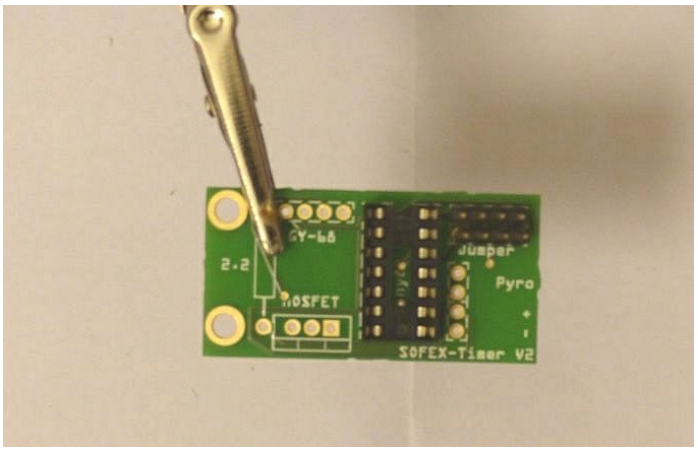
Zusammenbau des „SOFEX-Timers“

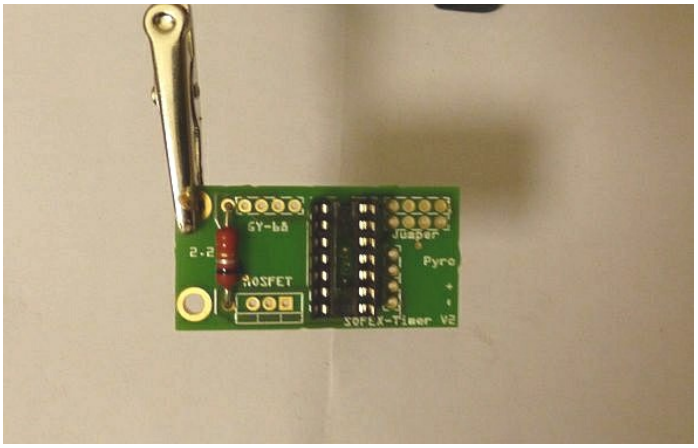
Wichtiger Hinweis: Den MOSFET fassen wir zunächst gar nicht an!!!

Die Platine wird so gelegt, dass unten rechts die Aufschrift „SOFEX-Timer“ lesbar ist.

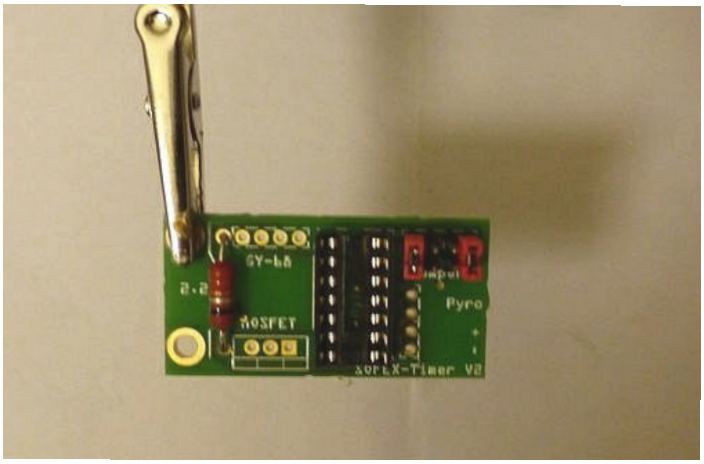
Als erstes wird der 14-polige IC-Sockel eingelötet. Dabei darauf achten, dass die Aussparung in der Mitte oben ist!



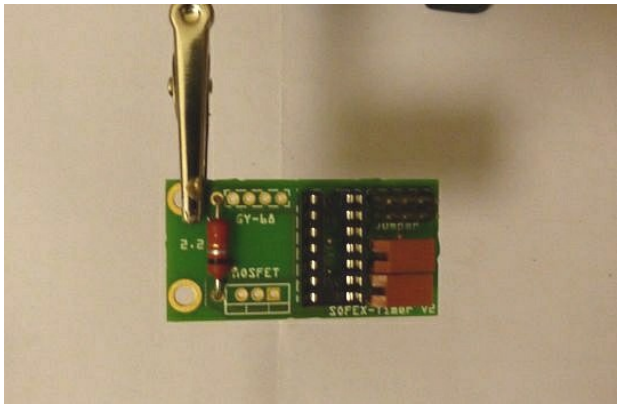
Danach wird der Widerstand eingelötet und die überstehenden Enden abgeknipst.



