nicht der Fall.

3.3.2. Verbesserung Verbindungsaufbau

Im Ausgangszustand der Dongle-App werden lediglich zwei Zustände im Verbindungsaufbau abgebildet: “unconnected” und “connected”. Aus der Zuordnung des Connection Handles, also dem Wechsel dieser Variable aus dem Default Wert 0xFFFF auf einen beliebigen anderen Wert, kann zusätzlich darauf geschlossen werden, dass die Service Discovery abgeschlossen wurde und mit der Subscription begonnen werden kann. Das Connection Handle ist dabei eine interne vom Softdevice vergebene Identifikationsnummer. Diese Abbildung des Verbindungsaufbaus ist mehrer Hinsicht unvollständig.

Der Zustand “connected” in der Dongle-App bildet den Zustand fehlerhaft ab. Er wird eingenommen, sobald ein erster Verbindungsaufbau durch die Anwendung angestoßen wurde. Zu diesen Zeitpunkt ist noch keine Kommunikation über die Scan Response hinaus erfolgt und folglich kann noch kein Connection Handle dem zu verbindenden Gerät zugeordnet werden. Im nrf_Base Framework wird dieser Zustand, getriggert durch das korrespondieren Event, erst nach der initialen Kommunikation eingenommen und das Connection Handle ist bereits zugeordnet. Das ist ein entscheidender Unterschied, da ab diesem Zeitpunkt ein Verbindungsabbruch auftreten und nur über das Connection Handle nachvollzogen werden kann. Das kann dazu führen, dass im Werkzeug der Verbindungsaufbau fehlgeschlagen ist, jedoch in der Dongle-App das Gerät im Zustand “connected” ohne Connection Handle festhängt. Nicht nur wird dieses Werkzeug nicht mehr vollständig verbunden, sondern solange es in diesem Zustand bleibt, wird bei keinem anderem Gerät ein Verbindungsaufbau angestoßen.

Wenn die Service Discovery ausgeführt wurde, wird in der Dongle-App mit diesem Event das Connection Handle zugeordnet und die Subscription angestoßen. Mit dieser Zuordnung ist in der Dongle-App das Gerät vollständig verbunden und die derzeitig eingestellte Messeinheit des Werkzeugs wird abgefragt. Dabei wird jedoch nicht darauf gewartet, dass das zu einer erfolgreichen Subscription korrespondieren Acknowledgement erhalten und der Zustand “Subscription” eingenommen wurde. Auch hier können bei einer fehlgeschlagenen Subscription Fehler in der Anwendung auftreten.

Die Zustände des Verbindungsaufbaus sollen jetzt korrekt und vollständig in der Anwendung abgebildet werden. Dabei wurde die Callback Funktionen bisher direkt im Eventhandler des Central aufgerufen. Diese Erweiterung hätte es erfordert, eine respektiven Callback in allen zugehörigen Events aufzurufen, was entgegen der Kapselung der Projekte geht. Jedoch gibt es im Central bereits eine ähnliche Funktionalität, die den Zustand des Central Moduls über UART ausgibt und bereits in den Events des Verbindungsaufbaus aufgerufen wird. Sie wird nun erweitert, sodass falls der Compilerschalter für die USB-App gesetzt ist, der Zustand des Moduls nicht über UART, sondern an einen Callback der USB-App überreicht wird. In der

USB-App wird der Zustand gespeichert und die dazugehörigen Folgeaktionen durchgeführt. Im Central Modul ist weiterhin nur ein einziger Aufruf einer USB-App Funktion vonnöten, um die Zustände des Verbindungsaufbaus in der USB-App abzubilden.

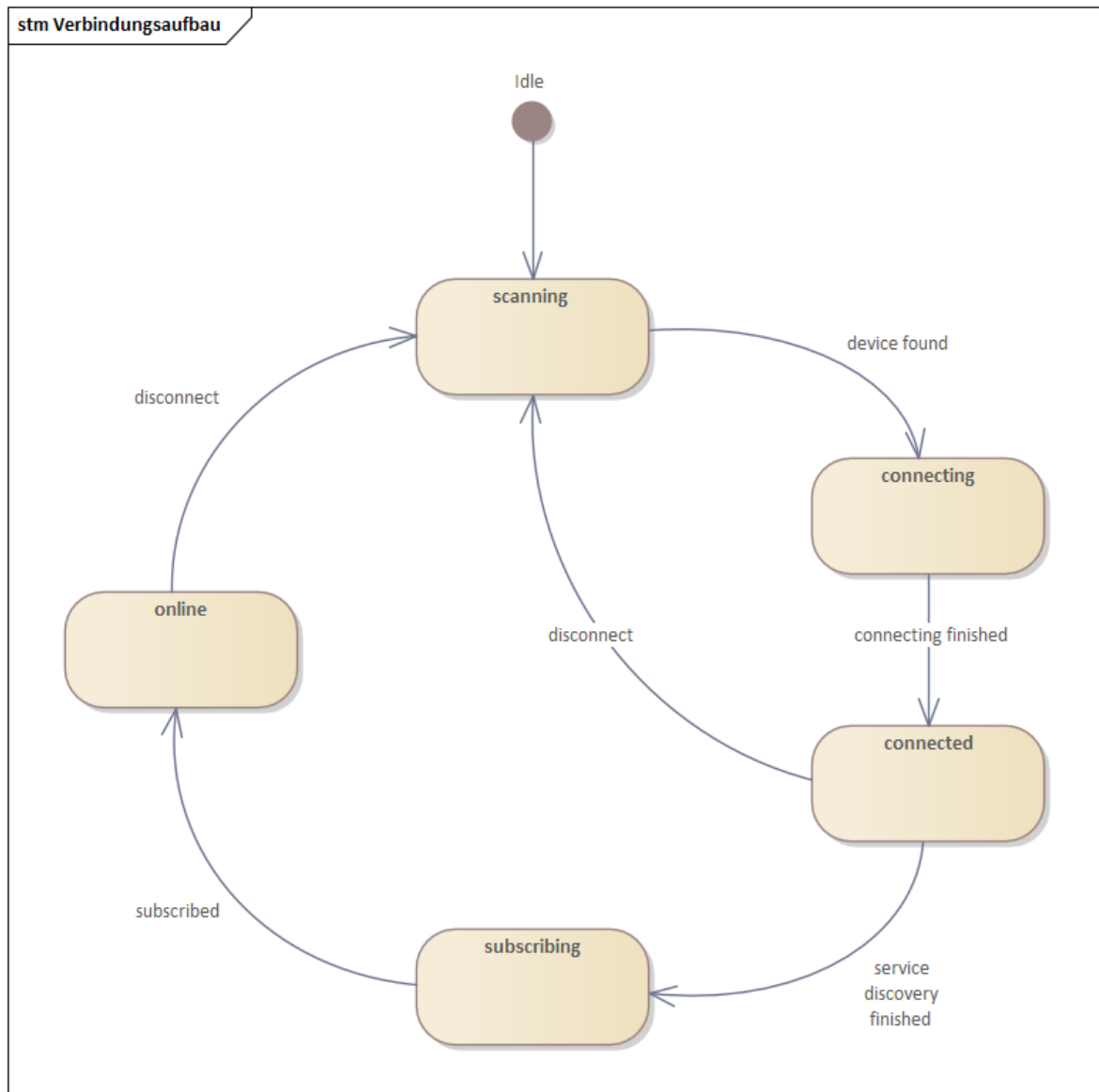


Abbildung 4: Zustandsdiagramm des Verbindungsaufbaus

3.3.3. Optimierung Abfrage Messeinheit

Jede Nachricht die über BLE gesendet wird braucht Rechenleistung und Energie und steigt mit der Anzahl der verbundenen Geräte linear an. Im Ausgangszustand der USB-Dongle App muss die Anwendung bei Werkzeug der Marke Horex jedes Mal, wenn sie ein Messergebnis

erhält, die Einheit der Messung abfragen. Bei Drehmomentschlüsseln der Marke Garant wird ein Datenblock mit der Messeinheit kurz nach Erhalten des Messergebnisses empfangen. Diese Nachricht bezieht sich jedoch auf die derzeitig eingestellte Messeinheit, welche im Fall eines Arbeitsablaufs mit sich ändernden Einheiten, die Einheit für die nächste Messung ist. In diesem Fall gibt die USB-Dongle App die falsche Einheit aus. Bei Geräten der Marke Horex wird nur bei einer Änderung der Messkonfiguration, diese übermittelt.

In beiden Fällen sendet das Werkzeug automatisch bei einer Änderung der Messkonfiguration einen Datenblock mit der Messeinheit an die Subscriber der HCT-Charakteristik. Die Kontrolle der Messeinheit kann daher verbessert werden, indem die derzeitig eingestellte Messeinheit nach dem Verbindungsaufbau einmal abgefragt wird und für jedes verbundene Gerät gespeichert wird. Bei einer Änderung der Messeinheit erhält die Dongle-App die neue Messkonfiguration und aktualisiert die gespeicherte Einheit. Der Messablauf ist schematisch in Abbildung 5 zu sehen. Dadurch wird eine fehlerhafte Einheit in der Ausgabe vermieden, wird damit die Anzahl an Nachrichten der Dongle-App an das Messgerät verringert und die kritischen Ressourcen Rechenleistung und Energie geschont.

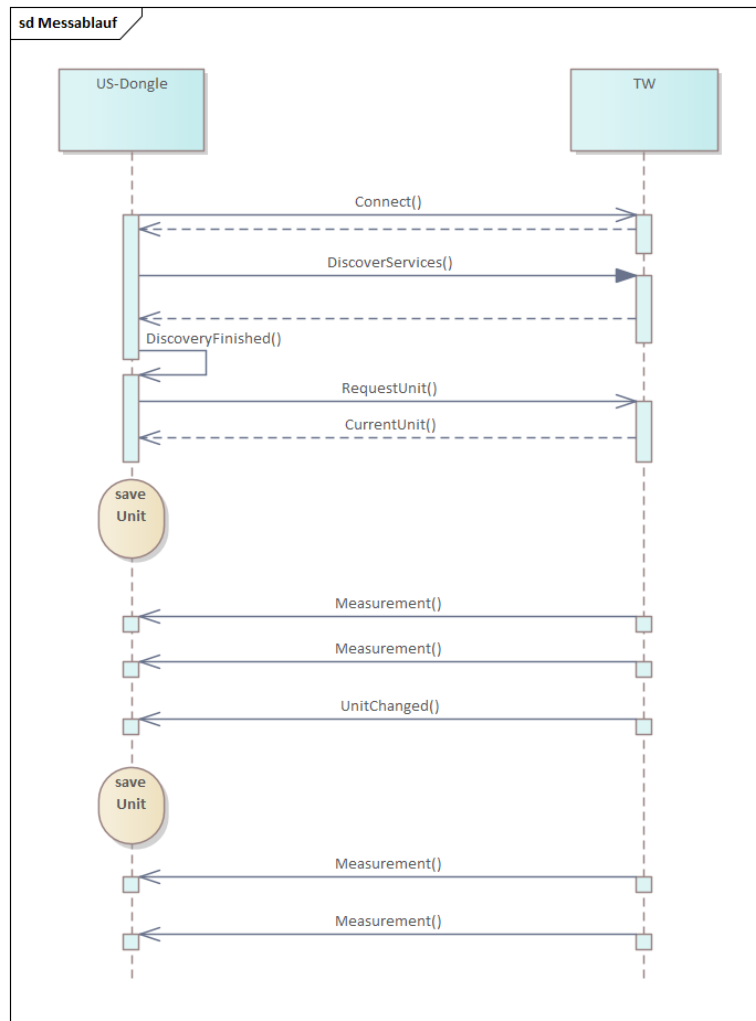


Abbildung 5: Messablauf

4. Methodik

Die Lösung der Problemstellung wird anhand einer exemplarischen Implementierung eines drahtlosen Fußschalters durchgeführt. Dieser soll als Produkt in das Sortiment der Hoffmann Group eingeführt werden, weswegen die Implementierungen, die für diese Arbeit durchgeführt wurden, über eine prototypische Implementierung hinausgehen. Dies zeigt sich insbesondere in der ausgiebigen Analyse der Probleme die während der Umsetzung aufgetreten sind.

4.1. Konfiguration des Fußschalters

Die durch den Fußschalter gesammelten Daten sollen in einem möglichst breiten Spektrum zur Weiterverarbeitung bereitgestellt werden. Auch durch den Fußtaster gegebene Funktionalitäten, wie das Ausgeben eines spezifischen und konfigurierbaren Zeichens bei Betätigung, soll abgedeckt werden. Der Anwender muss die Art wie der Fußschalter eingesetzt werden soll, spezifizieren können und die Datenverarbeitung innerhalb der Anwendung des Fußschalters muss dementsprechend flexible und anpassbar gestaltet werden. Eine weitere Herausforderung dabei ist, das dem Anwender nur äußerst begrenzte Interaktionsmöglichkeiten mit dem Fußschalter zu Verfügung stehen.

Um diesen Anforderungen gerecht zu werden, wurde die Entscheidung getroffen Messmodi einzuführen. Der Messmodus, in dem der Fußschalter arbeiten soll, kann in einer zusätzlichen globalen Konfigurationsdatei angegeben werden und legt fest welche Funktionalität bereitgestellt wird. Es folgt dem bereits bei der Konfiguration der zu verbindenden Geräte verwendeten Schema, dass das Fußschalter durch die Dateien, die im Massenspeichermedium liegen, konfiguriert wird und liefert dem Anwender visuelles Feedback über die derzeitigen Einstellungen. Durch Kommentare in der Konfigurationsdatei können zudem die möglichen Einstellungsoptionen dem Anwender leicht zugänglich gemacht werden.

Eine grundlegend andere Möglichkeit verschiedenste Einstellungen durchzuführen, ist es durch Kombinationen aus langen und kurzen Betätigungen des Fußtasters nicht sichtbare Menüs aufzurufen und mit Hilfe einer Bedienungsanleitung die gewünschten Optionen auszuwählen. Auch wenn diese Möglichkeit in Produkten wie Kaffeemaschinen oder digitalen Weckern eine weite Verbreitung findet, hat sie den entscheidenden Nachteil, dass der Anwender keinerlei grafisches Feedback bekommt. Auch ist es hierfür von Vorteil, wenn das Gerät mehrere Tasten besitzt um die Wahrscheinlichkeit des versehentlichen Ansteuern von Menüs und Umkonfiguration des Fußschalters zu minimieren. Aufgrund des grafischen Feedbacks und der Konsistenz mit der Konfiguration der zu verbindenden Geräte wurde sich für die erste Möglichkeit dazu entschieden. In Kapitel 5.1 wird die Umsetzung der Messmodi beschrieben.

Aus der Konfiguration des Messmodus über eine Konfigurationsdatei ergibt sich die Problemstellung, dass festgestellt werden muss, dass eine Umkonfiguration durchgeführt wurde. Zum Beginn dieser Arbeit muss der Dongle jedes Mal, wenn die Konfigurationsfiles geändert wurden, ab- und wieder eingesteckt werden, damit die Anwendung neugestartet wird und die Dateien erneut eingelesen werden. Beim Fußschalter bekommt jedoch bei Verlust der USB-Verbindung automatisch den benötigt Strom von seiner eingebauten Batterie, weswegen ein

harter Reset durch den Anwender nicht ohne weiteres möglich ist. In Kapitel 5.2 werden daher Möglichkeiten evaluiert, wie die Anwendung des Fußschalters Änderungen an den Konfigurationsdateien detektieren kann, damit anschließend automatisch ein Neustart des Geräts durchgeführt werden kann.

4.2. Einbindung der Werkzeuge

Mit dem Fußschalter soll ein Gerät geschaffen werden, dass die Messdaten von einem möglichst breiten Spektrum an Werkzeug sammeln kann. Dabei bestehen grundlegende Abhängigkeiten zum Medium der Verbindung (BLE) und zum Protokoll (HCT), das über diese Verbindung gesprochen wird. Diese werden aufgrund der Spezifikation des Fußschalters nicht aufgelöst. Jedoch soll für diese Geräte Mechanismen geschaffen werden, um sie möglichst vollständig einzubinden und für in Zukunft entwickelte Geräte erweiterbar zu bleiben. Dabei gilt es außerdem die oft sehr unterschiedlichen Funktionen der Geräte im Fußschalter zu vereinen.

Bei der Einbindung von neuen Geräten in die Anwendung des Fußschalters zeigt sich schnell ein grundlegendes Problem. Während das HCT-Protokoll den Verbindungsaufbau vollständig abstrahiert und damit alle HCT-fähigen Geräte verbunden werden können, bestehen in den Messdaten schwerwiegende Unterschiede. So findet sich das eigentliche Messergebnis oft an einer unterschiedlichen Stelle im übermittelten Datenblock, weswegen für die Einbindung eines neuen Werkzeugs, Mechanismen geschaffen werden müssen, die eine flexible Verarbeitung der Messdaten erlaubt.

Desweiteren besteht zwischen Messuhren und Drehmomentschlüssel ein grundlegender funktioneller Unterschied. Zwar messen beide Geräte kontinuierlich einen Wert, jedoch ist das eigentliche Messergebnis, das ausgegeben und gespeichert wird, ein jeweils anderes. Eine Messuhr oder Messschieber zeigt ausschließlich den derzeitigen Messwert an, während der in einer bestimmten Zeit maximal erreichte Messwert von keinem Interesse ist. Das ist bei einem Drehmomentschlüssel genau anders herum, da er nachdem eine Schraube angezogen wurde, die maximal erreichte Kraft, die während der Verschraubung erreicht wurde, als Messergebnis anzeigen soll und es auch solange anzeigt, bis erneut eine Kraft am Drehmomentschlüssel anliegt. Die Funktionalität des Fußtasters den Anwender zu befähigen ein Messergebnis zeitgenau und präzise auszulösen, hat bei der Art wie das Messergebnis bei Drehmomentschlüssel vorliegt, also keine besondere Bedeutung. Es wurde sich daher entschieden bei Betätigung des Fußtasters nicht das Messergebnis von Drehmomentschlüssel abzufragen, sondern die Funktionalität ganz auf Messuhren zuzuschneiden. Dabei ergibt sich das Problem, dass die ausgelösten Messergebnisse immer in der gleichen Reihenfolge ausgegeben werden müssen, um sie einer Messuhr zuordbar zu machen. Um dieses Problem zu lösen wird eine spezielle Funktionalität entwickelt, die Gruppenfunktion. Die Mechanismen zur flexiblen Verarbeitung der Messdaten, sowie die Entwicklung der Gruppenfunktion werden in Kapitel 5.3 vorgestellt und anhand der Einbindung der Messuhren beziehungsweise Messschieber umgesetzt.

4.3. Fußschalter Hardware

Für den Fußschalter steht eine Reihe an Hardware Funktionen zur Verfügung. Deren Funktionalität muss evaluiert und eingebunden werden um die Funktionsweise der Anwendung sinnvoll zu ergänzen. Der Fußtaster stellt neben der Änderung der Konfigurationsdateien die einzige Möglichkeit für den Anwender da, um mit dem Fußschalter zu interagieren, weshalb sein Funktionsumfang maximiert werden soll. Die Drei-Farben-LED soll den internen Zustand des Fußschalters anzeigen und Aufschluss über die Prozessvorgänge geben.

Wenn der Fußschalter nicht über USB mit einer Stromquelle verbunden ist, bekommt er den benötigten Strom von der fest eingebauten Batterie. Um diese nicht unnötig zu belasten, muss ein Energiemanagement geschaffen werden, dass die bestehenden Ressourcen optimiert. Die Erhöhung der Effizienz der Batterienutzung kann dabei auf zwei Weisen erreicht werden. Einerseits kann bei Nutzung des Fußschalter die Energieeffizienz verbessert werden, was in Kapitel 3.3.3 mit einer Optimierung der Abfrage der Messeinheit umgesetzt wurde. Andererseits kann bei Inaktivität des Fußschalters dessen Funktionalität abgeschaltet werden, um ebenfalls die Laufzeit der Batterie zu verlängern. Dazu muss festgelegt werden was Inaktivität bedeutet und nach welcher Zeit der Inaktivität drastische Stromsparmaßnahmen, wie das vollständige Ausschalten des Fußtasters, erfolgen.

Inaktivität als solche muss klar definiert werden, da sie einerseits programmatisch festgestellt werden soll und andererseits gewisse Aktivitäten, wie eine aktive Verbindung, nicht unbedingt darauf hindeuten, dass der Fußschalter tatsächlich in Gebrauch ist. Anstatt alle Tätigkeiten des Fußschalters zu kategorisieren, wird stattdessen festgelegt welche Ereignisse ein Herunterfahren des Geräts verhindern sollen. Diese sind zum Ende der Arbeit einerseits das Erhalten eines Messergebnis und andererseits der angestoßene Verbindungsaufbau zu einem Werkzeug.

Für die Dauer der Inaktivität nach der der Fußschalter heruntergefahren werden soll kann keine allgemeingültige Aussage getroffen, sondern eine angemessene Zeitdauer hängt von persönlichen Präferenzen und dem Anwendungsfall ab. Daher wurde sich an dieser Stelle dazu entschieden, diese Zeitdauer durch den Anwender konfigurierbar zu machen. Dazu wurde in der Konfigurationsdatei config.ini ein Attribut angelegt, durch das der Anwender Zeit nach der das Gerät heruntergefahren wird in Sekunden angegeben kann.

5. Implementierung

5.1. Messmodi

In diesem Kapitel werden die implementierten Messmodi vorgestellt und die Herausforderungen der Umsetzung beschrieben. Zum Ende dieser Arbeit sind folgende Modi konfigurierbar:

- 0: USB-HID-singleKey
- 1: USB-HID
- 2: CDC (COM-Port)
- 3: BLE-HID
- 4: BLE-HID-singleKey

Sie werden in die Kapitel USB-HID und BLE-HID zusammengefasst und es wird auf den Modus BLE-Windows-App eingegangen, der nicht umgesetzt wurde, jedoch für eine spätere Version geplant ist. Der Modus CDC war zu Beginn dieser Arbeit bereits implementiert.

Zum Zeitpunkt der Implementierung der Dongle-App, war diese die oberste Abstraktionsschicht aller USB-Funktionalität. Das ist mit Einführung der Messmodi nicht mehr der Fall. Daher bedarf es einer neuen Abstraktionsschicht, die über der Dongle- und Fußschalterapp steht und deren verschiedenen Funktionalitäten dirigiert. Dadurch können die Operationsmodi programmatisch getrennt werden, sodass bestimmte Schritte der Initialisierung, wie das Einlesen der Konfigurationsdatei der zu verbindenden Geräte, in einem Modus wie HID einzelnes Zeichen nicht ausgeführt werden. Außerdem bleibt die Anwendung dadurch leicht um neue Messmodi erweiterbar.

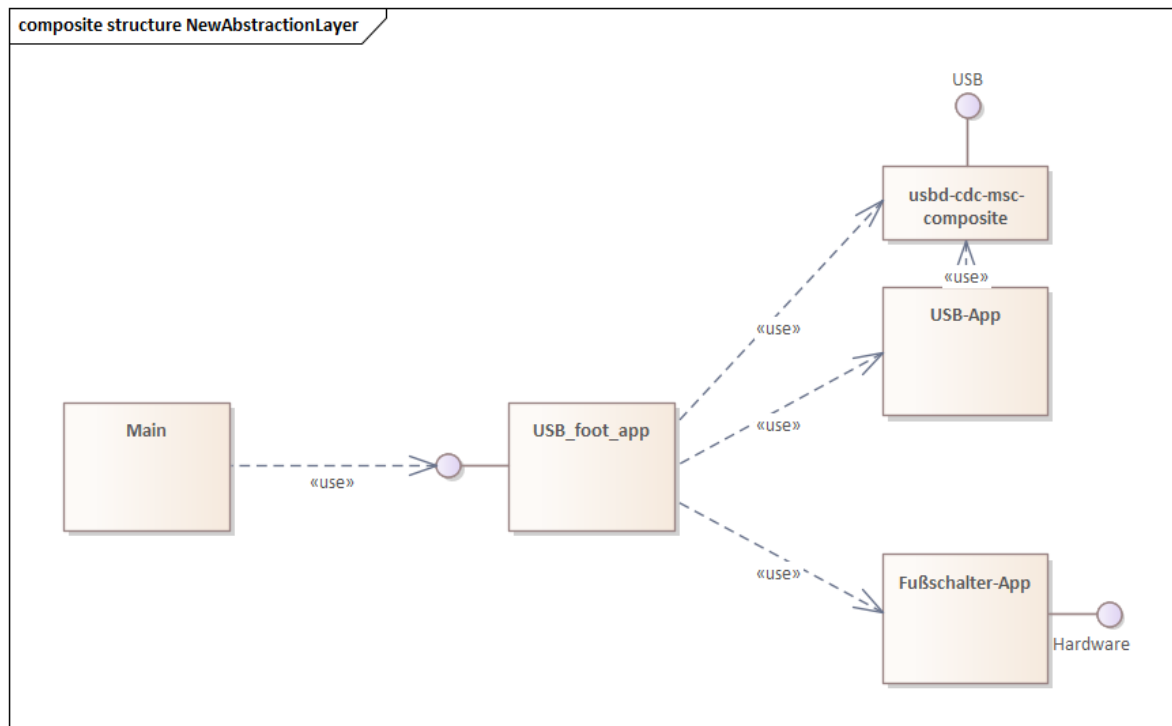


Abbildung 6: Neue Abstraktionsschicht

5.1.1. USB-HID

Ein Feature, das sowohl für den Dongle als auch für den Fußschalter implementiert werden soll, ist das Human Interface Device (HID) über USB. In diesem Modus gibt die Anwendung die Messergebnisse nicht mehr über den virtuellen COM-Port aus, sondern ist über USB als Tastatur mit dem Computer verbunden. Über sie werden die Zeichen der Messung als Tastendrücke, gefolgt von einem konfigurierbaren Terminierungszeichen, ausgegeben. Dadurch können Messungen in einem Tabellen Programm oder einem Texteditor aufgefangen werden.

