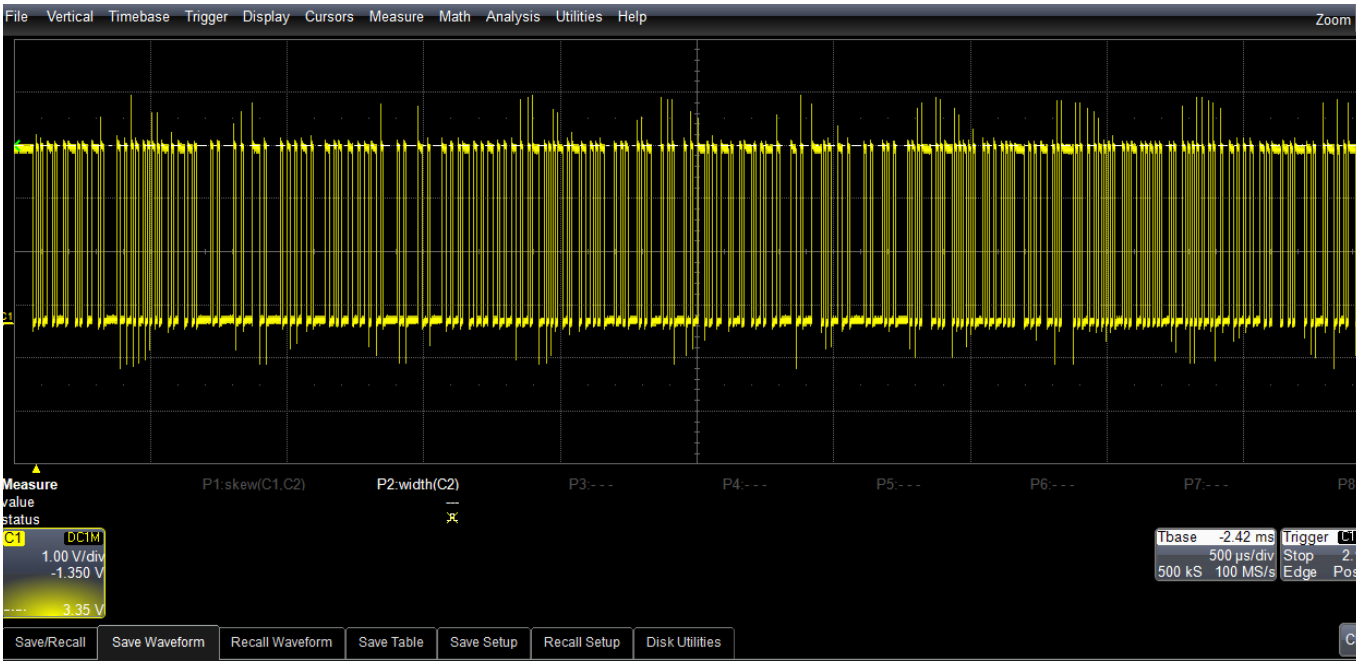
Decoding Lecroy oscilloscope traces with sigrok

Sigrok is an open source signal analysis software suite which is very helpful for the analysis of hardware protocols. The two main tools provided by sigrok are PulseView and sigrok-cli. PulseView is almost the graphical interface equivalent of sigrok-cli. They allow acquiring signal traces from various instruments or from various trace formats and then they allow decoding th with numerous protocol decoders. I would recommend using AppImages if you want to test the latest version without having to build them from source.

