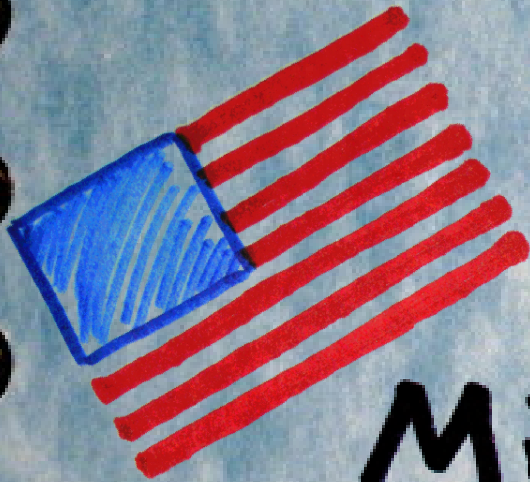


Building an All-Channel Bluetooth Monitor



Michael Ossmann
and
Dominic Spill

I'm Mike



Institute for
Telecommunication Sciences

mike@ossmann.com

I'm Dominic



University College London
Imperial College London
dspill@doc.ic.ac.uk

Certain commercial equipment, materials, and software are identified to specify technical aspects of the reported procedures and results. In no case does such identification imply recommendations or endorsement by the U.S. Government, its departments, or its agencies; nor does it imply that the equipment, materials, and software identified are the best available for this purpose.

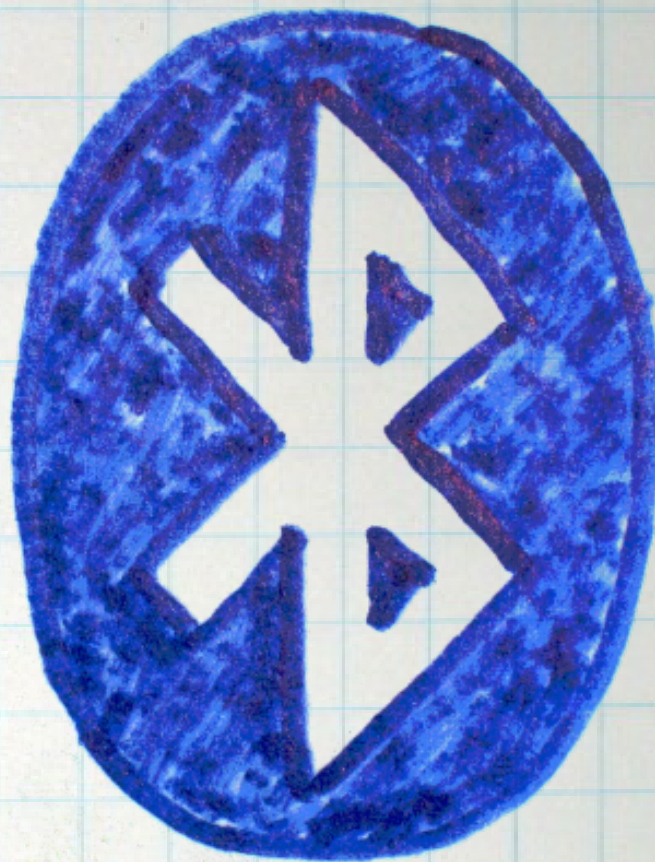
Sniffing
Bluetooth
is
Hard

Bluetooth

PAN

2.4 GHz

Frequency
Hopping



79 MHz

channel

frequency

79
channels

0

2402 MHz

1

2403

⋮

⋮

78

2480

1 MHz
wide

1600 hops per second

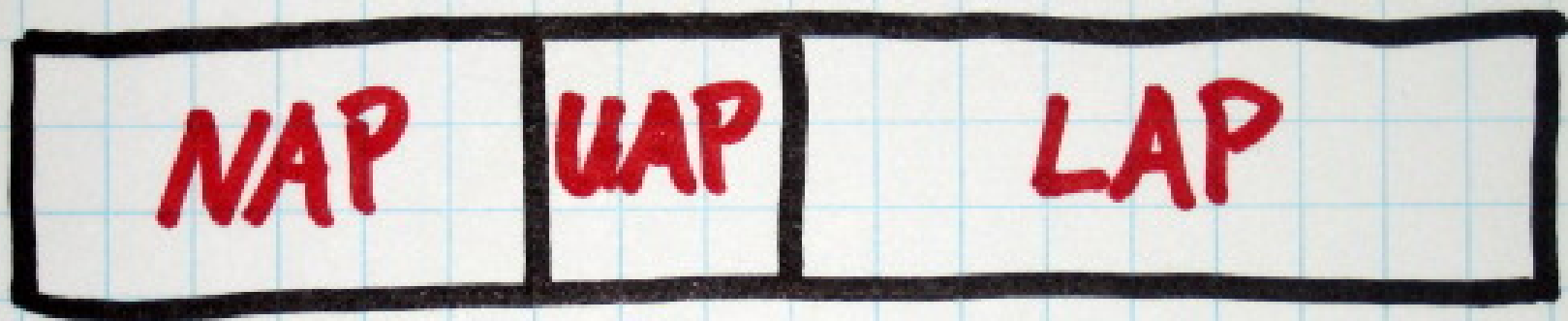
clocks

increment 3200 times
second

hops every other clock

master device provides
clock for piconet

BD-ADDR



16
bits

8
bits

24
bits

company ID

sniffing hardware

protocol analyzer

\$10,000

Bluetooth ODM

\$10

USRP

\$1000

protocol analyzer

test equipment
for Bluetooth
manufacturers

expensive
or
illegal

doesn't do
what we want

Bluetooth

OEM

flashable
chipset

custom
firmware

one
channel

hardware
implementation

Software radio

Wideband receiver

USRP 8 MHz

USRP2 25 MHz

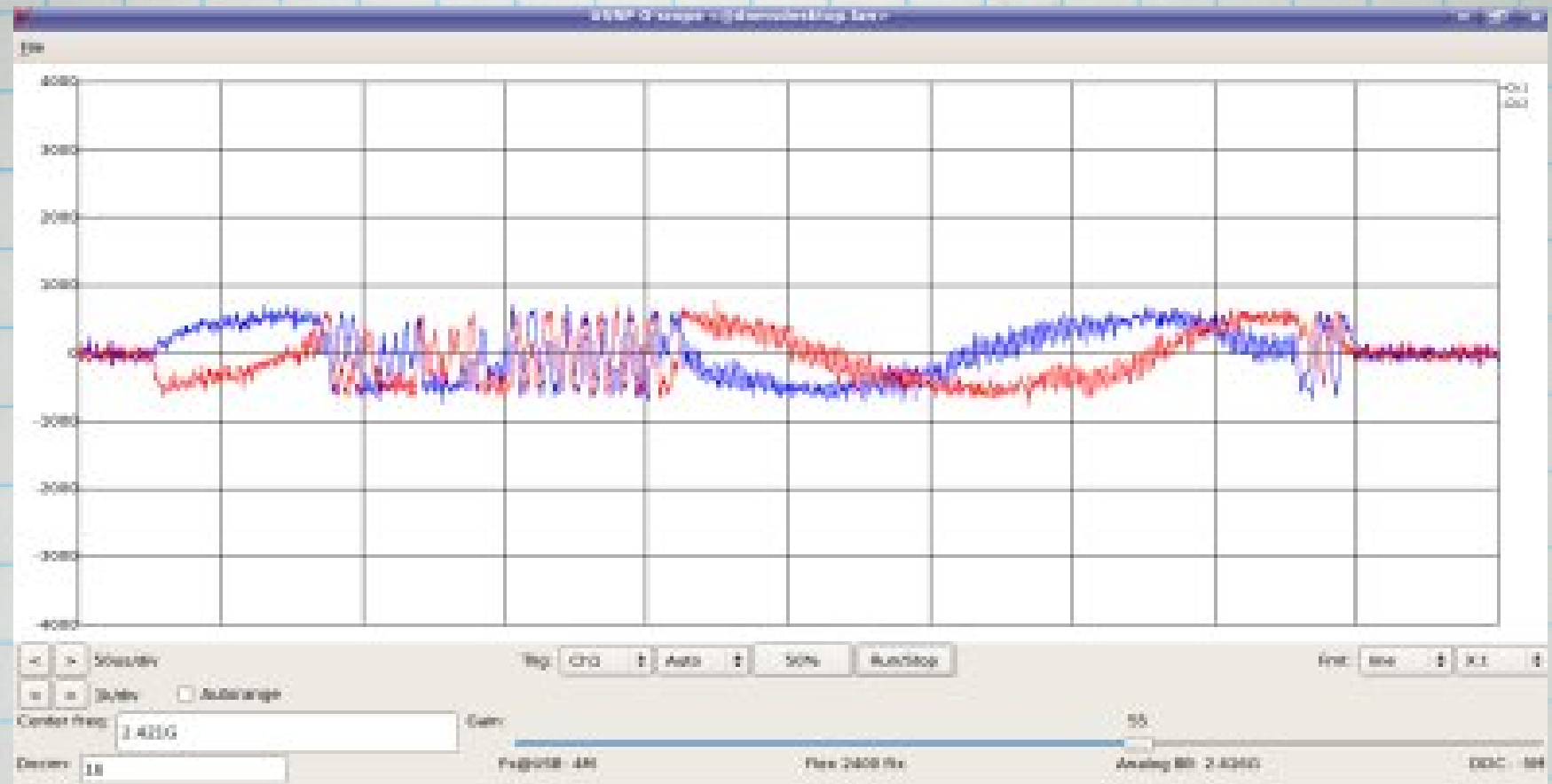
GNU
Radio

software signal processing
we can do what we want

this has potential...

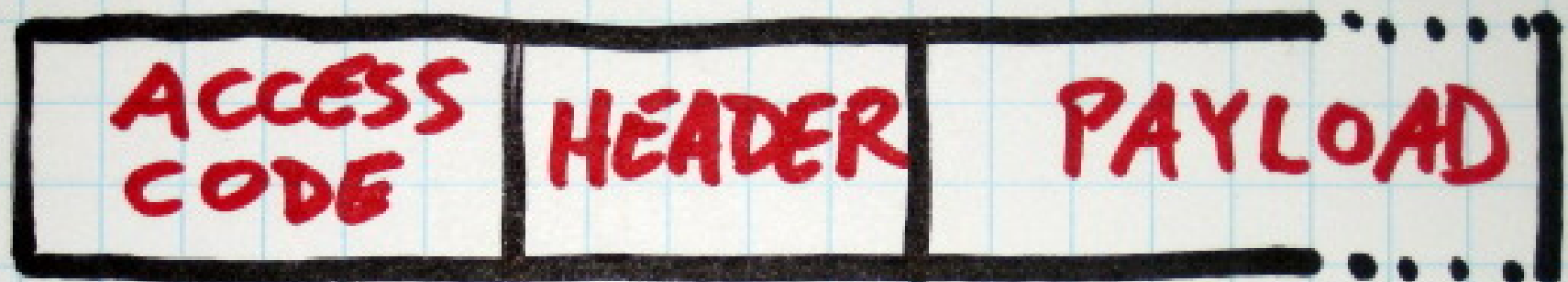
Sniffing
a
Single
channel

how?



demodulate in software

WOO Packets!



72
bits

54
bits

0-2745
bits

includes LAP

demonstration

finding
packets

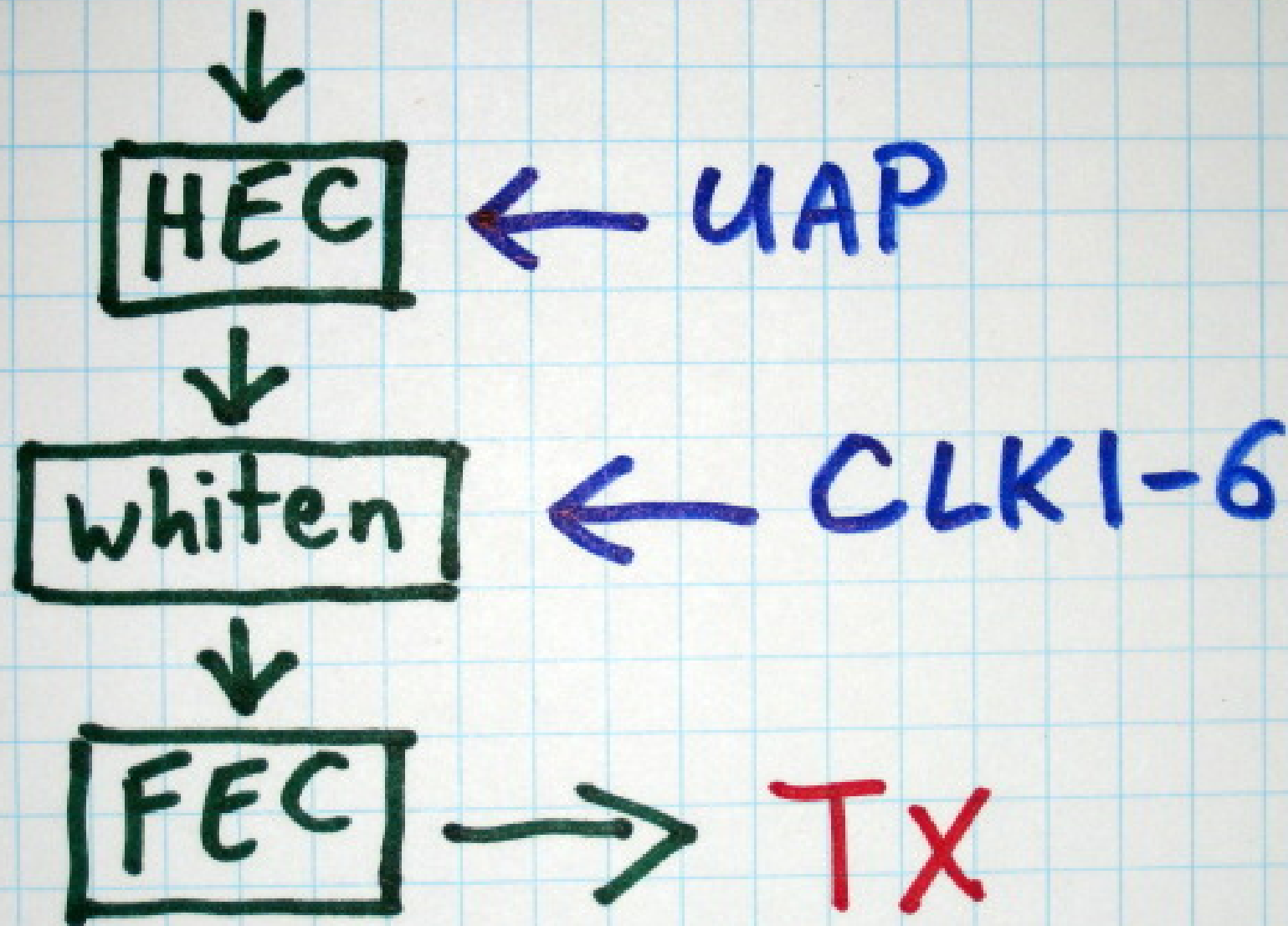
UAP discovery

not as simple
as LAP

not
transmitted
in the
clear

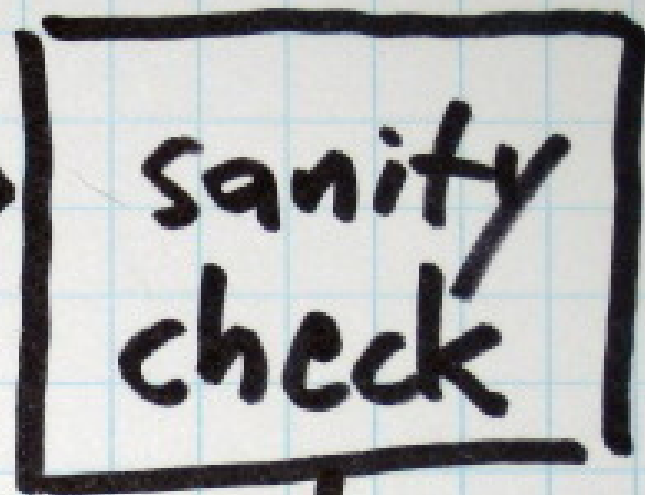
used for
error
checking

header processing



UAP retrieval

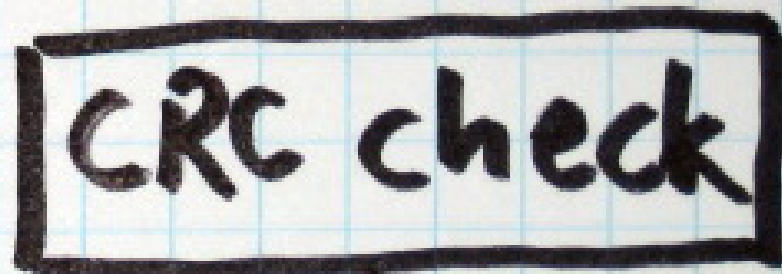
64 candidate packets



sanity
check



candidate
UAP



CRS check



single correct UAP

demonstration

retrieving
the
UAP

payload data

got LAP and UAP?

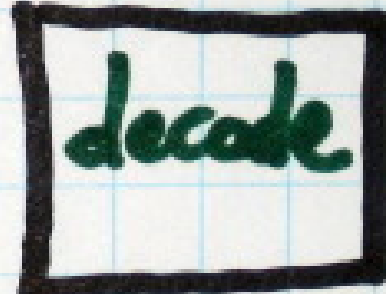
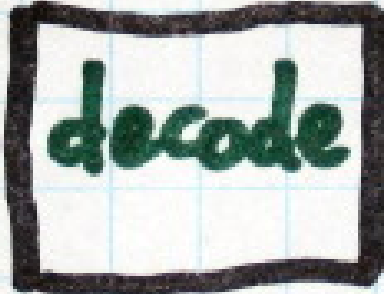
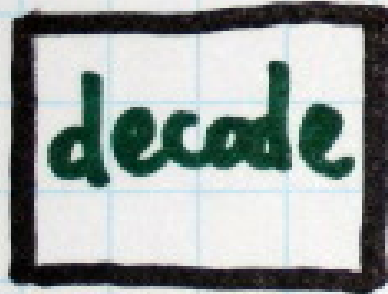
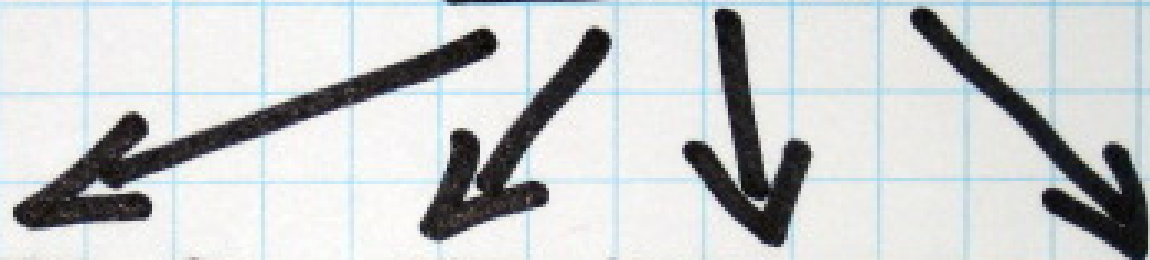
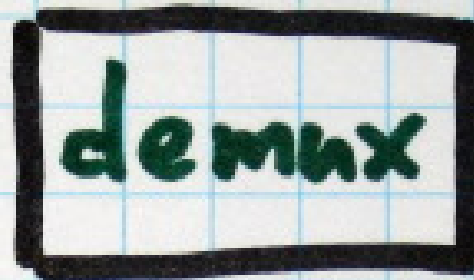
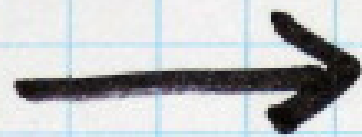
now we can:

- sniff packets
- verify checksums
- discover CLK1-6

Extending
to
Several
Channels

one input, several outputs

RX



...

channel 1

channel 2

channel 3

...

how many channels?

USRP

8 channels



USRP2

25 channels



79 Bluetooth channels

demonstration

finding packets
on multiple
channels

all 79 channels?

10 USRPs

or

4 USRP2s



CPU requirements

1 CPU \approx 1 channel

bus speeds,
storage,
etc.

