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: 29. Juni 2023

Abstract

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

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.

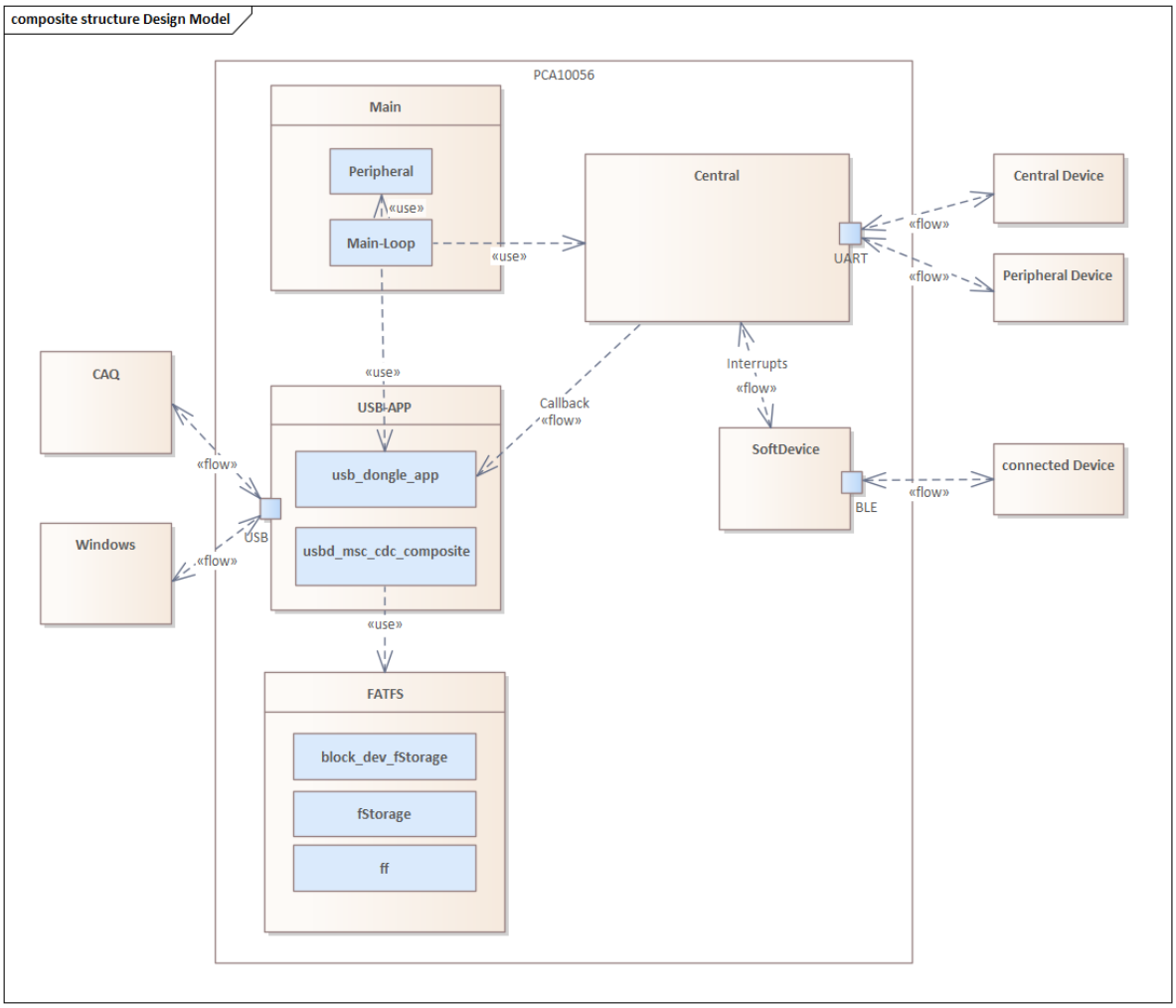


Abbildung 1: Aufbau USB-Dongle App

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.

3 Implementierung

3.1 Einbindung Messuhren

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.

3.1.1 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 es alte Messschieber gibt, die zwar unter einem korrekten Name advertisen aber nicht das HCT-Protokoll sprechen. Jedoch sollten diese Geräte aufgrund der Inkompatibilität der Protokolle letztendlich nicht verbunden werden.
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. 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.

