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: 12. Juli 2023

Abstract

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

1 Einleitung

In diesem Kapitel wird zunächst vorgestellt, welche die Geräte sind, mit denen der Fußschalter interagiert und welche 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 ermöglicht eine Automatisierung der Qualitätskontrolle, sowie einen Nachweis, dass Standards in der Fertigung eingehalten wurden, was in Branchen wie der Automobilindustrie oder der Luftfahrt von höchster Bedeutung ist.
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 über BLE-HID 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 CSV-Datei, in der alle durchgeführten Messungen mit einer großen Anzahl an zusätzlichen Daten gespeichert werden. Wird das Gerät über USB mit dem Computer verbunden, zeigt es sich als 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 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 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.

1.2 Motivation

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 drahtgebunden sowohl an das Werkzeug als auch an den Computer, dieses Senden der Messung auslösen können. Durch die drahtgebundene 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 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 in der Lage ist sich selbstständig mit mehreren Drehmomentschlüsseln zu verbinden. Sie stellt die Messergebnisse bereits über einen virtuellen COM-Port im MUX50 bzw. DMX16 Protokoll zur Verfügung und die zu verbindenden Geräte sind über eine CSV-Datei, die über MSC dem Nutzer zur Verfügung gestellt wird, konfigurierbar.

2.1 Aufbau

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 wurden und reiht die Daten, die von Interesse sind, in eine 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.

