PandwaRF

User Guide

Contents

Warning

- This guide only covers features available in the PandwaRF "regular" or PandwaRF Rogue Pro versions.
- This guide does NOT cover features available in the PandwaRF Rogue Gov or PandwaRF Marauder versions.
- Please refer to *PandwaRF Rogue Gov User Guide* for description of features of the PandwaRF Rogue Gov version.
- Please refer to *PandwaRF Marauder User Guide* for description of features of the PandwaRF Marauder version.

What is the PandwaRF

PandwaRF is a pocket-sized, portable RF analysis tool operating the sub-1 GHz range.

It allows the capture, analysis and re-transmission of RF via an Android device or a Linux PC.

Practically, it removes the 'standard SDR Grind' of capturing, demodulating, analyzing, modifying and replaying by hand – replacing it with a simple but powerful Android interface.

The PandwaRF system consists of two elements: the hardware device and the Android application.

PandwaRF can be connected to Android using Bluetooth Low Energy – BLE (preferred) or USB.

General Overview

PandwaRF is a RF hacking tool used to:

Receive

- Capture any data in ASK/OOK/MSK/2-FSK/GFSK modulation from the frequency range: 300-348 MHz, 391-464 MHz and 782-928 MHz
- Transfer the captured data to your smartphone & save/share it
- Run a Spectrum Analyzer on a specific frequency band
- Send the captured data in JSON to your own server for post-processing
- Write your own JS scripts or use a provided one

Transmit

- Transmit previously captured data or write your own
- Transmit data from a smartphone or directly from PandwaRF
- Brute force with a predefined transmission pattern (encoders or devices)
- Transmit power: +10dBm

Analyze

- Visualize the frequency used by any device using the PandwaRF built-in Spectrum Analyzer
- Directly show the maximum and average RSSI for a specific frequency band

Possible applications include:

- Receive keyfobs transmission (car, alarm, gate opener, …)
- Replay captured transmission from keyfobs
- Replay a modified captured transmission
- Transmit your own custom payload
- Capture RF data and transmit it on another frequency
- Brute force wireless devices (alarms, gate openers etc.)
- Spectrum Analyzer
- Find the frequency used by a RF device
- Reverse engineer unknown protocols

- Measure the data rate of a transmission
- Check the RF jam-resistance of your own devices
- Send captured data to a server for post-processing
- Write custom JavaScript scenarios
- Develop your own Android application

Warning: PandwaRF is a test equipment for RF systems. It has not been tested for compliance with the regulations governing the transmission of radio signals. You are responsible for using your PandwaRF legally.

The intentional jamming of RF signals is ILLEGAL. PandwaRF is not designed for RF jamming and should only be used for testing the robustness of your own devices.

Good practices

It is recommended that you familiarize yourself with PandwaRF features by using a SDR device. That will show you what happens for example when PandwaRF performs:

- a brute force with different Function Mask or symbols Encoding,
- a transmission with different data rates, different modulations or different deviation values,
- etc.

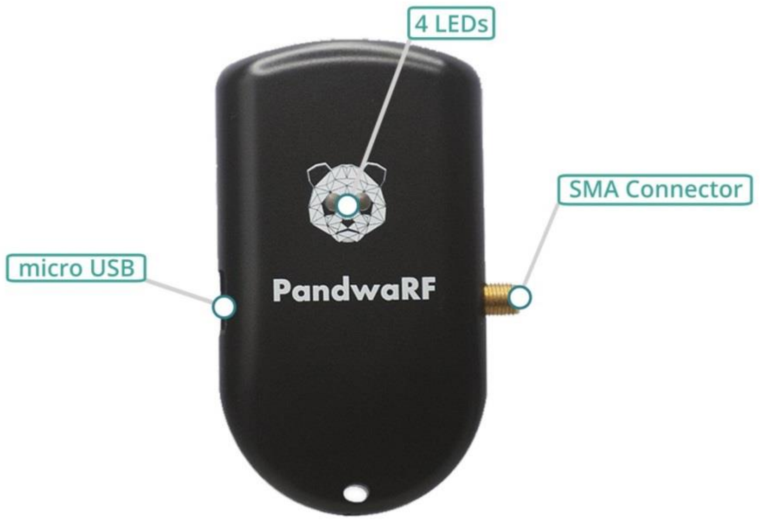
What's in the box

If you ordered a **PandwaRF Essentials Kit or a Rogue Pro**, you should have:

- 1x PandwaRF/Rogue Pro
- 1x Micro USB OTG Male-Male cable
- 1x Micro USB female to USB Type-C male Converter
- 1x 315MHz antenna
- 1x 433MHz antenna
- 1x 868MHz antenna
- 1x protective case

If you ordered a **PandwaRF or a PandwaRF Bare**, only the device is included.

PandwaRF



Antennas

Using the proper antenna is critical to have good RF performance.

Here is how to identify each antenna:

Antennas

Antenna for 315 MHz

Antenna for 433 MHz

Antenna for 868-915 MHz

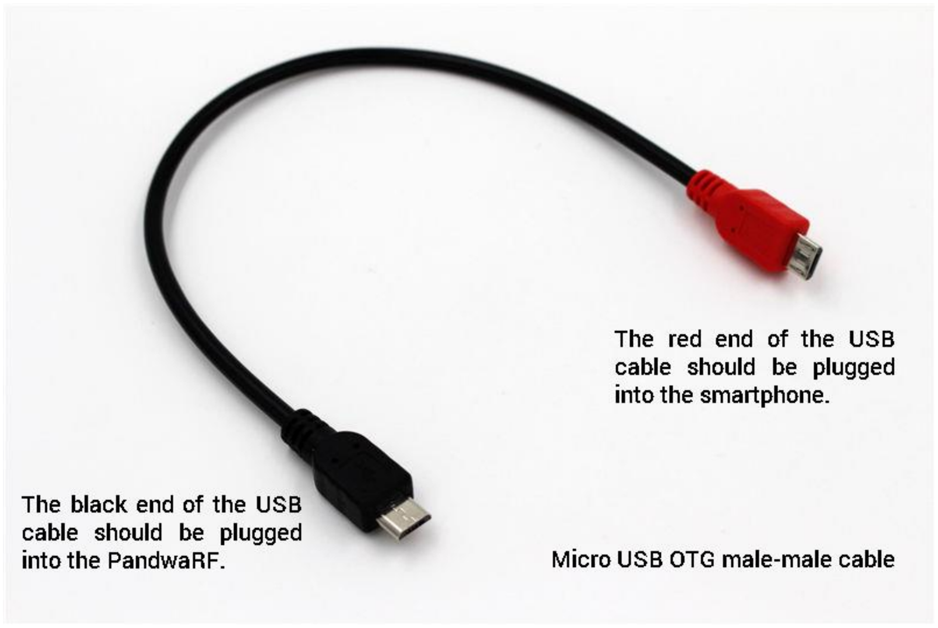
Antennas are usually labelled with the first digit of their frequency band: 3 for 315 MHz, 4 for 433 MHz, 8/9 for 868/915 MHz.

Note: Non-contractual pictures. The antennas might vary in number, shape and size based on the supplier we use at the moment.

Micro USB OTG Male-Male cable

If you decide to connect PandwaRF to Android using a USB cable, you need a Micro USB OTG Male-Male cable (provided). Both ends of the cable are not equal:

- The red end should be connected to the Smartphone
- The black end is connected to PandwaRF


The red end of the USB cable should be plugged into the smartphone.
The black end of the USB cable should be plugged into the PandwaRF.
Micro USB OTG male-male cable

Do not try to assemble 2 normal USB cables. The result is highly uncertain. Use a Micro USB OTG Male-Male cable as indicated in page 6 Micro USB OTG Male-Male cable

Quick Start

Here are some quick steps to get you started with PandwaRF.

