```
/* Filname is:  16 bit proof version 3 */
/*  Version: V 3.0  */
/* programmer: OZ7EC */
/* This program shows, by a fair distribution of the samples,
 that the actual connected Arduino har enough noise to qualify
 delivering 16 bit Arduino resultion */
/* If the samples a doesn NOT hold a fair destibution in the 7 classes,
 additional noise to get the 10 but LSB to flicker must be added
 artifically to the input voltage  */

/* Display connections:
 pin 11 LCD pin D4 to Arduino digital pin 5
 pin 12 LCD pin D5 to Arduino digital pin 4
 pin 13 LCD pin D6 to Arduino digital pin 3
 pin 14 LCD pin D7 to Arduino digital pin 2
 pin 3 LCD to kontrast potentiometer
 pin 4 LCD pin RS to Arduino digital pin 12
 pin 6 LCD Enable pin to Arduino digital pin 11
 pin 5 LCD R/W pin to ground    */

/* Backlight needs 150 mA at 5 V.
 Arduino Uno R3 CANNOT deliver this.
 Use an external 7805 med køleplade
 */

#define chars_per_line 16
#define count_of_lines 2

/* You can measure the voltage og the internal Band Gap reference voltage
 ( use; ADC reference) by running the program Band Gab Reference, OZ 2013 /12 */
```

```
/* This program uses ADC interrupt
 ADC clock runs at 500 kHz
 without loosing bits (ENOB, Effective Nuber Of Bits */

/* Note: we are handling 917504 samples in total to get the final ressults */

/* this program runs JUST ONCE,
 hence is the function LOOP empty */

boolean fresh_ADC_data;
int inputpin = 0;
double millivolt;
float sum_of_samples;
int count_of_samples = 4096;

int x;
int outer_loop_control;
int y;
int z;

int number_of_words = 224;
double array_of_words[224];
int point_at_upper;

boolean up;
boolean sundown;
boolean sunup;
double interrim;
double lowest_val;
double highest_val;
```

```
double first;
double second;
double third;
double fourth;
double fifth;
double sexth;
double seventh;

float ADC_voltage_reference = 1.074218 ; // Measured on my own Arduiono, type in yours please

/* libraries in use */
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);
#include <interrupt.h >

void setup()
{
  lcd.begin (chars_per_line,count_of_lines);
  lcd.setCursor(0,0);  // upper line, first position
  lcd.print (" Bevis16bit 3.0");
  lcd.setCursor(0,1);  // lower line, first position
  lcd.print ("              ");

  pinMode(inputpin,INPUT); // Tell ADC where to pick analog input
  /* Set prescaler to 32, ADC Clock = 500 kHz */
  ADCSRA |= (1<<ADPS2)|(0<<ADPS1)|(1<<ADPS0);
  /* Set reference voltage to ADC to be Band Gap internal 1,1 volt,
   1.1 / 1024 = 1,074218 mV per bit */
  ADMUX |= B11000000;
  /* Clear ADLAR, ADCH og ADCL in opposite order */
```

```
ADMUX &= B11011111;
/* Input pin er som defaultt A0, just set it anyhow */
ADMUX &= B11110000;
/* Enble AD */
ADCSRA |= (1<<ADEN);
/* Tillad ADC interrupt */
ADCSRA |= (1<<ADIE);
/* Enable globale interrupt */
sei();
/* Kick first conversion off */
ADCSRA |= (1<<ADSC);
/* Now setup usually finishes and LOOP takes over
 BUT we only need one single tour through the program,
 so all  programming continous in SETUP */

Serial.begin(9600);
/* Press: TOOLS and press: Serial monitoring
 when the progam starts runnung.