Für die Implementierung werden die Funktionen des NRF-Library Files `app_usb_hid_kbd.h` verwendet („nRF5 SDK v17.0.2 USB HID keyboard“, 2020). Es abstrahiert die Low-Level USB-Aufrufe und stellt Funktionen zur Verfügung, die bei Aufruf eine Taste drücken oder loslassen. Es fehlt jedoch eine Funktion die ganze Strings serialisiert, weshalb diese implementiert werden muss. Die Scancodes, die Nummern der Tastertasten, sowie der Modifier, eine Taste die Tastendrücke modifiziert, wie die Shift-Taste, werden dabei von einer Funktion erhalten, die im `nrf_Base` Projekt enthalten ist. Sie wurde für die HID Implementierung über BLE geschrieben und gibt den Scancode und zugehörigen Modifier für einen ASCII Character zurück.

Zwischen den Tastendrücken muss eine gewisse Zeit gewartet werden, da der Computer Tastendrücke zwischen denen zu wenig Zeit verstreicht nicht registriert. Dieser Delay wurde

auf 1 Millisekunden festgelegt, was der maximalen Übertragungsrate über USB entspricht („Device Class Definition for Human Interface Devices“, 2001, S. 17) und da bei einem längeren Delay die Anzahl an fehlerhaften Tastendrücken zunahm. Diese Art darauf zu warten, dass die Tastendrücke vom Computer registriert wurde, stellte sich jedoch als sehr fehleranfällig heraus. Nicht nur wurde doppelte Zeichen nicht korrekt erkannt, sondern nach ungefähr zehn Ausgaben schien der USB-Bus überlastet zu sein und es erschienen keine Ausgaben mehr der Messergebnisse. Daher musste die Implementierung geändert werden. Statt die Tastendrücken ohne Rücksicht auf die zugehörigen Events dem USB zu übergeben, muss nach jedem Zeichen auf ein Event der USB-HID Library gewartet werden, das die erfolgreiche Übertragung signalisiert. Dazu muss der zu serialisierende String in einem Buffer hinterlegt und bei Empfangen des Events das nächste Zeichen übertragen werden.

Eine Abwandlung dieses Modus, ist der Modus USB-HID-singleKey. In diesem Modus gibt der Fußschalter bei Betätigung des Tasters nur ein einzelnes konfigurierbares Zeichen aus, beziehungsweise bei einem Doppelklick ein Zweites, wie in Abschnitt 5.2 beschrieben ist.

5.1.2. BLE-HID

Ein weiterer Modus, in dem der Fußschalter arbeiten soll, ist HID über BLE. Dabei simuliert der Fußschalter oder Dongle eine über BLE verbindbare Tastatur und serialisiert nach dem Verbinden wie bei HID über USB die Messergebnisse als Tastendrücken. Dazu muss das Gerät nun zusätzlich zur Central Rolle in der Peripheral Rolle agieren. Dazu muss einerseits das Advertising korrekt konfiguriert werden und in den bestehenden Code des Peripheren Verbindungsaufbaus, die Fußschalter Applikation eingebunden werden. Für das eigentliche Schreiben des Messergebnisses über BLE, gibt es bereits eine bestehenden Funktionen des nrf_Base Projekts die einen String vollständig serialisiert und diese muss nur in der Fußschalter Applikation aufgerufen werden.

Dabei erfordert dieser Modus es nun, dass die Bonding Informationen der Verbindung von Fußschalter zu Computer gelöscht werden kann. Werden sie nicht gelöscht, jedoch auf dem Computer, können sich die beiden Geräte nicht mehr miteinander verbinden. Das muss zum Beispiel geschehen, wenn der Fußschalter mit einem anderen Computer verbunden werden soll, sich der erste Computer jedoch noch in Reichweite befindet. Trotzdem sollen nicht bei jedem Reset des Fußschalter, also bei einer Änderung der Konfigurationsdatei, die Bonding Informationen gelöscht werden, da sonst bei jedem Verbinden, der Fußschalter im Computer erst entkoppelt und dann wieder neugekoppelt werden muss. Es kann also nicht direkt gesagt werden wann die Bonding Informationen gelöscht werden sollen und wann nicht, stattdessen wäre der Idealfall, dass der User selbst die Informationen löschen kann. Das ist jedoch aufgrund der begrenzten Interaktionsmöglichkeiten des Users mit dem Fußschalter nur schwierig umsetzbar. Ein Kompromiss ist, dass die Bonding Informationen gelöscht werden, wenn der Modus BLE-HID verlassen wird. Der Anwender kann dann auch um die Bonding Informationen zu löschen aus dem Modus 3 herauswechseln und dann wieder hineinwechseln. Dabei ist das Problem, dass bei einem Reset nicht direkt ersichtlich ist, in welchen Modus gewechselt wird, da die Konfigurationsdatei erst beim Neustart neugelesen wird. Daher wird direkt vor dem Reset die Konfigurationsdatei gelesen und falls aus dem Modus 3 oder 4 in einen anderen Modus gewechselt wird, die Bonding Informationen gelöscht. Ein alternativer

Lösungsansatz der ohne das ressourcenfordernde Einlesen der Datei auskommt, ist die Bonding Informationen immer zu löschen, wenn sich der Fußschalter beim Reset in einem anderen Modus als 3 oder 4 befindet. Jedoch war dann der “Peer Manager” nicht initialisiert, der zum Löschen der Bonding Informationen zwingend benötigt wird, weshalb doch der erste Lösungsansatz umgesetzt wurde. Der Peer Manager ist dabei eine Library von NRF die sich um Verschlüsselung, Pairing und Bonding kümmert („nRF5 SDK v17.0.2 Peer Manager“, 2020).

Erste Tests zeigen, dass die Geschwindigkeit der Übertragung, einerseits die Dauer bis zum ersten Tastendruck und andererseits die Zeit zwischen den einzelnen Tastendrücken, zu hoch ist. Die Untersuchung wo genau zuviel Zeit verwendet wird, fand mithilfe eines Oszilloskop und dem togglen eines freien Pins, der zu diesen Debugzweck belegt wurde, statt. Es konnte dadurch bereits ein Bug im nrf_Base Projekt gefunden werden. Dabei wird in der Funktion, zwischen den einzelnen Zeichen eine Pause gemacht, um ähnlich im Modus USB-HID dem Betriebssystem Zeit zu geben die Tastendrücke zu verarbeiten. Dieser Delay ist abhängig von Connection Interval, welches wenn nur die Verbindung zum Computer besteht, korrekt gesetzt ist. Jedoch stellte sich heraus, dass diese globale Variable von allen Verbindungen stets überschrieben wird. Dabei ist diese Variable auch unabhängig von der HID Funktionalität nur für die peripheren Verbindungen von Bedeutung und dieses Verhalten ein Fehler. Es wurde daher an der Stelle, an der diese Variable durch einen “Connection Update Request” überschrieben wird, die Unterscheidung eingeführt, ob es sich bei der Verbindung um eine periphere Verbindung handelt.

Auch von diesem Modus, gibt es die Abwandlung BLE-HID-singleKey, die wie im Modus USB-HID-singleKey, ein einzelnes Zeichen bei Betätigung des Tasters ausgibt.

5.1.3. BLE-Windows-App

Der letzte Modus, in dem der Fußschalter agieren soll, ist als ein an die HCT-Windows-App angebundenes Gerät. Dabei soll das Signal der Betätigung des Tasters als eine HCT-Nachricht an die Windows-App gesendet werden, welchen dann ein Messergebnis bei den mit ihr verbundenen Messgeräten triggert. Dazu muss ein HCT-Model für den Fußschalter geschaffen werden. Das Model stellt folgende Werte bereit:

- Device Class
- Protocol type, version
- Version of Hardware, Software, BLE
- Battery level, status
- Reset

Werte des Config.ini Konfigurationsfiles:

- Operating Mode
- CDC protocol

- HID Keyboard Language ID
- HID data set separator
- HID number separator
- HID single key single press
- HID single key double press
- Inactivity timeout
- sequentiell group triggering

Für die Übertragung des eigentlichen Signals, dass der Fußschalter betätigt wurde, muss eine HCT-Charakteristik angelegt werden, auf welche die HCT-Windows-App sich subscriben kann. Über diese Charakteristik wird sie dann über die Betätigung des Tasters notifiziert. Im Advertising muss sich der Fußschalter dann nicht als Tastatur, sondern als HCT-Fußschalter erkenntlich zeigen.

Es wurde jedoch beschlossen, diesen Modus erst in einer späteren Version des Fußschalters zu implementieren, wie im Abschnitt 8.2 genauer erklärt wird.

5.2. Überarbeitung des MSC

In diesem Kapitel wird gezeigt durch welche Schritte die finale Lösung zur Detektion einer Änderung der Konfigurationsdateien erarbeitet wurde.

5.2.1. Evaluierung der Möglichkeiten zur Detektion

Der erste Versuch der unternommen wurde um dem Anwendungsfall gerecht zu werden, ist über die File Information des File allocation Table (FAT) Filesystems den Änderungszeitpunkt auslesen und falls er sich im Vergleich zum Zeitpunkt, der beim erstmaligen Einlesen festgestellt wurde, geändert hat, ein Systemneustart durchzuführen. Dabei hat sich jedoch gezeigt, dass eine Änderung an der Datei keine Veränderung am Änderungszeitpunkt hervorruft. Erst nach einem manuellen Reset zeigt sich das korrekte Änderungsdatum in der File Information. Ein weiterer Versuch war es, direkt zu überprüfen ob neue Daten über das Blockdevice geschrieben wurden. Das Blockdevice ist dabei eine Zwischenschicht zwischen Filesystem und dem physischen Speicher („nRF5 SDK v17.0.2 Block device“, 2020). Das führte jedoch dazu, dass der Systemneustart durchgeführt wurde, bevor alle Daten vollständig geschrieben wurden. Zudem hat diese Implementierung weitere Probleme, wie zum Beispiel, dass ein Formatieren des Datenträgers, wie er bei der ersten Inbetriebnahme durchgeführt werden muss, nicht mehr möglich war.

Ein periodisches Neueinlesen der Daten war hingegen nicht möglich, weil beim Lesen der Daten die Anwendung in einer Warteschleife festhing. Es zeigte sich, dass sowohl MSC als auch das FAT Filesystem auf die gleiche Instanz des Blockdevice versucht haben zuzugreifen, was grundsätzlich nicht möglich ist. Ein Anlegen einer weiteren Instanz des Blockdevice für

das Filesystem behob dieses Problem.

In einer zwischenzeitlichen Lösung des Problems werden die Konfigurationsfiles periodisch neugelesen und über die Daten ein Hashwert gebildet. Anhand dieses Wertes wird dann eine Änderung festgestellt. Hat sich das globale Konfigurationsfile geändert wird ein Systemreset durchgeführt, während bei einer Änderung des Files der zu verbindenden Geräte, die Verbindung zu allen Geräten getrennt wird und das File anschließend neueingelesen, wodurch der Verbindungsaufbauprozess neu gestartet wird. Jedoch zeigte sich, dass Änderungen, die am Ende der Datei stattgefunden haben, nicht detektiert werden. Durch genaue Analyse des Prozess konnte festgestellt werden, dass die Länge der Datei nicht erneut eingelesen wird, wenn die Datei aus dem Massenspeichermedium heraus geändert wurde und nicht aus dem Filesystem auf dem Chip. Daher können Änderungen an den Dateien, die ausschließlich hinten an dem bestehenden Text angefügt werden, nicht detektiert werden, da die Datei mit der alten Länge eingelesen wird.

5.2.2. Manuelles Einlesen des Änderungszeitpunkts

Es konnte festgestellt werden, dass die Information, wie lang die Konfigurationsdatei ist, korrekt im Speicher vorhanden ist, aber nicht ins interne Filesystem übernommen wird. Es besteht also die Möglichkeit die Informationen selbstständig einzulesen. Dazu muss als Erstes das "Directory Entry" gefunden werden. Es steht nach den FAT und beinhaltet Informationen zur Datei, wie die Länge in Bytes und das Änderungsdatum („The FAT filesystem“, 2002, Abschnitt 1.4). Daher müssen folgende Informationen aus der Boot Section ausgelesen werden und folgende Berechnung durchgeführt werden:

- Sa: Startadresse Filesystem
- Sg: Sektorengroße
- nrS: Anzahl reservierter Sektoren
- nF: Anzahl FAT
- nSF: Anzahl Sektoren pro FAT

$$Sa + Sg \cdot nrS + Sg \cdot nF \cdot nSF = Sa + Sg \cdot (nrS + nF \cdot nSF)$$

Diese Informationen stehen jedoch immer an der gleichen Stelle im Bootsektor und ändern sich während des Betriebs des Fußschalters nicht.

Im Directory Entry wird jeweils für die Informationen einer Datei 32 Bytes verwendet und innerhalb dieser 32 Bytes befinden sich die Informationen immer an der gleichen Stelle, weshalb mit festen Offsets gearbeitet werden kann. Jedoch wird ein Eintrag nicht sofort gelöscht, wenn die Datei gelöscht wird, sondern die Filenamen durch Ersetzen des ersten Buchstaben durch 0x5a invalidiert („The FAT filesystem“, 2002, Abschnitt 1.4). Daher muss der valide Eintrag immer wieder neu gesucht werden. Dadurch kann das Directory Entry größer als ein Sektor werden, auch wenn er nur zwei Dateinamen beinhaltet. Es wird

daher nacheinander die Sektoren, die für das Directory Entry allokiert sind, eingelesen und auf die korrekten Dateinamen hin untersucht. Ist der korrekte Eintrag gefunden, wird der Änderungszeitpunkt eingelesen und nach der bereits beschriebenen Methode überprüft. Es wurde hier wieder der Änderungszeitpunkt zur Detektion einer Änderung verwendet, da die Länge der Datei nur dazu benutzt werden könnte, die Datei “händisch” einzulesen, also direkt über Flash Zugriffe und nicht über die FAT Filesystem Library. Was kurzzeitig in Angesicht der weiterhin bestehenden Probleme in Erwägung gezogen wurde, aber aufgrund des hohen Entwicklungsaufwands versucht wurde zu vermeiden. Dabei kann aus dem Directory Entry auch der Startsektor der Datei ausgelesen werden und ähnlich der Adresse des Directory Entry, die Adresse des Sektors berechnet werden. Auf den ersten Blick erscheint das nicht übermäßig kompliziert, jedoch wird die Datei, falls sie größer als ein Sektor wird, auf nicht zwangsläufig aufeinanderfolgende Sektoren verteilt, was nur anhand der FAT nachvollzogen werden kann.

5.2.3. Finale Lösung

Das Hauptproblem des MSC ist, dass es immer wieder dazu kommt, dass Schreibbefehle fehlschlagen und die Daten unvollständig in den Speicher übertragen werden. Dabei wurde davon ausgegangen, dass die Schreibbefehle fehlschlagen aufgrund, weil der Neustart des Fußschalters zu früh erfolgt und die zu schreibenden Daten nicht vollständig in den Speicher übertragen wurden. Schließlich zeigte sich das als eine grundlegende Fehlannahme und stattdessen ergeben Nachforschungen, dass die Zugriffe auf den Flash vom Softdevice abgearbeitet werden müssen und dass aufgrund mehrere aktive Verbindungen die Wahrscheinlichkeit von Fehlerhaften Schreibzugriffen stark ansteigt („nRF5 SDK v17.0.2 Flash Storage“, 2020, Abschnitt SoftDevice Backend). Getätigte Änderungen an den Konfigurationsfiles werden dann nicht übernommen und sind selbst nach langen Wartezeiten, nach dem Reset verloren. Deshalb werden, falls Schreibzugriffe gequeueet sind, alle Aktivitäten des Softdevice gestoppt, um die Gefahr von Fehlern beim Schreiben und die benötigte Zeit zu minimieren. Das umfasst das Trennen aller Verbindungen, sowie das Stoppen von Advertising und Scanning. Da durch einen gestarteten Timer der Fußschalter ohnehin neugestartet werden soll, damit die Konfigurationsfile neueingelesen werden können, ist die Beeinträchtigung der User Experience vernachlässigbar.

Diese Problematik besteht auch im Modus BLE-HID, da die Bonding Informationen über die selbe Library vom Softdevice in den Flash geschrieben werden („nRF5 SDK v17.0.2 Peer Manager“, 2020). Hat sich der Fußschalter zum Zeitpunkt des Verbindens mit dem Computer bereits mit mehreren Werkzeugen verbunden, besteht eine hohe Wahrscheinlichkeit, dass die Bonding Informationen fehlerhaft geschrieben werden. Der Fußschalter kann sich dann nicht mehr mit dem Computer verbinden beziehungsweise kann keine Zeichen bei einer bestehenden Verbindung übertragen. Die Lösung dieses Problems besteht darin, das Verbinden des Werkzeugs erst zu starten, wenn die periphere Verbindung zum Computer, bereits besteht. Da ohne die Verbindung zum Computer dieser Modus nicht nutzbar ist, ist die Userexperience dadurch nicht beeinträchtigt.

5.3. Einbindung Messuhren

In Vorbereitung auf die Implementierung der fußschalterspezifischen Features wird die Dongle-App um die Unterstützung der Messuhren bzw. Messschieber erweitert, da sich die meiste Funktionalität des Fußschalters speziell auf die Messuhren bezieht. Messuhren und Messschieber sind in diesem Kontext dabei gleich bedeutend, weil das HCT-Interface für beide Geräte identisch ist. Es wird sich nachfolgend jedoch stets auf Messuhren bezogen, da sie für die Anwendungsfälle des Fußschalters und Dongles bedeutender sind.

5.3.1. Unterscheidung der Geräte

Im einfachsten Anwendungsfall ist der Fußschalter mit mehreren Werkzeugen verbunden und sammelt passiven deren Messergebnisse auf. Unabhängig vom eingestellten Messmodus benötigt die Dongle-App von den Geräten einerseits das Messergebnis und andererseits die Messeinheit, um die Ergebnisse vollständig an den Computer weitergeben zu können. Diese befinden sich abhängig vom Gerät an folgenden HCT-Protokoll-Adressen:

	Datenblock	Adresse
Garant Drehmomentschlüssel		
Messergebnis	Measurement data block	0x00000B04
Messeinheit	Setpoint data block	0x00000C1F
Horex Drehmomentschlüssel		
Messergebnis	Device measurement result	0x0000240C
Messeinheit	Device measurement case config	0x0000221B
Messuhren/Messschieber		
Messergebnis	Device measurement result	0x00002408
Messeinheit	Device measurement case config	0x0000220F

Tabelle 1: Adressen Messergebnis und Messeinheit

In Tabelle 1 wird deutlich, dass die Adressen von Messergebnis und Messeinheit für diese drei Geräte jeweils unterschiedlich sind, auch wenn sie im Fall der Messuhren und Horex Drehmomentschlüssel im gleichen Datenblock liegen. Wird dieser als Binärdaten erhalten kann die Adresse des Datenblocks gelesen werden, die Information von welchem Gerät die Daten stammen und damit an welcher Stelle genau die jeweils die Daten zu finden sind, kann jedoch nicht festgestellt werden. Es bleibt also keine andere Möglichkeit als die verbundenen Geräte ihrem Typ zuzuordnen und die erhaltenen Daten entsprechend zu interpretieren.

Diese Unterscheidung mit welcher Art von Gerät kommuniziert wird, kann auf zwei verschiedenen Weisen erfolgen. Einerseits kann anhand des Namens das Gerät entweder der Klasse Drehmomentschlüssel oder Messuhr zugeordnet werden. Das ist jedoch unter Umständen fehleranfällig, da das Sortiment an HCT-Werkzeug stetig wächst und es nicht auszuschließen ist, dass in der Zukunft Geräte auf den Markt kommen, mit deren Namen es zu falschen Zuordnung kommt. Zudem gibt es alte Messschieber, die zwar unter einem korrekten Name advertise aber nicht das HCT-Protokoll sprechen, wobei diese Geräte aufgrund der

Inkompatibilität der Protokolle letztendlich nicht verbunden werden sollten.

Eine andere Methode ist, die “Device Information” abzufragen und anhand der Klassenidentifikationsnummer das Gerät einem Typ zuzuordnen. Das ist die präferierte Lösung, da neben der genaueren und sicheren Zuordnung, in der Device Information andere Informationen, wie die Protokoll Version mitgeliefert werden, die bei späteren Features unter Umständen benötigt werden. Zudem bleibt die Datenverarbeitung leichter erweiterbar um neue Geräte. Jedoch muss diese Abfrage in den asynchronen und mehrteiligen Ablauf des Verbindungsaufbaus eingefügt werden, wodurch ein höherer Entwicklungsaufwand entsteht. Auch benutzen Drehmomentschlüssel der Firma Garant eine andere Klassifikation der Geräte als das Werkzeug der Firma Horex, welche innerhalb der Dongle-App vereinheitlicht werden muss.

Da die Erweiterbarkeit und Flexibilität jedoch dem höheren Entwicklungsaufwands überwiegen, wurde sich zugunsten dieser Lösung entschieden. Dabei wird nach dem Verbindungszustand-Callback, der vom Central Device nach einer erfolgreichen Subscription der Anwendung auf die BLE-Charakteristik des Werkzeugs aufgerufen wird, eine Nachricht zu Abfrage der Device Information gesandt. Die erhaltenen Daten werden dann in der zum Gerät gehörenden Struktur gespeichert und anschließend die Abfrage der Messeinheit durchgeführt.

5.3.2. Anpassung der Messdatenverarbeitung

Auch die Datenverarbeitung, also die Interpretation der erhalten Daten über BLE muss angepasst werden, da das Messergebnis, wie bereits in Tabelle 1 gezeigt, zwar innerhalb des gleichen Datenblocks wie bei dem Horex Drehmomentschlüssel liegt, jedoch innerhalb des Datenblocks an einer anderen Stelle. Diese Offsets sind als Konstanten im Headerfile der Dongle-App definiert und müssen um die entsprechenden Einträge für die Messuhren erweitert werden. Werden die Daten erhalten, wird als Erstes die Adresse ausgelesen und dann je nach Typ des zugehörigen Geräts, die Interpretierung der Daten durchgeführt. Sollen also weitere Geräte eingebunden werden, können zusätzliche Sonderbehandlungen der Daten anhand des Typs hinzugefügt werden. Zudem handelt es sich bei dem Messergebnis, das von Interesse ist, beim Drehmomentschlüssel um den “Peak Torque”, der als Gleitkommazahl codiert ist, während bei der Messuhr das Ergebnis die “Measurement Distance” als Ganzzahl codiert ist. Dabei handelt es sich abhängig von der Messeinheit um Mikrometer oder Mikroinch. Es muss auf Millimeter beziehungsweise Inch umgerechnet werden, da die Ausgabe über CDC oder HID der Anzeige auf dem Gerätedisplay gleichen soll. Das binäre Messergebnis wird daher zunächst mit einem Umrechnungsfaktor von 1000 in Millimeter beziehungsweise Milliinch als Gleitkommazahl umgerechnet. Anschließend muss überprüft werden, ob es sich bei der derzeitige Einheit des Geräts um Inch handelt, da dann der Wert erneut durch 1000 dividiert werden muss, um von Milliinch auf die gewünschte Einheit Inch zu gelangen. Es wird also letztendlich eine Gleitkommazahl erhalten, wodurch sie ohne Anpassungen in der Nachrichten Struktur der Dongle-App gespeichert werden kann.

Die Einheitenkodierung der Messuhr ist komplementär zur Kodierung der Drehmomentschlüssel und daher kann die if-Cascade, welche die Zuordnung vornimmt, um die Einheiten der Messuhr erweitert werden. Jedoch befindet sich die Information welche Einheit verwendet wird, wie das Messergebnis, ebenfalls an einer anderen Stelle innerhalb des Datenblocks und muss entsprechend angepasst werden.