1. Download the PandwaRF Android application (https://play.google.com/store/apps/details?id=com.comthings.pandwarf).

2. Connect an antenna to the PandwaRF

3. If it is the first use after unpacking the device, you need to charge and wake up PandwaRF. To wake up:

    - **Preferred method**: Plug PandwaRF onto an USB power source (the Orange & Blue lights will blink slowly to indicate charging & BLE advertising), or
    - **Alternative method**: Open the plastic enclosure and press any button (the Blue light will blink slowly to indicate BLE advertising).

3. Check that your Android phone has GPS enabled. See why in next chapter.

4. Start the PandwaRF Android app. If Android pops up a dialog to request for location and storage permission, you need to grant access. (Cf. Android Application permissions)

5. From the **Scan** tab, all the nearby PandwaRF devices will be searched automatically. PandwaRF can be used while charging.

6. Choose the PandwaRF to connect to by clicking on it. The blue LED on PandwaRF device will stop blinking and remain ON. The app screen will change to the **Bus Service** tab, showing you the device information (MAC address, battery level, RSSI level, enabled features etc.).

7. The status in **BUS Service** tab should be *Ready* and link icon should be green.

8. Navigate to **RX/TX** or **SpecAn** (Spectrum Analyzer) pages to start having fun...

9. When you are done having fun, you can disconnect from the device using the **Disconnect** button.

You will find many more details in our wiki: https://github.com/ComThings/PandwaRF/wiki

Android Application permissions

The PandwaRF application uses 2 types of permissions:

Storage

Storage permission is used to read/write files (JavaScript, saved sessions, captured data, …) onto the device.

Location

On Android version 6.0 Bluetooth Low Energy (BLE) scanning will only work if Location services are enabled on the device. This is a requirement from Google. If you don't grant location permission to the app, the scan may not find any BLE device. We are aware that this is not a convenient situation and we are working on an alternate solution that does not require users to grant Location permission to the app.

Hardware

Opening the enclosure

PandwaRF can be easily opened as the top enclosure part is not stuck to the bottom enclosure part.

You can use a coin to open the enclosure.

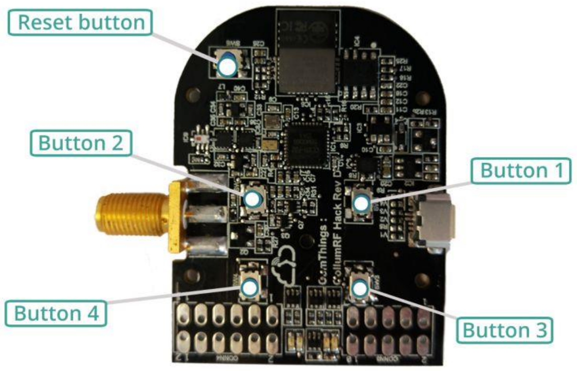
Using the internal buttons

On some very particular occasions, you need to press a button.

These occasions are:

- If you want to link/pair your PandwaRF to your phone and prevent any other phone to connect to it
- Manually shutdown the PandwaRF
- Reset the PandwaRF if it becomes unstable

Button position



Button	Short press	Long press (>2s)
SW1	TX config 1	Shutdown PandwaRF
SW2	TX config 2	Whitelist bypass mode
SW3	TX config 3	Delete all bonding information
SW4	TX config 4	
SW5	Reset	

TX configs are data which has been sent to PandwaRF internal memory and can be retransmitted without the need of the Android application. Cf. What to do with your captured data.

Whitelist bypass mode

Temporarily turn off usage of the whitelist until the next connection. Pressing ***SW2*** for 2s allows connecting to a phone which is not in the whitelist. This is valid for a single connection. To be granted access to PandwaRF permanently, this phone must then bond.

Delete all bonding information

- Pressing ***SW3*** for 2s or
- Pressing ***SW3*** and press/release ***SW5*** reset at the same time to clear all bonding information. This allows any phone to connect to PandwaRF.

Shutdown

Pressing ***SW1*** for 2s will force PandwaRF to power off until either a button is pressed again, or a USB cable is plugged-in.

Signification of the LEDs

For an explanation on the LEDs' meaning, check the LEDs Indication States page in our wiki:
https://github.com/ComThings/PandwaRF/wiki/Hardware-LEDs-Indication-State

Eye	Color	Used for
Right	🟠 Orange	USB Charging status
Right	🔵 Blue	BLE state
Left	🟢 Green	RX
Left	🔴 Red	Blink: TX, On: Error

Power Management

PandwaRF low power mode

When not plugged onto USB and not connected in BLE:

- PandwaRF enters low power mode
- PandwaRF only advertises to allow BLE connection
- In this mode the Blue LED blinks every 5s.

When plugged onto USB:

- the Orange LED blinks every 1s while charging and stays on when charge is complete.

When connected in BLE:

- the Blue LED stays on

PandwaRF shutdown

In case you need to shut down PandwaRF completely, there are 2 methods:

- open the enclosure and press any of the 4 buttons (not the reset button) for 2s
- when connected to your PandwaRF, go to **Bus Service** page and click on **Power Off**

PandwaRF will shut down completely and will stop advertising.

Pressing any of the 5 buttons or providing USB power will wake up the PandwaRF.

Warning: in this mode, your smartphone will not be able to discover or connect to your PandwaRF.

Battery charging

PandwaRF has an integrated Battery Gas Gauge, allowing to precisely measure the remaining battery capacity. For the measurement to be precise, PandwaRF needs to be fully charged at least once. It will then initialize its coulomb counter to 100%. Once unplugged from the power source, it will start monitoring its own consumption.

- When charging, PandwaRF’s Orange LED blinks once per second.
- When fully charged, PandwaRF’s Orange LED remains ON.

Using the Android application

Scanning and connecting

1. In the **Scan** page, press the **Scan** button
2. The app will search for all PandwaRF in range. Use the MAC address to tell the difference between several PandwaRF devices.
3. The received signal level (RSSI) is displayed, in dBm negative value (the higher the RSSI, the closer the device is)
4. Click on the device to connect
5. Status will change to *Connected*, then *Ready*
6. Device cannot operate until status is *Ready*

7. Once device is *Ready*, the **BUS Service** page is displayed. It gives you information about your PandwaRF model, FW versions, battery level etc.

Spectrum Analyzer

This feature is useful to quickly identify on which frequency a device is communicating.

Steps:

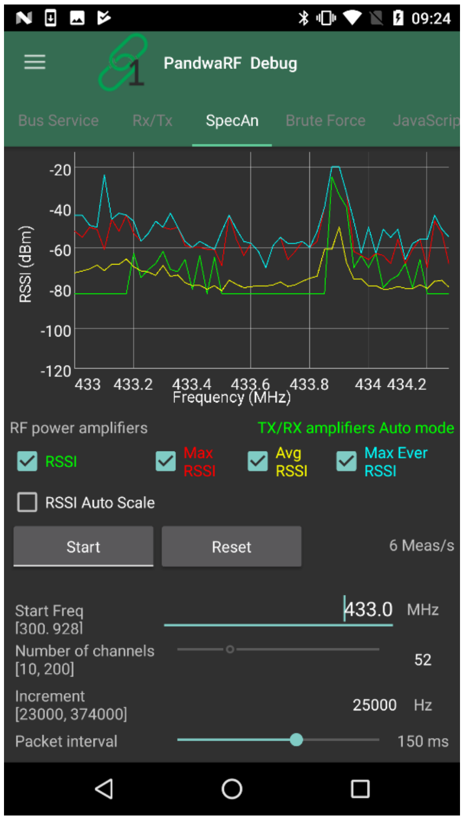
1. Select the frequency band (315MHz, 433MHz, …)
2. Start the **Spectrum Analyzer**
3. Click on the highest peak in the graph. The corresponding frequency will be automatically picked and used when capturing or transmitting data, brute forcing etc.

Parameters:

- RSSI: real time RSSI value, in dBm
- Max RSSI: highest RSSI value observed for the last X measurements, in dBm
- Avg RSSI: Average RSSI, computed from the last X measurements, in dBm
- Max Ever RSSI: highest RSSI value observed since the beginning of the capture, in dBm
- RSSI Auto Scale: if the set RSSI range is automatically computed from min/max RSSI observed. Use this mode in case of very low signals.
- Start Freq: base frequency, in MHz
- Number of channels: number of RSSI samples captured, starting at Start Freq
- Increment: channel spacing (separation between each measure), in Hz.
- Packet interval: delay between each capture of a full frequency range, in ms. Only used in BLE mode.

