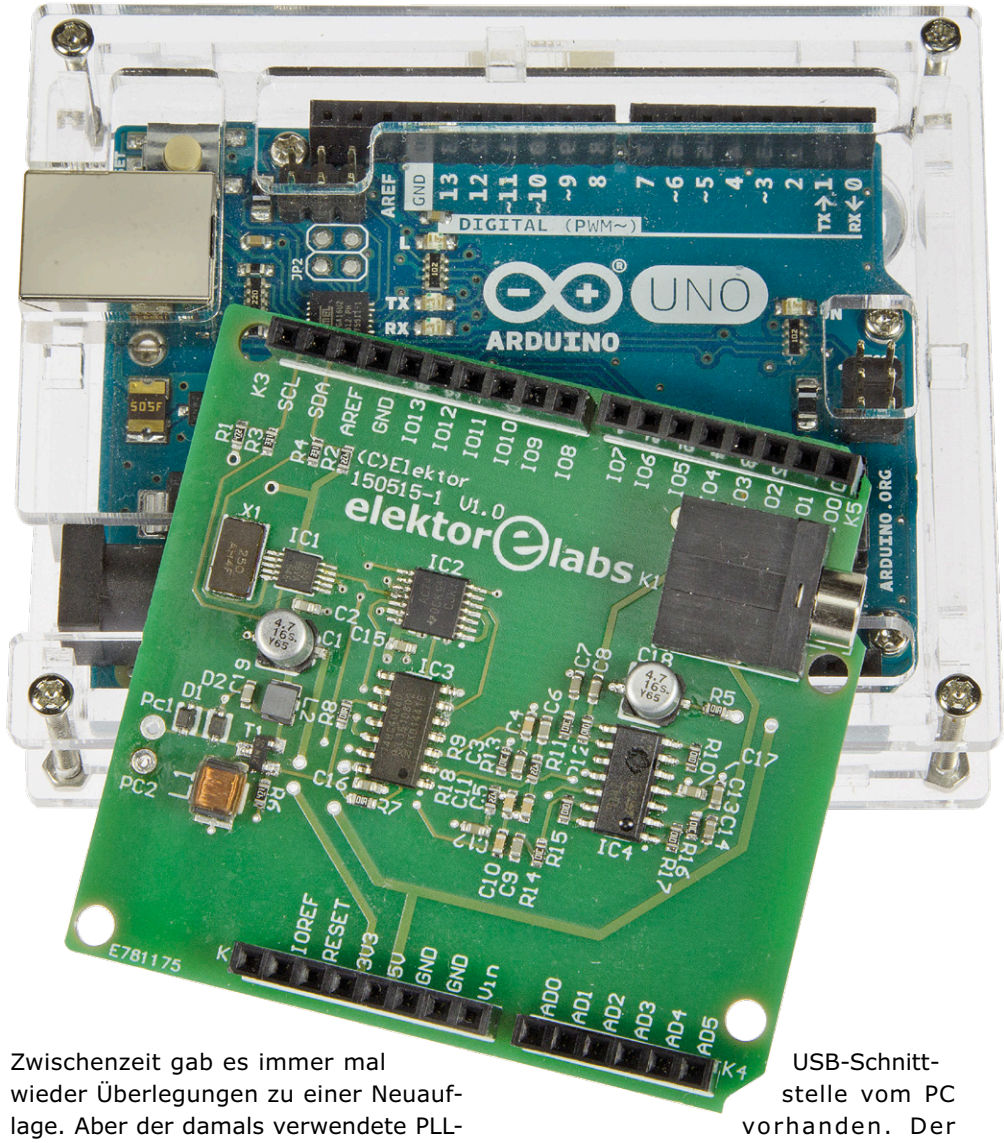
Elektor SDR Reloaded

SDR-Shield für Arduino

Von **Burkhard Kainka** (Deutschland)

Ein Software Defined Radio ist ein universelles Werkzeug in der HF-Technik, das auch für Messungen einsetzbar ist. Die Eigenschaften des Empfängers werden durch Software definiert. In unserer heutigen Zeit bietet es sich an, das Frontend als Arduino-Shield zu realisieren.



Eigenschaften

- Betriebsspannung: 5 V und 3,3 V vom Arduino
- Frequenzbereich: 150 kHz bis 30 MHz
- Empfindlichkeit: 1 µV
- Gesamtverstärkung: 40 dB
- Maximaler Antennenpegel: 10 mV
- Dynamikumfang: 80 dB

Auch wenn immer mehr Rundfunkdienste die AM-Bereiche Lang-, Mittel- und Kurzwelle verlassen, bleibt es interessant, mit eigenen Empfängern auf die Wellenjagd zu gehen. Jetzt erst recht, könnte man sagen, denn viele weit entfernte Stationen treten nun erst deutlich hervor, weil sie nicht mehr von deutlich stärkeren Signalen verdrängt werden. Oft ist es auf der Kurzwelle so still, das man meinen möchte, der Empfänger wäre taub. In manchen Bereichen produzieren dann die Funkamateure die stärksten Signale. Und man findet immer wieder Neues, von Piratensendern über SSB-Sprechfunk bis zu den neuen digitalen Betriebsarten. Das macht neugierig!

Elektor hat schon viele Radios und Empfänger gebaut. Ein Software Defined Radio mit USB-Schnittstelle wurde schon in 2007 [1] vorgestellt. In der Zwischenzeit gab es immer mal wieder Überlegungen zu einer Neuauflage. Aber der damals verwendete PLL-Chip wird nicht mehr gebaut. Deshalb musste eine neue Lösung her. Sie kam in Form des Silicon Lab Chips SI5351, eines CMOS-Clock-Generators von 8 kHz bis 160 MHz mit I²C-Bus.

Die ersten Versuche gelangen mit einem Breakout-Board von Adafruit. Die vorhandenen Software-Beispiele waren für den Arduino geschrieben. Und deshalb wurden auch die ersten Gehversuche mit dem Arduino unternommen. Der neue VFO wurde einfach an die alte SDR-Platine angeklemmt und konnte seine Tauglichkeit beweisen (**Bild 1**).

Und so entstand die Idee: Warum nicht gleich den ganzen Empfänger als Arduino-Shield bauen? Dann ist schon mal die Stromversorgung geklärt und die USB-Schnittstelle vom PC vorhanden. Der Arduino übernimmt die Ansteuerung des VFO und kann sozusagen im Klartext angesprochen werden (bitte einmal 6030 kHz). Und was vielleicht noch wichtiger ist, so hat man sogar eine reelle Chance, einen Stand-Alone-Empfänger zu bauen. Die Bedienung könnte relativ leicht vom PC auf den Arduino wandern. Und wer weiß, vielleicht auch eines Tages die Dekodierung des IQ-Signals?

