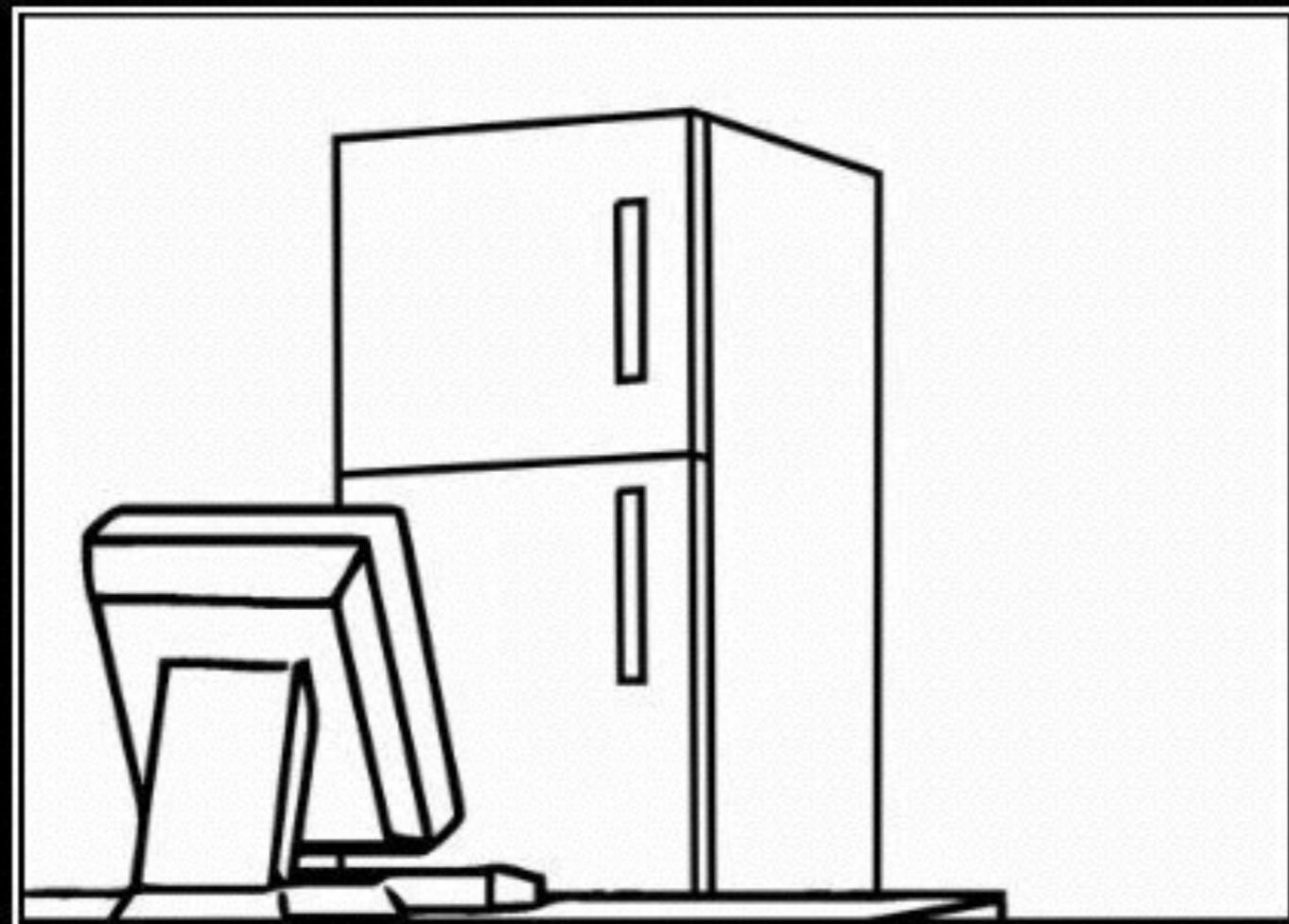


“Wait, what?”

A Practical Introduction to the Internet of Things



ON THE INTERNET OF THINGS

NOBODY KNOWS YOU'RE A FRIDGE

Gareth
@gyaresu

Tasmania

Go away, you wouldn't like it.



Imagery ©2015 Data SIO, NOAA, U.S. Navy, NGA, GEBCO, Landsat, U.S. Geological Survey, PGC/NASA, Map data ©2015 Google



I wore that head in the 5th film (jah, srsly)



“He who shall not answer your stupid questions”







Greenpeace International

“A beige building in Amsterdam”

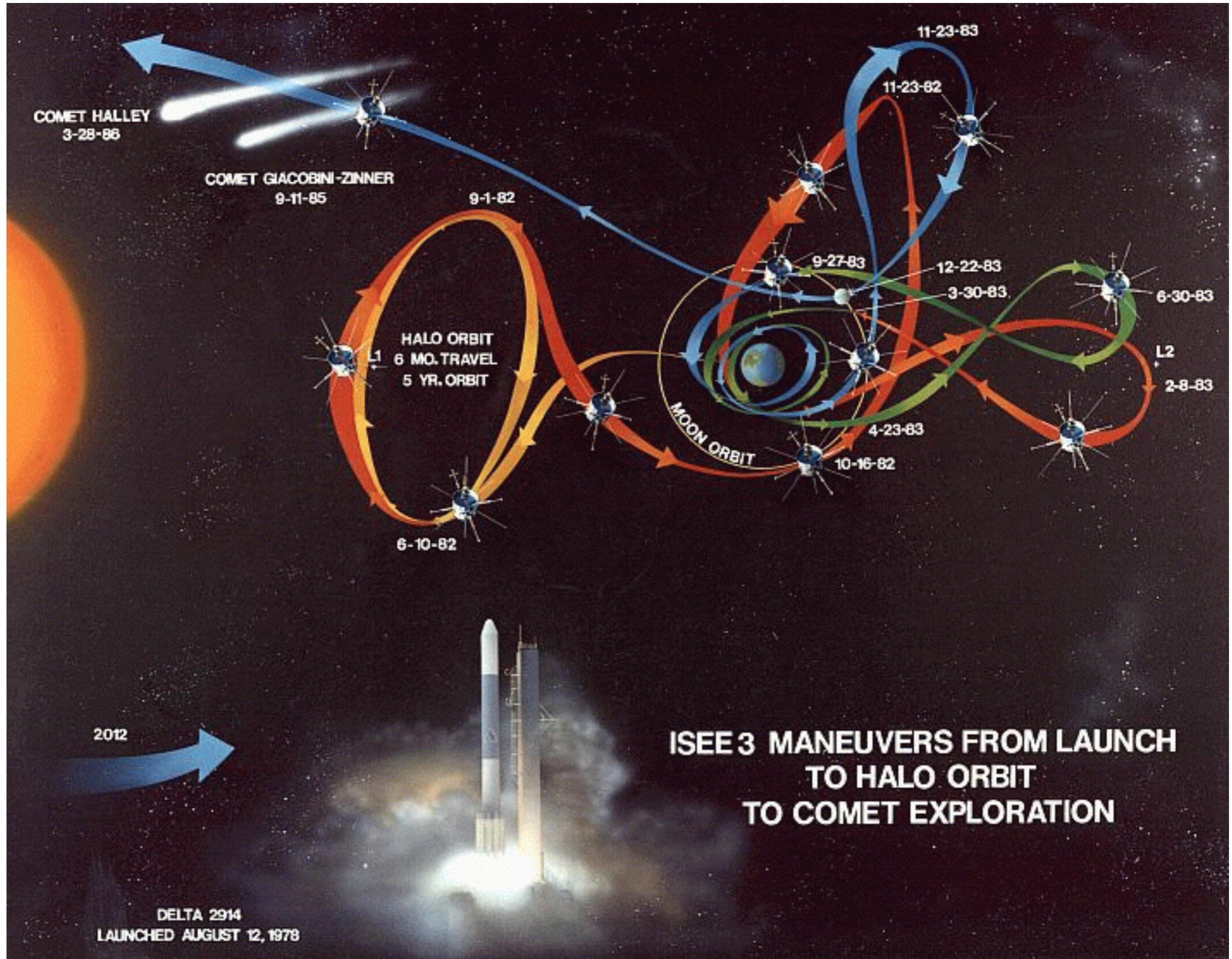


Computer Security Researcher





NASA 50-000618



ICEE-3 Reboot Project

Partnership: Ovation TV & RocketHub Team Up [Find Out More](#) X

 **ROCKETHUB**
An EFactor Group Company

Success School Our Movement News Join Login 🔍

ISEE-3 Reboot Project by Space College, Skycorp, and SpaceRef

Description About Conversations Activity



A large image showing three men in yellow hard hats and casual clothing standing outdoors on what appears to be a construction or industrial site. They are all smiling and giving a thumbs-up gesture towards the camera. The background shows various metal structures, pipes, and a view of trees and hills in the distance.

128% **\$159,602**
Raised towards \$125,000 goal

2238 Funders Ended 05/23/14

Complete!

PROJECT COMPLETE

Funders

    [View All](#)

Goods

Fund \$10 and more **Project Supporter**





Puerto Rico

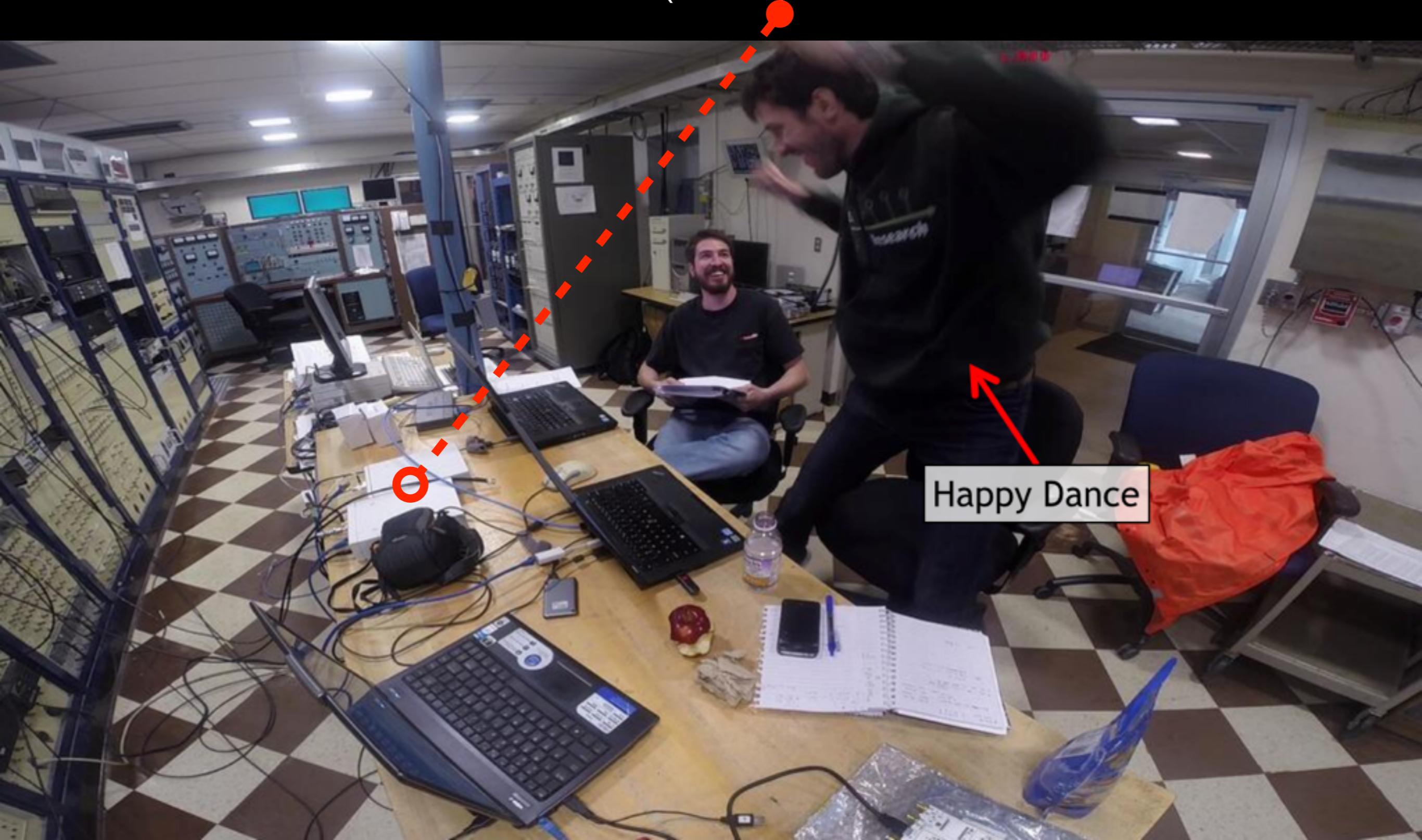
Arecibo Observatory
Observatory & national
research center



Arecibo Radio Telescope

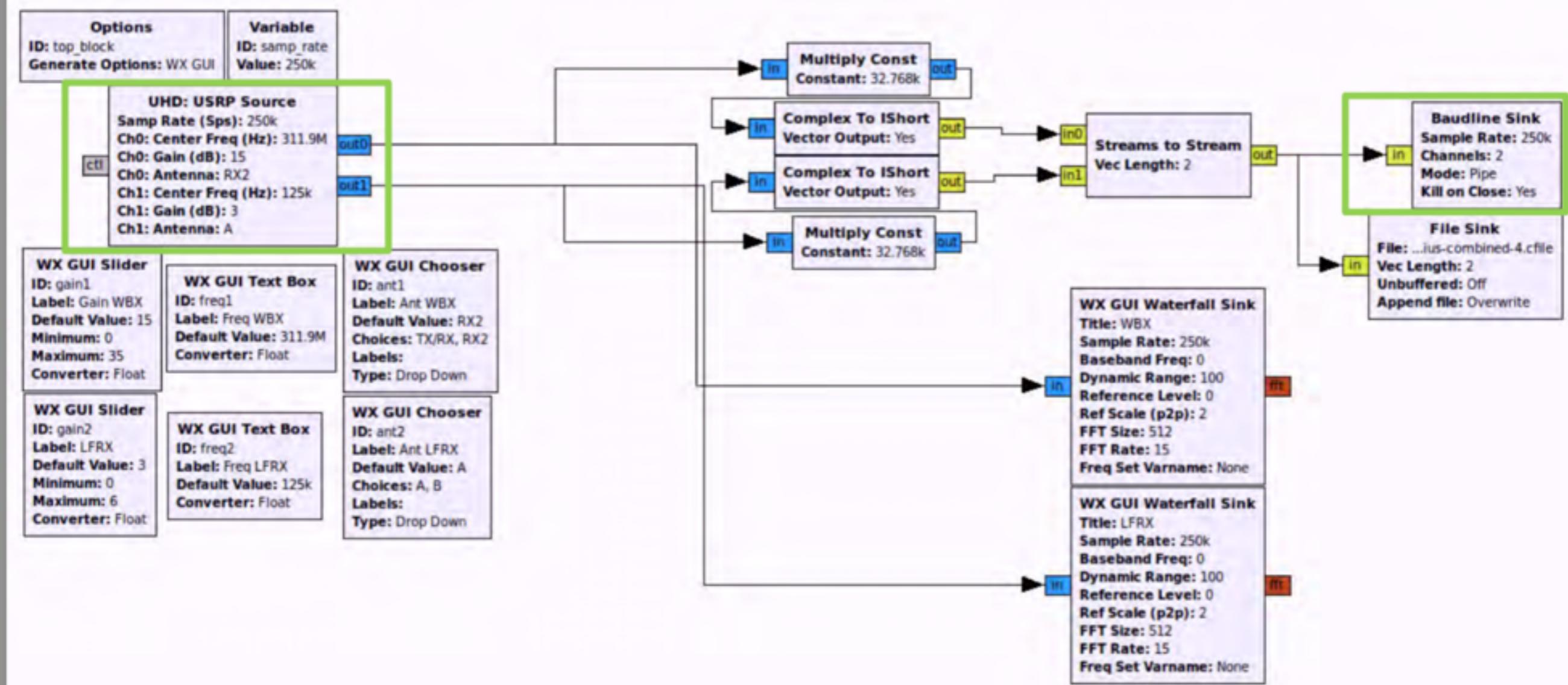


Software Defined Radios (SDR)



GNU Radio looks wacky but is totally boss

GNU Radio + baudline



“RTLSDR”
Realtek RTL2832U chip
Used in DVB-T Dongles (5yrs)



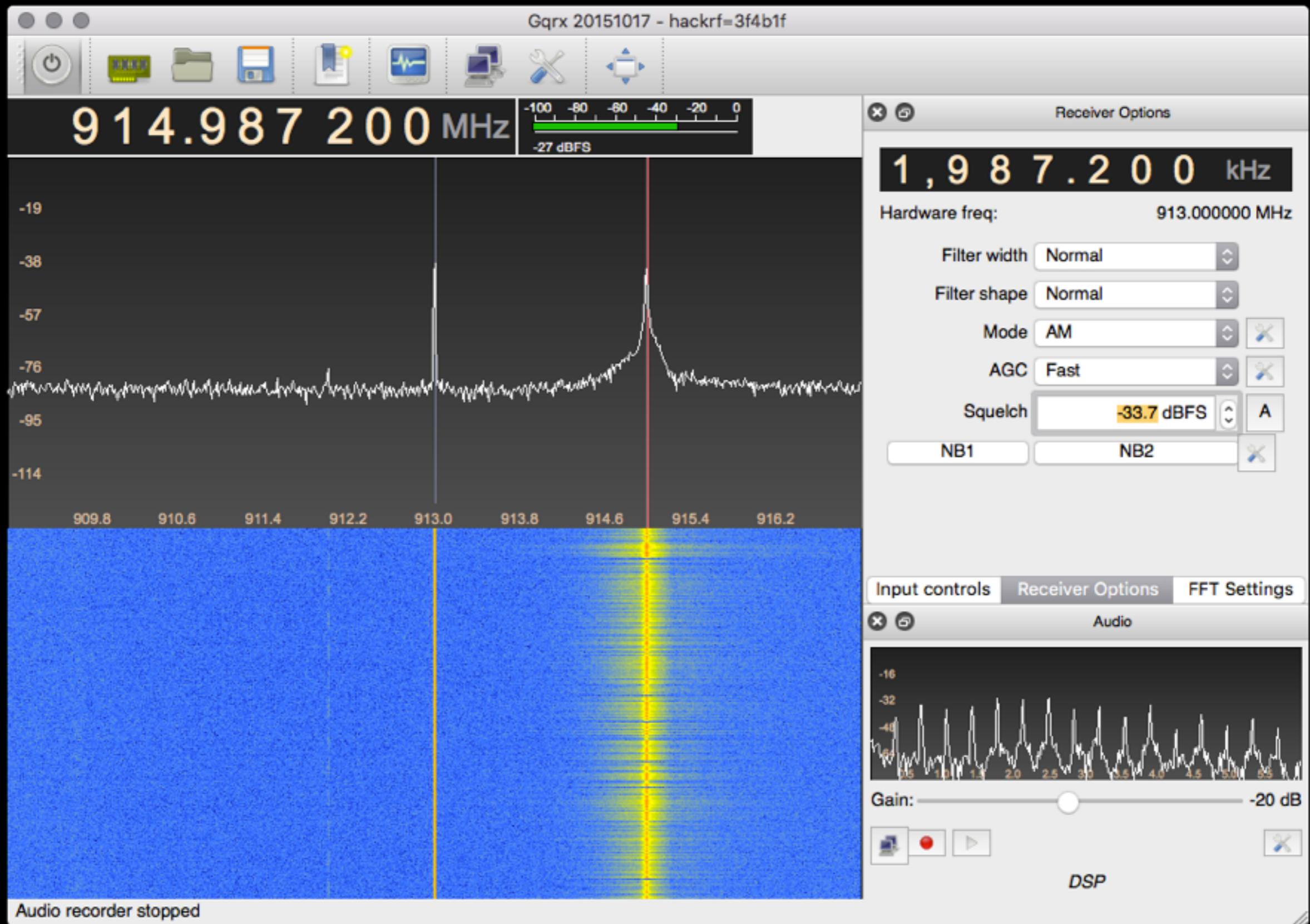
HackRF One

0 - 6GHz (\$300)