Note: End Freq = Start Freq + (Number of channels * Increment)

Spectrum Analyzer in BLE mode



RF power amplifiers — TX/RX amplifiers Auto mode

✓ RSSI ✓ Max RSSI ✓ Avg RSSI ✓ Max Ever RSSI

☐ RSSI Auto Scale

6 Meas/s

Start Freq [300, 928]: 433.0 MHz

Number of channels [10, 200]: 52

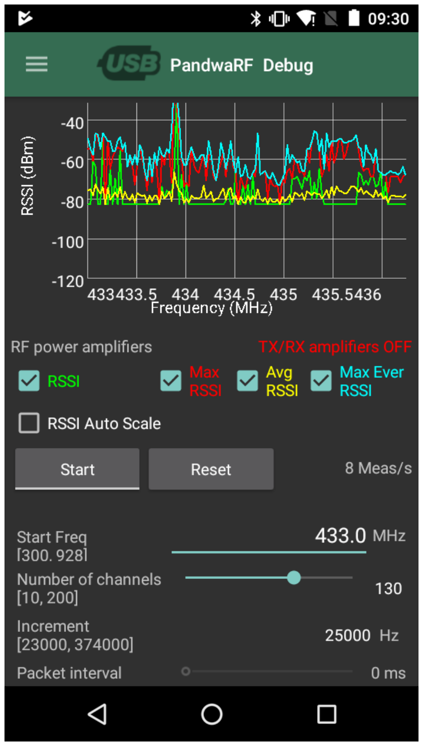
Increment [23000, 374000]: 25000 Hz

Packet interval: 150 ms

BLE has a limited bandwidth so some constraints are set:

- the number of channels is fixed to 52 to optimize BLE bandwidth
- the packet interval is set to default to 100 ms to not saturate the BLE link. You can change this value if you think you have a much better BLE connection with your phone.

Spectrum Analyzer in USB mode



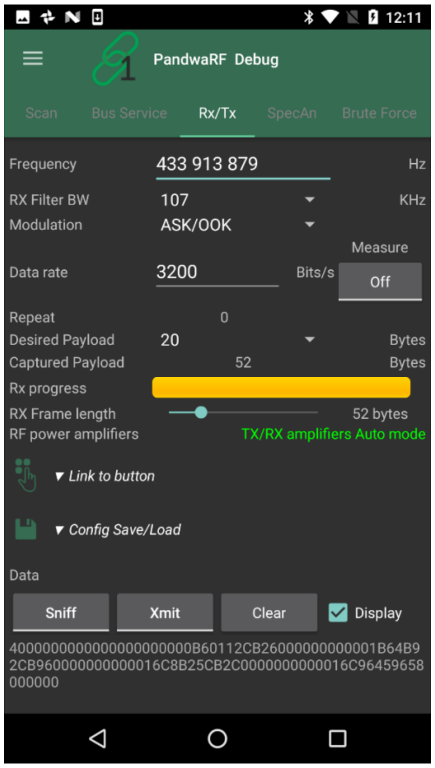
Start Freq [300. 928] 433.0 MHz
Number of channels [10, 200] 130
Increment [23000, 374000] 25000 Hz
Packet interval 0 ms

Since USB has a much higher throughput rate than BLE, the number of channels can be adjusted from 10 to 200. The Packet interval is also reduced to 0 ms.

RX/TX

This tab is used to:

- capture data from your PandwaRF
- transmit data to your PandwaRF



Frequency 433 913 879 Hz
RX Filter BW 107 KHz
Modulation ASK/OOK
Data rate 3200 Bits/s
Measure Off
Repeat 0
Desired Payload 20 Bytes
Captured Payload 52 Bytes
Rx progress
RX Frame length 52 bytes
RF power amplifiers TX/RX amplifiers Auto mode
Data
Display [✓]

```
40000000000000000000000B60112CB260000000000001B64B9
2CB960000000000016C8B25CB2C0000000000016C96459658
000000
```

Parameters

Common(RX or TX)

- **Frequency** in Hz: the range is [300000000, 928000000] Hz
- **Modulation**: 2-FSK, GFSK, MSK, ASK, and OOK modulation formats are supported
- **Deviation**:
  - RX: Specifies the expected frequency deviation of the incoming signal. It should be the same as the target device TX deviation for demodulation to be performed reliably and robustly.
  - TX: Specifies the nominal frequency deviation from the carrier frequency for a 0 (-DEVIATN) and a 1 (+DEVIATN)
- **Data rate**: do not go higher than 10000 Bits/s if you are doing data capture. Remember: it is always better to know the target device data rate. See *Data Rate Measurement* for more information.
  - In RX, it corresponds to the data sampling rate.
  - In TX, it corresponds to the data rate for sending the data.
- **Frame Length**:
  - In RX mode: you can specify the size of the packet that the CC1111 transceiver needs to capture before sending it to Android Application.
  - In TX mode: This indicates how data present in the Data section will be split into smaller chunks and sent to CC1111 for RF transmission.
  - Be aware that there will be some blank data between 2 split chunks, for RX mode or TX mode.
- **Frame compression:**

  - Allows support for higher data rates. When enabled, the data between PandwaRF and the Android App is compressed before being sent. Available for Rogue Pro only.

RX parameters

- **RX Filter BW** in KHz: Receiver Channel Filter Bandwidth.

  For best performance, the channel filter bandwidth should be selected so that the signal bandwidth occupies at most 80% of the channel filter bandwidth. If RX capture with Auto value gives incorrect results (lot of 0), it probably means that the **Frequency** value is not centered onto the signal you are trying to capture. The solution can be to adjust the **Frequency** or to increase the RX Filter BW (but doing so may also capture some side signals). The range is [54-750 KHz].

- **Desired Payload**: Indicate how many bytes you want to capture. This is only valid for RX. The capture's duration will be: Desired Payload (in byte) x 8 x Data rate seconds.

  - If the value is too low, the captured sequence may not be complete.

  - If the value is too high, the PandwaRF app will not stop the capture by itself and you will have to stop it manually.

  The Desired Payload is only used to indicate to PandwaRF how many bytes to capture. When PandwaRF transmits data, it sends all data currently present in the Data section.

- **Captured Payload**: indicates the number of bytes already captured

TX parameters

- **Repeat**: Number of SW or HW repeat times.
  - 1 = single TX (no repetition)
  - 2 = 2 transmissions, etc.
  - If the data buffer fits into 1 frame, repeat is handled by FW (no delay).
    If the data buffer is longer than 1 frame, repeat is handled by SW (with some delay).

Capture and transmit data

- **Sniff**: send capture data order to the PandwaRF dongle, using setup parameters

- **Xmit**: transmit whatever data is shown below this button, using setup parameters

- **Clear**: erase RX/TX data buffer (Android side)

- **Display**: uncheck if you are not interested in viewing the captured data. The most useless parameter of this app.

Details on RF data reception (aka RF sniffing)

It is important to remember that every RF data you request PandwaRF to capture will then travel to the phone using a Bluetooth Smart (BLE) connection. This connection is relatively slow and depends on what smartphone you have (see Measuring the throughput of the BLE link between your Android phone and PandwaRF).

So it is important to capture data at a correct rate. A correct rate means:

- not too high as it would be a waste of bandwidth and might overload the BLE link
- not too low as it could cause the captured data to be wrong (missing bits due to under-sampling)

To measure the data rate of the RF device you want to sniff, check *Rx Data Rate Measurement*.

