Swagway

af

Carl Emil Grøn Christensen and Mathias Dannesbo

8. maj 2012

Resumé

FiXme: Skriv resume

Forord

Gennem teksten vil der være angivet link til "Issues" på GitHub siden for projektet.[1] Disse Issues har været brugt som projektstyring samt bugtracker igennem projektforløbet. Links er angivet som i enden af denne sætning.[#1] Det er ikke nødvendigt for forståelsen af denne rapport at følge linksne.

Vi vil gerne takke Steffen Linnerup Christiansen for hjælp med mekanikken, samt Kristian Holm Nielsen for hjælp med forståelsen af PID.

[1] `https://github.com/neic/Swagway`

Indhold

1 Indledning

1.1 Problemformulering

2 Input

2.1 Sensor

To sensorere, drift.

Sensor hardware

Pull-up, Bus capasistance, level shifter,

Figur 1: IMU breakoutboard fra Sparkfun. (CC BY-NC-SA 3.0, Sparkfun)

Sensor software

I²C, wire.h, libraries

2.2 Styring

Potentiometer, gaffelsensor, strain-gate

3 Control

3.1 Filter

Mål: At samle data fra gyroskop og accelerometer og regne en vinkel ud

Målet med filtret er at samle dataen fra gyroskopet, accelerometeret og ud fra disse beregne en vinkel, som vi kan benytte til at regulere PWM efter. Ud fra dette mål, udvalgte vi tre filtre, som har de egenskaber vi leder efter.

Komplementær filter

Kalman filter

Modificeret Kalman filter

3.2 Regulering

Mål: At omsætte en vinklen til en PWM værdi.

Lineær

PID

Eksponentiel

4 Output

Mål: Køre motorer i begge retninger med variabel hastighed.

4.1 H-broens virkemåde

H-bro teori

4.2 PWM

Pulsbreddemodulation (Pulse-Width-Modulation): Er en metode til at kontrollere spændingen i elektriske apparater. Det er en metode til at levere spænding igennem en række impulser i stedet for der konstant er strøm igennem systemet. Ved at øge eller formindske bredden af impulsen kan man kontrollere en motor.

Man kan altså sige, at PWM er et on/off system hvor man styrer ved hurtigt, at slukke og tænde for spændingen. Tiden der går imellem at den er on/off er så kort, at en LED som sådan ikke mærker det. Så den vil ikke blinke, men derimod lyse mindre hvis man øger bredden imellem impulserne.

Måden vi styre dette på i en Arduino er via `analogWrite(pin,…);`. Her har vi mulighed for at give en værdi fra 0--255. Dette betyder at `analogWrite(pin,255);` er 100% og `analogWrite(pin,127);` er 50%. Dette kan også ses på figur 2.

FiXme: Hele afsnittet skal omskrives

4.3 Overvejelser

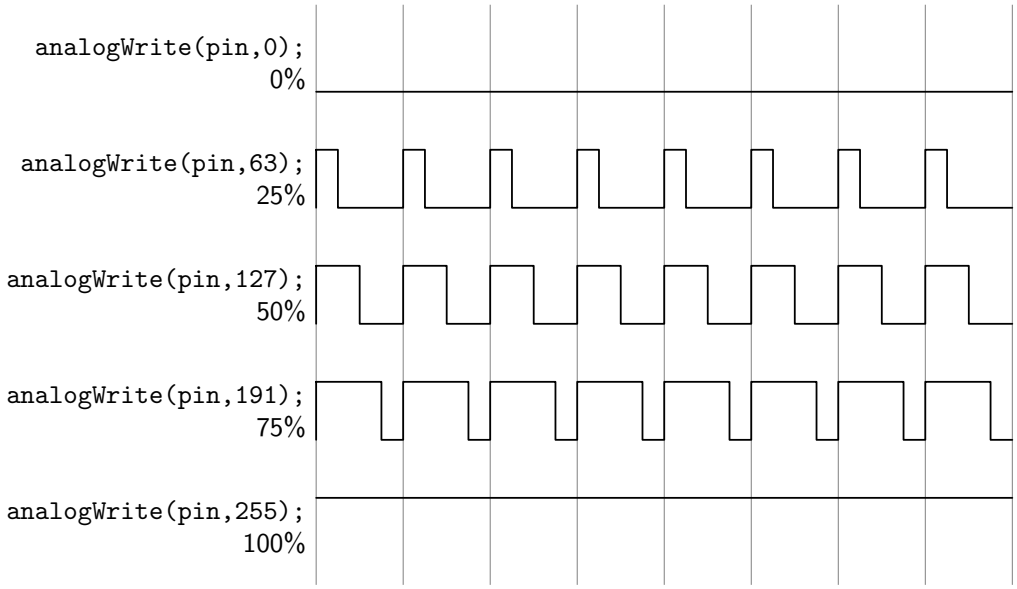
Vores valg: Dobbelt H-bro med mange chip eller bygge selv. Med eller uden PWM "i bunden".

4.4 Motorcontroller

H-bro, PWM, PWM-kondensator, beskyttelses dioder, 4000 serie, optocopler

Samlet board

Det var upraktisk at have alle funktioner på samme board. H-broerne og optocouplerne blev flyttet på sit eget board "Motorcontroller v1.0".



