Indhold

[Projektformulering 4](#_Toc405203398)

[Termliste 5](#_Toc405203399)

[1.1.1 Body 5](#_Toc405203400)

[1.1.2 Rock 5](#_Toc405203401)

[1.1.3 BodyRock3000 5](#_Toc405203402)

[1.1.4 Lydpakke: 5](#_Toc405203403)

[1.1.5 MIDI-parameter 5](#_Toc405203404)

[1.1.6 Lydmodul 5](#_Toc405203405)

[1.1.7 Mapping scheme 5](#_Toc405203406)

[1.1.8 Sensorkonfiguration 5](#_Toc405203407)

[1.1.9 Buffer 5](#_Toc405203408)

[1.1.10 Preset 5](#_Toc405203409)

[Kravspecifikation 6](#_Toc405203410)

[Use Case diagram 6](#_Toc405203411)

[1.2 Aktørbeskrivelse 6](#_Toc405203412)

[1.2.1 Bruger 6](#_Toc405203413)

[1.2.2 MIDI-modtager 6](#_Toc405203414)

[1.2.3 Højtalersystem 7](#_Toc405203415)

[1.4 Use Case beskrivelser 8](#_Toc405203416)

[1.4.1 Forbind Body og Rock 8](#_Toc405203417)

[1.4.2 Installér lydpakker 10](#_Toc405203418)

[1.4.3 Konfigurer sensorer 11](#_Toc405203419)

[1.4.4 Konfigurer preset 13](#_Toc405203420)

[1.4.5 Vælg preset 15](#_Toc405203421)

[1.4.6 Indsamle sensordata 16](#_Toc405203422)

[1.4.7 Generér MIDI 17](#_Toc405203423)

[1.4.8 Afspil lyd 18](#_Toc405203424)

[Ikke funktionelle krav 20](#_Toc405203425)

[1. Generelt 20](#_Toc405203426)

[2. Body 20](#_Toc405203427)

[3. Rock 20](#_Toc405203428)

[Systemarkitektur Hardware 21](#_Toc405203429)

[Overordnet arkitektur 21](#_Toc405203430)

[Domæne model BodyRock3000 21](#_Toc405203431)

[Body HW Arkitektur 22](#_Toc405203432)

[BDD Body 22](#_Toc405203433)

[Allokeringsdiagram Body 23](#_Toc405203434)

[IBD Body 24](#_Toc405203435)

[Sensor IBD (Generisk) 25](#_Toc405203436)

[Rock HW Arkitektur 25](#_Toc405203437)

[BDD Rock 25](#_Toc405203438)

[Allokeringsdiagram Rock 26](#_Toc405203439)

[Grænsefladebeskrivelse 27](#_Toc405203440)

[Systemarkitektur Software 28](#_Toc405203441)

[Detaljeret domænemodel for software-moduler 29](#_Toc405203442)

[Klasse-identifikation 30](#_Toc405203443)

[Body 32](#_Toc405203444)

[Sekvensdiagram Body 32](#_Toc405203445)

[Klassediagram Body 32](#_Toc405203446)

[Rock 33](#_Toc405203447)

[Tråd kommunikation 33](#_Toc405203448)

[Slow Lane 34](#_Toc405203449)

[Fast Lane 36](#_Toc405203450)

[Overordnet Klassediagram Rock 39](#_Toc405203451)

# Projektformulering



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, samt 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, som kommunikerer trådløst:

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

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.

Systemets fleksibilitet og muligheder 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.

# Termliste

### Body

Body er den bærbare enhed for systemet BodyRock3000. Den består af en controller-enhed, som kan sende sensordata til den stationære enhed Rock. På controller-enheden kan der tilsluttes et vilkårligt antal sensorer, der kan foretage accelerometer-, gyroskop-, proksimitets- og taktile trykmålinger. Da det er et krav at anvende PSoC4-boardet i projektet, vil denne blive brugt som Body-enhed.

### Rock

Rock er den stationære enhed for systemet BodyRock3000. Denne enhed modtager sensordata fra Body-enheden via trådløs kommunikation. Sensordataene konverteres herefter til MIDI, som fastlægger lydoutputtet. Det er ligeledes et krav at anvende DevKit8000 i projektet, derfor vil DevKittet blive brugt som Rock-enhed.

### BodyRock3000

BodyRock3000 er en samlet betegnelse for systemet. Denne bruges bl.a. som initiator ved Use Case 6, og indikerer at Use Casen startes af interne processer op systemet.

### Lydpakke:

Bestemmer hvilke lyde der skal syntetiseres, på baggrund af de modtagne MIDI-signaler

### MIDI-parameter

Der er generelt tre typer MIDI-parametre:

* **Note**: Angiver hvilken tone, der skal afspilles i intervallet [0-127]
* **Control Change (CC):** Angiver hvilken parameter, der ønskes ændret i intervallet [0-127]
* **Program Change**: Angiver hvilket preset, der ønskes i intervallet [0-127]

### Lydmodul

Et digitalt datamodul, der agerer som bindeled mellem de valgte lydpakker og MIDI-parameter-inputs, således at den korrekte lyd afspilles analogt.

### Mapping scheme

En agenda for hvorledes et givet sensor-input mappes til en given MIDI-parameter

### Sensorkonfiguration

En sensorkonfiguration indeholder:

* ID for en specifik sensor-enhed
* MIDI-parameter
* Mapping scheme
* Lydpakke

### Buffer

Et midlertidigt hukommelseslager mellem to moduler i BodyRock3000.

### Preset

Et sæt af sensorkonfigurationer, der simultant[[1]](#footnote-1) kan indlæses på systemet

# Kravspecifikation

Kravspecifikationen er udfærdiget med basis i en Use Case baseret tilgang. Den indeholder en beskrivelse over relevante aktører for systemet samt Fully Dressed Use Cases.

## Use Case diagram

Herunder ses et UC diagram for systemet BodyRock3000. Bemærk at UC7 og UC8 er systeminitierede via UC6.



