Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Bachelorarbeit zum Thema:

Implementierung eines drahtlosen Fußschalters

Zur Erlangung des akademischen Grades Bachelor of Science

Vorgelegt von: Wolfram Barth

Matrikelnummer: 03708119

Studiengang: Informatik

Betreuer: Prof. Dr. Stefan Wallentowitz

Abgabedatum: 8. August 2023

Abstract

Das Ziel der vorliegenden Arbei ist es zu zeigen welche Implementierungsschritte durchgeführt werden müssen, um die Softwareanwendung für einen drahtlosen Fußschalter zu schaffen. Dazu wird die folgende Forschungsfrage gestellt: Wie muss eine bestehende bestehende Anwendung erweitert werden um

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

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

1 Einleitung

In diesem Kapitel wird zunächst vorgestellt, mit welchen Geräten der Fußschalter interagiert und die zusammen die HCT-Plattform bilden. Es wird gezeigt, wie sich der Fußschalter in diese Plattform einfügt, sowie Motivation der Implementierung erklärt.

1.1 HCT-Plattform

Die Digitalisierung von Werkzeug 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 mit 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.
Dabei wurde sich für Bluetooth low energy (BLE) enschieden, da den Hand- und Messwerkzeugen, aufgrund ihrer handlichkeit und nicht kabelgebundenheit, nur begrenzte Batteriespeicherkapazitäten zu Verfügung stehen. BLE ist eine Abwandlung des klassischen Bluetooth und ist besonders auf Energiesparkeit hin optimiert, wodurch es auch auf dem Hand- und Messwerkzeug lange Batterielaufzeiten garantieren kann.
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

Sie stellen dem Anwender die Daten der Messungen in verschiedener Weise zu Verfügung. Zum Einen können die Geräte als ein Human interface device (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. Die letzte 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 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.

1.2 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, dass entwickelt wurde um über BLE gesprochen zu werden und stellt ein virtuelles Speichermodel der Geräte da. Dabei besitzten Werkzeuge unterschiedlicher Produktreihen und Hersteller jeweils verschiedene Speichermodelle. Über “read” und “write” Befehle auf die Speicheradressen kann dann der interne Zustand des Werkzeug abgefragt und verändert werden. So können auch komplexe Operationen effizient über BLE durchgeführt werden. Zudem stehen automatisierte Tools zur Verfügung mit deren Hilfe die Frameworks, die das Sprechen des HCT-Protokoll abstrahieren, um die Speichermodelle neu eingeführter Werkzeuge erweitert werden kann.

Protocol version 2.4.12 for Holex_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) Holex basic (2) Garant plus (3) Holex plus (4) Hazett (5) Facom (6)	HolexBasic	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 featurs are supportet (nessesary 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 1: Auszug aus HCT-Protokoll für Messuhren und Messschieber

1.3 Motivation

Die Digitalisierung von Werkzeug führt dazu, dass eine wachsende Anzahl von Geräten im Arbeitsumfeld der Fertigung ihre Messdaten zur Verarbeitung durch einen Computer zur Verfügung stellen. Dabei treten bei der Zusammenführung dieser Daten eine Reihe an praktischen Problemen auf. 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.
Es hat sich gezeigt, dass die Einführung der HCT-Windows-App immer wieder auf großen Widerstand von IT-Abteilungen trifft. Diese haben Sicherheitsbedenken, da die App direkt in der Fertigung eingesetzt wird und es müssen daher langwierige interne Prozesse für die Einführung angestoßen werden. Der Fußschalter soll ohne einer Installation die Funktion der Windows-App, die Serialisierung von Messergebnisse über einen virtuelle COM-Port im MUX50 bzw. DMX16-Protokoll, zur Verfügung stellen.
Des weiteren muss zum Senden eines Messwerts an den Computer bei Messschiebern und Messuhren 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. Durch die Entwicklung eines Fußschalters der sich über Bluetooth mit dem Messwerkzeug verbindet, wird der Einsatzbereich deutlich vergrößert und der Einsatz dem Anwender erleichtert.