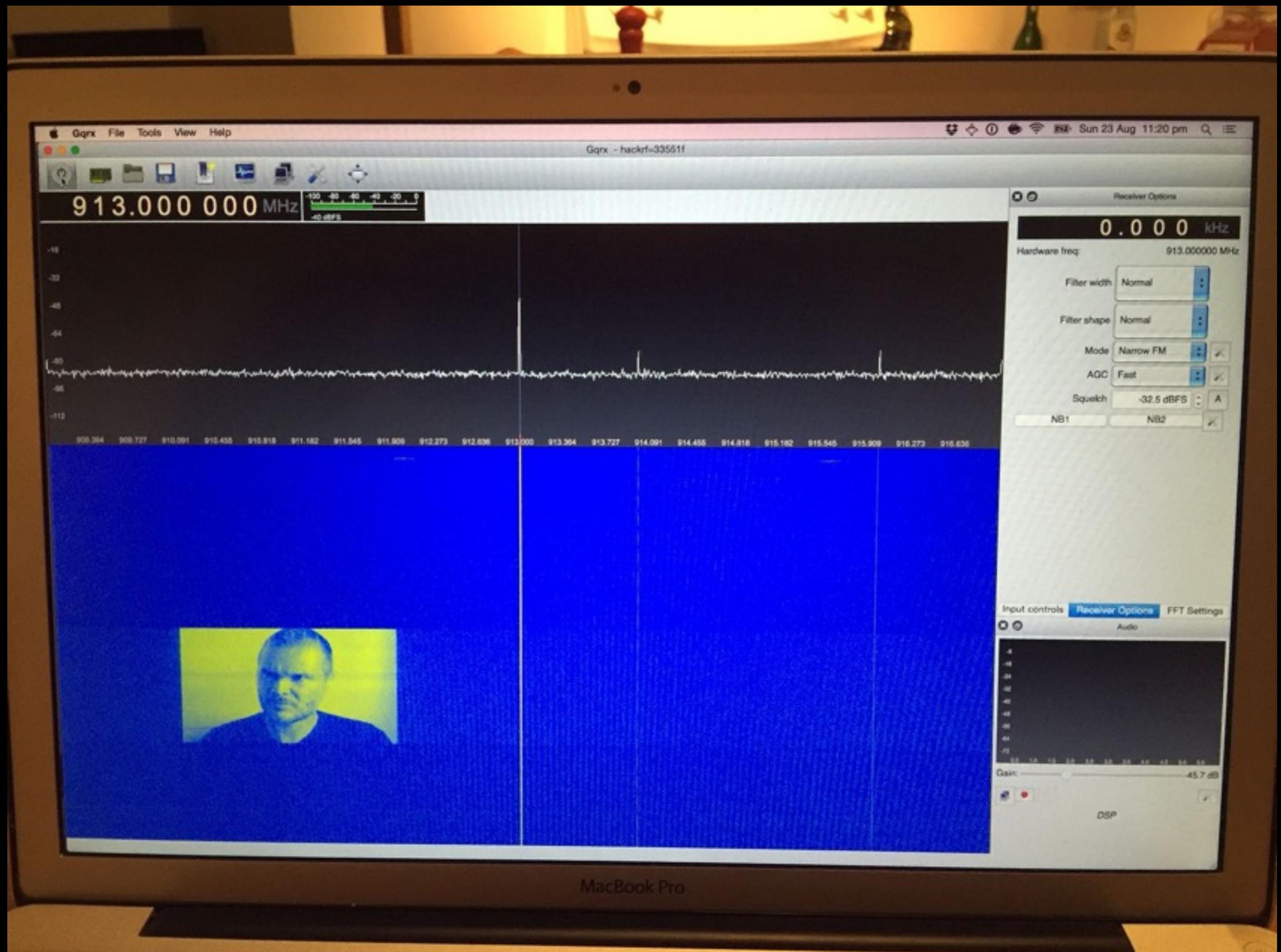
<http://greatscottgadgets.com/sdr/>

GQRX

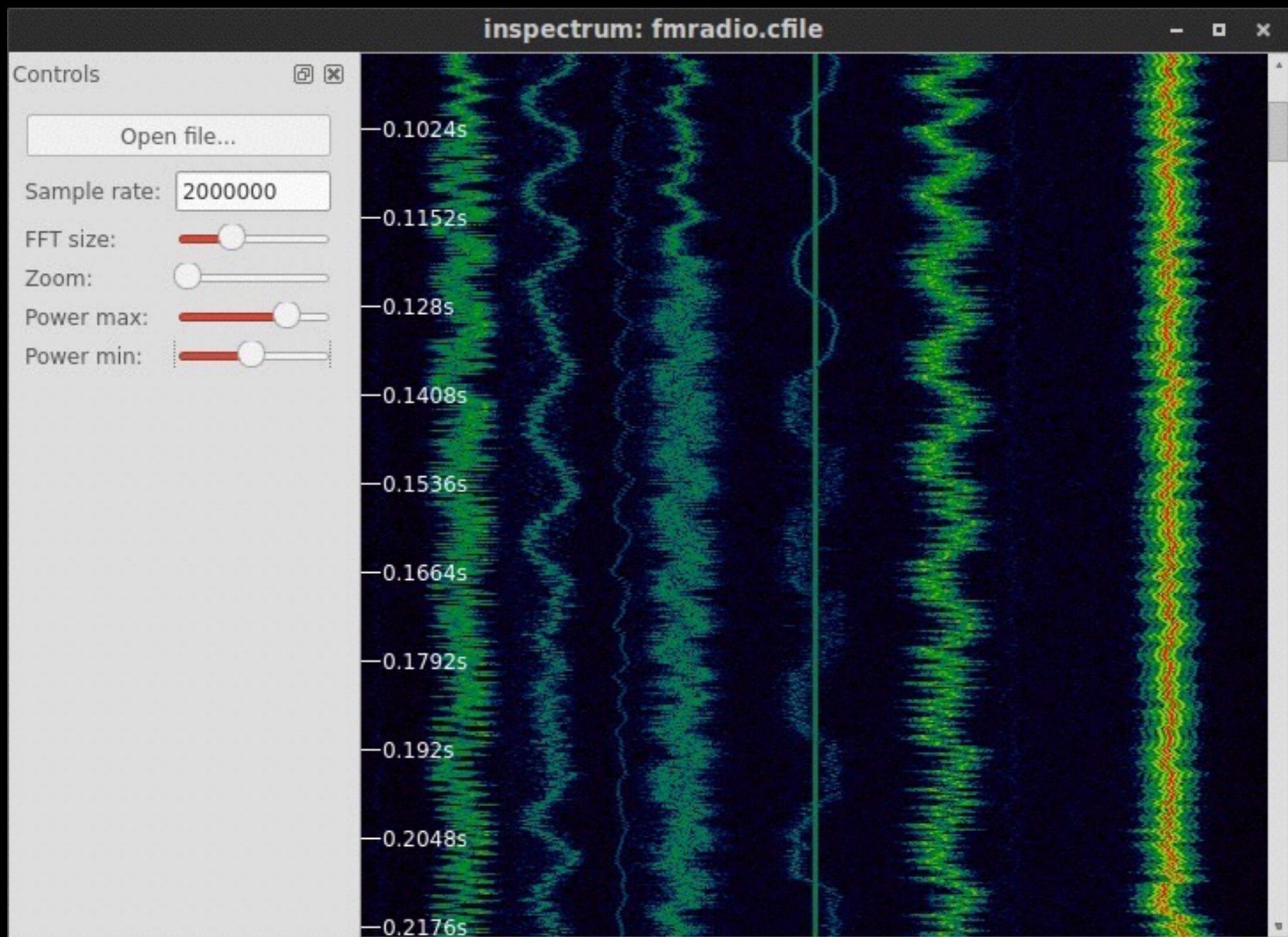


Spectrum Painter

polygon/spectrum_painter



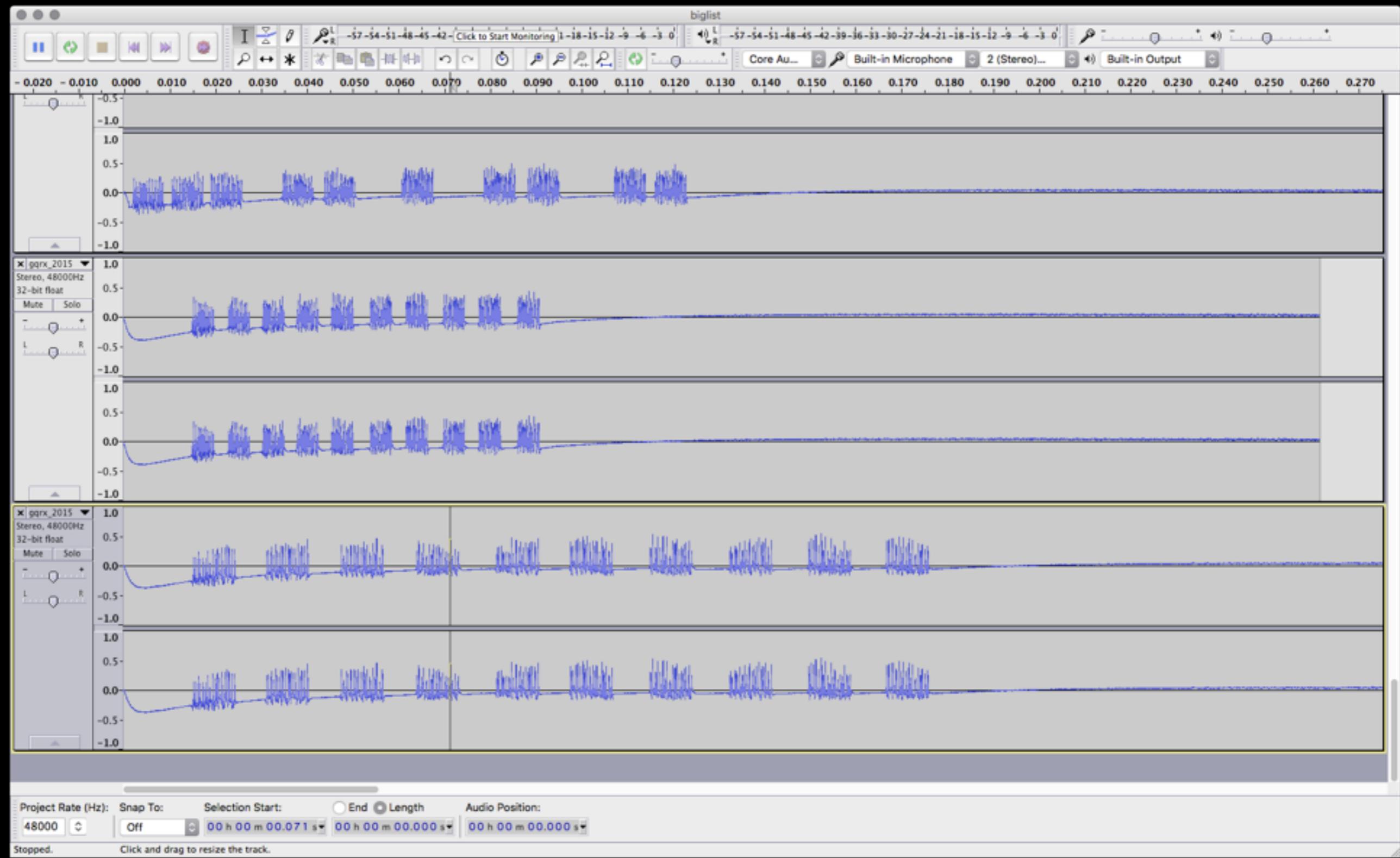
miek/inspectrum



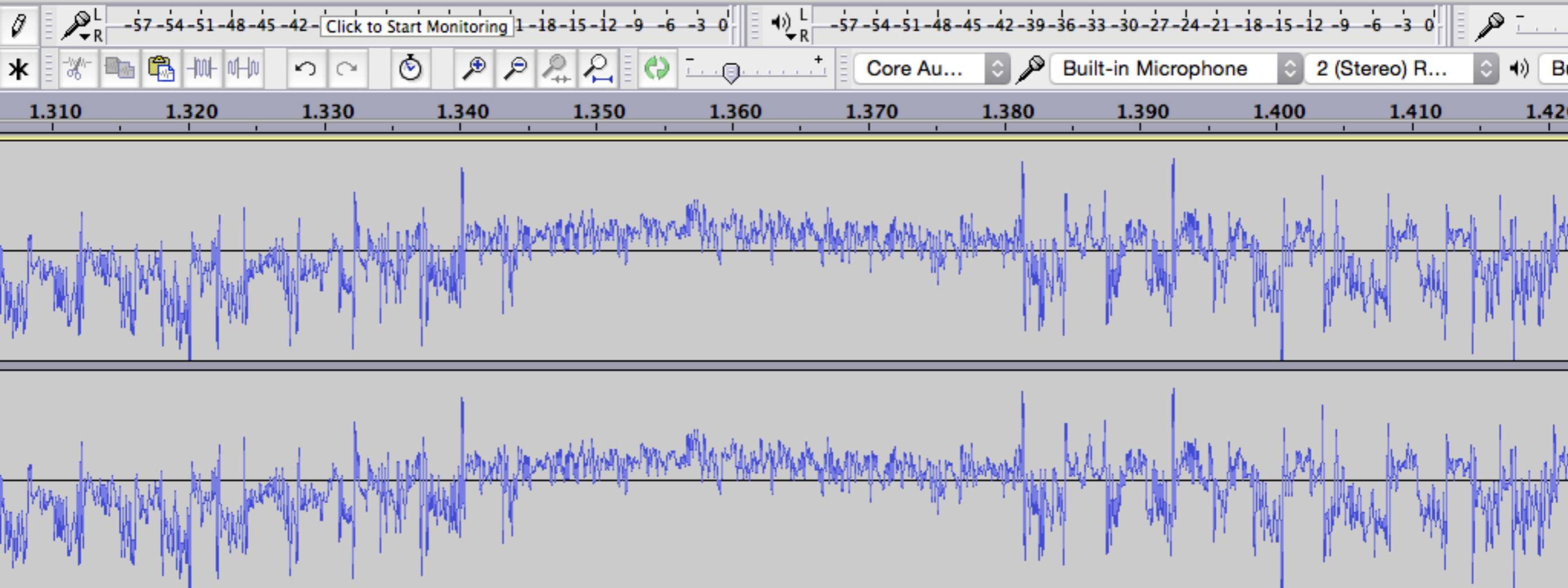
\$ hackrf_transfer

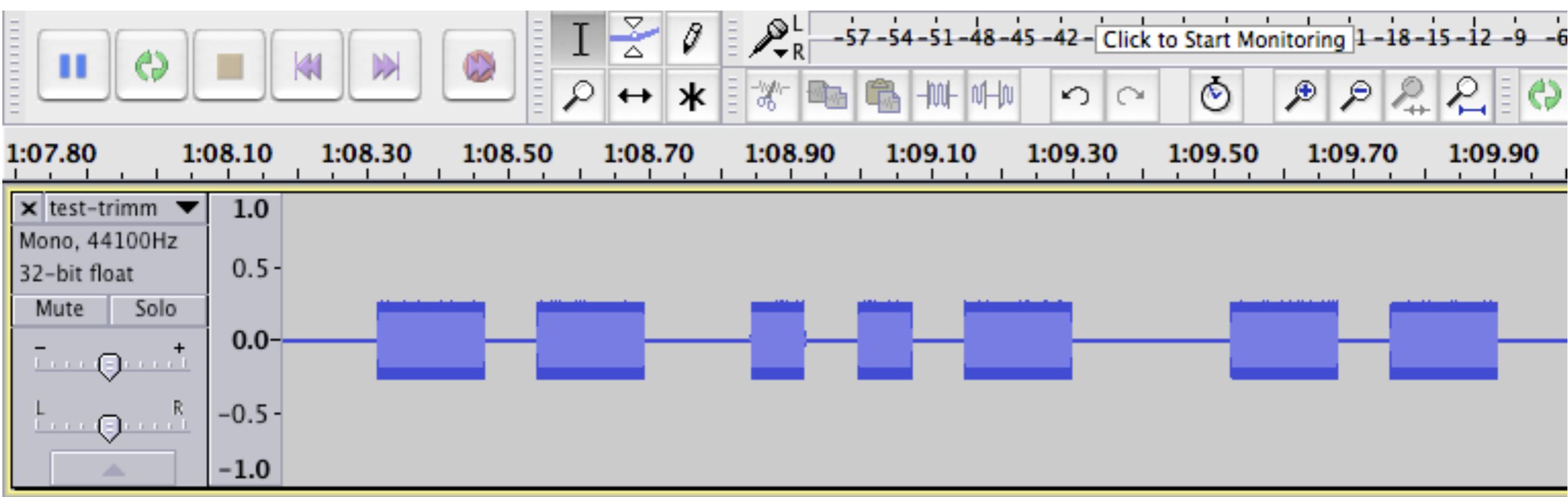
```
tmux (tmux) 301 ~ (zsh) 302 1. TERM=screen-256color-bce tmux attach (tmux)
7732 hackrf_transfer -r 391MHz-8M-8bit.iq -f 391000000 -s 8000000
7733 hackrf_transfer -r 392MHz-8M-8bit.iq -f 392000000 -s 8000000
7746 vi transfer-bits.js
7748 node transfer-bits.js
7900 hackrf_transfer -h
7901 hackrf_transfer -c 127 -f 850000000
7902 hackrf_transfer
7903 hackrf_transfer -R -c -f 870000000
7904 hackrf_transfer -c -f 870000000
7905 hackrf_transfer -R -c 127 -f 870000000
7906 hackrf_transfer -R -c 127 -f 870000000 -b 1
7907 hackrf_transfer -R -c 127 -f 870000000 -b 2
7908 hackrf_transfer -R -c 127 -f 870000000 -b 3
7909 hackrf_transfer -R -c 127 -f 870000000 -b 1000
7910 hackrf_transfer -R -c 127 -f 870000000 -b 1000000
7911 hackrf_transfer -R -c 127 -f 870000000 -b 2000000
7912 hackrf_transfer -R -c 127 -f 870000000 -b 8000000
7913 hackrf_transfer -R -c 127 -f 870000000 -b 1750000
7920 hackrf_transfer -R -c 127 -f 915000000 -b 1750000
7921 hackrf_transfer -R -x 40 -f 915000000 -b 1750000
7922 hackrf_transfer -c 100 -f 915000000 -b 1750000
9655 hackrf_transfer -r test-431MHz-8M-8bit.iq -f 431000000 -s 8000000
9737 hackrf_transfer -r test-390MHz-8M-8bit.iq -f 390000000 -s 8000000
9739 hackrf_transfer -r test-389MHz-8M-8bit.iq -f 389000000 -s 8000000
9751 hackrf_transfer -r test-912MHz-8M-8bit.iq -f 912000000 -s 8000000
gyaresu@zaphod:~/programming|⇒
```

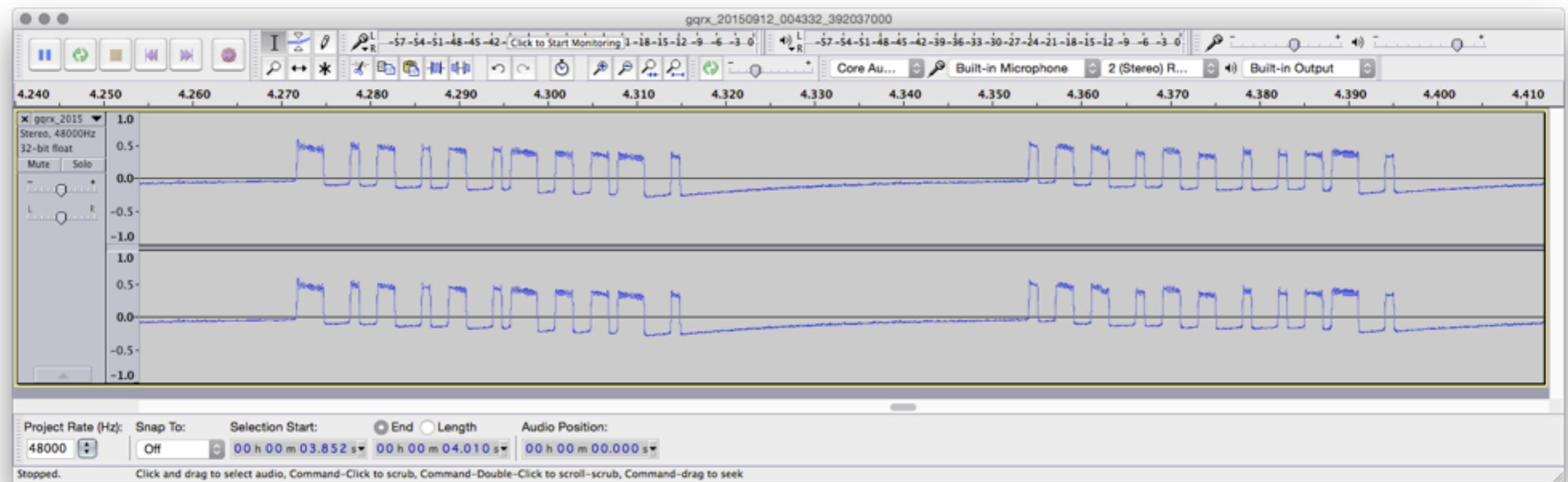
Audacity



gqrx_20150911_202358_390000000

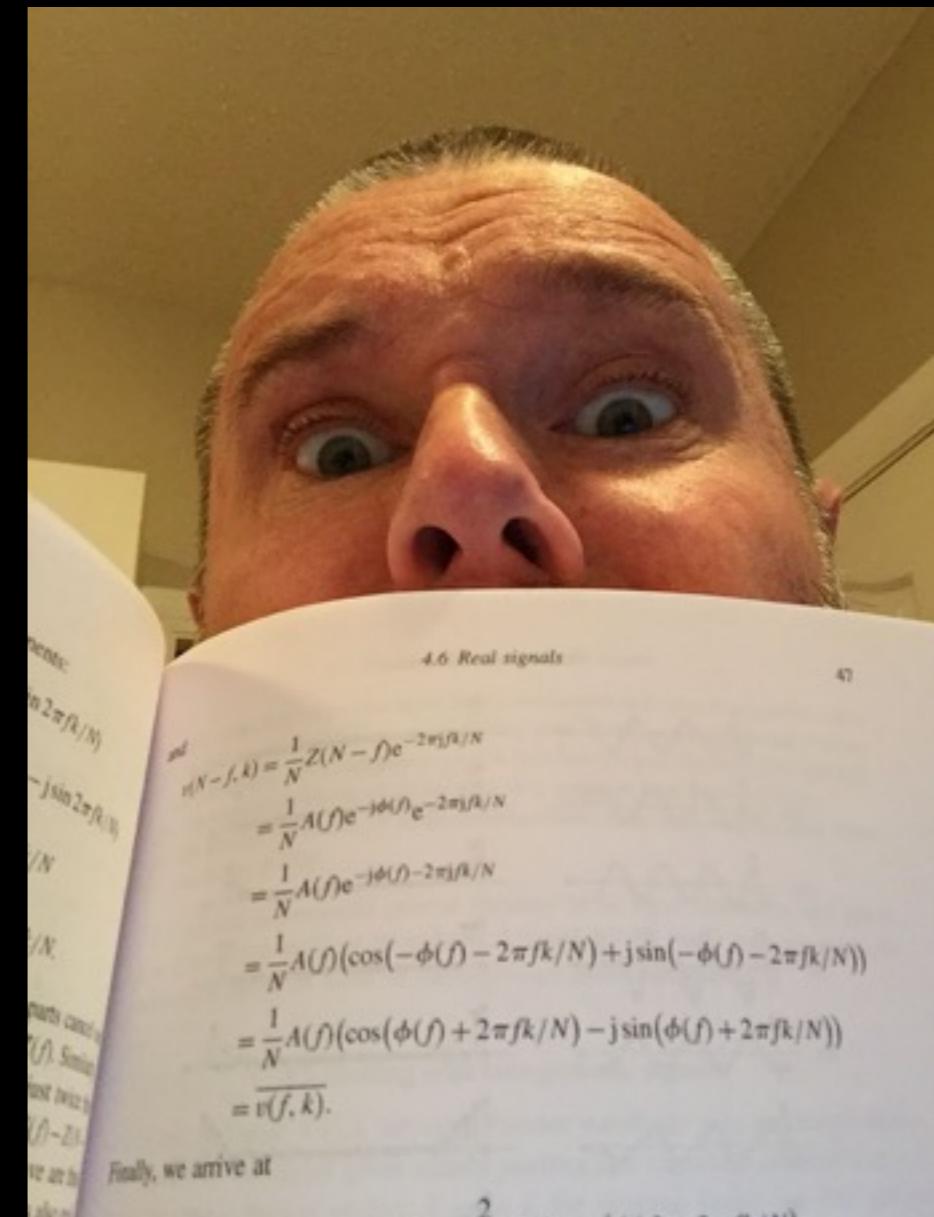
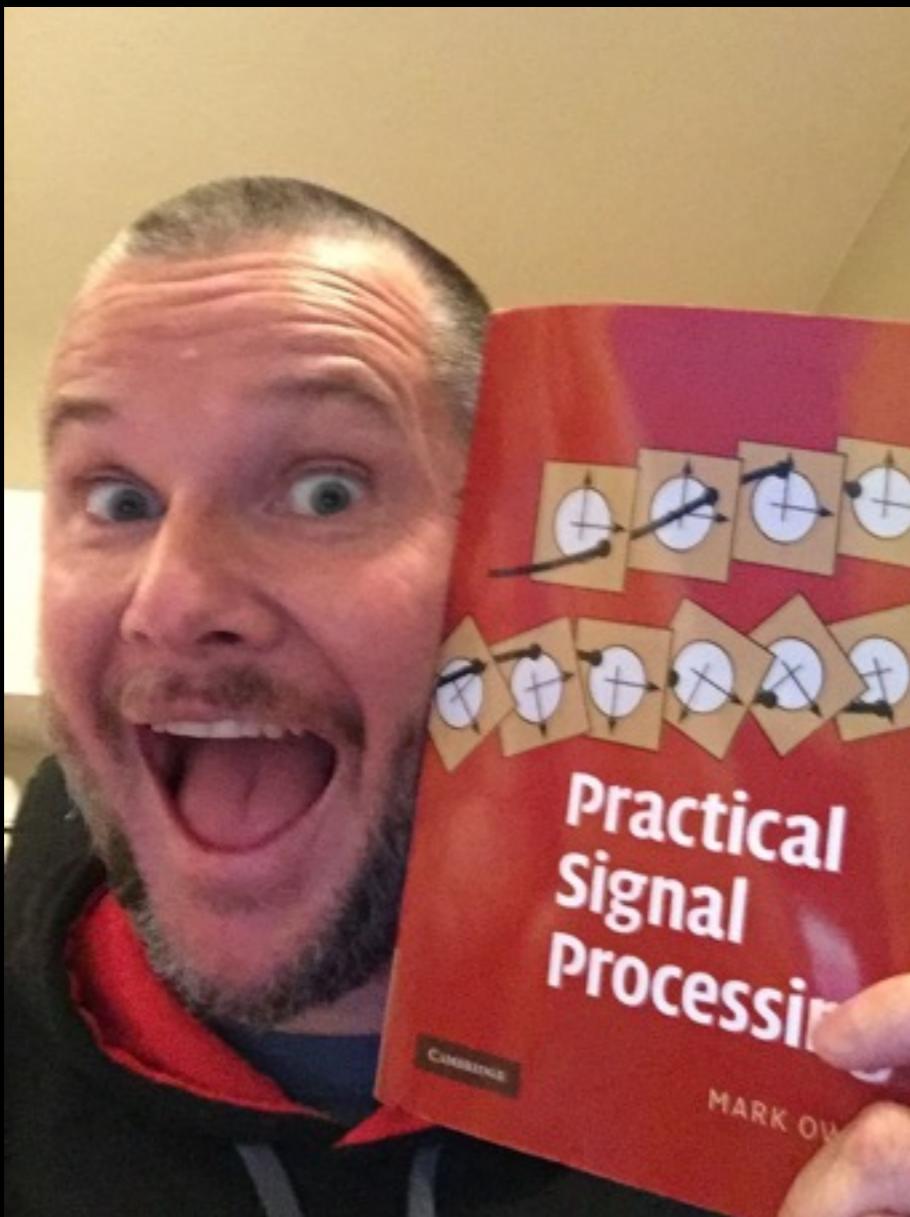






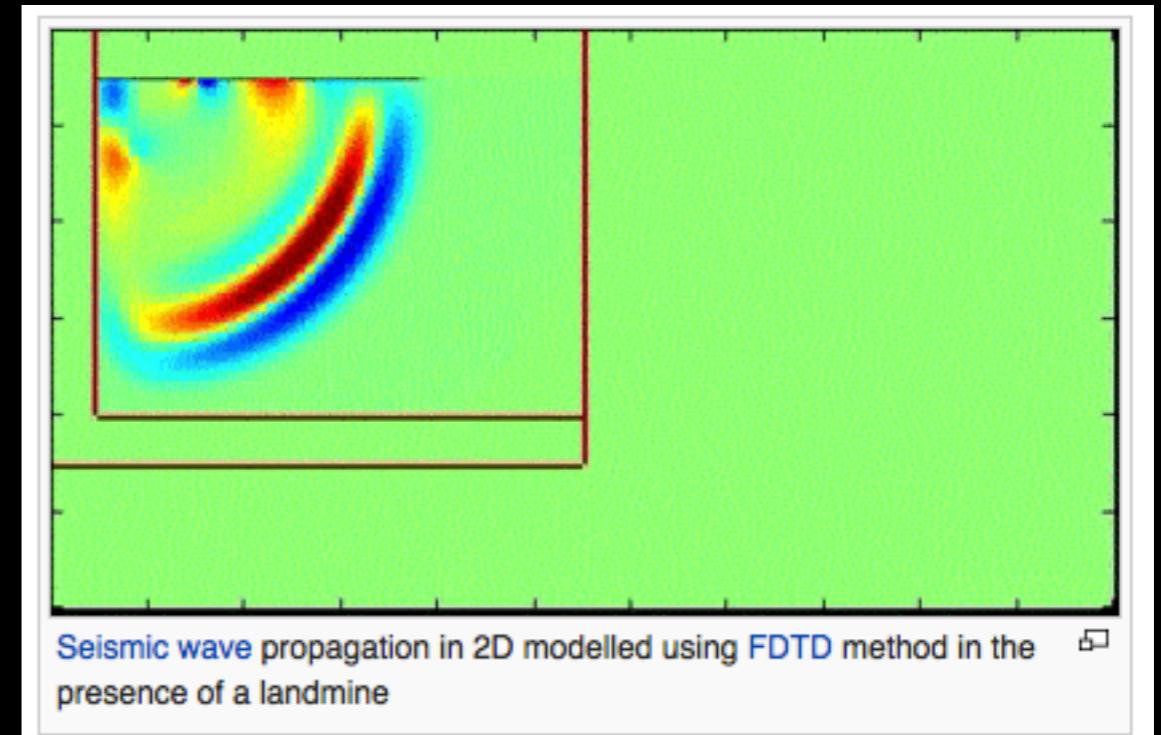
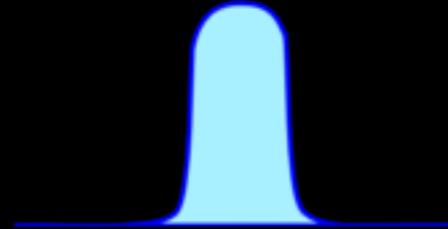
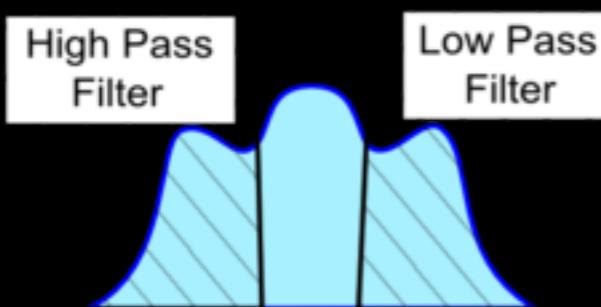
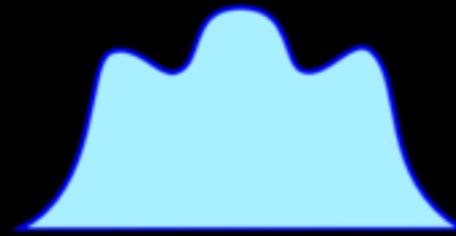
What's a signal?