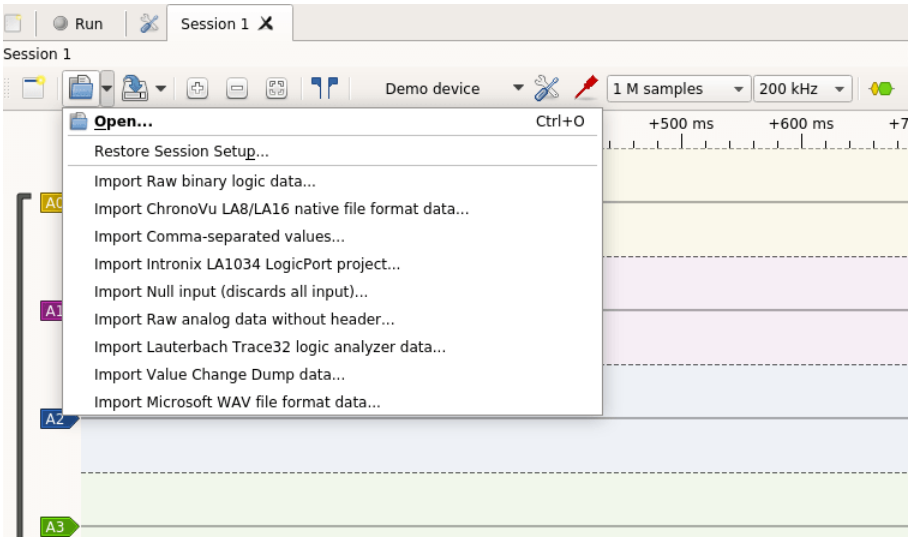
During a hardware analysis, it may happen that we have to spy communication at the hardware level on a PCB. For example, you m want to spy the SMBUS of your computer to see if there is a hardware implant. A convenient tool for that is the oscilloscope. However, as soon as we found an interesting signal, we may want to decode the data which is exchanged to understand what is going on. Let's suppose we have recorded a waveform with a Lecroy oscilloscope and want to make sense of the data exchange



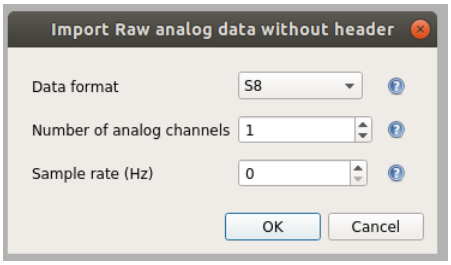
The Lecroy oscilloscope interface allows saving the waveform to a file for further post-processing.

If the data is saved in the "Binary" format, it will create a file with the .trc extension. Its format is fully documented b Lecroy.

Then the PulseView software allows importing external data and supports various file formats:



However, none of the file formats from Lecroy oscilloscopes are (yet) supported out of the box by PulseView. Nevertheless, it allows importing Raw analog data which is a good candidate for us. It only asks for a data format, the number of channels and the sample rate.



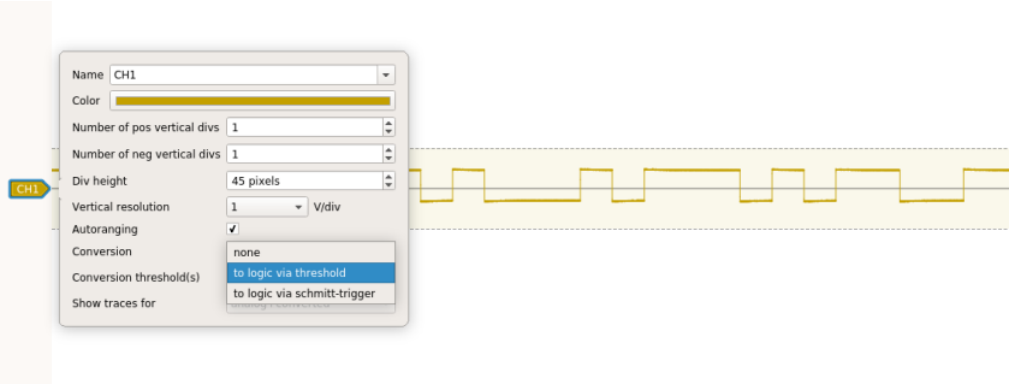
Another very interesting tool I used in the past is lecroyparser. It allows importing a Lecroy waveform saved in the trc file format in Python. Let's have a look at how it interprets the file.

```
if self.commType == 0:
    y = np.fromstring(self.file.read(self.waveArray1), dtype = np.dtype((self.endianness + "i1", self.waveArray1)))[0]
else:
    length = self.waveArray1 // 2
    y = np.fromstring(self.file.read(self.waveArray1), dtype = np.dtype((self.endianness + "i2", length)))[0]
y = self.verticalGain * np.array(y) - self.verticalOffset
x = np.linspace(0, self.waveArrayCount * self.horizInterval, num = self.waveArrayCount) + self.horizOffset
```

```
1
2
3
1
2
4
1
2
5
```

So commType gives the value format (8 or 16 bits) and horizInterval gives the sampling time. I made an open source script adapted from lecroyparser to convert a Lecroy trc trace to a raw analog trace. The script will also display the data format a the sample rate parameters needed to import the trace in PulseView. The generated file can be imported in PulseView and the trace should appear as an analog signal. The voltage values will not be correct since they were not converted, but these are important for the following.

