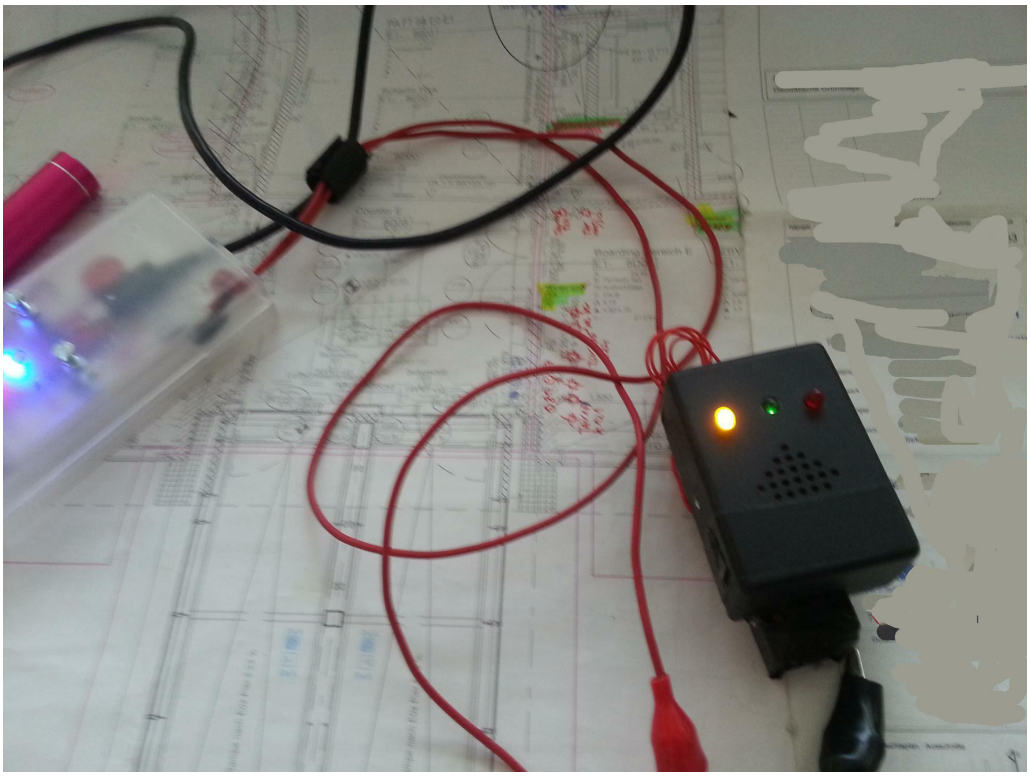
Adventskalender- Wettbewerb 2015/16

Kabel – Sucher

Von: Frank Schacht



Mein Projekt zum diesjährigen Wettbewerb ist ein Kabelsuchgerät, unter Fachleuten wird es oft auch als „Wobbler“ bezeichnet, wohl weil es in der Regel einen wechselnden Piepton abgibt. Nun was ist das, was macht man damit? Kurz gesagt: Kabel suchen! z.B. versteckt in der Wand (unter Putz) Das kann man auch mit anderen Geräten. Geräte, wie meines hier verwenden in der Regel Profis, wenn man z.B. ein Gebäude aus- oder umbaut und keine aktuellen Pläne vorhanden hat. Dann gibt es eine mehr oder weniger große Menge an Kabeln, wo selbst der Elektroinstallateur rätselt, wo geht dieses oder jenes Kabel hin. Dann wird der Elektriker im Normalfall einen solchen Kabelsucher zu Rate ziehen.

Hier will ich nur sehr kurz erklären wie das geht: Das Gerät besteht normalerweise aus zwei Teilen, 1. einem Sender, der an das zu suchende Kabel angeschlossen wird und einen modulierten (gewobbelten) Ton auf das Kabel gibt. 2.:Der Empfänger, der diesen Ton dann mehr oder weniger stark empfängt und wieder gibt. Die Suche sollte dann der Art laufen, Ton leise = Kabel weiter weg, Lauter Ton = gesuchtes Kabel (in der Wand oder im Bündel auf der Kabeltrasse) gefunden.

Was selbst Profigeräte nicht können, bei Erfolg oder Abbruch der Suche auf Fernschaltbefehl ausschalten, oder auf ein weiteres Kabel umschalten. Profigeräte können je nach Ausführung durch Umschalten bedingt zur Durchgangsprüfung verwendet werden. Aber auch nicht aus der Ferne! Wenn ich mit dem Wobbelton mein Kabel gefunden habe, muß ich zurück gehen und umklemmen

