Funksteckdosen mit dem Arduino steuern

Dieser Artikel war die Inspiration zum Miniprojekt im QG NWT-Wiki. Leider ist die Seite inzwischen aus dem Netz verschwunden.

- https://web.archive.org/web/20190606165653/http://www.dserv01.de/howtos/funksteckdose-fernsteuern-mit-arduino/
- https://wiki.qg-moessingen.de/faecher:nwt:bluetooth_steckdosen:433_mhz:funksteckdose:start

---

Funksteckdose fernsteuern mit Arduino

Achtung: Diesen Artikel habe ich waehrend meiner Schulzeit mit nur laienhaften Kenntnissen geschrieben. Es koennen grobe Fehler enthalten sein.

Kaffee fertig zum Weckerklingeln oder Geraete einfach per Computer ein und ausschalten. Ich steuer alles gerne ueber meinen Homeserver/DesktopPC, der aufgrund seines Stromverbrauchs von nur ca 10Watt immer laeuft, und da ich mir letztens eine kleine Kaffeemaschine gekauft habe war es nur eine Frage der Zeit bis mein PC auch Herr ueber die Kaffeemaschine/Steckdosen werden sollte. Ich hatte noch ein Set Funksteckdosen von Elro, das ich mal fuer 10€ bei Media Markt gekauft hatte, das sich hierfuer anbot. Aufgrund des niedrigen Preises und der hohen Sicherheit (vgl. zusammengebastelte 240V Relays), zusaetzlich Draht- und trotzdem Batterielos, eignet sich so ein Funksteckdosenset fuer solche Vorhaben am besten. Funksteckdosen mit einem Arduino verbinden mag auf den ersten Blick zwar kompliziert wirken, aber eigentlich duerfte das jeder Anfaenger koennen. Das noetige Wissen dazu werde ich euch in diesem Tutorial versuchen zu vermitteln. Ich habe im Internet 1-2 Artikel gefunden, die sich auch mit diesem Thema beschaeftigen, jedoch beschraenken die sich meist einfach auf eine Library deren Befehle man sich ersteinmal anschauen muss (und dann vielleicht trotzdem nichts mit den ausgegebenen Werten anfangen kann), obwohl eine genaue Beschaeftigung und selbststaendige Messung+Programmierung genausoschnell gehen wuerde (und man hierbei noch einen deutlichen Lerneffekt hat).

Fangen wir also an:

Anforderungen

Wir brauchen: Ein 434Mhz Funksteckdosenset (Es kann auch 433Mhz oder 433.92Mhz etc. sein), einen 434Mhz Empfaenger und einen 434Mhz Sender. Ich werde mit dem Watterott Empfaenger und Sender arbeiten und ich empfehle diese auch zu kaufen. Mit Versand kommt ein Gesammtpreis von 11.20€ zustande, der nun wirklich nicht hoch ist (Abgesehn davon hat Watterott ne Menge gute Sachen, die die Versandkosten druecken wuerden). Dies ist nahe an der guenstigsten Moeglichkeit und hat den Vorteil, dass sich diese Modelle besonders leicht verwenden lassen. Ein Arduino sollte auch noch in Besitz sein und du solltest wenigstens Wissen, wie man damit Leds leuchten laesst. Sollte dies nicht der Fall sein, solltest du dich erstmal in langweiligen Led-Tutorials (z.B. Hier) damit vertraut machen. Natuerlich brauchst du auch noch was zum Verbinden, sprich Kabel/Draht oder aehnliches. Zum Messen werden wir den Mikrofonanschluss deines PCs benutzen, also solltest du irgendwie ein beidendiges 3.5mm(Klinke)Kabel haben oder irgendwas anderes um an die Kontakte zu kommen.
Fuer den PC brauchst du noch die Software Audacity.
Optimal: Breadboard (ich nutze meist ein kleines 3€(Endpreis!) Board von Ebay). Es geht auch ohne, aber das kann zu Frust fuehren. Die von mir genannte Hardware ist Breadboard ausgelegt. Da ein Breadboard guenstig ist und immer wieder gebraucht wird, sollte dies nirgendswo fehlen.

