Roboter-Nachrichten 01 / 2015

Vorwort:

Liebe Leser, etwas verspätet, aber trotzdem möchten wir Ihnen ein gutes Neues Jahr wünschen. Die fünfte Ausgabe der Roboter-Nachrichten wird einiges aufarbeiten, dass wir in 2014 versprochen hatten, aber aus Zeitgründen nicht liefern konnten. Außerdem haben wir den Liniensensor vereinfacht, so dass kein extra ATtiny Mikrocontroller nötig ist. Das macht den Sensor günstiger und viel einfacher im Nachbau. Dank der netten Firma (bekannter Deutscher Kaffeeröster) sind wir nun im Besitz eines 3D-Druckers. Leider kommen wir kaum zum selber Drucken, da Kollegen und Freunde sich melden und „mal schnell“ etwas aus dem 3D-Drucker wollen. Aber trotzdem ist es gelungen, ein Rad für den neuen Roboter R2PT4 (Schwarmroboter) zu entwickeln und auch schon erfolgreich zu drucken. In einer der nächsten Ausgaben werden wir ausgiebig über dieses Projekt berichten.

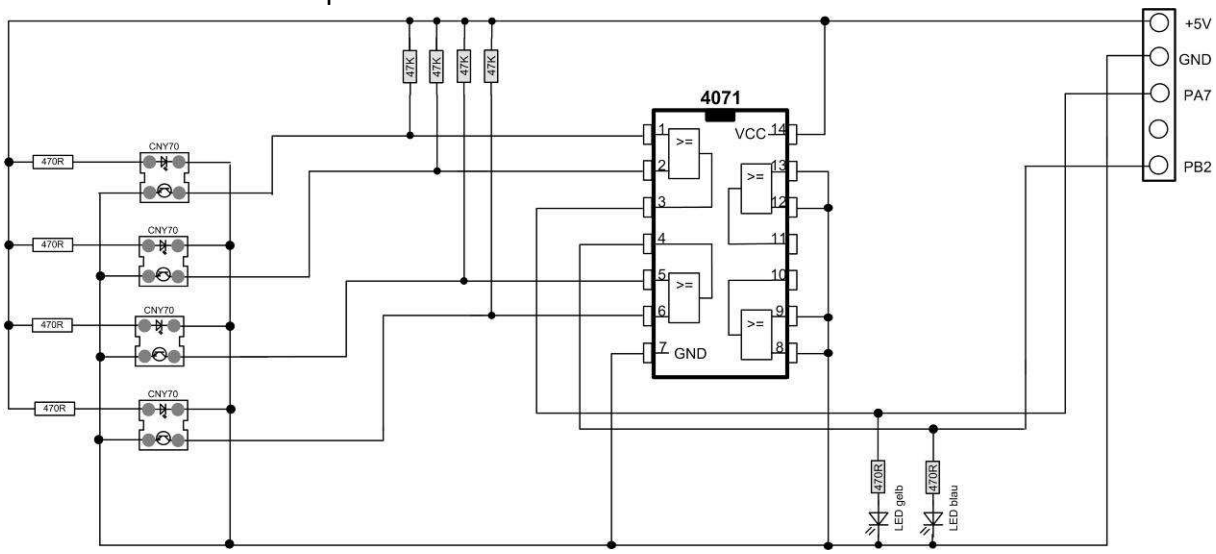
Die Roboternachrichten stehen auch als Download auf den Webseiten des VTH Verlages zur Verfügung www.vth.de oder auf den Webseiten von Schneider Engineerings www.ps-robotics.de .

Inhaltsangabe:

Projekt: Linien folgen (Update)

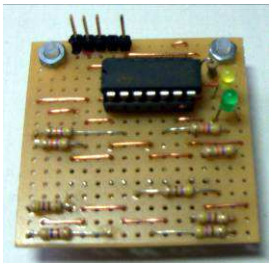
Die gute Nachricht zuerst, an der Software zur Auswertung, falls Sie diese schon geschrieben haben, muss nichts geändert werden. Dem Leser, der erst jetzt mit dem Gedanken spielt sich einen neuen Sensor zu bauen, empfehlen wir den Artikel über Linien folgen in den Roboternachrichten 04 / 2014 zu lesen. Wie gesagt, die Funktion bleibt gleich, allerdings entfällt der ATtiny44 Mikrocontroller, der durch einen MOS IC 4071 ersetzt wird. Jetzt eine interessante Beobachtung, der 4071 ist hochohmiger, als der Mikrocontroller und somit verändert sich die Empfindlichkeit der Sensoren vom Type CMY70. Das bedeutet, dass wir den Widerstandswert der IR-Sendediode anpassen müssen, da sonst der Abstand zwischen Boden und Sensor zu groß wird. Der Sensor reagierte schon bei einem Abstand von mehr als drei Zentimetern und nicht wie geplant bei 5-8mm. Merke, muss der Abstand zwischen Sensor CNY70 und Boden angepasst werden, dann einfach den Widerstand der Sendediode anpassen. In unserem Fall haben wir den Widerstandswert von 220Ohm auf 470Ohm erhöhen müssen. Bitte den max. Strom durch die Diode beachten (siehe Datenblatt) wenn an den Widerständen Veränderungen durchgeführt werden !!!