2 Bluetooth Low Energy

asjdf

3 Ausgangszustand

Zum Beginn dieser Arbeit wurde bereits in meiner vorangegangen 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. 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 öffnet und andererseits über BLE mehrere HCTWerkzeuge 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 HCT-Dongle-App

Die Anwendung nutzt das Bluetooth Modul der Hoffmann Group. Es stellt dabei die Anwendungsschicht des BLE-Stack da und abstrahiert somit die BLE spezifischen Aufrufe zum Softdevice. 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 Kanalzuweisung gespeichert werden. Die Kanalzuweisung wird für das MUX50 bzw. DMX16 Protokoll benötigt. 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 (CDC) initialisiert und konfiguriert. Der Code

wurde nur mit kleinen Änderungen aus den Beispielen des NRF_SDKs übernommen, 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, musste der Treiber block_device_ram.c 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 einreiht. 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 ërased"werden, bevor sie geschrieben werden kann und die Block Logik des MSC wird auf die Flash Page Logik des Speichers übertragen.

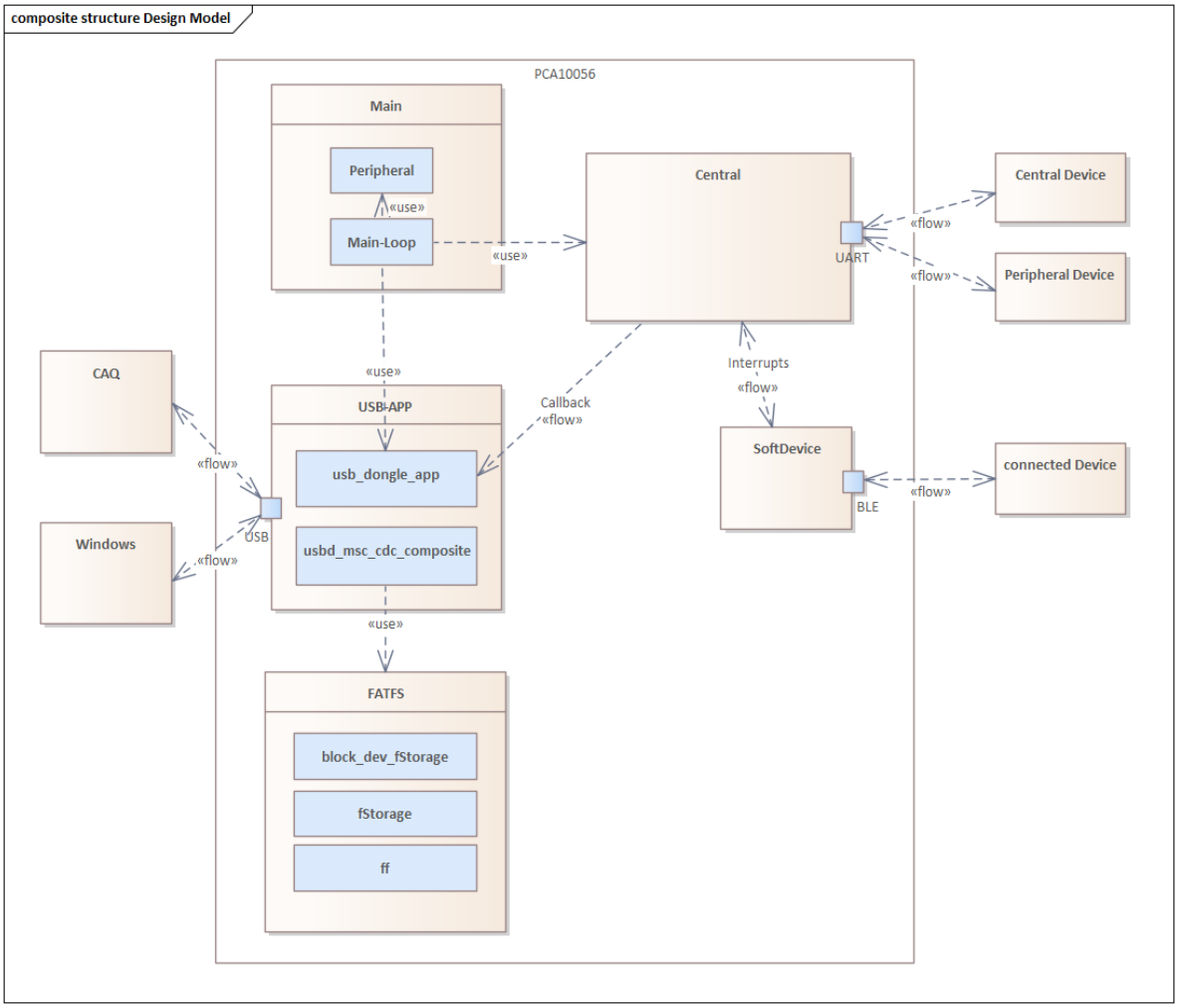


Abbildung 2: Aufbau des Projekts im Ausgangszustand

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.

4 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.

4.1 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. Insbesonders 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 beeinhalten. 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 Software architektonischen 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 Initialisierungsaufruf 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.

4.2 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. 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. Zum 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 Wekrzeug 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.