Encodieren der Tasten

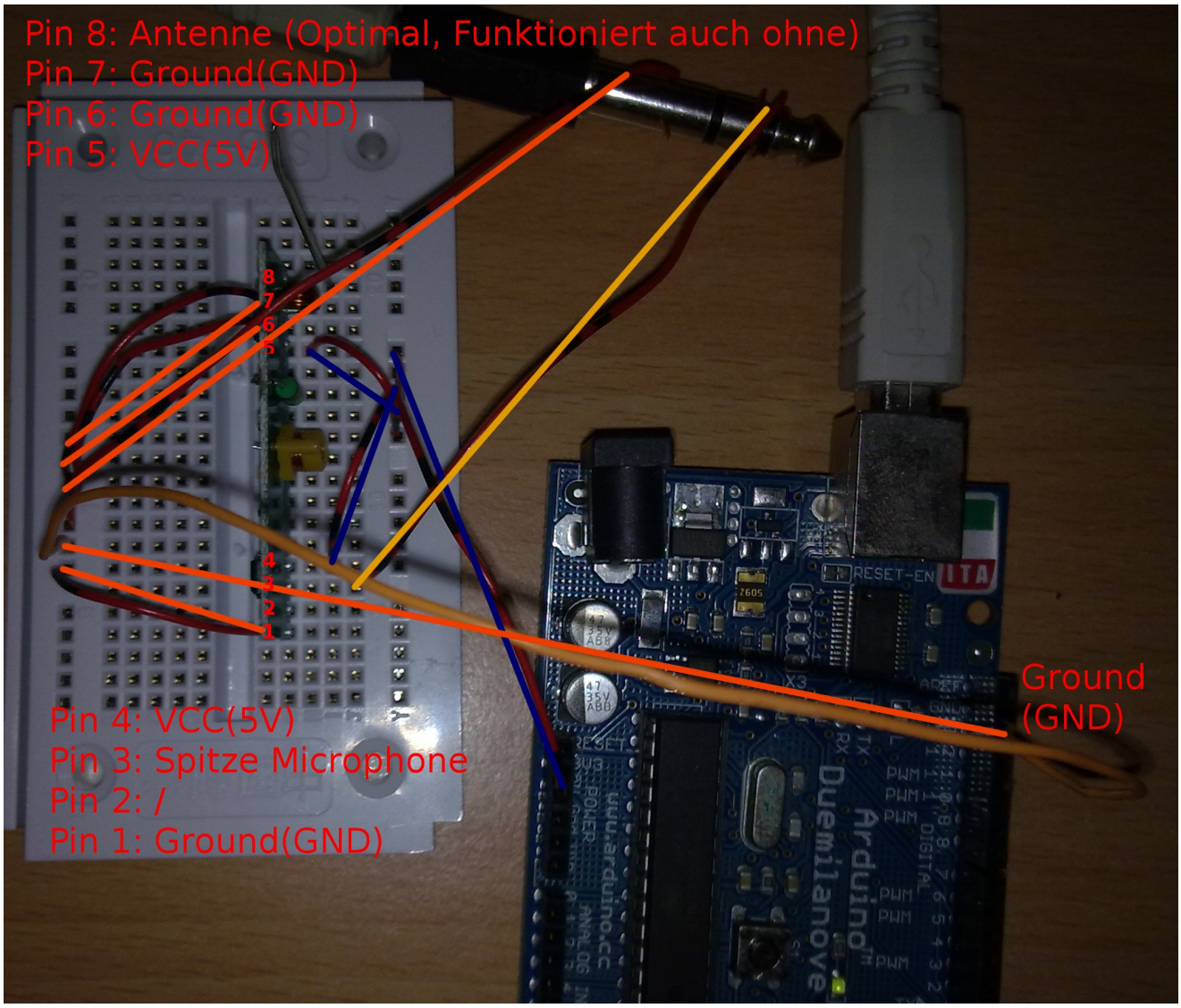
Wenn du alle Bestandteile hast und dich auch ein wenig mit der Funktionsweise der Funksteckdose vertraut gemacht hast, sprich die beigelegte Anleitung, gelesen hast [damit es nicht daran scheitert, dass du vergessen hast, Fernbedienung und Steckdosen

aufeinander abzustimmen], koennen wir anfangen den Empfaenger anzuschliessen. Zuvor moechte ich dir aber noch erklaeren was wir genau machen:

Ein Mikrofon wandelt eigentlich nur Schalldruck in Spannungsaenderungen um (mehr?: Wikipedia). Unser Empfaenger wandelt Elektromagnetische Wellen der Frequenz 434Mhz in Spannungsaenderungen um. Da den PC nur die Spannungsaenderung interessiert kann, bietet es sich an den Empfaenger als Mikrofon anzuschliessen um dann die Fernbedienung(bzw. die Tastendruecke) aufzunehmen und zu analysieren. Dies geht mit Audacity meiner Meinung nach am besten. Hier koennen wir uns die Signale genau anschauen. Erstmal muessen wir jedoch den Empfaenger anschliessen:

Es ist hilfreich wenn ihr euch erstmal das Dokument zum Modul durchliest.

Ich habe aufgrund der schlecht unterscheidbaren Kabel, einfach mal farbige Linien drueber gezeichnet. (Orange: Ground | Gelb: Daten | Blau: 5V)

