|  |
| --- |
| Stegräknare  **Projekt: Pedometer med AVR**  Kurs Programmering av inbyggda system  2017-04-18 |
|  |
| |  |  |  | | --- | --- | --- | | **Dennis Bunne** [LinkedIn](https://www.linkedin.com/in/dennis-bunne-259a9783/) | **Johan Kämpe** [LinkedIn](https://www.linkedin.com/in/johankampe/) | **Simon Karlsson** [LinkedIn](https://www.linkedin.com/in/simon-karlsson-54509aa9/) |   Mjukvaruutvecklare Inbyggda System [MÖLK Utbildning AB](http://dev.molk.com/) |

# Sammanfattning

Skriv när klar

Innehållsförteckning

[**1** **Inledning** 5](#_Toc480312030)

[1.1 Syfte 5](#_Toc480312031)

[1.2 Noteringar och information 5](#_Toc480312032)

[1.3 Bakgrund 6](#_Toc480312033)

[1.3.1 Definitioner av begrepp och förkortningar 6](#_Toc480312034)

[1.3.2 Stegräknare 7](#_Toc480312035)

[1.3.3 I²C 7](#_Toc480312036)

[1.3.4 Pull up-resistorer 8](#_Toc480312037)

[1.3.5 X-Y-Z-riktning 8](#_Toc480312038)

[1.3.6 MPU-9250 8](#_Toc480312039)

[1.4 Länkar 9](#_Toc480312040)

[1.4.1 GitHub 9](#_Toc480312041)

[1.4.2 Använda funktionsbibliotek 9](#_Toc480312042)

[1.4.3 Datablad 9](#_Toc480312043)

[1.4.4 Andra länkar 9](#_Toc480312044)

[**2 Genomförande och resultat** 10](#_Toc480312045)

[2.1 Använd programvara 10](#_Toc480312046)

[2.2 Använd hårdvara 10](#_Toc480312047)

[2.3 Avgränsningar 10](#_Toc480312048)

[2.4 Metod 11](#_Toc480312049)

[2.4.1 Skrivning av källkod, kompilering och uppladdning till mikrokontroller 11](#_Toc480312050)

[2.4.2 Studie av datablad 11](#_Toc480312051)

[2.4.3 Studie av funktionsbibliotek 11](#_Toc480312052)

[2.5 Prototyp 1 12](#_Toc480312053)

[2.6 Prototyp 2 13](#_Toc480312054)

[2.7 Stegräknarens funktion 13](#_Toc480312055)

[2.8 Projektets kod 14](#_Toc480312056)

[2.8.1 Egeninkluderade standardbibliotek 14](#_Toc480312057)

[2.8.2 Inkluderade icke-standardbibliotek 14](#_Toc480312058)

[2.8.3 Macron definierade i main.c 15](#_Toc480312059)

[2.8.4 Macron från andra bibliotek 16](#_Toc480312060)

[2.8.5 Variabler 17](#_Toc480312061)

[2.8.6 Funktioner 18](#_Toc480312062)

[2.8.7 Main-funktionen 19](#_Toc480312063)

[**3 Bilagor** 20](#_Toc480312064)

[3.1 Kopplingsschema för prototyp 1 20](#_Toc480312065)

[3.2 Kopplingsschema för prototyp 2 21](#_Toc480312066)

Figurförteckning

[Figur 1 I2C-ansluting på Atmega328 7](#_Toc480312893)

[Figur 2 Riktningar i X- Y och Z-led 8](#_Toc480312894)

[Figur 3 MPU-9250, framsida 8](#_Toc480312895)

[Figur 4 MPU-9250, baksida med anslutningsbeteckningar 8](#_Toc480312896)

[Figur 5 Uppkoppling av MPU-9250 till Arduino UNO 12](#_Toc480312897)

[Figur 6 Utskrift av stegräkning till Serial Monitor 12](#_Toc480312898)

[Figur 7 Kodprojektets egenskaper – Defined symbols 15](#_Toc480312899)

# Inledning

## Syfte

Syftet med projektet är att tillverka en bärbar, batteridriven stegräknare. Stegräknaren ska ha modulära enheter, det vill säga att dessa går att ta bort eller byta ut. En modulär enhet kan vara en display, blåtandsmodul eller accelerometer.

En display används för att visa antal tagna steg. Stegen ska avläsas med hjälp av en accelerometer, eller alternativt med en vibrationssensor.

## Noteringar och information

Projektet utfördes på under perioden mars till april, 2017.

I rapporten hänvisas själva stegräknaren ofta till som *projektet* elle*r produkten*.

I rapporten hänvisas MPU-9250-enheten, som används för att mäta acceleration, ofta som *MPU-enheten*, eller *MPU-n*.