5.3.3. Gruppenfunktion

Durch das Betätigen des Fußtasters des Fußschalters soll bei allen verbundenen Messuhren das derzeitige Messergebnis abgefragt werden. Während im Modus 2 (CDC) ein Messergebnis, anhand der Kanalnummer einem Werkzeug zugeordnet werden kann, ist dies in den HID-Modi nicht der Fall. Es muss daher die korrekte Reihenfolge der Ausgabe der Messergebnisse sichergestellt werden. Es wurde sich entschieden, das `devices.csv` Konfigurationsfile um eine Spalte mit einer Gruppennummer zu erweitern, da somit der Anwender sowohl die Reihenfolge als auch welche Messuhren in der Gruppe sind, konfigurieren kann. Da durch das Abschicken der Nachrichten zur Abfrage der Messungen in der korrekten Reihenfolgen, das tatsächliche Erhalten in der gleichen Reihenfolge nicht sichergestellt ist, muss davon ausgegangen werden, dass die Nachrichten in einer zufälligen Reihenfolgen erhalten werden. Stattdessen muss bei der Ausgabe die Nachrichten umsortiert werden. Dazu bedarf es eines Zählers, der durch Betätigung des Fußschalters, von 0 auf 1 gesetzt wird, wodurch der Start der Gruppenfunktion später beim Erhalten der Messergebnisse erkennbar ist. Bei der Abarbeitung der erhaltenen Nachrichten, wird dann über die Zuordnung zum Device, die Nachricht ausgewählt und weitergegeben, die zum Counter korrespondiert. Die Gruppenids, die keinem der konfigurierten Geräte zugeordnet werden können, sowie die Gruppenids die zu unverbundenen Geräten gehören, werden dabei übersprungen. Zusätzlich soll ein Feature der Messeruhren genutzt werden, um die Gruppennummer auch auf der Messuhr anzuzeigen. Dazu werden den Messuhren ihre Gruppennummer nach dem Verbindungsaufbau via dem HCT-Protokoll übermittelt.

Durch spätere Anregungen von Messtechnikern, die für den Vertrieb von Messwerkzeug zuständig sind, ergab sich jedoch, dass wenn ein Gerät der Gruppe zwar konfiguriert, jedoch nicht verbunden ist, die präferierte Lösung ist die gesamte Gruppe nicht zu triggern. Das ergibt sich einerseits dadurch, dass die Messuhren sich aufgrund ihrer relativ kleinen Batterien bei Inaktivität schnell ausschalten und die Verbindung zu ihrem Central trennen. Andererseits soll möglichst jede durchgeführte Messung korrekt sein, da sie durch die CAQ-Software verarbeitet und gespeichert wird. Das Ziel der Sicherstellung einer korrekten Messung überwiegt hier also dem Gedanken der Benutzerfreundlichkeit, dass eine Messung auch dann gemacht werden kann, wenn einer der Geräte unverbunden ist. Der Fußschalter blinkt dann zwei Mal kurz rot auf, um diesen Fehler zu signalisieren.

Des Weiteren könnte es passieren, dass eine Messung zwar angefragt, aber nicht erhalten wurde. Die Anwendung würde dann blockieren, da auf die Nachricht gewartet wird und müsste neugestartet werden. Es wurde sich entschieden, einen Timer zu starten, falls eine Nachricht nicht erhalten wurde und bei seinem Ablauf statt dem Messergebnis eine Fehlermeldung auszugeben. Die restlichen Messergebnisse können dann korrekt ausgegeben werden.

Ebenfalls aus dem Feedback der Messtechnikern heraus, wurde ein sequentielles Triggern der Gruppe implementiert. Dazu muss der entsprechende Eintrag in der Konfigurationsdatei mit einer Eins auf aktiv gesetzt werden. Es wird bei den Geräten der Gruppe nacheinander, jeweils bei Betätigen des Fußtaster, die Abfrage eines Messergebnis ausgelöst.

5.4. Einbindung Hardware

Die Prototypen des Fußschalters der Firma Brecht wurden bereits vor Beginn dieser Arbeit erhalten und somit konnte direkt mit der Inbetriebnahme der neuen Hardware begonnen werden. Der Chipsatz des Fußschalters ist ebenfalls der PCA10056 von Nordic semiconductor, somit muss an der Software des USB-Dongles keine Änderungen vorgenommen werden. In diesem Kapitel werden die hardwarebezogenen Funktionalitäten des Energiemanagements, des Fußtaster und der light-emitting diode (LED) vorgestellt. Im Anhang finden sich die ursprünglichen Hardwarezeichnungen der Platine.

5.4.1. Energie Management

Um den Fußschalter herunterzufahren und den Stromverbrauch zu senken, sah eine erste Idee vor das im nrf_Base Projekt vorhandene Energiemanagement zu benutzen. Dieses fährt aus dem Main-Loop heraus getriggert den Chip bei Inaktivität des Softdevice weitestgehend herunter, wodurch der Stromverbrauch stark sinkt. Ist der USB-Port also nicht verbunden und es wurde keine Aktivität in der Fußschalter Anwendung registriert, wird ein Timer gestartet. Läuft dieser Timer ab, werden alle BLE-Verbindungen getrennt und falls notwendig Scanning und Advertising gestoppt. Wird während dem Laufen des Timers Aktivität registriert, wird er neugestartet, während er vollständig gestoppt wird, falls USB wieder verbunden ist. Dieser Implementierung lag die Vermutung zugrunde, dass wenn der Chip vollständig heruntergefahren wurde, er auch nicht durch Betätigung des Fußtasters wieder neugestartet werden kann.

Zunächst zeigte sich jedoch das Problem, dass bei der Hardware des Fußschalters, der Akku auf der Datenleitung des USB liegt. Dadurch wird in der Anwendung nicht wie bei dem EvalBoard die Events für USB connected und USB disconnect erhalten. Daher musste am Fußschalter geringfügige Hardwareänderungen durchgeführt werden. Dabei wurde die Eingangsspannung bereits vorher abgegriffen und auf den Pin des Fußtasters gelegt. Der Fußtaster erhält einen unbelegten Pin. Mit einem Analog-to-digital converter (ADC) wird dann überprüft, ob eine Spannung auf diesem Pin anliegt.

Es zeigte sich bei der Einbindung des Fußtasters, dass die Anwendung durch Betätigung des Fußtasters selbst aus einem Shutdown heraus wieder aufgeweckt wird. Daher wird nach Ablauf des Inaktivitätstimers die Anwendung vollständig heruntergefahren, was noch energieeffizienter ist.

5.4.2. Fußtaster Funktionalität

Nach Überarbeitung der Codebasis und Einbindung der Messuhren, stehen alle Funktionalitäten bereit um den eigentlichen Fußtaster einzubinden. Während bei der initialen Einbindung des Fußtasters, eine Betätigung das Abfragen der Messergebnisse bei allen verbundenen Messuhren triggerte, wuchs die Funktionalität stetig.

Zum Ende dieser Arbeit wird durch ein kurzes Betätigen oder einem “einfachen Klick” die bereits beschriebene Gruppenfunktion ausgeführt. Im Modus 0 (USB-HID-singleKey) und 4 (BLE-HID-singleKey) wird draufhin ein Timer gestartet, welcher derzeit auf 500ms gesetzt ist. Wird während seines Laufen eine zweite Betätigung getätigt, wird ein “Doppelklick”

registriert und das dafür in der Konfigurationsdatei hinterlegte Zeichen ausgegeben. Dadurch kann der User in diesem Modus schnell Dialogoptionen in der HCT-Windows-App auswählen oder in einem Textfile die Seiten wechseln. In einem anderen Modus führt das Warten auf die zweite Betätigung, jedoch zu einer Verzögerung der Ausgabe um die Dauer des Timers, weshalb dort die Doppelklick Funktionalität zugunsten des Ansprechverhalten des Tasters gestrichen wurde.

Wird der Taster hingegen für 3 Sekunden durchgängig gehalten, wird das Gerät heruntergefahren. Diese Funktionalität wurde ursprünglich eingeführt, da der Fußschalter auf einer Messe vorgestellt wurde, aber der automatische Reset des Geräts bei Änderung der Konfigurationsfiles noch nicht implementiert war. Sie wurde aufgrund von positiven Feedback beibehalten.

5.4.3. Inbetriebnahme LED

Auf dem Board des Fußschalters befindet sich eine LED die durch einen Lichtkanal nach außen hin durch das Gehäuse sichtbar gemacht wird und die dazu benutzt werden soll den internen Zustand des Geräts darzustellen. Folgende Zustände sollen abgebildet werden:

Zustand	LED-Farbe
Gerät im Sleep Modus	Aus
Alle zu verbindenden Geräte verbunden	Blau
Min. ein Gerät verbunden, es wird nach den fehlenden Geräten gescannt	Blau blinkend
Kein Gerät verbunden, Scanning inaktiv	Grün
Kein Gerät verbunden, Scanning aktiv	Grün blinkend
MSC-Schreibvorgang detektiert	Gelb
Min. ein Konfigurationsfile nicht gefunden	Rot
Fehler in den Konfigurationsfiles	Rot blinkend

Tabelle 2: LED-Zustände

Zudem blinkt der Fußschalter, wie in Kapitel 5.3.3 beschrieben, zwei mal kurz rot auf, wenn bei Betätigung des Fußtasters nicht alle zu der Gruppe gehörenden Geräte verbunden sind. Außerdem blinkt die Drei-Farben-LED des Fußschalters vor einem Neustart oder dem Ausschalten des Geräts zweimal kurz grün.

6. Ergebnisse

In diesem Kapitel werden die Ergebnisse der Implementierung vorgestellt. Zunächst werden die zentralen Funktionalitäten der Gruppenfunktion und HID vorgestellt. Daraufhin die Ergebnisse diskutiert und im Abschluss der Stand der Produktentwicklung vorgestellt.

6.1. Evaluierung

6.1.1. Gruppenfunktion

Bei der Gruppenfunktion war die größte Herausforderung der Implementierung, dass gerade in den HID Modi die Reihenfolge der Gruppe eingehalten werden muss damit die Messergebnisse in der Ausgabe wieder einem Werkzeug zuordbar sind. In Abbildung 7 ist die Ausgabe der Gruppenfunktion über den virtuelle COM-Port im Modus CDC zu sehen. Die Ausgabe wurde in Terminalprogramm RealTerm aufgefangen und die einzelnen Ausgaben der Gruppenfunktion durch rote Striche getrennt. Durch Betätigen des Fußtasters werden also jeweils die drei Zeilen zwischen den roten Strichen ausgegeben. Durch die Kanalnummern, die nur in diesem Modus Verwendung finden und jeweils am Anfang der Zeilen stehen, sind die Messergebnisse den einzelnen Geräten zuordbar. Zusätzlich wurden die Messuhren auf Messwerte eingestellt, die zu ihren Kanalnummern und Gruppennummern korrespondieren.

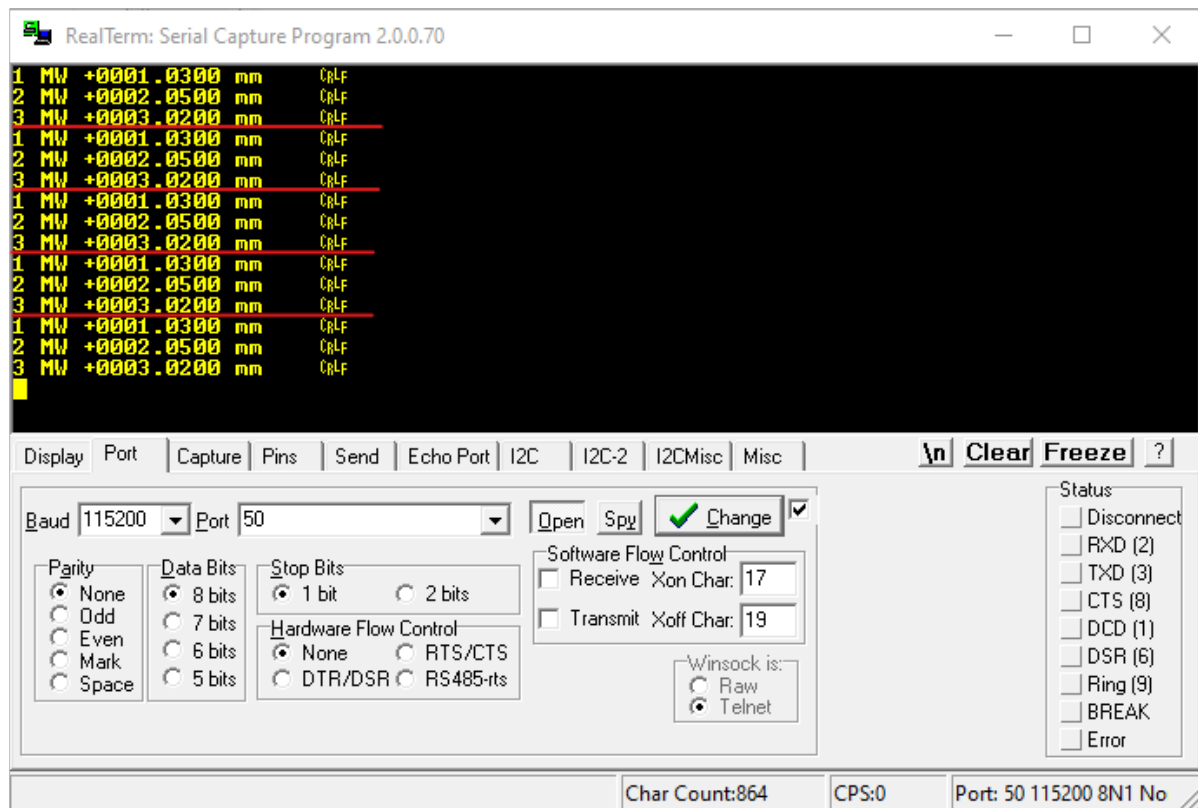
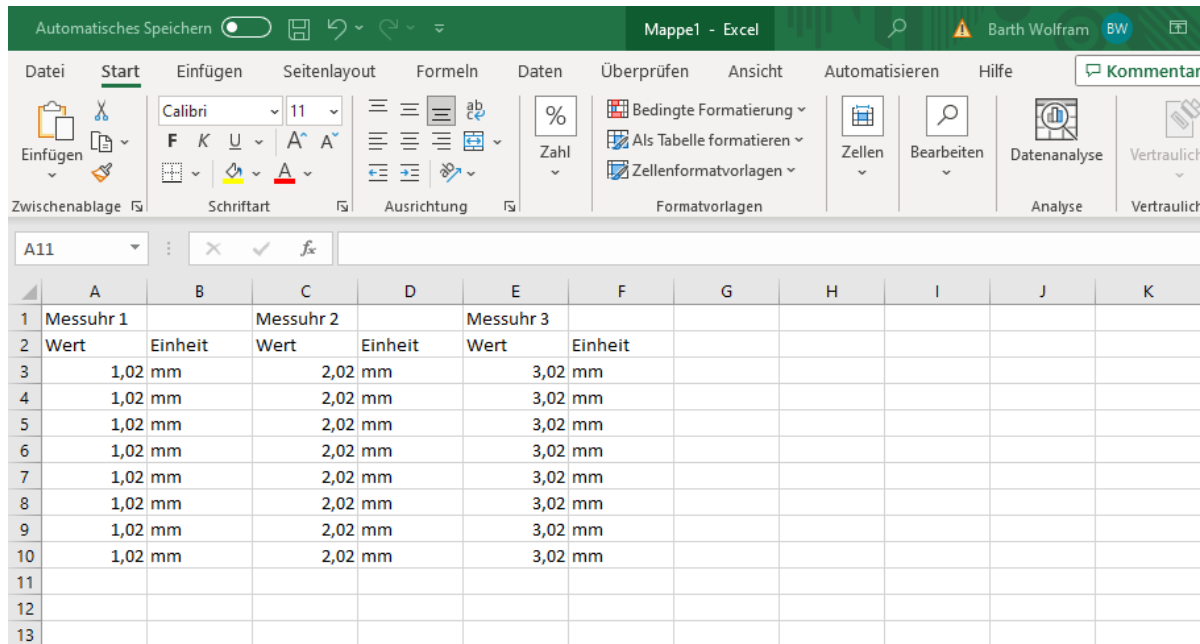


Abbildung 7: Durchführung von Messungen mit der Gruppenfunktion über CDC

Die gleichen Messungen sind in Abbildung 8 im Modus USB-HID durchgeführt. Durch Betätigung des Fußtasters wird jeweils eine Zeile Messwerte ausgegeben. Es ist ersichtlich, dass bei einer Vertauschung der Messergebnisse, bei echten Messdaten die von Zeile zu Zeile jeweils unterschiedlich sind, dies nicht nachvollzogen werden kann. Da hier jedoch die Einhaltung der Reihenfolge gezeigt werden soll, sind die Messergebnisse immer die Gleichen und korrespondieren zu ihren Gruppennummern. Es wird deutlich, dass auch bei mehreren durchgeführten Gruppenmessungen die Reihenfolge der Gruppe eingehalten wird.



	A	B	C	D	E	F	G	H	I	J	K
1	Messuhr 1		Messuhr 2		Messuhr 3						
2	Wert	Einheit	Wert	Einheit	Wert	Einheit					
3	1,02	mm	2,02	mm	3,02	mm					
4	1,02	mm	2,02	mm	3,02	mm					
5	1,02	mm	2,02	mm	3,02	mm					
6	1,02	mm	2,02	mm	3,02	mm					
7	1,02	mm	2,02	mm	3,02	mm					
8	1,02	mm	2,02	mm	3,02	mm					
9	1,02	mm	2,02	mm	3,02	mm					
10	1,02	mm	2,02	mm	3,02	mm					
11											
12											
13											

Abbildung 8: Durchführung von Messungen mit der Gruppenfunktion über HID

6.1.2. Zeitmessungen HID

Für die Modi USB-HID und BLE-HID ist die Zeit die verstreicht bis das gesamte Messergebnis eingetippt wurde das wichtigste Gütekriterium der Implementierung neben der offensichtlichen Fehlerfreiheit. In Abbildung 9 sind daher die Logausgaben der Aufrufe zur Funktion, die die einzelnen Tastendrucke an die USB-HID-Library übergibt zu sehen. In Klammern ist die verstrichene Zeit seit dem ersten Tastendruck in Millisekunden zu sehen. Abgesehen von der Zeitspanne zwischen dem ersten Tastendruck und Tasten loslassen, in der weniger als eine Millisekunde vergangen ist, liegen zwischen jedem einzelnen Aufruf genau zwei Millisekunden. Es ist zu vermuten, dass die Zeitspanne zwischen den ersten beiden Aufrufen niedriger ist, da die Daten noch nicht über USB gesendet wurden, sondern zwischengespeichert sind. Sobald begonnen wird die Daten tatsächlich über USB zu senden, vergehen zwischen zwei Tastendrucke genau zwei Millisekunden. Das Messergebnis wird schnell und gleichmäßig über USB eingetippt. Zu den Zeitstempeln 22 und 24 wird dabei die Tabulatortaste gedrückt um in einer Tabelle in die nächste Spalte zu wechseln, während zum Zeitpunkt 34 und 36 die Zeile terminiert wird und in die nächste Zeile gesprungen wird.

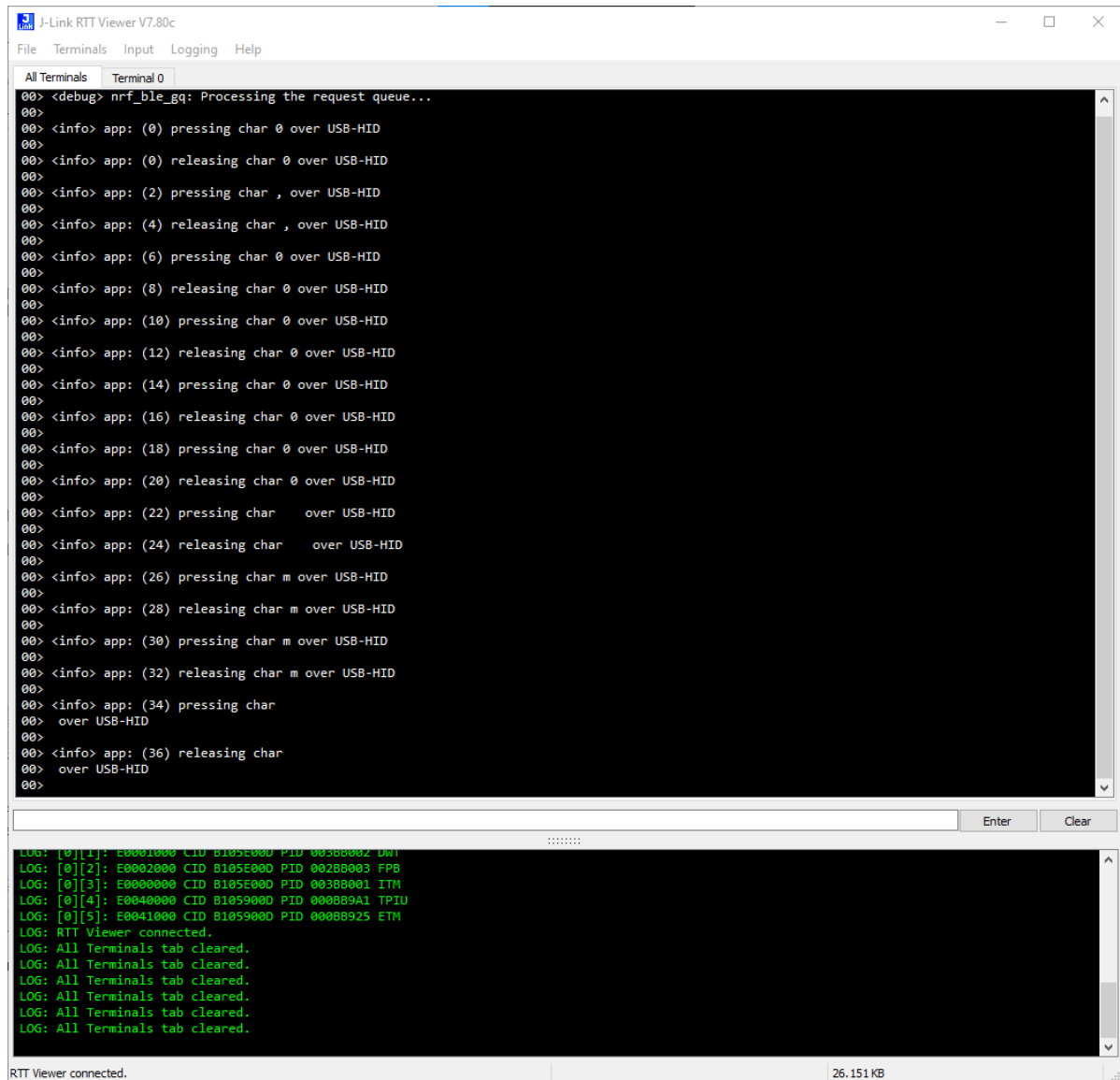


Abbildung 9: Logging einer Ausgabe eines Messergebnis über USB-HID

Anders ist das zeitliche Verhalten bei Eingabe des Messergebnis über BLE-HID. In Abbildung 10 ist der Mitschnitt der BLE-Nachrichten des Fußschalters an den Computer bei einer verbundenen Messuhr zu sehen. Es wurde der Ellisys Bluetooth Analyzer verwendet. Hier ist auch zu sehen wie zum Zeitpunkt 13:30:35.883 das Messergebnis als HCT-Nachricht von der Messuhr an den Fußschalter übertragen wird. Ab dem Zeitpunkt 13:35.948 werden die Tastendrücke vom Fußschalter an den Computer übertragen. Dabei werden die Codes für Tastaturtasten übertragen und nicht die Codes der ASCII-Zeichen. Der Code 0 steht dabei für das loslassen einer Taste. Es wird deutlich, dass die einzelnen Tastendrücke mit unregelmäßigen Abständen eingetippt werden. Die Telegramme zwischen denen weniger als eine Millisekunde verstreicht werden dabei in einem Zeitslot übertragen. Es kommt in unregelmäßigen Zeitabständen dazu,

dass mit ungefähr 110 Millisekunden eine verhältnismäßig lange Zeitspannen zwischen den einzelnen Nachrichten verstreicht, vorallem da das Connection Interval hier 30 Millisekunden beträgt und somit der Fußschalter deutlich mehr Zeitslots für weitere Übertragungen zur Verfügung hätte. Es wird beobachtet, dass die Ausgabe stottert.