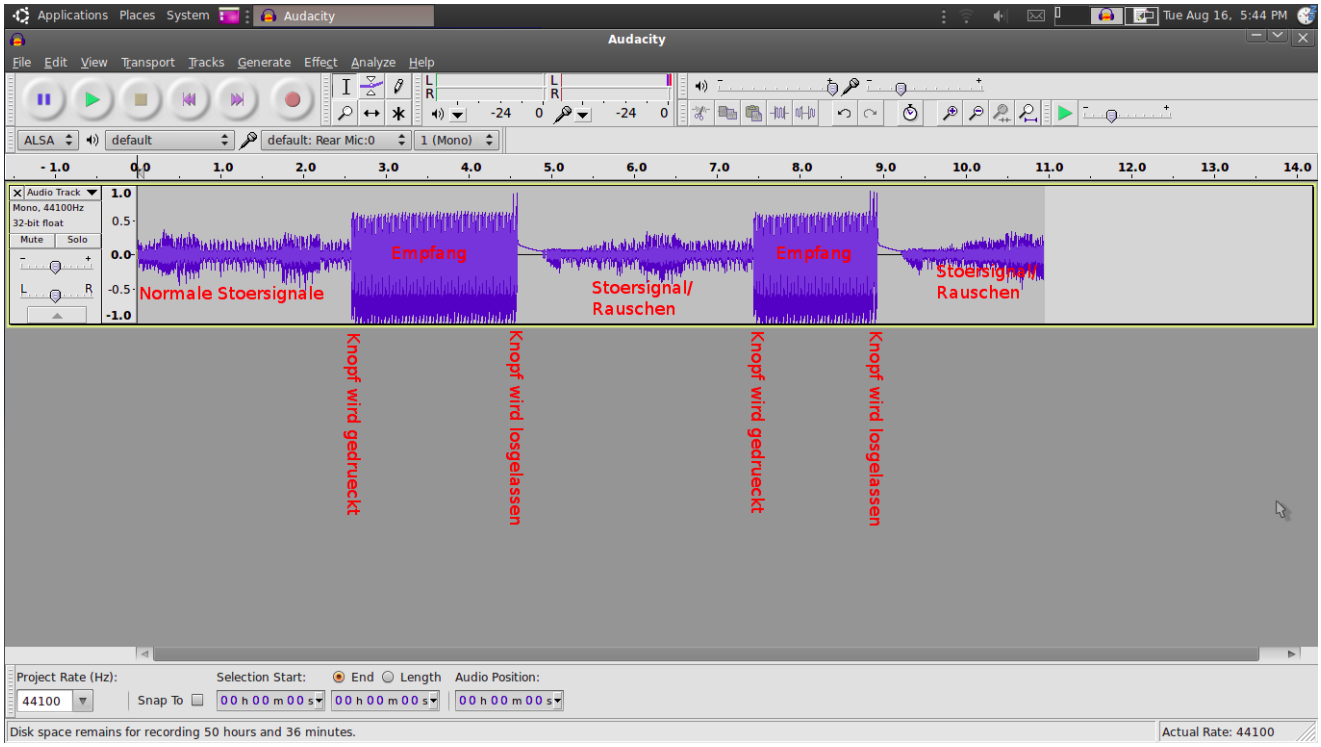


Der Arduino dient jetzt nur als Stromzufuhr, da der Empfaenger leider nicht Passiv arbeitet. Achtet bitte genau auf die Verkabelung, sodass kein Kurzschluss entsteht. Ich habe einen Adapter verwendet, weil der 3.5mm Anschluss mir ein wenig zu klein war. Sollte dich die Verkabelung vor ein Raetzel stellen, solltest du dich eventuell ueber Breadboards

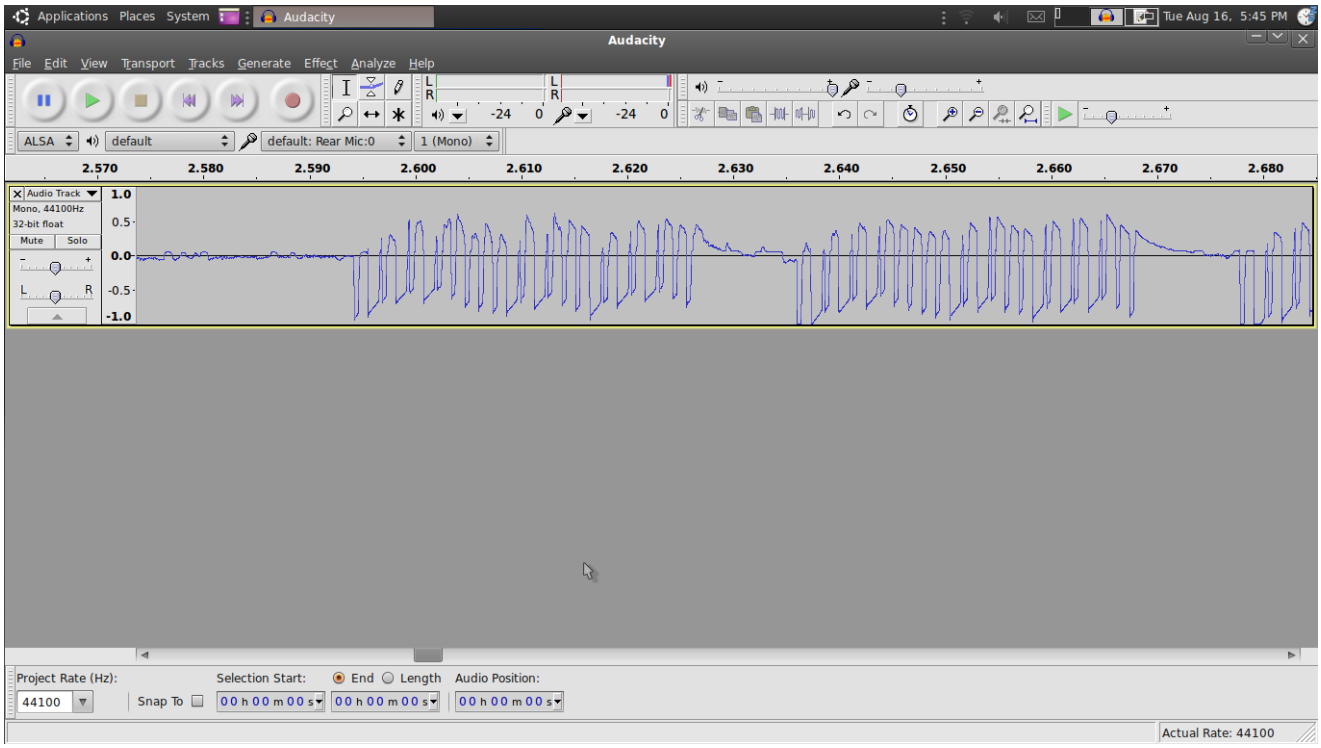
informieren. Wenn trotzdem etwas unverstaendlich sein sollte, zoeger nicht die Kommentarfunktion zu benutzen, damit ich dieses Tutorial ausbesseren kann.