Hier nun der neue Schaltplan:

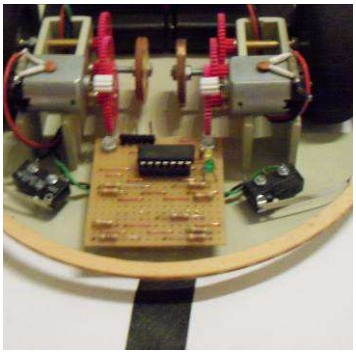


Die Anzahl der Bauteile hat sich geändert, z.B. entfällt der Taster und so kann die Platine auf weniger Platz realisiert werden.

Hier ein Bild von dem Prototype Platine und eingebaut im Chassis



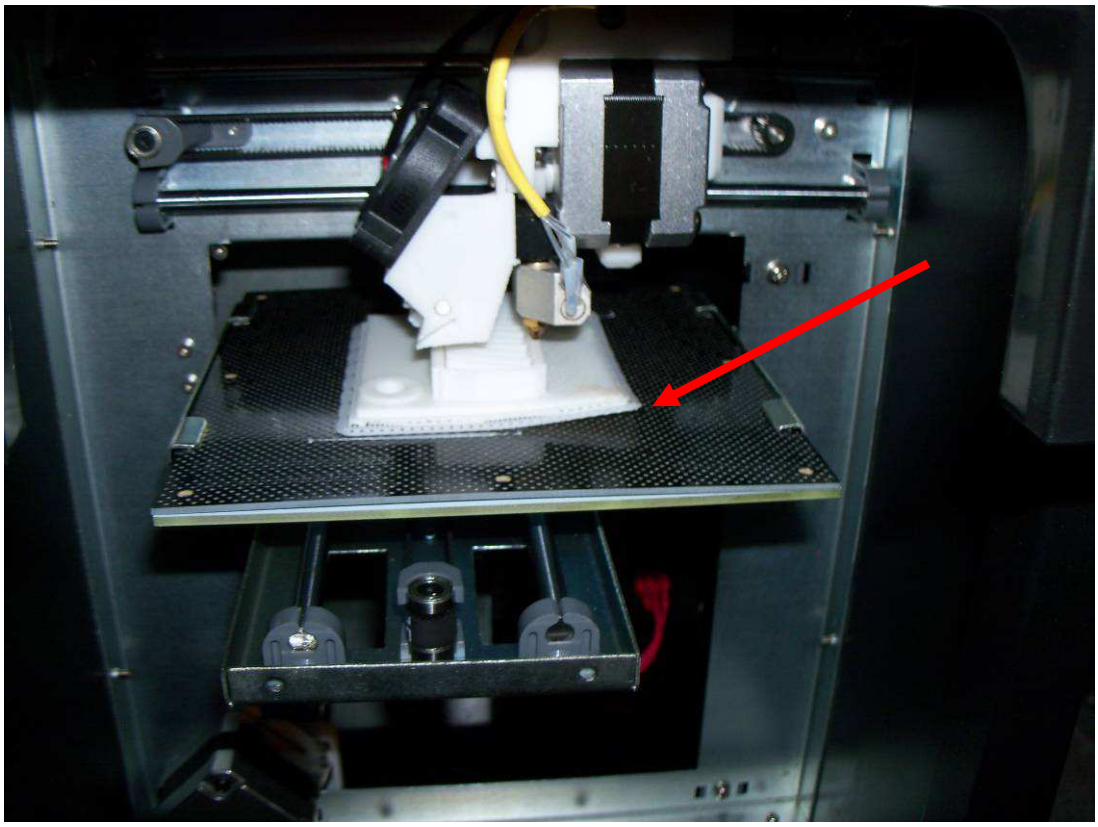
des R2PT3 Roboters. Der Liniensensor kann als Bausatz von Schneider-Engineerings bezogen werden, siehe www.ps-robotics.de im Bereich Produkte. Übrigens ein weiteres Projekt für dieses Jahr, die Webseite ist etwas in die Jahre gekommen und soll in ein modernes Design überführt werden. Aber alles zu seiner Zeit.



