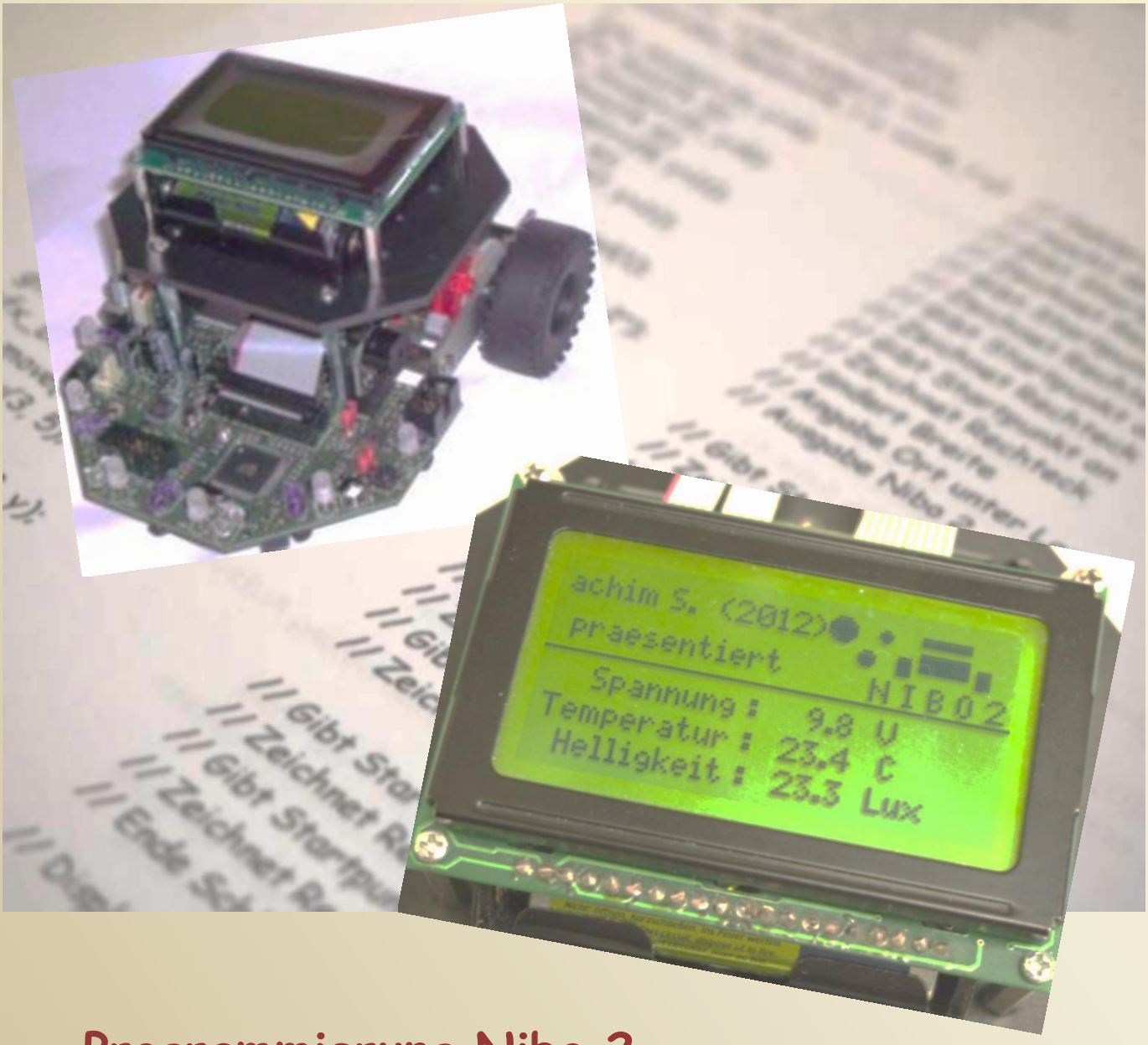
# NIBO 2*

# PROGRAMMIERUNG




## Programmierung Nibo 2
## Teil 3 – Display

* by nicai-systems

# Nibo 2

# Programmierung Teil 3 – Display

Notwendige Programme:

- AVR Studio 4 ( mit den aktuellen Versionen und Updates )
- WinAVR ( in der aktuellen Version )
- Nibo Library ( in der aktuellen Version )