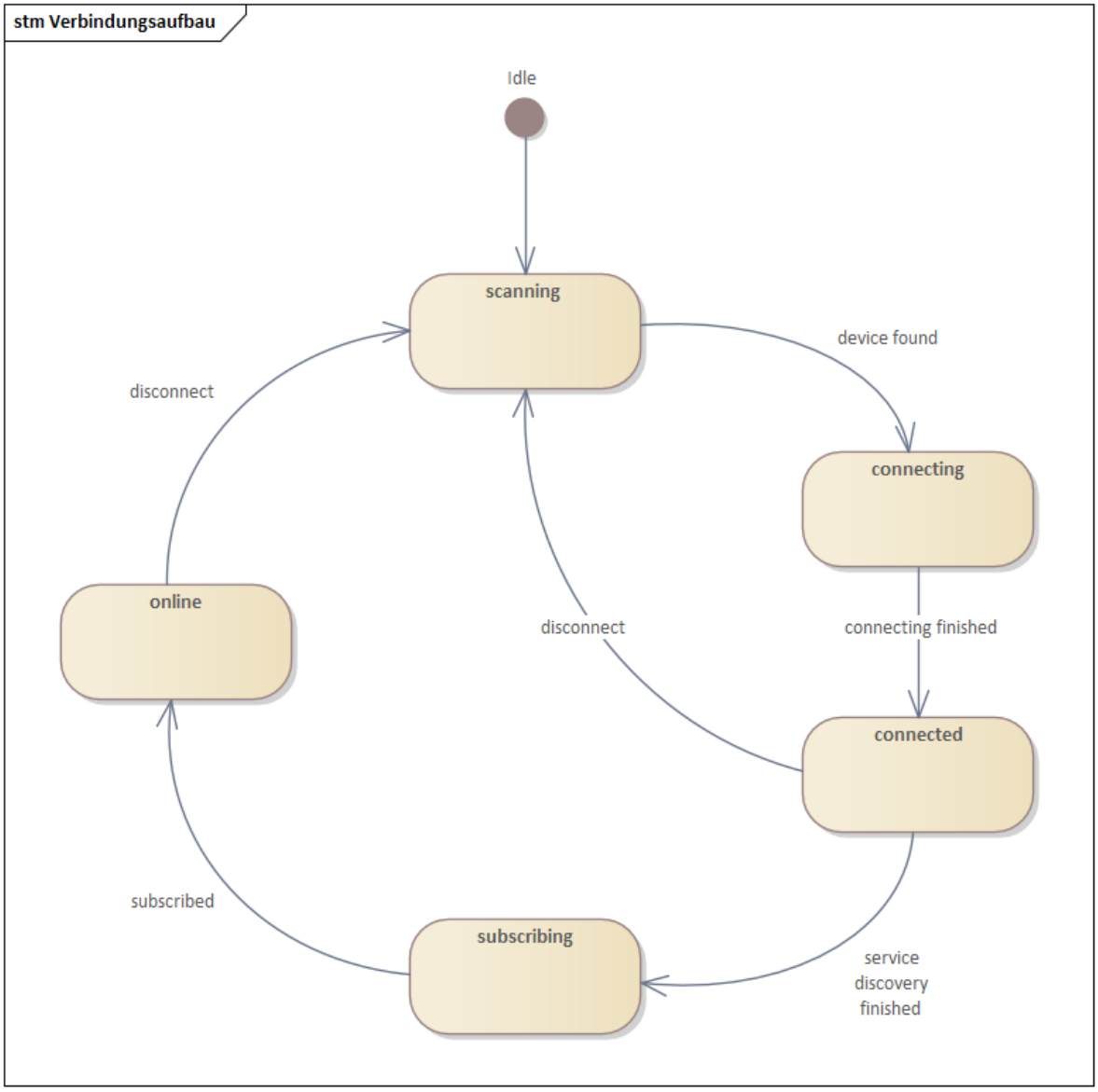