Unser „neues“ Spielzeug der 3D-Ducker

Eine tolle Sache, aber:

- Die Einrichtung (kalibrieren) des Druckers hat mal locker drei bis vier Stunden gedauert.
- Die Suche nach einen geeignetem Programm, mit dem man die Modelle erstellen kann war das nächste große Hindernis. Der Erfolg im Internet war etwas bescheiden. Die meisten Programme sind nicht gerade selbst erklärend und so sind wir bei unserem TurboCad Professional hängengeblieben, das wir schon seit Jahren für den Entwurf der Roboter nutzen.
- Natürlich war die verwendete TC Version V14 Professional nicht in der Lage, das entsprechende File-Format zu speichern, welche für die Ansteuerung eines 3D-Druckers benötigt wird. STL (Stereolithographie) ist das Format, das wir benötigen und erst ab Version TC V19 Professional verfügbar. Also noch mal ca. 250 Euro ausgeben, oder mit Programmen aus dem Internet arbeiten.
- Erster Druck eines größeren Teils und natürlich löste sich eine Ecke ab. Diesen Vorgang nennt man „Wraping“ und ist das Größte Problem bei dem 3D-Druck, siehe Bild.

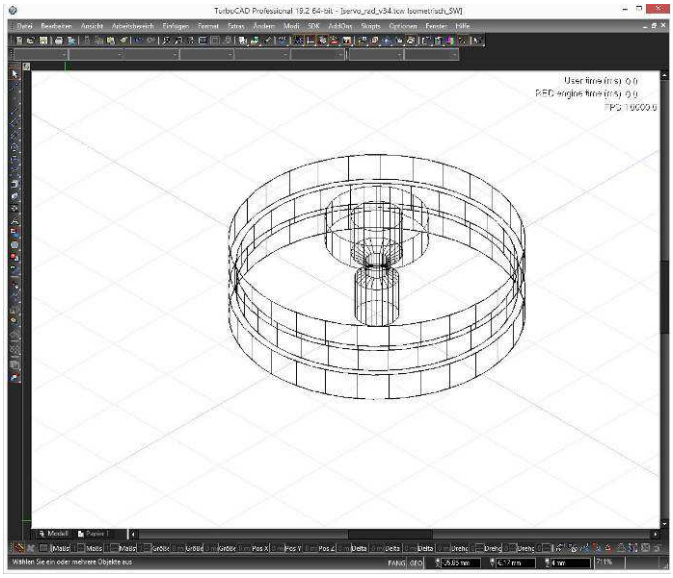


- Tja, hätte man das Buch von Herrn Oliver Bothmann „3D-Druck-Praxis“, erhältlich im Shop des VTH Verlages, mal besser zuerst gelesen … Im Buch wird als Auslöser des Wraping beschrieben, dass ein zu schnelles Abkühlen des gedruckten Teils zu Ablösungen und Verformungen führen kann.

- Auch ganz hilfreich sind Foren im Internet. Dort kann man nach lesen, wie man den Drucker richtig anspricht oder ihn modifiziert, um eine höhere Temperatur der Arbeitsplatte zu erreichen. Auch die Qualität des Filaments (ABS Kunststoff der zum Druck benötigt wird) wirkt sich stark auf das Wraping und die Qualität des Druckes aus.

Nun gut, aus Fehlern lernt man ja auch. Die Entwicklung eines neuen Chassis für den R2PT4 wird somit eine Herausforderung werden, da die Grundplatte ca. 90mm (rund) werden soll. Hier mal ein Foto mit dem Prototypen als Vorschau für eine der nächsten Roboternachrichten. Der Antrieb wird mittels „gehackten“ Servos erfolgen. Siehe entsprechendes Kapitel im Buch „Der Weg zum eigenen Roboter“. Dort wird der Servo-Hack beschrieben (Siehe Seite 46). Die Räder des neuen Roboters sind übrigens aus dem 3D-Drucker, siehe folgende TurboCAD Zeichnung. Der schwarze Ring um das Rad ist eine Dichtung aus dem Baumarkt, die verhindern soll, dass der Roboter auf glatten Flächen durchdreht.

So, mehr wird noch nicht verraten!