## Bakgrund

### Definitioner av begrepp och förkortningar

|  |  |
| --- | --- |
| **Term** | **Förklaring** |
| **Acceleration** | Förändring av hastighet per tidsenhet. Kan vara positiv (hastighetsökning) eller negativ (minskad hastighet). |
| **Arduino UNO** | Utvecklingskort. |
| **Atmega328p** | En mikrokontroller (MCU) med 28 anslutningar, från tillverkaren Atmel |
| **Bibliotek** | En samling av kod-funktioner och/eller definitioner. |
| **Breadboard** | Kopplingsplatta, används för att ansluta enheter till varandra. |
| **C** | Det programmeringsspråk som används i projektet. |
| **DIL** | Dual in-line. Standard för mikrochip med två rader av ben. |
| **Git** | Ett versionshanteringssystem |
| **GitHub** | Molnlagring för Git-projekt |
| **I²C** | Ett sätt att koppla samman enheter, via två ledare: SCL och SDA. |
| **IDE** | *Integrated Development Environment*. Ett datorprogram eller en programsvit som vanligtvis innehåller en texteditor, kompilator och debugger. För att underlätta vid programmering. |
| **Lib** | Se *Bibliotek.* |
| **Library** | Se *Bibliotek.* |
| **MCU** | Microcontroller Unit, se*Mikrokontroller.* |
| **Mikrokontroller** | En liten enchippsdator med CPU, arbetsminne och programminne. |
| **Mikroprocessor** | SKRIV |
| **MPU** | Microprocessor Unit, Se *mikroprocessor* |
| **MPU-6050** | Mikroprocessor med sensor för gyro och acceleration i tre riktningar vardera. |
| **MPU-9250** | Mikroprocessor med sensor för gyro, acceleration och kompass i tre riktningar vardera. |
| **Pull up-reistor** | Används i logiska kretsar för att se till att en ingång håller sig hög när den inte har någon anslutning. |
| **Reistans** | Värdet på ett motstånd / resistor. Anges i Ohm (Ω). |
| **Resistor** | Motstånd i elektriska kretsar. |
| **SCL** | Se *I²C.* |
| **SDA** | Se *I²C.* |
| **Standardbibliotek** | Kodbibliotek som alltid är tillgängliga för ett specifikt programspråk. Exempelvis *stdio.h* för programspråket C. |
| **Stegräknare** | En enhet som mäter hur många steg en människa tar |
| **TWI** | Se *I²C.* |
| **Utvecklingsmiljö** | Se *IDE.* |
| **Serial Monitor** | Funktion i Arduinos IDE, för att kommunicera med inkopplad Arduino-enhet. Används ofta för debugging. |
| **AVR** | Familj av Mikrokontrollers som tillverkas av företaget Atmel. |
| **ISP** | *In-system programming*, programmering av redan installerade eller inbyggda system. |

SORTERA+FYLLPÅ

### Stegräknare

En stegräknare är en vanligtvis portabel enhet. Som mäter hur många steg en människa tar. Den kan användas vid exempelvis löpning, promenad, eller för att se hur många steg en person tar under en dag.

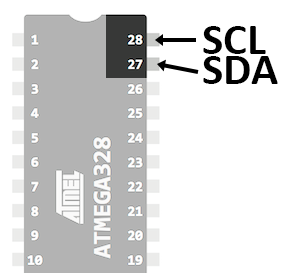
Stegräknare kan också kallas pedometer.

### I²C

I²C, *Inter-Integrated Circuit*, är ett sätt att koppla enheter till moderkort, inbyggda system, mobiltelefoner eller andra enheter.

Kommunikation sker via två ledare, SDA (*Serial Data Line*, datasignal) och SCL (*Serial Clock*, klocksignal). Flera I²C-enheter kan anslutas till samma SDA- och SCL-ledare.

På en mikrokontroller av typen Atmega328p, är anslutningar 27 och 28, SDA och SCL, respektive.



Figur 1 I2C-ansluting på Atmega328

I²C kallas även TWI, *Two Wire Interface*, eller *IIC*.

### Pull up-resistorer

Pull up-resistorer används i logiska kretsar för att se till att en ingång håller sig hög när den inte har någon anslutning.

Utan pull up kan ingången anta vilket läge som helst, hög eller låg. I projektet används pull up-resistorer för I²C-anslutningarna.

Resistorer som används ska vara av hög resistans.

### MPU-9250

Enheten som används i projektet för att mäta acceleration kallas *MPU-9250 IMU Breakout* och är tillverkad av Sparkfun. På enheten finns en MPU-9250.

|  |  |
| --- | --- |
| SparkFun IMU Breakout - MPU-9250  Figur 2 MPU-9250, framsida | SparkFun IMU Breakout - MPU-9250  Figur 3 MPU-9250, baksida med anslutningsbeteckningar |

De anslutningar som används i projektet är VDD och GND för drivning, samt SDA och SCL för I2C-kommunikation.

