First use of Universal Radio Hacker

Joshua S Curry and Denis A Nicole
Electronics and Computer Science
University of Southampton*

16th July, 2021

Table of Contents

1 Introduction

Universal Radio Hacker is a relatively small program for the analysis of simple data protocols received over the air on an SDR. It does not support sophisticated modulation schemes such as QAM; it is really designed for basic amplitude and frequency digital modulation. It is an ideal tool for the analysis, "cracking", and "spoofing" of legacy remote control protocols. If you need something more advanced, you should switch to the *Gnu Radio* suite.

In common with Gnu Radio, Universal Radio Hacker's graphical interface uses the PyQt5 library which wraps the Qt framework in Python. There are some issues with the stability of some applications written in this way when running under Microsoft Windows; we have known programs to exit unexpectedly, and have observed an inability to forward screens using, for

* This project was funded under *Cybersecurity Body of Knowledge (CyBOK)* subaward 2021–2471.

example, Microsoft Teams. The effect on Universal Radio Hacker is not serious, but you should follow good practice in regularly saving and backing up your work.

This document uses a 433.92MHz key fob and relay receiver board as our introductory example. The signal is amplitude modulated with on-off keying by an encoder chip.

2 Software Installation

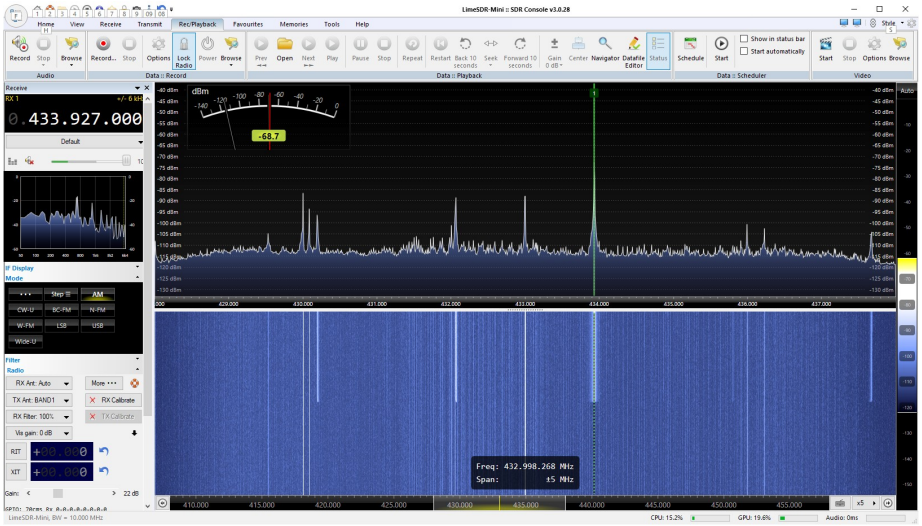
We assume you are running an x64 Windows 10 system. If you already have *SDR Console* installed and working, you need only install *Universal Radio Hacker* from: https://github.com/jopohl/urh/releases

3 Running *Universal Radio Hacker*

It is important that you exit SDR Console before starting Universal Radio Hacker.

Universal Radio Hacker is designed around a specific workflow. The expectation is that, after selecting an SDR device, you will find your signal with the *Spectrum Analyser,* use *Record Signal* to make a recording, and then proceed through the main tabs *Interpretation* and *Generator*. The A*nalysis* and *Simulator* tabs are for more advanced protocols and are not described here. The intent is that you capture a data message, retransmit it for confirmation, understand it, regenerate it, and then modify it to achieve a desired effect.

3.1 Spectrum Analyser



The signal from our keyfob as seen on SDR Console. If we tune to it and select AM, we can actually hear the modulation, which is at about 700Hz.

