elektronisches Männchen

Karsten Müller
email@kmkcl.de

27. Juli 2013

Eine Version dieser Anleitung mit Bild gibt es auf: http://kmkcl.de/maennchen.html

Inhaltsverzeichnis

1 Vorwort

Das elektronische Männchen ist ein kleines Spielzeug, dass sich zum verschenken eignet. Die Materialkosten halten sich stark in Grenzen und trozdem kann dem Männchen durch die Software eine eigene Note gegeben werden.

Ich habe diese Anleitung mitverschenkt und versucht die Komponenten und Funktionsweise auf einem (hoffentlich) verständlichem Niveau für Interessierte zu erklären. Zum Teil leidet dadurch die Fachsprache etwas, wofür ich um Entschuldigung bitten möchte.

2 Funktionen

Das Männchen kann unter anderem ein Lied trällern und eignet sich als Würfel oder Taschenlampe. Die Funktionen werden mit der sogenannten Diagnosenadel aktiviert. Die Diagnosenadel ist das Ende des frei liegenden Drahtes. Die zugehörigen Diagnosepunkte sind die Knubbel aus Lötzinn an Becken, Knien und Füßen. Jeder Punkt löst eine andere Funktion aus:

Tabelle 1: Funktionen

Körperteil	Funktion
linkes Knie	Lied abspielen
linker Fuß	Würfel
rechter Fuß	Lauflicht
rechtes Knie	Zombie
Becken	Herzschlag

Anmerkung: Alle Angaben der Körperteile sind aus der Sicht des Männchens.

2.1 Würfel

Jeder Zahl von 1 bis 6 ist eine LED zugeordnet. Wird der Diagnosepunkt berührt, leuchten alle LEDs und es wird gewürfelt. Sobald der Diagnosepunkt losgelassen wird, leuchtet nur noch eine der LEDs und es ertönt abhängig von der gewürfelten Zahl ein kurzer Ton.

2.2 Lauflicht

Dabei blinken alle LEDs in einer gewissen Reihenfolge auf und das Männchen gibt kurze Pieptöne von sich. Die Funktion startet erst nach loslassen des Diagnosepunktes und kann durch eine andere beendet werden.

2.3 Zombie

Genau wie bei dem Lauflicht startet die Funktion erst nach loslassen des Diagnosepunktes und kann durch jede andere Funktion beendet werden. In dem Zombie-Modus leuchten die Augen unheimlich auf und die Taschenlampe in des Männchens rechter Hand spukt gespenstisch.

