Fachhochschule
Nordwestschweiz

Fachbericht

Team 1

Projekt 4

Fachhochschule Nordwestschweiz FHNW

11. Juni 2018

Studiengang:	Elektro- und Informationstechnik EIT
Auftraggeber/in:	Prof. Hans Gysin Jana Kalbermatter
Fachexperten:	Matthias Meier Prof. Dr. Pascal Schleuniger Pascal Buchschacher Dr. Roswitha Dubach Dr. Anita Gertiser Bonnie Domenghino
Projektteam:	Adrian Annaheim Benjamin Ebeling Jonas Rosenmund Michael Schwarz Samuel Wey Andres Minder

Inhaltsverzeichnis

1 Einleitung **1**

2 Gesamtsystem **2**

3 Software **3**
3.1 Datenverwaltung . . . 3
3.2 Auslesen, Schreiben . . . 4
3.3 Validierung . . . 4

4 Energieversorgung **6**
4.1 Technische Grundlagen . . . 6
4.2 Akku, Ladesteuerung und Spannungsregelung . . . 6
4.3 Validierung . . . 8
4.4 Fazit und Empfehlungen für das weitere Vorgehen . . . 10

5 USB **11**
5.1 Technische Grundlagen . . . 11
5.2 Konzept . . . 11
5.3 USB zu UART . . . 12
5.4 USB zu SDIO . . . 12
5.5 Validierung UART-Pfad . . . 13
5.6 Validierung SDIO-Pfad . . . 13

6 Audioausgabe **14**
6.1 Technische Grundlagen . . . 14
6.2 Konzept . . . 15
6.3 Hardware . . . 15
6.4 Firmware . . . 16
6.5 Validierung . . . 17

7 Bluetooth **19**
7.1 Technische Grundlagen . . . 19
7.2 Beacons im Museum . . . 19
7.3 IBeacons auslesen und identifizieren . . . 20
7.4 Validierung . . . 21

8 Firmware **22**
8.1 Statemachine . . . 24
8.2 Datenverwaltung . . . 25
8.3 Validierung . . . 25

9 Schlusswort **26**

10 Verzeichnisse **27**

11 Anhang **29**
11.1 Benutzeranleitung 29
11.1.1 `maketicket inventory_file preset_file` 29
11.1.2 `makepackage inventory_file preset_file package_path` 29
11.1.3 `loadsd package_path` 29

1 Einleitung

Museen haben die Aufgabe, mit Ausstellungen spezifische Themenbereiche den Interessierten zugänglich und erlebbar zu machen. Dabei soll oft ein breites Publikum vom wenig Informierten bis zur Expertin angesprochen werden. Um den Besuchenden zu ermöglichen, sich mit den persönlich relevanten Objekten auseinanderzusetzen, werden unter anderem Audioguides verwendet. Diese Audioguides werden meist zusammen mit Standard-Kopfhörern eingesetzt. Um den Hygienestandards zu entsprechen, müssen die Kopfhörer entweder aufwändig gereinigt oder Einwegkopfhörer eingesetzt werden. Der Kostenaufwand ist verhältnismässig gross, zudem bieten die bisher verwendeten Audioguides keine Möglichkeit, eine Ausstellung aufzuteilen. Eine Unterteilung der Ausstellung würde den Museen neue Angebote ermöglichen, z. B. Eintrittskarten für ausgewählte Ausstellungsbereiche mit entsprechend individualisierten Audioguides anzubieten. Dies könnte eine Museumsausstellung attraktiver gestalten.

Ziel dieser Projektarbeit war es, einen Audioguide namens Dojo mit der notwendigen Elektronik auszustatten. Das Konzept Dojo wurde bereits erarbeitet, das heisst im Rahmen dieser Arbeit stand die technische Umsetzung des Konzeptes von Dojo im Zentrum. Dojo soll eine Ausstellung eingrenzen können, indem es nur die gewählten Audiodateien abspielt und allenfalls als Zutrittsberechtigung fungiert. Die Audioübertragung vom Gerät ans Gehör soll über einen Knochenschallgeber realisiert werden, um die Unterhaltskosten der Geräte signifikant zu senken. Mit Hilfe des Audioguides soll es möglich sein, Sounddateien abzuspielen, sobald man sich in der Nähe eines Ausstellungsobjektes befindet. Dojo soll zudem eine Funktion, welche einem Like-Button ähnlich ist, besitzen. Diese Funktion soll dem Besucher und der Besucherin ermöglichen, Objekte auszuwählen und später mehr Informationen darüber z. B. in Form einer Broschüre zu erhalten. Die zu entwickelnde Elektronik muss in dem bereits bestehenden Design Platz finden.

Um die gewünschten Audiodateien freizuschalten und den richtigen Ausstellungsobjekten zuzuordnen, wird am Anfang des Besuches ein Schlüssel auf das Dojo geladen, welcher anhand einer Korrespondenztabelle die ausgewählten Dateien freischaltet. Mit Hilfe von Bluetooth-Beacons, welche im Museum bei den Objekten befestigt sind, können den Ausstellungsobjekten Audiodateien zugeordnet werden. Durch einen Vibrationsmotor wird signalisiert, dass eine Audiodatei abspielbereit ist. Die Audiodatei, welche sich auf einer μSD-Karte befindet, wird nach Betätigung des Start-Buttons über einen WTV020 Chip auf dem Knochenschallgeber abgespielt. Wenn der Like-Button während dem Rundgang betätigt wird, kann anschliessend am Ausgang über die Korrespondenztabelle eine individuelle Broschüre ausgedruckt werden oder man kann sich die Broschüre als PDF per E-Mail zusenden lassen.

Mit Hilfe des hergestellten Prototyps ist es möglich, Beacons in einem Abstand von fünf Metern korrekt zu erfassen und die zugeordnete Audiodatei abzuspielen. Auf dem Dojo können maximal 512 Audiodateien abgespeichert werden. Am Eintritt können je nach Wunsch die verschiedenen Audiofiles ausgewählt werden und für den Besuch freigegeben werden. Den Besuchern und den Besucherinnen wird die Möglichkeit gegeben, für gewünschte Objekte mit Hilfe eines Like-Buttons am Ende des Museumsrundgangs Zusatzinformationen in Form einer Broschüre zu erhalten.

Im nachfolgenden Kapitel Gesamtsystem, wird die Aufteilung des Berichtes dargelegt.

2 Gesamtsystem

Abbildung 2.1 zeigt das grobe Gesamtsystem des Dojos. Die einzelnen Kapitel dieses Berichts orientieren sich an den Blöcken und deren Funktion im Gesamtsystem. In Kapitel **3 Software** wird die Computersoftware erklärt, welche verwendet wird, um das Dojo zu konfigurieren. Das Kapitel **4 Energieversorgung** befasst sich mit dem Akku, der Ladeschaltung und -überwachung, sowie der Spannungsversorgung. Im Kapitel **5 USB** wird erläutert, wie über die USB-Schnittstelle mit dem Mikrocontroller kommuniziert wird und wie Audiofiles auf die μSD-Karte übertragen werden. Das Kapitel **6 Audioausgabe** behandelt die Verarbeitung und Ausgabe der Audiofiles. Das nächste Kapitel **7 Bluetooth** befasst sich mit den Bluetooth-Komponenten, das heisst mit den Beacons zur Erkennung der Kunstobjekte und dem Bluetoothmodul im Dojo für den Empfang. Wie die Firmware als Gesamtsystem funktioniert, ist in Kapitel **8 Firmware** dargelegt. Die Validierung der einzelnen Blöcke wird im jeweiligen Kapitel abgehandelt.