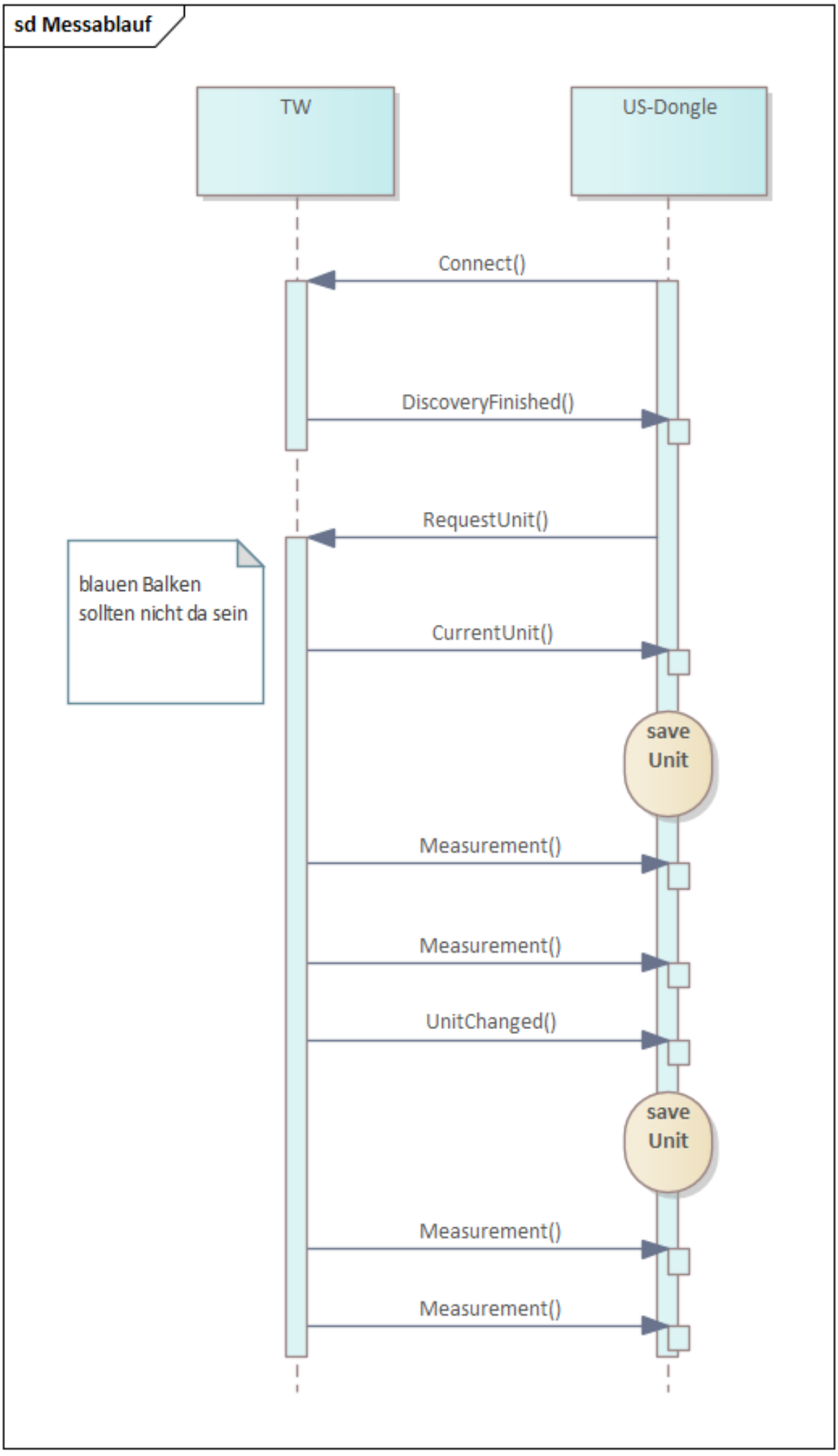