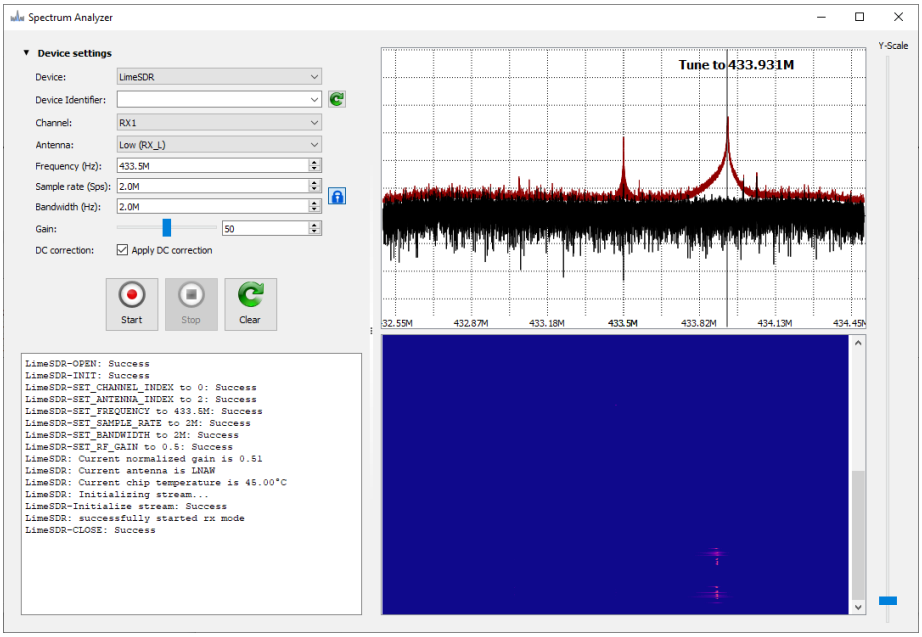
You may already know the frequency of your transmitter or have measured it with SDR Console but you should nevertheless check that everything is working correctly by selecting *File* → *Spectrum Analyzer...* For The LimeSDR Mini, you will need

- Device: LimeSDR
- Channel: RX1
- Antenna: Low (RxL)
- Frequency: perhaps 433.5MHz
- Sample Rate and Bandwidth: 2.0M
- Gain: perhaps around 50.
- DC Correction: Applied

Many SDRs show a *zero frequency* noise peak at the centre of their receiving band. Throughout the capture phases, it is better to offset the centre *Frequency* from the frequency you want to receive, but you must make sure it is within the receiving band. Here we are offset by 420kHz which, being less than 1MHz, is fine.

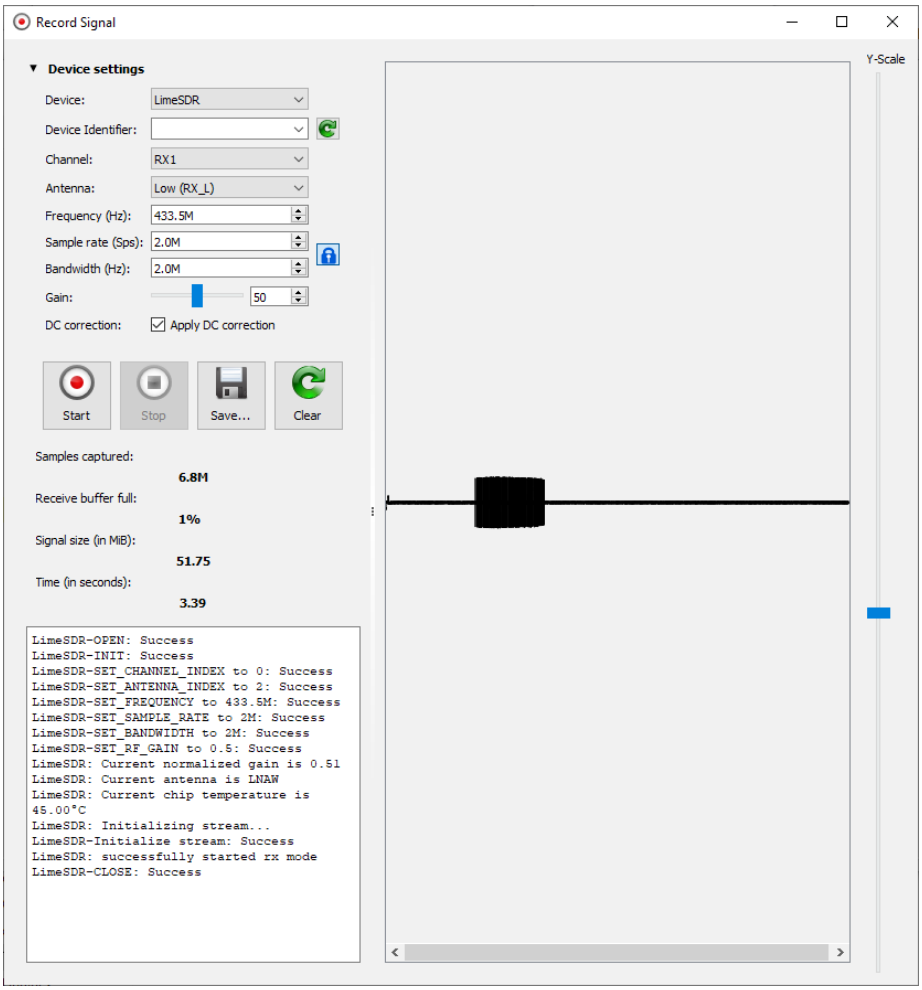
You might have to experiment with your choice of gain to get a strong signal without significant other *spurious* responses.
When you are ready, make sure a suitable aerial is connected to the Rx port on the LimeSDR Mini and click *Start*. Press and release the **A** button on the