Wenn wir nun alles angeschlossen haben (moeglichst nicht direkt vor der Tastatur, sonst koennte es gleich schnell wieder kaputt gehen) koennen wir nun anfangen den Funk zu entschluesseln. Dafuer nutzen wir nun Audacity. Du solltest evtl. vorher schon ein wenig Erfahrung mit Audacity sammeln indem du einfach ein Micro anschliesst du irgendwas aufnimmst. Audacity ist etwas komplexer als einfache Aufnahmeprogramme auch wenn wir eigentlich nur den leicht erkennbaren Aufnahme- und Stopbutton brauchen werden. Stelle einfach sicher, dass der Mikrofonanschluss nicht in den Systemeinstellungen gemuted ist oder du den falschen Anschluss ausgewaehlt hast. Dies ist keine grosse Herrausforderung, aber es waere aerglich, wenn man nichts empfaengt und glaubt man haette etwas auf dem Breadboard falsch verkabelt. Dieses Wissen ist auch fuer jede Menge anderer Projekte sinnvoll. Falls du also noch keine Ahnung von Audacity hast, ist es keinenfalls Zeitverschwendung.

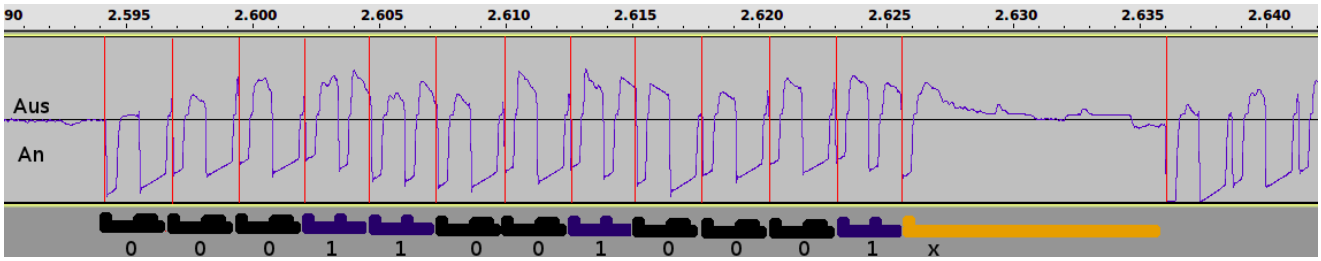
So, jetzt solltest du einfach den Empfaenger als Mikrofon anschliessen und mit Audacity aufnehmen koennen. Du solltest ein leichtes Rauschen sehen koennen (auf dem „Diagram“ der Audiospur von Audacity) wenn du auf Aufnehmen drueckst. Wenn du nun die Fernbedienung in die Naehe des Empfaengers haelst und eine Taste drueckst, sollte es einen staerkeren Ausschlag geben. Nach der Aufnahme drueckst du auf den Stopbutton um die Aufnahme zu stoppen und dir die Aufnahem anzusehen.



Wenn du mit bzw. (je nach Tastaturlayout) und Scrollrad reinzoomst kannst du dir die Aufnahme genauer ansehen.



Du wirst merken, dass jeder Tastendruck aus mehreren Staffeln mit den selben Ausschlag-Mustern besteht. Du musst dir eigentlich nur die negativen Ausschlage ansehen, die kleinen positiven sind lediglich eine physikalische Wechselwirkung auf die negativen. Die genaue Ausschlagsstaerke kannst du auch vernachlaessigen, denn du solltest aus dem Muster erkennen, dass es sich um einen Binaercode handelt, der mehrere Male (um Fehlsignale zu verhindern muss der Code mehrmals empfangen werden) gesendet wird. Obwohl Binaercode eigentlich aus An und Aus (1 und 0) besteht, kann man dies nicht genau hierdrauf anwenden, denn sonst wuerden ja die halbe Zeit Code gesendet (der zwar nur aus 0en besteht, aber dadurch koennte man z.B. 0010 und 0100 nicht unterscheiden). Deshalb bestehen die 1en und 0en nochmal selbst aus einem Code. Zusaetzlich kommt noch ein Synchronisationssignal, damit der Empfaenger erkennt, wo der Anfang und das Ende ist. Du kannst selber versuchen den Code zu dechiffrieren, was relativ schwer ist, wenn man es zum ersten Mal macht, oder einfach dir die Loesung angucken:



Dies ist ein Beispielcode. Du kannst einen anderen haben. Sollte der Code ganz anders aussehen, sprich du kannst deinen Code nicht so mit 1en und 0en ausfuellen, hast du evtl. ein anderes Modell das neben 1 und 0 [+x(sync)] noch ein weiteres Zeichen hat. Da ich so ein Modell nicht habe, kann ich leider auch nichts dazu abbilden, aber es laeuft nach dem selben Prinzip ab. Du solltest improvisieren koennen. Wenn du nicht weiterkommst, kannst du mich notfalls mit Screenshots versorgen und ich erweiter dann das Tutorial.