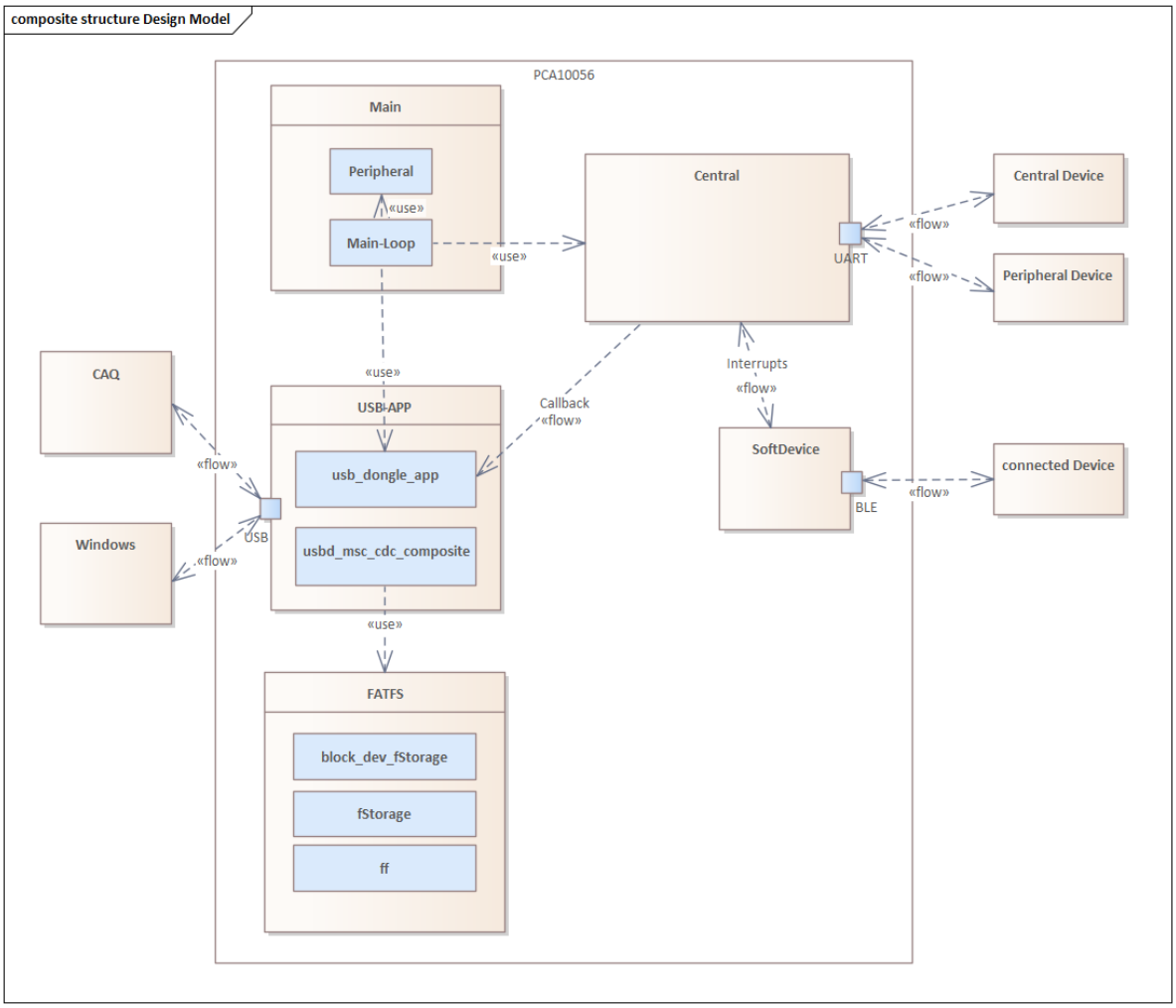


Abbildung 1: Aufbau USB-Dongle App

2.2 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. Zudem sollen die Messmodi USB-HID, USB-HID-Single-Character, BLE-HID und BLE-Windows-App geschaffen werden. Diese sollen sich neben den HID-

spezifischen Konfigurationen in einer eigenen Datei befinden. Des Weiteren müssen die Schreibbefehle des MSC optimiert werden, da sie nur sehr langsam 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 den geringeren Hardwareumfang durchgeführt werden.

3 Überarbeitung der Dongle-App

Die Dongle App dient neben dem nrf_Base Projekt als Basis für die Anwendung des Fußschalters. Sie diente als Proof-of-Concept und ermöglichte als solche die Bereitschaft des Projektmanagements die Ressourcen für die Implementierung des Fußschalters zu bewilligen. Im ersten Schritt der Implementierung wird sie um Grundfunktionalitäten erweitert und die Datenverarbeitung flexibler gestaltet.

3.1 Trennung der Projekte

Das Framework der Hoffmann Group zur Abstraktion der BLE-Schnittstelle nrf_Base wird in allen HCT-fähigen Produkten eingesetzt und stetig weiterentwickelt. Um die fehlerfreie Kommunikation des Fußschalters zu den HCT-Werkzeugen auch in Zukunft sicherzustellen, müssen diese Updates auch in das Projekt des Fußschalters übernommen werden. Während in der Proof-of-Concept Anwendung des Dongles darauf noch keine Rücksicht genommen wurden, soll in der Entwicklung des Fußschalters von Anfang an diese Abhängigkeit berücksichtigt werden und die Dongle-App entsprechend angepasst werden. Des weiteren soll auch der Umfang der Anwendung, die auf dem Dongle läuft, minimiert werden und der Code für die Peripherie des Fußschalters abgetrennt werden. Natürlich darf in anderen Produkten als dem Dongle und dem Fußschalter, deren Code nicht zu finden sein.
Folgende Peripherie darf dabei nur in den jeweiligen Projekten initialisiert werden:

- nrf_Base: UART
- HCT-FootSwitch: Fußschalter, Akku Power Management, Drei-Farben LED, USB
- Dongle: Ein-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. Während die erste Möglichkeit den Vorteil hat, 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 Softwarearchetektonischen Gründen die Software bereits gekapselt, wodurch die Programmatische Abtrennung der Teile keinen außerordentlichen Entwicklungsumfang 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 gesamter 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 entfernt 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.

