jopohl / urh

Universal Radio Hacker: investigate wireless protocols like a boss

security signal-processing wireless iot sdr hacking

2,462 commits **3** branches **52** releases **7** contributors GPL-3.0

Branch: **master** New pull request Find file Clone or download

jopohl refactor code Latest commit `39d1ee1` 3 days ago

.github	Cleanup directories (#377)	5 months ago

data	Multi device support (#432)	6 days ago
src/urh	refactor code	3 days ago
tests	Multi device support (#432)	6 days ago
.gitignore	Windows GNU radio backend fix with custom python2 interpreter (#372)	5 months ago
.travis.yml	use travis for osx builds (#429)	20 days ago
LICENSE	Update LICENSE	8 months ago
README.md	remove circle ci badge from readme	20 days ago
appveyor.yml	remove pip upgrade	25 days ago
setup.py	cleanup compiler warnings (#404)	2 months ago
urh.desktop	remove version from desktop file	5 months ago

README.md

Universal Radio Hacker Black Hat Arsenal USA 2017

Tests Linux/OSX	Tests Windows	Test Coverage	Latest Release
build passing	build passing	coverage 86%	pypi package 2.0.4

The Universal Radio Hacker (URH) is a software for investigating unknown wireless protocols. Features include

- **hardware interfaces** for common Software Defined Radios
- **easy demodulation** of signals

- **assigning participants** to keep overview of your data
- **customizable decodings** to crack even sophisticated encodings like CC1101 data whitening
- **assign labels** to reveal the logic of the protocol
- **fuzzing component** to find security leaks
- **modulation support** to inject the data back into the system

To get started, download the official userguide (PDF), watch the demonstration videos (YouTube) or check out the wiki for more information and supported devices. Scroll down this page to learn how to install URH on your system.

Want to stay in touch? chat on slack

If you find URH useful, please consider giving this repository a ⭐ or even donate via PayPal. We appreciate your support!

Installation

Universal Radio Hacker can be installed via *pip* or using the *package manager* of your distribution (if included). Below you find more specific installation instructions for:

- Linux
  - Via Package Manager
  - Generic way with pip (Ubuntu/Debian)
- Windows
  - MSI Installer
  - Pip
- Mac OS X
- Updating your installation
  - Updating with Pip

    - Updating with MSI
- Running from source

Linux

Via Package Manager

For the following linux distributions you can install URH using your package manager.

Distribution	Install with
Arch Linux	`yaourt -S urh`
Gentoo / Pentoo	`emerge -av urh`
Fedora 25+	`dnf install urh`
openSUSE	`zypper install urh`

Generic way with pip (Ubuntu/Debian)

URH you can also be installed with **pip** using `pip3 install urh`. In case you are running Ubuntu or Debian read on for more specific instructions.

In order to use native device backends, make sure you install the **-dev** package for your desired SDRs, that is `libairspy-dev`, `libhackrf-dev`, `librtlsdr-dev`, `libuhd-dev`.

If your device does not have a `-dev` package, e.g. LimeSDR, you need to manually create a symlink to the `.so`, like this:

```
sudo ln -s /usr/lib/x86_64-linux-gnu/libLimeSuite.so.17.02.2 /usr/lib/x86_64-linux-gnu/libLimeSuite.so
```

before installing URH, using:

```
sudo apt-get update
sudo apt-get install python3-numpy python3-psutil python3-zmq python3-pyqt5 g++ libpython3-dev python3-pip
sudo pip3 install urh
```

Windows

MSI Installer

The easiest way to install URH on Windows is to use the `.msi` installer available here.

It is recommended to use the **64 bit version** of URH because native device support is not available on 32 bit windows. If you get an error about missing `api-ms-win-crt-runtime-l1-1-0.dll`, run Windows Update or directly install KB2999226.

Pip

If you run Python 3.4 on Windows you need to install Visual C++ Build Tools 2015 first.

It is recommended to use Python 3.5 or later on Windows, so no C++ compiler needs to be installed.

1. Install Python 3 for Windows. Choose a **64 Bit** Python version for native device support.
2. In a terminal, type: `pip install urh`.
3. Type `urh` in a terminal or search for `urh` in search bar to start the application.