Abbildung 2.1: Blockschaltbild Gesamtsystem

Software	Energieversorgung	USB	Audioausgabe	Bluetooth	Firmware

Tabelle 2.1: Legende Gesamtsystem

3 Software

Im Kapitel Software wird erläutert, wie die Audiodateien auf das Dojo gelangen und die Korrespondenztabelle angepasst werden kann. Es wird erklärt, wie die Likes ausgewertet und verarbeitet werden können.

3.1 Datenverwaltung

Alle Ausstellungsobjekte sind in einer Textdatei aufgelistet, in der vermerkt wird, welche Audio- und Textdatei zu welchem Beacon gehört. Die Form dieser Inventardatei ist in Abbildung 3.1 zu sehen.
Zudem wird in einer weiteren Textdatei ein Preset definiert, mit der die μSD-Karte beschrieben werden kann. Die Form dieser Presetdatei ist in Abbildung 3.2 zu sehen.

```
exhibition romans{
    233(
        de  roemischerhelm.mp3  roemischerhelm.tex
        en  romanhelmet.mp3     romanhelmet.tex
    )
    10(
        de  becher_rom.mp3      becher_rom.tex
        en  cup_rome.mp3        cup_rome.tex
    )
}

exhibition middle_ages{
    15(
        de  mittelalterlicher_schaedel.mp3  mittelalterlicher_schaedel.tex
        en  skull_medieval.mp3              skull_medieval.tex
    )
    11(
        de  morgenstern.mp3         morgenstern.tex
        en  primitive_weapon.mp3    primitive_weapon.tex
    )
}
```

Abbildung 3.1: Inventardatei für eine Testausstellung

```
languages: de, en
exhibitions: romans, middle_ages
```

Abbildung 3.2: Presetdatei für eine Testausstellung

Der CLI-Befehl `makepackage` nimmt die Inventardatei und Presetdatei und generiert einen Ordner, in dem sich alle Audiodateien befinden, die in der Inventardatei aufgelisteten sind. Die

Dateien des generierten Audiopakets haben als Namen nur noch eine Zahl, die bestimmt an welcher Stelle sie auf der μSD-Karte abgespeichert wird.
Der CLI-Befehl `loadsd` nimmt das vorher generierte Audiopaket und schreibt es auf die μSD-Karte des Audioguides. Dazu schickt es dem Mikrocontroller den Befehl, dass er den Multiplexer umschaltet. Das Betriebssystem des PC sollte dann die μSD-Karte mounten und dem Python Skript verfügbar machen, worauf dieses die Daten kopiert. Es wird dabei angenommen, dass nur ein Audioguide am PC angeschlossen ist und dass alle μSD-Karten "dojo-sd"heissen.
Der CLI-Befehl `maketicket` nimmt die Inventardatei und die Presetdatei als Argumente und fragt den Nutzer, welche Ausstellungen dem Besucher in welcher Sprache zur Verfügung stehen sollen. Mit diesen Inputs wird ein Ticket generiert, das über die serielle Schnittstelle auf das EEPROM des Mikrocontrollers geladen wird.
Auf dem EEPROM des Dojos sind jedem Beacon zwei Bytes zugeordnet, die die Beacon-ID, Like- und Zutrittsinformation beinhalten. Die Likes sind zu Beginn des Besuchs alle auf null und können vom Besucher gesetzt werden. Die Zutrittsbits der bezahlten Ausstellungen werden zu Beginn des Besuchs gesetzt und können vom Besucher nicht verändert werden.

3.2 Auslesen, Schreiben

Das EEPROM des Mikrocontrollers wird über eine serielle Schnittstelle beschrieben, wobei alle Befehle das gleiche Muster haben:

```
[1 char] op-code | [3 char] arg1 | [3 char] arg2 ;
```

Wert x=10 in Addresse y=2 schreiben:	`p002010;`
Wert x aus Addresse y=60 lesen:	`r060000;`

Um ein Byte mit Wert 42 in die Adresse 2 zu schreiben, muss der String `p002042` auf die serielle Schnittstelle geschrieben werden. Um das Byte an der Adresse 2 auszulesen, muss der String `r002000;` auf die serielle Schnittstelle geschrieben werden. Wichtig ist, dass alle Befehle mit einem Semikolon terminiert werden.
Das Python Modul beinhaltet neben Schreib- und Lesefunktionen auch noch Funktionen, die ein ganzes Ticket auf das EEPROM schreiben bzw. wieder einlesen können. Hier ist es wichtig, dass das vordefinierte Muster des EEPROMs in Abbildung 3.3 eingehalten wird.

Anzahl gespeicherter Beacons	Sprachcode
ID Beacon1	Like Beacon1, Zutritt Beacon1
ID Beacon2	Like Beacon2, Zutritt Beacon2
⋮	⋮

Abbildung 3.3: EEPROM Adressierungsmuster

3.3 Validierung

Für die Validierung der Datenverwaltung wurden Python Unittests eingesetzt, die mit einer Testdatenbank alle Funktionen testen. Neben den einzelnen Funktionen wurde auch ein kompletter Durchlauf mit der Testdatenbank gemacht.

Die EEPROM-Operationen werden mit einem Arduino Uno-Board getestet, wobei das EEPROM vom PC aus beschrieben und wieder eingelesen wird. Die Unittests stellen sicher, dass das EEPROM die gewünschten Daten beinhaltet und dass diese mit der spezifizierten Geschwindigkeit übertragen werden.
Es werden zudem Benchmarks durchgeführt, welche sicherstellen, dass nicht unnötig gewartet wird. Ein Test, der misst wie lange das Beschreiben von verschieden langen Tickets dauert, zeigt, dass das Beschreiben eines vollen Tickets mit 250 Beacons $6.17s$ dauert.

4 Energieversorgung

Wie in allen Bereichen des vorliegenden Projekts geht es auch im Bereich der Energieversorgung darum, aus dem verfügbaren Raum, kosteneffizient eine möglichst gute Performance zu erhalten. Um dies zu erreichen, muss der Energiebedarf der Elektronik auf ein Minimum reduziert werden. Mit der Auswahl von energieeffizienten Komponenten und einem effizienten Ablauf im Programmcode, kann dem Energieverbrauch entgegengewirkt werden. Der Energiebedarf von Komponenten, die gerade nicht in Verwendung sind, kann so erheblich reduziert werden. Wird zum Beispiel der Mikrocontroller nicht aktiv verwendet, versetzt er sich in einen sparsamen Bereitschaftsmodus, der einen wesentlich geringeren Verbrauch hat.

4.1 Technische Grundlagen

Die maximale Grösse des Akkus war durch das designte Gehäuse bereits vorgegeben. Somit konnte eine einfache Auswahl getroffen werden. Um den Energiebedarf des Dojos abzuschätzen, wurde zu Beginn der Projektarbeit eine Energiebedarfsberechnung anhand von Datenblattangaben der verwendeten Komponenten gemacht. Die Berechnungen haben gezeigt, dass das Wunschziel, die Akkulaufzeit des Dojos auf einen ganzen Museumstag auszulegen, ins Auge gefasst werden kann. Dies führt im Museum zu einer hohen Verfügbarkeit der Geräte. Das Museum benötigt so wesentlich weniger Geräte und muss nicht ständig die Geräte auswechseln und aufladen. Ausserdem verringert sich der Arbeitsaufwand, da die Dojos nicht in jeder freien Minute ins Ladegerät gesteckt werden müssen, sondern ein Ladezyklus über Nacht ausreicht.