3.1.2 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 float codiert, während es sich beim Messschieber bzw. Messuhr um „Measurement Distance“ als int32 codiert handelt. Im Fall der Messuhr wird das binäre Messergebnis erst als int32 intrepretiert, da es sich um die Distanz in Mikrometern handelt, anschließend durch 1000 dividiert, um Millimeter zu erhalten. 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 von Mikroinch auf Inch zu gelangen. Abschließend wird der Wert im float32 der Nachrichten Struktur gespeichert, welche nicht angepasst werden musste. Zwischen Messuhr und Messschieber muss keine weitere Unterscheidung gemacht werden.
Die Einheitenkodierung der Messuhr ist praktischerweise komplementär zur Kodierung des

HCT Drehmomentschlüssels 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.

3.1.3 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, dadurch ist der Start der Gruppenfunktion später erkennbar. Bei der Abarbeitung der erhaltenen Nachrichten, wird dann über die Zuordnung zum Device, die Nachricht ausgewählt und weitergegeben, die zum Counter korrespondiert. Unvergebene Gruppenids 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 BLE übermittelt.

3.2 Einbindung Hardware Fußschalter

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. Da der Fußschalter zwar die USB-Dongle App beinhaltet, jedoch der Dongle auch als eigenständiges Produkt angeboten werden soll, muss der Code der beiden Projekt von Anfang an getrennt bleiben. Dazu muss ein neues Projektfile eingeführt werden und Abhängigkeiten zwischen den Files so gering wie möglich gehalten werden. Unvermeidbare Abhängigkeiten werden durch Compilerschalter getrennt.

3.2.1 Triggern des Messergebnis

Das erste Fußschalter spezifische Feature ist, dass durch Betätigung des Fußschalters eine Messung bei Messuhren bzw. Messschiebern ausgelöst wird. Das Senden bzw. das Schreiben der Daten, die den Messvorgang in einer Messuhr auslösen, wurde dabei bereits im Zuge der Einbindung der Messuhr implementiert. Dieser Vorgang muss nun durch das Betätigen des Fußschalters ausgelöst werden. Dazu muss in einem ersten Schritt analysiert werden in welcher Form und an welchen Pin das Signal genau vorliegt und wie es abgegriffen werden kann.

3.2.2 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. In der bestehenden Anwendung ist bereits ein Energiemanagement vorhanden, dieses schaltet den BLE-Chip nach einiger Zeit der Inaktivität aus. Der Chip der Anwendung verbraucht nur wenig Strom. Daher soll nach einer gewissen Zeit der Innaktivität die Verbindungen getrennt werden, Scanning und Advertising gestoppt werden, damit das Softdevice ausgeschaltet werden kann. Durch Betätigung des Fußtasters wird das Softdevice wieder gestartet und die Messmittel erneut verbunden. Wenn USB verbunden ist, sollen keine Energiesparmaßnahmen getroffen werden.
Nach ersten Tests zeigt 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 muss am Fußschalter geringfügige Hardwareänderungen durchgeführt. Dabei wird 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.

3.2.3 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

3.3 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.

