

# Elektor SDR Reloaded

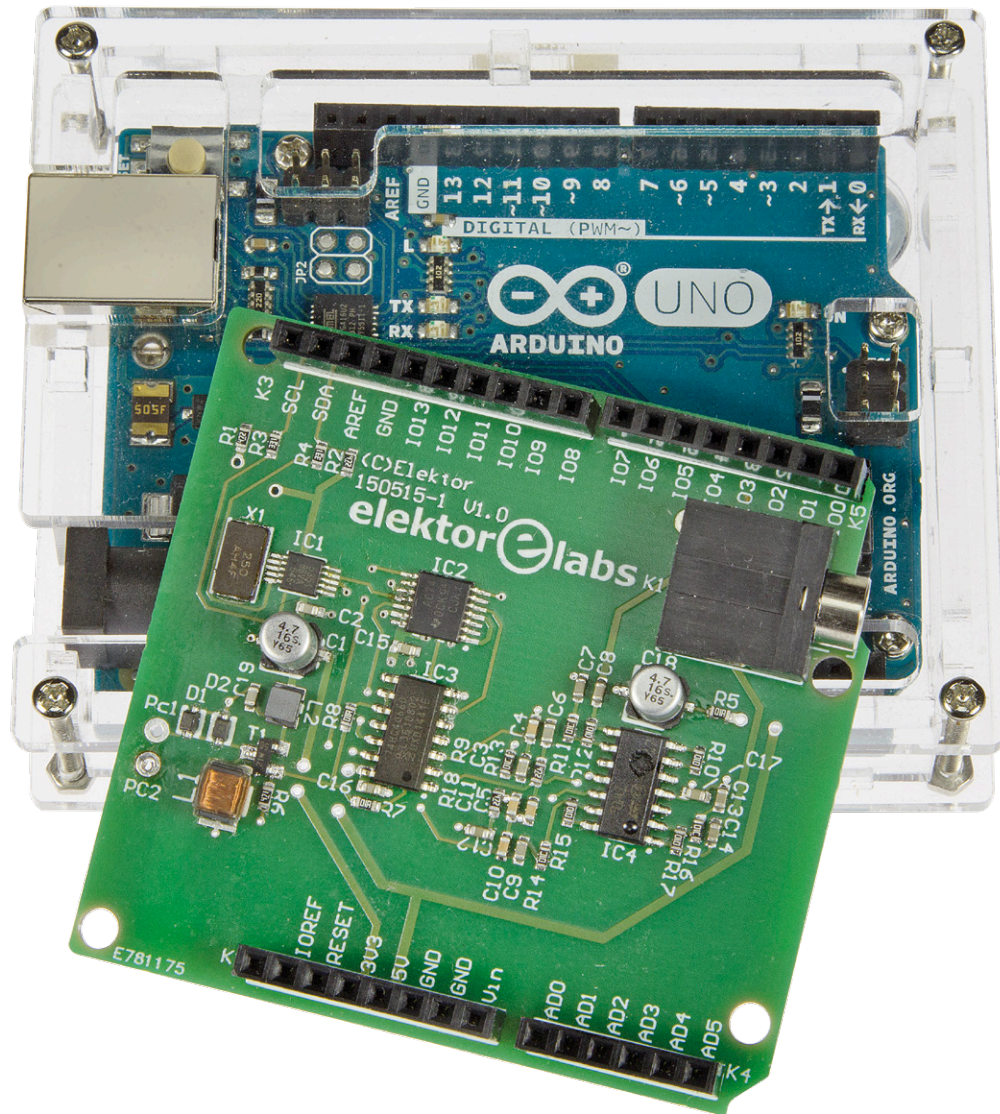
## SDR Shield for the Arduino

By **Burkhard Kainka** (Germany)

A Software Defined Radio is a universal tool in RF technology circles, one that can also be put to use for making measurements. The characteristics of the receiver are defined in software, which now gives us the opportunity to use an Arduino Shield as a front-end.

### Technical Characteristics

- Supply voltage: 5 V and 3.3 V as for Arduino
- Frequency range: 150 kHz up to 30 MHz
- Sensitivity: 1  $\mu$ V
- Total amplification: 40 dB
- Maximum antenna signal level: 10 mV
- Dynamic range: 80 dB



Even though broadcast services are deserting the AM domains in the long, medium and short wavebands, there is still plenty of interest to be found surfing the radio waves with a home-constructed receiver. Now more than ever you might say, because many distant stations now come up far more clearly because they are no longer swamped by stronger signals. In fact it is often so quiet on the short waves that it's easy to imagine your receiver has gone deaf. On some bands it is the radio amateurs who produce the strongest signals. And there is always something new to find, from pirate radio stations through SSB radiotelephony to the new digital modes. That just has to make you curious!

Elektor has already published many radio and receiver projects. A Software Defined Radio with USB interface was introduced as long ago as 2007 [1]. In the meantime much thought has been devoted to conceiving updates for this design. However, the PLL chip we used originally is no longer made, making it necessary to find a new solution. This has arrived in the form of the Silicon Lab SI5351 chip, a CMOS clock generator from 8 kHz to 160 MHz with I<sup>2</sup>C bus.

First investigations revolved around a break-out board from Adafruit. The available sample software was written for the Arduino, so our first steps were undertaken with the Arduino. The new VFO was

simply hooked up to the old SDR PCB and proved its suitability (**Figure 1**).

And then there came an idea: why not simply build the entire receiver as an Arduino Shield? This decided the power supply requirements, using the USB interface already available on the PC. The Arduino would look after controlling the VFO and could be addressed in plain language so to speak (6030 kHz please). And what is perhaps even more exciting, this even gives you a real chance to build a totally standalone receiver. Operation could be migrated from the PC to the Arduino relatively simply. And who knows, perhaps one day the decoding of the IQ signal as well?

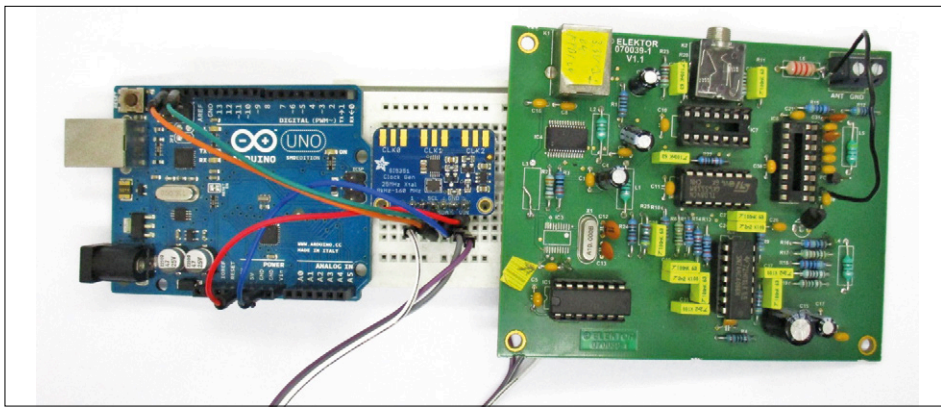


Figure 1. First preliminary test for an SDR2: an SI5351 PLL chip hooked up to an Arduino Uno and the 'old' SDR receiver.

### So how does it work?

First, let's go back to basics. What fundamentally is a Software Defined Radio? For quite some time the development of digital electronics left radios entirely untouched. When home computers first became available, most radios were still analog. Then development began to take place, at least for digitizing their tuning. Today's radios are often equipped with a PLL synthesizer that simplifies tuning and guarantees precise conformity to channel spacing. The rest of the circuitry remains analog as previously.

Subsequently digital electronics appeared inside commercial RF equipment and amateur radio gear. Ever more of the analog functions in devices were replaced by software. In most cases a digital signal processor (DSP) with appropriate software operates out of sight from the user and takes care of optimal filter curves, variable bandwidth, signal decoding, interference suppression and much more. The equipment is altogether improved and with less hardware overhead. Further examples of this kind of development can be found in smartphones and other portable end-devices. At the same time it's evident that hobby constructors can no longer keep pace with this technology. In fact things don't need to be so sophisticated, however. All you need is a rapid A-D converter connected direct to the antenna. The entire spectrum is digitalized and then further processed digitally. In fact technology of this kind is available for the entire frequency range from 0 to 30 MHz. It's software alone that filters out specified frequencies and demodulates the desired signal. Regard-

less of whether we are dealing with AM or DRM broadcast stations or whether we're receiving SSB signals, CW Morse transmissions, teletype (RTTY), weather fax or whatever else, everything is feasible. There is appropriate software for everything. Sure, it must be conceded that with such a large bandwidth, the hardware can be quite expensive, and the further processing necessary for this broad spectrum imposes high demands.

Nevertheless a cunning way around this challenge can be found in the sound cards of modern PCs. Using the 96 kHz sampling rate that's normal today you can already receive the whole frequency range up to 48 kHz. Instead of using a microphone,

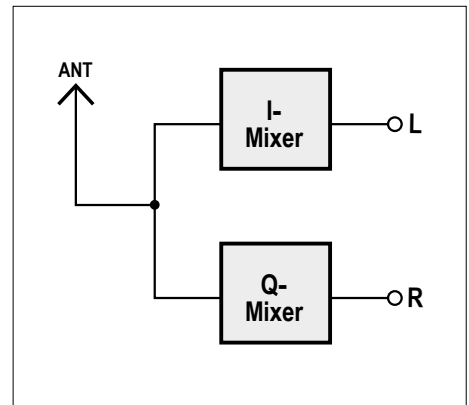


Figure 2. Working principles: the front-end consists of a dual direct mixer with signals shifted 90 degrees out of phase.

you simply connect a large coil as an antenna and immediately you're able to receive the VLF band. Down there are plenty of interesting signals, even transmitters installed on submarines.

If you want to use the sound card for higher frequencies, you must first down-convert the signals. The process is akin to a superhet with a lower intermediate frequency (IF). The PC handles the IF stages, filtering, automatic gain control (AGC) and demodulation. In principle a simple direct mixer with a diode ring mixer or the well-known NE612 would be adequate for this. Only a stable variable oscillator (VFO) would be needed in addition. For special applications you could

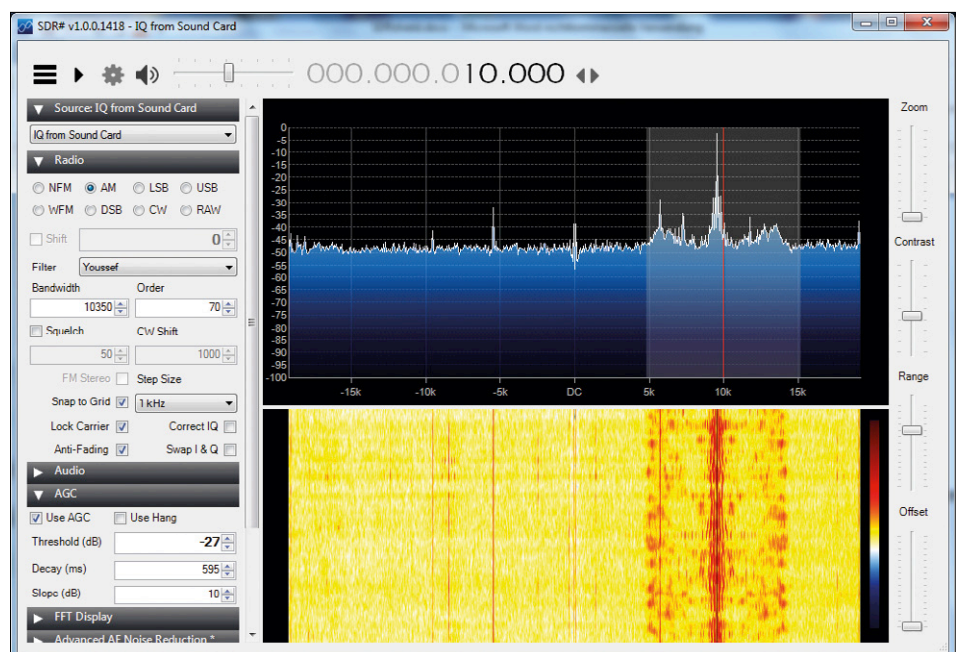


Figure 3. The SDR# program receiving an AM signal.