3 Die Elektronik

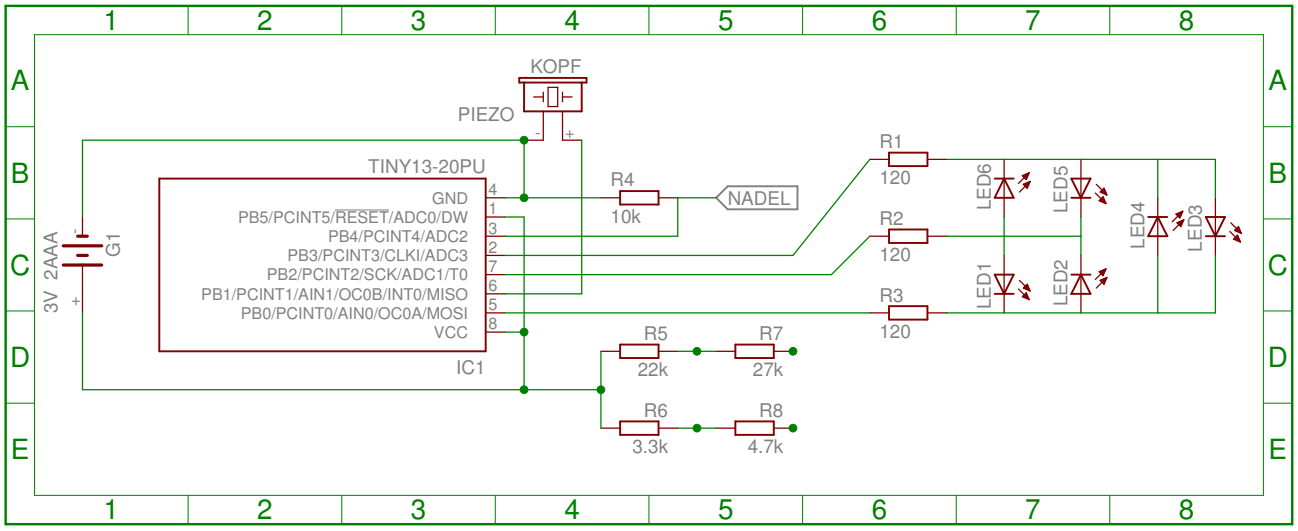

Abbildung 1: Schaltplan

Tabelle 2: Bauteile

Bauteil	Stelle
R1	rechter Unterarm
R2	linker Unterarm
R3	rechter Oberarm
R4	linker Oberarm
R5	rechter Oberschenkel
R6	linker Oberschenkel
R7	rechter Unterschenkel
R8	linker Unterschenkel
LED1	Penis
LED2	Herz rot
LED3	linkes Auge
LED4	rechtes Auge
LED5	Herz grün
LED6	rechte Hand

Angaben aus Sicht des Männchens

3.1 Körper

Das zentrale Element des Männchens bildet ein Mikrocontroller. Mikrocontroller sind programmierbare Bauelemente, die eine Vielzahl von Funktionen übernehmen können. Die Besonderheit liegt in der umfangreichen Peripherie, die neben einem Prozessor auf dem Chip integriert ist. Neben den notwendigen Elementen zum Betreiben ei nes Prozessors, wie Programmspeicher (Flash) und Arbeitsspeicher (SRAM) ist oft auch Elektronik für Takterzeugung und verschiedene Ein und Ausgänge, wie zum Beispiel AnalogDigitalWandler, vorhanden. So wird der externe schaltungstechnische Aufwand auf ein Minimum reduziert. Programmiert werden die Mikrocontroller in den gleichen Programmiersprachen wie Computerprogramme mit dem PC. Über einen speziellen Programmieradapter wird das Programm auf den Mikrocontroller übertragen. Der hier verwendete Mikrocontroller ist ein Attiny13. Er bietet nur einen kleinen Programmspeicher (1kB) und wenige Funktionen, ist dafür aber relativ günstig und klein. Von den Funktionen werden der Taktgenerator, Zähler und der Analog-Digital-Wandler benutzt. Letzteres ist quasi ein Messgerät für Spannungen. Der verwendete interne Takt beträgt 1,2 MHz.

3.2 Beine

Die Beine bestehen aus den Widerständen R5 bis R8, wobei R5 und R6 an Plus angeschlossen sind. Mit einem Messgerät ließe sich an allen Punkten nur die normale Betriebspannung messen, da kein Strom durch die Widerstände fließt. Werden die Punkte allerdings mit der Nadel berührt, so fließt ein Strom über den Widerstand R4 zu Minus. Da jeder Widerstand anders ist, ergibt sich immer eine andere Reihenschaltung, je nachdem welcher Punkt berührt wird. Somit fällt an dem Widerstand R4 immer eine andere Spannung ab, die vom Mikrocontroller erfasst werden kann. Die Widerstände sind so berechnet, dass die entstehenden Spannungen in etwa gleichmäßigen Abständen den kompletten Bereich der Betriebsspannung abdecken. Der Widerstand R4 sorgt weiterhin dafür, dass am Eingang des Mikrocontrollers immer eine definierte Spannung anliegt.
Die Füße bestehen im Übrigen aus kleinen Spulen. Auf Gleichspannung haben sie keine Wirkung und können als Drähte angesehen werden.

3.3 LEDs

Die LEDs sind in einer speziellen Art und Weise mit dem Microcontroller verbunden. Diese Verschaltung wird Charlie-Plexing genannt. Charlieplexing ist die effektivste Methode um mit wenigen Ein- und Ausgängen die meisten LEDs anzusteuern.
Dies funktioniert so:

Eine LED lässt Strom nur in eine Richtung durch. Werden 2 LEDs entgegengesetzt parallel verschaltet, so lässt sich über die Kombination von Plus und Minus steuern, welche LED leuchten soll. Weiterhin ist der Mikrocontroller so beschaffen, dass ein Anschluss noch einen 3. Zustand haben kann, bei dem weder Plus noch Minus angeschlossen sind. Somit ist es möglich jede LED einzeln anzusteuern. Bei genauer Betrachtung fällt jedoch auf, dass immer 2 LEDs in Reihe parallel zu einer anderen LED liegen. Damit nicht alle 3 LEDs leuchten, muss die einzelne LED die Spannung so weit absenken, dass die zwei LEDs in Reihe zu wenig Spannung zum leuchten bekommen. LEDs leuchten nämlich erst ab einer gewissen Spannung, die vorallem von der Farbe, aber auch der Technologie abhängt. Anders als bei einem Widerstand kann die Spannung einer LED als konstant angesehen werden. Die Differenz zu der allgemeinen Betriebsspannung muss an einem Vorwiderstand abfallen.