## Aktørbeskrivelse

### Bruger

|  |  |
| --- | --- |
| **Aktørnavn:** | **Bruger** |
| **Type:** | Primær |
| **Beskrivelse:** | Ønsker at benytte systemet, og indstille diverse konfigurationer for systemet ved hjælp af Rock-enheden. Herudover kan Bruger installere lydfiler |

### MIDI-modtager

|  |  |
| --- | --- |
| **Aktørnavn:** | **MIDI-modtager** |
| **Type:** | Sekundær |
| **Beskrivelse:** | Ekstern MIDI-enhed som overholder MIDI-standarden[[2]](#footnote-2). Denne vil i praksis kunne være alt fra synthesizere, drum machines, pedal-board loopere, samt nogle lys-systemer. |

### Højtalersystem

|  |  |
| --- | --- |
| **Aktørnavn:** | **Højtalersystem** |
| **Type:** | Sekundær |
| **Beskrivelse:** | Det eksterne højtalersystem modtager lydsignal fra audio-outputtet, og afspiller den ønskede lyd |

## Use Case beskrivelser

For alle use cases gælder det, at i enhver undermenu, findes der altid mulighed for at annullere og vende tilbage til hovedmenuen.

### Forbind Body og Rock

|  |  |
| --- | --- |
| **Navn** | **Forbind Body og Rock** |
| **Use Case ID** | 1 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Bruger |
| **Sekundær aktør** | Ingen |
| **Initiator** | Bruger |
| **Prækondition** | Bruger har adgang til Body og Rock |
| **Formål** | Body og Rock er forbundne og kan anvendes |
| **Postkondition** | Body og Rock er klar til brug |

|  |
| --- |
| **Hovedscenarie** |
| 1. Bruger tænder Body og Rock 2. Systemet viser hovedmenuen på Rocks display 3. Bruger vælger ”Forbind enhed” på Rocks touch-displayet 4. Rock søger efter Body 5. Bruger vælger Body på touch-displayet   [Undtagelse 5.a: Rock kunne ikke finde en Body-enhed]   1. Systemet forsøger at oprette forbindelse mellem Body og Rock 2. Systemet giver besked om godkendt forbindelse på displayet   [Undtagelse 7.a: Enhederne kunne ikke forbindes]   1. Systemet returnerer til hovedmenu, og UC1 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 5.a: Rock kunne ikke finde en Body-enhed]   1. 1. Systemet giver besked om at der ikke er nogen Body-enheder   2. Bruger slukker enhederne, og der fortsættes fra punkt 1 i hovedscenariet (ved gentagne fejl kontaktes kundeservice)  [Undtagelse 7.a: Enhederne kunne ikke forbindes]   1. 1. Systemet giver besked om ikke oprettet forbindelse på displayet 2. 2. Bruger fejlfinder på enhederne, og der fortsættes fra punkt 2 i hovedscenariet |

### Installér lydpakker

|  |  |
| --- | --- |
| **Navn** | **Installér lydpakker** |
| **Use Case ID** | 2 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Bruger |
| **Sekundær aktør** | Ingen |
| **Initiator** | Bruger |
| **Prækondition** | Der er fysisk forbundet en USB-hukommelsesenhed til Rock |
| **Formål** | Der installeres en lydpakke |
| **Postkondition** | En lydpakke er installeret |

|  |
| --- |
| **Hovedscenarie** |
| 1. Bruger vælger ”Installér lydpakke” i Rocks hovedmenu 2. Bruger vælger ”Importer lydpakke” i menu   [Undtagelse 2.a: Bruger vælger ”Slet lydpakke”]   1. System opretter forbindelse til USB-hukommelsesenhed   [Undtagelse 3.a: Der kan ikke oprettes forbindelse]   1. Bruger vælger lydpakke fra liste over kildens lydpakker 2. System installerer lydpakke, og UC2 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 2.a: Bruger vælger ”Slet lydpakke”]  1. Bruger vælger lydpakke fra liste over systemets lydpakker  2. Bruger bekræfter sletning af lydpakke  3. System sletter lydpakke, og UC2 afsluttes  [Undtagelse 3.a: Der kan ikke oprettes forbindelse]   1. 1. Der vises en fejlmeddelelse, og UC2 afsluttes |

### Konfigurer sensorer

|  |  |
| --- | --- |
| **Navn** | **Konfigurer sensorer** |
| **Use Case ID** | 3 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Bruger |
| **Sekundær aktør** | Ingen |
| **Initiator** | Bruger |
| **Prækondition** | Rock er tændt, og mindst én lydpakke er installeret |
| **Formål** | Der oprettes, nedlægges eller ændres i en sensorkonfiguration |
| **Postkondition** | En sensorkonfiguration er oprettet, nedlagt eller ændret |

|  |
| --- |
| **Hovedscenarie** |
| 1. Bruger vælger ”Konfigurer sensor” i Rocks hovedmenu 2. Bruger vælger ”Opret ny” i menuen   [Undtagelse 2.a: Bruger vælger ”Rediger eksisterende”]  [Undtagelse 2.b: Bruger vælger ”Slet”]   1. Bruger indtaster konfigurationens navn   [Undtagelse 3.a: Det indtastede navn er optaget]   1. Bruger vælger sensor på liste over sensorer 2. Bruger vælger MIDI-parameter på liste over MIDI-parametre 3. Bruger vælger mapping scheme på liste over mapping schemes 4. Bruger vælger lydpakke på liste over installerede lydpakker 5. Bruger vælger at tilføje sensor til et eller flere presets på liste over eksisterende presets 6. Bruger præsenteres for sine opsætningsvalg, og UC3 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 2.a: Bruger vælger ”Rediger eksisterende”]  1. Bruger vælger sensorkonfiguration fra liste over sensorkonfigurationer  2. Der fortsættes fra punkt 4 i hovedscenariet, hvor alle menuer herefter er udfyldt med  data fra den valgte opsætning  [Undtagelse 2.b: Bruger vælger ”Slet”]  1. Bruger vælger sensorkonfiguration fra liste over sensorkonfigurationer  2. Bruger bekræfter valg   1. 3. Sensorkonfiguration slettes, og UC3 afsluttes   [Undtagelse 3.a: Det indtastede navn er optaget]  1. Systemet viser fejlmeddelelse og der fortsættes fra hovedscenariets punkt 3 |

