#### initSensors

|  |
| --- |
| Skal kalde init-funktionerne for de ønskede sensorer |

#### 

#### Kode:

void initSensors()

{

/\*\*\*\*\* INIT INDIVIDUAL SENSORS \*\*\*\*\*/

initADXL345(); //Accellerometer

//initMPU6050(); //Gyroscope

// ..osv

}

#### initADXK345

|  |
| --- |
| Skal kalde funktionen setSensArray indtil alle seks registre fra accelerometeret er læst ind i funktionen.  Skal opsætte accelerometeret op til I2C |

#### Kode:

//Accelerometer ADXL345 defines

#define ACCEL\_ADDRESS 0x53 // jumperen sættes til GND

#define PWR\_CTRL\_REG 0x2D //Power control registeret Datablad side 26

#define PWR\_MODE (1<<3) // Sættes i measurement mode, standby til den skal måle (Strømbesparende)

#define DATAX0 0x32 //least significant

#define DATAX1 0x33 //most significant

#define DATAY0 0x34

#define DATAY1 0x35

#define DATAZ0 0x36

#define DATAZ1 0x37

//init accelerometer

void initADXL345(void)

{

// Fylder ADXL345 ind i sensor array'et.

setSensArray(ACCEL\_ADDRESS, DATAX0, DEFAULT\_DATA);

setSensArray(ACCEL\_ADDRESS, DATAX1, DEFAULT\_DATA);

setSensArray(ACCEL\_ADDRESS, DATAY0, DEFAULT\_DATA);

setSensArray(ACCEL\_ADDRESS, DATAY1, DEFAULT\_DATA);

setSensArray(ACCEL\_ADDRESS, DATAZ0, DEFAULT\_DATA);

setSensArray(ACCEL\_ADDRESS, DATAZ1, DEFAULT\_DATA);

//Opsætter til I2C

if(I2C\_1\_I2CMasterSendStart(ACCEL\_ADDRESS, I2C\_1\_I2C\_WRITE\_XFER\_MODE) == I2C\_1\_I2C\_MSTR\_NO\_ERROR //Tjekker om transfer er fuldført uden fejl

&& I2C\_1\_I2CMasterWriteByte(PWR\_CTRL\_REG) == I2C\_1\_I2C\_MSTR\_NO\_ERROR // POWER CONTROL REGISTER

&& I2C\_1\_I2CMasterWriteByte(PWR\_MODE) == I2C\_1\_I2C\_MSTR\_NO\_ERROR) // Sættes i measurement mode. Standby til den skal måle (Strømbesparende)

handleI2CError();

#### }

#### initMPU6050

|  |
| --- |
| Skal kalde funktionen setSensArray indtil alle seks registre fra gyroskopet er læst ind i funktionen.  Skal opsætte gyroskopet op til I2C |

#### Kode:

//Gyroscope MPU6050 defines

#define GYRO\_ADDRESS 0x68 //Jumperen sættes til GND

#define GYRO\_XOUT\_H 0x43 // Most significant

#define GYRO\_XOUT\_L 0x44 // Least significant

#define GYRO\_YOUT\_H 0x45

#define GYRO\_YOUT\_L 0x46

#define GYRO\_ZOUT\_H 0x47

#define GYRO\_ZOUT\_L 0x48

//init gyroscope

void initMPU6050(void)

{

// Fylder ADXL345 ind i sensor array'et.

setSensArray(GYRO\_ADDRESS, GYRO\_XOUT\_H, DEFAULT\_DATA);

setSensArray(GYRO\_ADDRESS, GYRO\_XOUT\_L, DEFAULT\_DATA);

setSensArray(GYRO\_ADDRESS, GYRO\_YOUT\_H, DEFAULT\_DATA);

setSensArray(GYRO\_ADDRESS, GYRO\_YOUT\_L, DEFAULT\_DATA);

setSensArray(GYRO\_ADDRESS, GYRO\_ZOUT\_H, DEFAULT\_DATA);

setSensArray(GYRO\_ADDRESS, GYRO\_ZOUT\_L, DEFAULT\_DATA);

//Opsætter til I2C

if(I2C\_1\_I2CMasterSendStart(GYRO\_ADDRESS, I2C\_1\_I2C\_WRITE\_XFER\_MODE) == I2C\_1\_I2C\_MSTR\_NO\_ERROR) //Tjekker om transfer er fuldført uden fejl

handleI2CError();

}

#### setupI2C

|  |
| --- |
| Skal kalde de to initieringsfunktioner fra I2C komponenten. |

#### Kode:

void setupI2C()

{

I2C\_1\_Start(); // Funktion fra I2C komponent datablad

I2C\_1\_I2CMasterClearStatus(); // Funktion fra I2C komponent datablad

}

#### setSensArray

|  |
| --- |
| setSensArray(deviceAddress, registerToRead, data)  Sæt plads [i][0] med sensorens device adresse  Sæt plads [i][1] med det sensor register som skal læses fra  Sæt plads [i][2] med data fra register som skal læses fra |

#### Kode:

void setSensArray(int deviceAddress, int registerToRead, int dataTarget)

{

sensArray[numOfRegToRead][0]=deviceAddress; //set device address

sensArray[numOfRegToRead][1]=registerToRead;//set register for future read

sensArray[numOfRegToRead][2]=dataTarget; //set where to save read data

++numOfRegToRead; //increment current number of places in array that are set

}

#### readAllSensors

|  |
| --- |
| Skal kalde funktionen readI2C i det antal af regsitre det ønskes læst fra |

#### Kode:

void readAllSensors()

{

int numOfReg; // Variabel til at inkrementere

for (numOfReg = 0; numOfReg < numOfRegToRead; numOfReg++)

{

readI2C(numOfReg); // Kalder readI2C for hver inkrementering

}

}

#### readI2C

|  |
| --- |
| Skal generisk kunne læse fra en sensor vha. |

#### Kode:

void readI2C(int numOfReg)

{

uint8 rawData;

if (I2C\_1\_I2CMasterSendStart(sensArray[numOfReg][0],

I2C\_1\_I2C\_WRITE\_XFER\_MODE) == I2C\_1\_I2C\_MSTR\_NO\_ERROR)

/\* Check if transfer completed without errors \*/

{

if(I2C\_1\_I2CMasterWriteByte(sensArray[numOfReg][1]) ==

I2C\_1\_I2C\_MSTR\_NO\_ERROR)

{

if(I2C\_1\_I2CMasterSendRestart(sensArray[numOfReg][0],

I2C\_1\_I2C\_READ\_XFER\_MODE) == I2C\_1\_I2C\_MSTR\_NO\_ERROR)

{

rawData = I2C\_1\_I2CMasterReadByte(I2C\_1\_I2C\_NAK\_DATA);

I2C\_1\_I2CMasterSendStop();

sensArray[numOfReg][2] = rawData;

// gemmer rawData på den tilhørende plads i sensArray.

}

}

}

}

#### handleI2CError

|  |
| --- |
| Skal håndtere hvis der kommer fejl på I2C ved at sende en stop |

#### Kode:

void handleI2CError(void)

{

I2C\_1\_I2CMasterSendStop();

}