Predicting
the
Hopping
Pattern

why?

sniffing all channels
is too costly

active
attacks

it's fun!

hopping sequence

UAP
0x65

clock

channel

0

20

2

60

4

53

6

62

⋮

⋮

LAP
0x87CBA9

how?

use algorithm in spec
requires LAP, UAP, clock
generate entire pattern
for all clock values (~24hr)

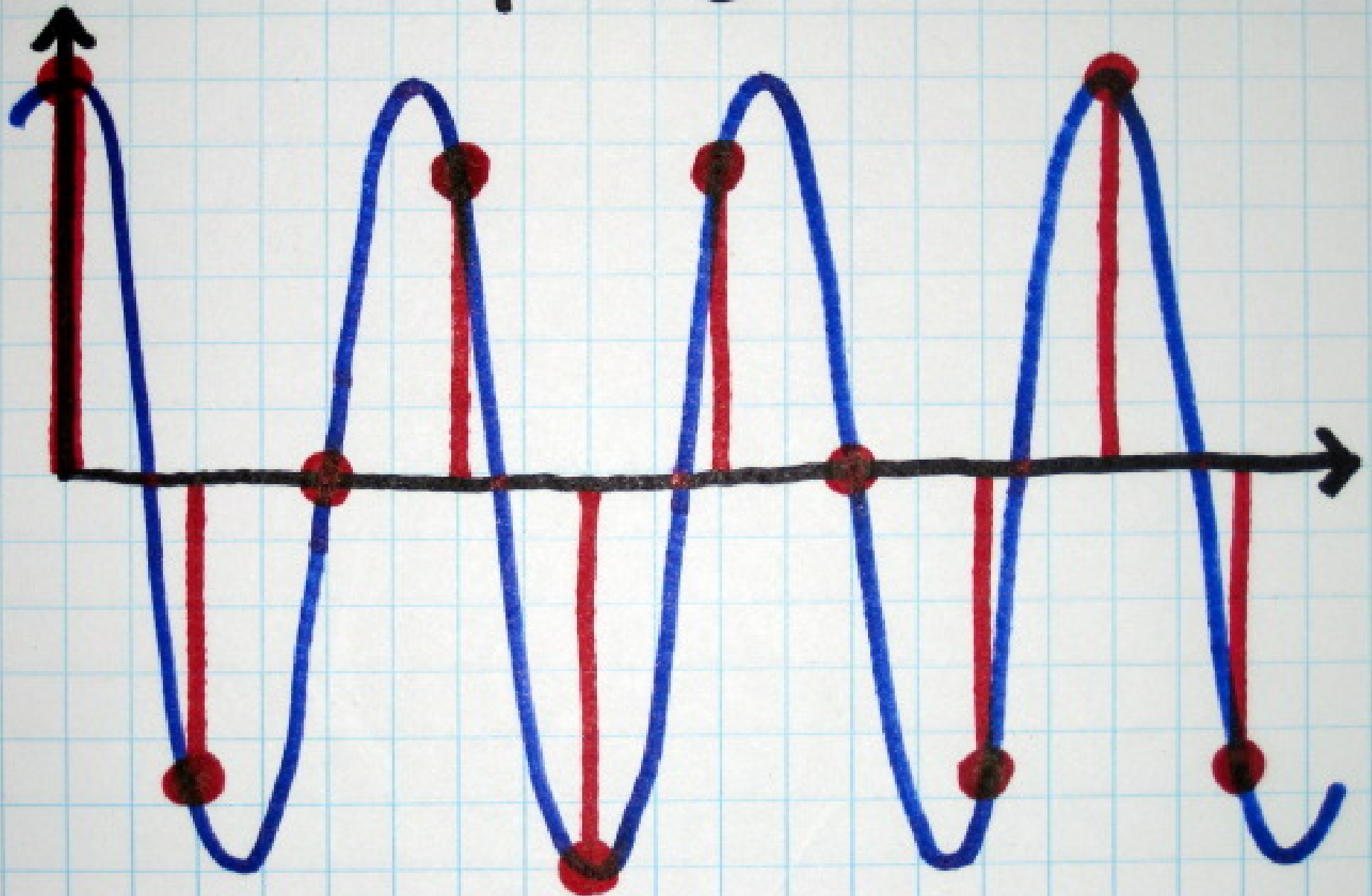
search the pattern
using observed hops

demonstration

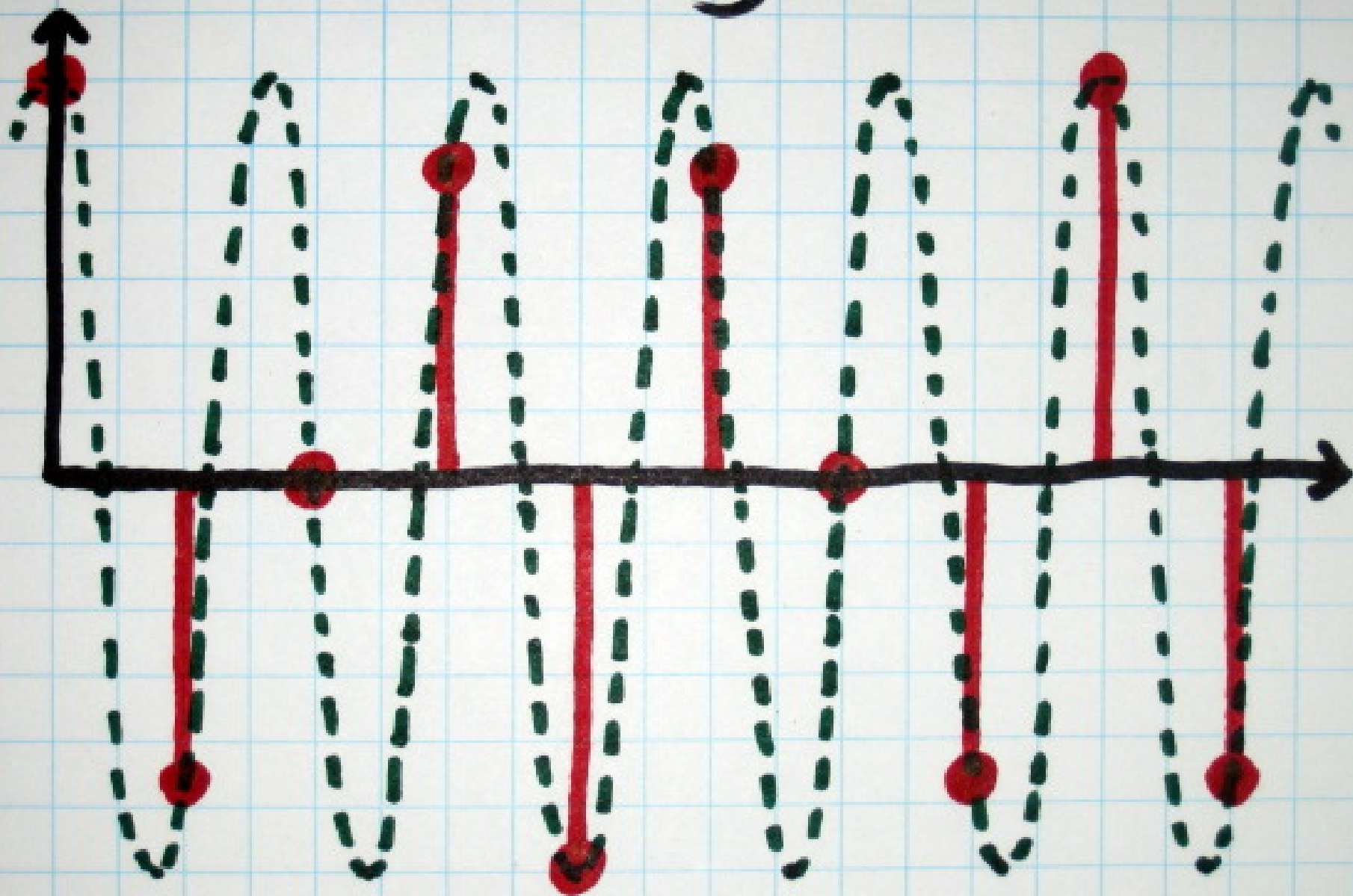
finding
the clock

Intentional
Aliasing

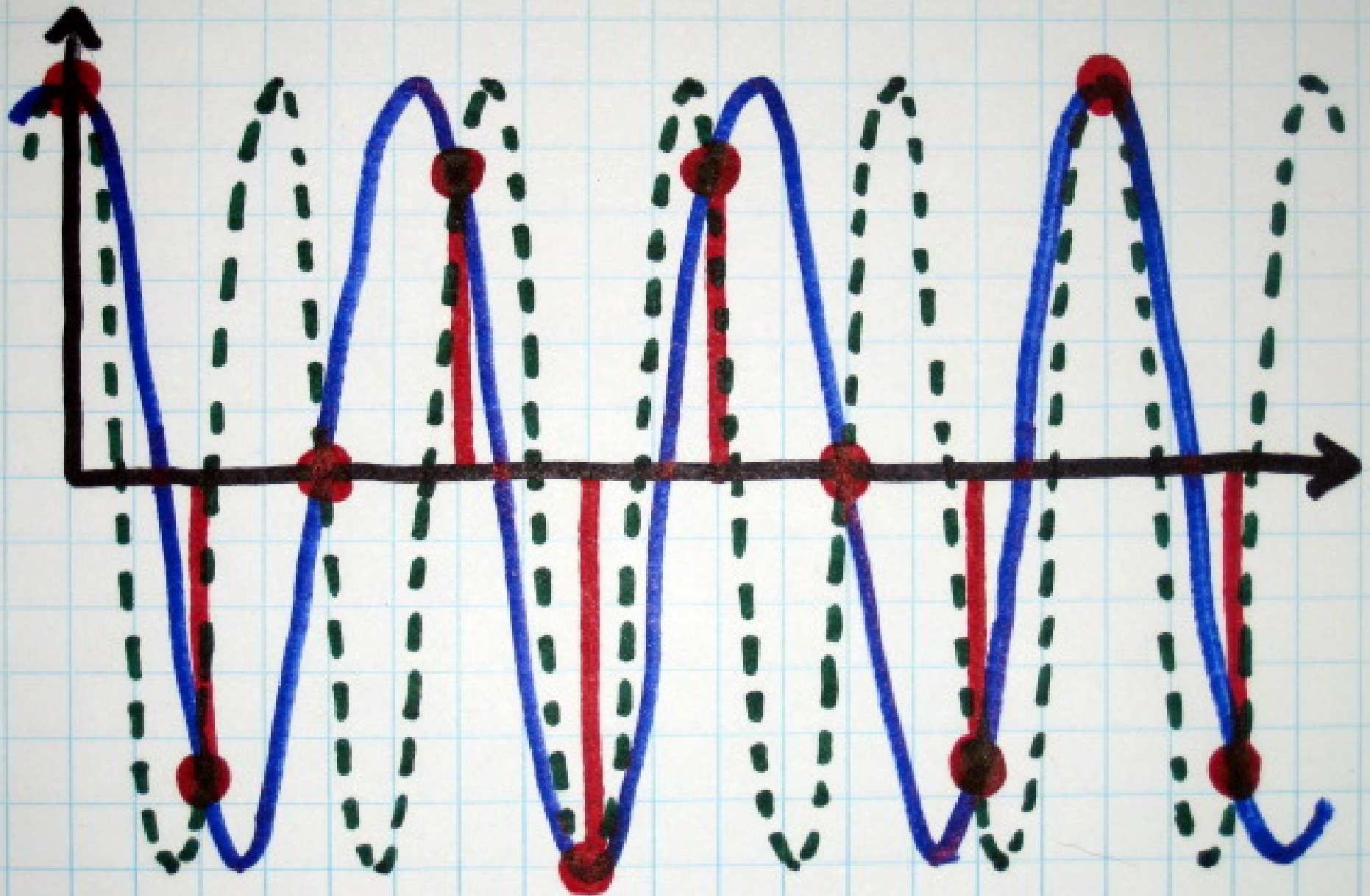
sampling 3 MHz



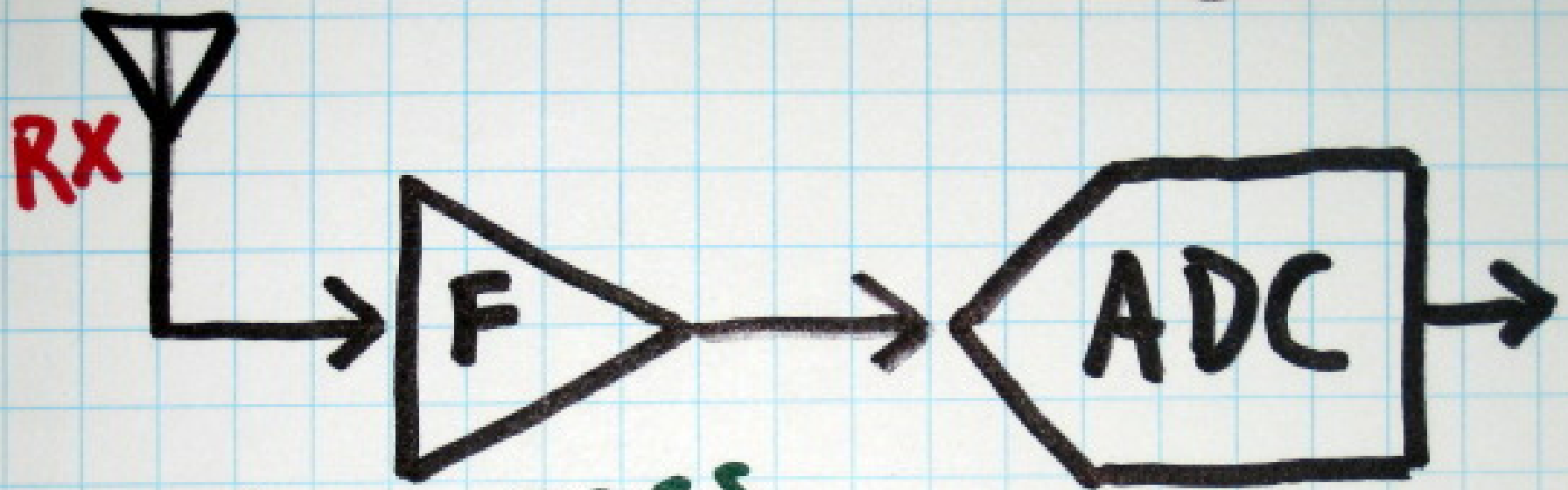
sampling 5 MHz



aliases



anti-aliasing



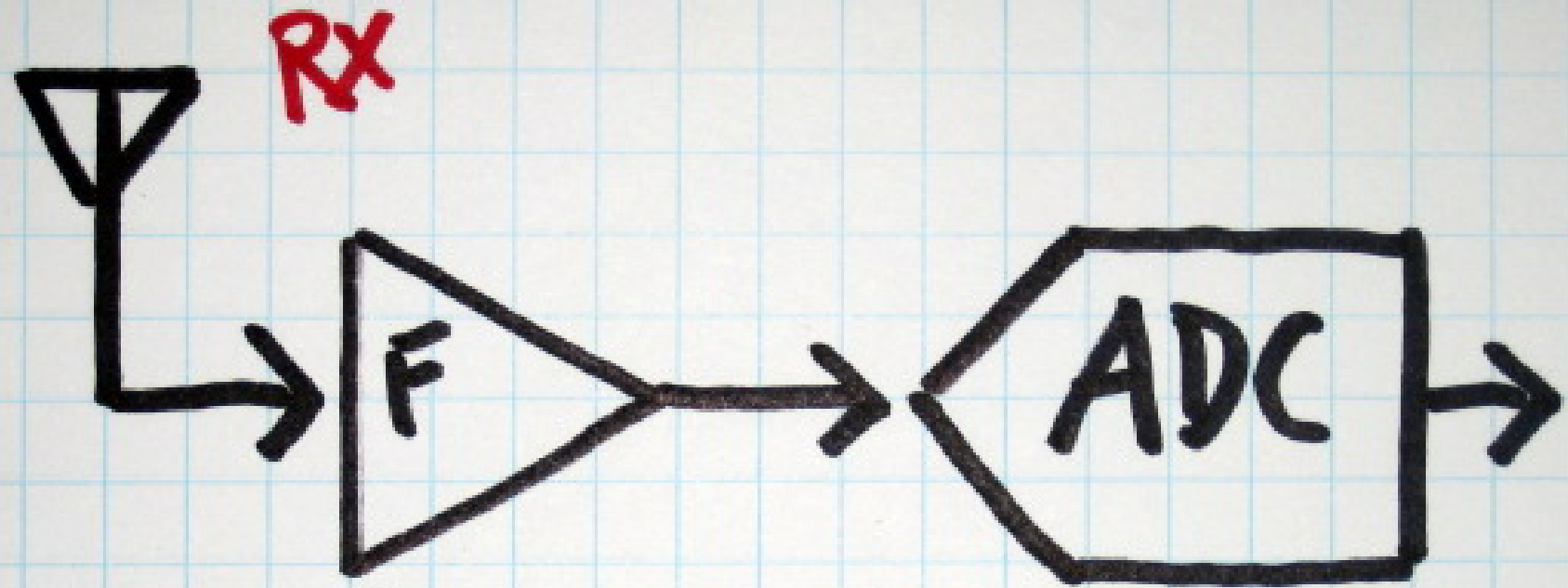
low-pass
filter

3 MHz