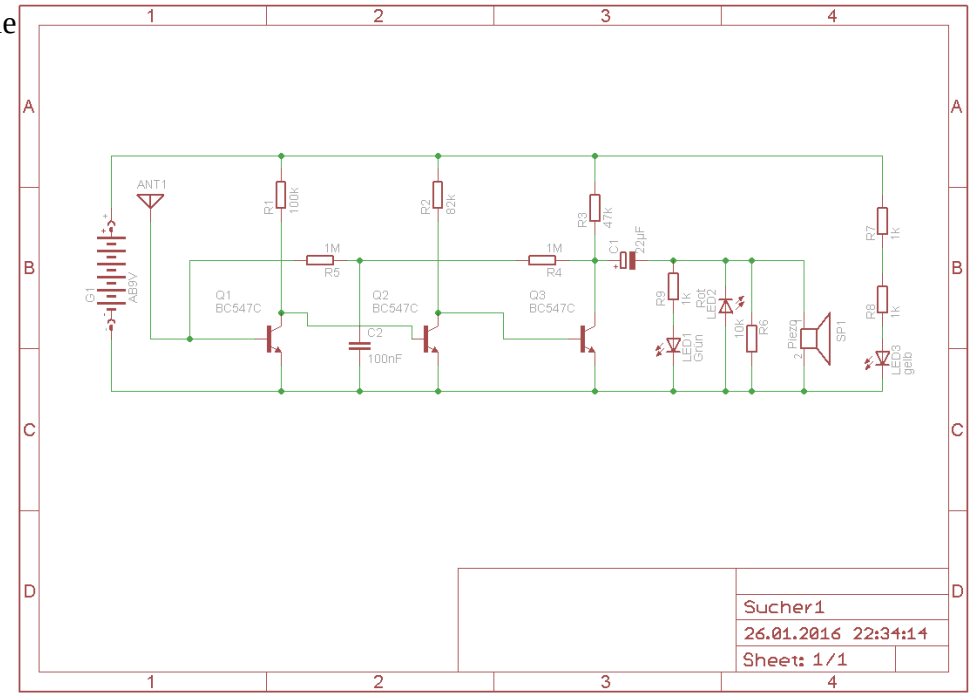
bzw. umschalten auf Durchgangsprüfung, da der Wobbelton bei dicht aneinander liegenden Kabeln gern auch auf Nachbarkabeln zu hören ist. Das macht man dann sogut wie immer zu zweit, da ich, als Monteur nicht das Piepsen des Prüfers über z.B. 50m hören kann.

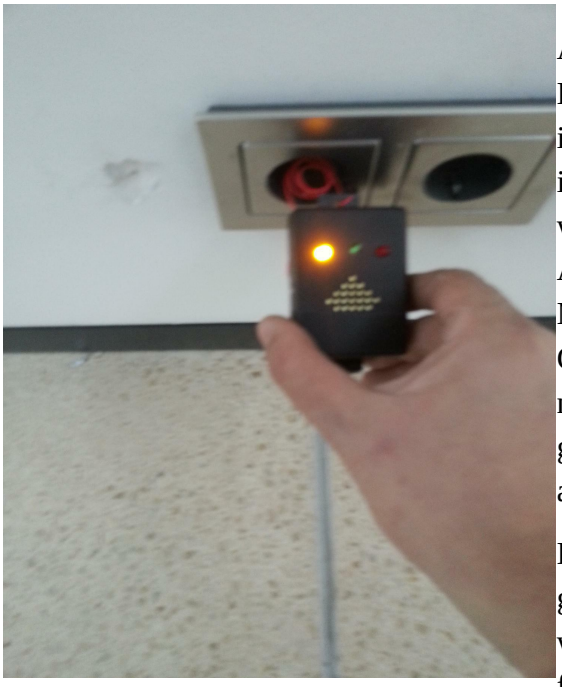
Mein Gerät wurde also Zweiteilig:

1. Der Empfänger:

Das Schaltbild erinnert sehr stark an den Elektrosmog-Sensor vom Tag 22, aus dem Conrad Adventskalender. Tatsächlich fand ich diese Schaltung am Überzeugendsten und erweiterte sie nur noch um den Piezo- “Lautsprecher“ und später noch um einen Widerstand zum entladen des Piezos. Eine Bereitschaftsanzeige spendierte ich der Schaltung aus der übrigen (gelben) LED und 2 x 1k Widerstand, da ich nicht gern unverhofft mit leerer Batterie da stehen mag.

Hier die Schaltung:

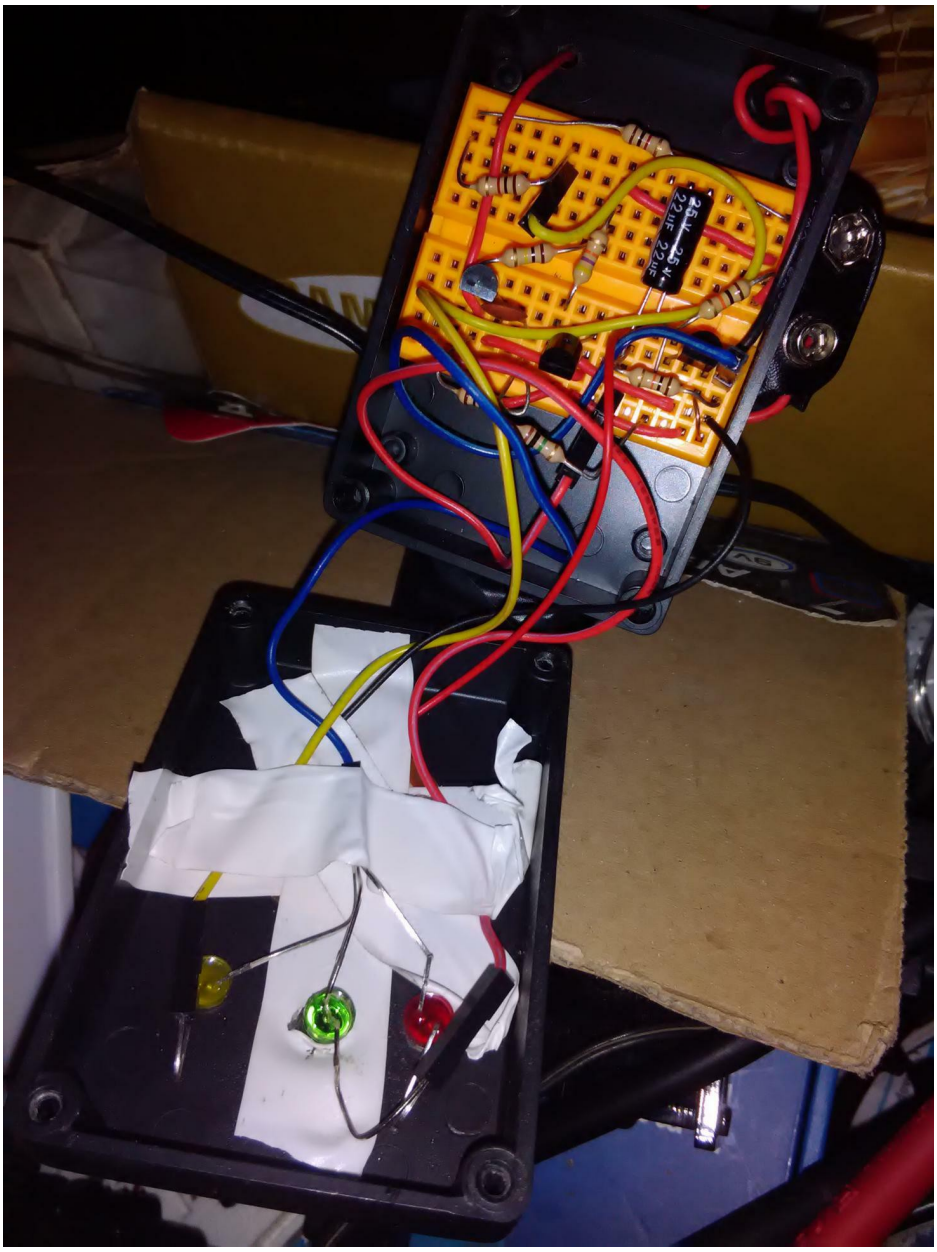




Als Antenne dient mir ein Stück Draht von etwa 15cm Länge, der zu einer kleinen Suchspule gewickelt worden ist. Da in den Adventskalendern kein Schalter vorhanden ist, und ich die beiden Taster im Sender benötige (dazu weiter unten mehr) mußte der Batterieklipp als Ausschalter herhalten. Die Schaltung wurde auf dem Miniboard aufgebaut und in ein gerade vorhandenes Conrad- Gehäuse eingebaut, um das ganze praktisch nutzen zu können. Nun hoffe ich nicht dadurch schon gegen die Wettbewerbsregeln verstossen zu haben, da auch der Sender ein Gehäuse von Conrad erhalten hat.

Das winzige Bredboard und das Modulgehäuse passten so gut zusammen, dass das Board nicht einmal geklebt werden mußte. Im Gehäusedeckel sind schon die Löcher für den „Lautsprecher“, nur noch drei Löcher für die

LEDs rein und zwei kleine für die Drahtantenne und das Versorgungskabel zum Batterieklip, die Batterie passt nicht mit hinein, sondern hält durch etwas Iso-Band hinten auf dem Gehäuse.



2. Der Sender:

Als Sender bot sich der IOT-Controller geradezu an. Ein Ton wird auf einen Portpin ausgegeben, dann eine winzige Pause und ein Ton doppelter Frequenz (Tonhöhe) wird für die gleiche Zeit ausgegeben, dann wieder kurze Pause und von vorn. Das gibt einen recht charakteristischen gewobbelten Ton. Mit dem kleinen Poti an A0 wird analog, ganz nach eigenem Gusto die ausgegebene Frequenz geregelt.

Der Taster an Pin D7 „S1“ wird mit internem PullUp-Widerständen gegen Masse betrieben, und dient alleine gedrückt, zur visuellen Anzeige der Suchfrequenz. Dazu wurde die weiße LED an Pin D9 angeschlossen und wird mit PWM angesprochen.

Da ein unbekanntes Kabel (BITTE IMMER die 5 Sicherheitsregeln einhalten!!!!) nicht nur eine

unbekannte Spannung haben kann (VORHER Prüfen !!! Unser Gerät ist auf gar keinen Fall Spannungsfest!!!) also Stromlos machen! Es kann auch ein Verbraucher, wie ein Boiler oder eine Leuchte angeschlossen sein, oder gar ein Kurzschluss kann vorkommen, das mag nämlich unser Controller auch nicht. Aber wir haben ja noch einen keramischen Kondensator, so kann ihm ein Kurzschluss nichts anhaben und der Such- Ton kommt trozdem noch gut durch. Der zweite Taster wurde aus Platzgründen an A2 gegen VCC eingebaut. Mit diesem kann der Status (wobbeln an oder aus) auch ohne WiFi – Gerät getestet werden. (Suche aus: schnelleres Blinken der LED, im anderen Fall wird ganz langsam geblinkt.) Zusammen mit dem Taster S1 kann nun „von Hand“ der Suchvorgang gestartet werden. Diese Funktion hat sich als äußerst nützlich erwiesen, wenn man a) kein WLAN-Gerät (z.B:Smartfone) zur Hand hat oder wenn sich ein Fehler beim Senden der Webseite ereignet hat.

Eigens dafür habe ich die „WLAN-LED“ onBoard zum blinken gebracht: langsames Blinken = „Website send OK“ (am SerialMonitor), 20 mal hektisches, schnelles Blinken = „Website send Error“

Das Umschalten von Hand erfolgt durch 1. S2 drücken und halten und dazu 2. S1 kurz drücken. Üblicher Weise wird aber auf der vom Controller erzeugten und im WiFi- Gerät angezeigten Webseite an- und ausgeschaltet.

Das Programm wurde so ausgelegt, das es zu erst in ein bestimmtes WLAN-Netz einwählt (im Smartfone wird dazu der WLAN- Hotspot eingeschaltet) wenn das fehl schlägt, schaltet das Modul kurzer Hand in den Accesspoint- Modus und bietet dann im WLAN „NanoEsp“ seine Webseite an.

Auf eine Versorgung des Senders aus einer 9Volt Batterie wurde verzichtet, da man ja besser den USB-(Micro)-Anschluss des Controllers benutzen kann. So kann das Gerät bei lang andauernder Suche aus einem USB-Steckernetzteil betrieben werden kann oder praktischer Weise an einer Powerbank Stundenlang arbeitet. Die Powerbank ihrerseits kann ja bequem wieder aufgeladen werden.