### Konfigurer preset

|  |  |
| --- | --- |
| **Navn** | **Konfigurer preset** |
| **Use Case ID** | 4 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Bruger |
| **Sekundær aktør** | Ingen |
| **Initiator** | Bruger |
| **Prækondition** | Der findes mindst én sensorkonfiguration i systemet |
| **Formål** | At oprette, redigere eller slette et preset |
| **Postkondition** | Der er foretaget en ændring i et preset |

|  |
| --- |
| **Hovedscenarie** |
| 1. Bruger vælger ”Konfigurer preset” i Rocks hovedmenu 2. Bruger vælger ”Opret nyt” i menuen   [Undtagelse 2.a: Bruger vælger ”Rediger eksisterende”]  [Undtagelse 2.b: Bruger vælger ”Slet”]   1. Bruger indtaster ønsket navn på preset 2. Bruger markerer de ønskede konfigurationer på liste over sensorkonfiguration, og tilføjer disse til det nuværende preset 3. Bruger trykker på ”Fortsæt” 4. Systemet præsenterer de valgte indstillinger på displayet, og anmoder om bekræftelse 5. Bruger accepterer ændringer, og UC4 afsluttes   [Undtagelse 7.a: Bruger vælger ”Fortryd”] |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 2.a: Bruger vælger ”Rediger eksisterende”]   * + - * 1. Bruger vælger preset fra liste over eksisterende presets         2. Der fortsættes fra punkt 4 i hovedscenariet, hvor alle menuer herefter er udfyldt med data fra det valgte preset   [Undtagelse 2.b: Bruger vælger ”Slet”]   * + - * 1. Bruger vælger preset fra liste over eksisterende presets         2. Bruger bekræfter valg   [Undtagelse 2.b.2.a: Bruger vælger ”Fortryd”]  [Undtagelse 2.b.2.a: Bruger vælger ”Fortryd”]   * + - * 1. Der fortsættes fra punkt 2.b.1 i Undtagelser   [Undtagelse 7.a: Bruger vælger ”Fortryd”]   * + - * 1. Der fortsættes fra punkt 4 i hovedscenariet |

### Vælg preset

|  |  |
| --- | --- |
| **Navn** | **Vælg preset** |
| **Use Case ID** | 5 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Bruger |
| **Sekundær aktør** | Ingen |
| **Initiator** | Bruger |
| **Prækondition** | Mindst ét preset er konfigureret |
| **Formål** | Bruger vælger et preset |
| **Postkondition** | Der er skiftet til et nyt preset |

|  |
| --- |
| **Hovedscenarie** |
| * 1. Bruger vælger preset på Body   2. Body sender et signal til Rock med det valgte preset   [Undtagelse 2.a. Afsendelse mislykkedes]   * 1. Rock opdaterer det aktive sæt af sensorkonfigurationer på baggrund af det valgte preset, og UC5 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 2.a: Afsendelse mislykkedes]  1. Body afgiver fejlmeddelelse  2. Bruger afvikler UC1 ”Forbind Body og Rock”, og der fortsættes fra punkt 1 i hovedscenariet. |

### Indsamle sensordata

|  |  |
| --- | --- |
| **Navn** | **Indsaml sensordata** |
| **Use Case ID** | 6 |
| **Samtidige forekomster** | 1..\* (inklusiv denne) |
| **Primær aktør** | BodyRock3000 |
| **Sekundær aktør** | Ingen |
| **Initiator** | BodyRock3000 |
| **Prækondition** | Body og Rock er forbundet, og der er forbundet mindst én sensor |
| **Formål** | At sende rådata fra Body til Rock |
| **Postkondition** | Systemet har indsamlet og gemt sensordata i buffer på Rock |

|  |
| --- |
| **Hovedscenarie** |
| 1. Sensor genererer rådata 2. Body sender rådata trådløst til Rock   [Undtagelse 2.a: Afsendelse mislykkedes]   1. Rock indlæser data i buffer 2. Rock er klar til at modtage nye data og UC6 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 2.a: Afsendelse mislykkedes]   1. 1. Body afgiver fejlmeddelelse |

### Generér MIDI

|  |  |
| --- | --- |
| **Navn** | **Generér MIDI** |
| **Use Case ID** | 7 |
| **Samtidige forekomster** | 1..\* (inklusiv denne) |
| **Primær aktør** | Ingen |
| **Sekundær aktør** | MIDI-Modtager |
| **Initiator** | Ingen |
| **Prækondition** | Mindst ét preset er konfigureret for den aktuelle sensor |
| **Formål** | At konvertere rådata til MIDI-signaler som sendes til hhv. MIDI-afspiller og MIDI-udgang |
| **Postkondition** | MIDI-signaler, baseret på rådata, er sendt til hhv. MIDI-afspiller og MIDI-udgang |

|  |
| --- |
| **Hovedscenarie** |
| 1. Konverteringsmodul læser data fra buffer   [Undtagelse 1.a: Aflæsning af buffer mislykkedes]   1. Konverteringsmodul[[3]](#footnote-3) omdanner rådata til MIDI-meddelelser 2. Konverteringsmodul henter lydpakken ift. sensorkonfiguration for den pågældende sensor 3. Signalet fra konverteringsmodul sendes til hhv. MIDI-afspiller og MIDI-udgang, og UC7 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 1.a: Aflæsning af buffer mislykkedes]   1. 1. Rock afgiver en fejlmeddelelse 2. 2. Der fortsættes fra punkt 1 i hovedscenariet. |

### Afspil lyd

|  |  |
| --- | --- |
| **Navn** | **Afspil lyd** |
| **Use Case ID** | 8 |
| **Samtidige forekomster** | 1 (inklusiv denne) |
| **Primær aktør** | Ingen |
| **Sekundær aktør** | Højtalersystem |
| **Initiator** | Ingen |
| **Prækondition** | Mindst én lydpakke er installeret, mindst ét preset et konfigureret, og MIDI-signal modtages fra konverteringsmodul |
| **Formål** | At afspille lyd genereret af BodyRock3000 |
| **Postkondition** | Et lydsignal, der repræsenterer valgt preset, lydpakke og MIDI-signal, er genereret og sendt til højtalersystem |

|  |
| --- |
| **Hovedscenarie** |
| 1. Lydmodul aflæser valgt lydpakke   [Undtagelse 1.a: Aflæsning af lydpakke mislykkedes]   1. Lydmodulet aflæser valgte preset   [Undtagelse 2.a: Aflæsning af preset mislykkedes]   1. Lydmodulet modtager MIDI-signaler   [Undtagelse 3.a: Modtagelse af MIDI-signaler mislykkedes]   1. Lydfil der skal manipuleres vælges ud fra MIDI-signal 2. Valgt lydfil manipuleres i overensstemmelse med indkommende MIDI-signal (LFO, pitch, volumen, tænd, sluk, eller lignende) 3. Lydsignal udsendes analogt på en port til højtalersystem, via en DAC, og UC8 afsluttes |

