# Body Control

Body Control er styresystemet på Body-delen af BodyRock3000. Systemet står for at samle sensordata fra de fire[[1]](#footnote-1) forskellige sensorer via I2C, konverterer disse data og videresende dem vha. en UART forbundet til et Bluetooth modul.



Figur 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.

## SerialUnit

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

## 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

|  |  |
| --- | --- |
| **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.[[2]](#footnote-2)  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 |

1. Denne iteration samler kun data fra en sensor. [↑](#footnote-ref-1)
2. REFERENCE [↑](#footnote-ref-2)