Wie funktioniert's?

Aber jetzt noch mal ganz von vorn. Was ist überhaupt ein Software Defined Radio? Von der Entwicklung der digitalen Elektronik blieben die Radios lange Zeit völlig unberührt. Als es schon Homecomputer gab, waren die meisten Radios noch

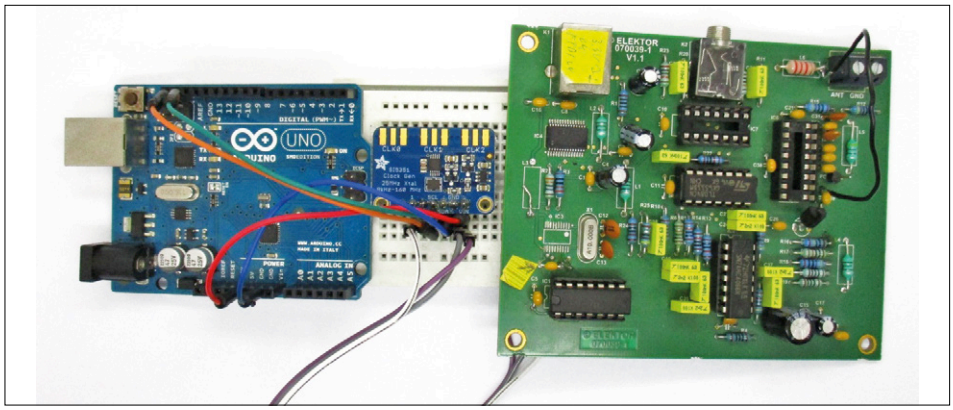


Bild 1. Der erste Vorversuch für einen SDR2: Ein SI5351 PLL-Chip angeschlossen an einen Arduino Uno und den „alten“ SDR-Empfänger.

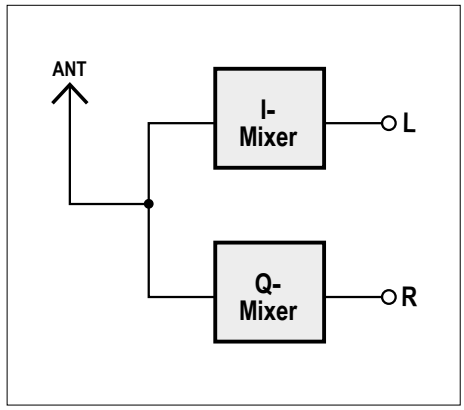


Bild 2. Prinzipschaltbild: Das Frontend besteht aus einem doppelten Direktmischer mit um 90 Grad phasenverschobenen Signalen.

analog. Dann setzte allmählich eine Entwicklung ein, zumindest die Abstimmung zu digitalisieren. Heutige Radios sind oft mit einem PLL-Synthesizer ausgestattet, denn das vereinfacht die Abstimmung und garantiert ein genaues Einhalten der Kanalraster. Der Rest der Schaltung ist aber nach wie vor analog.

Inzwischen ist die digitale Elektronik in der kommerziellen HF-Technik und im Amateurfunk angekommen. Immer mehr analoge Funktionen der Geräte werden durch Software ersetzt. Meist arbeitet ein Digitaler Signal-Prozessor (DSP) mit der passenden Software weitgehend unsichtbar für den Benutzer und sorgt für optimale Filterkurven, variable Bandbreite, Signaldekodierung, Entstörung und vieles mehr. Die Geräte werden damit insgesamt besser bei weniger Hardware-Aufwand. Andere Beispiele für diese Entwicklung sind Smartphones und andere mobile Endgeräte. Da sieht man gleich, da kann kein Hobby-Elektroniker mehr mithalten. Aber eigentlich muss es gar nicht so aufwendig sein. Man braucht nur einen schnellen AD-Wandler direkt an der Antenne. Das ganze Spektrum wird digitalisiert und dann digital weiter verarbeitet. So etwas gibt es tatsächlich schon für den ganzen Bereich von 0 bis 30 MHz. Erst die Software filtert bestimmte Frequenzen heraus und demoduliert das gewünschte Signal. Ganz egal, ob es um AM- oder DRM-Rundfunksender geht oder ob SSB-Signale, CW-Morsesender, Fernschreiber, Wetterfax oder anderes empfangen werden soll, alles ist möglich, und für alles gibt es die passende Software. Allerdings ist die Hardware bei einer so großen Bandbreite recht teuer, und die Weiterverarbeitung des breiten Spektrums stellt hohe Anforderungen.

Einen Ausweg bietet die Soundkarte moderner PCs. Bei einer heute üblichen Abtastrate von 96 kHz kann man bereits den ganzen Bereich bis 48 kHz empfangen. Statt eines Mikrofons schließt man einfach eine große Spule als Antenne an, und schon kann man den VLF-Bereich empfangen. Da gibt es viele interessante Signale bis hin zum U-Boot-Sender.

Wenn man die Soundkarte für höhere Frequenzen verwenden will, muss man die Signale zuerst heruntermischen. Der Aufbau entspricht dann einem Superhet mit einer niedrigen Zwischenfrequenz. Die ZF-Stufen, Filterung, automatische Verstärkungsregelung und die Demodulation übernimmt der PC. Im Prinzip reicht dafür ein einfacher Direktmischer mit einem Dioden-Ringmischer oder dem bekannten NE612. Nur ein stabiler variabler Oszillator (VFO) wird zusätzlich gebraucht. Für spezielle Anwendungen nimmt man einen Quarzoszillator. Aber wenn man den ganzen Bereich abstimmen können möchte, sollte es ein DDS-Generator oder ein PLL-Baustein ein.

Bei einem IQ-Mischer handelt sich vereinfacht gesagt um einen doppelten Direktmischer mit zwei um 90 Grad phasenverschobenen Signalen. Das Oszillatorsignal liegt immer nahe bei der Empfangsfrequenz. Die Ausgangssignale liegen daher im NF-Bereich, meist zwischen 0 kHz und 24 kHz. Die beiden Signale werden mit I und Q bezeichnet (siehe **Bild 2**). Man gibt