|  |
| --- |
| **Undtagelser** |
| [Undtagelse 1.a: Aflæsning af lydpakke mislykkedes]   1. 1. Rock afgiver fejlmeddelelse   2. Bruger trykker på ”Forsøg igen”, og der fortsættes fra punkt 1 i hovedscenariet.  [Undtagelse 1.a.2.a: Bruger trykker på ”Afbryd”]  1. Der returneres til Hovedmenu, og UC8 afsluttes  [Undtagelse 1.a.2.a: Bruger trykker på ”Afbryd”]  1. Der returneres til Hovedmenu, og UC8 afsluttes  [Undtagelse 2.a: Aflæsning af preset mislykkedes]   1. 1. Rock afgiver fejlmeddelelse   2. Bruger trykker på ”Forsøg igen”, og der fortsættes fra punkt 2 i hovedscenariet.  [Undtagelse 2.a.2.a: Bruger trykker på ”Afbryd”]  1. Der returneres til Hovedmenu, og UC8 afsluttes  [Undtagelse 2.a.2.a: Bruger trykker på ”Afbryd”]  1. Der returneres til Hovedmenu, og UC8 afsluttes  [Undtagelse 3.a: Modtagelse af MIDI-signal mislykkedes]  1. Der fortsættes fra punkt 3 i hovedscenariet |

## Krav til Mapping Scheme

Følgende features skal kunne indstilles for et givent Mapping Scheme

**Note:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Skala** | Kromatisk | | | Dur | | | | Mol | | | |
| **Grundtone** | c | c# | d | d# | e | f | f# | g | g# | a | a# |
| **Retning** | Opadgående | | | | | Nedadgående | | | | | |

**Velocity:**

|  |  |
| --- | --- |
| **Lower Threshold** | 0-127 |

**Control Change:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | Absolute | Relative | |
| **Channel num** | 0-127 | | |
| **Min value (for absoulte)** | 0-127 | | |
| **Max value (for absoulte)** | 0-127 | | |
| **Følsomhed (for relative)** | Slow | Medium | Fast |

**Midi channel:**

|  |  |  |
| --- | --- | --- |
| **Type** | Omni | Specific channel |
| **Channel no (for specific)** | 1-16 | |

## Ikke funktionelle krav

### Generelt

* 1. Den trådløse forbindelse mellem Body og Rock skal have en rækkevidde af minimum 10 meter.

### Body

* 1. Skal ved aktiv brug kunne holde strøm i min. 2 timer.
  2. Skal forsynes af et 9V batteri?
  3. Skal have en power-indikator i form af en diode, der lyser grønt når enheden er tændt.
  4. Skal have en fejl-indikator i form af en diode, der lyser rødt ved fejl.[[4]](#footnote-4)
  5. Skal kunne styre op til 16 sensorer på én gang.

### Rock

* 1. Skal have en strømforsyning, der tilsluttes et 220V 50 Hz el-net.
  2. Skal have en power indikator i form af en diode, der lyser grønt når enheden er tændt.
  3. Skal have en fejlindikator i form af en diode, der lyser rødt ved fejl.
  4. Skal have en brugergrænsefladen der er en GUI baseret på Linux Kubuntu.
  5. Skal have et MIDI output af typen ”5-pin DIN hun”
  6. Skal have et lydoutput af typen ”stereo mini-jack hun”.

# Systemarkitektur Hardware

## Overordnet arkitektur

Dette afsnit beskriver systemarkitekturen for ”BodyRock3000”. Yderligere beskrivelse kan ses i projektformuleringen og kravspecifikationen.

Formålet er at identificere:

* De overordnede komponenter, samt at bestemme deres grænseflader
* De eksterne komponenter, samt at beskrive dem
* Arbejdsopgaver for projektets design- og implementeringsfase

### Domæne model BodyRock3000



På figuren herover ses en overordnet domæne model med konteksten for systemet ”BodyRock3000”.

Notér at Body og Rock har hver sit bluetooth-modul, der benyttes som interface mellem deres indbyrdes kommunikation.

## Body HW Arkitektur

### BDD Body



**Body** består af Controller, Spændingsforsyning, Bluetooth-modul og Preset knapper.

**Controller** er hjernen i Body. Denne står for indsamling, pakning og videre afsendelse af data fra sensorBus til Bluetooth-modul. Den afkoder og videresender desuden information fra Preset knapper.

**Spændingsforsyning** leverer spænding til de resterende komponenter i Body.

**Bluetooth-modul** videresender data fra Controller til Rocks Bluetooth-modul.

**Preset knapper** bruges af brugeren til vælge preset.

### Allokeringsdiagram Body



**Controller** implementeres på en **PSoC4**

**Spændingsforsyningen** implementeres som hhv. en **Reguleringskreds** og et **Batteri**

**Bluetooth-modul** implementeres med komponenten **HC-05**

**Preset knapper** implementeres med en **3x4 knapmatrix**

### IBD Body

Herunder ses et internt block diagram for Body. Dette illustrerer interne forfindelser i Body:



**Batteri** generer den strøm til systemet.

**Reguleringskreds** bruges til at transformere den indgående batterispænding til 3,3V, som de resterende Body enheder samt eksterne sensorer kører på.

**PSoC4** er systemets kontrol-enhed. Den benyttes til at indsamle, bearbejde og pakke data fra sensorbus og 3x4 knapmatrix, og sender disse til bluetooth-modulet RN42. Den kontrollerer desuden systemets statusLEDer.

**StatusLEDer** består af består af hhv. en grøn- og en rød LED, der indikerer hhv. power status (hvorvidt enheden er tændt) og error status (hvorvidt der opstår fejl på enheden).

**3x4 knapmatrix** benyttes af brugeren til at skifte mellem forskellige systempresets.

**HC-05** Forbinder Body og Rock trådløst via bluetooth. Den er forbundet til PSoC4 med en RX og en TX forbindelse

**Sensorbus** er en I2C bus, der består af hhv. GND, VDD, SDA og SCL forbindelserne.[[5]](#footnote-5)

## Sensor IBD (Generisk)

Alle sensor-enheder, kobles på sensorbus som følger:



Sensorenhederne er alle koblet parrallelt op med en strømforsyning (GND og VCC) samt I2C forbindelserne SCL (serial clock) og SDA (serial data). SDA forbindelsen er non-atomic, da både master (Body) og slave (den pågældende sensorenhed) kan tage styring over linjen. Sensorenhedernes interfaces er lavet således, at de kan kobles i forlængelse af hinanden.