TurboCad V19 Professional

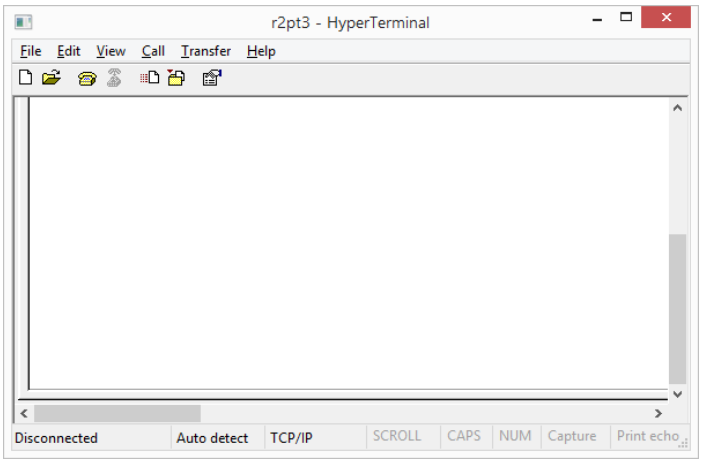
Fertig gedrucktes Rad, noch nicht vom Trägermaterial entfernt

Sind Sie interessiert am 3D-Drucken? Gerne druckt Schneider-Engineerings Ihre Entwürfe für „kleines Geld“. Das spart erst mal die Kosten für einen eigenen 3D-Drucker. Genaue Größen, File-Formate und Kosten klären wir individuell. Kontaktieren Sie uns z.B. via Email: info@ps-robotics.de

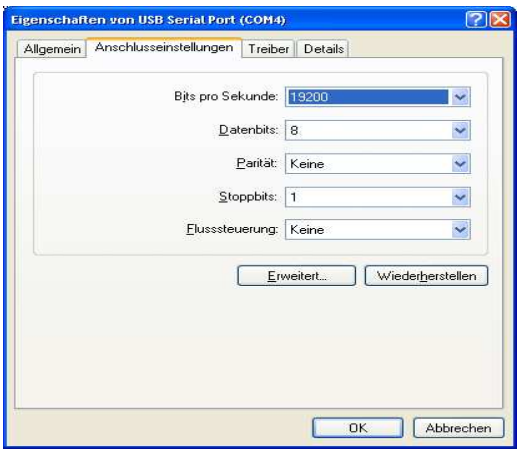
Die serielle Schnittstelle RS232 oder auch UART genannt

Das größte Problem bei dem Design der Software für den Roboter ist das gleiche, wie mit dem Jogger und dem Hundebesitzer. Der will nur (harmlos) spielen und dann der Satz „das hat er aber noch nie gemacht“. Wie passt das zum Roboter? Wir können nicht ohne zusätzlichen Aufwand in einen Roboter hineinschauen und so wissen wir nicht welche Programmschritte der Roboter gerade durchführt, oder banal gesagt, wo es gerade dran hängt, dass der Roboter etwas von uns nicht geplantes tut. Klar, Sie können jetzt sagen, dass Programm ist schlecht entworfen oder nicht bis zu Ende durchdacht, aber können Sie immer alle äußeren Parameter, die durch z.B. Sensoren einfließen, immer genau im Voraus bestimmen? Ich kann das jedenfalls nicht!

Wie kann man sich helfen das Problem zu lösen? Zum einen mit ein paar extra LEDs, wie zum Beispiel bei dem Linien-Sensor. Die LEDs zeigen an, ob eine Linie für den Sensor sichtbar ist. Für ein großes Programm mit all seinen Unterroutinen brauchen Sie allerdings eine Menge LEDs, um das Programm zu erfassen! Wäre es nicht besser, dass für die Versuchsphase, der Roboter uns Informationen direkt aus dem Programm heraus liefert. Falls sich der Roboter nicht frei im Raum bewegen muss für Tests und der Bewegungsradius begrenzt werden kann, bietet es sich an den Roboter über die Serielle Schnittstelle an ein Computerterminal anzubinden. Nein, Sie benötigen kein altes VT100 Terminal, eine Terminalsimulation auf



Ihrem PC reicht schon aus. Falls Sie den Roboter über eine serielle Schnittstelle und mit einem Ladeprogramm (Boot-loader) betreiben, haben Sie alle Vorrausetzungen bereits erfüllt. Schließen Sie den Roboter über Ladekabel an die RS232 oder den Dongel (UBS zu RS232) an und starten das Terminal-programm am Computer. Das Program Hyperterminal bietet sich als einfache Variante an und kann kostenlos aus dem Internet geladen werden. Folgende Einstellungen müssen auf Seite des PCs vorgenommen werden.