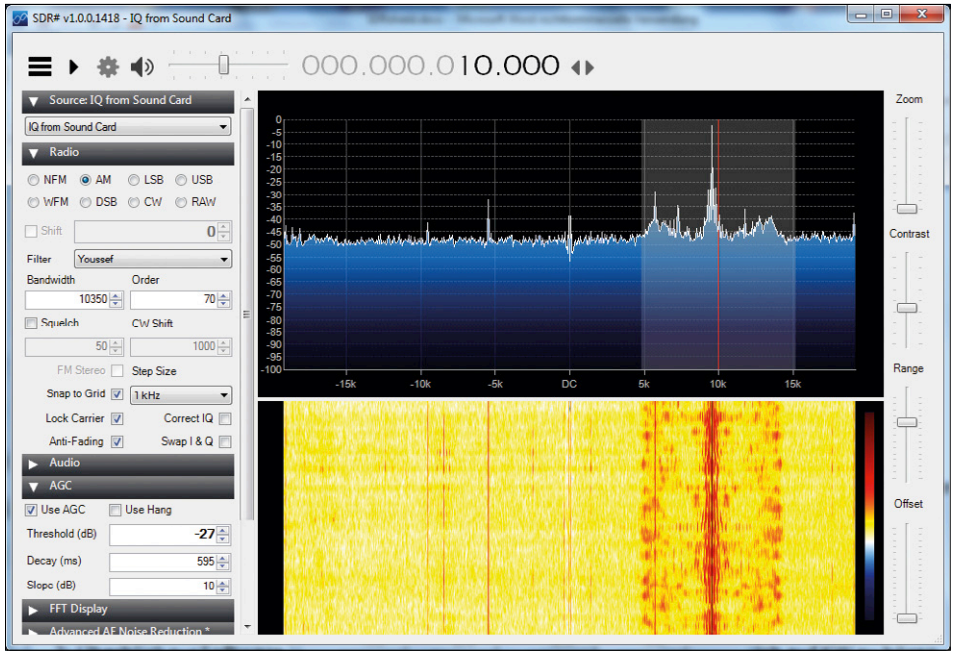


Bild 3. Das Programm SDR# beim Empfang eines AM-Signals.

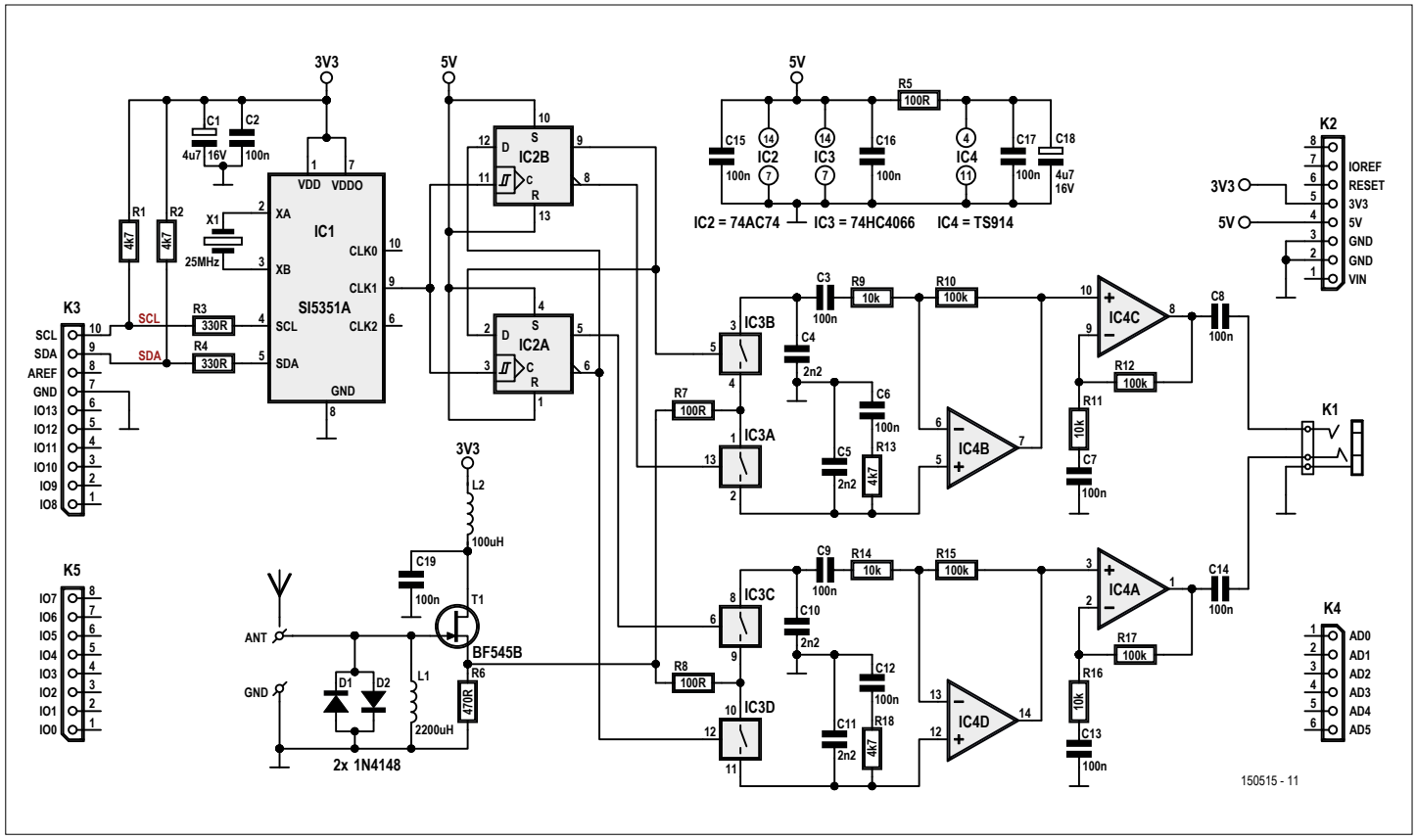


Bild 4. Schaltplan des neuen SDR-Empfängers.

sie direkt auf den linken und den rechten Kanal eines Soundkarten-Eingangs. Der Rest wird dann per Software erledigt. Ein einfacher Mischer würde den Bereich unterhalb und oberhalb der Oszillatorfrequenz in denselben Bereich mischen. Es gäbe also ein Spiegelfrequenz-Problem. Durch die doppelte Mischung mit einer Phasenverschiebung kann jedoch die Software die Spiegelfrequenz herausrechnen. So kann ein Bereich zwischen –24 kHz und +24 kHz empfangen wer-

Stückliste

Widerstände:

R1,R2,R13,R18 = 4k7, 1 %/100 mW, SMD 0603
R3,R4 = 330 Ω, 1 %/100 mW, SMD 0603
R5,R7,R8 = 100 Ω, 1 %/100 mW, SMD 0603
R6 = 470 Ω, 1 %/100 mW, SMD 0603
R9,R11,R14,R16 = 10 k, 1 %/100 mW, SMD 0603
R10,R12,R15,R17 = 100 k, 100 mW, SMD 0603

Kondensatoren:

C1,C18 = 4µ7/16 V, SMD case B
C2,C3,C6,C7,C8,C9,C12,C13,C14,C15,C16,C17 ,C19 = 100 n/50 V, X7R, SMD 0603
C4,C5,C10,C11 = 2n2/50 V, X7R, SMD 0603

