**E3 PRJ3**

**Efterår 2014**

**BODY ROCK 3000**

**Rapport**

**Gruppe 9**

**Deltagere:**

|  |
| --- |
| ***#1***  Stud.nr.: 201370738 Navn: Kristian Boye Jakobsen |
| ***#2***  Stud.nr.: 201205998 Navn: Lasse Fisker |
| ***#3***  Stud.nr.: 201270810 Navn: Mathias Siig Nøregaard |
| ***#4***  Stud.nr.: 201370768 Navn: Lukas Hedegaard Jensen |
| ***#5***  Stud.nr.: 201370914 Navn: Jonas Evers Nikolajsen |
| ***#6***  Stud.nr.: 201370801 Navn: Jeppe Hofni Hansen |
| ***#7***  Stud.nr.: 201371008 Navn: Felix Blix |
| ***#8***  Stud.nr.: 201370952 Navn: Kristoffer Lerbæk Pedersen |

**Vejleder: Peter Høgh**

12. december 2014

# Abstract

Resuméet skal udarbejdes **både** på dansk og på engelsk (abstract). Resuméet skal være en kortfattet sammenfatning af projektrapporten og bør indeholde følgende:

* Projektets formål
* Projektets problemstilling
* Valgte løsninger
* Anvendte metoder
* De væsentligste resultater

Resuméet skal beskrive arbejdet mere præcist, end det er muligt i den korte titel, således at læseren herudfra kan afgøre, om det er umagen værd at læse hele rapporten. Det skal kunne læses selvstændigt og må ikke indeholde henvisninger til afsnit i rapporten. Resuméets omfang må ikke overstige 2-300 ord og under **ingen** omstændigheder 500 ord. Skriv så konkret som muligt, og undgå vage udtryk eller fyldord.

Indhold

[Forside 1](#_Toc406148954)

[Abstract 1](#_Toc406148955)

[Indledning 4](#_Toc406148956)

[Opgaveformulering 4](#_Toc406148957)

[Projektafgrænsning 5](#_Toc406148958)

[Systembeskrivelse 5](#_Toc406148959)

[Krav 5](#_Toc406148960)

[Projektgennemførelse 6](#_Toc406148961)

[Udviklingsmodel 6](#_Toc406148962)

[Projektstyring 7](#_Toc406148963)

[Scrum 7](#_Toc406148964)

[Tidsplan 8](#_Toc406148965)

[Mødestruktur 8](#_Toc406148966)

[Dokumentorganisering og log 9](#_Toc406148967)

[Metoder 9](#_Toc406148968)

[SysML 9](#_Toc406148969)

[Scrum 11](#_Toc406148970)

[Specifikation og analyse 12](#_Toc406148971)

[Systemarkitektur 12](#_Toc406148972)

[Design, implementering og test af HW 13](#_Toc406148973)

[Sensorer 13](#_Toc406148974)

[I2C Bus 13](#_Toc406148975)

[Bluetooth 13](#_Toc406148976)

[PSoC shield 13](#_Toc406148977)

[Strømforsyning 13](#_Toc406148978)

[Design, implementering og test af SW 13](#_Toc406148979)

[Body 14](#_Toc406148980)

[Sensorer 14](#_Toc406148981)

[Bluetooth 14](#_Toc406148982)

[Control 14](#_Toc406148983)

[Slow Lane 14](#_Toc406148984)

[Controller 14](#_Toc406148985)

[Data Storage 14](#_Toc406148986)

[GUI 14](#_Toc406148987)

[Hovedmenu 14](#_Toc406148988)

[Sensorer 14](#_Toc406148989)

[Ny sensorkonfiguration 15](#_Toc406148990)

[Fast Lane 15](#_Toc406148991)

[Receiver 15](#_Toc406148992)

[MidiModule 17](#_Toc406148993)

[ALSA 20](#_Toc406148994)

[Integrationstest 20](#_Toc406148995)

[Body + I2C + Sensorer 20](#_Toc406148996)

[GUI 🡪 Controller 🡪 DataStorage 21](#_Toc406148997)

[Sensor 🡪 Receiver (Bluetooth) 21](#_Toc406148998)

[Receiver 🡪 MidiModule 21](#_Toc406148999)

[MidiModule 🡪 ALSA 21](#_Toc406149000)

[Udviklingsværktøjer 21](#_Toc406149001)

[PSoC Creator 21](#_Toc406149002)

[Atmel Studio 21](#_Toc406149003)

[Linux sampler 21](#_Toc406149004)

[Multisim 21](#_Toc406149005)

[Eagle CAD 21](#_Toc406149006)

[QT Creator 21](#_Toc406149007)

[Git 22](#_Toc406149008)

[Andre software biblioteker 22](#_Toc406149009)

[Resultater og diskussion 22](#_Toc406149010)

[Opnåede erfaringer 22](#_Toc406149011)

[Fælles 22](#_Toc406149012)

[Individuelt 22](#_Toc406149013)

[Konklusion Jonas 22](#_Toc406149014)

[Fremtidigt arbejde 23](#_Toc406149015)

[Konklusion 23](#_Toc406149016)

[Referencer 23](#_Toc406149017)

# Indledning

Denne rapport er skrevet på baggrund af et projektoplæg, som stiller visse krav til hvad projektet skal indeholde, men selve emnet er frit.

Dette projekt omhandler hvorledes sensorer, PSoC og Rasberri Pi kan benyttes til at opbygge et avanceret musikinstrument, som på sigt bl.a. kan skifte mellem flere lydpakker, etc.. Systemet skal bringe glæde og leg til musikken, og er blevet døbt ”BodyRock3000”.

Emnet for rapporten er blevet valgt på baggrund af et ønske om at se gruppens egne originale idéer blive realiseret fra bunden. Flere idéer blev overvejet, og BodyRock3000 blev valgt ud fra kriterier som:

* Integration af flere forskellige sensorer
* Mulighed for diverse software-databanke
* Brug af flere forskellige lydpakker
* Mulighed for at have en fungerende prototype ved slut

Opgaven udføres ved hjælp af de forskellige fag, som både 1., 2. og 3. semester på Ingeniørhøjskolen Aarhus Universitet har budt på, med særligt udgangspunkt i fagene på 3. semester. Først vil der blive udarbejdet en kravspecifikation, hvorefter gruppen arbejder med elementer af Scrum, hvor gruppemedlemmerne arbejder i iterationer.

# Opgaveformulering

Opgaven i dette projekt er at udvikle et intuitivt musikinstrument, baseret på kropslige bevægelser, der kan implementeres på diverse synthesizere, drummachines og DAW's (Digital Audio Workstation).