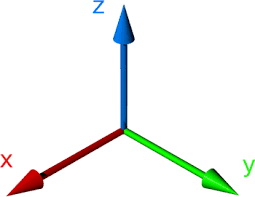
Länk till tillverkarens produktsida finns i kapitel 1.4.4

### 

### X-Y-Z-riktning

X, Y och Z används inom projektet för att beskriva riktningar i en tredimensionell miljö. MPU-9250-enheten har sensorer för acceleration i dessa tre riktningar.

När enheten förs åt ett visst håll mäts rörelsens acceleration i denna riktning.



Figur 4 Riktningar i X- Y och Z-led

## Länkar

### GitHub

Länkar till projektets GitHub-sida

**GitHub-sida för projektet**  
<https://github.com/GoblinDynamiteer/stepCounter>

**GitHub-sida, huvudkod**  
<https://github.com/GoblinDynamiteer/stepCounter/blob/master/Kod/Huvudkod/stepCounter/mpu_conntest/main.c>

**GitHub-sida, kod för prototyp 1**  
<https://github.com/GoblinDynamiteer/stepCounter/blob/master/Kod/Testkod/Arduino/stepCounterPrototype/stepCounterPrototype.ino>

### Använda funktionsbibliotek

Länkar till bibliotek som har använts i projektets kod

**MPU 6050 library**   
[http://davidegironi.blogspot.se/2013/02/avr-atmega-mpu6050-gyroscope-and.html#.WOtTeojyhPY](http://davidegironi.blogspot.se/2013/02/avr-atmega-mpu6050-gyroscope-and.html%23.WOtTeojyhPY)

**U8glib**<https://github.com/olikraus/u8glib>

**I2C master library**  
[http://homepage.hispeed.ch/peterfleury/avr-software.html#libs](http://homepage.hispeed.ch/peterfleury/avr-software.html%23libs)

### Datablad

Lista!!

### Andra länkar

**MPU-6050 Accelerometer + Gyro**Kodexempel<http://playground.arduino.cc/Main/MPU-6050>

**SparkFun IMU Breakout - MPU-9250**Produktbeskrivning<https://www.sparkfun.com/products/13762>

# Genomförande och resultat

## Använd programvara

|  |  |
| --- | --- |
| **Programnamn** | **Beskrivning** |
| Atmel Studio 7.0 | IDE för att skriva kod och kompilera till Atmel MCU. |
| Atom | Texteditor, för att skriva och analysera kod. |
| Git | Versionshanteringsverktyg |
| GitKraken | Grafiskt gränssnitt (GUI) för Git |
| Fritzing | Program för att skapa kretsscheman |
| Microsoft Word | Dokumentering och rapportskrivning |
| Google Docs | rapportskrivning |
| AVRDUDE, AVR Downloader/UploaDEr | Program för att ladda upp kompilerad kod till MCU. |
| AVRDUDESS | Grafiskt gränssnitt (GUI) för AVRDUDE |
| Autodesk Fusion 360 | CAD-program för att konstruera ett hölje till stegräknaren. |

## Använd hårdvara

|  |  |
| --- | --- |
| **Komponent** | **Beskrivning** |
| Atmel ATMega328-PU DIP-28N 8-bit MCU | Mikrokontroller. |
| SparkFun MPU-9250 | Enhet som mäter acceleration i X, Y och Z-riktning. Innehåller även gyroskop, temperatur och kompass-funktionalitet. |
| Micro OLED Display SSD1306 128x64 | Display med upplösning 128x64 pixlar. |
| AVR pocket programmer | Enhet för att programmera Atmega-MCU via en dators USB-port. |
| DIL-hållare 28-pin 0.3" | Sockel för Atmega-MCU |
| Resistor 4,7kΩ, 2 st. | Pull up-resistorer för I²C-kanaler. |
| Kristalloscillator 16 MHz | Kopplas till MCU för att öka klockfrekvensen från 1 MHz till 16 MHz |
| Kondensator 22pF, 2 st. | Används för inkoppling av kristalloscillator. |
| Litiumbatteri 18650 | Strömförsörjning. |
| Batterihållare för litiumbatteri 18650 |  |
| Grovekontakter med 4-stift | För modularisering av I²C-komponenter. |
| Arduino UNO | Prototyping |

## Avgränsningar

SKRIV!!!!!

## Metod

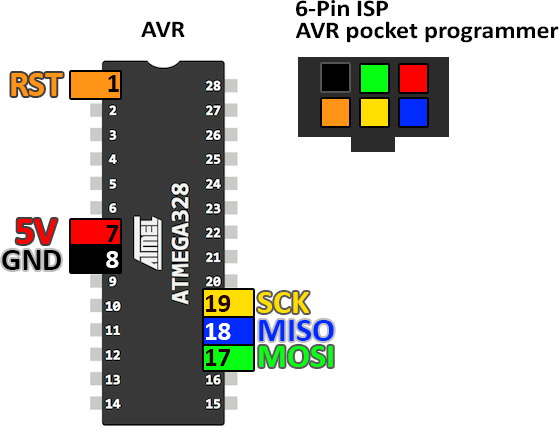
### Skrivning av källkod, kompilering och uppladdning till mikrokontroller

Programmets källkod skrevs i utvecklingsmiljön Atmel Studio, som ett *GCC C Executable Project*-projekt. Källkoden är skriven i programspråket C.