Abbildung 3: Zustandsdiagramm des Verbindungsaufbaus

4.3 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 Holex 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 Holex 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. 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.


sd Messablauf
TW
US-Dongle
Connect()
DiscoveryFinished()
RequestUnit()
blauen Balken
sollten nicht da sein
CurrentUnit()
save
Unit
Measurement()
Measurement()
UnitChanged()
save
Unit
Measurement()
Measurement()

5 Messmodi

Die durch den Fußschalter gesammelten Daten sollen in einem möglichst breiten Spektrum zur Weiterverarbeitung bereitgestellt werden. Auch durch den Fußtaster gegebene Funktionalitäten, wie aus 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 sich entschieden Messmodi einzuführen. Dieser kann in einer zusätzlichen globalen Konfigurationsdatei angegeben werden und legt fest welche Funktionalität durch den Fußschalter bereitgestellt wird. Es folgt dem bereits bei der Konfiguration der zu verbindenden Geräte verwendeten Schema, dass das Gerät 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 Menüs aufzurufen und mit Hilfe eines Bedienungsanweisung 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. Zum Zeitpunkt der Implementierung der Dongle-App war die App 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 die die 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. 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

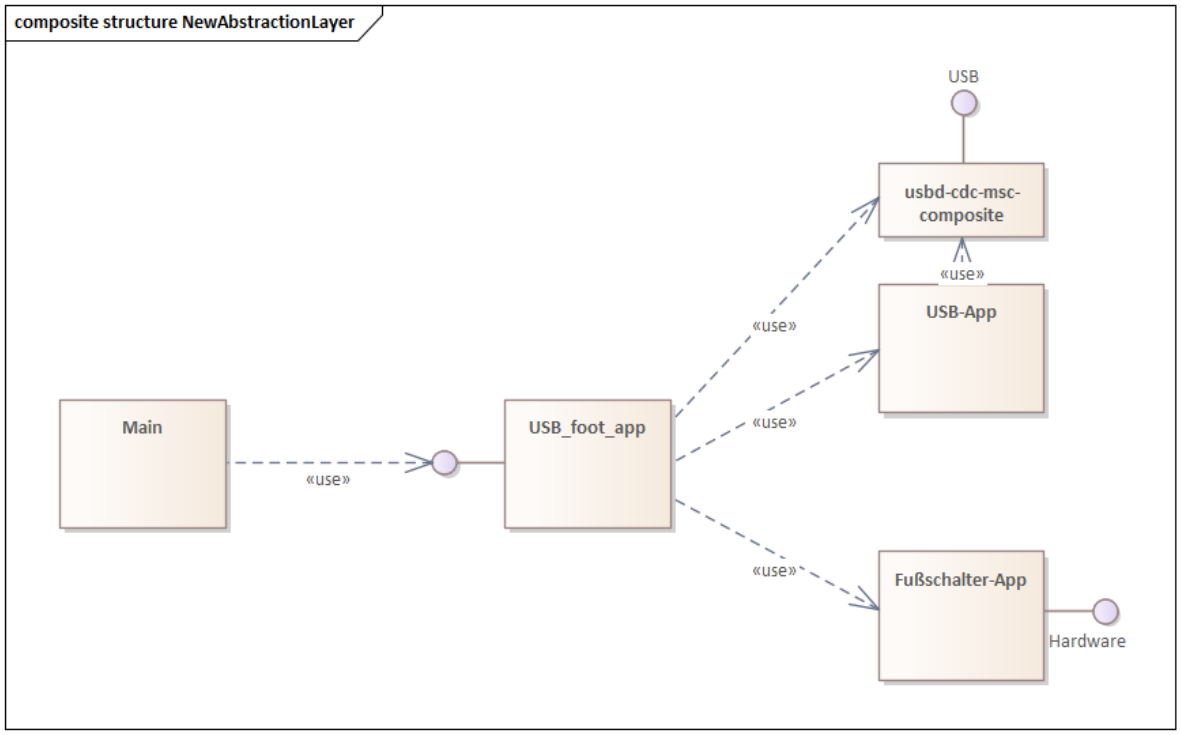


Abbildung 5: Neue Abstraktionsschicht

5.1 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 einfach in Excel oder einem Texteditor aufgefangen werde.
Für die Implementierung werden die Funktionen des NRF-Library Files app_usb_hid_kbd.h verwendet. 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 Tasturtasten, 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, 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 dopplete Zeichen nicht korrekt erkannt, sondern nach ungefähr 10 Ausgaben schien der USB-Bus überlastet zu sein. 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 kofigurierbares Zeichen aus, beziehungsweise bei einem Doppelklick ein Zweites, wie in Abschnitt 5.2 beschrieben ist.

5.2 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 Neutstart 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 Lösungansatz 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