Mac OS X

1. Install Python 3 for Mac OS X. *If you experience issues with preinstalled Python, make sure you update to a recent version using the given link.*

2. (Optional) Install desired native libs e.g. `brew install librtlsdr` for corresponding native device support.
3. In a terminal, type: `pip3 install urh`.
4. Type `urh` in a terminal to get it started.

Update your installation

Updating with Pip

If you installed URH via pip you can keep it up to date with `pip3 install --upgrade urh`, or, if this should not work `python3 -m pip install --upgrade urh`.

Updating with MSI

If you experience issues after updating URH using the `.msi` installer on Windows, please perform a **full uninstallation**. That is, uninstall URH via Windows and after that remove the installation folder (something like `C:\Program Files\Universal Radio Hacker`). Now, install the new version using the recent `.msi`.

Running from source

If you like to live on bleeding edge, you can run URH from source.

Without installation

To execute the Universal Radio Hacker without installation, just run:

```
git clone https://github.com/jopohl/urh/
cd urh/src/urh
./main.py
```

Note, before first usage the C++ extensions will be built.

Installing from source

To install from source you need to have `python-setuptools` installed. You can get it e.g. with `pip install setuptools`. Once the setuptools are installed use:

```
git clone https://github.com/jopohl/urh/
cd urh
python setup.py install
```

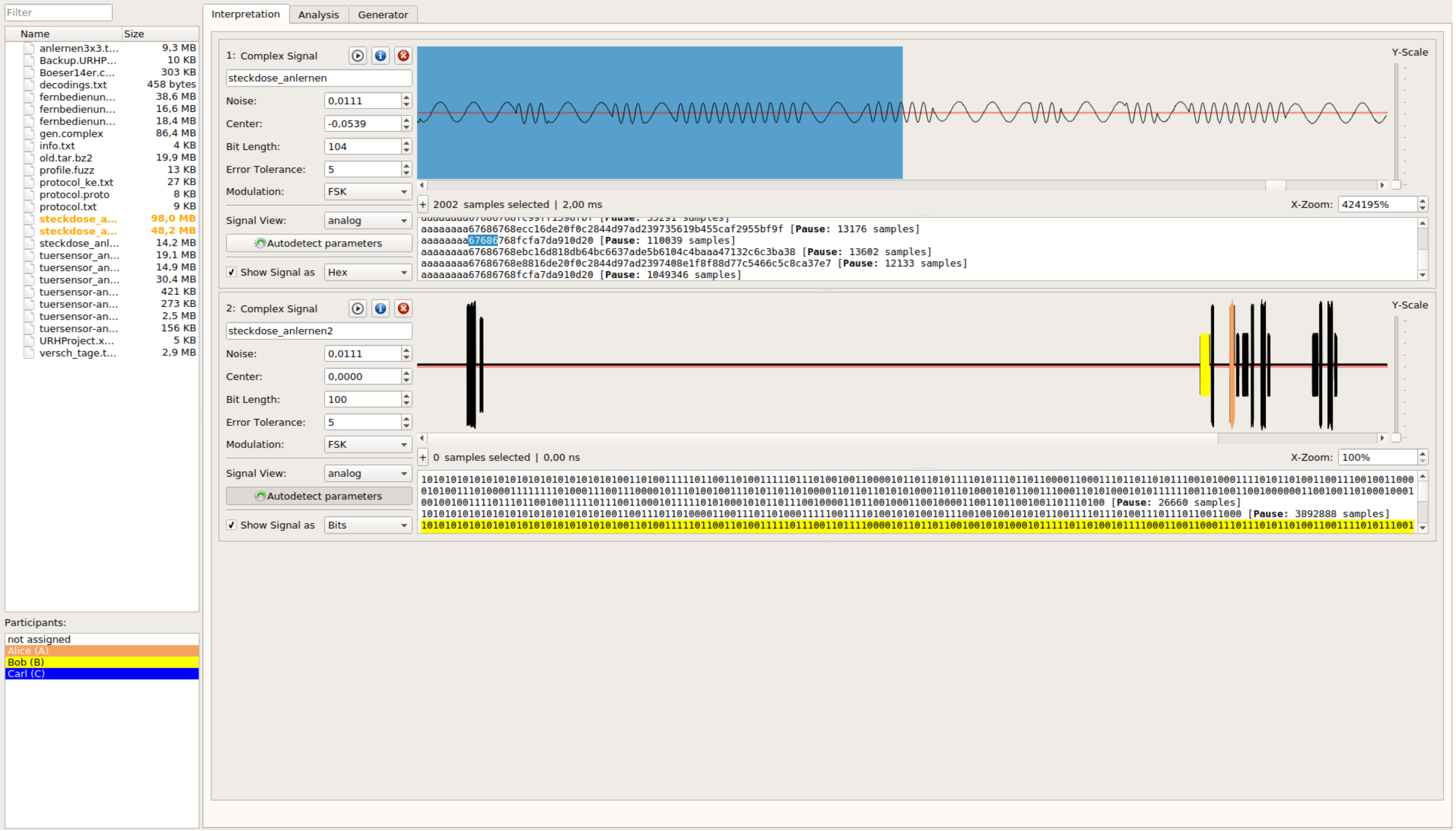
And start the application by typing `urh` in a terminal.