All egenskriven kod finns i projektets källkodsfil *main.c*.

Vid kompilering av koden skapas en fil med filändelsen *hex* i Atmel Studio-projektets *Debug*-underkatalog*.* Denna fil laddas upp till mikrokontrollern med datorprogrammet AVRDUDE och programmeringsenheten *AVR pocket programmer*.

Programmering utförs via ISP, följande anslutningar på mikrokontrollerns används.



Figur 5 Inkoppling av programmeraren till mikrokontrollern

Programmeringsenheten kan även användas som strömkälla.

### Studie av datablad

Tidigt i projektet samlades datablad in för införskaffad hårdvara. Databladen studerades och relevant information noterades för att kunna användas vid framtagning av prototyp.

I synnerlighet studerades datablad för:

* **MPU-9250**
* **Atmega328p**
* **OLED Display**

### Studie av funktionsbibliotek

Tidigt i projektet letades diverse färdiga kodbibliotek upp, för projektets hårdvara. Koden i dessa bibliotek studerades, eller användes för att undersöka funktionen hos olika komponenter.

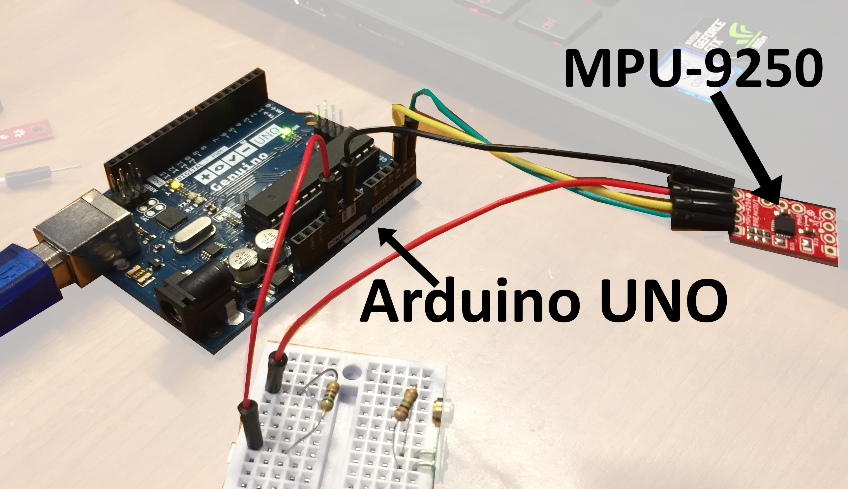
Exempel på bibliotek som studerades eller användes i projektets utveckling:

|  |  |
| --- | --- |
| **Bibliotek** | **Beskrivning och syfte** |
| **Wire.h** | I2C-bibliotek för Arduino-enheter. För att funktionstesta I2C-enheter med Arduino UNO. Användes även i koden för prototyp 1. |
| **twitest.c** | I2C-test/demo för AVR-enheter. |
| **mpu6050.h** | MPU-6050 (Accelerometer, mm.) för AVR-enheter. |
| **i2cmaster.h** | I2C-bibliotek för AVR-enheter. |
| **ssd1306.c** | Kod som använt sig av **i2cmaster.h** för att skriva ut text till display. |

Koden studerades med texteditorn Atom och i utvecklingsmiljön Atmel Studio.

## Prototyp 1

Projektets första prototyp, kallad *prototyp 1*, byggdes för att säkerställa att en stegräknare kunde konstrueras med en MPU-9250. MPU-enheten kopplades direkt till en Arduino UNO-enhet via en breadboard.