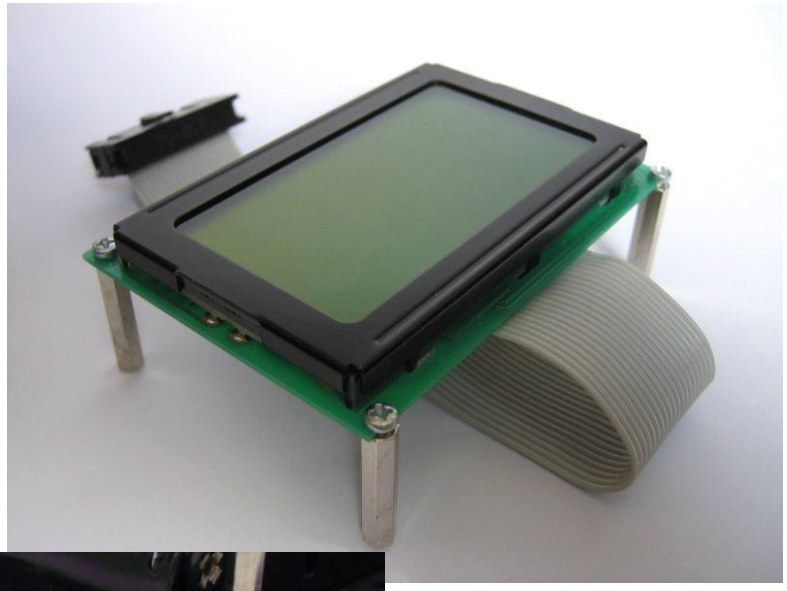
Bitte diese Programme nach Anweisung des Herstellers oder des Tutorials von Nicai, installieren.
Ich arbeite mit Windows 7 und den angegebenen Programmen.
Es können auch andere Programme, z.B. Linux, verwendet werden.
Vom AVR Studio ( 5,6 ) sind in der Zwischenzeit neuere Versionen von Atmel veröffentlicht worden. Diese sind teilweise sehr viel grösser und dadurch auch in der Funktion anders. Welches Programm ihr nutzt, müsst ihr selber entscheiden. Alle Fotos bei hjs und nicai.
Die Beispielprogramme findet ihr bei Roboter.cc, unter Öffentliche Projekte. Alle Programme habe ich auf Funktion getestet.

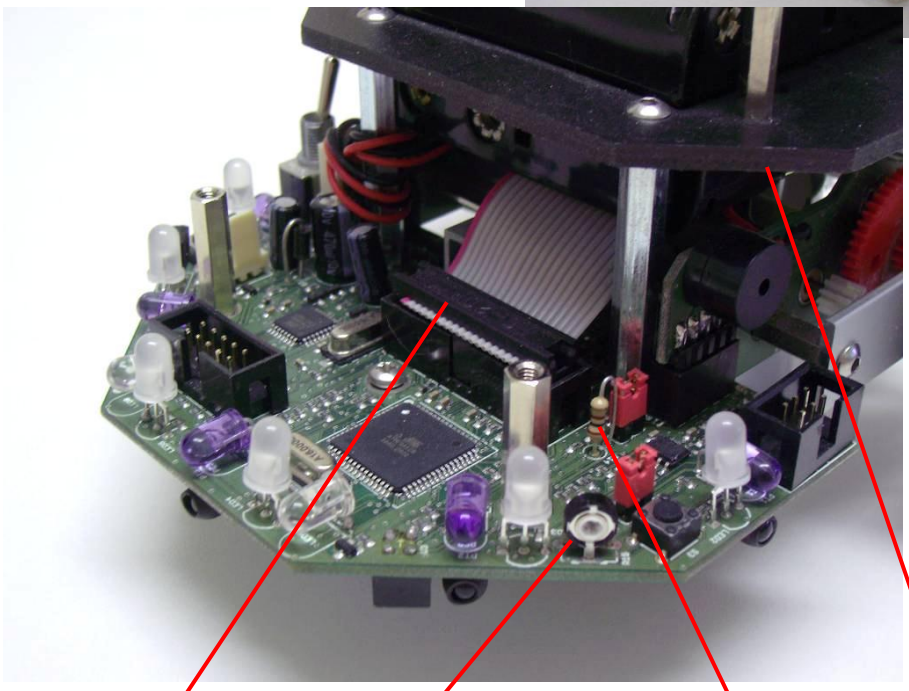
**by H.J.Seeger**

## 1. Graphikdisplay GFX

( Bild Nicai )

Der Nibo 2 kann mit einem Grafikdisplay erweitert werden. Dies hat eine Auslösung von 64 x 128 Pixel. Zur Darstellung der Zeichen stellt die Bibliothek des Nibo Zeichen in einer 5 * 8 Pixelmatrix bereit (siehe gfx.c). Auf diese Weise können bei einem Pixel Zeichenabstand maximal acht Textzeilen mit jeweils 21 Zeichen ausgegeben werden.