## Rock HW Arkitektur

### BDD Rock



**Rock** består af Functionalities, Bluetooth-modul og Touchscreen.

**Functionalities** er en abstraktion for alle Rocks logiske funktionaliteter. Se ”BDD Functionalities”[[6]](#footnote-6) for yderligere beskrivelse.

**Bluetooth-modul** modtager data fra Bodys Bluetooth-modul.

**Touchscreen** benyttes af brugen til at operere Rocks GUI og ændre indstillinger for Rock.

### Allokeringsdiagram Rock



**Functionalities** implementeres på et **Raspberry Pi** samt en **ESI MIDI Mate** til output af MIDI.

Som **Touchscreen** benyttes en **Adafruit PiTFT.**

**Bluetooth-modul** implementeres med komponenten **RN42**

## Grænsefladebeskrivelse

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Navn** | **Beskrivelse** | **Signaler** | **Område** | **Kommentar** |
| sensorbus | I2C [[7]](#footnote-7) bus forbundet med RJ11-stik | GND | 0 V |  |
| VCC | 3,3 V |  |
| SCL | 0-3,3 V | I2C seriel clock signal.  Følger 3,3V CMOS standard |
| SDA | 0-3,3 V | I2C seriel data signal Følger 3,3V CMOS standard |
| BluetoothCon | Forbindelse fra Controller til RN42-modul | GND | 0 V |  |
| VCC | 3,3 V |  |
| RXD | 0-3,3 V | RS232 seriel recieve data signal Følger 3,3V CMOS standard |
| TXD | 0-3,3 V | RS232 seriel transmit data signal Følger 3,3V CMOS standard |
| Reconnect | 0-3,3 V | 0V =deaktiveret, 3,3 V aktiveret. Ved aktiveret genetablerer RN-42 modulet bluetoothforbindelsen mellem Body og Rock[[8]](#footnote-8) |
| lineOut | Ubalanceret signal med 3.5 mm (1/8") 3-conductor TRS minijack stik. | Left channel (tip) | -0.447-0.447 V | Consumer audio. [[9]](#footnote-9)  Nominal level: -10 dBV |
| Right channel (ring) | -0.447-0.447 V |
| Ground (sleeve) | 0V |  |

**MIDI out / ekstern MIDI**

MIDI beskeder, der overholder MIDI protokollen som defineret af Midi Manufacturers Association[[10]](#footnote-10), sendes via ALSA[[11]](#footnote-11) til et ESI MIDI Mate USB til MIDI lydkort.[[12]](#footnote-12) Denne har både et input-kabel og et output-kabel. Det er kun output-kablet der benyttes. Det anvender følgende pin layout:

|  |  |
| --- | --- |
| Pin | Signal |
| 1 | Ikke anvendt |
| 2 | Ground |
| 3 | Ikke anvendt |
| 4 | Nuværende Sync |
| 5 | Nuværende Kilde |



### Trådløs kommunikations protokol

Den trådløse kommunikationsprotokol er til for at Rock kan tolke hvilke sensordata og preset(REFERENCE til preset) der modtages fra Body. Fra Body sendes der to typer char arrays.

Type 1:   
Et sensordata array der indeholder seks pladser. Der er oprettet et sensordata array pr tilsluttet sensor.

Type 2:  
Et preset array der indeholder tre pladser.

Sensordata array:

|  |  |  |
| --- | --- | --- |
| Plads: | Indhold: | Beskrivelsen: |
| 0 | 0x0F | Definition på sensordata forsendelsen |
| 1 | 0-15 | Sensor ID |
| 2 | 0-127 | Sensorkoordinat x værdi |
| 3 | 0-127 | Sensorkoordinat y værdi |
| 4 | 0-127 | Sensorkoordinat z værdi |
| 5 | 0 | Nul-terminering |

Preset array:

|  |  |  |
| --- | --- | --- |
| Plads: | Indhold: | Beskrivelsen: |
| 0 | 0xF0 | Definition på presetdata forsendelsen |
| 1 | 1-9 | Preset værdi |
| 2 | 0 | Nul-terminering |

Forsendelsen af sensordata arrayene skal ske 50 gange i sekundet(REFERENCE til HARDWARE OPSÆTNINGEN FOR BT). Presetdata forsendelsen skal kun ske når der er ændret preset ved at trykke på knap-matrix’en.

(AFGRÆNDSNING INGEN PRESET)

**Oversigt over ID på Body samt sammenhæng mellem type, navn og parametre.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SensorID | Definition | Sensor navn | Type | Parametre |
| 0 | ACC1\_ID | ADXL345 | Accelerometer | (X,Y,Z) |
| 1 | GYR1\_ID | MPU6050 | Gyroskop | (X,Y,Z) |
| 2 | FLE1\_ID | FLEXIFORCE A301 | Tryksensor | (X) |
| 3 | PRX1\_ID | HC-SR04 | Proximitysensor | (X) |
| 4 | ACC2\_ID | ADXL345 | Accelerometer | (X,Y,Z) |
| 5 | GYR2\_ID | MPU6050 | Gyroskop | (X,Y,Z) |
| 6 | FLE2\_ID | FLEXIFORCE A301 | Tryksensor | (X) |
| 7 | PRX2\_ID | HC-SR04 | Proximitysensor | (X) |
| 8 | ACC3\_ID | ADXL345 | Accelerometer | (X,Y,Z) |
| 9 | GYR3\_ID | MPU6050 | Gyroskop | (X,Y,Z) |
| 10 | FLE3\_ID | FLEXIFORCE A301 | Tryksensor | (X) |
| 11 | PRX3\_ID | HC-SR04 | Proximitysensor | (X) |
| 12 | ACC4\_ID | ADXL345 | Accelerometer | (X,Y,Z) |
| 13 | GYR4\_ID | MPU6050 | Gyroskop | (X,Y,Z) |
| 14 | FLE4\_ID | FLEXIFORCE A301 | Tryksensor | (X) |
| 15 | PRX4\_ID | HC-SR04 | Proximitysensor | (X) |

# Systemarkitektur Software

I følgende afsnit beskrives softwarearkitekturen for systemet ”BodyRock3000”. Softwarearkitekturen er opbygget på baggrund af projektformuleringen og kravspecifikationen.