Der Anschluss des zu prüfenden Kabels wurde durch zwei Litzenkabel ausgeführt, die mit einer in jedem Baumarkt erhältlichen Krokoklemme versehen sind. Bredboardseitig mit ein paar cm Draht zum einstecken angelötet. Die Kabel wurden zur Zugentlastung einfach verknotet.

Ergebnisse:

Das Geräte Duo wurde auf einer großen Baustelle erfolgreich getestet! Der Ton des Suchers wude als viel zu leise eingestuft, aber die Erkennbarkeit der grünen LED (besser, als bei der Roten) ist auch in heller Umgebung überzeugend. Die Reichweite des Suchsignals konnte mit weit über 90m bestimmt werden. Die WLAN-Reichweite konnte mit über 50m ermittelt werden. Was ich als hervorragende Fernbedienbarkeit bezeichnen würde.

Die Frage: „Wie geht es meiner Batterie?“ (am Sender) führte mich nach relativ kurzer Suche auf der Arduino Webseite zum „secret voltmeter /thermometer“ und damit zu der Anzeige hier:

hier, die aktuelle Webseite des NanoESP

Die Funktion des Durchgangsprüfers wurde noch nicht erfolgreich getestet, da dem Controller zu schnell der Speicher aus ging. Es wird aber daran noch immer gearbeitet.

Ein Vergleich des Gerätepaars mit einem Profi-Gerätepaar ergab, bis auf die viel viel zu leise „Lautstärke“, keine Nennenswerten Nachteile des Eigenbaues. Im Gegenteil, Betrieb mit Netzteil oder Powerbank und extreme Fernbedienbarkeit zeigen Vorteile auf.

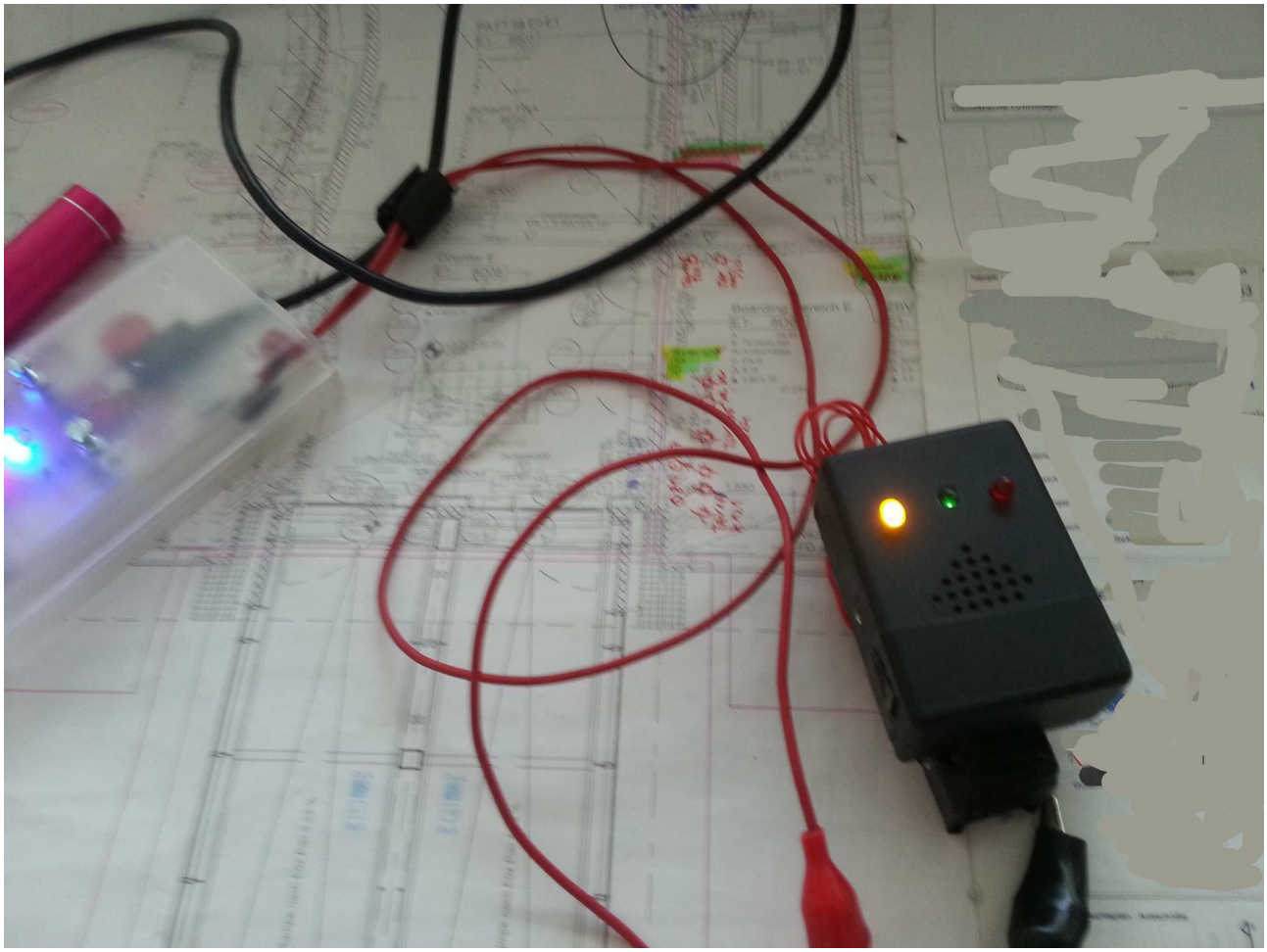
Als Vorteil erwies sich die verwendeten IP- Adressen auf die Rückseite des SenderGehäuses zu schreiben.

