00 Ausgangszustand

Als Grundlage für die Entwicklung des HCT-Fußschalters dient der USB-Dongle, sowie das Basisprojekt nrf-Base. Das Basisprojekt ist dabei die Anwendungsschicht der BLE-Stacks und abstrahiert die BLE spezifischen Aufrufe zum Softdevice. Es kann sowohl als peripheral device und central device agieren und führt den Verbindungsprozess mit HCT-Geräten durch. Es wird in allen HCT-Geräten eingesetzt, wo es eine Vermittlerfunktion zwischen dem eigentlichen Geräte Chip und dem Softdevice übernimmt. Darauf aufbauend befindet sich die USB-Dongle App. Sie sitzt ebenfalls auf dem Chip des Softdevice und macht den Chip, verbunden über USB, als Massenspeichermedium verfügbar, wo sie eine Konfigurationsdatei im CSV-Format dem Anwender zur Verfügung stellt. Über diese Datei kann unter Anderem angegeben werden mit welchen Geräten sich der Dongle verbindet. Sind die Geräte verbunden gibt die USB-App durchgeführte Messergebnisse über einen virtuellen COM-PORT im Mux50 bzw. Dmx16 Protokoll aus. Dadurch werden die Messergebnisse durch CAQ-Software verwertbar, was durch den Anwendungsfall erforderlich ist.

01 Compilerschalter

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()

#include „usb\_dongle\_app.h“ ???

Central File:

Alle Aufrufe zu „app\_uart\_put“

Alle Aufrufe zu „SendCentralModuleConnectionState“

Alle USB App Callbacks

App\_uart\_flush() Aufrufe können entfernt werden

In einer ersten Ausbaustufe wird mit einem Compilerschalter „USE\_UART“ in Central File die Aufrufe zur UART abgetrennt, während mit einem Compilerschalter „USE\_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.

02 Überarbeitung der Nachrichtenverarbeitung

Im derzeitig Stand wird für empfange Messergebnisse ein Objekt vom Typ CDCMsg erstellt und in einem Array gequeuet, da die Messergebnisse nicht die Einheit der Messung beinhalten. Die Einheit muss durch Anfordern der „Device measurement case configuration“ Daten überprüft werden. Dies ist jedoch nur bei Drehmomentschlüsseln der Marke Holex der Fall. Bei Garant Drehmomentschlüsseln wird die Einheit in einer eigenen BLE-Nachricht mit den „Setpoint“ Daten mitgesendet. Das ist jedoch nicht korrekt, da diese Daten sich auf eine nächste Messung beziehen, was bei der Durchführung von einzelnen Messungen kein Problem darstellt, jedoch in einem „Arbeitsablauf“ mit Messungen verschiedener Einheit zu falsch interpretierten Daten führen kann. Daher muss auch beim Garant Drehmomentschlüssel die Einheit immer abgefragt werden.

Durch tiefere Analyse wurde jedoch festgestellt, dass die derzeitige Implementierung der Queue nicht threadsafe ist. Ist eine Nachricht der Queue erfolgreich abgearbeitet, wird der erste Platz im Array frei und die restlichen Nachrichten müssen jeweils ihren Platz im Array um eine Stelle verringern. Dieser Prozess kann durch ein hochpriorer Interrupt vom Softdevice gestört werden und eine Nachricht verloren gehen.

Eine vorgeschlagene Lösung dieses Problems ist, die Messergebnisse in Geräte Objekten aufzufangen und auf ihre Vollständigkeit zu warten. Dadurch würde auch die Codegröße verringert und weniger Arbeitsspeicher benötigt werden, da das Array der Queue wegfallen würde.

03 Einbindung Messuhr

Bisher ist die Dongle App daraufhin ausgelegt, dass mit einem Drehmomentschlüssel kommuniziert wird. Der Anwendungsfall erfordert es jedoch, dass der Fußschalter mit Messuhren und Messschiebern funktioniert.

Erste Tests mit einer Messuhr zeigen, dass die Verbindung zur Messuhr korrekt aufgebaut wird und ein Messergebnis bereits empfangen wird. Dieses wird jedoch offenbar falsch interpretiert, da sich im Terminalprogramm der Messwert 0.0 mit Einheit „Err“ ablesen lässt.

Ein Blick in die Protokollbeschreibung der Messuhr zeigt, dass das Messergebnis zwar innerhalb des gleichen Datenblocks wie bei dem Holex Drehmomentschlüssel liegt, 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. Zwischen Messuhr und Messschieber muss keine weitere Unterscheidung gemacht werden.

Im Code muss daher überprüft werden um welche Art von HCT-Gerät es sich handelt um die Daten korrekt zu verarbeiten. Das kann auf zwei verschiedene 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 in der Device Information andere Informationen, wie die Protokoll Version mitgeliefert werden, die bei späteren Features unter Umständen benötigt werden.