Die ursprüngliche Energiebedarfsberechnung der wichtigsten Komponenten hat bei einer durchschnittlichen Nutzung von zehn Stunden einen Speicherbedarf von $400mAh$ ergeben. Es wurde angenommen, dass das Dojo während einem Drittel der Zeit aktiv genutzt und in der übrigen Zeit in Standby-Modus versetzt wird.

4.2 Akku, Ladesteuerung und Spannungsregelung

Aufgrund der Voraussetzungen wurde bei der Suche auf Akkus des Typen LI14500 fokussiert, Lithium-Akkus mit einem Durchmesser von $14mm$ und einer Länge von $50mm$. Diese Standardgrösse ist für das vorgegebene Gehäuse etwas zu gross, weshalb es etwas angepasst werden muss. Die Grösse solcher Akkus mit den Anforderungen der Energiekapazität sind auf dem Markt auffindbar.

Das Dojo wird über den integrierten USB-Anschluss aufgeladen. Die Ladesteuerung ist Aufgabe des MCP7383 von Microchip . Dieser Ladesteuerungs-IC benötigt wenige zusätzliche Bauteile und ist daher ideal für portable Geräte wie das Dojo geeignet, er arbeitet nach dem Prinzip constand-current / constant-voltage. Die Ladeschaltung ist auf der Hauptplatine integriert. Die verwendete Standardbeschaltung ist auf Abbildung 4.1 ersichtlich, die Ladekurve ist in Abbildung 4.2 dargestellt. In der Ladekurve ist gut zu erkennen, wie die Spannung gegen Ende der Ladung konstant gehalten und der Strom reduziert wird.

Abbildung 4.1: Schema der Ladeschaltung

Abbildung 4.2: Ladekurve des Emmerich-Akkus

Um die interne Spannungsversorgung auf einem konstanten Niveau zu halten, ist dem Akku der LDO (Low Drop-Out) TC1262 von Microchip nachgeschaltet. Dieser regelt die Betriebsspannung auf $3.3V$, liefert bis zu $500mA$ und ist selbst sehr energiesparend.

4.3 Validierung

Nach Empfang des Emmerich-Akkus konnte dieser erfolgreich mit dem angefertigten Prototyp des Dojo-Prints in Betrieb genommen werden. Um die tatsächliche Kapazität des Akkus zu bestimmen und um zu verifizieren, ob die Lieferantenangabe stimmt, wurde der Akku in verschiedenen Durchläufen getestet. Die Tests wurden mit dem "Battery Charging Centre ALC 8500-2 Expert" von der ELV Elektronik AG durchgeführt (Abbildung 4.3). Zu diesem Testgerät gibt es eine PC-Software (Abbildung 4.4), mit der Parameter wie die Art des Akkus, Kapazität, Spannung aber auch das gewünschte Lade- / Entladeprofil eingestellt werden können. Während dem Vorgang, also dem Laden oder Entladen des Akkus, speichert das Gerät alle fünf Sekunden die Werte der aktuellen Spannung, des aktuellen Stromes sowie der summierten Kapazität in mAh. Anschliessend können die abgespeicherten Werte zur weiteren Auswertung über die PC-Software in eine Tabelle exportiert werden.

Abbildung 4.3: Das verwendete Akkutestgerät ALC 8500 Expert

Abbildung 4.4: Bildschirmausschnitt der PC-Software zum Akkutestgerät

Der Akku wurde mit unterschiedlichen Entlade-/Ladezyklen getestet. Für den geprüften Akku von Emmerich konnte bei Entladung mit einer anfänglichen Spannung von $4.1V$ eine Kapazität von $827mAh$ gemessen werden bis der Tiefentladeschutz die Entladung bei $2.7V$ gestoppt hat. Bis zum Erreichen der internen Versorgungsspannung von $3.3V$, die im Betrieb des Dojos gehalten werden muss, konnte eine Kapazität von immerhin noch $788mAh$ festgestellt werden. Mit diesen gemessenen Werten stimmt der Akku mit den Datenblattwerten überein. Eine Entladekurve wird in Abbildung 4.5 gezeigt.

Abbildung 4.5: Entladekurve des Emmerich-Akkus bei konstantem Entladestrom

Neben der Kapazitätsüberprüfung des Akkus wurden auch die anfänglich nach Datenblattangaben berechneten Strombedarfswerte verifiziert. Dazu wurden am Prototyp Strommessungen durchgeführt. Die Strommessung direkt am Akku hat ergeben, dass im Standby-Betrieb, wenn also der Knochenschallgeber nicht in Betrieb ist, ein Strom von $35mA$ fliesst. Ist der Knochenschallgeber in Betrieb, steigt der Stromfluss auf $130mA$. Beim Test wurde eine Sprachdatei abgespielt, die Lautstärke war eher am oberen Limit.

Die Berechnung mit den gemessenen Werten zeigt, dass die angestrebten Ziele mit dem Prototyp erreicht werden. Die Speicherkapazität des eingesetzten Akkus reicht für einen Betrieb während zehn Stunden, bei einer aktiven Nutzung während 40% der Zeit, also vier Stunden reine Abspielzeit und sechs Stunden Standby-Betrieb.

Des Weiteren wurde das Ladeverhalten des Akkus, wenn er am Prototyp angeschlossen ist, überprüft. Nach dem Einstecken des USB-Kabels fliesst während den ersten zehn Sekunden ein Strom von $330mA$, danach erhöhte sich der Stromfluss bei der Versuchsmessung auf $406mA$. Die Anhebung des Ladestromes erfolgt in einem Schritt und nicht allmählich über die Zeit. Anschliessend reduzierte sich der Strom über die Zeit allmählich, da bereits die spannungsbedingte Strom-

begrenzung erreicht war. Als Gegenstück diente ein USB 3.0-Anschluss eines Laptops.

Abbildung 4.6: Anordnung für die Strommessung am Akku

4.4 Fazit und Empfehlungen für das weitere Vorgehen

Wie mit den vorhergehenden Berechnungen gezeigt, konnte das Ziel erreicht werden, Energiereserven für einen ganzen Betriebstag im Dojo zu speichern. Allerdings wurde ein Akku verwendet, der mit dem LI14500-Standard etwas zu gross ist für das vorgegebene Gehäusedesign. Falls bei der Weiterentwicklung der vorliegenden Dojo-Elektronik weitere Einsparungen im Energiebedarf realisiert werden können, kann der Wechsel auf einen Akku der nächstkleineren Bauart geprüft werden. Der gewählte Akku stellte sich als gut und den Herstellerangaben entsprechend heraus. Für eine Serienproduktion des Dojos ist der Preis des Akkus von Emmerich mit CHF 13.95 auf jeden Fall zu hoch. Der Preis müsste durch Verhandlung und Mengenrabatte mindestens um die Hälfte reduziert werden. Allenfalls ist in Erwägung zu ziehen, ob eine reine Speicherzelle verwendet werden soll, die keine integrierte Schutzelektronik besitzt, denn in der im Rahmen dieses Projektes entwickelten Elektronik ist ebenfalls eine Schutzbeschaltung enthalten. Die Überladeschutzschaltung hat sich im Praxistest bewährt. Sie lässt den angeschlossenen Akku mit einer zuverlässigen Ladekurve aufladen. Auch die Schaltung für das Bereitstellen der Versorgungsspannung hat sich bewährt und die Elektronik mit einer konstanten Spannung versorgt.