Dazu öffnen wir unter Windows Systems den Geräte-Manager und wählen die entsprechende „COM“ Schnittstelle aus, an der unser USB-zu-RS232- Adapter hängt, oder wenn Sie noch eine RS232-Schnittstelle (serielle Schnittstelle) am PC haben, wählen Sie die Schnittstelle aus, an dem Sie unsere Test- und Programmierplatine anschließen wollen. Das kann z.B. bei Windows XP folgendermaßen aussehen (siehe Bild): Die Ansicht zeigt die Konfiguration eines PCs, der mit dem Betriebssystem Windows XP läuft: Die Baudrate ist

19200, 8 Datenbits, Parität keine, 1 Stoppbit und Flusssteuerung (Xon/Xoff) auf „keine“ setzen. Bei Windows 7 oder Windows 8 sieht es vom Bild her etwas anders aus, aber die einzugebenden Daten wie Baudrate, Datenbits, Parität & Stoppbit sind gleich. Somit wäre der PC vorbereitet, was uns noch fehlt ist die Ansteuerung des UART (RS232) auf dem Mikrocontroller. Wie schon in einem der letzten Newsletter (Roboter-Nachrichten) beschrieben, kann man sich das Leben schwer machen und etwas neues entwickeln mit Hilfe der Datenblätter oder man greift auf die bewährten Routinen von Peter Fleury zurück und lädt die entsprechenden Software-Bibliotheken von seiner Webseite herunter http://homepage.hispeed.ch/peterfleury/avr-software.html .
In dem Directory in dem Sie Ihren Programmcode geschrieben haben müssen folgende Dateien hinzugefügt werden: „uart.c“ und „uart.h“. Hier nun ein Beispielcode für ein Programm, das die Zeichenfolge „1m“ überträgt.

```
// ******* Header dateien********
#include  <avr/io.h>
#include  <avr/interrupt.h>
#include  <avr/pgmspace.h>
#include  <stdlib.h>

#define   F_CPU   8000000
#define   UART_BAUD_RATE 9600         // Festlegen der Baudrate

#include  "uart.h"
#include  "uart.c"

char       buff2[10];

int main (void)
{
  DDRD  = 0xff;               // Bits als Ausgang setzen

  // 9600Baut, No Parity, 1 Stopbit, Xon/Xoff ausgeschaltet
  uart_init(UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU));
  sei();             // Wichtig: Enable global Interrupt

  // **** Endlosschleife des Hauptprogrammes ****
  while (1)
  {
       buff2[0] = 49;              // ASCII Code fuer eine 1
       buff2[1] = 'm';             // m als Zeichen übergeben
       buff2[2] = '\0';            // Zeichenkettenende

       uart_putc (buff2[0]);        // Schnittstelle (UART)
       uart_putc (buff2[1]);        // Schnittstelle (UART)
       uart_putc (buff2[2]);        // Schnittstelle (UART)
       uart_putc (13);              // CR Return
       uart_putc (10);              // Line Feed

       // ********** O D E R ***********

       uart_puts (buff2);           // gibt das gleiche Ergebnis
       uart_putc (13);              // CR Return
       uart_putc (10);              // Line Feed
  }
  return (0);

} // Ende Programm
```

Sieht nicht nur einfach aus, ist es auch.
Tipp: Die häufigsten Fehler sind falsche Einstellungen der Schnittstellenparameter.

Kolumne von Klaus Wellmann

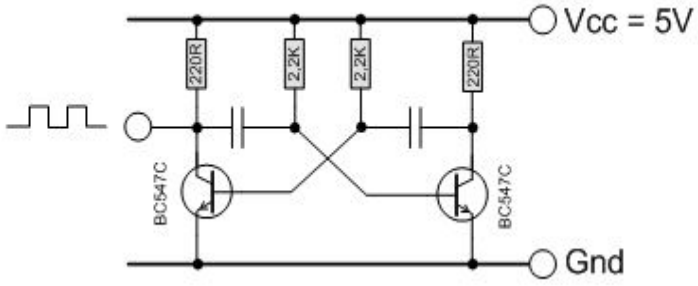
Methoden, um ein Blinklicht zu herzustellen.

Der Bedarf an Blinklichtsignalen ist so alt, wie die Menschheit. Ich behaupte, das wohl bekannteste Blinklicht war eines der sieben antiken Weltwunder, nämlich der Leuchtturm von Alexandria! Es gibt wohl historische Beschreibungen des Turms (ca. 300 v. Chr), jedoch kein Geschichtsschreiber hat die Lichtquelle genau beschrieben. Möglicherweise war es ein Hohlspiegel, der bis 55 km weit leuchten konnte – nichts genaues weiß man nicht ...
Egal: Ich behaupte frech, dass die Gelehrten der Antike schon damals wussten, dass (1.) eine rotierende Lichtquelle, oder (2.) eine fixe Lichtquelle, vor der rotierende Abdeckungen kreisen, (also Blinklichteffekt) auf hoher See besser zu erkennen ist, als eine konstante Lichtquelle. Wem das ein zu unsicherer Beweis für das Bedürfnis der Menschen nach Blinklichtern ist, dem reiche ich die römischen Beschreibungen des Limes (Hadrianwall um 122 n. Chr) nach. ….nachts erfolgte die Signalgebung auf den Wachtürmen mittels Lichtzeichen... (also Blinklichter) Ich hoffe, dass ich jetzt den letzten Zweifler überzeugt habe...