3.2 Verbesserung Verbindungsaufbau

Im ersten Prototyp der Dongle-App wird, sobald die Konfigurationsdatei eingelesen wurde, überprüft ob die jeweiligen Geräte bereits verbunden sind. Falls ein Gerät unverbunden ist, wird das Scanning gestartet und periodisch die vom Central gefundenen Geräte auf das zu verbindende Gerät hin untersucht. Wird es schließlich gefunden, wird der Verbindungsaufbau angestoßen und sobald die Service discovery stattgefunden hat, wird sich auf die HCT-Charakteristik subscribed und das Connection Handle des Geräts aus dem Central Device übernommen.
Dabei werden jedoch zwei Zustände nicht abgebildet. Zum einen wird zwischen der Verbindungsaufnahme und der Subscription auf die HCT-Charakteristik, der Zustand “Connected” eingenommen. Hier wird bereits das Connection Handle zugeordnet und sollte gespeichert werden, da ab der Einnahme dieses Zustand ein disconnect auftreten kann und nur über das Connection Handle nachvollzogen werden. Zum anderen wird nach der erfolgreichen Subscription ein Acknowledgement übermittelt und im Central der Zustand “Subscription” eigenommen. Hier wurde in der Dongle-App bereits das Connection Handle dem Gerät zugeordnet, weswegen ein Disconnect nachvollzogen werden kann, jedoch wird die initiale Kommunikation mit der Abfrage der Messeinheit forgesetzt. Dabei wurde bisher nicht gewartet, dass die Subscription erfolgreich abgeschlossen wurde, was jedoch die Zuverlässigkeit des Verbindungsaufbaus erhöht.