Induktivitäten:

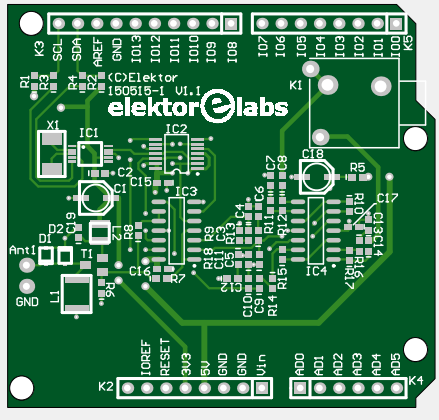
L1 = 2200 µH (Fastron L-1812AF)
L2 = 100 µH (Murata LQH32CN101K23L)

Halbleiter:

D1,D2 = 1N4148WS, SOD-323
T1 = BF545B, SOT-23
IC1 = SI5351A-B-GT, MSOP-10
IC2 = SN74AC74PW, TSSOP-14
IC3 = 74HC4066, SOIC-14
IC4 = TI914IDT, SOIC-14

Außerdem:

K1 = Stereo-Klinkenbuchse 3,5 mm für Platinenmontage
K2,K3,K4,K5 = 1 Satz Steckverbinder, Arduino-kompatibel (1·6 Pins, 2·8 Pins, 1·10 Pins)
X1 = 25-MHz-Quarz (Abracon ABM7)
Platine 150515-1
Oder
Platine mit vormontierten SMDs: 150515-91



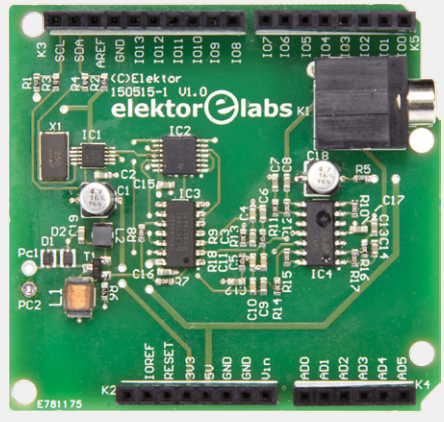


Bild 5. Die doppelseitige Platine für den SDR3-Empfänger ist als Arduino-Shield ausgeführt.

den, wenn die Soundkarte eine Abtastrate von 48 kHz hat. **Bild 3** zeigt, was ein Programm wie SDR# daraus macht (siehe auch Textkasten „SDR-Software“).

Schaltplan

Ein Blick auf den Schaltplan in **Bild 4** zeigt die einzelnen Baugruppen. Der PLL-Generator SI5351 (IC1) liefert das Oszillatorsignal mit der vierfachen Empfangsfrequenz an den Teiler 74AC74 (IC2B). Dieser teilt die Frequenz durch vier und liefert die um 90 Grad phasenverschobenen Signale an den Mischer 74HC4066 (IC3). Dieser Analogschalter ist wie ein Umschalter verdrahtet und legt das HF-Signal abwechselnd an den invertierenden und den nicht-invertierenden Eingang des Operationsverstärkers TS914 (IC4B/IC4D). Damit wird das Signal in den NF-Bereich heruntergemischt. Nach einer geringen Filterung und Verstärkung (IC4C/IC4A) gelangt das Signal an den Audio-Ausgang. Die HF-Eingangsstufe bildet ein Source-Folger mit dem JFET BF545B (T1), das SMD-Äquivalent des BF245B.

Wer den alten Elektor-SDR kennt, sieht eine gewisse Vereinfachung im Signalzweig. Am HF-Eingang gab es damals mehrere umschaltbare Tiefpassfilter. Jetzt ist der Eingang breitbandig und mit zwei Dioden gegen Überspannung geschützt. Das reicht für den Kurzwellenempfang mit einer Drahtantenne völlig aus. Der Überspannungsschutz ist der Erfahrung mit dem ersten SDR geschuldet; bei einem Gewitter können die Eingangsstufen beschädigt werden. Für besondere Aufgaben kann man noch externe Filter und Vorverstärker verwenden. Und die NF-Verstärkung konnte beim alten Entwurf in drei Stufen eingestellt werden. Diesmal gibt es nur die mittlere Verstärkung, die sich als allgemein nützlich erwiesen hat. So ist alles etwas einfacher geworden und passt nun problemlos auf das Shield.

Für die ersten Versuche muss man einfach nur eine Drahtantenne anschließen. Ideal ist ein frei abgespannter Antennendraht mit einer Länge ab drei Metern. Wenn das nicht möglich ist, bringt auch schon ein längerer Draht etwas, der einfach nur irgendwie im Zimmer liegt. Allerdings zeigen Innenantennen meist einen schlechteren Störabstand. Der Bau optimierter Antennen wird noch ein Thema in Elektor sein.

Aufbau

Die Platine (**Bild 5**) ist als Arduino-Shield ausgeführt, so dass man sie direkt auf einen Arduino Uno stecken kann. Da der Si5351 nur in einem sehr kleinen 10-poligen SMD-Gehäuse erhältlich ist, haben wir uns entschlossen die ganze Schaltung mit SMDs aufzubauen und die Platine bestückt im Elektor-Shop anzubieten [3]. Man braucht nur noch die vier Arduino-kompatiblen Steckverbinder auf die Platine zu löten. Wer die Platine lieber komplett selbst aufbauen möchte, kann von [3] das Platinen-Layout downloaden oder eine separate Platine im Elektor-Shop kaufen.