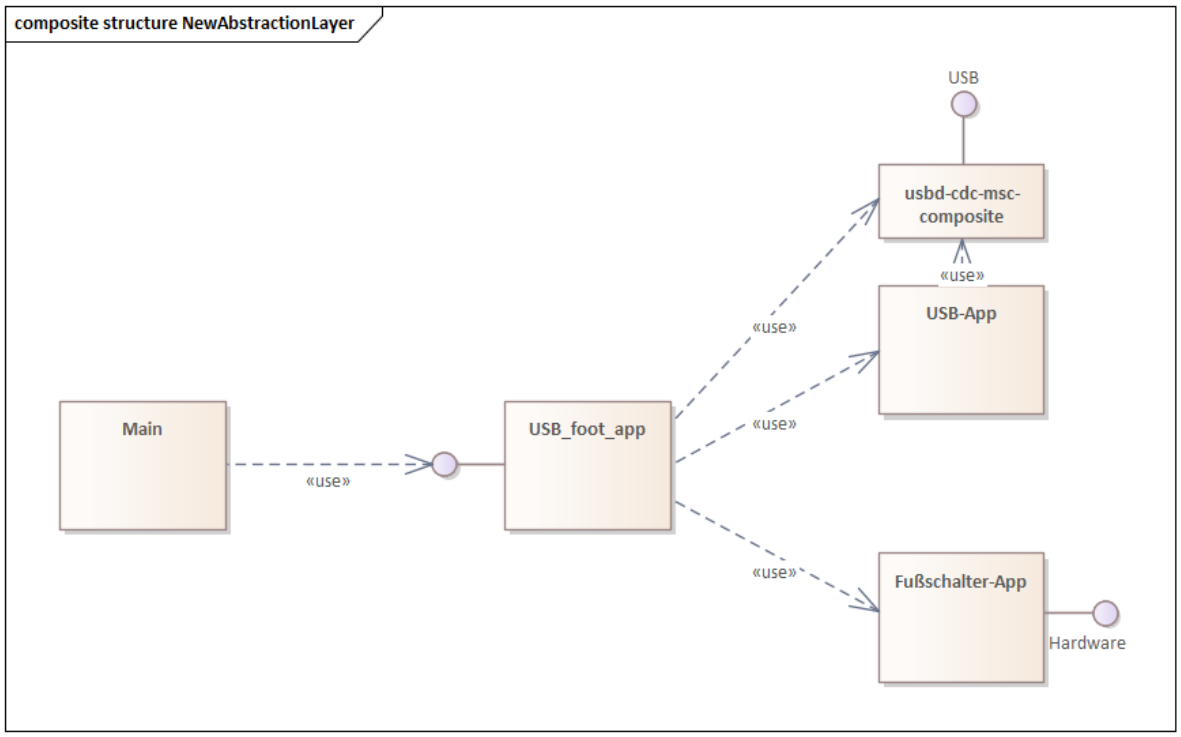


Abbildung 2: Neue Abstraktionsschicht

3.3.1 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.

3.3.2 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.

3.3.3 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. Dazu

3.4 Allgemeine Erweiterungen

3.4.1 Trennung der Projekte

Die Grundlage des Projekts nRF52_base wird unterschiedlichsten Produkte eingesetzt. Im Drehmomentprüfgerät als Central Device und im Drehmomentschlüssel als Peripheral Device für übergeordnete Device Applikationen mit der über serielle UART kommuniziert wird. Im USB-Dongle und im HCT-Fußschalter ist das Basisprojekt direkt auf einem Chip mit der Anwendung des Dongles und des Fußschalters.
Da im Dongle und im Fußschalter die UART-Kommunikation nicht benötigt wird, während im Drehmomentprüfgerät und im Drehmomentschlüssel die Anwendung des Dongles nicht benötigt wird, werden entsprechende Codestellen durch Compilerschalter aufgetrennt. So wird gewährleistet, dass für alle Projekte nur einen Codebasis mainted werden muss.
Die USB-App ist bereits gut gekapselt von dem Basisprojekt. Folgende Funktionsaufrufe müssen abgetrennt werden.

Main File:

- Usb_app_init()
- Usb_app_process()

Central File:

- Alle Aufrufe zu „app_uart_put“
- Alle USB App Callbacks

In einer ersten Ausbaustufe wird mit einem Compilerschalter „UART“ in Central File die Aufrufe zur UART abgetrennt, während mit einem Compilerschalter „USB_APP“ die Funktionalität der USB-App gekapselt wird. Letztendlich sollte die Verwendung der UART durch geeignete syntaktische Mittel weiter gekapselt werden. Nach Trennung der beiden Projekte durch Compilerschalter zeigt sich, dass die compilierten Binärfiles zwar gleich groß sind, jedoch nicht übereinstimmen, was eine weitere Analyse nach sich zieht.

3.4.2 Verbesserung Verbindungsaufbau

Im derzeitigen Zustand, sobald die Konfigurationsdatei eingelesen ist, geht die App die zu verbindenden Geräte durch und überprüft jeweils deren Flag über den Verbindungszustand. Wird ein Gerät gefunden, das nicht verbunden ist, wird das Scanning gestartet. Wird ein

Gerät von dem Central gefunden, wird der Verbindungsaufbau angestoßen. Sobald die Service discovery stattgefunden hat, wird sich auf die HCT-Charakteristik subscribed und das Connection Handle des Geräts aus dem Central Device übernommen.