Der Anschluss des Displays erfolgt durch einen 20 - poligen Wannenstecker mit einem Flachkabel zum X9 auf der Platine des Nibo 2.
Auf dem oberen Bild seht ihr das Display. Auf dem unteren Bild seht ihr die Anschlüsse und Regler zum Display.
( Angabe Nicai )

## 2. Möglichkeiten

Welche Möglichkeiten habe ich und was kann ich darstellen?


( Darstellung Schrift – oben )

( Nase ) ( Biene -> )

Das fängt beim Zeichensatz an, geht über einfache Striche bis zur Darstellung von Quadraten und Bildern. Mit den Bildern wollen wir uns in diesem Teil nicht beschäftigen, das kommt später.
Es gibt auch die Möglichkeit, den Zeichensatz bzw. einige Buchstaben daraus gedreht darzustellen. Und noch einiges anderes …

## 3. Programme

Womit fängt man an? Bei C ist es fast schon Pflicht, natürlich mit **Hello World**.

**PRG_GFX_1 – bringt den Text „Hello World" auf das Display**

```
#include <nibo/niboconfig.h>          // Laden der Header Dateien
#include <nibo/display.h>             // stammt von nicai
#include <nibo/gfx.h>
int main() {                          // Beginn Programm
  display_init();                     // initiiert Display
  gfx_init();                         // initiiert gfx
  gfx_move(30, 20);                   // Angabe Position
  gfx_set_proportional(0);            // Angabe Schriftbreite breit
  gfx_print_text("Hello World");      // Ausgabe Text ( breit )
  gfx_move(30, 30);                   // Angabe Position
  gfx_set_proportional(1);            // Angabe Schriftbreite schmal
  gfx_print_text("Hello World");      // Ausgabe Text ( schmal )
  return 0;                           // Ende Programm
}
```

In diesem Programm wird der Text „Hello World" breit und schmal auf dem Display dargestellt. Habe dazu das Original von nicai genommen. Es hat leider einen kleinen Nachteil. Da das Display nicht beleuchtet ist, kann ich es schlecht erkennen.

**PRG_GFX_2 – bringt den Text „Hello World" auf das Display mit Beleuchtung**

```
#include <nibo/niboconfig.h>              // Laden der Header Dateien
#include <nibo/leds.h>
#include <avr/interrupt.h>
#include <nibo/gfx.h>
#include <nibo/display.h>
#include <nibo/pwm.h>
int main()                                // Start Programm
 {
   sei();                                 // Einbinden der Dateien
   display_init();
   pwm_init();
   gfx_init();
   leds_set_displaylight(1023);           // Setzt Helligkeit Display
   gfx_move(30, 20);                      // Angabe Ort
   gfx_set_proportional(0);               // Angabe Schrift
   gfx_print_text("Hello World");         // Ausgabe Text ( breit )
   gfx_move(30, 30);                      // Angabe Ort
   gfx_set_proportional(1);               // Angabe Schrift
   gfx_print_text("Hello World");         // Angabe Text ( schmal )
   return 0;                              // Ende
 }                                        // Schleife Ende
```

Bei diesem Programm wird wieder der gleiche Text auf dem Display ausgegeben und zusätzlich wird die die Displaybeleuchtung mit der maximalen Helligkeit eingeschaltet.

Als nächste wollen wir uns mal den Zeichensatz ansehen. Dazu habe ich ein kleines Programm geschrieben, das den Zeichensatz auf dem Display darstellt.