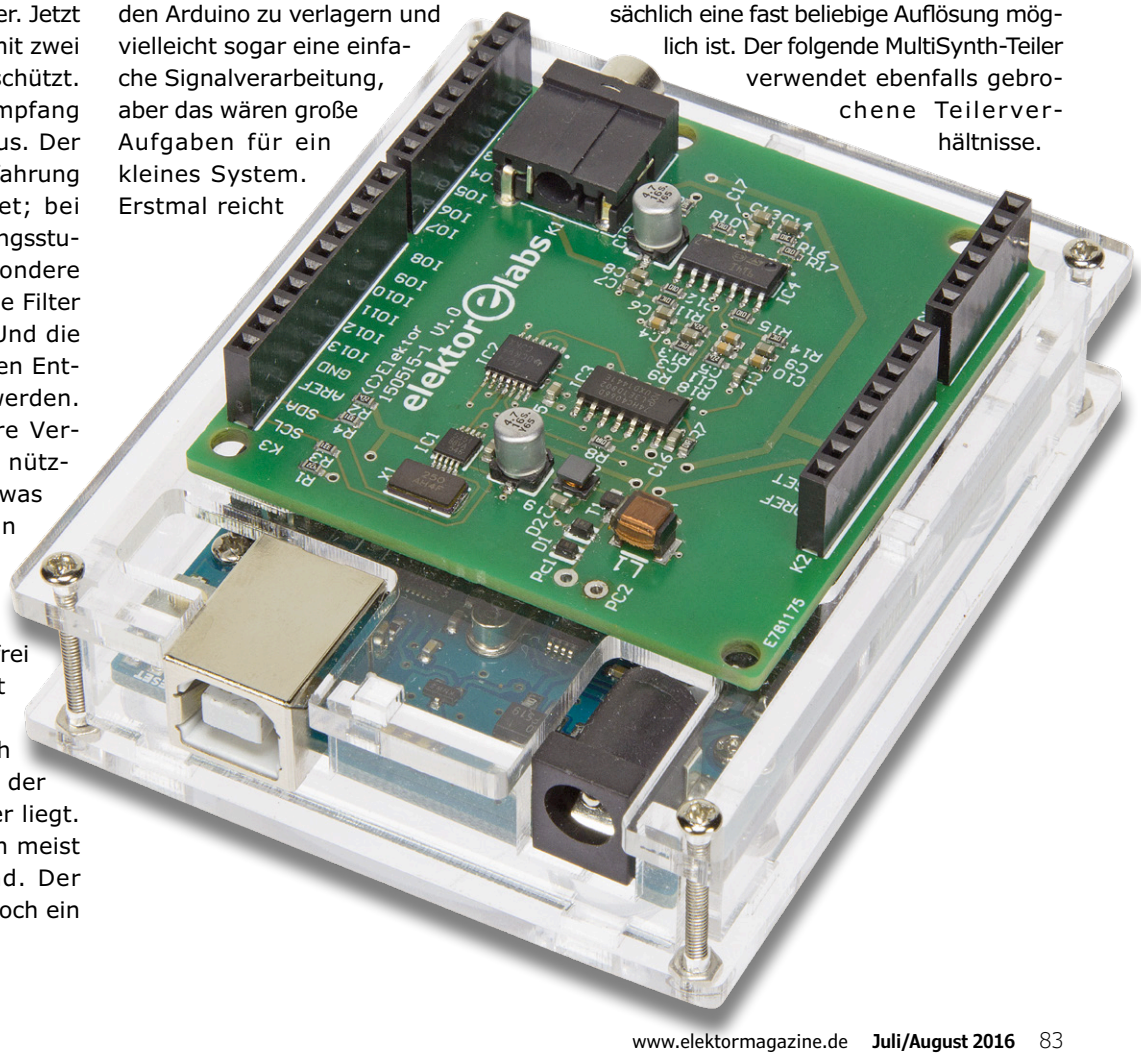
Frequenz einstellen

Der Arduino dient hier zusammen mit dem SDR-Shield als Interface zwischen Antenne und PC. Seine Aufgabe besteht allein darin, den VFO abzustimmen. Und dazu sagt ihm ein PC-Programm, welche Frequenz gerade gewünscht ist. Vom PC zum Arduino gelangen Informationen über die USB-Leitung. Das heruntergemischte Nutzsignal geht dann zur weiteren Verarbeitung über ein Stereokabel an den Soundkarten-Eingang. Zwar könnte man versuchen, auch die Bedienung auf den Arduino zu verlagern und vielleicht sogar eine einfache Signalverarbeitung, aber das wären große Aufgaben für ein kleines System. Erstmal reicht es, wenn der Arduino Kommandos vom PC empfängt und den VFO einstellt.

Wie man mit dem Arduino umgeht, soll nicht Thema dieses Beitrags sein. Der Umgang mit der Arduino-IDE wird vorausgesetzt. Denn zuerst muss ein passendes Arduino-Programm geladen werden. Was genau in der Software passiert, wird im Folgenden erläutert. Wer aber lieber schnell zu den ersten Empfangsergebnissen kommen möchte, kann diese Informationen überspringen und einfach die Software laden [3].

Die entscheidende Aufgabe besteht darin, den SI5351 zu überreden, eine passende Frequenz zu generieren. Das IC hat zwei interne PLLs und drei Ausgänge (siehe Blockschaltbild, **Bild 6**). Verwendet wird hier nur die PLLA und der Ausgang CLK1. Der Sketch verwendet die Library von Adafruit, was die ganze Sache erfreulich einfach macht. Bevor man loslegen kann, muss die Library von [1] geladen und eingebunden werden.

Der SI5351 hat einen Quarzoszillator mit 25 MHz und zwei PLLs, die zwischen 600 MHz und 900 MHz eingestellt werden können. Die PLL-Teiler arbeiten mit gebrochenen Teilerverhältnissen, so dass tatsächlich eine fast beliebige Auflösung möglich ist. Der folgende MultiSynth-Teiler verwendet ebenfalls gebrochene Teilerverhältnisse.



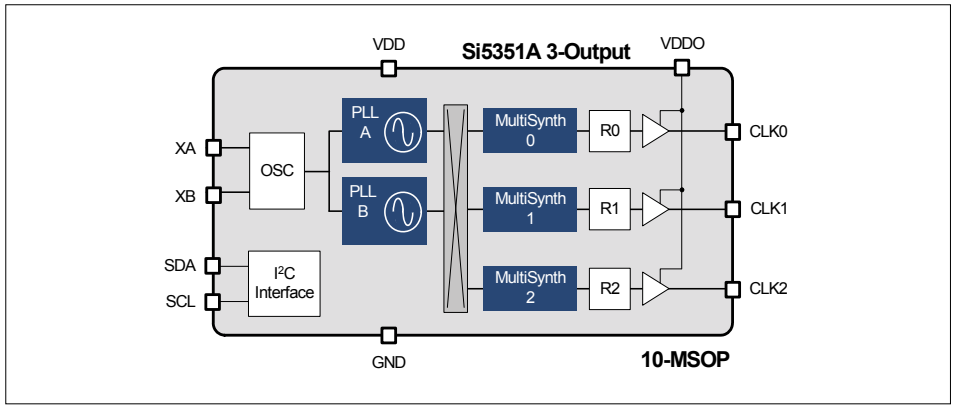


Bild 6. Blockschaltbild aus dem Datenblatt des Si5351A.

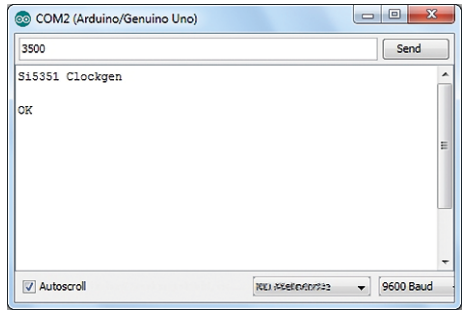


Bild 7. Ansteuerung des Clock-Generators mit dem Arduino-Terminal.