~~5 MHz~~

~~11 MHz~~

band-pass sampling



band-pass
filter

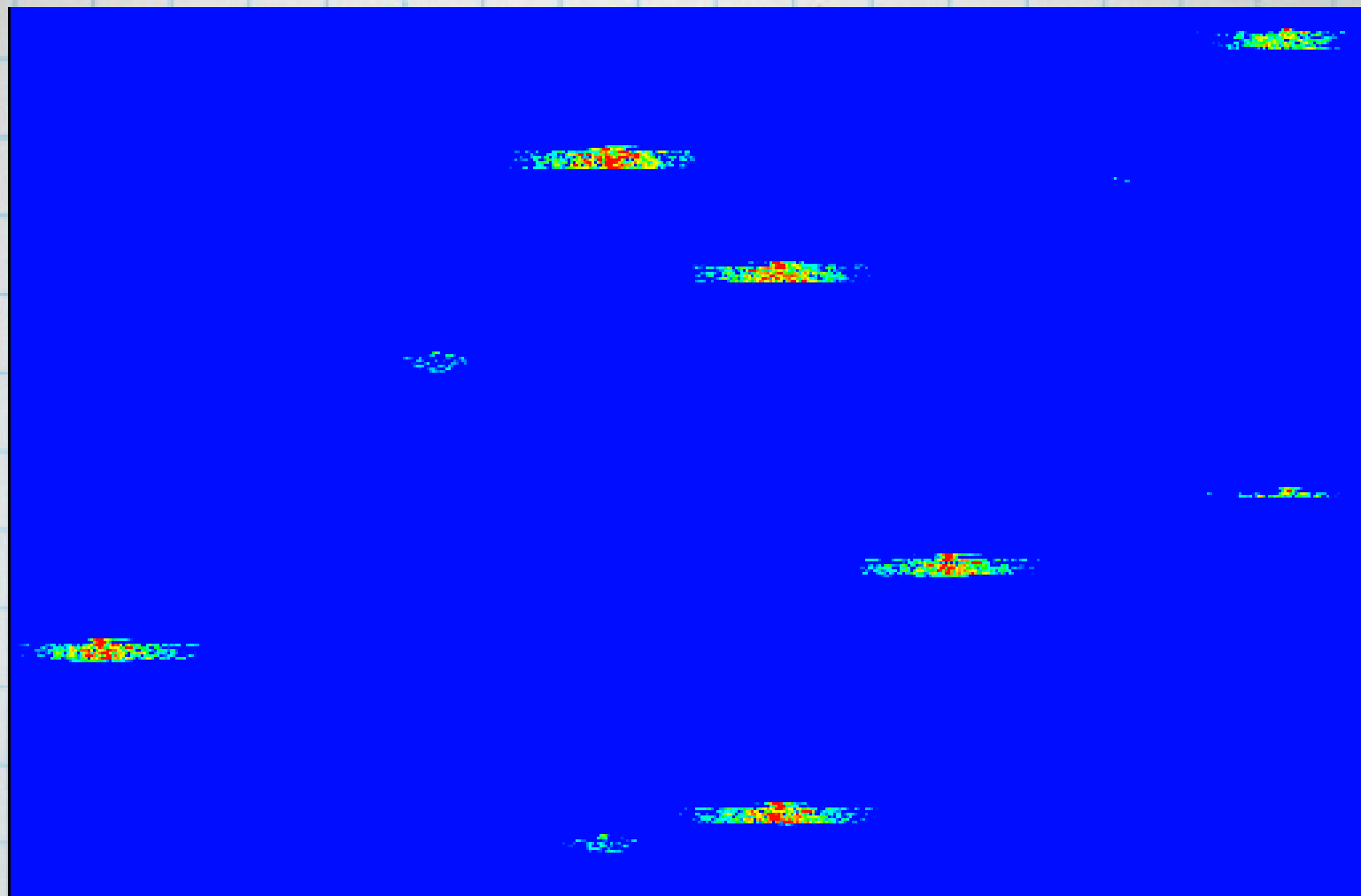
~~3 MHz~~

5 MHz

~~11 MHz~~

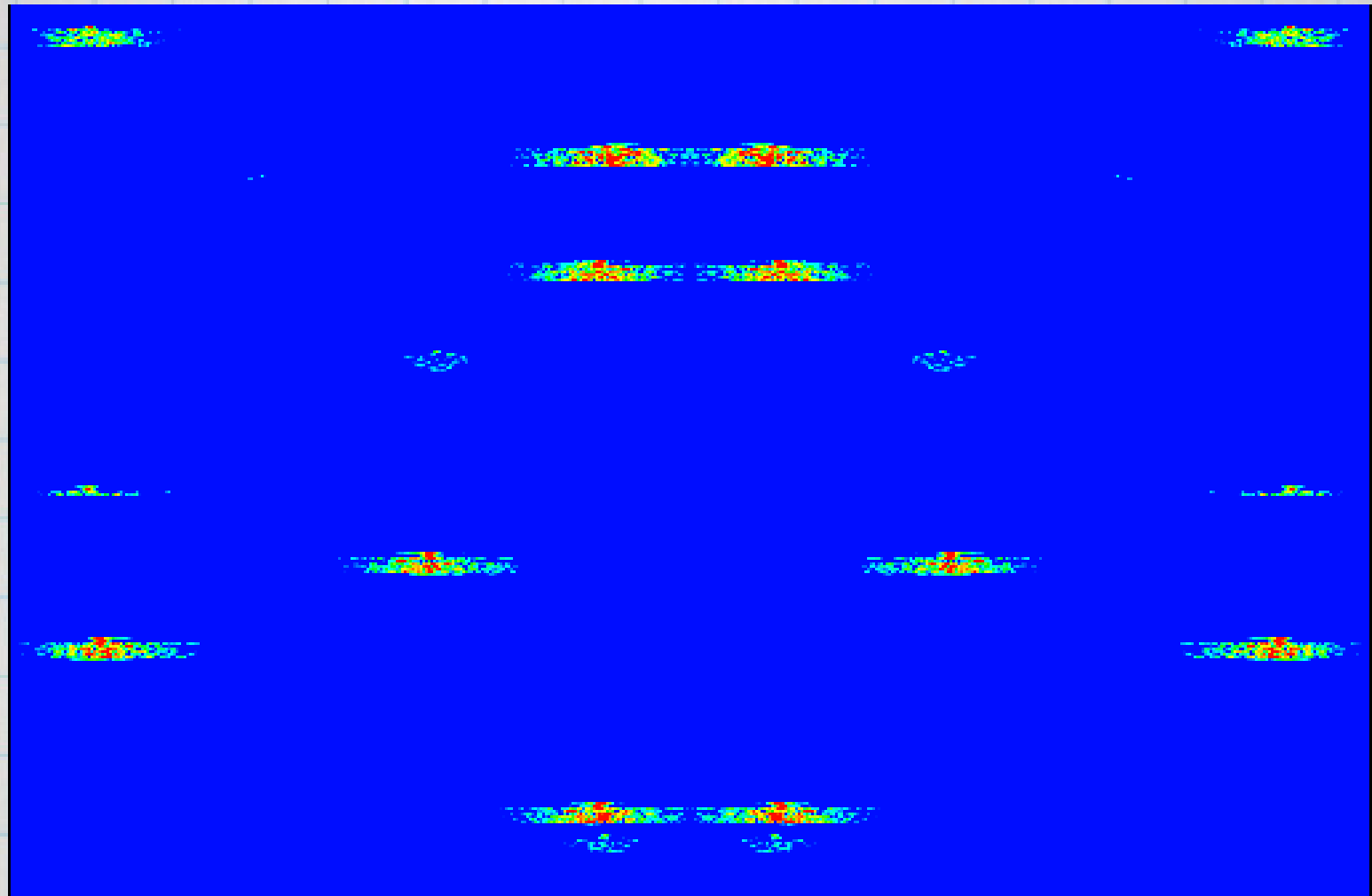
frequency hopping

time
↓



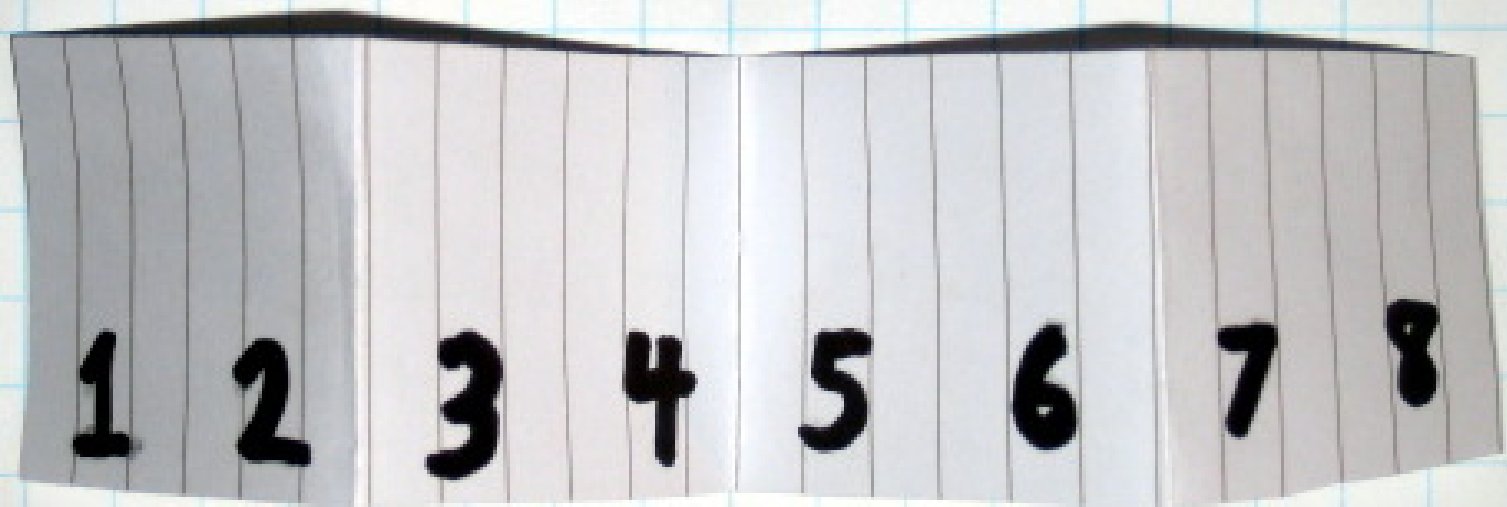
frequency →

aliased frequency hopping



frequency →

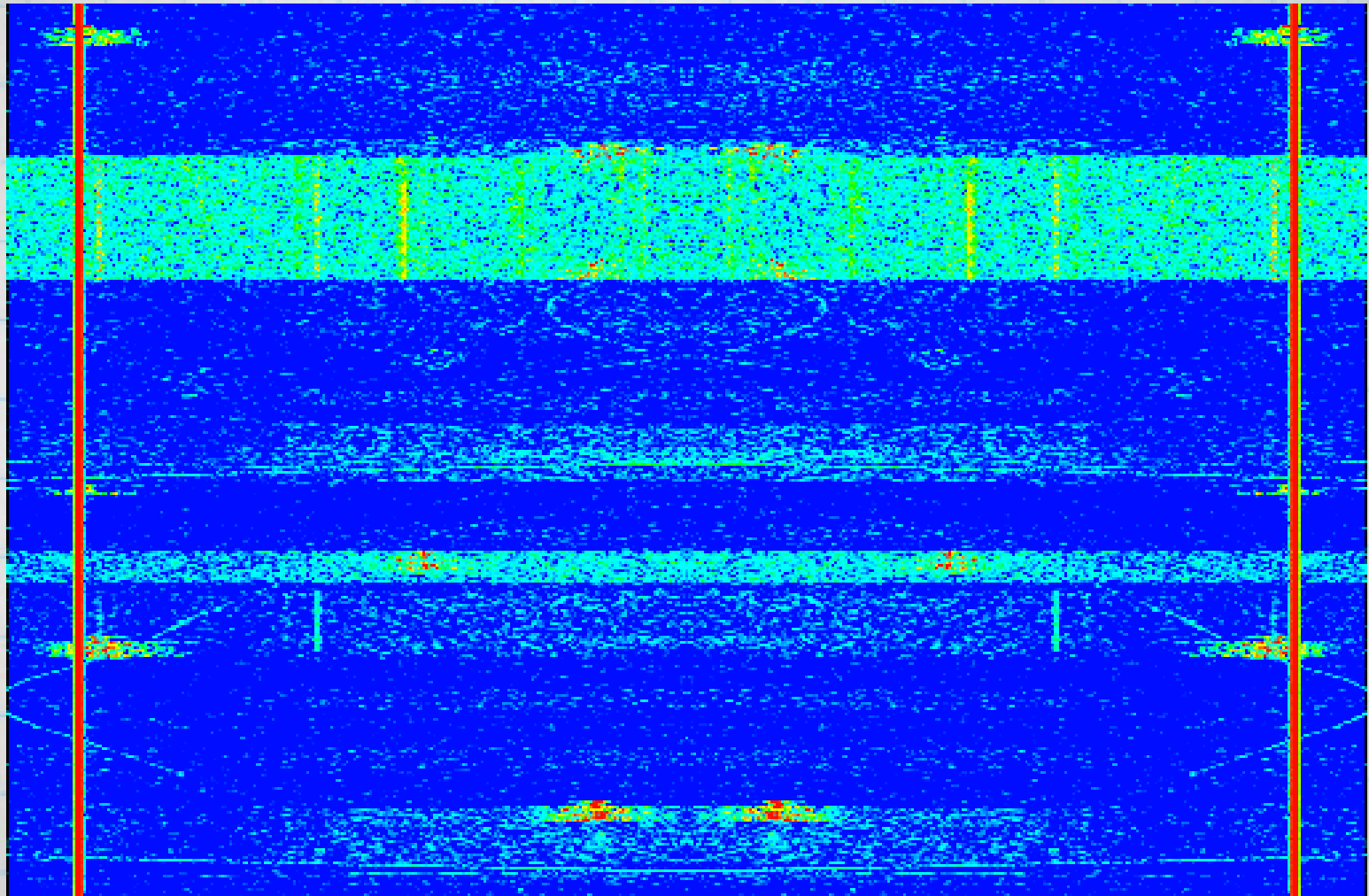
more aliases



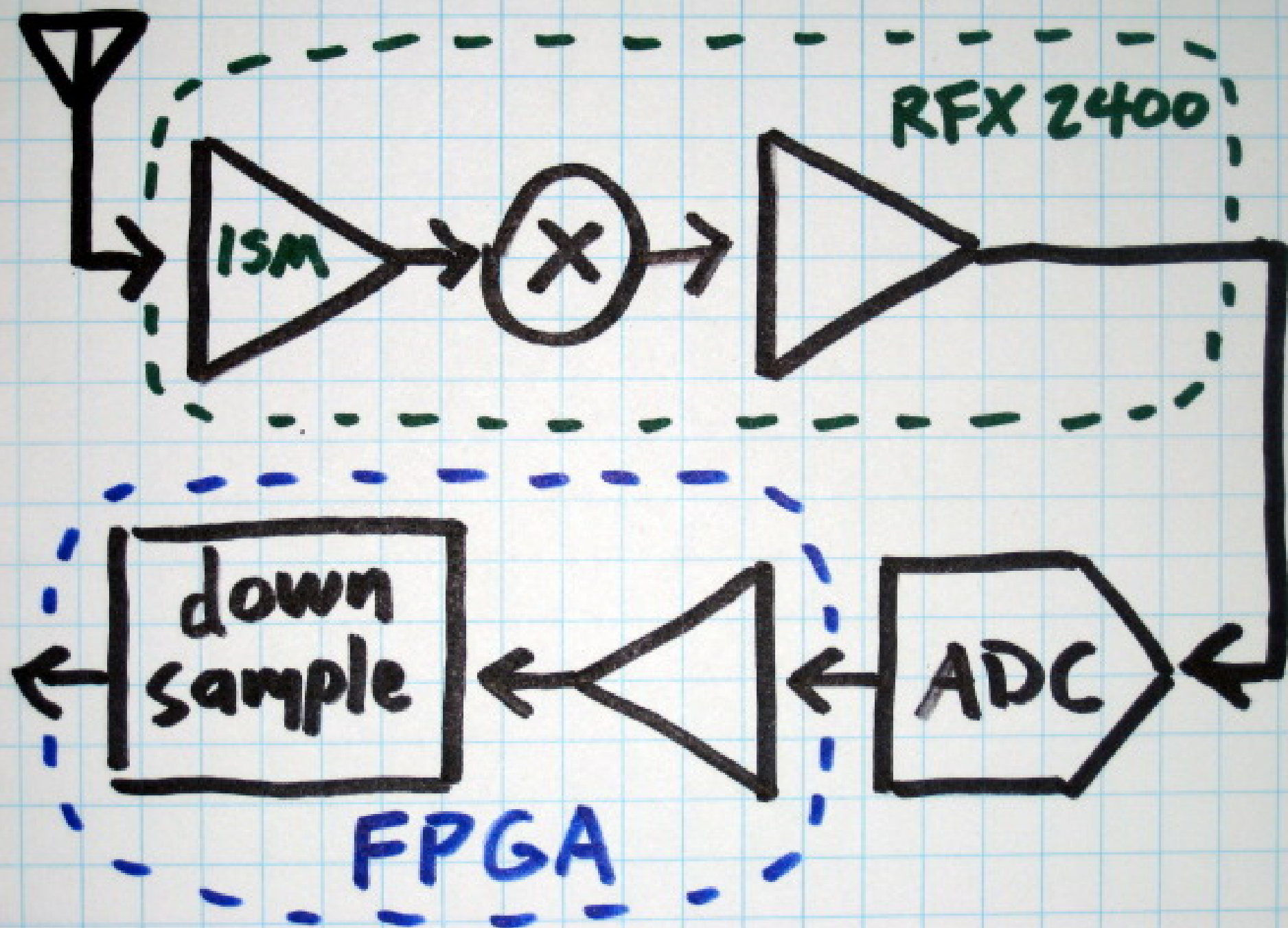
more aliases



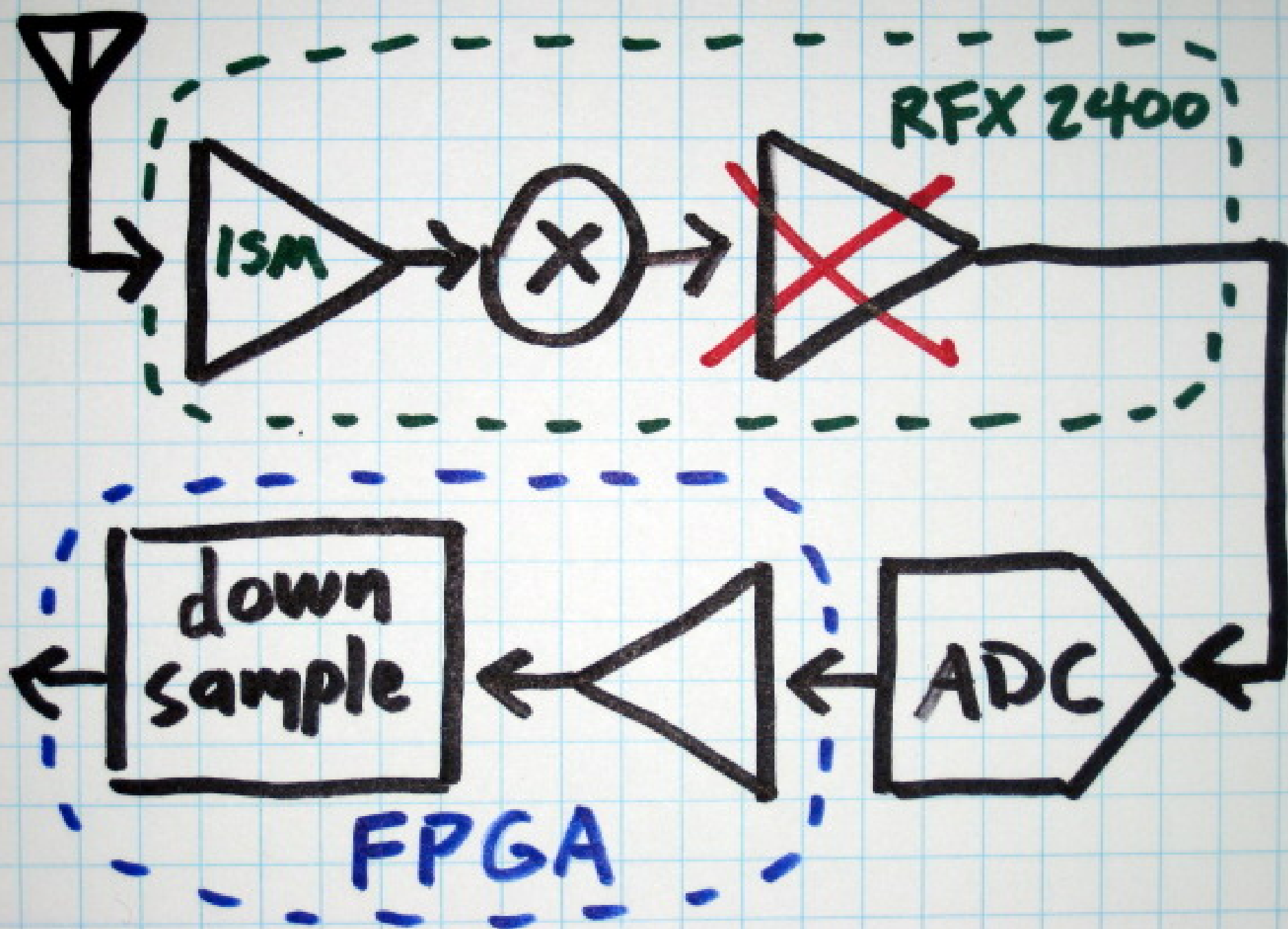
the good, the bad, and
the ugly



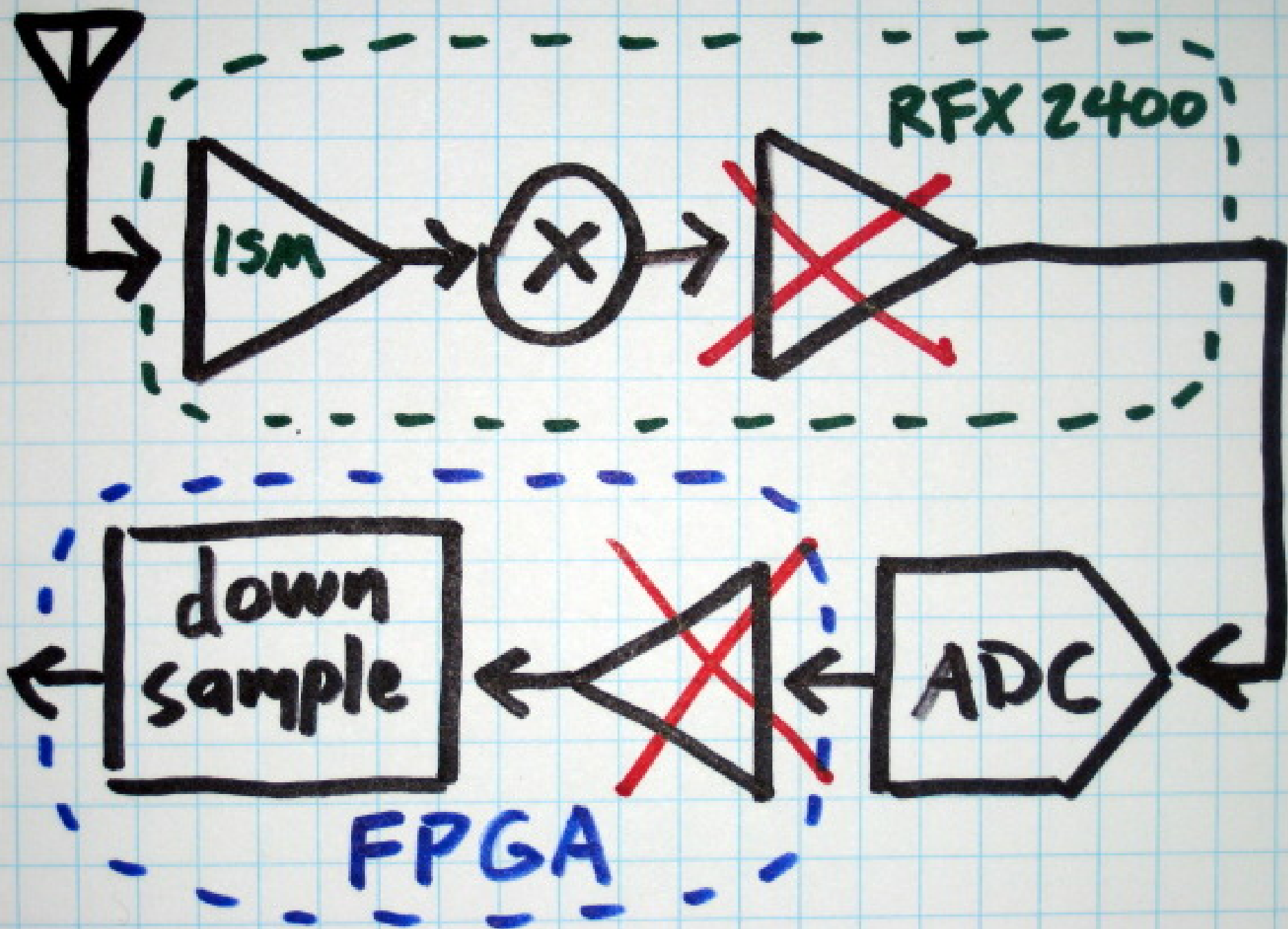
Our method



Our method



Our method



demonstration

aliased
hopping

security implications

non-discoverable == discoverable

active attacks

got encryption?