Ausblick:

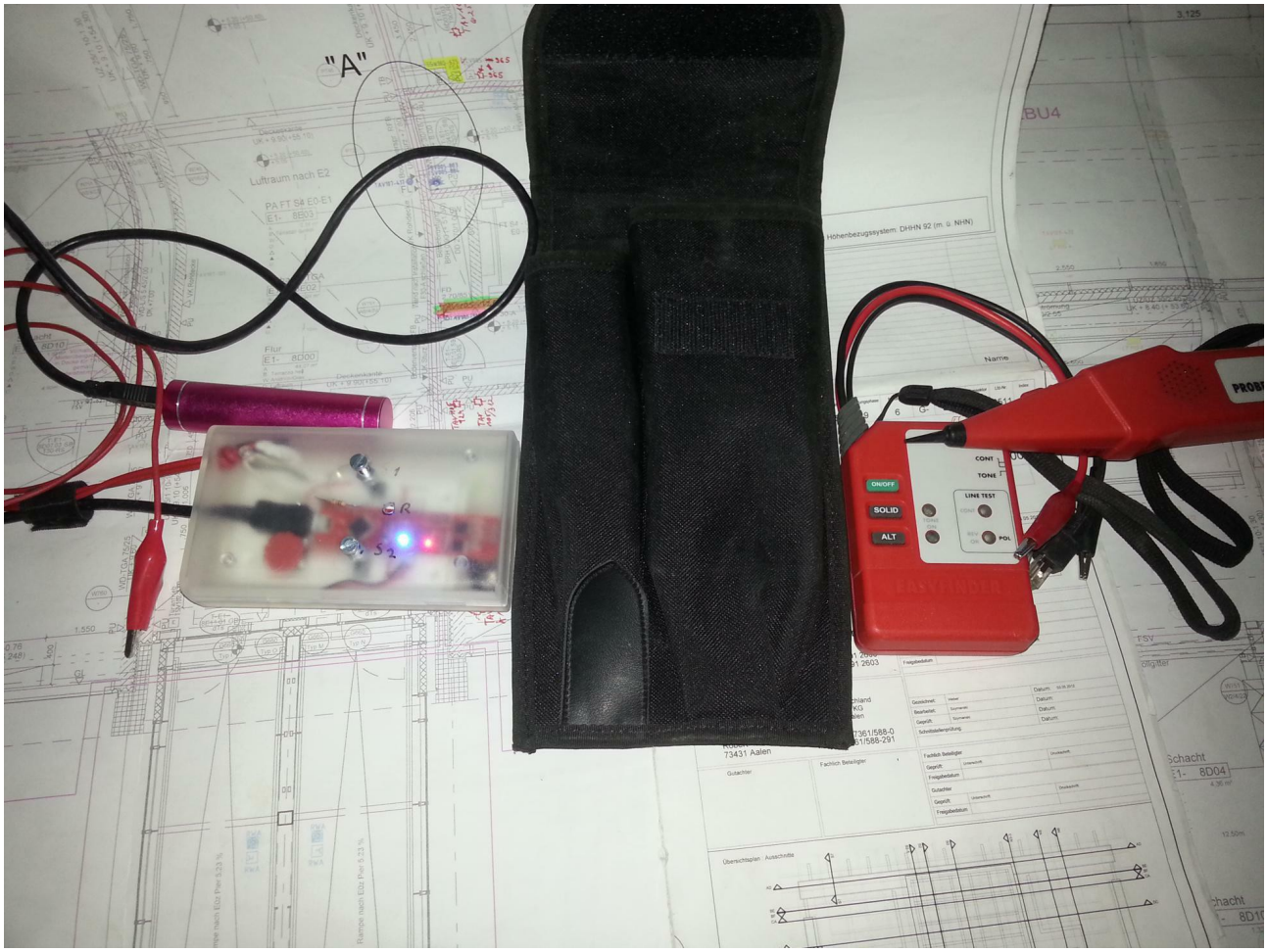
Eine Version ohne die Wettbewerbsbeschränkungen bekommt dann einen Schutz vor Fremdspannung am Sender und einen NF-Verstärker, sowie einen besseren Lautsprecher im