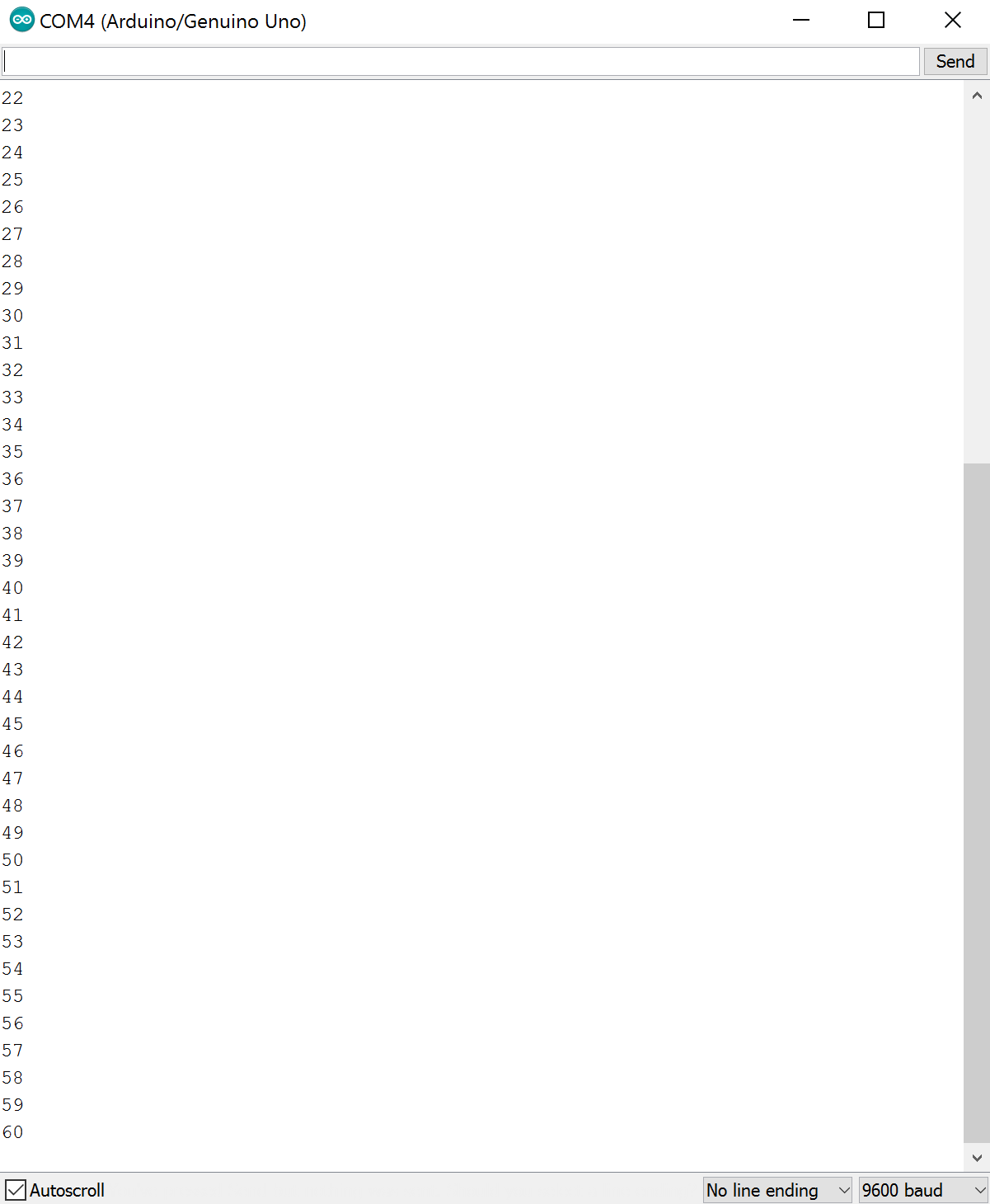


Figur 6 Uppkoppling av MPU-9250 till Arduino UNO

Koden för prototyp 1 baserades på kod skapad av JohnChi [[1]](#footnote-1), som skriver ut värden från MPU-6050-enheter till Arduino-utvecklingsmiljöns *Serial Monitor.* Koden skrevs om så att endast värden för acceleration hämtades, dessa användes för stegräkning.

Funktionsbiblioteket **wire.h** används för att ansluta MPU-n via I2C till Arduino-enheten. Portarna A4 och A5 används för SDA och SCL, respektive. En resistor med resistansvärdet 560 Ohm används för att ta ner spänningsmatningen till MPU-n från 5V till ca 3V. Enligt tillverkarens hemsida[[2]](#footnote-2) ska enheten drivas med 2.4 – 3.6 V.

En bärbar dator användes för strömförsörjning och för att avläsa *Serial Monitor*. Avläsning av steg gjordes genom att placera prototypen i en byxficka och promenera runt.



Figur 7 Utskrift av stegräkning till Serial Monitor

En vibrationssensor testades också för att avläsa steg. Även om den kunde användas för avläsning, valdes den att inte användas, då MPU-enheten ansågs vara mer pålitlig.

Prototyp 1 färdigställdes 2017-03-21

Kopplingsschema för prototyp 1 kan ses i rapportens bilagor.

## Prototyp 2

LED användes för indikation av steg.

Prototyp 2 används som bas för den färdigställda produkten.

Kopplingsschema för prototyp 2 kan ses i rapportens bilagor.