Q&A

<http://gr-bluetooth.sf.net/>





Building an All-Channel Bluetooth Monitor

Michael Ossmann & Dominic Spill



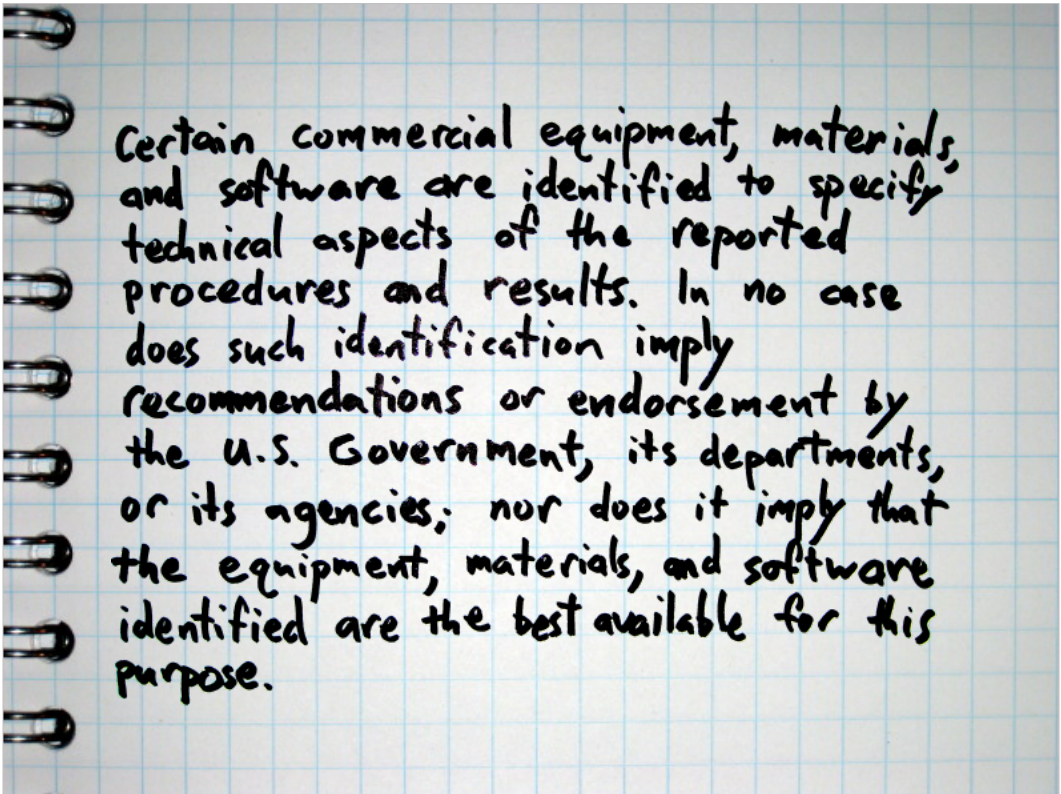
I'm Mike

Institute for Telecommunication Sciences
mike@ossmann.com



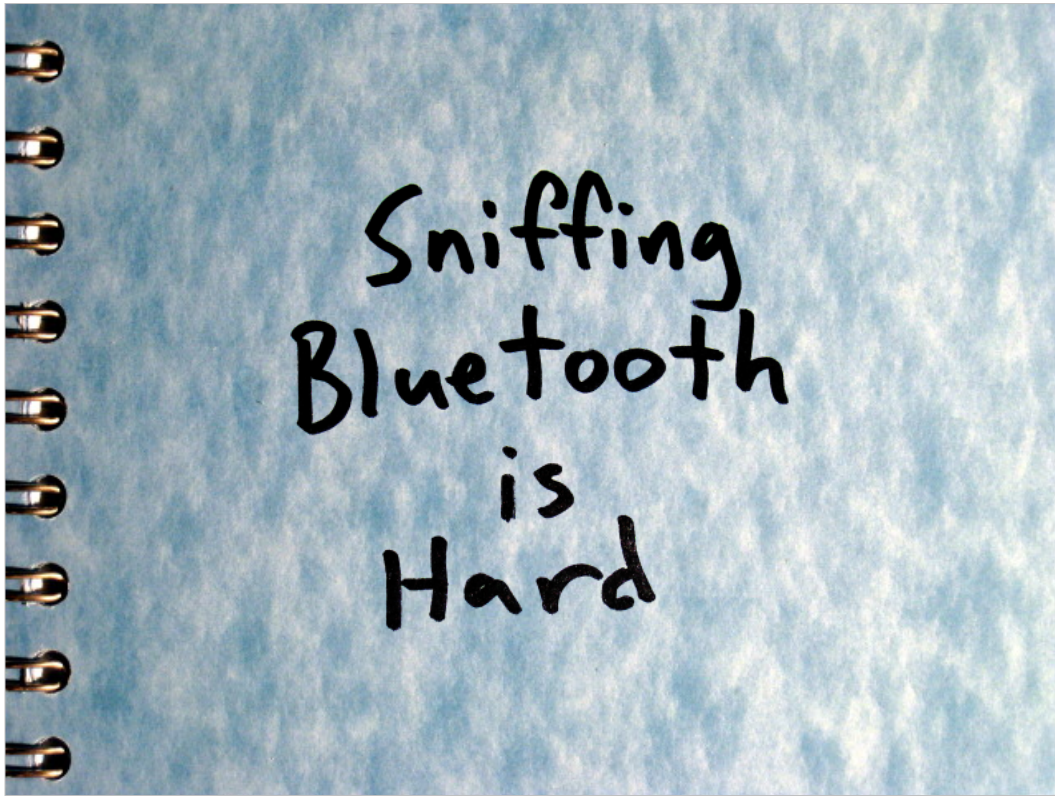
I'm Dominic

University College London
Imperial College London
dspill@doc.ic.ac.uk

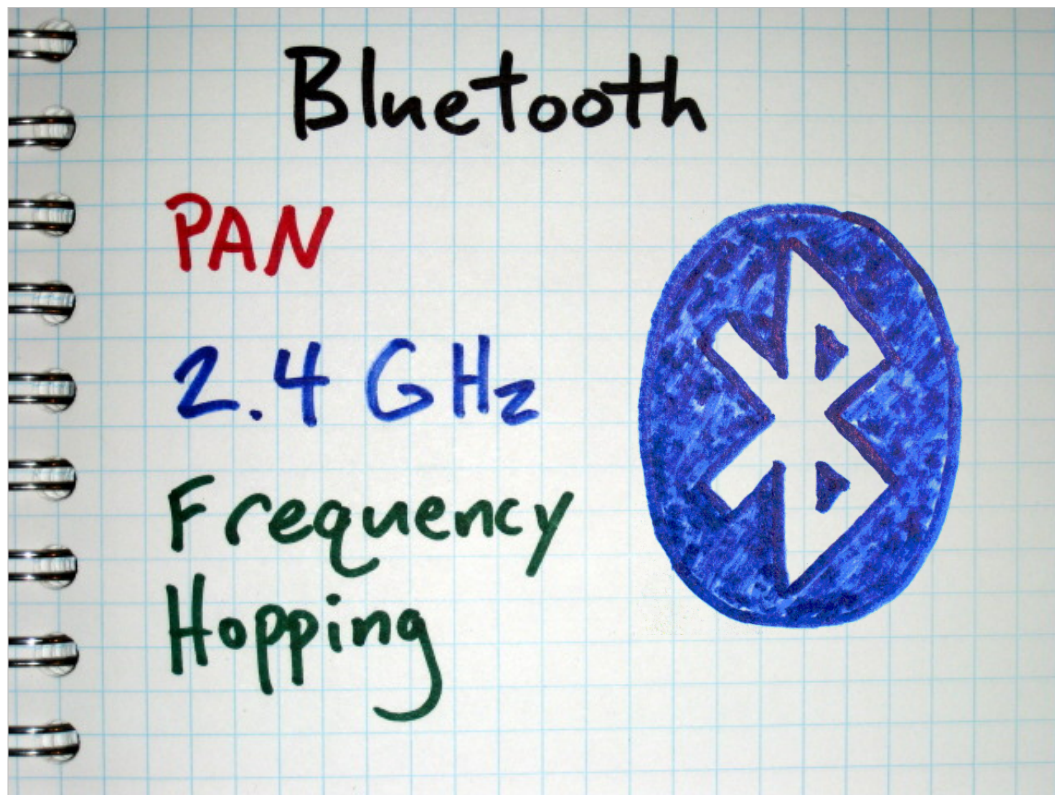
A photograph of a spiral-bound notebook with a light blue grid pattern. The notebook is open, and the text is written in black ink on the right-hand page. The text is a single paragraph that reads: "Certain commercial equipment, materials, and software are identified to specify technical aspects of the reported procedures and results. In no case does such identification imply recommendations or endorsement by the U.S. Government, its departments, or its agencies; nor does it imply that the equipment, materials, and software identified are the best available for this purpose." The handwriting is in a cursive, slightly slanted style. The spiral binding is visible on the left side of the page.

Certain commercial equipment, materials, and software are identified to specify technical aspects of the reported procedures and results. In no case does such identification imply recommendations or endorsement by the U.S. Government, its departments, or its agencies; nor does it imply that the equipment, materials, and software identified are the best available for this purpose.

Certain commercial equipment, materials, and software are sometimes identified to specify technical aspects of the reported procedures and results. In no case does such identification imply recommendations or endorsement by the U.S. Government, its departments, or its agencies; nor does it imply that the equipment, materials, and software identified are the best available for this purpose.



Sniffing Bluetooth is Hard



About Bluetooth

- PAN technology
- 2.4 GHz ISM band
- Frequency hopping

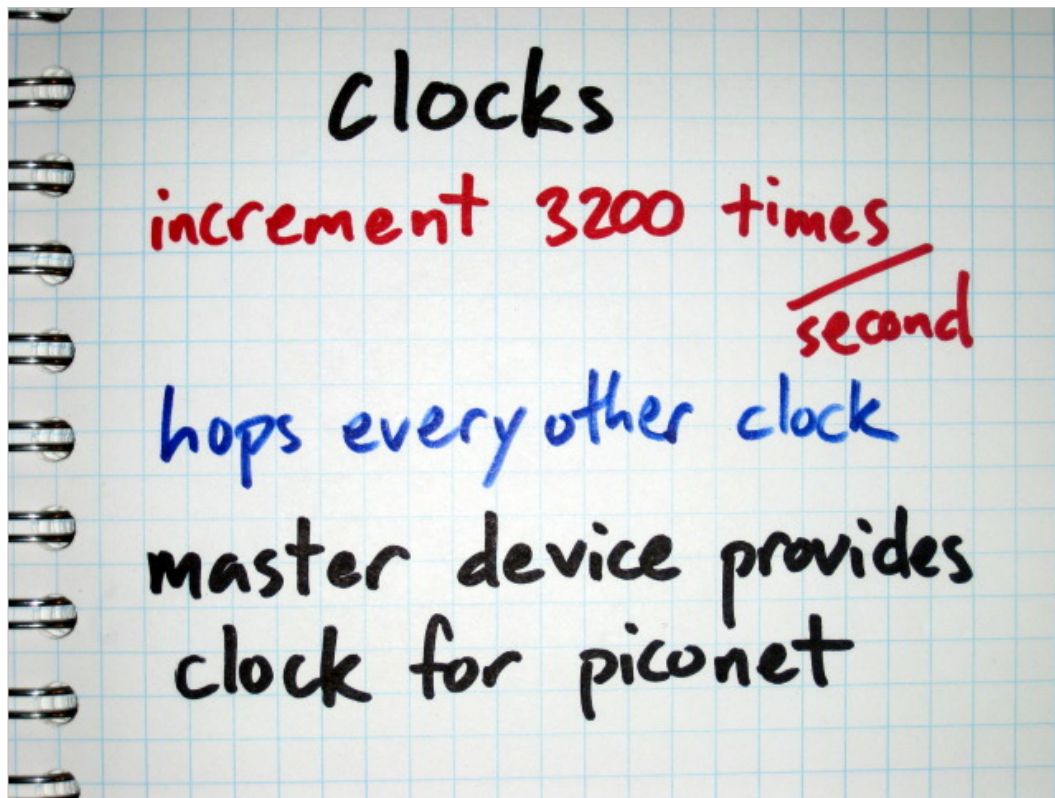
79 MHz

<u>channel</u>	<u>frequency</u>	
0	2402 MHz	79 channels
1	2403	
⋮	⋮	
78	2480	1 MHz wide

1600 hops per second

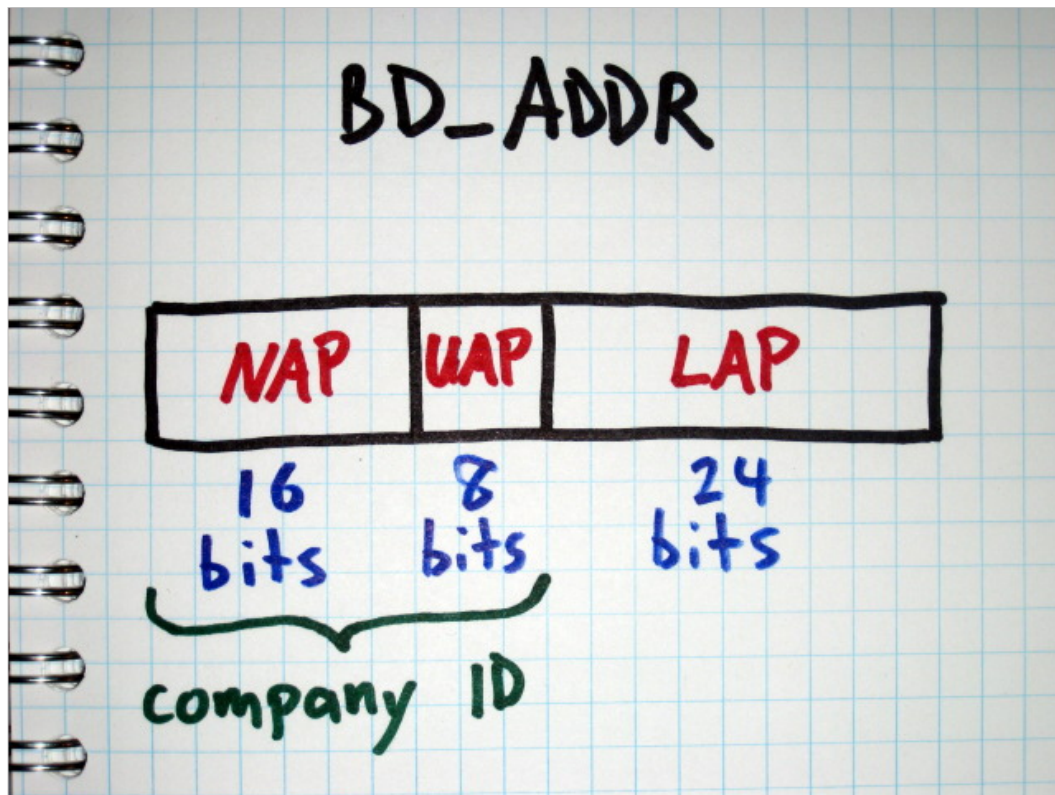
79 MHz

- Bluetooth hops through 79 channels
- Each channel is 1 MHz wide
- A piconet hops 1600 times per second



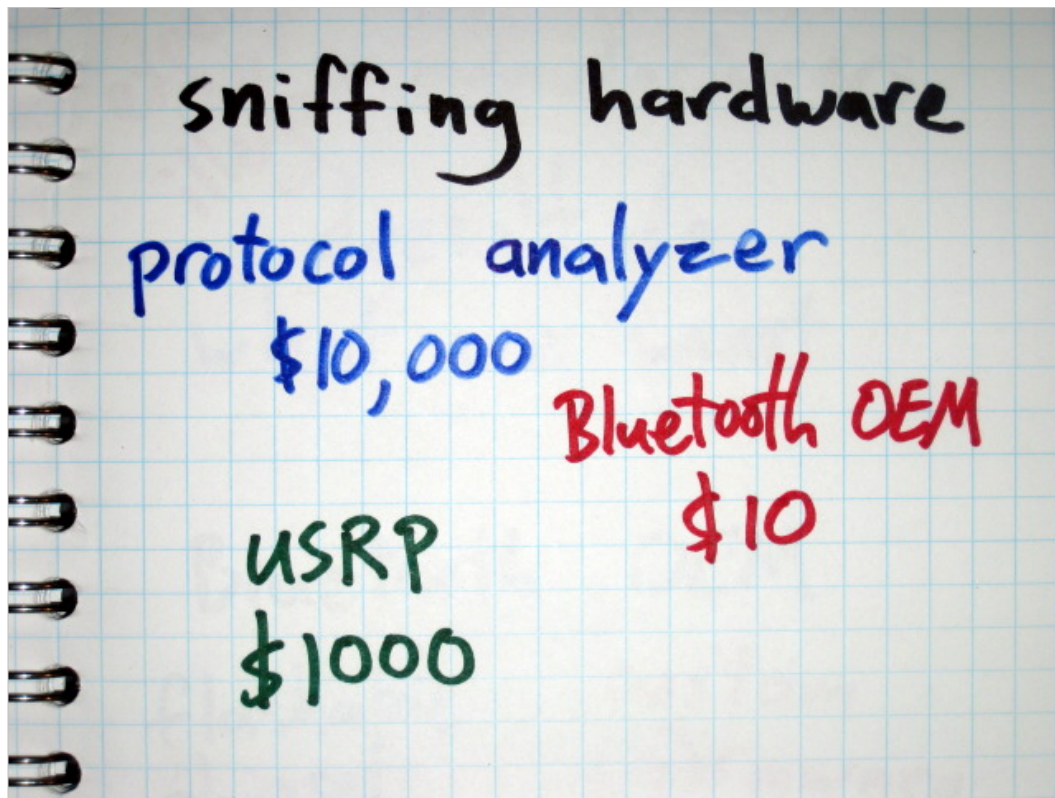
Clocks

- Every Bluetooth device has a clock
- Clocks increment 3200 times per second
- Hops happen every other clock cycle
- The master device dictates the clock of the piconet
- Subordinate devices keep track of their clocks' offset from the master's



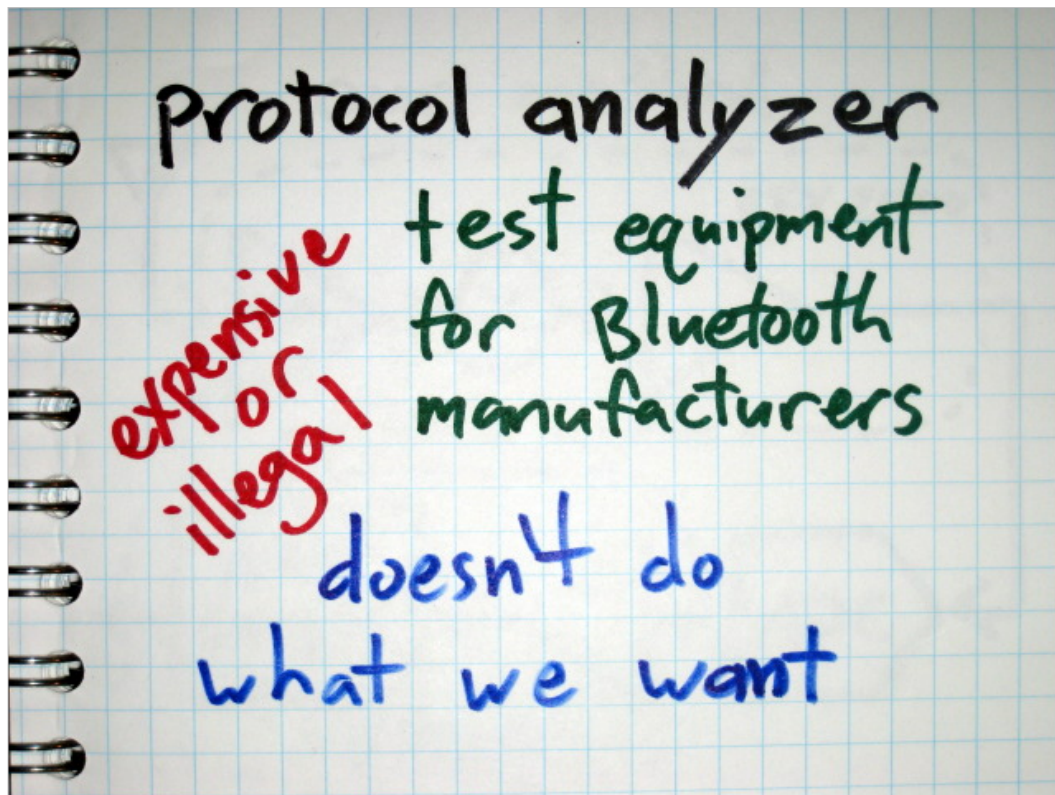
Addresses

- BD_ADDR is like a MAC address
- 48 bits
- Comprised of three parts
 - NAP
 - UAP
 - LAP



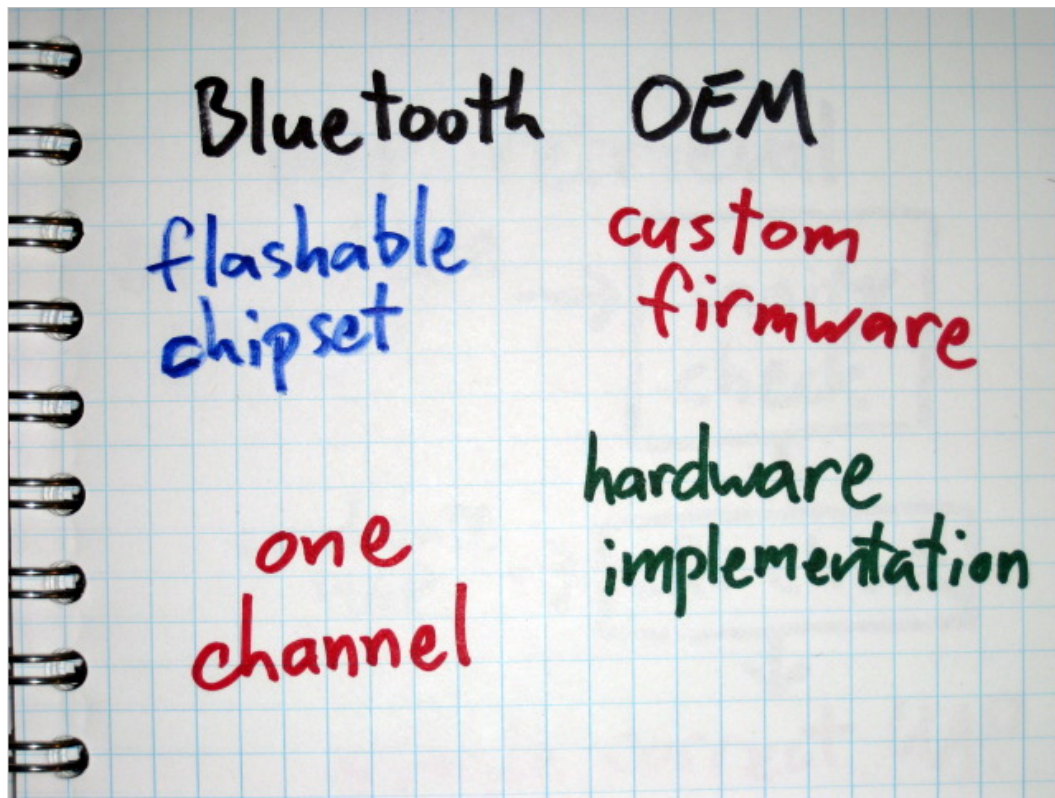
Possible Sniffing Methods

- Protocol analyzer
 - \$10,000
- Bluetooth OEM
 - < \$10
- Wideband receiver
 - USRP (\$1000)
 - USRP2 (\$1800)



Protocol Analyzer

- Test equipment for Bluetooth manufacturers/developers
- Expensive or illegal
 - \$10,000
 - Flash CSR dongles
- Doesn't do what we want
 - Sniff any connection
 - Sniff all connections



Bluetooth OEM

- Flashable chipset
- Custom firmware
- Hardware implementation
 - Cannot sniff arbitrarily
- Not wideband
 - One channel at a time



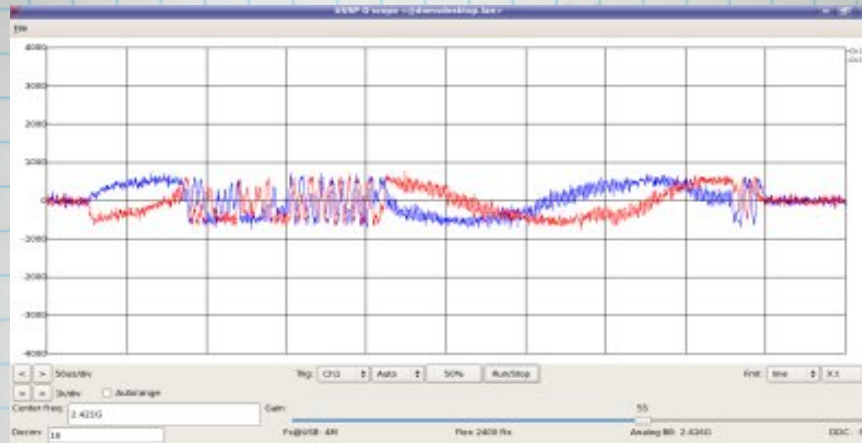
Software Radio (GNURadio/USRP)

- Wideband receiver
 - USRP - 8MHz
 - USRP2 - 25MHz
- Software signal processing
 - Can do whatever we like with data
- This has potential...