Figur 2: Eksempel på PWM med varierende dutycycle

Tabel 1: Motorcontroller sandhedstabel

Arduino pin			HEXFET spænding				HEXFET on/off				
P7	P6	P5	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	
P8	P9	P10	Q5	Q6	Q7	Q8	Q5	Q6	Q7	Q8	
0	0	0	0	1	0	0	0	0	0	1	Off (↻)
1	0	0	0	0	0	1	0	1	0	0	Off (↺)
0	1	0	1	1	0	0	1	0	0	1	↻
1	1	0	1	0	0	1	1	1	0	0	Short
0	0	1	0	1	1	0	0	0	1	1	Short
1	0	1	0	0	1	1	0	1	1	0	↺
0	1	1	1	1	1	0	1	0	1	1	Short
1	1	1	1	0	1	1	1	1	1	0	Short

Tabellen viser hvordan motorcontrolleren opføre sig hvis den får inputtet angivet under "Arduino Pin"

Motorcontroller v1.0

24. januar 2012 Boardet virkede ikke. Det opførte sig som det var kortsluttet. Det viste sig, at efter boardet var skilt helt af igen, at det plus tegn der skulle vise polariteten var sat ved den forkerte pol. Printet havde taget skade af at blive loddet på flere gange.

FiXme: Indsæt dia
Motorcontroller v1
FiXme: Indsæt figu
Motorcontroller v1

Der var desuden nogle af ledningene for tætsiddene og loddeøerne var lidt underdimensionerede. Der manglede også en mulighed for at se, hvilken vej strømmen løber i H-broerne. Dette blev rettet i v2.0.

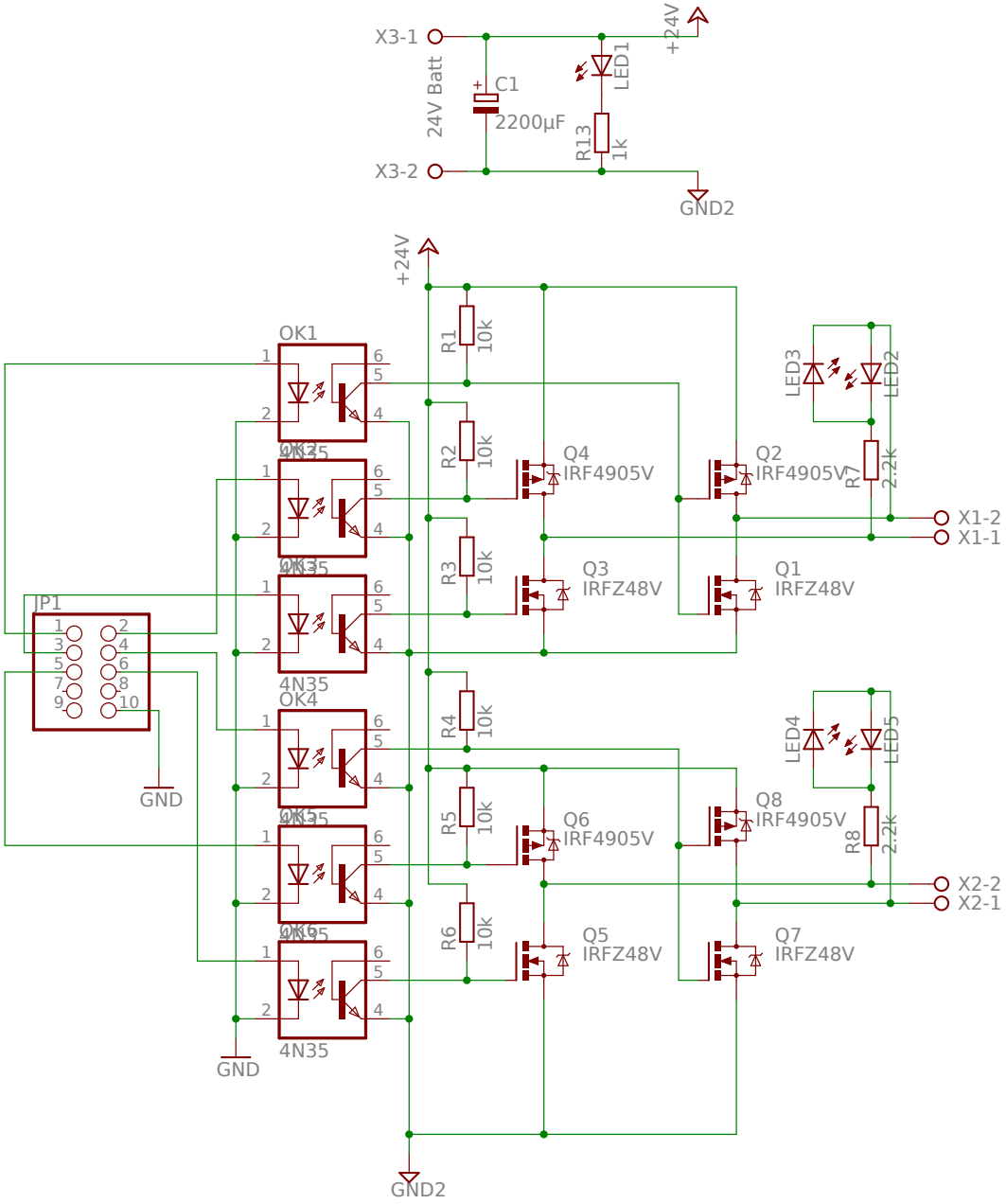
Motorcontroller v2.0

8. marts 2012 Dette board blev aldrig lavet færdig; Ledningerne omkring pinheaderen var for tæt efter at loddeøeren blev forstørret. Diagram og figur over printet kan findes i bilag.