Lösungansatz umgesetzt wurde. Der Peer Manager ist dabei eine Library von NRF die sich um Verschlüssung, Pairing und Bonding kümmert (NordicSemiconductors, 2021).
Erste Tests zeigen, dass die Geschwindigkeit der Übertragung, einerseits die Dauer bis zum ersten Tastendruck und anderseits 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.

5.3 BLE-HCT-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 seperator
- HID number seperator

- HID single key

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.

6 Einbindung Messuhren

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.
Daher wird auch in Vorbereitung auf die Implementierung der fußschalterspezifischen Features die Dongle-App um die Unterstützung der Messuhren bzw. Messschieber erweitert. 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.

6.1 Unterscheidung der Geräte

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 darüber hinausgehenden Daten schwerwiegende Unterschiede.
Im einfachsten Anwendungsfall ist der Fußschalter mit mehren 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
Holex 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 Holex 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 advertisen 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 Holex, 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. Dies muss mithilfe eines Debuggers überprüft werden, da vorerst die Ausgabe der Messergebnisse von Messuhren nicht funktioniert. Dazu werden einen Drehmomentschlüssel und eine Messuhr mit einem nRF52840 DK development board verbunden. Dann kann mit dem Debugger der IAR-Workbench Integrated Development Environment (IDE) direkt die Werte in der Gerätestrukturen überprüft werden.

6.2 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 Holex 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.

6.3 Gruppenfunktion

Zwischen Messuhren und Drehmomentschlüssel besteht ein 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, 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.
Durch das Betätigen des Fußtasters des Fußschalters soll bei allen verbundenen Messuhren das derzeitige Messergebnis abgefragt werden. Während im Modus 2 (CDC) ein Messergebnisse, 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 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 kleine Batterien bei Inaktivität schnell ausschalten und die Verbindung zu ihrem Central trennen. Anderseits 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 Ablaufen 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 des Messergebnis ausgegelöst.

