```
/* Filnavnet er: Band_gap_volt */

/*  Version: 1.0*/
/* Færdiggjort den 8.juli 2013 */

/* Dette program omdanner vores Arduino til et voltmeter, som måler
 den DC spænding der findes på chippens interne Band Gap Reference*/

/* Vi skal bruge mange konverteringer i en fart, så jeg bruger chip programmering,
 det vil sige jeg går direkte ind i A/D konverterens registre,
 og bruger derfor ikke højniveau statementet analogRead.  */
/* Justering/kablibrering::
 Til denne måling bruger jeg +5 volt som reference til A/D konverteren.
 Du tager derfor dit bedste voltmeter og måler din 5 volt.
 Husk at måle på AREF pinnen på din Arduino
 En afvigelse fra nominelle 5 volt kan du så selv korrigere den værdi af 1,1 volten på Ban Gap du
aflæser med softwaren */
/* Husk A/D konverterens særhed: smid de første målinger væk */
/* Hvis du vil sløve systemet, og se de rå samples i venstre side af displayet,
 så ændrer du parameteren i delay umiddelbart i starten af loop.
 Prøv med 10000 ! */


/* Display forbindelser
 pin 11 LCD pin D4 til Arduino digital pin 5
 pin 12 LCD pin D5 til Arduino digital pin 4
 pin 13 LCD pin D6 til Arduino digital pin 3
 pin 14 LCD pin D7 til Arduino digital pin 2
 pin 3 LCD til 10 kilo potmeter mellem + 5V og stel, de kræver de nyere display.
 pin 4  LCD pin RS til Arduino digital pin 12
 pin 6 LCD Enable pin til Arduino digital pin 11
 pin 5 LCD R/W pin til stel    */

/* Backlight kræver 150 mA ved 5 V, som stabilisatoren på
 Arduino Uno R3 IKKE kan bære. Den er en SMD kreds.
 Derfor separat 5 volt stabilisator 7805 med køleplade
 fødet direkte af V(in) på Arduino
 V(in) er den rå spænding fra net tilslutningen, IKKE fra USB
 Som Vin bruges en af de sorte net forsyninger,
 plus i midten
 en 6 Vdc er nok., en 12 volt giver varm regulator
 */

#define bufferantal 256
#define karaktererperlinie 16
#define antallinier 2

double resultat[bufferantal];
boolean friskedata;
double dethele;
int AntalSamples;
```

```
/* Vi referer til biblioteker */
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);   // Arduino's

void setup()
{
  lcd.begin (karaktererperlinie,antallinier);
  lcd.setCursor(0,0);  // øverste linie, første position
  lcd.print ("BandGap volt 1.0");  // skriv på øverste linie
  lcd.setCursor(0,1);  // første position, 2. linie
  lcd.print ("                "); // skriv på nederste linie

  /* Set prescaler til 128, ADC Clock = 125 kHz */
  ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);

  /* Set referencespænding til ADC til at være ekstern 5 volt,
   Det svarer til 5.0 / 1024 = 4.8828125 mV per bit.
   Sæt MUX0 til 0 og MUX1, MUX2 og MUX3 til 1, derved vælges Band Gap til ADC inout.
   og ADLAR til 0, ADLAR sand betyder ADC data baglæns  */
  ADMUX |= B01001110;

  /* Enble AD */
  ADCSRA |= (1<<ADEN);

  /* Tillad ADC interrupt */
  ADCSRA |= (1<<ADIE);

  /* Enable globale interrupt */
  sei();

  /* Spark den første konvertering i gang, resten følger af sig selv */
  ADCSRA |= (1<<ADSC);

  /* det var så setup, nu cycler vi derudad */
  friskedata = false;
  AntalSamples = 0;
}

void loop ()
{
  while(bit_is_set(ADCSRA,ADSC));
  delay(1); /* Sus i skørterne ? */

  /* Vi kommer her hele tiden,
   men hvis der har været ADC interrupt siden sidst vi var her,
   så er friske data jo sand = true
   I så tilfælde skal der arbejdes ! */
  if (friskedata == true)
    bump(); /* Hent og parker nyt sample */

  /* Denne tekst kommer så tit, at det er meget svært at se på displayet når det er nye data
```

```
   Hvis du vil følge og læse dine data for hver konvertering kan du neddrosle
   systemet ved at øge parameteren til "delay" umiddelbart indenfor start af "loop"*/
  lcd.setCursor(0,1);  // nederste linie, første position
  lcd.print (resultat[0],2);  // skriv på nederste linie i venstre side
  lcd.print (" ");

  /* nu skal min CPU så sandelig på rigtigt computerarbejde */
  if (AntalSamples >= bufferantal )
    beregnsnit();

  /* Sæt konvertering i gang igen */
  ADCSRA |= (1<<ADSC);
}

/*****************************************************************
 ** Dette er en interrupt vektor for ADC interrupt, husk din compiler hjælper dig
 ** ISR = Interrupt Service Routine
 *****************************************************************/
ISR(ADC_vect)
{
  friskedata = true;
}

/*****************************************************************
 ** Skub gamle samples en position opad og hæld den nye sample ind i position 0
 *****************************************************************/
void bump()
{
  friskedata = false;
  AntalSamples = AntalSamples + 1;

  /* Skub alle data i "resultat" eeen hel position opad for at lave resultat om til en FIFO buffer
   den der ryger ud over kanten foroven er jo den første sample */
  for (int x = 0; x < bufferantal; x++)
    resultat[x+1] = resultat[x];
  resultat[0] = (4.8828125)*ADC;
  /* Hvert bit har værdien 4.8828125, som er 5000.0 / 1024.0 */
}

/*****************************************************************
 ** Rigtigt computerarbejde, beregn gennemsnit af dine ”bufferantal” samples.
 *****************************************************************/
void beregnsnit()
{
  AntalSamples = 0;
  dethele = 0;
  for (int x = 0;x < bufferantal;x++)
    dethele = dethele + resultat[x];
  dethele = dethele/ bufferantal;
  lcd.print (dethele,2); /* Skriver på nederste linie yderst til højre */
}
/* Slut på endnu et Arduino prgram */
```