Instrumentet skal kunne afspille lydsamples, og generere MIDI-tone- og CC-signaler på baggrund af data fra accelerometer-, gyroskop-, proksimitets- og taktile trykmålinger.

Der er stillet følgende krav i den udleverede opgaveformulering:

* Systemet *skal* via sensorer/aktuatorer interagere med omverdenen
* Systemet *skal* have brugerinteraktion
* Systemet *skal* indeholde faglige elementer fra semesterets andre fag
* Systemet *skal* anvende Devkit 8000 og PSoC-teknologi

Dette medfører at det færdige produkt skal kunne opfange:

* Accelerationer
* Tilt
* Afstande
* Taktile tryk

Og på baggrund af disse:

* Afspille samples
* Generere MIDI-signaler

Efter konsultation med, og tilladelse fra, vejleder, er det besluttet at udskifte Devkit 8000 med en Raspberry Pi model B+. Denne beslutning er taget på baggrund af problemer med implementering af ALSA-biblioteker, det eksterne MIDI-lydkort og Linux-sampleren på Devkit 8000.

Visionen bag projektet er at skabe et nyt udtryks-medie for musikere, foruden at inkludere hidtil ulærte musiktalenter i den kreative og musikalske glæde, systemet vil medføre.

# Projektafgrænsning

Ingeniørhøjskolen Aarhus Universitet har opstillet følgende obligatoriske krav til 3. semesterprojekt:

* *Systemet skal via sensorer/aktuatorer interagere med omverdenen*
* *Systemet skal have brugerinteraktion*
* *Systemet skal indeholde faglige elementer fra semesterets andre fag*
* *Systemet skal anvende Devkit 8000- og PSoC-teknologi.(REFERENCE – CITAT)*

Ud fra kravene udvikles et elektronisk musikinstrument, der bruger sensorteknologi til at frembringe lyd. For yderlige uddybning, se projektbeskrivelsen (REFERENCE). Det skal nævnes at der for gruppe 9 er givet dispensation til at benytte en Raspberry Pi B+(REFERENCE) i stedet for Devkit 8000.

Projektet afgrænses til en at bestå af en prototype. Prototypen overholder kravene fra kravspecifikationen(REFERENCE) med undtagelse af:

* Antal sensorer er begrænset til én sensor af typen accelerometer
* Hverken Body- eller Rock-enheden indeholder ”Preset” funktionalitet(REFERENCE)
* Det vil ikke være muligt gøre Rock-enheden lydløs
* Det vil ikke være muligt at tilføje andre lydpakker end standard-lydpakken

# Systembeskrivelse

BodyRock3000



Opgaven i dette projekt er at udvikle et intuitivt musikinstrument, baseret på kropslige bevægelser, der kan implementeres på diverse synthesizere, drummachines og DAW's (Digital Audio Workstation).

Instrumentet skal kunne afspille lyd-samples og generere MIDI tone- og CC-signaler, på baggrund af data fra accelerometer-, gyroskop-, proksimitets- og taktile trykmålinger.

Dette medfører at det færdige produkt skal kunne opfange:

* Accelerationer
* Tilt
* Afstande
* Taktile tryk

Og på baggrund af disse:

* Afspille samples
* Generere MIDI-signaler

Systemet består af to dele, der kommunikerer trådløst:

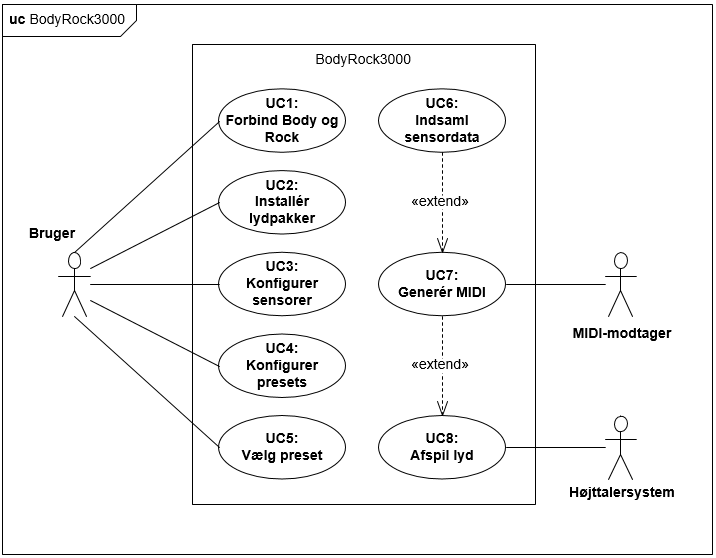
1. En bærbar enhed med sensorer til aflæsning af kropslige bevægelser og positioner
2. En stationær enhed, til processering af aflæste sensorer, afspilning af samples fra downloadede lydpakker, generering af MIDI-signaler og styring af system-indstillinger

Visionen bag projektet er at skabe et nyt udtryks-medie for musikere, samt at inkludere hidtil ulærte "skabsmusikanter" i den kreative og musikalske glæde, systemet vil medføre.

Systemets fleksibilitet og mulighed for udvidelse, samt den lette integration med eksisterende synthesizer- og sample-systemer, gør det til en kærkommen udvidelse af repertoiret for midler til musikalsk udtryk.

# Krav

# Krav

Ud fra opgaveformuleringen, er der udarbejdet en række use cases, som beskriver aktørernes interaktion med systemet. Disse use cases fungerer som kravspecifikation, og bruges i den tidlige del af udviklingsfasen, til at bestemme systemets funktionalitet. For at se de fully dressed use cases, henvises til dokumentationen.[[1]](#footnote-1)

Figur 1: Use case diagram over BodyRock3000

## Aktørbeskrivelse

På use case-diagrammet ses en række aktører. Disse er beskrevet i følgende aktørbeskrivelse:

**Bruger** er en primær aktør, som ønsker at benytte systemet BodyRock3000, ved at indstille diverse konfigurationer etc.[[2]](#footnote-2)

**MIDI-modtager** er en sekundær aktør, som kan transformere de generede MIDI signaler.[[3]](#footnote-3)

**Højtalersystem** er en sekundær aktør, som afspiller den ønskede lyd.[[4]](#footnote-4)

## Use case- beskrivelse

Use cases fra use case-diagrammet er beskrevet i følgende afsnit. Hver use case beskriver et scenarie hvor en aktør interagerer med systemet.