0= Kurzer Ausschlag, Lange Pause, Langer Ausschlag, kurze Pause
1= Kurzer Ausschlag, Lange Pause, Kurzer Ausschlag, Lange Pause
x(Sync-Signal)=Kurzer Ausschlag, Sehr lange Pause

Ein Code besteht also aus 12Bits(0en und 1en)+1Sync-Signal(x). Die einzelnen Bits(Mit der Zeitleiste abgemessen: ca 2.8ms) koennte man selbst nochmal in 8Bits(a 350us) (Diesmal An/Aus) unterteilen. Dies macht das spaeter Umsetzen mit dem Arduino einfacher.

0=10001110 (8bits a'350us; insg.: 2.8ms) [An: 350us | Aus: 1050us | An: 1050us | Aus: 350us]
1=10001000 (8bits a'350us; insg.:2.8ms) [An: 350us | Aus: 1050us | An: 350us | Aus: 1050us]
x=10000000000000000000000000000000 (32Bits a'350us; insg.: 11.2ms) [An: 350us | Aus: 10.85ms]

Du solltest nun aus jeder Taste so einen Code bilden koennen. Bei mir sehen die Codes so aus:

On	Off
A 000111000010x	000111000001x
B 000110100010x	000110100001x
C 000110010010x	000110010001x
D 000110001010x	000110001001x

Man erkennt, dass die ersten 5 Bits immer gleich sind. Dies ist die Verschluesselung die man in der Fernbedienung und den Empfaengern einstellen kann. Praktischerweise passt die 5 Bits genau zu den 5 Schaltern (mit 0=Unten und 1=Oben).

Die Naechsten 5 Bits sind bei jedem Buchstaben gleich. Sie entsprechen also dem Buchstaben. Auch hier passt die Reihenfolge wie angegossen [A:10000 | B:01000 |…]

Die letzten zwei Bits sind immer bei On oder Off gleich. [On:10 | Off: 01]

Das x am Ende ist wie schon vorher angesprochen das Sync-Signal, dass die Bitcodes trennt und somit Anfang und Ende einer Reihenfolge definiert. Dies ermoeglicht das mehrfache Wiederholen des Codes, denn die Steckdosen wollen den Code mehrmals hintereinander empfangen, damit sie wissen, dass es nicht nur ein zufaellig passendes Stoersignal ist (siehe Elektrosmog).

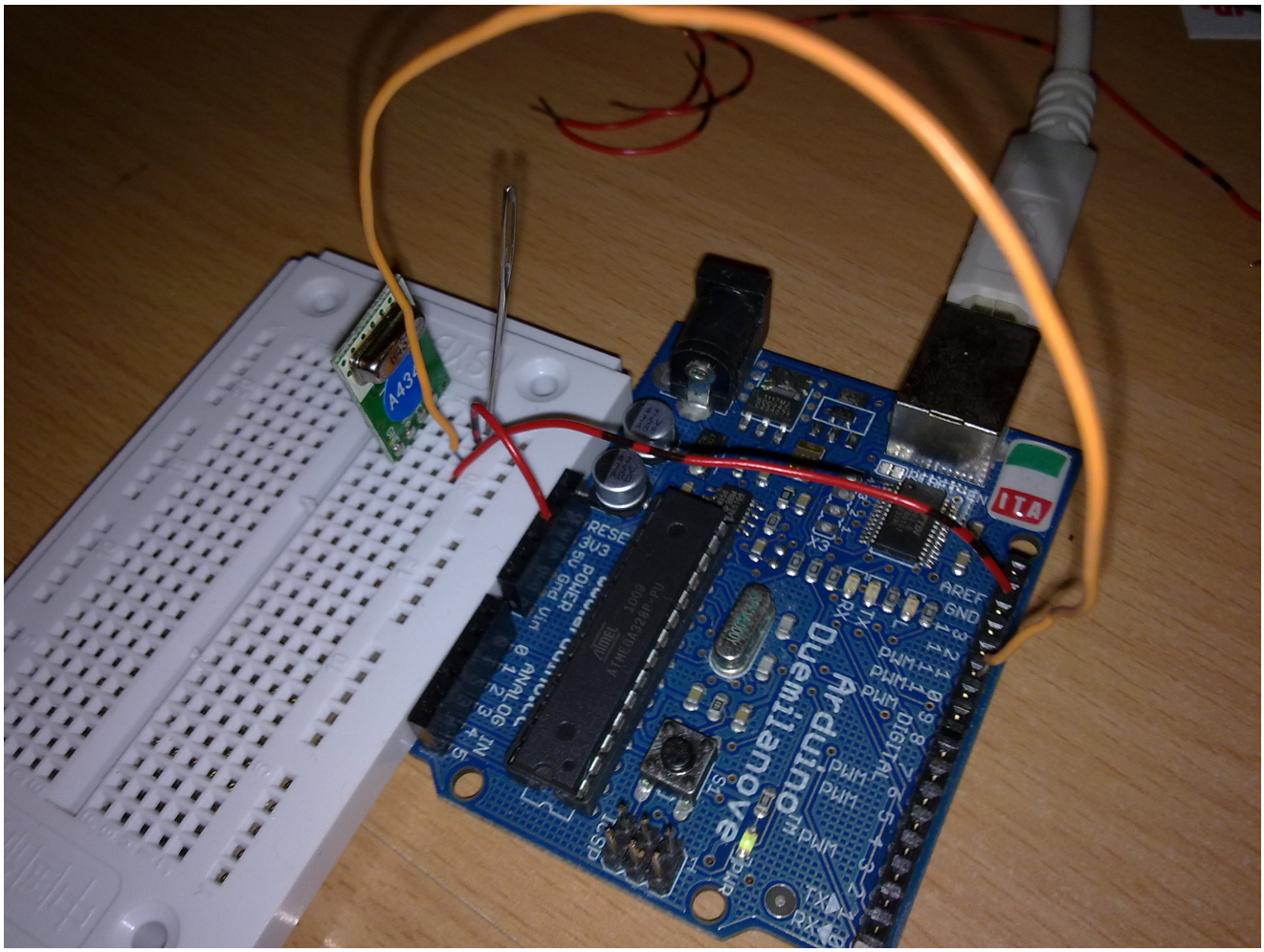
So koennten wir uns jetzt auch ohne messen den Code ausrechnen, was darauf hinweisst, dass unser Codesystem auch richtig ist. Wir sind mit diesem Kapitel also fertig. Als naechstes wollen wir den Code auch senden.