External decodings

See wiki for a list of external decodings provided by our community! Thanks for that!

Screenshots

Get the data out of raw signals



Name	Size
anlernen3x3.t...	9,3 MB
Backup.URHP...	10 KB
Boeser14er.c...	303 KB
decodings.txt	458 bytes
fernbedienun...	38,6 MB
fernbedienun...	16,6 MB
fernbedienun...	18,4 MB
gen.complex	86,4 MB
info.txt	4 KB
old.tar.bz2	19,9 MB
profile.fuzz	13 KB
protocol_ke.txt	27 KB
protocol.proto	8 KB
protocol.txt	9 KB
steckdose_a...	98,0 MB
steckdose_a...	48,2 MB
steckdose_anl...	14,2 MB
tuersensor_an...	19,1 MB
tuersensor_an...	14,9 MB
tuersensor_an...	30,4 MB
tuersensor-an...	421 KB
tuersensor-an...	273 KB
tuersensor-an...	2,5 MB
tuersensor-an...	156 KB
URHProject.x...	5 KB
versch_tage.t...	2,9 MB

not assigned
Alice (A)
Bob (B)
Carl (C)

1: Complex Signal
steckdose_anlernen
Noise: 0,0111
Center: -0,0539
Bit Length: 104
Error Tolerance: 5
Modulation: FSK
Signal View: analog
☑ Show Signal as Hex