**PRG_GFX_3 - Nibo 2 Zeichensatz darstellen by hjs**

```
#include <nibo/niboconfig.h>              // Aufruf der Bibliotheken
#include <nibo/leds.h>
#include <avr/interrupt.h>                // by HJS
#include <nibo/gfx.h>
#include <nibo/display.h>
#include <nibo/pwm.h>
#include <nibo/delay.h>
#include <nibo/iodefs.h>
#include <stdio.h>
int main()                                // Beginn Programm
 {
  sei();                                  // Aufruf sei, als erstes
```

```
  leds_init();                          // Aufruf leds
  pwm_init();                           // Aufruf pwm
  display_init();                       // Aufruf Display
  gfx_init();                           // Aufruf Graphik
  leds_set_displaylight(1024);          // Setzt Displaylicht
   {                                    // Bringt die darstellbaren Zeichen auf den Schirm
    gfx_term_goto(0,0);                 // setzt Anfang oben links
    for (uint8_t i=0x21; i<0x80; ++i)   // Beginn Schleife mit Angabe Zeichen
     {                                  // Beginn Schleife
       char text[2];                    // Variable Text mit min. Grösse 2
       text[0]=i;                       // Wird mit i -> 0x?? gefüllt
       text[1]=0;                       // Ende wird mit 0 gesetzt
       gfx_term_print(text);            // gibt Text aus
       delay_ms(150);                   // Angabe Zeit zwischen Zeichen
     }
    gfx_move(0,42);                     // Angabe Ort
    gfx_hline(128);                     // Zeichnet Linie nach rechts + Länge
    gfx_set_proportional(0);            // ändert Breite
    gfx_move(10, 45);                       // Angabe Ort
    gfx_print_text("Zeichensatz Nibo 2 ");  // Ausgabe Text
    gfx_move(2, 55);                        // Angabe Ort
    gfx_print_text("(leider ohne Umlaute)");// Ausgabe Text
   }
  return 0;                                 // Habe fertig
}
```

Die Zeichen werden in einer Schleife aufgerufen. Diese geht von 0x21 bis 0x80 und dadurch werden die Zeichen nacheinander aufgerufen und durch „**text**" auf dem Display dargestellt. Ihr könnt gern einmal die Angaben bei **gfx_move**(0,42); verändern und überraschen lassen, wo der Strich erscheint und was danach noch alles verändert wird.

Kommen wir als nächstes zu einfachen Formen. Sehr beliebt sind dabei Rechtecke und Striche. Diese lassen sich einfach aufrufen und in der Grösse beliebig verändern. In dem nachfolgenden Programm habe ich ein einfaches Logo vom Nibo 2 einmal durch verschiedene Rechtecke dargestellt. Dabei erfolgt durch die Anweisung **gfx_move** (57, 25); **gfx_box** (14, 3); einmal die Angabe des Startpunkte (57,25) und als zweites die Angabe der Grösse und Form (14,3) des Rechteckes.

**PRG_GFX_4 - Logo einfach by hjs**

```
#include <nibo/niboconfig.h>            // Aufruf der Header
#include <nibo/leds.h>
#include <avr/interrupt.h>              // by HJS
#include <nibo/gfx.h>
#include <nibo/display.h>
#include <nibo/pwm.h>
#include <stdio.h>
```

```
int main()                                  // Beginn Hauptprogramm
 {
  sei();                                    // Aufruf sei, als erstes
  leds_init();                              // Aufruf leds
  pwm_init();                               // Aufruf pwm
  display_init();                           // Aufruf Display
  gfx_init();                               // Aufruf Graphik
  leds_set_displaylight(1023);              // Setzt Displaylicht
   {                                        // Beginn Programm
    gfx_move(57, 25); gfx_box(14, 3);       // Angabe Ort, Zeichnet Rechteck
     gfx_move(57, 30); gfx_box(14, 6);      // Angabe Ort, Zeichnet Rechteck
     gfx_move(51, 32); gfx_box(4, 6);       // Angabe Ort, Zeichnet Rechteck
     gfx_move(73, 32); gfx_box(4, 6);       // Angabe Ort, Zeichnet Rechteck
     gfx_set_proportional(1);               // Angabe Breite
     gfx_move(46,40);                       // Angabe Ort
     gfx_print_text("N I B O 2");           // Ausgabe Nibo 2
     gfx_move(45,49);                       // Angabe Ort
     gfx_hline(40);                         // Strich mit Länge
   }
  return 0;                                 // Habe fertig
}
```

In diesem Programm sind alle Orte absolut angegeben. Das bedeutet, das für die Angabe der Ort (Startpunkte) jedes Mal eine Zahl verwendet wird. Das ist natürlich auch anders möglich.

