### MidiModule

#### Design

MidiModule har til opgave at styre konverteringen af data, der modtages fra receiver, samt skift af preset. For et givent preset oprettes et antal SensorConfiguration objekter ved preset initiation, og nedlægges ved preset afslutning. Ved modtagelse af data fra Receiver, gennemløbes SensorConfiguration objekternes map funktion, hvorved MidiModules MidiSignal vector opdateres med de relevante værdier. Denne MidiSignal vector afsendes dernæst til AlsaAdapater, der står for interfacing til systemets ALSA-funktioner.

##### Klassediagram

Herunder ses et klassediagram for MidiModule og dets nærmeste klasser. Udeladt er klassen DataMsg[[1]](#footnote-1), lagringsstructs i MappingScheme[[2]](#footnote-2), samt AlsaAdapterens association til frameworket ALSA. Controller og Receiver har adgang til MidiModule gennem MsgQueue.



##### Funktionsbeskrivelser

|  |  |
| --- | --- |
| **Funktion** | start(): void |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** | join(): void |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** | getMsgQueue(): MsgQueue\* |
| **Parametre** | Ingen |
| **Returværdi** | MsgQueue \* |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** | eventDispatcher(): void |
| **Parametre** | Ingen |
| **Returværdi** | Ingen |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** | setPreset(list<SensorConfiguration>& sensConfList): void |
| **Parametre** | list<SensorConfiguration>& sensConfList: En reference til de SensorConfigurationer, der ligger i det preset, man ønsker at aktivere |
| **Returværdi** | Ingen |
| **Beskrivelse** | Denne funktion opdaterer MidiModule’s atributter list<SensorConfiguration>& og vector<MidiModule> |

|  |  |
| --- | --- |
| **Funktion** | handleDataMsg(DataMsg\* msg): void |
| **Parametre** | DataMsg\* msg |
| **Returværdi** | Ingen |
| **Beskrivelse** | Kaldes automatisk når en DataMsg påfyldes MsgQueue. Den har til opgave at opdatere MidiModules MidiSignal vector på baggrund af det modtagne data. |

|  |  |
| --- | --- |
| **Funktion** | handleShutDownMsg(DataMsg \* msg): void |
| **Parametre** | DataMsg \* msg: |
| **Returværdi** | Ingen |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** | BR3K\_error(int errorNum, std::string msg): void |
| **Parametre** | int errorNum: std::string msg: |
| **Returværdi** | Ingen |
| **Beskrivelse** |  |

|  |  |
| --- | --- |
| **Funktion** |  |
| **Parametre** |  |
| **Returværdi** |  |
| **Beskrivelse** |  |

#### Implementering

##### start

##### join

##### getMsgQueue

##### setPreset

Implementeringen for setPreset er ligetil takket være STL containerne list og vector. Disse muliggør opdatering af MidiModule attributterne som følge:

sensorConfList\_ = sensConfList;

midiSignalVector\_ = std::vector<MidiSignal>(sensConfList.size());

Bemærk at der oprettes et MidiSignal for hver SensorConfiguration i sensConfList.

##### handleDataMsg

MidiSignal vector opdateres ved at itterere gennem MidiSignals sensConfList og kalde dennes map-funktion med det modtagne data. Dernæst afsendes det den opdaterede midiSignalVector til AlsaAdapter.

Herunder ses et sekvensdiagram for funktionen



##### handleShutDownMsg

##### BR3K\_error

#### Test

Der henvises til Fast Lane integrationstest.

1. Se Receiver s. XXX [↑](#footnote-ref-1)
2. Se MappingScheme s. XXX [↑](#footnote-ref-2)