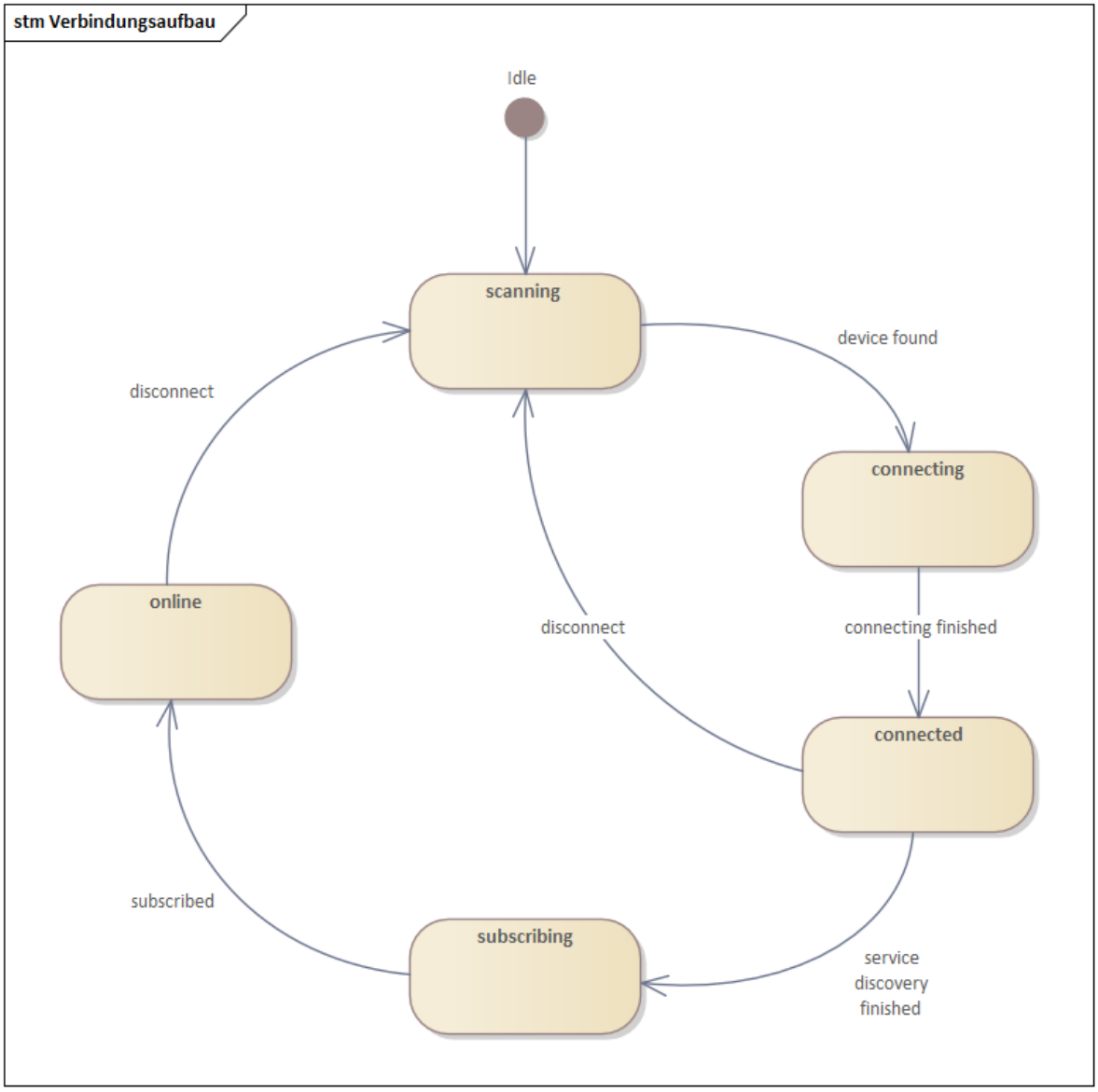