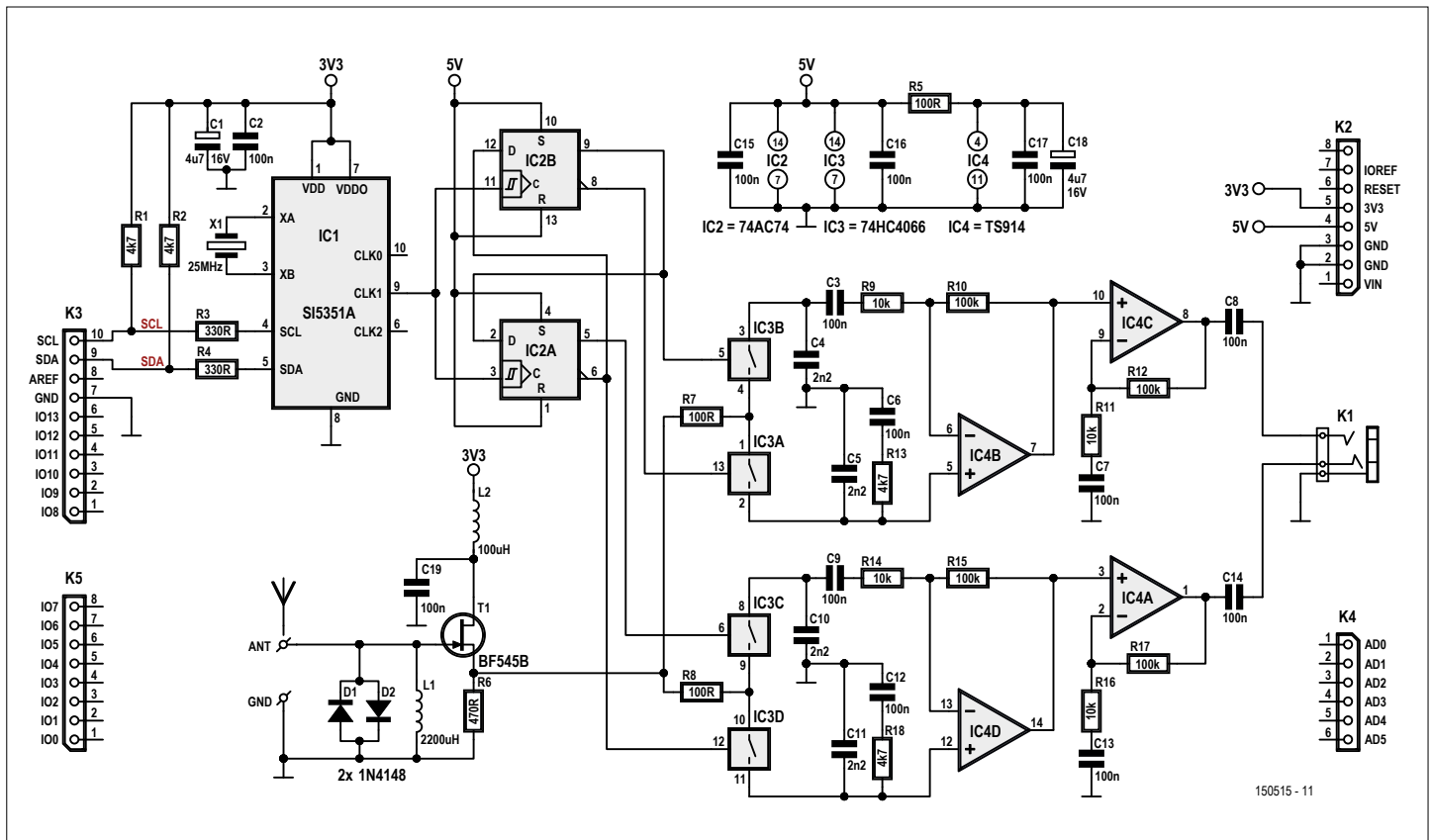


Figure 4. Schematic of the new SDR receiver.

use a crystal oscillator. But if you want to be able to tune an entire band, it ought to be a DDS generator or a PLL module. Simply stated, an IQ mixer is a matter

of a dual direct mixer with two signals phase-shifted by 90 degrees. The oscillator signal is always adjacent to the reception frequency. The output signals

therefore lie within the AF range, mostly between 0 kHz and 24 kHz. The two signals are designated I and Q (see **Figure 2**). These are applied direct to the

## Component List

### Resistors

R1,R2,R13,R18 = 4.7kΩ 1%, 0.1W, SMD 0603  
 R3,R4 = 330Ω 1%, 0.1W, SMD 0603  
 R5,R7,R8 = 100Ω 1%, 0.1W, SMD 0603  
 R6 = 470 Ω 1%, 0.1W, SMD 0603  
 R9,R11,R14,R16 = 10kΩ 1%, 0.1W, SMD 0603  
 R10,R12,R15,R17 = 100kΩ, 0.1W, SMD 0603

### Capacitors

C1,C18 = 4.7μF 16V, SMD case B  
 C2,C3,C6,C7,C8,C9,C12,C13,C14,C15,C16,C17  
 ,C19 = 100nF 50V, X7R, SMD 0603  
 C4,C5,C10,C11 = 2.2nF 50V, X7R, SMD 0603

### Inductors

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

### Semiconductors

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

### Miscellaneous

K1 = stereo jack socket, 3.5mm, PCB mount  
 K2,K3,K4,K5 = connector set, Arduino compatible (1 pc. 6-pin, 2 pcs. 8-pin, 1 pc. 10-pin)  
 X1 = 25MHz quartz crystal (Abracon ABM7)

PCB # 150515-1

Or  
 PCB with preassembled SMD parts: 150515-91

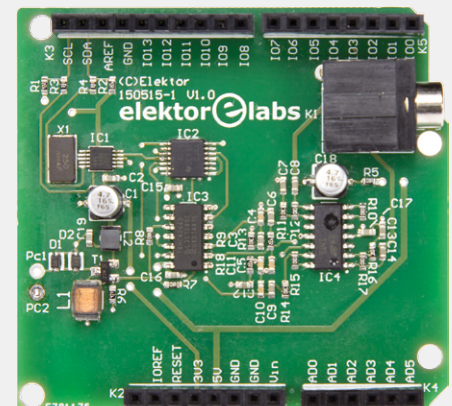
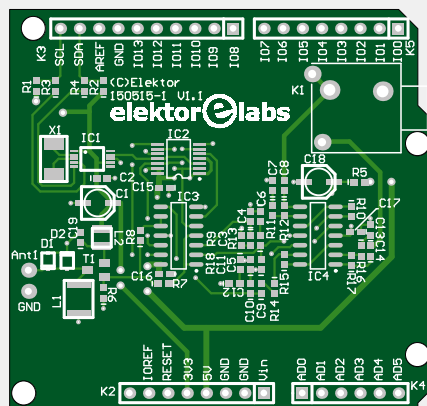


Figure 5. Double-sided PCB for the SDR3 executed as an Arduino Shield.

left and right channels of the sound card input. The rest is dealt with in software. A simple mixer would mix the range below and above the oscillator frequency into the same range, in which the dreaded image frequency problem would arise. By carrying out dual mixing and phase shifting, the software is able to cancel out and eliminate the image frequency, however. In this way a range between  $-24$  kHz and  $+24$  kHz can be received if the sound card has a sampling rate of 48 kHz. **Figure 3** shows what a program like SDR# makes out of this (see also the text panel *SDR software*).

### Circuit

A glance at the schematic in **Figure 4** shows the individual building blocks. The SI5351 PLL generator (IC1) delivers the oscillator signal with the 4x receive frequency to the 74AC74 divider (IC2B). This divides the frequency by four and delivers the signals phase-shifted by 90 degrees to the 74HC4066 mixer (IC3). This analog switch is wired as a changeover switch and applies the RF signal alternately to the inverting and non-inverting inputs of the TS914 op-amp (IC4B/IC4D). In this way the signal is mixed down into the AF region. After some modest filtering and amplification (IC4C/IC4A), the signal reaches the audio output. The RF input stage creates a source follower using the BF545B JFET (T1), the SMD equivalent of the BF245B. Anyone familiar with the old Elektor SDR will see a certain simplification in the signal path. On its RF input it had several switchable lowpass filters. The new design has a wideband input and is protected against over-voltage by two diodes. This is completely adequate for shortwave reception with a wire antenna. The addi-

tional over-voltage protection was born from experience with the first SDR; in a thunderstorm the input stages could be damaged. For specific purposes you can also use additional external filters and a preamp. In the old version the AF amplification could be adjusted in three steps. This time around there is only the middle level, which proved itself to be fine for general use. So everything has become a bit simpler and now works well with the Shield.

For first trials you simply need to connect up a wire antenna. Some wire with a length of three meters hung from the ceiling will be fine. If this is impossible a longer piece of wire lying anywhere in the room should work. Admittedly indoor antennas suffer greater interference levels and advice on how to make optimized antennas will be printed soon in Elektor.

### Construction

The PCB (**Figure 5**) is designed as an Arduino Shield, enabling it to be plugged straight into an Arduino Uno. As the SI5351 is available only as a miniature 10-pin SMD package, we took the decision to design the complete circuit with SMDs and to offer the ready-assembled PCB in the Elektor Shop [3]. Beyond this your only other action is to solder the four Arduino-compatible female headers onto the PCB. Anyone who would prefer to build the PCB completely unassisted can download the PCB layout at [3] or you can buy the PCB on its own from the Elektor Shop.

### Setting the frequency

The Arduino, in conjunction with the SDR Shield, serves as an interface between antenna and PC. Its sole task is to tune

the VFO and to do this, a PC program tells it which frequency is currently desired. Data from the PC reaches the Arduino via the USB cable. The downconverted signal of interest is then sent for further processing along a stereo cable to the sound card input. You could certainly attempt to relocate the control function to the Arduino as well, perhaps even some simple signal processing but this would be hard labor for a small system. For the moment it is sufficient for the Arduino to receive commands from the PC and adjust the VFO.

How you deal with the Arduino is not a topic for this article. Using the Arduino IDE is presupposed. First of all a suitable Arduino program is loaded. Exactly what happens in the software will be explained presently. However, if you're in a rush for practical results, you can skip this information and simply load the software [3]. The critical task lies in persuading the SI5351 to generate an appropriate frequency. The IC has two internal PLLs and three outputs (see block diagram, **Figure 6**). Here only PLL A and the output CLK1 are used. The concept makes use of the Adafruit Library, which makes the whole business delightfully simple. Before you let loose, you must download the Library from [2] and integrate it.

The SI5351 has a 25 MHz crystal oscillator and two PLLs that can be set between 600 MHz and 900 MHz. The PLL dividers operate with fractional division ratios, so it is in fact possible to achieve almost any desired resolution. The following multi-synth divider also uses fractional division ratios. This gives you two ways to generate the desired frequency:

- You can set the PLL on a fixed frequency e.g. 900MHz and then divide

— Advertisement



HAMMOND  
MANUFACTURING

Housings for Raspberry  
Pi, Arduino and many  
other bareboard  
computers

- Enclosure
- Platform

+ 44 1256 812812

sales@hammondmfg.eu /1593HAM.htm



The choice is yours.

- Enclosure for all round protection
- Platform for all round access
- Design-specific versions for all popular models
- Visit hammondmfg.com for full details



/1593HAMEGG.htm

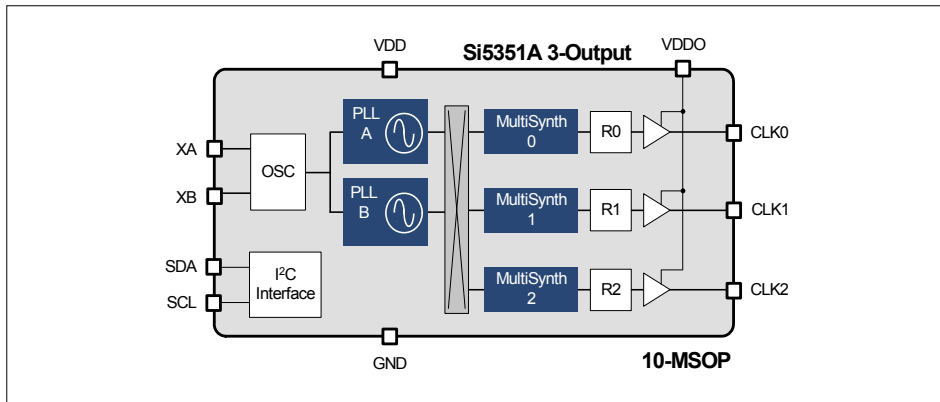


Figure 6. Block diagram from the data sheet for the SI5351A.

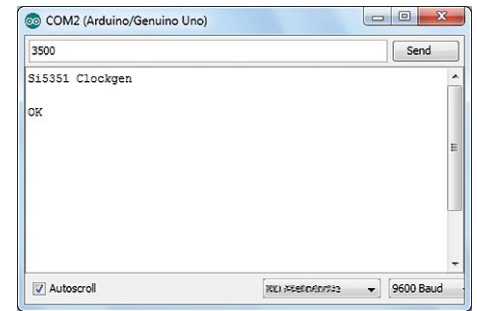


Figure 7. Activating the clock generator with the Arduino terminal.

down with fractional numbers.

- You can adjust the PLL in small steps and then divide by integers to reach the final frequency.