Jetzt aber einen Sprung in die Neuzeit.

Erste Methode
Mit dem Aufkommen der ersten technisch ausgereiften Transistoren Anfang der 1950-er, war es mit der Röhrentechnik (weil zu groß, zu hoher Strombedarf, zu schwer) vorbei. Zum Beispiel: Das erste bezahlbare kleine Transistorradio für jedermann kam 1954 auf den Markt – eine Sensation! Welche verschiedenste Schaltungen entwickelten sich? Ja, klar unter anderem: das Blinklicht mittels einem astabilen Multivibrator, oder auch Kippstufe genannt. Die Schaltung besitzt keine stabilen Zustände, wenn der Transistor Q1 leitet, wird dadurch Q2 gesperrt. Durch das Zurückkoppeln des Ausgangssignales vom Kollektor auf die Basis des Transistors Q1, wird dieser nun gesperrt und folglich wird Q2 leitend. Das geschieht selbstständig in einem Rhythmus, der durch die Elko-Kondensatoren C1 + C2 und die Widerstände R2 und R3 bestimmt wird. Also, mal selbst ausprobieren und die angegebenen Kondensatoren und Widerstände einfach mal tauschen bzw. verändern....

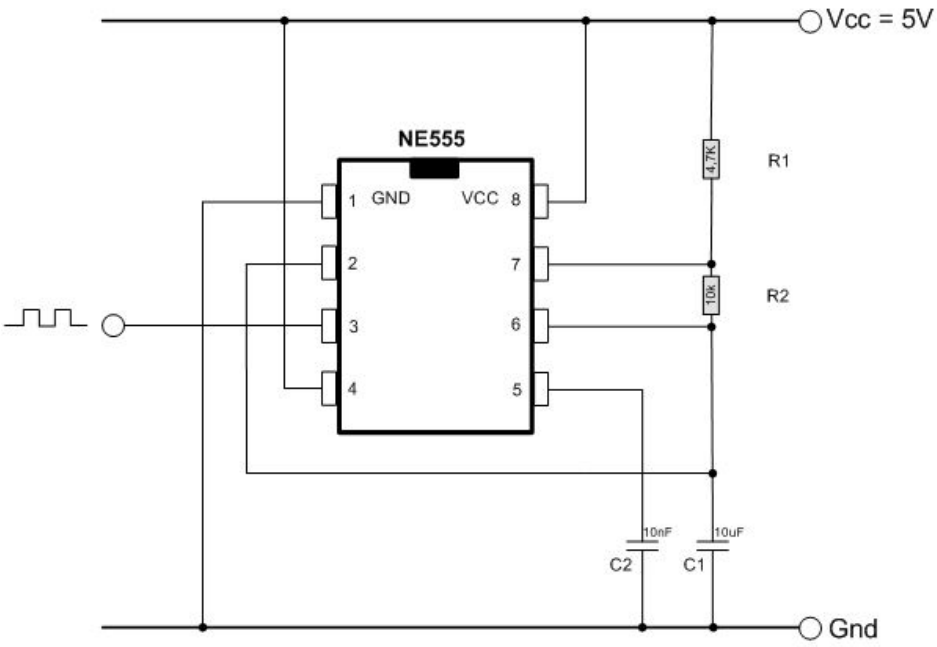
Mein Versuchsaufbau: Stromquelle 5,0 Volt DC (man kann auch viermal 1,5 Volt = 6Volt verwenden) Transistoren: BC 547 NPN (aus Restbeständen habe ich versuchsweise zwei 2SC1740 NPN genommen funktioniert auch!) . Also, wenn man irgendwelche zwei identische Kleinsignal-transitoren vom Typ NPN hat, dann dürfte es keine Probleme geben!!!