Steps for capturing RF data from a target keyfob

1. Find the frequency of the keyfob to capture
    - Always try to know the center frequency onto which the keyfob is transmitting (!).
    - Setting the correct frequency has an impact on RX data (of course). The less you know the exact frequency, the more you need to increase the RX channel filter bandwidth to get a chance to capture the keyfob signal. But you also get more RF noise...
    - For example 433 MHz is not enough information, you need to know if it is 433.42 MHz, 433.92 MHz, etc...
    - If you don't know exactly the frequency, you will need to fine tune the RX channel filter bandwidth and increase it step by step until you receive correct data. Data is assumed correct when it is not all zeros.
    - As a general rule, 80% of the signal to capture should be within Frequency +/- RX channel filter bandwidth. Quoting the CC1111 specification: "For best performance, the channel filter bandwidth should be selected so that the signal bandwidth occupies at most 80% of the channel filter bandwidth."

    If you think it is around 433.x, but don't really know, I suggest trying 433.92 with a RX filter bandwidth set to 150KHz.

    Note: You can measure the exact frequency by using the PandwaRF Spectrum analyzer. It is less precise than a SDR, but it should do the trick.

    Cf. Spectrum Analyzer section.

2. Modulation

    You also need to know what the used modulation is. Keyfobs are mostly OOK, but we have also seen PSK or 2-FSK for some keyfobs.

3. Data rate
    - Measure the keyfob data rate by pressing the data rate **Measure** button. Press the keyfob’s button to force transmission until the data rate stops changing.
    - Increase the desired payload to 250 bytes to be sure to receive enough data. You can always reduce it later if you see your keyfob transmits less than 250 bytes.

4. Start the capture
    - Press the **Sniff** button and wait until PandwaRF is ready. The status phases are:
        - RX Setup: PandwaRF device is preparing for data acquisition

    - Receiving: PandwaRF is ready to capture data
- Force transmission by pressing the keyfob button

The captured data should look like some random data.

- If there is no data captured, it means that the frequency is not correct at all.
- If you get all zero, it means that the frequency is almost correct, or the RX filter bandwidth is too small. In both cases, try again with another frequency/RX filter bandwidth as described previously.

What to do with your captured data

Once the data is captured, you have several options.

- **Re-transmit:** if you want to perform a replay attack, just press the **Xmit** button. Data will be transmitted as displayed. If the capture has been made with an incorrect data rate, then the transmitted data will not be understandable by the target receiver.
- **Link to button**: pressing a button will copy a previously captured data into the PandwaRF's internal memory associated to a button (1 to 4 slots). You can then transmit this data by:
    - pressing the physical button on the PandwaRF board. You have to open the enclosure cf. Button position, or
    - navigate to **Bus Service Extended** page. You have to enable Developer Mode.
- **Post to API**: you can send the captured data to your own server for processing.
- **Save**: save the captured RX data to Android data storage
- **Load**: load RX data from Android data storage
- **Delete all**: erases previously saved RX data from internal storage. Be careful if you want to keep some data...

Rx Data Rate Measurement

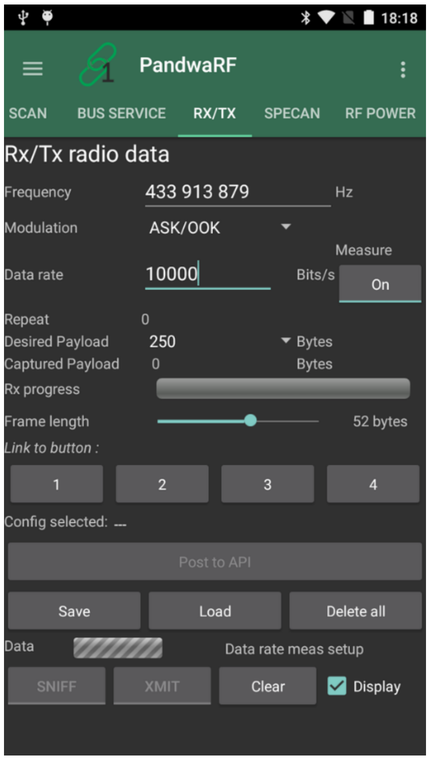
Before trying to capture RF data, you need to know what the data rate of the RF data being sent by your device is.

Set up the data rate measurement

In the **RX/TX** tab:

- Set up the frequency your device is using to transmit.
- Click on the **Measure** button
- Warning: the data rate which was previously set in the **Data rate** Edit Text will be erased.

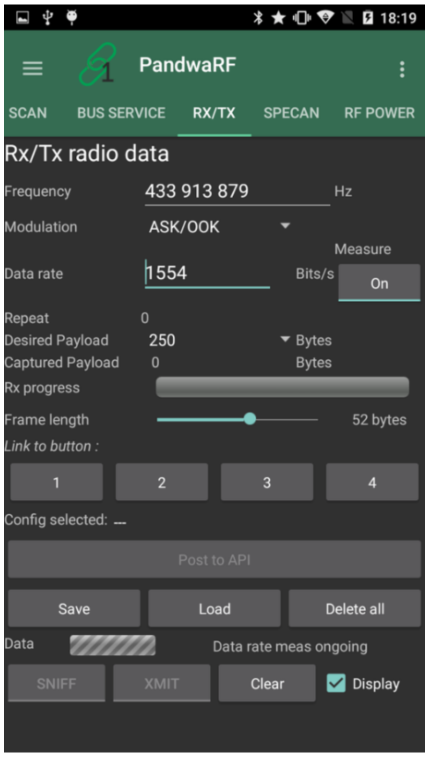
Note: PandwaRF uses a fixed internal sampling rate of 100 Kbits/s. You cannot change this.



Rx/Tx radio data
Frequency 433 913 879 Hz
Modulation ASK/OOK
Data rate 10000 Bits/s Measure On
Repeat 0
Desired Payload 250 Bytes
Captured Payload 0 Bytes
Rx progress
Frame length 52 bytes
Link to button :
Config selected: ---
Data Data rate meas setup

Result of the data rate measurement

- The data capture indication will display "Data rate meas setup" and the progress bar will be animated
- Force a transmission on the device you want to measure, eg. if it is a remote control, press the button
- The data rate will slowly converge to whatever the data rate of your device is, in our case: 1554 Bits/s
- Note: no RX data will be sent by PandwaRF when doing a Data rate measurement. PandwaRF measures the data rate internally and only sends the results to the Smartphone.



Rx/Tx radio data
Frequency 433 913 879 Hz
Modulation ASK/OOK
Data rate 1554 Bits/s Measure On
Repeat 0
Desired Payload 250 Bytes
Captured Payload 0 Bytes
Rx progress
Frame length 52 bytes
Link to button :
Config selected: ---
Data Data rate meas ongoing

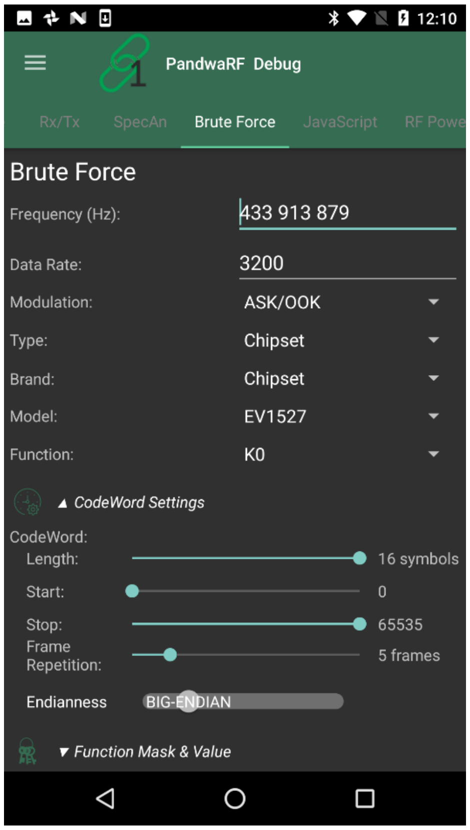
You can now use this optimized data rate as a basis for capturing RF data for your device. Of course you can always use a higher data rate, but it will use more bandwidth of the BLE link. If you want correct RF data capture, do not use a data rate lower than the measured value. For example if you think your device is transmitting at 3000 Bits/s, you must sample at least at twice this rate just to have enough precision.