FiXme: ref

Motorcontroller v2.1

8 marts 2012



Figur 3: Diagram over Motorcontroller v2.1

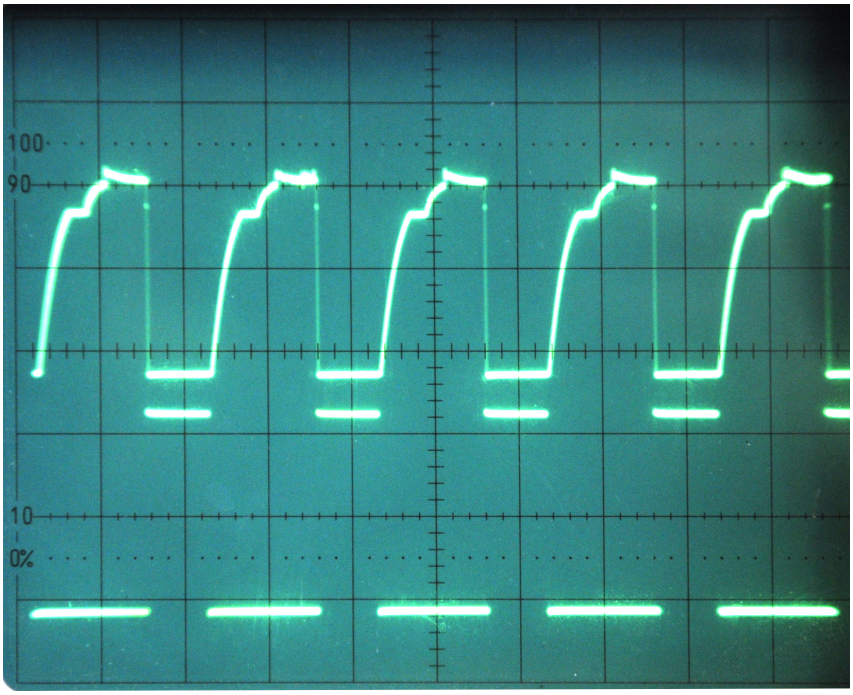
FiXme: ref til fig:mosch2.1

Boardet fungerede umiddelbart. Motoren kunne køre i begge retninger og farten kunne styres med PWM. Dog startede motoren på ca. 30% fart i den ene retning. Ved at måle på PWM signalet fra mainboardet og signalet til motoren kunne problemet indskrænkes til at være på Motorcontrolleren.

Det viste sig efter megen debugging, at spændingen på gaten på P-kanal HEXFETerne (IRF4905) ikke gik `HIGH` lige så hurtigt som forventet. Der blev opstillet et forsøg på et breadboard med en P-kanal HEXFET, en optocoupler og en Arduino.

FiXme: Indsæt diagram over forsøg med optocoupler og HEXFET

Forsøget viste, at når optocoupleren sad mellem HEXFETen og Arduinoen var der en kapacitet mellem HEXFETens gate og source. Figur 4 viser nederst PWM signalet fra Arduinoen og øverst signalet på P-kanal HTXFETens gate. Man ser tydeligt at det tager en ubelejlig tid før signalet på gaten stiger.



Figur 4: Oscilloskop billede af stigetid på en P-kanal HEXFET gate. Nederst ses PWM-signalet fra Arduinoen, øverst ses signalet på gaten.

Ved at sætte en mindre pull-up-modstand på kunne den aflades hurtigere, men det var ikke muligt at få den tilpas langt ned til at kunne styre motoren godt. Ved at fjerne optocoupler og køre HEXFETens gate direkte fra Arduinoen eller ved fjerne HEXFETen og måle direkte på optocoupleren, var stigningstiden ≈ 0. Det var kun i kombination mellem HEXFETen og transistoren i optocoupleren at stigningstiden ikke var ≈ 0.

Der blev forsøgt med en 4N25 optocoupler istedet for 4N35 og en BC547 istedet for optocoupleren; der var samme stigningstid.

Det har ikke været muligt, selv med hjælp fra vejleder, at forklare hvorfor denne kapacitet er der.

Problemet blev ikke løst, det blev bare gjort ubetydeligt: Istedet for at bruge en N- og en P-kanal HEXFET til at bestemme retning og køre PWM på de andre to N- og P-kanal HEXFETer, blev det lavet om til at begge P-kanal HEXFETer blev brugt til at bestemme retning og at N-kanal HEXFETerne bliver brugt at køre PWM. Det er ikke et problem at stigetiden på P-kanal HEXFETerne er langsom da de kun ændre sig når der skiftes retning og ikke med høj frekvens som ved PWM.