Sniffing (Single Channel)

how?

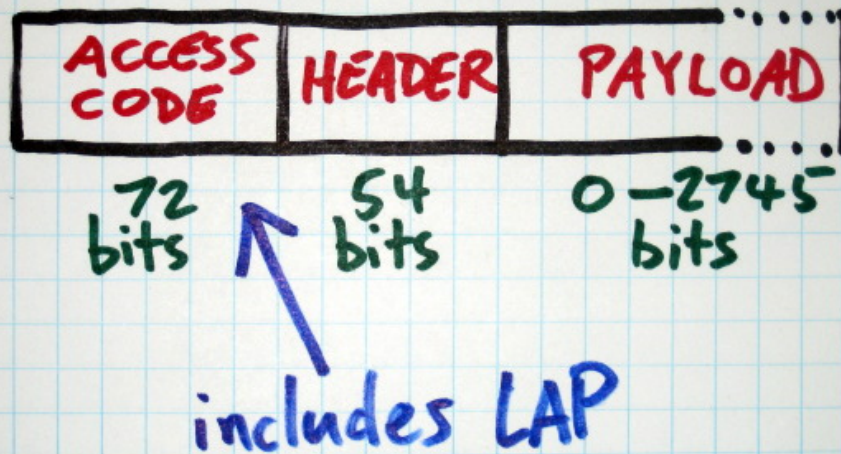


demodulate in software

How?

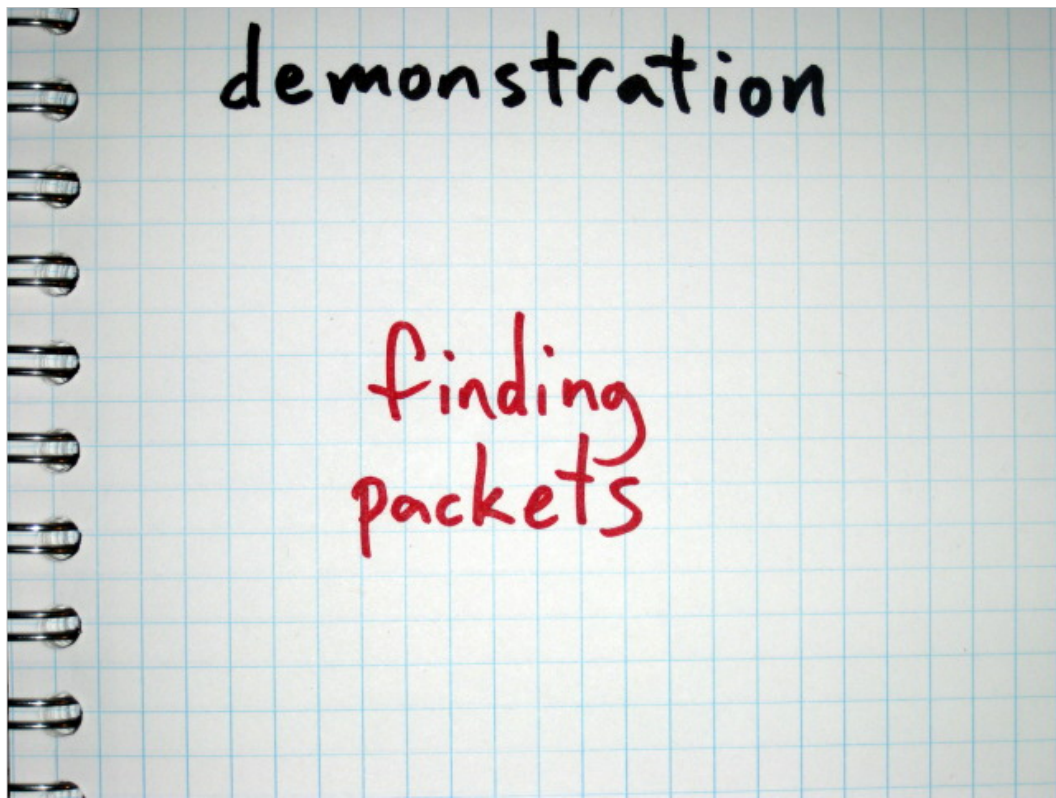
- Step 1: Capture a packet and demodulate
 - Device debug mode and visual tools helpful

Woo Packets!



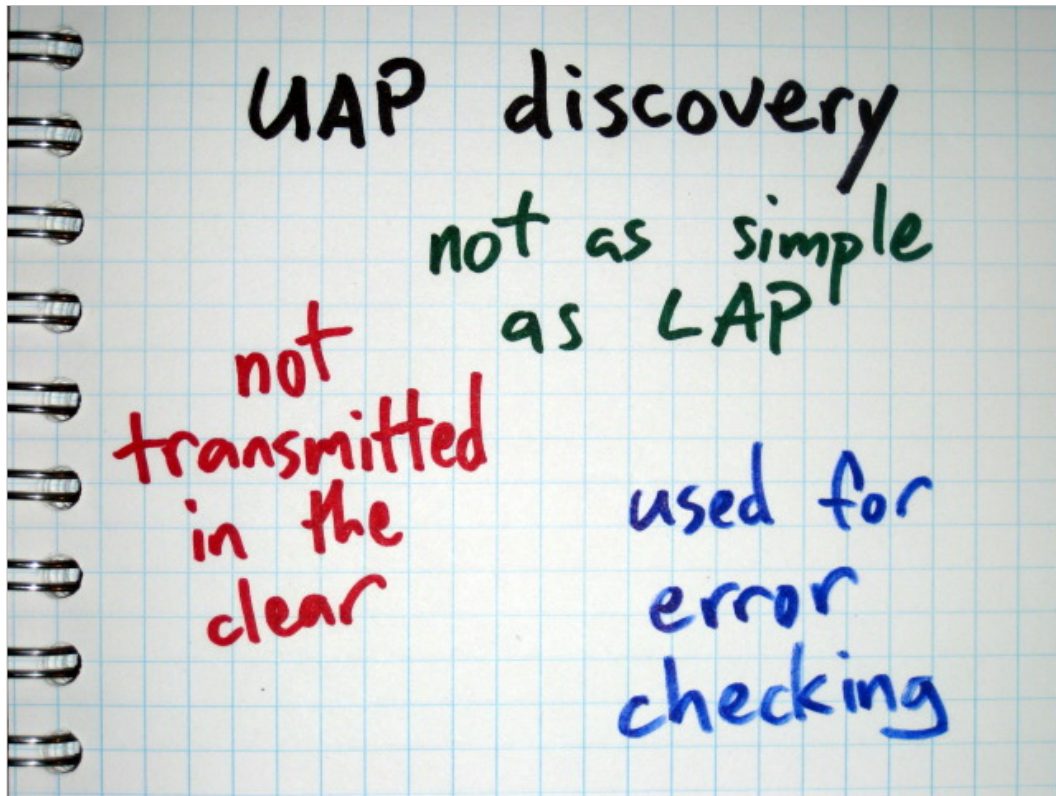
Woo Packets!

- Now let's get some data
- LAP from access code



Demo: Finding Packets

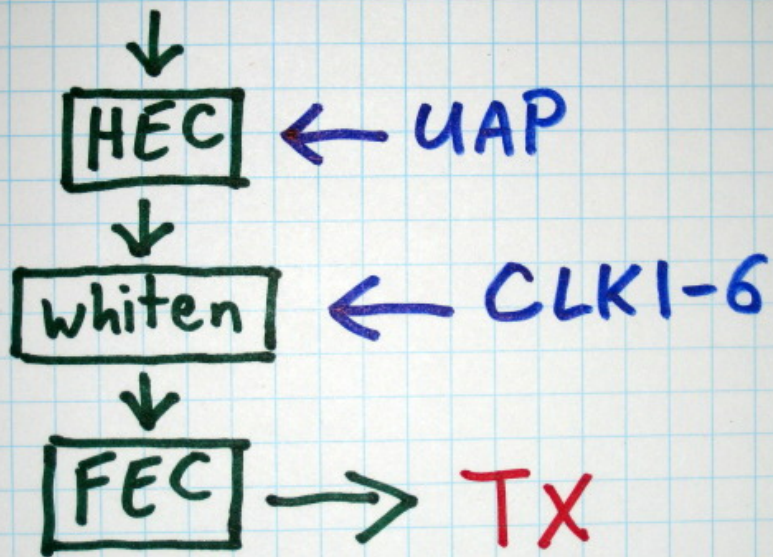
- `btrx.py -i headset3.cfile` (or similar)

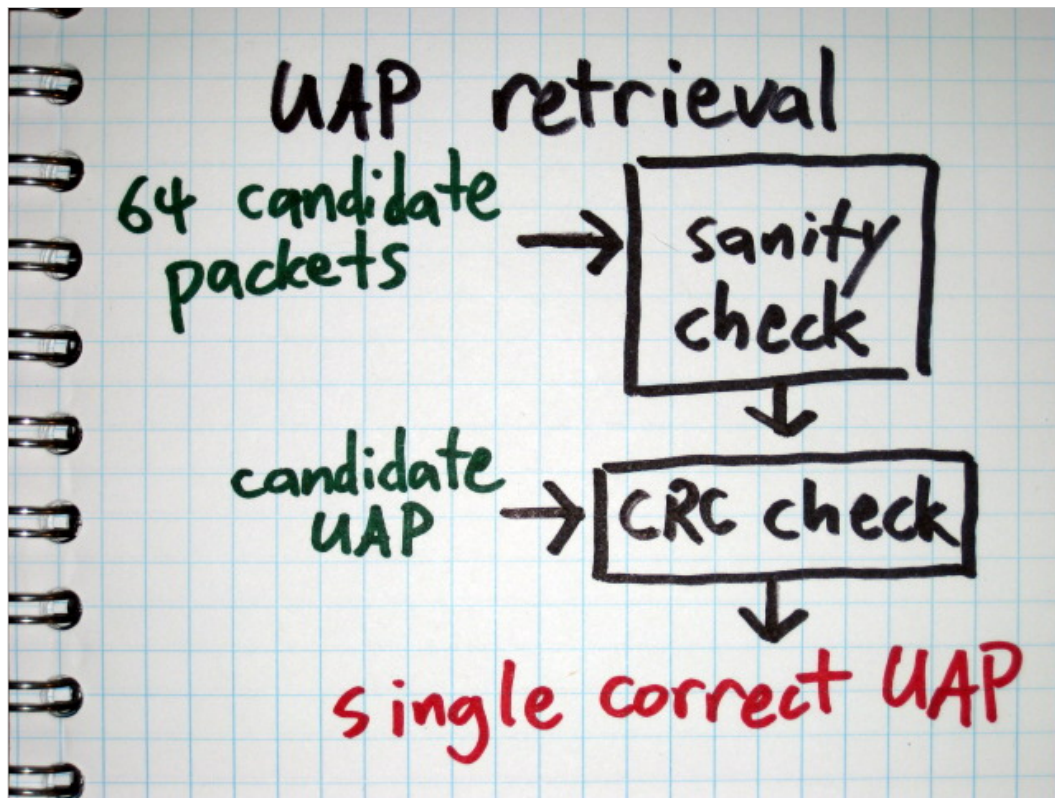


UAP discovery

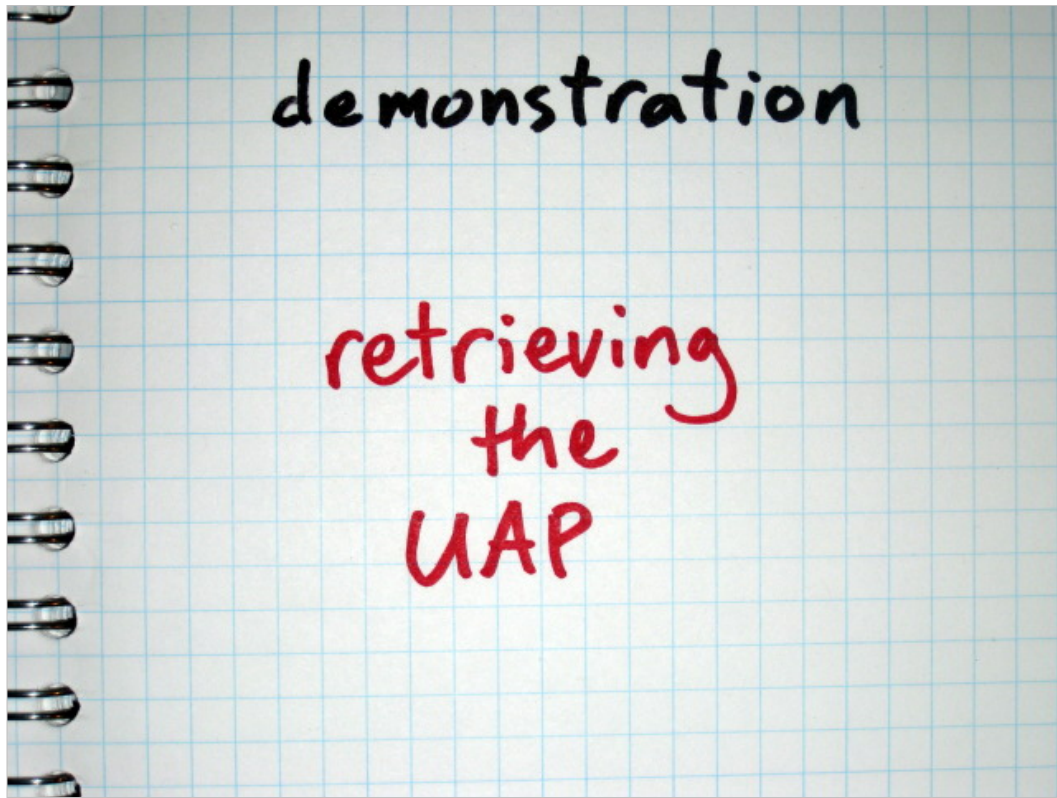
- Not simple like LAP
- Used for checksum calculations
 - Simple algorithms
 - Run in reverse
- Data whitening
 - 64 clock values
 - 64 candidate UAPs

header processing



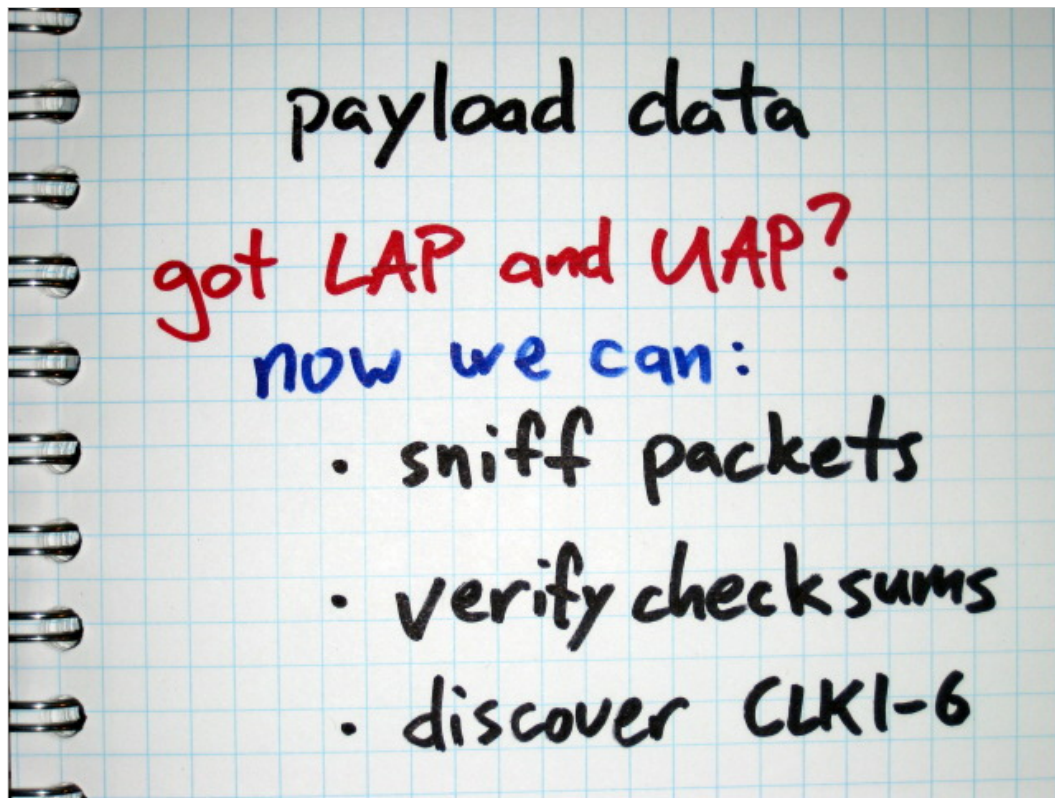


UAP retrieval



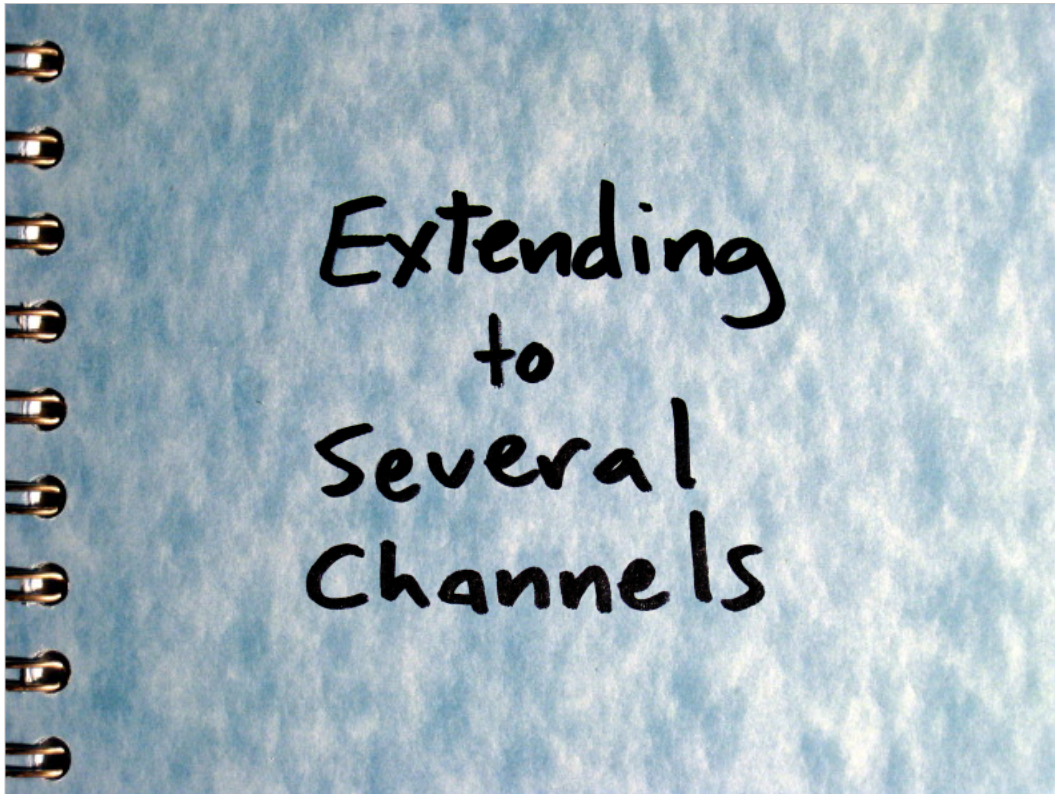
Demo: Retrieving the UAP

- `btrx.py -d 16 -i headset2.cfile -l 24d952` (or similar)

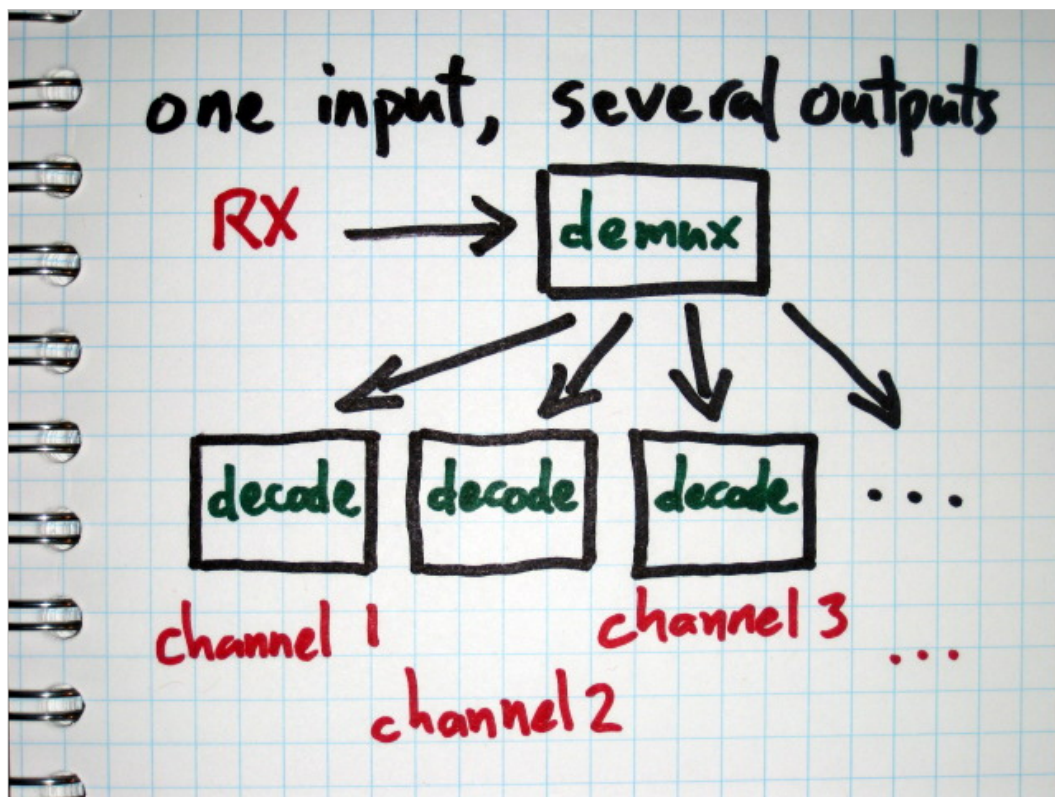


Payload Data

- Got both UAP and LAP
 - Sniff packets from piconet
 - Verify header / payload checksums
 - Discover lower clock bits
- Half way to full clock
 - Hopping



Extending to Several
Channels



One Input, Several Outputs

- USRP can do 8 channels
- USRP2 can do 25

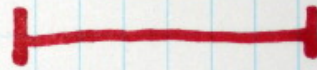
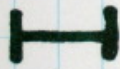
how many channels?