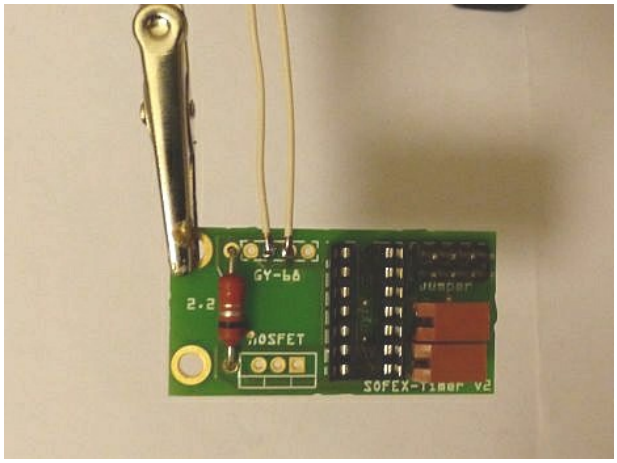
Jetzt sind die Pins für die Jumper dran. Es geht am einfachsten, wenn man den linken und den rechten Jumper vor dem Einlöten auftsteckt.



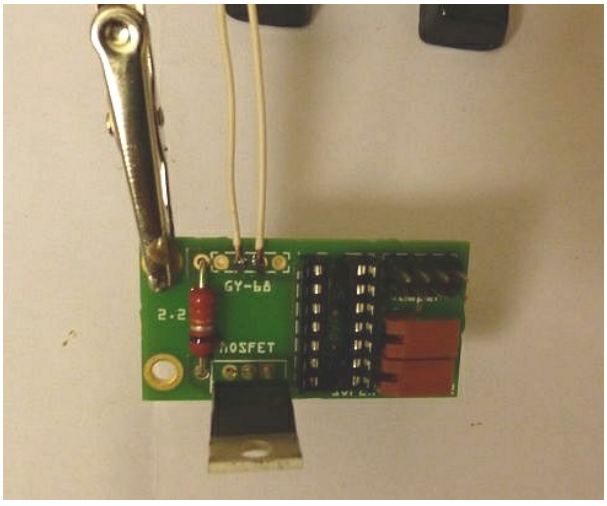
Als nächstes beide 2-poligen Header für die Steckverbinder einlöten.



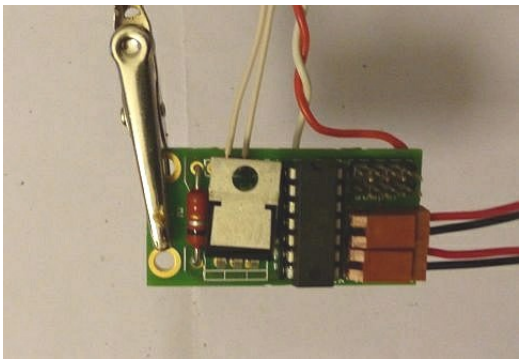
Die beiden Kabel, die als Abreißkontakt dienen, werden an die beiden mittleren Ösen angelötet, und zwar dort, wo die Beschriftung GY-68 steht.



Der MOSFET kommt zuletzt. Die Dinger reagieren sehr empfindlich auf statische Entladungen! Wir fassen sie so wenig wie möglich an. Wenn jemand merkt, dass die Kombination Teppichboden/Schuhe statische Aufladung bewirkt, sollte er ihn gar nicht anfassen, sondern lieber jemand anders!
Der MOSFET wird so eingelötet, dass seine Beschriftung nach oben zeigt.



Der MOSFET wird dann nach oben umgebogen:



Der Prozessor Attiny84 wird ganz zum Schluss eingesetzt. Dabei muss bei Draufsich auf die Platine die Kerbe nach oben zeigen! Erst leicht einsetzen und prüfen, ob alle Beinchen richtig sitzen, dann erst fest eindrücken.