Formålet er at skabe en ramme for softwaren-udviklingen, hvorfra delopgaver kan uddelegeres til udviklere.

Softwarearkitekturen består af:

* Identifikation af problemer, klasser og metoder med udgangspunkt i en detaljeret domænemodel for system og det software-moduler.
* Oprettelse af skelet for videre implementering af Body SW vha. et simpelt sekvensdiagram for program flow.
* Oprettelse af skelet for videre implementering af Rock SW vha. applikationsmodeller for systemets moduler

Det er valgt at lave sidstnævnte applikationsmodellerne for modulerne/modulgrupperne internt frem for direkte Use Case baseret. Applikationsmodellerne indeholder således funktionaliteter for op til flere Use Cases, men kun afgrænset til det/den pågældende modul(-gruppe). Denne tilgang leder op til en uddelegering af design og implementering af systemet til mindre teams, der står for hver deres modul.

## Detaljeret domænemodel for software-moduler

Herunder ses en domænemodel for kommunikationen mellem systemets interne SW-moduler.



## Klasse-identifikation

På baggrund af domænemodellen for softwaremoduler, identificeres følgende klasser for hhv. Body og Rock:

**Body** generer data ud fra brugerens påvirkning af sensorer og tryk på knapmatrix. Disse data pakkes i Body’s Serial Unit og videresendes via Bluetooth-modul.

* **Main** er Body’s central for indsamling af sensordata og håndtering af knap-tryk.
* **SerialUnit** står for at pakke de fra Body genererede data og adaptere dem til Bluetooth-moduls interface
* **Sensor** står for at aflæse alle tilkoblede sensorer, standardiserer sensordata samt sende dem til SerialUnit
* **Knap** er 4x3 knapmatrix afkoderen. Den meddeler Main at der er ønsket nyt preset.
* **Bluetooth-modul** er driveren til det fysiske Bluetooth-modul HC-05, der muliggør afsendelse af datapakker fra Body til Rock.

**Rock** afkoder data fra Body og generer MIDI og lyd på baggrund af de indstillede værdier i dataStorage. Sidstnævnte indstilles fra GUI.

* **GUI** er systemets bruger-interface, der benyttes til bruger-indstillinger for systemet.
* **Controller** er Rocks styrings-hub. Den konfigurerer dataStorage på baggrund af brugeren interaktion med GUI, og konfigurer Receiver (dataAdapter), MIDI-module (MIDI-mapper) og Sound-module (Linux Sampler) ved modtagelse af besked om preset-skift fra Receiver.
* **DataStorage** opbevarer data på systemet. Denne består af Preset-bank, Sensorkonfiguration og Sample Bibliotek.
  + **Preset-bank** indeholder systemets samlede presetindstillinger. Den består af op til 9 Presets, der hver kan tilknyttes op til 16 Sensorkonfigurationer
  + **SensorConfiguration** indeholder systemets brugerindstillinger for hver sensor-enhed, der kan tilkobles.
    - **MappingScheme** indeholder policy for hvordan der for den givne sensorkonfiguration genereres MIDI.
  + **Samplebibliotek** indeholder de lyd-samples, som Linux Sampler kan afspille.
* **Receiver** står for modtagelse og udpakning af data fra Body via Bluetooth, samt bestemmelse af datatype og heraf modtager på videre afsendelse. Receiver består af følgende sub-moduler:
  + **Bluetooth-modul** er driveren til det fysiske Bluetooth-modul RN-42, der modtagelse afsendelse af datapakker fra Body til Rock.
  + **SerialUnit** står for at udpakke de fra Body modtagne data.
  + **dataAdapter** står for genkendelse af typen for de udpakkede data (hhv. sensor-data eller preset-valg) samt adaptering og afsendelse af disse til det relevante modul i systemet. Adaptering kan i fremtidige systemudgaver indeholde implementering af velocity-kurve for sensorData. Presetvalg og systeminfo videresendes til Controller og sensordata videresendes til MIDI-module.
* **MidiModule** står for at klargøre MIDI-signaler til videre afsendelse på baggrund af modtagne sensordata og nuværende preset. Disse videresendes til software frameworket ALSA, der står for videre processering af data. MIDI-module består af følgende sub-moduler:
  + **MIDI Mapper** holder den funktionalitet, der lader systemet konvertere sensordata til MIDI signaler. Jf. det nuværende aktive preset, består denne af en liste af SensorConfiguration, der hver har et tilknyttet sensorID til aflæsning af sensor data og et tilknyttet plads i MIDI Signal Bufferen, der mappes til.
  + **MIDI Signal Buffer** indeholder de MIDI signaler, der skal videresendes til ALSA.
  + **ALSA Adapter** står for at interface til ALSA, så værdierne fra MIDI Signal Buffer aflæses korrekt.
* **ALSA-functions** er de i systemet benyttede funktionaliteter fra ALSA **(Advanced Linux Sound Architecture)**. ALSAer et software framework, der leverer et API for systemets lyd-kort drivere, og har i dette tilfælde til opgave at afspille lydsamples samt generere line- og MIDI output.
  + **Linux Sampler** afspiller lyd-samples fra Sample Bibliotek på baggrund af modtaget MIDI Signal og givne indstillinger i Sensorkonfiguration.
  + **MIDI-generator** er driveren, der genererer den MIDI på Rocks MIDI output.

## Body

Body Control er styresystemet på Body-delen af BodyRock3000. Systemet står for at samle sensordata fra de fire[[13]](#footnote-13) forskellige sensorer via I2C, konverterer disse data og videresende dem vha. en UART forbundet til et Bluetooth modul.

### Klassediagram Body



Figur 1 UML af Body Control

#### Main

Main er Bodys control klasse. Main står for at initerer I2C, UART og de tilkoblede sensorer samt at kalde funktioner der kan læse sensorer, standardisere sensordata og sende sensor- og presetdata via SerialUnit. På nuværende tidspunkt skal initieringsfunktionerne til sensorene stå i Main klassen for at funktionerne kan implementeres.

#### Sensor

Sensor står for at klargøre I2C, læse sensordata via I2C, gemme sensordata i en midlertidig buffer, konvertere sensordata til et generisk interval fra 0-127. Sensor sender også de generiske sensordata til SerialUnit. Sensor indeholder også I2C fejlhåndtering.

#### SerialUnit

Står for at klargøre sensordata til afsendelse via UART og videresende sensordata’en over uart.