3.4 Kopf

Der Kopf wird durch einen Piezo-Schallwandler dargestellt. Dieser besteht aus einem hauchdünnen Quarzplättchen mit Metallflächen auf beiden Seiten. Der Piezo-Effekt bechreibt ein Phänomen, dass sich einige Kristalle, wie auch der Quarz bei Anlegen einer elektrischen Spannung verformen und umgekehrt bei Verformung eine Spannung erzeugen. Die Schwingung wird dann auf einen Resonanzkörper übertragen. Elektrisch gesehen ist der Schallwandler ein Kondensator, der Ladungsträger speichern kann. Im Gegensatz zu normalen Lautsprechern fließt immer nur dann ein Strom, wenn sich das angelegte Signal ändert, er ist somit energiesparender.

4 Das Programm

Das Programm ist in C geschrieben. Diese Programmiersprache zählt mit zu den ältesten und dennoch am weitest verbreiteten Programmiersprachen. Im folgenden eine kurze Beschreibung der einzelnen Programmteile.

4.1 Initialisierung

Programmzeile 58-77

Alle Ein- und Ausgänge und die meisten Funktionen der Mikrocontroller werden über sogenannte Register eingestellt bzw. ausgeführt. Ein Register ist als Variable einer ganzen Zahl zu sehen, die in binärer Form gespeichert wird. Wird so zum Beispiel in die Variable für die Ausgänge (PORTB) die Zahl 3 geschrieben, so werden nach der binären Darstellung (3=11) die Anschlüsse Null und Eins auf Plus gelegt. Ein anderes Beispiel ist der Analog-Digital-Wandler. Dort beginnt der Wandler zu arbeiten, nachdem in einem speziellen Register eine bestimmte Zahl geschrieben wurde. Wenn er fertig ist, steht eine bestimmte Zahl in einem anderen Register. Für die Funktion des Programmes müssen eine ganze Reihe mehr Einstellungen getroffen werden.

4.2 Analog-Digital-Wandler

Programmzeile 19-35, 80-94

Der Analog-Digital-Wandler ist das zentrale Element in dieser Anwendung, der es ermöglicht mit nur einem Anschluss mehrere Zustände zu identifizieren. Der AD-Wandler ist im prinzip ein Spannungsmessgerät. Zwischen Minus und Plus kann dieser 1024 Stufen an Spannungswerten erfassen. Diese Messung geht so schnell, dass bei Berührung oder loslassen eines Kontaktpunktes nicht vorhersehbare andere Spannungen gemessen werden. Durch einfügen einer 200ms Pause und Mittelwertbildung aus jeweils 10 Messungen wird dieses Problem minimiert.

4.3 Lied

Programmzeile 97-122

Das Abspielen des Liedes ist mit der aufwändigste Programmteil. Aus dem Programmspeicher muss die Frequenz und Dauer der nächsten Note geholt werden, die Frequenz muss abgespielt und gleichzeitig auf die Dauer geachtet werden.

Die Frequenz wird über den internen Zähler erzeugt. Dieser zählt immer von 0 bis zu einer gewünschten Zahl, wobei der Takt kann in sehr groben Stufen eingestellt werden kann. Wird die obere Grenze erreicht, beginnt der Zähler wieder bei 0 und der Ausgang des Schallwandlers wechselt von Plus nach Minus oder andersrum. Dadurch entsteht das Rechtecksignal und der damit typische Klang. Dieses spezielle Verhalten des Zählers muss nicht extra programmiert werden, da es eine integrierte Funktion des Mikrocontrollers ist, die nur richtig konfiguriert werden muss.

Die Dauer des Tones wird durch eine Schleife bestimmt, die nur aus “Tue nichts”-Anweisungen besteht, die unterschiedlich oft ausgeführt wird.