For ikke at tilføje flere optocouplere og bruge flere pins på arduinoen blev der, på Motorcontrolleren tilføjet to invertere. Se figur

FiXme: ref: dia:v3.0

Motorcontroller v3.0

FiXme: Indset diagram over Motorcontroller v3.0 (Figur over printet kan findes i bilag)

27. marts 2012 Efter at der blev tilføjet en inverter på to af gatesne til P-kanal HEXFETerne er denne low når der ikke er spænding på optocouplerne (fx når den ikke er koblet til Mainboardet). Det tænder HEXFETen, det, sammen med N-kanal HEXFETerne som også er tændt uden spænding

på optocouplerne, kortslutter H-broen. Motorcontroller v3.0 var fungerede, men det var upraktisk at den var kortsluttet unden at være koblet sammen med Mainboardet.

Pull-up modstandende blev erstattet af pull-down.

Motorcontroller v4.0

12. april 2012 Dette board blev aldrig lavet færdigt. En stor del af boardet blev re-routed da der var blevet rodet efter mange versioner. Diagram og Figur over printet kan findes i bilag.

FiXme: ref

Motorcontroller v4.1

13. april 2012 Der var en ledning der ikke var routed og der var noget mindre re-routing.

Motorcontroller v4.2

17. april 2012 Dette board blev aldrig lavet færdigt. Formodstandene til optocouplerne blev flyttet fra mainboardet til Motorcontrolleren, for at undgå, at disse blev brændt af.

Motorcontroller v5.0

24. april 2012 Dette board blev aldrig lavet færdigt. Der blev tilføjet LEDer til optocouplerne så man kan se hvornår de er tændt, og for at lette debugging.

Motorcontroller v5.1

24. april 2012 Dette board blev aldrig lavet færdigt. Nogle pins blev flyttet i fladkablet.

Motorcontroller v5.2

24. april 2012

Dette board fungerer og sidder i Swagwayen.

FiXme: Board brænder af
FiXme: ref til fig:mosch5.2

Motorcontroller v6.0

24. april 2012

5 Auxiliary

5.1 Mainboard

Mainboard v1.0

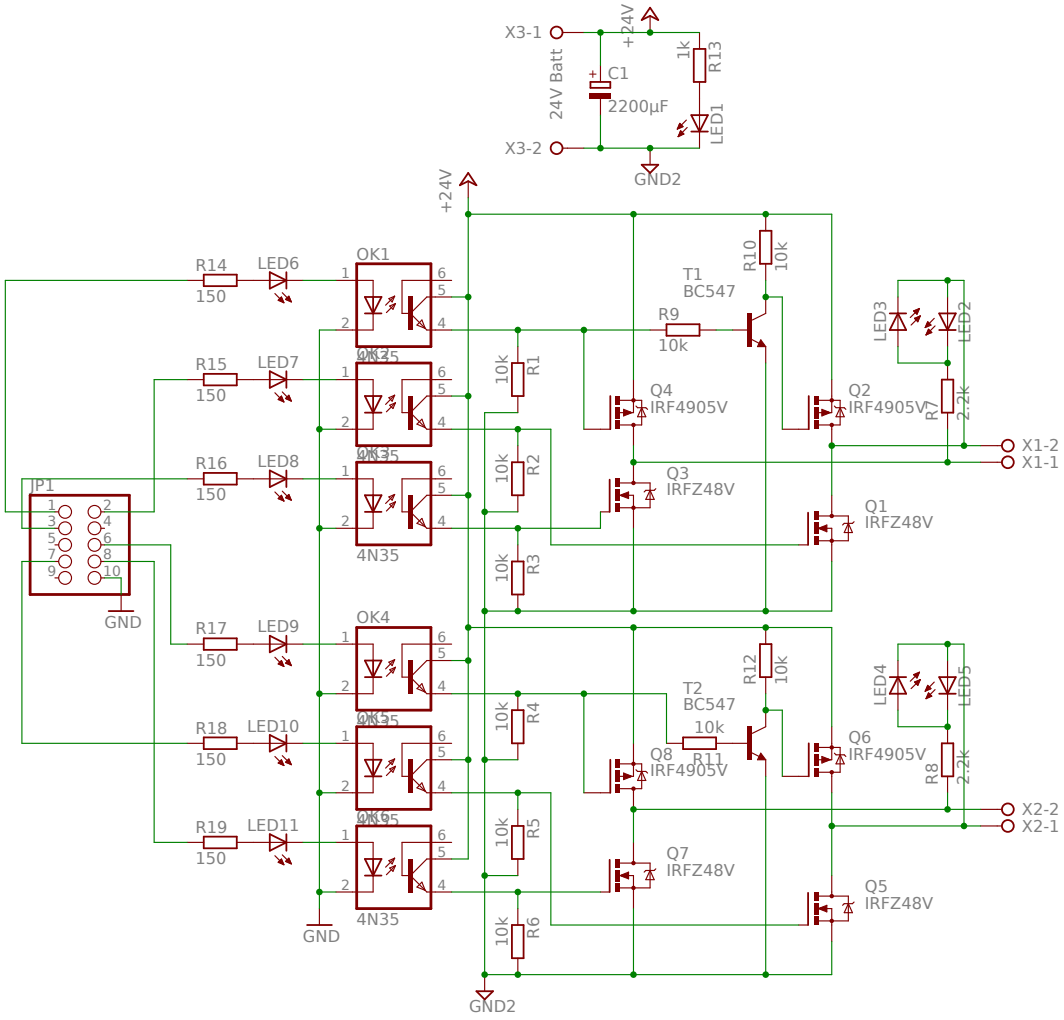
24 jan 2012 Loddeøerne var underdimmentioneret[#1] og det var ikke til at komme til at trykke på resetknappen på Arduinoen da bordet dækkede overdet.[#2] Der blev tilføjet en resetknap på mainboardet samt et stik til at læse data fra styret.[#12]