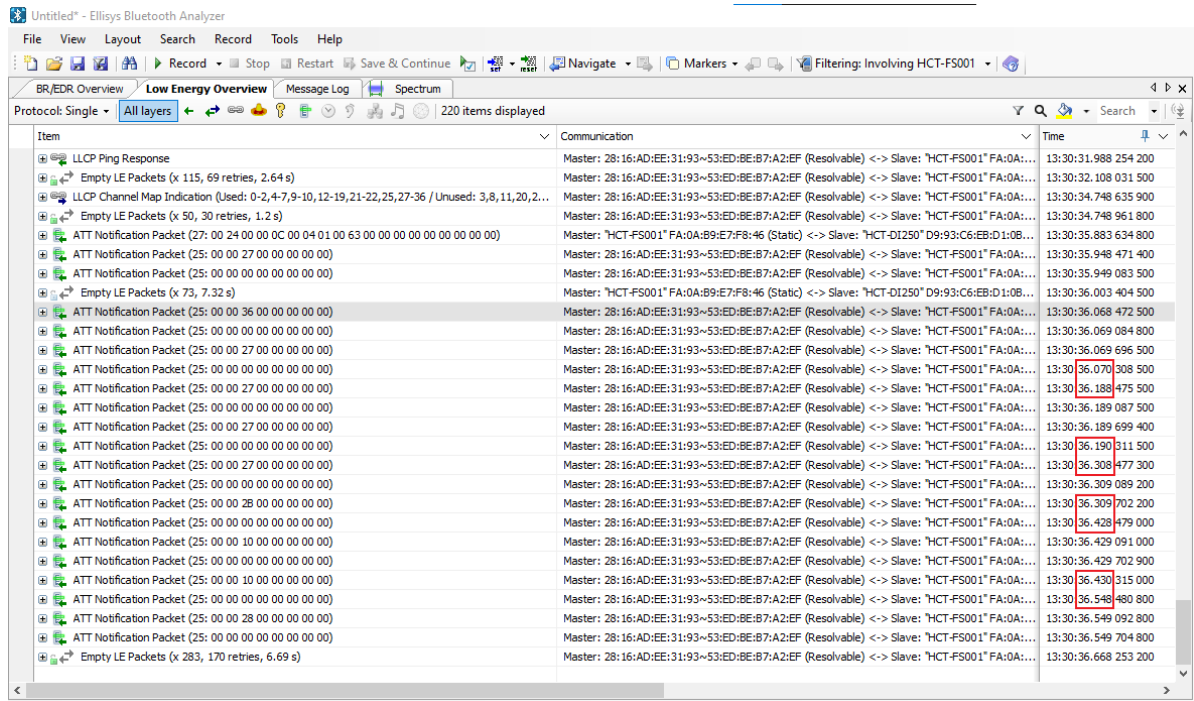


Abbildung 10: Mitschnitt der BLE Nachrichten eines BLE-HID Messergebnis von einer verbundenen Messuhr

Dieses Verhalten verschärft sich, wenn mehrere Messuhren verbunden sind. In Abbildung 11 ist der Mitschnitt der BLE-Nachrichten der Tastendrucke bei drei verbundenen Messuhren zu sehen. Die Übertragungen treten wieder gehäuft auf. In einem Zeitslot werden mehrere Zeichen übertragen zwischen denen weniger als eine Millisekunde verstreicht, während die Zeit bis zur nächsten Übertragung mit 236 Millisekunden fast achtmal so groß wie das Connection Interval von 30 Millisekunden ist. Interessanterweise beträgt die Zeitdauer in allen der drei roten Kästen jeweils exakt 236 Millisekunden.

Item	Communication	Time
ATT Notification Packet (25: 00 00 2B 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.408 795 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.409 407 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.410 019 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.410 631 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.411 243 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.411 855 600
ATT Notification Packet (25: 00 00 2B 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.412 468 500
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.648 187 400
ATT Notification Packet (25: 00 00 38 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.648 799 400
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.649 411 400
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.650 023 400
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.650 636 500
ATT Notification Packet (25: 00 00 36 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.651 248 400
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.651 861 400
ATT Notification Packet (25: 00 00 23 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.652 473 300
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.888 190 900
ATT Notification Packet (25: 00 00 1E 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.888 802 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.889 414 900
ATT Notification Packet (25: 00 00 27 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.890 026 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.890 638 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.891 250 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.891 862 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:10.892 474 900
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:11.128 194 800
ATT Notification Packet (25: 00 00 10 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:11.128 806 800
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:11.129 418 800
ATT Notification Packet (25: 00 00 10 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:11.130 030 800
ATT Notification Packet (25: 00 00 00 00 00 00 00 00)	Master: 28:16:AD:EE:31:93~>4D:E3:B7:93:51:8F (Resolvable) 53:ED:8E:B7:A2:EF (Resolvable...	13:59:11.130 642 800

Abbildung 11: Mitschnitt der BLE Nachrichten der BLE-HID Messergebnisse von drei verbundenen Messuhr

6.2. Diskussion

Die Koordinierung von mehreren Messuhren mit der Gruppenfunktion, sowie die Modi USB-HID und CDC funktionieren fehlerfrei und werden als vollständig implementiert betrachtet. In Verbindung mit den HID Modi ist die Gruppenfunktion eine Funktionalität die es noch in keinem anderen Produkt gibt und ist damit eines der zentralen Verkaufsargumente. Bislang müssen Messuhren einzelnen entweder als HID-Device oder über die HCT-Windows-App, welche jedoch keine HID Funktionalität zur Verfügung stellt, verbunden werden.

Im Modus BLE-HID kommt es dazu, dass die Ausgabe stottert also in unregelmäßigen Abständen verhältnismäßig große Zeitspannen verstreichen bis die Ausgabe fortgeführt wird. Dabei steht in Verdacht, dass die Connection Intervalle bei mehreren bestehenden Verbindungen nicht gut koordiniert werden. In Abbildung 12

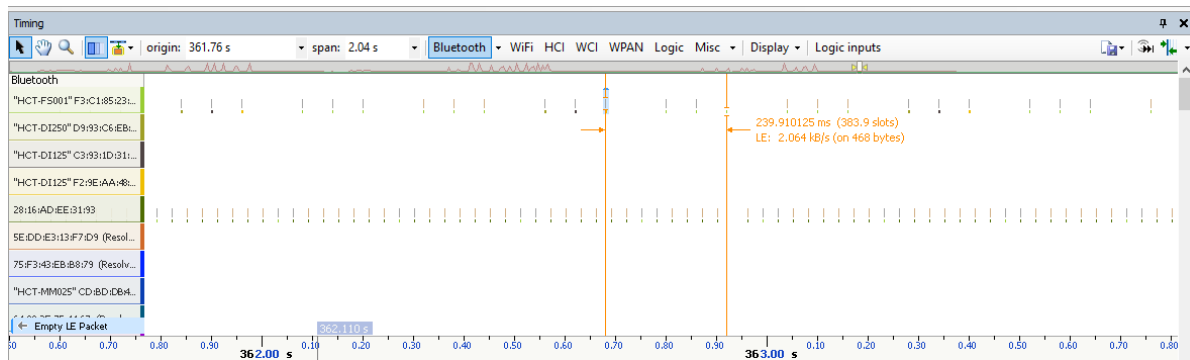


Abbildung 12: Diagramm der Connection Intervalle bei drei Messuhren und einer Verbindung zum Computer

6.3. Stand Produktentwicklung

Der Fußschalter und der Dongle sollen als eigenständige Produkte in das Sortiment der Hoffmann Group übernommen werden. Für sie werden jeweils Projekte vom Produktmanagement eröffnet. Dabei werden die Hardwareänderungen wie in Kapitel 5.4.1 beschrieben, in das Layout der Platine übernommen. Zusätzlich sollen die Pins, die den Chip in den Bootloader Mode versetzt, durch eine Taste von außen erreichbar gemacht werden. Außerdem sollen auch die SWE-Pads, die das Debugging auf dem Chip ermöglichen, erreichbar gemacht werden. Derzeitig sind sie auf der Seite mit der der Dongle auf die Platine gelötet wurde, weswegen sie nur sehr schwer abgreifbar sind. Die Prototypen dieser neuen Hardwareversion wurden gegen Ende dieser Arbeit erhalten und es wurde bereits festgestellt, dass die Taste die den Fußschalter in den Bootloadermodus versetzt nicht wie spezifiziert funktioniert. Der Zulieferer der Platine muss diese daher erneut überarbeiten. Des weiteren werden umfassende Funk- und Reichweitenmessungen durchgeführt, da die ursprüngliche Platine dort nicht die Anforderungen erfüllte.

Der Modus BLE-Windows-App wurde auf eine spätere Version verschoben, da die Integration der Messuhren und Messschieber in die Windows App andauert, sowie neue Geräte auf den Markt kommen sollen und deren Integration priorisiert wird. Eine Integration des Fußschalters ist daher noch nicht absehbar. Daher wurde die Implementierung des Modus zugunsten anderer Features bis auf Weiteres verschoben.

Insbesondere der Dongle ist bereits zu ersten Testläufen an ausgewählte Kunden gegeben worden und hat sich dort bereits bewährt. Desweiteren werden Codereviews mit anderen Entwicklern der Hoffmann Group durchgeführt.

Für die Funktionalität des Fußschalter beziehungsweise des Dongles wird eine Erfindungsmeldung herausgegeben und der Prozess der Patentierung der Technik wurde angestoßen. Siehe Anhang A.

7. Fazit

Der Implementierungsumfang für diese Arbeit war von Anfang an ambitioniert. Dennoch wurde alle geplanten Features außer dem Modus BLE-Windows-App implementiert und sogar noch zahlreiche zusätzliche Features, wie die Gruppenfunktion, entwickelt und umgesetzt. Während einige Funktionen, wie der Modus BLE-HID, noch Verbesserungspotenzial besitzen und für eine tatsächliche Produkteinführung noch ein umfangreiches Testen aller Funktionalitäten aussteht, war die Erarbeitung der Konzepte zur Anbindung der Werkzeuge und Verarbeitung der Messdaten ein voller Erfolg, was sich einerseits durch die bereits erwähnte Erfindungsmeldung (Anhang A) und andererseits durch die Allokierung von Ressourcen durch das Produktmanagement zeigt.

Die Anwendung die für diese Arbeit entwickelt wurde, ist auf Werkzeug der Hoffmann Group und damit verbunden auf das HCT-Protokoll limitiert. Sie deckt zum Ende dieser Arbeit zwei wichtige Messmittel ab: den Drehmomentschlüssel und die Messuhr. Es wurden Mechanismen geschaffen, um die verbleibenden und zukünftigen Messmittel der Hoffmann Group schnell und einfach einzubinden. Zudem gibt es Bestrebungen sowohl der Hoffmann Group als auch anderer Hersteller, Werkzeug das nicht das HCT-Protokoll spricht mit der HCT-Plattform kompatibel zu machen. Während die Datenverarbeitung des Fußschalters und des Dongles weitestgehend unabhängig von dem HCT-Protokoll ist, muss für eine Erweiterungen auf andere Protokolle, zusätzliche Mechanismen und Konzepte geschaffen werden.

Während die Hoffmann Group den Fußschalter und den Dongle also als Produkt verkaufen wird, kann die Forschung die Konzepte dieser Anwendung weiterführen und die Anbindung von nicht HCT-fähigen Geräten ermöglichen. Tatsächlich ist die Hoffmann Group in diesem Zusammenhang an der Entwicklung der Bluetooth Special Interest Group (SIG) eines generischen Profils für industrielles Werkzeug beteiligt. Mit dem Fußschalter steht dabei ein Gerät bereit, dass leicht darauf erweitert werden kann, Werkzeug welches dieses Profil implementiert an einen Computer anzubinden.

7.1. Ausblick

Derzeit sind bis zu fünf Verbindungen mit Werkzeugen und eine Verbindung mit einem Computer möglich. Hier kommt der Chip der Anwendung bereits an seine Grenzen, was in Zeitverzögerungen bis zur Ausgabe des Messergebnis im Computer führt. Da die Anzeige der Gruppennummer auf den Messuhren lediglich bis fünf geht, ist dies eine ausreichende Anzahl an Verbindungen, jedoch besteht das Potenzial die Performance der Anwendung zu verbessern und möglicherweise noch mehr Verbindungen zu ermöglichen.

Die Anwendung wurde erweiterbar für neue Messmodi gehalten. Fest geplant ist dabei der bereits beschriebene Modus BLE-Windows-App, jedoch sind noch weitere Modi denkbar, wie ein Modus in dem die durchgeführten Messungen in einer CSV-Datei persistiert werden. Die Detektierung von Änderungen an den Konfigurationsdateien birgt immernoch eine Reihe von Gefahren. Im Endprodukt wird es durch einen kleinen Schalter, der für den Lufttransport eingeführt wurde, möglich sein die Batterie des Fußschalters vom USB abzutrennen. Dadurch ist es möglich den Fußschalter vollständig von Außen auszuschalten. Werden die Daten durch den Computer auf das Massenspeichermedium geschrieben, wird dies dem

Anwender über die LED angezeigt. Schaltet er den Fußschalter dennoch aus kommt es zum derzeitigen Implementierungsstand dazu, dass die Konfigurationsdateien korumpiert werden. In diesem Fall ist es jedoch nicht sicher, in wie weit ein Verlust von Dateien verhindert werden kann. Eine Lösung des Problems könnte sein, beim erfolgreichen Einlesen der Dateien eine Sicherungskopie der Schlüsseldaten des Filesystems zu machen und diese zu Laden falls eine Inkonsistenz festgestellt wird, was im Falle des Fußschalters aufgrund der geringen Größe des Massenspeichermediums durchaus praktikabel sein könnte. Jedoch sollte auch eine weiterführende Analyse durchgeführt werden, was der Stand der Technik in diesem Zusammenhang ist, da dieses Problem für alle Massenspeicher Geräte existieren muss. Im Falle des Fußschalters können an einen Nutzer im industriellen Arbeitsumfeld jedoch etwas höhere Ansprüche gestellt werden, als an einen privaten Anwender und es wird erwartet, dass der Anwender eine Nutzungsanleitung liest oder in den Gebrauch eingewiesen wird. Dennoch werden die Konfigurationsdateien zum Download auf der Webseite der Hoffmann Group verfügbar gemacht, sodass beschädigte Dateien ersetzt werden können.

Eine weitere zusätzliche Funktionalität die denkbar ist, jedoch als Niederprior eingestuft wurde, ist es die Konfigurationsdateien in Hypertext Markup Language (HTML) neuzudesignen. Diese könnten dann in einem Browser angezeigt werden und würde es ermöglichen die Schlüssel der Attribute nicht editierbar zu machen, sowie drop-down-Menüs für Attribute mit begrenzten Auswahlmöglichkeiten einzuführen. Mithilfe von Javascriptcode könnte dann die Konfiguration in einer Datei persistiert und dann von der Anwendung eingelesen werden. Diese Funktionalität wurde noch nicht weiter verfolgt, weil einerseits die zeitliche Beschränkung dieser Arbeit dies nicht zugelassen hat und weil sie anderseits nicht ohne eine Javascript Komponenten auskommt. Dies wurde von einem Kollegen in einem ersten Gespräch behauptet und wirft das Problem auf, ob die stark gesicherten Computerumgebungen in der Fertigung die Ausführung von Javascriptcode zulassen. Des weiteren wurde diese Verbesserung des grafischen Interface grundsätzlich als niederprior eingestuft, weil der Fußschalter im industriellen Bereich eingesetzt wird und das Ausmaß seiner Interaktion damit fraglich ist.

Die Hoffmann Group steht in engen Kontakt mit ihren Kunden und bindet deren Feedback direkt in die Produktentwicklung ein. Es ist davon auszugehen, dass aufgrund des User feedbacks noch zahlreiche Fehlerbehebungen und Erweiterungen in den nächsten Jahren für den Fußschalter durchgeführt werden.

A. Erfindungsmeldung



MELDUNG EINER ERFINDUNG

Gemäß §5 des Gesetzes über Arbeitnehmererfindungen melde(n) ich/wir der
Firma

Hoffmann Engineering Services

nachstehende Erfindung mit der Bezeichnung;

Trigger mit Funkverbindung zu Werkzeugen und PC/Bluetooth Verbindung zwischen zwei
Werkzeugen

1. Erläuterung der Erfindung

Problemstellung oder technische Aufgabe

Momentan können Werkzeuge/ Messmittel mit einem PC verbunden werden, um erfasste Messwerte digital an eine auf dem PC installierte SW zu übermitteln. Um die eventuelle Verfälschung der Messung durch das händische Auslösen am Gerät zu verhindern oder eine möglichst zeitsynchrone Messung mehrerer Geräte zu erreichen, werden externe Trigger, wie z.B. Fußschalter verwendet, mit denen sich die verbundenen Werkzeuge fernauslösen (triggern) lassen, so dass ein aktueller Wert (Messwert) übertragen wird. Die Fußschalter/Trigger sind meist kabelgebunden, was deren Handhabung in einem industriellen Arbeitsplatz einschränkt. Bei nicht kabelgebundenen Triggern und Messgeräten muss eine SW installiert werden, die ihrerseits das Trigger-Signal entgegennimmt und an die konfigurierten Geräte weiterleitet. Sofern nur die Bluetooth-HID-Schnittstelle zur Datenübermittlung verwendet werden soll, entsteht zudem das Problem, dass bei einem zeitsynchronen Auslösen an mehreren Messgeräten die Messwerte nicht serialisiert und somit verfälscht und dem jeweiligen Messgerät nicht zugeordnet werden können. Durch die Erfindung muss einerseits keine zusätzliche SW installiert werden und andererseits müssen die Werkzeuge nicht mehr direkt mit dem PC verbunden werden. Der Trigger beinhaltet ein Bluetooth-Modul, mit welchen die Werkzeuge verbunden werden. Durch eine Bluetooth- oder Kabelverbindung des Triggers zu einem PC können dann die Werte an diesen PC übertragen werden. Die Übertragung kann über verschiedene Protokolle stattfinden (HID, HCT, MUX50, DMX16, ...). Zudem werden bei einer HID-Schnittstellen-Einbindung zeitgleich getriggerte Messwerte serialisiert und können den Messgeräten zugeordnet werden.

Stand der Technik

Bisher wird der Trigger nicht mit den Werkzeugen verbunden. Es benötigt zusätzliche HW/ SW um die Fernauslösung der Werkzeuge einzurichten.

Lösung der technischen Aufgabe bzw. der grundsätzliche Erfindungsgedanke

Durch die direkte Funk-Verbindung der Werkzeuge mit dem Trigger kann eine Fernauslösung ohne zusätzliche SW am PC erfolgen. Der Trigger dient als Bindeglied zwischen den Werkzeugen und dem PC.

Ausführungsbeispiel

Klicken oder tippen Sie hier, um Text einzugeben.

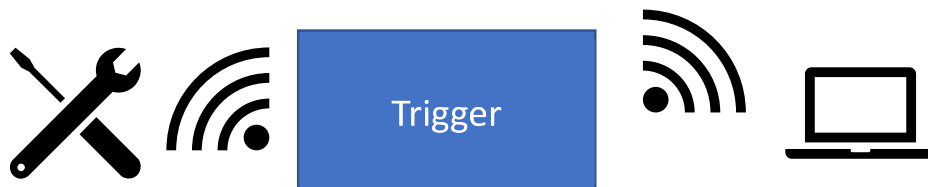
Trigger mit Funkverbindung

Seite 1 von 6

schematische Zeichnung

Klicken oder tippen Sie hier, um Text einzugeben.

A – HCT Fußschalter



Neuartig:

- Standard Schnittstelle für CAQ Systeme ohne SW Installation
- Unkomplizierte Konfiguration welche Geräte verbunden sein sollen (über cdc mit .ini und .csv Datei)
- Konfiguration von Geräten zu einer Gruppe
- Funk Verbindung zwischen Trigger und Werkzeug und Trigger und PC ohne SW Installation am PC (Standard SW kann verwendet werden, z.B. Microsoft Office, Notepad, ...)
-

Nutzen:

- Kein Kabel, (Werte werden über BLE-HID an den PC geschickt)
- Keine SW muss installiert werden
 - o da Trigger SW enthält
 - o da Kommunikation über Standardschnittstelle mit PC
- Keine Kollision der Messwerte, wenn mehrere Messgeräte gleichzeitig getriggert werden
- Zuordnung der Messwerte zu den Messgeräten ist durch Spaltenzuordnung oder Identifizierung möglich

B – HCT Fußschalter



Neuartig:

- Bi-direktionale Übertragung von Informationen (über verschiedene Protokolle)

Nutzen:

- Direkte Übergabe der Messwerte an eine CAQ SW oder ein MES System
- Konfiguration von Geräten zu einer Gruppe
- Zuordnung mehrerer Geräte möglich

C – HCT Fußschalter

Neuartig:

- Unterschiedlich Trigger-Möglichkeiten durch Klick, Doppelklick und Long Press am Fußschalter
 - o Konfiguration unterschiedlicher Tastatur-Signale (Enter, Tab, PgUp PgDown)
 - o Konfiguration unterschiedlicher Messtrigger für die Geräte (Messwert-Trigger, Preset,)
 - o Konfiguration unterschiedlicher Messmodi für die Geräte möglich (Indicating, Min, Max, Delta)

D – HCT Dongle

Neuartig:

- Standardschnittstelle für CAQ Systeme ohne SW Installation
- Unkomplizierte Konfiguration welche Geräte verbunden sein sollen (über cdc mit .ini und .csv Datei)
- Konfiguration von Geräten zu einer Gruppe
- Konfiguration unterschiedlicher Messmodi für die Geräte möglich (Indicating, Min, Max, Delta)
- Keine Kollision der Messwerte, wenn mehrere Messgeräte gleichzeitig benutzt werden
- Zuordnung der Messwerte zu den Messgeräten ist durch Spaltenzuordnung oder Identifizierung möglich

2. Dringlichkeitsvermerk für die Einreichung einer Patentanmeldung beim Deutschen Patentamt

Besteht die Gefahr, dass eine Weitergabe der Erfindung an Dritte (= Nicht-Angehörige der Firma z.B. Kunden, (externe Kooperationspartner) kurz bevorsteht? Ja

wenn ja, wann? 06.06.2023

an wen? Über Messtechniker an Kunden (z.B. ZF)

wird eine Geheimhaltungserklärung für Dritte benötigt? Ja

3. Voraussichtliche Anwendungsmöglichkeit der Erfindung:

- a.) innerhalb der Firma (Projekt, Baureihe, Gegenstand): Klicken oder tippen Sie hier, um Text einzugeben.
- b.) bei anderen Firmen (Drittfirmen): Klicken oder tippen Sie hier, um Text einzugeben.

4. Personalien des oder der Erfinder(s)

Name, Vorname	Privatanschrift	Anteil an der Erfindung
1. Künstler, Marko	Franziska-Schmitz-Str. 4 80634 München	33%
2. Lippach, Freddy	Buchendorfer Str. 3, 81475 München	33%
3. Barth, Wolfram	Thalkirchnerstr. 88 80337 München	33%
4.		

weitere Erfinder (ggfs. bitte auf separatem Zusatzblatt aufführen)

5. Einordnung der Erfindung

Die Erfindung soll in der HCT Plattform eingesetzt werden.

Ist die Erfindung am Fremdprodukt feststellbar? Ja

Welches Fremdprodukt? Klicken oder tippen Sie hier, um Text einzugeben.

Kann die Lösung der Erfindung technisch umgangen werden? Ja

Kann die Erfindung als Sperrpatent gegen Konkurrenten dienen? Ja

6. Wie wurde die der Erfindung zugrunde liegende Aufgabe veranlasst?

Hinweis: von jedem Erfinder ist nur jeweils eine der folgenden Möglichkeiten a) bis c) anzukreuzen:

	Erfinder			
	1.)	2.)	3.)	4.)
a) Die Aufgabe wurde gestellt				
mit Angabe des Lösungswegs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ohne Angabe des Lösungswegs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
b) Die Aufgabe wurde nicht gestellt; sie ergab sich durch infolge der Betriebszugehörigkeit erlangte Kenntnis von Mängeln und Bedürfnissen, welche vom Erfinder	1.)	2.)	3.)	4.)
nicht selbst festgestellt wurden	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
selbst festgestellt wurden	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
c) Der Erfinder hat sich die Aufgabe gestellt	1.)	2.)	3.)	4.)
innerhalb seines Aufgabengebiets	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
außerhalb seines Aufgabengebiets	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

7. Wie wurde die Lösung der Aufgabe gefunden?

Hinweis: von jedem Erfinder können mehrere oder keine Möglichkeit zutreffen:

	Erfinder			
	1.)	2.)	3.)	4.)
Die Lösung der Aufgabe wurde gefunden				
a) mit Hilfe der dem Erfinder beruflich geläufigen Überlegungen	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
b) aufgrund betrieblicher Arbeiten oder Kenntnisse	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

c) mittels der Unterstützung des Betriebs mit technischen Hilfsmitteln

☐ ☐ ☐ ☐

8. Betriebliche Stellung des/der Erfinder

	Erfinder			
	1.)	2.)	3.)	4.)
ohne Vorbildung (z.B. ungelernte Arbeiter, Hilfsarbeiter, Angelernte, Lehrlinge, ...)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
handwerklich-technische Ausbildung (z.B. Facharbeiter, Laboranten, Monteure, einfache Zeichner, ...), evtl. mit kleineren Aufsichtspflichten (z.B. Vorarbeiter, Untermeister, Schichtmeister, ...)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
untere betriebliche Führungskräfte (z.B. Meister, Obermeister, Werkmeister, ...), ggf. gründlichere technische Ausbildung (z.B. Chemotechniker, Techniker, ...)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fertigung: Tätigkeit, gehobene technische Ausbildung (z.B. Universität, Technische Hochschule, höhere technische Lehranstalten, Ingenieur-/Fachhochschulen)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Fertigung: leitende Tätigkeit (z.B. Ingenieure, Chemiker als Gruppenleiter) Entwicklung: Tätigkeit (z.B. Ingenieure, Chemiker)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Fertigung: Leiter einer Fertigungsgruppe (z.B. technischer Abteilungsleiter, Werksleiter), Entwicklung: Gruppenleiter von Konstruktionsbüros und Entwicklungslaboratorien, Forschung: Tätigkeit (z.B. Ingenieure, Chemiker)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Entwicklung: Leiter der Entwicklungsabteilung Forschung: Gruppenleiter	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Forschung: Leiter der gesamten Forschungsabteilung, technische Leiter größerer Betriebe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Ich/Wir erkläre(n) hiermit, die vorstehenden Angaben nach bestem Wissen und Gewissen gemacht zu haben.