Brute Force

PandwaRF has an integrated brute force feature. It can send many RF codes consecutively, and supports multiple types of encoding. The brute force mechanism runs entirely on the PandwaRF board, not on the smartphone, making it faster than a normal data transmission from the smartphone (**RX/TX** page) or a JavaScript.

Note: Have a look also at the Android Brute Force Tutorial (https://github.com/ComThings/PandwaRF/wiki/Android-Brute-Force-Tutorial).

Note: while the PandwaRF is able to perform Brute Force, it is not as fast as the PandwaRF Rogue, which has a reworked firmware optimized for brute forcing wireless devices efficiently.



Brute Force

Frequency (Hz): 433 913 879

Data Rate: 3200

Modulation: ASK/OOK

Type: Chipset

Brand: Chipset

Model: EV1527

Function: K0

CodeWord Settings

CodeWord:

Length: 16 symbols

Start: 0

Stop: 65535

Frame Repetition: 5 frames

Endianness BIG-ENDIAN

Function Mask & Value

Parameters to set up before use

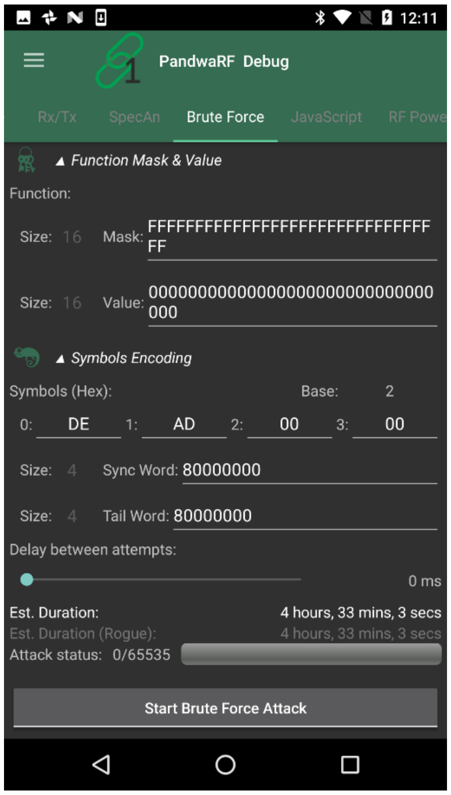
RF parameters

- Frequency in Hz: the range is [300000000, 928000000] Hz
- Data rate: you can go as high as 100000 Bits/s
- Modulation: 2-FSK, GFSK, MSK, ASK, and OOK modulation formats are supported

Codeword Settings

- Length: Number of Symbols. This is the key size that PandwaRF will attack. As the code length increases, the amount of time to find the correct code increases exponentially.

- Frame Repetition: Number of Frames you want to send for each Brute Force attempt (you can adapt this parameter if you want to go faster, but sometimes the receiver needs at least 5 frames for example to recognize the signal).
- Endianness: The byte-order you want to use for transmitted data, generally Big-Endian.



Function Mask & Value

Function:

Size: 16 Mask: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Size: 16 Value: 00000000000000000000000000000000

Symbols Encoding

Symbols (Hex): Base: 2

0: DE 1: AD 2: 00 3: 00

Size: 4 Sync Word: 80000000

Size: 4 Tail Word: 80000000

Delay between attempts: 0 ms

Est. Duration: 4 hours, 33 mins, 3 secs

Est. Duration (Rogue): 4 hours, 33 mins, 3 secs

Attack status: 0/65535

Function Mask & Value

The general logic is: Transmitted data = (data_to_send AND Function_Mask) OR (Function_Value). Note that these are bitwise AND/bitwise OR.

- Function Mask : This is like a Mask IP Address. Every symbol noted FF is brute-forced, and every symbol noted 00 is fixed.
- Function Value : This is the "contrary" of mask. Every symbol noted 00 is not fixed, and every symbol that you want to be fixed needs to be set here.

Symbols Encoding

This is how you encode your signal which is {0x88, 0xEE, 0xE8, 0x8E} so : [0: 88] [1: EE] [2: E8] [3: 8E] The corresponding encoding base is automatically displayed based on the number of symbols (from 2 to 4 symbols). Sync Word (in hex): If you have a synchronization word, blank in this example. Tail Word (in hex) : If you have a tail word, I put 800000 to ensure time between every codeword.

- Symbols: this defines how a **logical** bit (0 or 1) should be converted before transmission into **physical** bits. The code key space is always scanned based on logical data. The corresponding encoding base is automatically displayed based on the number of symbols (from 2 to 4 symbols). You can choose you own values of symbol mapping, eg. how a logical bit will be converted for transmission.
- Synchro word (in hex): Data that needs to be sent **before** each code word.

- Tail Word (in hex): Data that needs to be sent **after** each code word.

Delay Between attempts

This is the delay between each frame you send. Minimum value is 100ms for a regular PandwaRF, 0ms for a PandwaRF Rogue Pro. These delays are approximate.

Start & stop values

Specify the range of possible codes to try. Can be used if you want to restart a previous brute force to where you stopped previously.

Autonomous Brute Force

PandwaRF Rogue Pro is able to continue a brute force even if the Smartphone disconnects from BLE, or the PandwaRF application is sent to background or killed.

Once the Brute Force is started, simply disconnect from the app, or put the app in background mode (pay attention to disable **Keep background connection** option from the **App settings**.

The Brute Force will continue normally until it completes or PandwaRF runs out of battery.

Powering up the PandwaRF again will trigger the Brute Force to continue from where it stopped.

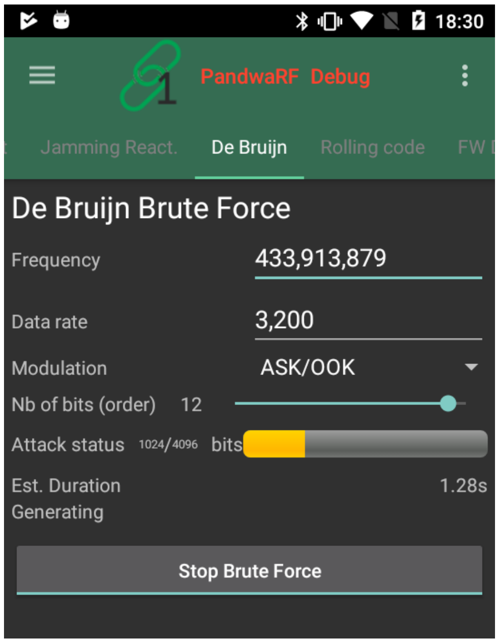
Note: the autonomous brute force is only available while starting a Brute Force from BLE connection.

De Bruijn Brute Force

The De Bruijn sequence is an algorithm used to efficiently produce every possible code in as few bits as possible. It is very effective against old receivers that contain shift registers. Using the De Bruijn mathematical algorithm, PandwaRF Rogue Gov is able to brute force a 12 bit code in 1.2 s instead of a normal brute forced duration of 8mn.

Besides classical RF parameters such as frequency, data rate and modulation, the only parameter needed is the length of the codeword to Brute force.

Old receivers usually use from 8 bit to 12 bit codewords.



De Bruijn Brute Force

Frequency 433,913,879

Data rate 3,200

Modulation ASK/OOK

Nb of bits (order) 12

Attack status 1024/4096 bits

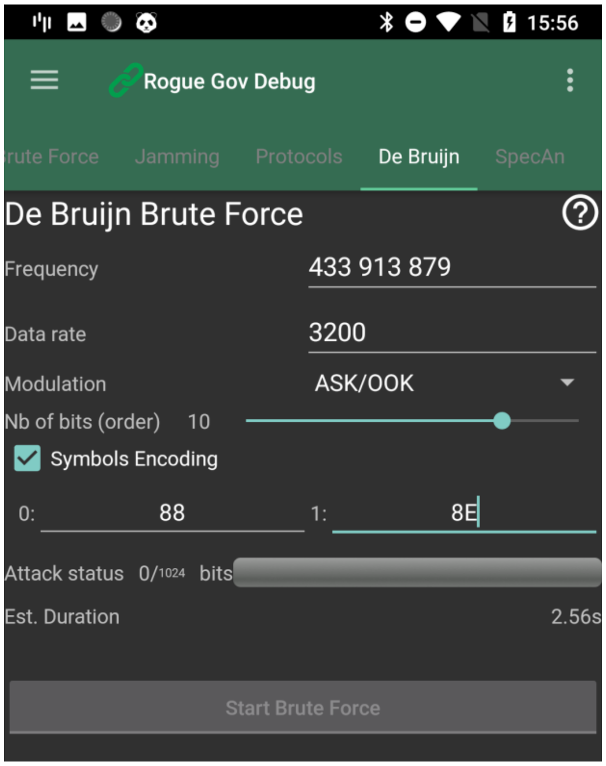
Est. Duration 1.28s

Generating

Note on De Bruijn attack: this attack only works on some old receivers.