Mainboard v2.0

1 marts 2012 Logik kredsløbet blev opgivet og efterfølgende blev der brugt tre Arduino pins per motor.[#21] Displayboardet blev ligeledes opgivet.[#9] Pinheaderne til 9V og IMUen var desuden for tæt sammen.[#13]

Mainboard v3.0

26 marts 2012 Dette board blev aldrig lavet færdig. Tilføjet pins til radio.[#29]



Figur 5: Diagram over Motorcontroller v5.2

Mainboard v3.1

29 marts 2012 Formodstandene til optocouplerne på motorboardet blev flyttet til motorboardet.[#37] Pinheaderen til IMUen blev lavet om fra 2×7 til 2×5

Mainboard v4.0

24 april 2012 Swagwayen kører med dette board, men den ene motor kørte hurtigere end den anden.[#42] Det lignede umiddelbart en mekanisk fejl, men ved at bytte om på ledningerne til motorene viste det sig at den ene kanal på kotorcontrolleren kørte langsommere end den anden. Problement blev isoleret til at Arduinoen ikke sendte PWM med samme frekvens til begge kanaler. I Arduino referencen for `AnalogWrite()` ser man også at pin 5 og 6 PWM kører fra en anden timer end de andre PWM pins. Tilfældigvis kørte den ene motor på begge af disse pins. Pin 5 og 9 blev byttet så Swagwayen kører lidt hurtigere forlens end baglens, men med samme forskel på begge hjul.

Opbytningen af de to pins blev gjort ved at bryde kobberbanerne på printet og lodde to lednin-

Tabel 2: Pin forbindelser på Arduino

Pin	Forbindelse	Egenskaber
0	USB Rx	
1	USB Tx	
2	Radio Rx	Interrupt
3		Interrupt, PWM
4	Radio Tx	
5	Motorcontroller R2	PWM[a]
6	Motorcontroller L2	PWM[a]
7	Motorcontroller L1	
8	Motorcontroller R1	
9	Motorcontroller L3	PWM
10	Motorcontroller R3	PWM
11		PWM
12		
13		LED
A0		
A1		
A2	Steering	
A3	Steering	
A4	IMU I²C SDA	SDA
A5	IMU I²C SCL	SCL

[a] PWM outputtet fra disse er lidt højere end forventet. De drives af en anden timer.[#42] Se under Mainboard 4.0 i sektion 5.1.

ger på, der blev ikke lavet et nyt board.

6 Mekanik

Motorer, batterier,

7 Konklusion

Vi satte os de og de mål, vi nåede de og de mål.

8 Perspektivering

Styring Kraftigere motorer med mindre gearkasser. Encodere

Tabeller

Figurer

Litteratur

A Kildekode

A.1 swagway.ino

```
/*************************************************/
/* swagway.ino -- Swagway onboard software       */
/*                                               */
/* Author: Mathias Dannesbo <neic@neic.dk> and   */
/*         Carl-Emil Grøn Christensen            */
/* Time-stamp: <2012-05-07 18:07:19 (neic)>      */
/* Part of the Swagway project                   */
/* https://github.com/neic/Swagway               */
/*                                               */
/*************************************************/

#include <Wire.h>
#include <math.h>
#include "ITG3200.h"
#include "ADXL345.h"

// IMU
ADXL345 acc = ADXL345();
float accSampleRate;
ITG3200 gyro = ITG3200();
float gyroSampleRate;

// General
int xa, ya, za;
float xg, yg, zg;

unsigned long sinceLastSend;

bool newAccData, newGyroData;

double accAngle, gyroAngle, estAngle;

// Kalman filter
const float Q_angle = 0.001; // Process noise covariance for the
    accelerometer - Sw
const float Q_gyro = 0.003; // Process noise covariance for the gyro - Sw
const float R_angle = 0.03; // Measurement noise covariance - Sv

double angle = 0; // It starts at 0 degrees
double bias = 0;
double P_00 = 0, P_01 = 0, P_10 = 0, P_11 = 0;
double dt, y, S;
double K_0, K_1;

// Motor

const int directionPinLeft = 7; //HIGH when forward
const int forwardPinLeft = 9;
const int backwardPinLeft = 6;

const int directionPinRight = 8; //HIGH when forward
const int forwardPinRight = 10;
const int backwardPinRight = 5;