First, here is method A. The VFO frequency is four times the mixer frequency,

which is 12 kHz below the receive frequency. The program is arranged to receive the radio frequency in kHz and implement in text format. In order to receive 3500 kHz the SI5351 must produce at output 1 of the SI5351 must produce an output frequency of  $4 \times (3500-$

$12) = 13952$  kHz. The PLL divider is set to 36 ( $25 \text{ MHz} \times 36 = 900 \text{ MHz}$ ) and the multisynth divider  $900000/13952 = 64.506$ . Using this method we can get down to 1 MHz. For even smaller frequencies the additional R\_DIV divider is set to 16. **Listing 1** indicates the relevant

#### Listing 1. Program for fixed-tuned 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);
  }
}
```

software for the Arduino; **Figure 7** shows the control panel on the Arduino terminal. Method A has the advantage that the VFO can be adjusted more or less continuously, i.e. there is no interruption when a change of frequency occurs.

On the other method B promises greater phase accuracy, adequate even for DRM. Against this, every frequency change is accompanied by a brief interruption of about a millisecond, which appears as an interference signal on the SDR. The method requires calculation of the optimal fractional division (**Listing 2**) in order to keep the PLL constantly in the range 600 MHz to 900 MHz.

Both programs can be controlled by any Terminal program of your choice. However, for really convenient operation a VB program was written in Visual Studio 2015 (SDRshield.zip, downloadable at [3]). This sends the desired frequency to the Arduino in text format (e.g. 3500) at 9600 Baud. The slider control (see **Figure 8**) operates in 9 kHz steps in a range up to 1.6 MHz and beyond that with 5 kHz resolution. Additionally you can enter a desired frequency direct or click the 'band' buttons at the beginning of the individual broadcast or amateur radio bands. The first time you do this, take care that you have selected the correct COM Port.

### SDR software

Here is a survey of the SDR software used. Practically all the programs used with the old Elektor SDR still work fine.

- *SDRadio* is still a good choice;
- *SoDoRa* can also decode DRM;
- *DREAM* still works but does not make use of the IQ signal and uses the receiver like a direct mixer;
- *HSDR* is a current and very powerful program;
- *SDRSharp* (*SDR#*) is distinguished by simple operation and good on-screen representation.

In a follow-up article we shall discuss these individual programs and their possibilities in detail.

### First results with reception

If you have no better antenna to hand, for your first tests you can simply connect a one-to-three meter length of wire to the antenna input. This will enable you to

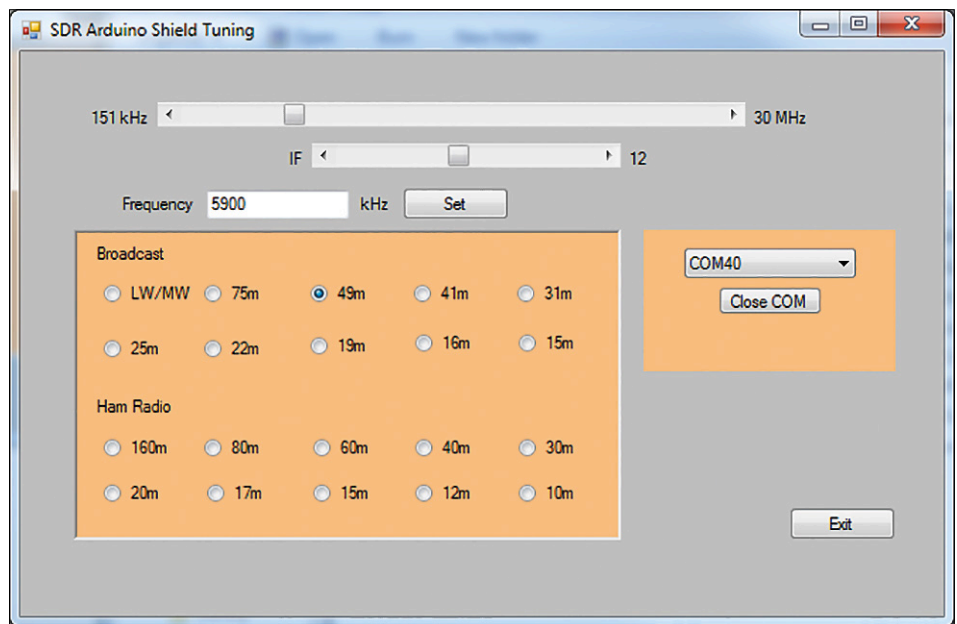


Figure 8. A short VB program ensures ease of operation.

receive all broadcast stations on all short-wave bands without any problem. Experience shows that there is more happening at night than during the daytime. And during the evening we find the main focus is on the lower bands, between 75 m and 41 m. Even amateur radio stations can be pulled in with just a short wire aerial. Normally you'll have the best luck in the 40 meter band, where you can hear some

CW and SSB stations. You can select the appropriate operational mode in the SDR software, plus audio volume, bandwidth, ALC settings and much more. With the correct settings you can often get better results than with an expensive analog receiver of the older kind.

A fundamental characteristic of all switching mixers is that signals on uneven (odd) multiples of the base frequency can also

### SDR software

The current stars in the SDR software firmament are *SDR#* [4] and *HSDR* [5]. Both programs follow the new trend for ever higher frequencies and can be driven using simple DVB-T [6] dongles. This is a good choice if you wish to poke around on the VHF and UHF bands. There have also been attempts to use this kind of hardware below 30 MHz. You can, for example, use an up-mixer that shifts every frequency 50 MHz higher. You do of course then have a multiple conversion superhet along with its well-known problems, such as countless phantom and spurious signals together with reduced dynamic range. A dedicated SDR for frequencies up to 30 MHz uses single conversion only and in that way manages to deliver very clean reception without 'birdy' whistles.

On your PC you have two programs running, namely the tuning program and the SDR software. Each SDR program has its own method of operation but the basic steps are nevertheless similar in each case. First you need to establish that the correct input is in use. For this you need to select the sound card and activate the chosen input (Line In). Next you boot up the SDR software. You'll know you have selected the correct input when you see a significant rise in the noise floor, which should increase still more after connecting the antenna. Most sound cards need to have their volume control throttled back, as the receiver can deliver up to a volt of output signal.

**Listing 2. Program for 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);
  }
}
```



be downconverted. If you want to receive a signal on 1 MHz, other signals on 3 MHz, 5 MHz, 7 MHz and so on can disturb your reception. For this reason people often use switchable lowpass filters. The SDR Shield doesn't include one of these, so it makes sense to use an antenna that's selective. Even so, things work astonishingly well with a wideband wire antenna. The reason for this is that at specific times of the day strong signals dominate on various bands and get through unscathed. An exception to this is reception on long and medium wave, which can be desensitized by signals in the short wave region. You can eliminate this problem by using a ferrite rod antenna with a rotary tuning capacitor. The theme of antennas, filters and pre-amplifiers needs to be examined in closer detail. This involves not merely large signal voltages but also the achievable signal-to-noise ratio. There's nothing better than a long wire antenna, erected as far as possible from your house for this. But because this is not always possible, we must look for compromises. And in this respect the magnetic loop antenna is the clear winner. These enable you to have yourself a relatively small and unobtrusive antenna indoors. More on this later. In your first trials with this receiver one particular question is bound to arise: won't the Arduino itself interfere with reception? It is after all in very close range. In fact great care was taken when laying out the PCB to achieve a high degree of decoupling. This includes a continuous ground plane on the underside of the PCB, whilst the 5 V and 3.3 V supply voltages are decoupled with L-C filters. In actual fact these measures are extremely effective and under normal conditions you won't notice anything from the Arduino.

### Eavesdropping with the Arduino

What you might at least 'receive', however, is the 16 MHz clock oscillator. This will occur when you have no antenna at all connected. The Shield can then demonstrate its ability to function as a test device. Actually there are two oscillators running simultaneously. One of these is the 16 MHz crystal oscillator on the Uno's USB chip with a discrepancy of less than 1 kHz. If you touch the underside of the Uno PCB at the spot where the crystal is soldered, you get a

slight amount of detuning. You know then that it was the signal in question. With a short piece of wire on the antenna input the signal will grow stronger, as will the noise floor. And we can exploit this fact: signals that arrive via the antenna input have good image frequency suppression, whereas it's different for those that creep into the signal path via the supply voltage. The latter exhibit twice the frequency but are significantly weaker.

The clock signal of the Mega328 is another thing to track down. This oscillator uses a ceramic resonator and can exhibit discrepancies of up to 50 kHz. In point of fact a weak signal was found on 15950 kHz with some sideband signals into the bargain, contributed by the controller. Touching the Arduino PCB in the region of the ceramic resonator additionally set off some broad FM modulation and further detuning, which proved that the resonator was temperature-dependent to some degree. Of course it's only when you have an SDR that you can sound out the Arduino so accurately!

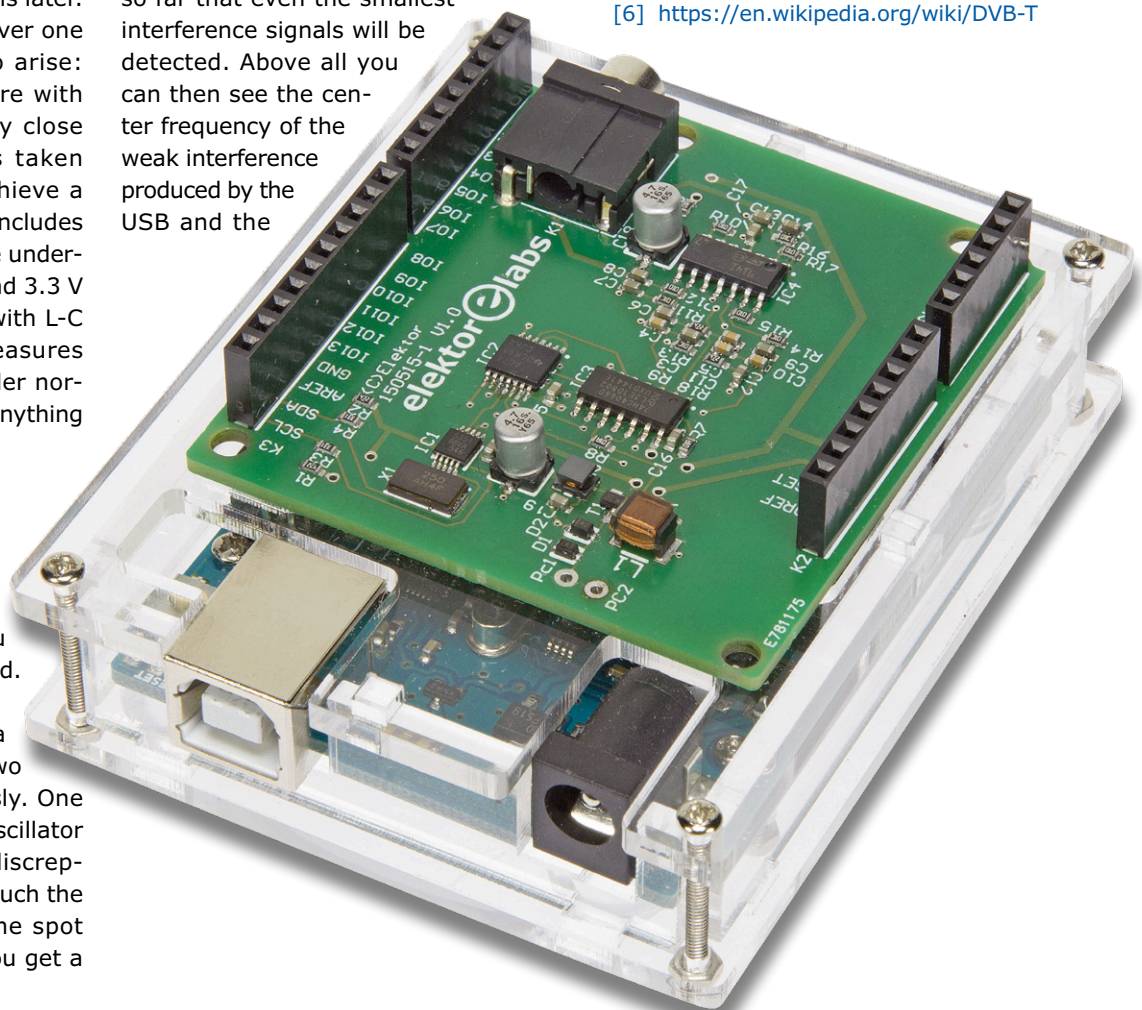
Without an antenna connected a SDR will normally crank up the amplification so far that even the smallest interference signals will be detected. Above all you can then see the center frequency of the weak interference produced by the USB and the

Arduino. If you wanted to separate out the signals caused by the Arduino from those coming from the USB, you could simply power up the Arduino and after tuning into your desired frequency, unplug the USB cable leaving the rest running. All internal interference signals are very weak though. As soon as you connect an antenna, the noise floor rises to the extent that all the interference is entirely masked. This shows the high sensitivity of the SDR. Even signals of only one microvolt can be received. Normally this level of sensitivity is entirely unnecessary, because the noise level of the antenna is significantly higher. Using long antennas can even lead to over-modulating the receiver and in these situations you will have to consider using an input attenuator. ◀

(150515)