5 USB

Im folgenden Kapitel wird erklärt, wie über die USB-Schnittstelle Daten auf den Dojo geladen werden und ausgelesen werden. In einem Unterkapitel wird sogleich die Validierung beschrieben.

5.1 Technische Grundlagen

Nachfolgend werden einige Begriffe zur Datenübertragung kurz erklärt, welche im Kapitel von Bedeutung sind.

UART Universal Asynchronous Receiver Transmitter (UART) realisiert eine digitale serielle Schnittstelle. Über die UART-Schnittstelle können Daten über einen Datenstrom gesendet und empfangen werden. Die Daten sind in einem fixen Rahmen, bestehend aus Start-Bit, fünf bis neun Datenbits, optionalem Parity-Bit zur Fehlererkennung und einem Stopp-Bit [1].

SDIO Secure Digital Input Output (SDIO) bezeichnet ein Interface für die Datenübertragung zwischen SD-Karten. Die Daten werden wahlweise im SPI, 1-Bit oder im 4-Bit Modus übertragen.

5.2 Konzept

Zur Kommunikation mit dem Mikrocontroller im Gerät besteht eine USB-UART-Schnittstelle. Um neue Audio-Dateien auf die SD-Karte zu schreiben, wurde ausserdem eine USB-SDIO-Schnittstelle realisiert. Folgende Abbildung 5.1 zeigt das Konzept mit den beiden Schnittstellen.

Abbildung 5.1: Konzept USB

Eine micro USB-Typ B Schnittstelle ist die gemeinsame Schnittstelle nach aussen. Dahinter schaltet ein USB-Switch, gesteuert durch den Mikrocontroller, einen der beiden Pfade durch. Es kann jeweils entweder der SDIO-Pfad oder UART-Pfad durchgeschaltet werden, nie beide Pfade gleichzeitig.
Als USB-Switch wird der Baustein TS3USB30E von Texas Instruments verwendet. Dieser Chip wurde dafür entwickelt, um high-speed USB 2.0 Signale in portablen Geräten und Konsumelektronik zu schalten. Gesteuert wird der USB-Switch über zwei Pins. Über den Enable-Pin kann der Baustein in einen hochohmigen Zustand gebracht werden, um den Bus nicht zu belasten und den Stromverbrauch zu senken. Mit dem Select-Pin wird der durchzuschaltende Pfad ausgewählt [2].
Folgende Tabelle 5.1 zeigt die Wahrheitstabelle für die entsprechenden Pfade:

Select (Pin24)	$\overline{Enable}$ (Pin23)	Funktion
X	HIGH	hochohmig
LOW	LOW	Pfad = UART
HIGH	LOW	Pfad = SDIO

Tabelle 5.1: Wahrheitstabelle USB-Pfade

5.3 USB zu UART

Der Baustein FT231X von FTDI stellt das Interface von USB zu UART bereit. Das ganze USB-Protokoll wird dabei auf dem Chip abgehandelt und es sind kaum zusätzliche Bauteile nötig (siehe Schema im Anhang). Über zwei Pins (Rx und Tx) werden die Daten seriell übertragen.
Bei aktivem UART-Pfad wird der Dojo am Computer als serielle Schnittstelle erkannt. Über diese Schnittstelle kommuniziert nun die Software auf dem Computer mit dem Mikrocontroller im Gerät, um beispielsweise die ‹Likes›auszulesen oder die Audiofileübertragung einzuleiten.

5.4 USB zu SDIO

Der VUB300 Chip vom Hersteller elan stellt das USB zu SDIO Interface bereit. Damit kann die SD-Karte im Dojo über den USB-Anschluss beschrieben und ausgelesen werden. Wird mit dem USB-Switch der SDIO-Pfad durchgeschaltet, erscheint die SD-Karte am Computer als Datenträger und kann gelesen sowie beschrieben werden. Zu beachten ist, dass zwischen dem VUB300 Chip und der SD-Karte noch ein weiterer Multiplexer (MUX) eingebaut ist. Mit diesem wird die SD-Karte zwischen dem VUB300 und dem Audio-Chip umgeschaltet. Genaueres dazu findet man in Abschnitt **6 Audioausgabe**. Damit die SD-Karte mit dem SDIO-Pfad verbunden ist, muss der MUX TS3A27518 (siehe Schema im Anhang) folgendermassen angesteuert werden:

$\overline{Enable}$ (Pin25)	IN1 (Pin26)	IN2 (Pin27)	Funktion
HIGH	X	X	hochohmig
LOW	LOW	LOW	Pfad = SDIO(VUB300)
LOW	HIGH	HIGH	Pfad = Sound-Modul

Tabelle 5.2: Wahrheitstabelle SD-Pfade

5.5 Validierung UART-Pfad

Um die korrekte Funktion der seriellen Schnittstelle zwischen dem USB-Anschluss und dem Mikrocontroller zu überprüfen, wurde eine kleine Testsoftware auf den Mikrocontroller geladen. Sobald der Mikrocontroller am RX-Pin Zeichen empfängt, sendet er diese über den TX-Pin wieder zurück. Mit dem Serial Monitor der Arduino IDE wurden die Zeichenketten erfolgreich gesendet und empfangen.

5.6 Validierung SDIO-Pfad

Mit einem weiteren Testprogramm wurde die Funktionsfähigkeit des SDIO-Pfades überprüft. Dafür wurde der USB-Switch und der MUX mit dem Mikrocontroller entsprechend angesteuert. Die SD-Karte wurde daraufhin von einem Computer mit Linux Betriebssystem als Datenträger erkannt und konnte erfolgreich ausgelesen und beschrieben werden. Leider stellte sich nach weiteren Tests heraus, dass auf einem Windows-Rechner keine SD-Karte erkannt wird. Die Gründe dafür sind nicht bekannt und das Problem konnte auch nicht behoben werden. Somit muss die Computer-Software auf einem Linux Betriebssystem gestartet werden.

Anmerkung
Durch die begrenzte Verfügbarkeit des VUB300 Chips, sowie der geringen Informationsdichte über den Baustein und dem Kompatibilitätsproblem mit Windows, ist dieser nicht für weitere Projekte oder Serienproduktionen zu empfehlen.

6 Audioausgabe

In diesem Kapitel werden die technischen Grundlagen welche sich auf das verwendete Audio-Modul beziehen, das Konzept sowie die Funktion der Audioausgabe beschrieben. Zudem wird die Validierung erläutert.

6.1 Technische Grundlagen

Das Konzept von Dojo sieht als Abspielgerät der Audiodateien einen Bone Conductor vor. Der Bone Conductor sowie das für den Prototyp verwendete Audio Modul WTV020 werden in diesem Kapitel beschrieben. Diese Informationen werden in den nachfolgenden Kapiteln benötigt.