```

```
54 // PID
55
56 const int targetAngle = 0;
57 const float Ex = 2.6; //Exponential value
58 const float Kp = 2; //Proportional value
59
60 void setup()
61 {
62   Serial.begin(115200);
63   Wire.begin();
64
65   //Init the acc
66   acc.init(ADXL345_ADDR_SDO_LOW);
67   acc.setFullRes(true);
68   acc.setRange(3);
69   acc.setVoltage(3.3);
70   acc.setOutputRate(10); //25Hz*2^(10-8)=100Hz. See table 7 in ADXL345
         datasheet
71
72   //Calculate the accSampleRate
73   accSampleRate = 25*pow(2,(acc.getOutputRate()-8)); //See table 7 in
         ADXL345 datasheet
74
75   //Init the gyro
76   gyro.init(ITG3200_ADDR_AD0_LOW);
77   gyro.setSampleRateDiv(79); //Set the sample rate to 8000Hz/(79+1)=100Hz
78
79   //Calculate the gyroSampleRate
80   if (gyro.getFilterBW() == BW256_SR8)
81   {
82     gyroSampleRate = 8000 / (gyro.getSampleRateDiv()+1);
83   }
84   else
85   {
86     gyroSampleRate = 1000 / (gyro.getSampleRateDiv()+1);
87   }
88
89   pinMode(directionPinLeft, OUTPUT);
90   pinMode(forwardPinLeft, OUTPUT);
91   pinMode(backwardPinLeft, OUTPUT);
92   pinMode(directionPinRight, OUTPUT);
93   pinMode(forwardPinRight, OUTPUT);
94   pinMode(backwardPinRight, OUTPUT);
95   //Calibration
96   gyro.zeroCalibrate(250,2);
97
98   //Dump settings
99   dumpIMUsettings();
100 }
101
102 void loop()
103 {
104   if (acc.isRawDataReady())
105   {
106     acc.readAccRaw(&xa,&ya,&za);
107     accAngle = atan2(xa,ya)*180/3.1415; // calcutalte the X-Y-angle
108     newAccData = true;
109   }
```

```
110
111   if (gyro.isRawDataReady())
112   {
113     gyro.readGyro(&xg,&yg,&zg);
114     gyroAngle += zg/gyroSampleRate;
115     newGyroData = true;
116   }
117
118   if (newAccData && newGyroData)
119   {
120     estAngle = kalman(accAngle, zg, micros()-sinceLastSend);
121     //      sendToGraph();
122     newAccData = newGyroData = false;
123     float pwm = pid(estAngle);
124     motorControl(pwm,pwm);
125     Serial.print(estAngle);
126     Serial.println(pwm);
127
128     sinceLastSend = micros();
129   }
130 }
131
132   double kalman(double newAngle, double newRate, double dtime) {
133     // KasBot V2 - Kalman filter module - http://www.arduino.cc/cgi-bin/
          yabb2/YaBB.pl?num=1284738418 - http://www.x-firm.com/?page_id=145
134     // with slightly modifications by Kristian Lauszus
135     // See http://academic.csuohio.edu/simond/courses/eec644/kalman.pdf
          and http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf for
          more information
136       dt = dtime / 1000000; // Convert from microseconds to seconds
137
138     // Discrete Kalman filter time update equations - Time Update ("
          Predict")
139     // Update xhat - Project the state ahead
140     angle += dt * (newRate - bias);
141
142     // Update estimation error covariance - Project the error covariance
          ahead
143     P_00 += -dt * (P_10 + P_01) + Q_angle * dt;
144     P_01 += -dt * P_11;
145     P_10 += -dt * P_11;
146     P_11 += +Q_gyro * dt;
147
148     // Discrete Kalman filter measurement update equations - Measurement
          Update ("Correct")
149     // Calculate Kalman gain - Compute the Kalman gain
150     S = P_00 + R_angle;
151     K_0 = P_00 / S;
152     K_1 = P_10 / S;
153
154     // Calculate angle and resting rate - Update estimate with
          measurement zk
155     y = newAngle - angle;
156     angle += K_0 * y;
157     bias += K_1 * y;
158
159     // Calculate estimation error covariance - Update the error
          covariance
```

```
    P_00 -= K_0 * P_00;
    P_01 -= K_0 * P_01;
    P_10 -= K_1 * P_00;
    P_11 -= K_1 * P_01;

    return angle;
  }

float pid(float input)
{
  float output;
  if (input>0)
  {
   output = pow(input,Ex) + Kp*input;
  }
  else
  {
    output = -pow(-input,Ex) + Kp*input;
  }
  return constrain(output, -255, 255);
}

void motorControl(int left, int right)
{
  if (left < 0)
  {
    digitalWrite(directionPinLeft, HIGH);
    digitalWrite(backwardPinLeft, LOW);
    analogWrite(forwardPinLeft, -left);
  }
  else
  {
    digitalWrite(directionPinLeft, LOW);
    digitalWrite(forwardPinLeft, LOW);
    analogWrite(backwardPinLeft, left);
  }
  if (right < 0)
  {
    digitalWrite(directionPinRight, HIGH);
    digitalWrite(backwardPinRight, LOW);
    analogWrite(forwardPinRight, -right);
  }
  else
  {
    digitalWrite(directionPinRight, LOW);
    digitalWrite(forwardPinRight, LOW);
    analogWrite(backwardPinRight, right);
  }
}
/* Serial communication */

void sendToGraph()
{
  Serial.print("<");
  Serial.print(gyroAngle); //0
  Serial.print(",");
  Serial.print(accAngle); //1
  Serial.print(",");
```

```
218   Serial.print(estAngle); //2
219   Serial.print(",");
220   Serial.print(micros()-sinceLastSend); //3
221   Serial.println(">");
222 }
223
224 void dumpIMUsettings()
225 {
226   Serial.println();
227   Serial.println("=======================================");
228   Serial.println("==============IMU Settings==============");
229   Serial.println();
230   Serial.println("               ---Gyro---                ");
231   Serial.print("Sample rate                    (Hz) = ");
232   Serial.println(gyroSampleRate,0);
233   Serial.println();
234   Serial.println("                ---Acc---                ");
235   Serial.print("Sample rate                    (Hz) = ");
236   Serial.println(accSampleRate,0);
237   Serial.print("Full resolution                     = ");
238   Serial.println(acc.getFullRes());
239   Serial.print("Range                           (g) = ");
240   Serial.println(pow(2,(1+acc.getRange())),0);
241   Serial.print("Scale factor X             (LBS/g) = ");
242   Serial.println(acc.scaleFactor[0],0);
243   Serial.print("Scale factor Y             (LBS/g) = ");
244   Serial.println(acc.scaleFactor[1],0);
245   Serial.print("Scale factor Z             (LBS/g) = ");
246   Serial.println(acc.scaleFactor[2],0);
247   Serial.println();
248   Serial.println("============end IMU Settings============");
249   Serial.println("=======================================");
250   Serial.println();
251 }
```