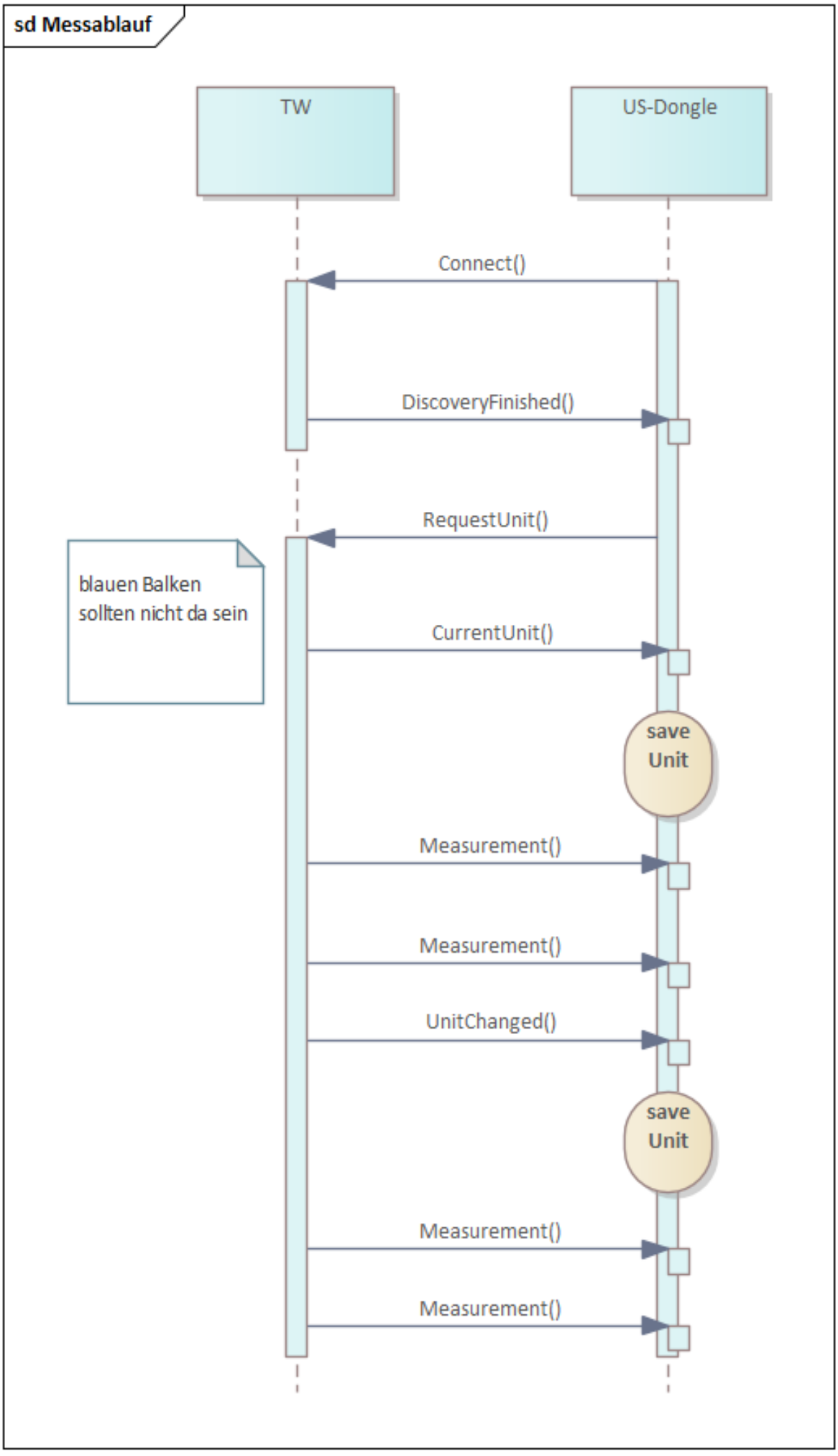