2002 samples selected | 2,00 ms
X-Zoom: 424195%

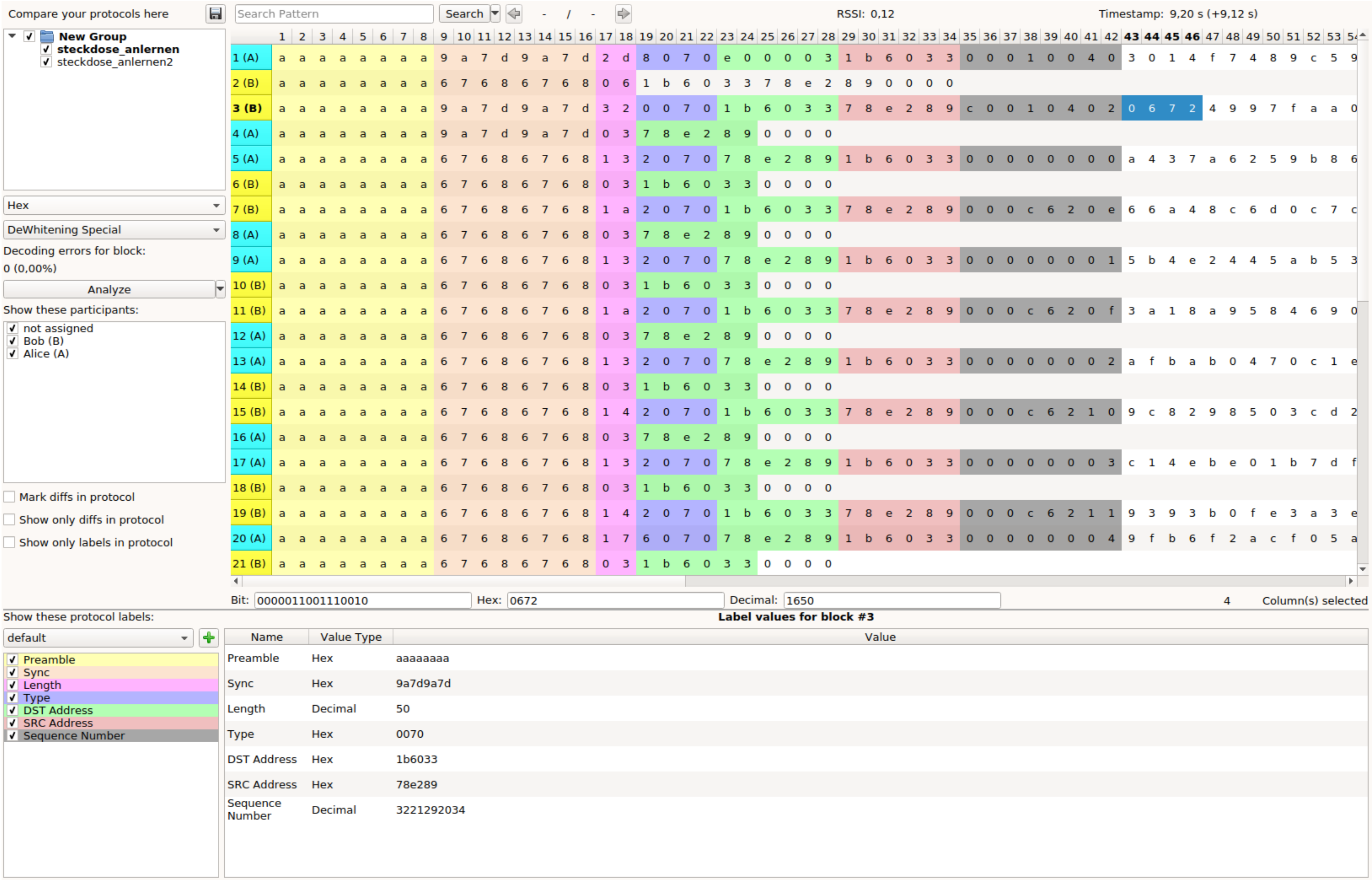
```
aaaaaaaa67686768ecc16de20f0c2844d97ad239735619b455caf2955bf9f [Pause: 13176 samples]
aaaaaaaa67686768fcfa7da910d20 [Pause: 110039 samples]
aaaaaaaa67686768ebc16d818db64bc6637ade5b6104c4baaa47132c6c3ba38 [Pause: 13602 samples]
aaaaaaaa67686768e8816de20f0c2844d97ad2397408e1f8f88d77c5466c5c8ca37e7 [Pause: 12133 samples]
aaaaaaaa67686768fcfa7da910d20 [Pause: 1049346 samples]
```

2: Complex Signal
steckdose_anlernen2
Noise: 0,0111
Center: 0,0000
Bit Length: 100
Error Tolerance: 5
Modulation: FSK
Signal View: analog
☑ Show Signal as Bits

0 samples selected | 0,00 ns
X-Zoom: 100%

```
10101010101010101010101010101010101001101001111101100110100111110111010010011000010110110101111010111011011000011000111011011010111001010001111010110100110011100100110000
0101001110100000111111110100011100111000010111010010011101011011010000110110110101010001101101000101011001110001101010001010111111001101001100100000001100100110100010001
00100100111101110110010011111011100110001011111010100010101101110010000110110010001100100001100110111001001101110100 [Pause: 26660 samples]
10101010101010101010101010101010101001100111011010000110011101101000011111001111010010101001011100100100101010110011110111010011101110110011000 [Pause: 3892888 samples]
101010101010101010101010101010101010011010011111011001101001111101110011011110000101101101100100101010001011111011010010111100011001100011101110101101001100111101011100 1
```

Keep an overview even on complex protocols



Compare your protocols here

Search Pattern | Search | - / - | RSSI: 0,12 | Timestamp: 9,20 s (+9,12 s)