WTV020 Das WTV020 Modul ist ein Soundmodul welches es ermöglicht, Audiodateien auf einem Aktor abzuspielen. Auf einer maximal 1GB grossen μSD-Karte können bis zu 512 Audiodateien abgespeichert werden. Die Audiodateien auf der μSD-Karte müssen jedoch dem .wav oder .ad4 Format entsprechen. Die Dateien müssen gemäss Vorgabe: 0000; 0001; 0002; ... nummeriert werden.
Der WTV020 Chip kann in zwei verschiedenen Modes betrieben werden, dem MP3 Mode und dem Two Line Serial Mode. Im MP3 Mode können direkt 6 Pins angesteuert werden. Durch die sechs Pins können folgende Funktionen umgesetzt werden: Reset, ±Volume , next, previous und play, pause. Für das Dojo wird jedoch der two line serial mode genützt. Dieser kann das Modul mit nur 3 Pins betreiben. Der Mikrocontroller muss an den Clock-, den Data- und den Busy-Pin angeschlossen werden. Im two line serial mode können die Audiodateien welche sich auf der μSD-Karte befinden abgespielt werden. Er ermöglicht zudem, ähnlich wie im MP3 Mode ein Lied zu pausieren und neu zu starten, sowie eine Lautstärkenregulation. [3]

Bone Conductor Die Audioausgabe erfolgt wie beschrieben über einen Bone Conductor. Der Bone Conductor besteht aus einem kleinen Metallstab, welcher mit einer Kupfer Spule umwickelt ist. Sobald ein pulsförmiger Strom durch die Spule fliest, dehnt sich ein Magnetfeld aus welches die benötigten Vibrationen auf ein flaches Metallstück auslöst.

Abbildung 6.1: Bone Conductor [?]

In der Abbildung 6.1 ist der für den Dojo benötigten Bone Conductor abgebildet. Der Bone Conductor ermöglicht es durch die Vibrationen, dass eine Audio Datei über den Schädelknochen

abgespielt wird und so nur für eine Person hörbar ist, diese jedoch immer noch die Umgebungsgeräusche wahrnimmt. [?]

6.2 Konzept

Das Konzept der Audioausgabe ist wie folgt aufgebaut:

Abbildung 6.2: Audio Konzept

Die auf einer μSD-Karte abgespeicherten Audiodateien werden über einen Multiplexer, welcher vom Mikrocontroller gesteuert wird, an einen WTV020 Chip übertragen. Dieser wird in einem Serial Mode 6.1 betrieben und kann ebenfalls vom Mikrocontroller angesteuert werden. Der Audiochip entschlüsselt die Daten und gibt diese an einen Klasse D Verstärker weiter. Dieser wird benötigt, um eine gut hörbare Lautstärke zu erreichen. Das Audiosignal wird schliesslich an einem Bone Conductor ausgegeben. Nachfolgend wird die Anordnung auf dem Print der einzelnen Komponenten dargelegt.

6.3 Hardware

Wie im Konzept 6.2 beschrieben, muss man, um Daten von der μSD-Karte auszulesen zuerst den Multiplexer ansteuern. Dies geschieht über den Mikrocontroller. Im ungesteuerten Zustand ist der MUX auf den WTV020-Chip geschaltet. Der WTV020SD-20S Chip kann nun auf die μSD-Karte zugreifen. Über vier Pins werden die Audiofiles ausgelesen und weitergegeben. Das Audiosignal wird auf den Klasse-D Verstärker gegeben. Dieser ist wie folgt aufgebaut.

Abbildung 6.3: Klasse D Verstärker

Um ein Rauschen an der Speisung zu vermeiden, wird ein 1μ Farad grosser Kondensator benötigt. In diesem Projekt wurde zuerst eine konstante Verstärkung mit dem Faktor ca.10 verwendet, da die Lautstärken Regelung über den WTV020-Chip geregelt werden soll. Dieser Faktor entsteht aus dem Verhältnis des Internen Widerstandes und den zugeschalteten Widerständen. Um das Rauschen der Eingänge, sowie zu hohe Frequenzen zu filtern wurden zuerst wie die Abbildung 6.3 zeigt, Kapazitäten eingeplant. Auf dem Prototyp wurden aufgrund von Tests welche in der Validierung aufgeführt werden, die Vorwiderstände durch Kapazitäten ersetzt und die geplanten Kapazitäten weggelassen.
Wie erwähnt wird die Lautstärkenregelung über den WTV020-Chip geregelt. Dies ist jedoch im serial mode fehleranfällig. Übergangsweise wurde um die Lautstärke zu dämmen ein 50 Ohm grosser Widerstand vor den Bone Conductor geschaltet. Die Audioausgabe kann also wie beschrieben mit dem Bone Conductor umgesetzt werden.

6.4 Firmware

Wie im Kapitel Hardware beschrieben muss der MUX angesteuert werden um dem WTV-Chip den Zugriff auf die μSD-Karte zu gewähren. Im Kapitel 5 in der Tabelle 5.2 ist beschrieben wie der MUX angesteuert werden muss, damit die benötigten Pins durchgeschaltet sind. Für die Audioausgabe müssen alle Öffner geschlossen werden. Dies wird über die Pins 25 – 27 (Port C) realisiert. Sobald Musik abgespielt werden muss, werden die benötigten Ausgänge gesetzt. Das WTV020 Modul vergleicht die Daten und Clock Pins wie folgt :

Abbildung 6.4: WTV Serial Mode

In der Abbildung 6.4 ist ein three line serial mode dargestellt. Für den benötigten two line serial mode wird der CS Pin nicht benötigt.

Sobald am Pin 7 über einen Taster Play ein low Signal erkannt wird, wird folgende Funktion aufgerufen:

```
void sendWTVcommand(unsigned int command){
digitalWrite(WTV_CLK, LOW);
_delay_us(1900);
for (byte i = 0; i < 16; i++)
{
_delay_us(100);
digitalWrite(WTV_CLK, LOW);
digitalWrite(WTV_DOUT, LOW);
if ((command & 0x8000) != 0)
{
digitalWrite(WTV_DOUT, HIGH);
}
_delay_us(100);
digitalWrite(WTV_CLK, HIGH);
command = command<<1;
}
}
```

Abbildung 6.5: Audio Datei abspielen [4]

Die benötigten Befehle welche sendWTVcommand mitgegeben werden müssen sind aufgrund des WTV020-Chips vorgefertigt. Nachfolgend werden die verwendeten Befehle und deren Funktion aufgeführt.

Funktion	Befehl
Play_Pause	0xFFFE
STOP	0xFFFF
VOL	0xFFF0-0xFFF7

Tabelle 6.1: WTV020 Funktionen

Bei der Funktion Play_Pause ist zu erwähnen, dass um die Ausgabe zu starten am Anfang die gewünschte File Nummer gesendet werden muss (0-512). Ohne File Nummer wird die aktuelle Audioausgabe pausiert und kann wieder gestartet werden.