### Forbind Body og Rock

Brugeren tænder for Body og Rock, og benytter herefter hovedmenuen til at forbinde Body til Rock.[[5]](#footnote-5)

### Installér lydpakker

Brugeren benytter Rocks hovedmenu til at importere og installere en lydpakke.[[6]](#footnote-6)

### Konfigurer sensorer

Brugeren benytter Rocks hovedmenu til at oprette en ny sensorkonfiguration.[[7]](#footnote-7)

### Konfigurer presets

Brugeren benytter Rocks hovedmenu til at oprette en ny preset konfiguration.[[8]](#footnote-8)

### Vælg preset

Brugeren vælger preset ved at trykke på knapmatrixen placeret på Body.[[9]](#footnote-9)

### Indsaml sensordata

Sensor genererer rådata, hvilket sendes trådløst til Rock hvor det gemmes i en buffer.[[10]](#footnote-10)

### Generér MIDI

Læser data fra buffer og omdanner det til MIDI signaler.[[11]](#footnote-11)

### Afspil lyd

Der bliver afspillet den lyd som er genereret af BodyRock3000.[[12]](#footnote-12)

Projektbeskrivelse

# Projektgennemførelse

## Udviklingsmodel

Projektet er gennemført med en modificeret udgave af *ASE*-modellen(REFERENCE) som den primære udviklingsmodel. ASE-modellen er som udgangspunkt en vandfaldsmodel. Man arbejder sig systematisk frem med at lave opgaveformulering, kravspecifikation og systemarkitektur, for så at designe og implementere modulerne i hardware og software hver for sig og til sidst laver man en samlet accepttest.

Figur : ASE process

I dette projekt ville man gerne undgå en ren vandfaldsmodel. Derfor blev elementer fra *Verification* and *Validation modellen* også kaldet V-modellen indført. Alle trin i ASE-modellen blevet parret med et tilsvarende afsluttende trin. Se figur XX.



Figur : Verification and Validation model

Kravspecificationen blev udarbejdet sammen med accepttesten, systemarkitekturen sammen integrationstestene og til designfasen/implementeringen laves modultestene til alle enhederne.

V-modellen ligger op til at hvert trin i modellen afsluttes før der påbegyndes et nyt. Dette valgte man undervejs at ændre i projektgennemførelsen da forudsætningen for at dette kan lade sig gøre kræver at man har et overblik over hvilke teknologier der skal bruges i projektet og hvordan disse kan implementeres. V-modellen er derfor blevet brugt for at sammenholde de forskellige trin, men ved siden af, har projektet haft iterærer processer til hvert enkelt delsystem.

Typisk har disse processer bestået af teknologiundersøgelser og forsøg med at udvikle fungerende funktionelle udgaver af delsystemerne.(REFERENCE til teknologi undersøgelserne). Iterationerne er blevet styret ved hjælp af SCRUMsprint og i de senere sprint lykkes det også at have visse delsystemer færdige til at vise kunden/vejlederen.

## Projektstyring

### Scrum

Fra starten af semesteret blev der besluttet at projektet skulle styres så vidt muligt vha. Scrum som er en agil udviklingsmetode, hvilket var en af grundene for at undgå en ren vandfaldsudviklingsmodel. Scrum har fordelene:

* At man ikke behøver at kende alle krav i begyndelsen af projektet før man kan begynder at udvikle.
* Man deler projektet op i iterationer kaldet sprint så projektet bliver lettere at overskue.
* Hvert sprint ender ud i et produkt man kan demonstrer for kunden/produktejeren.
* Hver dag (eller hver anden dag for denne gruppe) mødes man til stå op møde og får overblik over opgaver og eventuelle problemer.
* Det er nemt og hurtigt at reallokerer resurser til der hvor der er mest brug for dem.

Det skal nævnes at Scrum er et softwareudviklingsværktøj men at gruppen også valgte at bruge det til hardwareudvikling.

Nøglebegreberne som gruppen har brugt fra Scrum er:

* Stå Op møde
* Denne mødeform består i at man, i denne gruppe, hver anden dag har mødtes stående og stillet hvert medlem tre sprøgsmål om hvad man lavede sidst, hvad skal man lave som det næste og om man har eventuelle problemer med at udføre sine opgaver.
* Roller
* Gruppen består af 8 medlemmer hvor et par af medlemmerne har delt Scrummasterrollen. Scrummasterrollen har således været fordelt med at et gruppemedlem har fungeret som primær kontaktperson til produktowner/vejleder og en anden har fungeret som ro og ordenholder.
* Produkt Backlog
* Backloggen består af alle opgaver som skal laves før projektet er færdig. I dette projekt er den blevet udfyldt løbende i fællesskab. Backloggens opgaver skal som udgangspunkt være delt op i Storys når det laves. Når et Sprint planlægges ligges opgaverne over på taskboarded.
* Taskboard og Sprint Backlog
* Taskboardet har fungeret som organistor for listen over opgaver for det pågældende sprint. Opgaverne er blevet defineret med beskrivelser, prioritering, estimeret tid og uddelegering. Taskboardet har også fungeret som kontrakt for hvad Scrumgruppen har valgt at forpligtige sig til af opgaver i sprintet.

Kort beskrivelse af hvert Scrum Sprint

* Sprint 1
* Længde: En uge.
* Formål: At lave kravspecifikationen, accepttesten og samt organiseringen af projektet.
* Sprint 2
* Længde: Tre uger.
* Formål: At lave teknologiundersøgelser af forbindelser, sensorer, MIDI, GUI samt at afslutte kravspecifikationen og accepttesten og påbegynde systemarkitekturen.
* Sprint 3
* Længde: To uger
* Formål: At lave de første implementeringer af delsystemer samt forsætte med at lave systemarkitekturen.
* Sprint 4
* Længde: Tre uger.
* Formål: At få lavet en stor del af hardwareimplementeringen færdig, få overblik over Rock delen af system, forsæt med systemarkitekturen og forsætte med implementeringen af delsystemerne.
* Sprint 5
* Længde: To uger
* Formål: At få overblik over Body delen af systemet, videreudvikle delsystemer samt gå ned i den dybere del af software opbygningen på Rock delen af system.
* Sprint 6
* Længde: En uge
* Formål: At få systemet til at generer lyd fra sensor til Body til Rock til lydafgiver.
* Sprint 7
* Længde: To uger
* Formål: At skrive produktdokumentation og projektrapport færdig, ???, ???

### Tidsplan

Den overordnede tidsplan(REFERENCE) der strækker sig igennem hele projektet er blevet tilegnet vha. Microsoft Projekt. Tidsplanen er blevet brugt til at holde overblikket over tidsrammen for projekt og er blevet opdateret efter hvert gruppe/vejledermøde.