Für eine erste Implementierungsstufe ist die Zuordnung anhand des Namens ausreichend. Bei der Verarbeitung des erhaltenen Messergebnisses wird überprüft, ob es sich bei dem sendenden Gerät um einen Drehmomentschlüssel handelt. Ist dies nicht der Fall werden das binäre Messergebnis erst als int32 intrepretiert, da es sich um die Distanz in Mikrometern handelt, anschließend durch 1000 dividiert um Millimeter zu erhalten und dann im float32 der CDCMsg gespeichert.

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.

Auslösen der Abfrage des Messergebnisses

Das nächste zu implementierende Feature ist das Auslösen der Abfrage des Messergebnisses bei den Messuhren. Dazu soll der Fußschalter auf den MUX-Befehl bestehend aus Kanalnummer gefolgt von Carriage Return Linefeed reagieren. Er schickt dann den HCT-Befehl „Write Device Measurement Control“ an die Messuhr und schreibt in das Attribut „Measurement Start Condition“ eine 3 für Snap shot. Die Messuhr sendet dann die Device Measurement Result Daten und die Daten werden wie gehabt verarbeitet.

Es muss die bestehende if-Cascade zur Verarbeitung von eingehenden Daten über USB-CDC erweitert werden und dann die korrekte HCT-Nachricht verschickt werden. Dies geschieht, wie bereits bei der Abfrage der Einheit des Messergebnisses indem die Nachricht in den ReceiveBuffer gelegt wird.

04 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.

Ein Bild, das Diagramm enthält.

Automatisch generierte Beschreibung

05 Einbindung Hardware Fußschalter

Inbetriebnahme Fußschalter

Die Prototypen des Fußschalters von Firma Brecht wurden bereits erhalten und somit kann 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.

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.

06 Einbindung 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.

[Problem doppelte Zeichen]

07 Implementierung Operationsmodi

Sowohl der Fußschalter als auch der USB-Dongle sollen in mehreren verschiedenen Operationsmodi laufen können. Dieser wird in einer zusätzlichen globalen Konfigurationsdatei spezifiziert. Die Modi sind:

Kabelgebunden:

HID einzelnes Zeichen

HID

USB CDC

Kabellos (nur Fußschalter):

HID einzelnes Zeichen à Fußschalter dient zum Triggern von Abläufen

HID à Fußschalter leitet die Messergebnisse an den HID-Client weiter

Peripheral device for HCT-Windows-App

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.

08 Nachbesserung 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.

09 Dokumentation Konfigurationsfiles

Im derzeitigen Zustand befinden sich zwei Konfigurationsfiles im Massenspeichermedium. Zum einen das File „config“ in dem spezifiziert wird mit welchen Geräten sich der Dongle bzw. Fußschalter in den Modi 1 und 2 verbindet und zum anderen das File „device“, das übergeordnete Einstellungen beinhaltet.

Das „config“ File ist im CSV-Format und kann somit benutzerfreundlich mit Excel bearbeitet werden. Der Eintrag „Name“ ist der Name des Geräts der zum Beispiel in der Hoffmann Connectivity App abgelesen werden kann und muss zwingend richtig sein, damit ein Verbindungsaufbau stattfinden kann. Die Seriennummer ist meist direkt auf dem Gerät zu finden und wird ebenfalls für den Verbindungsaufbau benötigt. Die Spalte „Channel“ gibt an über welchen Kanal der Fußschalter ein Messergebnis im Modus 2 ausgibt. Die Spalte „Winkel Channel“ gibt an über welchen Kanal die Winkelmessung ausgegeben wird. Ist sie 0, wird keine Winkel Ergebnis ausgegeben. Die Spalten können umbenannt werden, müssen aber in der gleichen Reihenfolge stehen bleiben. Sollte das File in einem Editor bearbeitet werden, muss darauf geachtet werden, dass keine Leerzeichen zwischen den „;“ hinzugefügt werden.

Im „device“ File kann der Modus, in dem der Dongle bzw. Fußschalter arbeitet, angegeben werden. Zudem können folgende HID spezifische Einstellungen getätigt werden:

Value seperator: Zeichen zwischen Messergebnis und Messeinheit

Data set seperator: Zeichen am Ende eines vollständigen Messergebnisses

Number seperator: Zeichen das die Dezimalstellen abtrennt. Punkt oder Komma

Im Modus 0 gibt die Anwendung, wenn der Fußschalter betätigt wird, ein einzelnes Zeichen über HID aus. Dieses Zeichen kann im Eintrag „key“ spezifiziert werden. Sonderzeichen wie „Enter“ können hier wie bei den vorherigen Einträgen angegeben werden.