### Web Links

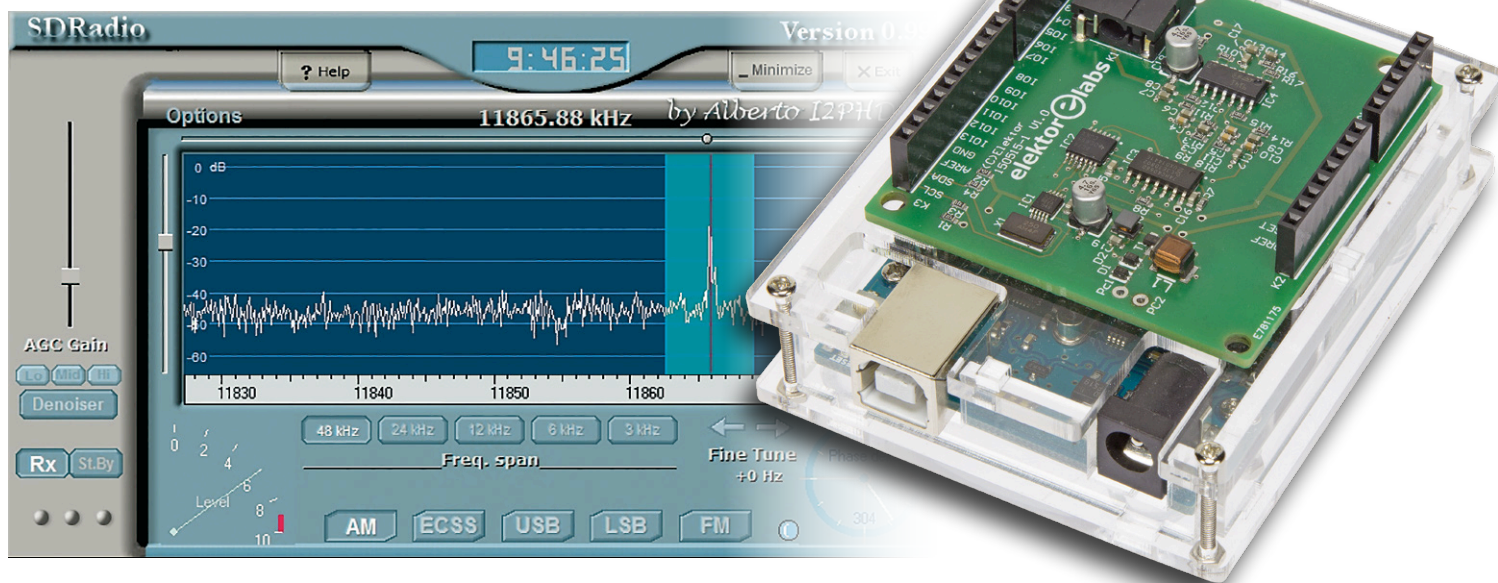
- [1] [www.elektormagazine.com/070039](http://www.elektormagazine.com/070039)
- [2] [https://github.com/adafruit/Adafruit\\_SI5351\\_Library](https://github.com/adafruit/Adafruit_SI5351_Library)
- [3] [www.elektormagazine.com/150515](http://www.elektormagazine.com/150515)
- [4] <http://airspy.com/download>
- [5] <http://www.hdsdr.de>
- [6] <https://en.wikipedia.org/wiki/DVB-T>





# Elektor SDR Reloaded (2)

## Software for the SDR Shield



Without appropriate software the 'Software Defined Radio' Shield simply cannot function. And there are now so many functions and variations to consider that it's not simple to keep track. Here we detail your first moves when putting the SDR receiver into successful operation.

By **Burkhard Kainka** (Germany)

The Arduino side of the software used was already discussed in the previous edition [2], also how the PC is used to tune the receiver. With the receiver now delivering the IQ signal to the sound card, what happens next depends entirely on which SDR software you use. Numerous programs are available, of which just three will be examined here.

### Preparations

In point of fact all you need do is connect a wire antenna to the receiver input. Grounding (earthing) is normally already provided by the ground (earth) wire of the PC. Everything else to do with antennas will follow in a later article. Here and

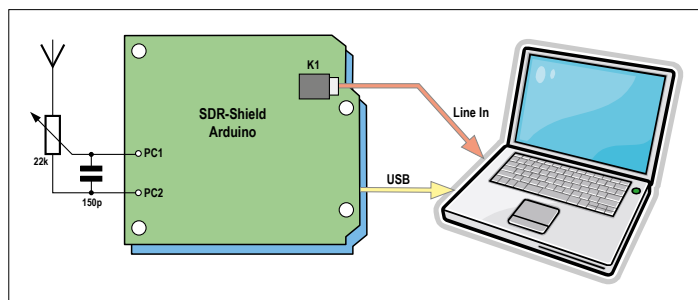


Figure 1. Attenuator and lowpass filter on antenna input.

now we'll describe a really simple method for assuring success with your first attempts at reception. It's an attenuator using a potentiometer (**Figure 1**) and this gadget can work wonders. Even a relatively short antenna can in fact, on many bands and at certain times, deliver signals so powerful that the receiver can be overdriven. So you just throttle back the pot a bit and straightaway the reception is crystal-clear again.

### SDRadio

Everyone who has already experimented with an SDR will have heard of the program SDRadio by Alberto di Bene [1]. Recently the author has written a significantly larger program (HRSDR, see next section), but SDRadio still works extremely well on the newer versions of Windows and stands out for its simplicity. There is no need to install the software as such; it's much simpler to save it in a folder of your choice and just start the program SDRadio.exe. The first thing to do is select the sound card you are using and the audio input (**Figure 2**). If the sound card supports a sampling rate of 96 kHz, you can choose whether to work at 48 kHz or 96 kHz. Clicking on RX starts the decoding.

Along with the SDR software we need to start the tuning program (**Figure 3**), which can be downloaded here [2]. When the VFO in this has been installed successfully on the Shield, the image displayed in SDRadio changes and at the very least you will notice a significant increase in the noise level.

For our first test we select the Long Wave band. The tuning software lets us switch to individual broadcast and amateur

radio bands. A click on LW-MW sets the frequency to 531 kHz. Using the slider control we can tune up or down in 9 kHz channel steps. Clicking on the ends of the slider alters the frequency in small steps of 9 kHz (in the higher bands by 5 kHz), whilst a click on the slider itself gives larger steps of 45 kHz. Because the SDR window displays 48 kHz at a time, this gives us the opportunity to scroll quickly across a band and take an overview to see where the action is.

In this case we are tuning downwards, on order to receive the BBC on Droitwich 198 kHz. The Long Wave has become more interesting now that many local-area Medium Wave transmitters have gone off the air. Another thing is that on Long Wave we have relatively constant propagation conditions during the daytime, meaning that we can expect to find some strong signals. With the antenna potentiometer fully opened up the carrier level reaches -20 dB (**Figure 4**) and the noise floor lies at -55 dB. Reception has relatively severe background noise. If you wind the pot back a bit to an optimum setting, the carrier is inevitably reduced, by 5 dB in fact, but the underlying noise is now -70 dB down (**Figure 5**). In the process the signal-to-noise ratio has been improved from 35 dB to 45 dB, a clearly audible enhancement. Now we can broaden or reduce the receive bandwidth to tease out the best-listening sound. The width of the filtering is indicated by a pale blue stripe. By clicking on the edges you can drag the bandwidth in or out to make it wider or narrower. In fact this ability to vary the bandwidth is a key advantage over traditional receiver concepts using filters of fixed bandwidth.

The BBC signal now occupies 12 kHz. At -3 kHz we see a sig-

nal that belongs to Europe 1 on 183 kHz and lies outside the 9-kHz channel spacing. In SDRadio we can switch directly to this frequency and listen in there without altering the tuning. That's the good thing with software defined radio: you can always see what's going on either side of the receive frequency you have set. And if any interfering signals do crop up, you can often remedy this by shifting the receive window slightly.

## HSDR

Also from Alberto di Bene is the significantly more potent program HSDR, downloadable at [3]. The program comes with an installer and the first time you start it, it opens with a window for setting the sound card (**Figure 6**).

Following this we need to select an input. In the Options Menu (F7) we click on *Select Input* and choose the *Sound Card* (**Figure 7**). The *Playback* window appears next, with additional settings (you can, however, close this without making changes). After this we can initiate reception with *Start*. Any time you re-start HSDR subsequently, the input must be selected afresh. The program starts initially in the SSB operating mode. So let's click on AM and slide the reception window to 12 kHz. The LO (Local Oscillator, VFO) is set initially at 28.5 MHz and needs to be reset to zero, making the display cover from -24 kHz to +24 kHz. To do this you guide the mouse pointer onto individual figures and change these using the mouse wheel.

An interesting feature is the AGC setting (automatic gain control), for which you can select four settings (Fast, Med, Slow and Off). *Slow* is particularly suitable for AM, although the regulation can react negatively to pulse interference. The Noise

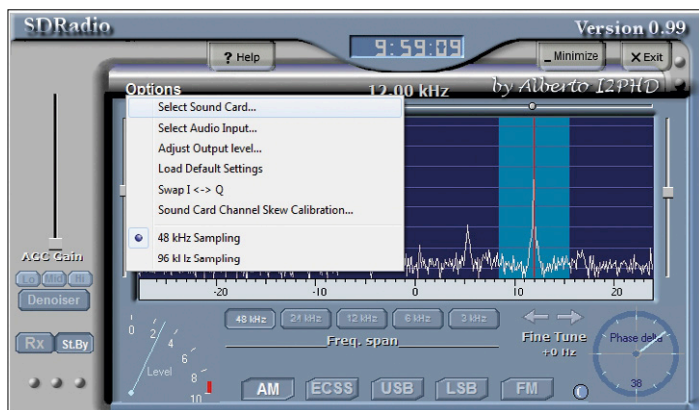


Figure 2. SDRadio options.

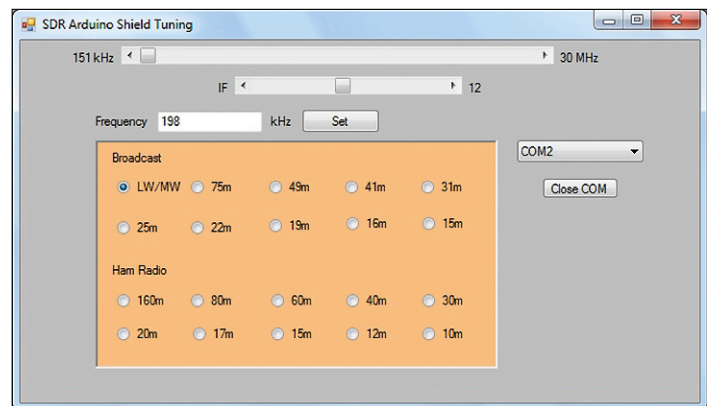


Figure 3. Tuning-in on BBC Droitwich 198 kHz.

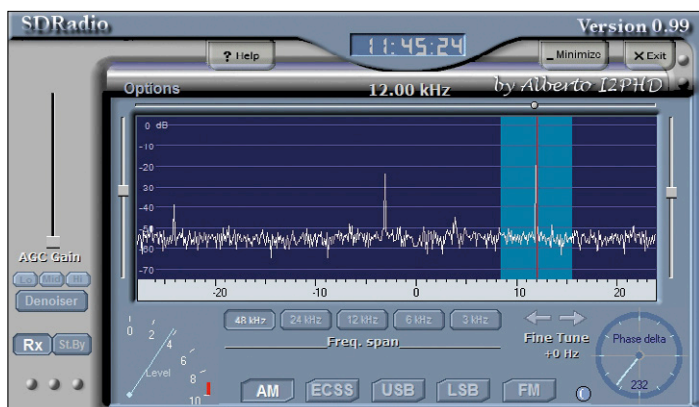


Figure 4. BBC on 198 kHz without attenuator.

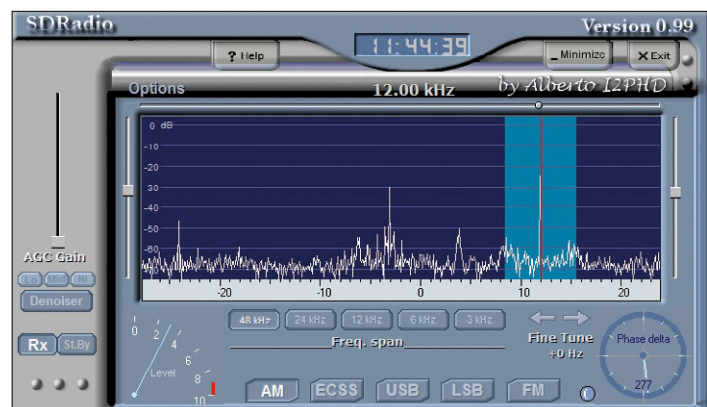


Figure 5. Optimally set attenuator with lowpass filtering.

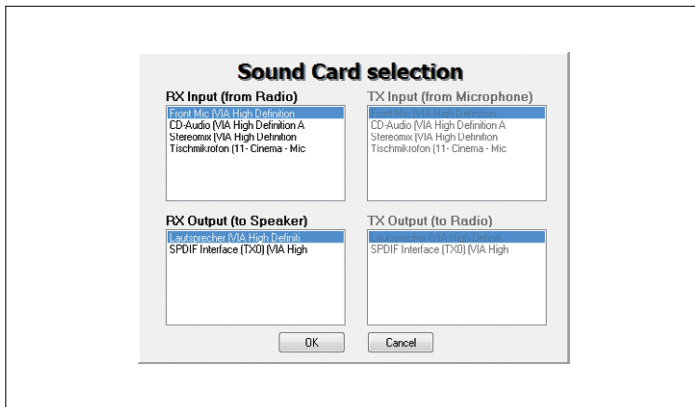


Figure 6. Settings in HSDR.