Abbildung 2: Zustandsdiagramm des Verbindungsaufbaus

3.3 Optimierung Abfrage Messeinheit

Im derzeitigen Stand der USB-Dongle App muss die Anwendung bei Drehmomentschlüsseln und Messuhren 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 über USB-CDC aus.
In beiden Fällen sendet der Drehmomentschlüssel automatisch bei einer Änderung der Messkonfiguration einen Datenblock mit der Messeinheit an die Dongle App.
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 wird die Dongle App notifiziert und die gespeicherte Einheit aktualisiert. Dadurch wird eine fehlerhafte Einheit in der Ausgabe über USB-CDC vermieden und die Anzahl an Nachrichten der Dongle-App an das Messgerät verringert.


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

4 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 spezielle auf die Messuhren bezieht. Messuhren und Messschieber können dabei gleich bedeutend verwendet werden, da das HCT-Interface für beide Geräte identisch ist. Bisher ist die Dongle App daraufhin ausgelegt, dass mit einem Drehmomentschlüssel kommuniziert wird und die erhaltenen Binärdaten werden entsprechen interpretiert. Für die Einbindung der Messuhren muss nun die Unterscheidung eingeführt werden mit welcher Art von Werkzeug kommuniziert wird und entsprechend die Kommunikation angepasst werden.

4.1 Unterscheidung Geräte

Die 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 bzw. Messschieber zugeordnet werden. Das ist jedoch unter Umständen fehleranfällig, da das Sortiment an HCT-Werkzeug stetig wächst und es nicht ausschließen 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 Class ID 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. Letztendlich wird nach dem Callback, der vom Central Device aufgerufen wird, wenn die erfolgreiche Subscription der Anwendung auf die BLE-Charakteristik des Werkzeugs stattgefunden hat, eine Nachricht zu Abfrage der Device Information gesandt. Die erhaltenen Daten werden dann in der zum Gerät gehörenden Struktur gespeichert.