4.4 Würfel

Programmzeile 124-136

Bei der Würfelfunktion werden in sehr schneller Folge die Zahlen von 1 bis 240 durchgezählt. Der Rest, der beim Teilen durch 6 entsteht, entspricht der gewürfelten Zahl. Dieses Vorgehen erzeugt eine höhere Gleichmäßigkeit im Programmablauf und somit einen besseren “Zufall”als jeweils von 1 bis 6 zu zählen. Jeder Zahl ist eine LED zugeordnet und durch die Trägheit des Auges scheinen zunächst alle LEDs gleichzeitig zu leuchten. Erst wenn der Diagnosepunkt nicht mehr berührt wird, hört das Zählen auf und die zuletzt leuchtende LED bleibt weiterhin an. Mit einer einfachen Formel wird zu jeder Zahl ein Ton errechnet, der dann für eine kurze Zeit wiedergegeben wird.

4.5 Lauflicht

Programmzeile 84-93

Das Lauflicht sollte auch ohne ständige Berührung weiterlaufen, daher weicht das Programm etwas von dem Würfelprogramm ab. Aktiviert wird das Lauflicht durch eine Berührung und loslassen des entsprechenden Diagnosepunktes. Für die Verzögerung des Laufens sorgt dann die Funktion des Analog-Wandlers, die auf ein Berühren eines Diagnose-Punktes wartet und für jedes stabile Messergebnis eine Weile benötigt. Genau wie bei dem Würfel wird bei jeder LED ein kurzer Ton erzeugt.

4.6 Zombie

Programmzeile 141-180

Der Zombie erzeugt zwei verschiedene Dreiecksschwingungen. Dabei zählt eine Variable immer wieder von 0 bis 150 und wieder zurück und eine von 0 bis 127 und zurück. Abhängig von diesen zwei Zahlen bleiben die LEDs während eines annähernd gleichen Zeitintervalles unterschiedlich lang eingeschaltet. Da das ein- und Ausschalten so schnell geht, dass es nicht wahrnehmbar ist, scheinen die LEDs einfach nur unterschiedlich hell zu leuchten. Dieses Verfahren heißt Pulsweitenmodulation (PWM) und taucht bei nahezu allen Anwendungen auf, bei denen LEDs eine unterschiedliche Helligkeit haben sollen.

4.7 Quelltext

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

Project : elektronisches Maennchen
Author  : Karsten Mueller
Date    : 27.07.2013

*******************************************/
#define F_CPU 1000000UL
#include  <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>

#include "lieder.h"

// Analog-Digital-Wandler
// 10 Messungen durchfuehren
// Ergebnis zuordnen
char ADCread()
{
   char i;
   uint16_t result=0;
   for(i=0; i<10; i++)
     {
       ADCSRA |= (1<<ADSC);
       while (ADCSRA & (1<<ADSC) ) {}
       result += ADCW;
     }
  if(result<101*10) return 0;
  if(result<204*10) return reFuss; // 1
  if(result<409*10) return reKnie; // 2
  if(result<700*10) return liFuss; // 3
  if(result<900*10) return liKnie; // 4
  return Becken; // 5
}

void LEDs(char leds)
{
   switch(leds)
     {
        case Penis  : DDRB=(1<<SPEAKER)|(1<<PB0)|(1<<PB2); PORTB=(1<<PB2); break; //LED 1
        case rHerz  : DDRB=(1<<SPEAKER)|(1<<PB0)|(1<<PB2); PORTB=(1<<PB0); break; //LED 2
        case liAuge : DDRB=(1<<SPEAKER)|(1<<PB0)|(1<<PB3); PORTB=(1<<PB3); break; //LED 3
        case reAuge : DDRB=(1<<SPEAKER)|(1<<PB0)|(1<<PB3); PORTB=(1<<PB0); break; //LED 4
        case gHerz  : DDRB=(1<<SPEAKER)|(1<<PB2)|(1<<PB3); PORTB=(1<<PB3); break; //LED 5
        case reHand : DDRB=(1<<SPEAKER)|(1<<PB2)|(1<<PB3); PORTB=(1<<PB2); break; //LED 6
    }
}

void warte(char zeit)
{
   while ( zeit-- )
     _delay_us(2);
}

void main(void)
{
   unsigned char modus = 0;
   unsigned char index=0;