Practical Signal Processing - Mark Owen

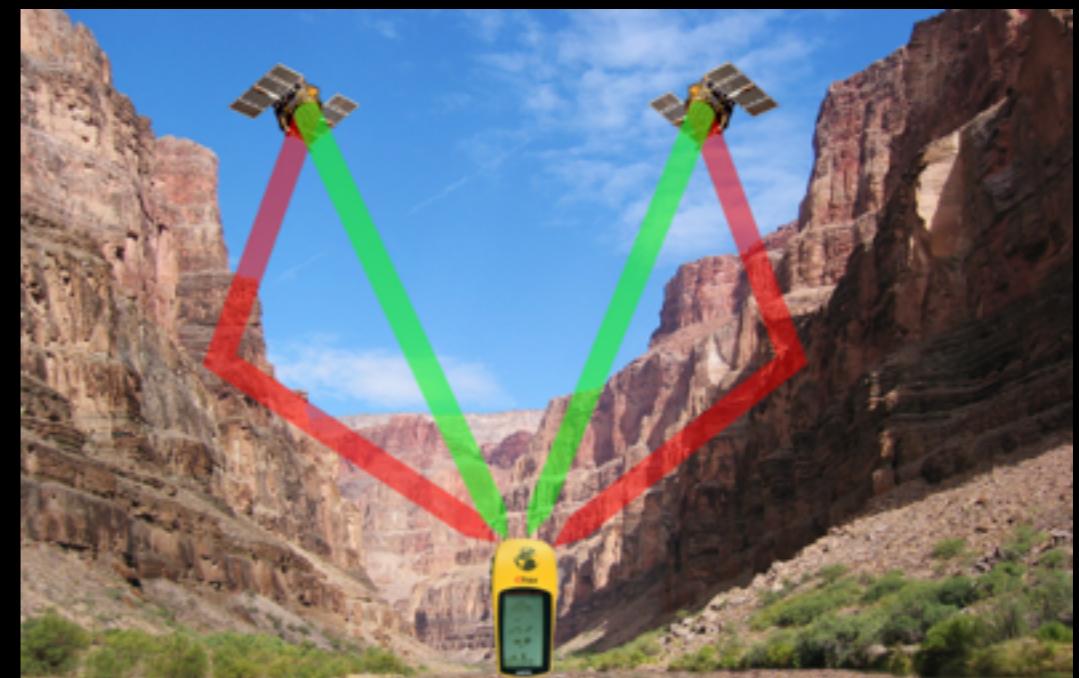


Wave Propogation

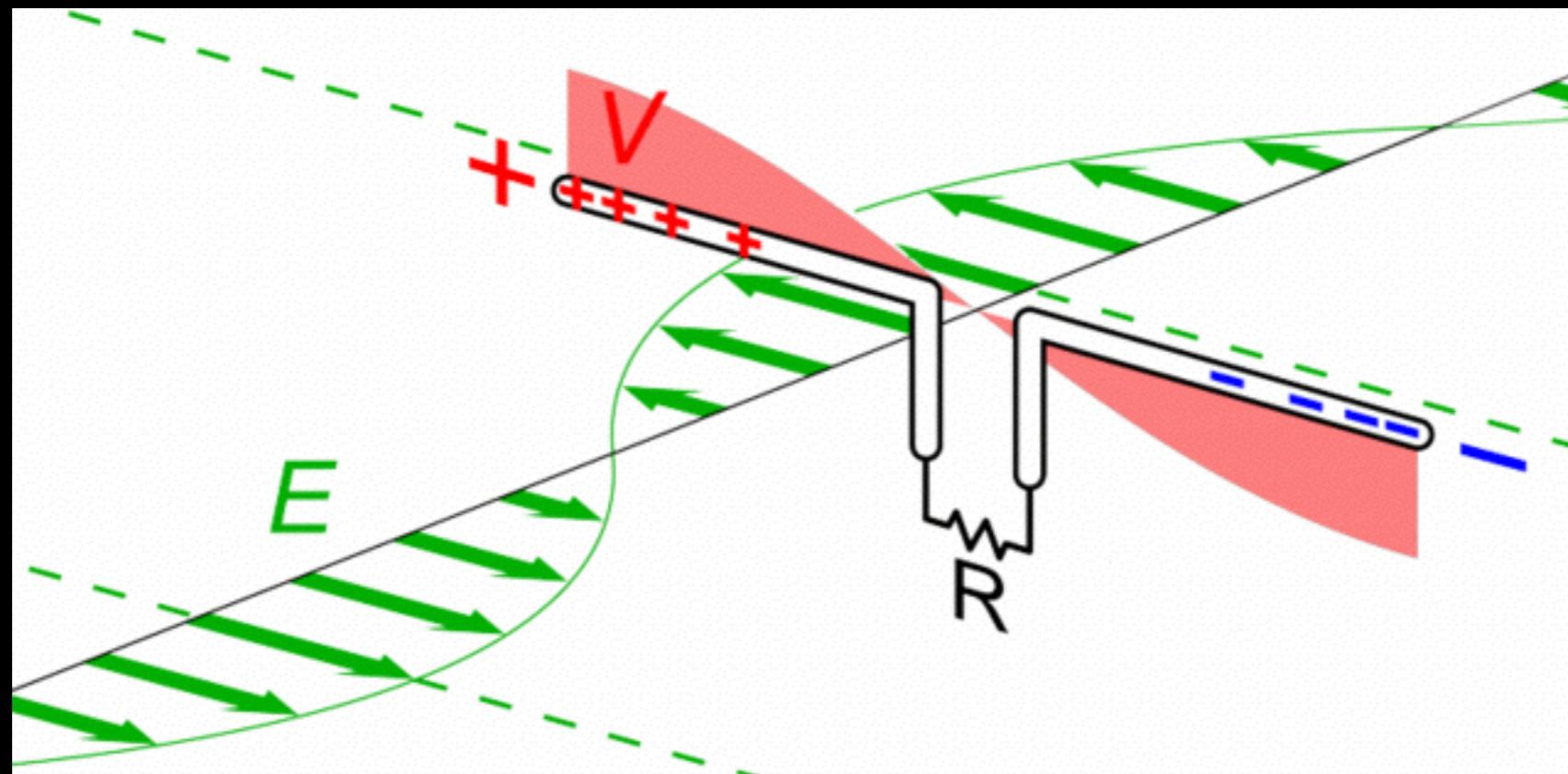
Passband



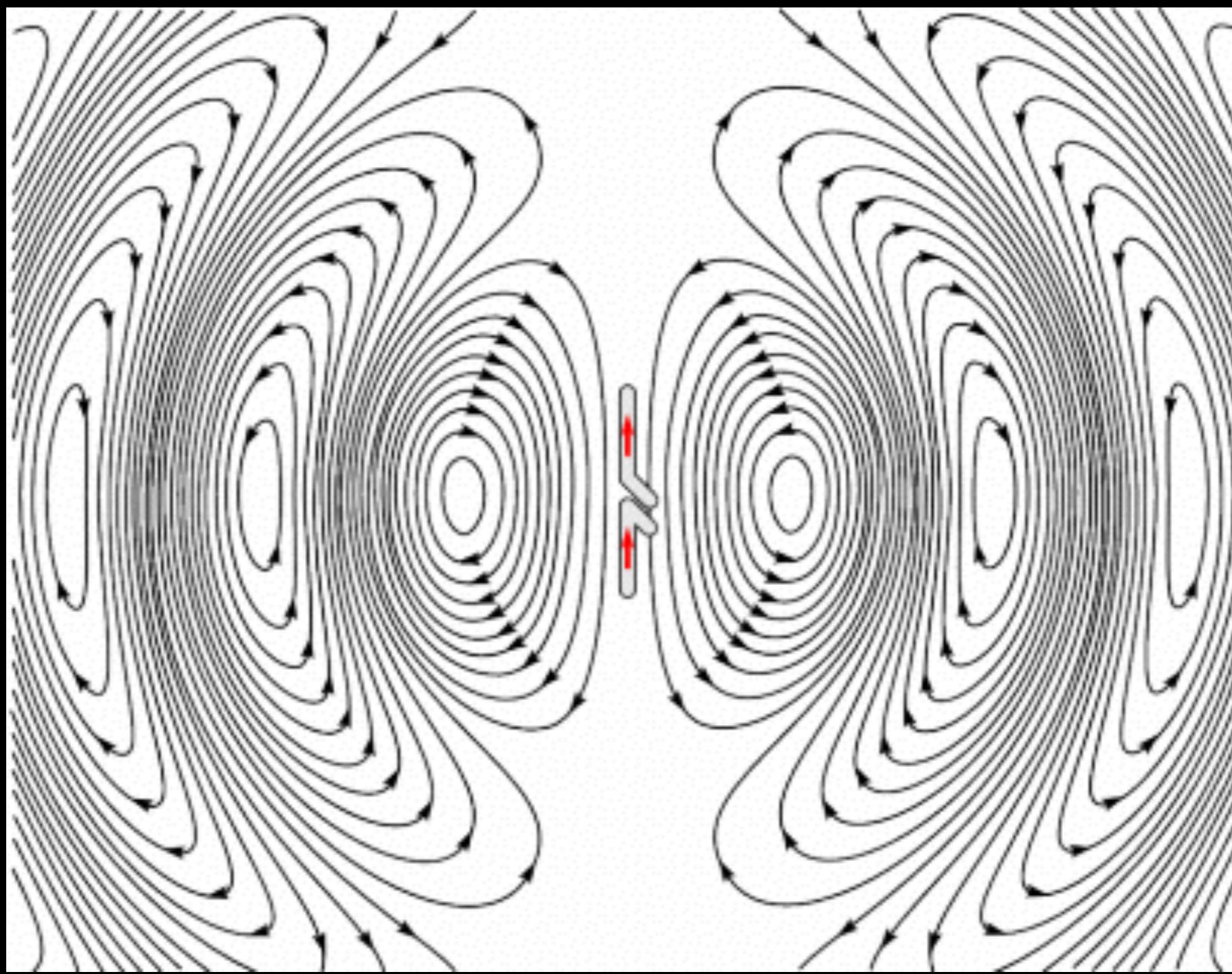
“Multipath effect”



Antennas are weird



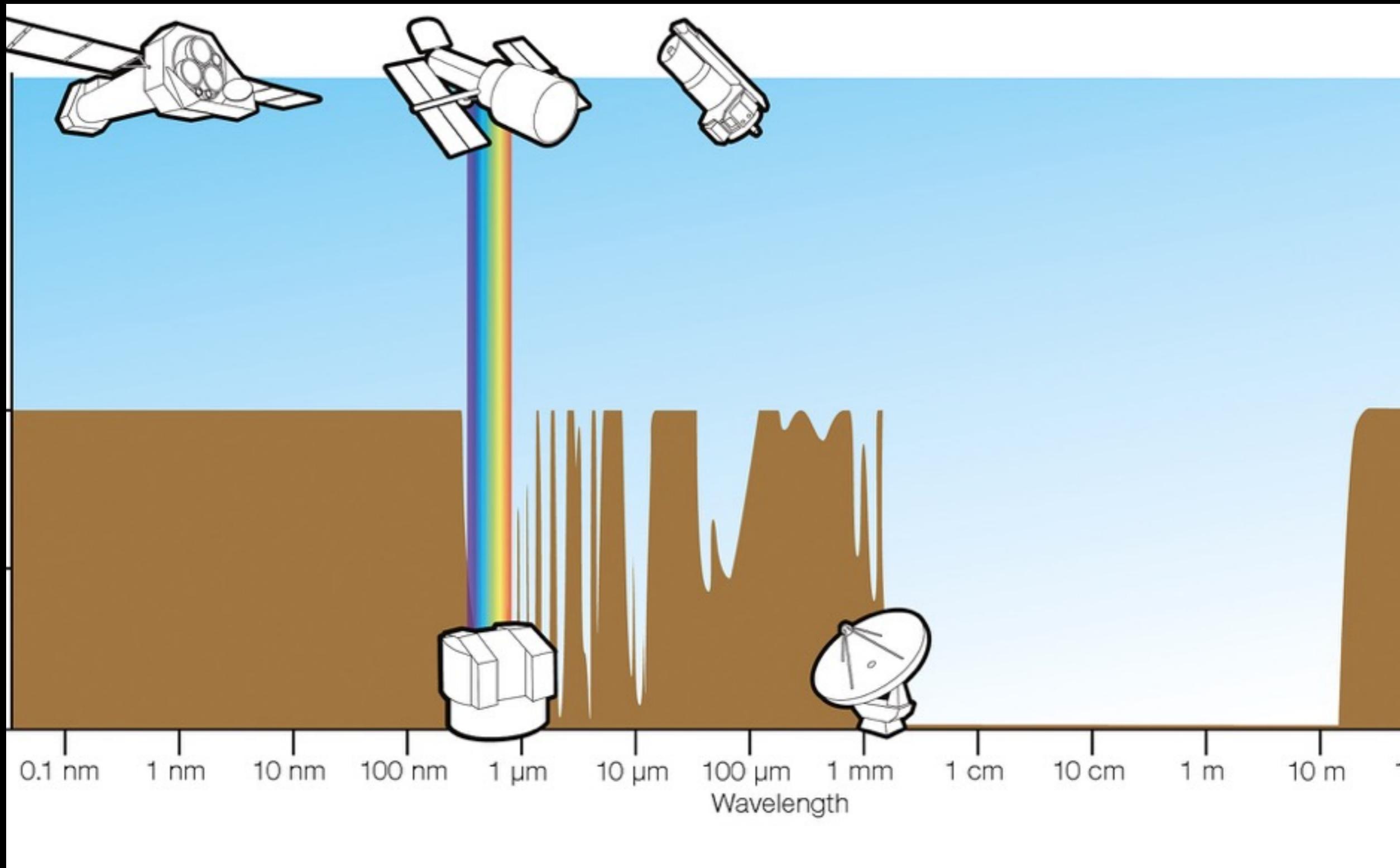
Woah



high frequency == short wavelength

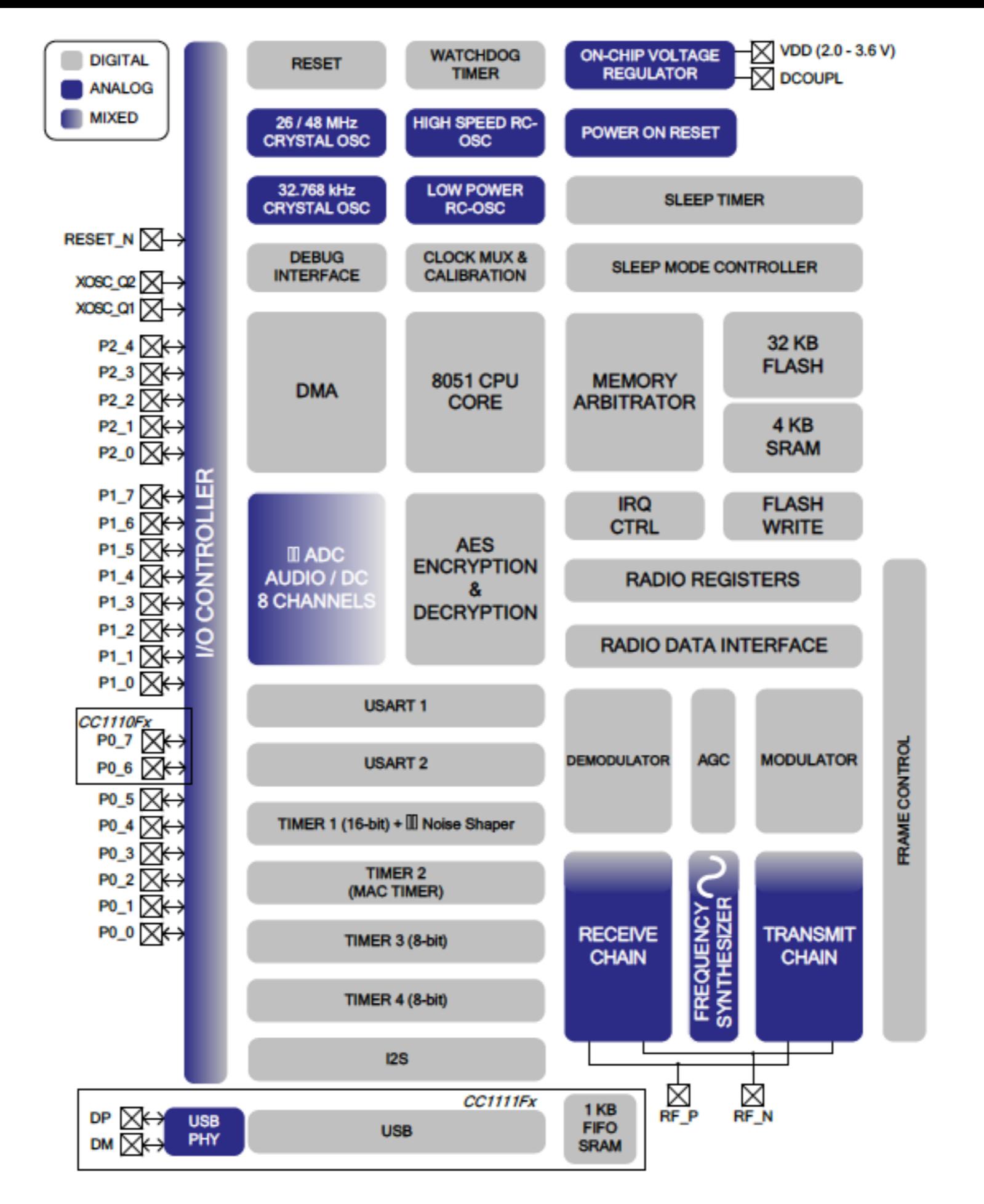
low frequency == long wavelength

300MHz == 1m / 30MHz == 10m





CC11xx



	Write		Read		
	Single Byte	Burst	Single Byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x00			IOCFG2		
0x01			IOCFG1		
0x02			IOCFG0		
0x03			FIFOTHR		
0x04			SYNC1		
0x05			SYNC0		
0x06			PKTLEN		
0x07			PKTCTRL1		
0x08			PKTCTRL0		
0x09			ADDR		
0x0A			CHANNR		
0x0B			FSCTRL1		
0x0C			FSCTRL0		
0x0D			FREQ2		
0x0E			FREQ1		
0x0F			FREQ0		
0x10			MDMCFG4		
0x11			MDMCFG3		
0x12			MDMCFG2		
0x13			MDMCFG1		
0x14			MDMCFG0		
0x15			DEVIATN		
0x16			MCSM2		
0x17			MCSM1		
0x18			MCSM0		
0x19			FOCCFG		
0x1A			BSCFG		
0x1B			AGCCTRL2		
0x1C			AGCCTRL1		
0x1D			AGCCTRL0		
0x1E			WOREVT1		
0x1F			WOREVT0		
0x20			WORCTRL		
0x21			FREND1		
0x22			FREND0		
0x23			FSCAL3		
0x24			FSCAL2		
0x25			FSCAL1		
0x26			FSCAL0		
0x27			RCCTRL1		
0x28			RCCTRL0		
0x29			FTEST		
0x2A			PTEST		
0x2B			AGCTEST		
0x2C			TEST2		
0x2D			TEST1		
0x2E			TEST0		
0x2F					
0x30	SRES		SRES		PARTNUM
0x31	SFSTXON		SFSTXON		VERSION
0x32	SXOFF		SXOFF		FREQEST
0x33	SCAL		SCAL		LQI
0x34	SRX		SRX		RSSI
0x35	STX		STX		MARCSTATE
0x36	SIDLE		SIDLE		WORTIME1
0x37					WORTIME0
0x38	SWOR		SWOR		PKTSTATUS
0x39	SPWD		SPWD		VCO_VC_DAC
0x3A	SFRX		SFRX		TXBYTES
0x3B	SFTX		SFTX		RXBYTES
0x3C	SWORRST		SWORRST		RCCTRL1_STATUS
0x3D	SNOP		SNOP		RCCTRL0_STATUS
0x3E	PATABL		PATABL		PATABL
0x3F	TX FIFO		TX FIFO		RX FIFO

R/W configuration registers, burst access possible
Command Strobes, Status registers (read only) and multi byte registers

15 Packet Handling Hardware Support

The **CC1101** has built-in hardware support for packet oriented radio protocols.

In transmit mode, the packet handler can be configured to add the following elements to the packet stored in the TX FIFO:

- A programmable number of preamble bytes
- A two byte synchronization (sync) word. Can be duplicated to give a 4-byte sync word (recommended). It is not possible to only insert preamble or only insert a sync word
- A CRC checksum computed over the data field.

The recommended setting is 4-byte preamble and 4-byte sync word, except for 500 kBaud data rate where the recommended preamble length is 8 bytes. In addition, the following can be implemented on the data field and the optional 2-byte CRC checksum:

- Whitening of the data with a PN9 sequence
- Forward Error Correction (FEC) by the use of interleaving and coding of the data (convolutional coding)

In receive mode, the packet handling support will de-construct the data packet by implementing the following (if enabled):

15.1 Data Whitening

From a radio perspective, the ideal over the air data are random and DC free. This results in

- Preamble detection
- Sync word detection
- CRC computation and CRC check
- One byte address check
- Packet length check (length byte checked against a programmable maximum length)
- De-whitening
- De-interleaving and decoding

Optionally, two status bytes (see Table 27 and Table 28) with RSSI value, Link Quality Indication, and CRC status can be appended in the RX FIFO.

Bit	Field Name	Description
7:0	RSSI	RSSI value

**Table 27: Received Packet Status Byte 1
(first byte appended after the data)**

Bit	Field Name	Description
7	CRC_OK	1: CRC for received data OK (or CRC disabled) 0: CRC error in received data
6:0	LQI	Indicating the link quality

**Table 28: Received Packet Status Byte 2
(second byte appended after the data)**

Note: Register fields that control the packet handling features should only be altered when **CC1101** is in the IDLE state.

With **CC1101**, this can be done automatically. By setting PKTCTRL0.WHITE DATA=1, all

Oh, ok... I can dig that.