### Mødestruktur

Gruppe- og vejledermøder er blevet styret ved hjælp af en mødeindkaldelse efterfulgt af et møde med dagsorden, dirigent og referent. De administrative roller er blevet fastlagt vha. en turnusordning(REFERENCE TIL TURNUSORDNING), hvor at de forskellige roller som mødeindkalder, referent og dirigent skifter fra møde til møde. Dette er gjort for at sikre at alle gruppemedlemmer får et indblik i det administrative arbejde. For at sikre konsensus i dokumenter, er der udarbejdet skabeloner til mødeindkaldelser og referater. Møderne er blevet afholdt efter behov, med udgangspunkt i et møde ved et sprints begyndelse og afslutning. Referatet fra forrige møde er blevet gennemgået og godkendt ved hvert møde.

## Dokumentorganisering og log

Gruppen har benyttet Git til organisering af dokumenter, kode og genereldokumentation for projektet. Git består af et repository som kan rumme alle disse data. Det har den fordel at alt hvad sendes til repository’et skal have en medfølgende beskrivelse og bliver stemplet med både dato/tidspunkt og præcise informationer om hvilke data bliver sendt med. Git’en har derfor også fungeret som gruppens fælles log for projektet. Gits funktion med at merge dokumenter er kun blevet brugt til dels.

(REFERENCE TIL GITloggen)

# Metoder

### SysML

Til formidling af kravspecifikation og systemarkitekturen har projektgruppen valgt at anvende SysML (Systems Modeling Language). Dette er gjort at formidle systemet bedst og bredest muligt, da SysML er industristandard og simpel og intuitiv at gå til for omverdenen.

SysML udspringer af UML (Unified Modeling Language), men er ulig UML, der centrerer sig om udvikling af SW, tiltænkt hele systemer (både SW og HW).

De af gruppen anvendte SysML-diagrammer kategoriserer sig i grupperne ”structure” og ”behavior” (adfærd).



#### Stucture Diagram

**Block Definition Diagram (BDD)** er benyttet til at vise nedbrydningen systemet i delelementer (blok).

**Internt Block Diagram (IBD)** er benyttet til at vise grænseflader mellem systemets blokke.

#### Behavior Diagram

**Use Case Diagram (UC)** er benyttet til at vise systemets krav til funktionalitet på baggrund af hvordan det benyttes af eksterne aktører til at opnå nogle mål.

**Sekvens Diagram (SD)** og **State Machine Diagram (STM)** er benyttet til at beskrivesystemets logisk funktionalitet.

#### Applikationsmodel

Der er for enkelte mere komplekse software-moduler gjort brug af applikationsmodeller. Disse tager udgangspunkt en udfærdiget domænemodel og viser det pågældende moduls funktionalitet ved forskellige brugssituationer. Produktet af en applikationsmode er et klassediagram samt SD og STM, der lægger et solidt fundament for efterfølgende software-udvikling.

For videre beskrivelse af SysML, se <http://www.sysmlforum.com/>

### Scrum

Følgende afsnit er en tilrettet version af det lignende afsnit i ”Projektgennemførelse” (det hører nemlig mere til i metoder)

Til projektstyring har gruppen har valgt at inddrage elementer fra den agile udviklingsmetode Scrum. Dette er gjort for at drage sig flg. fordele:

* En hurtig igangsættelse af systemudvikling: Det er ikke nødvendig at kende alle krav inden.
* Overskuelighed: Der vælges for hvert *sprint* et specifikt fokuspunkt.
* Konstant værdiskabelse: Hvert sprint ender ud i et produkt man kan demonstrer for kunden*/produktejeren*.
* Hyppig kommunikation: Hver dag (hver anden dag for netop denne projektgruppe) mødes gruppen til *stå op møde* og får overblik over opgaver og eventuelle problemer.
* Fleksibilitet: Reallokerering af resurser kan ske hurtigt som følge af hyppige *stå op møder*.

#### Scrum nøglebegreber

Se bilag: ”The Scrum Guide.pdf”.

#### Afvigelser i brugen af Scrum i dette projekt

**Hardwareudvikling**: Selvom Scrum i klassisk forstand er et softwareudviklingsværktøj, har gruppe valgt også at bruge den til hardwareudvikling.

**Frekvens af *stå op møder***: Grundet projektmedlemmernes sideløbende skolegang, har det ikke været muligt at arbejde nok til at retfærdiggøre daglige møder. I stedet har der i projektets begyndelse været 2 ugentlige møder og mod projektets afslutning 3-4 ugentlige møder.

**Scrum Master rollen**: Denne har været fordelt på flere af gruppens medlemmer; en kommunikator til product owner (i dette tilfælde projekt-vejleder) samt en opretholder af Scrums værdisæt og stå op mødernes fremgang. Sidstnævnte rolle blev som projektet skred frem mere kollektiv idet gruppen fik en fælles forståelse for arbejdsmetoden Scrum.

#### Scrum Sprints

Herunder ses en oversigt over gruppen Scrum sprints og varigheden af disse:

|  |  |  |
| --- | --- | --- |
| Sprint # | Varighed | Mål |
| 1 | 1 uge | Påbegyndelse af kravspecifikationen og accepttesten.  Projektorganisering |
| 2 | 3 uger | Teknologiundersøgelser af brugerønsker, forbindelser, sensorer, MIDI og GUI. Færdiggørelse af kravspecifikationen og accepttesten.  Påbegyndelse af systemarkitektur. |
| 3 | 2 uger | Videre arbejde med systemarkitektur.  Første implementeringer af delsystemer |
| 4 | 3 uger | Fortsat implementering af delsystemer, herunder hardwareimplementering.  Rock SW overordnet design. |
| 5 | 2 uger | Body SW overordnet design.  Udvikling af Rock SW moduler. |
| 6 | 1 uge | Integration af moduler med slutmål at generere lyd ud fra sensormåling. |
| 7 | 2 uger | Færdiggørelse af projektdokumentation og –rapport.  Videre arbejde med modul integration. |

# Specifikation og analyse

I dette afsnit vil vi komme med en kort beskrivelse over vores valgte løsninger. For yderligere begrundelse og analyse for hvorfor disse er valgt henvises til dokumentationen (REFERENCE)

Der er fortaget Teknologiundersøgelser for følgende teknologier:

## Sensortyper

## Bus-teknologier

## Trådløse teknologier:

Der var taget en beslutning om at kommunikationen mellem Body og Rock enhederne skulle foregå trådløst. Der blev taget højde for følgende krav i kravspecifikationen: Den trådløse forbindelse skulle kunne række minimum 10 meter. Ud fra dette blev forskellige hardwaremoduler undersøgt, og ud fra disse undersøgelser valgt to moduler til trådløs kommunikation.