Test

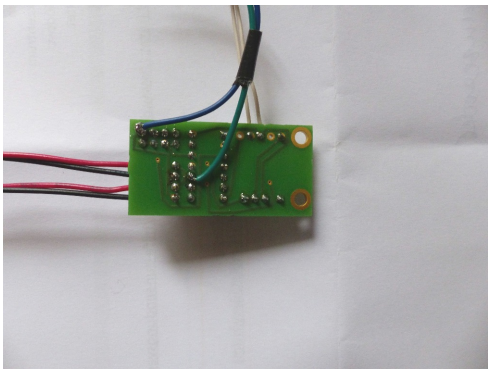
Testen können wir, indem wir
- als Abreißkontakt einen Jumper mit den beiden Kabeln (an GY-68) verbinden
- einen Lipo (nur 1S-Lipo, also 3,7 V Nennstrom!) an die mit „+“ und „-“ gekennzeichneten Stifte des unteren Steckverbinders anschließen; dabei auf richtige Polung achten!!!
- und eine LED an den oberen Steckverbinder: plus an den oberen, minus an den unteren Stift

Unterschiedliche Jumper-Kombinationen müssen dann zu entsprechender Wartezeit bis zum Einschalten der LED führen.

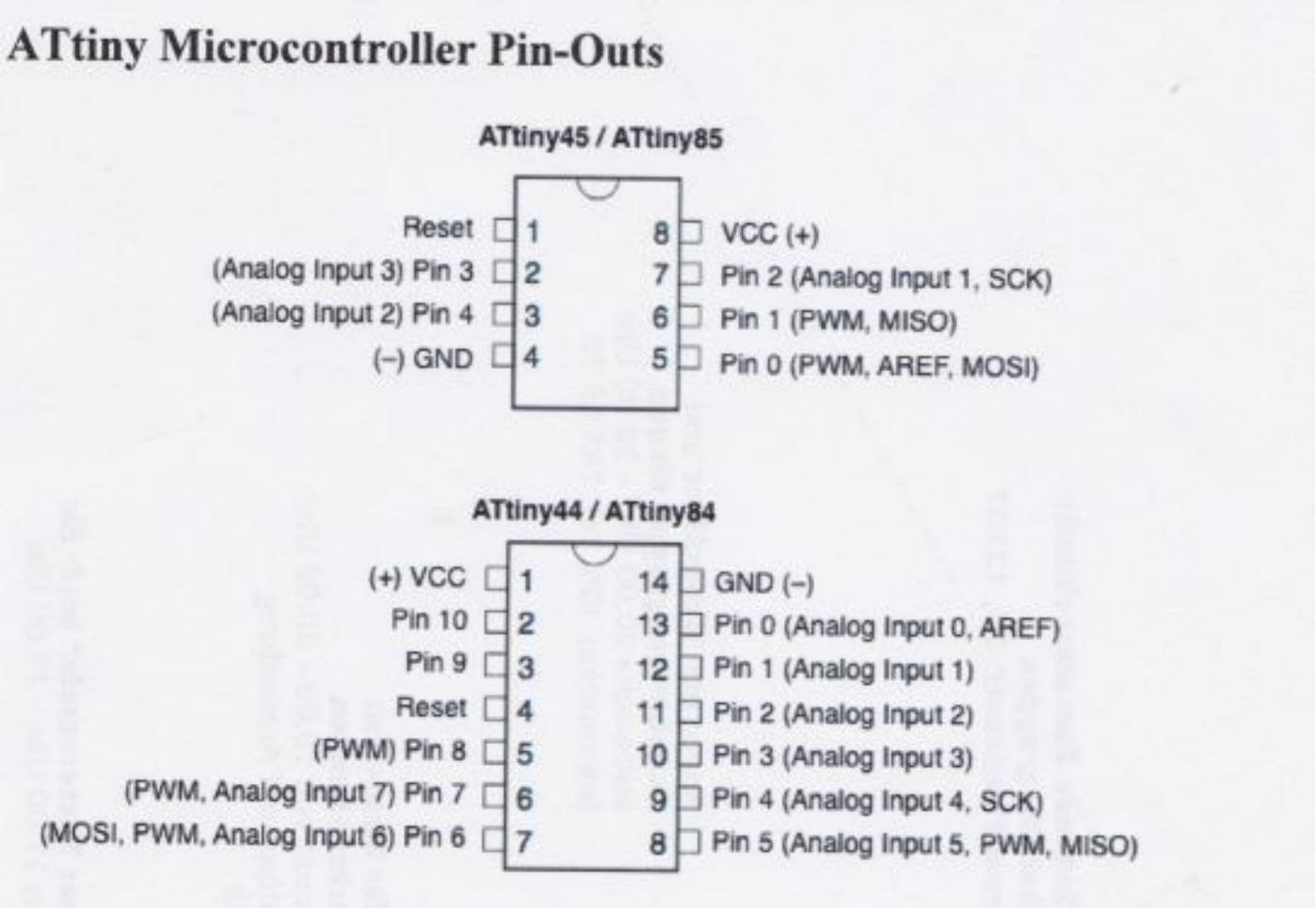
Wird ohne geschlossenen Abreißkontakt eingeschaltet, darf nichts passieren, also die LED nicht aufleuchten. Es dient dazu, dass nicht aus Versehen auf der Rampe kurz nach Einschalten eine Auslösung erfolgt und der Raketeur womöglich verletzt wird!