1. 2.07.08.2023 W. Baurth
Datum / Unterschrift

2.07.08.2023 / W. Baurth
Datum / Unterschrift

3. 07.08.2023 W. Baurth
Datum / Unterschrift

4. _____
Datum / Unterschrift

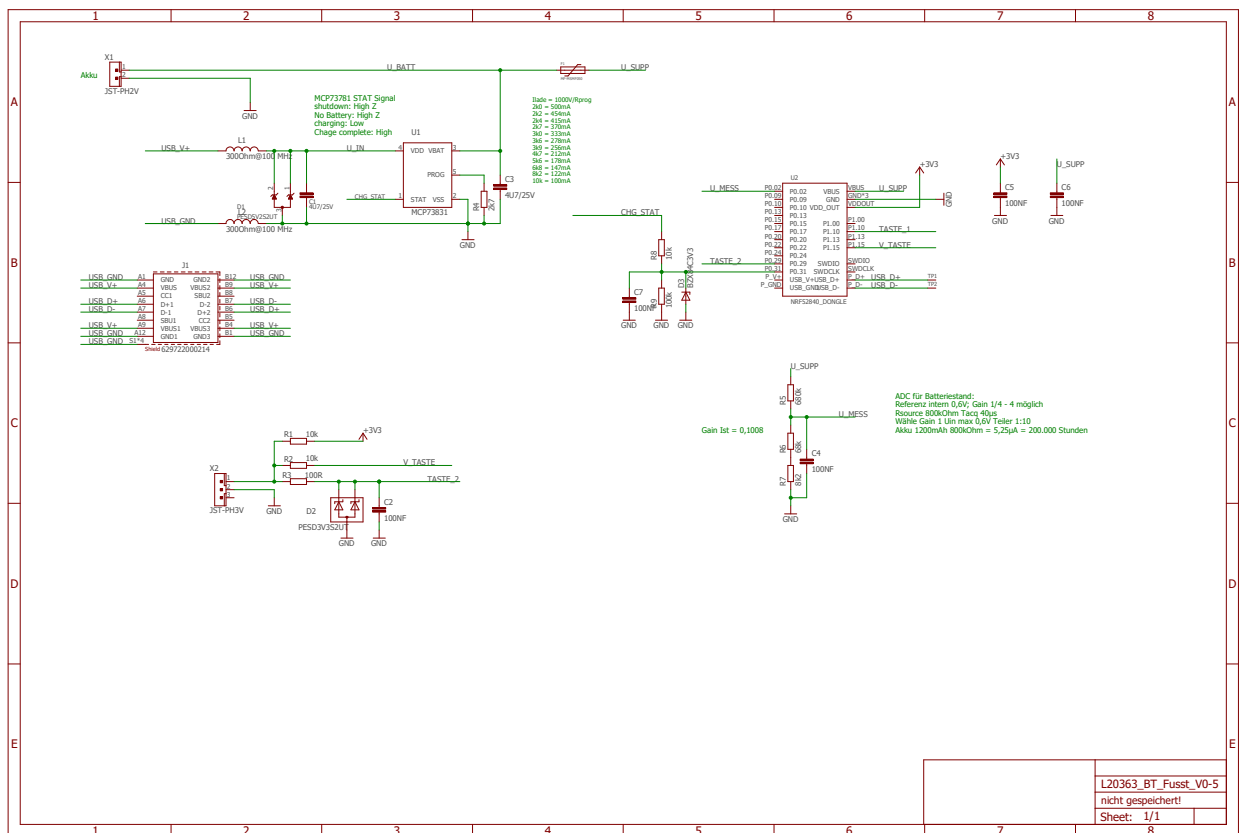
Bitte die Erfindungsmeldung bei ip@hoffmann-group.com und dem Vorgesetzten einreichen.

Bestätigung des Eingangs:

Datum: 08.08.2023  Hoffmann Group
Hoffmann Engineering Services GmbH
Haberlandstraße 55 · D-81241 München
www.hoffmann-group.com

Unterschrift: 

B. Hardwarezeichnung



C. HCT-Protocol-Description



Version	Date	Name	Description of Change
0.0.12	22.11.2020	Lippach	Derived from Protocol description Horex
0.0.13	24.11.2010	Lippach	First revision after internal review
0.0.14	06.12.2020	Lippach	Second revision after internal review
0.0.15	16.12.2020	Lippach	Extensions regarding Service UUID's
0.0.16	12.01.2021	Lippach	Definition of Product/Tool ID
0.0.17	17.02.2021	Lippach	Extension regarding SGTIN
0.0.18	19.03.2021	Lippach	Extension for SCAN-Response
0.0.19	14.04.2021	Lippach	Serial Number in Local Name
0.0.20	08.06.2021	Lippach	Extension Identification flag
0.0.21	11.06.2021	Lippach	Identification flags bitwise
1.0.0	08.08.2021	Lippach	Final revision after release of FW Version 1.0.0
1.0.1	27.04.2022	Lippach	Company Prefix in Advertisement corrected
1.0.2	27.04.2022	Lippach	Proposal for ProtocolType in Advertisement

HCT Protocol Description

V1.0.2



Inhalt

1	Glossary	4
2	Executive Summary	4
3	Requirements	4
4	BLE-Interfaces of the HCT-BLE-MODULE	6
4.1	Advertising Structure of the HCT-BLE-Module	7
4.1.1	GTIN (EAN)	9
4.1.2	DMC-Definition	10
4.1.3	Local Name – Identification for HID on Windows.	10
4.2	General description of the HCT-Service-Protocol	11
4.3	The HCT BLE Protocol-Frame	12
4.4	Virtual Address Map	16
4.5	HCT Live Data Service	17
5	Interface from Tool Application Module to HCT-BLE-Module	18
5.1	Embedding of the HCT-BLE-Module in a Tool - Conceptual View	18
5.2	Necessary HW Interfaces from and to the HCT-BLE-Module	18
5.2.1	Serial Interface	18
5.2.2	GPIO's (From App-Module to HCT-BLE-Module)	18
5.2.3	Interface Required for Testing Purpose on the Tool-Controller	19
5.2.4	UART-HW Interface Description	19
5.3	UART-SW Interface Description	20
5.4	The UART-Data Frame	20
5.5	Definition and Meaning of the Link-Handles	20
5.6	Handling of Link-Handles	22
6	Use Cases	23
6.1	Startup-Synchronization between HCT-BLE-Module and Tool-Application Module	23
6.2	Initialization of HCT-BLE-Module	24
6.2.1	Check Serial Interface Version	26
6.2.2	Advertisement Data	27
6.2.3	Security Data	28
6.2.4	SIG Service Configuration	28
6.2.5	User Service Configuration	30
6.3	Download Initialization	32
6.3.1	Method 1: Triggered by User via Serial Protocol	32
6.3.2	Method 2: Triggered by the User if no Application or a not Working or Corrupt Application is Loaded	33
6.4	Connection Status	34
6.5	Live Data	34
6.5.1	Combination of Sending of Live Values	35



6.6	Additional Use Cases that Will be Defined by Hoffmann	36
7	<i>Proxy-Library for Easier Integration</i>	36
8	<i>Documents</i>	37



1 Glossary

Server	The server is the one who holds data and can offer parts of its data. In the context of BLE the Server will be the unit which represents the slave role. The Server will send the data if the client asks for or send data notifications if internal data have been changed and client is interested in.
Client BLE_Client	The client can send data to the server and asks for data in order to remain updated. The client will usually also receive data as notifications, if data have been changed.
BLE-Proxy-Library	Collection of C-based Header and source code files which help to serialize and deserialize the serial protocol to the BLE-Controller and the data flow from and to the BLE-clients.
Master	= BLE – Client / Central Device
Slave	= BLE – Server / Peripheral Device

2 Executive Summary

The current document describes the usage of the HCT-BLE-Module in order to be used in a tool for embedding it into the Hoffman-Connect-Tools infrastructure. It gives an overview about the BLE-Interfaces provided by the module, especially the HCT-Service-Protocol, as well as the interface between the Tool Application Module and the HCT-BLE-Module. Detailed information about the API will be provided by another document.

3 Requirements

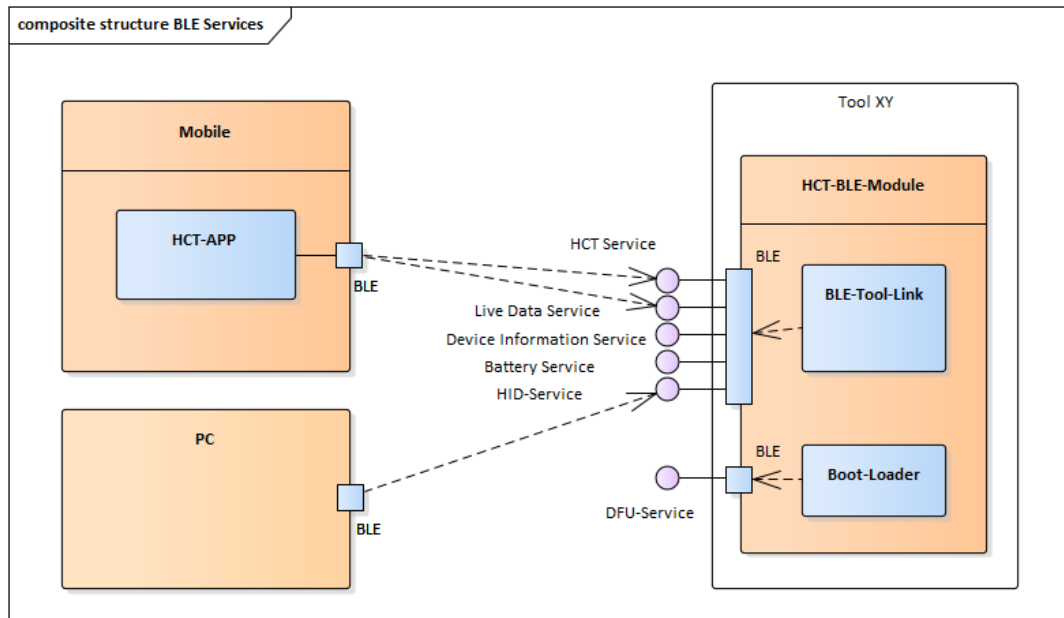
The following functional and non-functional requirements are the basis for the design of SW- and interface architecture.

- Functional Requirements
 - Writing/Reading and publishing of data
 - Creation, Reading, Writing and Deleting of list entries
 - Reading (and Writing) of entire lists
 - Sending live data asynchronously
 - Sending HID-Data asynchronously
- Non-Functional requirements
 - Encapsulation of BLE-Complexity towards the supplier of a tool
 - Simple interface to Supplier Controller
 - Notification when data has been changed
 - Multi-Client-Capability
 - Dynamic Advertisement
 - Dynamic Configuration of Services (e.g. HID)
 - Performance
 - Energy saving by optimized power consumption
 - Security → Dynamic Configuration of Security-Levels
 - Reducing of data transmission load to a minimum for energy saving





4 BLE-Interfaces of the HCT-BLE-MODULE



Currently there are five services: the HCT-Service, the Live Data Service and the HID-Service. It is planned to use the SIG adopted Battery- and Device Information Service and in future develop generic services that fit to the interoperability with other devices. For Firmware Update the DFU-Service is used.



4.1 Advertising Structure of the HCT-BLE-Module

The advertising data (28 Byte) compiled by the HCT-BLE-Module has the following structure, explained by the example:

02 01 06 03 03 12 18 0A 08 48 43 54 2D 54 57 30 31 32 08 FF A3 08 02 01 02 00 02 FE FE

# of bytes	Explanation	Value (Hex)	Content
2	Length and type	03 19	3 Byte Type: Appearance
2	Appearance	C1 03	0x03C1 → Keyboard
2	Length and type	02 01	2 Byte, Type: Flags
1	Flags (Capabilities, bit coded)	06	LE Limited Discoverable Mode → No LE General Discoverable Mode → Yes BR/EDR Not Supported → Yes Simultaneous LE and BR/EDR (Controller) → No Simultaneous LE and BR/EDR (Host) → No
2	Length and type	0A 08	10 Byte, Type: Shortened Local Name
9	Local Name	48 43 54 2D 54 57 30 31 32	ASCII Values for "HCT-<TT><VVV>" TT = Tool Type; VVV = Variant or Version "HCT-TW012", see [Fehler! Verweisquelle konnte nicht gefunden werden.]
2	Length and type	08 FF	8 Byte, Type: Manufacturer Specific Data
2	Company ID	A3 08	0x08A3 → (Hoffmann SE Manufacturer ID)
1	Brand Indication	00 .. FF	00: Unknown 01: Garant 02: Horex 03: tbd
1	Identification Flags	00 .. 03	Bit 0 = 0: APP-Connection is deactivated Bit 0 = 1: APP-Connection activated, Bit 1 = 1: Tool is in usage (e.g. when button is pushed)
1	ProtocolType	02	0x0 = unknown or not implemented 0x1 = HCT 1 (old DTW) 0x2 = HCT 2 (current Generation) 0x10 = Sylvac-Protocol 0x11 = Mahr-Protocol (0xFE = HCT 2 (current Generation Horex))
2	Reserve	FE FE	
2	Length and Type	02 0A	2 Byte, Type: TX-Power Level
1	TX-Power	00	0 dBm
31	Gesamt		



For the tool identification the scan response data must be used. This data can be acquired by a scan request from the central device, if the peripheral indicates that more advertising data are available.



Scan response data example

# of bytes	Explanation	Value (Hex)	Content
2	Length and type	03 03	3 Byte, Type: Complete List of 16-bit Service Class UUIDs (here only one will be listed)
2	UUID	12 18	0x1812 → HID Service UUID
2	Length and type	12 FF	18 Byte, Type: Manufacturer Specific Data
2	Company ID	A3 08	0x08A3 → (Hoffmann SE Manufacturer ID)
4	GTIN (EAN) GS1 Company Prefix	C6 FC 3D 00	0x3D FCC6 → "4062406" 7 decimal places
4	GTIN (EAN) Item Reference	ED 7E 01 00	0x00017EED → "98029" 5 decimal places
6	Serial Number	81 F9 52 BA 00 00	0x00BA52F981 → "003126000001" 12 decimal places
9	Reserve	00	
31 Bytes			

4.1.1 GTIN (EAN)

The GTIN is an individual, purely numeric article number from the GS1 numbering system. It identifies each product uniquely worldwide.

It contains the company prefix which identifies the producer and the item reference which identifies the product type and its sub-types uniquely.

An additional serial number is added in a way, that it compiles a SGTIN, that is used by the HCT-App in order to distinguish between advertising devices of the same type by reading a DMC-Code which is lasered on a metal part of the tool, printed on a sticker or shown on the display of the tool.



4.1.2 DMC-Definition

The Data Matric Code (DMC) shall be used for identifying the respective tool by the camera of a mobile device in order to establish the right connection.

The DMC-Code complies to the VDMA 34193.

Template and Example of a character string with leading FNC1 character and serialized identification number (SGTIN) for encoding in the GS1 DataMatrix

Template:

<FNC1>01 0CCCCCCC RRRRR2 21 SSSSSSSSSSSS

C = Company Prefix with 7 numerical characters

R = Item Reference with 5 numerical characters

S = Serial Number with 12 numerical characters

Example with reference to the GTIN and Serial Number in Scan response data

<FNC1>01 04062406 980292 21 003126000001

4.1.3 Local Name – Identification for HID on Windows.

The “Local Name” in the HCT advertisement is the so called ‘shortened local name’.

It only gives an information about the tool-type or tool- characterization, not about its unique identification. It must comply to the HCT advertisement naming convention, defined in [[Fehler! Verweisquelle konnte nicht gefunden werden.](#)].

The full local name consists of the shortened local name and can have up to 30 characters, e.g.

“HCT-<TT><VVV>-<Serial Number>”

whereby the possible name extension up to 20 characters shall contain a unique information that makes it possible to distinguish between two or more tools of the same type, especially when connected as a HID-device to Windows. It shall contain the serial number of the tool which is also part of the SGTIN of the DMC.

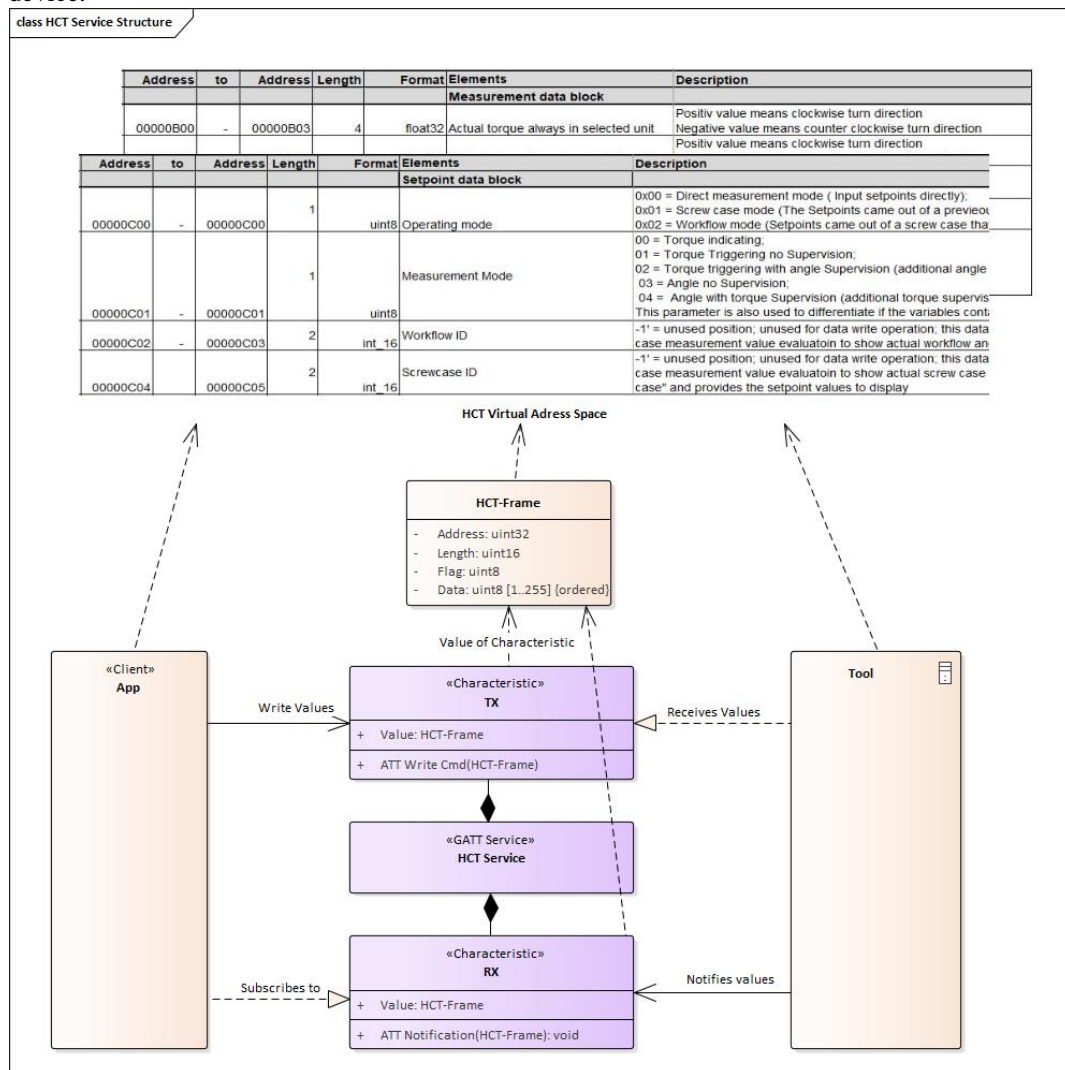
The complete name can be acquired by reading the device name characteristic after the connection has been established using GATT.



4.2 General description of the HCT-Service-Protocol

The protocol itself in fact describes the access to a common process data interface. This process data interface is represented by a virtual memory. The virtual memory is divided into different blocks which contain data to a specific meaning or function, like structures or classes, for example measurement data or setpoint data. The meaning and structure of the data at the specific addresses in the data blocks is currently described in a generic device-xml data and device specific data -xml file which extends the generic data. This data is individual for each device and depends on the device capabilities. The start addresses of the respective blocks, also called base addresses or base identifier, will be used as a reference into the internal memory structure, whereby, depending on the development environment the respective block can be implemented as class or structure and assigned to its base address.

XML or equivalent excel-file can give a description what kind of data is available from the device.





The figure above shall illustrate the basic structure and philosophy of the HCT Service that consists of two proprietary GATT-Characteristics, the HCT-TX and HCT-RX characteristics. Both contain a value of the type *HCT-Frame* which is in fact a structure, that points to a certain element and number of subsequent elements of the virtual address space or parts of it and indicates whether these data needs to be transferred from the client to the server or vice versa. This frame shall be transferred, either by a Write Command of the TX-Characteristic from the Client (App) to the Server (Tool) or by a Notification of the value (HCT-Frame) of the RX-Characteristic from the Client to the Server.

In order to access this virtual memory and its blocks or classes and manipulate them, some generic methods are available which are described in the next chapter.

HCT_PROTOCOL_SERVICE_UUID	10150100-34bb-41c7-bc33-78855b62d28e	Description and Access Method
HCT_PROTOCOL_TX_UUID	10150101-34bb-41c7-bc33-78855b62d28e	Data transfer from Client (central device) to the Server (peripheral device) ATT Cmd Write (no Request)
HCT_PROTOCOL_RX_UUID	10150102-34bb-41c7-bc33-78855b62d28e	Data transfer from Server (peripheral device) to the client (central device) ATT Notification

4.3 The HCT BLE Protocol-Frame

The protocol frame which also describes the structure of the HCT BLE TX and RX-characteristics consists of:

HEADER			DATA
ADDRESS	LENGTH	FLAG	DATA
4 Byte	2 Byte	1 Byte	MAX:256 Byte

ADDRESS:

The address is in fact a virtual address into the process data space. The Address space is divided into 256 Byte pages. Only one page can be written at once in a standard transmission procedure. The address map has some generic blocks respectively classes but can be extended by tool specific information. The available data content is in the device specific documentation.

LENGTH:

The Length gives the amount of DATA in byte that should/could be transferred, without header.

FLAG:

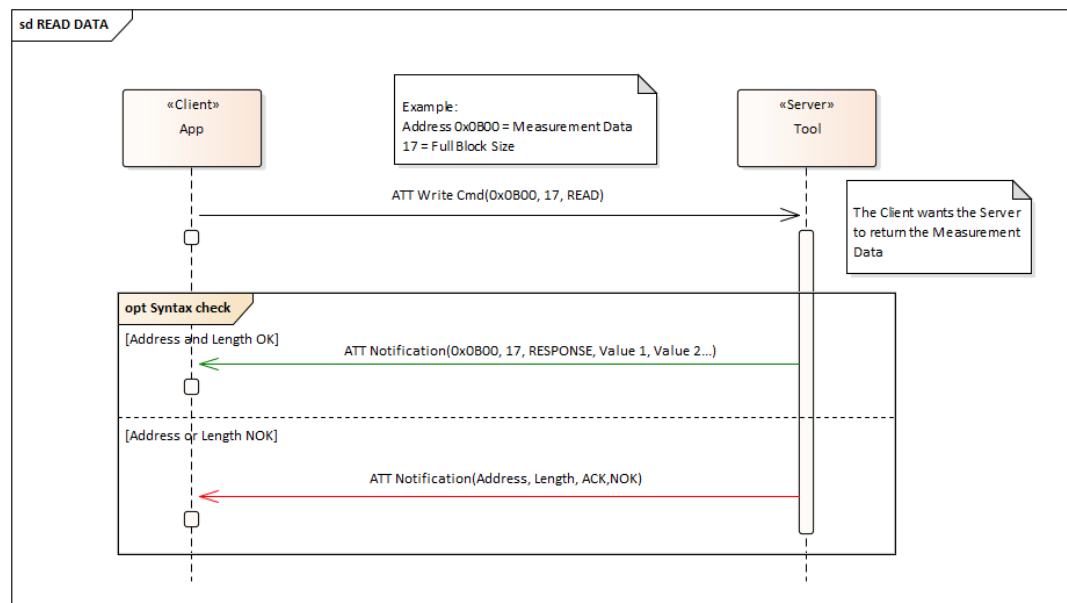
The Flag indicates the type of transmission.