USRP

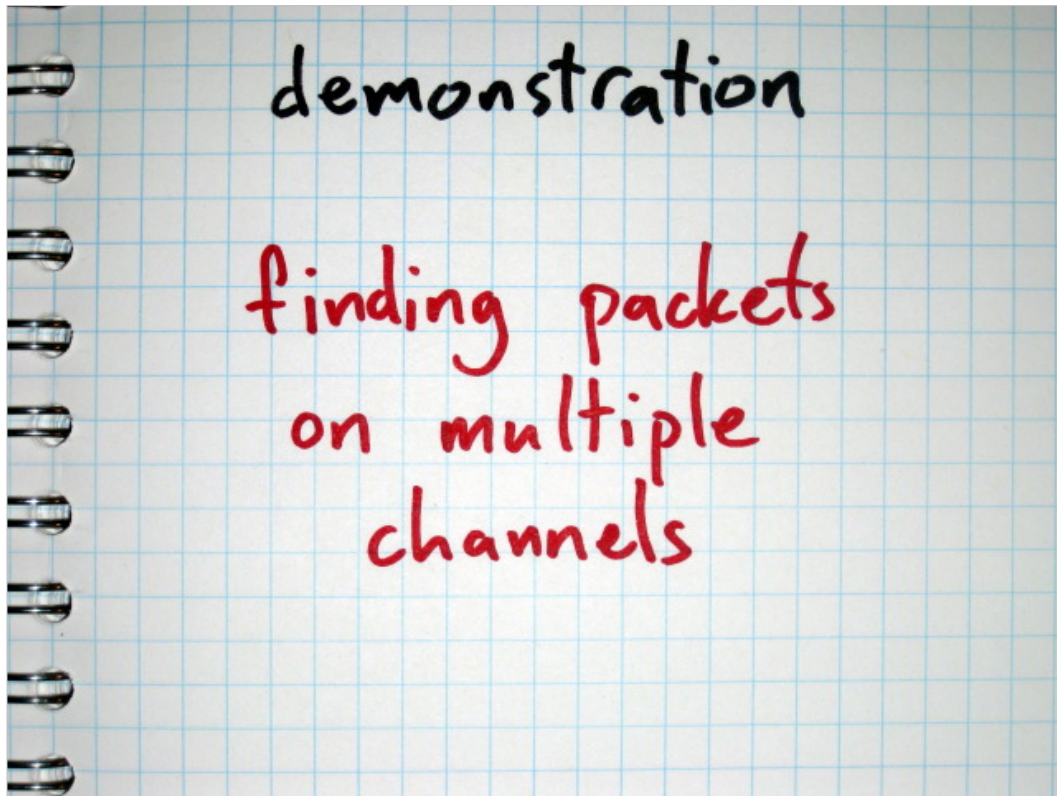
8 channels

USRP2

25 channels



79 Bluetooth channels



Demo: Finding Packets on Multiple Channels

- `multibtrx.py -d 8 -i headset1.cfile -f 2476.5e6 -l 24d952` (or similar)

all 79 channels?

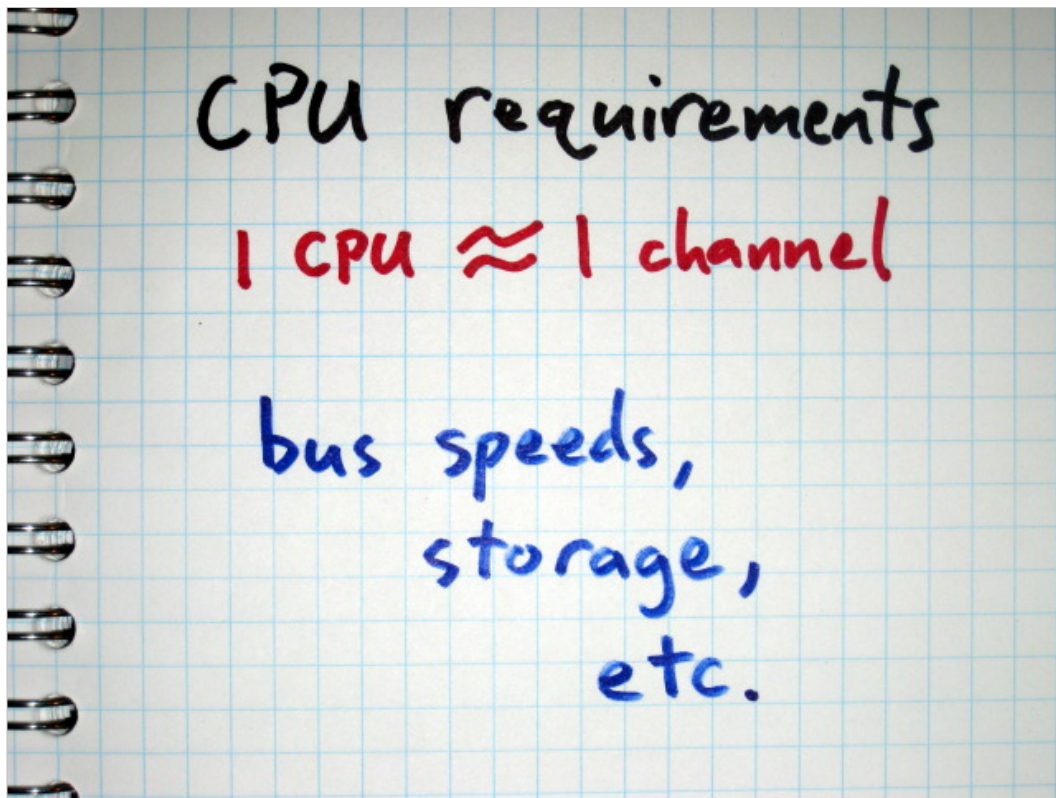
10 USRPs

or

4 USRP2s



- Use 10 USRPs
- Use 4 USRP2s (or 3 USRP2s and 1 USRP)

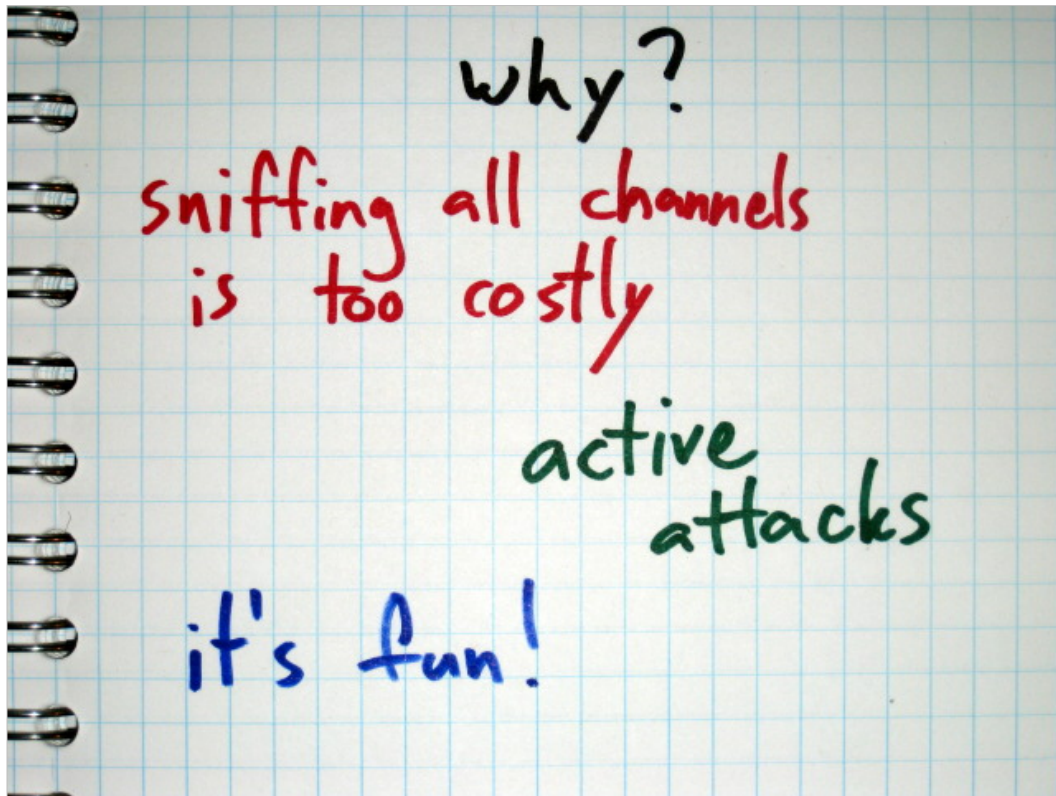


CPU requirements

- Roughly speaking, 1 CPU core can decode 1 channel in real time
- 79 channels == 79 cores
- Bus/storage speeds must also be considered



Predicting the Hopping
Pattern



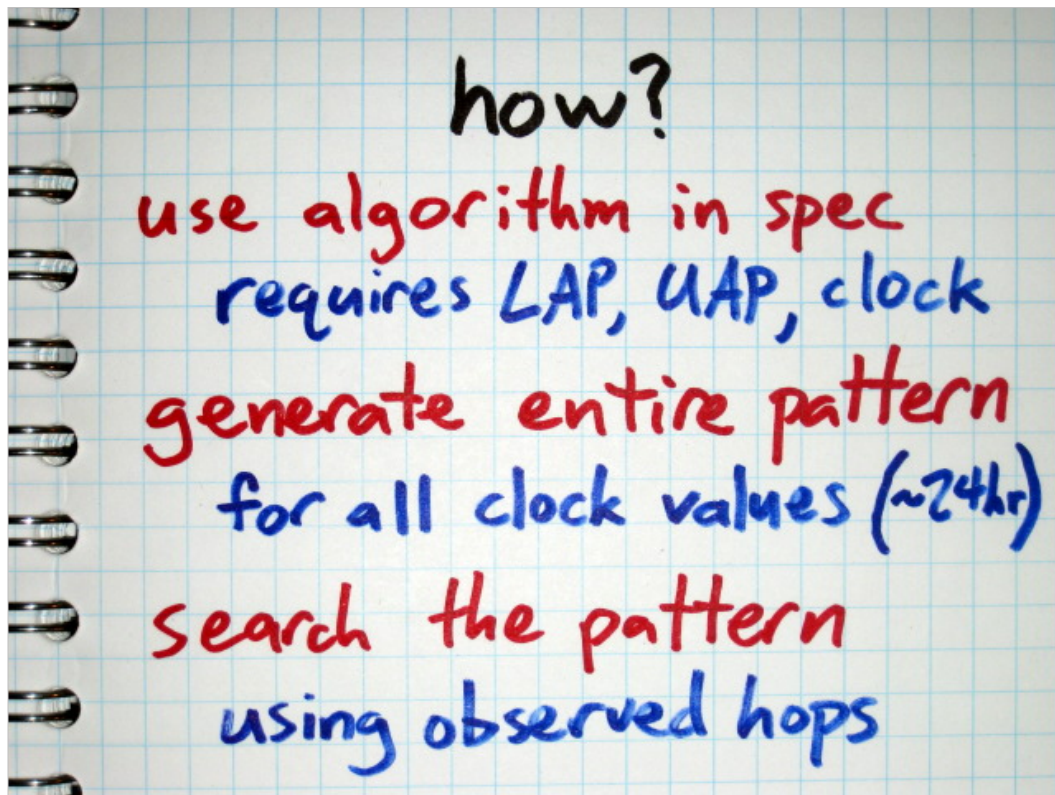
Why?

- Can't sniff all channels
 - Bandwidth limited to 25MHz
- Processing speed
 - One channel per core is slow
- Active attacks
 - Hopping is a prerequisite
- It's fun
 - It's a challenge

hopping sequence		
	<u>clock</u>	<u>channel</u>
UAP 0x65	0	20
	2	60
LAP 0x87CBA9	4	53
	6	62
	⋮	⋮

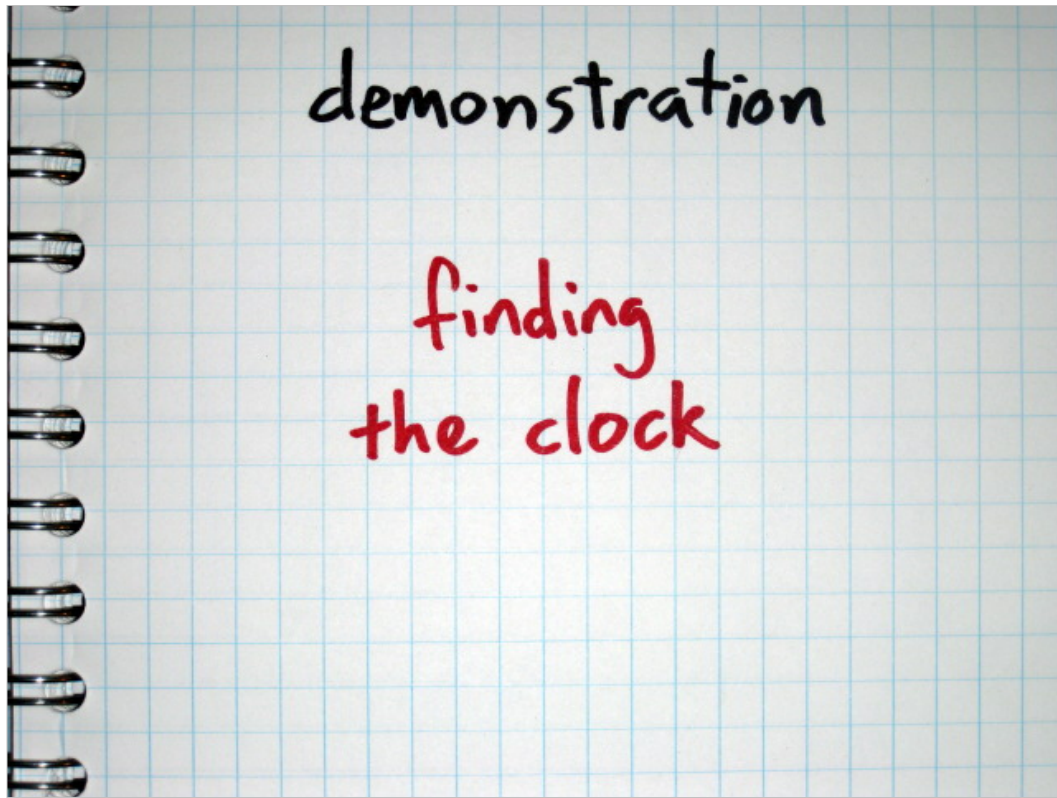
Hopping Sequence

- Pseudo-random sequence based on
 - CLK1-27 of master
 - Low 28 bits of BD_ADDR (LAP and part of UAP)
- If you know these two values, you can predict the sequence



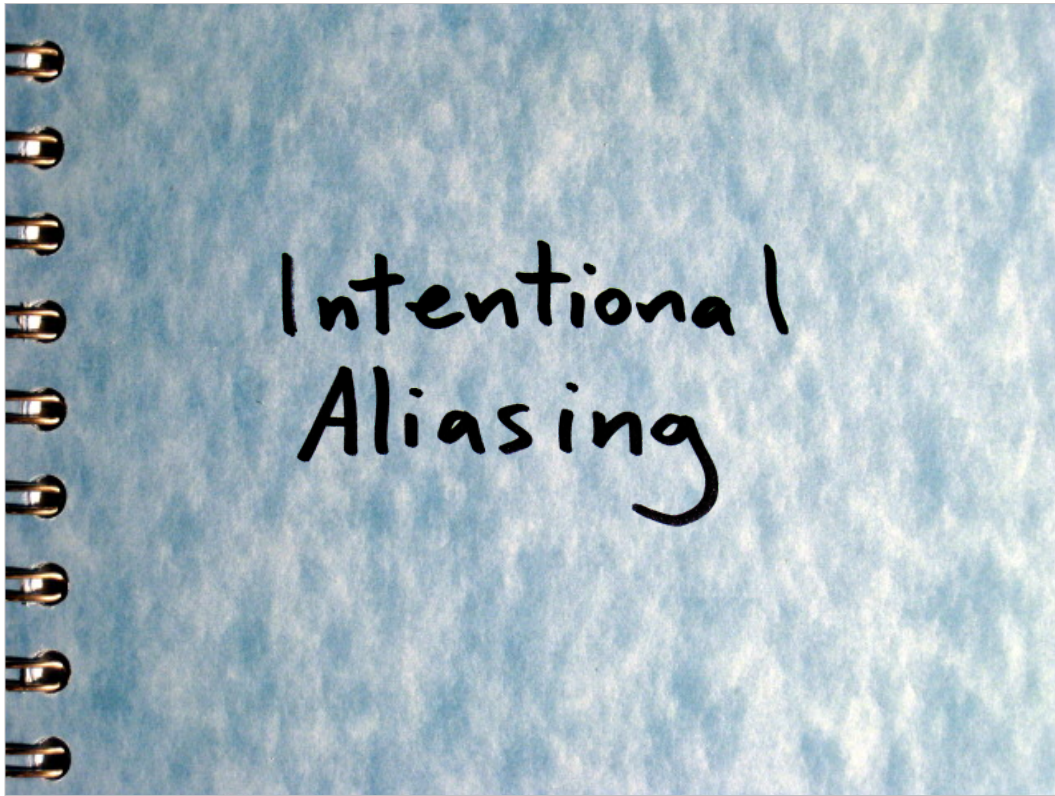
How?

- Algorithm given in Bluetooth spec
 - LAP / UAP
 - Clock
- Generate entire pattern
 - Loops in ~24hrs
- Where in the pattern are we?
 - Know lower 6 bits and corresponding channels
 - Test packet gaps and channels

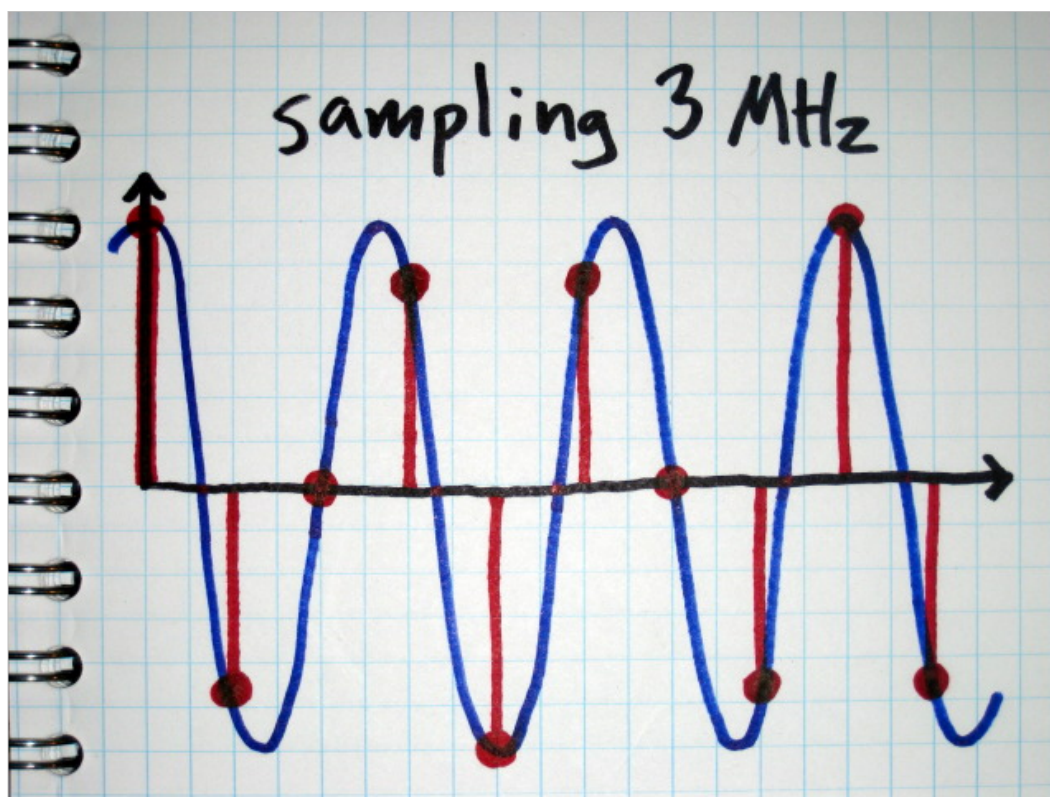


Demo: Finding the Clock

- `btrx.py -d 16 -i headset2.cfile -l 24d952 -n 74` (or similar)



Intentional Aliasing



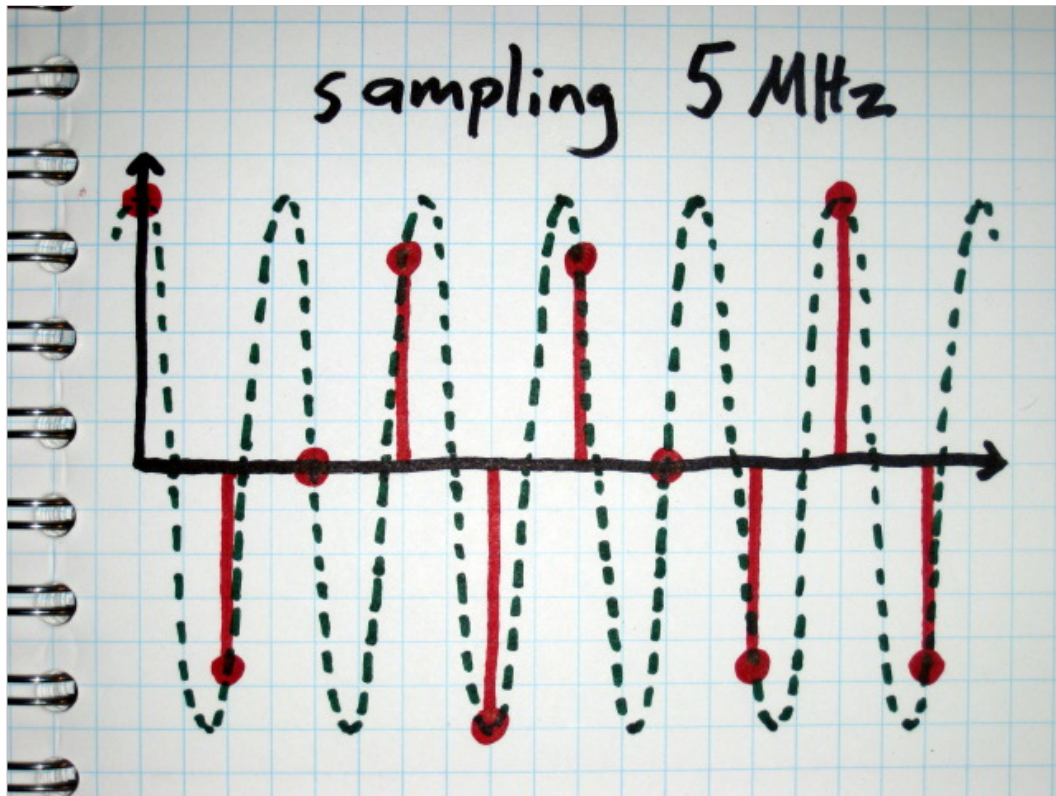
Sampling 3 MHz

One of the fundamental tools of digital signal processing (DSP) is the sampling of analog signals.

