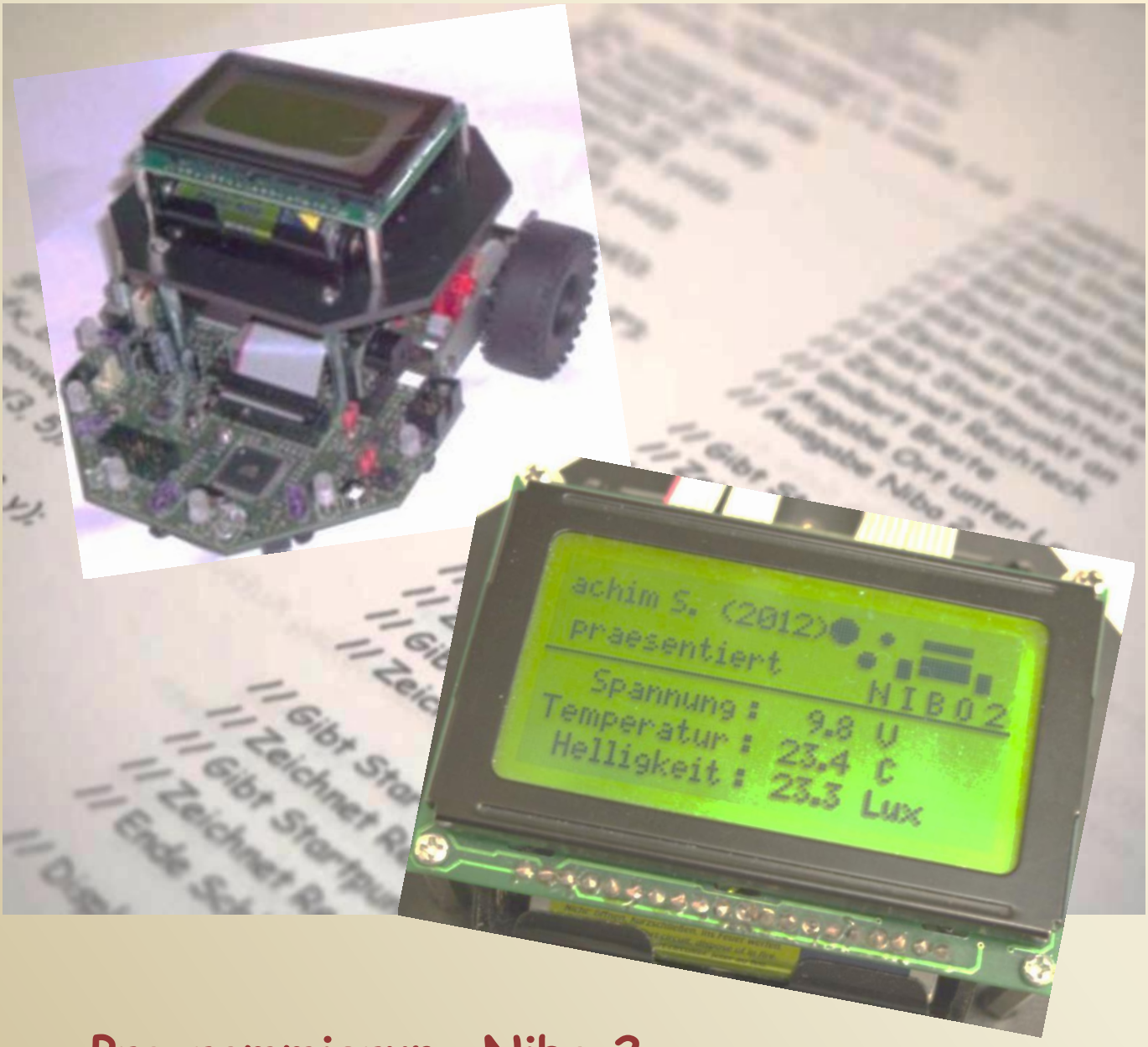
# NIBO 2*

# PROGRAMMIERUNG




## Programmierung Nibo 2
## Teil 2 – LED`s

* by nicai-systems

# Nibo 2

# Programmierung Teil 2 – LED`s

Mögliche Programme:

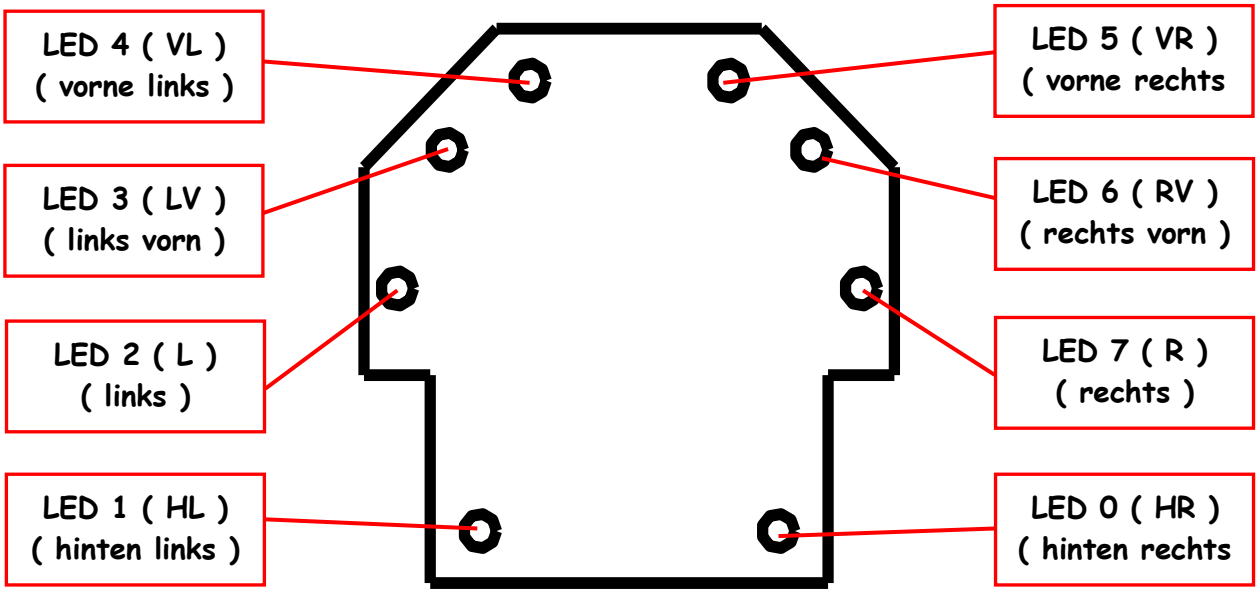
- AVR Studio 4 oder 6 ( mit den aktuellen Versionen )
- WinAVR ( in der aktuellen Version )
- Nibo Library ( in der aktuellen Version )

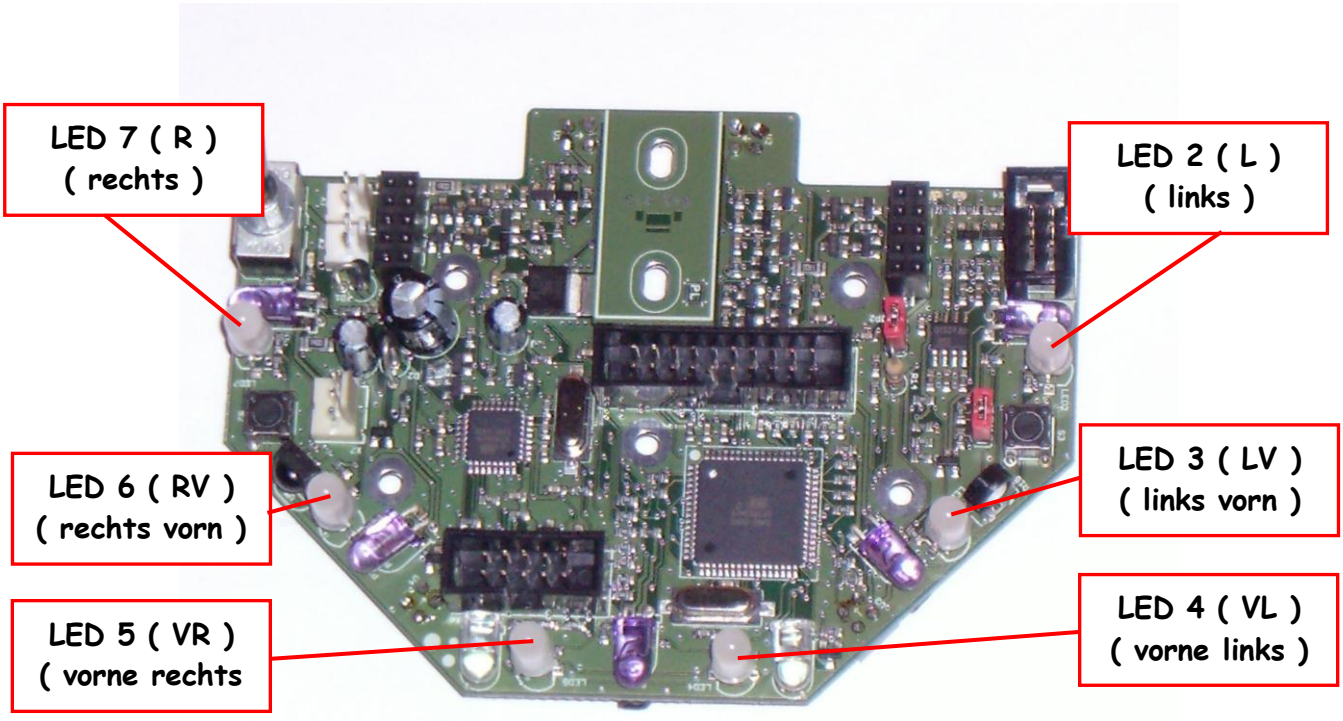
Bitte diese Programme nach Anweisung des Herstellers oder des Tutorials von Nicai, installieren.
Ich arbeite mit Windows 7 und den angegebenen Programmen.
Es können auch andere Programme, z.B. Linux, verwendet werden.
Vom AVR Studio sind in der Zwischenzeit neuere Versionen von Atmel veröffentlicht worden. 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.
Die Beispielprogramme findet ihr bei Roboter.cc. Bei den Beispielen wurde kein Display verwendet. Alle Programme habe ich getestet auf Funktion.