FLAG - Byte			
Value	Short name	Explanation	Reaction of Receiver
0x00	READ	This is a query for data from the client to be returned, from the server	RESPONSE or ACKNOWLEDGE with NOK or other Error code

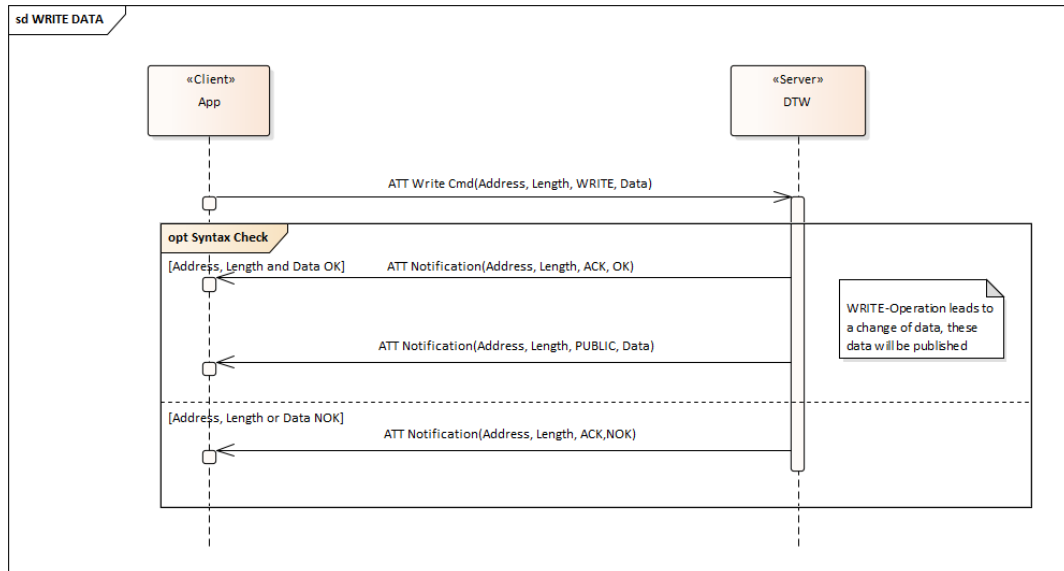


0x01	WRITE	Write/Take over Data: The server takes over the data sent by the client.	ACKNOWLEDGE
0x02	ACKNOWLEDGE	Acknowledge message, message contains ACK/NACK or other Error Codes	NO
0x03	RESPONSE	Answer to a READ request	NO
0x04	PUBLISH	Spontaneous sent after a data change	NO
0x05...0xFF	RESERVE	Reserved	

For a better understanding of the Flag-mechanism, see figures below.

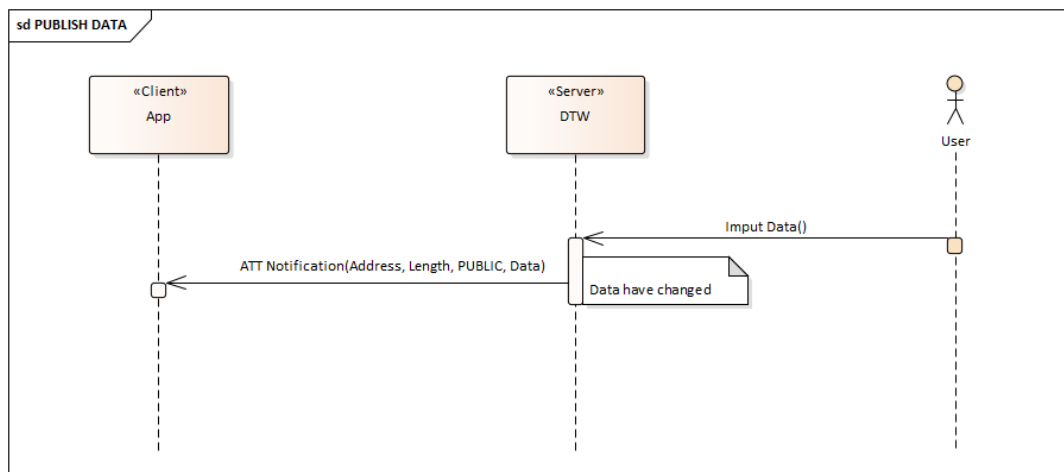


Reading of data will be done with setting the READ-Flag, the server can either return the requested data or set the Acknowledge-Flag and one data byte with the Acknowledge code if the parameters from the client (address and length) where not OK and data cannot be transferred.



By setting the WRITE-Flag the client wants the server to take over its data. If the parameters are checked successfully the server will apply the data which will be acknowledged. If acknowledged with OK, the client knows that his data has been applied. In a multiple client connection where another client also subscribed for possible changes of data a WRITE operation must lead to a PUBLISH operation once these data are different from the current content in the moment of receiving the data.

If the data will lead to an internal status change or a change of other data, these changed data sets must also be published in order to retain the consistency. In case the parameter check failed the client will get a message with an Acknowledge NOK or a dedicated error number.

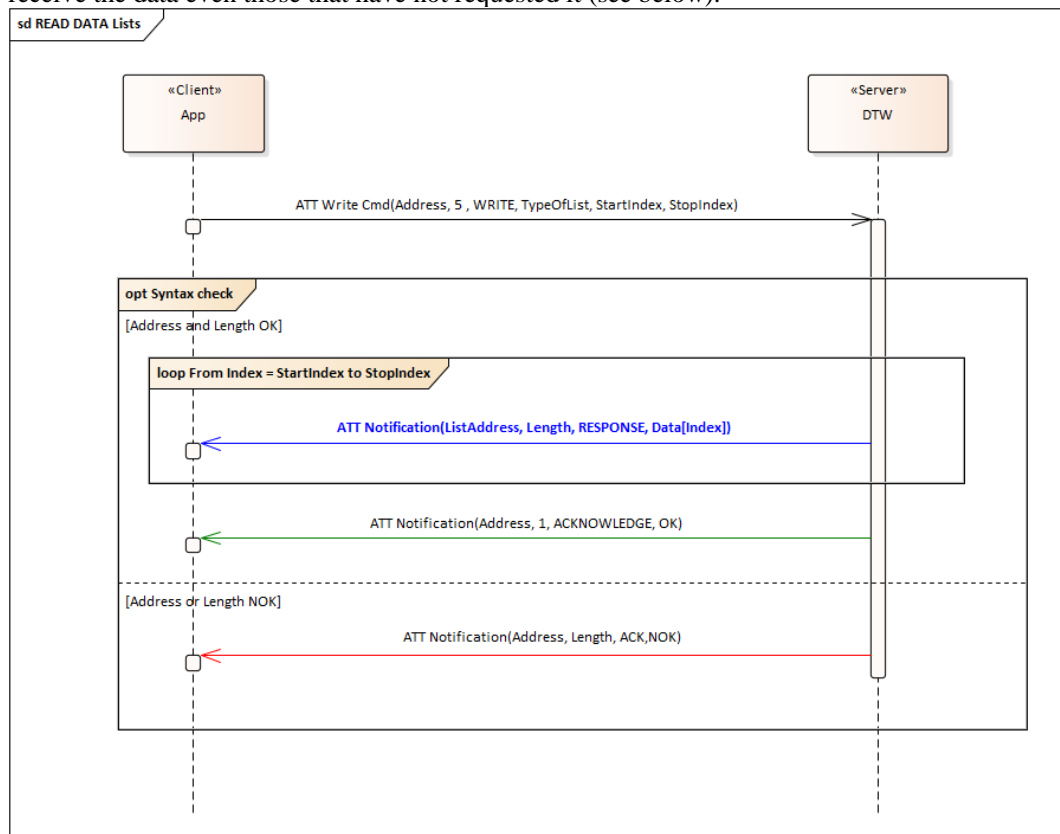


Data also will be published if the user has made changes on it, for example input a new measurement mode via buttons on the tool.

There is currently a special treatment in reading list entries. After the query for certain list entries has been received the server will send these entries one by one having the RESPONSE flag set.



In a multiple client environment, this should be avoided, since usually all connected clients receive the data even those that have not requested it (see below).



The Acknowledge to the WRITE action, that triggered the query and led to the RESPONSE of data will be sent after the data have been send. The WRITE/ACKNOWLEDGE pair effectively encapsulates the RESPONSE data sets and signals to the client that the batch of responses has finished.

Data:

In the context of the GATT characteristic the data are unspecified, they only will be specified by knowing the dedicated structure behind the ADDRESS-information of the virtual memory. How the data should be interpreted by the device is described in the device specific documentation.

In case of a message with the READ-flag set, client wants to retrieve data from the service and no data is sent. Otherwise at least one byte is necessary, for example if the message indicates an acknowledge to a writing of data, by sending the WRITE-Flag.



4.4 Virtual Address Map

The virtual address map is structured in certain data sets or blocks which represent a certain classification of the data. Every block has a certain start address, also called base address which can also be seen as an identifier for the specific data structure, block or class. Some of these blocks have a predefined base address which cannot be changed. Within the address range of the blocks sub structures can be defined, depending on the device specifics.

Address range		Description
0x00000000	0x000000FF	Standardized data block with device information. Description of that block is in the device specification. Even small devices/microcontrollers can handle this message - Includes e.g. information about device type
0x00000100	0x000FFFFF	Description of that block is in the device specification. Device specific data blocks - Containing the measurement data and all parameters for configuration
0x00100000	0x0010FFFF	Reserved – (Record Access Control Point (RACP) for LIST ACCESS) – SET
0x00110000	0x0011FFFF	Reserved – (Record Access Control Point (RACP) for LIST ACCESS) – RESPONSE
0x00120000	0x001FFFFF	
0x00200000	0x002FFFFF	Address range for device specific “special” communication capabilities. E.g. BLE-characteristics for “Screen-Mirroring”, TCP/IP, UART
0x00210000	0x002FFFFF	SIG Defined Characteristics - Keyboard (Max. buffer length 100 Bytes) - Batterie (Resolution 1%) - Bond management - Date/Time - Digital I/O (n Bit ?) ...
0x00300000	0x003FFFFF	- Reserved

For the 1st version there are fixed services and characteristics for BLE and only a subset of the described capabilities are available.

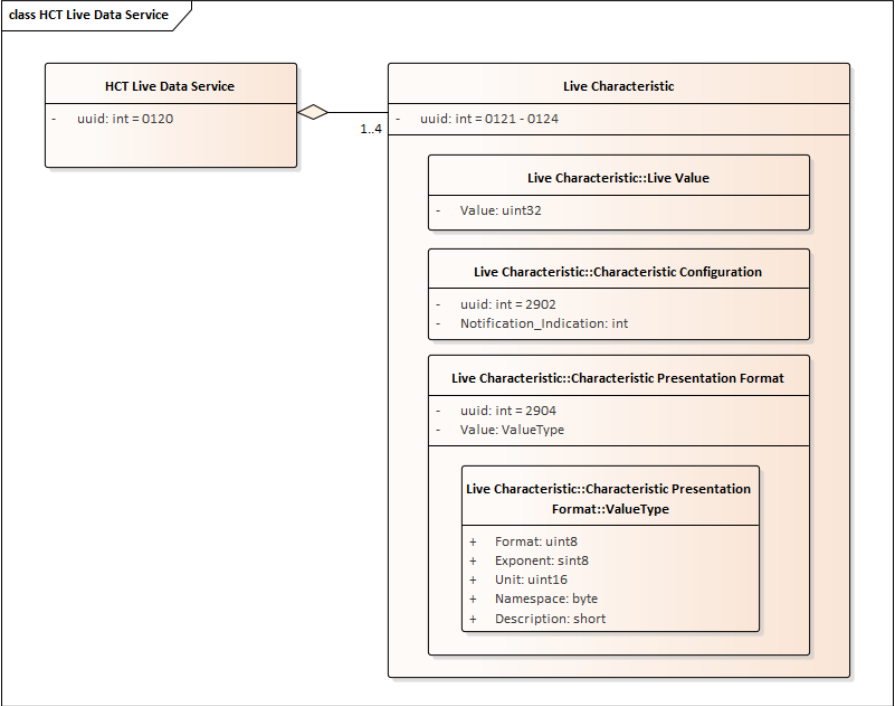
They are described in the Software_communication_protocol_definition_V1.1.1.xls

The use of a 4-byte address gives the possibility to have direct access to a data range of 4.294.967.295 bytes. This “virtual memory” space can be used for user accessible data and internals like updates and system configurations.



4.5 HCT Live Data Service

The HCT Live Data Service contains currently four characteristics which can be used for notifying live values. Clients must subscribe to the characteristics descriptor 2902. Each characteristic contains a presentation format descriptor in which the format, exponent and unit of the live value is accessible.



The value – type is currently four octets which can be used as a standard 32 bit unsigned or signed value with an exponent to represent the decimals of the value, or as a float32 value. This should be described by using the SIG specified Presentation Format.



5 Interface from Tool Application Module to HCT-BLE-Module

For BLE communication the SoC from Nordic Semiconductor is used.

5.1 Embedding of the HCT-BLE-Module in a Tool - Conceptual View

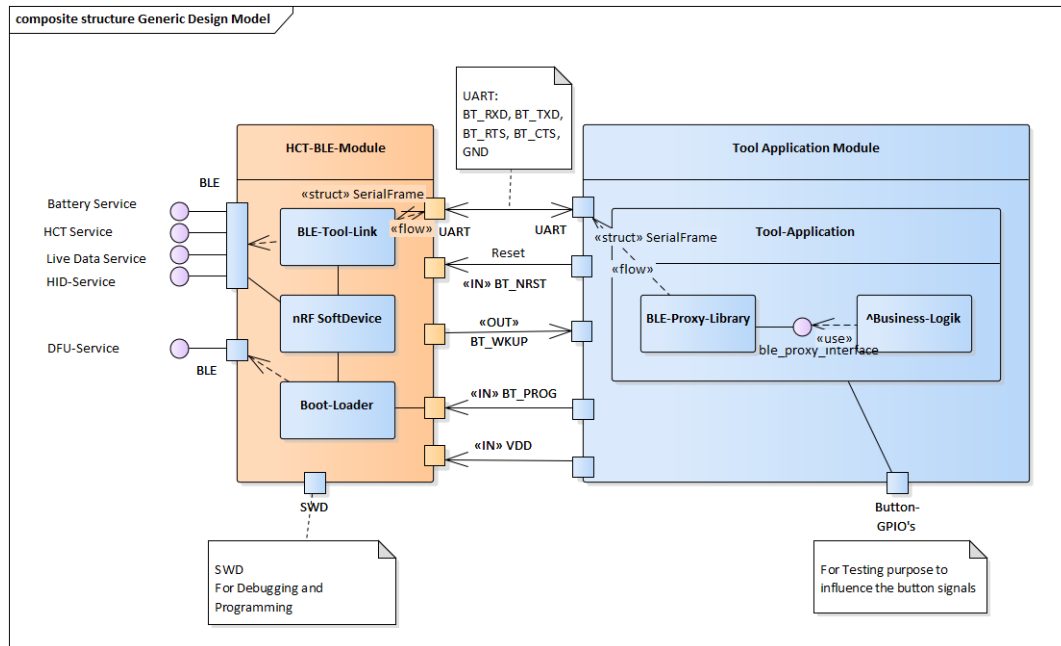


Figure 1: Conceptual View

As shown in Figure 1 the system is derived in two modules, the HCT-BLE- and the Application Module which communicate via the UART interface. Data transferred to the HCT-BLE-Module will be linked by the BLE-Tool-Link to the respective BLE-Services and characteristics.

5.2 Necessary HW Interfaces from and to the HCT-BLE-Module

5.2.1 Serial Interface

- BT_RX
- BT_TX
- BT_RTS → flow control
- BT_CTS → flow control
- GND
- VCC
- 115200 baud

5.2.2 GPIO's (From App-Module to HCT-BLE-Module)

- BT_VDD → Enable/switch on the BT-Controller
- BT_NRST → Reset the BT-Controller (low active)
- BT_PROG → Signal BT-Controller to activate DFU-Service for Download



- BT_WKUP → Signal from BT to Tool controller to wake up tool controller if a new BLE message has arrived and needs to be transferred via UART

For all lines additional scratch pads for connecting additional lines to the PC instead of the BLE-controller are necessary to test the BLE_Controller by itself and independently.

SWD-Interface with TAG-Connect 6 Pin (Pinning + Footprint will be provided)

- SWD-IO
- SWD-CLK
- Reset

5.2.3 Interface Required for Testing Purpose on the Tool-Controller

- TAG-Connect for programming the processor

Option 1:

- Serial interface for additional Test-Input in order to simulate HW-Peripherals such as Buttons, Strain-Gauge, ...

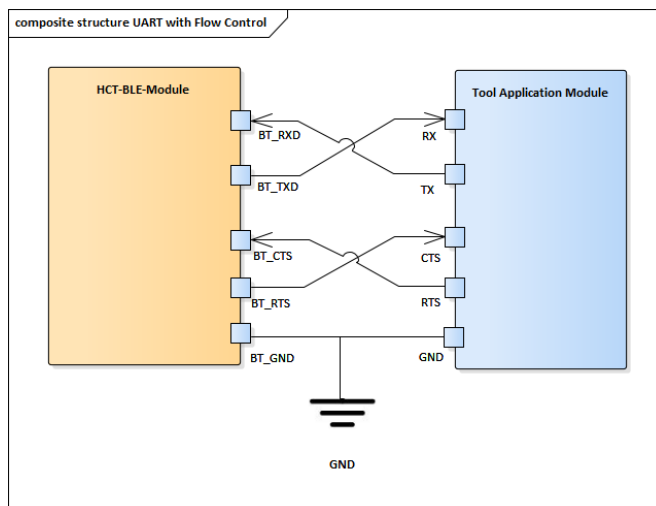
Option 2:

- Additional Scratch Pads adding the possibility for actuating the Button-Signals automatically

5.2.4 UART-HW Interface Description

Following features will be available:

- 115200 baud
- 8 Data-Bits, 1 Stop-Bit, No parity
- Hardware flow control – RTS /CTS,
- BT_CTS and BT_RTS signals are active low

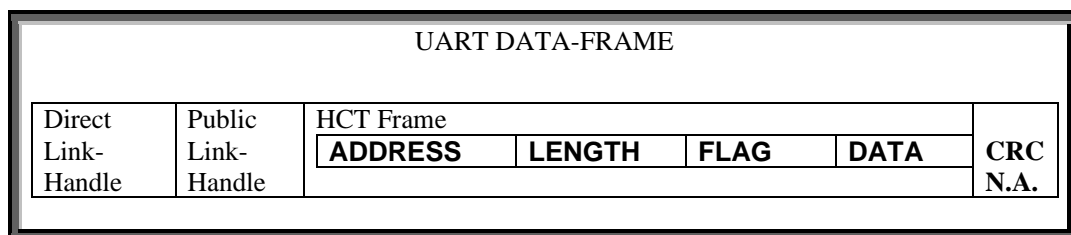




5.3 UART-SW Interface Description

5.3.1 The UART-Data Frame

The data frame of the UART Protocol is in fact a specialization of the HCT-BLE-Protocol-Frame described in 4.3. It is extended by two elements, called link handles. The meaning of the link-handles is described in a dedicated chapter.



5.3.2 Definition and Meaning of the Link-Handles

For use in the serial interface the BLE-Protocol frame is extended by two additional parameters in the front.

Internal communication between the Tool-Application and the BLE-Tool-Link is necessary, for example for setting up the BLE-connection. At the same time the BLE-Controller manages multiple BLE-connection links to several clients. Link handles represent these connections and are used to distinguish between them.

A general feature of the BLE GATT characteristic is, that all clients who have registered with a characteristic will always receive a notification message when the characteristic is changed.

The following examples illustrate the usage of link handles in typical situations in order to save communication load. The protocol is based on the publisher subscriber principle. If a client sends data to the server, the server applies these data. Then the client receives an ACKNOWLEDGE with necessary OK or NOK information. Supported by the link handles all other clients subscribed to the RX-characteristic will NOT receive the acknowledge information.

In a similar situation a client requests data by using the READ-operation. Using the link handles it is avoided that other clients receive the respective RESPONSE data without having requested it. In another use case data has changed by a WRITE-action of one certain client. This data must be published to all other subscriber clients. Using the link handles this (known) information is not send to the client that has written the data.

Direct Link-Handle:

Type	Value- Range	Description
UInt8	0	Used for direct communication between BLE-Controller and Tool-Controller
	0<n<0xFF	Any link number; added by the BLE-Controller and initiated indirectly from a BLE-Client by a READ or WRITE-Message, the handle must be repeated from Tool-Controller side, when answered with RESPONSE or ACKNOWLEDGE
	0xFF	Currently not supported, Reserve

**Public Link-Handle:**

Type	Value- Range	Description
UInt8	0	No publication
	0<n<=0xFF	Any link number; added by the BLE-Controller and initiated indirectly from a BLE-Client by a WRITE-Message if another client has subscribed for the same characteristic. The handle <u>must be repeated</u> from Tool-Controller side, when answered with RESPONSE, ACKNOWLEDE.
	0xFF	If data change is initiated by the tool-controller and PUBLISH-Message is sent in order to inform the BLE-Controller to publish it to all subscribers of the respective characteristic.

5.3.3 Optional CRC-Check

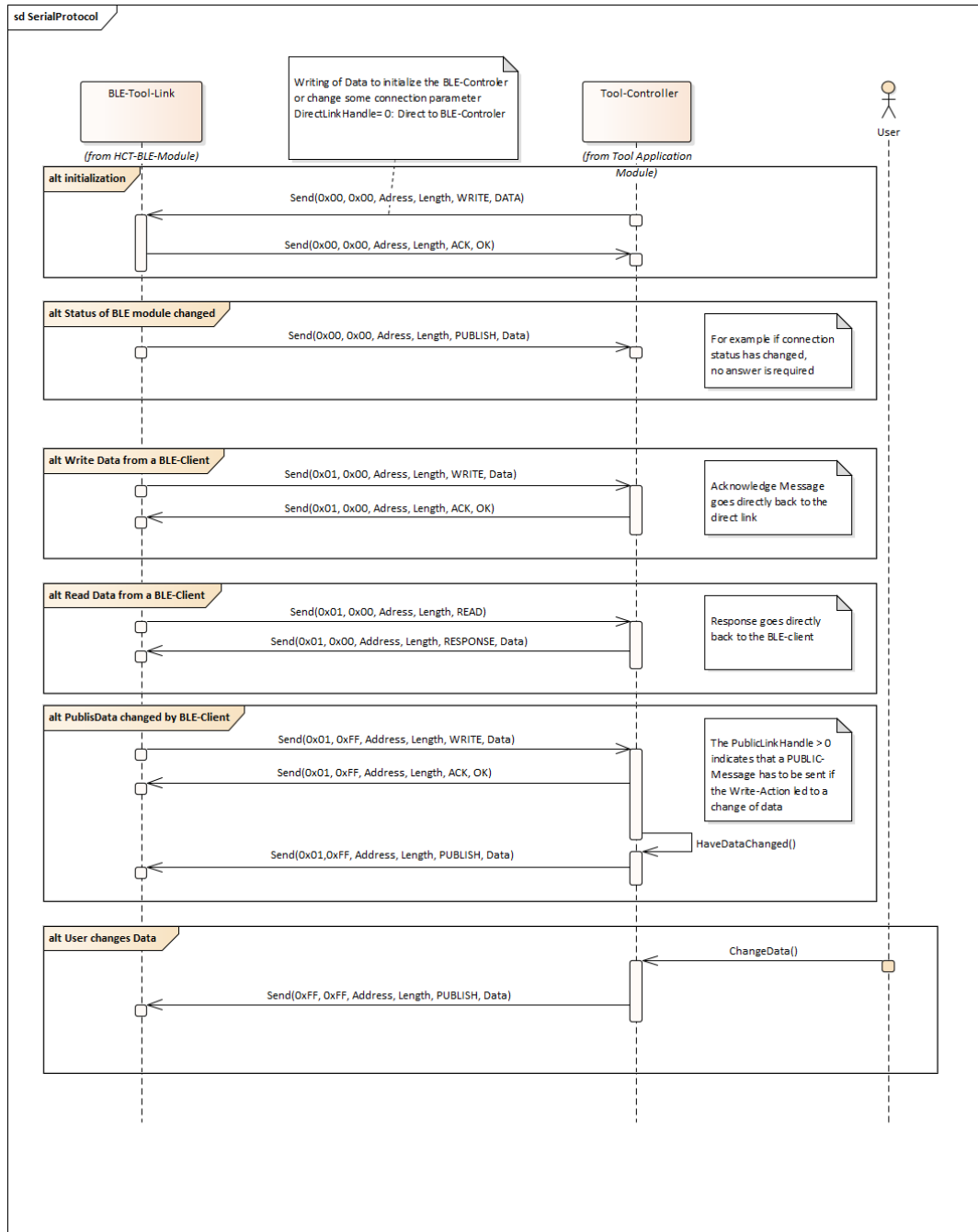
To harden the communication against distortions like electric and/or magnetic fields especially at higher transmission rates a checksum can be used optionally.