6.5 Validierung

Um die Audioausgabe zu testen, wurden verschiedene Versuche durchgeführt. Zuerst wurde der Bone Conductor über eine kleine Verstärkerschaltung direkt an einem Laptop angeschlossen, um die benötigte Leistung und die Lautstärke abschätzen zu können. Bei diesen Messungen ergab sich, bei einer sehr gut hörbaren Lautstärke, eine maximale Scheinleistung von 0.28 VA.
Um das WTV020 Modul mit geringem Aufwand zu testen wurde dieses zuerst im MP3 Mode betrieben. So konnten schon am Anfang nicht kompatible μSD-Karten aussortiert werden.
Alle Funktionen des WTV020-Chips wurden separat und im two line serial mode überprüft. Alle

verlangten Funktionen des Chips laufen Wunschgemäss. Zudem wurde ein Versuchsaufbau mit dem D-Klasse Verstärker durchgeführt, mit welchem die aktuell maximal benötigte Leistung von 1W gemessen wurde.
Trotz der funktionierenden Tests ist aktuell am Prototyp die Lautstärkenregelung nicht möglich. Das Audiomodul übernimmt die eingestellte Lautstärke, setzt diese jedoch, sobald das Gerät an einen Verstärker angeschlossen ist, vorherigen Wert zurück. Zudem ist zu erwähnen, dass für eine Massenproduktion ein WTV020 Modul nicht geeignet wäre, aufgrund der erwähnten Eigenheiten. Das Modul nimmt trotz gleicher Formatierung und gleichem Hersteller nicht jede μSD-Karten an. Zudem treten beim serial mode immer wieder kleiner Ungereimtheiten auf, wie z.B. die Lautstärkenregelung.

7 Bluetooth

In diesem Kapitel wird dargelegt, wie die Kommunikation über Bluetooth im Gerät stattfindet.

7.1 Technische Grundlagen

Da eine drahtlose Verbindung mit Beacons hergestellt werden muss, wird dazu ein Bluetooth-Modul, das HM-11, verwendet. Es ermöglicht das Übertragen der Identifikationsnummer vom Beacon auf das Dojo. Der Vorteil eines vorgefertigten Moduls liegt in der einfachen Ansteuerung Hardware-, sowie Firmware-mässig.

Receiver Signal Strength Indicator (RSSI)

Der RSSI ist ein dimensionsloser Wert. Er beschreibt die empfangene Leistung bei kabelloser Kommunikation. Angegeben wird der Wert in dBm, was das Verhältnis in Dezibel zwischen der Leistung mW und der Referenzleistung 1 mW darstellt.

Technische Daten

Das HM-11 Bluetooth-Modul benötigt eine Versorgungsspannung von 3.3 V DC und verfügt über das Kommunikationsprotokoll UART. Die Standard Baudrate beträgt 9600bps und nimmt beim Senden einen Strom von 15 mA auf. In offener Umgebung beträgt die Reichweite bis zu 30 m und unterstützt AT Kommandos, um das Bluetooth-Modul zu konfigurieren.

Bluetooth-Konfiguration

Das verwendete HM-11 Bluetooth-Modul kann im Auslieferungszustand noch nicht wunschgemäss verwendet werden. Um eine reibungslose Kommunikation zu gewährleisten, müssen einige Konfigurationen im Bluetooth-Modul vorgenommen werden. Das Bluetooth-Modul wird über die serielle Schnittstelle, einem FTDI Chip, mit dem Computer verbunden. Mit den AT-Kommandos wie AT+Name, AT+ROLE1, AT+IMME1 und AT+DISI? kann das Bluetooth-Modul nun eingestellt werden. Falls das Bluetooth-Modul bereits auf dem Print bestückt ist, kann das Modul über die UART-Schnittstelle konfiguriert werden.

7.2 Beacons im Museum

Beacons sind kleine und kompakte Bluetooth-Sender die auf der Low Energy Spezifikation basieren. Sie können entweder batteriebetrieben oder mit permanentem Stromanschluss an Räumen oder Objekten installiert werden. Bewegt sich ein Museumsbesucher in die Nähe eines Beacons mit diesem Dojo, so kann er Audiobeiträge empfangen und mit dem Dojo abspielen. Zudem kann an der Kasse des Museums entschieden werden, welche Zonen aktiviert werden sollen. Das Dojo beinhaltet somit auch die Zutrittsberechtigungen. Der Vorteil dieses Systems ist der Preis. Der Kunde bezahlt nur, was ihn auch interessiert.

Minew E7

Mit dem Beacon Minew E7 konnte ein Museum simuliert werden, um den Prototypen auf die Funktionalität zu prüfen. Dieser Beacon lässt sich über die gratis BeaconSET+ App konfigurieren. Dieser Beacon bringt einige Vorteile mit sich. Zum einen kriegt er eine maximale Reichweite von 100 m hin, doch noch viel wichtiger ist das Protokoll. Der Minew E7 unterstützt iBeacon und Eddystone Protokolle die entweder separat oder gleichzeitig benutzt werden können.

Protokollbeschreibung

Das iBeacon-Protokoll wurde von Apple entwickelt und baut auf der Bluetooth-Low-Energy-Spezifikation auf. Das Paket das ausgestrahlt wird hat eine Länge von 30 Byte. In diesem Paket stecken die ganzen Informationen wie Geräte-ID, Beacon-Typ, Referenzempfangsleistung und noch viele weitere Daten drin. Um die Beacons zu scannen, wurde die Software so geschrieben, dass es nur die UUID, den Majorvalue und den RSSI-Wert benötogt. Nachfolgend wird der Aufbau des iBeacons-Paket graphisch dargestellt.

Abbildung 7.1: Aufbau eines iBeacon-Pakets

7.3 IBeacons auslesen und identifizieren

Mittels der Funktion scan() im State SCAN wird mit dem hm-11 nach den verfügbaren IBeacons in der Umgebung gesucht. Über eine serielle Schnittstelle (UART) wird vom MCU der Befehl AT+DISI? dem hm-11 geschickt, wobei als Antwort dann alle umliegenden IBeacons in einem String zurückkommen (siehe Abbildung 7.2). Dafür wird vom MCU jeder char, rsp. jedes

```
Send: AT+DISI?
Recv: OK+DISCS (Scan start)
Recv: OK+DIS[P0:P1:P2:P3:P4] (if have one device)
Recv: OK+DIS[P0:P1:P2:P3] (if have two devices)
...........
Recv: OK+DISCE (Scan end)
```

Abbildung 7.2: P0: Factory ID (8 Byte); P1: IBeacon UUID (32 Byte); P2: Major-, Minorvalue, Measured Power (10 Byte); P3: Media-Access-Control-Adresse MAC (12 Byte); P4: RSSI (4 Byte) [5]

Byte vom Datenbuffer ausgelesen und verwertet. Es wird zuerst nach einer UUID eines IBeacons

gefiltert und dann die ersten drei Zahlen in einen Integer gecastet. Anschließend soll das Majorvalue[1], welches einen bestimmten, selbst wählbaren Wert hat, abgeglichen werden, damit nur die IBeacons berücksichtig werden, die relevant sind. Ist der empfangene RSSI-Wert grösser als -90dbm, wird der IBeacon eingespeichert. Dies wird mit jedem empfangenen IBeacon wiederholt und immer die RSSI-Werte miteinander verglichen, wobei dann der IBeacon mit dem grösseren RSSI-Wert ins System gespeichert und als Returnvalue von der scan() Funktion zurückgegeben wird.

7.4 Validierung

Für die Validierung wurden zwei IBeacons mit Androidhandys simuliert (Beacon Simulator App). Dafür wurden diese mit einer Distanz von ungefähr $6m$ zueinander auf einer Höhe von ca. $2m$ platziert. Um die detektierten IBeacons direkt auszuwerten, wurde die UUID auf einen Emulator (Putty) über eine serielle Schnittstelle (USB Typ micro B) herausgeschrieben. Somit konnte verifiziert werden, dass der vom Dojo detektierte IBeacon auch der sich am Dojo nächsten befindliche IBeacon war.