**by H.J.Seeger**

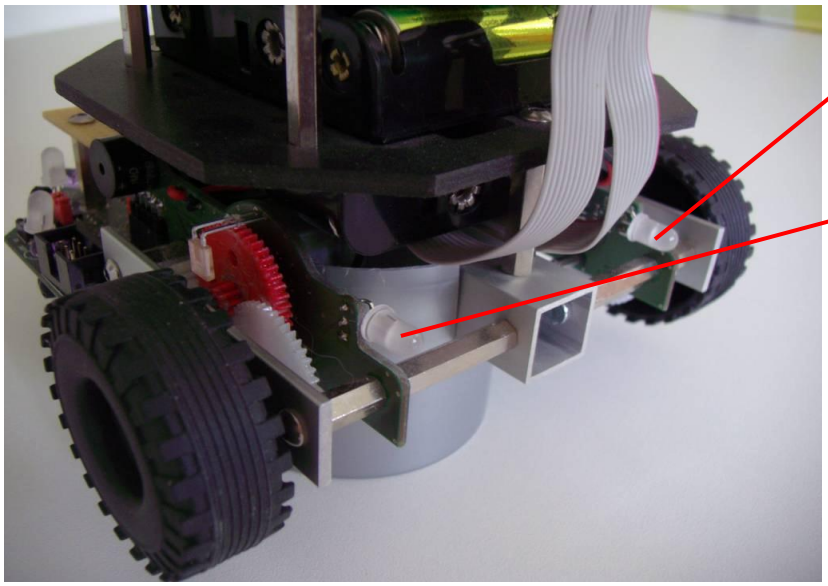
## 1. Hardware

Sehen wir uns einmal die Lage der LED auf dem Nibo 2 an. Ich habe dazu einmal ein kleines Bild gezeichnet mit den Standorten der einzelnen LED`s.




Auf dieser Platine sind nur 6 LED´s, die zwei anderen LED`s sind auf den anderen Platinen




Achtet bitte darauf, dass es sind Zweifarbige – LED`s. Sie haben als Farbe rot und grün und wenn beide LED anschalte bildet sich orange. Sie haben jeweils 3 Beine und müssen korrekt eingelötet werden.

Genug Hardware, kommen wir zum ersten Programm.

## 2. LED 0 bis 7

**PRG_LED_1 ( steht so im Netz ) – LED 5 abwechseln rot und grün schalten**

```
#include <nibo/niboconfig.h>                  // Laden der Header Dateien
#include <nibo/leds.h>                        // PRG_LED_1
#include <nibo/delay.h>

int main()                                    // Start des Programmes
 {
  leds_init();                                // LED`s initiiert
  while(1==1)                                 // Endlosschleife
   {                                          // Klammer Anfang
    delay(500);                               // Pause 500 ms
    leds_set_status(LEDS_RED, 5);             // setzt LED 5 auf rot
    delay(500);                               // Pause 500 ms
    leds_set_status(LEDS_GREEN, 5);           // setzt LED 5 auf grün
   }                                          // Klammer Ende
  return 0;                                   // Zurück, Programmschleife von vorn
 }                                            // Ende Programm
```