Senden

Eigentlich koennten wir den Sender auch mit passender Stromversorgung am Audioausgang des PCs anschliessen und einfach die Aufnahme abspielen. So wuerden wir genau das

senden, was wir auch empfangen/aufgenommen haben. Dies ist jedoch eine sehr unelegante Loesung und wenn wir Musik abspielen wuerden, wuerde der Sender die ganze Zeit dazu irgendwas senden und viel Elektrosmog produzieren. Wir modulieren das Signal also mit dem Arduino nach. Dies ist durch unseren Binaercode ziemlich einfach.

Zuerst muessen wir den Sender anschliessen:



Du solltest alles erkennen koennen, zusaetzlich steht auf dem Sender selbst noch, was wohin muss. Ansonsten guck einfach in das Dokument von oben.

Nun basteln wir uns den Code fuer den Arduino:

Zuerst schreiben wir eine Funktion, die 1,0 oder x sendet.

```
void sendByte(char i) { //Diese Funktion soll 0,1 oder x senden koennen. Wir
speichern die gewuenschte Ausgabe in der Variabel i
  switch(i){ //nun gucken wir was i ist
  case '0':{ //Der Code fuer '0'
    digitalWrite(rc_pin,HIGH);
    wait(1); //da die Pausen x*350us lang sind, machen wir daraus eine Funktion
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
    wait(3);
```

```
    digitalWrite(rc_pin,LOW);
    wait(1);
    return;
  }
  case '1':{ //Der Code fuer '1'
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(3);
    return;
  }
  case 'x':{ //Der Code fuer x(sync)
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(31);
  }

}
}

void wait(int x) {
  delayMicroseconds(x*350); //warte x*350us
}
```

Nun brauchen wir noch eine Funktion die einen Code in die Bits unterteilt.

```
boolean sendCode(char code[]){ //empfange den Code in Form eines Char[]
  for(short z = 0; z<7; z++){ //wiederhole den Code 7x
    for(short i = 0; i<12; i++){ //ein Code besteht aus 12bits
      sendByte(code[i]);
    }
  sendByte('x'); //da der code immer mit x/sync abschliesst, brauchen wir den
nicht im code und haengen es automatisch immer hinten ran.
  }
  return true;
}
```

Nun fuegen wir den Code zusammen:

```
short rc_pin=13; //der Pin auf dem der Datenpin des Senders angeschlossen ist.

void setup() {
  pinMode(rc_pin, OUTPUT);  //definiere rc_Pin als Ausgang (schliesslich wollen
wir senden)
}

void loop() {
sendCode("000110010001"); //sende den Beispiel Code[Aus]
delay(1000);
sendCode("000110010010"); //sende den Beispiel Code[An]
delay(1000);
}
boolean sendCode(char code[]){ //empfange den Code in Form eines Char[]
  for(short z = 0; z<7; z++){ //wiederhole den Code 7x
    for(short i = 0; i<12; i++){ //ein Code besteht aus 12bits
      sendByte(code[i]);
    }
  sendByte('x'); //da der code immer mit x/sync abschliesst, brauchen wir den
nicht im code und haengen es autisch immer hinten ran.
  }
  return true;
}
void sendByte(char i) { //Diese Funktion soll 0,1 oder x senden koennen. Wir
speichern die gewuenschte Ausgabe in der Variabel i
  switch(i){ //nun gucken wir was i ist
  case '0':{ //Der Code fuer '0'
    digitalWrite(rc_pin,HIGH);
    wait(1); //da die Pausen x*350us lang sind, machen wir daraus eine Funktion
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
    wait(3);
    digitalWrite(rc_pin,LOW);
    wait(1);
    return;
  }
  case '1':{ //Der Code fuer '1'
    digitalWrite(rc_pin,HIGH);
    wait(1);
```

```
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(3);
    return;
  }
  case 'x':{ //Der Code fuer x(sync)
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(31);
  }

}
}

void wait(int x) {
  delayMicroseconds(x*350); //warte x*350us
}
```

Nun sollten wir problemlos eine Funksteckdose mit dem Arduino ein und ausschalten koennen. Leider nicht auf Befehl, aber dazu kommen wir nun im naechsten Kapitel.