Test-Ausgaben

Um die im Programm einegefügten Test-Ausgaben nutzen zu können, löten wir uns zwei Kabel an Masse (-) und Pin3:



Die Test-Ausgaben erfolgen also auf Pin3. Wir können z. B. einen Arduino Nano nehmen, in den eine Serial-Relay-Software geladen wurde, ihn entsprechend mit den Kabeln verbinden und diesen wiederum an den PC anschließen. Im Konsolfenster erscheinen dann unsere Ausgaben. Darauf achten, dass das Konsolfenster auf 9600 Baud eingestellt ist!



HINWEISE
Ein Abreißkontakt muss vorhanden sein, der zunächst geschlossen ist und beim Start der Rakete öffnet. Typischerweise verwendet man dafür eine Büroklemmer.

ACHTUNG: Wenn der Kontakt beim Einschalten nicht geschlossen ist, wird keine Auslösung erfolgen, eine Rakete könnte daher später abstürzen. Es dient dazu, eine Auslösung kurz nach Einschalten – also noch auf der Rampe – zu vermeiden, da der Raketeur noch nahe der Rampe stehen könnte.

An die Pyro-Kontakte dürfen nur sogenannte "Brückenanzünder A" angeschlossen werden. Es sind keine Maßnahmen gegen Vertauschung der Pole beim Anschluss einer Batterie vorhanden. Die Batteriespannung kann zwischen 3,7 und 5,5 V liegen. Daher ist ein 3,7 V Lipo (1S) dringend empfohlen. Auch gegen den Anschluss einer Batterie mit zu hoher Spannung gibt es keinen Schutz!

DISCLAIMER
Der SOFEX-Timer und die zugehörige Software sind Teil eines Einführungskurses in die Programmierung von ATtinies. Es ist Kursmaterial, kein kommerzielles Produkt! Ein Echt-Einsatz geht auf Risiko des Benutzers. Insbesondere darf es nicht für Mid- oder High-Power-Raketen verwendet werden, da wichtige Eigenschaften fehlen, wie
- Möglichkeiten, einen Arming-Schalter einzusetzen
- Durchgangsprüfung des Brückenanzünders
- Maßnahmen gegen Anschluss einer Batterie (eines Akkus) mit falscher Polung
- Maßnahmen gegen Anschluss einer Batterie (eines Akkus) mit falscher Spannung

Obwohl das Programm sorgfältig getestet wurde, übernehme ich keine Verantwortung für mögliche Schäden, die durch den Einsatz der Software und/oder der Hardware entstehen können. Es sei darauf hingewiesen, dass Batterien, speziell Lipo-Akkus heiß werden können und in solch einem Fall Feuer fangen können. Auch können Raketen Schaden anrichten an Personen und Sachen, falls sie in unvorhergesehener Weise abstürzen.