7 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 Hardwarezeichnungen der Platine.

7.1 Energie Management

Wenn der Fußschalter nicht über USB mit einer Stromquelle verbunden ist, bekommt er den benötigten Strom von dem fest eingebauten Akku. Um diesen nicht unnötig zu belasten, muss ein Energiemanagement geschaffen werden, dass die bestehenden Ressourcen optimiert. Die Erhöhung der Effizienz der Batterienutzung kann dabei auf Weisen erreicht werden. Einerseits kann bei Nutzung des Fußschalter die Energieeffizienz verbessert werden. So wurde bereits, wie in Kapitel 3.3 beschrieben, eine Optimierung der Abfrage der Messeinheit durchgeführt. 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 angemesse Zeitdauer hängt von persöhnlichen 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.
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 Central und Peripheren 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 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 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 Ablaufen des Inaktivitätstimers die Anwendung vollständig heruntergefahren, was noch energieeffizienter ist.

7.2 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.

7.3 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 4.3 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.

8 Überarbeitung des MSC

Zum Beginn dieser Arbeit muss der Dongle jedes Mal, wenn die Konfigurationsfiles geändert wurden, ab- und wieder eingesteckt werden, damit die Anwendung neugestartet und die Dateien erneut eingelesen werden. Beim Fußschalter ergibt sich nun das Problem, dass das Gerät einen Akku besitzt, weswegen ein harter Reset durch den Anwender nicht ohne weiteres möglich ist. Es müssen daher Möglichkeit evaluiert werden, wie die Anwendung des Fußschalters Änderungen an den Konfigurationsdateien detektieren kann, damit anschließend automatisch ein Neustart des Geräts durchgeführt werden kann.

8.1 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 Systemreset durchzuführen. Dabei hat sich jedoch gezeigt, dass eine Änderung am File 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. Dies führte jedoch dazu, dass der System Reset 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 bei dem Read Befehl 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.
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.

8.2 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. Daher müssen folgende Informationen aus der Boot Section ausgelesen werden und folgende Berechnung durchgeführt werden:

- Sa: Startadresse Filesystem
- Sg: Sektorengröße
- 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 Ersetzten des ersten Buchstaben durch 0x5a invalidiert. 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 beeinhaltet. 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.

8.3 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. Nachforschungen ergeben, dass die Zugriffe auf den Flash vom Softdevice abgearbeitet werden müssen. Bestehen also mehrere aktive Verbindungen steigt die Wahrscheinlichkeit von Fehlerhaften Schreibzugriffen stark an. Getätigte Änderungen an den Konfigurationsfiles werden dann nicht übernommen und sind selbst nach langen Wartenzeiten, nach dem Reset verloren. Deshalb werden, falls Schreibzugriffe gequeuet 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. 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.

9 Schlusswort

9.1 Stand Produktentwicklung

Der Fußschalter und der Dongle sollen als eigenständige Produkte in das Sortiment der Hoffmann Group übernommen werden. Sie werden jeweils als Projekte von dem Produktmanagement übernommen. Dabei werden die Hardwareänderungen wie in Kapitel 5.1 beschrieben, in das Layout der Platine übernommen. Zusätzlich soll der Pin, der den Chip in den Bootloader Mode versetzt, 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 Anfang August erhalten.
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.
Für die Funktionalität des Fußschalter beziehungsweise des Dongles wird eine Erfindungsmeldung herausgegeben und der Prozess der Patentierung der Technik wurde angestoßen.

9.2 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, umgesetzt. Eine weiterführende Optimierung des Modus BLE-HID steht noch aus, aber steht einer Produkteinführung nicht im Weg.

Literatur

NordicSemiconductors. (2021). *nRF5 SDK v17.1.0 Peer Manager*. Verfügbar 27. Juli 2023 unter https://infocenter.nordicsemi.com/index.jsp?topic=%2Fsdk_nrf5_v17.1.0%2Flib_peer_manager.html

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

München, den 8. August 2023