Sebastian hat in den Kommentaren noch einen Code für die Lidl Funksteckdosen RC-710 von Libra gepostet, den ich euch nicht vorenthalten will. Da Copy&Paste aus den Kommentaren nur begrenzt geht (z.B. , “ , wird mit einem anderen Symbol ersetzt), gibt es den auch hier in formatierte und kopierbarer Form:

```
#define ch1on "1011000000000010001"
#define ch1off "1011000000000000000"
#define ch2on "1011000000010010011"
#define ch2off "1011000000010000010"
#define ch3on "1011000000001010000"
#define ch3off "1011000000001000001"
#define ch4on "1011000000011010010"
#define ch4off "1011000000011000011"
#define chMon "1011000000011110000"
#define chMoff "1011000000011100001"
#define dimmUp1 "1011000000000001010"
#define dimmDn1 "1011000000000011011"
```

```
#define dimmUp2 "10110000000010001000"
#define dimmDn2 "10110000000010011001"
#define dimmUp3 "10110000000001001011"
#define dimmDn3 "10110000000001011010"
#define dimmUp4 "10110000000011001001"
#define dimmDn4 "10110000000011011000"

short rc_pin=2;

void setup() {
  pinMode(rc_pin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  while(Serial.available()>0){
    byte command = Serial.read();
    switch(command){
    case '1':
      {
        sendCode(ch1on);
        Serial.println("command ch1on executed");
        break;
      }
    case '2':
      {
        sendCode(ch1off);
        Serial.println("command ch1off executed");
        break;
      }
    case '3':
      {
        sendCode(ch2on);
        Serial.println("command ch2on executed");
        break;
      }
    case '4':
      {
        sendCode(ch2off);
        Serial.println("command ch2off executed");
        break;
```

```
    }
  case '5':
    {
      sendCode(ch3on);
      Serial.println("command ch3on executed");
      break;
    }
  case '6':
    {
      sendCode(ch3off);
      Serial.println("command ch3off executed");
      break;
    }
  case '7':
    {
      sendCode(ch4on);
      Serial.println("command ch4on executed");
      break;
    }
  case '8':
    {
      sendCode(ch4off);
      Serial.println("command ch4off executed");
      break;
    }
  case '9':
    {
      sendCode(chMon);
      Serial.println("command chMon executed");
      break;
    }
  case 'a':
    {
      sendCode(chMoff);
      Serial.println("command chMoff executed");
      break;
    }
  case 'b':
    {
      sendCode(dimmUp1);
      Serial.println("command dimmUp1 executed");
```

```
        break;
      }
    case 'c':
      {
        sendCode(dimmDn1);
        Serial.println("command dimmDn1 executed");
        break;
      }
    case 'd':
      {
        sendCode(dimmUp2);
        Serial.println("command dimmUp2 executed");
        break;
      }
    case 'e':
      {
        sendCode(dimmDn2);
        Serial.println("command dimmDn2 executed");
        break;
      }
    case 'f':
      {
        sendCode(dimmUp3);
        Serial.println("command dimmUp3 executed");
        break;
      }
    case 'g':
      {
        sendCode(dimmDn3);
        Serial.println("command dimmDn3 executed");
        break;
      }
    case 'h':
      {
        sendCode(dimmUp4);
        Serial.println("command dimmUp4 executed");
        break;
      }
    case 'i':
      {
        sendCode(dimmDn4);
```

```
        Serial.println("command dimmDn4 executed");
        break;
      }
    default:
      Serial.println("command not found");
    }
  }
}
boolean sendCode(char code[]){ //empfange den Code in Form eines Char[]
  sendSymbol('x');
  for(short z = 0; z<7; z++){ //wiederhole den Code 7x
    for(short i = 0; i<20; i++){ //ein Code besteht aus 20bits
      sendSymbol(code[i]);
    }
    sendSymbol('x'); //da der code immer mit x/sync abschliesst, brauchen wir den
nicht im code und haengen es autisch immer hinten ran.
  }
  return true;
}
void sendSymbol(char i) { //Diese Funktion soll 0,1 oder x senden koennen. Wir
speichern die gewuenschte Ausgabe in der Variabel i
  switch(i){ //nun gucken wir was i ist
  case '0':
    { //Der Code fuer '0'
      digitalWrite(rc_pin,LOW);
      delayMicroseconds(600);
      digitalWrite(rc_pin,HIGH);
      delayMicroseconds(1400);
      return;
    }
  case '1':
    { //Der Code fuer '1'
      digitalWrite(rc_pin,LOW);
      delayMicroseconds(1300);
      digitalWrite(rc_pin,HIGH);
      delayMicroseconds(700);
      return;
    }
  case 'x':
    { //Der Code fuer x(sync)
      digitalWrite(rc_pin,LOW);
```

```
      delay(81);
      digitalWrite(rc_pin,HIGH);
      delayMicroseconds(800);
    }

  }
}
```

Desweiteren gibt es auch eine Library von picoPAN

.