For Body enheden blev valgt modulet: **Generic HC-05 Bluetooth RF Transceiver Module RS232**

For yderligere beskrivelse se dokumentationen (REFERENCE)

For Rock enheden blev valgt modulet: **RN42-I/RM Bluetooth 2.1 Module, -86dBm Receive Sensitivity, 4dBm Output**

For yderligere beskrivelse se dokumentationen (REFERENCE)

* **Grafisk user interface (GUI)**
* **Lydsamppler**
* **MIDI teknologi**

# Kunde undersøgelse

Eftersom projektgruppen ikke kun agerer udviklere, men også projektinitiativtagere og –indehavere, er der lavet en undersøgelse af ønskede features fra en musikers perspektiv. Undersøgelsen er foretaget af projektgruppemedlem Lukas Hedegaard, der selv er udøvende musiker og dermed har brugererfaring inden for feltet.

**Ønskede features**:

* Styring af toner (Note)
  + Retning: hhv. opadgående, hvor lav sensorværdi giver dyb tone, høj sensorværdi giver lys tone, samt nedadgående (modsat opadgående).
  + Skala valg, herunder kromatik, dur og mol.
  + Ændring af toneart
* Styring af anslagskraft (Velocity) med justerbar sensitivitet
* Styring af expression-parametre (Control Change) med hhv. absolut- og relativ ændring af parameter
* Mulighed for benyttelse af en enkelt sensor til justering af flere parametre
* Integration med eksisterende instrumenter/synthesizere over MIDI.
* System presets med let adgang, så systemets indstillinger kan skifte momentant, for at lette systemets brug i et live-set med flere på hinanden følgende sange med forskellige lyde.
* Intern samplebank med justerbare lyde (sekundært ønske).

Kundeundersøgelsen udmunder sig i kravspecifikationens afsnit ”Krav til Mapping Scheme” på s. 20 i projektdokumentatioen. (REFERENCE TJEK)

# Systemarkitektur

## Overordnet arkitektur

Den overordnede systemarkitektur ses på nedenstående domænemodel. Denne model giver et godt overblik over det samlede system og hvordan det agere med hinanden.



Figur

Systemet består bland andet af vores to ”hovedenheder” Body (PSoC4)(REFERENCE) og Rock (Raspberry Pi)(REFERENCE), disse to enheder er vores arbejdsmaskiner i systemet. Body modtagere data input fra vores sensorer, pakker dette og sender det via sit bluetooth-modul(REFERENCE) til Rock enhedens bluetooth-modul(REFERENCE). Rock udpakker så denne data og omdanner denne til et lydoutput. (MIDI/ALSA beksrivelse?????????????????????)(REFERENCE)

Brugeren har muligheden for at opsætte diverse konfigurationer via den grafiske brugeroverflade (GUI)(REFERENCE) på Rock enheden. På denne måde kan der konfigureres hvordan Rock enheden skal håndtere de data input den modtager.

# Design, implementering og test af HW

I det følgende beskrives det detaljerede hardwaredesign for projektet. Det detaljerede design tager udgangspunkt i kravsspecifikationen og systemarkitekturen. Designprocessen samt hvilke overvejeler og valg der er taget vil ligeledes blive beskrevet i dette afsnit.

## Indledende designovervejelser

I domain model BodyRock3000(REFERENCE TIL DOKUMENTATION), ses hvilke blokke det samlede system består af og hvilken funktionalitet de skal udfylde.

Systemet kan nedbrydes til to hardwareblokke en Body blok og en sensor blok.

Body blokken kan nedbrydes til følgende:



Figur : BDD for body

**Body** består af hardwareblokkene: Spændingsforsyning, Bluetooth-modul og Presetknapper. For yderlige beskrivelse af blokkene henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION).

## Overvejelser omkring sensorer

I projektformuleringen(REFERENCE TIL DOKUMENTATION) blev det fastlagt, at det endelige produkt skulle kunne opsamle data fra bevægelser herunder, accelerationer, tilt, afstande og taktil tryk.

Ud fra ovenstående blev det besluttet, at følgende sensortyper skulle anvendes:

* Accelerometer
* Gyroskop
* Proximity sensor
* Tryksensor

## Sensorer

Da alle sensorer kobles til I2C bussen gennem et fire polet RJ11 stik, kan følgende generiske IBD for en sensorenhed tegnes.



Figur : IBD for generisk sensorenhed

Som det ses af figur **XX** er alle sensorenhederne koblet til den eksterne 3.3V spændingsforsyning med tilhørende GND, via to af polerne fra RJ11 stikket. Herudover er sensorerne koblet til henholdsvis SCL og SDA.

Det ses ligeledes heraf at der er sensorenheden er koblet til I2C to steder, hvilket som sagt giver mulighed for at serieforbinde flere sensorer.

I det følgende afsnit beskrives de forskellige sensorenheder. Kun accelerometeret er fuldt beskrevet, for at se fuld beskrivelse for de andre sensorer henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION)

### Accelerometer

Sensoren som benyttes i dette projekt er et 3-akset accelerometer af typen **ADXL345[[13]](#footnote-13)**. Denne model er ultralow power, den kan gå så lavt som til i *measure mode* og kun i *standby mode* disse værdier er fundet i databladet.[[14]](#footnote-14) ADXL345 understøtter i forvejen I2C[[15]](#footnote-15) og er derfor at foretrække at benytte da der ikke skal tilføjes noget ekstra til enheden for at benytte I2C.

Det fremgår desuden af databladet at ADXL345 opererer ved 2V til 3.6V, hvilket passer til vores spændingsforsyning som leverer 3.3V[[16]](#footnote-16).

#### I2C

Når man skal oprette forbindelse til en sensorenhed via I2C, er det vigtig at man kender komponentens I2C—adresse. ADXL345 har en pin kaldet ALT ADDRESS[[17]](#footnote-17), denne pin bruges til at styre ADXL345’s to forskellige I2C adresser.

|  |  |
| --- | --- |
| **I2C adresse (hex)** | **ALT ADDRESS PIN** |
| 0x1D | Koblet til VCC |
| 0x53 | Koblet til GND |

Tabel : I2C adresser og kobling til ALT ADDRESS pin

Ud fra denne viden benyttes en 1x3 Harwinpin med tilhørende jumper, således at brugeren hurtig og nemt kan skifte mellem de to alternative I2C adresser.

For at se breakoutboard samt forbindelser henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION)

### Gyroskop

Gyroskopet har til formål at generere data op baggrund af tilt. For fuld beskrivelse af gyroskopets design og implementering henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION).

### Proximity sensor

Proximity sensoren har til formål at generere data på baggrund af afstand. For fuld beskrivelse af proximity sensor design og implementering henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION)

### Tryksensor

Tryksensoren har til formål at generere data på baggrund af taktilt tryk. For fuld beskrivelse af tryksensor design og implementering henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION)

## I2C Bus

## Bluetooth

## PSoC shield

## 

## Spændingsforsyning

Systemet BodyRock3000, opererer på spændingen 3.3V.

