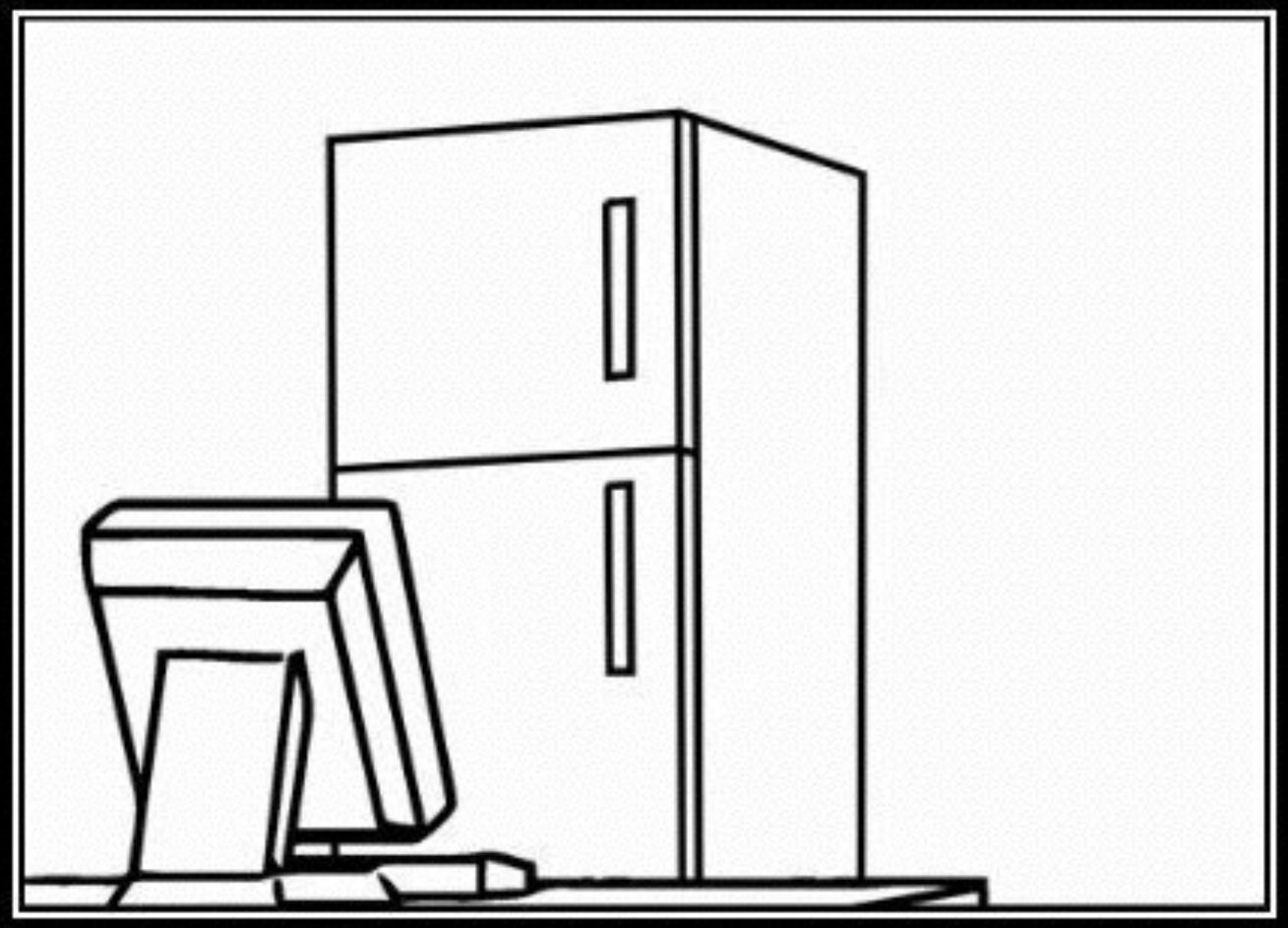
“Wait, what?”

A Practical Introduction to the Internet of Things



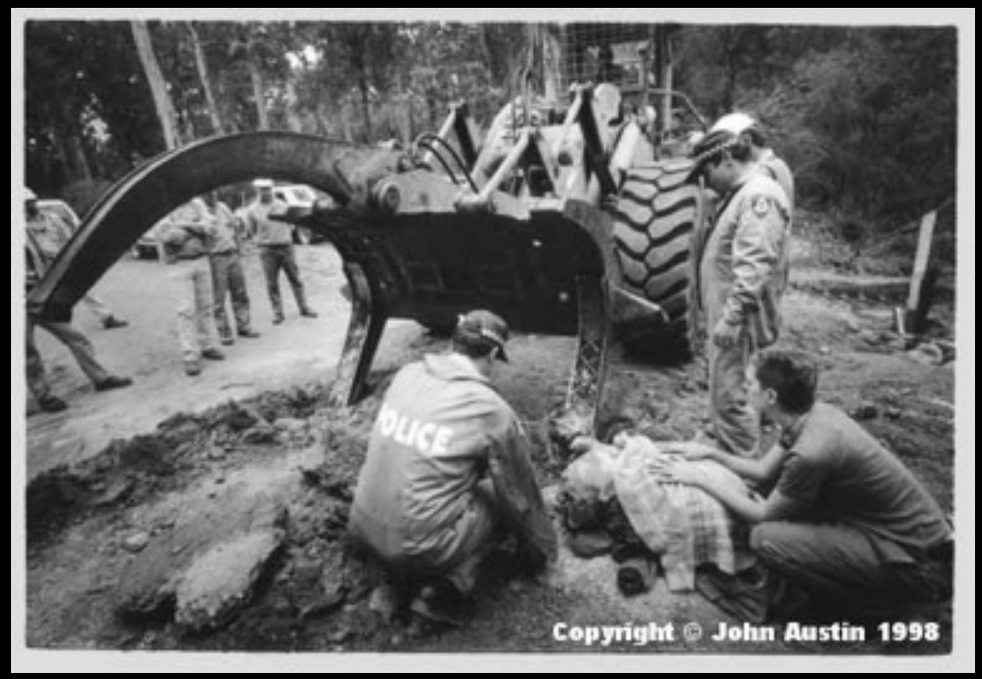
Gareth

@gyaresu

Tasmania

Go away, you wouldn't like it.


Copyright © John Austin 1998


POLICE
Copyright © John Austin 1998


Copyright © John Austin 1998

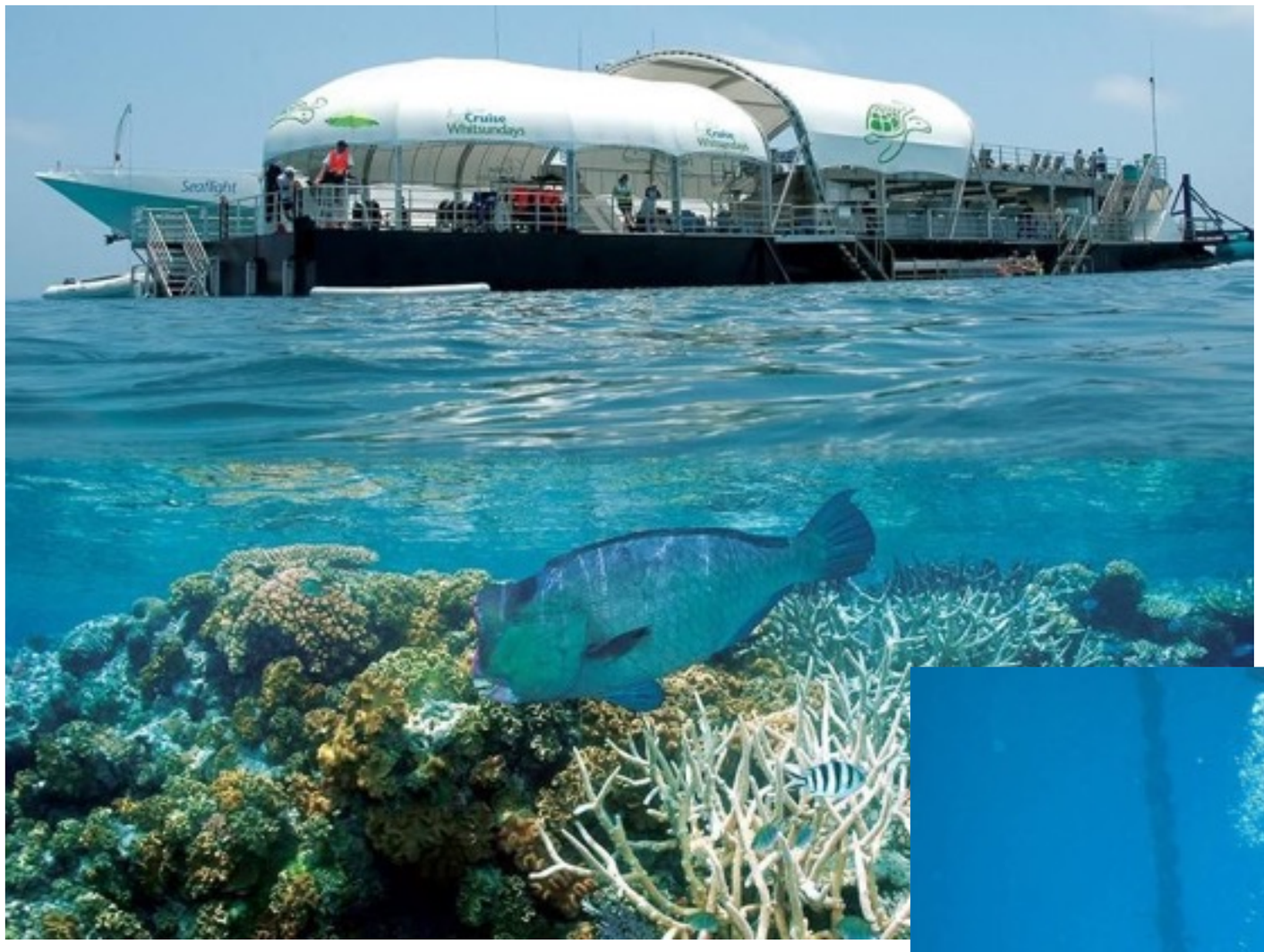

Copyright © John Austin 1998


Copyright © John Austin 1998


Copyright © John Austin 1999

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

“He who shall not answer your stupid questions”