4.2 Korrigierung Abfrage Messeinheit

Die Protokollbeschreibung der Messuhr zeigt, dass das Messergebnis zwar innerhalb des gleichen Datenblocks wie bei dem Holex Drehmomentschlüssel liegt, mit dem sich den gleichen Daten Callback teilt, jedoch innerhalb des Datenblocks an anderer Stelle. Diese Offsets sind als Konstanten im Headerfile der USB-App definiert und müssen um die entsprechenden Einträge für die Messuhren bzw. die Messschieber erweitert werden. Zudem handelt es sich bei dem Messergebnis, das von Interesse ist, beim Drehmomentschlüssel um den „Peak Torque“, der als Gleitkommazahl codiert, während es sich beim Messschieber bzw. Messuhr um „Measurement Distance“ als Ganzzahl codiert handelt. Im Fall der Messuhr wird das binäre Messergebnis erst als Ganzzahl intrepretiert, da es sich um die Distanz in

Mikrometern handelt und anschließend auf Millimeter umgerechnet. Es wird als Nächstes überprüft, ob es sich bei der derzeitige Einheit des Geräts um inch handelt, da dann der Wert erneut durch 1000 dividiert werden muss, um den korrekten Umrechnungsfakor zu erhalten und von Mikroinch auf Inch zu gelangen. Es wird letztendlich eine Gleitkommazahl erhalten, wodurch sie der Nachrichten Struktur gespeichert werden kann und diese nicht angepasst werden muss.
Die Einheitenkodierung der Messuhr ist komplementär zur Kodierung der Drehmomentschlüssel und daher kann die if-Cascade, die 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.

4.3 Gruppenfunktion

Während im Modus 2 (CDC) die Messergebnisse von mehreren, durch Betätigung des Fußschalters getriggerten Messungen, anhand der Channelnummer einem Werkzeug zugeordnet werden können, ist dies in den HID-Modi nicht der Fall. Daher wird das devices.csv Konfigurationsfile um eine Spalte mit einer Gruppennummer erweitert. Werden die Geräte durch Betätigung des Fußschalters als Gruppe getriggert, werden die Messerergebnisse so sortiert, dass sie gemäß dieser Gruppenreihenfolge ausgegeben werden, wodurch sie wieder einer Messuhr zuordbar sind. Dazu bedarf es einem Counter, der durch Betätigung des Fußschalters, von 0 auf 1 gesetzt wird, wodurch der Start der Gruppenfunktion später 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 unverbundenen Geräte 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 Test und Anregungen von Messtechniker ergab sich außerdem, dass wenn ein Gerät der Gruppe zwar konfiguriert, jedoch nicht verbunden ist, es besser ist die gesamte Gruppe nicht zu triggern. Das ergibt sich einerseits dadurch, dass durch ihre relativ kleine Batterie der Messuhren, sich bei Inaktivität schnell ausschalten und die Verbindung zu ihrem Central trennen. Anderseits jedoch soll möglichst jede Messung korrekt sein, da sie in der 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 lediglich zwei kurz rot auf, um diesen Fehler zu signalisieren.

5 Einbindung Hardware

Die Prototypen des Fußschalters von 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.

5.1 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, soll der Fußschalter nach einer gewissen Zeit so weit wie möglich heruntergefahren werden.
Um diese Anforderung zu erfüllen, 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, wie das erhalten einer Nachricht oder dem Verbinden eines Geräts, 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, auch nicht durch Betätigung des Fußtasters wieder neugestartet werden.
Zunächst zeigte sich jedoch das Problem, dass bei der Hardware des Fußschalters, der Akku auf der Datenleitung des USBs 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. 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.

5.2 Fußschalter Funktionalität

Nach Überarbeitung der Codebasis und Einbindung der Messuhren, stehen alle Funktionalitäten bereit um den eigentlichen Fußschalter einzubinden. Während bei der initialen Einbindung des Fußtasters, eine Betätigung das Abfragen der Messergebnisse bei allen Messuhren triggerte, wuchs die Funktionalität stehtig.
Zum Ende dieser Arbeit wird durch ein kurzes Betätigen oder einem “einfachen Klick” die bereits beschriebene Gruppenfunktion ausgeführt. Im Modus 0 (single Character HID) 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 hinterlegt 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 dem 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 Messer vorgestellt wurde, aber der automatische Reset des Geräts bei Änderung der Konfigurationsfiles noch nicht zuverlässig funktionierte. Sie wurde beibehalten, da sie als sehr benutzerfreundlich empfunden wurde.