While there are some DSP applications that are entirely digital, most DSP systems include both analog and digital components. For example, the sound card on your computer can receive an analog signal from a microphone and sample it to produce a digital signal. It can also convert a digital signal into an analog signal destined for speakers. Software radio systems operate in essentially the same way but with antennas instead of microphones and speakers.

Digital sampling is the very simple process of measuring the value of something at many discrete moments over time, usually at regular intervals. If you measure the rainfall at your home every day, you are acting as a digital sampler; nature provides the analog signal (the rainfall that varies continuously over time), and you convert it by periodic sampling into a digital signal (a sequence of discrete values). Your sound card and microphone measure variations in air pressure, but, instead of sampling at a rate of one sample per day, they sample at perhaps 48 thousand samples per second. The USRP2 we use for Bluetooth monitoring samples radio signals at a rate of 100 million samples per second.

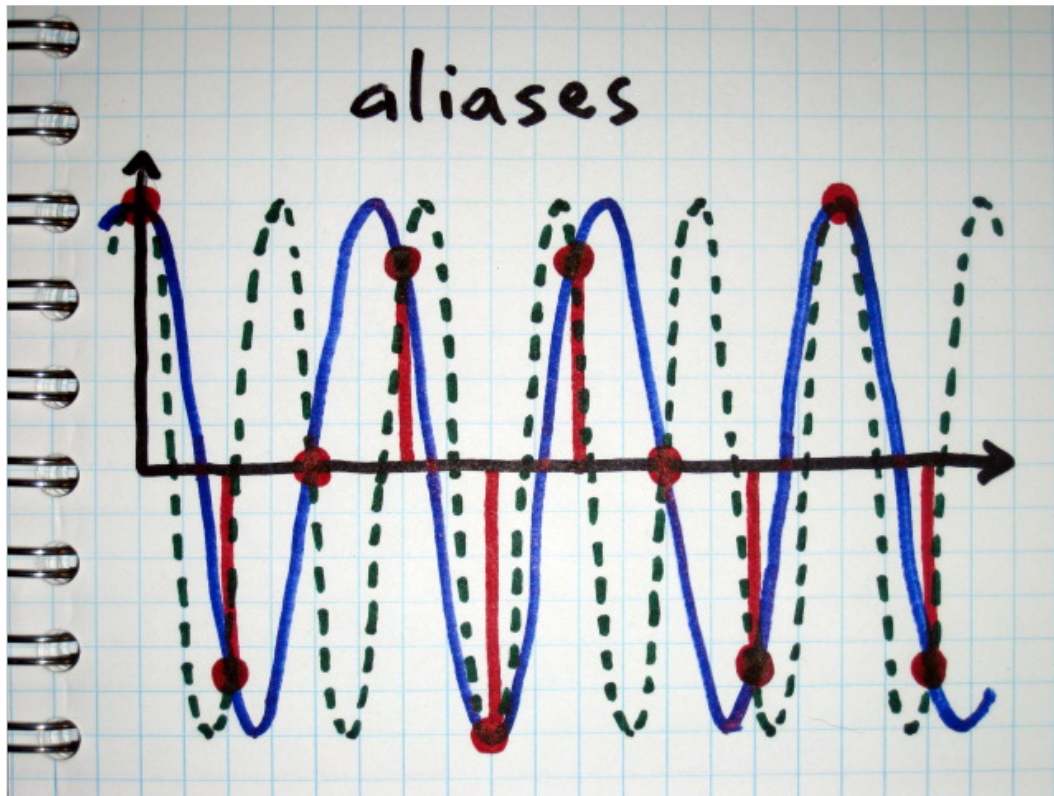
Let's say we have an ADC (Analog to Digital Converter) that samples at a rate of 8 million samples per second (MSPS). If a pure sine wave (the blue analog signal) comes along with a frequency of 3 MHz (3 million cycles per second), our sampler will produce a stream of digital values like so (the red dots). 3 MHz is $\frac{3}{8}$ of our sample rate, so there are 8 samples taken every 3 periods of the analog signal. We can clearly see that the digital (red) signal repeats itself every 8 samples, and, even without the blue line present, it isn't hard to envision the 3 MHz analog signal that the red sequence represents (with a period of 2.67 samples). So now we have a way (the red dots) to digitally represent a pure sine wave with a frequency of 3 MHz. If an analog signal were to arrive with a frequency of 3.2 MHz, the resultant digital signal would certainly be distinct from this one (although the difference might not be apparent for the first few samples).



Sampling 5 MHz

Now suppose that a 5 MHz analog signal comes along (the dashed green line). Our 8 Msps sampler converts the analog signal into a digital signal like so (the red dots). Since 5 MHz is $5/8$ of our sample rate, there are 8 samples taken every 5 periods of the analog signal. The digital signal once again clearly repeats itself every 8 samples. This sequence of values (the red dots) can be thought of as a digital representation of a 5 MHz sine wave. The sequence is distinct from those that represent other nearby frequencies such as 5.2 MHz.

You might notice, however, that the digital sequence that represents a 5 MHz sine wave looks familiar. It is the same sequence that represents a 3 MHz sine wave!



Aliases

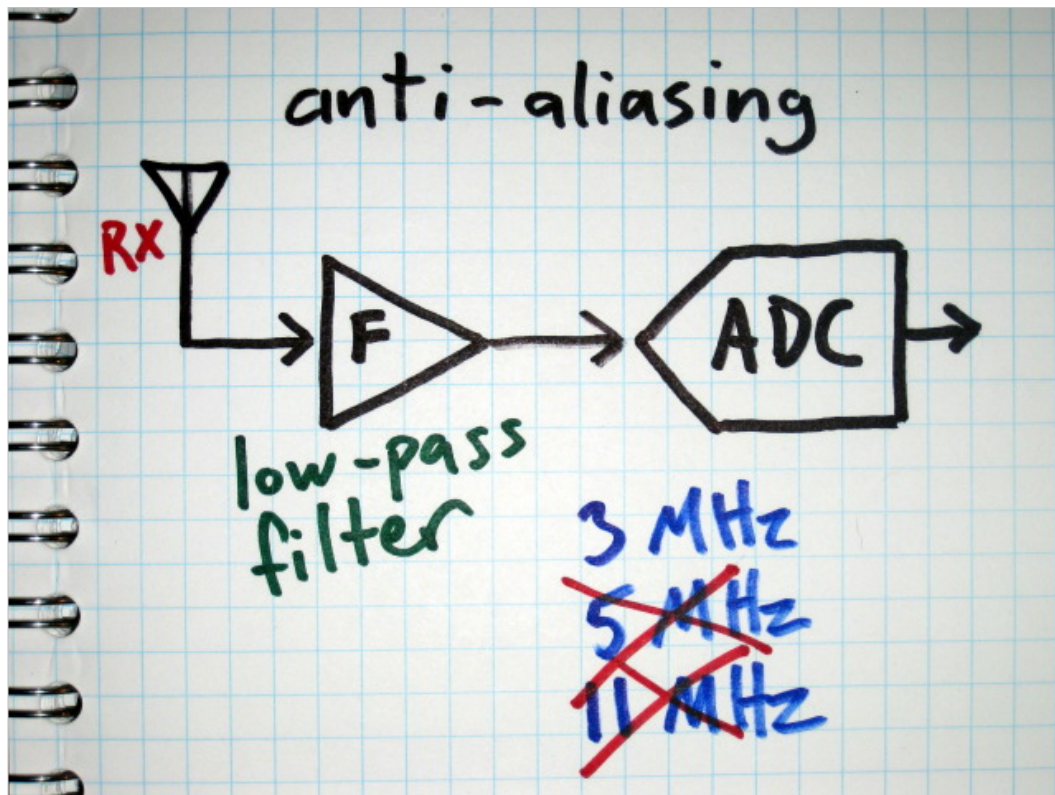
In the analog domain, a 3 MHz signal is quite distinct from a 5 MHz signal, but, in the digital domain, the two are indistinguishable (with a sample rate of 8 Msps). If you take the time to draw a sine wave with a frequency of 11 MHz (you'll need a finer pen and steadier hand than mine), you'll find that it, too, passes through the same red dots. In fact, the set of frequencies that share these points is infinite and includes 3, 5, 11, 13, 19, 21, 27, 29, and 36 MHz and so forth. These frequencies are all 3 MHz away from integer multiples of the sample rate (8 Msps).

Our digital sequence (the red dots) exhibits ambiguous frequency. In the analog domain, this would be weird, but, in the digital domain, it is completely normal. Every digital sequence can be thought of as a set of ambiguous frequency components. There is no such thing as a digital signal with a single frequency because every frequency component is really an infinite set of frequencies. The frequencies aren't completely ambiguous, however. (A 3.2 MHz signal can be distinguished from a 3 MHz signal.) They are only ambiguous with respect to integer multiples of the sample rate.

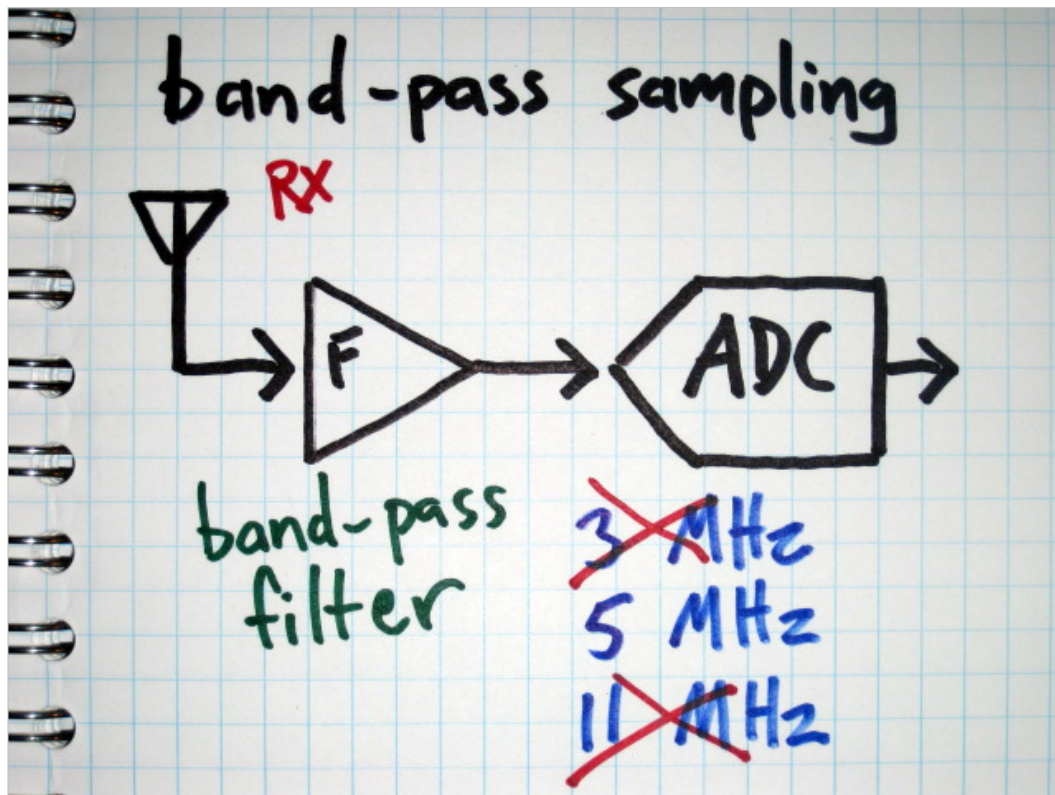
This is the basis of the Nyquist sampling theorem. One way to state the theorem is that, in order to unambiguously represent an analog signal in the digital domain, you have to sample at a rate that is at least twice the frequency of your analog signal of interest. In other words, our 8 MHz sampler is only able to unambiguously represent signals with frequencies lower than 4 MHz. (A 4.1 MHz signal would be indistinguishable from a 3.9 MHz signal, and a 6.2 MHz signal would be indistinguishable from 1.8 MHz.)

If we were comfortably above the Nyquist limit, that is, if our signal of interest were well below 4 MHz, we might run into trouble if a neighboring signal between 4 and 8 MHz showed up. In the analog domain, the neighboring signal wouldn't interfere with our signal below 4 MHz. In the digital domain, however, we might detect a 5 MHz signal, and it would directly interfere with a 3 MHz signal. Because of the ambiguity in the digital domain, the 5 MHz signal would be indistinguishable from a 3 MHz signal.

When a 5 MHz analog signal produces an apparent 3 MHz signal in the digital domain, we call the apparent 3 MHz signal an "alias" of the 5 MHz signal. The signal has an infinite number of aliases at 3 MHz, 11, 13, 19, 21, and so forth. This phenomenon of "aliasing" is usually considered a problem to be avoided, typically by filtering in the analog domain prior to sampling.



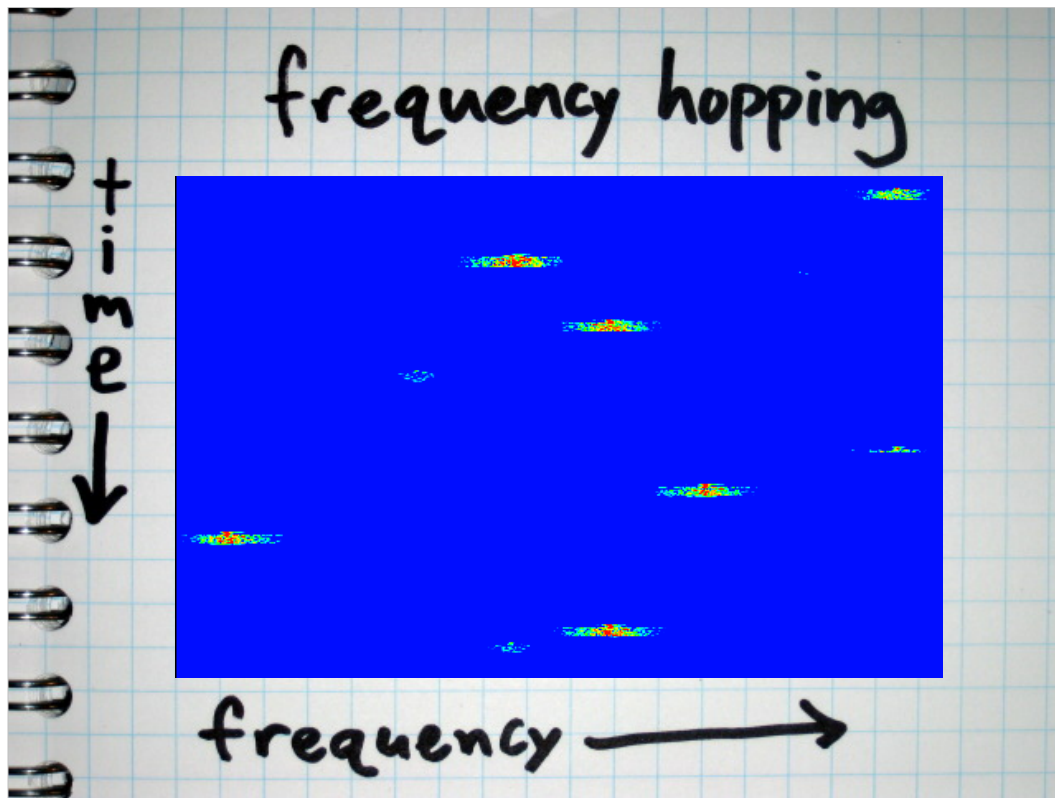
If our signal of interest is entirely below 4 MHz, then we can use an analog "anti-aliasing" filter that filters out all frequencies above 4 MHz, allowing only lower frequencies to pass through (a low-pass filter) to the ADC. By using this anti-aliasing filter before sampling, we can effectively eliminate any ambiguity in the digital domain. If we detect a digital frequency component at 3 MHz, we can be certain that there really was a 3 MHz analog signal and that we are not seeing an alias of an analog signal at 5 MHz or higher. In this way we are able to eliminate interference from analog signals above 4 MHz.



Band-Pass Sampling

Now let's suppose that we would like to sample an analog signal of interest between 4 and 8 MHz. As long as the signal does not contain frequency components beyond these boundaries, we should be able to sample it with our 8 Msps ADC. In the digital domain, our signal is indistinguishable from signals in the 0 to 4 MHz range, the 8 to 12 MHz range, and so forth, but frequencies between 4 and 8 MHz are distinguishable from each other. To ensure that we receive only those signals that are between 4 and 8 MHz, we can use an anti-aliasing filter that filters out frequencies below 4 MHz as well as those above 8 MHz (a band-pass filter). Because the analog signal arriving at the ADC is limited by such a filter, this technique is called band-pass sampling. We can band-pass sample signals in any 4 MHz wide range bordered by an integer multiple of the sample rate (4 to 8 MHz, 8 to 12 MHz, 12 to 16 MHz, and so on), but at some point there is an absolute maximum due to the limited precision of our ADC timing.

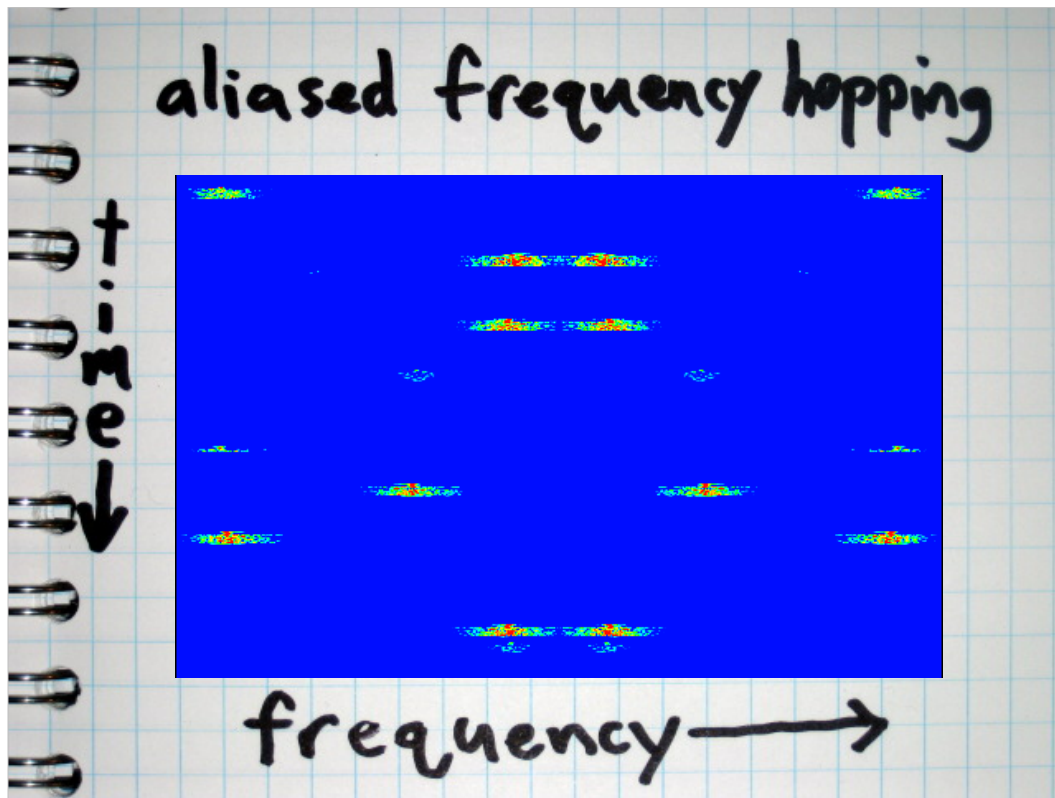
A more general statement of the Nyquist sampling theorem that allows for band-pass sampling is this: In order to unambiguously represent an analog signal in the digital domain, you have to sample at a rate that is at least twice the bandwidth of your analog signal of interest. Bandwidth is a measure of the range of frequency components of a signal. A signal that fits between 4 and 8 MHz has a bandwidth of no more than 4 MHz, therefore it can be sampled by an ADC operating at a sample rate of at least 8 Msps.



Frequency Hopping

Bluetooth is a frequency hopping system. At any given moment, a Bluetooth piconet uses a single channel with a bandwidth of 1 MHz, but the network switches among many different channels (adjacent 1 MHz bands) 1600 times per second. A normal Bluetooth system uses 79 channels at frequencies of 2402 MHz through 2480 MHz, but let's suppose we wanted to monitor an unusual Bluetooth piconet that uses only 8 channels at much lower frequencies between 0 and 8 MHz (channel one is centered at 0.5 MHz, channel 2 at 1.5 MHz, and so forth).