A.2 ADXL345.h

```
1  /***********************************************/
2  /* ADXL345.h -- ADXL345/I2C libary for Arduino   */
3  /*                                               */
4  /* Author: Mathias Dannesbo <neic@neic.dk> and   */
5  /*         Carl-Emil Grøn Christensen            */
6  /* Time-stamp: <2012-04-04 17:37:05 (neic)>      */
7  /* Part of the Swagway project                   */
8  /* https://github.com/neic/Swagway               */
9  /*                                               */
10 /* Inspired by the ITG3200 Arduino library at    */
11 /* http://code.google.com/p/itg-3200driver       */
12 /***********************************************/
13
14 #ifndef ADXL345_h
15 #define ADXL345_h
16
17 #if defined(ARDUINO) && ARDUINO >= 100
18 #include "Arduino.h"
19 #else
20 #include "WProgram.h"
21 #endif
```

```
22
23 #define ADXL345_ADDR_SDO_HIGH 0x1D
24 #define ADXL345_ADDR_SDO_LOW 0x53
25
26 // Registers
27 #define BW_RATE         0x2C // RW SETUP: Output rate and low power mode
28 #define POWER_CTL       0x2D // RW SETUP: Power control
29 #define INT_SOURCE      0x30 // R  INTERRUPT: Status
30 #define DATA_FORMAT     0x31 // RW SETUP: Self-test and data format
31 #define DATAX0          0x32 // R  SENSOR: Data
32
33 // Bitmaps
34 #define STANDBY_MODE    0x00 // 0000 0000
35 #define MEASURE_MODE    0x08 // 0000 1000
36
37 class ADXL345
38 {
39  public:
40   float scaleFactor[3];
41   float voltage;
42
43   ADXL345();
44
45   void init(unsigned int address);
46
47   // SETUP: Mode
48   void setStandbyMode();
49   void setMeasureMode();
50
51   // SETUP: Output Rate
52   byte getOutputRate();
53   void setOutputRate(byte _SampleRate);
54
55   // SETUP: Data format
56   bool getFullRes();
57   void setFullRes(bool fullRes);
58   int getRange();
59   void setRange(int range);
60
61   // INTERRUPT
62   bool isRawDataReady();
63
64   // SETUP: Data processing
65   void setVoltage(float _voltage);
66   void updateScaleFactor();
67
68   // SENSOR: Read
69   void readAccRaw(int *_AccX, int *_AccY, int *_AccZ);
70   void readAcc(float *_AccX, float *_AccY, float *_AccZ);
71
72   void writemem(uint8_t _addr, uint8_t _val);
73   void readmem(uint8_t _addr, uint8_t _nbytes, uint8_t __buff[]);
74
75  private:
76   uint8_t _dev_address;
77   uint8_t _buff[6];
78 };
79
```

```
80 #endif /* ADXL345_h */
```