#### Knap

Ikke implementeret da denne klasse omhandler preset.

#### Funktioner

|  |  |
| --- | --- |
| Funktioner \* | Beskrivelse |
| initSensors() | Kalder de forskellige initieringsfunktioner for hver sensor. |
| initADXL345() | Initialiserer accelerometeret. |
| initMPU6050() | Initialiserer gyroskopet. |
| initUART() | Starter UART komponenten. |
| setupI2C() | Starter I2C komponenten. Returnerer master status og clearer status flaget. |
| readAllSensors() | Kalder funktionen readI2C for hvert sensorregister der er, at læse fra. |
| readI2C(int numOfReg) | Modtager det register nummer der skal læses fra og følger herefter I2C protokollen for læsning og gemmer det læste data. |
| convSensData() | Samler sensoraksernes most-significant og least-significant byte til en.  Konverterer herefter til MIDI-skalaen 0-127.  Kalder funktionen setdataArray(int, int, int, int). |
| setdataArray(int, int, int, int) | Fylder array med sensor ID og data fra convSensData() |
| sendDataArray() | Sender et array med data for hver sensor. |
| handleI2CError() | Funktion som kaldes ved fejl i I2C forbindelsen. Genererer en stop condition. |
| setSensArray(int, int, int) | Fylder det multidimensionelle array. |

\*Alle preset funktioner er ikke skrevet da man har valgt ikke at implementere preset I den iteration.

#### Uddybende funktionsbeskrivelser

##### void initSensors(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Kalder de initieringsfunktioner til de specifikke sensorer som er sat op. |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void initADXL345(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Sætter de første 6 pladser i sensArray, plads [numOfSens][0] med ADXL345 ID, plads [numOfSens][1] med registeret som der skal læses fra og plads [numOfSens][2] med en default data værdi -1.  Opsætter sensoren til I2C jf. I2C-protokollen. |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void initMPU6050(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Sætter de første 6 pladser i sensArray, plads [numOfSens][0] med ADXL345 ID, plads [numOfSens][1] med registeret som der skal læses fra og plads [numOfSens][2] med en default data værdi -1.  Opsætter sensoren til I2C jf. I2C-protokollen. |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void initUART(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Starter SCB\_init() og SCB\_Enable(). Efter denne funktion er kaldt er UART komponenten enabled og klar til at operere.  Når konfigurationen er sat til ”Unconfigured SCB”, skal komponenten først initialiseres til at operere i en af følgende konfigurationer: I2C, SPI, UART eller EZ I2C. Ellers kan denne funktion ikke enable komponenten  Returnerer master status og clearer status flaget. |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void setupI2C(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Starter SCB\_init() og SCB\_Enable(). Efter denne funktion er kaldt er UART komponenten enabled og klar til at operere.  Når konfigurationen er sat til ”Unconfigured SCB”, skal komponenten først initialiseres til at operere i en af følgende konfigurationer: I2C, SPI, UART eller EZ I2C. Ellers kan denne funktion ikke enable komponenten |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void readAllSensors(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Kalder funktionen readI2C for hvert sensorregister der er, at læse fra. |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |
|  |  |

##### void readI2C(int)

|  |  |
| --- | --- |
| **Beskrivelse** | Følger I2C protokollen for at læse fra et register. Protokollen er generisk ved at bruge pladserne i det multidimensionelle array, sensArray.  Eks. Sendes der en start til sensArray[numOfReq][0], som er adressen på sensoren.  For yderligere beskrivelse se I2C protokol.[[14]](#footnote-14)  Dataen som læses gemmes i sensArray[numOfReq][2]. |
| **Parametre** | En int indeholdende den række der ønskes læst fra i sensArray. |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void convSensData(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Tager data fra sensArray, og samler most significant og least significant for de givne akser: x, y og z.  Konverterer herefter til MIDI-skalaen 0-127.  Kalder funktionen setdataArray(int, int, int, int). |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void setdataArray(int, int, int, int)

|  |  |
| --- | --- |
| **Beskrivelse** | Fylder dataArray plads 0 med sensor ID fra 0-15 og de næste 3 pladser med data fra x, y og z aksen fra sensoren. De resterende pladser i dataArray sættes til 1 |
| **Parametre** | Fire int: ID, x\_data, y\_data, z\_data |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void sendDataArray(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Opretter et array for hver sensor. Funktionen fylder array plads 0 med START\_SENSDATA, de næste 4 pladser sættes med værdierne fra dataArray i samme rækkefølge som de er sat i setdataArray(). Den sidste plads i arrayet sættes til NUL. Herefter sendes array over BT vha. UART\_UartPutString() |
| **Parametre** | ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void handleI2CError(void)

|  |  |
| --- | --- |
| **Beskrivelse** | Sender en stopkondition |
| **Parametre** | ingen |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

##### void setSensArray(int, int, int)

|  |  |
| --- | --- |
| **Beskrivelse** | Fylder det multidimensionelle array, sensArray på følgende vis:  Den første int fyldes i plads [numOfReqToRead[0]. Den næste int fyldes i plads [numOfRegToRead[1] og den sidste int i plads [numOfReqToRead[2].  Den globale variabel numOfRegToRead inkrementeres hver gang funktionen kaldes. |
| **Parametre** | Tre ints: device adressen, registeret som ønskes læst fra og data. |
| **Returværdi** | Ingen |
| **Påvirkninger** | Ingen |

## Rock

Softwaren på Rock implementeres event-baseret og indeholder således en række selvstændige tråde. De vil være opdelt i to hovedkategorier, bestemt ud fra hvor kritisk afviklingshastigheden er for den musikalske oplevelse ved brug af systemet. Disse er benævnt som følger:

* **Slow Lane**
* **Fast Lane.**

Visse software-moduler vil gå igen for begge baner, men hastigheden af disse (prioriteten i systemets scheduler), vil afhænge af hvilken tråd, de indgår i.

Der vil i følgende afsnit være udfærdiget applikationsmodeller for de individuelle moduler. Hver af disse moduler har sin egen tråd, og applikationsmodellerne rummer dermed de enkelte trådes funktionalitet.

### Tråd kommunikation

I dette afsnit skitseres overordnet hvordan kommunikation trådene imellem vil foregå.

### Slow Lane

Denne bane håndterer de software-moduler i systemet, hvis afvikling ikke er kritisk for den musikalske oplevelse ved benyttelse af BodyRock3000.