Figure 18: Data Whitening in TX Mode

15.2 Packet Format

The format of the data packet can be configured and consists of the following items (see Figure 19):

- Preamble
- Synchronization word

- Optional length byte
- Optional address byte
- Payload
- Optional 2 byte CRC

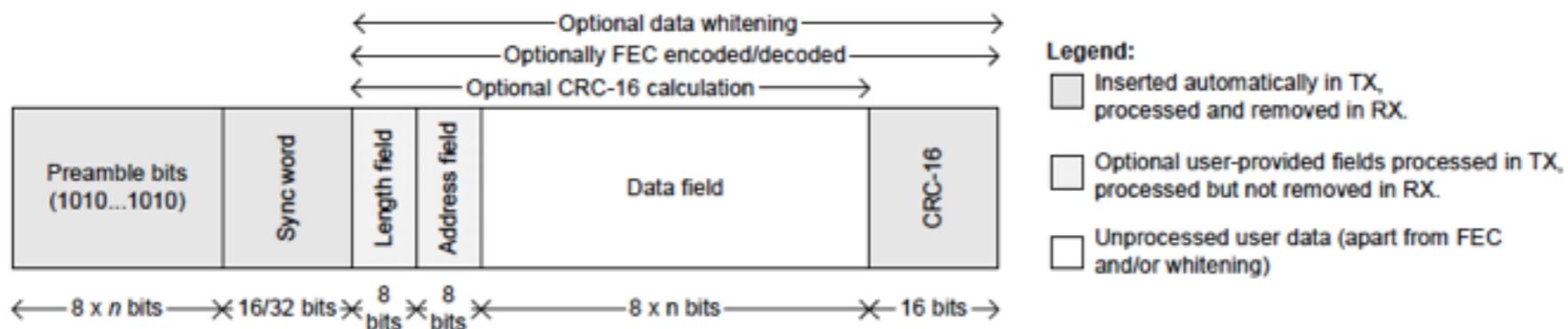


Figure 19: Packet Format

The preamble pattern is an alternating sequence of ones and zeros (10101010...). The minimum length of the preamble is programmable through the value of `MDMCFG1.NUM_PREAMBLE`. When enabling TX, the modulator will start transmitting the preamble. When the programmed number of preamble bytes has been transmitted, the modulator will send the sync word and then data from the TX FIFO if data is available. If the TX FIFO is empty, the modulator will continue to send preamble bytes until the first byte is written to the TX FIFO. The modulator will then send the sync word and then the data bytes.

The synchronization word is a two-byte value set in the `SYNC1` and `SYNC0` registers. The sync word provides byte synchronization of the incoming packet. A one-byte sync word can be emulated by setting the `SYNC1` value to the preamble pattern. It is also possible to emulate a 32 bit sync word by setting `MDMCFG2.SYNC_MODE` to 3 or 7. The sync word will then be repeated twice.

CC1101 supports both constant packet length protocols and variable length protocols. Variable or fixed packet length mode can be used for packets up to 255 bytes. For longer

Nibble the bits in hex

```
1. TERM=screen-256color-bce tmux attach [tmux]
tmux (tmux) 361 ~ (zsh) 362

204
gyaresu@zaphod:~|⇒ echo "obase=16; ibase=2; 1100" | bc
C
gyaresu@zaphod:~|⇒ echo "obase=10; ibase=2; 1100" | bc
12
gyaresu@zaphod:~|⇒ echo "8 + 4 + 0 + 0" | bc
12
gyaresu@zaphod:~|⇒ echo "obase=10; ibase=2; 11001100" | bc
204
gyaresu@zaphod:~|⇒ echo "128 + 64 + 0 + 0 + 8 + 4 + 0 + 0" | bc

204
gyaresu@zaphod:~|⇒ echo "obase=16; ibase=10; 204" | bc
CC
gyaresu@zaphod:~|⇒ █
□ 0 → gyaresu <GERcc1101 > 2 > ~ < 20:11 < 18 Nov ← zaphod
```

Packet identification

Pulse coding: Short pulse length 255 - Lo
Short distance: 245, long distance: 613,
p_limit: 418
bitbuffer:: Number of rows: 8
[00] {11} 6c c0 : 01101100 110
[01] {11} ab c0 : 10101011 110
[02] {11} 6c c0 : 01101100 110
[03] {11} ab c0 : 10101011 110
[04] {11} 6c c0 : 01101100 110
[05] {11} ab c0 : 10101011 110
[06] {11} 6c c0 : 01101100 110
[07] {11} ab c0 : 10101011 110
Test mode file issued 3 packets
gyaresu@zaphod:~/programming/
⇒ █
108
">>>>
█ 0 ➤ gyaresu ➤ king/modernrc 1 debian
values of internal variables or
mark measurement
, windowing it, F

short 0 (?)
long 1
long 1
short 0
long 1
long 1
short 0
short 0
long 1
long 1
short 0

2203125 Hz 2336 ms

Gareth @gyaresu · Sep 11
Ideas on encoding?
Garage door. rtl_433. Baudline.
12 Dipswitch Uni remote: 1 OFF, 2-12 ON.
#RTLSR #RFCat #GNURadio

2 ...

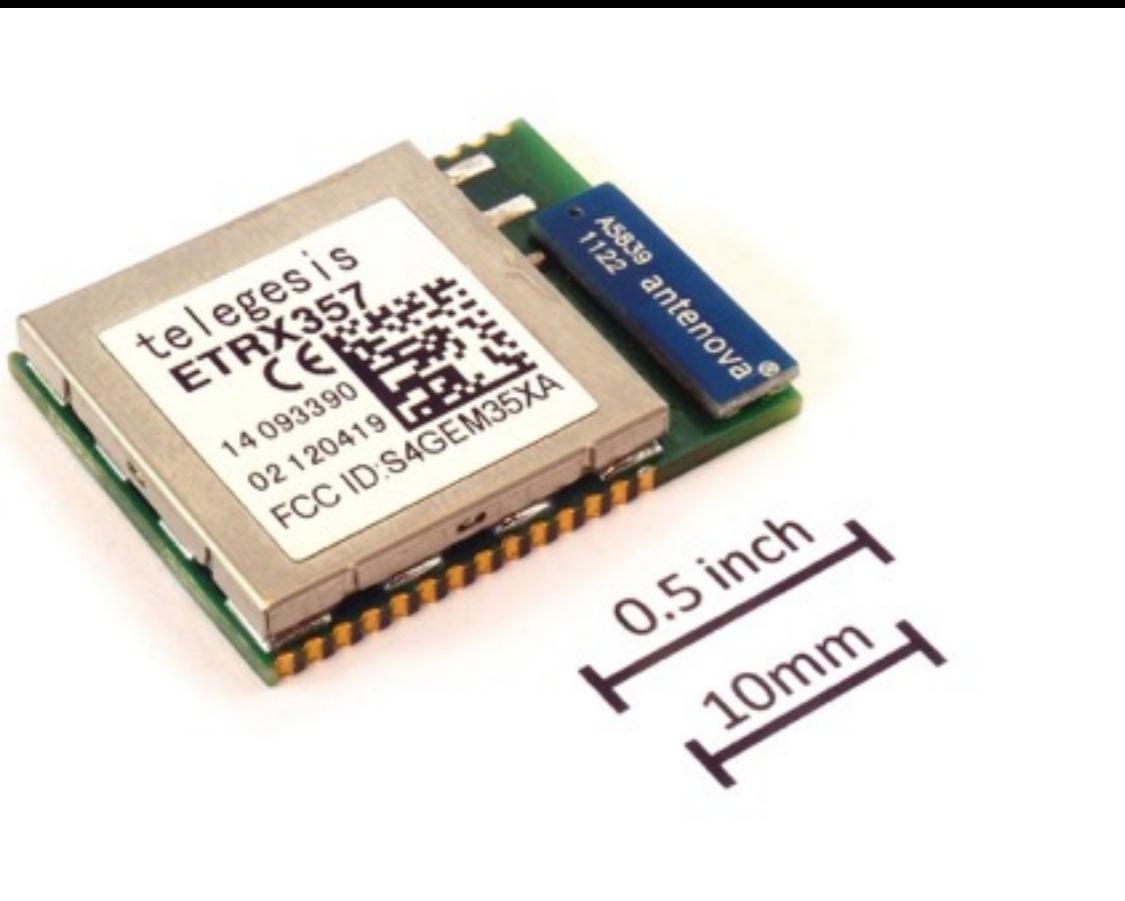
ZigBee

<protocols>

2.4 GHz in most jurisdictions worldwide;
784 MHz in China
868 MHz in Europe
915 MHz in the USA and Australia.

Z-Wave

Residential.
~900MHz

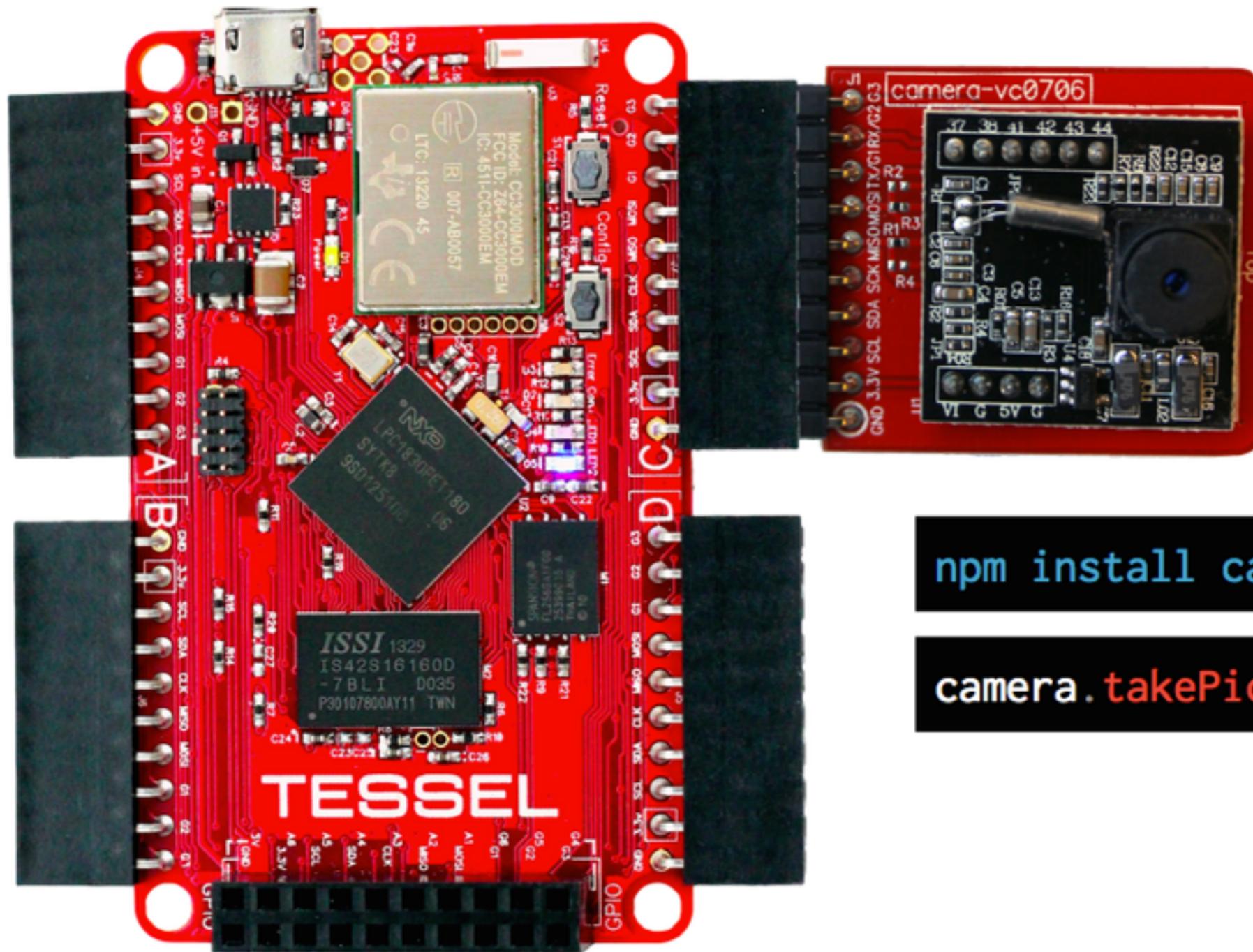


Brand: XBee



Brand: Sigma Designs

Wait... how did JavaScript get in here?



Camera

```
npm install camera-vc0706
```

```
camera.takePicture(function (){})
```

JavaScript is robots

The screenshot shows the GitHub page for the **Johnny-Five** repository. At the top, there's a navigation bar with links for **News**, **API**, **Examples**, **Articles**, and **Platform Support**. A yellow button labeled **J5** is on the far left. In the top right corner, there's a link to **Fork me on GitHub**. The main content area has a yellow background with a circuit board pattern. On the left, the repository name **Johnny-Five** is displayed in a large, bold, black font inside a dark box. Below it, the text **The JavaScript Programming Framework** is written in a smaller, dark font. To the right of the text is a cartoon illustration of a robot with multiple arms, holding a wrench and a screwdriver. The entire page is set against a yellow background with a faint, repeating circuit board pattern.

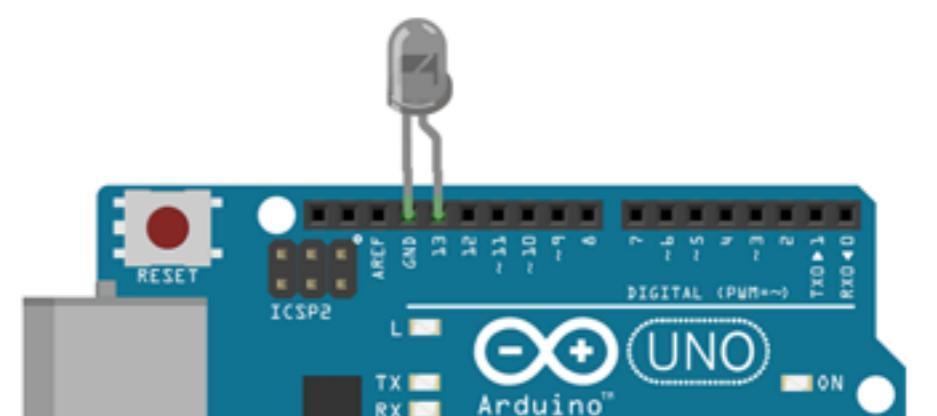
Johnny-Five is the original [JavaScript Robotics](#) programming framework. Released by [Bocoup](#) in 2012, Johnny-Five is maintained by a community of passionate software developers and hardware engineers. Over 75 developers have made contributions towards building a robust, extensible and composable ecosystem.

[Star](#) [Fork](#) [645](#)

Hello World!

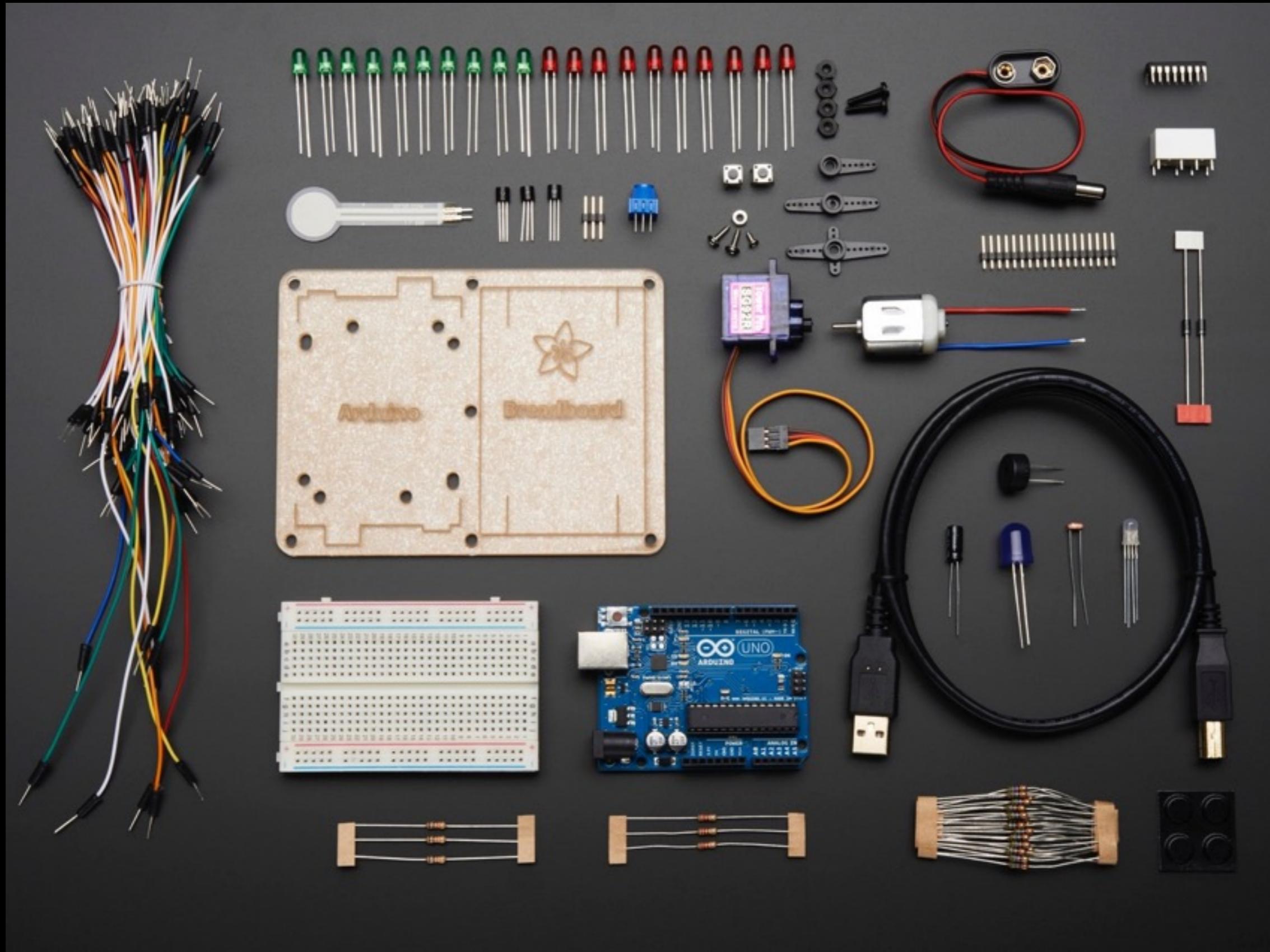
The ubiquitous "Hello World" program of the microcontroller and SoC world is "blink an LED!". The following code demonstrates how this is done using the Johnny-Five framework.

1. Install [Node.js](#) or [io.js](#).
2. [Setup your board](#).
3. `npm install johnny-five`



ARDX

Arduino Experimenter's Kit



Introduction

REFERENCE

Electronics Primer

JavaScript Primer

EXERCISES

1. Getting Started

2. 8 LED Fun

3. Spin Motor Spin

4. A Single Servo

5. 8 More LEDs

6. Music

7. Button Pressing

8. Twisting

9. Light

10. Temperature

11. Larger Loads

12. Colorful Light

13. Measuring Bends

14. Fancy Sensing

Arduino Experimenter's Guide for NodeJS

ABOUT THIS GUIDE

NodeBots are Arduino-based robots that are controlled by [node.js](#).

This guide will step you through assembling and programming a number of projects using an Arduino-compatible microcontroller and node.js, to help you get started building your own NodeBots. This guide has been designed to be used with the Arduino Experimenter's Kit, which is available from several suppliers, including SparkFun, AdaFruit, SEEED Studio and Freetronics.

The overall goal of this guide is fun. Beyond this, the aim is to give you a wide range of electronic components through small, simple and easy-to-understand projects. Working through the guide will give you the tools to figure out how it works and how to extend it.