## Slutgiltig produkt

Text

## Stegräknarens funktion

Vid stegräknarens start kalibreras den genom att hämta 20 stycken värden för acceleration från MPU-9250-enheten. Stegräknaren bör ligga stilla under kalibreringen. Medelvärdet av de 20 värdena sätts till att vara stegräknarens värde för när den är passiv. Detta värde benämns som *idle*-värde.

Meddelandet "Calibrating.." visas på stegräknarens display under kalibreringen. När kalibreringen är färdig visas meddelandet "Idle.*.*" på displayen, och stegräknaren kan börja användas.

Under användning hämtas accelerationsdata kontinuerligt från MPU-9250-enheten. Skillnaden mellan detta värde och *idle*-värdet beräknas som ett absolutvärde. Om skillnaden mellan dessa två värden överskrider ett visst satt tröskelvärde, så anses det att ett steg har tagits.

Vid varje taget steg uppdateras stegräknarens display med det totala antalet tagna steg.

Stegräknarens display sätts automatiskt i sömnläge om inget steg har tagits på ca 30 sekunder. Displayen vaknar automatiskt när ett steg tas.

## Projektets kod

Utförligare kommentering av koden finns i källkodsfilen *main.c*.

### Egeninkluderade standardbibliotek

|  |  |
| --- | --- |
| **Bibliotek** | **Beskrivning** |
| **<avr/io.h>** | Diverse I/O definitioner för mikrokontrollern. |
| **<util/delay.h>** | För funktionen *\_delay\_ms*() som pauserar programmet i angiver antal millisekunder. |
| **<avr/interrupt.h>** | Används för funktionen *sei(),* som aktiverar interrupt på mikrokontrollern. Behövs för kommunikation med MPU-9250. |
| **<math.h>** | Diverse matematikfunktioner och macron. Används för funktionen *fabs()*, som ger absolutvärdet av ett tal. |
| **<stdio.h>** | För funktionen *sprintf(),* som skriver en formaterad textsträng till en char-array. Används för att skriva ut stegräkning till display. |

### Inkluderade icke-standardbibliotek

Flera av dessa bibliotek har egna inkluderingar av olika standardbibliotek.

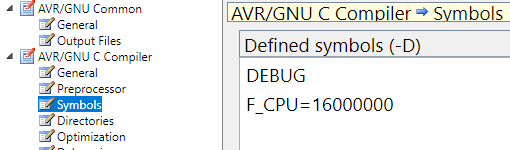
|  |  |
| --- | --- |
| **Bibliotek** | **Beskrivning** |
| **"i2chw/i2cmaster.h"**  *I2C master library using hardware TWI interface* Skapat av Peter Fleury | I2C-funktionalitet för AVR-enheter. För kommunikation mellan mikrokontroller och enheter anslutna till dess I2C-portar SDA/SCL. |
| **"mpu6050/mpu6050.h"**  *MPU6050 lib 0x02* Skapat av Davide Gironi | För accelerometer-enheten, via I2C. Viss egen kod har skrivits genom att bryta ut kod från detta bibliotek. |
| **"u8g/u8g.h"**  *Universal 8bit Graphics Library*  Skapat av olikraus | Funktionsbibliotek till AVR för diverse displayer. |

### Macron definierade i main.c

Macron som är definierade i main.c

|  |  |  |
| --- | --- | --- |
| **Namn** | **Värde** | **Beskrivning** |
| **X** | **0x3B** | Address för accelerometerdata i X-riktning i MPU-9250 |
| **Y** | **0x3D** | Address för accelerometerdata i Y-riktning i MPU-9250 |
| **Z** | **0x3F** | Address för accelerometerdata i Z-riktning i MPU-9250 |
| **STEP\_ACC\_TRIGGER** | **2.3** | Tröskelvärde för acceleration, att bestämma om ett steg har tagits. |
| **SET\_IDLE\_LOOP** | **20** | Antal värden som ska hämtas från accelerometern för att bestämma “idle”-värde.  Det värde som accelerometern ger när den anses ligga stilla. |
| **DISPLAY\_LINE\_HEIGHT** | **16** | Pixelhöjd för en rad på displayen. Används vid utskrift av text. |
| **DISPLAY\_MIDDLE** | **(DISPLAY\_LINE\_HEIGHT \* 2 + 2)** | Ungefärligt pixelvärde för utskrift på displayens mitt. |
| **DISPLAY\_SLEEP\_DELAY** | **3000** | När värdet i variabeln *displaySleepTimer* når detta värdet sätts displayen i sömnläge. |

Macrot F\_CPU behövs vara angivet för standardbiblioteket *delay.h, detta* sätts i kodprojektets egenskaper till *F\_CPU=16000000*, i programmet Atmel Studio.