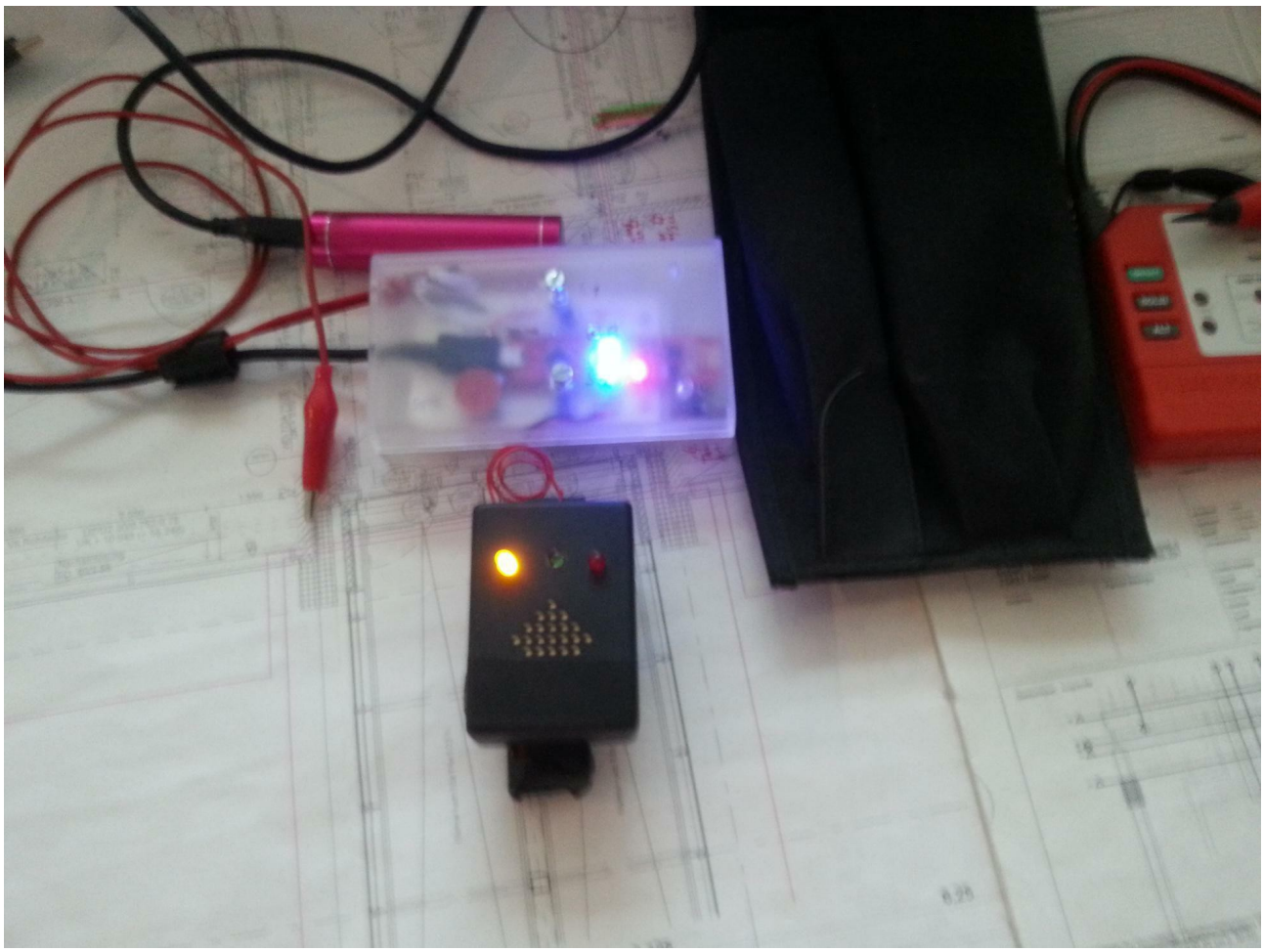
Empfänger.

Hier noch ein paar Bilder des Gerätes im Einsatz, und im Vergleich zum Profi-Gerät...

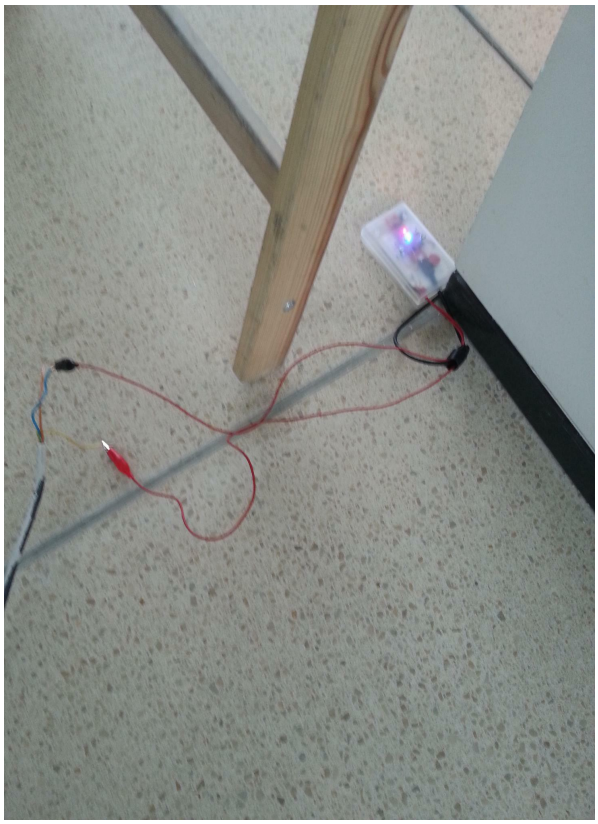


Unschwer zu erkennen: Die Tasche des Profis und daneben das Profi-Gerät, hinter welchem der Eigenbau sich nicht verstecken braucht.