Til at forsyne systemet med 3.3V designes en spændingsforsyning, som består af et **XX**V batteri og en reguleringskreds.

### Reguleringskreds

Til reguleringskreds benyttes **LM317**(REFERENCE). LM317 som er en 3 terminal justerbar regulator med en outputrange fra 1.2V til 25V.



Figur : Typisk opsætning af LM317 (fra datablad LM317)

**R2** er den modstand som skal justeres for at få den ønskede udgangsspænding. Størrelsen af modstand R2 findes vha. følgene formel fra databladet:

Som det ses af ovenstående formel er VOUT ikke afhængig af inputtet, hvilket er en fordel da man i så fald kan bruge batterier med forskellig spændingsstørrelser. Det gælder dog at indgangsspændingen som minimum skal være 1.5V større end den ønskede outputspænding.(REFERENCE TIL DATABLAD)

For beregninger samt simulering af reguleringskredsen henvises til projektdokumentationen(REFERENCE TIL DOKUMENTATION).

### Batteri

Vælges efter det er bestemt hvorledes reguleringskredsen skal laves. Da reguleringskredsen kræver at inputspændingen minimum skal være 1.5V større end den ønskede outputspænding er batteriet valgt til at være på 9V.

# Design, implementering og test af SW

I det følgende afsnit beskrives det detaljerede softwaredesign for projektet. Designprocessen, samt hvilke overvejelser og valg der er blevet truffet, vil ligeledes blive beskrevet.

For uddybende beskrivelser af design og implementering af software, henvises til dokumentationen (REFERENCE)

Identifikation af ønskede softwaredele:

I prototypen af vores produkt er der kun implementeret vejen fra hovedmenuen og ned til oprettelse af en ny sensorkonfiguration

**Grafisk brugeroverflade (GUI) – Rock**, bestående af grafisk platform med følgende afgrænset indhold:

* Hovedmenu: Generelt overblik over de mulige konfigurationer  
   Herunder: Sensorer, Presets og Lydpakker
* Sensorer: Menuen for mulige konfigurationer af sensorer   
   Herunder: Ny sensorkonfiguration, Rediger sensorkonfiguration og Slet sensorkonfiguration
* Ny sensorkonfiguration: Muligheden for at konfigurere indstillingerne for en ny sensorkonfiguration
* messageHandler: Denne klasse håndtere beskederne mellem GUI klasserne og Controller klassen

**Controller klassen – Rock,**

kontrollerer trådkommunikation for systemet, fungere som bindeled mellem GUI’s messageHandler og Datastorage.

## Body

### Sensorer

### Bluetooth

### Control

## Slow Lane

### Controller

### Databanken

Databanken har til formål at gemme eller hente konfigurationer. Konfigurationerne kan være sensorkonfigurationer (REFERENCE), presets (REFERENCE) eller lydpakker (REFERENCE). Hvilke attributter konfigurationerne indeholder kan ses i referencerne. Databanken er implementeret som en klasse med tre underliggnede klasser, sensorkonfigurationsbanken, presetbanken og samplebanken. Disse indeholder konfigurationerne nævnt tidligere. Controller-klassen har en instans af Databanken, således at den blot kalder funktioner i Databanken så den kan få de relevante data og sende dem videre til GUI’en. For at gemme og indlæse konfigurationerne bruges der Boost-bilbioteket. Boost indeholder funktioner til serialisere konfigurationerne til XML-filer. De samme filer omdannes tilbage til sensorkonfigurationer, presets og samples vha. Boost.

Nedenunder ses en overordnet skitse for databanken.



For detaljeret beskrivelse af interaktion henvises til sekvensdiagrammet (REFERENCE) og klassediagrammet (REFERENCE).

### GUI

### Hovedmenu

Hovedmenuen er implementeret som klassen mainWindow. Der oprettes en instance af denne klasse i vores main program. Inde i mainWindows konstruktor oprettes det grafiske UI som danner rammerne for at vi kan designe diverse elementer i UI’et, såsom knapper, dropdowns og lignende.

Brugeren bliver i hovedmenuen præsenteret for følgende tre valgmuligheder:

1. Sensorer Danner en instans af sensorWindow
2. Lydpakker Danner en instans af lydpakkeWindow
3. Presets Danner en instans af presetWindow

Alt efter brugerens valg dannes det grafiske ui for den tilhørende klasse.

For yderligere beskrivelse af Hovedmenuens funktionalitet, henvises der til dokumentationen (REFERENCE)

### Sensorer

Efter brugeren har valgt menu punktet Sensorer dannes der en instance af klassen sensorWindow, der har samme funktionalitet i konstruktoren som mainWindow.

Brugeren bliver i Sensorkonfigurationer(instancen af sensorWindow) præsenteret for følgende tre valgmuligheder:

1. Ny Sensorkonfiguration
2. Rediger Sensorkonfiguration
3. Slet Sensorkonfiguration

Alt efter brugerens valg dannes det grafiske ui for den tilhørende klasse, og der sendes besked til messegeHandler klassen efter den nødvendige info, da der tages udgang i prototypen er det Ny sensorkonfiguration der er implementeret.

### Ny sensorkonfiguration

Efter brugeren har valgt Ny Sensorkonfiguration, dannes der en instance af klassen nySensorkonf, der har samme funktionalitet i konstruktoren som de ovenstående klasser. Derudover kaldes en funktion der ligger i klassen messageHandler, der efterspørge den nødvendige information der skal bruges til at oprette en ny sensorkonfiguration. Denne information er følgende:

1. Sensor type
2. Mapping scheme
3. Akse
4. Lydpakke

Sensor type og akse er ”hardcoded” i prototypen, da der ikke er implementeret mere end en type sensor og der kun er tre mulige akser (x, y, z), de to andre informationer hentes gennem messegeHandleren’s funktion getSensorKonfInfo, der henter en struct med lister af strings med navne på de Mapping schemes og Lydpakker der ligger i Datastorage (Klassen der håndtere vores gemte data). Derved dannes der et ui med mulighed for at vælge den information vi skal bruge til at oprette en ny sensorkonfiguration.

BILEDE!!!!!!!!

Efter brugeren har valgt sin ønskede Sensorkonfigurationer har han muligheden for at gemme disse ved anvendelse af en gem knap. Når denne anvendes sendes de fire valgte informationer til messageHandleren der sender det videre til Controller klassen der gemmer informationen i Datastorage.

Den nyoprettede Sensorkonfiguration er nu klar til at blive benyttet i den yderligere del af systemet.