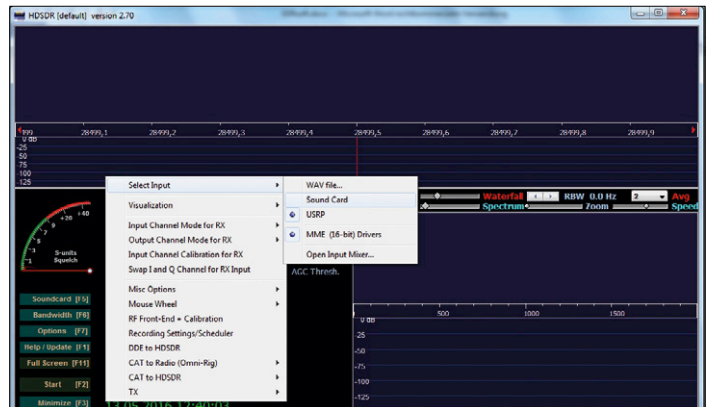


Figure 7. Selecting the sound card as signal input.

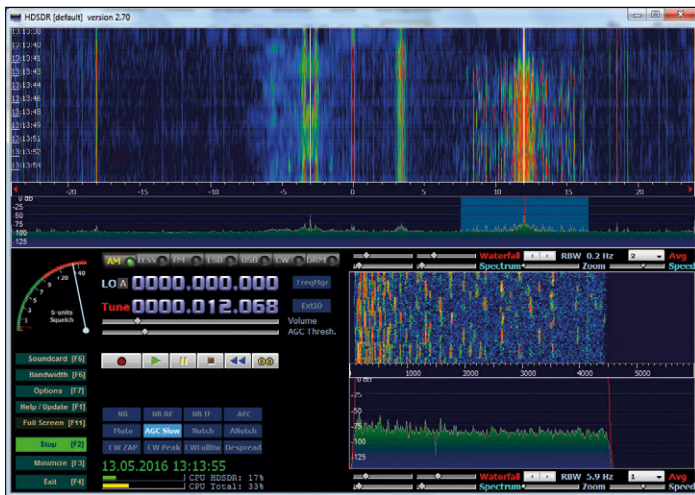
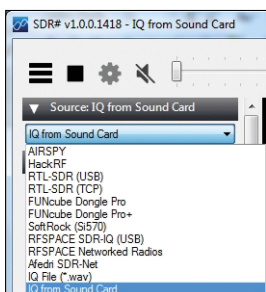


Figure 8. Optimal AM reception.

Blanker (NB RF) can help mitigate this, whilst Noise Reduction (NR) attenuates the noise floor noticeably. When you have found the ideal settings for everything, you can switch out the AGC and thereby achieve a very agreeable, uniform sound. The program has a window of its own for displaying the demodulated audio signal (**Figure 8**). A thin red line indicates the flanks of the filter. You can adjust these boundaries with the left mouse button and thus alter the bandwidth. In the evening time the Short Waves come alive and become fascinating. It's well worth wandering around the broadcast bands and seeking out far-off stations of interest. But you can also listen to amateur radio stations and commercial services, where many transmissions are in SSB or CW.



SDR# stands out by its simplicity...

Figure 9. Selecting the inputs.

## SDR-Sharp (SDR#)

SDR# is another piece of software for universal application. The program supports cheapo USB DVB-T dongles and other UHF receivers, and tends to be used mostly on the higher frequencies. However, it is equally suitable for use with the SDR Shield. The software can be downloaded from [4]. At startup you select the hardware to be used along with the sound card as input (**Figure 9**). These settings are retained the next time you use the software.

Decoding begins by using the Start/Stop symbol seen at upper left in **Figure 10**. You have in addition a mute function and the loudspeaker volume control. Using the Menu symbol you can switch the menu on and off, so as to rationalize everything down to the most mission-critical (**Figure 11**). The receive frequency in the 48-kHz range and the bandwidth can also be set then. To later the bandwidth you click using the left mouse button on the edge of the filter and reposition it according to your needs. The maximum AM filter width is 32 kHz, which in fact enables you to hear several adjacent transmitters. This setting makes sense if you want to check out individual Short Wave bands rapidly. During the daytime you may find only gaping emptiness here. But if you keep pressing and maintain the wide AM setting, you can scan across the broadcast bands in rapid 5-kHz steps and find the few signals of interest in a short time. After sunset everything looks entirely different again. It's time now to keep focused on the 41-m band, but also do watch out elsewhere for signals like All India Radio on 7550 kHz. Among the Menu functions you should examine the AGC settings. All the important functions here can be tried out individually. Normally the Noise Blanker should be deactivated, as it can degrade the sound quality of an AM station. Nevertheless, it can help effectively when you are fighting intensive interference. The same goes for the Noise Reduction options. What would Short Wave be like without amateur radio traffic? On the 40 m band above all others countless SSB speech and CW Morse signals can be found. Normally you would look for an interesting segment using the tuning software and then move the precise receive frequency into the SDR window. In this way you always maintain sight of the current band occupancy. In the CW operating mode (**Figure 12**) you can nominate a very narrow bandwidth in order to suppress close-by signals and fish extremely weak signals out of the noise. A specialized hobby of many radio amateurs is radio contact



using very low transmit power (QRP), often sent at very low speed (QRS) in order that they can still be read among the noise. In the waterfall diagram you can frequently see weak signals that you can scarcely even hear.

SSB speech traffic (**Figure 13**) operates without any carrier, transmitting only one out of the two sidebands. On the lower bands as far as 40 m people use the lower sideband (LSB) and on the upper bands the upper one (USB) is used. To receive these knowing the correct frequency is decisive, since imprecise tuning otherwise leads to a kind of Mickey Mouse voice effect. However, it is very simple to spot the exact position in the spectrum because normally the deeper components of speech predominate. Using the mouse you can adjust the receive window at lightning speed in order to clarify the speech. Once again this is another advantage of being able to see the current band occupancy. In this way you can rapidly flip over to the adjacent station and listen in for a moment. The waterfall also displays clearly those stations that are almost already slipping down in the noise.

Reception software is an extensive field and this article can focus on only the most important programs along with the settings for the most important applications. So far we have covered the crucial first steps but plenty more lies ahead. You need think only about the many digital operating modes like weatherfax, radio teletype (RTTY) and DRM broadcasting. With the Elektor SDR Shield all roads are open.

(160048)

## Web Links

- [1] <http://digilander.libero.it/i2phd/sdradio/>
- [2] [www.elektormagazine.com/150515](http://www.elektormagazine.com/150515)
- [3] [www.hdsdr.de](http://www.hdsdr.de)
- [4] <http://airspy.com/download/>

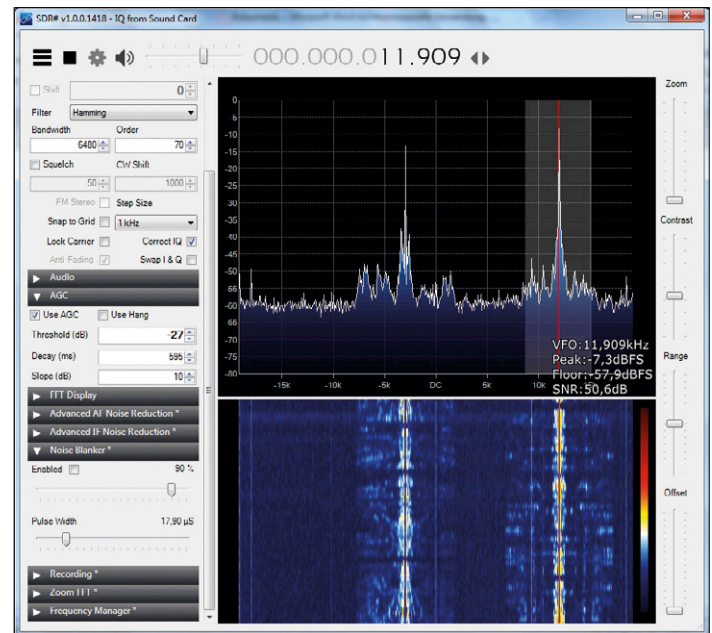


Figure 10. AM reception and SNR display.

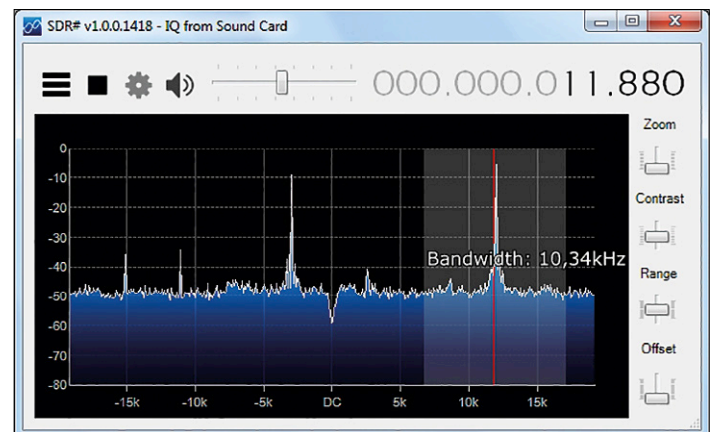


Figure 11. Compact SDR# window.

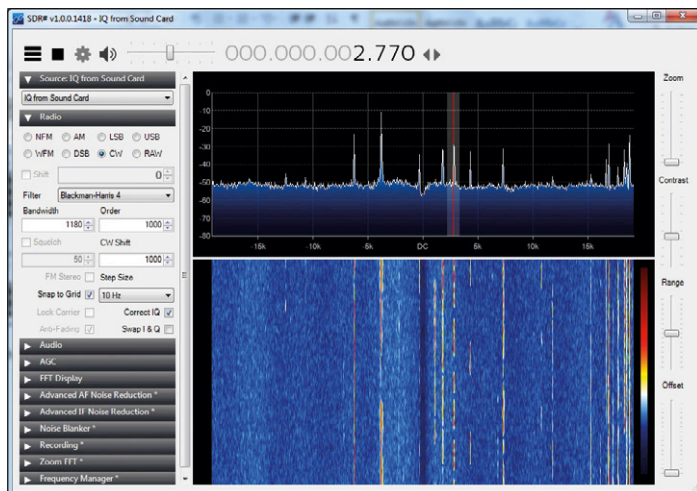


Figure 12. Narrow bandwidth and CW signals.

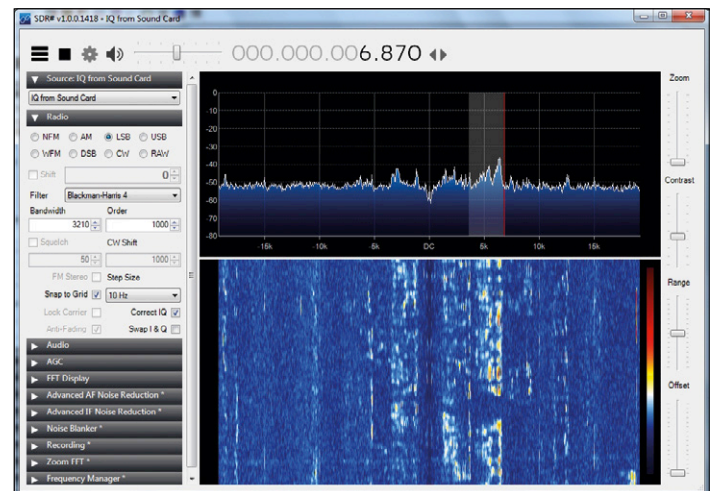


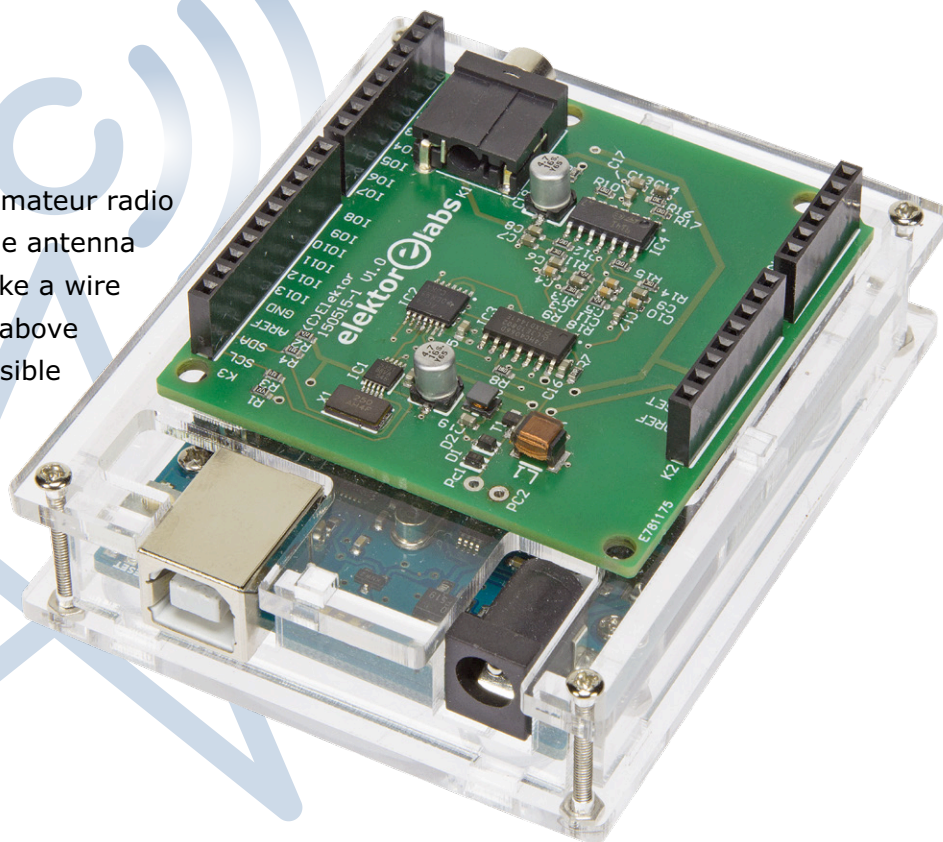
Figure 13. SSB reception on the 40 m band.