UART doesn't provide an inherent error correction neither a checksum to correct or detect errors during the transmission. The BLE module therefore has a configurable checksum mechanism to include a standardized checksum value at the end of the protocol frame in a UART transmission. The checksum is placed directly after the last byte of the transmission payload.

CRC will be configurable, in the current released version (1.0.0) it is not supported.



5.3.4 Handling of Link-Handles





6 Use Cases

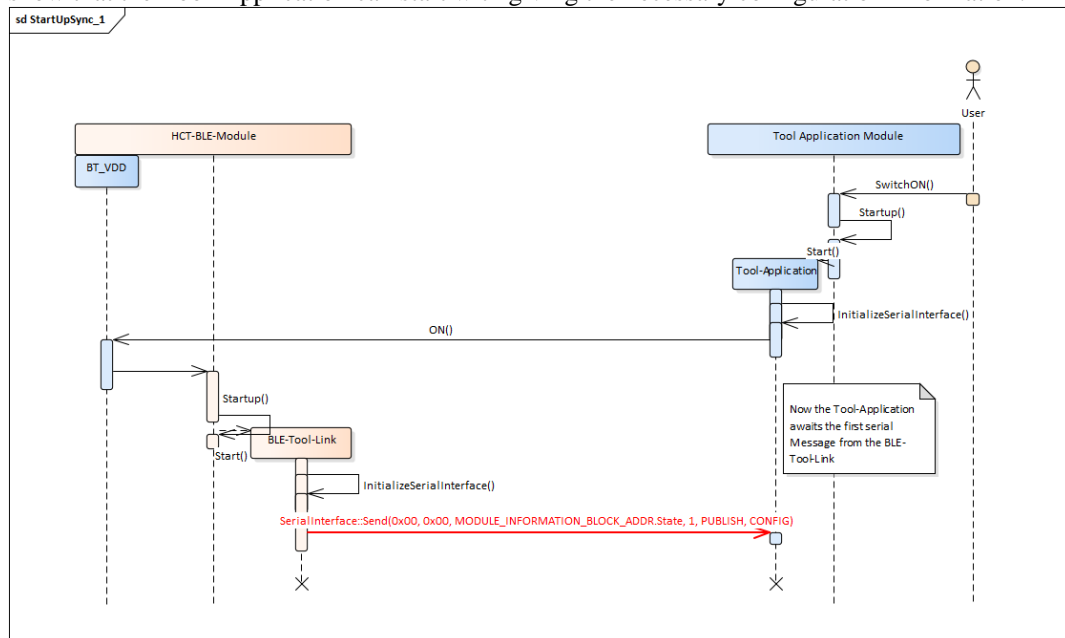
In the following chapter some use cases shall be described by the help of sequence diagrams. These sequence diagrams visualize a possible interaction between the Tool-Application Module and the HCT-BLE-Module which are not fixed yet and shall be discussed in order to find the best solution.

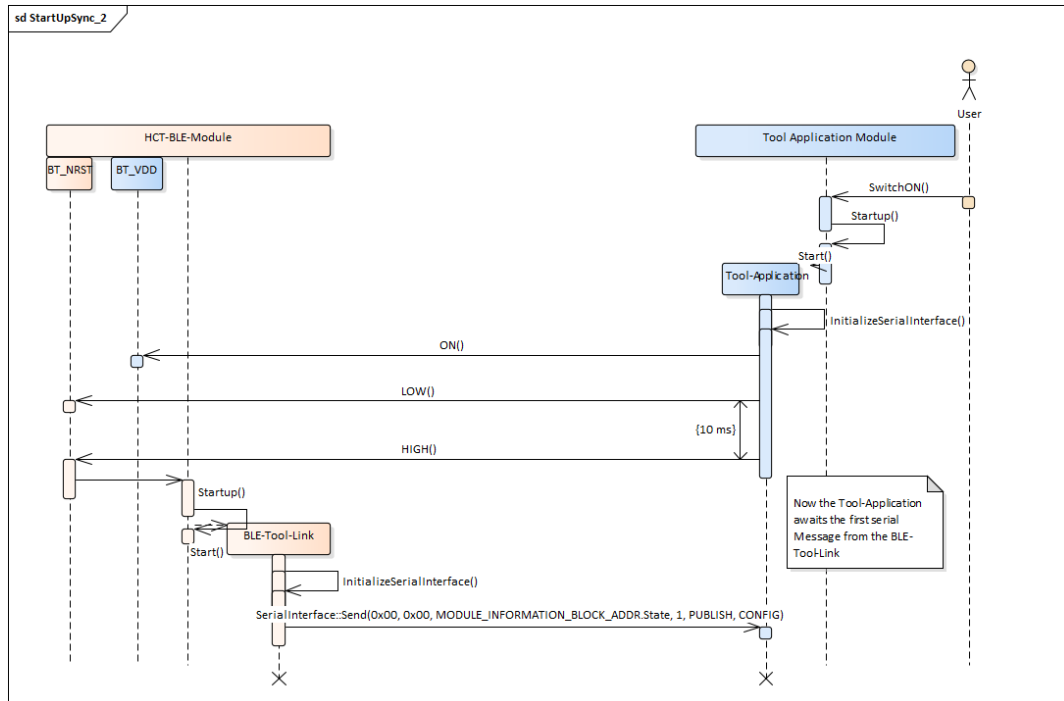
6.1 Startup-Synchronization between HCT-BLE-Module and Tool-Application Module

After startup both controllers must initialize their UART interface in order to start communication.

One possible way is, that after its own startup and initialization of the UART-Interface the Tool-Controller starts the BLE-Controller. Then it waits for the initialization of the BLE-Controller which sends its module status which signals, that it must be configured now.

After the BLE-Tool-Link has initialized the UART, it will send the state CONFIG in order to show that the Tool-Application can start with giving the necessary configuration information.





The second option is to use the BT_NRST additionally, see figure above. Since the BT_NRST is a software configurable PIN it must be configured in the Bootloader. That means, that the Bootloader must be able to start and configure the Pin in order to be recognized as a reset-pin.

6.2 Initialization of HCT-BLE-Module

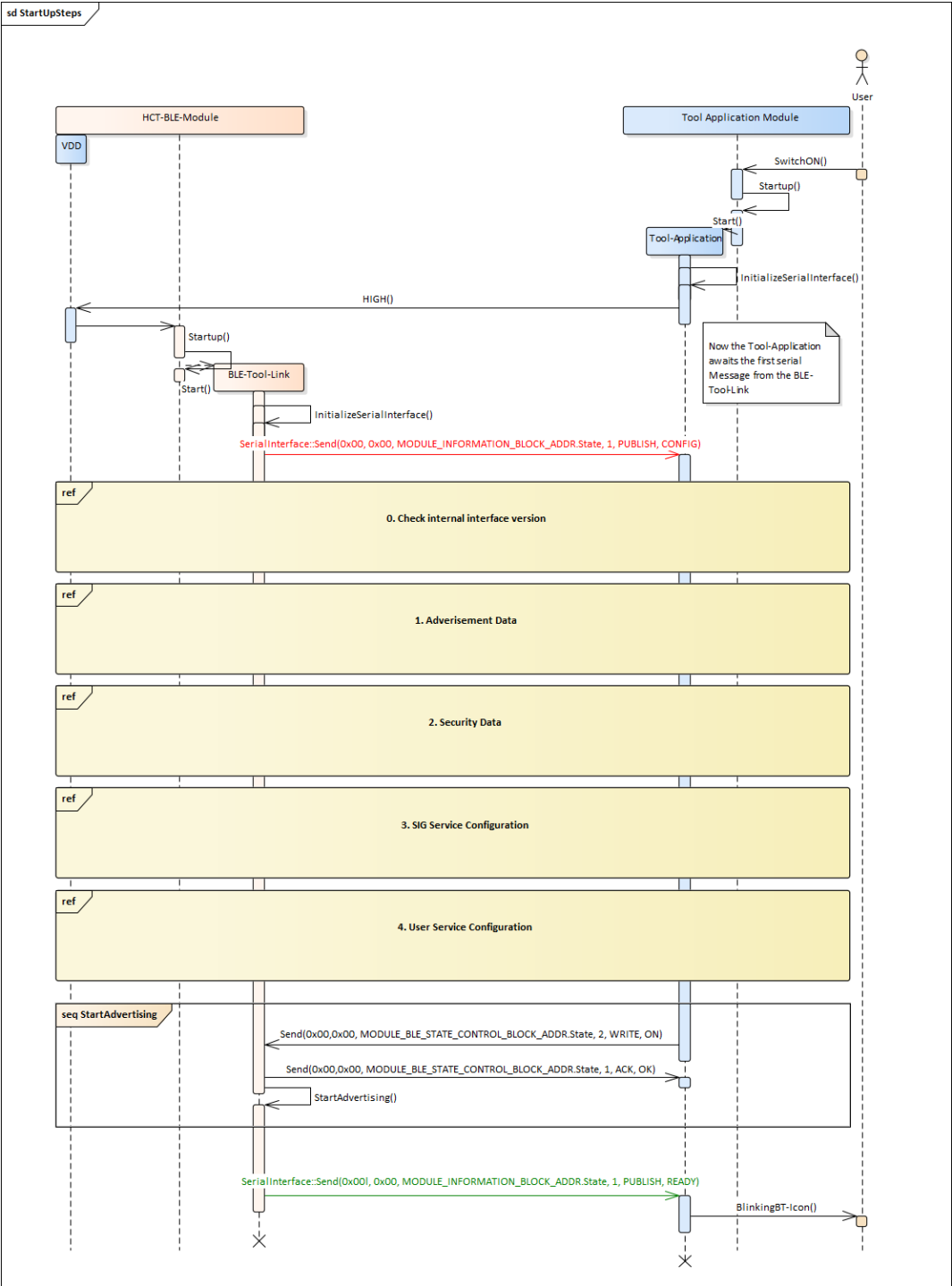
The tool-application starts the BLE-specific configuration of the HCT-BLE-Module.

The compilation of the necessary data is supported by our BLE-Proxy-Library, see [2]. It contains the necessary data and only exposes data to be changed, i.e. tool specific data such as the tool name in the advertisement.

For a better understanding, the configuration is divided in five parts:

- Advertisement Data,
- Security Data,
- Configuration of predefined SIG-Services,
- Configuration of proprietary, user defined services.

After having send this information the Tools SW will signalize the BLE-Controller that it can start advertising.

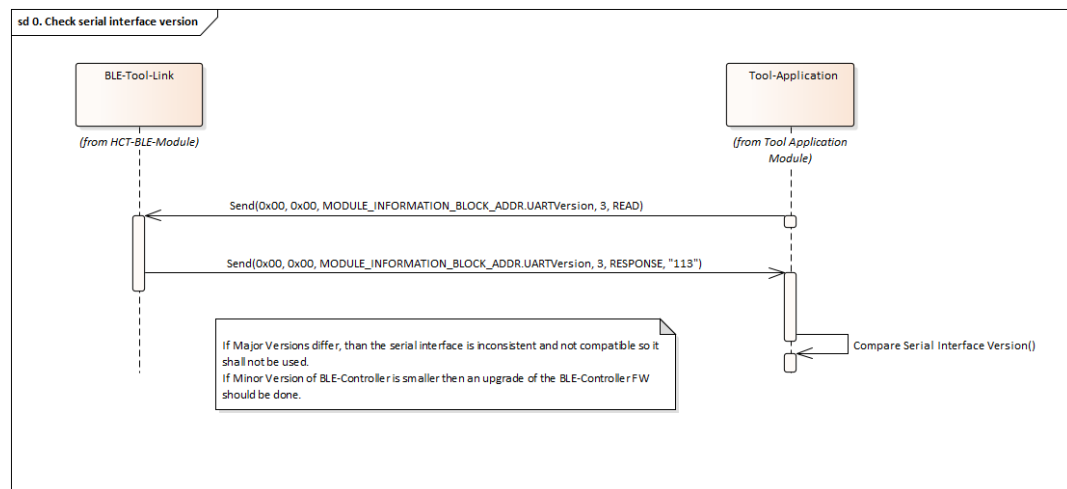




In the following section the Start-Up steps will be described in detail:

6.2.1 Check Serial Interface Version

In order to be able to exchange the internal data for setting up the BLE-Controller, the consistency of the data-structure interface must be ensured. Therefore the Application controller must query first the serial protocol version.



The version is divided into <Major>.<Minor>.<Step>., see Software_communication_protocol_definition_V1.1.1.

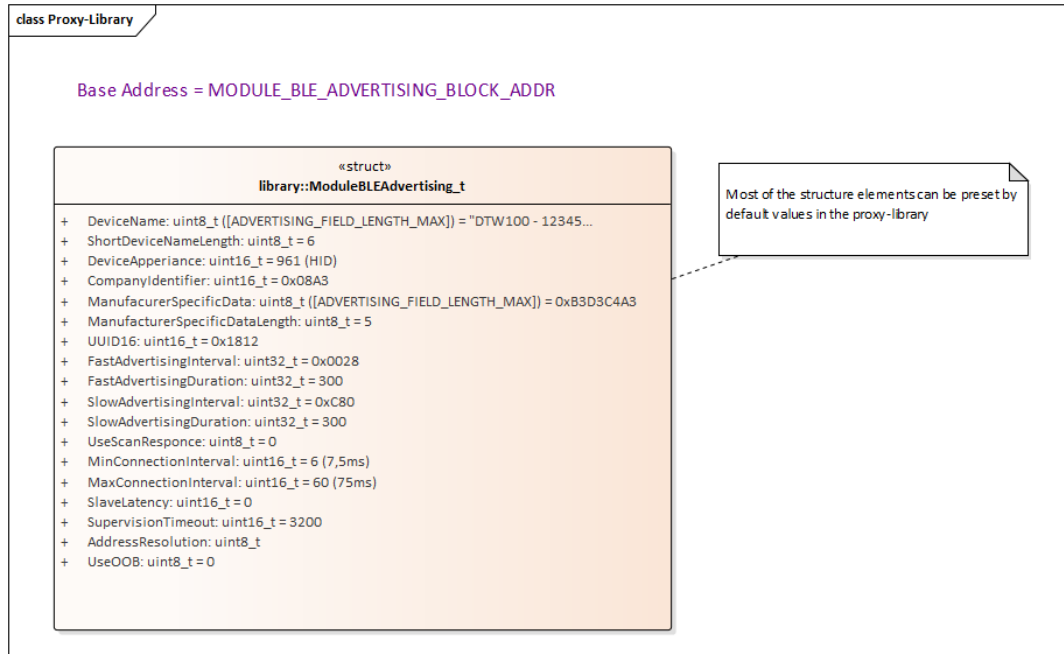
The Major-Version Number of the Interface should be the same between BLE-Tool-Link and Tool Application, as a difference indicates a not backward-compatible change between the structures. The Minor-Version of the BLE-Tool-Link should not be smaller than the Minor-Version of the Library in the Tool-controller. If the Minor-Version of the BLE-Tool-Link is higher it indicates a backward-compatible change that does not require an Upgrade of the FW, see also Table.

BLE-Tool-Link Serial Version		Tool- Application:: Proxy-Library- Version	Action required.
Major	>	Major	Serial Interface cannot be used due to inconsistency. Application Controller FW upgrade with newer Proxy-Library required or downgrade of the BLE-Controller FW-Version to the same Major (by using BT_PROG and BT_NRST Pin)
Major	<	Major	Serial Interface cannot be used due to inconsistency. BLE-Controller FW-Version must be upgraded by using BT_PROG and BT_NRST Pin
Minor	>	Minor	Backward compatible change/extension, upgrade of the Tool-Controller FW possible but not necessary



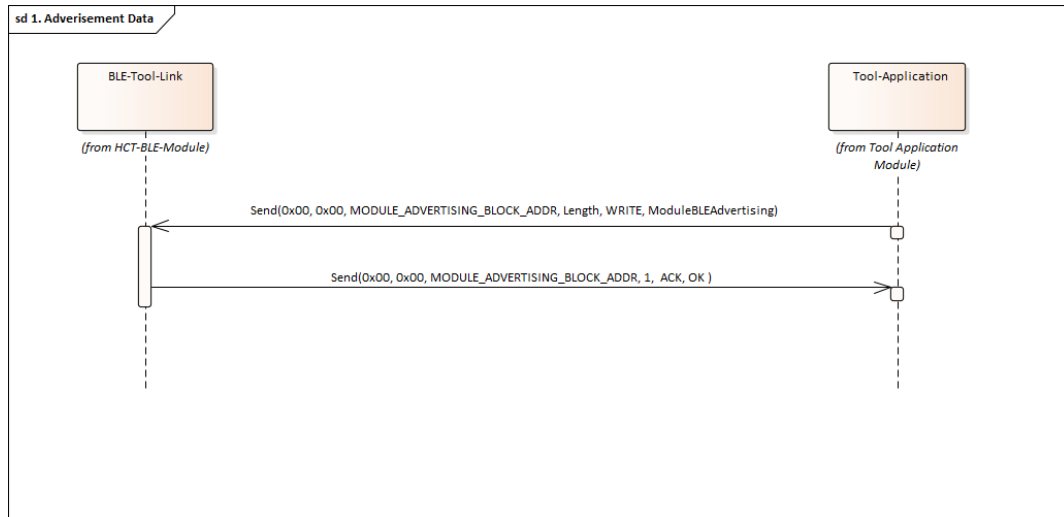
Minor	<	Minor	Serial interface can be used, but new extended functionality is not supported. Upgrade of BLE-Controller FW needed if extended functionality shall be used. BLE-Download can be initiated by the serial interface.
Step	<>	Step	Serial interface can be used. Changes which do not affect the structure and the compatibility. Will be used to trace some steps during the development or smaller bug fixes.

6.2.2 Advertisement Data



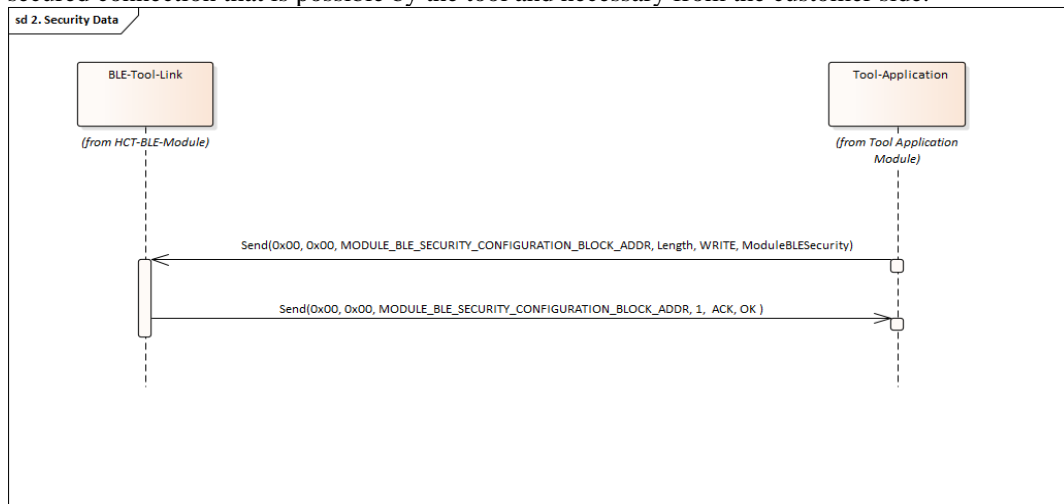
Advertisement configuration data are hosted in the virtual Hoffmann-Protocol address-space with its base address, see figure above or [1]. The structure can and will be preset with default values by the BLE-Proxy-Library, however some values need to be filled explicitly, for example the DeviceName.

The Tool-Application then must send the data via the serial protocol to the BLE-Tool-Link and will get an Acknowledge with OK, if data was taken over.



6.2.3 Security Data

The current structure of the security data configuration block at address `MODULE_BLE_SECURITY_CONFIGURATION_BLOCK_ADDR` is very open and offers a variety of possibilities to configure security options, however only some presets are helpful. Here the intention is, to collect parameters, for example the type of address resolution (random, resolvable, private,...) or which authentication method must be chosen. The principle is the same as with the advertisement configuration data – the structure will be preset by the proxy-library and can be changed for a few dedicated parameters, depending on the type of authentication and secured connection that is possible by the tool and necessary from the customer side.



6.2.4 SIG Service Configuration

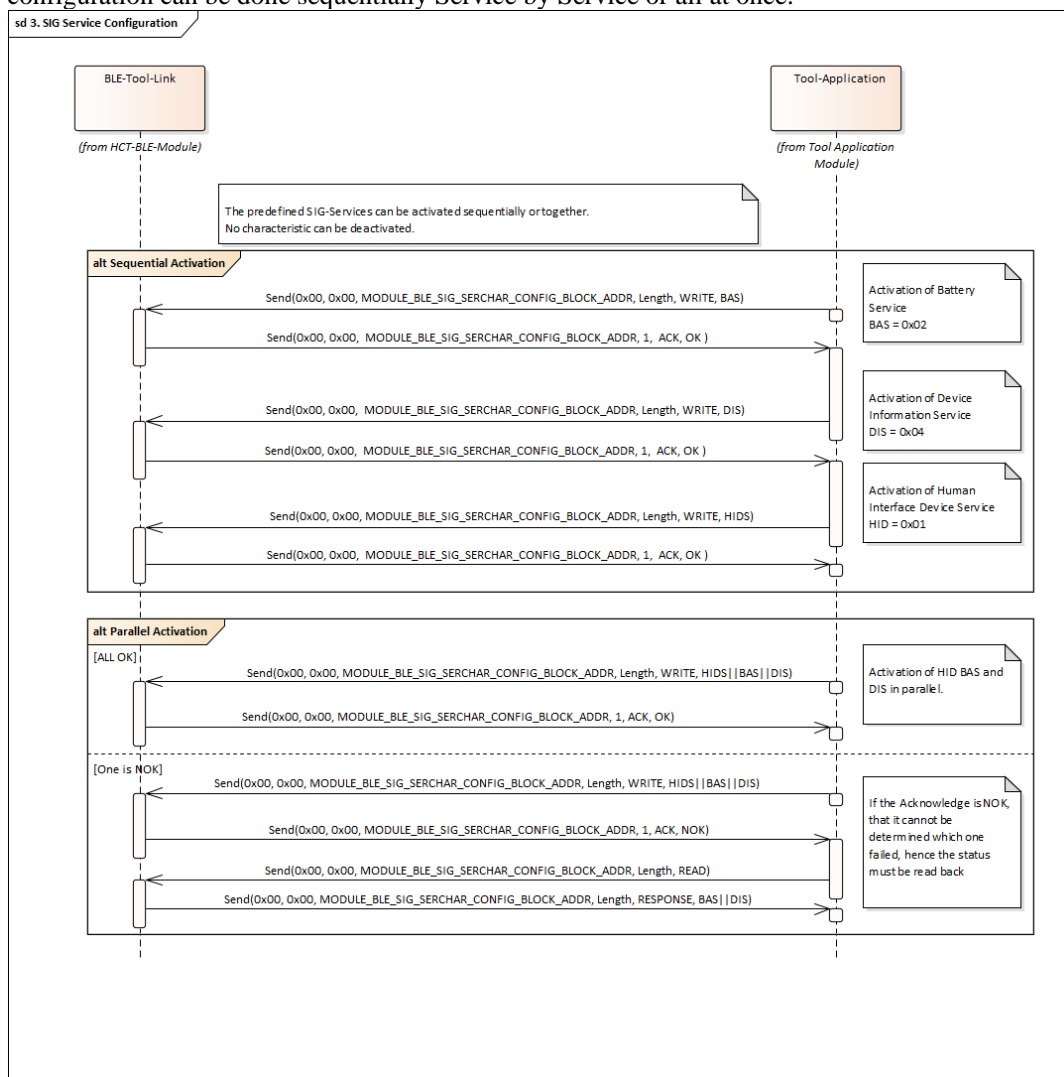
The Sig Service Configuration block can be found at `MODULE_BLE_SIG_SERCHAR_CONFIG_BLOCK_ADDR`.

SIG defined Services are preset by the BLE-Controller and don't need additional parameters. For the tools, the Battery-Service (BAS), Device Information Service (DIS) and Human Interface Device Service (HIDS) will be available and can be activated by the configuration.



The control-parameter SerCharConfiguration_1_32 is a bitfield where each SIG-Service will be represented with a dedicated DEFINE. By setting the configuration and sending it to the BLE_Tool-Link it will setup the activated Services in the BLE-Stack.

Important to know is, that a Service can only be added to but not removed from the BLE-Stack. If for example the HID is not yet activated on the DTW it can startup and connect to a mobile application. If then later the HID is needed, it can be activated by sending the SIG Service Configuration block again with only the activated HID-Bit. If a service is supposed to be deactivated, that only can be done by performing shut down of all connections and a reset of the BLE-Controller and a restart of the configuration steps. As shown in the figure below, the configuration can be done sequentially Service by Service or all at once.