Eine definitive Distanzbestimmung zwischen dem IBeacon und dem Dojo ist schwierig, da die Sendeleistung der simulierten IBeacons etwas schwanken. Auch die Auslegung des Raumes, sowie die Einrichtung kann das Signal abschwächen, was direkten Einfluss auf die Erkennungsdistanz hat. Es kann aber festgehalten werden, dass ein IBeacon in einem Umkreis von $3m$ garantiert erkennt wird, solange sich keine Gegenstände unmittelbar zwischen den beiden Objekten befinden.

[1] Major- und Minorvalues dienen hauptsächlich zur zusätzlichen Identifikation eines IBeacons

8 Firmware

Die Firmware wurde auf einen ATMega 328p Mikrochip geflashed (siehe Abbildung 8.1). Für die Kommunikation (UART) mit dem Bluetoothmodul ist eine zusätzliche Softwareserial (Pins PD2 & PD3) zur üblichen seriellen Schnittstelle (Pins PD0 & PD1) implementiert, da der ATMega 328p nur ein UART Interface (TXD & RXD) auf den Pins PD0 & PD1 hat. Der ATMega 328p wird mittels einem ISP (AVR-Dragon) über den sechs poligen ICSP-Header (Anhang: Schema Sheet: 3/3) JP1 geflashed.
Folgende Einstellungen wurden im Device Programming unter Fuses im AtmelStudio dafür vorgenommen:

- EXTENDED.BODLEVEL: Brown-out detection disabled
- HIGH.SPIEN: enabled
- HIGH.BOOTZ: Boot Flash size=2048 words start address=$3800
- LOW.SUT_CKSEL: Ext. Crystal Osc. 8.0- MHz; Start-up time PWRDWN/RESET: 1K CK /14 CK + 65ms
- EXTENDED: 0xFF
- HIGH: 0xD9
- LOW: 0xCF

Abbildung 8.1: ATMega 328p [6]

Die Firmware ist so designed, dass alle $\sharp defines$, die extern foreward declarated Funktionen und alle weiteren inkludierten Headerfiles in der Dojo.h Datei implementiert sind. Diese wird dann von der Dojo_Functions.cpp und der Sketch.cpp Datei eingebunden. In der Abbildung 8.2 ist der allgemeine Aufbau der Firmware in einem UML-Diagramm visualisiert.

Abbildung 8.2: UML-Diagramm Firmware

In der Tabelle 8.1 ist die Auslastung des ATMegas mit der Endversion der geflashten Firmware dargestellt. Diese ist relativ niedrig und bietet für einen weiteren Aufbau genügend freie Memory.

Program Memory Usage:	5064 bytes	15.5 % Full
Data Memory Usage:	429 bytes	20.9 % Full

Tabelle 8.1: Auslastung des ATMega 328p

8.1 Statemachine

Auf dem Mikrocontroller läuft eine Mealy-Statemachine mit sechs Zuständen (siehe Abbildung 8.3):

INIT: Hier wird das setup() der Firmware ausgeführt. Dabei werden alle nötigen Pinkonfigurationen initialisiert und und das WTV-Modul konfiguriert.

SCAN: Es wird vom Dojo nach IBeacons in naher Umgebung gesucht. Falls einer gefunden wurde, dann wird direkt das dazugehörige Audiofile bereitgestellt. Wird der Playbutton gedrückt, ändert sich der State zu PLAY. Ansonsten wird weiter gescanned.

PLAY: Hier werden die bereitgestellten Audiofiles abgespielt. Wird der Playbutton nochmals gedrückt, wird das Abspielen abgebrochen und der State wechselt wieder zu SCAN.

Wird das Dojo an den Computer angeschlossen, ist der Zustand des Dojos abhängig von der gewünschten Tätigkeit. Dafür wird vom Computer aus ein Befehl über das USB-Kabel gesendet, woraufhin sich der State ändert:

GET_LIKES: Die vom Besucher getätigten Likes werden vom EEPROM auf den Computer transferiert.

LOAD_SD: Sobald das Dojo in diesem State ist, schaltet der Multiplexer um und die mikroSD-Karte kann wie ein normaler Datenträger beschrieben werden.

LOAD_CONFIG: Ein Ticket wird auf das interne EEPROM geladen.

Abbildung 8.3: Mealy-Statemachine

8.2 Datenverwaltung

Das Ticket auf dem EEPROM beinhaltet für jedes Ausstellungsobjekt zwei Bytes, das erste beinhaltet die Beacon-ID und das zweite das Zutritts- und Like-Bit. Die ersten zwei Bytes sind reserviert und beinhalten die Anzahl gespeicherter Beacons und ein Sprachcode: 0 für die erste Sprace, 1 für die zweite. Da auf der mikroSD-Karte die Dateien der zwei Sparchen abewechslungsweise und in gleicher Reihenfolge wie im EEPROM aufgelistet sind, muss nur der Index der Beaon-ID übergeben werden damit die korrekte Audiodatei abgespielt werden kann: `(BeaconIndex - 2) * 2 + Sprachcode`.

8.3 Validierung

Die Validierung der Datenverwaltung wurde mit Unittests vom PC aus gemacht. Dabei wurde ein mit Zufallszahlengenerator generiertes Ticket über die serielle Schnittstelle auf das EEPROM geschrieben und dann wieder eingelesen. Diese Validierung wurde direkt auf dem Arduino UNO Board gemacht und hat funktioniert.

Auf dem MCU sind am Ende noch Probleme beim Auslesen des Receivebuffer von der seriellen Schnittstelle über das USB-Kabel zum Computer aufgetaucht. Dabei konnten über mySerial.available() die vom Computer gesendeten Befehle nicht ausgelesen werden. Mit dem Oszilloskop sind die gesendeten Befehle auf dem PD0 zwar messbar, allerdings können von der Seite des MCUs aus diese „Daten“ nicht detektiert werden. Deshalb funktioniert die Kommunikation vom Computer zum MCU auf dem Prototyp nicht wie geplant. Umgekehrt vom MCU zum Computer können aber Daten gesendet werden. Um die States zu ändern, wurden nun für die gesamte Funktion Taster zur Stateänderung verwendet, damit die Statemachine und der Ablauf der Firmware funktioniert.

Dabei sind die Tasterfunktionen:

SW1: Ändert den State zu LOAD_SD

SW2: Ändert den State zu PLAY

SW3: Ändert den State zu LOAD_CONFIG

Ein weiterer Aspekt, welcher dazu führt, dass für einen nächsten Prototyp des Dojos möglicherweise ein anderer Mikrochip benutzt wird ist, dass der ATMega 328p nur zwei external Interrupts hat (Abbilung 8.1 Pins PD2-INT0 & PD3-INT1). Diese sind momentan aber mit der Softwareserial belegt, weshalb mit den Taster nicht auf diese zugegriffen werden kann. Genauso gab es Konflikte mit der Softwareserial Library, die Tasterinterrupts über Pin Change Interrupts anzusteuern. Darum werden jetzt für die gesamte Funktion des Prototyps die Taster nach jedem scan() abgefragt und die Statusled zum togglen gebracht, wenn der Taster detektiert wurde.

Die micro SD-Karte kann als normaler Datenträger mit den Audiofiles beschrieben werden. Zu beachten ist einfach, dass dies nur über einen Computer mit Linux als Betriebssystem funktioniert.