Introduction

EXERCISE 11

REFERENCE

Electronics Primer

JavaScript Primer

EXERCISES

1. Getting Started

2. 8 LED Fun

3. Spin Motor Spin

4. A Single Servo

5. 8 More LEDs

6. Music

7. Button Pressing

8. Twisting

9. Light

10. Temperature

11. Larger Loads

12. Colorful Light

13. Measuring Bends

14. Fancy Sensing

PARTS

CIRCUIT LAYOUT

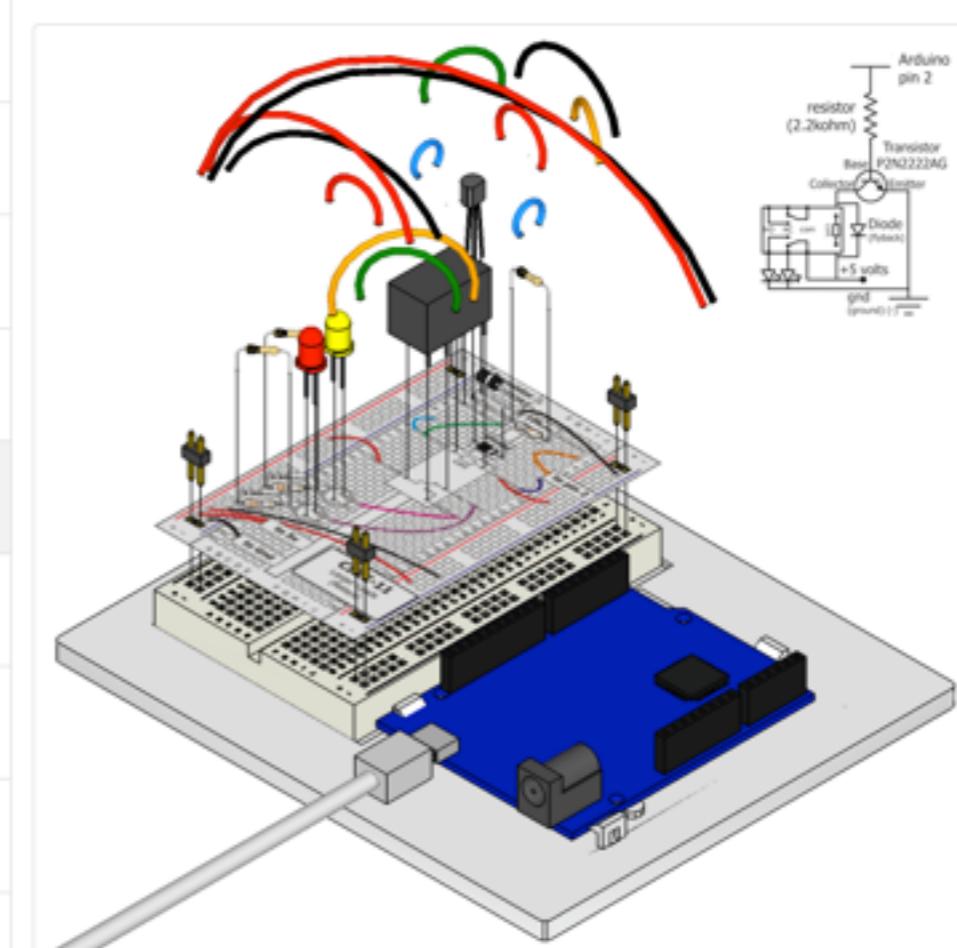
ASSEMBLY

CODE

TROUBLESHOOTING

EXTENDING THE CODE

MORE



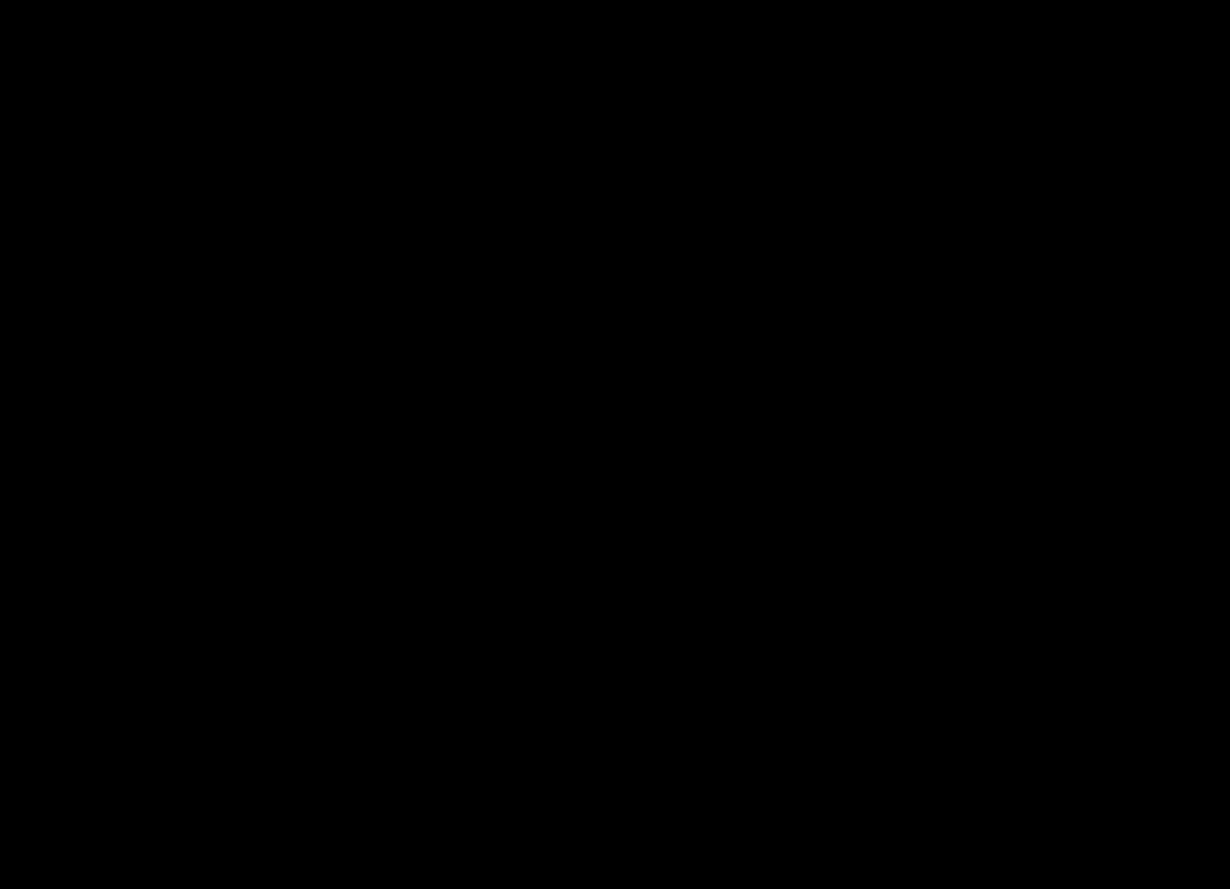
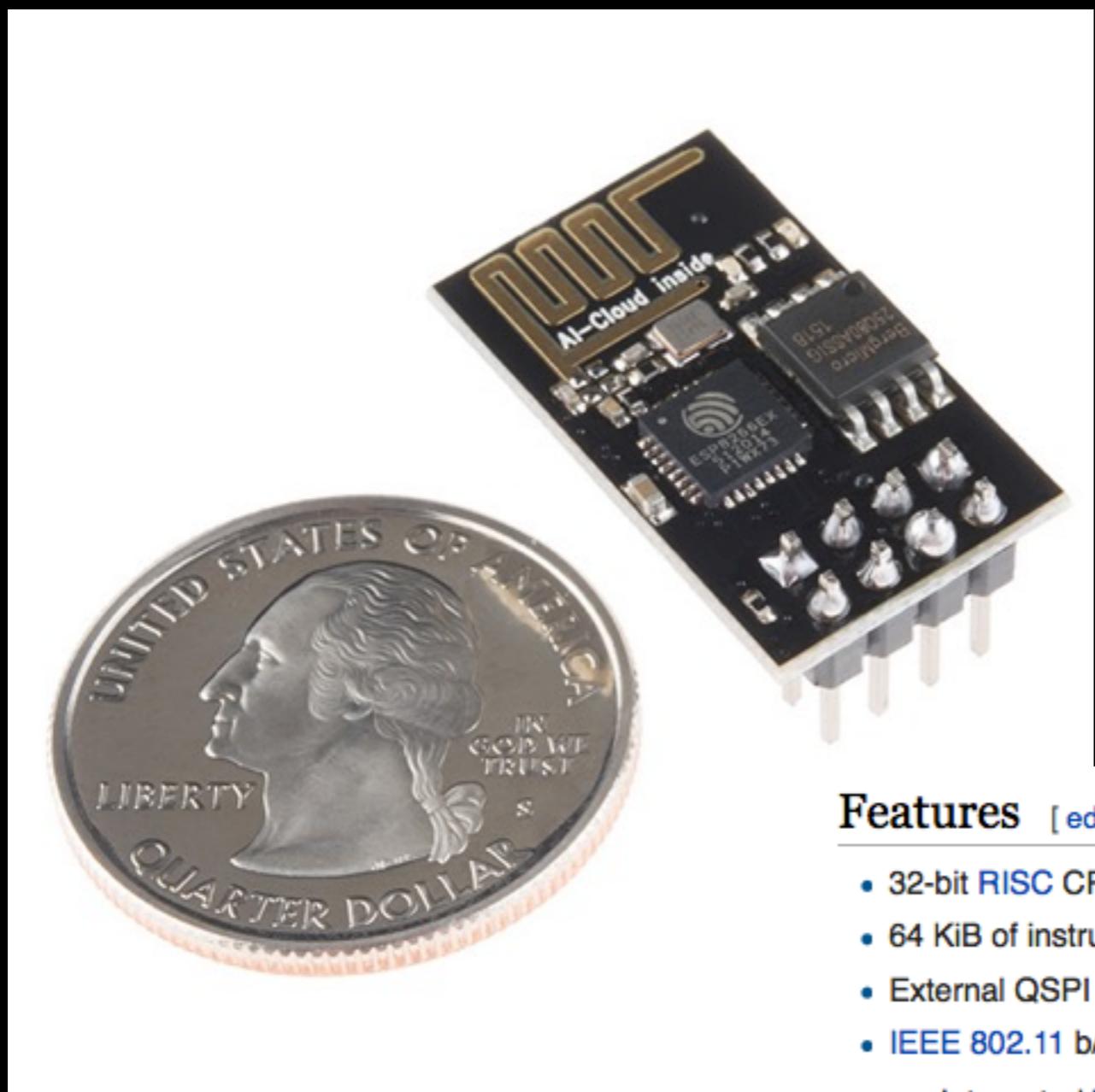
Assembly video: <http://ardx.org/VIDE11>

CODE

You can find this code in [code/CIRC-11-code-relay.js](#)

```
1 var five = require("johnny-five");
2 var myBoard = new five.Board();
3 myBoard.on("ready", function() {
4   var val = 0;
5   var relay = new five.Relay(2);
6   this.loop(1000, function() {
7     if (val % 1) {
8       relay.on();
```

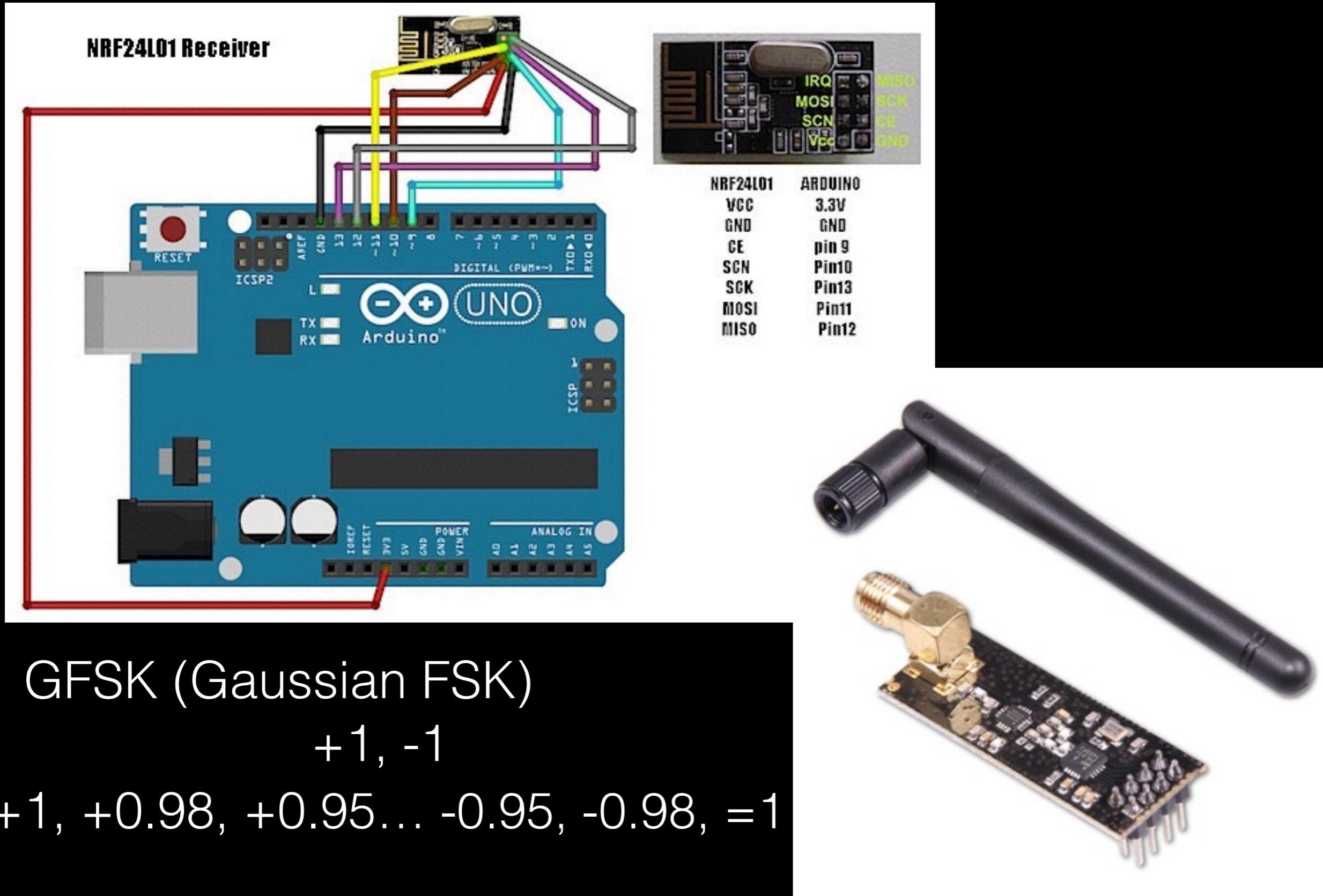
ESP8266 (2.4GHz WiFi™)



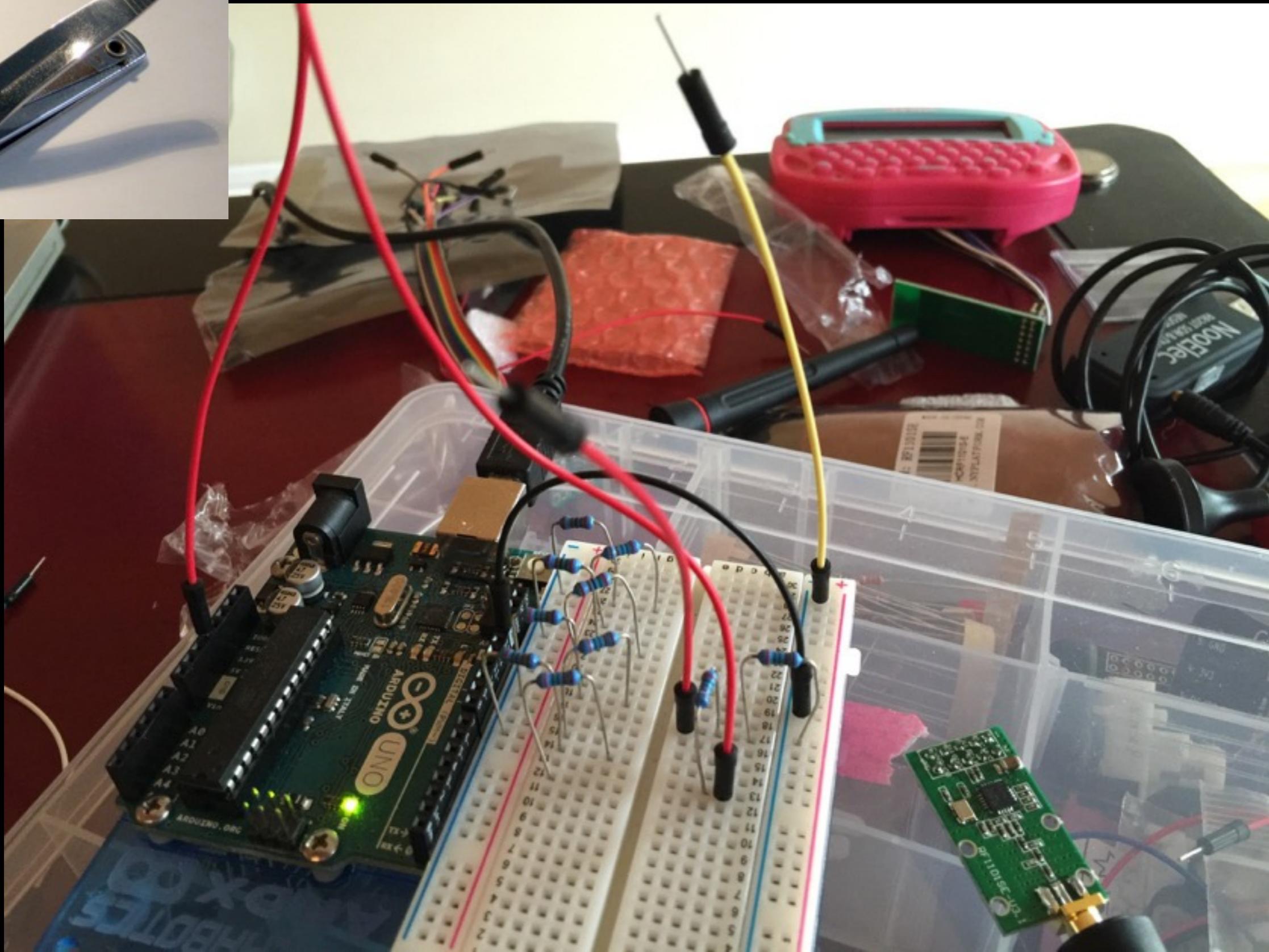
Features [edit]

- 32-bit RISC CPU: [Tensilica Xtensa LX106](#) running at 80 MHz
- 64 KiB of instruction RAM, 96 KiB of data RAM
- External QSPI flash - 512 KiB to 4 MiB
- IEEE 802.11 b/g/n Wi-Fi
 - Integrated TR switch, balun, LNA, power amplifier and matching network
 - WEP or WPA/WPA2 authentication, or open networks
- 16 GPIO pins
- SPI, I²C,
- I²S interfaces with DMA (sharing pins with GPIO)
- UART on dedicated pins, plus a transmit-only UART can be enabled on GPIO2
- 1 10-bit ADC

NRF24L01 (2.4GHz != Wifi™)



Lesson #1: Plug it in



Voltage Dividers (WAT?)

Screenshot of the Wikipedia article on Voltage dividers.

Voltage divider

From Wikipedia, the free encyclopedia

In electronics, a voltage divider (also known as a potential divider) is a passive linear circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). Voltage division is the result of distributing the input voltage among the components of the divider. A simple example of a voltage divider is two resistors connected in series, with the input voltage applied across the resistor pair and the output voltage emerging from the connection between them.

Resistor voltage dividers are commonly used to create reference voltages, or to reduce the magnitude of a voltage so it can be measured, and may also be used as signal attenuators at low frequencies. For direct current and relatively low frequencies, a voltage divider may be sufficiently accurate if made only of resistors; where frequency response over a wide range is required (such as in an oscilloscope probe), a voltage divider may have capacitive elements added to compensate load capacitance. In electric power transmission, a capacitive voltage divider is used for measurement of high voltage.

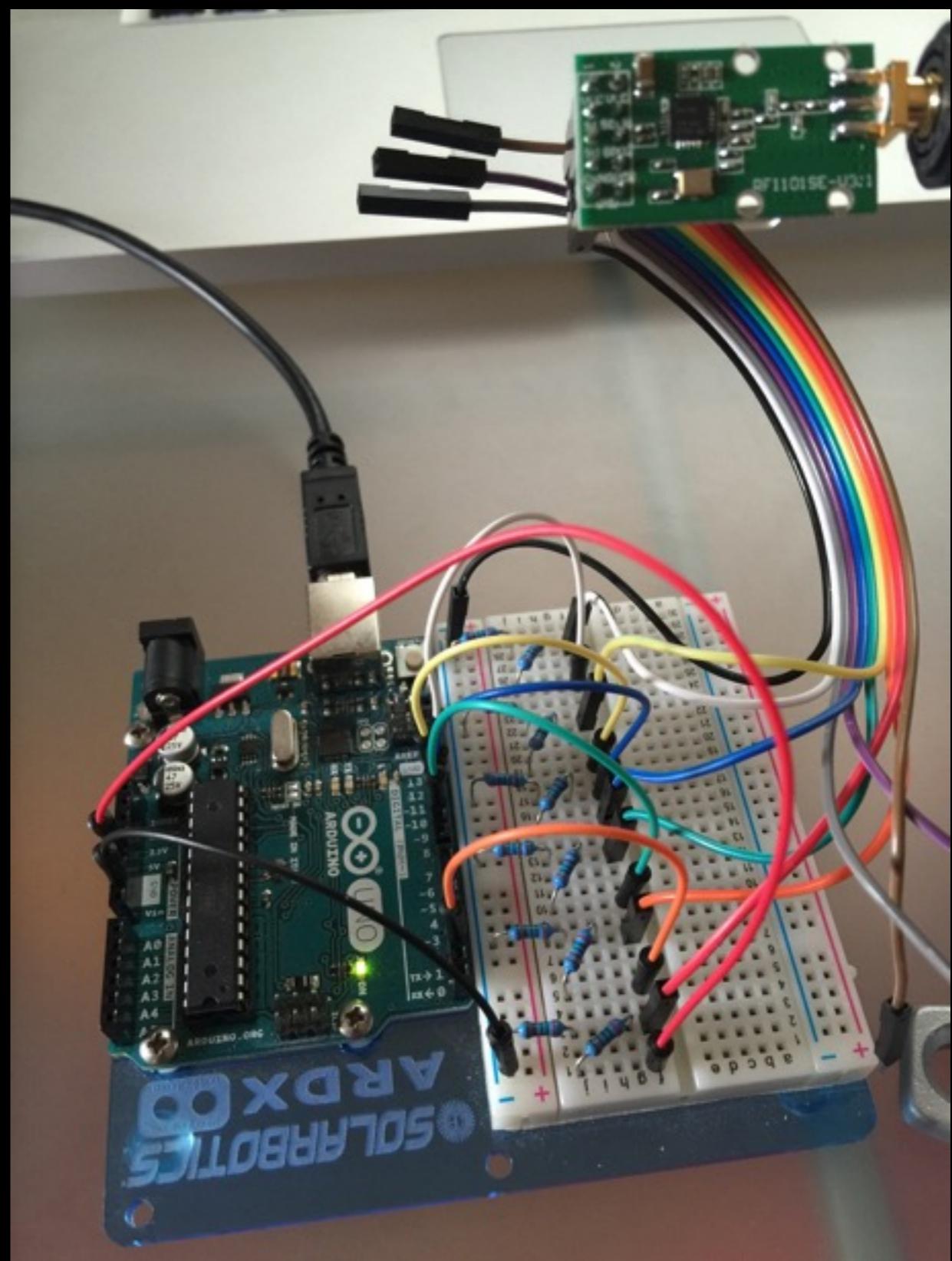
Contents [hide]

- 1 General case
- 2 Examples
 - 2.1 Resistive divider
 - 2.2 Low-pass RC filter
 - 2.3 Inductive divider
 - 2.4 Capacitive divider
- 3 Loading effect
- 4 Applications
 - 4.1 Sensor measurement
 - 4.2 High voltage measurement
 - 4.3 Level shifting
- 5 References
- 6 See also
- 7 External links

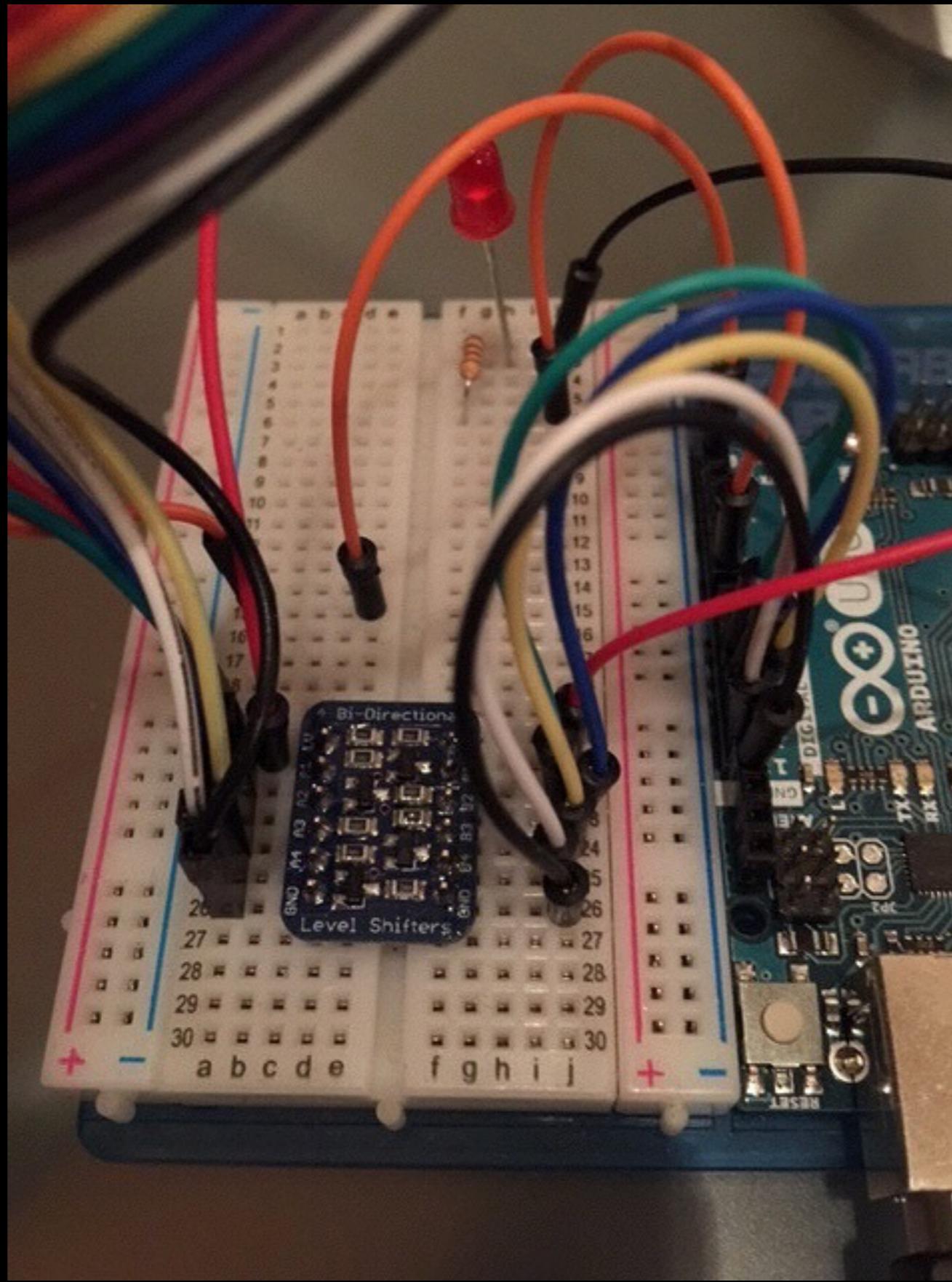
General case [edit]

Figure 1: A simple voltage divider

```
graph LR; Vin((Vin)) --- R1[Z1]; R1 --- Node1(( )); Node1 --- R2[Z2]; R2 --- Gnd(( )); Node1 --- Vout((Vout));
```

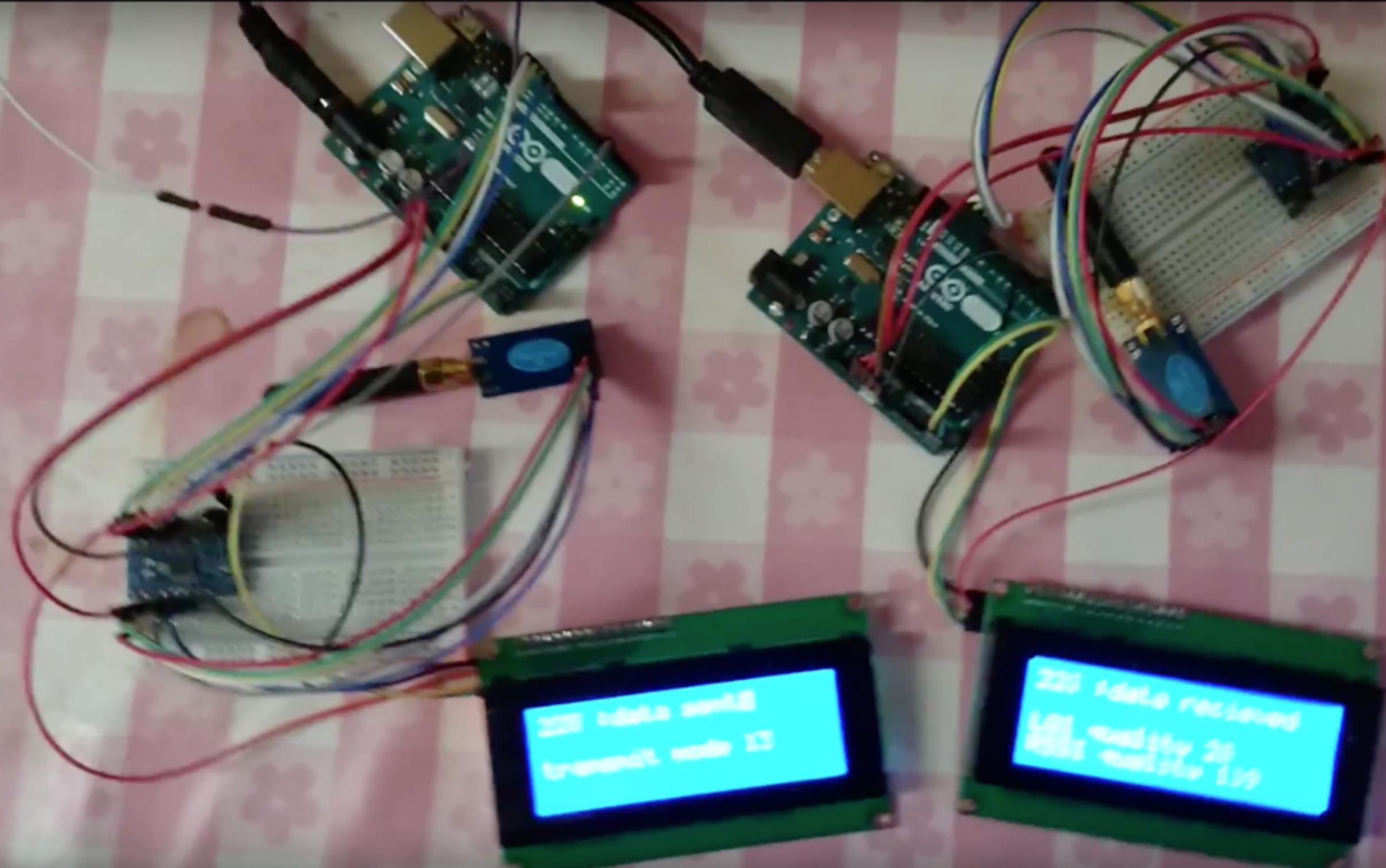


Logic Level Converters



It's alive!