In den ersten 3 Zeilen werden die Header-Dateien eingebunden.
Das eigentliche Programm beginnt ab „ **int main()** “. Danach werden die LED`s initiiert. Die Schleife beginnt ab „ **while(1==1)** “. Innerhalb den Klammern { } steht das eigentliche Programm. Durch die folgenden Anweisungen ( **leds_set_status (LEDS_RED, 5);** und **leds_set_status (LEDS_GREEN, 5);** ) wird die Farbe der LED zwischen Rot und Grün umgeschaltet. Die **5** gibt die Nr. der LED an. Durch die Anweisung **delay(500)** legt der Prozessor eine Pause von 500ms ein. Bei **return 0;** endet die Schleife und springt zurück auf **while(1==1)**. Die letzte Klammer **}** wird nicht erreicht. Für weitere Erklärungen der einzelnen Anweisungen verweise

ich auf ein Lehrbuch der Sprache C oder dem Internet.
Gewöhnt euch gleich an den Stil beim Schreiben eines Programmes. Die geschweiften Klammern werden eingerückt, damit man besser sehen kann, in welchem Bereich sie gültig sind und keine vergessen wird.
Zur besseren Sicht habe ich die Kommentare farbig **(blau)** hinterlegt. Zur Funktion sind sie nicht notwendig.
Es sind insgesamt 8 LED´s. Die Zählung beginnt aber bei 0 und geht bis 7.

**PRG_LED_2 ( steht so im Netz )- LED sollen nacheinander reihum in einer Farbe eingeschaltet werden. Die Farbe soll bei jedem Umlauf zwischen Aus, Rot, Grün und Orange wechseln. ( rechts drehend )**

```
#include <nibo/niboconfig.h>                  // Laden der Header Dateien
#include <nibo/leds.h>                        // PRG_LED_2
#include <nibo/delay.h>
int main()                                    // Start des Programms
 {
  leds_init();                                // LED`s initiiert
  while(1==1)                                 // Endlosschleife
   {                                          // Klammer Anfang
    int farbe;                                // Variable farbe
    for (farbe=0; farbe<4; farbe++)           // Zähler farbe von 0 bis 3
     {                                        // 0 = aus, 1 = grün. 2 = rot,3 = orange
      int ledNr;                              // Variable ledNr
      for (ledNr=0; ledNr<8; ledNr++)         // Zähler ledNr von 0 bis 7
       {
        leds_set_status(farbe, ledNr);        // setzt entsprechende LED
        delay(150);                           // Pause 150 ms
       }
     }
   }                                          // Klammer Ende
  return 0;                                   // Zurück, Programmschleife von vorn
 }                                            // Ende Programm
```

Sehen wir uns das Programm einmal näher an. Als erstes werden wieder die Header Dateien eingebunden. Danach erfolgt der Start des Programmes und das initiieren der LED`s.
Als nächstes werden zwei Schleifen gebildet. In der ersten wird die Variable **farbe** deklariert und diese von **0 bis 3** gezählt. In der zweiten Schleife wird die Variable **ledNr** deklariert und von **0 bis 7** gezählt. Beide Schleifen sind ineinander verschachtelt. Es wird zuerst die Schleife mit der Variablen **farbe** aufgerufen, diese wird auf **0** gesetzt. Dann wird die Schleife mit der Variablen **ledNr** aufgerufen. Diese zählt von **0 bis 7** durch. Nach dem **7** erreicht wurde, beginnt das Programm von vorn und erhöht die Variable **farbe** auf **1**. Es folgt wieder die Variable **ledNr**. Nach dem die Variable **farbe** auf **3** erhöht wurde und die **ledNr** ausgeführt wurde, wird **farbe** wieder auf **0** gesetzt und alles beginnt von vorn. Durch die Variable **farbe** wird die Farbe der LED festgelegt. Der Bereich geht von **0 bis 3**. Dabei bedeutet:

- 0 = aus
- 1 = grün
- 2 = rot
- 3 = orange

Die Variable **ledNr** legt die Nummer der LED fest. Der Bereich geht von **0 bis 7**.

**PRG_LED_3 ( steht so im Netz )- LED sollen nacheinander reihum in einer Farbe eingeschaltet werden. Die Farbe soll bei jedem Umlauf zwischen Aus, Rot, Grün und Orange wechseln. ( links drehend )**

```
#include <nibo/niboconfig.h>                  // Laden der Header Dateien
#include <nibo/leds.h>                        // PRG_LED_3
#include <nibo/delay.h>

int main()                                    // Start des Programms
 {
  leds_init();                                // LED`s initiiert
  while(1==1)                                 // Endlosschleife
   {                                          // Klammer Anfang
    int farbe;                                // Variable farbe
    for (farbe=0; farbe<4; farbe++)           // Zähler farbe von 0 bis 3
     {                                        // 0 = aus, 1 = grün. 2 = rot,3 = orange
      int ledNr;                              // Variable ledNr
      for (ledNr=7; ledNr>-1; --ledNr)        // Zähler ledNr von 7 bis 0
       {
        leds_set_status(farbe, ledNr);        // setzt entsprechende LED
        delay(150);                           // Pause 150 ms
       }
     }
   }                                          // Klammer Ende
  return 0;                                   // Zurück, Programmschleife von vorn
 }                                            // Ende Programm
```