Schaltplan NR. 1

Zweite Methode

Mit dem Aufkommen der ersten technisch ausgereiften ICs (Integrierte Schaltungen), im Jahr1972, z.B. der N555 eröffneten sich völlig neue Aspekte und Anwendungsmöglichkeiten. Nur durch das Verwenden von variablen Widerständen (Potentiometer) und Kondensatoren in der Peripherie des 8 beinigen ICs, kann man ungleich lange Impulse und Pausen in einer sauberen z.B. Rechteckspannung (high-low, ohne ansteigende/fallende Flanke) produzieren. Spannungsversorgung 5 Volt (man kann den N555 aber auch bis 12 Volt betreiben)



Schaltplan NR.2

Je geringer der Widerstandswert des Widerstandes R2 ist, desto höher die gemessene Frequenz an der LED. Hierbei zeigt sich, dass die Grenze zwischen dem extrem schnellen Blinken und Dauerlicht für meine Augen etwa bei 25 HZ liegt. Und wie gut sehen Sie? Ein Sehtest der anderen Art!
Die neuen Begriffe: Frequenz, Taktgeber, Takt, 3/4-Takt, Wiener Walzer, Impulslänge, Länge der Pause...müssen erst einmal logisch sortiert werden; denn man kann die produzierte Frequenz eines N555 auch berechnen!
Das muss man sich aber auch nicht als Hobbyelektroniker antun, denn man kann bereits automatische Formelrechner zur Frequenzberechnung benutzen.
z.B. http://www.dieelektronikerseite.de/Tools/NE555.htm

Der aufmerksame Leser merkt sofort, hier findet der Einstieg in die Digitaltechnik statt, bzw. das elektronische Basteln mit tollen Möglichkeiten der Verkettung mit weiteren ICs (z.B. Treiber IC CD4017 für LEDs) findet nur noch an den Ausgängen (Pin 3) statt. Jedoch wie tatsächlich die Schaltung im inneren des ICs aussieht – das wird zur unwichtigen Nebensache....

Hauptsache das Basteln macht Spaß! Tipp. im Internet, hier Google, http: N555 Schaltungen oder N555 CD4017 ; dann unter Bilder findet man tolle Projekte zum Nachbauen.

Dritte Methode

In den 1990-er Jahren kamen die Mikrocontroller mit löschbaren und programmierbaren ROM (EEPROM) Speicher und Flash-Speicher auf. Bezogen auf unser Thema der Erzeugung von Blinklichtern bedeutet das für den Elektronikbastler: „Widerstände und Kondensatoren = unwichtig – heute nur noch PC und Programmier-sprache“.
Naja, ganz so einfach ist es dann doch nicht - jedoch an dieser Stelle eine umfassende Anleitung zu geben, würde den Rahmen des Newsletter sprengen!

Gestatten Sie mir, dass ich Ihnen am Beispiel des ATtiny13 (8Pins) der Firma Atmel die drei wichtigsten Programmiersprachen Bascom,C und Assembler zeige, die im Grunde alle nur eines bewirken : Eine Lampe (LED+Vorwiderstand) zum Blinken zu bringen.

Hardware:
Programmierboard : z.B. myAVR MK2 oder Atmel STK500

C-Programmcode

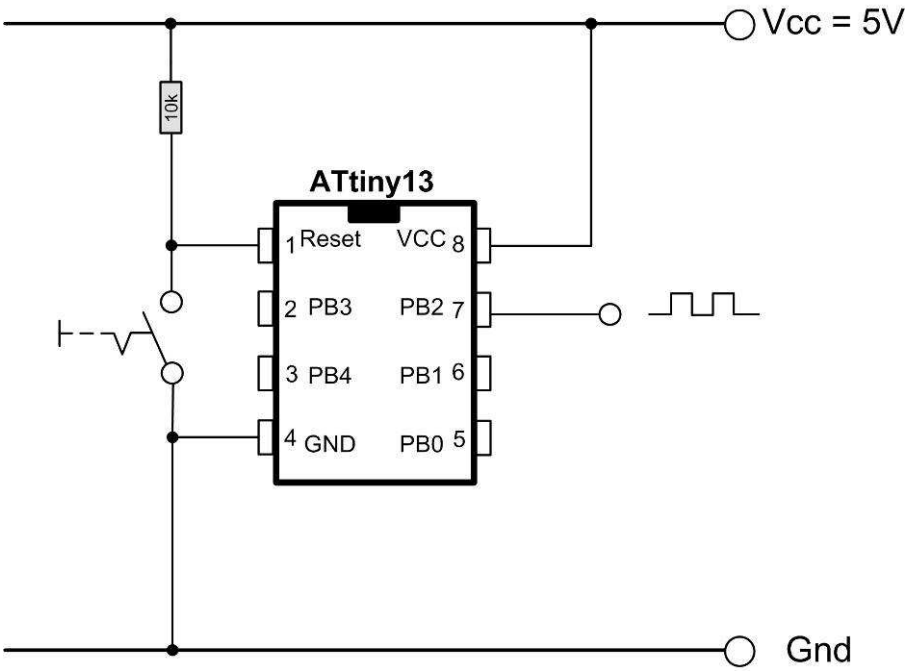
```
//-------------------------------------------------------------------------
// Title      : AVR C Grundgerüst für ATtiny13
//-------------------------------------------------------------------------
// Funktion   : Blinklicht an PortB.2
// Schaltung  : Sisy-Experimentierboard, Löbau
//-------------------------------------------------------------------------
// Prozessor  : ATtiny13
// Takt       : 3,6864 MHz
// Sprache    : C
// Date       : 5.11.2014
// Version    : AVR mikrocontroller in C programmieren...
// Autor      : Klaus Wellmann
//-------------------------------------------------------------------------

// Blink LED ATtiny13

#define F_CPU   3686400

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
  DDRB = 0b11111111;    // Datenrichtungsregister willkürlich
                       //PortB2 als output

  while (1)
  {
      PORTB = 0b0000100; _delay_ms(300);//Bit gesetzt,somit LED an
      PORTB = 0b0000000; _delay_ms(400);//Bit gelöscht,somit LED aus
  }
  return 0;
}
```