Im nächsten Programm wollen wir das Nibo-Logo einmal anders darstellen. Die Sprache C lässt den Aufruf von Unterprogrammen zu. Das kann man durch die Anweisung **void** machen. Beachtet bitte dabei, dass keine Rückgabe von Werten möglich ist. (Normalfall) Beim Aufruf können auch verschiedene Parameter mit übergeben werden. Im folgenden Programm habe ich das bekannte Nibo-Logo noch mal um die 3 Punkte erweitert und in ein Unterprogramm geschrieben. Der Aufruf des Unterprogramms erfolgt dabei durch **nibo_logo(x,y);** Durch **nibo_logo** erfolgt der Aufruf des Unterprogrammes **void nibo_logo(int x,int y)** und die Übergabe der Ausgangsposition. Durch eine Veränderung dieser Werte, kann das Logo an jeder Stelle des Displays dargestellt werden. Es ist auch ein mehrmaliger Aufruf aus dem Programm mit unterschiedlichen Werten möglich. Dabei sind **x** und **y** die minimalen und maximalen Positionen, die auf dem Display möglich sind. Zur Sicherheit habe ich diese Werte angegeben und zusätzlich erfolgt durch **if (x>70)** und folgende Zeilen, noch mal eine Kontrolle der Werte und Begrenzung.

**PRG_GFX_5 - Darstellung Logo mit Unterprogramm - by hjs**

```
#include <nibo/niboconfig.h>                // Aufruf der Bibliotheken
#include <nibo/leds.h>
#include <avr/interrupt.h>                  // by HJS
#include <nibo/gfx.h>
#include <nibo/display.h>
#include <nibo/pwm.h>
#include <nibo/iodefs.h>
#include <stdio.h>
```

```
// Eingabe der Position
int8_t x=40;                                  // waag. min=0, max=70
int8_t y=20;                                  // senk. min=0, max=35

// ***   Unterprogramm LOGO NIBO 2  ***
void nibo_logo(int x,int y)                   // nibo_logo mit x + y
 {                                            // macht das Logo von Nicai
   gfx_move(x+31, y+4); gfx_box(14, 3);       // Zeichnet Rechteck
   gfx_move(x+31, y+9); gfx_box(14, 6);       // Zeichnet Rechteck
   gfx_move(x+25, y+11); gfx_box(4, 6);       // Zeichnet Rechteck
   gfx_move(x+47, y+11); gfx_box(4, 6);       // Zeichnet Rechteck
   gfx_set_proportional(1);                   // ändert Breite
   gfx_move(x+19,y+19);                       // Angabe Ort unter Logo
   gfx_print_text("N I B O 2");               // Ausgabe Nibo 2
// Punkt 1
   gfx_move(x+6, y+3); gfx_box(6, 6);         // Zeichnet Rechteck
   gfx_move(x+7, y+2); gfx_box(4, 8);         // Zeichnet Rechteck
   gfx_move(x+5, y+4); gfx_box(8, 4);         // Zeichnet Rechteck
// Punkt 2
   gfx_move(x+14, y+12); gfx_box(5, 3);       // Zeichnet Rechteck
   gfx_move(x+15, y+11); gfx_box(3, 5);       // Zeichnet Rechteck
// Punkt 3
   gfx_move(x+19, y+5); gfx_box(4, 2);        // Zeichnet Rechteck
   gfx_move(x+20, y+4); gfx_box(2, 4);        // Zeichnet Rechteck
 }                                            // Ende Schleife
int main()                                    // Beginn Hauptprogramm
 {
   sei();                                     // Aufruf sei
   leds_init();                               // Aufruf leds
   pwm_init();                                // Aufruf pwm
   display_init();                            // Aufruf Display
   gfx_init();                                // Aufruf Graphik
   leds_set_displaylight(1023);               // Setzt Displaylicht
   if (x>70)                                  // Abfrage x
    {
      x=70;                                   // Begrenzung x
    }
   if (y>35)                                  // Abfrage y
    {
      y=35;                                   // Begrenzung y
    }
  nibo_logo(x,y);                             // Aufruf Unterprg. ab Position
 return 0;                                    // Habe fertig
}
```

Durch den Aufruf als Unterprogramm kann ich es als Baustein in jedem anderem Programm verwenden.

Kommen wir zu einer weiteren Möglichkeit zur Darstellung von verschiedenen Strichen und Figuren. Ich habe es im unteren Programm dargestellt. Dazu verwende ich wieder die Unterprogramme **nibo-Logo** und als weiteres **nibo-Schema.** In dem Unterprogramm nibo-Schema beginne ich an einem Startpunkt und ziehe einen Strich bis zum nächsten angegeben Punkt. Dieser Punkt wird durch eine Linie verbunden. Dabei spielt die Richtung keine Rolle. Der Strich kann gradlinig, hoch, runter oder schräg verlaufen. Bei einem schrägen Verlauf wird die Linie aus kurzen Stücken gradlinigen gebildet. Sieht nicht so schön aus.