For yderligere information om GUI klasserne henvises til dokumentationen (REFERENCE)

## Fast Lane

### Receiver

Receiver-klassen har til opgave at modtage, sortere og videresende alle meddelelser fra Body, modtaget over Bluetooth.

Når et objekt af klassen oprettes, oprettes en forbindelse der læser på noden *ttyAMA0*, som håndterer UART-forbindelser på en Raspberry Pi. Denne forbindelse lukkes igen ved nedlæggelse af objektet.

**Figur XX** viser et sd-diagram over operationerne som udføres af klassen Receiver.



**Figur 8** Sekvensdiagram over operationerne udført af klassen Receiver

Receiver kan udføre to typer af operationer, som begge initieres af indkommende beskeder over bluetooth. Operationerne er indrammet i de to yderste **opt**-kasser, og initieres af indholdet af den først byte i forsendelsen, startbyten.

Operationen i **opt1** initieres af brugeren, når der på Body-enheden skiftes preset. Der modtages en startbyte, som indikerer at den følgende byte er det ønskede preset, og preset-valget sendes hernæst videre til Controller-klassen.

Operationen i **opt2** initieres for hver aktive sensor 50 gange i sekundet af Body-enheden, når outputtet fra den aktuelle sensor er blevet aflæst. Der modtages en startbyte, som indikerer at den følgende byte er sensorens ID, mens de efterfølgende tre bytes er sensor-outputtet i op til tre dimensioner. Hvis ID’et er større end sidst modtagede ID, tilføjes dataene til en datapakke. Er ID’et derimod mindre end eller lig med sidst modtagede ID, sendes datapakken til MIDImodule-klassen, og en ny datapakke påbegyndes med de medsendte data.

### MidiModule

#### MidiModule control

#### SensorConfiguration

#### Mapping Scheme

#### Design

Mapping Scheme klassens har til funktion af løse to opgaver

1. Lagre brugerindstillinger for hvordan data fra en given sensor omdannes til et MidiSignal
2. At syntetisere et MidiSignal ud fra en given sensor måling.

##### Klassediagram

Herunder ses et klassediagram for MappingScheme og dets lagringsstructs[[18]](#footnote-18) samt for MidiSignal, som MappingScheme også benytter i sin map() –funktion. Bemærk at der for klassediagrammet herunder er udeladt set- og get-metoder.



For funktionsbeskrivelser, se projektdokumentation s. XXX

#### Implementering

Herunder foreligger et udsnit af beskrivelsen af implementeringsfremgangen for klassens funktioner. For funktionerne mapVelocity, mapCCAbs og mapCCRel, se projektrapport s. XXX

Se bilag MappingScheme.h og MappingScheme.c for den endelige MappingScheme implementering.

##### map

map() har til ansvar at kalde den underfunktion, der svarer til den mapping parameter, brugeren har indstillet.

##### mapKey

Programflowet i mapKey er som vist i nedenstående flowchart



Tilrettelse af data: Systemet kan generere toner fra oktaverne -2 til 9 = 12 oktaver. I hver oktav er der 12 toner. Dette giver mulighed for at generer i alt 120 forskellige toner. Derfor benyttes kun sensorData i intervallet 3-122.

quantizeDiatonic: beskrives i flg. afsnit.

Dataforskydning muliggør korrekt kvantisering ved forskellige grundtoner.

Sæt Midisignal: For at muliggøre brug af flere note inputs på samme polyfoniske MIDI instrument, skal den forrige tone for sensoren slukkes inden en ny igangsættes. Der tjekkes derfor først om tonen er forskellig fra den gamle inden den sættes. Er den ny, sættes kommandoen til ”note off”. Dermed kræves to gennemløb med samme modtagne data før tonen ændres.

##### quantizeDiatonic

Herunder vises et udsnit af dokumentationen for quantizeDiatonic. Udover nedenstående, forefindes en beskrivelse af mol-kvantiseringen i projektrapporten s. XXX

Under et betegner man dur- og mol-skalaer som diatoniske skalaer – deraf navnet.

Når det indkomne data (som indeholder alle oktavens toner) skal kvantiseres, skiftes de toner, der ikke ligger i skalaen op- eller ned til en tone i skalaen. Kvantiseringen er designet således, at de mest brugte toner favoriseres: Grundtone (1. trin), terts (3. trin), kvint (5. trin), herunder markerede som hhv. 1, 3 og 5.

Dur kvantiseringen foregår som på illustrationen af en oktav herunder:



Påskrevet er durskalaens trin (1-7) for en c-dur skala samt hvilken vej det ønskes at tonerne uden for skala skal kvantiseres til (røde pile).

##### Test

Test af MappingScheme er for de fleste funktioner testede for et begrænset, men nøje udvalgt sæt input data. De er som udført af designer med udgangspunkt i en white box approach, hvor alle signalveje testes, om end ikke alle mulige udfald afprøves; dette ville for funktionen mapKey være *12 gruntoner x 3 skalaer x 2 retninger x 127 mulige datainput = 9144 muligheder* for testens udfald. Netop denne funktion er således testet for et begrænset antal inputs for en specifik grundtone-skala kombination. Der henvises til endelig integrationstest for auditiv validering af signalvej for forskellige grundtoner og skalaer.

Alle tests er at finde i projektdokumentationen s. XXX

#### AlsaAdapter

### ALSA

# Integrationstest

## Body + I2C + Sensorer

## GUI 🡪 Controller 🡪 DataStorage

## Sensor 🡪 Receiver (Bluetooth)

## Receiver 🡪 MidiModule

## MidiModule 🡪 ALSA

# Udviklingsværktøjer

Herunder gives en kort beskrivelse af relevante udviklingsværktøjer benyttet i projektet.

## PSoC Creator

Givet at brugen af PSoC var et projekt krav, har Cypress’ PSoC Creator været væsentlig for projektet. Demme IDE tillader udvikling af hele systemer til PSoC chippen fra et sammenhængende miljø. Dette dækker over hardware mapping af de digitale og analoge subsystemer, automatisk generering af software API ‘er til indbyggede komponenter samt udvikling af programkode.

## Atmel Studio

Til at forsyne vores sensor kredsløb med I2C interfaces er der blevet benyttet to ATtiny microcontrollere, til hvilke programmerne er blevet skrevet i Atmel Studio. Dette tillader at programmerne kan bygges med de relevante Atmel biblioteker og passende compiler alene ved at fortælle IDE’en hvilken chip man skriver til. Derudover har AS også et værktøj til at programmere chippen gennem vores STK500 board.

## Linux sampler

For at kunne omsætte de genererede midi signaler til lyd skal der benyttes en midi sampler. Til vores projekt har vi valgt at benytte Linux sampler, da denne er en letvægts sampler lavet til at køre på Linux. Dette gjorde den ideel til vores projekt da dette benytter sig af en indlejret Linux platform med begrænsede system ressourcer.