Schalten mit dem PC(Linux)

Natuerlich kann man auch mit dem SerialMonitor der ArduinoIDE arbeiten, aber die ist eher zum Testen als zum richtigen Umsetzen. Der Code laesst sich trotzdem auch mit dem SerialMonitor verwenden.

Damit der Arduino weiss, wann er was senden muss, muessen wir ihm irgendwie Informationen senden. Dies machen wir am besten mit einer seriellen Verbindung ueber USB. Ich habe hierfuer mal den Anfang des Codes leicht veraendert:

```
short rc_pin=13;
char code[12]; //neu: hier wird der Code gespeichert

void setup() {
  pinMode(rc_pin, OUTPUT);
  Serial.begin(9600); //neu: hier stellen wir die Baudrate fuer die serielle USB
Verbindung ein
}
void loop() {
  while(Serial.available() <= 0){ delay(10);} //Warten bis eine USB-Verbindung $
  while(Serial.read() != 'b'){ delay(10);} //Warten bis ein 'b' empfangen wird
  for(short i=0; i<12; i++){ //Nun werden die naechsten 12 Zeichen als Code erk$
    while(Serial.available() <= 0){ delay(10);} //Warten bis eine USB-Verbindun$
    code[i] = Serial.read();
  }
  Serial.println(code); //zur Kontrolle gibt der Arduino nun den Code nochmal w$
  sendCode(code); //Den Code senden
}
```

Zuerst wartet der Arduino auf ein ‚b' und wartet dann auf einen 12stelligen Code aus ‚1' und ‚0'. Alle anderen Zeichen werden ignoriert. Du kannst nun im Serialmonitor einfach z.B.

„b000110010001“ eingeben und dann wird der Code „00011 00100 01“ gesendet und anschliessend zur Kontrolle nochmal ausgegeben. Das ganze nun nochmal in komplett:

```
short rc_pin=13;
char code[12]; //neu: hier wird der Code gespeichert

void setup() {
  pinMode(rc_pin, OUTPUT);
  Serial.begin(9600); //neu: hier stellen wir die Baudrate fuer die serielle USB
Verbindung ein
}
void loop() {
  while(Serial.available() <= 0){ delay(10);} //Warten bis eine USB-Verbindung $
  while(Serial.read() != 'b'){ delay(10);} //Warten bis ein 'b' empfangen wird
  for(short i=0; i<12; i++){ //Nun werden die naechsten 12 Zeichen als Code erk$
    while(Serial.available() <= 0){ delay(10);} //Warten bis eine USB-Verbindun$
    code[i] = Serial.read();
  }
  Serial.println(code); //zur Kontrolle gibt der Arduino nun den Code nochmal w$
  sendCode(code); //Den Code senden
}

boolean sendCode(char code[]){ //empfange den Code in Form eines Char[]
  for(short z = 0; z<7; z++){ //wiederhole den Code 7x
    for(short i = 0; i<12; i++){ //ein Code besteht aus 12bits
      sendByte(code[i]);
    }
  sendByte('x'); //da der code immer mit x/sync abschliesst, brauchen wir den
nicht im code und haengen es autisch immer hinten ran.
  }
  return true;
}

void sendByte(char i) { //Diese Funktion soll 0,1 oder x senden koennen. Wir
speichern die gewuenschte Ausgabe in der Variabel i
  switch(i){ //nun gucken wir was i ist
  case '0':{ //Der Code fuer '0'
    digitalWrite(rc_pin,HIGH);
    wait(1); //da die Pausen x*350us lang sind, machen wir daraus eine Funktion
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
```

```
    wait(3);
    digitalWrite(rc_pin,LOW);
    wait(1);
    return;
  }
  case '1':{ //Der Code fuer '1'
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(3);
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(3);
    return;
  }
  case 'x':{ //Der Code fuer x(sync)
    digitalWrite(rc_pin,HIGH);
    wait(1);
    digitalWrite(rc_pin,LOW);
    wait(31);
  }

}
}

void wait(int x) {
  delayMicroseconds(x*350); //warte x*350us
}
```

Um das ganze jetzt per Tastenkombination oder aehnliches zu machen muessen wir hierraus erstmal einen Terminal-Befehl machen. Dies geht am einfachsten mit ttyUSB. Hierzu hatte ich bereits mal ein Tutorial geschrieben. Doch ich fasse es nochmal kurz hier zusammen:

Zuerst muessen wir den Arduino initalisieren. Initalisieren tun wir mit dem Befehl: stty -F /dev/ttyUSB0 cs8 9600 -hupcl (Pfad: /dev/ttyUSB0 Baudrate: 9600)

Anschliessend koennen wir mit echo ‘Code’ > /dev/ttyUSB0 senden. z.B. echo ‘b000110010001’ > /dev/ttyUSB0

Das Initalisieren muss nach jedem Systemstart einmal gemacht werden. Hier eignet sich ein Autostart. Den Sendebefehl kannst du nun ueberall einfuegen.