9 Schlusswort

Der realisierte Prototyp ist in der Lage über BLE Beacons zu erkennen, eine entsprechende Rückmeldung über den Vibrationsmotor oder eine LED zu geben und ein zugehöriges Audiofile abzuspielen. Ausserdem kann die eingebaute μSD-Karte gelesen und beschrieben werden. Mit einem einfachen Command Line Interface, programmiert in Python, kann der Prototyp konfiguriert und ‹Likes›ausgelesen werden. Aus zeitlichen Gründen konnte die Übertragung des Konfigurationsfiles sowie die Auswertung der ‹Likes›noch nicht in das Gesamtsystem eingebettet werden. Das Konzept und die Software ist vorhanden, muss jedoch noch auf die Firmware des Mikrocontrollers abgestimmt werden.
Am Anfang des Projekts, wurde die USB Verbindung zur μSD-Karte, sowie die Umschaltung des Signals mit dem USB-Switch als sehr anspruchsvoll und fehleranfällig eingeschätzt. Durch sorgfältige Recherche und exaktem Layout wurden diese Hürden gut gemeistert. Durch die geringe Informationsmenge über den Audiochip WTV020 und seinen Eigenheiten, stellte sich seine Handhabung als anspruchsvoll heraus. Daraus resultierten einige Probleme im Verlaufe der Entwicklungsphase.
Bei einer allfälligen Weiterentwicklung des Produkts, sollte das Konzept überarbeitet werden. Eine Variante mit leistungsfähigerem Mikrocontroller bietet sich dabei an. Die Handhabung der μSD-Karte und Dekodierung der Audiofiles würde auf dem Mikrocontroller stattfinden. Dadurch kann das Kompatibilitätsproblem mit dem VUB300 Chip und Windows gelöst werden und man umgeht die Eigenheiten und Einschränkungen des WTV020 Soundchips.
In diesem Projekt konnten alle Teammitglieder viel dazulernen. Besonders herausfordernd war die Tatsache, ein bestehendes Design mit passender Elektronik auszustatten, welche alle Anforderungen erfüllt. Das realisierte Produkt erfüllt zwar im aktuellen Zustand nicht alle Anforderungen, jedoch bietet es eine gute Basis um schlussendlich einen voll funktionsfähigen Dojo zu realisieren. Das Konzept bietet, mit Ausnahme der Einschränkungen durch die Bausteine WTV020 und VUB300, die Möglichkeit, alle Anforderungen zu erfüllen. Abschliessend können wir sagen, dass wir zufrieden mit unserem Produkt sind und uns die Arbeit meistens Freude bereitet hat.

10 Verzeichnisse

Abbildungsverzeichnis

2.1 Blockschaltbild Gesamtsystem . . . 2

3.1 Inventardatei für eine Testausstellung . . . 3
3.2 Presetdatei für eine Testausstellung . . . 3
3.3 EEPROM Adressierungsmuster . . . 4

4.1 Schema der Ladeschaltung . . . 7
4.2 Ladekurve des Emmerich-Akkus . . . 7
4.3 Das verwendete Akkutestgerät ALC 8500 Expert . . . 8
4.4 Bildschirmausschnitt der PC-Software zum Akkutestgerät . . . 8
4.5 Entladekurve des Emmerich-Akkus bei konstantem Entladestrom . . . 9
4.6 Anordnung für die Strommessung am Akku . . . 10

5.1 Konzept USB . . . 11

6.1 Bone Conductor . . . 14
6.2 Audio Konzept . . . 15
6.3 Klasse D Verstärker . . . 15
6.4 WTV Serial Mode . . . 16
6.5 Audio Datei abspielen [4] . . . 17

7.1 Aufbau eines iBeacon-Pakets . . . 20
7.2 Rückgabe des AT+DISI? Befehls . . . 20

8.1 ATMega 328p . . . 22
8.2 UML-Diagramm Firmware . . . 23
8.3 Mealy-Statemachine . . . 24

11.1 Abmessungen WTV020 . . . 29

Literaturverzeichnis

[1] Matthias Meier. Digitaltechnik 1. Skript, 2016. [Unpubliziertes Dokument].

[2] Texas Instruments. TS3USB30. `https://www.ti.com/lit/ds/symlink/ts3usb30.pdf`, 2007. [Online Available: 11.06.2018].

[3] Emartee. WTV010 & WTV020. Datenblatt, April 2011.

[4] Sparkfun. Audio Sound Breakout WTV020SD. `https://github.com/sparkfun/Audio-Sound_Breakout-WTV020SD`, keine Angaben.

[5] JN Huamao Technology Company. Bluetooth 4.0 BLE module. Datenblatt, July 2017.

[6] Atmel. ATMega328/P. Datenblatt, November 2011.

11 Anhang

Abbildung 11.1: Abmessungen WTV020

11.1 Benutzeranleitung

11.1.1 `maketicket inventory_file preset_file`

Dieser CLI-Befehl schreibt ein "Ticketäuf das EEPROM. Das inventory_file beschreibt die ganze Datenbank, das preset_file beschreibt, welche Audiodateien sich schon auf der SD-Karte befinden. Gibt man den Befehl ein, promptet das Programm den Nutzer, welche Ausstellungen bezahlt sind und anschliessend welche Sprache gewünscht ist. Sind diese Daten erfasst schreibt das Programm das Ticket über die serielle Schnittstelle auf das EEPROM.

11.1.2 `makepackage inventory_file preset_file package_path`

Dieser CLI-Befehl kopiert die Audiodateien aller in der Presetdatei spezifizierter Ausstellungen in den Ordner package_path. Die Dateien sind dann alle umbenannt, damit sie in der richtigen Reihenfolge auf die SD-Karte geschrieben werden.

11.1.3 `loadsd package_path`

Dieser CLI-Befehl schreibt alle Audiodateien im Ordner package_path auf die SD-Karte des Dojos. Alle schon auf der SD-Karte vorhandenen Dateien werden dabei gelöscht.

Auf den folgenden Seiten befinden sich das Schema, das Layout und der Bestückungsplan der Printplatte.

Spannungsversorgung

TITLE: Pro4E_Schema

Document Number:

REV:

Date: 10.06.2018 17:01

Sheet: 1/3

USB & SD-Karte

TITLE: Pro4E_Schema

Document Number:

REV:

Date: 10.06.2018 17:01

Sheet: 2/3

Mikrocontroller & Audio

TITLE: Pro4E_Schema

Document Number:

REV:

Date: 10.06.2018 17:01

Sheet: 3/3

J1
1
U$5
C26
U$7
R22
C28
JP3
C36
R11
U$9
C34
R12
R13
C30
R8
C35
U$10
R19
JP1
D1
R14
SW1
C33
C32
Q2
C31
R21
SW2
U$6
C29
16
1
R20
SW3
R10
R15
R9
Q3
U1
R2
U$4
U$2
C5
*
C7
R3
C6
C8
C3
C4
R6
C22
C13
C17
C16
C15
IC1
C11
R4
C20
R5
Q1
C23
C9
C19
C18
R7
JP2
R18
J2
1
L1
L2
C25
C2
C1
R1
U$1
U$3

C27
Q5
R16
R17
C24

J1
U$5
U$9
U$10
JP3
JP1
SW1
SW2
SW3
PHNW prod
Team 1
Prototyp V1.0
U$6
U1
U$2
U$4
U$3
U$1
JP2

FHNW pro4E
Team 1
Prototyp V1.0

FHNW pro4E
Team 1
Prototyp V1.0