## Implementering af Body Control

### Main

For beskrivelse af mains funktion se system arkitektur (REFERENCE)

|  |
| --- |
| Main  Tænd power-indikator  Initier I2C, sensorer og UART  Loop  Hvis nyt preset\*  Send preset  Læs alle sensorer via I2C  Konverter sensordata til rette format og placer i dataarray  Send dataarray via UART  Vent så forsendelsesraten er på 50Hz |

\*Da preset ikke er med i denne iteration er den heller ikke implementeret i den endelige udgave af mainkoden for projektet.

#### Main kode

int main()

{

//Initialisering

initUART();

initSensors();

// Tænd Power indikator

POWERPIN\_Write(1);

while(1)

{

// Klargøre I2C før hver aflæsning sensorer

setupI2C();

// Læs sensorer

readAllSensors();

// Standartiser sensordata

convSensData();

// Send sensordata

sendDataArray();

// Tilpas afsending til 50 Hz

CyDelay(19); // 19 ms

}

return 0;

}

Det skal nævnes at tilpasning til 50Hz i første udgave af Main koden var lavet med et timer interrupt der satte et flag højt hvert 20 ms (= 50Hz). Men at dette interrupt lavede fejl i I2C-aflæsningen af sensorerne.

#### initSensors

#### Kode:

Kristian

#### initADXK345

#### Kode:

Kristian

#### initMPU6050

#### Kode:

Kristian

### SerialUnit

Beskrivelser af SerialUnits funktioner kan ses i system arkitektur (REFERENCE)

|  |
| --- |
| Opret klasse-lokalt dataarray[16] // Med plads til fire sensorer\* |

\*I projektet var der oprindeligt afgrænset til fire sensorer i den endelige udgave er der afgrænset til én (REFERENCE).

#### Kode:

// Klasse-lokalt array

char dataArray[MAXSENSDATA]; // Array with id-number and data to send via BT

#### initUART

|  |
| --- |
| initUART  Start og initer UART komponenten |

#### Kode:

void initUART()

{

// Enable og initier UART komponenten

UART\_1\_Start();

}

#### setDataArray

|  |
| --- |
| setDataArray(ID, X-sensordata, Y-sensordata, Z-sensordata)  Sæt klasse-lokale dataarray til ID, X-sensordata, Y-sensordata,  Z-sensordata  Sæt ubrugte pladser til kendt værdi = 1 |

#### Kode:

void setdataArray(int ID, int x\_data, int y\_data, int z\_data)

{

// Klasse-lokale dataarray til ID, x\_data, y\_data, z\_data

dataArray[0] = ID+1;

dataArray[1] = x\_data+1;

dataArray[2] = y\_data+1;

dataArray[3] = z\_data+1;

//Sætter de resterende pladser i array'et til 1.

int i;

for(i = 4; i<MAXSENSDATA; i++)

{

dataArray[i] = 1;

}

}

#### sendDataArray

|  |
| --- |
| sendDataArray  Opret et array med seks pladser pr sensor jf. Trådløs kommunikations  protokol(REFERENCE)  Sæt hvert sensorarray med de tilsvarende pladser fra det  klasse-lokale dataarray\* //Alle data der sendes bliver +1 pga.  //datavalidering i receiver klassen på Rock  Valider sensordata // Hvis der er fejl tænd rød Error diode  Send sensordata  Reset dataarray til kendt værdi // ASCII ’0’ = 48 |

\*Da der er afgrænset til en sensor i denne udgave af projektet vil alle andre arrays end accelerometer arrayet blive sat til 1.

#### Kode:

void sendDataArray()

{

// Opret et array med seks pladser pr sensor

char accelArray[6];

char gyroArray[6];

char proxArray[6];

char flexArray[6];

// Sætter accelerometer array.

accelArray[0] = START\_SENSDATA;

accelArray[1] = dataArray[0]; // dataArray[0] - ID 0

accelArray[2] = dataArray[1]; // dataArray[1] - x

accelArray[3] = dataArray[2]; // dataArray[2] - y

accelArray[4] = dataArray[3]; // dataArray[3] - z

accelArray[5] = 0; // Nul terminering

// Sætter gyroskop array

gyroArray[0] = START\_SENSDATA;

gyroArray[1] = 1+1; // dataArray[4] - ID 1

gyroArray[2] = 1+1; // dataArray[5] - x

gyroArray[3] = 1+1; // dataArray[6] - y

gyroArray[4] = 1+1; // dataArray[7] - z

gyroArray[5] = 0; // Nul terminering

// Sætter Proximity array

proxArray[0] = START\_SENSDATA;

proxArray[1] = 2+1; // dataArray[8] - ID 2

proxArray[2] = 1+1; // dataArray[9] - x

proxArray[3] = 1+1; // dataArray[10] - y

proxArray[4] = 1+1; // dataArray[11] - z

proxArray[5] = 0; // Nul terminering

// Sætter flex array

flexArray[0] = START\_SENSDATA;

flexArray[1] = 3+1; // dataArray[12] - ID 3

flexArray[2] = 1+1; // dataArray[13] - x

flexArray[3] = 1+1; // dataArray[14] - y

flexArray[4] = 1+1; // dataArray[15] - z

flexArray[5] = 0; // Nul terminering

//Error Check om I2C sensordata er valid

if((dataArray[1] == 65)&&(dataArray[2] == 65)&&(dataArray[3] == 65))

{

ERRORPIN\_Write(1);

}

//Send sensordata

UART\_1\_UartPutString(accelArray);

UART\_1\_UartPutString(gyroArray);

UART\_1\_UartPutString(proxArray);

UART\_1\_UartPutString(flexArray);

//Reset dataArray til kendt acii værdi: 48 = '0';

int i;

for(i = 0; i < MAXSENSDATA; i++)

{

dataArray[i] = 48;

}

}

### Sensor

Beskrivelser af Sensors funktioner kan ses i system arkitektur (REFERENCE)

#### setupI2C

#### Kode:

#### setSensArray

#### Kode:

#### readAllSensors

#### Kode:

#### convSensdata\*

|  |
| --- |
| convSensdata  Opret variabler  Saml most significant og least significant for hver sensorkoordinat  X, Y, Z i variabler  Konverter fra værdier fra [-512..511] til værdier [0-127] og læg i  8bit variabel  Kald setdataArray fra SerialUnit og sæt accelerometer ID samt  konverteret X, Y og Z værdier |

\*Denne funktion er kun lavet til at standardiser accelerometerets(ADXL345) sensordata da de andre er afgrænset væk.

#### Kode:

void convSensData(void)

{

//ADXL345

// Opret variabler

int16 x, y, z; // Da det er to 8 bit registre som skal lægges sammen

unsigned char x1, y1, z1;

//Samler most significant og least significant for X

x = sensArray[1][2]<< 8;

x+= sensArray[0][2];

//Samler most significant og least significant for Y

y = sensArray[3][2]<< 8;

y+= sensArray[2][2];

//Samler most significant og least significant for Z

z = sensArray[5][2]<< 8;

z+= sensArray[4][2];

//Konvertering fra int16 til uint8

x1 = (x+512)/8;

y1 = (y+512)/8;

z1 = (z+512)/8;

// Sæt dataarray i SerialUnit

setdataArray(ACC1\_ID, x1, y1, z1);

}

#### readI2C

#### Kode:

#### handleI2CError

#### Kode: