



CANSAT KURSUS

efterår 2021

Jens Dalsgaard Nielsen, AAU
Henrik Bay Madsen, NVH
Steen Eiler Jørgensen, Astra

se også

<http://jensd.dk/cansat>



tentativ agenda

- velkommen
- udlevering af kit
 - install af SW
 - arduino IDE
- gennemgang af cansat shield
 - bibliotekter for cansat shield
- Afdækning af folks programmeringserfaringer
 - gætter på fra ingenting til meget kompetente
- enkle hello world programmer
- start på kit
- trykmåler (bmp085/bmp180/...)
- accelerometer (MPU6050)
- gyro (MPU6050)
- magnetometer (hmc5983)
- logning på sd card
- radiolink
- apc220
- configuration
- den store test
- pewwww



ARDUINO INSTALL

- <http://arduino.cc>
- <https://www.arduino.cc/en/software>

The screenshot shows the Arduino website's software download page. The URL in the address bar is https://www.arduino.cc/en/software. The page features a navigation bar with links for TION, STORE, HARDWARE, SOFTWARE (which is highlighted), DOCUMENTATION, COMMUNITY, BLOG, and ABOUT. Below the navigation bar, there is a search bar labeled "Search on Arduino.cc". The main content area displays the Arduino Web Editor interface, which includes an editor window with code and a sidebar with options like Sketchbook, Examples, Libraries, and Serial Monitor.

The screenshot shows the Arduino Web Editor interface. It features a toolbar at the top with "EDITOR", "NEW SKETCH", "SEARCH SKETCHBOOK", and "LIBRARIES". The main workspace shows a sketch named "sketch_feb10" with some code. The sidebar on the left includes "Sketchbook", "Examples", "Libraries", and "Serial Monitor". Below the editor, there are two buttons: "CODE ONLINE" and "GETTING STARTED".

Downloads

The screenshot shows the Arduino IDE 1.8.13 download page. At the top, there is a large "Arduino IDE 1.8.13" heading with a circular icon containing two overlapping circles. Below the heading, there is a brief description: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board." A link to the "Getting Started" page for installation instructions is provided. Further down, there is a "SOURCE CODE" section with a note about development being hosted on GitHub and instructions for building the code. A "Mac OS X" section is also present.

The screenshot shows the "DOWNLOAD OPTIONS" page for the Arduino software. It lists download links for various operating systems:

- Windows** Win 7 and newer
- Windows** ZIP file
- Windows app** Win 8.1 or 10 [Get](#)
- Linux** 32 bits
- Linux** 64 bits
- Linux** ARM 32 bits
- Linux** ARM 64 bits
- Mac OS X** 10.10 or newer

Below these options, there are links for "Release Notes" and "Checksums (sha512)".

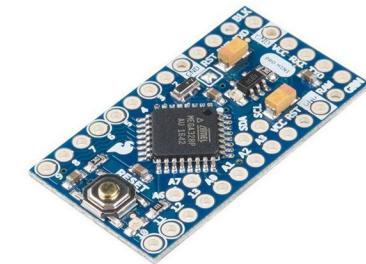


C - a programming language

- Small language
- Used everywhere
 - from micro-controllers running slower than 1MHz in clock
 - to super computers running more than 3 GHz
 - A span of more than 3000 times (not ocunting CPUs)
 - from less than 1 kByte memory in systems
 - to much more than TBytes
- Language for coding linux operating system
- Language for coding MAC OS
- Windows ??
- You name it



Az-Delivery





nogle links

- <https://www.arduino.cc/en/Tutorial/HomePage>
- <https://microcontrollerslab.com/arduino-programming-tutorial-beginners/>
- <http://jensd.dk/cansat/c-course>
- <http://jensd.dk/cansat>



Programming in an imperative language

- Imperative language
- Execution of statement programmed by you
- One at a time
- In the order given by you - Order is important - change in order might change result.
- Programs are structured in blocks or functions
- Blocks of functions can execute other blocks



Programming in an imperative language

- Imperative language
- Execution of statement programmed by you
- One at a time
- In the order given by you - Order is important - change in order might change result.
- Programs are structured in blocks or functions
- Blocks of functions can execute other blocks



Seq ordered execution

- 1) start code (invisible)
- 2) execute **setup** statement by statement
- 3) back to invisible part
- 4) execute loop statement by statement
- 5) goto 4) and execute again
- 6) until infinity ...

```
File Edit Sketch Tools Help
sketch_aug26b §
1
2
3
4 void setup() {
5
6 }
7
8 void loop() {
9
10 }
11 }
```

A statement is either

- execution of a block/function
 - and return back to where it was called
- execution of a statement



Functions

Order of execution

```
1) setup  
2) loop  
   1) aBlock  
      1) nextLevelBlock  
      2) anotherBlock  
   3) back to 2)
```

```
setup  
loop  
  aBlock  
    nextLevelBlock  
    anotherBlock  
loop (again)  
  aBlock  
    nextLevelBlock  
    anotherBlock  
loop (again)  
  aBlock  
    nextLevelBlock  
loop (again)  
  aBlock  
    nextLevelBlock  
  anotherBlock  
...
```

The screenshot shows the Arduino IDE interface with a sketch titled "sketch_aug26b". The code is as follows:

```
1  void nextLevelBlock() // head or interface  
2 {  
3   // code missing  
4 }  
5  
6  
7  void aBlock() // head or interface  
8 {  
9   // body - the functionallity  
10  // ...  
11  nextLevelBlock();  
12 } // ...  
13 void anotherBlock() // head or interface  
14 {  
15   // here will come som code  
16 }  
17  
18  
19  
20 void setup() {  
21   // missing code  
22 }  
23  
24 void loop() {  
25   aBlock(); // calling or execute aBlock  
26   anotherBlock();  
27 }
```



Functions “1”

Consists

- a head : your interface to the function
- a body: which holds the code the function execute

head - syntax:

```
void NAME ()      - name might be a, bec vrst  
                  - case sensitive Jens, jens,jeNS,jENS are different names  
                  - you can't use national letters like jørgen (ø :-()
```

examples

```
void setup()    :-)  
void myFirstfunction()
```

body - syntax:

```
{  
  // statement or calling another function  
}
```



Functions “2”

```
void x()  
{  
    // do something  
}
```

execution of function x

...
x(); // name + () + ; ;" is end of a statement (like “.” in danish)



the C/function flow

- /&%¤/%¤/&#"¤¤%(/&%
- loop calls aBlock
 - aBlock calls nextLevelBlock
 - nextLevelBlock executes and return to ...
 - aBlock which returns to ...
- loop
- loop calls anotherBlock
 - anaotherBlocks executes and return to ...
- loop
- loop return (after last statement)
- loop “loops” to start of loop and
- loop calls aBlock (first bullet)
- ... etc

Line numbers

24

25->7 (call aBlock)

8,9,10->2

2,3,4,5 return to 10+

11 return to 25+

26->13 (call anotherBlock)

14,15,16 return to 26+

27 loop to 24

The screenshot shows the Arduino IDE interface with a sketch titled "sketch_aug26b". The code is as follows:

```
File Edit Sketch Tools Help
sketch_aug26b.ino
1
2 void nextLevelBlock() // head or interface
3 {
4     // code missing
5 }
6
7 void aBlock() // head or interface
8 {
9     // body - the functionallity
10    // ...
11    nextLevelBlock();
12    // ...
13
14 void anotherBlock() // head or interface
15 {
16     // here will come som code
17
18
19
20 void setup() {
21     // missing code
22 }
23
24 void loop() {
25     aBlock(); // calling or execute aBlock
26     anotherBlock();
27 }
```



Need some debugging - need to see what is going on

The image shows the Arduino IDE interface. On the left is the code editor with the following C++ code:

```
File Edit Sketch Tools Help
aaaa
1 void setup() {
2   Serial.begin(115200);
3   delay(1000); // relax
4 }
5
6 void aaa()
7 {
8   Serial.println("aaa");
9   delay(1000); // wait one second here
10  return;
11 }
12
13 void aa()
14 {
15   Serial.println("aa");
16   delay(1000); // wait one second here
17   aaa();
18   return;
19 }
20
21 void a()
22 {
23   Serial.println("a");
24   delay(1000); // wait one second here
25   aa();
26   return;
27 }
28
29 void loop() {
30   Serial.println("loop");
31   a(); // calls function a
32   delay(5000); // calls delay which delay execution by 5000 mill
33 }
```

On the right is the串行监视器 (Serial Monitor) window, which displays the following output:

```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot
configSIP: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
loop
a
aa
aaa
```

Arrows from the code editor point to the `Serial.println` statements in the `aaa()`, `aa()`, and `a()` functions. Arrows from the serial monitor point back to the same lines in the code editor, indicating the flow of data from the code to the monitor.



Serial.println - a function

- prints :-)
- only one item: text or numbers.
- numbers
 - 1 2 456 -34 integers (heltal)
 - 1.23 -1.23e7 floating point (kommatal)
- text
 - "letters in between"
- Serial.println("ost"); //prints ost and force a linebreak after
- Serial.println(123); // prints 123 and force a ...
- Need to initialize port with writing speed
- Serial.begin(115200); // for ce nerds 115200 bit pr second



```
aaaa
1 void setup() {
2   Serial.begin(115200);
3   delay(1000); // relax
4 }
5
6 void aaa()
7 {
8   Serial.println("aaa");
9   delay(1000); // wait one second here
10  return;
11 }
```



serial interface

- 115200 bits pr second
- delay pause execution
- “return” returns from a function

```
aaaa
1 void setup() {
2   Serial.begin(115200);
3   delay(1000); // relax
4 }
5
6 void aaa()
7 {
8   Serial.println("aaa");
9   delay(1000); // wait one second here
10  return;
11 }
```



serial terminal (in menu “tools”)

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with various icons. The main area contains a code editor with the following sketch:

```
aaaa
1 void setup() {
2   Serial.begin(115200);
3   delay(1000); // relax
4 }
5
6 void aaa()
7 {
8   Serial.println("aaa");
9   delay(1000); // wait one second
10  return;
11 }
12
13 void aa()
14 {
15   Serial.println("aa");
16   delay(1000); // wait one second
17   aaa();
18   return;
19 }
20
21 void a()
22 {
23   Serial.println("a");
}
```

To the right of the code editor is the Serial Terminal window. It has a title bar showing the port name: /dev/ttyUSB0. The window contains a text input field and a "Send" button. At the bottom of the terminal window are several configuration options: Autoscroll, Show timestamp, Both NL & CR, 115200 baud, and Clear output.



TEST - 1

- Arduino UNO
- testprogram
- File -> examples -> digital -> BlinkWithoutDelay
- Tools-> Board -> Uno
- Tools -> Port -> den USB port din Arduino sidder på
- ctrl-u (compiler og upload)
- blinker LED
- JA: mission completed



første rigtige program

-
-
- Lav ADC eksempel
- print måling med 5 msec delay imellem
- sæt en ledning til adc indgang og hold ved den
- print og plot og se resultat
- BRUG FUNKTIONER !!!
- med parametre (Jens viser eksempel online)



```
imp03-ex01
1 void setup() {
2     Serial.begin(115200);
3     delay(1000);
4 }
5
6 int maaling;
7 float voltage;
8
9 float omregn(int val, int valMax, float vMax)
10{
11     return ( val * vMax / valMax );
12 }
13
14 void loop() {
15     maaling = analogRead(36); // nodeMCU adc0
16     voltage = omregn(maaling, 4095, 3.3);
17     Serial.println(voltage, 3);
18     delay(20);
19 }
20
21 /* TRICKS
22 For at undgaa debug beskederne når man rebooter
23 kan man sætte GPIO15 til stel(gnd).
24 HVis du vil serial plotte ovenstående så
25
26 1. hold RESET knappen nede
27 2. start upload
28 3. slip RESET knappen lidt efter
29
30 efter upload
31
32 1. hold EN knappen nede
33 2. start seriell plotter
34 3. slip EN knappen
35
36 JDN
37 */
```



```
1 void setup() {
2     Serial.begin(115200);
3     delay(1000);
4 }
5
6 int maaling;
7 float voltage;
8
9 float omregn(int val, int valMax, float vMax)
10{
11    return ( val * vMax / valMax );
12}
13
14 void loop() {
15    maaling = analogRead(36); // nodeMCU adc0
16    voltage = omregn(maaling, 4095, 3.3);
17    Serial.println(voltage, 3);
18    delay(20);
19 }
```



exercise 2

- Lav ADC eksempel
- print måling med 5 msec delay imellem
- sæt en ledning til adc indgang og hold ved den
- print og plot og se resultat
-
- Lav midlingsrutine så du mäter 5 gange med 1 millisek imellem og midler
- print
 - alle 5 värder
 - och medelvärde 5 gange
 - hint Serial.print(arr[i]); Serial.print(" "); Serial.println(average);
 - så kan du plotta dem
- ledning i adc indgang og hold ved ledning



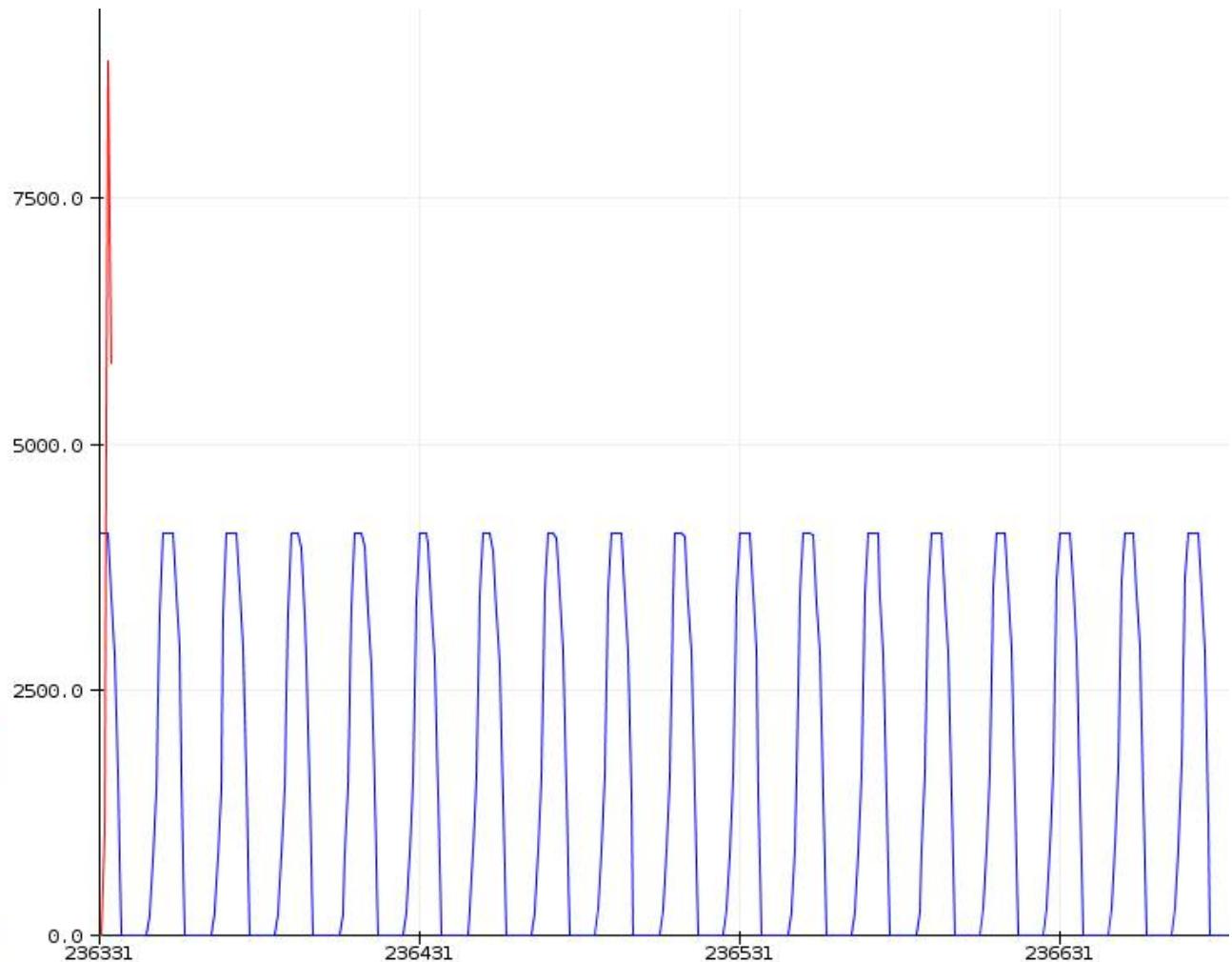
analog Input rå

- har i ledning på pin 36 og holder ved enden af ledningen
- jeg opfanger 50 Hz støj fra verden (får port i mætning da maks jo er 1023)
- ACD får i mætning
- fra 236431 til 236531 er der 100 målinger dvs 100 millisekunder ca (se kode)
- der er 5 perioder som varer $5 \times 20 \text{ msec} = 100 \text{ msec}$ ($50\text{Hz} \sim 20 \text{ msec}$ pr periode) :-) (NB figur er fra en micro med 12 bit oplosning dvs 4095 niveauer)

```
1 void setup() {  
2     Serial.begin(115200);  
3 }  
4 // menu tools(værktøjer på dansk) ->  
5 // ctrl+shift+l  
6  
7 void loop() {  
8     delay(1);  
9     Serial.println(analogRead(36));  
10 }  
11
```

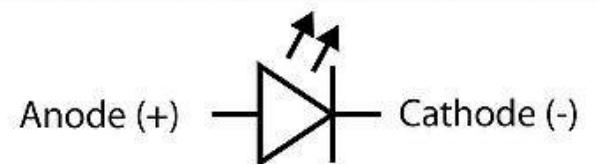
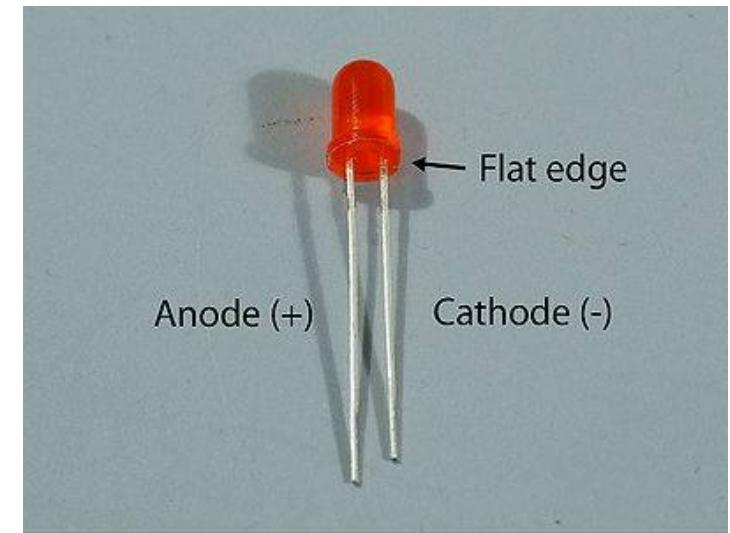
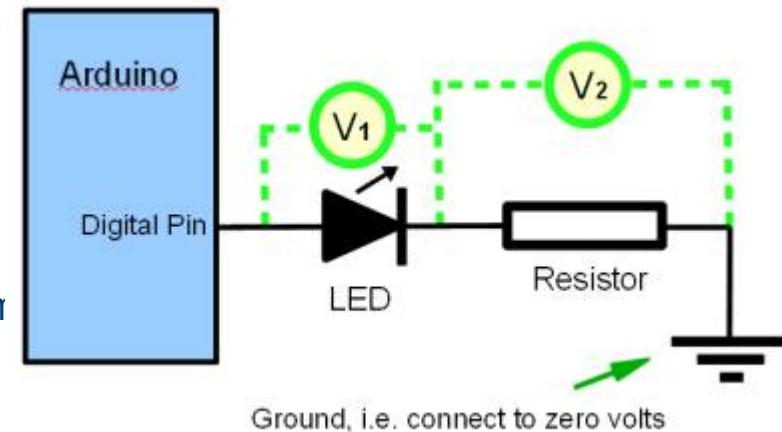
Done uploading.

Invalid library found in /home/jdn/q/code
Invalid library found in /home/jdn/q/code





- ESP32 digital pin HIGH = 3.3V
- LED Voltage drop ~ 2.2V
- 1.1 V left for the diode
- [https://ardustore.dk/produkt/led-diode-3mm/red/](https://ardustore.dk/produkt/led-diode-3mm-red/)
- from the sheet
 - continuous forward current max 30 mA
- $R = (3.3 - 2.2) / 0.03A = 36 \text{ ohm}$
- $i = 10 \text{ mA} \rightarrow R = (3.3 - 2.2) / 0.01A = 110 \text{ ohm}$
- higher current == more light
- pin -> LED -> resistor or
- pin -> resistor -> LED - no difference





Variable - hvad er det

- Storage. Vi skal have en måde at gemme data på mens programmet kører
- Program = kode + data !!!
- Variable skal erklæres **FØR** man bruger som (i modsætning til python)
- Du skal angive **hvilken type** variabel det er (i modsætning til python)
- Der er ret **strenge typecheck** så kan ikke blander æbler og pærer (i mod...)



variabeltype no 1 - heltal

- heltal - tælletal: 1 2 3 4 654 6875 0 -123 -444 -23

Erklæring:

```
<typenavn> <variabelNavn>;  
...  
(std man skriver <> når man forklarer)
```

```
int etHeltal;  
int et andet heltal, kurt;
```

```
1 int helTal; ←  
2  
3 void setup()  
4 {  
5   helTal = 33; ←  
6   Serial.begin(115200);  
7   delay(1000);  
8 }  
9  
10  
11 void loop() {  
12  
13   Serial.print("tal er ");  
14   Serial.println(helTal);  
15  
16   helTal = helTal + 1; ←  
17   // udregn til højre for lighedstegn  
18   // gem i venstre side  
19   delay(500);  
20 }
```



Variable II

- simpel matematik til rådighed

```
int a,b,c,d,res;
```

```
res = (a+b) / (c + d);
```

eller

```
res = res * ((a+b) / (c + d));
```

kan også skrives som

```
res *= ((a+b) / (c + d));
```

BRUG RIGELIGT MED PARANTESER

SLØVER IKKE DIN KODE NED !

```
1 int helTal;
2
3 int andetTal, hest;
4
5 void setup()
6 {
7     helTal = 33;
8     hest = helTal + 4;
9     andetTal = hest + 4*helTal;    // + - * /
10    Serial.begin(115200);
11    delay(1000);
12 }
13
14
15 void loop() {
16
17     Serial.print("tal er ");
18     Serial.println(helTal);
19
20     helTal = helTal + 1;
21     // udregn til højre for lighedstegn
22     // gem i venstre side
23     delay(500);
24 }
```



paranteser

- De to følgende er ens da der er rækkefølge i udførelse af regneoperationer.
- kaldes på engelsk operator precedence
 - https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm
- * og / udføres FØR + -
- Sidste operator der udføres er =



Category	Operator	Associativity
Postfix	() [] -> . ++ --	Left to right
Unary	+ - ! ~ ++ -- (type)* & sizeof	Right to left
Multiplicative	* / % ←	Left to right
Additive	+ - ←	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR		Left to right
Logical AND	&&	Left to right
Logical OR		Left to right
Conditional	? :	Right to left
Assignment	= += -= *= /= %= >>= <<= &= ^= =	Right to left
Comma	,	Left to right



heltal - er der mere ???

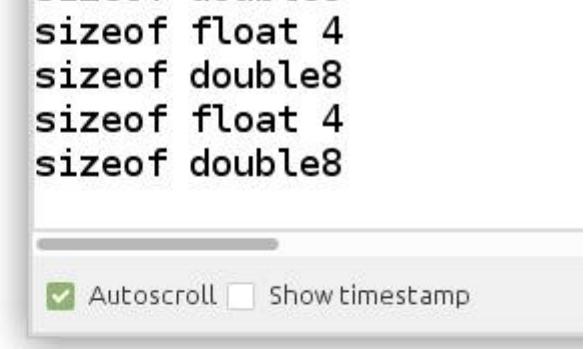
- Et heltal har max positiv og max negativ værdier den kan holde
- Arduino
 - `int`: fylder 16 bit - dvs der er 2^{16} værdier = 65536 værdier
 - 32768 negative tal: -32768 til -1
 - 0 (tallet 0)
 - 32767 positive tal: 1 til 32767
 - ialt 65536 tal
- ESP32
 - `int`: fylder 32 bit dvs der er 2^{32} værdier = 4294967296 værdier (ca 4 mia)
 - -2147483648 negative tal: -2147483648 til -1
 - 0 (tallet 0)
 - 2147483647 positive tal: 1 til 2147483647
 - ialt 4294967296 tal
- 64 bit
 - $2^{64} \approx 1,8 \cdot 10^{19}$ eller ca $\pm 9 \cdot 10^{18}$
 - puhaa mange tal
- *Jordens befolkning er lige nu ca 7800000000 (7,8 mia)*
- *Så den kan representeres vha int 16 ? eller 32 ? eller 64 ?*



kommatal - float

- Man kan regne med kommatal med + - * /

```
float etKommaTal;  
  
etKommaTal = 1.24;  
// NB “.” som decimaloperator  
  
etKommaTal = 2* 3.4;  
  
etKommaTal = - 1.4e17;
```



The screenshot shows the Arduino IDE with the following code:

```
1  sizeof float 4  
2  sizeof double8  
3  sizeof float 4  
4  sizeof double8
```

The serial monitor window displays the output of the code above, showing the sizes of float and double variables as 4 and 8 bytes respectively. The checkbox "Autoscroll" is checked.

```
1  float kommaTal;  
2  
3  void setup()  
4  {  
5    Serial.begin(115200);  
6    delay(1000);  
7  }  
8  
9  void loop() {  
10  
11    Serial.print("sizeof float ");  
12    Serial.println(sizeof(float));  
13  
14    Serial.print("sizeof double");  
15    Serial.println(sizeof(double));  
16  
17    delay(2000);  
18 }
```

- Er 32 bit float ikke nok
- er der 64 bit double



enkle typer

Type & Description
char Typically a single octet(one byte). It is an integer type.
int The most natural size of integer for the machine.
float A single-precision floating point value.
double A double-precision floating point value.
void Represents the absence of type.



Analog input

- Kan måle spændinger 0-5V som udgangspunkt
- 10 bit ADC (analogtilDigitalConverter) dvs $2^{10} = 1024$ niveauer
- $0 \rightarrow 5.0V \Rightarrow 0 \rightarrow 1023$

```
void setup()
{
    Serial.begin(9600);
}

void loop()
{
int i;
float voltage;
i = analogRead(A0);  (UNO A0,A1,... A5 - du kan også bare skrive 0,1,...)

voltage = 5.0*i/1023;

Serial.println(voltage);

delay(100); // sov 100 millisekunder
}
```



rdy - go

- åben serial terminal
- luk serial terminal
- åben serial plotter



BREAK



Digital input/output

- Se arduino comic
-



BMP085 / BMP180

- Trykmåler
- +- 30cm nøjagtige
 - (omregnet til luftryk/højde)
- I2C interface s

```
1 #include <Wire.h>
2 #include <bmp085.h>
3
4 /*
5 1. call bmp085_init
6   - parameter pressure at sealevel (in Pascal) or 0
7   if 0 then 101325 (std pressure at sea level) is used
8
9 2. Measurement by
10  a. call bmp085_measure()
11  b. now pressure and temperature is in float variables bmp085_temp and br
12 */
13
14
15 void setup()
16 {
17   Serial.begin(9600);
18   Wire.begin();
19
20   bmp085Init(101300.0); // ress at sealevel today
21 }
22
23 float temp, atm, alt, pres;
24
25
26 #define LOOPTIME 200
27 unsigned long t1, t3;
28
29 void loop()
30 {
31   float tryk, temp, højde;
32   bmp085Measure(&temp, &tryk, &højde);
33
34   Serial.print("temp(C) "); Serial.print(temp);
35   Serial.print("\t tryk(Pa) "); Serial.print(tryk);
36   Serial.print("\t højde(m)"); Serial.println(højde);
37   delay(100);
38 }
```



```
1 #include <Wire.h>
2 #include <bmp085.h>
3
4
5 void setup()
6 {
7     Serial.begin(9600);
8     Wire.begin();
9
10    bmp085Init(101300.0); // ress at sealevel today
11 }
12
13 float temp, atm, alt, pres;
14
15 void loop()
16 {
17     float tryk, temp, højde;
18     bmp085Measure(&temp, &tryk, &højde);
19
20     Serial.print("temp(C) "); Serial.print(temp);
21     Serial.print("\t tryk(Pa) "); Serial.print(tryk);
22     Serial.print("\t højde(m)"); Serial.println(højde);
23     delay(100);
24 }
```



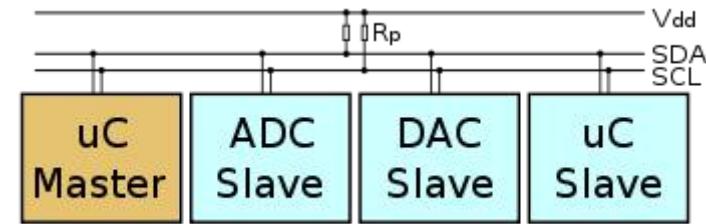
så var der lige Openlogger

- Logger alt hvad der bliver skrevet hvis
 - openlog er konfigureret til rette hastighed (med tomt sd-kort kører den 9600)
 - Jumper J4 er sat på
- hvad er dagens lufttryk i Bjerringbro ?
- Ret i koden i bmp08test
- compiler og upload kode
- Sæt jumper på J4
- Sæt SDcard i Openlogger
- Sæt batteri til
- gå en tur ned ad trappen og op igen
- fjern batteri
- Hvad er der på sd kort ?



MPU6050

- 3 akses acceleration og gyro
- I2C interface
-
-





- example: accgyrotes in MPU6050
- to code

```
5
4 #include "MPU6050.h"
5 #include "Wire.h"
6 MPU6050 accelgyro(0x68);
7
8 int16_t gx, gy, gz;
9 int16_t ax, ay, az;
10
11 #define LED_PIN 13
12 bool blinkState = false;
13
14 void setup() {
15   Serial.begin(38400); // start serial port
16
17   Wire.begin(); // start i2c bnetwork
18
19   accelgyro.initialize(); // inititalize mpu6050
20
21   if (accelgyro.testConnection()) {
22     Serial.println("MPU6050 ok ");
23   }
24   else {
25     Serial.println("MPU6050 fail ");
26     while (1); // stop here
27   }
28
29   pinMode(LED_PIN, OUTPUT);
30 }
31
32 void loop() {
33
34   // read raw accel/gyro measurements from device
35   accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
36   Serial.print("a/g:\t");
37   Serial.print(ax); Serial.print("\t");
38   Serial.print(ay); Serial.print("\t");
39   Serial.print(az); Serial.print("\t");
40   Serial.print(gx); Serial.print("\t");
41   Serial.print(gy); Serial.print("\t");
42   Serial.println(gz);
43
44   // blink LED to indicate activity
45   blinkState = !blinkState;
46   digitalWrite(LED_PIN, blinkState);
47 }
48
```



HMC5983

- 3 akses magnetometer
- Husk feltet over Danmark peger ned med en vinkel på ca 60 grader

```
#include <HMC5983.h>
#include <Wire.h>

HMC5983 compass;

void setup () {
    Serial.begin(9600);
    Serial.println(" heading / x y z kompasanter");
    delay(3000);
    compass.begin(); // use "true" if you need some debug information
}

void loop() {
    float x, y, z;
    float c = -999;
    c = compass.readAll(&x, &y, &z);
    if (c == -999) {
        Serial.print("Reading error, discarded");
        return;
    } else {
        Serial.print(c);
    }
    Serial.print(" ");
    Serial.print(x); Serial.print(" ");
    Serial.print(y); Serial.print(" ");
    Serial.println(z);
}
```



HMC5983

- output x,y,z -2048 - 2047
- et count == 0.92 mG
- range default is +- 1.3 Gauss
- earth magn field 0.25 - 0.65 Gauss



BREAK



RADIO - APC220

- transmitterer alt hvad der skrives mpå serielle port
- kan også modtage kommunikation den anden vej
- J2 hvis du vil sende data hjem
- J3 hvis du vil modtage data hjemmefra
- kommunikations til Arduino 1200 og op til 57600 baud
- kommunikation i luften 2400 og op til 19200
 - højere hastighed i luften == kortere rækkevidde
- ANBEFALING
- I luft max 9600 - gerne 4800 hvis databudget er til det
- Internt max 19200
- Frekvenser 433,05 MHz – 434,79 MHz (center 433,92 MHz) (stærkt benyttet)
- Radio kan 418-455 MHz - beware !!!
- Kan/er (lidt) anderledes i udlandet.



endelig

- **velkommen**
- **udlevering af kit**
 - install af SW
 - arduino IDE
- **gennemgang af cansat shield**
 - bibliotekter for cansat shield
- **Afdækning af folks programmeringserfaringer**
 - gætter på fra ingenting til meget kompetente
- **enkle hello world programmer**
- **start på kit**
- **trykmåler (bmp085/bmp180/...)**
- **accelerometer (MPU6050)**
- **gyro (MPU6050)**
- **magnetometer (hmc5983)**
- **logning på sd card**
- **radiolink**
- **apc220**
- **configuration**
- **den store test**
- **pewwww**