# Elektor SDR Reloaded (3)

## Antennas for the SDR Shield

By **Burkhard Kainka** (Germany)

Radio reception on AM, shortwave and amateur radio bands stands or falls by the quality of the antenna used. Good results are easy: you just take a wire 30 meters long and string it ten meters above ground between two trees, as far as possible away from your house. In reality this is not always achievable. So we need to find the ideal compromise: an antenna that is effective, unobtrusive — and small.



As experience shows conclusively, this requires pulling in not just maximum signal voltage but (even more so) attaining the best signal-to-noise ratio. Interference is pervasive and its amount grows daily. Living quarters in particular abound with sources of interference. Computers, power supplies, consumer electronics, motors and much more besides generate a dense layer of electronic noise that overwhelms much of what goes on in the amplitude modulation (AM radio) region.

### Aerial attenuator

On the SDR Shield we already have a FET impedance converter (T1) with an RF choke (L1) and diode limiter (D1/D2). The simplest, yet adequate, antenna for the SDR is a piece of wire about three meters long, which is easy to sling over a cupboard or even on the floor of the room. Using just this you can already hear a dense array of radio stations on the short waves in the evening. Some signals on the 75-meter band may even be strong enough to overload the receiver. A helpful remedy is an attenuator in the form of a 'pot' (potentiometer) that is simply connected in parallel with the antenna input (**Figure 1**).

As soon as you connect the antenna wire, the noise level increases discernibly. The noise floor level varies from band to band and can swamp weak signals with ease. The antenna itself is not only the source; chiefly it arises from the 'polluted' ground or earth wire. Normally your PC is connected to earth via a protective ground wire. But between the actual earth and the power outlet (mains socket) on the wall, it picks up a whole load of 'crap' along the way. Then we have the USB cable carrying all manner of wideband signals, meaning that the noise level at the Arduino's ground connection is even greater. Even a perfectly ideal outdoor antenna cannot solve this problem, because the GND interference and the antenna signal are stacked up at the input. The best a good antenna can do is deliver signals that are stronger than those producing the background noise.

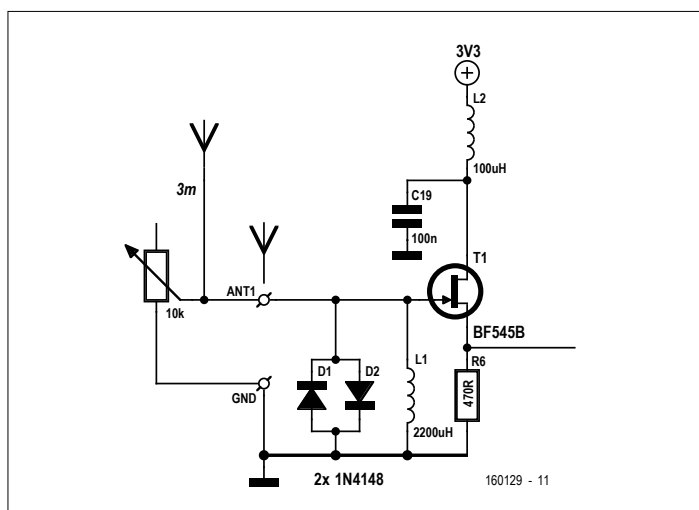


Figure 1. Indoor antenna with attenuator.

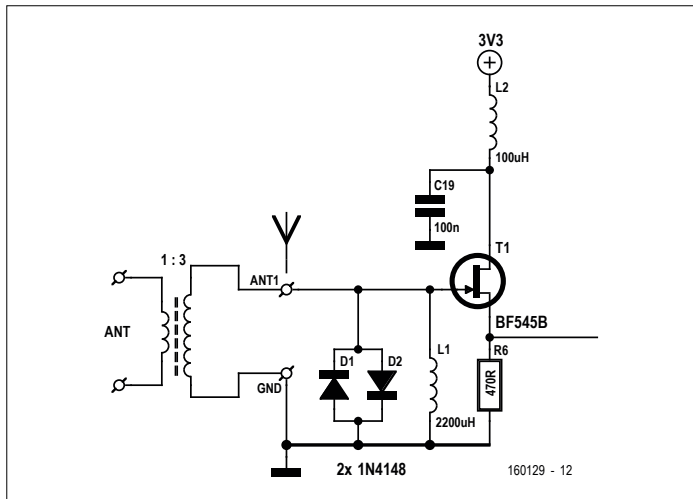


Figure 2. Using an RF transducer.

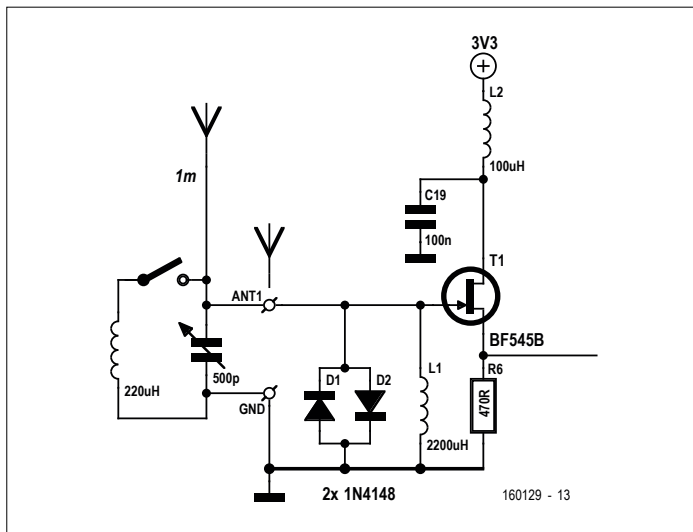


Figure 3: Tuned input circuit.

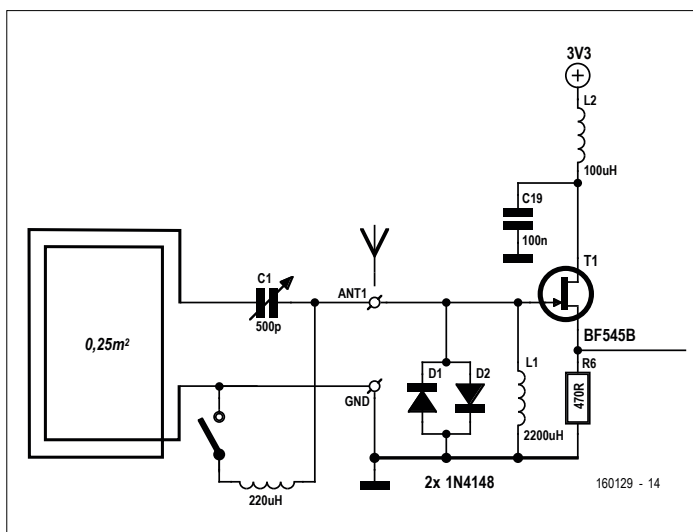


Figure 4: Frame antenna for the long and medium waves.

### Isolating transformer

Let's assume you have a perfect outdoor antenna fed with coaxial cable or an aerial setup with an AM antenna on the roof. You always still need to ensure that interference is not being introduced by the ground connection. The best way to do this is with an RF isolating transformer (Figure 2). You can make one of these using a ring core or a double-hole core. Guide values for winding these are two turns on the antenna side and six turns on the SDR side.

### Resonant circuits

Signals on the long and medium waves are improved considerably when you employ a resonant circuit (Figure 3). In the state of resonance a tuned circuit will simultaneously raise the signal voltage and degrade undesired signals occurring at multiples of the target frequency. With a variable capacitor of 500 pF the internal choke coil (2.2 mH) can tune down to the lower end of the long-wave broadcasting band at 150 kHz. A short antenna just one meter long will suffice now. Admittedly the circuit is relatively narrowband, meaning that you have to retune it each time you change frequency.

For the medium waveband from 500 kHz you simply hook up a coil or fixed inductance of 220 µH in parallel. A medium wave coil can also be wound on a ferrite rod, removing the need for a wire antenna. In the process it becomes plain that the signal-to-noise ratio is improved noticeably. This is because the ferrite rod picks up predominantly the *magnetic* component of radio waves, whereas near-field interference signals occur mainly as alternating *electrical* fields.

In reception tests it is always valuable to make comparisons using other radios. It might turn out for example then that a fairly basic transistor radio (battery operated) might receive weak medium wave signals more clearly than the SDR. That's predictable really: this simple radio uses a ferrite antenna, so it cannot pick up interference related to a ground connection. So we need to do a bit more to make the SDR equally interference-resistant.

A different subject for making comparative tests is a mini stereo system with AM coverage. These tend to use a small frame antenna with multiple turns of wire, connected by a fairly long cable. The instructions tell you to set up this antenna at a suitable spot away from the hi-fi system. The digital modules in the system produce so much interference that using an internal ferrite antenna simply would not work. Tests indicate that adequate distance makes all the difference. You need at least half a meter or so of separation to achieve good reception. On VHF these problems are far less pronounced of course, because the majority of interference occurs on the lower frequencies.

### Frame antenna

A good frame antenna could be the solution. And a coil with a physically greater capture area will produce a higher voltage in the antenna. The antenna loop used should have two turns and a frame length of 0.5 m (Figure 4). For this we need four meters of wire or better still, eight meters, so that the antenna can be placed two meters away from the computer. Used like this, you have a very usable wideband short wave antenna. But on the long and medium waves the inductance is too low.



You can cure this with a variable capacitor connected in series. The loop antenna now becomes a component of the input circuit and can be brought into resonance on long wave. Using a smaller inductance of 220  $\mu\text{H}$  makes it right for medium wave.

### Rod antenna

The familiar car aerial is proof that rod antennas work well. Finding yourself in a parking area away from general habitation one evening and tuning the medium waves on the car radio, you may be amazed how good the reception can be out there. Using this kind of antenna, finding the right location is your only challenge. The best spot would be on the roof or on an aerial mast as far as possible away from the house. A compromise might be an antenna placed at least outside the window or, if necessary, fixed on the inside of the pane.

You will certainly need a fair length of screened cable. And this will work then only with a matching preamplifier or impedance converter (**Figure 5**). In the simplest case a JFET will suffice both as impedance converter and preamplifier, which can be powered remotely over the cable. The Arduino produces the appropriate voltage of 3.3 V, which is filtered in addition. To make the antenna usable in all situations we will need to add a pot as attenuator and a simple low-pass filter.

An ideal installation of this antenna will deliver a fully adequate signal voltage on all bands. Admittedly one problem remains to be solved: wideband interference will be picked up, via the ground conductor, from the computer and other gadgets connected to the AC supply.

### Loop antenna

All these tests have proved that (1) magnetic antennas give the best results indoors, (2) they need to be placed as far as possible away from the computer and (3) the best antenna location in the home is next to an external wall or a window. All of these insights point in the direction of a magnetic loop with a preamplifier, powered remotely over a coaxial cable (**Figure 6**).

The actual loop consists of four meters of coax cable, fixed to a window in a square that measures one meter along each edge. The inner conductor of the cable serves as the loop. The braiding of the cable is connected to ground and shields the antenna from electrical fields. At the upper end we need to open and break the screening so that it does not become a closed circuit in itself.

The preamplifier employs a BC547B audio transistor, the amplification of which extends up to higher frequencies. The low input impedance of the transistor represents practically a short circuit to the antenna feeder, which makes the antenna extremely wideband. The preamplifier is connected and powered using a coaxial cable of any length you choose. Extracting the signal, with the attenuator and optional lowpass filter, remains unaltered.

In actual fact there is plenty in this simple project that might be worthy of improvement. The impedance is not an exact match to the cable and improvements could still be made in matters like linearity and overload rejection. Both aspects call for greater collector current. But the amplifier stands out by its simplicity and works very satisfactorily.

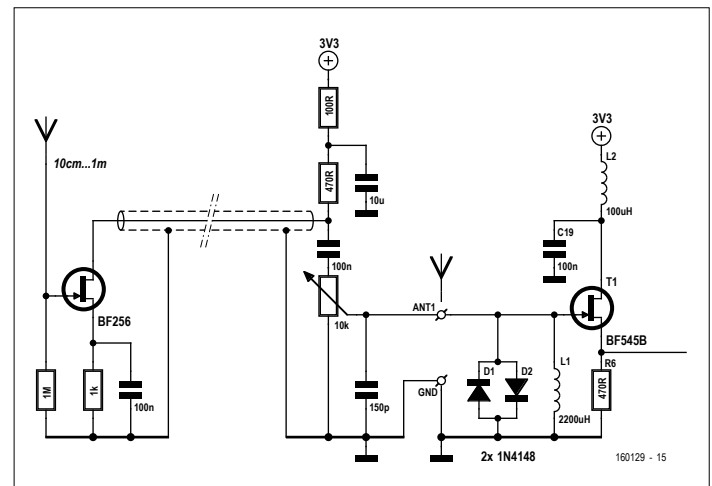


Figure 5: Rod antenna with preamplifier.