 You will the see the final 7 results on your screen */

/**********************************************************************
 * Initialise array of words
 **********************************************************************/
x = 0 ;
while (x < number_of_words )
{
  array_of_words[x] = 0.0;
  x++;
}
```

```
/*********************************************************************
 * We are running the outer loop "number_of_words" times
 *********************************************************************/
outer_loop_control = 0 ;
while (outer_loop_control < number_of_words )
 /* we pick 224 dfferent volts */
{

 /*********************************************************************
  * We are running the Inner loop "count_of_samples" times
  *********************************************************************/
 sum_of_samples = 0 ;
 int i = 0;
 while (i < count_of_samples )
 {
  fresh_ADC_data = false;
  sum_of_samples = ( sum_of_samples + ADC);
  ADCSRA |= (1<<ADSC);  /* sæt konvertering i gang igen */
  i++;
 }
 /* Now we have some "count_of_samples" samples */
 /*  Time to decimate by right shifting 6 times = division by 64 */
 sum_of_samples = sum_of_samples / 64.0; /* decimation */

 millivolt = ( sum_of_samples * ADC_voltage_reference ) / 65.536 ;

 array_of_words[outer_loop_control] = millivolt;

 /* The foolowing is just to tell You that the program is progressing !!!*/
 lcd.setCursor(0,1);  // første position, 2. linie
 lcd.print (outer_loop_control);  // skriv på nederste linie
```

```
  lcd.print ("  ");  // skriv på nederste linie
  lcd.print (millivolt,5);  // skriv på nederste linie

  outer_loop_control++;
}
/*************************************************************************
 * End of outer loop
 *************************************************************************/



/*************************************************************
 * Here starts "Buppel", which puts the values in "array_of_words" in rising order,
 * starting in positio 0 (ZERO)
 *************************************************************/

point_at_upper = 1;
up = true;
sundown = false;

while ( sundown == false )
{
  highest_val = array_of_words[point_at_upper];
  lowest_val = array_of_words[point_at_upper - 1];

  if ((highest_val > lowest_val) or (highest_val == lowest_val))
  {
    up = true;
    point_at_upper = point_at_upper + 1;

    /*****************************************************
```

```
   * if you want to watch the sort mechanism working:
   * Uncomment this, but it slows a lot
   * Serial.print("Up");
   *
   * Serial.print("  ");
   * Serial.println(point_at_upper);
   *
   * lcd.setCursor(0,1);  // første position, 2. linie
   * lcd.print ("       "); // skriv på nederste linie
   * lcd.print (x);
   ********************************************************/
}
else
{
  /*********************************************************
   * Swap and step one position down
   *********************************************************/
  interrim = array_of_words[point_at_upper - 1];
  array_of_words[point_at_upper - 1] = array_of_words[point_at_upper];
  array_of_words[point_at_upper] = interrim;

  /*********************************************************
   * display sort mechanism
   * Serial.print("Ned");
   * Serial.print("  ");
   * Serial.println(point_at_upper);
   *********************************************************/
  up = false;
  point_at_upper = point_at_upper - 1;
}
if (point_at_upper > 223 and up == true)
```

```
   sundown = true;
}

/*************************************************************
 * end of "Buppel".
 *************************************************************/

/*************************************************************
 * Put all "array_of_words" values out on serial.
 *************************************************************/
y = 0;
while (y < number_of_words)
{
  Serial.print("  ");
  Serial.print(y);
  Serial.print("  ");
  Serial.println(array_of_words[y],5);
  y++;
}
/*************************************************************
 * Job done
 *************************************************************/

/*************************************************************
 * add 7 sets of 32 values
 *************************************************************/
/*
 0-31
 32-63
 64-95
 96-127
```

```
 128-159

 160-191

 192-223


 Add up these sets of 32 values to facilitate

 a way of watching "Normal Distribution"


 96-128 is the middle one

 alle values shoul be just below 1000 millivolt*/


sunup = false;

while (sunup == false)

{

  first = 0.0;

  z = 0;

  while (z < 32 )

  {

    first = first + array_of_words[z];

    z++;

  }

  first = first / 32.0;


  second = 0.0;

  z = 32;

  while (z < 64 )

  {

    second = second + array_of_words[z];

    z++;

  }

  second = second / 32.0;
```

```
third = 0.0;
z = 64;
while (z < 96 )
{
 third = third + array_of_words[z];
 z++;
}
third = third / 32.0;

fourth = 0.0;
z = 96;
while (z < 128 )
{
 fourth = fourth + array_of_words [z];
 z++;
}
fourth = fourth / 32.0;

fifth = 0.0;
z = 128;
while (z < 160 )
{
 fifth = fifth + array_of_words[z];
 z++;
}
fifth = fifth / 32.0;



sexth = 0.0;
z = 160;
```

```
while (z < 192 )
{
  sexth = sexth + array_of_words[z];
  z++;
}
sexth = sexth / 32.0;



seventh = 0.0;
z = 192;
while (z < 224 )
{
  seventh = seventh + array_of_words[z];
  z++;
}
seventh = seventh / 32.0;

/*************************************************************
 * additions complete
 ************************************************************/



/*************************************************************
 * Show user the results on Serial
 ************************************************************/
Serial.println("  ");
Serial.println("  ");
Serial.println("  ");
Serial.println(first,5);
Serial.println(second,5);
Serial.println(third,5);
```

```
    Serial.println(fourth,5);
    Serial.println(fifth,5);
    Serial.println(sexth,5);
    Serial.println(seventh,5);

    sunup = true;
  }

} /* This is the endof programming, sorry folks */



void loop()
{
}
/* The most lazy LOOP function I ever wrote */



ISR(ADC_vect)
{
  fresh_ADC_data = true;
}
/* What a workhorse !!*/
```