SOFEXTimer | Arduino 1.8.3

Datei Bearbeiten Sketch Werkzeuge Hilfe

SOFEXTimer

```
/*
 SOFEXTimer Software Copyright (C) Hans-Joachim Oelkers 2017.

 Timer starts when liftoff occurs. This is indicated by a contact that is opened at lauch.
 When time elapsed, an igniter is burnt.
 Can be used e.g.
 - instead of deploying the parachute by motor exhaust
 - for an airstart

 HINTS
 A contact has to be provided that is closed at first and opens at lauch. Typically a paperclip is
 used for this purpose.

 ATTENTION: If the contact is not closed at switch-on, nothing will happen, so a rocket may fall down later.
 This is to avoid burning the igniter soon after switch-on, when a person doing some handling of the rocket
 maybe stands close to the lauchner!

 To the Pyro contacts only a so-called 'bridge igniter A' may be connected.
 There is no protection against connecting a battery the wrong way, i.e. interchanging + and - pole!
 The battery voltage may have 3.7 to 5.5 V, Therefore, a 3.7 V (1S) LiPo battery is strongly recommended.
 There is no provision against damage by connecting a battery with a voltage higher than 5.5 V!

 DISCLAIMER
 This program is part of an introduction course into ATtiny programming. The SOFEX-Timer therefore is
 course material, not a commercial product! Using it is at the user's own risk. Especially it must not
 be used with Mid or High Power Rockets, as it lacks important features like
 - Arming Switch provision
 - check of igniter continuity
 - provision for connecting a battery the wrong way, i.e. interchanging + and - pole
 - provision against connecting a battery with a too high voltage etc.
 Although the program was carefully written and tested, I don't take any responsibility for
 possible damages that may occur by using the software and/or hardware.
 Be warned that batteries, especially LiPo batteries may get hot and possibly burn in such a case.
 Also, rockets may cause damage to people and any belongings in case they fall down in an unexpected way.
*/
```

```
#include <SoftwareSerial.h>

// Definitions for Serial debug
#define rxPin 7
#define txPin 3

SoftwareSerial mySerial(rxPin, txPin);    // for test print

boolean contactWasClosed = false; // status of the contact to open at launch
boolean startDetected = false;    // start has not yet been detected
boolean timeElapsed = false;      // time has not yet been elapsed

long startTime;
long pyroTimeSec;                 // time read from jumper settings in secs
long pyroTime;                    // time in ms


/*          ATtiny84
 *        +----  ----+
 *    VCC |          | GND
 * Pin 10 |          | Pin 0
 *  Pin 9 |          | Pin 1
 *  Reset |          | Pin 2
 *  Pin 8 |          | Pin 3
 *  Pin 7 |          | Pin 4
 *  Pin 6 |          | Pin 5
 *        +----------+
 */

// define pins used
const int pinTime3 = 2;
const int pinTime2 = 1;
const int pinTime1 = 9;
const int pinTime0 = 10;
const int pinPyro = 0;
const int pinContact = 4;

// digits of the jumper settings
int digit3;
int digit2;
int digit1;
int digit0;
```

```
void setup()
{
  mySerial.begin(9600);

  // just to clear a possibly connected 2-line display
  mySerial.println();
  mySerial.println();

  //Initialise input pins
  pinMode(pinTime3, INPUT_PULLUP);
  pinMode(pinTime2, INPUT_PULLUP);
  pinMode(pinTime1, INPUT_PULLUP);
  pinMode(pinTime0, INPUT_PULLUP);
  pinMode(pinContact, INPUT_PULLUP);
  pinMode(rxPin, INPUT);              // we don't use the RX pin, just to show it

  // Initialise output pins
  pinMode(pinPyro, OUTPUT);
  pinMode(txPin, OUTPUT);


  //read time value from jumper settings
  if (digitalRead(pinTime3) == 0)
    digit3 = 1;
  else
    digit3 = 0;
  if (digitalRead(pinTime2) == 0)
    digit2 = 1;
  else
    digit2 = 0;
  if (digitalRead(pinTime1) == 0)
    digit1 = 1;
  else
    digit1 = 0;
    if (digitalRead(pinTime0) == 0)
    digit0 = 1;
  else
    digit0 = 0;
```

```
  pyroTimeSec = (8 * digit3 + 4 * digit2 + 2 * digit1 + digit0);  // time in s
  pyroTime = pyroTimeSec * 1000;                                  // time in ms
  mySerial.print("Time(s) = ");     // for test
  mySerial.println(pyroTimeSec);       // for test

  if (digitalRead(pinContact) == 1)
  {
    // contact was open at start! we do nothing to avoid early pyro ignition!
    contactWasClosed = false;
  }
  else
    contactWasClosed = true;
}

void loop()
{
  if (contactWasClosed)             // avoid ignition if contact open at start
  {
    if (!startDetected)
    {
      if (digitalRead(pinContact) == 1)
      {
        startDetected = true;       //contact opened: start has been detected
        startTime = millis();       //now time is running
        mySerial.println("Contact opened!");   // for test
      }
    }
    if (startDetected && (!timeElapsed) && ((millis() - startTime) > pyroTime))
    {
      // time has elapsed now, we want to do the ignition just once
      mySerial.println("Ignition!");    // for test
      digitalWrite(pinPyro, HIGH);
      delay(2000);                      // in general, we should not delay the
                                        // loop() routine for such a long time,
                                        // but this is the last thing we do ...
      digitalWrite(pinPyro, LOW);
      timeElapsed = true;
    }
  }
```