At this point, the signal is considered analog and no decoder can be applied on analog signals in PulseView. Thankfully, clicking on the signal label allows us to convert it to logic data.

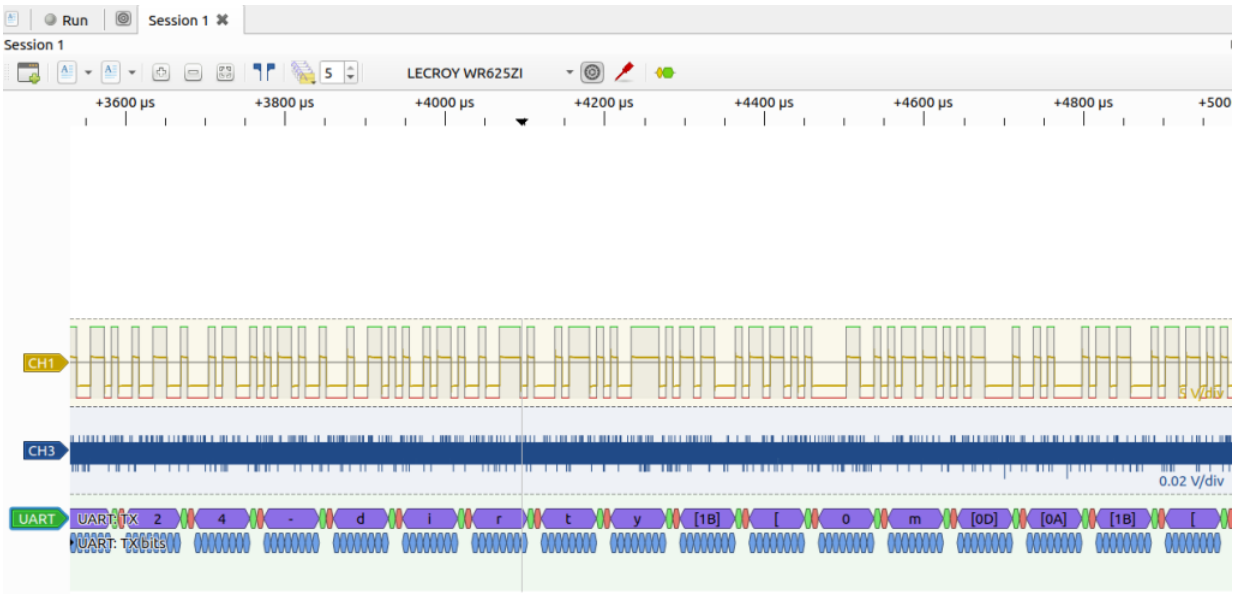


We have chosen to convert the signal using the signal average threshold which works fine on our signal. Then we can add any supported decoders (here UART) and even stack decoders:

The ASCII data can be read directly from the decoder.

If you want to perform the decoding live on your oscilloscope, you can plug a USB cable to the USBTMC connector of the oscilloscope.

USBTMC stands for USB Test & Measurement Class. It is a class of USB devices that allows communication with USB measurement instruments. Basically, it enables you to control and get output data from your instruments. Sigrok has support for various devices using USBTMC and in particular for Lecroy oscilloscopes. After having connected the oscilloscope to the PC and restarting PulseView, it recognizes the oscilloscope directly. When clicking on the Run tab, PulseView will acquire traces fr the oscilloscope. Then the same method as before can be applied to decode the traces:



The main advantage of USBTMC is that the voltage values are correct with this method and also that you can capture several frames.

This article showed how to use PulseView from the Sigrok project to decode traces from Lecroy oscilloscope traces or directly from USBTMC. The presented methodology is powerful since it can be used with all of Sigrok's supported decoders and there are plenty of them.