Cruise Whitsundays
Seatlight

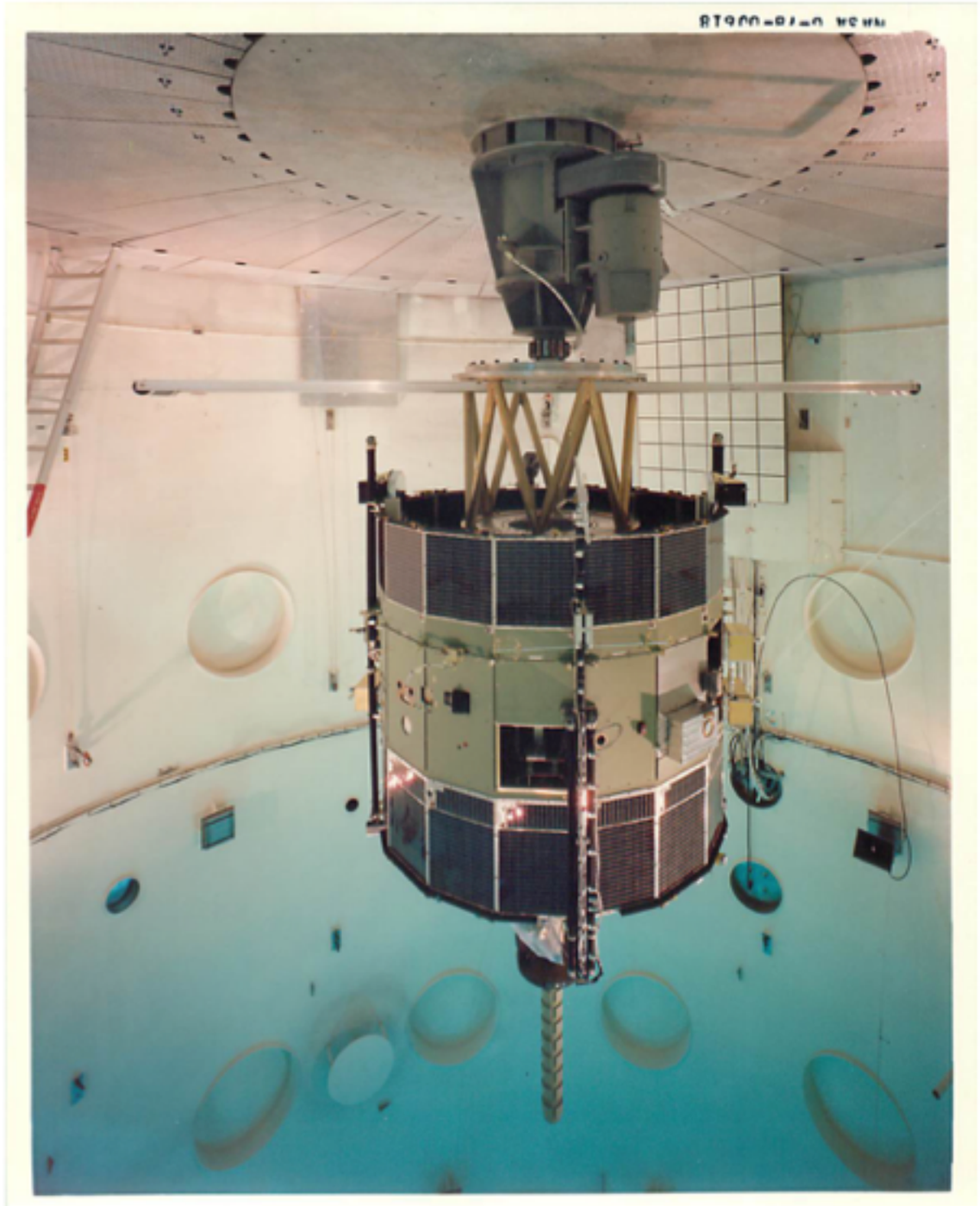

NORTHERN TERRITORY
QUEENSLAND
Australia

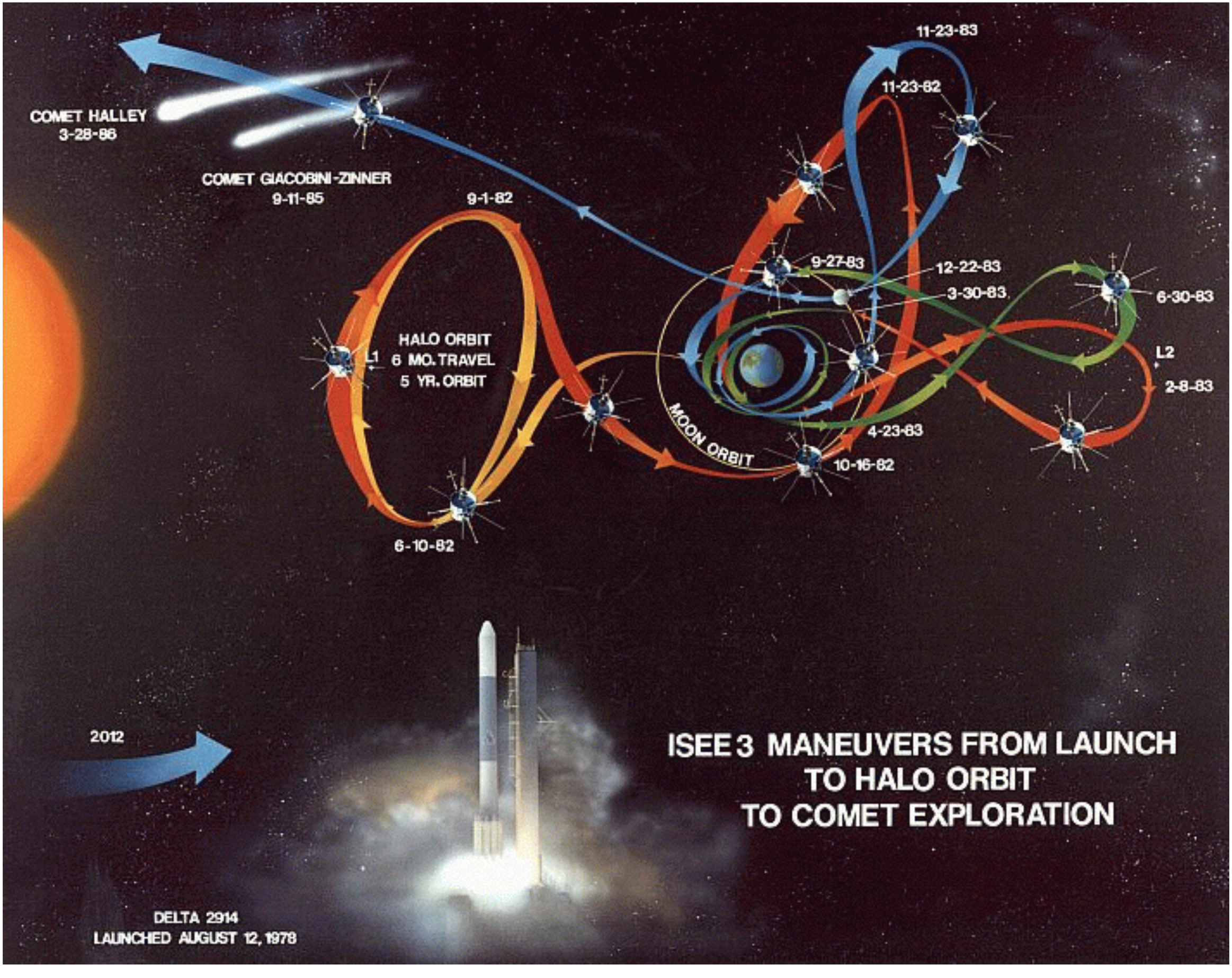

GROUNDHOG
DAY

Greenpeace International

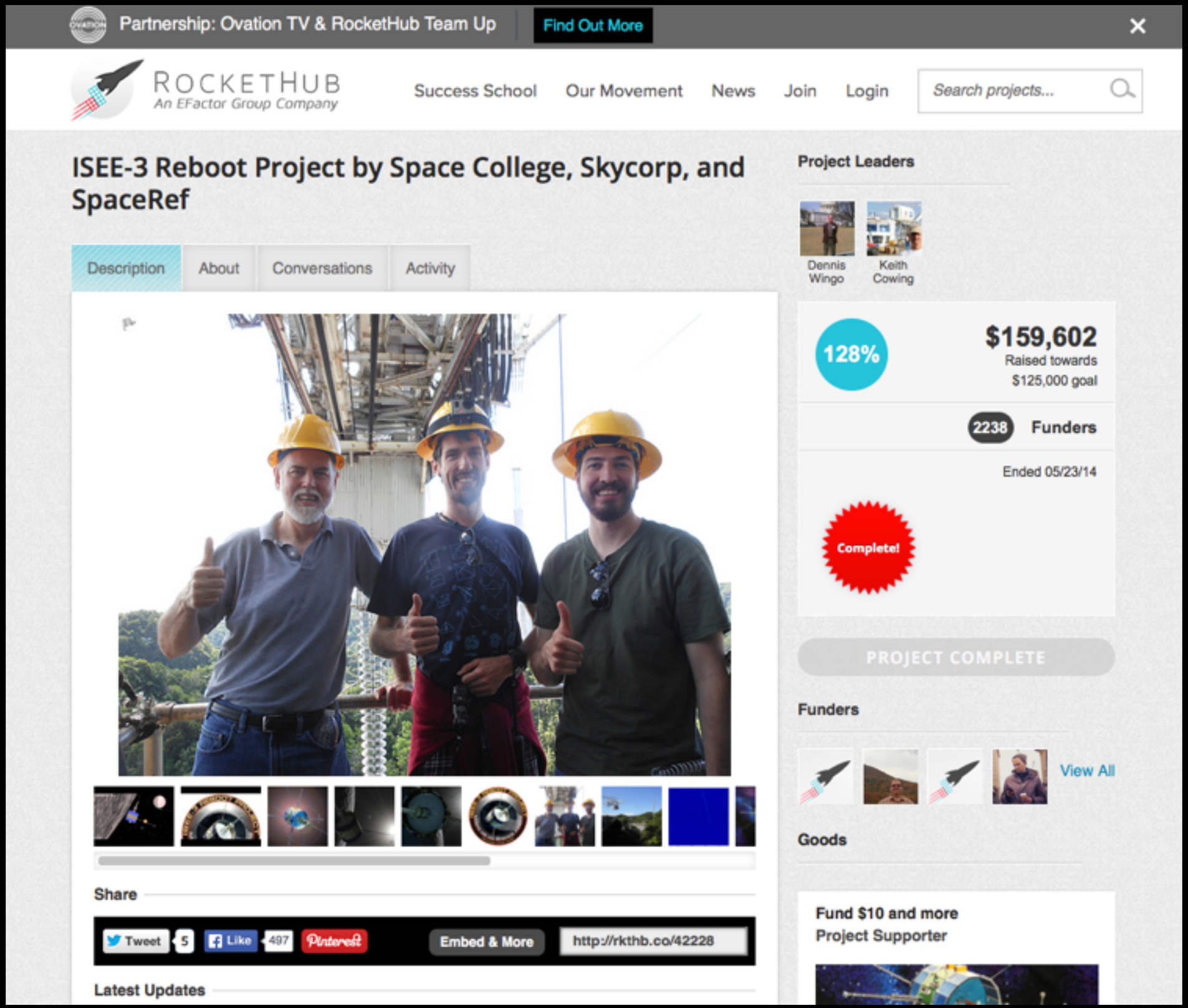
"A beige building in Amsterdam"