keyfob and watch for the signal. You can identify its frequency by dragging the mouse over the plot.



The keyfob signal as seen on Universal Radio Hacker's Spectrum Analyser

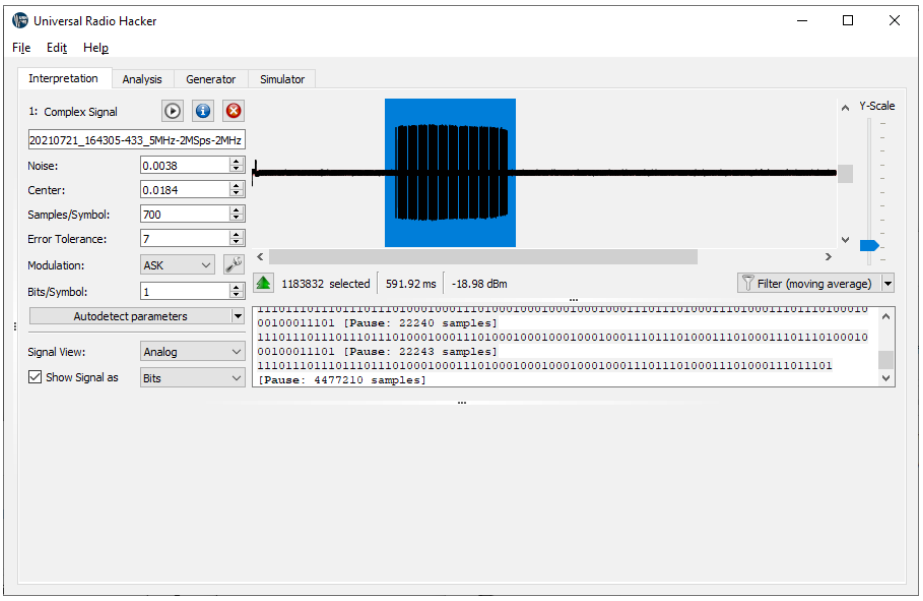
3.2 Capture



A captured keypress

Make sure a suitable aerial is connected to the Rx port on the LimeSDR Mini and select *File* → *Record Signal...* You will want the same settings as used on the Spectrum Analyser window; in particular the *Frequency* should again be offset from the actual keyfob frequency. Click *Start*, record a keypress, press *Stop,* and *Save* the recording. When you close this window, a file should appear in the interpretation window.