**PRG_GFX_6 - Linie nibo by hjs**

```
#include <nibo/niboconfig.h>              // Laden der Bibliotheken
#include <nibo/display.h>
#include <nibo/gfx.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <avr/interrupt.h>

// ***   LOGO NIBO 2  *** //
void nibo_logo(uint8_t x, uint8_t y)      // nibo_logo mit Parameter x + y
 {                                        // macht das Logo von Nicai
   gfx_move(x+31, y+4);gfx_box(14, 3);    // Gibt Startpunkt an
   gfx_move(x+31, y+9);gfx_box(14, 6);    // Gibt Startpunkt an
   gfx_move(x+25, y+11);gfx_box(4, 6);    // Gibt Startpunkt an
   gfx_move(x+47, y+11);gfx_box(4, 6);    // Gibt Startpunkt an
   gfx_set_proportional(1);               // ändert Breite
   gfx_move(x+19,y+19);                   // Angabe Ort unter Logo
   gfx_print_text("N I B O 2");           // Ausgabe Nibo 2
// Punkt 1
   gfx_move(x+6, y+3);gfx_box(6, 6);      // Gibt Startpunkt an
   gfx_move(x+7, y+2);gfx_box(4, 8);      // Gibt Startpunkt an
   gfx_move(x+5, y+4);gfx_box(8, 4);      // Gibt Startpunkt an
// Punkt 2
   gfx_move(x+14, y+12);gfx_box(5, 3);    // Gibt Startpunkt an
   gfx_move(x+15, y+11);gfx_box(3, 5);     // Gibt Startpunkt an
// Punkt 3
   gfx_move(x+19, y+5);gfx_box(4, 2);     // Gibt Startpunkt an
   gfx_move(x+20, y+4); gfx_box(2, 4);    // Gibt Startpunkt an
 }                                        // Ende Schleife
void nibo_schema()
 {
   gfx_move (5,5);                        // Startpunkt 1
   gfx_lineTo (5,60);                     // Strich zu
   gfx_lineTo (120,60);                   // Strich zu
   gfx_lineTo (120,5);                    // Strich zu
   gfx_lineTo (5,5);                      // Strich zu Start
   gfx_move (10,10);                      // Startpunkt 2
   gfx_lineTo (10,55);                    // Strich zu
```

```
  gfx_lineTo (115,55);                  // Strich zu
  gfx_lineTo (115,10);                  // Strich zu
  gfx_lineTo (10,10);                   // Strich zu Start
 }
int main()
 {                                      // Beginn Hauptprogramm
  sei();                                // Definiert Bibliotheken
  leds_init();
  pwm_init();
  display_init();
  gfx_init();
  if (display_init()!=0)                // Abfrage Display
   {                                    // Display (init) ist wahr, wenn nicht null
    leds_set_displaylight(50);          // setzt Wert für Displaybeleuchtung auf 50
    if (display_type==2)
     {                                  // Rückgabe nach display init, 2=graphicdisplay
       gfx_init();                      // initialisiert Graphik Display
     }                                  // Ende Klammer
   }                                    // Ende Klammer
  leds_set_displaylight(1023);          // Angabe Helligkeit
  gfx_fill(0x00);                       // löscht Display
  nibo_logo(33,20);                     // Aufruf logo
  nibo_schema();                        // Aufruf Nibo Schema
 }
```

Es gibt natürlich auch die Möglichkeit der Bewegung. In dem nachfolgenden Programm wird das Nibo-Logo zuerst stark vergrössert dargestellt. Danach wird es immer kleiner bis zur richtigen Grösse. Das wird durch die Berechnung in der Formel erreicht. Danke meinem Kollegen aus dem Netz für dieses Programm.