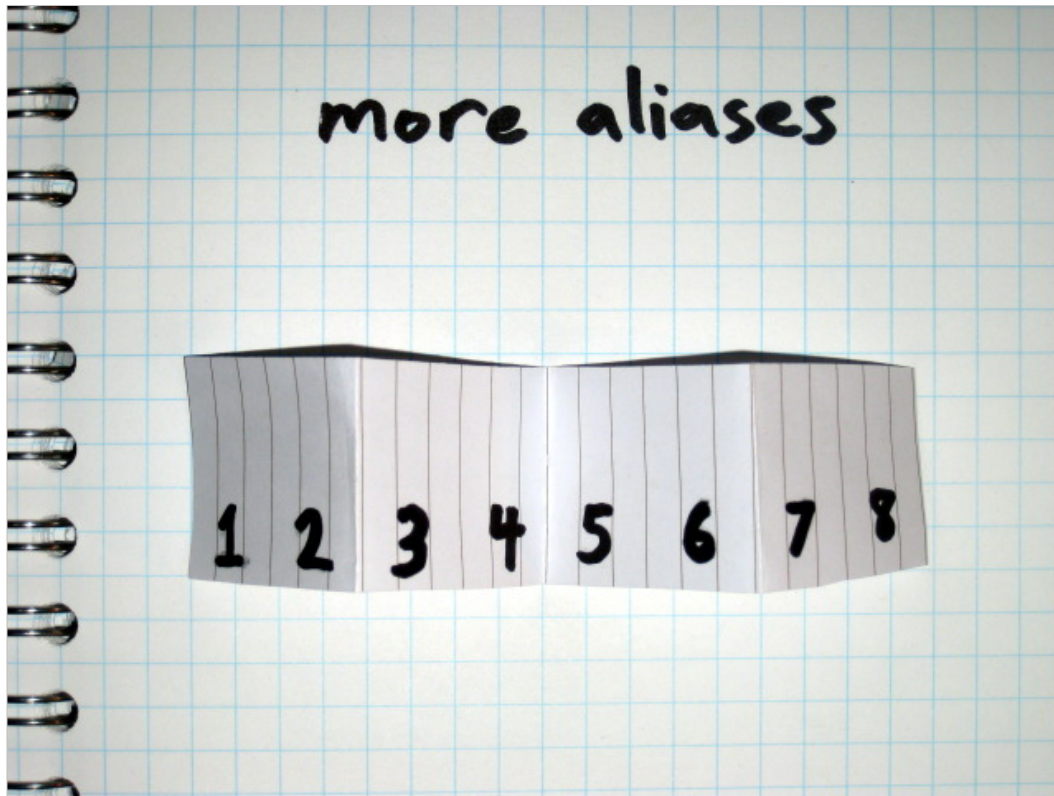
If we only wanted to monitor channels 1 through 4, we could use a low-pass anti-aliasing filter. To monitor channels 5 through 8, however, we would need a band-pass anti-aliasing filter. Nyquist says that we can only monitor one of these two sets of channels at a time.



Aliased Frequency Hopping

Without filtering the two ranges from each other, our 8 Msps sampler would fail to distinguish channel 1 from channel 8, channel 2 from channel 7, and so forth. If a transmission on channel 1 occurs at the same time as a transmission on channel 8, they would interfere with each other in the digital domain.

Lucky for us, a Bluetooth piconet never transmits on more than one channel at a time! If we are willing to live with the fact that we can't distinguish certain channels from one another, there is nothing stopping us from monitoring all 8 channels with our 8 Msps ADC. We just need a double wide anti-aliasing filter, a low-pass filter at 8 MHz. Although they are usually thought of as things to be avoided, aliases can be our friends. When we monitor channel 1 with this system, we sometimes receive frames transmitted on channel 1 and sometimes receive aliases of frames transmitted on channel 8. We can't tell them apart, but they should never interfere with each other.

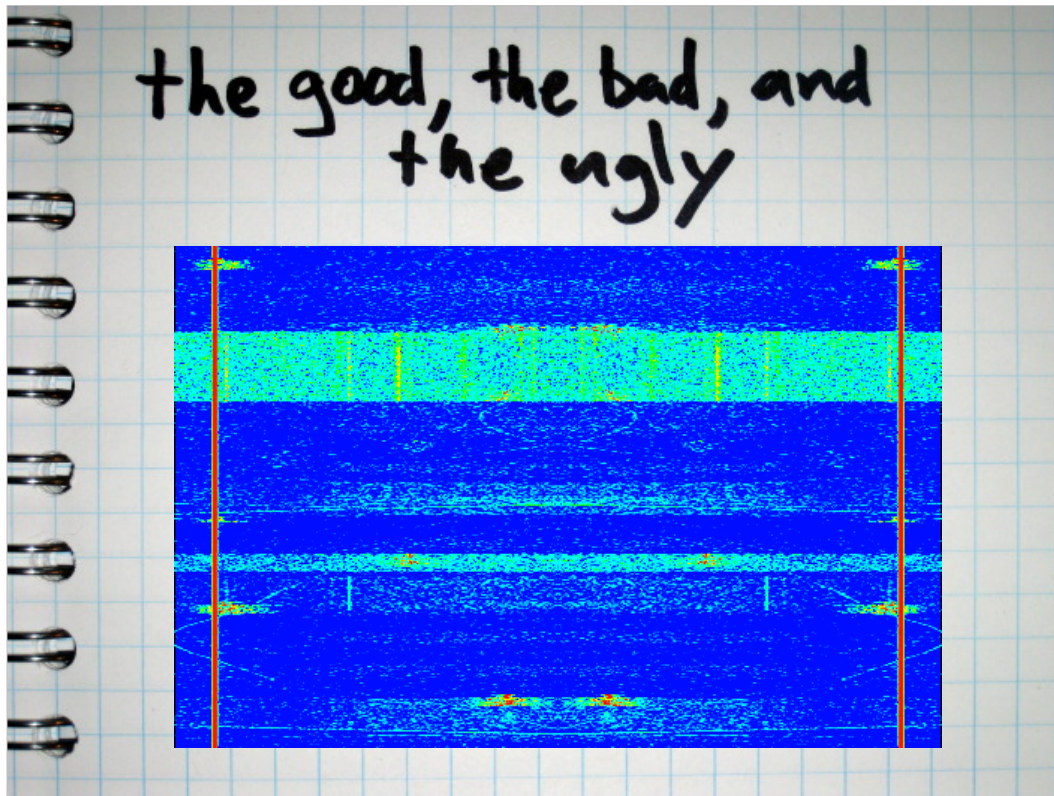


Aliases, Aliases, and More Aliases

We could go crazy and extend this idea further. If our ADC only operates at 4 Msps, we could use the same technique (with a low-pass filter at 8 MHz) and double up our aliased channels a second time. Imagine folding a piece of paper over on itself once and then folding it over a second time. Channels 1, 4, 5, and 8 would all be aliased on top of each other (and thus indistinguishable from one another), and Channels 2, 3, 6, and 7 would be aliased on top of each other as well. Our monitoring software would lose even more information about which channel is which, but it would only have to monitor two channels in the digital domain. The ultimate extension of this technique would be to cut the sample rate in half yet again. If we operate our ADC at 2 Msps (retaining the 8 MHz low-pass anti-aliasing filter), all 8 channels would be aliased on top of each other, and our monitoring system would see them all as one channel in the digital domain.

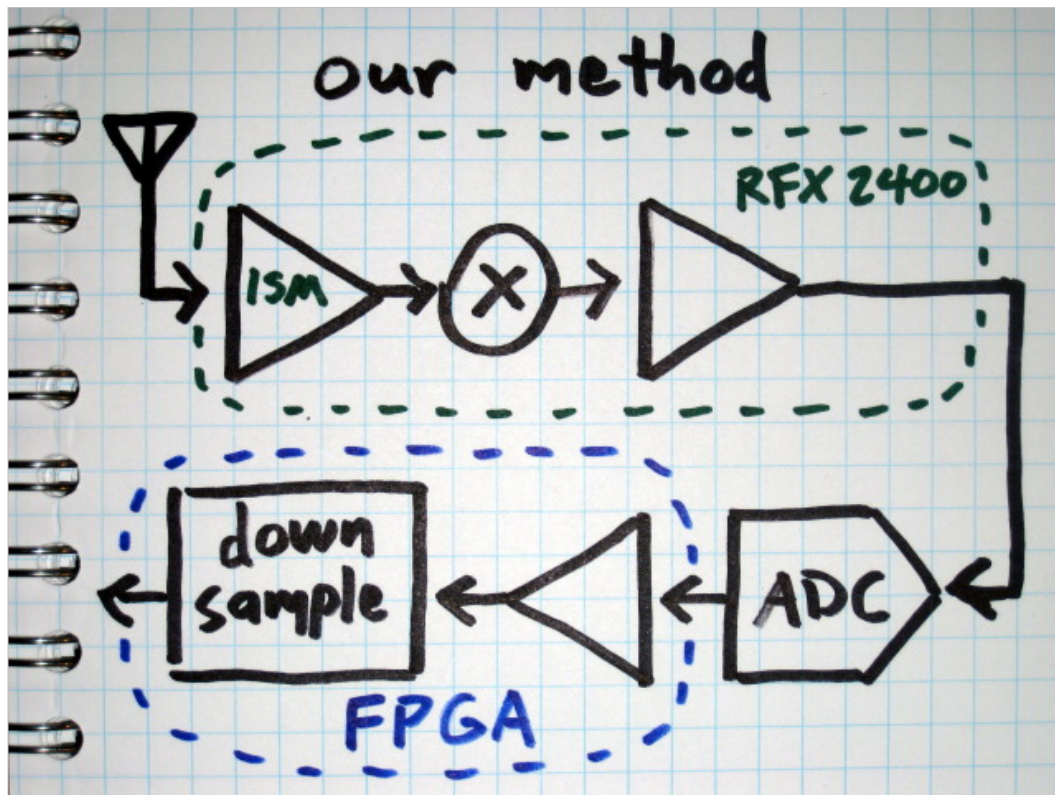
more aliases





The Good (more channels), the Bad (more noise), and the Ugly (more interference)

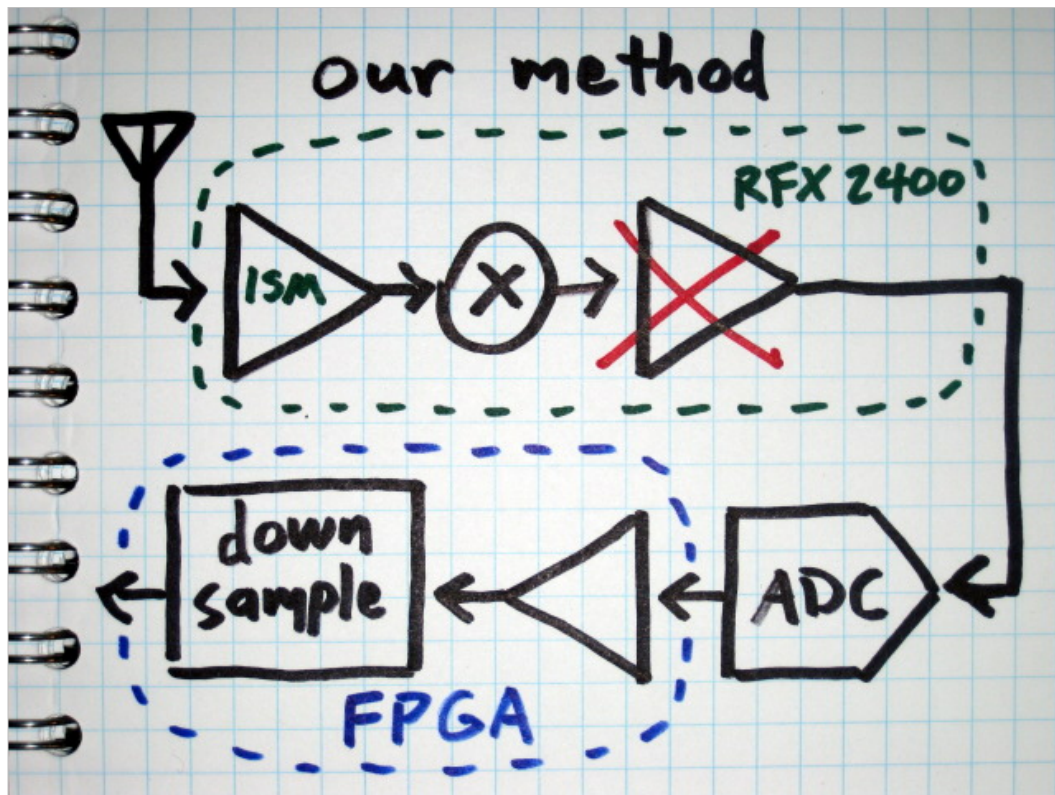
There is a reason not to use the aliasing technique any more than we have to, however. Every time we increase the number of overlapping aliases, we double up the bad along with the good. If there is interference, say from an 802.11 network, on a particular channel, the aliases of the interfering signal can affect multiple channels in the digital domain. In the extreme case, if we alias all Bluetooth channels onto a single digital channel, it only takes one interferer on any single channel to ruin our ability to receive transmissions on every channel (which is exactly the situation that frequency hopping is intended to avoid). The 2.4 GHz ISM band tends to be busy, so this problem is very likely. Even without the presence of interference, each alias adds more background noise, reducing our signal to noise ratio, thereby increasing the likelihood of decoding errors. If we want to maximize our ability to correctly decode Bluetooth transmissions, especially if we want to monitor more than one piconet at once, we should use the aliasing technique as little as possible.



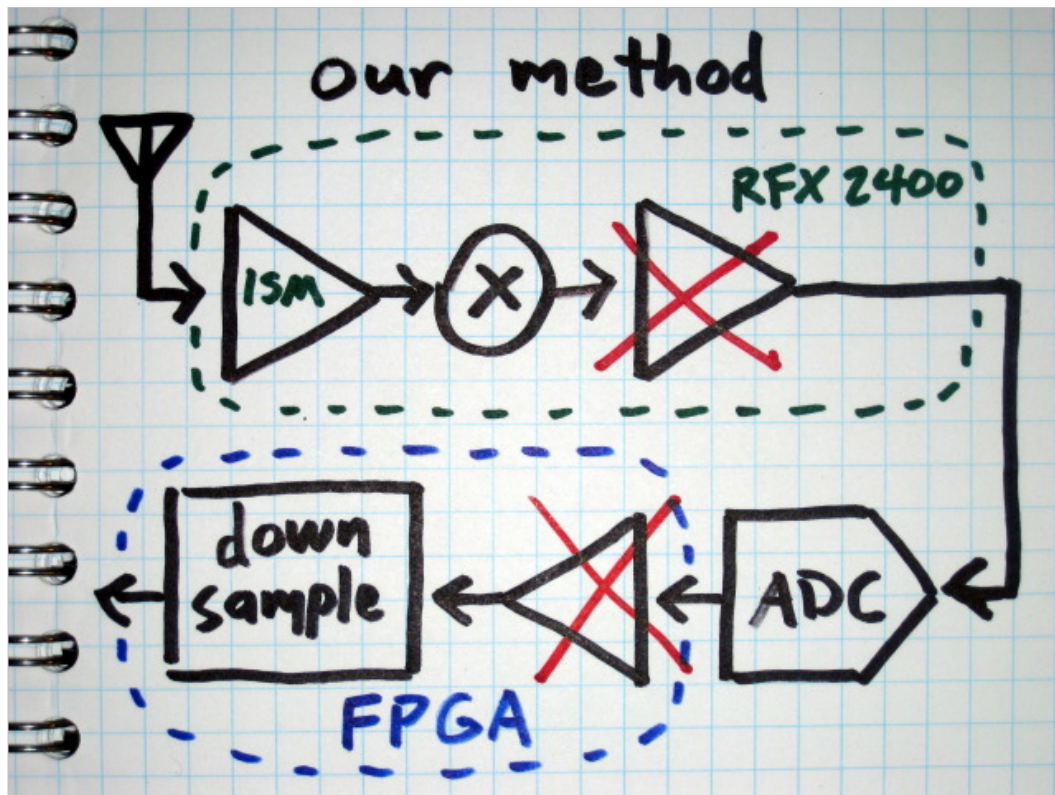
Our Method

This example of an 8 channel Bluetooth piconet is a simplified version of our real world challenge. Our goal is to monitor 79 channels operating at a much higher frequency. Our principal limitation is that our fastest software radio system, the USRP2, is capable of delivering a maximum bandwidth of 25 MHz to the host computer. That means that our software can see only 25 out of 79 channels unambiguously. By configuring the USRP2 to alias four 25 MHz bands on top of each other, however, we can deliver all 79 channels to the host computer with each of the 25 distinct channels in the digital domain carrying 3 or 4 aliased channels from the analog domain.

This requires two modifications:



The RFX2400 daughterboard is modified by removing the 20 MHz analog anti-aliasing filter from the receive path.

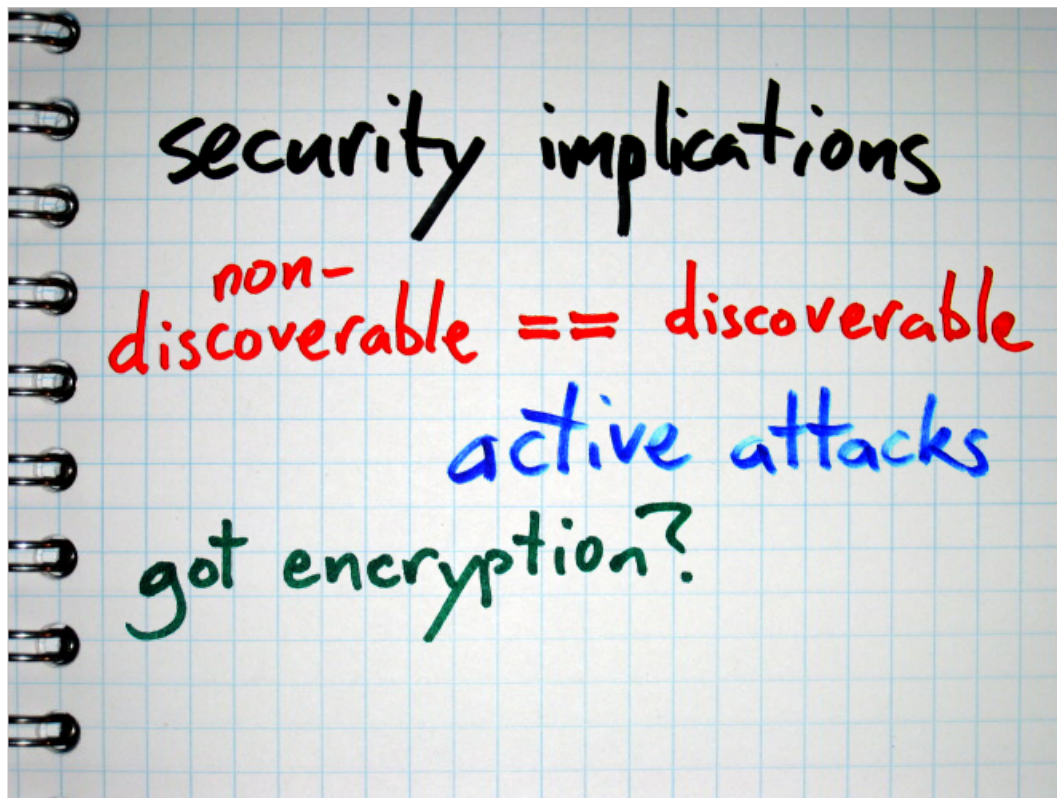


The USRP2 FPGA code is modified by zeroing some coefficients in the halfband decimators (only effective when using a decimation rate of 4).



Demo: Aliased Hopping

`multibtrx.py -pa2d 4 -i alias1.cfile -f 2440e6 -l 24d952`
(or similar)

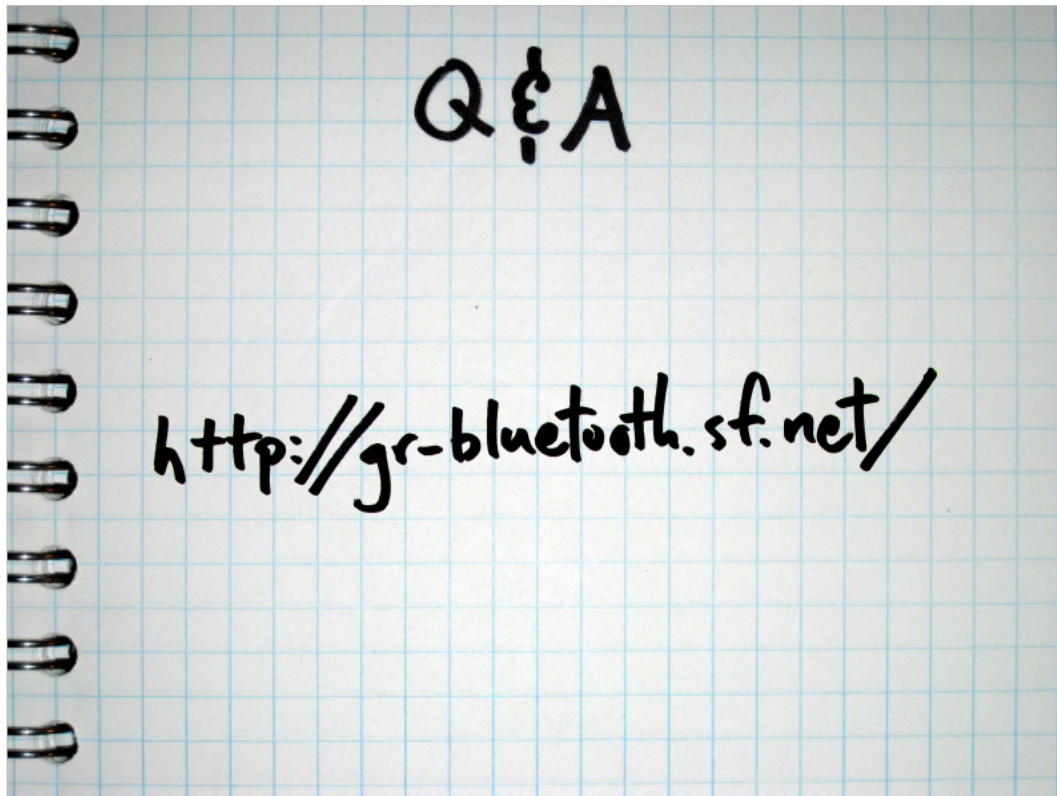


Security Implications

Non-discoverable devices are just as easily monitored as discoverable devices

Knowledge of the clock (thus hopping sequence and whitening) opens the door to active attacks

Got encryption? Most devices don't.



Q&A

The gr-bluetooth project (our code) is located at
<http://gr-bluetooth.sf.net/>