## Multisim

Til kredsløbne for afstands- og tryksensoren benyttes der en række hardware komponenter da disse er adskilt fra PSoC ’en af en I2C bus. Disse er simuleret i Multisim før implementeringen for at forhåndsteste designet.

## Eagle CAD

Til PCB layout er der i projektet blevet benyttet Eagle. Dette er gratis at bruge til nonkommercielle formål og har rigeligt med features til dette projekt. Programmet tillader udarbejdelse af skematiske diagrammer, samt print layout med automatisk kontrol af overensstemmelse med skematisk design.

## QT Creator

Til udvikling af den grafiske brugerflade på Rock enheden blev det valgt at benytte QT biblioteket, da dette er forholdsvis udbredt til GUI udvikling på indlejrede Linux systemer. QT Creator er et IDE der tillader at man kan designe sin QT brugerflade gennem et grafisk værktøj, hvorefter koden til dette genereres automatisk.

## Git

Til organisation af filer, både kode og dokumentation, er der blevet benyttet et Git repository fra GitHub. Dette giver et system til fildeling hvor der samtidig tages højde for filhistorik. Da dokumentationen ikke er skrevet i LaTeX eller lign. er der ikke blevet draget nytte af Gits potentiale ift. at tillade flere at arbejde på den samme fil på samme tid. Dette fungerer kun i plaintext formater, og ikke i formater som f.eks. .docx.

## Andre software biblioteker

Til udviklingen af softwaren på Rock enheden er der gjort stor nytte af C++’s Standard Template Library, specielt til brogrammets datastrukturer. Da programmet også har behov for at kunne gemme data ved system genstart er der ud over standard bibliotekerne også benyttet Boost’s serialisations bibliotek til lagring af objekter som XML kode.

# Resultater og diskussion

Beskrivelse af projektets resultater i kort form bl.a. ved anvendelse af tabeller, grafer eller billeder. Det er vigtigt, at man her klart og nøgternt præsenterer sine resultater. Det er vigtigt at udpege og diskutere relevante dele af de opnåede resultaterne og deres betydning. Bl.a. en samlet vurdering af resultaterne i lyset af problemstillingen og formålet med – eller hypotesen for projektet. Der må også gerne være en beskrivelse af de dele af projekt man er specielt stolt af.

Hvad er lykkedes hvad er ikke?

Fælles.

# Opnåede erfaringer

## Fælles

## Individuelt

### Konklusion Jonas

Dette projekt har for mig været et af de sværere. Dette skyldes at jeg har haft svært ved meget af undervisningen, og har derfor haft vanskeligheder ved at overføre det faglige stof til projektet. Derfor har jeg været meget afhængig af at have en sparingspartner når jeg skulle arbejde.

Projektet som helhed tror jeg vi har fået op i en lidt for ambitiøs størrelse, dette har også medført at vi har haft svært ved at færdigøre en virkende prototype.

Selvom det har været en sværd del at komme igennem føler jeg stadig jeg har fået noget lærerigt ud af det. Jeg har anvendt udviklingsværktøjet QT og lært en masse om dette program.

Jeg har også kunne mærke at struktureringen omkring rapport og dokumentation ikke er på samme niveau som sidste semester, hvor dette netop også var fokus punktet. Dette har været lidt forvirrende.

Vi har anvendt Scrum til den daglige strukturering af opgaver og møder, dette har været lærerigt og en god arbejds metode.

Personligt synes jeg det har været knap så godt et projekt fra min side af.

# Fremtidigt arbejde

Første mål for en fremtidigt arbejde vil være at tilfredsstille alle de i kravspecifikationen stillede krav. Disse er:

* Implementere preset skift fra Rocks GUI.
* Live presetskift fra Body over systemets bluetooth forbindelse.
* Implementere af forskellige lydpakker i ALSA og at være i stand til at skifte disse med presetvalg.

Næste målsætning er en fuld implementering af MIDI med eksempelvis ”pitch bend” (mulighed for at ”bøje” tonerne) og ”patch change”, så det ved presetskift også er muligt at ændre preset på eksterne MIDI-instrumenter.

En mulig implementering i fremtidige systemiterationer kunne endvidere være opkobling af flere Body enheder per Rock enhed, for at facilitetere brugen af systemet til sociale aktiviteter.

Derudover er der mulighed for udvidelse af justeringsmulighed i Mapping Scheme:

* Skalaer: Heltone, pentaton, harmonisk- og melodisk mol.
* Velocitetskurver: Lineær, logaritmisk, eksponentiel, fuld.

# Konklusion

Her gives en samlet konklusion på projektarbejdet. Hvad er lykkedes, hvad er evt. ikke lykkedes og årsagen til dette. Konklusionen skal gerne indeholde et klart budskab og forholde sig objektivt til de opstillede krav og opnået resultater.

Desuden sammenfattes de slutninger, der kan drages af de resultater, som er omtalt i rapportens tidligere afsnit. Konklusioner kan være såvel positive som negative. Man skal tage sig i agt for ikke at undertrykke de negative fund (hvis f.eks. en metode har vist sig uegnet, bør det opfattes som et bidrag til ens erfaringsmateriale, ikke som et personligt nederlag).

I konklusionen trækkes desuden de store linier op. Væsentlige kvantitative resultater kan nævnes, hvorimod den detaljerede redegørelse og argumentationen henvises til diskussionen i rapportens hoveddel.

# Referencer

Brug words funktion

Harward citation:

<http://guides.is.uwa.edu.au/harvard>

1. REFERENCE [↑](#footnote-ref-1)
2. REFERENCE [↑](#footnote-ref-2)
3. REFERENCE [↑](#footnote-ref-3)
4. REFERENCE [↑](#footnote-ref-4)
5. REFERENCE [↑](#footnote-ref-5)
6. REFERENCE [↑](#footnote-ref-6)
7. REFERENCE [↑](#footnote-ref-7)
8. REFERENCE [↑](#footnote-ref-8)
9. REFERENCE [↑](#footnote-ref-9)
10. REFERENCE [↑](#footnote-ref-10)
11. REFERENCE [↑](#footnote-ref-11)
12. REFERENCE [↑](#footnote-ref-12)
13. REFERENCE [↑](#footnote-ref-13)
14. REFERENCE [↑](#footnote-ref-14)
15. REFERENCE [↑](#footnote-ref-15)
16. RERERENCE [↑](#footnote-ref-16)
17. REFERENCE [↑](#footnote-ref-17)
18. Som specificeret i afsnittet ”Krav til Mapping Scheme” [↑](#footnote-ref-18)