Symbol encoding

The De Bruijn Brute Force can also be made using Symbols Encoding. While normal De Bruijn brute force is made using binary symbols, using Symbol Encoding allows the translation of symbols onto a byte instead of a bit.


De Bruijn Brute Force

Frequency 433 913 879

Data rate 3200

Modulation ASK/OOK

Nb of bits (order) 10

☑ Symbols Encoding

0: 88 1: 8E

Attack status 0/1024 bits

Est. Duration 2.56s

Bit encoding (normal De Bruijn)

If bit encoding is used, one De Bruijn symbol is converted into one bit and user cannot change the mapping.

- symbol 0 => bit ‘0’
- symbol 1 => bit ‘1’

The duration of a symbol is the same as the duration of one bit (1/3200 bits/s = 312 µs).

For an order of 10 (codeword of 10 bits) the length of the complete De Bruijn sequence is 2^{10} + (10 -1) = 1033 symbols.

As each symbol is also a bit, the sequence is 1033 bits long, which is 322 ms (1033 * 312 µs).

Symbol encoding (Enhanced De Bruijn)

PandwaRF Rogue has an enhanced mode for De Bruijn attack, which allows using De Bruijn sequence on more recent receivers)

If byte encoding is used, one De Bruijn symbol is converted into eight (8) bits and user can change the mapping.

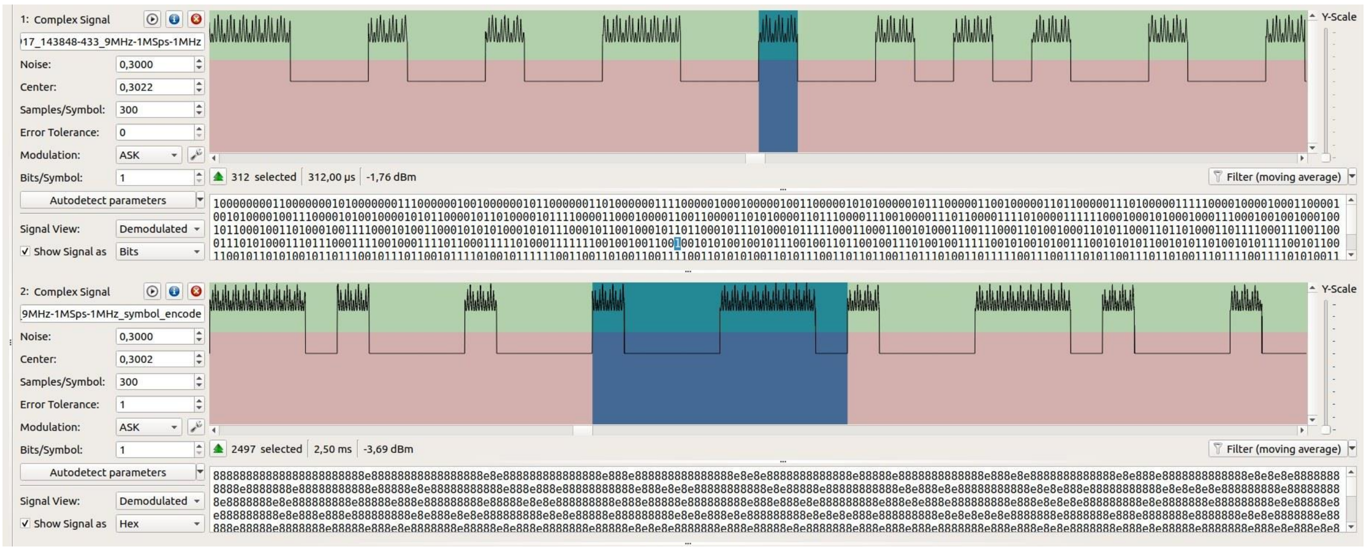
Ex:

- symbol 0 => bit ‘0x88’
- symbol 1 => bit ‘0x8E’

As each symbol is converted to a byte, the sequence of 1033 symbols become 1033 * 8 bits, which is 2.5 s (322 ms * 8).

Note that the duration of a symbol is 8 times the duration of one bit (8 * 1/3200 bits/s = 2.5 ms).

Comparison - normal De Bruijn vs enhanced De Bruijn

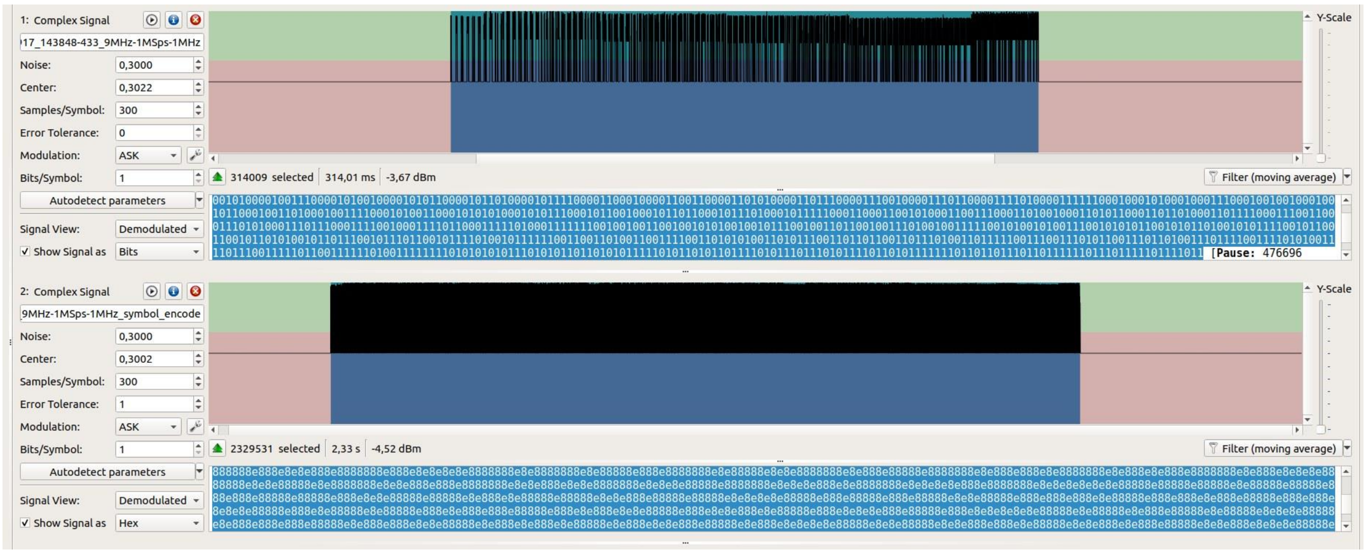


```
1: Complex Signal
17_143848-433_9MHz-1MSps-1MHz
Noise: 0,3000
Center: 0,3022
Samples/Symbol: 300
Error Tolerance: 0
Modulation: ASK
Bits/Symbol: 1
Signal View: Demodulated
Show Signal as: Bits
312 selected | 312,00 µs | -1,76 dBm
1000000001100000000101000000001110000000100100000001011000000011010000000111100000010001000000100110000001010100000010111000000110010000001101100000111010000011111000010000100011000001
001010000100111000010100100001010110000101101000010111110000110001000011001100001101010000110111000011100100001110110000111101000011111100010001010001000111000100100100100100
101100010011010001001111000101001100010101010001010111000101100100010110110001011101000101111110001100011001010001100111000110100100011010110001101101000110111110001110011001100
011101010001110111000111100100011110110001111101000111111100100100110010010101001001011100100110110010011101001001111100101001010011100101010110010101101001010111100101100
110010110101001011011100101110110010111101001011111100110011010011001111001101010100110101110011011011001101110100110111110011100111010110011101101001110111100111101010011
2: Complex Signal
9MHz-1MSps-1MHz_symbol_encode
Noise: 0,3000
Center: 0,3002
Samples/Symbol: 300
Error Tolerance: 1
Modulation: ASK
Bits/Symbol: 1
Signal View: Demodulated
Show Signal as: Hex
2497 selected | 2,50 ms | -3,69 dBm
888888888888888888888888e8888888888888888e8e88888888888888e888e888888888888888e8e8e8888888888888e88888e888888888888e888e8e888888888888e8e888e888888888888e8e8e8e8888888
8888e8888888e888888888e88888e8e8888888888e888e888e8888888888e888e8e8e8888888888e8e88888e8888888888e8e888e8e88888888888e8e8e888e888888888e8e8e8e8e8888888888e88888888
8e8888888e8e88888888e88888e888e888888888e88888e8e8e888888888e888e88888e88888888e888e888e8e888888888e888e8e888e88888888e888e8e8e8e88888888e8e8888888e88888888e8e88888e8
e8888888e8e888e888e88888888e8e888e8e8e888888888e8e8e8888e88888888e8e8e888e8e88888888e8e8e888e88888888e8e8e8e8e8888888e888888e888e888888e888888e8e8e8888888e88
888e88888e8888888e88888e888e8e8888888e88888e8e888e8888888e88888e8e8e8e8888888e888e88888e8e8888888e888e888e888e8888888e888e888e8e8e8888888e888e8e88888e8888888e888e8e888e8e8
```