Sehen wir uns dieses Programm einmal genauer an. Welcher Unterschied besteht zum **LED_2**? Eigentlich nicht viel. Es werden die gleichen Variablen verwendet. Der Wertebereich ist gleich. Nur die Zählrichtung unterscheidet sich. Bei dem Programm **LED_2** wird die Variable **ledNr** von **0 bis 7** gezählt. Bei dem Programm **LED_3** wird die Variable **ledNr** von **7 bis 0** gezählt. Dabei wird bei **7** begonnen und bis **0** gezählt. Dadurch beginnt bei LED 7 die Funktion und es dreht sich links herum. Genauso kann auch kann ich auch die Variable **farbe** verändern.

Es ist auch eine Kombination aus beiden Programmen möglich. In dem nachfolgenden Programm habe ich es so gemacht. Dabei wird links und rechts verwendet. Es werden aber nur die LED 2,3,4 und LED 5,6,7 verwendet. Dadurch entsteht ein Eindruck wie bei „ Knight Rider "

**PRG_LED_4 ( steht so im Netz )- LED sollen jeweils links und rechts an und aus gehen**

```
#include <nibo/niboconfig.h>                // Laden der Header Dateien
#include <nibo/leds.h>                      // PRG_LED_4
#include <nibo/delay.h>
#include <avr/interrupt.h>
#include <nibo/iodefs.h>
#include <stdio.h>
int main()                                  // Beginn Hauptprogramm
 {
   sei();                                   // Einbinden der Dateien
   leds_init();
   while(1)
    {                                       // Beginn Schleife
      int Llf;                              // Variable Llf = LEDlauf
      for (Llf=2;Llf<=4;Llf++)              // Schleife LED an
       {                                    // 0 = aus, 1 = grün. 2 = rot,3 = orange
         leds_set_status(2,Llf);            // LED an 2,3,4
         leds_set_status(2,9-Llf);          // LED an 5,6,7
         delay(250);                        // Pause 250 ms
       }
     for (Llf=4;Llf>=2;Llf--)               // Schleife LED aus
      {
        leds_set_status(0,Llf);             // LED aus 2,3,4
        leds_set_status(0,9-Llf);           // LED aus 5,6,7
        delay(250);                         // Pause 250 ms
      }
     delay(150);                            // Pause 150 ms
    }
   return 0;
 }
```

Sehen wir uns mal die Funktion an. Es sind wieder 2 Schleifen vorhanden. In der ersten Schleife gehen die Zahlen von **2 bis 4** und durch die Subtraktion ( **9-Llf** ) von **7 bis 5** und schalten die LED`s ein. In der zweiten Schleife gehen die Zahlen von **4 bis 2** und wieder durch die Subtraktion von ( **9-Llf** ) von **5 bis 7** und schalten die LED`s aus.

Um ein Muster zu gestalten, gibt es die verschiedensten Möglichkeiten. Man kann die Farben, die Richtung, die Geschwindigkeit, die Helligkeit, die Ein- und Ausgänge, die Zeit und eine Kombination aus allem machen. Es bleibt jedem selbst überlassen, entsprechende Änderungen an den Beispielen zu machen. Ich habe noch ein paar Beispiele angefügt.

```
// PRG_LED_5 by h.j.seeger - Baustellenlicht

#include <nibo/niboconfig.h>                    // Laden der Header Datein
#include <nibo/leds.h>
#include <nibo/delay.h>
#include <avr/interrupt.h>
#include <nibo/iodefs.h>
#include <stdio.h>
static int leds_left[]={1,2,3,4};
static int leds_right[]={0,7,6,5};

int main()                                      // Beginn Hauptprogramm
 {
   sei();                                       // Einbinden der Datein
    leds_init();                                // IO-Ports für LEDs initiiert
    while(1)
     {                                          // Beginn Schleife
      for(int c=1;c<=3;c++)
        {
         for(int i=0; i<=5; i++)
          {
            if(i<=3)
             {
               leds_set_status(c,leds_right[i]);
               leds_set_status(c,leds_left[i]);
             }
            delay(120);
            if(i==2)
             {
               leds_set_status(0,leds_left[0]);
               leds_set_status(0,leds_right[0]);
             }
           if(i==3)
             {
               leds_set_status(0,leds_left[1]);
               leds_set_status(0,leds_right[1]);
             }
           if(i==4)
            {
              leds_set_status(0,leds_left[2]);
              leds_set_status(0,leds_right[2]);
            }
          if(i==5)
            {
              leds_set_status(0,leds_left[3]);
              leds_set_status(0,leds_right[3]);
            }
```

```
          delay(30);
         }
        leds_set_status(0,leds_left[3]);
        leds_set_status(0,leds_right[3]);
      }
   }
return 0;
}
```