Computer Security Researcher




COMET HALLEY
3-28-86
COMET GIACOBINI-ZINNER
9-11-85
9-1-82
HALO ORBIT
6 MO. TRAVEL
5 YR. ORBIT
L1
6-10-82
MOON ORBIT
11-23-83
11-23-82
9-27-83
12-22-83
3-30-83
6-30-83
L2
2-8-83
4-23-83
10-16-82
2012
DELTA 2914
LAUNCHED AUGUST 12, 1978
ISEE 3 MANEUVERS FROM LAUNCH
TO HALO ORBIT
TO COMET EXPLORATION

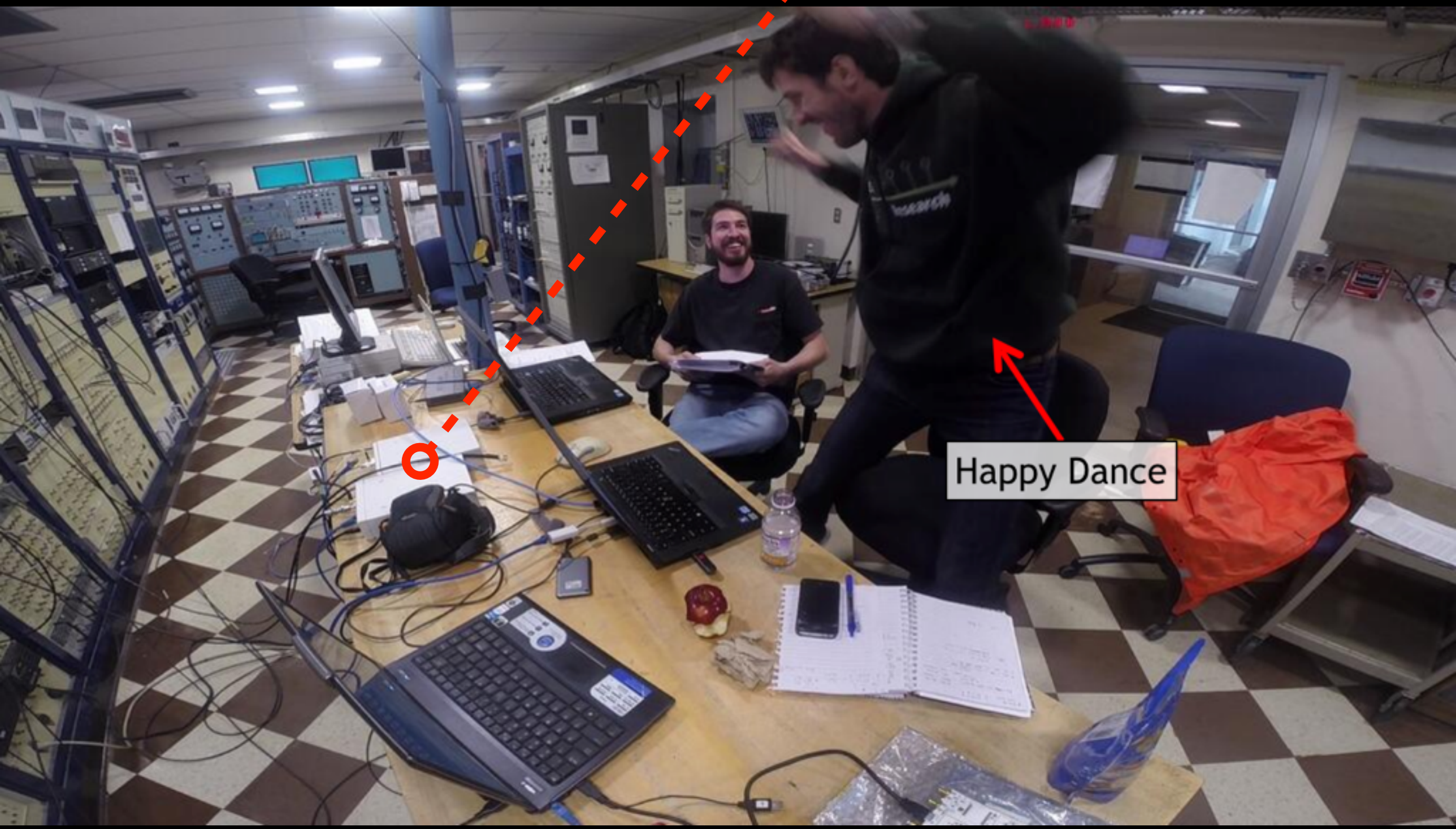
ICEE-3 Reboot Project




Gareth
Puerto Rico
United States
Mexico
Gulf of Mexico
Cuba
Arecibo Observatory
Observatory & national research center
Guatemala
Honduras
Nicaragua
Costa Rica
Panama
Caribbean Sea
Venezuela
Colombia
Ecuador
Peru
Brazil
Bolivia
French Guiana
Guyana
Suriname
North Atlantic Ocean
Spain
Morocco
Western Sahara
Mali
Burkina Faso
Senegal
The Gambia
Guinea
Sierra Leone
Liberia
Google

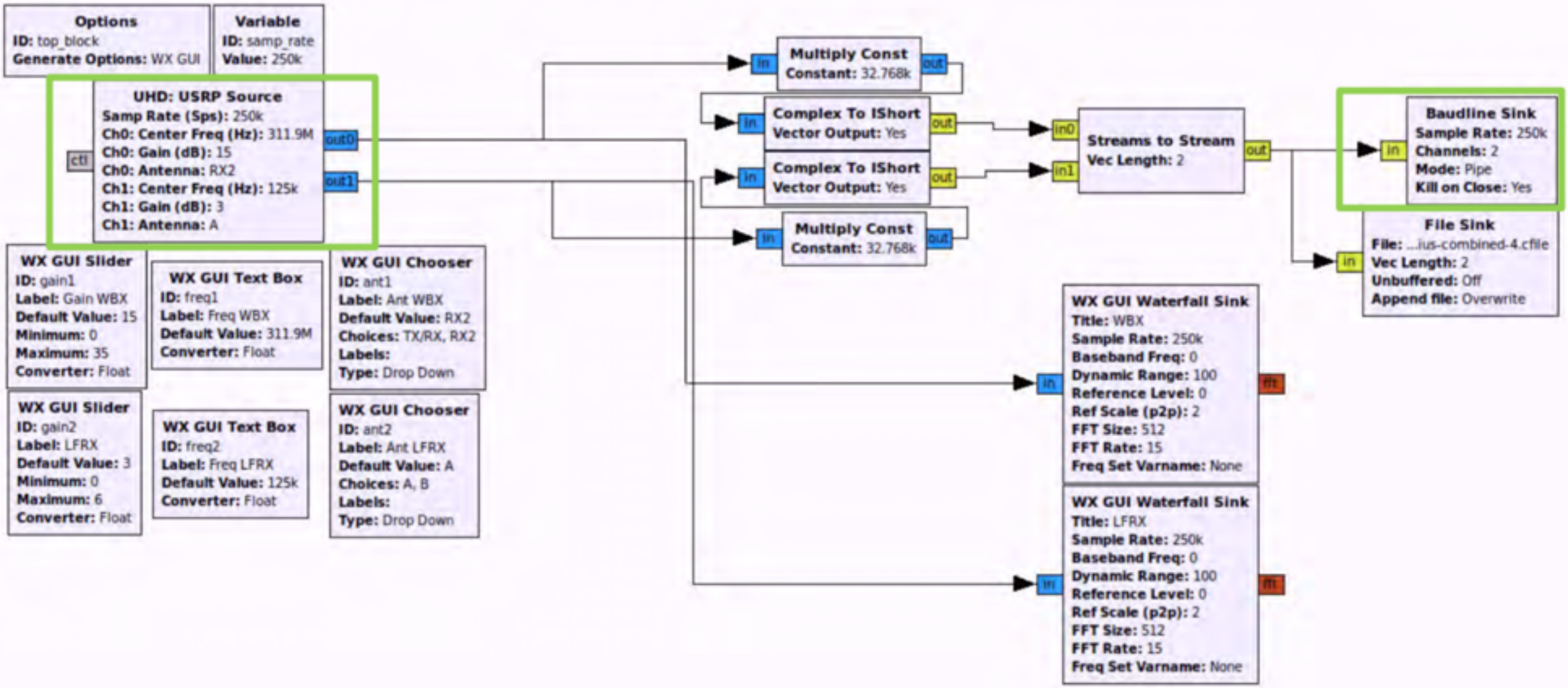
Arecibo Radio Telescope

Software Defined Radios (SDR)