A.3 ADXL345.cpp

```
1  /*********************************************/
2  /* ADXL345.cpp -- ADXL345/I2C libary for Arduino */
3  /*                                           */
4  /* Author: Mathias Dannesbo <neic@neic.dk> and   */
5  /*         Carl-Emil Grøn Christensen        */
6  /* Time-stamp: <2012-04-04 18:04:52 (neic)>      */
7  /* Part of the Swagway project               */
8  /* https://github.com/neic/Swagway           */
9  /*                                           */
10 /* Inspired by the ITG3200 Arduino library at    */
11 /* http://code.google.com/p/itg-3200driver       */
12 /*********************************************/
13
14 #include "ADXL345.h"
15 #include <Wire.h>
16
17 ADXL345::ADXL345()
18 {
19 }
20
21 void ADXL345::init(unsigned int address)
22 {
23   _dev_address = address;
24   setStandbyMode();
25   setMeasureMode();
26
27 }
28
29 void ADXL345::setStandbyMode()
30 {
31   writemem(POWER_CTL, STANDBY_MODE);
32 }
33
34 void ADXL345::setMeasureMode()
35 {
36   writemem(POWER_CTL, MEASURE_MODE);
37 }
38
39 byte ADXL345::getOutputRate()
40 {
41   readmem(BW_RATE, 1, &_buff[0]);
42   return(_buff[0]);
43 }
44
45 void ADXL345::setOutputRate(byte _rate)
46 {
47   _rate %= 16; //Prevent overflow
48   writemem(BW_RATE, _rate);
49 }
50
51 bool ADXL345::getFullRes()
52 {
53   readmem(DATA_FORMAT, 1, &_buff[0]);
54   return(_buff[0] >> 3);
```

```
55  }
56
57  void ADXL345::setFullRes(bool _fullRes)
58  {
59    readmem(DATA_FORMAT, 1, &_buff[0]);
60    writemem(DATA_FORMAT, ((_buff[0] & ~(1 << 3)) | (_fullRes << 3)));
61  }
62
63  int ADXL345::getRange()
64  {
65    readmem(DATA_FORMAT, 1, &_buff[0]);
66    return(_buff[0] & B00000011);
67  }
68
69  void ADXL345::setRange(int range)
70  {
71    range %= 4; //Prevent overflow
72    readmem(DATA_FORMAT, 1, &_buff[0]);
73    writemem(DATA_FORMAT, ((_buff[0] & ~3) | range));
74  }
75
76
77  bool ADXL345::isRawDataReady()
78  {
79    readmem(INT_SOURCE, 1, &_buff[0]);
80    return(_buff[0] >> 7);
81  }
82  void ADXL345::setVoltage(float _voltage)
83  {
84    voltage = _voltage;
85    updateScaleFactor();
86  }
87
88  void ADXL345::updateScaleFactor()
89  {
90    int rangeScale=256;
91    if (!getFullRes())
92      {
93        rangeScale = pow(2,(8-getRange()));
94      }
95    scaleFactor[0] = rangeScale*0.89013671875+rangeScale*0.0439453125*
         voltage;
96    scaleFactor[1] = rangeScale*0.89013671875+rangeScale*0.0439453125*
         voltage;
97    scaleFactor[2] = rangeScale;
98  }
99
100 void ADXL345::readAccRaw(int *_AccX, int *_AccY, int *_AccZ)
101 {
102   readmem(DATAX0, 6, &_buff[0]);
103   *_AccX = _buff[1] << 8 | _buff[0];
104   *_AccY = _buff[3] << 8 | _buff[2];
105   *_AccZ = _buff[5] << 8 | _buff[4];
106 }
107
108 void ADXL345::readAcc(float *_AccX, float *_AccY, float *_AccZ)
109 {
110   int x, y, z;
```

```
111   readAccRaw(&x,&y,&z);
112   *_AccX = x / scaleFactor[0];
113   *_AccY = y / scaleFactor[1];
114   *_AccZ = z / scaleFactor[2];
115 }
116
117 void ADXL345::writemem(uint8_t _addr, uint8_t _val) {
118   Wire.beginTransmission(_dev_address);   // start transmission to device
119   Wire.write(_addr); // send register address
120   Wire.write(_val); // send value to write
121   Wire.endTransmission(); // end transmission
122 }
123
124 void ADXL345::readmem(uint8_t _addr, uint8_t _nbytes, uint8_t __buff[]) {
125   Wire.beginTransmission(_dev_address); // start transmission to device
126   Wire.write(_addr); // sends register address to read from
127   Wire.endTransmission(); // end transmission
128
129   Wire.beginTransmission(_dev_address); // start transmission to device
130   Wire.requestFrom(_dev_address, _nbytes);// send data n-bytes read
131   uint8_t i = 0;
132   while (Wire.available()) {
133     __buff[i] = Wire.read(); // receive DATA
134     i++;
135   }
136   Wire.endTransmission(); // end transmission
137 }
```

B Status log

B.1 13. marts

Mainbord er fungerende. v2.0 af motorboardet er næsten færdig.
Kredsløbet uden om printne er næsten færdig.
Vi kan læse data fra IMUen og vi har et halvt implementert kalman-filter.
Efter kalmanfilteret fungere skal der implementeres PID med wrapper kode.