Eine tiefere Analyse zeigt jedoch, dass zwischen dem Anstoß des Verbindungsaufbau und dem Subscribing auf die HCT-Charakteristik ein weiterer Zustand eingenommen wird. Wenn der Verbindugsaufbau vollzogen wurde, wird ein Flag übermittelt, sowie erstmals dem Gerät das Connection Handle zugeordnet. Ebenfalls kann in diesem Zustand bereits ein Disconnect stattfinden, was im derzeitigen Zustand nicht in der USB-App nachvollzogen werden kann, da das Connection Handle noch nicht gespeichert wurde. Daher muss dieser Zustand programmatisch abgebildet werden.

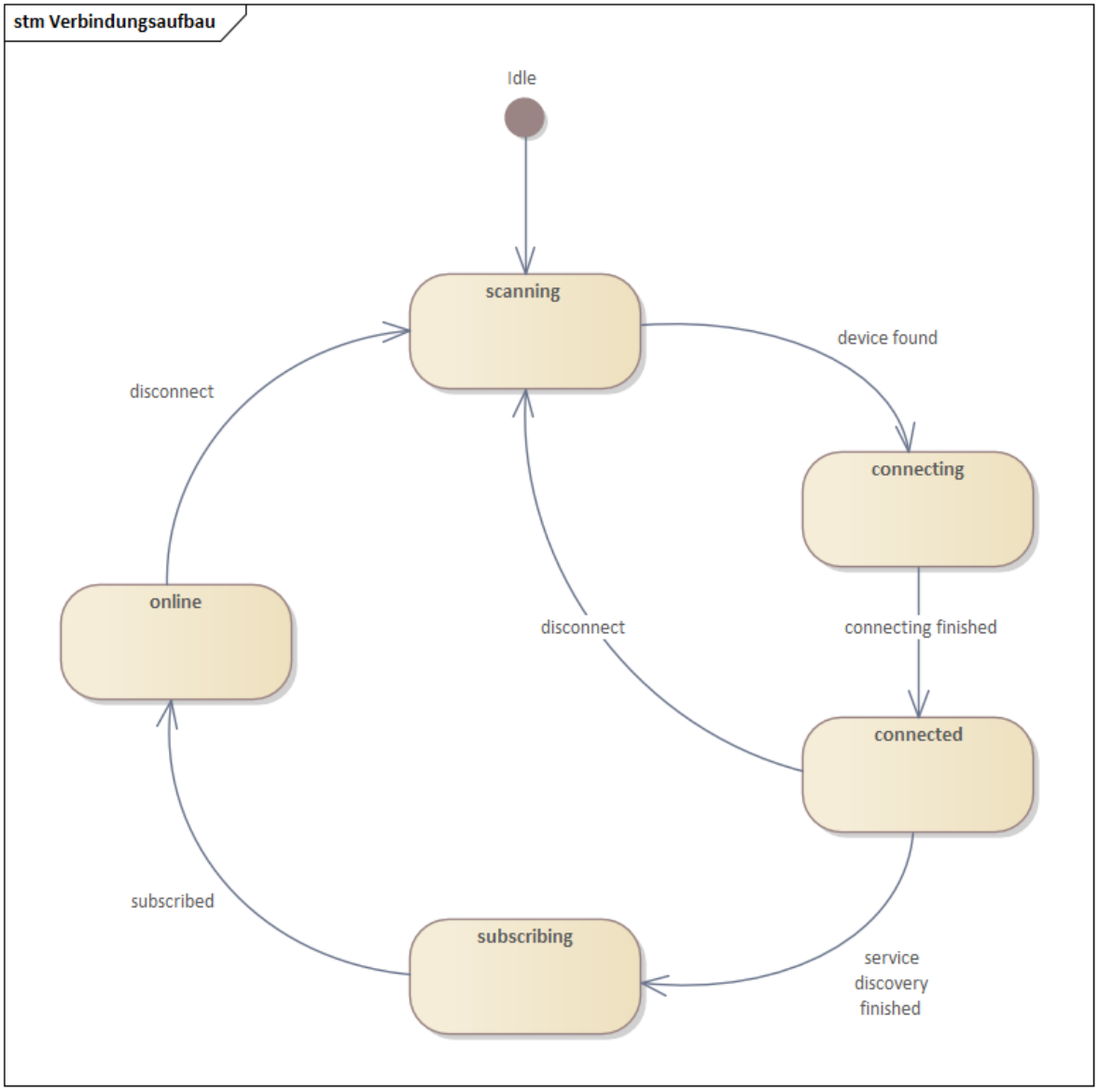
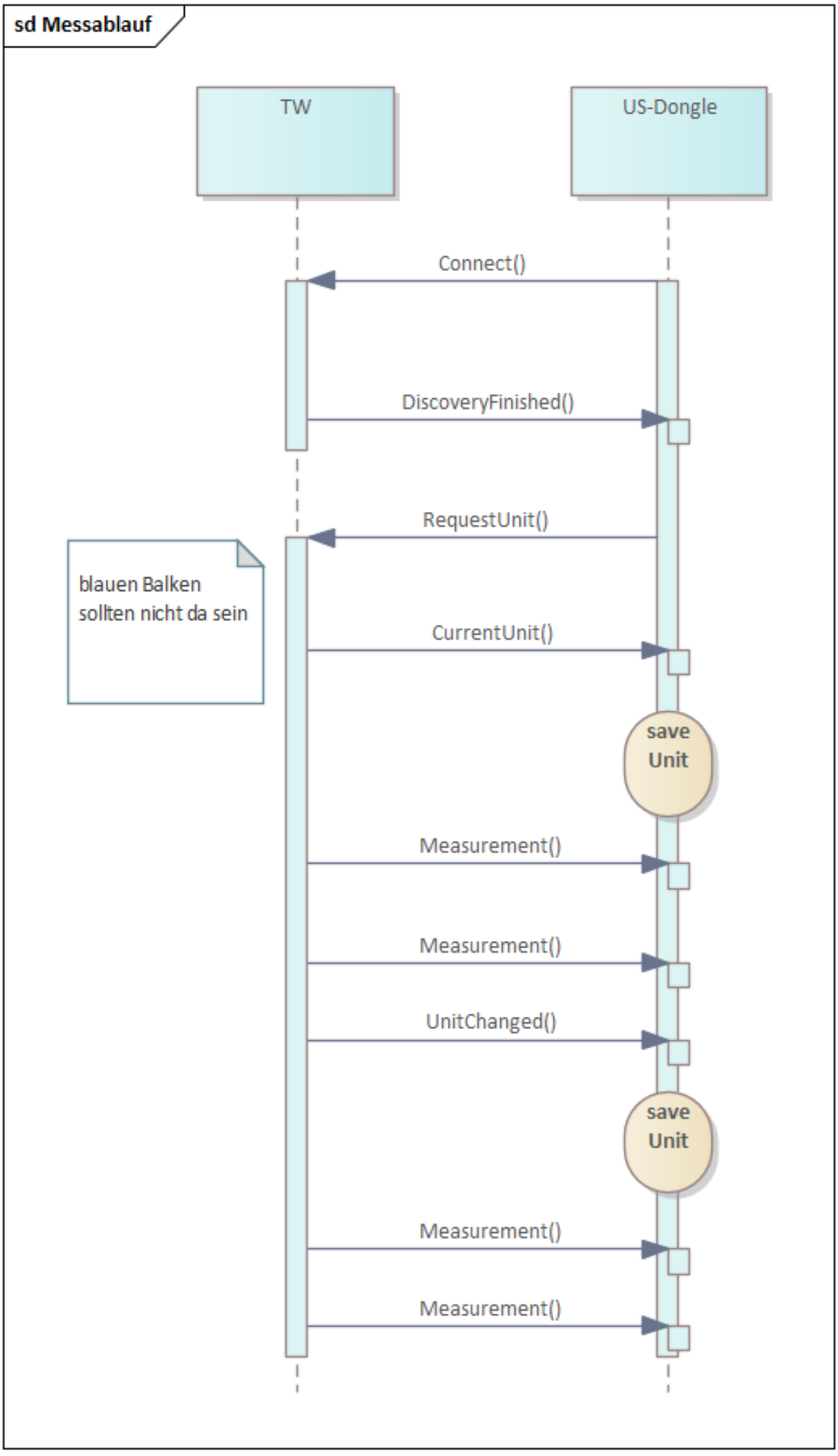


Abbildung 3: Aufbau USB-Dongle App

3.4.3 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()

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 29. Juni 2023