GNU Radio looks wacky but is totally boss

GNU Radio + baudline



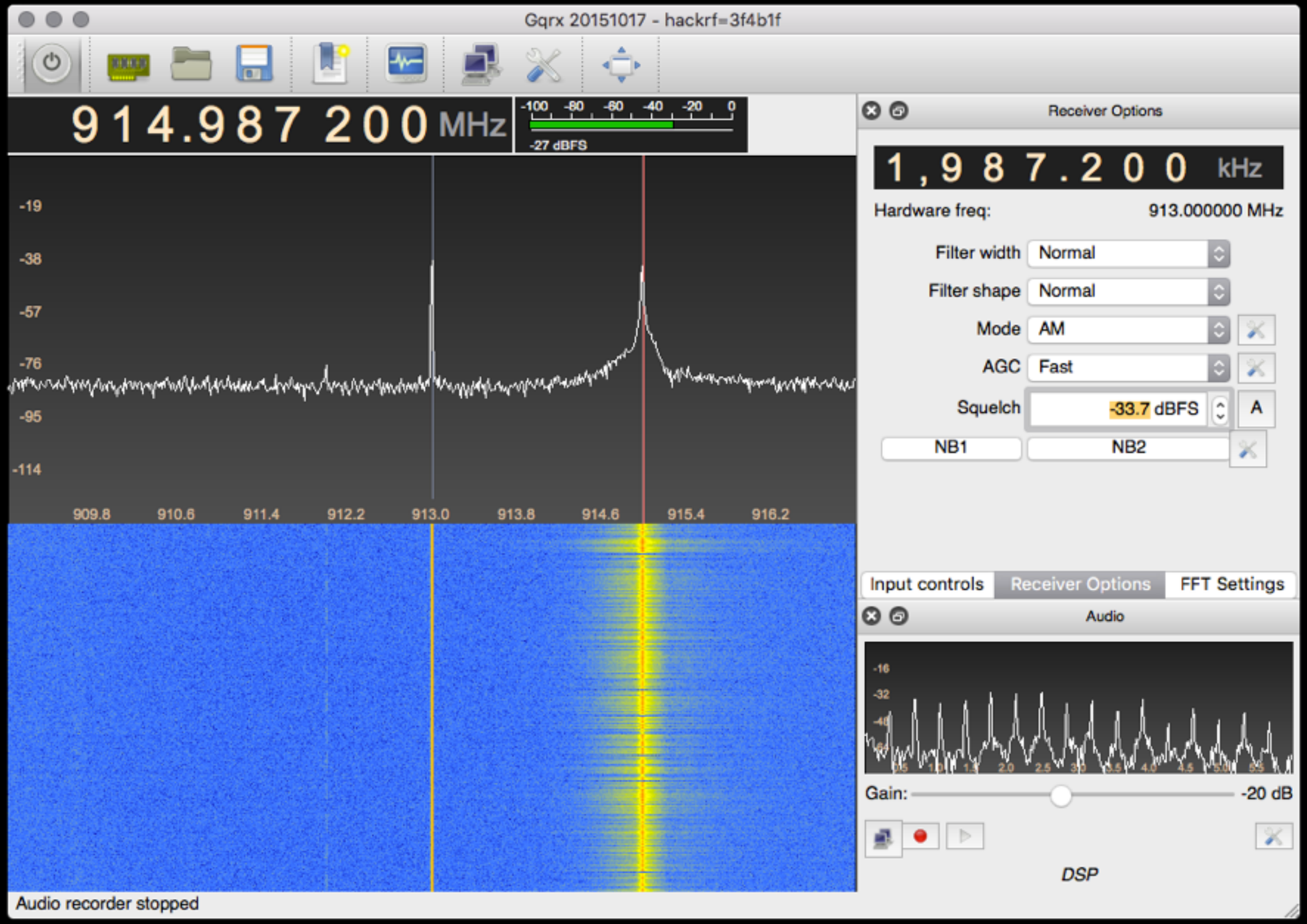
“RTLSDR”
Realtec RTL2832U chip
Used in DVB-T Dongles (5yrs)

HackRF One

0 - 6GHz ($300)

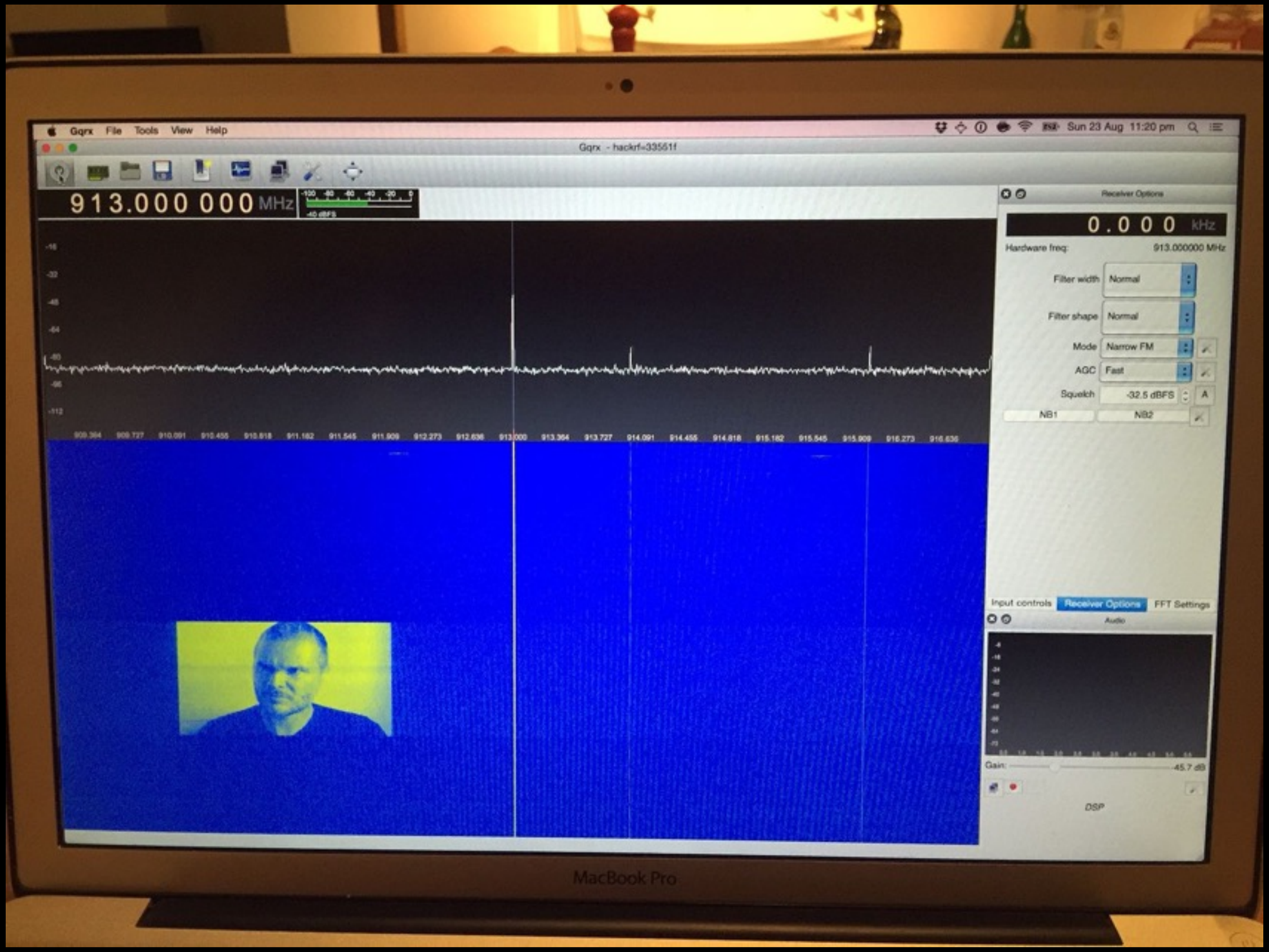
http://greatscottgadgets.com/sdr/

GQRX

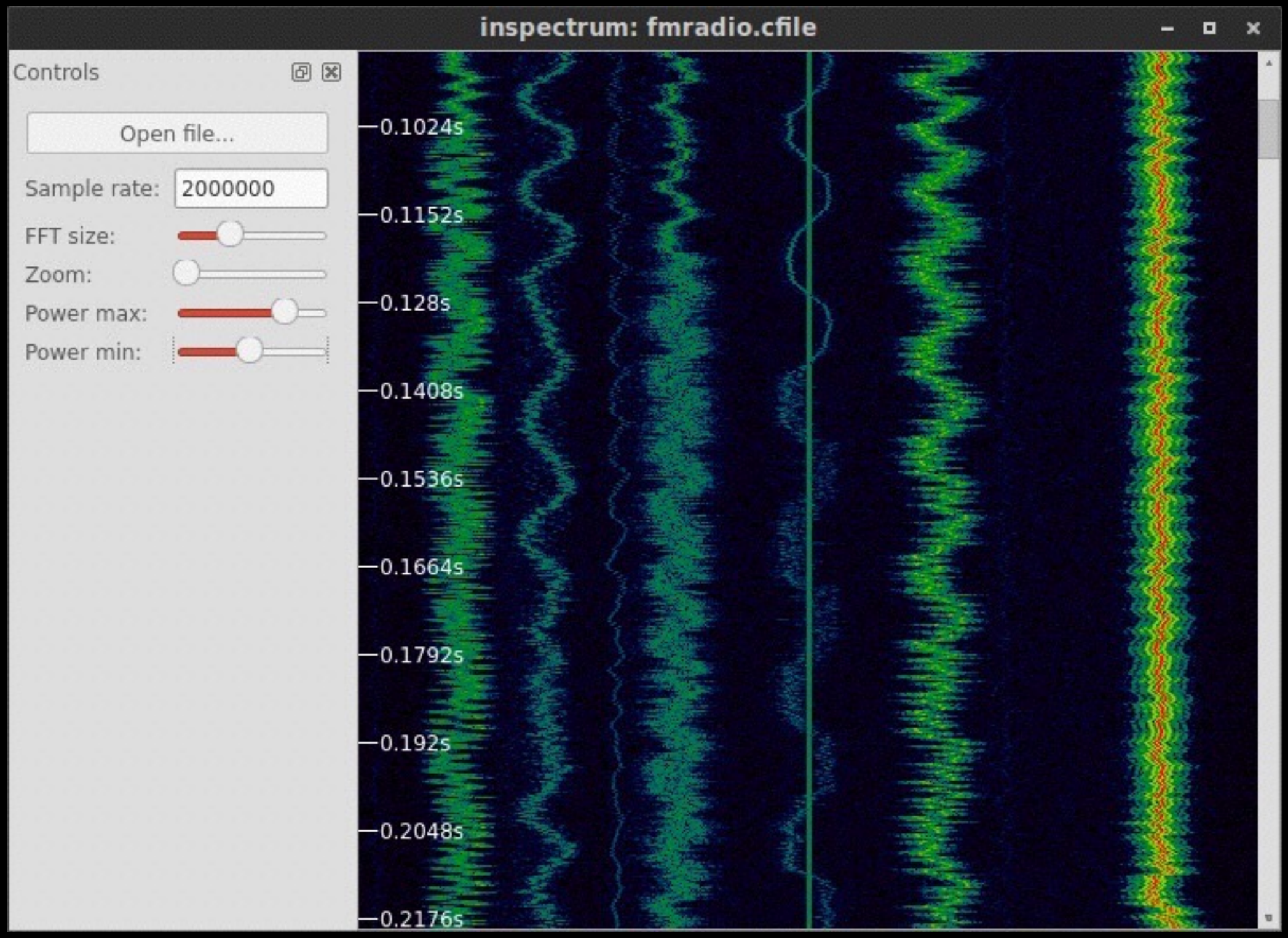

Gqrx 20151017 - hackrf=3f4b1f
914.987 200 MHz
-100 -80 -60 -40 -20 0
-27 dBFS
-19
-38
-57
-76
-95
-114
909.8 910.6 911.4 912.2 913.0 913.8 914.6 915.4 916.2
Receiver Options
1,987.200 kHz
Hardware freq: 913.000000 MHz
Filter width Normal
Filter shape Normal
Mode AM
AGC Fast
Squelch -33.7 dBFS
A
NB1
NB2
Input controls
Receiver Options
FFT Settings
Audio
-16
-32
Gain: -20 dB
DSP
Audio recorder stopped