**PRG_GFX_7 - Nibo Logo bewegen**

```
#include <nibo/niboconfig.h>
#include <nibo/display.h>
#include <nibo/gfx.h>
#include <nibo/delay.h>
#include <nibo/iodefs.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <avr/interrupt.h>
void nibo_logo(uint8_t xm, uint8_t y, uint8_t r)
 {
   uint8_t x = xm - 14 * r;                          // Formel Berechnung
   gfx_move(x+6*r, y+0);   gfx_box(14*r,3*r);        // Ausgabe Box
   gfx_move(x+6*r, y+5*r); gfx_box(14*r,6*r);        // Ausgabe Box
   gfx_move(x+0,   y+7*r); gfx_box(4*r, 6*r);        // Ausgabe Box
   gfx_move(x+22*r,y+7*r); gfx_box(4*r, 6*r);        // Ausgabe Box
 }
```

```
int main()
 {
   sei();
   leds_init();
   display_init();
   gfx_init();
   pwm_init();
   leds_set_displaylight(1023);

   while(1==1)
    {
      gfx_fill(0x00);                    // BS löschen
      gfx_set_proportional(1);           // Angabe Schrift
      for(uint8_t n=6; n>0; --n)         // Schleife von 6 zu 0
       {
         gfx_fill(0x00);                 // BS löschen
         nibo_logo(60+n,2+n*2,n);        // Nibo Logo verkleinern
         delay(500);
       }
      gfx_set_proportional(1);           // ändert Breite
      gfx_move(43,19);                   // Angabe Ort unter Logo
      gfx_print_text("N I B O 2");       // Ausgabe Text
      gfx_move(42,28);                   // Angabe Ort
      gfx_hline(40);                     // Zeichne Strich
      delay(3000);                       // Pause
    }
   return 0;
 }
```

Auch für ganz mutige Programmierer habe ich was da. Könnt ja mal das nächste Programm ausprobieren. Damit kann ich ein Smiley erzeugen und nach Wahl auf dem Display ausgeben. Da man die entsprechenden **1** und **0** verändern kann, kann man so ziemlich jedes mögliche Smiley darstellen oder auch ein anderes Bild. Bitte die Längen der einzelnen Zeilen beachten. Am besten, ihr holt euch das Programm direkt von **Roboter.cc**, das gibt die wenigsten Fehler.

**PRG_GFX_8** - **Smiley**

```
// achim S. und BirgerT.
// h.j.seeger@web.de

#include <nibo/niboconfig.h>
#include <nibo/display.h>
#include <nibo/gfx.h>
#include <nibo/iodefs.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <stdio.h>
#include <avr/interrupt.h>
```

```
void nibo_smile1(uint8_t smx, uint8_t smy)
 {
  #define _XCAT(x,a,b,c,d,e,f,g,h) x##a##b##c##d##e##f##g##h
  #define XBM16(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) (uint8_t) _XCAT(0b,i,j,k,l,m,n,o,p),(uint8_t)
_XCAT(0b,a,b,c,d,e,f,g,h)
  #define xbm_smile1_width 16
  #define xbm_smile1_height 16
  static unsigned char xbm_smile1_data[] PROGMEM =
    {
      XBM16(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0),          // Smiley normal
      XBM16(0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0),
      XBM16(0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0),
      XBM16(0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0),
      XBM16(0,1,0,0,1,1,1,0,0,1,1,1,0,0,1,0),
      XBM16(1,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1),
      XBM16(1,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1),
      XBM16(1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1),
      XBM16(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1),
      XBM16(1,0,0,1,0,0,0,0,0,0,0,1,0,0,1),
      XBM16(1,0,0,1,0,0,0,0,0,0,0,1,0,0,1),
      XBM16(0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0),
      XBM16(0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,0),
      XBM16(0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0),
      XBM16(0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0),
      XBM16(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0)
    };
  gfx_move(smx,smy);                               // Grafik Ausgabeposition
  gfx_draw_xbm_P(xbm_smile1_width, xbm_smile1_height, (PGM_P) xbm_smile1_data);
 }
int main()
 {
   sei();
   leds_init();
   display_init();
   gfx_init();
   pwm_init();
   leds_set_displaylight(1024);
   while(1==1)
   nibo_smile1(55,25);                             // Angabe Ort
   return 0;
 }
```

Ich habe wieder einige Sachen farbig markiert, damit man sich ein wenig zurecht findet. Die Grösse des Smileys beträgt 16x16 Pixel. So lange ich diese Grösse einhalte, kann ich auch anderes damit machen, z.B. Bilder, Taster, Schalter, Akkus usw. Mein Dank gilt BirgerT für dieses schöne Programm.

Sehen wir uns zum Ende mal ein bisschen grösseres Programm an. Hierbei werden nacheinander die unterschiedlichen Mode dargestellt. Als Bild verwende ich ein Smiley. Dabei kann man deutlich erkennen was alles möglich ist. Bleibt eigentlich nur noch ein Problem. Es muss alles in ein Programm verpackt werden und der Compiler darf nicht meckern.