CC1101 RF module tests with arduino logic level converters PART4



rf1101.cpp

Wat?

ABC

Always Be Checking the damn data sheet

cc1101.pdf

Open Tools Fill & Sign Comment

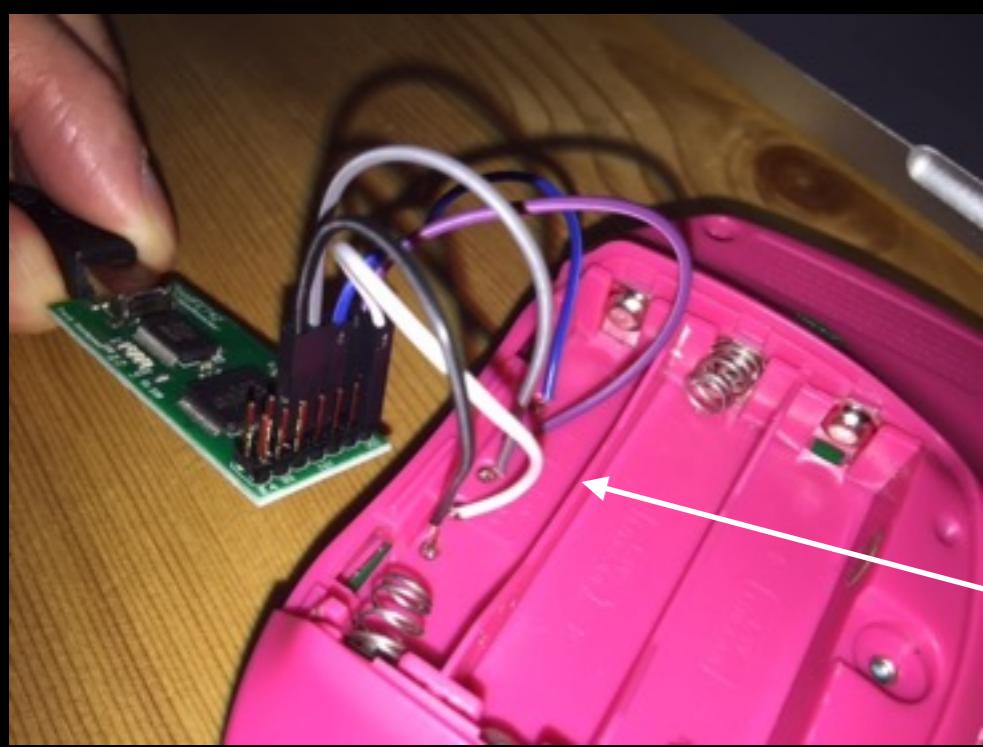
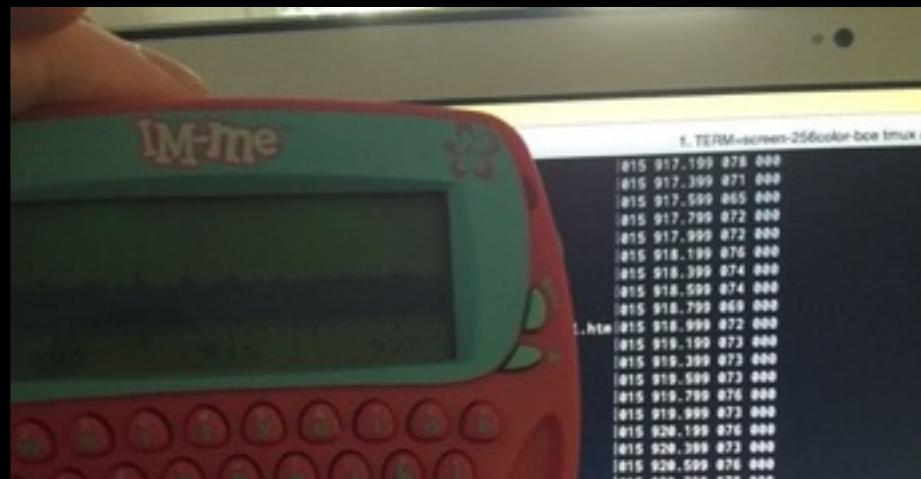
62 / 105 125%

GDOx_CFG[5:0] Description

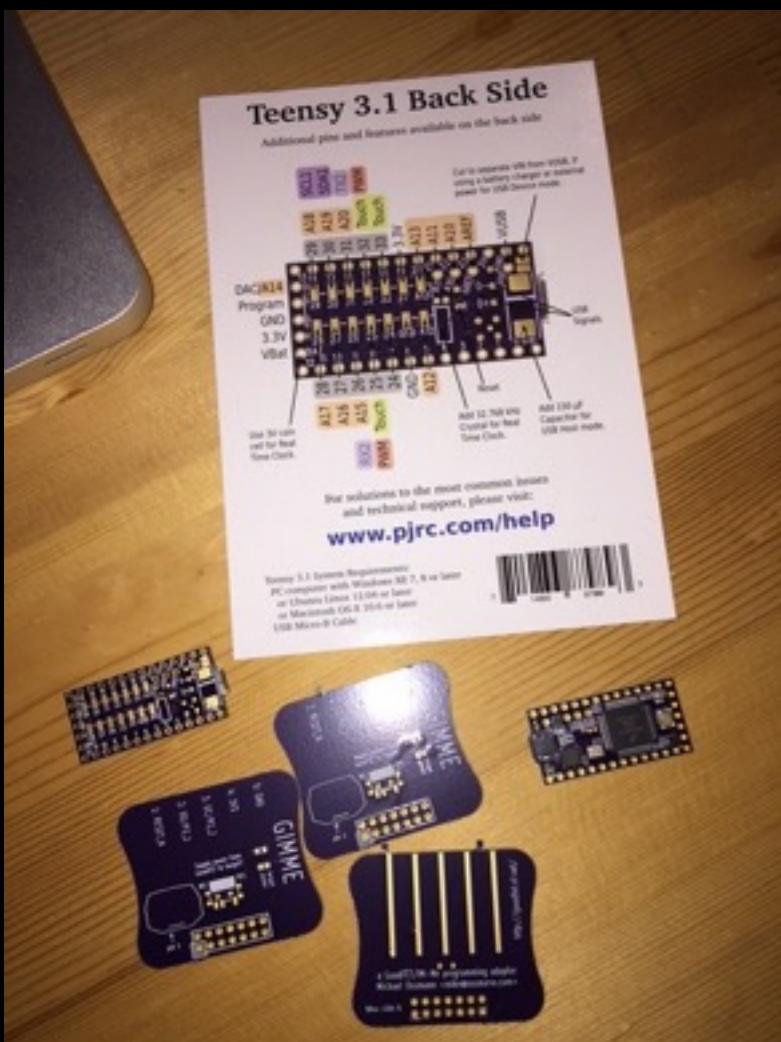
0 (0x00)	Associated to the RX FIFO: Asserts when RX FIFO is filled at or above the RX FIFO threshold. De-asserts when RX FIFO is drained below the same threshold.
1 (0x01)	Associated to the RX FIFO: Asserts when RX FIFO is filled at or above the RX FIFO threshold or the end of packet is reached. De-asserts when the RX FIFO is empty.
2 (0x02)	Associated to the TX FIFO: Asserts when the TX FIFO is filled at or above the TX FIFO threshold. De-asserts when the TX FIFO is below the same threshold.
3 (0x03)	Associated to the TX FIFO: Asserts when TX FIFO is full. De-asserts when the TX FIFO is drained below the TX FIFO threshold.
4 (0x04)	Asserts when the RX FIFO has overflowed. De-asserts when the FIFO has been flushed.
5 (0x05)	Asserts when the TX FIFO has underflowed. De-asserts when the FIFO is flushed.
6 (0x06)	Asserts when sync word has been sent / received, and de-asserts at the end of the packet. In RX, the pin will also de-assert when a packet is discarded due to address or maximum length filtering or when the radio enters RXFIFO_OVERFLOW state. In TX the pin will de-assert if the TX FIFO underflows.
7 (0x07)	Asserts when a packet has been received with CRC OK. De-asserts when the first byte is read from the RX FIFO.
8 (0x08)	Preamble Quality Reached. Asserts when the PQI is above the programmed PQT value. De-asserted when the chip re-enters RX state (<code>MARCSTATE=0x0b</code>) or the PQI gets below the programmed PQT value.
9 (0x09)	Clear channel assessment. High when RSSI level is below threshold (dependent on the current CCA_MODE setting).
10 (0x0A)	Lock detector output. The PLL is in lock if the lock detector output has a positive transition or is constantly logic high. To check for PLL lock the lock detector output should be used as an interrupt for the MCU.
11 (0x0B)	Serial Clock. Synchronous to the data in synchronous serial mode. In RX mode, data is set up on the falling edge by <code>CC1101</code> when <code>GDOx_INV=0</code> . In TX mode, data is sampled by <code>CC1101</code> on the rising edge of the serial clock when <code>GDOx_INV=0</code> .
12 (0x0C)	Serial Synchronous Data Output. Used for synchronous serial mode.
13 (0x0D)	Serial Data Output. Used for asynchronous serial mode.
14 (0x0E)	Carrier sense. High if RSSI level is above threshold. Cleared when entering IDLE mode.
15 (0x0F)	CRC_OK. The last CRC comparison matched. Cleared when entering/restarting RX mode.
16 (0x10) to 21 (0x15)	Reserved – used for test
22 (0x16)	RX_HARD_DATA[1]. Can be used together with RX_SYMBOL_TICK for alternative serial RX output.
23 (0x17)	RX_HARD_DATA[0]. Can be used together with RX_SYMBOL_TICK for alternative serial RX output.
24 (0x18) to 26 (0x1A)	Reserved – used for test
27 (0x1B)	PA_PD. Note: PA_PD will have the same signal level in SLEEP and TX states. To control an external PA or RX/TX switch in applications where the SLEEP state is used it is recommended to use <code>GDOx_CFGx=0x2F</code> instead.
28 (0x1C)	LNA_PD. Note: LNA_PD will have the same signal level in SLEEP and RX states. To control an external LNA or RX/TX switch in applications where the SLEEP state is used it is recommended to use <code>GDOx_CFGx=0x2F</code> instead.
29 (0x1D)	RX_SYMBOL_TICK. Can be used together with RX_HARD_DATA for alternative serial RX output.
30 (0x1E) to 35 (0x23)	Reserved – used for test
36 (0x24)	WOR_EVNT0
37 (0x25)	WOR_EVNT1
38 (0x26)	CLK_256
39 (0x27)	CLK_32k
40 (0x28)	Reserved – used for test
41 (0x29)	CHIP_RDYn
42 (0x2A)	Reserved – used for test
43 (0x2B)	XOSC_STABLE
44 (0x2C)	Reserved – used for test
45 (0x2D)	Reserved – used for test
46 (0x2E)	High impedance (3-state)
47 (0x2F)	HW to 0 (HW1 achieved by setting <code>GDOx_INV=1</code>). Can be used to control an external LNA/PA or RX/TX switch.
48 (0x30)	CLK_XOSC/1
49 (0x31)	CLK_XOSC/1.5
50 (0x32)	CLK_XOSC/2

Find 2e Previous Next

Real men hack pink



Teensy don't care



pjrc.com/teensy

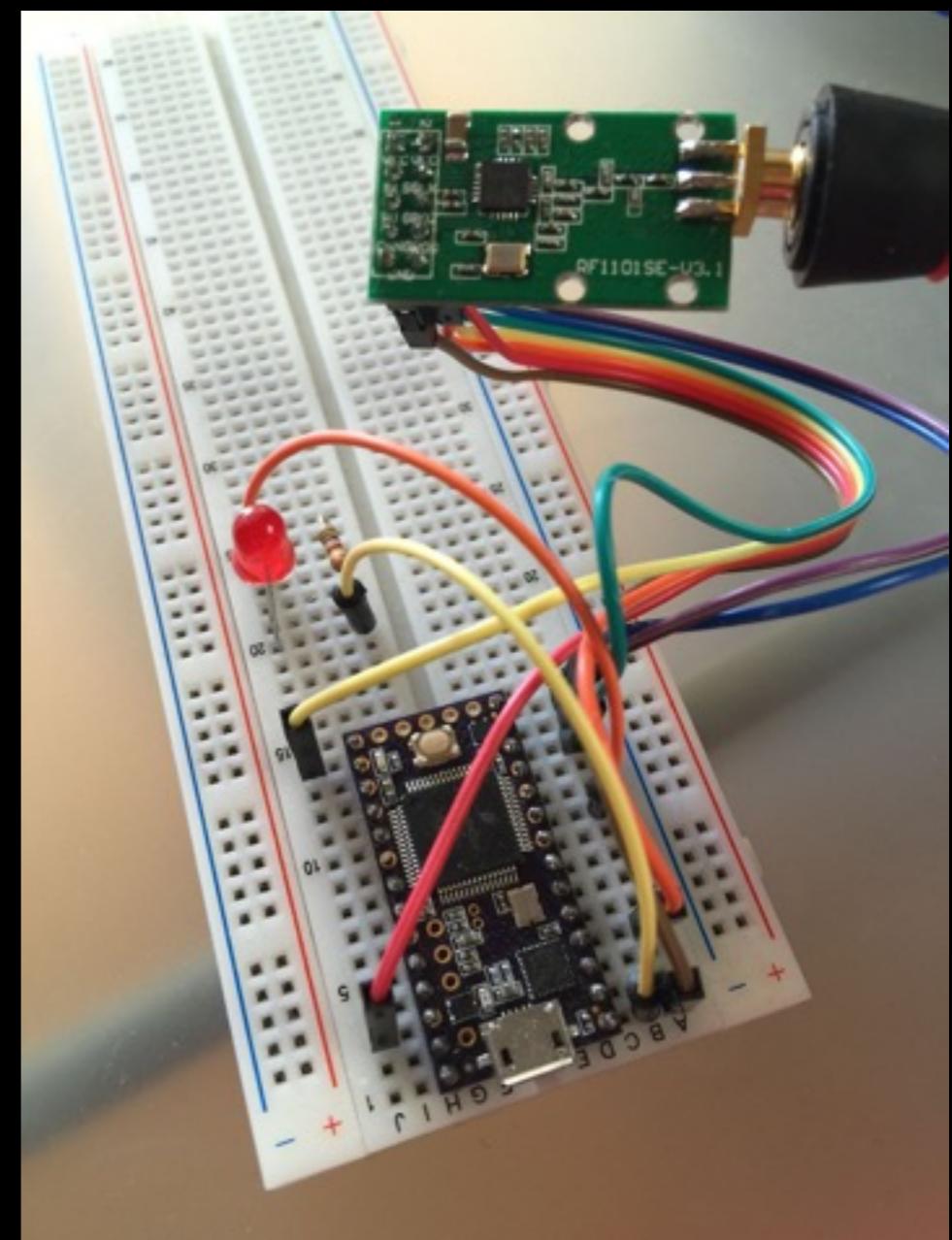
OSHWPark.com

OSH Park
BLEKey

BLEKey

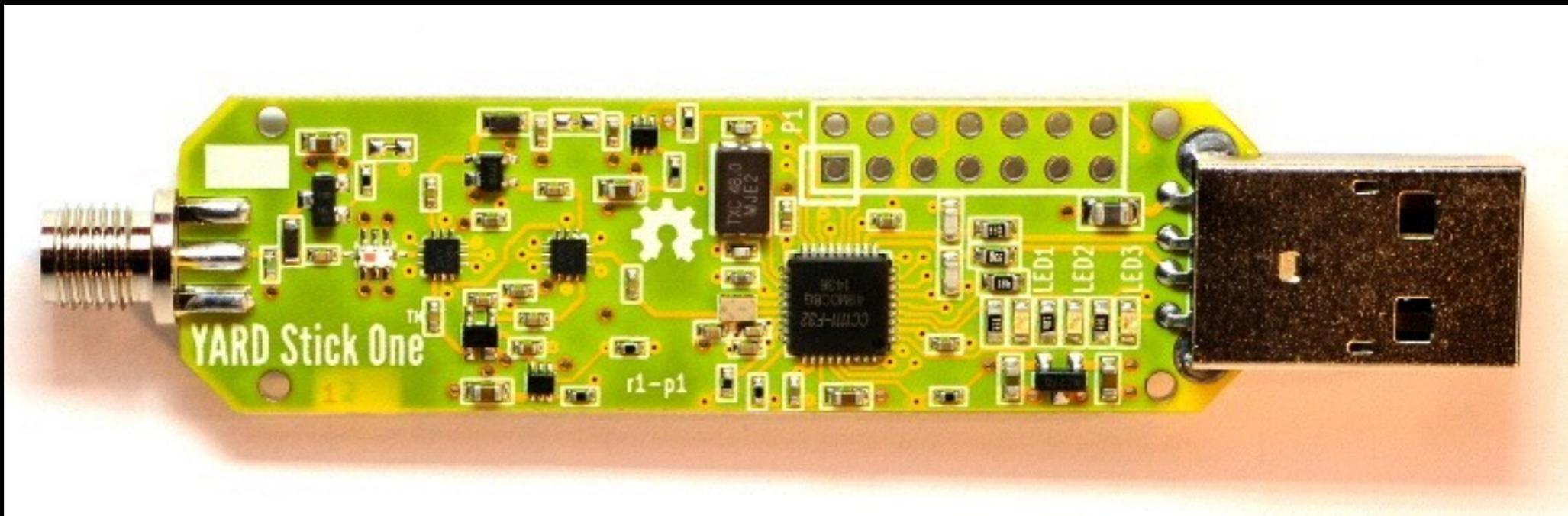
by EvilMug.
2 layer board of 0.67x1.20 inches (16.92x30.51 mm). Shared on August 20th, 2015 16:48.
Bluetooth Low Energy Wiegand Key
Order now. Download. Permalink.

Designed and developed by Resistor.

A screenshot of the OSH Park website showing a product page for the BLEKey. The page features a purple header with the OSH Park logo and a circular gear icon. Below the header, the product name "BLEKey" is displayed in large white text. A detailed description follows, including the author's name, the board's dimensions, the date it was shared, its function as a Bluetooth Low Energy Wiegand Key, and links for ordering, downloading, and viewing the permalink. At the bottom, there is a note about the design and development being done by Resistor.

96MHz
256 KB Flash
64 KB OF RAM!

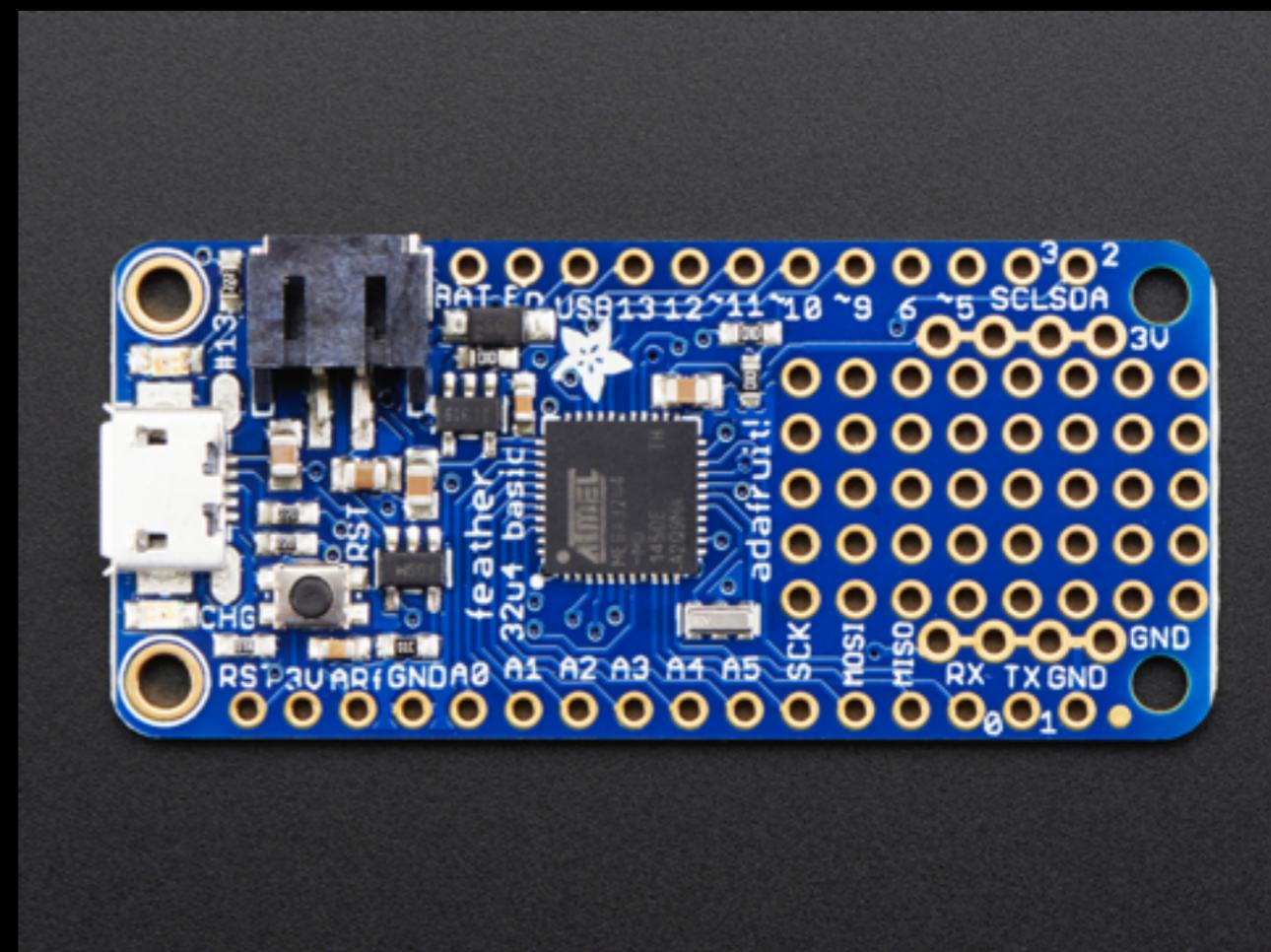
YARD Stick One



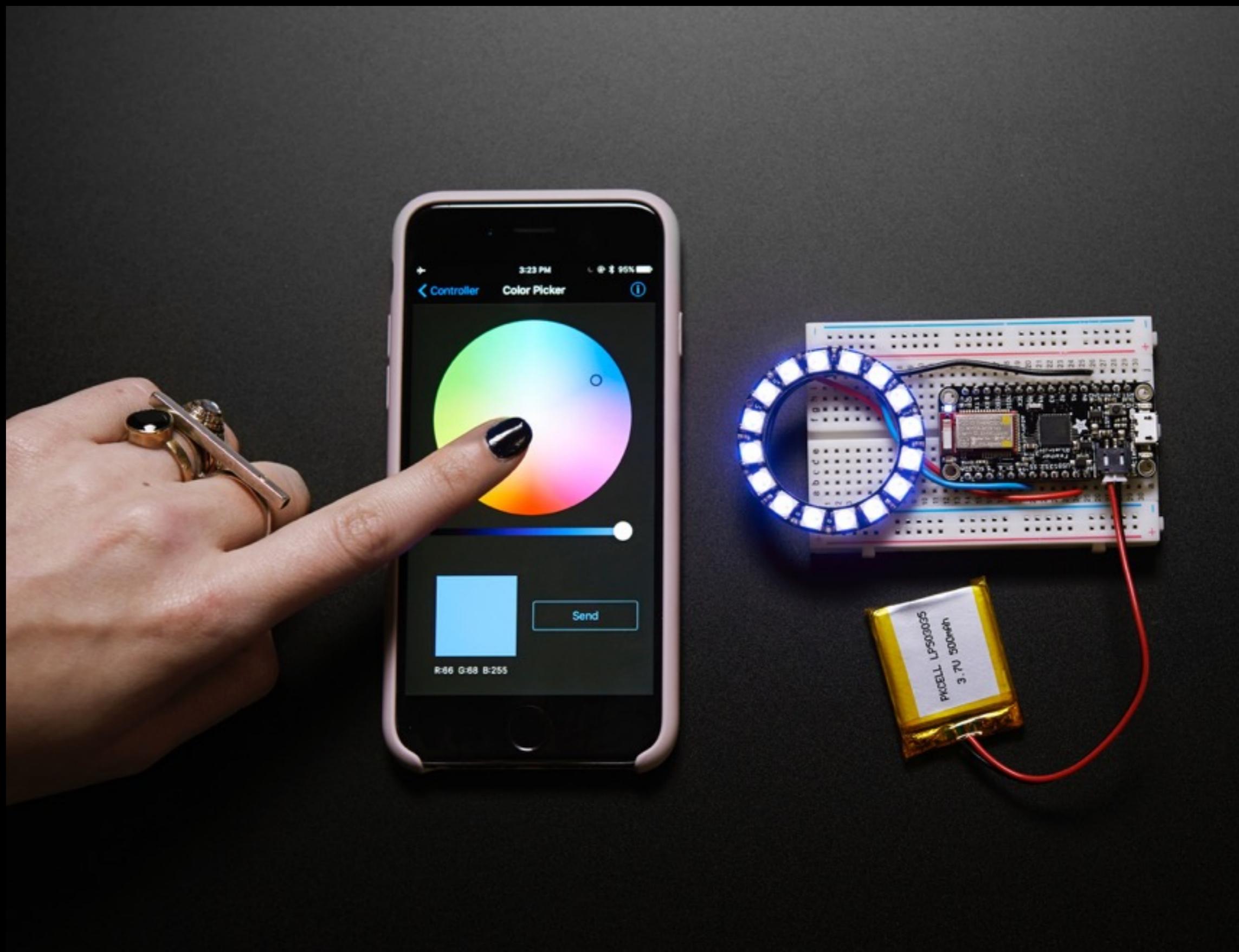
- SMA connector for external antennas such as [ANT500](#)
- receive amplifier for improved sensitivity
- transmit amplifier for higher output power
- strong RF performance across the entire operating frequency range
- low pass filter for elimination of harmonics when operating in the 800 and 900 MHz bands
- antenna port power control for compatibility with antenna port accessories designed for [HackRF One](#)
- [GoodFET](#)-compatible expansion and programming header
- [GIMME](#)-compatible programming test points