3.3 Interpretation



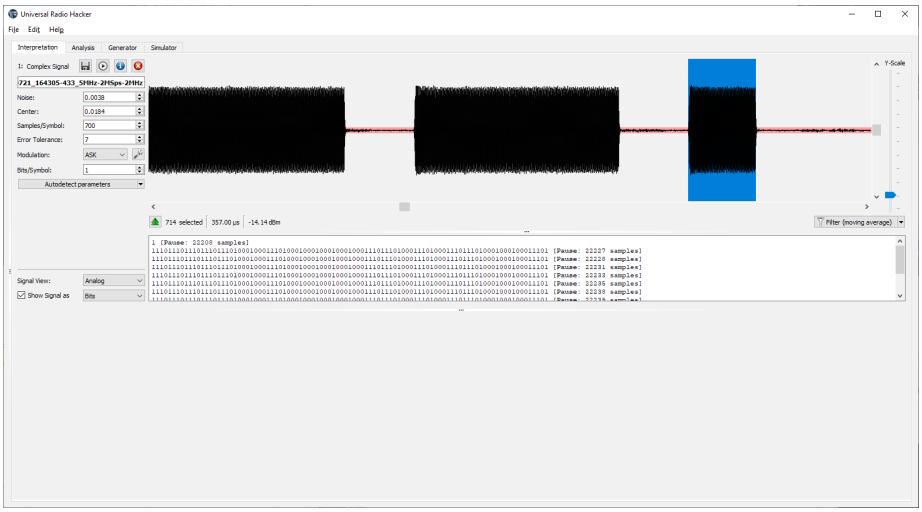
The interpretation window

This is the most delicate step in the capture. With luck, the program will have autodetected the signal parameters, including its *Frequency* and *Samples/Symbol* and displayed some lines of message bits in the lower pane of the window. If not, you can try recapturing the signal, or adjusting the detected (or undetected) parameters and trying to autodetect again. The *noise* and *centre* values can also be dragged on the upper pane.

If you connect an aerial to the Tx port of the LimeSDR Mini you can check your capture using the *Play* button (▶). This retransmits the captured signal unmodified; if it does not activate the receiver, you will need to re-record. Settings for the transmission window should match those used for recording, with *Repeat* set to a small number and *Gain* set near 100. Use the same *Frequency* as for the recording (here 433.5MHz), not the keyfob frequency.

When we have an interpretable signal, we can clean it up and examine it. Use the mouse to select the time period of interest (with the key pressed) and *Right-Click → Crop to Selection.*

The various on and off periods can be measured by selecting them and looking at the sample count. You can do this even if the parameter capture failed and you have to interpret manually.

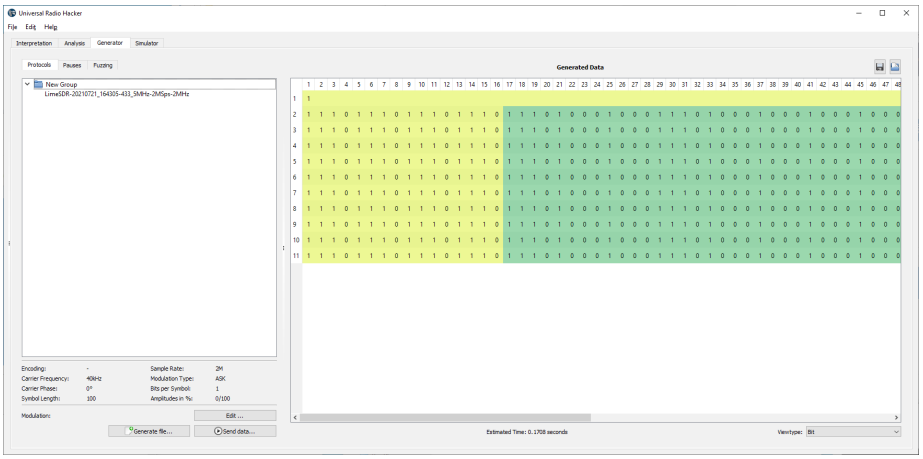


The shortest pulse is about 714 samples.