Figur 8 Kodprojektets egenskaper – Defined symbols

### Macron från andra bibliotek

Macron som används i main.c, vilka är definierade i andra bibliotek.

|  |  |  |  |
| --- | --- | --- | --- |
| **Namn** | **Värde** | **Från bibliotek** | **Beskrivning** |
| **MPU6050\_ADDR** | **(0x68 <<1)** | *mpu6050.h* | I²C-address för MPU-9250 |
| **MPU6050\_AGAIN** | **16384.0** | *mpu6050.h* | Värde för att omvandla ”rå” accelerationsdata från MPU-9250 till g-krafter. |
| **I2C\_WRITE** | **0** | *i2cmaster.h* | Datariktning för I²C - för att skriva till en enhet. |
| **I2C\_READ** | **1** | *i2cmaster.h* | Datariktning för I²C - för att läsa från en enhet |

### Variabler

Variabler i källkoden, sätts i main-funktionen

|  |  |
| --- | --- |
| int columnPin[3] = {  COL1\_PIN,  COL2\_PIN,  COL3\_PIN  }; | Array för anslutningar på MCU som är anslutna till knappsatsens kolumner. Används i programmets huvudloop för att testa knapptryckningar. |
| int rowPin[3] = {  ROW1\_PIN,  ROW2\_PIN,  ROW3\_PIN  }; | Array för anslutningar på MCU som är anslutna till knappsatsens rader.Används i programmets huvudloop för att testa knapptryckningar. |
| int num = 1; | Används i programmets huvudloop för att avgöra vilken knapp som tryckts på, på knappsatsen. |

### Funktioner

Egenskrivna funktioner i källkoden

**void blinkLed(int n);**Blinkar lysdioden som är kopplad till mikrokontrollerns anslutning PC5. Lysdioden blinkar *n* antal gånger, vilket skickas som argument vid anrop av funktionen.

**int checkPin(int pin);**Returnerar 1 eller 0, beroende på om anslutning *pin* på mikrokontrollerns PD-anslutningar är hög eller låg. Argumentet *pin* skickas som argument vid anrop av funktionen.

### Main-funktionen

I detta kapitel beskrivs grundläggande vad som händer i programmets kod

Initialt sätts mikrokontrollerns anslutning PC5 till att användas som utgång:  
DDRC = 0b00100000;

DDR innebär *Data Direction Register* och *C* den samling av åtta anslutningar som tillhör PC0 – PC7 på mikrokontrollern.

Alla åtta anslutningar sätts samtidigt, en etta innebär utgång, och en nolla ingång. 0b001000000 innebär att anslutning 5 sätts till utgång och alla andra som ingångar.

PC5 används för att styra lysdioden som används för indikering av vilken knapp som har tryckts på, på knappsatsen.

Mikrokontrollerns PD-anslutningar sätts till:   
DDRD = 0b11000100;

Detta innebär att PD2, PD6 och PD7 används som utgångar. Portarna är anslutna till knappsatsens anslutningar som motsvarar dess rader.

En oändlig while-loop körs sedan.   
while(1){

I loopens början sätts variabelnint num = 1;  
num används som argument vid anrop av funktionen blinkLed(num);

Sedan körs en for-loop ROWS antal gånger:  
for(int j = 0; j < ROWS; j++){

I loopen sätt mikrokontrollerns PD-utgångar PD2, PD7, PD6 sätts till 1 (hög/på) respektive. PORTD = rowPin[j];

En nästlad for-loop kör *COLS* antal gånger:  
for(int i = 0; i < COLS; i++){

I varje varv anropas funktionen checkPin, för att se om PD-ingången PD3, PD1, PD5 är hög, respektive. Om ingången är hög har en knapp tryckts in och blinkLed anropas med rätt siffervärde.   
if(checkPin(columnPin[i])){

blinkLed(num);