**PRG_GFX_9 - Anzeige Draw Mode**
**// by hjs und BirgerT.**

```
#include <nibo/niboconfig.h>
#include <nibo/display.h>
#include <nibo/gfx.h>
#include <nibo/delay.h>
#include <nibo/iodefs.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>

#define _XCAT(x,a,b,c,d,e,f,g,h) x##a##b##c##d##e##f##g##h
#define XBM8(a,b,c,d,e,f,g,h) (uint8_t) _XCAT(0b,h,g,f,e,d,c,b,a)
#define XBM16(a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p) (uint8_t) _XCAT(0b,i,j,k,l,m,n,o,p),(uint8_t)
_XCAT(0b,a,b,c,d,e,f,g,h)
#define xbm_smile1_width 16
#define xbm_smile1_height 16

static unsigned char xbm_smile1_data[] PROGMEM =
 {
   XBM16(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0),
   XBM16(0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0),
   XBM16(0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0),
   XBM16(0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0),
   XBM16(0,1,0,0,1,1,1,0,0,1,1,1,0,0,1,0),
   XBM16(1,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1),
   XBM16(1,0,0,0,1,1,1,0,0,1,1,1,0,0,0,1),
   XBM16(1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1),
   XBM16(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1),
   XBM16(1,0,0,1,0,0,0,0,0,0,0,1,0,0,1),
   XBM16(1,0,0,1,0,0,0,0,0,0,0,1,0,0,1),
   XBM16(0,1,0,0,1,0,0,0,0,0,0,1,0,0,1,0),
   XBM16(0,1,0,0,0,1,1,1,1,1,1,0,0,0,1,0),
   XBM16(0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0),
   XBM16(0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0),
   XBM16(0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0)
 };
```

```
void clear_screen(void)
 {
   gfx_fill(0x00);                                    // Clear Screen
   gfx_move(118,0);
   gfx_box(10,60);
 }

int main()
 {
   sei();
   leds_init();
   display_init();
   gfx_init();
   pwm_init();
   leds_set_displaylight(1024);

   while(1==1)
    {
      gfx_fill(0x00);                                 // Clear Screen
      gfx_set_proportional(1);
      clear_screen();
      uint8_t y = 0;

      // Schleife über alle Modi 0,1,2 und 4,5,6
      for(uint8_t i=0; i<7; i++)
       {                                              // Modus 3 gibt es nicht
        if(i==3)
         {
           i++;                                       // Modus 3 gibt es nicht
           clear_screen();
           y = 0;                                     // wieder von oben beginnen
         }
        gfx_draw_mode(i);                             // Modus setzen
        gfx_move(0,y+4 );                             // Text Ausgabeposition

        switch(i)
         {
           case 0:  gfx_print_text("GFX_DM_JAM1 = 0"); break;
           case 1:  gfx_print_text("GFX_DM_JAM2 = 1"); break;
           case 2:  gfx_print_text("GFX_DM_COMP = 2"); break;
           case 4:  gfx_print_text("GFX_DM_JAM1_INV = 4"); break;
           case 5:  gfx_print_text("GFX_DM_JAM2_INV = 5"); break;
           case 6:  gfx_print_text("GFX_DM_COMP_INV = 6"); break;
         }
```

```
      gfx_move(110,y);                          // Grafik Ausgabeposition
       gfx_draw_xbm_P(xbm_smile1_width, xbm_smile1_height, (PGM_P) xbm_smile1_data);
       gfx_draw_mode(GFX_DM_JAM2);              // Modus setzen "ueberschreiben"
       y += 20;                                 // naechste vertikale Ausgabeposition
       delay(2000);                             // Zeit zum ansehen
      }

    }
  return 0;
}
```

In diesem kleinen Tutorial sind wir nun am Ende. Ich habe jedes Programm mit AVR Studio 4 erstellt und getestet. Hoffe, dass keine Fehler drin sind.
Noch eines zu den Programmen.
Versucht jede Zeile, jeden Punkt, jedes Semikolon zu verstehen. Jedes Zeichen hat seine Bedeutung. Ergründet die Funktion und nutzt es.
Ihr könnt die Programme auf Roboter.cc, unter Öffentliche Projekte laden und ausprobieren.

Viel Spass beim lesen und Programmieren, wünscht

Achim Seeger