Spectrum Painter

polygon/spectrum_painter



miek/inspectrum



$ hackrf_transfer

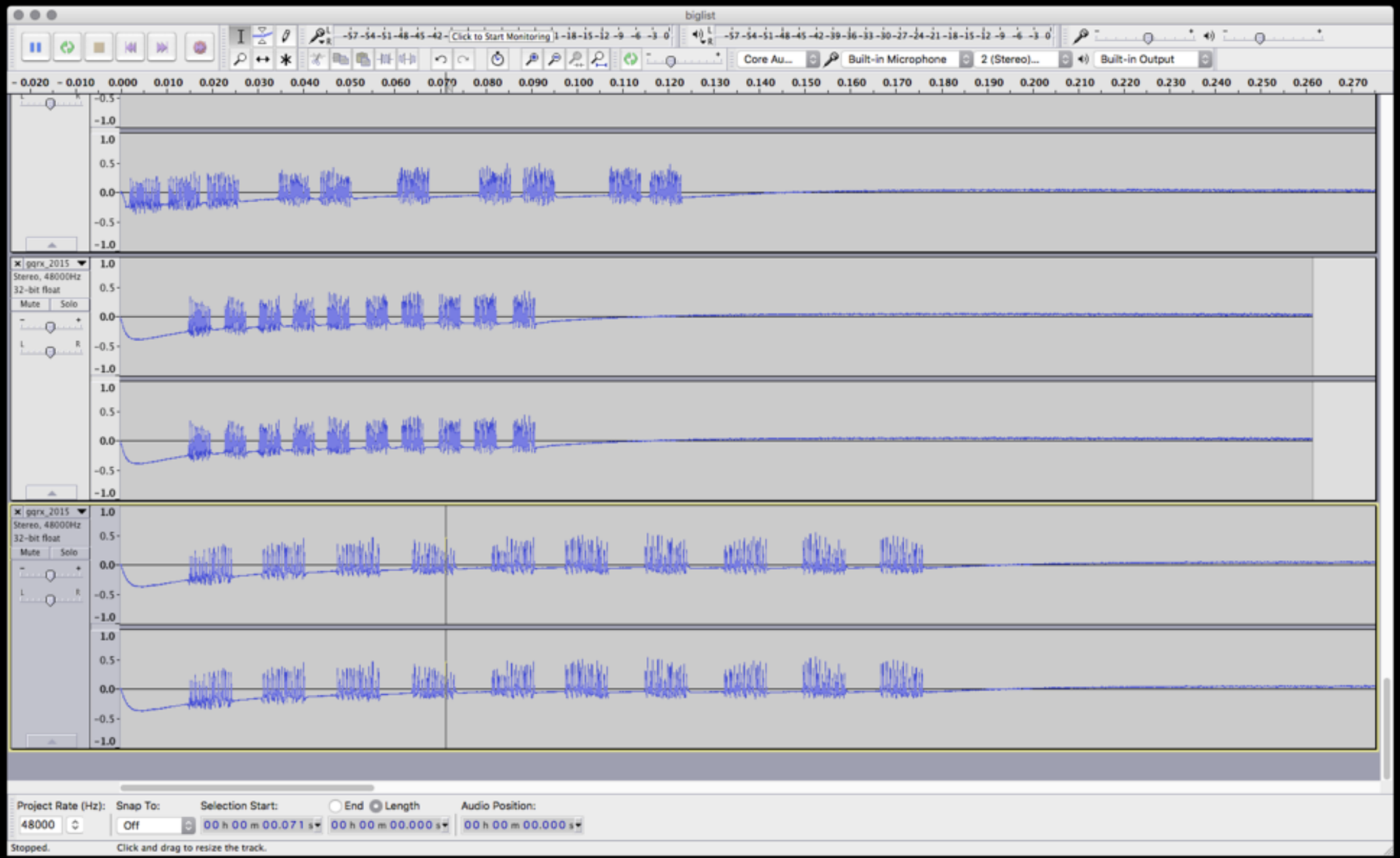
1. TERM=screen-256color-bce tmux attach (tmux)

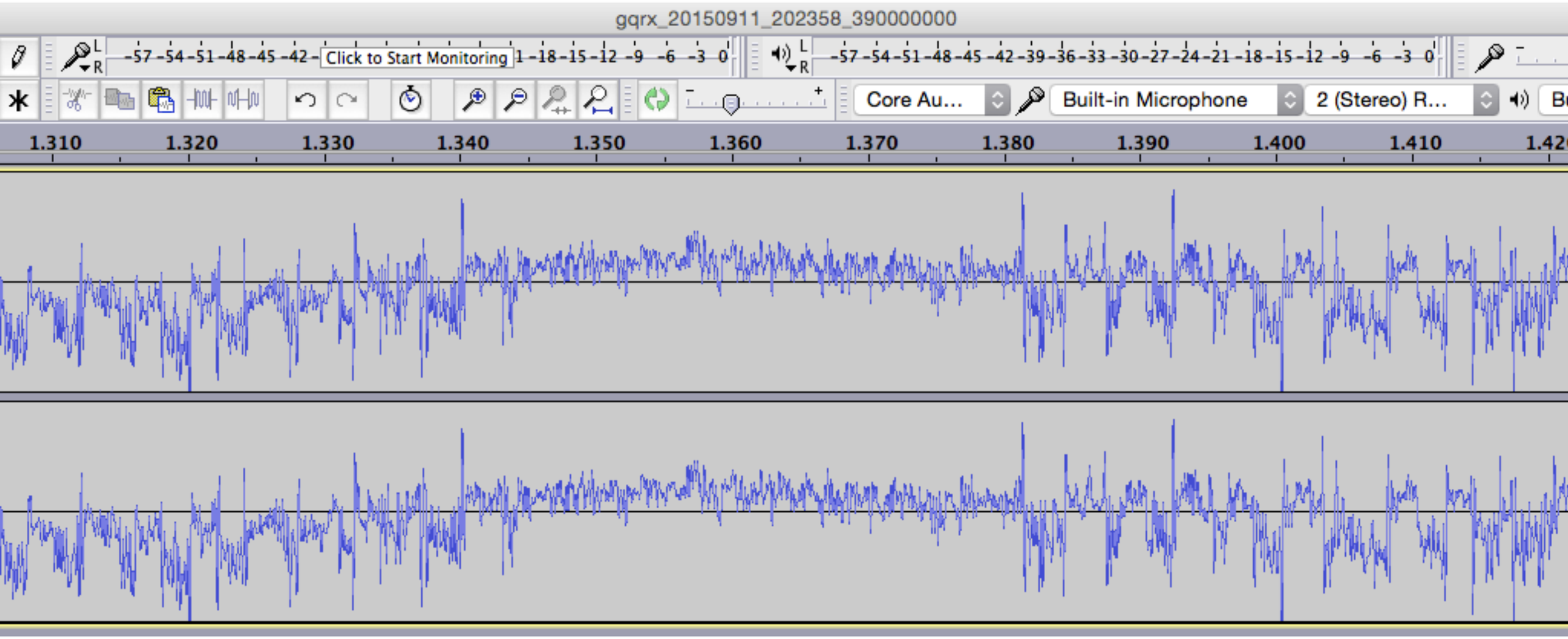
tmux (tmux) ⌘1 ~ (zsh) ⌘2

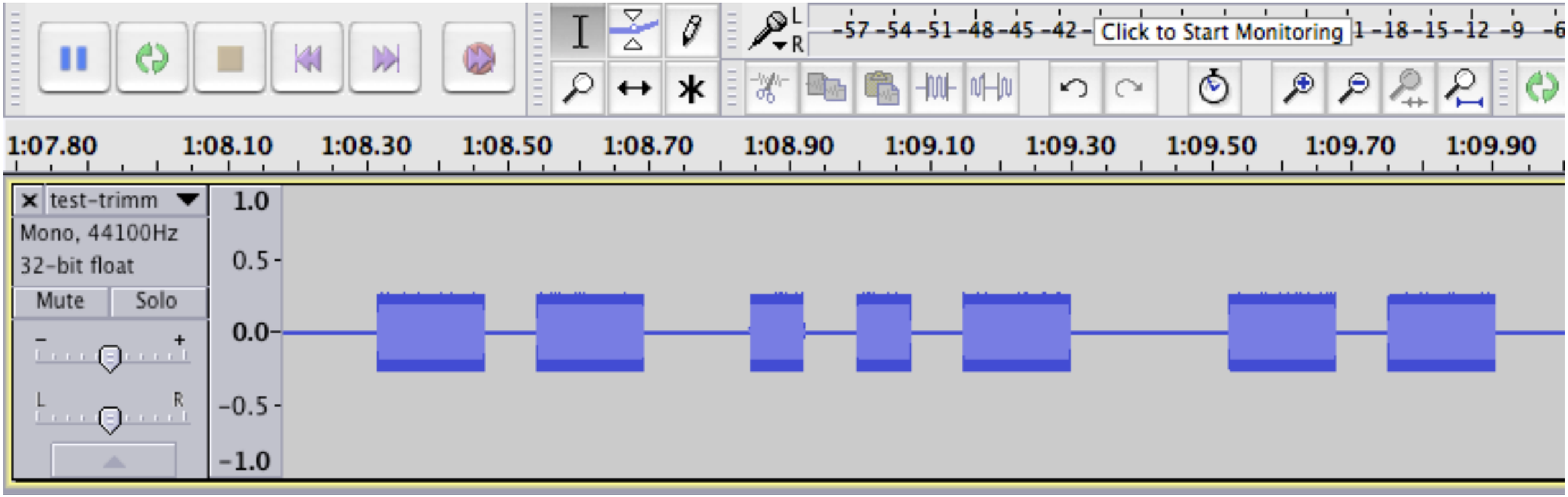
```
7732  hackrf_transfer -r 391MHz-8M-8bit.iq -f 391000000 -s 8000000
7733  hackrf_transfer -r 392MHz-8M-8bit.iq -f 392000000 -s 8000000
7746  vi transfer-bits.js
7748  node transfer-bits.js
7900  hackrf_transfer -h
7901  hackrf_transfer -c 127 -f 850000000
7902  hackrf_transfer
7903  hackrf_transfer -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|⇒
```