#### SW-moduler i Slow Lane

Bruger-indstillinger af systemet (UC 2-4) tildeles en lav prioritet i systemets scheduler, da disse kun anvendes sjældent, og ikke er latens-kritiske. Følgende softwaremoduler tager sig af disse indstillinger:

* GUI
* DataStorage
* Controller

Controller og DataStorage vil blive brugt i hhv. Slow- og Fast Lane afhængigt af hvilken sammenhæng de benyttes i.

#### Applikationsmodel for Slow Lane

*Slow Lane* benyttes i systemet som betegnelse for trådene ”GUI” og ”Controller”. Denne del af systemet står for at modtage input fra brugeren, samt for at administrere *Databank* i henhold til disse.

Herudover administrerer Controlleren ligeledes trådene i *Fast Lane*, hvilket bl.a. indebærer at modtage presets fra *Receiver*, samt at sende opdaterede data fra *Databank* til *MidiModule* og *ALSA-Functions*.

##### Sekvensdiagram

##### 

I dette diagram ses en beskrivelse af kommunikation mellem GUI og Controller tråden når brugeren skal oprette en ny sensorkonfiguration.  
Denne kan betragtes som repræsentativ for den gennerelle kommunikation mellem de to tråde.

GUI delen i diagrammet er en samlet betegnelse for diverse klasser i vores User Interface. Disse klasser kommunikere alle sammen igennem MsgHandler og ned til Controller klassen.

### Fast Lane

Denne bane håndterer de software-moduler i systemet, hvis afvikling er kritisk for den musikalske oplevelse ved benyttelse af BodyRock3000.

#### SW-moduler i Fast Lane

Benyttelse af systemet som instrument (UC 5-8) tildeles en høj prioritet i systemets scheduler, da disse er latens-kritiske. Følgende softwaremoduler benyttes ved brug af Rocks MIDI- og lydgenerations funktion:

* Receiver
* MidiModule
* ALSA-functions
* (Controller)
* (DataStorage)

Controller og DataStorage indgår i UC 5, ”Vælg preset”, der er en afstikker-funktionalitet i Fast Lane, idet den kan anses som en (re-)initieringsfunktion for banen Fast Lane, hvis hovedfunktion er at omdanne sensor data til hhv. MIDI-signaler og lyd.

#### Receiver

##### Sekvensdiagram

##### Klassediagram

##### Funktionsbeskrivelser

#### Applikationsmodel for MidiModule

I dette afsnit findes en applikationsmode for MidiModule tråden.

##### Klasseidentifikation

MidiModulet opdeles i underklasser for at vise systemets interne virkemåde.

**Boundary klasser:**

* ***MsgQueue***: Inderholder messages fra hhv. Receiver (data) og Controller (preset)
* ***AlsaAdapter***: Interface til ALSA-funktionerne

**Domæne klasser:**

* ***Timer***: Systemet interne clock, der holder takten for omdannelse af data til MidiSignaler.
* ***SensorConfig***: Et objekt af klassen SensorConfiguration i det aktive preset, der indeholder konfiguration for hvilket datapunkt et specifikt MidiSignal skal generers ud fra.
* ***MappingScheme:*** Et objekt af klassen MappingScheme i et specifikt SensorConfig objekt. Denne indeholder konfiguration- of funktionaliteten for hvordan et bestemt MidiSignal genereres.

**Controller klasse:**

* ***MidiModule:*** Det overordnede objekt, de indeholderden samlede funktionalitet for hvordan en mængde af datapunkter omdannes til en række MidiSignaler jf. det nuværende aktive preset.

##### Sekvens diagram

Herunder ses et sekvensdiagram for programflowet internt i MidiModule. Ved hvert taktslag, tjekkes MsgQueue. Hernæst håndteres den modtagne message jf. dens type.

For data messages loopes gennem de aktive sensorkonfigurationer, hvor SensorConfig tilgår et specifikt datapunkt, og MappingScheme dernæst opdaterer det givne SensorConfig objekts MidiSignal.

For preset messages nedlægges først de aktive sensorkonfigurationer, hvorefter et preset svarende til den modtagne message findes og initieres.



##### State Machine Diagram

MidiModule har to hovedstates: ”Idle” og ”Handling”. Handling igangsættes ved hvert taktslag og igangsættes med substate ”Handling Queue”, hvori msgQueue tømmes. Afhængigt af beskedens type, igangsættes så enten ”Handling Data” eller ”Handling Preset”. Når msgQueue er håndteret, går MidiModule i state ”Idle”.



##### Klassediagram

Herunder er angivet et UML klassediagram for MidiModule og dens bestanddele.



### Overordnet Klassediagram Rock

# Design og implementering (HW)

# Test (HW)

## Enhedstests

## Integrationstests

# Design og implementering (SW)

# Test (SW)

## Enhedstests

## Integrationstests

# Accepttest

# Bilag

Bilag forefindes på CD-rom

Oversigt noteres her

1. Teknisk set ikke simultant, men virtuelt simultant, da mennesket vil opfatte det sådan [↑](#footnote-ref-1)
2. http://www.midi.org/techspecs/midispec.php [↑](#footnote-ref-2)
3. Konverteringsmodulet indeholder én til flere konverteringstråde, afhængig af antal sensorer i systemet [↑](#footnote-ref-3)
4. Fejl defineres som XXXXX [↑](#footnote-ref-4)
5. For beskrivelse af I2C, se <http://www.nxp.com/documents/user_manual/UM10204.pdf> [↑](#footnote-ref-5)
6. Indsæt reference [↑](#footnote-ref-6)
7. For en detaljeret beskrivelse af I2C se <http://www.nxp.com/documents/user_manual/UM10204.pdf> [↑](#footnote-ref-7)
8. For en detaljeret beskrivelse af funktionen se bluetooth\_cr\_UG-v1.0r (RN 42).pdf s. 11 [↑](#footnote-ref-8)
9. http://en.wikipedia.org/wiki/Line\_level [↑](#footnote-ref-9)
10. http://www.midi.org/techspecs/midimessages.php [↑](#footnote-ref-10)
11. http://www.alsa-project.org/main/index.php/Main\_Page [↑](#footnote-ref-11)
12. http://www.esi-audio.com/products/midi\_mate/ [↑](#footnote-ref-12)
13. Denne iteration samler kun data fra en sensor. [↑](#footnote-ref-13)
14. REFERENCE [↑](#footnote-ref-14)