5.3 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 1: LED-Zustände

6 Messmodi

Sowohl der Fußschalter als auch der USB-Dongle sollen in mehreren verschiedenen Operationsmodi laufen können. Dieser wird in einer zusätzlichen globalen Konfigurationsdatei spezifiziert. Zum Zeitpunkt der Implementierung der USB-App war die App die oberste Abstraktionsschicht aller USB-Funktionalität. Das ist mit Einführung der Fußschalter Funktionalitäten nicht mehr der Fall. Daher bedarf es einer neuen Abstraktionsschicht, die über USB- und Fußschalterapp und die Funktionalitäten beider dirigiert. Dadurch können die Operationsmodi programmatisch getrennt werden, sodass bestimmte Schritte des Initialisierungsprozesses, wie das Einlesen der Konfigurationsdatei der zu verbindenden Geräte, in einem Modus wie HID einzelnes Zeichen nicht ausgeführt werden.

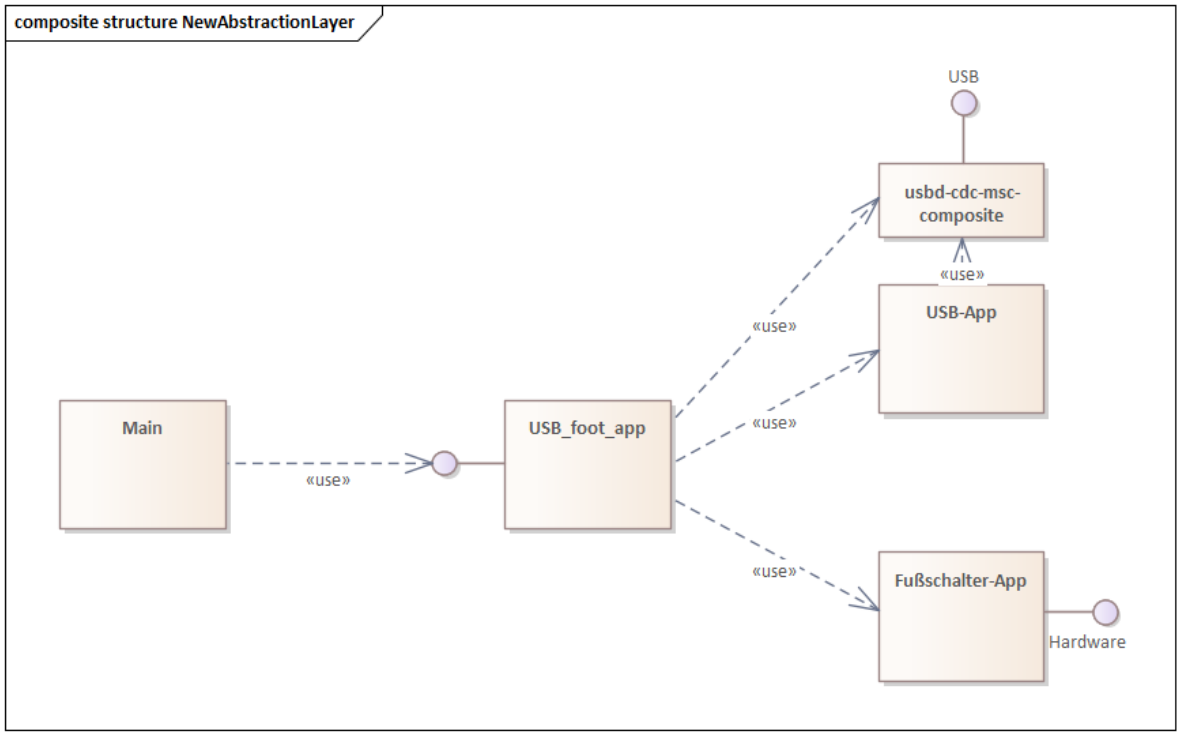


Abbildung 4: Neue Abstraktionsschicht

6.1 USB-HID

Ein Feature, das sowohl für die USB-App als auch für den Fußschalter implementiert werden soll, ist das Human Interface Device (HID). In diesem Modus gibt die Anwendung die Messergebnisse nicht mehr über den virtuellen COM-Port aus, sondern ist über USB als Tastatur mit Computer verbunden und gibt die Zahlen des Ergebnisses als Tastendrücke, gefolgt von einem konfigurierbaren Terminierungszeichen, ein. Dadurch können Messungen einfach in Excel oder einem Texteditor aufgefangen werde.

In einem ersten Schritt wird grundsätzlich die Funktionalität und das Messergebnis zusätzlich zur Ausgabe über den virtuellen COM-Port ausgegeben. Dies ist eine Vorbereitung für die Implementierung der verschiedenen Operationsmodi.

6.2 BLE-HID

Ein weiterer Modus, in dem der Fußschalter arbeiten soll, ist HID über BLE. Das bedeutet, dass sich der Fußschalter als Tastatur im kabellosen Zustand präsentieren kann und wie bei HID über USB die Messergebnisse als Tastatur Tastendrücken in einen Editor oder Excel eintippt. 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 Peripheral Verbindungsaufbaus, die Fußschalter Applikation eingebunden werden. Für das eigentliche Schreiben des Messergebnisses über BLE wird gibt es bereits bestehenden Funktionen und diese müssen nur in der Fußschalter Applikation aufgerufen werden. Erste Tests zeigen, dass die Geschwindigkeit der Übertragung, einerseits der Dauer bis angefangen wird das Messergebnis zu schreiben und anderseits wie schnell die einzelnen Tastendrücke erfolgen, stark von USB oder einen mitlaufenden Debugger beeinträchtigt wird. Beides sollte im eigentlichen Anwendungsfall dieses Modus nicht vorhanden sein.

6.3 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 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.

7 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 Files erneut eingelesen werden. Beim Fußschalter ergibt sich nun das Problem, dass das Gerät einen Akku besitzt, weswegen ein harter Reset durch den Anwender nicht möglich ist.

7.1 Evaluierung der Möglichkeiten zur Detektion

Der erste Versuch der Unternommen wurde um dem Anwendungsfall gerecht zu werden, ist über die File Information des 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 es Änderungen, die am Ende der Datei stattgefunden haben nicht detektiert werden.

7.2 Einlesen der Länge der Datei

Es hat sich gezeigt, dass die Länge der Datei nicht erneut eingelesen wird, wenn die Datei aus Windows heraus geändert wird 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 können, da die Datei mit der alten Länge eingelesen wird. Jedoch konnte festgestellt werden, dass die Information 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 “File Allocation Tables” (FAT). Daher müssen

folgende Informationen aus der Boot Section ausgelesen werden und folgende Berechnung durchgeführt werden:
$StartadresseFS + Sektorengroee * AnzahlreservierterSektoren + Sektorengroee * AnzahlFAT * AnzahlSektorenproFAT = StartadresseFS + Sektorengre * (AnzahlreservierterSektoren + AnzahlFAT * AnzahlSektorenproFAT)$
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. Es zeigt sich zudem, dass das Directory Entry größer als ein Sektor werden, auch wenn er nur zwei Dateinamen beeinhaltet. Das liegt daran, dass bei Löschen und Neuanlegen der Dateien, zum Beispiel beim Kopieren der Dateien aus einem Verzeichnis auf dem Computer, die alten Einträge nicht gelöscht sondern entvalidiert werden. Es wird daher nacheinander die Sektoren, die für das Directory Entry allkiert sind einzulesen und auf die korrekten Dateinamen hin zu untersuchen.

7.3 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 gegangen. 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.

8 Abschluss

8.1 Stand Produktentwicklung

8.2 Fazit?

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 12. Juli 2023