Bascom-Programmcode

```
' ATtiny 13 in Bascom ,  Funktion : LED blinkt an PortB.1
' Klaus Wellmann
' 05.11.2014

$regfile = "attiny13.dat"                                   'Chip festlegen
$crystal = 1200000                                          'Frequenz festlegen
$hwstack = 32
$swstack = 8
$framesize = 24
Config Portb = Output                                       'alle PortB auf Ausgang

Do                                                          'Schleifenanfang
  Portb.0 = 1                                               'Portb.1 auf high
  Waitms 200                                                'warte 200 ms Sekunden
  Portb.0 = 0                                               'Portb.1 auf low
  Waitms 200                                                'warte 200 ms Sekunden
Loop                                                        '200 ms Sekunde warten

End                                                         'Programmende
```

Assembler-Programmcode

```
;+----------------------------------------------------------------------------
;| Title     : Assembler Grundgerüst für ATtiny13
;+----------------------------------------------------------------------------
;| Funktion  : Attiny13 in Assembler
;| Schaltung : Blinkt an PortB.O
;+----------------------------------------------------------------------------
;| Prozessor : ATtiny13
;| Takt      : 3,6864 MHz
;| Sprache   : Assembler
;| Version   : an Sisy-Board ,Löbau
;+----------------------------------------------------------------------------
.include   "AVR.H"
; oder wer das AVR-Studio bevorzugt, dann lautet der
; include Befehl:
;.include "tn13def.inc"
;----------------------------------------------------------------------------
main:   ldi r16, 0xFF
        out DDRB,r16
         ldi r16, 0b00000001
        out PORTB,r16
         ldi r17,200

schleife1:  ldi r18,200 ; "200" ist eine relative Zeitangabe
schleife1.2: dec r18
      brne schleife1.2;falls nicht, dann Sprung zu Schleife 1.2
      dec r17
      brne schleife1  ;falls nicht, dann Sprung zu Schleife 1
;--------------------------------
ledaus: ldi r16, 0xFF
        out DDRB, r16
        ldi r16,0b00000000
        out PORTB, r16
         ldi r17,240; "240 ist eine relative Zeitangabe
schleife2: ldi r18,240
schleife2.1 :dec r18
          brne schleife2.1
          dec r17
          brne schleife2
;-----------------------------
          rjmp main;Springe zu main
```



Schaltplan Nr 3: Exemplarisch für den Programmcode, geschrieben in C

Welche der Versionen Ihnen am sympathischsten ist, weiß ich nicht - ich wünsche mir nur, dass Ihnen ein „Licht“ aufgegangen ist, was man so in der dunklen Jahreszeit alles Basteln kann.

Literaturangaben und Quellenangaben:

- Die 7 Weltwunder der Antike, Kai Brodersen
- Philips Experimentierkasten
- http://www.dieelektronikerseite.de

- AVR und Bascom, Stefan Hoffmann
- Attiny13, Atmega8 in C , Dr. Spanner
- http://www.elektronik-bastelkeller.de/

Ausblick auf die nächste Ausgabe des Newsletters

- Erfahrungen im 3D-Druck

- Projekt: Navigation in geschlossenen Räumen (Indoor-Navigation)
  (Theorie und Praxisansatz)

Mit freundlichen Grüßen
Die Robotniks
www.ps-robotics.de/