**// PRG_LED_6 by h.j.seeger - // Led lauflicht, wie bar_graph anzeige**

```
#include <nibo/niboconfig.h>                    // Laden der Header Dateien
#include <nibo/leds.h>
#include <nibo/delay.h>
#include <avr/interrupt.h>
#include <nibo/iodefs.h>
#include <stdio.h>
int leds_left[]={1,2,3,4};
int leds_right[]={0,7,6,5};
int main()                                      // Beginn Hauptprogramm
{                                               // Beginnt mit {
sei();                                          // Einbinden der Datein
leds_init();                                    // LEDs initiiert
while(1)
 {                                              // Beginn Schleife
   for(int c=1;c<=3;c++)
    {
      for(int i=0; i<=3; i++)
       {
         leds_set_status(c,leds_left[i]);
         leds_set_status(c,leds_right[i]);
         delay(100);
       }
      delay(100);
      for(int i=3; i>=0; i--)
       {
         leds_set_status(0,leds_left[i]);
         leds_set_status(0,leds_right[i]);
         delay(100);
       }
    }
 }
return 0;
}
```

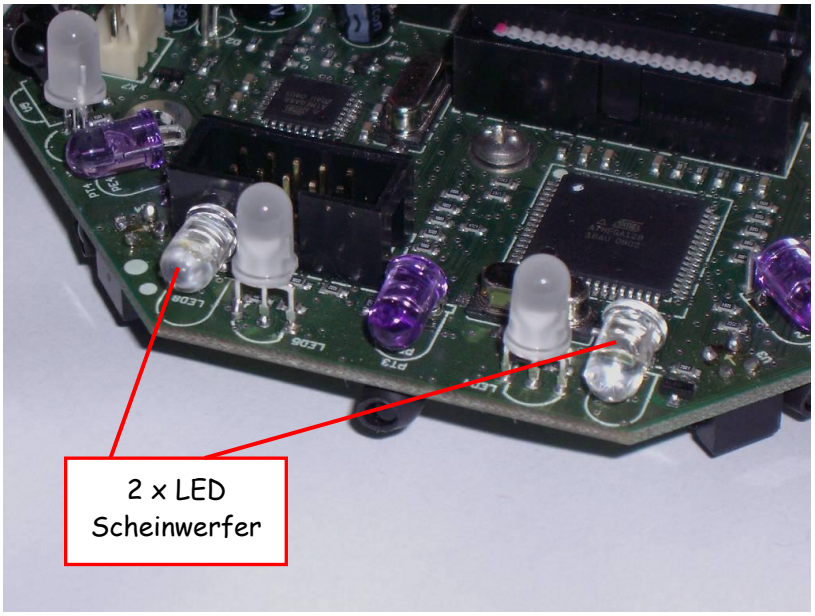
```
// PRG_LED_7 by h.j.seeger – LED laufen vor und zurück und wechseln Farbe

#include <nibo/niboconfig.h>
#include <nibo/leds.h>                  // Ladet Header Dateien
#include <nibo/delay.h>
#include <avr/interrupt.h>
#include <nibo/iodefs.h>
#include <stdio.h>
int main()                              // Beginn Hauptprogramm
 {                                      // Beginnt mit {
   sei();                                     // Einbinden der Datein
   leds_init();                         // IO-Ports für LEDs initiiert
   while(1)
    {                                   // Beginn Schleife
      int lednr;                        // Variable llf = LEDlauf
      for (lednr=2;lednr<=7;lednr++)    // Zähler vorwärts
          {
        leds_set_status(1,lednr);       // LED grün an
        delay(100);                     // Pause
        leds_set_status(0,lednr);       // LED grün aus
         }
        for (lednr=7;lednr>=2;lednr--)  // Zähler rückwärts
         {
           leds_set_status(1,lednr);    // LED grün an
           delay(100);                  // Pause
           leds_set_status(0,lednr);    // LED grün aus
         }
       for (lednr=2;lednr<=7;lednr++)   // Zähler vorwärts
        {
          leds_set_status(2,lednr);     // LED rot an
          delay(100);                   // Pause
          leds_set_status(0,lednr);     // LED rot aus
        }
      for (lednr=7;lednr>=2;lednr--)    // Zähler rückwärts
       {
         leds_set_status(2,lednr);      // LED rot an
         delay(100);                    // Pause
         leds_set_status(0,lednr);      // LED rot aus
       }
     for (lednr=2;lednr<=7;lednr++)     // Zähler vorwärts
      {
        leds_set_status(3,lednr);       // LED orange an
        delay(100);                     // Pause
        leds_set_status(0,lednr);       // LED orange aus
       }
    for (lednr=7;lednr>=2;lednr--)      // Zähler rückwärts
```

```
      {
        leds_set_status(3,lednr);          // LED orange an
        delay(100);                        // Pause
        leds_set_status(0,lednr);          // LED orange aus
      }
  }
return 0;
}
```