Figure 1 Duration of De Bruijn symbol. Bit encoding (top) vs symbol encoding (bottom)

Figure 2 shows the duration of the Full De Bruijn sequence in the 2 cases (bit encoding vs symbol encoding).

- On the top part, the normal De Bruijn sequence has a duration of 314 ms (vs 322 ms theory)
- On the bottom part, the Symbol encoded De Bruijn sequence has a duration of 2.33 s (vs 2.5 s theory)

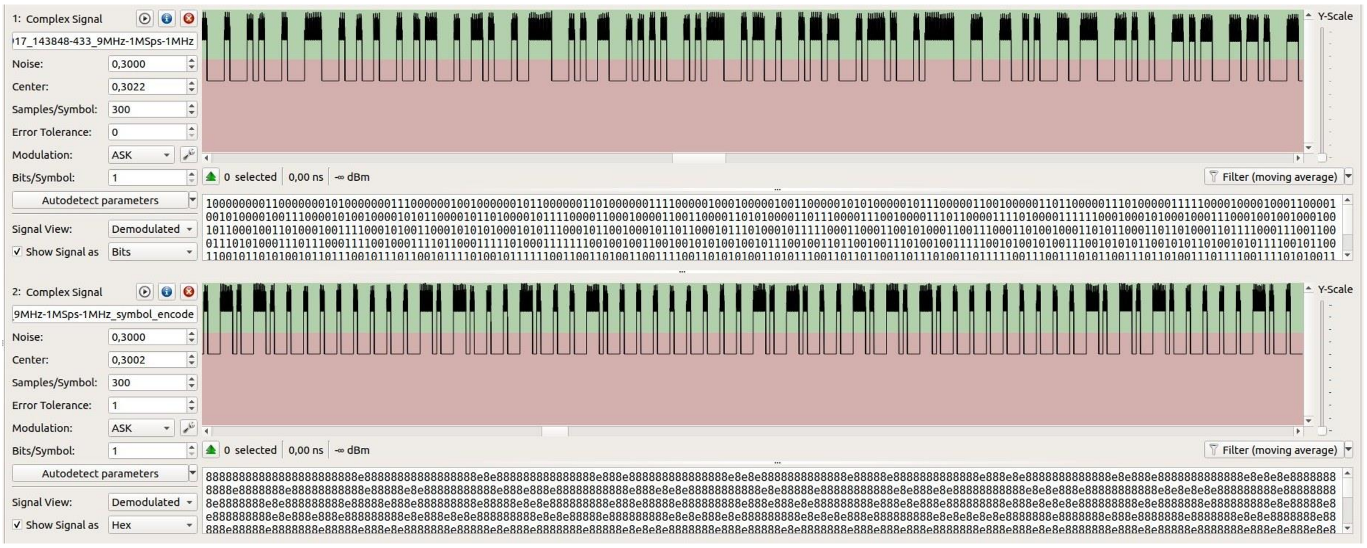


```
1: Complex Signal
17_143848-433_9MHz-1MSps-1MHz
Noise: 0,3000
Center: 0,3022
Samples/Symbol: 300
Error Tolerance: 0
Modulation: ASK
Bits/Symbol: 1
Signal View: Demodulated
Show Signal as: Bits
314009 selected | 314,01 ms | -3,67 dBm
001010000100111000010100100001010110000101101000010111110000110001000011001100001101010000110111000011100100001110110000111101000011111100010001010001000111000100100100100100
101100010011010001001111000101001100010101010001010111000101100100010110110001011101000101111110001100011001010001100111000110100100011010110001101101000110111110001110011001100
011101010001110111000111100100011110110001111101000111111100100100110010010101001001011100100110110010011101001001111100101001010011100101010110010101101001010111100101100
110010110101001011011100101110110010111101001011111100110011010011001111001101010100110101110011011011001101110100110111110011100111010110011101101001110111100111101010011
110111001111011001111110100111111110101010101110101011011010101111101011010110111110101110111010111101101011111110110110111011011111011101111101111011 [Pause: 476696
2: Complex Signal
9MHz-1MSps-1MHz_symbol_encode
Noise: 0,3000
Center: 0,3002
Samples/Symbol: 300
Error Tolerance: 1
Modulation: ASK
Bits/Symbol: 1
Signal View: Demodulated
Show Signal as: Hex
2329531 selected | 2,33 s | -4,52 dBm
88888e888e8e8e888e8888888e888e8e8e8e8e8888888e8e8888888e8e88888e888e8888888e8e88888e8e8e8888888e8e888e88888e8888888e8e888e888e8e8888888e8e888e8e888e8888888e8e888e8e8e8e88
8888e8e8e88888e8e8888888e8e8e888e888e8888888e8e8e888e8e8e8888888e8e8e8e88888e8888888e8e8e8e88888e8888888e8e8e8e8e88888e8e8e8e8e8e88888e8888888e8e8e88888e8e8e88888e8888888e
88e888e88888e88888e888e8e8e88888e88888e8e888e88888e88888e8e8e888e88888e88888e8e8e8e8e88888e888e88888e888e88888e8e8e88888e888e888e888e88888e888e888e888e888e88888e888e888e
8e8e8e88888e888e8e88888e8e88888e888e8e888e888e88888e888e8e888e8e8e88888e888e8e8e888e8e88888e888e8e888e888e88888e888e8e8e8e8e8e88888e8e8e88888e8e8e88888e8e88888e8e8e8e8888
e8e888e888e888e88888e8e888e888e8e8e88888e8e888e8e888e8e88888e8e888e8e8e888e88888e8e888e8e8e8e8e8e88888e8e8e88888e8e8e888e888e8e88888e8e8e888e888e88888e8e8e888e8e8e88888e
```

Figure 2 Duration of De Bruijn sequence. Bit encoding (top) vs symbol encoding (bottom)

Figure 3 shows the content of the De Bruijn sequence in both encodings:

- Bit encoding: the sequence contains only 0/1 bits
- Symbol encoding: the sequence contains only 0x88/0x8E bytes (b10001000/b10001110) bits.