▶ A Magnetic Loop antenna gives very good results.

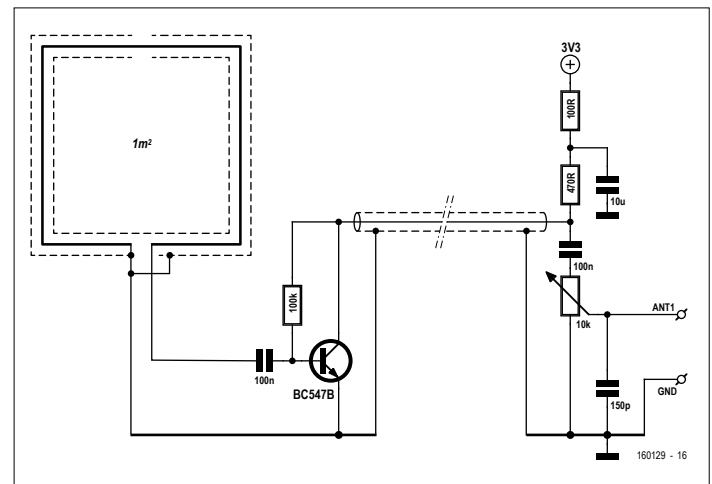


Figure 6: Wideband Loop with preamplifier.

This antenna provides very good results on all frequencies from 150 kHz to 30 MHz. Excellent signal-to-noise ratio is achieved on long wave even during the daytime. Evaluation tests with other antennas have demonstrated best results with the loop almost every time. For comparison an out-of-use wire antenna that the author had set up outdoors was used. At an elevation of about 30 feet this antenna certainly produced great signal voltages, simultaneously, however, with a broad spectrum of interference. For that reason reception using the screened loop gave the better results in most cases. ◀

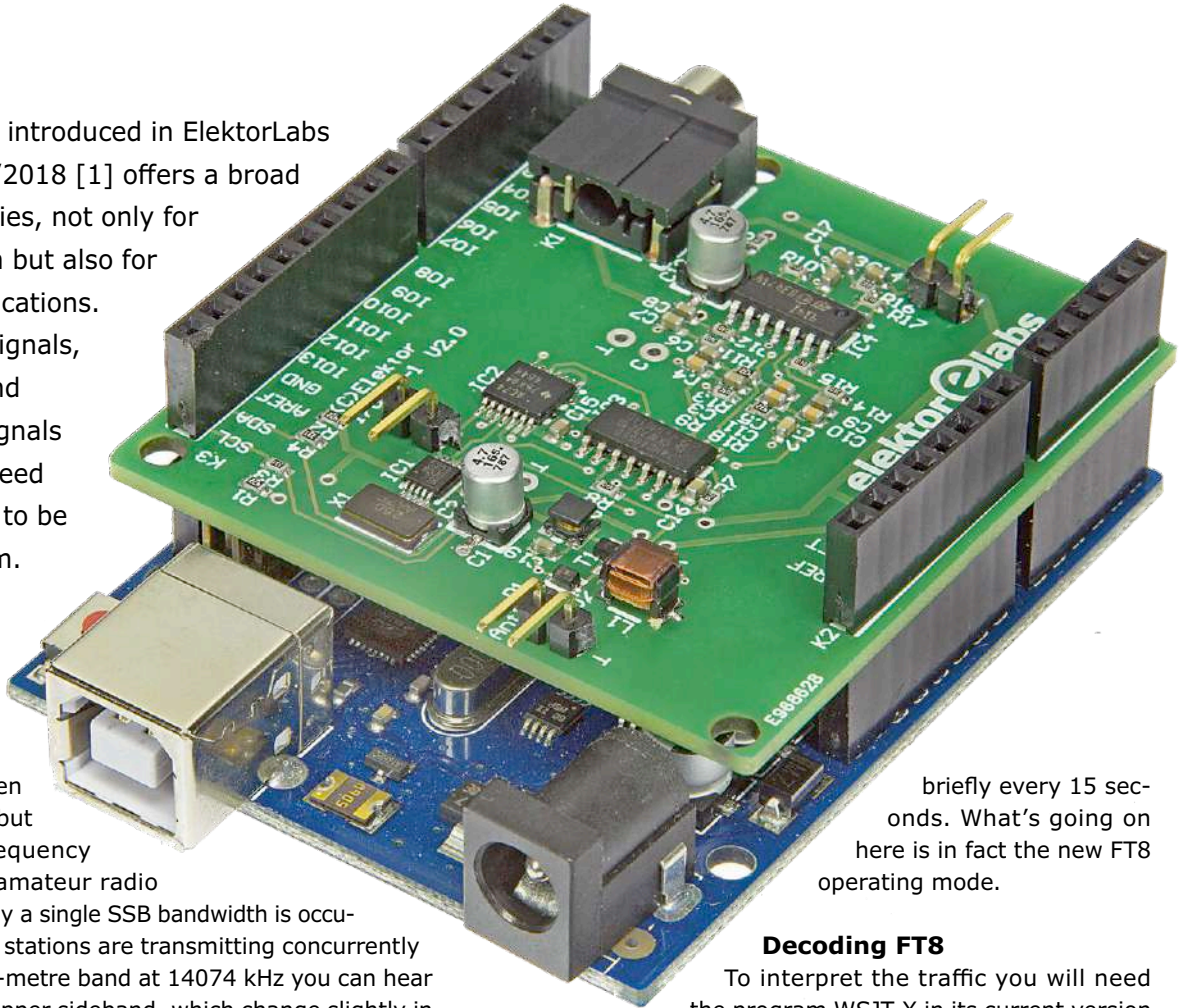
(160129)

# Elektor SDR Shield 2.0 (2)

## Digital data transmissions on the short waves

Burkhard Kainka

The SDR Shield 2.0 introduced in ElektorLabs Magazine edition 4/2018 [1] offers a broad range of opportunities, not only for broadcast reception but also for amateur radio applications. You can hear SSB signals, also listen to CW and numerous digital signals for which you will need additional software to be able to decode them.



Lately people have been discovering a narrow but particularly active frequency spot on many of the amateur radio bands. In each case only a single SSB bandwidth is occupied, on which several stations are transmitting concurrently (**Figure 1**). On the 20-metre band at 14074 kHz you can hear strange tones on the upper sideband, which change slightly in rapid succession. You'll also notice how the signals are silenced

briefly every 15 seconds. What's going on here is in fact the new FT8 operating mode.

### Decoding FT8

To interpret the traffic you will need the program WSJT-X in its current version (1.8.0). You can download this for Windows,

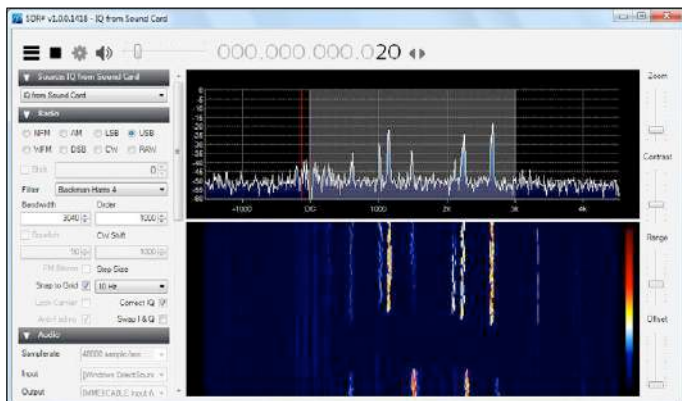


Figure 1. FT8 signals on the 20-m band.

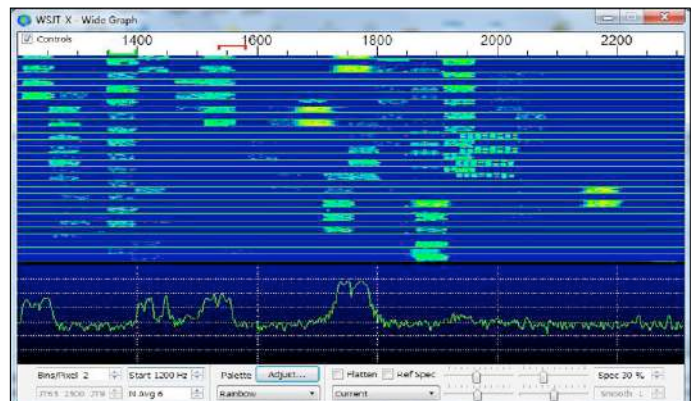


Figure 2. FT8 signals in their timeframe.



Linux and Mac machines at [2]. Installation is plain sailing. The crucial settings can be found in the (not very easy to find) menu by pressing the F2 key. WSJT-X works in a number of modes and for our purpose you need to select FT8. The software produces its own spectrum display (**Figure 2**). FT8 signals can be recognised by their width of around 50 Hz, in which eight frequencies make up an FSK signal. Additionally, each station occupies two transmission blocks, a scant 15-s long, per minute. Its partner station replies during the intervening time. A typical radio connection lasts for only one minute more or less. If the level lies in the middle range, messages should be decoded. It's essential, however, that the system time is set absolutely exactly. If nothing is decoded, this is usually a problem with the PC clock, which still needs to be synchronised. As soon as data is displayed, the current time deviation DT is also displayed in fractions of a second (**Figure 3**).

If everything is installed correctly, you can now sit back and, without needing to make further frequency searches, quietly observe who is contacting whom. Of particular interest is your own station's signal-to-noise ratio in dB compared with those reported by the stations it's in contact with. In this way you get an impression of far removed your own receive set-up is from the optimum. This depends largely on your receive location, which may suffer a high level of QRM (interference), whilst other stations may be enjoying better conditions and fully optimised antennas.

You can check out the system even without a transmitter or amateur radio licence, so to speak, by clicking on a station that is calling CQ. The software sets up the five text messages, Tx1 to Tx5. If you have arranged everything correctly and press Enable-TX, the transmit signal will be heard direct through the loudspeaker. Using the PWR control you can select the volume, just as you would the transmit power of an SSB transmitter. With good hearing you can also detect the eight adjacent frequencies.

### Decoding WSPR

The WSPR transmission technique (Weak Signal Propagation Reporter, pronounced 'whisper') was developed to span long distances with low power and minimal bandwidth. There are always many stations active, who report their reception results to [3]. Very rapidly you can gain an overview of where a signal can be received and how 1000 km (600 miles) is genuinely possible with only 10 mW. With 200 mW you can reach anywhere in Europe without difficulty.

Signals can also be read using WSJT-X. With a bandwidth of only about 5 Hz, they are extremely narrow and in total occupy only 200 Hz (from 1400 Hz to 1600 Hz). The spectrum in **Figure 4** indicates that the calibration of the receiver is correct, which is quite a challenge.

Stations received are displayed with the signal-to-noise ratio achieved, their call sign, location, transmit power and the distance in km. For this purpose, your own location in the configuration must be specified as a QTH locator, which you can determine, for example, at [4]. The locator code for my receiver in Essen was JO31MK.

Among the stations received (**Figure 5**) were several with a transmit power of only 200 mW or less, with which many managed distances of 100 km. The check mark (tick) for 'Upload spots' should be set only if you are logged into Wsprnet. Then your own reception results will be stored in the database. Oth-

erwise you may prefer to listen in privately. But it is always interesting to refer to the Wsprnet website to see which other stations were heard.

The example in **Figure 6** shows where the station DJ5AM was received. A comparison of the displayed reception frequency

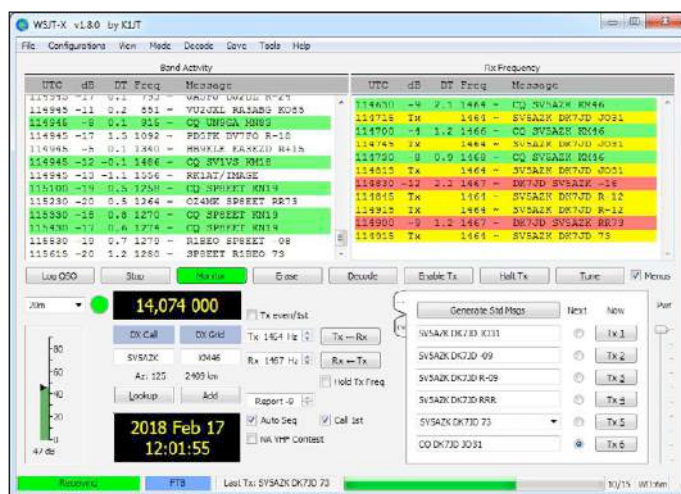


Figure 3. FT8 radio traffic across Europe.

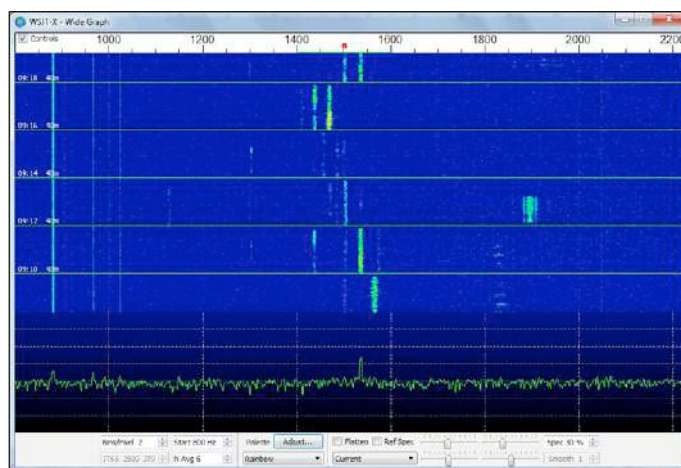


Figure 4. Waterfall diagram of WSPR signals.