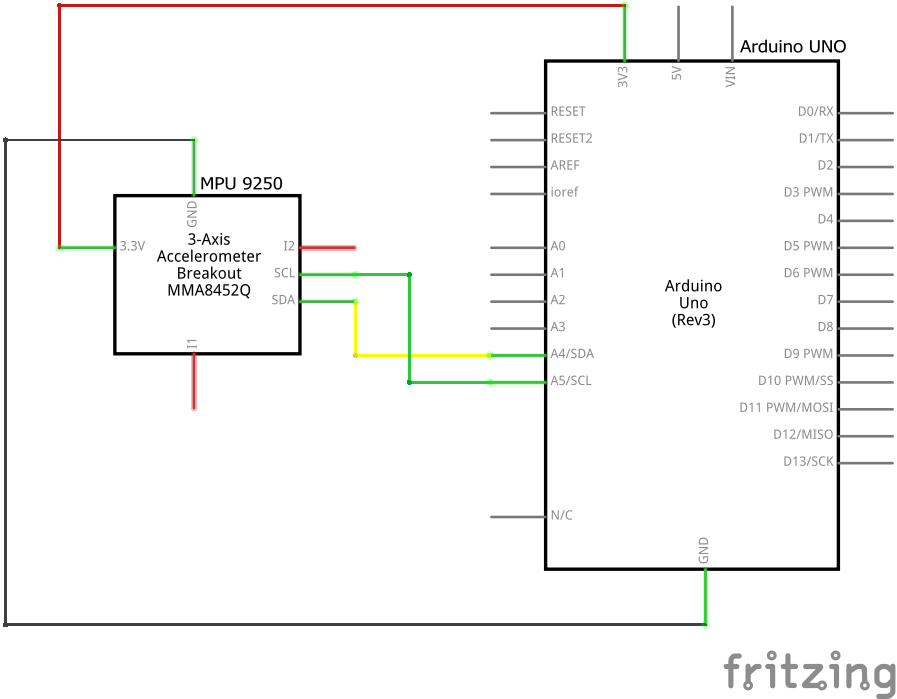
}

I varje varv ökas variabeln num med 1  
num++;

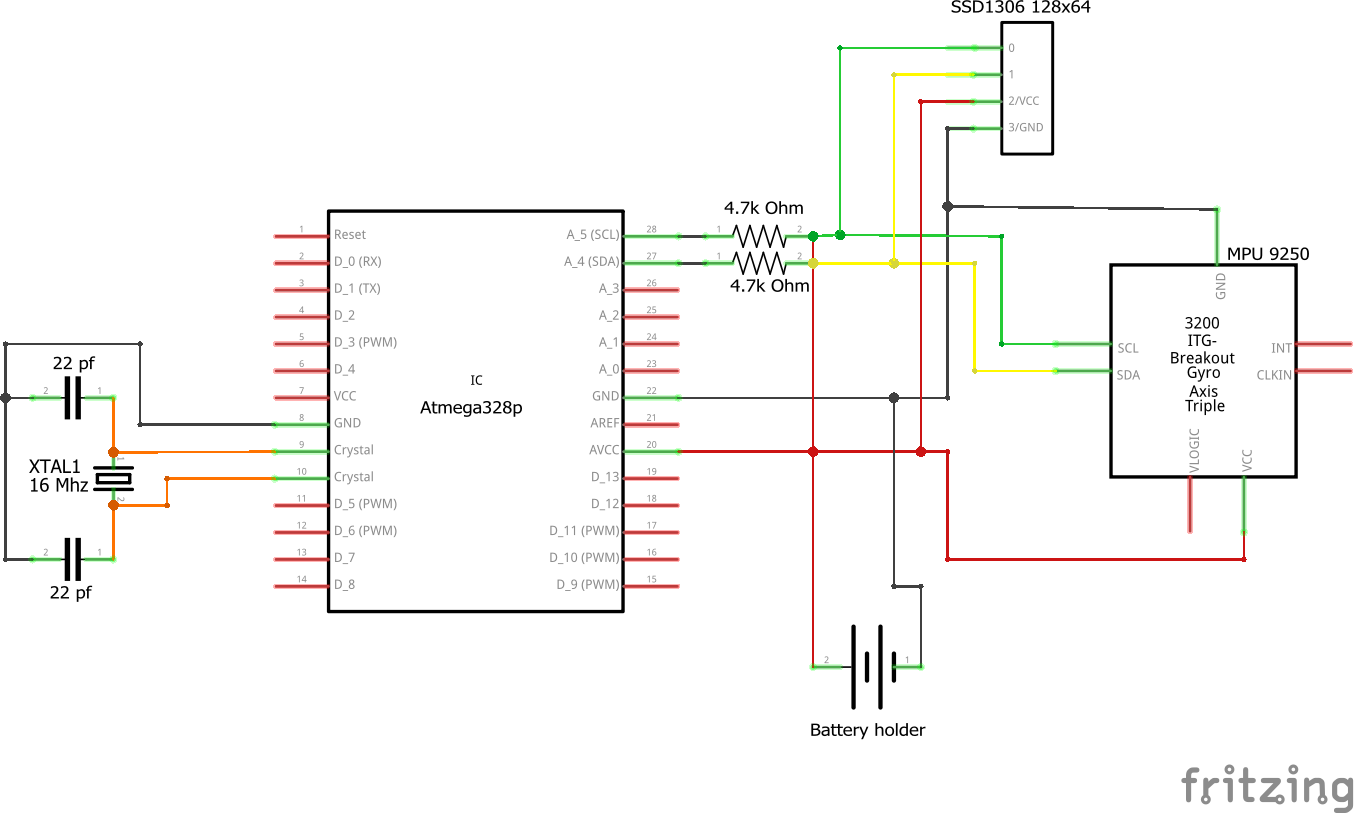
Slut while-loop.

# Bilagor

## Kopplingsschema för prototyp 1



## Kopplingsschema för prototyp 2



1. Se länk i kapitel 1.4.4 [↑](#footnote-ref-1)
2. Se länk i kapitel 1.4.4 [↑](#footnote-ref-2)