Some careful counting and checking for multiples will give you the "shape" of the modulation.

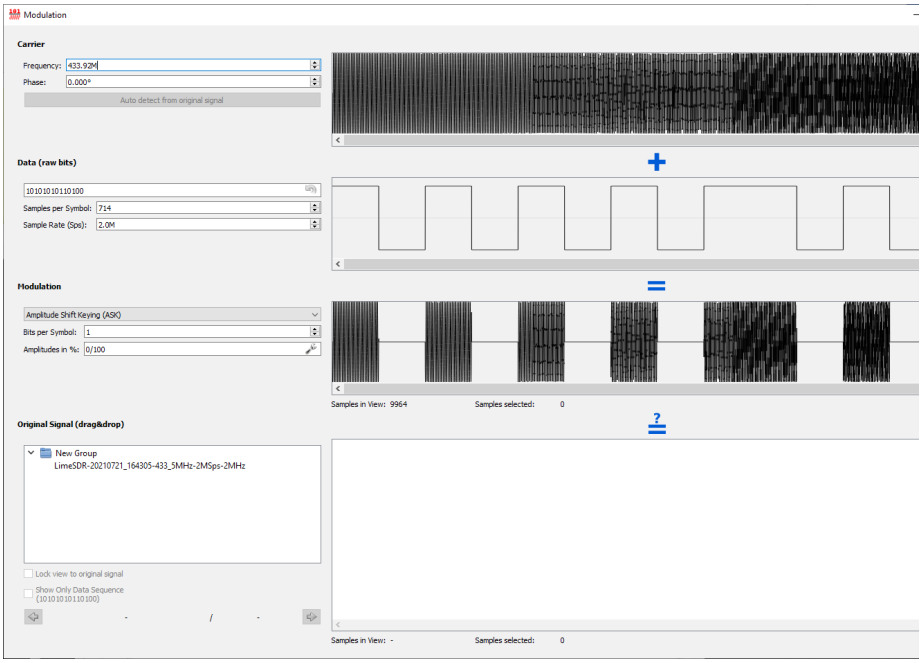
Before we go on to generation, we need to tidy up the bottom window. Delete any lines that do not match one next to them; our aim is to have five or so identical lines.

3.4 Generation



Generation: If you have not used the Analysis tab, your messages will not be colour coded.

Select the *Generation* tab, and drag the filename from the left window into the right. Select all five rows and click *Edit...* The *Modulation* window will open.



Now, for the first time, we set the frequency to the actual 433.92MHz sent by the fob. You can leave the *Phase* at 0°. The *Data (raw bits)* is ignored, but you need to make sure that the *Samples per symbol* and *Sample Rate* are correct at about 714 (as detected) and exactly 2M respectively. *Modulation* is ASK with *Bits per symbol* 1 and *Amplitude* 0/100.

We can now send our regenerated signal through the SDR. Make sure the Tx aerial is connected, click *Send Data...,* confirm that the transmission window parameters are still good and click *Start.* The receiver should again respond.

The key difference from our earlier playback is that, instead of a raw replay, the data bits from the *Generate* window are being modulated anew onto a clean carrier. We now have complete freedom to modify the bits of the messages and try to get different responses from the receiver. You **must** however close the *Modulation* and *Transmission* subwindows before you change a message bit or the old bit pattern will be used.

4 Advanced use

Universal Radio Hacker expects to start its workflow with a live message capture. This is not, however, necessary. If you create a simple `.txt` text file consisting of lines containing only the characters `0` and `1`, you can *File* → *Open* it and drag it into the right hand pane of the *Generator* tab. From there, you can proceed to modify it by inserting or deleting rows and changing bits, before Selecting *Edit...* to set up the modulation and then transmitting your message. This is a good work-around if automatic interpretation fails.

It is also possible to record the raw SDR (I and Q) data in a `.wav` file which can also be read and written by SDR Console. This is a nice capability, for which we have not yet found a use.