Damit hat man zwei Möglichkeiten, die Wunschfrequenz zu erzeugen:

- Man kann die PLL fest auf z.B. 900 MHz einstellen und dann mit gebrochenen Zahlen herunterteilen.
- Man kann die PLL kleinschrittig einstellen und dann ganzzahlig auf die Endfrequenz teilen.

Hier zunächst die Methode A. Die VFO-Frequenz liegt beim Vierfachen der Mischerfrequenz, die 12 kHz unterhalb der Empfangsfrequenz steht. Das Programm soll die Empfangsfrequenz in kHz im Textformat empfangen und umsetzen. Um 3500 kHz zu empfangen muss der Si5351 eine Ausgangsfrequenz von 4 · (3500 - 12) kHz = 13952 kHz am Ausgang 1 erzeugen. Der PLL-Teiler wird auf 36 gestellt (25 MHz · 36 = 900 MHz), der Multisynth-Teiler auf 900000/13952 = 64,506. Mit dieser Methode kommt

Listing 1. Programm für fest eingestellte PLL.

```
//SI5351_vfo  PLL fixed at 900 MHz (si5351vfo2.zip)

#include <Adafruit_Sensor.h>
#include <Wire.h>
#include <Adafruit_SI5351.h>

Adafruit_SI5351 clockgen = Adafruit_SI5351();
void setup(void)

{
  Serial.begin(9600);
  Serial.println("Si5351 Clockgen"); Serial.println("");


  /* Initialise the sensor */
  if (clockgen.begin() != ERROR_NONE)
  {
    Serial.print("Error");
    while(1);
  }
  Serial.println("OK");
  clockgen.enableOutputs(true);
  clockgen.setupPLL(SI5351_PLL_A, 36, 0, 1000);  //900
MHz
  setfreq (6000);
}

void setfreq (unsigned long freq)
{
  unsigned long f2;
  unsigned long f3;
  unsigned long f4;
  unsigned long f5;
  unsigned long div2;
  unsigned int Divider2;
  unsigned int rdiv;

   if (freq > 0)    {
    f2=(freq-12)*4;
    if (f2<1000) {
      rdiv = 16;
      f2 = f2 * 16;
      }
    else  {
      rdiv = 1;
      }
    div2 = 900000000/f2;
    f4 = div2/1000;
    f5=div2-(f4*1000);
    clockgen.setupMultisynth(1, SI5351_PLL_A, f4, f5,
1000);
    if (rdiv == 16) {
      clockgen.setupRdiv(1, SI5351_R_DIV_16);
      }
    if (rdiv == 1) {
      clockgen.setupRdiv(1, SI5351_R_DIV_1);
      }
   }
}

void loop(void)
{
  unsigned long freq;
  if (Serial.available()) {
   freq = Serial.parseInt();
   setfreq (freq);
   }
}
```

man runter bis 1 MHz. Für noch kleinere Frequenzen wird der zusätzliche R_DIV-Teiler durch 16 eingesetzt. **Listing 1** zeigt die zugehörige Software für den Arduino, **Bild 7** die Ansteuerung mit dem Arduino-Terminal.
Die Methode A hat den Vorteil, dass der VFO quasi kontinuierlich abgestimmt werden kann, d.h. es gibt keine Unterbrechung bei einem Frequenzwechsel.
Die Methode B verspricht dagegen eine größere Phasenreinheit, die auch für DRM ausreicht. Allerdings wird jeder Frequenzwechsel mit einer kurzen Unterbrechung von etwa einer Millisekunde begleitet, die als Störsignal im SDR erscheint. Die Methode erfordert eine Berechnung des optimalen Nachteilers (**Listing 2**), damit die PLL immer im Bereich 600 MHz bis 900 MHz bleibt.
Beide Programme können aus einem beliebigen Terminal heraus angesteuert werden. Für eine bequemere Bedienung wurde jedoch in Visual Studio 2015 ein VB-Programm geschrieben (SDRshield.zip, Download unter [3]). Es sendet die Wunschfrequenz mit 9600 Baud im Textformat (z.B. 3500) an den Arduino. Der Schieberegler (siehe **Bild 8**) verwendet im Bereich bis 1,6 MHz Schritte von 9 kHz und darüber das Raster 5 kHz. Zusätzlich kann man eine Wunschfrequenz direkt eingeben oder mit den Bänder-Schaltflächen auf den Anfang der einzelnen Rundfunk- oder Amateurfunkbänder klicken. Achten Sie darauf, dass beim ersten Gebrauch der richtige COM-Port gewählt wird.

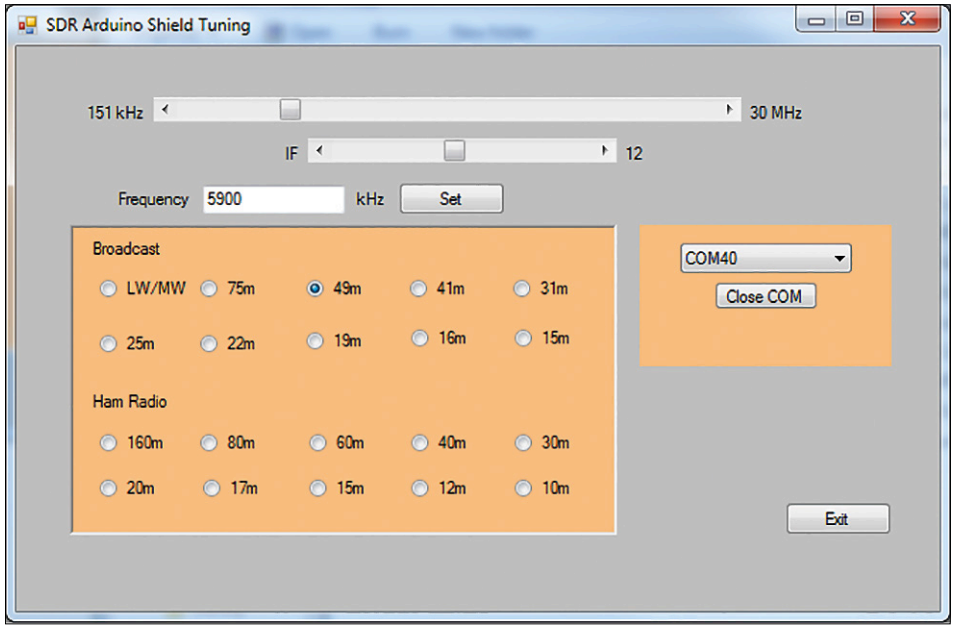


Bild 8. Ein kleines VB-Programm sorgt für eine bequeme Bedienung.

SDR-Software

Hier ein Überblick der verwendbaren SDR-Software. Praktisch alle Programme, die mit dem alten Elektor-SDR verwendet wurden, funktionieren auch jetzt noch.