Der Eintrag „protocol“ gibt an, ob Messergebnisse im MUX50 Protokoll oder im DMX16 Protokoll ausgegeben werden. Sie unterscheiden sich vor allem darin wie viele Kanäle zur Verfügung stehen.

Zeilen die mit „#“ beginnen zeigen Kommentare an und werden nicht eingelesen. Es kann vom Benutzer zusätzliche Zeilen mit Kommentaren hinzugefügt werden. Es muss darauf geachtet werden keine zusätzlichen Leerzeichen hinzuzufügen, weder vor oder nach dem „:“ eines Eintrags, sowie nach dem Wert eines Eintrags. Nach dem Wert eines Eintrags muss stattdessen immer ein „Enter“ getätigt werden. Die Namen der Einträge (z.B. mode) dürfen nicht verändert werden, sonst kann er nicht von der Anwendung eingelesen werden. Es gibt eine Maximale Anzahl an Zeichen, die das File beinhalten darf, derzeit sind das 400 Zeichen. Sollten sie überschritten werden, kann das File nicht mehr vollständig eingelesen werden. Wurden Änderungen an den Files vorgenommen, muss die Anwendung im derzeitigen Zustand der Implementierung neugestartet werden. Das kann bei dem USB-Dongle durch Aus- und Einstecken des Dongles erfolgen. Auch braucht die Anwendung bis zu mehreren Sekunden, um die Änderung zu speichern.

10 Detektieren von Änderungen an den Konfigurationsfiles

Im derzeitigen Stand der Implementierung muss der Dongle bzw Fußschalter jedes Mal, wenn Dateien im MSC geändert wurden, resettet werden. Erst dann werden die Änderungen übernommen. Beim Fußschalter ergibt sich nun das Problem, dass das Gerät einen Akku besitzt, weswegen ein Resett durch den Anwender nicht möglich ist.

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. Problem.

In der finalen Lösung des Problems wird den Konfigurationsfiles neugelesen und über die Daten ein Hashwert gebildet. Anhand dieses Wertes wird dann eine mögliche 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.

11 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.

12 Anbindung 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 könnte folgende Werte bereitstellen:

Device Class

Protocol type, version

Version of Hardware, Software, BLE

Battery level, status

Reset

Werte der Config.ini:

Operating Mode

CDC protocol

HID Keyboard Language ID

HID value seperator

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

13 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.

Der Chip der Anwendung kann nicht heruntergefahren werden, da sonst ein Tastendruck ihn nicht aufwecken würde.

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 Leitung 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.

Uart muss abgetrennt werden, wegen HSClock

Nach Zeit der Innaktivität vergrößerung Connection Interval

14 Verbesserung Detektion Änderung der Konfigurationsdateien

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:

Startadresse FS + Sektorengröße\*Anzahl reservierter Sektoren + Sektorengröße\*Anzahl FAT\*Anzahl Sektoren pro FAT = Startadresse FS + Sektorengröße \* (Anzahl reservierter Sektoren + Anzahl FAT\*Anzahl Sektoren pro FAT)

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 Offstes 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.

15 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:

Alle zu verbindenden Geräte verbunden: Blau

Es ist mindestens ein Gerät verbunden, aber es wird nach den fehlenden Geräten gescannt: Blau blinkend

Es ist kein Gerät verbunden, Scanning aktiv: Grün blinkend

Gerät im Sleep Modus: Aus

Keine Konfigurationsfiles gefunden: Rot

Fehler in den Konfigurationsfiles: Rot blinkend

16 Gruppen Feature

Während im Modus 2 (CDC) die Messergebnisse von mehreren, durch Betätigung des Fußschalters getriggerten Messungen, anhand der Channelnummer einem Device 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 als Gruppe getriggert, werden die Messerergebnisse so sortiert, dass sie gemäß dieser Gruppenreihenfolge ausgegeben werden, wodurch sie wieder einer Messuhr zuordbar sind. Zusätzlich soll ein Feature der Messeruhren genutzt werden, um diese Gruppennummer auch auf der Messuhr anzuzeigen. Dazu wird der Messuhr ihre Gruppennummer nach dem Verbindungsaufbau via BLE übermittelt. Unvergebene Gruppenids werden übersprungen.

17 Bugfix MSC/FStorage

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. 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.

18 Aktivierung Geräte

Soll ein neues Gerät in die Konfigurationsdatei aufgenommen werden, müssen verschiedene Kenndaten der zu verbindenden Geräte in die Konfigurationsdatei eingetragen werden. Damit der User nicht verholt diese Kenndaten seiner Geräte eintragen muss, wird ein zusätzlicher Wert in das devices.csv File aufgenommen, mit dem Geräte aktiv und inaktiv geschaltet werden können.