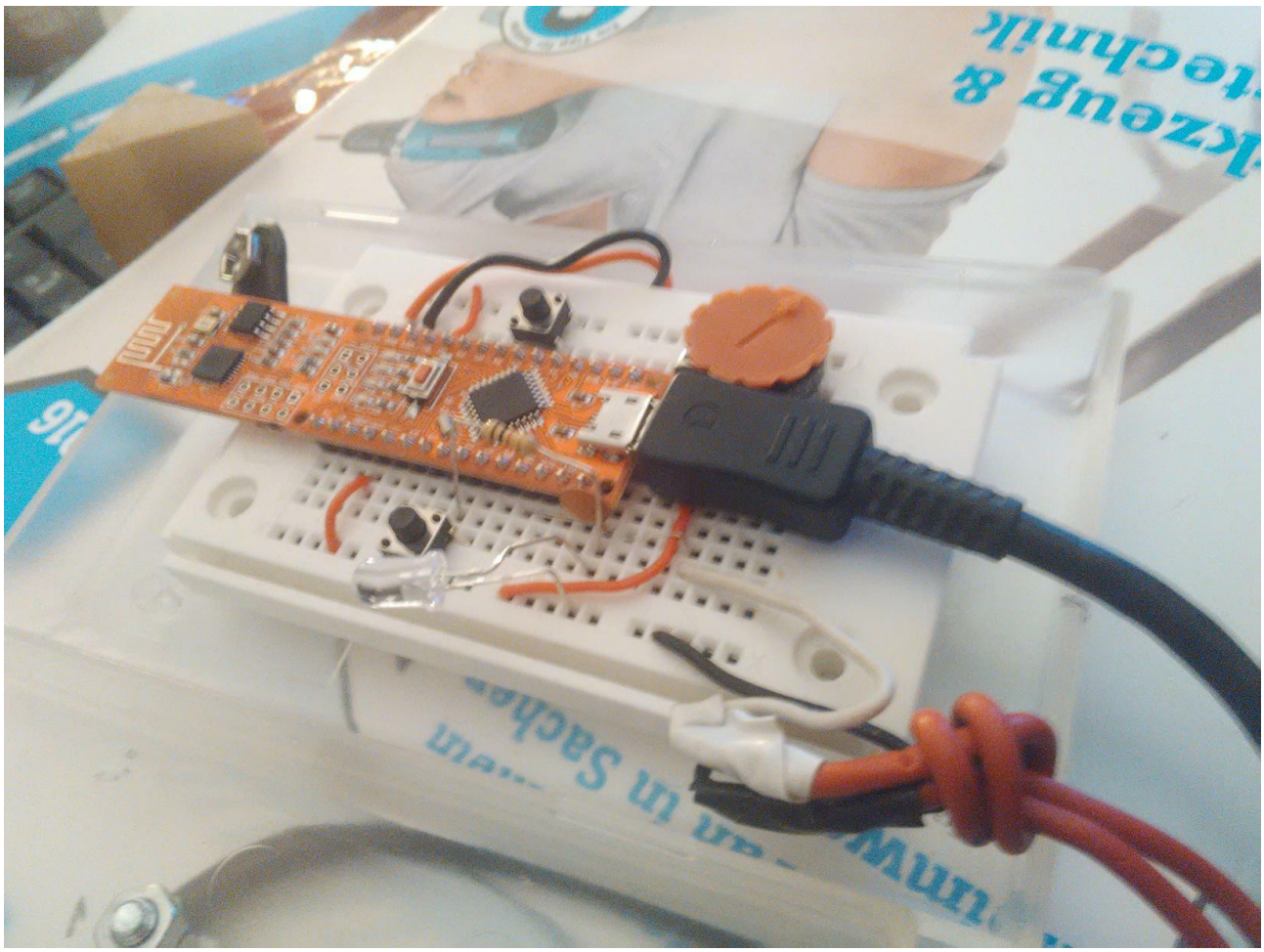


Hier im Bild zeigen unsere zwei, das sie an einer (pinkfarbenen) Powerbank arbeiten. Die Schrauben im Gehäuse des Senders sind nur meine einfache Lösung um die Tasterknöpfe zu verlängern.



Hier mein Foto des Steckboards mit dem NanoESP:

Ein Blick von der Seite zeigt deutlich u.a. den Schutzkondensator 100nF am Ausgang des NanoESP