## 3. Scheinwerfer - LED

An der vorderen Seite befinden sich 2 LED`s. Diese sind nach vorn gebogen. Auf Grund der grossen Helligkeit beider LED`s werden sie als Scheinwerfer genutzt.
Die Ansteuerung erfolgt für beide LED`s über einen Transistor. Dadurch können sie nur gemeinsam geschaltet werden.
Durch ein PWM-Signal kann aber die Helligkeit gesteuert werden.
Ein direkter Blick in die eingeschalteten LED`s kann zu einem Blenden führen. Seit bitte vorsichtig. Es ist nicht gefährlich, aber unangenehm.




**PRG_LED_8 ( steht so im Netz )- Die LED der Scheinwerfer sollen in Stufen verschiedenen Helligkeiten schalten.**

```
// PRG_LED_8 by h.j..seeger
#include <avr/interrupt.h>                 // Einbinden der Header Datein
#include <nibo/niboconfig.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <nibo/delay.h>
int main()                                 // Beginn Programm
 {
   sei();
   leds_init();
   pwm_init();
   while(1==1)                             // Endlosschleife
    {
      leds_set_headlights(0);              // SW aus
      delay(1000);                         // Pause 1 s
```

```
    leds_set_headlights(64);          // SW 1/16 Helligkeit
     delay(1000);                     // Pause 1 s

     leds_set_headlights(256);        // SW 1/4 Helligkeit
     delay(1000);                     // Pause 1 s

     leds_set_headlights(1023);       // SW 1/1 Helligkeit
     delay(1000);                     // Pause 1 s
   }
  return 0;
}
```

Sehen wir uns auch dieses Programm einmal genauer an. In den ersten 5 Zeilen werden die Header Dateien eingebunden. Danach werden die einzelnen Teile initiiert. Ab **while (1==1)** beginnt die Endlosschleife und das eigentliche Programm. Durch die Anweisung **leds_set_headlights()** wird die Helligkeit der beiden SW-Led gesetzt. Dabei sind insgesamt **1024** Werte möglich. Da wieder bei **0** begonnen wird, ist der maximale Wert **1023**.
Ich schalte die Helligkeit der SW-LED in mehreren Stufen von aus bis max. Man kann ohne Problem noch andere Werte eingeben oder erweitern.

Es geht natürlich auch besser:

**PRG_LED_9 ( steht so im Netz )- Die LED der Scheinwerfer sollen in sehr kleinen Stufen Helligkeiten ein- und ausschalten schalten.**

```
// PRG_LED_9  by achim S. nach Lutz

#include <nibo/niboconfig.h>                    // Einbinden der Header Dateien
#include <nibo/iodefs.h>
#include <nibo/delay.h>
#include <nibo/leds.h>
#include <nibo/pwm.h>
#include <avr/interrupt.h>
int main(void)
 {                                              // Beginn Programm
  sei();
  leds_init();
  pwm_init();
  while(1==1)
   {
    int Headlight;                              // Variable Headlight
    for (Headlight=0; Headlight<1025; Headlight++)  // Zähler vorwärts
     {
      leds_set_headlights (Headlight);          // setzt SW-LED
      delay(2);                                 // Pause 2 ms
     }
    for (Headlight=1024; Headlight>-1; Headlight--)  // Zähler rückwärts
   {
   leds_set_headlights (Headlight);             // setzt SW-LED
```

```
    delay(2);                                   // Pause 2 ms
  }
 }
return 0;
}
```

Wie bei den anderen Programmen, werden zuerst die Header Dateien eingebunden. Das Programm wird gestartet und einiges initiiert. Danach kommt die Endlosschleife. Dort wird zuerst ein **vorwärts Zähler** gestartet. Mit diesem Wert wird **Headlight** erhöht bis zum Maximum. Nach einer kurzen Pause wird ein **rückwärts Zähler** gestartet, der **Headlight** vom Maximum auf **>1** zählt. Auch dieser Wert wird für die Helligkeit der SW-LED genutzt. Nach dem Erreichen des minimalen Wertes beginnt das Programm von vorn.