1: Complex Signal
17_143848-433_9MHz-1MSps-1MHz
Noise: 0,3000
Center: 0,3022
Samples/Symbol: 300
Error Tolerance: 0
Modulation: ASK
Bits/Symbol: 1
Autodetect parameters
Signal View: Demodulated
✓ Show Signal as Bits
0 selected | 0,00 ns | -∞ dBm
Filter (moving average)

```
1000000001100000000101000000001110000000100100000001011000000011010000000111100000010001000000100110000001010100000010111000000110010000001101100000011101000000111110000010000100011000 01
00101000010011100001010010000101011000010110100001011110000110001000011001100001101010100001101110000111001000011101100001111010000111111000100010100010001110001001001000100
10110001001101000100111100010100110001010101000101011100010110010001011011000101110100010111110001100011001010001100111000110100100011010110001101101000110111100011100110 0
01110101000111011100011110010001111011000111110100011111110010010011001001010100100101110010011011001001110100100111110010100101001110010101011001010110100101011110010110 0
11001011010100101101110010111011001011110100101111110011001101001100111100110101010011010111001101101100110111010011011111001110011101011001110110100111011110011110101001 1
```

2: Complex Signal
9MHz-1MSps-1MHz_symbol_encode
Noise: 0,3000
Center: 0,3002
Samples/Symbol: 300
Error Tolerance: 1
Modulation: ASK
Bits/Symbol: 1
Autodetect parameters
Signal View: Demodulated
✓ Show Signal as Hex
0 selected | 0,00 ns | -∞ dBm
Filter (moving average)

```
888888888888888888888888e888888888888888888e8e8888888888888888e8888888e888e888888888888888e8e8e8888888888888e88888e8888888888888e888888888888e888e8e8888888888888e8e888e888888888888e8e8e8e8888888
8888e8888888e88888888888e88888e8e88888888888e888e888e8888888888e888888888888e888e8e8e88888888888e8e88888e88888888888e8e888e8e88888888e8e8888888888e8e8e888e88888888888e8e8e8e8e888888888e88888888
8e8888888e8e888888888e88888e888e888888888e88888e8e8e888888888e888888888e888e88888e888888888e888e888e8e888888888e888e8e888e8888888888e888888888e888e8e8e8e888888888e8e8888888e888888888e8e88888e8
e88888888e8e888e888e888888888e8e888e8e8e88888888e8e8e88888e88888888e8e88888e888888888e8e888e8e888888888e8e8e888e88888888e8e8e8e8e8e8e8e8888888e8888888e888e8888888e8888888e8e8e8888888e88
888e88888e8888888e88888e888e8e8888888e88888e8e888e8888888e888888e88888e8e8e8e8888888e888e88888e8e8888888e888e888e8888888e88e8888888e888e888e8e8e8888888e888e8e88888e8888888e888e8e888e8e8
```

Figure 3 Inside a De Bruijn sequence. Bit encoding (top) vs symbol encoding (bottom)

Of course in the example above, encoding onto bytes ‘88’ and ‘8E’ is arbitrary and can be changed by user.

For more information about the De Bruijn attack and vulnerable devices, refer to http://samy.pl/opensesame/

Android USB Connection

We have designed PandwaRF to be connected to an Android smartphone either using Bluetooth Smart or a USB cable.

Connecting in USB gives you more throughput than in BLE connection. This is clearly visible in RX capture and Spectrum Analyzer mode.

But if you connect in USB, you **cannot** access some PandwaRF services that are only available in BLE. List of features not available in USB mode:

- Bus Service:
    - Device information
    - Firmware information
    - Battery measurement
- FW update:
    - Nordic FW update
    - CC1111 FW update
- Self-Test
- Bus Service Extended

- BLE throughput measurement
- BLE parameters
- BLE errors

To connect to PandwaRF using USB, your phone needs to have the USB Host feature.

Note also that the PandwaRF Android app can connect to:

- PandwaRF. This is the preferred device as it gives access to all features: RX/TX/BF/SpecAn/DRM
- Yard Stick One. Limited features: RX/TX

Settings

You can control some default behavior of the application.

Bluetooth Connectivity

Auto scan devices when app starts

- Automatic BLE scan when application is launched or resumed.
- Do not start automatic BLE scan when application is launched or resumed. Manual scan only.

Background connection

Close the PandwaRF BLE connection when the app is in background.

Auto reconnect when app starts

Automatically reconnect to the last successfully connected PandwaRF when the application is launched.

Auto reconnect when app resumes

Automatically reconnect to the last successfully connected PandwaRF when the application is resumed from background and becomes visible.

Auto reconnect on BLE error

Automatically attempt a BLE reconnection to the last successfully connected PandwaRF when there is an unexpected BLE disconnection (GATT error, reset, tsunami...)

Auto bonding

Automatically bond PandwaRF upon connection.

Network Connectivity

Error reporting

Report PandwaRF Nordic FW errors. No personal data is sent.

Note: you cannot change this setting.

Radio settings

RF power amplifiers check

- Automatically check for correct RF power amplifiers prior to TX/RX RF action.
- Do not check RF power amplifiers settings prior to TX/RX RF action. User must check the correct setting of RF power amplifiers.

FSK Deviation

Display all supported values of FSK deviation, or only display commonly used values.

Features

Developer Mode

- Only the basic features: Scan/RX/TX/SpecAn/Brute Force.
- Enable experimental and dangerous features: BLE throughput measurement, CC1111 RF registers access, BLE errors, Self-test, BUS service, BLE Parameters, Log.

Display

Tab names in view pager

- Do not display page name in the tab located on top of the screen. Saves space on small screens.
- Display page name in the tab located on top of the screen. This is redundant with menu drawer, so uncheck this option if you have a small screen.

Display tab names in alternate view pager (Tablet only)

- Do not display page name in the tab located on top right of the screen. Saves space on small screens.
- Display page name in the tab located on top right of the screen. Uncheck this option if you have a small screen.

Force split mode

- One page fits the entire screen.
- Split UI in 2 parts: left page with Core features, right page with settings. Uncheck this option if you have a small screen. Changes will take effect only after restarting the app.

Changes will take effect only after restarting the app.

Tip of the Day

- Do not show Tips on Startup.
- Show Tips on Startup.

MAC address hiding

- Display all MAC addresses completely.
- Hide lower bytes of all MAC addresses displayed in the app (for privacy).

Tweaking

NPI timeout

Define the timeout policy when communicating with PandwaRF over BLE/USB

BLE TX enqueue mode

- Normal mode: Enqueue all TX packets into a FIFO queue. Faster, but can freeze sometimes.

- Fallback mode: Send one BLE packet at a time, waiting for the previous packet to be sent before queuing the new one. Safer but slower. Use this mode if you experience issues with the BLE connection.

Changes will be effective at the next connection.

Periodic RSSI measurement of the connected PandwaRF

- RSSI is measured every 1s. Can cause disconnect issues on some phones (Samsung Galaxy S5, …)
- RSSI is not measured when PandwaRF is connected.

Reset

Clear user input history

Clear all user input data. Eg. frequency, data rate.

Reset settings

This will reset the application settings to default values. All preferences regarding showing or hiding pop-up dialogs are cleared, and all pop-up dialogs will now be displayed until user's choice.

More Information

1. PandwaRF website (https://pandwarf.com/)
2. Wiki (https://github.com/ComThings/PandwaRF/wiki)
3. Chat (https://gitter.im/ComThings/Lobby)
4. Forum (http://pandwarf.boards.net/)
5. Demo videos (https://www.youtube.com/c/comthings/)

Note: You can find the solution to the most common issues in our wiki. Make sure to also check the PandwaRF forum and our Gitter chat room. If you still have the issue after doing so, please report it using our tracking system (https://github.com/ComThings/PandwaRF/issues)**.**

Still have questions? Feel free to contact us at pandwarf@comthings.com.

Happy hacking! :)

Document Revision History

Revision	Date	Status and Description
0.1	2019-03-19	Initial version.
0.2	2019-06-04	
0.3	2019-10-16	
0.4	2020-09-10	
0.5	2020-09-17	Add Autonomous Brute Force Add De Bruijn Brute Force