6.2.5 User Service Configuration

In the User Service Configuration the proprietary services and their characteristics will be activated. For each characteristic some more data has to be sent to the BLE-Tool-Link.

Note: Most of the data is already filled with default values by the BLE-Proxy-Library.

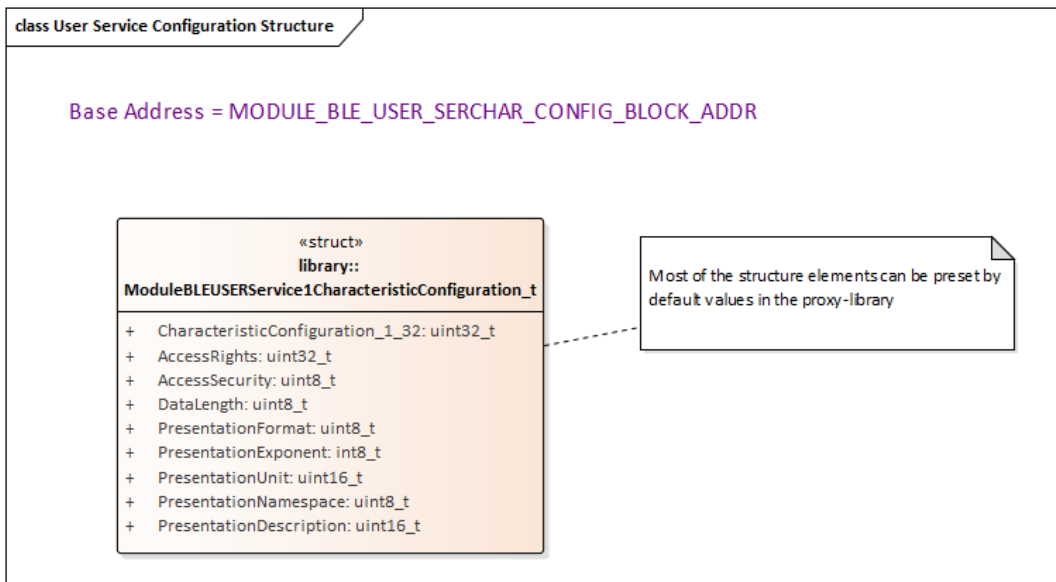
Every added characteristic included in a certain service will be represented in the bitfield CharacteristicConfiguration_1_32.

([UserServiceX]-> CharacteristicConfiguration_1_32)

The current proprietary services and characteristics are:

Characteristic description	included in (service)	Currently used for	Value of the bit in bitfield in HEX
Protocol TX (BLE Central Out -> peripheral In)	HOS	Communication data/command	1
Protocol RX (BLE Central In -> Peripheral Out)	HOS	Communication data/command	2
Measurement Characteristic 1	LDS	Actual Torque value live view	1
Measurement Characteristic 2	LDS	Peak Torque value live view	2
Measurement Characteristic 3	LDS	Actual Angle value live view	4
Measurement Characteristic 4	LDS	Peak Angle value live view	8

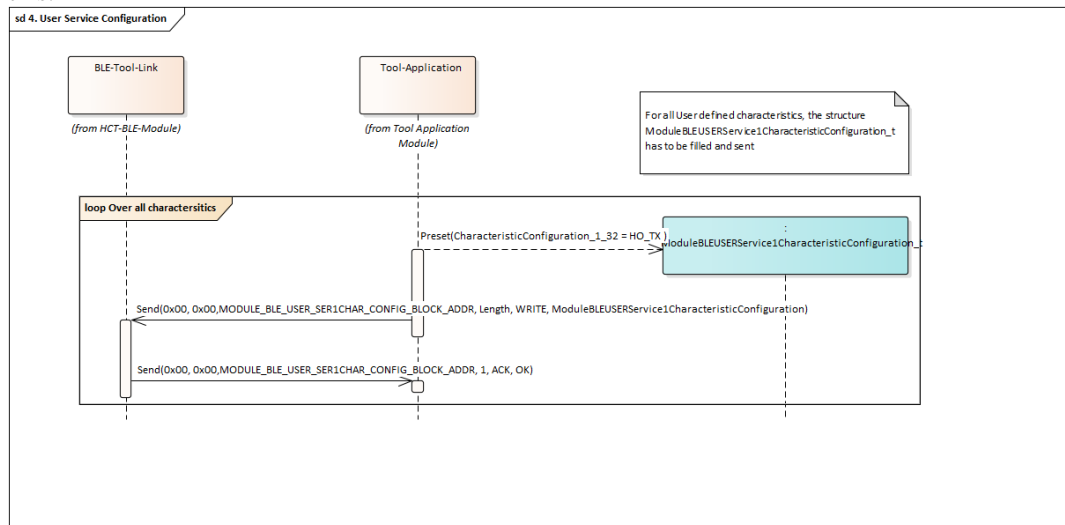
For each characteristic, the access right as well as the presentation format can be specified.





6.2.5.1 Dynamic Length Configuration of User Service Characteristics

Especially in regard to an optimized sending of live values via the serial interface there is a possibility to define a dedicated length and a virtual start address of these live values in the HCT protocol address space (`VirtualStartAddress`) for each user defined characteristic configuration. These values can now directly be placed sequentially in the virtual address map of the protocol. So it is possible to send for example four values in one serial telegramm. In the dynamic configuration each subsequent characteristic value's virtual start adress must have an offset to the predecessor by exactly the length in bytes of the predecessor. The BLE-Tool link will not verify this.



As shown in the figure above the configuration of the characteristics must be done sequentially, if the parameter set differs from the other characteristics. For the live characteristics torque or angle one set can be sent by bitwise OR-junction in the *CharacteristicConfiguration_1_32*. In the BLE-Proxy-Library, the data will be preset by default values.



6.3 Firmware Update Initialization

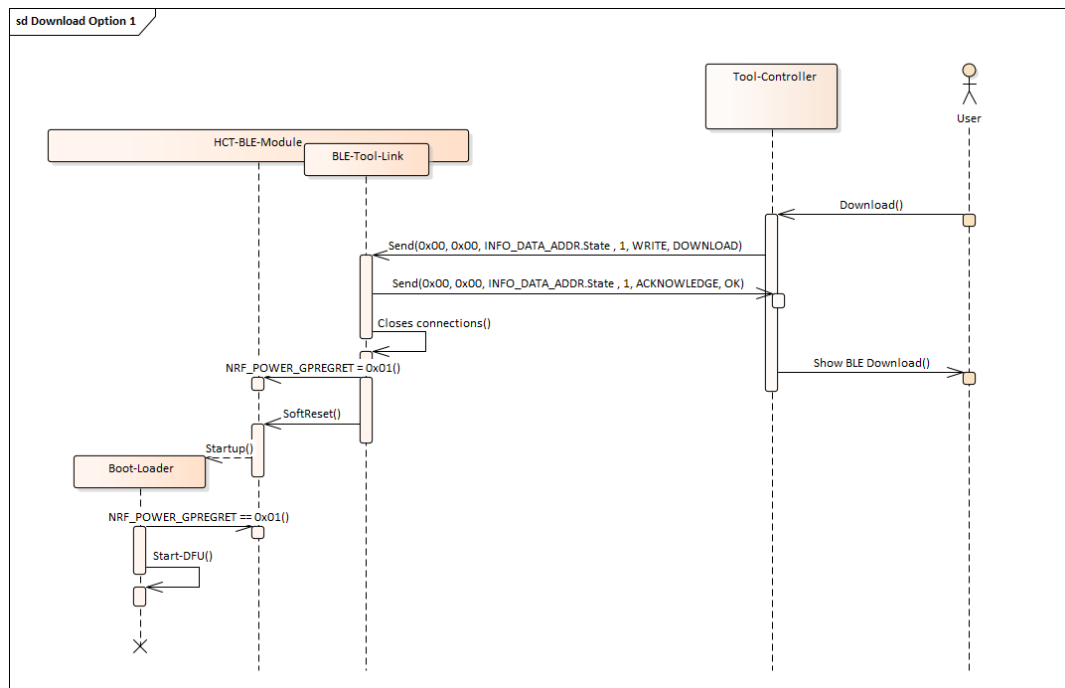
The firmware update initialization can and must be done in one of two ways.

6.3.1 Method 1: Triggered by User via Serial Protocol

The first method can be used if the BLE-Tool-Link is already started up and working. The download can also be initiated by a client connected to the Hoffmann-Service.

If the BLE-Tool-Link is connected to the Tool-Controller's software, the user can trigger or initiate the BLE-Download via a button, an external tool or a certain menu point. The tool-SW then must send the described command by setting or requesting the DOWNLOAD – State. When the Tool-Controller gets the respective acknowledge, it can show the user by means of display or LED-sign that the BLE-download has been initiated and that the user now can connect to the advertised download-service, for example DFU-Service.

The figure shown visualizes the necessary sequence for initiating this process.

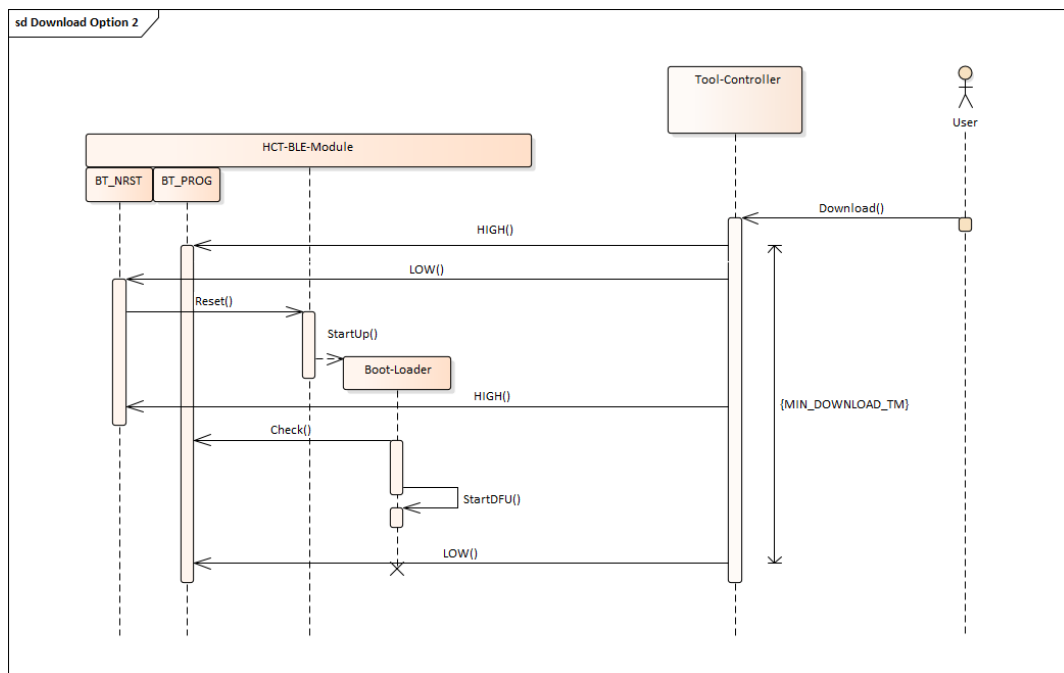




6.3.2 Method 2: Triggered by the User if no Application or a not Working or Corrupt Application is Loaded

In this case the application software on the HCT-BLE-Module might not be working. The reason could be that the application is corrupt and will crash after start, the bootloader cannot recognize this, or the trigger functionality of the serial interface is faulty, so that the application will not recognize the serial or BLE-trigger.

In such case the Tool-Controller must set a dedicated (BT_PROG)-Input and initiate a reset by switching the BT_NRST to LOW. In consequence the HCT-BLE-Module will start up and the started Boot-Loader will check the BT_PROG-Input. The initiating controller must hold this BT_PROG-Input to high for a minimum time-frame, here called MIN_Download_TM.

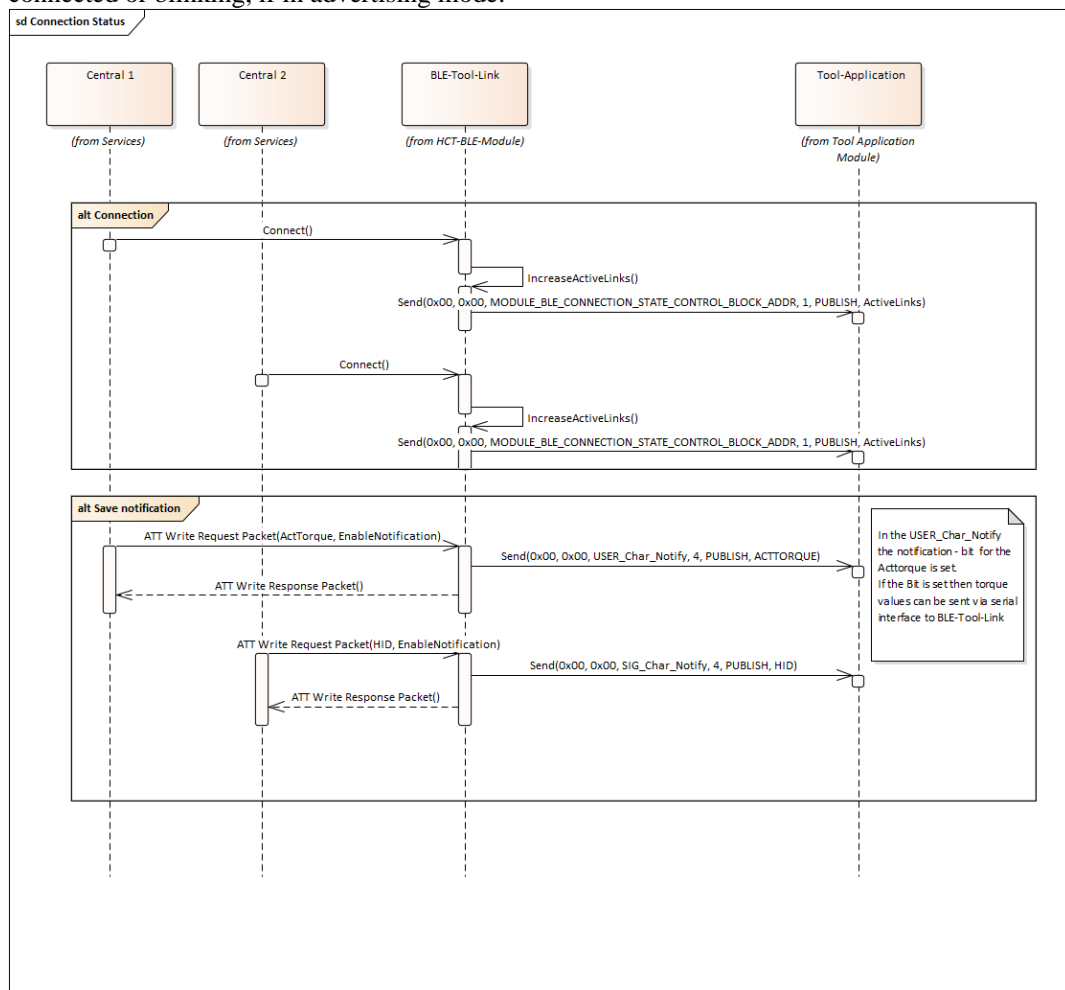




6.4 Connection Status

The connection status of the BLE-Tool-Link is represented in the ModuleBLEConnectionStateControl-Block at address `MODULE_BLE_CONNECTION_STATE_BLOCK_ADDR` of the virtual address space of the SW_Communication_Protocol.

The parameter `ActiveLinks` will be increased once a client connects to the device and will be decreased once a client disconnects. If a client subscribes to a configured SIG or User-defined characteristic for data notification the respective bit in the parameter `SIG_Char_Notify`- or `User_Char_Notify` parameter is set. Every change of it is published via the serial interface to the Tool-Application. The state is held consistently in the proxy-library. It is needed from the Tool-Application to show the connection state, for example the solid Bluetooth-Symbol when connected or blinking, if in advertising mode.



6.5 Live Data

Live data is sent by using the same protocol frame as the other data-commands.

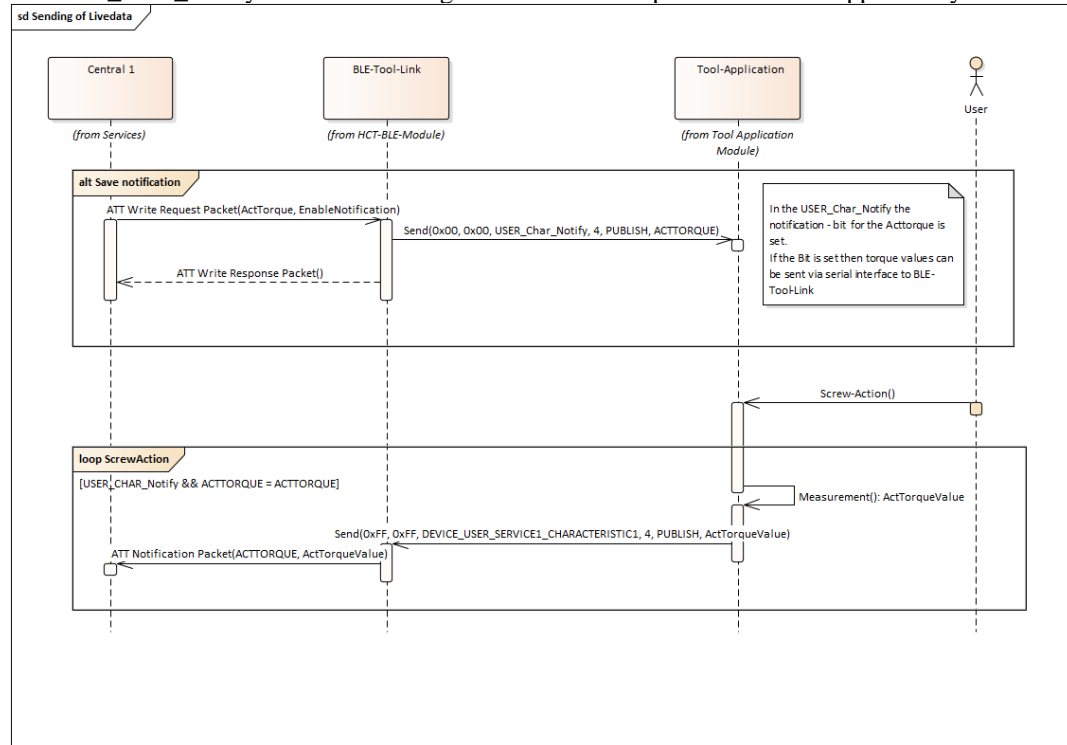
For each characteristic the Client must subscribe to the notification using the Client Characteristic Configuration descriptor. BLE-Tool-Link sets the respective information in the



USER_Char_Notify-Bitfield of the ModuleBLEConnectionState-block. Indication is currently not supported but also not needed.

Goal is to prevent sending of Live-Values via the serial interface, if no client has subscribed to the respective characteristics.

If the user starts a measurement on the tool, the Live-Values will be sent depending on the set bit in USER_Char_Notify-Bitfield. Sending of live values is optional and not supported by all tools.

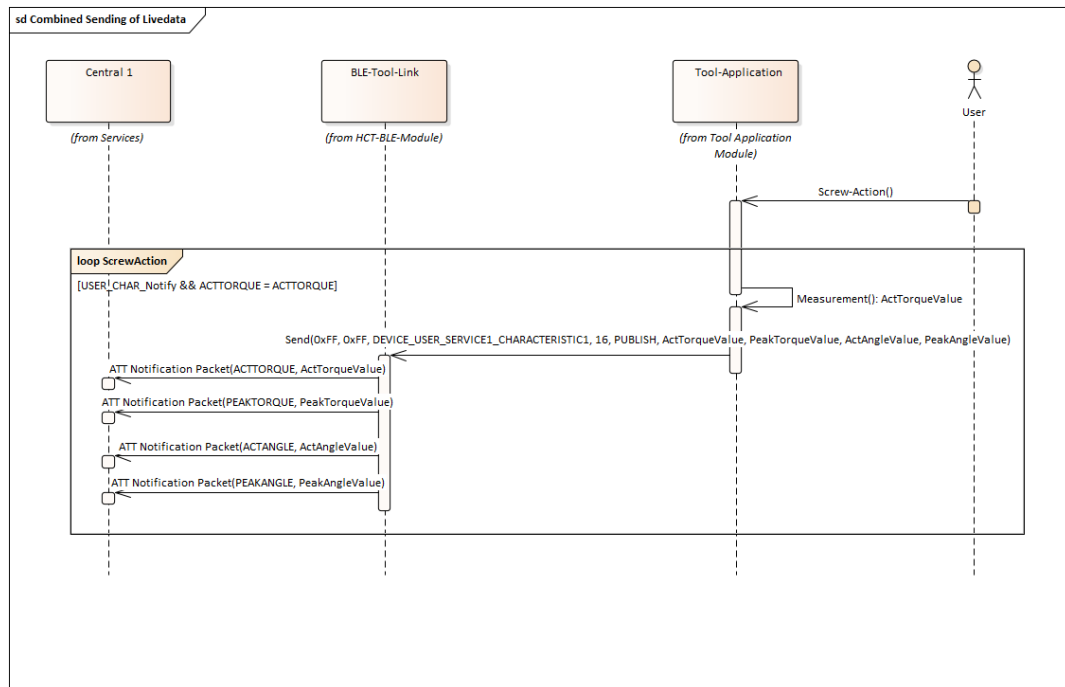


6.5.1 Combination of Sending of Live Values

In order to save overhead and to ensure a deterministic (equidistant time triggered) sending of for example up to four Torque- and Angle- live values in one serial message, the respective User-Characteristics must be configured as described in 6.2.5.1. If this is ensured, then the address of the first of the live values in the virtual address map must be used as the address in the serial protocol frame. Please note that the BLE-Tool-Link does the mapping of the subsequent values according to their virtual address and length information.

It sends all characteristic values when

- Subsequently received with the serial message from the Application Module and
- a subscription for the characteristic value is active.



6.6 Additional Use Cases that Will be Defined by Hoffmann

- Trigger Download via Hoffmann-Protocol
- HID Switch ON/OFF
- Switch OFF or reset BLE-Controller

7 Proxy-Library for Easier Integration

To make it easy for our suppliers to integrate BLE into their products, we offer an easy-to-integrate proxy library called BLE-Proxy-Library.

First, a fixed module configuration is used, later a dynamic configuration and a more general approach will be implemented.



8 Documents

1.	Address-Map for protocol description: Software_communication_protocol_definition_V2.3.0.xlsx
2.	BLE-Proxy-Library_description_Vx.x.x.docx
3.	BT advertising name concept of HCT tools.xls
4.	https://www.bluetooth.com/wp-content/uploads/Sitecore-Media-Library/Gatt/Xml/Descriptors/org.bluetooth.descriptor.gatt.characteristic_presentation_format.xml

Literatur

- Bluetooth Wireless Technology*. (2023). Verfügbar 10. August 2023 unter <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/>
- Sauter, M. (2022). *Grundkurs Mobile Kommunikationssysteme*. Springer Vieweg.
- Bluetooth Core Specification*. (2023, 31. Januar). Version 5.4.
- Carles Gomez, J. P., Joaquim Oller. (2012). *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology*. Verfügbar 10. August 2023 unter <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3478807/>
- HCT Windows App Nutzeranleitung*. (2022). Verfügbar 21. August 2023 unter https://www.hoffmann-group.com/media/media/de/downloads_1/manuals_1/20221109_HCT_Windows_App_Nutzeranleitung_rel_V0011.pdf
- nRF5 SDK v17.0.2 USB Examples*. (2020). Verfügbar 16. August 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/examples_usb.html
- Nordic Devzone Forum Case ID 241111*. (2019). Verfügbar 17. August 2023 unter <https://devzone.nordicsemi.com/f/nordic-q-a/55572/usb-msc-using-internal-flash>
- nRF5 SDK v17.0.2 Block device RAM*. (2020). Verfügbar 23. August 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/group__nrf__block__dev__ram.html
- nRF5 SDK v17.0.2 USB HID keyboard*. (2020). Verfügbar 23. August 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/group__app__usbd__hid__kbd.html
- Device Class Definition for Human Interface Devices*. (2001). Verfügbar 20. August 2023 unter https://usb.org/sites/default/files/hid1_11.pdf
- nRF5 SDK v17.0.2 Peer Manager*. (2020). Verfügbar 27. Juli 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/group__peer__manager.html
- nRF5 SDK v17.0.2 Block device*. (2020). Verfügbar 23. August 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/group__nrf__block__dev.html
- The FAT filesystem*. (2002). Verfügbar 28. August 2023 unter <https://www.win.tue.nl/~aeb/linux/fs/fat/fat-1.html>
- nRF5 SDK v17.0.2 Flash Storage*. (2020). Verfügbar 24. August 2023 unter https://infocenter.nordicsemi.com/topic/sdk_nrf5_v17.0.2/lib_fstorage.html

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die Bachelorarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, den 28. August 2023