- SDRadio ist immer noch eine gute Wahl
- SoDoRa kann auch DRM dekodieren
- DREAM funktioniert noch, verwendet aber nicht das IQ-Signal und nutzt den Empfänger wie einen Direktmischer
- HDSDR ist eine aktuelle und sehr mächtige Software
- SDRSharp (SDR#) zeichnet sich durch einfache Bedienung und gute Darstellung aus

In einem weiteren Artikel werden wir die einzelnen Programme und ihre Möglichkeiten detailliert besprechen.

Erste Empfangserfahrungen

Wenn keine bessere Antenne in Reichweite ist, kann man für die ersten Versuche einfach einen Draht von ein bis drei Metern Länge an den Antenneneingang anschließen. Damit empfängt man problemlos Rundfunksender auf allen Kurzwellenbändern. Die Erfahrung zeigt, dass am Abend mehr los ist als am Tage. Und der Schwerpunkt wandert am Abend hin zu den tieferen Bändern 75 m bis 41 m. Auch Amateurfunkstationen lassen sich bereits mit einer kurzen Drahtantenne empfangen. Meist hat man im 40-m-Band am meisten Glück und kann einige CW- und SSB-Stationen hören. Die passende Betriebsart wählt man in der SDR-Software aus, dazu Lautstärke, Bandbreite, ALC-Einstellungen und vieles mehr. Oft lässt sich mit den richtigen Einstellungen mehr herausholen als mit einem teuren analogen Empfänger älterer Bauart.
Eine grundlegende Eigenschaft aller schaltenden Mischer ist es, das sie auch Sig-

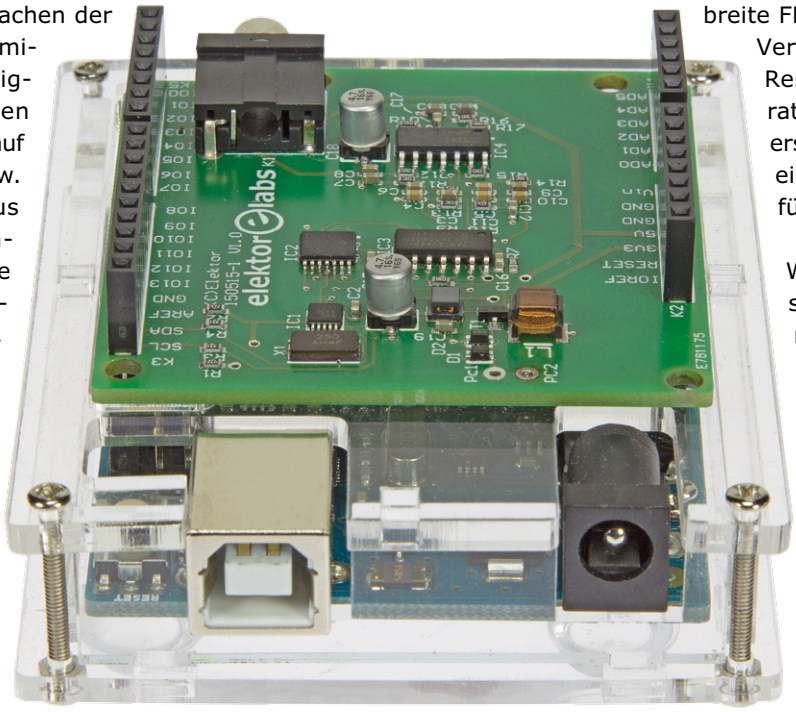
SDR-Software

Die aktuellen Sterne am Himmel der SDR-Software heißen SDR# [4] und HDSDR [5]. Beide Programme folgen dem neuen Trend zu immer höheren Frequenzen und lassen sich auch mit einfachen DVBT-Dongeln betreiben. Das ist eine gute Wahl, wenn man sich im VHF-und UHF-Bereich umsehen will. Es gibt auch Versuche, solche Hardware unterhalb 30 MHz einzusetzen. Man kann z.B. einen Up-Mischer verwenden, der alles um 50 MHz nach oben setzt. Dann hat man allerdings einen Mehrfach-Super mit den bekannten Problemen wie zahlreiche Phantomsignale und Nebenempfangsstellen sowie eine eingeschränkte Dynamik. Ein SDR speziell für den Bereich bis 30 MHz mischt jedoch nur einmal und liefert daher einen sehr sauberen Empfang ohne Pfeifstellen.
Am PC laufen zwei Programme, nämlich das Abstimm-Programm und die SDR-Software. Jedes SDR-Programm hat seine eigene Bedienung. Die ersten Schritte sind aber immer ähnlich. Zuerst muss man sicherstellen, dass der richtige Eingang verwendet wird. Dazu muss die Soundkarte gewählt werden, und deren verwendeter Eingang (Line In) aktiviert werden. Dann startet man die SDR-Software. Dass man den richtigen Eingang ausgewählt hat, sieht man an einem deutlichen Anstieg des Grundrauschens, das mit dem Anschluss der Antenne noch einmal deutlich ansteigen sollte. Meist müssen die Soundkarten-Regler auf eine reduzierte Lautstärke eingestellt werden, weil der Empfänger Signale bis zu einem Volt liefern kann.

Listing 2. Programm für variable PLL.

```
//SI5351_vfo, variable PLL (si5351vfo3.zip)

#include <Adafruit_Sensor.h>

#include <Wire.h>
#include <Adafruit_SI5351.h>

Adafruit_SI5351 clockgen = Adafruit_SI5351();
void setup(void)

{
  Serial.begin(9600);
  Serial.println("Si5351 VFO"); Serial.println("");

  if (clockgen.begin() != ERROR_NONE)
  {
    Serial.print("Error");
    while(1);
  }
  Serial.println("OK");
  clockgen.enableOutputs(true);
  setfreq (6000);
}

void setfreq (unsigned long freq)
{
  unsigned long f2;
  unsigned long f3;
  unsigned long f4;
  unsigned long f5;
  unsigned int Divider2;
  unsigned int rdiv;

   if (freq > 0)
   {
   f2=(freq-12)*4;
   // f2=freq;
    if (f2>120000) {
      f2=120000;
    }
    if (f2<800) {
       rdiv = 16;
       f2 = f2 * 16;
    }
    else  {
       clockgen.setupRdiv(1, SI5351_R_DIV_1);
       rdiv = 1;
    }
    if (f2 >= 100000) {
      Divider2 = 6;
     }
     if (f2 < 90000) {
      Divider2 = 10;
     }
     if (f2 < 60000) {
      Divider2 = 15;
     }
     if (f2 < 50000) {
      Divider2 = 18;
     }
     if (f2 < 45000) {
      Divider2 = 20;
     }
     if (f2 < 30000) {
      Divider2 = 30;
     }
     if (f2 < 20000) {
      Divider2 = 45;
     }
     if (f2 < 15000) {
      Divider2 = 60;
     }
     if (f2 < 10000) {
      Divider2 = 90;
     }
     if (f2 < 6000) {
      Divider2 = 150;
     }
    if (f2 < 4000) {
      Divider2 = 220;
     }
    if (f2 < 2700) {
      Divider2 = 330;
     }
    if (f2 < 1800) {
      Divider2 = 500;
     }
    if (f2 < 1500) {
      Divider2 = 600;
     }
    if (f2 < 1000) {
      Divider2 = 900;
     }
    f2=f2*Divider2;
    f2=f2*1000/25;
    f3=f2 /1000;
    f4 = f3/1000;
    f5=f3-(f4*1000);
    clockgen.setupPLL(SI5351_PLL_A, f4, f5, 1000);
    clockgen.setupMultisynth(1, SI5351_PLL_A, Divider2,
0, 2);
    if (rdiv == 16) {
      clockgen.setupRdiv(1, SI5351_R_DIV_16);
    }
  }
}

void loop(void)
{
  unsigned long freq;
  if (Serial.available()) {
   freq = Serial.parseInt();
   setfreq (freq);
  }
}
```