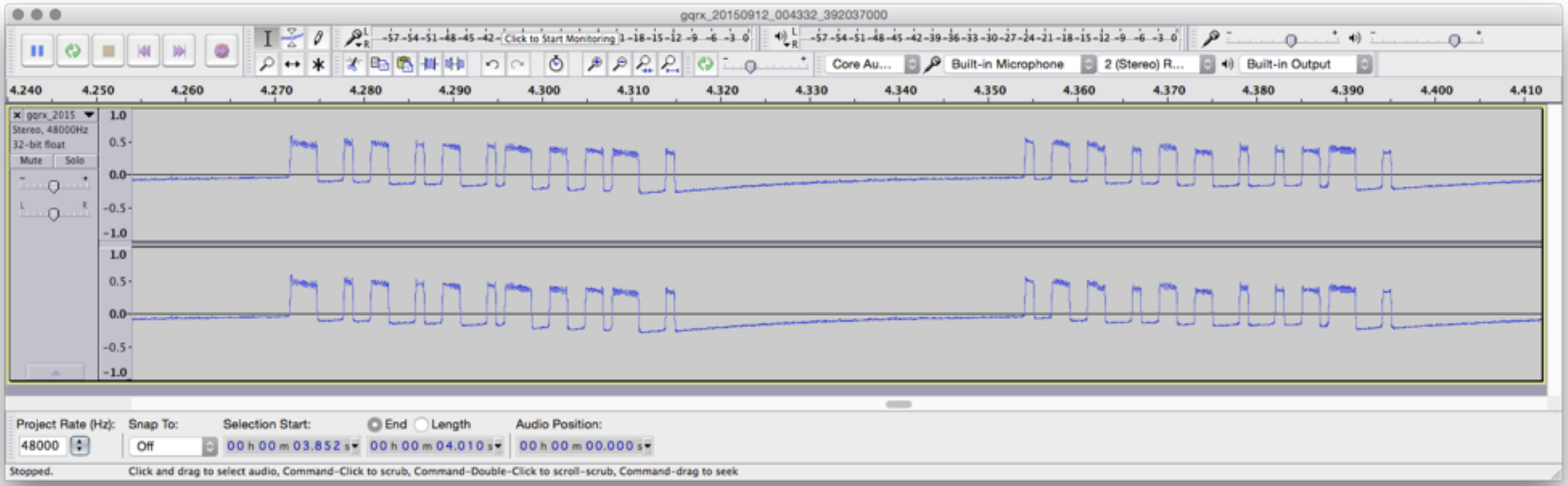
❐ 0 gyaresu 0 shodan 1 ..noPAGERcc1101 2 ~/programming 17:26 18 Nov zaphod

Audacity


biglist
Click to Start Monitoring
Core Au...
Built-in Microphone
2 (Stereo)...
Built-in Output
gqrx_2015
Stereo, 48000Hz
32-bit float
Mute
Solo
Project Rate (Hz):
48000
Snap To:
Off
Selection Start:
00 h 00 m 00.071 s
End
Length
00 h 00 m 00.000 s
Audio Position:
00 h 00 m 00.000 s
Stopped.
Click and drag to resize the track.

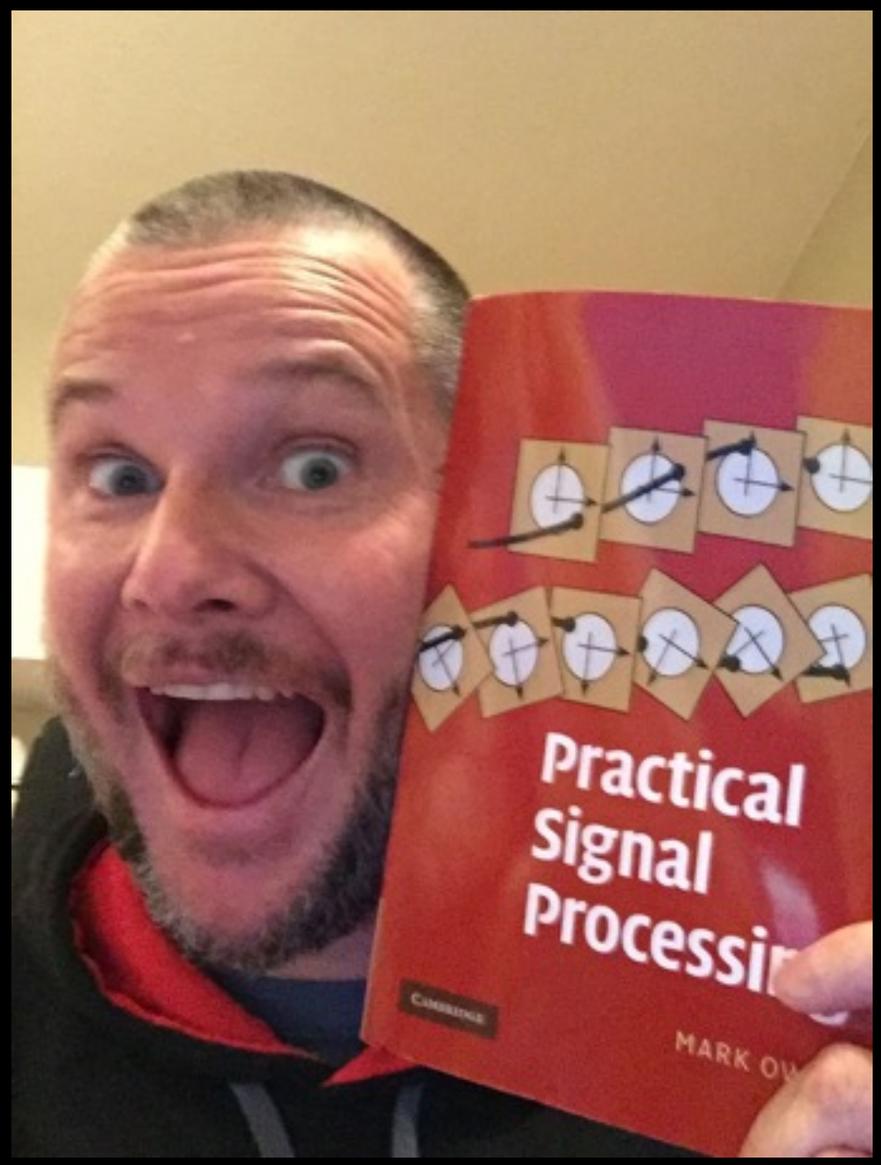

gqrx_20150911_202358_390000000
Click to Start Monitoring
Core Au...
Built-in Microphone
2 (Stereo) R...
1.310
1.320
1.330
1.340
1.350
1.360
1.370
1.380
1.390
1.400
1.410

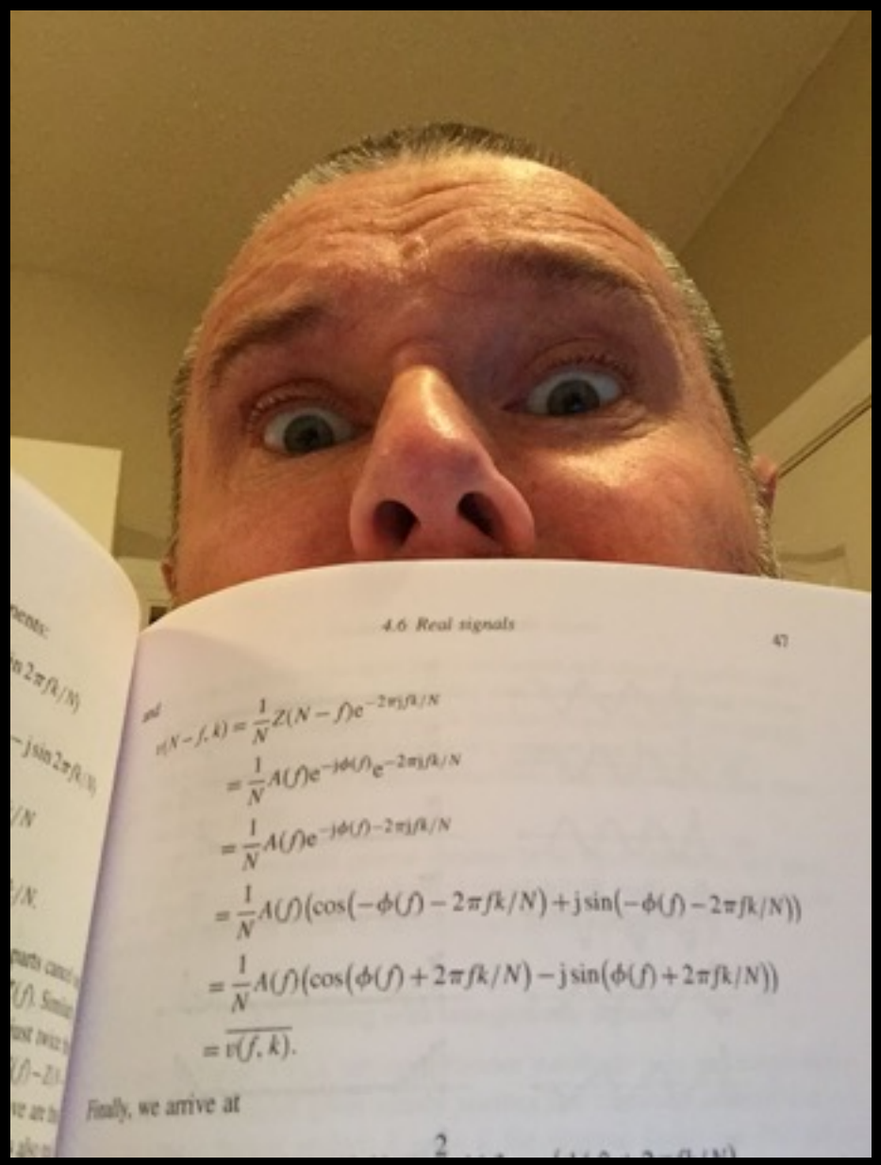

Click to Start Monitoring
1:07.80
1:08.10
1:08.30
1:08.50
1:08.70
1:08.90
1:09.10
1:09.30
1:09.50
1:09.70
1:09.90
test-trimm
Mono, 44100Hz
32-bit float
Mute
Solo
1.0
0.5
0.0
-0.5
-1.0


gqrx_20150912_004332_392037000
Click to Start Monitoring
Core Au...
Built-in Microphone
2 (Stereo) R...
Built-in Output
4.240 4.250 4.260 4.270 4.280 4.290 4.300 4.310 4.320 4.330 4.340 4.350 4.360 4.370 4.380 4.390 4.400 4.410
gqrx_2015
Stereo, 48000Hz
32-bit float
Mute
Solo
1.0
0.5
0.0
-0.5
-1.0
Project Rate (Hz):
48000
Snap To:
Off
Selection Start:
End
Length
Audio Position:
00 h 00 m 03.852 s
00 h 00 m 04.010 s
00 h 00 m 00.000 s
Stopped.
Click and drag to select audio, Command-Click to scrub, Command-Double-Click to scroll-scrub, Command-drag to seek