Adafruit Feather

- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather – 4.8 grams
- ATmega32u4 @ 8MHz with 3.3V logic/power
- 3.3V regulator with 500mA peak current output
- USB native support, comes with USB bootloader and serial port debugging
- You also get tons of pins – 20 GPIO pins
- Hardware Serial, hardware I2C, hardware SPI support
- 8 x PWM pins
- 10 x analog inputs (one is used to measure the battery voltage)
- Built in 100mA lipoly charger with charging status indicator LED
- Pin #13 red LED for general purpose blinking
- Power/enable pin
- 4 mounting holes
- Reset button

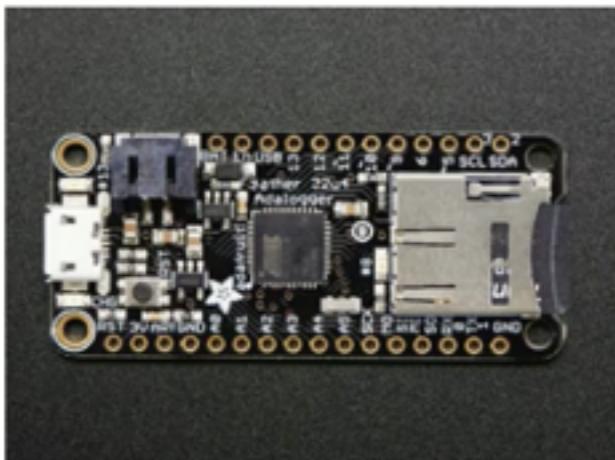


Adafruit Feather 32u4 Bluefruit LE



Adafruit's pretty great

BOARDS / FEATHER



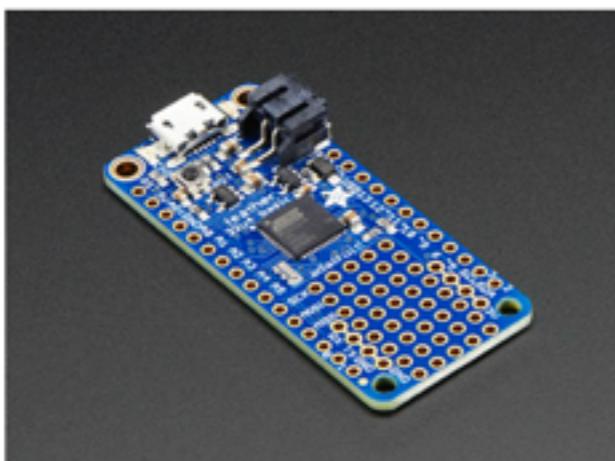
Adafruit Feather 32u4 Adalogger

PRODUCT ID: 2795

Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you standard for portable microcontroller cores. This is the Adafruit Feather 32u4 Adalogger - our take on reader) with built in USB and battery charging. Its an Adafruit Feather 32u4 with a microSD holder ready the...

[NOTIFY ME](#)

\$21.95
OUT OF STOCK



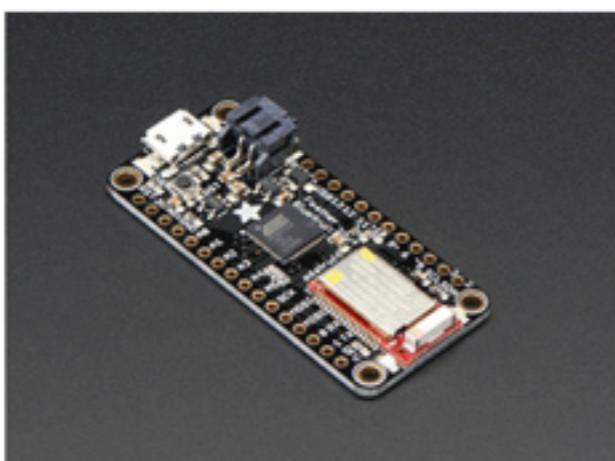
Adafruit Feather 32u4 Basic Proto

PRODUCT ID: 2771

Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you standard for portable microcontroller cores. This is the Feather 32u4 Basic Proto, it has a bunch of proto other boards in the Feather family, check'em out here! At the Feather 32u4's heart is an ATmega32u4 core chip...

[ADD TO CART](#)

\$19.95
39 IN STOCK



Adafruit Feather 32u4 Bluefruit LE

PRODUCT ID: 2829

Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you standard for portable microcontroller cores. This is the Adafruit Feather 32u4 Bluefruit - our take on Bluetooth Low Energy with built in USB and battery charging. Its an Adafruit Feather 32u4 with a BTLE

[ADD TO CART](#)

\$29.95
IN STOCK

Cheap!

SUPERDEALS BESTSELLING

www.aliexpress.com/item/high-quality-UNO-R3-MEGA328P-CH340-CH340G-for-Arduino-UNO-R3-USB-CABLE/

AliExpress™ I'm shopping for... All Categories Cart (0) Wish List Sign in Join My AliExpress

Home > All Categories > Electronic Components & Supplies > Other Electronic Components

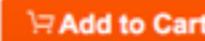

Shenzhen Bing Electronic Co., Ltd

Price: C\$ 4.84 / piece

Shipping: Free Shipping to Canada via China Post Ordinary Small Packet Plus
Estimated Delivery Time: 15-34 days (ships out within 5 business days)

Quantity: 1 piece (986277 pieces available)

Total Price: C\$ 4.84

Add to Wish List (1368 Adds)

Return Policy: Returns accepted if product not as described, buyer pays return shipping fee; or keep the product & agree refund with seller. View details

Seller Guarantees: On-time Delivery 60 days

 **Buyer Protection**

Full Refund if you don't receive your order
 Full or Partial Refund, if the item is not as described

Learn More

This Seller's Categories

- promotion
- 3D Printer Accessories
 - J-head hotend & Throat
 - 3D Nozzle
 - Heat bed & Heat Sink
 - Fan & Tape & Tube
 - Timing belt & Timing pulley

Product Details Feedback (2211) Shipping & Payment Seller Guarantees Report item

Item specifics

Brand Name: Bing
Model Number: SKIT-65
Name: UNOR3
Type: MEGA328P

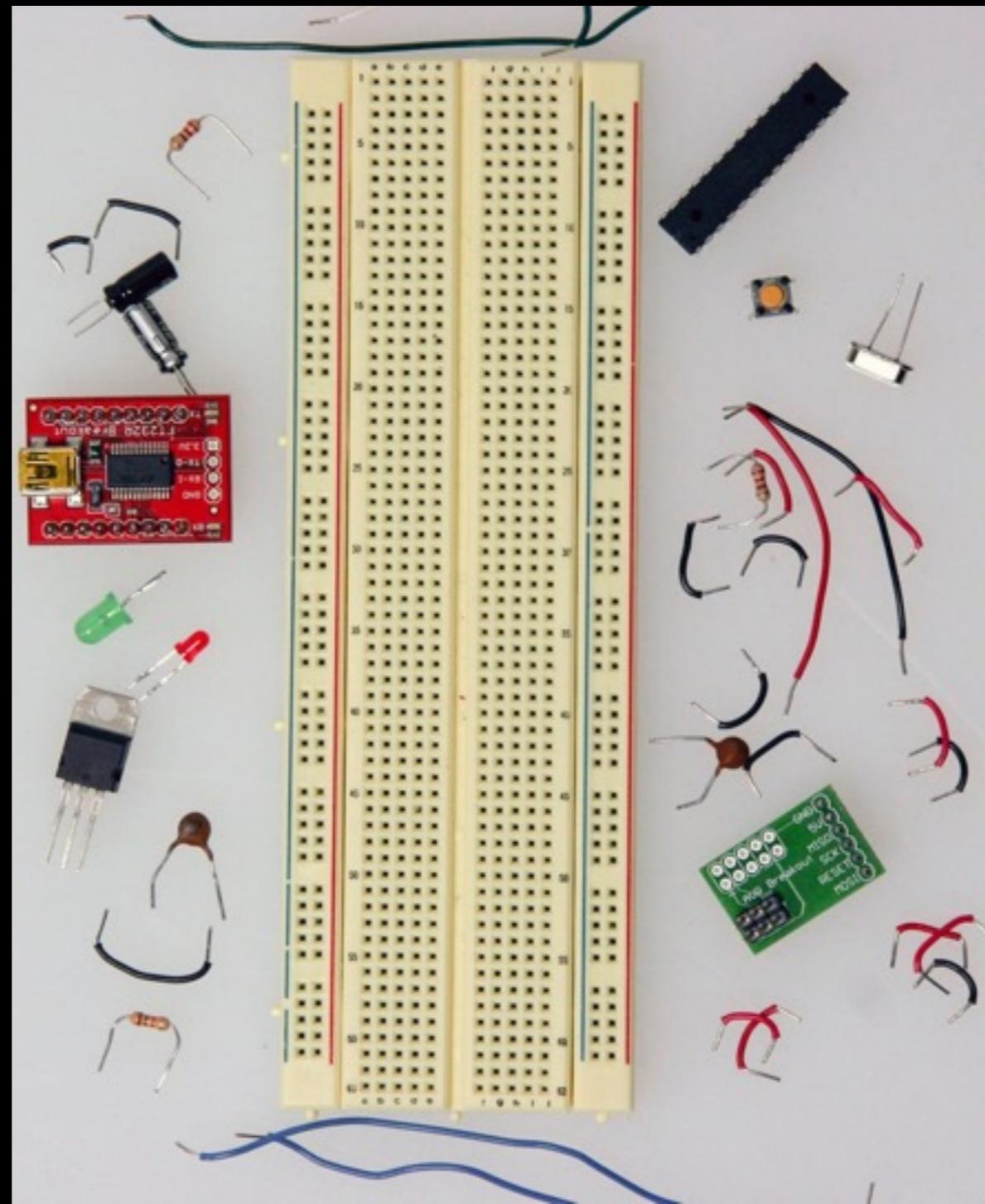
Recently Viewed

USB to Serial: ch430 vs FTDI Universal asynchronous receiver/transmitter (UART)

The screenshot shows a web browser window displaying the WCH (Weinzierl Communications) website. The URL in the address bar is www.wch.cn/download/CH341SER_ZIP.html. The page title is "WCH 沁恒". The main navigation menu includes "关于我们" (About Us), "产品中心" (Product Center), "应用方案" (Application Solutions), "BBS", "在线下载" (Online Download), "招贤纳士" (Recruitment), and "联系我们" (Contact Us). A large graphic of a blue arrow pointing upwards with small cubes representing data is centered above the "DOWNLOADS/在线下载" link. The breadcrumb navigation shows "首页 > 在线下载 > CH341SER.ZIP". On the left, there is a sidebar with links for "在线搜索" (Online Search), "技术手册" (Technical Manual), "应用资料" (Application Materials), "其他资料" (Other Materials), and "联系我们" (Contact Us) with a phone icon. The main content area displays the file details for "CH341SER.ZIP":
- 资料名称: CH341SER.ZIP
- 资料类型: 辅助资料
- 资料大小: 198KB
- 资料版本: 3.4
- 更新时间: 2015-10-30
- 软件简介: USB转并口CH341的WINDOWS驱动程序和DLL动态库, 支持WINDOWS 98/ME/2000/XP/Vista/7/8.1/10/2003/2008/2012 -32位/64位, 通过微软数字签名认证, 支持USB转EPP/EEMEM并口, 支持USB转同步串口: IIC/I2C、SPI等, 可用于USB转异步串口代替仿真串口驱动。
A large blue "DOWNLOAD" button with a downward arrow is prominently displayed. Below it, under "相关资料" (Related Materials), are links to other files:
- CH341SER.EXE: USB转并口CH341的WINDOWS驱动程序
- CH341SER_LINUX.ZIP: USB转串口CH340/CH341的虚拟串口
- CH341SER_MAC.ZIP: USB转串口CH340/CH341的苹果驱动
- CH341DS1.PDF: CH341技术手册 用于USB转串口/并口

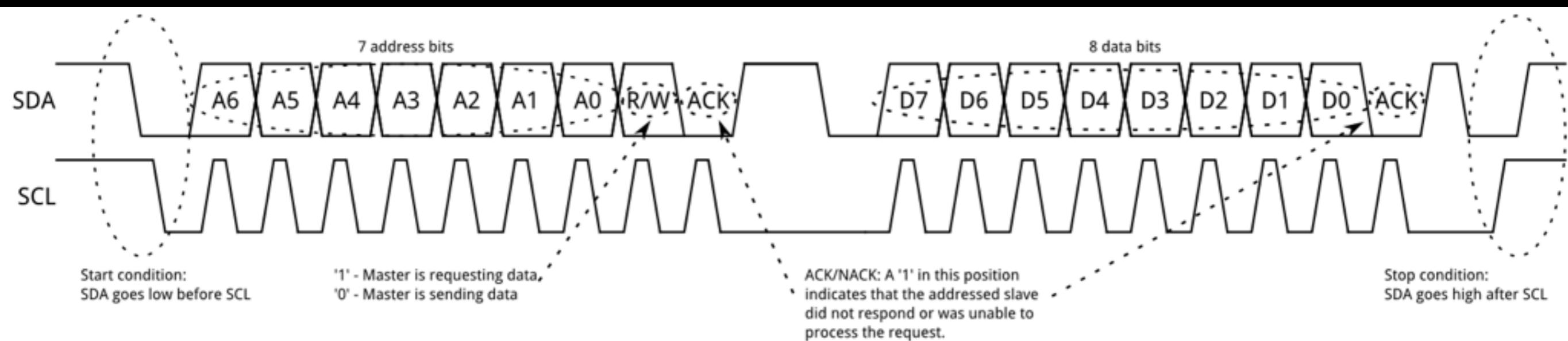
sudo nvram boot-args="kext-dev-mode=1"

Arduino from scratch

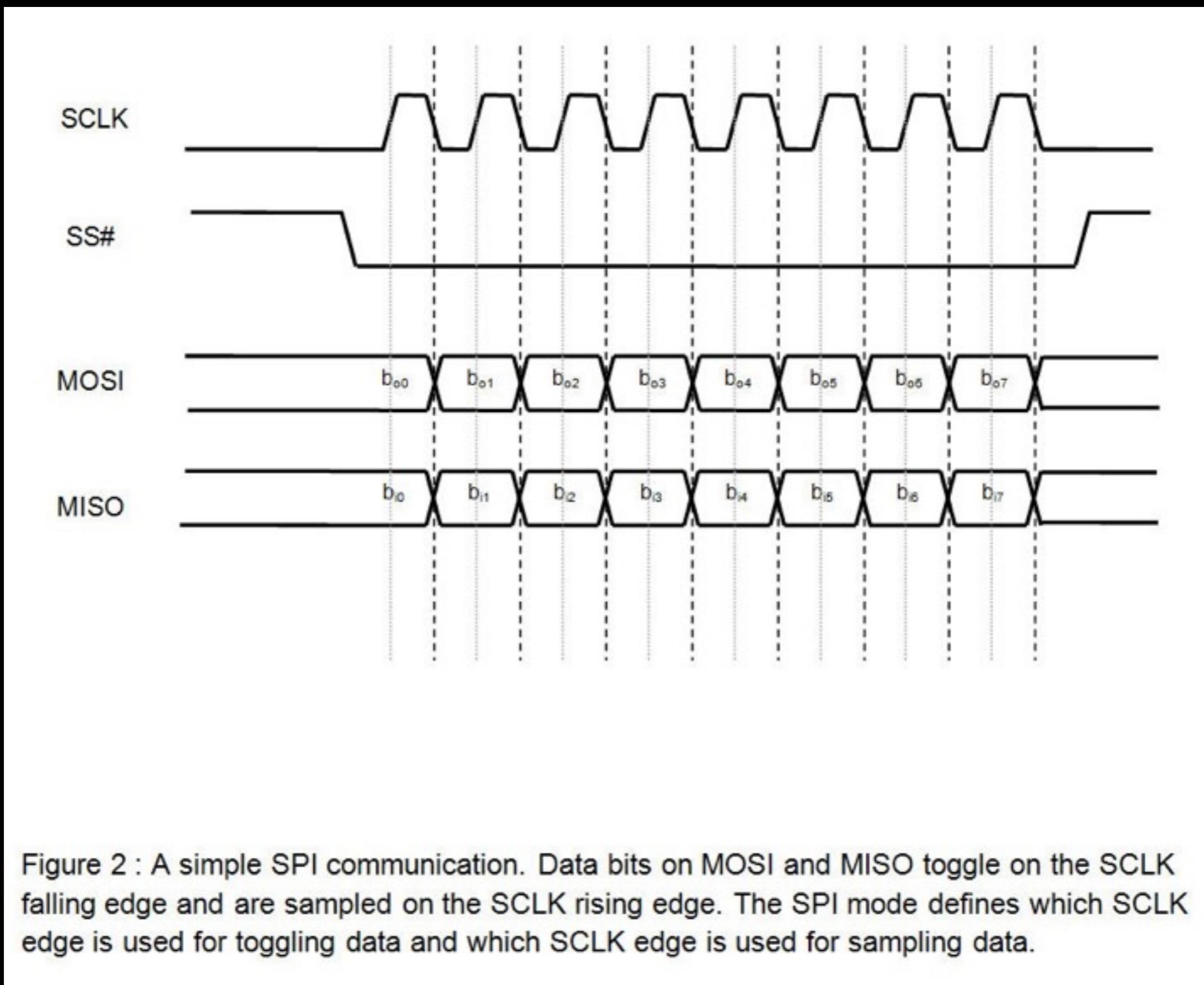


“Oh, you want power? Yeah, that’s extra.”

I2C (Inter-Integrated Circuit)

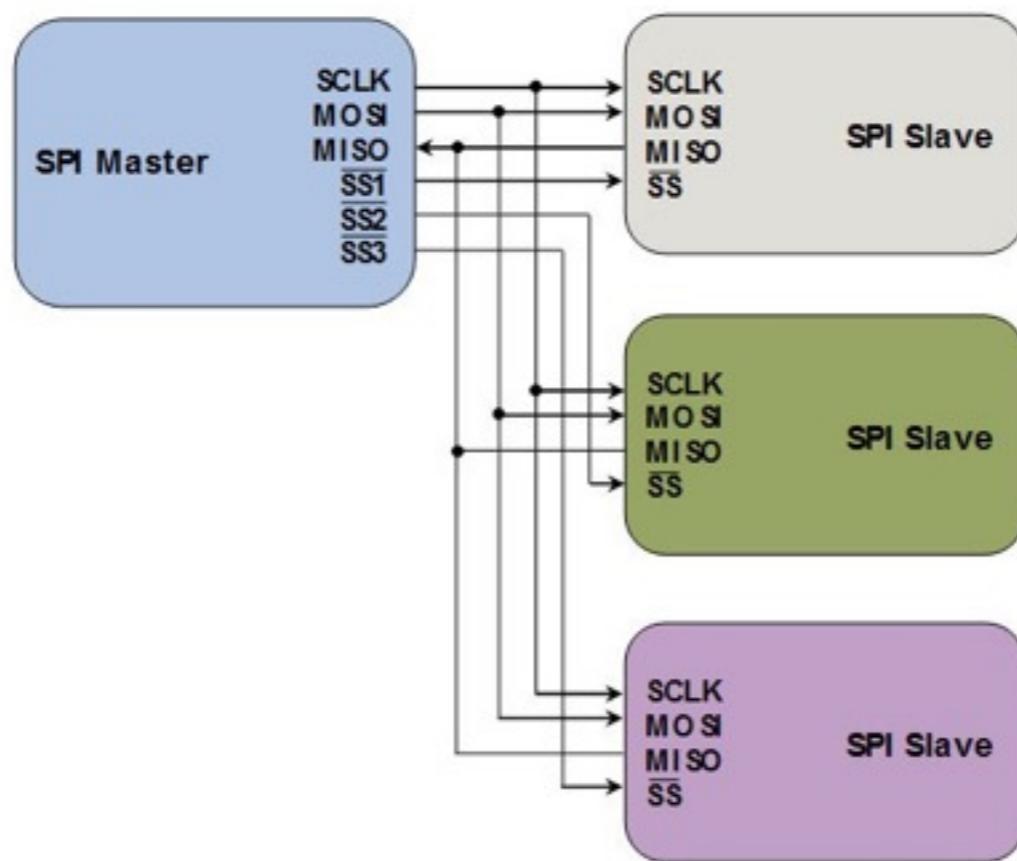
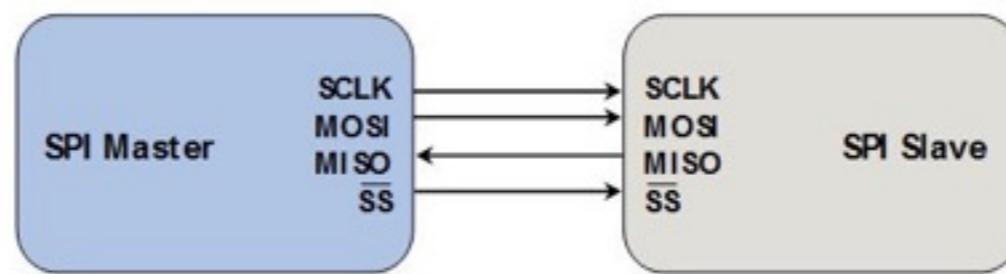


SPI (Serial Peripheral Interface)

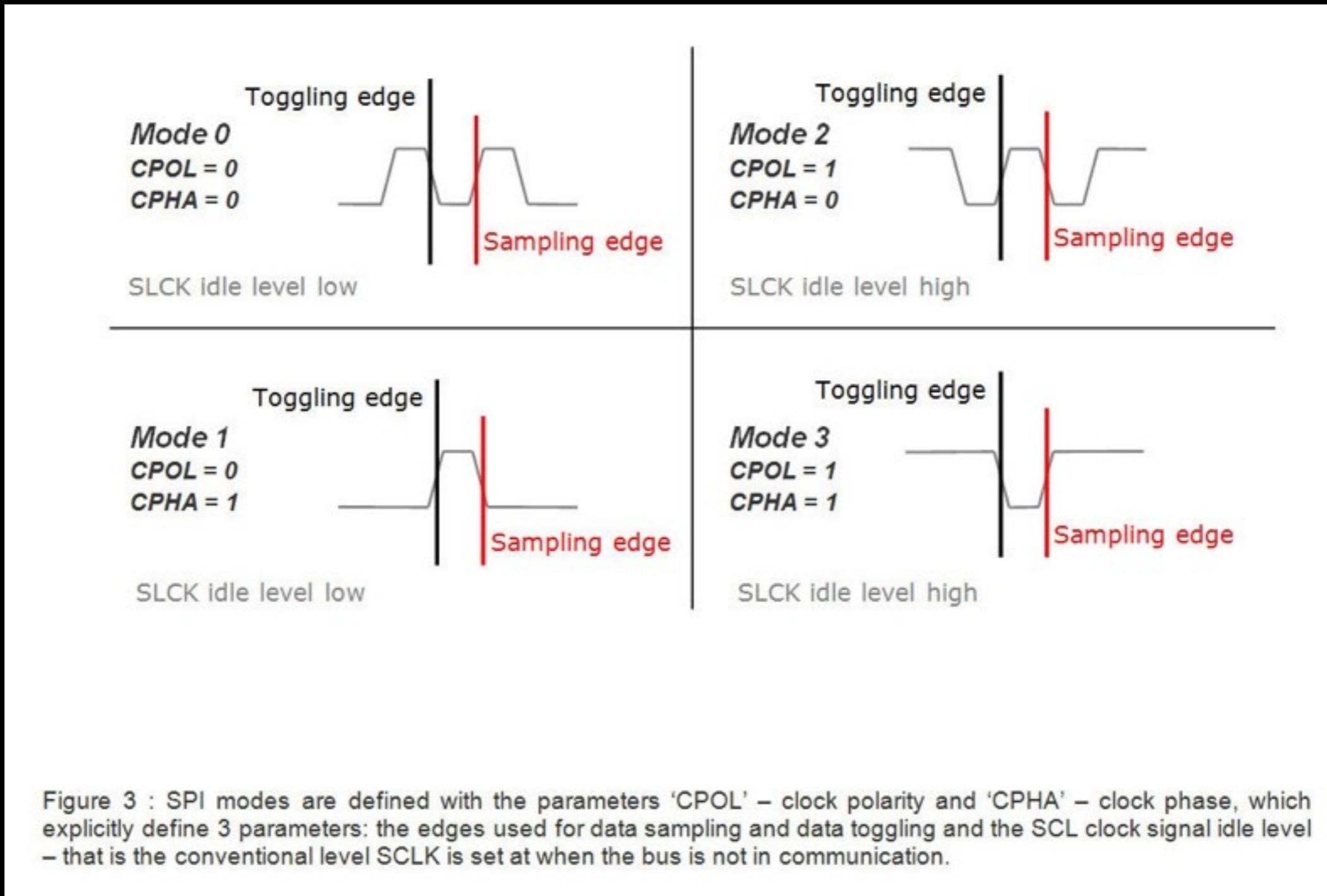


SPI wiring

Figure 1 : Two SPI busses topologies. The upper figure shows a SPI master connected to a single slave (point-to-point topology). The lower figure shows a SPI master connected to multiple slaves.