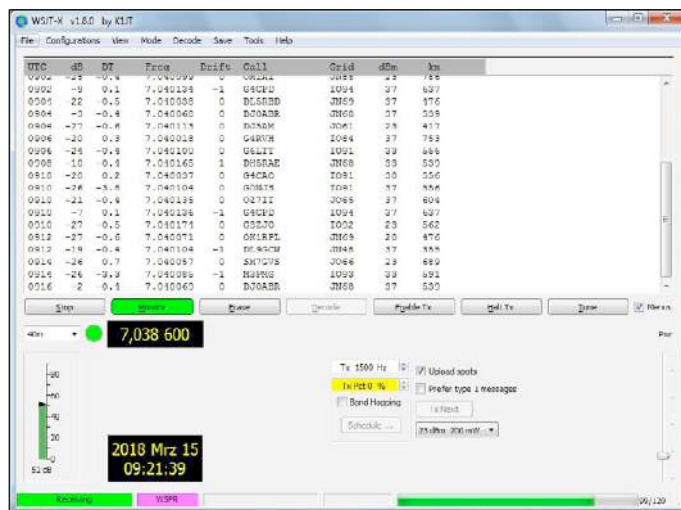


Figure 5. WSPR reception stations.



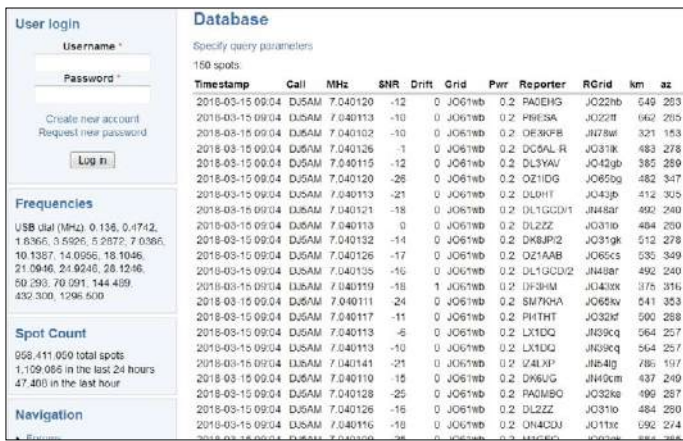


Figure 6. Reception reports from multiple receiving stations.

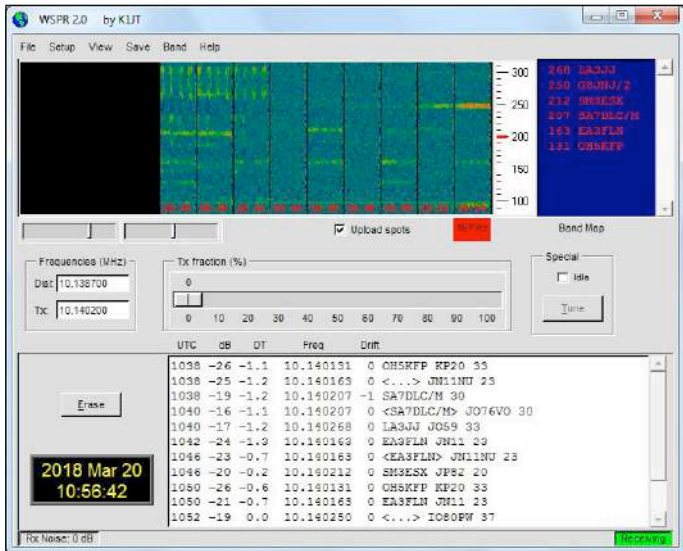


Figure 7. WSPR signals direct and via SDR#.

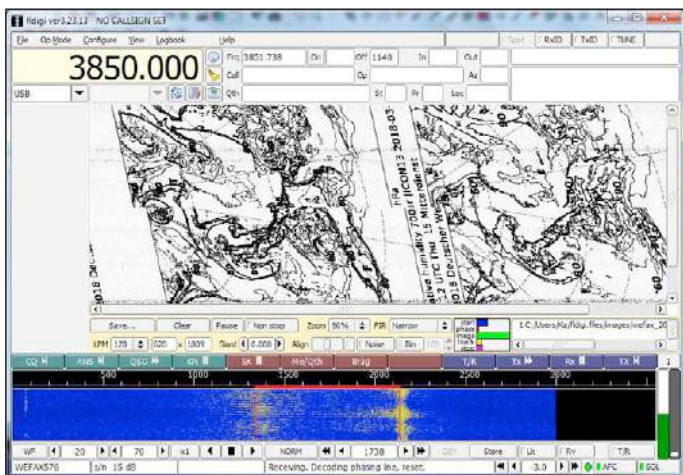


Figure 8. Weather chart with minor errors.

proves that my own display of 7040113 Hz is well in range. The calibration is therefore in line with achievable accuracy. In my first reception attempts, despite my initial calibration, a deviation of 100 Hz was discovered, which was then easy to correct. A comparison of the measured signal-to-noise ratios is also interesting. On average, other receive stations at comparable distances had signal-to-noise figures that were better by 10 dB to 20 dB, which indicates better antenna positions with less background noise. In my case, a shortened dipole plus balun were used in the garden. Despite this there is still plenty of wideband noise on the antenna input.

### Virtual audio cable

Many programs that have been developed specially for amateur radio require the use of an SSB receiver. Usually these provide reception from 300 Hz to 2700 Hz. You switch the receiver to USB and feed the receiver's audio signal into the sound card. The SDR Shield does have a connection for the sound card but as such it is not a fully-fledged SSB receiver. Nevertheless, you can use it for this purpose. There are two methods for this: a direct connection or else using SDR software such as SDR# or G8JCFSDR.

The direct connection is the simplest. You set the VFO so that the wanted signal lies within the range of the SSB bandwidth. The Shield then produces the desired SSB signal direct, albeit with the addition of the other sideband. The decoding software does not normally extract an IQ signal, only a mono signal. In many situations this works very well nevertheless, because there is no signal present in the other sideband.

Sometimes the better method is to use an SDR program with an IQ input and pass the already processed audio signal to the decoder. In this way you achieve effective suppression of the unwanted sideband. But now a new problem appears. You really need an extra sound card or a second PC, since the first sound card is already tied up with the SDR Shield. A solution lies in a virtual audio cable, such as the VB CABLE virtual audio device [5]. In this way you get an extra output device and in the SDR program you enter the *Cable Input* instead of the usual sound card outputs. Correspondingly, in the decoding software you select the *Cable Output* instead of the sound card input. Theoretically halving the noise bandwidth should produce a signal improvement of 3 dB. A comparison with WSPR signals confirms this differential exactly. On occasions the improvement was even clearer, because other signals present in the wrong sideband were now suppressed. In other cases the signal was definitely degraded, as the programs involved did not work optimally with the virtual audio cable.

Teaming up SDR# with WSPR2.0 produced good results. **Fig-**

### Web Links

- [1] [www.elektormagazine.com/160577](http://www.elektormagazine.com/160577)
- [2] <https://physics.princeton.edu/pulsar/k1jt/wsjt.html>
- [3] <http://wspnnet.org/>
- [4] [www.hb9eyz.ch/g\\_locator.php](http://www.hb9eyz.ch/g_locator.php)
- [5] [www.vb-audio.com/Cable/](http://www.vb-audio.com/Cable/)
- [6] <https://sourceforge.net/projects/fldigi/>

Figure 7 shows the difference made. First the signal was received direct from the sound card. Here you can see the interfering FT8 signals in the wrong sideband. Then we switched over to the virtual audio cable, that is, to the already processed USB signal. In that way the desired signals were decoded free of interference.

If you feel like working with G8JCFSDR, you cannot use direct tuning of the VFO to the USB frequency because the program shows its centre IF of 12 kHz in the middle of the spectrum. However, the decoded USB signal can be routed to any program of your choice for further processing.

### Digital operating modes with fldigi

The 'classic' operating modes in amateur radio are CW and SSB. Somewhat later radio teletype (RTTY) came along. In the meantime numerous new digital operating modes have appeared. Many of these can be received using fldigi, which you can also obtain on the Net [6]. Besides CW, the program also decodes RTTY and the much-loved PSK31, weather fax plus many other signals.

It's not only amateur radio signals that can be decoded with fldigi. Figure 8 shows a reception test of a weather chart of the German weather service. The result indicates some fine tuning is necessary here.

Morse telegraphy can be read too. This works well with relatively strong signals. On the other hand, weak Morse signals are one of the last bastions where humans remain superior to machines. Nevertheless, fldigi can help if you cannot keep up with high Morse speeds or are just starting out. After a while the program adjusts to the current transmission speed. During any gaps in transmission, it attempts to read something from the noise, which can produce somewhat hit or miss results.

Figure 9 shows an example with, in places, perfectly legible decoding.

Gaining in popularity lately is PSK31, a phase shift technique of 31 Baud, which corresponds more or less to the speed of typing on a keyboard. Because PSK31 corresponds to a bandwidth of only 31 Hz, its dependability is even better than CW, for which you need around 200 Hz. The outcome is you can make contacts even with strong noise and low transmit power (Figure 10).

PSK31 uses phase shift keying and shifts the phase by 180 degrees for smooth transitions. The oscillogram (Figure 11) shows that the amplitude has a zero crossing at each phase shift. The bandwidth thus corresponds only to the baud rate. Incidentally, the signal comes from fldigi, which generates transmit signals via the sound card, without connection to a transmitter.

(180284)

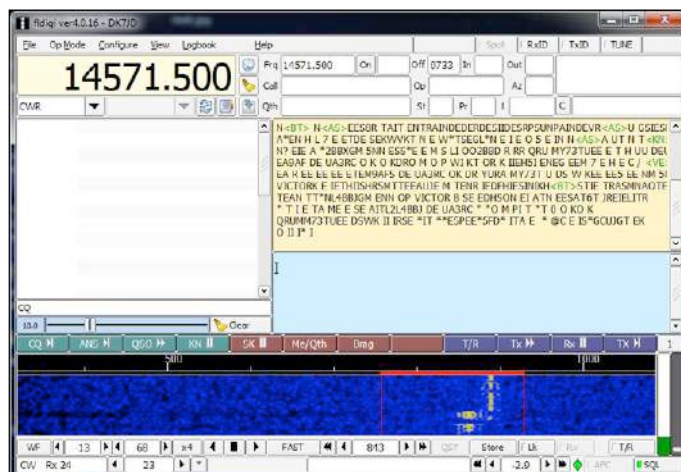


Figure 9. Decoding Morse code signals.

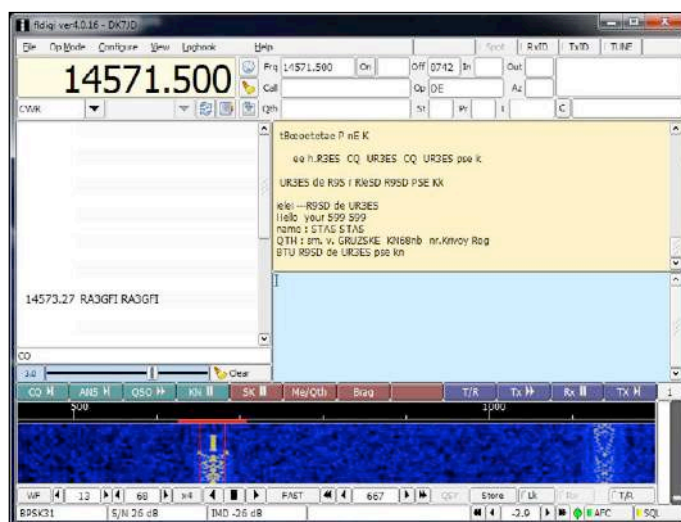


Figure 10. Decoding a PSK31 signal.

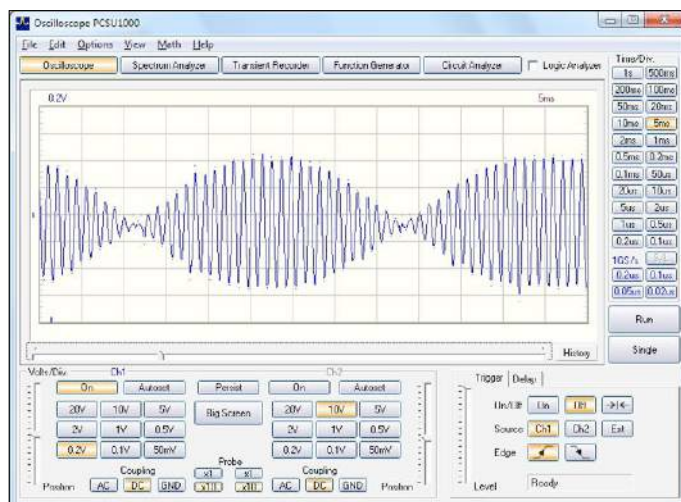


Figure 11. Oscilloscope image of a PSK31 signal.