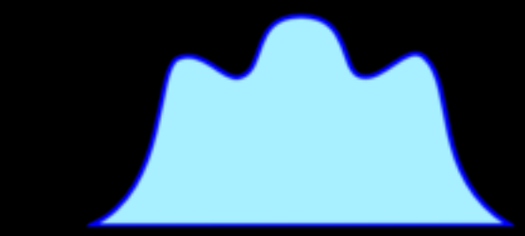
What's a signal?

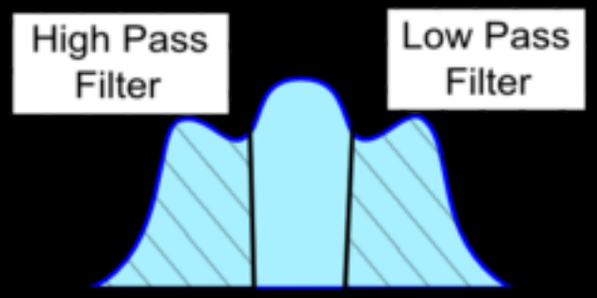
Practical Signal Processing - Mark Owen

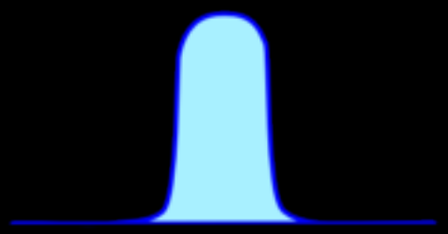




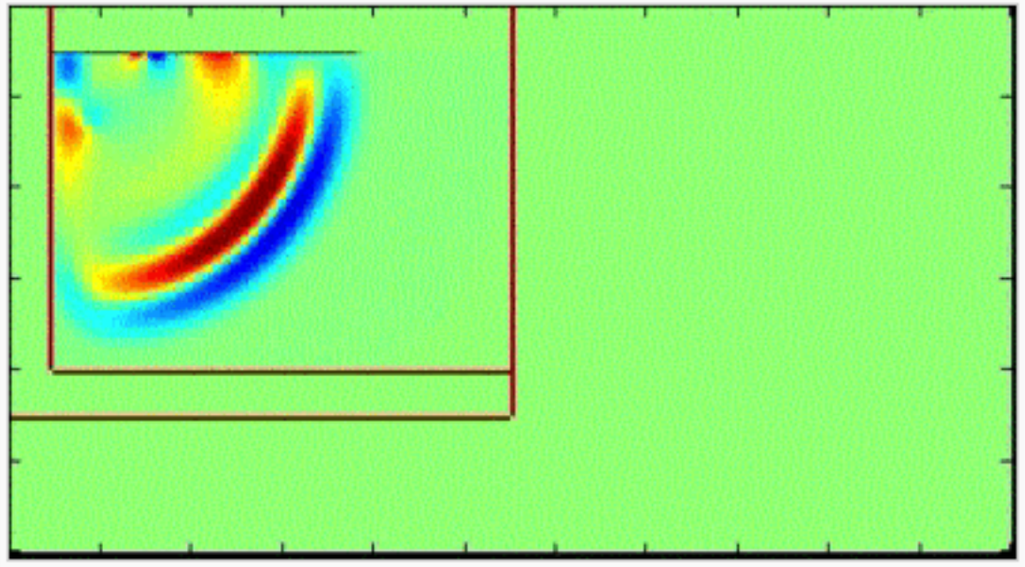
Passband







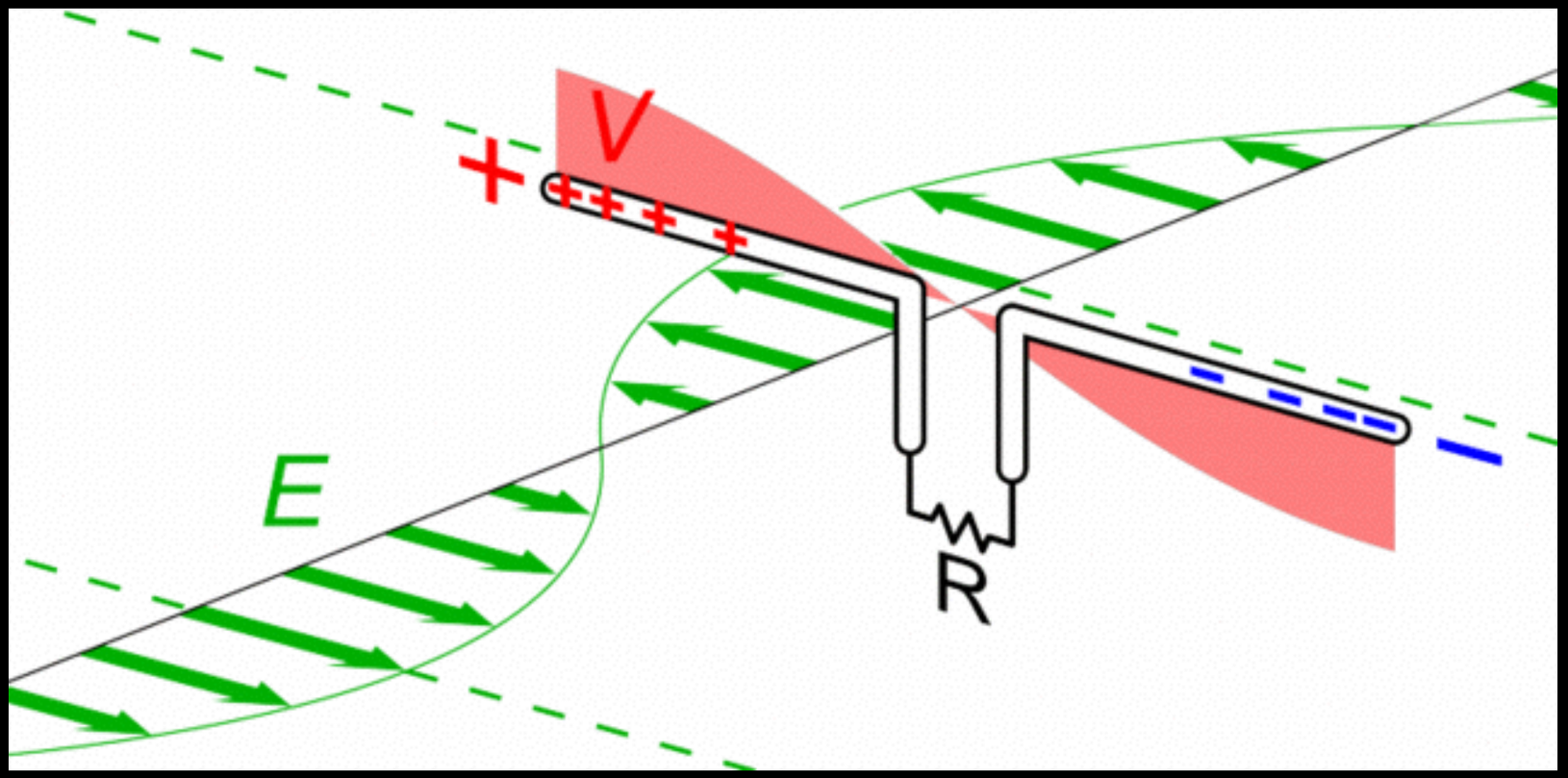
Wave Propogation



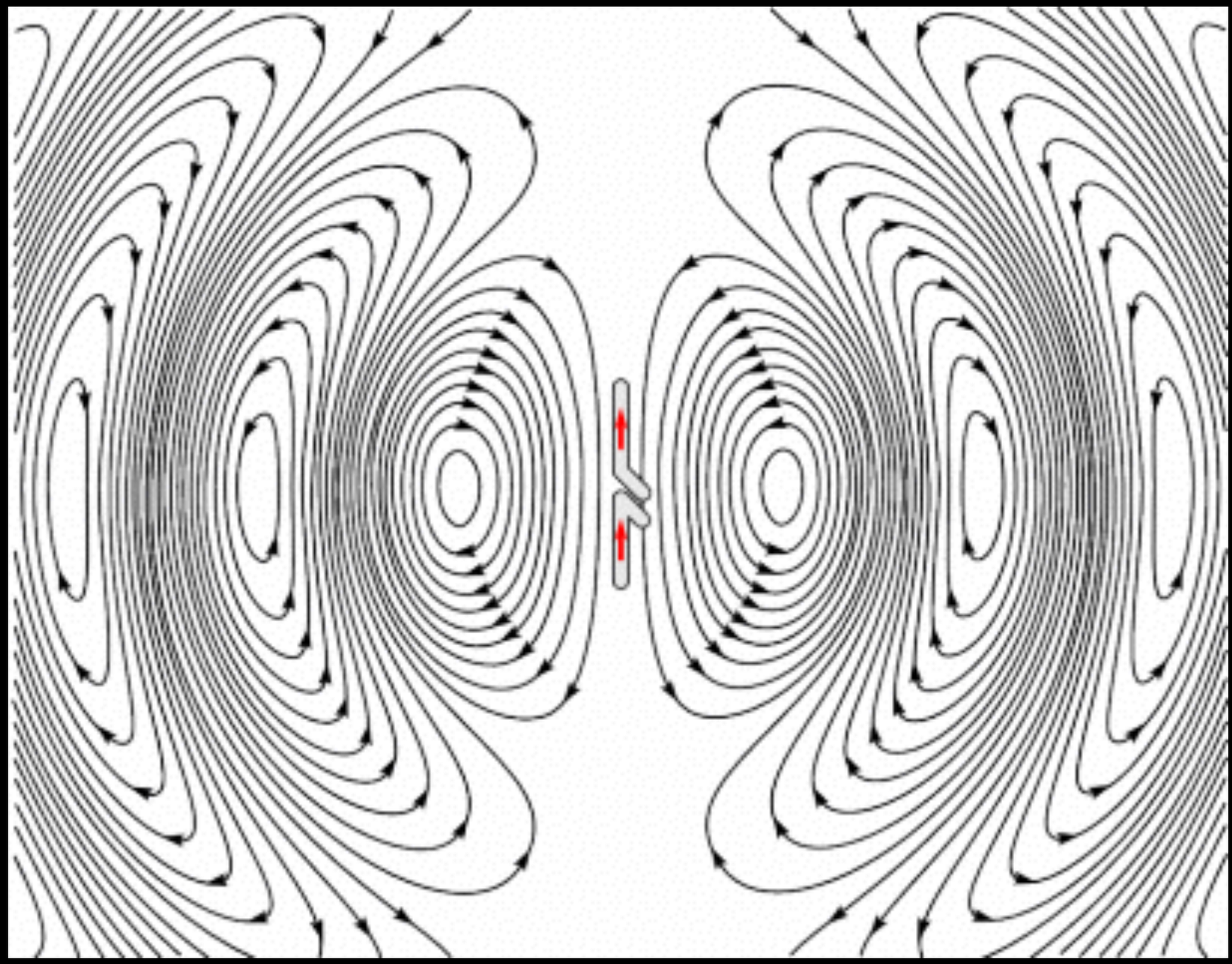
Seismic wave propagation in 2D modelled using FDTD method in the presence of a landmine