Zu gutter Letzt noch die Software als Mix aus Selbsterdachten und auf der Arduino.cc gefundenen Stücken (Deswegen sind manche Kommentare in Englisch):

```
/*
  TCP-Server und SuchTon-Ausgabe fÃ¼r Kabelsuche
  Change SSID and PASSWORD.
*/

#define SSID "Total_Commander"
#define PASSWORD "MyPasswort"

#define LED_WLAN 13

#define SENSOR A0
#define LED 6
#define button 7
#define GEN_AUS 9
#define GEN_AU2 10
#define button2 A2
#define DEBUG true
```

```
#include <SoftwareSerial.h>
#include "Webseite.h"

SoftwareSerial esp8266(11, 12); // RX, TX


long readTemp()

{
  long result; // Read temperature sensor against 1.1V reference
  ADMUX = _BV(REFS1) | _BV(REFS0) | _BV(MUX3);
  delay(20); // Wait for Vref to settle - 2 was inadequate
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA, ADSC));
  result = ADCL;
  result |= ADCH << 8;
  result = (result - 125) * 1075; //Don't forget to change "x" with multiplication sign
  return result;
}

long readVcc()

{
  long resultVcc; // Read 1.1V reference against AVcc

  ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) | _BV(MUX1);
  delay(20); // Wait for Vref to settle - 2 was inadequate
  ADCSRA |= _BV(ADSC); // Convert
  while (bit_is_set(ADCSRA, ADSC));
  resultVcc = ADCL;
  resultVcc |= ADCH << 8;
  resultVcc = 1126400L / resultVcc; // Back-calculate AVcc in mV
  return resultVcc;

}


boolean schalter;


void setup() {
  pinMode(LED, OUTPUT);
  pinMode(GEN_AUS, OUTPUT);
```

```
  pinMode(GEN_AU2, OUTPUT);
  pinMode(button, INPUT_PULLUP);
  pinMode(SENSOR, INPUT);
  pinMode(button2, INPUT);
  Serial.begin(19200);
  esp8266.begin(19200);
  schalter = false;
  debug("Suche WLAN:"); debug(SSID);
  if (!espConfig()) serialDebug();
  else BlinkWLAN(LED_WLAN, 1, 10000);          //WLAN-LED fÃ¼r 10 sek an. Danach Stromsparen

  if (configTCPServer())  debug("Server Aktiv"); else debug("Server Error");
}

void loop() {
  String xBuffer; int hlf = 0;
  int Frequenz = analogRead(SENSOR);
  if (esp8266.available()) // check if the esp is sending a message
  {
    if (esp8266.find("+IPD,"))
    {
      debug("Incomming Request");
      int connectionId = esp8266.parseInt();
      if (esp8266.findUntil("&START_STOP=", "\n")) hlf = esp8266.parseInt();
      debug(String(hlf));
      schalter = hlf > 0;
      if (sendWebsite(connectionId, createWebsite())) {
        debug("Website send OK");
        BlinkWLAN(LED_WLAN, 3, 500);
      } else {
        debug("Website send Error");
        BlinkWLAN(LED_WLAN, 20, 100);
      }
    }

  }



  if (!digitalRead(button))
  {
    noTone(GEN_AUS);                          //Button S1 alleine zeigt die verwendete SuchFrequenz als
Helligkeit an
    analogWrite(LED, map(Frequenz, 0, 1023, 0, 255)); //dafÃ¼r Muss aber der Ton AUS sein, da sich PWM und
tone gegenseitig beeinflussen
```

```
  }
  else
  {
    digitalWrite(LED, LOW);
    if (schalter)
    {
      digitalWrite(GEN_AU2, LOW);
      tone(GEN_AUS, Frequenz + 31, 400);
      delay(100);
      tone(GEN_AUS, Frequenz * 2, 400);
      delay(100);

    }
  }
  if (analogRead(button2) > 1000)              //Zu Erst ButtonS2 drÃ¼cken und kurz S1 dazu schaltet den
Suchmodus An und Aus
  {
    if (!digitalRead(button))
    {
      while (!digitalRead(button)) {}
      schalter = !schalter;
    }
    else
    {
      //Nur Button S2 zeigt den Status an
      ZeigeStatus();
    }
  }
}

void ZeigeStatus()
{
  if (schalter)
  {
    BlinkWLAN(LED, 2, 1000);
    //Wenn der Sucher gestartet worden ist, dann Leuchte 2mal fÃ¼r 1 sek.

  }
  else BlinkWLAN(LED, 2, 100);  //an sonsten 2mal schneller
}

void BlinkWLAN(int LEDs, int zahl, int zeit)        // Kleiner debug-Ersatz (ohne delay)
{ long z, t;
  for (int i = 0; i < zahl; i++)
```

```
  {
    digitalWrite(LEDs, HIGH);
    z = millis(); t = z;
    while (t <= z + zeit)
    {
      t = millis();
    }
    digitalWrite(LEDs, LOW);
    z = millis();
    while (t <= z + zeit)
    {
      t = millis();
    }
  }
}
//----------------------------Webseite

boolean sendWebsite(int connectionId, String webpage)
{
  boolean succes = true;

  if (sendCom("AT+CIPSEND=" + String(connectionId) + "," + String(webpage.length()), ">"))
  {
    esp8266.print(webpage);
    esp8266.find("SEND OK");
    succes &= sendCom("AT+CIPCLOSE=" + String(connectionId), "OK");
  }
  else
  {
    succes = false;
  }
  return succes;
}

String createWebsite()
{
  String xBuffer;
  float tmp;
  float vcc;

  tmp = readTemp() / 10000.0;
  delay(20);
  vcc = readVcc() / 1000.0;
  for (int i = 0; i < sizeof(site); i++)
```

```
  {
    char myChar = pgm_read_byte_near(site + i);
    xBuffer += myChar;
  }

  xBuffer.replace("*bright*", String(analogRead(SENSOR)));
  xBuffer.replace("*Temp*", String(tmp));
  xBuffer.replace("*Volts*", String(vcc));
  if (schalter)
    xBuffer.replace("*AN_AUS*", "AN");
  else
    xBuffer.replace("*AN_AUS*", "AUS");
  return xBuffer;
}


//----------------------------------------Config ESP8266----------------------------------

boolean espConfig()
{
  boolean succes = true;
  esp8266.setTimeout(5000);
  succes &= sendCom("AT+RST", "ready");
  esp8266.setTimeout(1000);
  if (configStation(SSID, PASSWORD)) {
    succes &= true;
    debug("WLAN Connected");
    debug("My IP is:");
    debug(sendCom("AT+CIFSR"));
  }
  else
  {
    if (configAP())
    {
      succes &= true;
      debug("WLAN AP aktiviert");
      debug("My IP is:");
      debug(sendCom("AT+CIFSR"));
    }
    else
      succes &= false;
  }
  //shorter Timeout for faster wrong UPD-Comands handling
  succes &= sendCom("AT+CIPMODE=0", "OK");
```

```
  succes &= sendCom("AT+CIPMUX=0", "OK");

  return succes;
}

boolean configTCPServer()
{
  boolean succes = true;

  succes &= (sendCom("AT+CIPMUX=1", "OK"));
  succes &= (sendCom("AT+CIPSERVER=1,80", "OK"));

  return succes;

}

boolean configTCPClient()
{
  boolean succes = true;

  succes &= (sendCom("AT+CIPMUX=0", "OK"));
  //succes &= (sendCom("AT+CIPSERVER=1,80", "OK"));

  return succes;

}



boolean configStation(String vSSID, String vPASSWORT)
{
  boolean succes = true;
  succes &= (sendCom("AT+CWMODE=1", "OK"));
  esp8266.setTimeout(20000);
  succes &= (sendCom("AT+CWJAP=\"" + String(vSSID) + "\",\"" + String(vPASSWORT) + "\"", "OK"));
  esp8266.setTimeout(1000);
  return succes;
}

boolean configAP()
{
  boolean succes = true;

  succes &= (sendCom("AT+CWMODE=2", "OK"));
  succes &= (sendCom("AT+CWSAP=\"NanoESP\",\"\",5,0", "OK"));
```

```
  return succes;
}


//-------------------------------------------------Controll
ESP-----------------------------------------------------


boolean sendCom(String command, char respond[])
{
  esp8266.println(command);
  if (esp8266.findUntil(respond, "ERROR"))
  {
    return true;
  }
  else
  {
    debug("ESP SEND ERROR: " + command);
    return false;
  }
}

String sendCom(String command)
{
  esp8266.println(command);
  return esp8266.readString();
}




//-------------------------------------------------Debug
Functions-----------------------------------------------------
void serialDebug() {
  while (true)
  {
    if (esp8266.available())
      Serial.write(esp8266.read());
    if (Serial.available())
      esp8266.write(Serial.read());
  }
}

void debug(String Msg)
```

```
{
  if (DEBUG)
  {
    Serial.println(Msg);
  }
}
```