## 4. Ports und LED´s

Die einzelnen LED´s können auch durch einen Port Befehl gesetzt werden. Im Einzelnen kann man diese zuordnen:

| | |
|---|---|
| 0 X 01 = LED **0** | 0 X 10 = LED **4** |
| 0 X 02 = LED **1** | 0 X 20 = LED **5** |
| 0 X 03 = LED **0 + 1** | 0 X 30 = LED **4 + 5** |
| 0 X 04 = LED **2** | 0 X 40 = LED **6** |
| 0 X 05 = LED **0 + 2** | 0 X 50 = LED **4 + 6** |
| 0 X 06 = LED **1 + 2** | 0 X 60 = LED **5 + 6** |
| 0 x 07 = LED **0 + 1 + 2** | 0 X 70 = LED **4 + 5 + 6** |
| 0 X 08 = LED **3** | 0 X 80 = LED **7** |
| 0 X 0A = LED **1 + 3** | 0 X A0 = LED **5 + 7** |
| 0 X 0B = LED **0 + 1 +3** | 0 X B0 = LED **4 + 5 + 7** |
| 0 X 0C = LED **2 + 3** | 0 X C0 = LED **6 + 7** |
| 0 X 0D = LED **0 + 2 + 3** | 0 X D0 = LED **4 + 6 + 7** |
| 0 X 0E = LED **1 + 2 +3** | 0 X E0 = LED **5 + 6 + 7** |
| 0 X 0F = LED **0 + 1 + 2 + 3** | 0 X F0 = LED **4 + 5 + 6 + 7** |

Der Aufruf kann durch **IO_LEDS_RED_PORT = 0 x FF** erfolgen. Bei **0 X FF** sind alle LED´s eingeschaltet. Bei **0 X 00** sind alle LED´s ausgeschaltet. Die Farben werden durch die Ports umgeschaltet. Für Grün verwendet man **IO_LEDS_GREEN_PORT**. Für orange verwendet man beide Ports. Dadurch kann man mit einer Anweisung mehrere LED`s schalten.
Ein kleines Beispiel zum Schalten der LED`s über Ports:

```
// PRG_LED_10 by h.j.seeger@web.de

#include <nibo/niboconfig.h>            // Laden der Header Dateien
#include <nibo/leds.h>                  // Beachte Gross- und kleinschreibung
#include <nibo/delay.h>
#include <nibo/iodefs.h>
int main()                              // Start des Programmes
 {
  leds_init();                          // LED`s initiiert
   while(1==1)                          // Endlosschleife
```

```
{
    IO_LEDS_RED_PORT = 0x84;        // LEDs 2,7 rot an
     delay(500);
    IO_LEDS_RED_PORT = 0x00;        // LEDs 2,7 rot aus

    IO_LEDS_RED_PORT = 0x48;        // LEDs 3,6 rot an
     delay(500);
    IO_LEDS_RED_PORT = 0x00;        // LEDs 3,6 rot aus

    IO_LEDS_RED_PORT = 0x30;        // LEDs 4,5 rot an
     delay(500);
    IO_LEDS_RED_PORT = 0x00;        // LEDs 4,5 rot aus

    IO_LEDS_GREEN_PORT = 0x84;      // LEDs 2,7 grün an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 2,7 grün aus

    IO_LEDS_GREEN_PORT = 0x48;      // LEDs 3,6 grün an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 3,6 grün aus

    IO_LEDS_GREEN_PORT = 0x30;      // LEDs 4,5 grün an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 4,5 grün aus

    IO_LEDS_GREEN_PORT = 0x84;      // LEDs 2,7 grün an
    IO_LEDS_RED_PORT = 0x84;        // LEDs 2,7 rot an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 2,7 grün aus
    IO_LEDS_RED_PORT = 0x00;        // LEDs 2,7 rot aus

    IO_LEDS_GREEN_PORT = 0x48;      // LEDs 3,6 grün an
    IO_LEDS_RED_PORT = 0x48;        // LEDs 3,6 rot an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 3,6 grün aus
    IO_LEDS_RED_PORT = 0x00;        // LEDs 3,6 rot aus

    IO_LEDS_GREEN_PORT = 0x30;      // LEDs 4,5 grün an
    IO_LEDS_RED_PORT = 0x30;        // LEDs 4,5 rot an
     delay(500);
    IO_LEDS_GREEN_PORT = 0x00;      // LEDs 4,5 grün aus
    IO_LEDS_RED_PORT = 0x00;        // LEDs 4,5 rot aus
  }                                 // Klammer Ende
 return 0;                          // Zurück, Programmschleife von vorn
}                                   // Ende Programm
```

In dem kleinen Tutorial sind wir am Ende. Ich habe jedes Programm mit AVR Studio 4 erstellt und getestet. Hoffe, das keine Fehler drin sind. Ihr könnt die Programme auf Roboter.cc laden und ausprobieren. Viel Spass beim lesen und Programmieren