nale auf ungeraden Vielfachen der Grundfrequenz heruntermischen. Wenn man ein Signal bei 1 MHz empfangen möchte, könnten Signale auf 3 MHz, 5 MHz, 7 MHz usw. den Empfang stören. Aus diesem Grunde verwendet man oft umschaltbare Tiefpassfilter. Das SDR-Shield verzichtet darauf, sodass der Einsatz selektiver Antennen sinnvoll ist. Trotzdem funktioniert es auch mit einer breitbandigen Drahtantenne erstaunlich gut. Das liegt daran, dass zu bestimmten Tageszeiten starke Signale auf unterschiedlichen Bändern dominieren und sich deshalb nicht in die Quere kommen. Eine Ausnahme ist der Empfang auf Lang- und Mittelwelle, der durch Signale aus dem Kurzwellenbereich beeinträchtigt wird. Aber man kann einfach eine Ferritantenne mit einem Drehkondensator anschließen, dann ist das Problem beseitigt.



Das Thema Antennen, Filter und Vorverstärker soll noch genauer behandelt werden. Dabei geht es nicht nur um möglichst große Signalspannungen, sondern besonders um den erreichbaren Störabstand. Es geht nichts über eine möglichst weit weg vom Haus aufgehängte, lange Drahtantenne. Aber weil das nicht überall möglich ist, muss man nach Kompromissen suchen. Und da gehört die magnetische Loop-Antenne zu den klaren Gewinnern. So lassen sich relativ kleine und unauffällige Antennen auch in Innenräumen betreiben. Aber dazu später mehr.

Bei den ersten Versuchen mit diesem Empfänger wird vermutlich eine Frage besonders interessieren: Sollte nicht der Arduino selbst Empfangsstörungen verursachen? Immerhin ist er ja ganz nah dabei. Beim Entwurf der Platine wurde große Aufmerksamkeit auf eine gute Entkopplung gelegt. Es gibt eine durchgehende Massefläche auf der Unterseite der Platine; die Betriebsspannungen 5 V und 3,3 V werden über LC-Filter entkoppelt. Und tatsächlich waren diese Maßnahmen sehr erfolgreich. Im Normalfall merkt man nichts vom Arduino.

Arduino belauscht

Aber wenigstens den Taktoszillator bei 16 MHz sollte man doch „empfangen“ können. Das geht tatsächlich schon, wenn man überhaupt keine Antenne anschließt. Das Shield kann dann gleich mal seine Fähigkeiten als Messgerät zeigen. Tatsächlich findet man gleich zwei Oszillatoren. Der eine ist der 16-MHz-Quarzoszillator am USB-Chip des Uno mit einer Abweichung unter 1 kHz. Wenn man von unten die Uno-Platine an der Stelle berührt, wo der Quarz eingelötet ist, gibt es eine kleine Verstimmung. Dann weiß man, dass es das besagte Signal war. Mit einem Stückchen Draht am Antenneneingang wird das Signal stärker, aber auch das Grundrauschen. Und bei der Gelegenheit sieht man noch eine Besonderheit. Signale, die über den Antenneneingang reinkommen, haben eine gute Spiegelfrequenzunterdrückung. Aber bei Signalen, die sich über die Betriebsspannung in den Signalweg schleichen, ist das anders. Sie erscheinen doppelt, wenn auch wesentlich schwächer.

Das Taktsignal des Mega328 muss man erstmal suchen. Er arbeitet nämlich mit einem Keramik-Oszillator, der Abweichungen bis 50 kHz zeigen kann. Und tatsächlich, bei 15950 kHz war ein schwaches Signal zu finden, gleich mit einigen Seitenbandsignalen, die der Controller beisteuert. Und wenn man die Arduino-Platine an der Stelle des Keramik-Resonators berührt, gibt es zusätzlich noch eine breite FM-Modulation und eine weitere Verstimmung, die zeigt dass der Resonator eine gewisse Temperaturabhängigkeit hat. Da musste erst ein SDR her, um dem Arduino einmal so genau auf den Zahn zu fühlen!

Wenn keine Antenne angeschlossen ist, dreht ein SDR im Normalfall seine Verstärkung so weit auf, dass auch kleinste Störsignale sichtbar werden. Dann sieht man vor allem um die Mittenfrequenz herum schwache Störungen, die vom USB und vom Arduino erzeugt werden. Wenn man unterscheiden möchte, welche Signale von Arduino und welche vom USB kommen, kann man einmal ein Netzteil an den Arduino anschließen und nach der Abstimmung auf eine Wunschfrequenz im laufenden Betrieb das USB-Kabel abziehen.

Alle internen Störsignale sind sehr schwach. Sobald eine Antenne angeschlossen wird, steigt das Grundrauschen soweit an, dass diese Störungen völlig überdeckt werden. Das zeigt die hohe Empfindlichkeit des SDR. Auch Signale mit nur einem Mikrovolt können empfangen werden. Meist ist so viel Empfindlichkeit aber gar nicht nötig, weil der Rauschpegel an der Antenne wesentlich höher liegt. Bei langen Antennen kann es sogar zu einer Übersteuerung des Empfängers kommen. Dann muss man bereits über einen Eingangs-Abschwächer nachdenken. ⏮

(150515)

Weblinks

[1] www.elektormagazine.de/070039

[2] https://github.com/adafruit/Adafruit_Si5351_Library

[3] www.elektormagazine.de/150515

[4] http://airspy.com/download

[5] http://www.hdsdr.de