4 modes of sampling



SAMD21 Mini/Dev Breakout Hookup Guide
NOVEMBER 12, 2015
An introduction to the Atmel ATSAMD21G18 microprocessor and our Mini and Pro R3 breakout boards. Level up your Arduino-skills with the powerful ARM Cortex M0+ processor.

See all upcoming classes →

Upcoming Classes

JULY 11, 2016 **Microcontrollers for Educators**
Connecting the STEM Dots: Microprocessors and Electrical and Computer Engineering in the Classroom. Colorado State University Dep...

New!  **SAMD21 Mini/Dev Breakout Hookup Guide**
NOVEMBER 12, 2015

New! 

Recent Blog Posts

NATIONAL ROBOTICS CHALLENGE 2015
APRIL 23, 2015 0 

SPARKFUN RETURNS TO THE WHITE HOUSE!
APRIL 8, 2015 0 

ON THE ORIGIN OF THERMAL SENSORS
MARCH 3, 2015 2 

SMALL SOLDERING CLASSES AT PORT CITY MAKERSPACE
FEBRUARY 25, 2015 0 

Start a Project
If you're eager to start building and learn as you go then this will help you find the best resources to hit the ground running.

Resources
Teaching an electronics class? Just eager to learn? We've got you covered with a variety of educational resources.

Tutorials
We've got a growing collection of tutorials covering everything from Arduino to Voltage Dividers.

Workshops
We teach professional development workshops for educators all over the country.

Shop **Learn** **A/C** **Forum** **Data**

LOG IN **REGISTER**

START A PROJECT EDU BLOG RESOURCES TUTORIALS CLASSES CALENDAR WORKSHOPS CONTACT search...

fccid.io 390MHz

The screenshot shows a web browser window with the URL fccid.io/HBW1127. The page title is "FCC ID HBW1127". The main content area displays the following information:

FCC ID HBW1127
HBW-1127, HBW 1127, HBW1127, HBWII27
Chamberlain Group Inc, The Garage Door Opener Receiver

An FCC ID is the product ID assigned by the FCC to identify wireless products in the market. The FCC assigns businesses 3 or 5 character "Grantee" codes to identify the business that created the product. For example, the grantee code for FCC ID: HBW1127 is . The remaining characters of the FCC ID, 1127, are often associated with the product model, but they can be arbitrary. In addition to the application, the FCC also publishes *internal images*, *external images*, *user manuals*, and *test results* for wireless devices. They can be under the "exhibits" tab below.

Chamberlain Group Inc, The

Full Company Details: Chamberlain Group Inc, The - HBW
Company Code: HBW
Address:
Chamberlain Group Inc, The
845 Larch Avenue
Elmhurst, IL 60126
United States

Application: Garage Door Opener Receiver
Equipment Class: CRR - Superregenerative Receiver

#	Purpose	Date	Unique ID
1	Class II Permissive Change	05/05/1997	rm/mnAUuldQ4dx691v4b/g==

Approved Operating Frequencies

App # (Line Item)	Lower Frequency	Upper Frequency	Tolerance	Rule Parts	Grant Notes
1 (1)	390.00000000	390.00000000	%	15B	37

Garage door operating frequencies

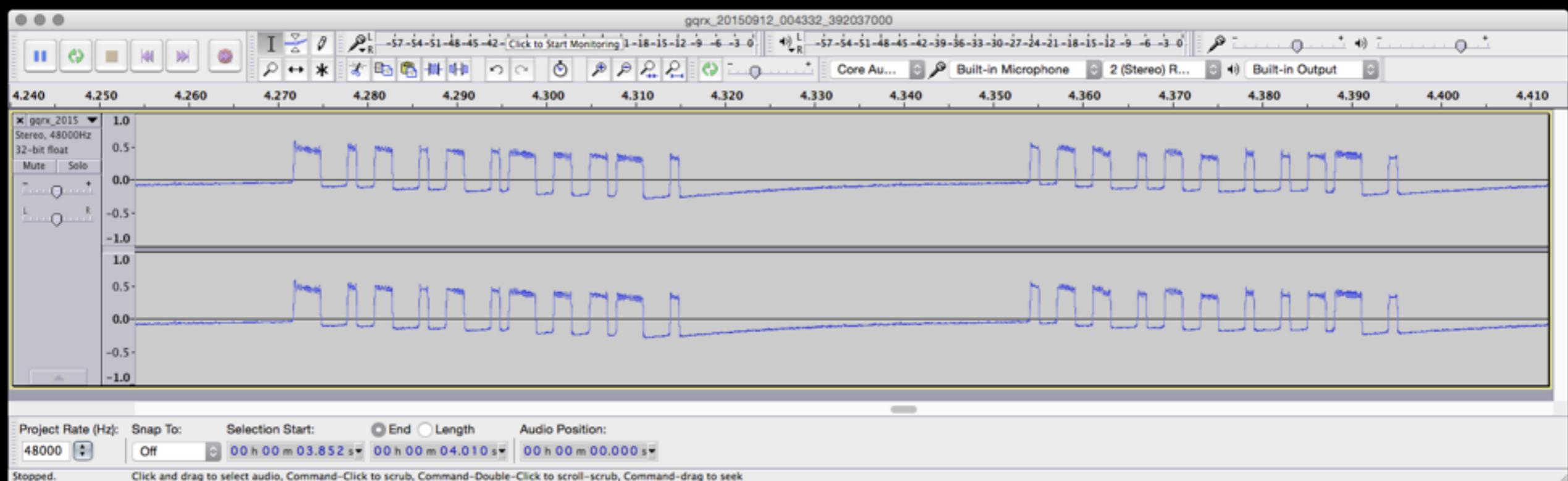
3.2 Operational Mode

For all tests, the test item was energized and placed on an 80cm high non-conductive stand. A plastic test fixture was used to continuously hold down a button on the transmitter. When the button was held down, the transmitter was transmitting continuously. When the button was released, the transmitter immediately ceased transmission. The following matrix provides a list of frequencies and modulations. The highlighted text shows the worst case frequencies and modulation codes that were tested.

Manufacturer	Code Format	Frequency
Linear Multicode	10 Position DIP Switch	300MHz
Stanley	Secure Code (NEW, Keeloq based)	310MHz
Linear/Moore-o-Matic	8 Position DIP Switch	310MHz
Stanley & Multicode	10 Position DIP Switch	310MHz
Chamberlain	NEW Rolling Code (E Code)	315MHz
Chamberlain	NEW Rolling Code (F Code)	315MHz
Chamberlain	Rolling Code (D Code)	315MHz
Genie	NEW, IntelliCode (Keeloq based)	315MHz
Chamberlain	9 Position DIP Switch, Canada	315MHz
Linear	NEW, Mega-Code	318MHz
Wayne Dalton	NEW, Rolling Code (Keeloq based)	372.5MHz
Chamberlain	Rolling Code (D Code)	390MHz
Genie	NEW, IntelliCode (Keeloq based)	390MHz
Chamberlain	7 Position DIP Switch	390MHz
Chamberlain	8 Position DIP Switch	390MHz
Chamberlain	9 Position DIP Switch	390MHz
Chamberlain	Billion Code (A Code)	390MHz
Genie	12 Position DIP Switch	390MHz
Genie	9 Position DIP Switch	390MHz

Thanks random data sheet from the internet!

Woot!



NodeSchool.io/vancouver

@nodeschoolyvr



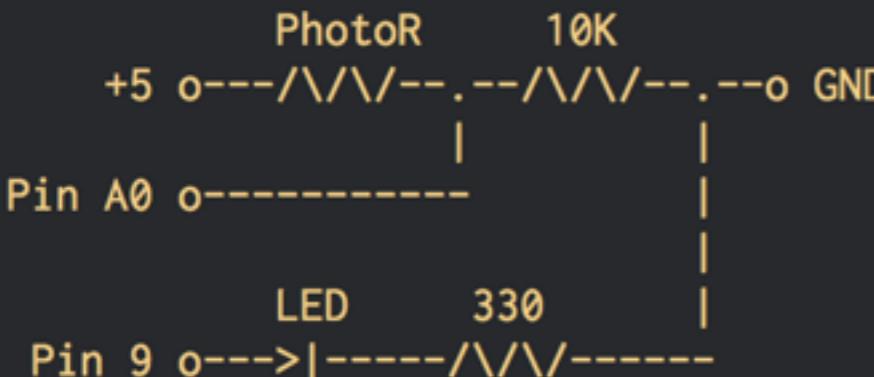
Fun and instructional

```
tmux (tmux) 301 - (zsh) 302
1. TERM=screen-256color-bce tmux attach (tmux)
Street Lamp [18/1590]
Exercise 5 of 9
```

Build a street lamp that turns on as it gets dark.

- * Use photoresistor and an LED
- * Connect the photoresistor to A0 and the LED to 9
- * Make the LED turn on when the photoresistor's value is greater than 600

Circuit diagram



Components

- * Photoresistor - <http://node-ardx.org/electronics-primer#photoresistor>
 - > Produces a variable resistance dependant on the amount of incident light.

Docs

0 gyaresu 0 shodan 1 ..noPAGERcc1101 > 2 > ~

< 18:08 < 18 Nov < zaphod

Works? Now plug it into an Arduino!

```
tmux (tmux) 301 - (zsh) 302 1. TERM=screen-256color-bce tmux attach (tmux)

## Components
* Photoresistor - http://node-ardx.org/electronics-primer#photoresistor
> Produces a variable resistance dependant on the amount of incident light.

## Docs
* Sensor - https://github.com/rwaldron/johnny-five/wiki/Sensor

## Hints
johnny-five has a generic Sensor object for handling various analog inputs.
It fires a data event with the current reading of the sensor.
The sensor value is available to the callback as this.value

-----
» To print these instructions again, run: nodebot-workshop print
» To execute your program in a test environment, run: nodebot-workshop run program.js
» To verify your program, run: nodebot-workshop verify program.js
» For help run: nodebot-workshop help

gyaresu@zaphod:~|⇒ █
□ 0 ➤ gyaresu ➤ 0 shodan 1 ..noPAGERcc1101 ➤ 2 ➤ ~ < 18:08 < 18 Nov ➤ zaphod
```

Open Source Hardware



ToorCon 13 Badge

The ToorCon 13 badge functions as an RF spectrum analyzer with the LEDs representing the 13 evenly spaced Wi-Fi channels in the 2.4 GHz band. Push the button to activate the badge for a short time. In addition to Wi-Fi, it can detect Bluetooth, ZigBee, microwave ovens, and anything else operating in the band.

The badge is designed to be enhanced. Only a few additional components are required to power it over USB. Fully populated, the badge functions as an **Ubertooth** capable of passive Bluetooth monitoring and more. Additionally, many options are open to those who would like to reprogram the microcontroller(s) on board.



[Home](#)
[Where to Buy](#)
[Upcoming Events](#)
[Free Stuff](#)
[About](#)
[Contact](#)

Products

[ANT500](#)
[HackRF One](#)
[Throwing Star LAN Tap](#)
[Ubertooth One](#)
[YARD Stick One](#)

Education

[SDR with HackRF](#)
[Technical Reports](#)

Current Projects

Loon Cup



CHANGING OUR PERIODS FOR THE BETTER



Automatically Checks
Menstrual Cycle



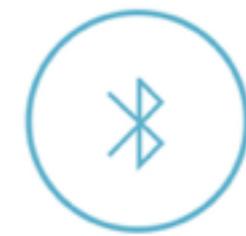
Menstrual Fluid
Volume



Menstrual Fluid
Color



Safe, 100%
Medical Grade
Silicone



Safe BLE 4.0

Internet Relay Chat (Freenode)

The screenshot shows a terminal window with a dark background. On the left is a sidebar titled 'ASDF' containing a list of channels with their names and user counts. The main area is a text-based IRC log. At the top of the log, a message from 'gyaresu' is visible: 'I've still got no shortage of things I need to learn, Programming in C, figuring out the sequence and way to programme the CC1110, and then there's the signal processing...'. The log continues with a conversation between 'AndrewMac' and 'sammy^A'. They discuss the use of rfcat for sending carrier waves via IM-ME or CC1110. AndrewMac asks if anyone has minimum viable code, and sammy^A responds that gyaresu has one. They also mention the 'im-me rfcat' and 'opensesame' tools. The conversation then shifts to the cost of the IM-ME, with AndrewMac mentioning he paid CA\$90 and sammy^A mentioning the TxMode() function in the rfcat firmware. They also discuss the availability of RF dongles and the need for a dongle. The log ends with AndrewMac asking about car remote stuff and sammy^A responding that it's not just a preamble.

Channel	Users
#ASDF	20
#rfcat	14
#goodfet	528
#hackrf	19
#portapack	1,450
#osmocom	251
#ubertooth	41
#cyberspectrum	99
#hackallthethings	1,865
#gnuradio	3
#rtl-sdr	1,787
#pentoo	10
#debian	212
#kali-linux	201
#rhel	1,666
#Greenpeace	1,432
#leveldb	703
#puppet	728
#python	5
#osx-server	1,787
#macit	10
#maedevops	207
#mingle	1,787
#centos	1,787
#centos-social	1,787
#Node.js	1,787
#gyaresu	1,787
#machomebrew	1,787
#vim	1,787
#nodeschool	1,787
#R	1,787
#leveldb	1,787
#fosdem	1,787
#google-corpeng	1,787
#raspberrypi	1,787
#ubuntu	1,787
#javascript	1,787
#bash	1,787
#occ	1,787
#hapi	1,787
#openbits	1,787
#hapijs	1,787
#python-unregistered	1,787
#ampersandjs	1,787
#ubuntu-unregged	1,787
#textual	1,787
#homebrew-cask	1,787
#nodebots	1,787

gyaresu on freenode IRC Network — #rfcat (20 users) (-cnt)

I've still got no shortage of things I need to learn, Programming in C, figuring out the sequence and way to programme the CC1110, and then there's the signal processing...

But I was just wondering if anyone had a minimum viable code for sending (for instance) a carrier wave via the IM-ME or CC1110 in general. Something that shows the sequence of 'things to know'. Bonus points if there's a video walkthrough :)

AndrewMac: gyaresu: samy has one

gyaresu: the im-me rfcat does this just steal it from opensesame

AndrewMac: yeah? I've been looking at the opensesame and specan speccan is a bit more complex but opensesame just TX's so just use the TX code to send out a signal?

gyaresu: AndrewMac yeah but I don't really understand enough C quite yet to know which functions are missing params etc. :) But the im-me rfcat is not a bad idea.

AndrewMac: do you have rfcat on your c1110 ? is it the EMK ?

samy^A: the TxMode() function in the rfcat firmware produces a carrier wave, though doesn't have all the initial settings you need (which you could gank from opensesame)

gyaresu: AndrewMac Nope. I've only got the im-me for now. I have a bunch of the RF dongle's coming from ebay and will be getting the YS1 soon enough.

AndrewMac: wanna sell your IM-ME? :P

gyaresu: AndrewMac I paid CA\$90 for it just to have a replicable device :p

AndrewMac: I'll trade you my CC1111EMK for it xD

gyaresu: AndrewMac And that was second hand without a dongle

AndrewMac: you dont need the dongle do you? yeah they are few and far between

gyaresu: samy^A Cool. That helps to know the initial settings are working. I obviously want to get a carrier wave going but I'm still at the "Which type of memory is goodfet.cc peek 0xFF actually looking at" stage.

AndrewMac: if anyone finds one please lemme know

gyaresu: AndrewMac correct. Dongle not required.

AndrewMac: But it doesn't load rfcat and I'm kind of ok wading around in the Weeds of Ignorance™ with regard to learning how to access the chip directly.

gyaresu: hmmm even with rfcat on it it gives you the option to set the registers directly to do what you want

AndrewMac: Don't get me wrong, I'm looking forward to calling methods instead of reflashing hex files :) I just still need to wrap my head around what the hell I'm doing. There's no real primer on this that I've found to be a single point of entry. It's the blog posts, code and help here that we all know and love.

samy^A: How was the crashspace gig? Did you have fun?

gyaresu: yeah, it was fun! cool to do the smaller environments where you can talk with everyone

gyaresu: samy^A Great. Good on you for doing that. Glad it went well.

AndrewMac: anyone besides samy played with car remote stuff?

gyaresu: looking at VW's there seems to be 5 *very* long code sents and then the remote just repeats a much smaller (more normal length) code

samy^A: AndrewMac: it's not just a preamble? got a wav/pic?

AndrewMac: sure

gyaresu: 1 sec

AndrewMac: prvtd

+ ⚙️ 🚭 Send message...

AndrewMac cd1zz cybergibbons dominlogs dragorn drone elasticninja gyaresu KvH Miek mossmann mybit RedBaron russm s7oneghos7 samy^A sharebrained souca_away ukid Zero_Chaos

People are amazing

2013 LCTHW

The screenshot shows a Mac OS X desktop with an iTerm window and a tmux session running in a terminal window.

iTerm: The title bar says "Exercise 9: Arrays And Strings". The content pane displays the C code for Exercise 9 from [Learn C The Hard Way](http://c.learncodethehardway.org/book/ex9.html).

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int numbers[4] = {0};
    char name[4] = {'a'};

    // first, print them out raw
    printf("numbers: %d %d %d %d\n",
           numbers[0], numbers[1],
           numbers[2], numbers[3]);

    printf("name each: %c %c %c %c\n",
           name[0], name[1],
           name[2], name[3]);

    printf("name: %s\n", name);

    // setup the numbers
    numbers[0] = 1;
    numbers[1] = 2;
    numbers[2] = 3;
    numbers[3] = 4;

    // setup the name
    name[0] = 'Z';
    name[1] = 'e';
    name[2] = 'd';
    name[3] = '\0';

    // then print them out initialized
    printf("numbers: %d %d %d %d\n",
           numbers[0], numbers[1],
           numbers[2], numbers[3]);

    printf("name each: %c %c %c %c\n",
           name[0], name[1],
           name[2], name[3]);

    // print the name like a string
    printf("name: %s\n", name);

    // another way to use name
    char *another = "Zed";

    printf("another: %s\n", another);

    printf("another each: %c %c %c %c\n",
           another[0], another[1],
           another[2], another[3]);
}
```

tmux: The title bar says "1. tmux (bash)". The content pane displays the C code for Exercise 9.

```
12 #include <stdio.h>
13
14 int main(int argc, char *argv[])
15 {
16     char full_name[] = {
17         'Z', 'e', 'd',
18         ' ', 'A', ' ', ' ',
19         'S', 'h', 'a', 'w', '\0'
20     };
21     int areas[] = {100, 12, 13, 14, 20};
22     char name[] = "Zed";
23
24     areas[0] = '42';
25
26     // WARNING: On some systems you may have to change the
27     // %ld in this code to a %u since it will use unsigned ints
28     printf("The size of an int: %ld\n", sizeof(int));
29     printf("The size of areas (int[]): %ld\n",
30           sizeof(areas));
31     printf("The number of ints in areas: %ld\n",
32           sizeof(areas) / sizeof(int));
33     printf("The first area is %d, the 2nd %d.\n",
34           areas[0], areas[1]);
35
36     printf("The size of a char: %ld\n", sizeof(char));
37     printf("The size of name (char[]): %ld\n",
38           sizeof(name));
39     printf("The number of chars: %ld\n",
40           sizeof(name) / sizeof(char));
41
42     printf("The size of full_name (char[]): %ld\n",
43           sizeof(full_name));
44     printf("The number of chars: %ld\n",
45           sizeof(full_name) / sizeof(char));
46
47     printf("name=\"%s\" and full_name=\"%s\"\n",
48           name, full_name);
49
50     return 0;
51 }
```

Terminal: The title bar says "ex8.c". The status bar shows "N ex8.c ERRORS (1) 13 unix < utf-5 < c 4 23"ex8.c"5413, warning: multi-character character constant [-Wmultichar]" and "vagrant < 0 > ..ramming_stuff < 18:27 < 02 Jun precise64".

2015 LCTHW

The image shows a Mac OS X desktop with two windows open. On the left is an iTerm window titled "TERM=screen-256color-bce tmux attach (tmux)". It displays the source code for "ex7.c" and its output from Valgrind. The code calculates bug counts and percentages. The Valgrind output shows memory usage statistics. On the right is a web browser window titled "c.learncodethehardway.org/boo...". It shows a page with text explaining the code, with numbered callouts pointing to specific parts of the text and code. Navigation buttons for "MAIN", "PREVIOUS", "NEXT", and "HELP" are visible on the right.

"Press ? for help
.. (up a dir)
</learn-c-the-hard-way/
2013-05-19/
ex1.c
ex3.c
ex4.asm
ex4.c
ex4.ihx
ex4.lk
ex4.lst
ex4.map
ex4.mem
ex4.rel
ex4.rst
ex4.sym
ex5.c
ex6.c
ex7.c
lmod.c
Makefile

```
1. TERM=screen-256color-bce tmux attach (tmux)
1 #include <stdio.h>
2
3 int main(int argc, char *argv[]) {
4     int bugs = 100;
5     double bug_rate = 1.2;
6
7     printf("You have %d bugs at the imaginary rage of %f.\n",
8           bugs, bug_rate);
9
10    long universe_of_defects = 1L * 1024L * 1024L * 1024L;
11    printf("The entire universe has %ld bugs.\n",
12          universe_of_defects);
13
14    double expected_bugs = bugs * bug_rate;
15    printf("You are expected to ahve %F bugs.\n",
16          expected_bugs);
17
18    double part_of_universe = expected_bugs /
19    universe_of_defects;
20    printf("That is only a %e portion of the universe.\n",
21          part_of_universe);
22
23    // this makes no sense, just a demo of something weird
24    char nul_byte = '\0';
25    int care_percentage = bugs * nul_byte;
26    printf("Which means you should care %d%%.\n",
27          care_percentage);
28
29    return 0;
30 }
```

~/programming/learn-c-the-har> NORMAL ➜ ex7.c ➤ unix < utf-8 < c ➤ 3% ➤ 1:1

==12729== Using Valgrind-3.11.0.SVN and LibVEX; rerun with -h for copyright info [6/1325]
==12729== Command: ./ex7
==12729==
You have 100 bugs at the imaginary rage of 1.200000.
The entire universe has 1073741824 bugs.
You are expected to ahve 120.000000 bugs.
That is only a 1.117587e-07 portion of the universe.
Which means you should care 0%.
==12729==
==12729== HEAP SUMMARY:
==12729== in use at exit: 38,839 bytes in 418 blocks
==12729== total heap usage: 518 allocs, 100 frees, 45,815 bytes allocated
==12729==
==12729== LEAK SUMMARY:
==12729== definitely lost: 0 bytes in 0 blocks
==12729== indirectly lost: 0 bytes in 0 blocks
==12729== possibly lost: 0 bytes in 0 blocks
==12729== still reachable: 148 bytes in 3 blocks
==12729== suppressed: 38,691 bytes in 415 blocks
0 ➜ gyaresu ➜ ..the-hard-way ➜ 1 ~/dotfiles

Inbox – Amazon Payments: A... TweetDeck Learn C The Hard Way Printf format strings - Cpro...

Here's what's going on in this little bit of nonsense:

ex7.c:1-4 The usual start of a C program.

ex7.c:5-6 Declare an `int` and `double` for some fake bug data.

ex7.c:8-9 Print out those two, so nothing new here.

ex7.c:11 Declare a huge number using a new type `long` for storing big numbers.

ex7.c:12-13 Print out that number using `%ld` which adds a modifier to the usual `%d`. Adding 'l' (the letter ell) means "print this as a long decimal".

ex7.c:15-17 Just more math and printing.

ex7.c:19-21 Craft up a depiction of your bug rate compared to the bugs in the universe, which is a completely inaccurate calculation. It's so small though that we have to use `%e` to print it in scientific notation.

ex7.c:24 Make a character, with a special syntax `'\0'` which creates a 'nul byte' character. This is effectively the number 0.

ex7.c:25 Multiply bugs by this character, which produces 0 for how much you should care. This demonstrates an ugly hack you find sometimes.

ex7.c:26-27 Print that out, and notice I've got a `%%` (two percent chars) so I can print a '%' (percent) character.

ex7.c:28-30 The end of the `main` function.

This bit of source is entirely just an exercise, and demonstrates how some math works. At the end, it also demonstrates something you see in C but not in many other languages. To C, a "character" is

Gareth
me@gareth.codes

[twitter.com](https://twitter.com/gyaresu)
[github.com](https://github.com/gyaresu)

@gyaresu