- ☑ New Group
  - ☑ steckdose_anlernen
  - ☑ steckdose_anlernen2

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
1 (A)	a	a	a	a	a	a	a	a	9	a	7	d	9	a	7	d	2	d	8	0	7	0	e	0	0	0	0	3	1	b	6	0	3	3	0	0	0	1	0	0	4	0	3	0	1	4	f	7	4	8	9	c	5	9
2 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	6	1	b	6	0	3	3	7	8	e	2	8	9	0	0	0	0																				
3 (B)	a	a	a	a	a	a	a	a	9	a	7	d	9	a	7	d	3	2	0	0	7	0	1	b	6	0	3	3	7	8	e	2	8	9	c	0	0	1	0	4	0	2	0	6	7	2	4	9	9	7	f	a	a	0
4 (A)	a	a	a	a	a	a	a	a	9	a	7	d	9	a	7	d	0	3	7	8	e	2	8	9	0	0	0	0																										
5 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	3	2	0	7	0	7	8	e	2	8	9	1	b	6	0	3	3	0	0	0	0	0	0	0	0	a	4	3	7	a	6	2	5	9	b	8	6
6 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	1	b	6	0	3	3	0	0	0	0																										
7 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	a	2	0	7	0	1	b	6	0	3	3	7	8	e	2	8	9	0	0	0	c	6	2	0	e	6	6	a	4	8	c	6	d	0	c	7	c
8 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	7	8	e	2	8	9	0	0	0	0																										
9 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	3	2	0	7	0	7	8	e	2	8	9	1	b	6	0	3	3	0	0	0	0	0	0	0	1	5	b	4	e	2	4	4	5	a	b	5	3
10 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	1	b	6	0	3	3	0	0	0	0																										
11 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	a	2	0	7	0	1	b	6	0	3	3	7	8	e	2	8	9	0	0	0	c	6	2	0	f	3	a	1	8	a	9	5	8	4	6	9	0
12 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	7	8	e	2	8	9	0	0	0	0																										
13 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	3	2	0	7	0	7	8	e	2	8	9	1	b	6	0	3	3	0	0	0	0	0	0	0	2	a	f	b	a	b	0	4	7	0	c	1	e
14 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	1	b	6	0	3	3	0	0	0	0																										
15 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	4	2	0	7	0	1	b	6	0	3	3	7	8	e	2	8	9	0	0	0	c	6	2	1	0	9	c	8	2	9	8	5	0	3	c	d	2
16 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	7	8	e	2	8	9	0	0	0	0																										
17 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	3	2	0	7	0	7	8	e	2	8	9	1	b	6	0	3	3	0	0	0	0	0	0	0	3	c	1	4	e	b	e	0	1	b	7	d	f
18 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	1	b	6	0	3	3	0	0	0	0																										
19 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	4	2	0	7	0	1	b	6	0	3	3	7	8	e	2	8	9	0	0	0	c	6	2	1	1	9	3	9	3	b	0	f	e	3	a	3	e
20 (A)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	1	7	6	0	7	0	7	8	e	2	8	9	1	b	6	0	3	3	0	0	0	0	0	0	0	4	9	f	b	6	f	2	a	c	f	0	5	a
21 (B)	a	a	a	a	a	a	a	a	6	7	6	8	6	7	6	8	0	3	1	b	6	0	3	3	0	0	0	0																										

Hex

DeWhitening Special

Decoding errors for block:
0 (0,00%)

Analyze

Show these participants:

- ☑ not assigned
- ☑ Bob (B)
- ☑ Alice (A)

☐ Mark diffs in protocol

☐ Show only diffs in protocol

☐ Show only labels in protocol

Bit: 0000011001110010 Hex: 0672 Decimal: 1650 4 Column(s) selected

Show these protocol labels:

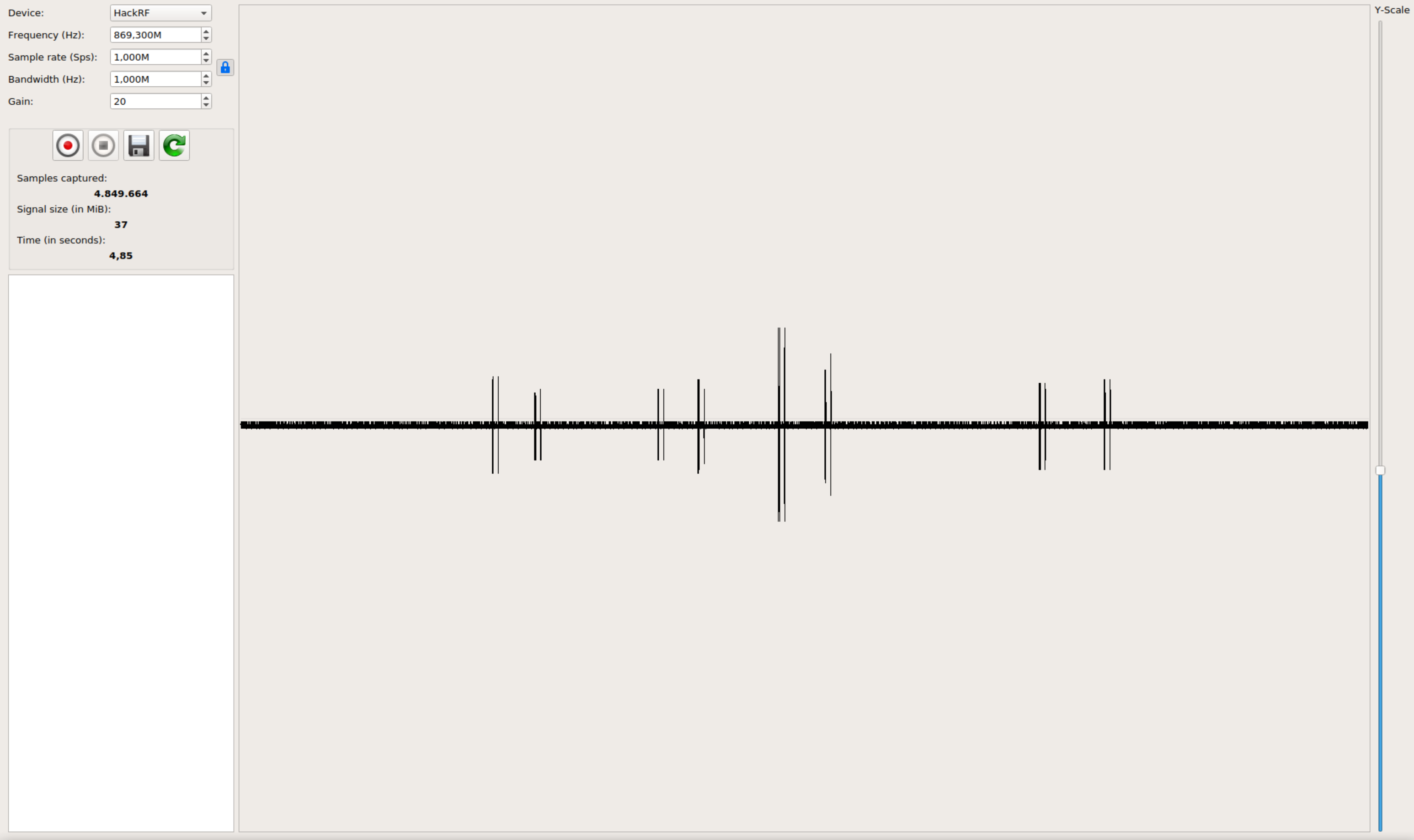
default

- ☑ Preamble
- ☑ Sync
- ☑ Length
- ☑ Type
- ☑ DST Address
- ☑ SRC Address
- ☑ Sequence Number

Label values for block #3

Name	Value Type	Value
Preamble	Hex	aaaaaaaa
Sync	Hex	9a7d9a7d
Length	Decimal	50
Type	Hex	0070
DST Address	Hex	1b6033
SRC Address	Hex	78e289
Sequence Number	Decimal	3221292034

Record and send signals



Device: HackRF

Frequency (Hz): 869,300M

Sample rate (Sps): 1,000M

Bandwidth (Hz): 1,000M

Gain: 20

Samples captured:
4.849.664

Signal size (in MiB):
37

Time (in seconds):
4,85