   //Ein-/Ausgaenge konfigurieren
   DDRB=(1<<SPEAKER);
   PORTB=0x00;

   // Timer/Counter 0 konfigurieren
   // CTC mode, toggle OC0B bei compare match
   // Takt: intern, prescale 8
   TCCR0A=(1<<WGM01) | (1<<COM0B0);
   TCCR0B=(1<<CS01);
```

```
70      TON=Silence;
71
72      //ADC konfigurieren
73      ADCSRA= (1<<ADEN);
74      ADCSRA|= (1<<ADPS1) | (1<<ADPS0);
75      ADMUX=2; //PB4
76      ADCread();
77
78      while (1)  //Hauptprogramm
79        {
80          while ( ADCread() == 0 ) //auf Beruehrung eines Diagnosepunktes warten
81            {
82              _delay_ms(200);
83
84              if ( modus == reFuss ) //Modus Lauflicht- nutzt delay gleich mit
85                {
86                  index++;
87                  if ( index > 6 )
88                     index=1;
89                  LEDs(index);
90                  TON=155-index*20;
91                  _delay_ms(20);
92                  TON=Silence;
93                }
94            }
95          modus=ADCread();
96
97          if ( modus == liKnie ) //Lied abspielen
98            {
99              char lang;
100
101             for ( index=0; index<lied_laenge; index++)
102               {
103                 //Note laden
104                 TON  = lied[index][0];
105                /* if ( TON < 45 )        //fuer Frequenzen unter 278 Herz (z.B. C1 und
                       tiefer)
106                   TCCR0B |= (1<<CS00);
107                 else
108                   TCCR0B &= ~(1<<CS00);*/
109                 lang = lied[index][1] * 6;
110
111                 //Laenge der Note abwarten
112                 for ( lang; lang; lang-- )
113                   _delay_ms(20);
114
115                 TON=Silence;
116
117                 //kurze Pause zwischen Noten
118                 LEDs( rHerz );
119                 _delay_ms(30);
120                 PORTB=0;
121               }
122           }
123
124         if ( modus == liFuss ) //Wuerfel
125           {
126             while ( ADCread() > 0 )
127               {
128                 index++;
129                 if ( index > 240 )
130                   index=1;
131                 LEDs( index%6+1 );
132               }
133             TON=135-(index%6)*20;
134             _delay_ms(200);
135             TON=Silence;
136           }
137
138         if ( modus == reFuss ) //Lauflicht, siehe ADC
139           _delay_ms(500);
140
```

```
141         if ( modus == reKnie ) //Zombie
142           {
143               _delay_ms(500);
144               char up=0;
145               index=0;
146               char lampe=0;
147
148               while( ADCread() == 0 ) //erzeugt mit zwei Dreiecksschwingungen PWM fuer LEDs
149                 {
150                    up++;
151                    if ( up == 5 || index == 0 ) //hochzaehlen
152                      {
153                        index++;
154                        up=0;
155                      }
156                    if ( up == 15 || index == 150) //runterzaehlen
157                      {
158                         index--;
159                         up=10;
160                      }
161                    if (lampe < 128 ) //Bit 7->hoch bzw. runterzaehlen, zweite Schwingungs
162                      lampe++;
163                    else
164                      {
165                         if ( lampe == 128 )
166                           lampe=255;
167                         if ( lampe == 129 )
168                           lampe=1;
169                         lampe--;
170                      }
171                    LEDs( reAuge );
172                    warte(index);
173                    LEDs( liAuge );
174                    warte(index);
175                    LEDs( reHand );
176                    warte( 80 - MAX( index/2 , 25 ) + ( lampe&127 )/4 );
177                    PORTB=0;
178                    warte(200-index); //kein echtes PWM, da PWM Frequenz nicht gleich
179                 }
180           }
181
182         if ( modus == Becken ) //Herzschlag
183           {
184                 index++;
185               if ( index == 10 )
186                 {
187                    LEDs( rHerz );
188                    TON=255-C2;
189                 }
190               if ( index > 11 )
191                 {
192                    LEDs( gHerz );
193                    index=0;
194                    TON=255-C2;
195                 }
196               _delay_ms(40);
197               TON=Silence;
198           }
199     }
200 }
```