“Multipath effect”

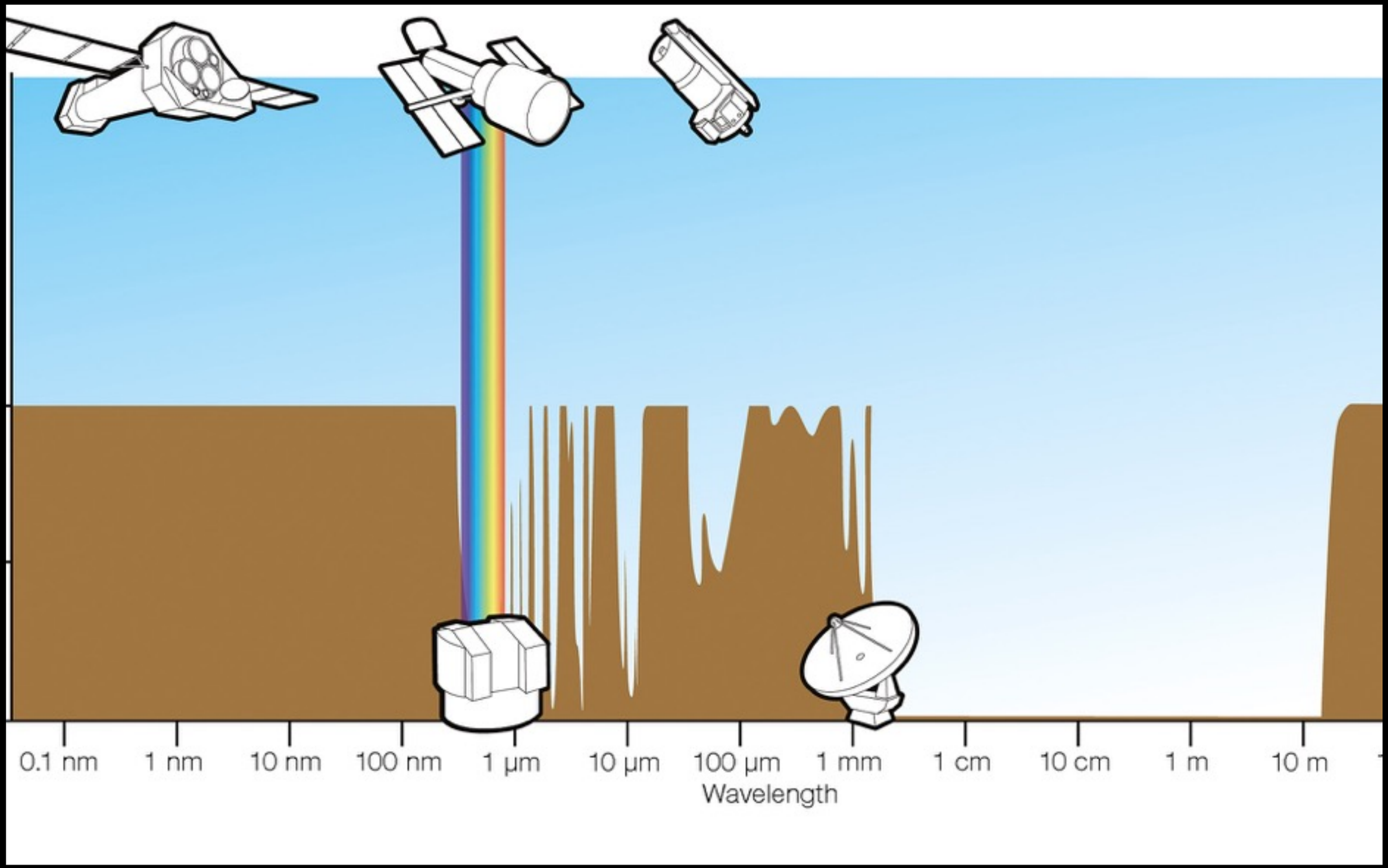
Antennas are weird



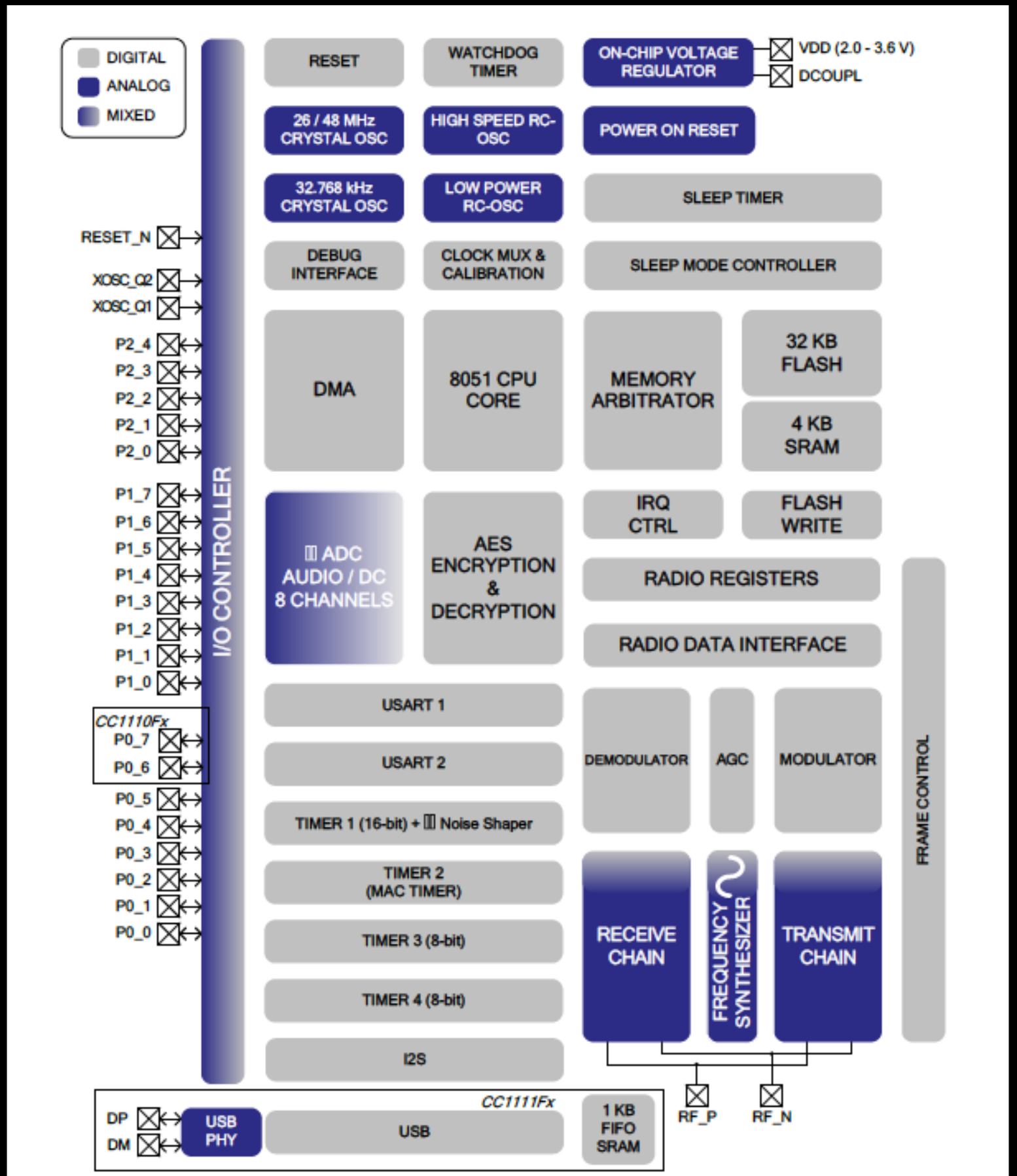
Woah



high frequency == short wavelength
low frequency == long wavelength
300MHz == 1m / 30MHz == 10m



CC11xx


DIGITAL
ANALOG
MIXED
RESET
WATCHDOG TIMER
ON-CHIP VOLTAGE REGULATOR
VDD (2.0 - 3.6 V)
DCOUPL
26 / 48 MHz CRYSTAL OSC
HIGH SPEED RC-OSC
POWER ON RESET
32.768 kHz CRYSTAL OSC
LOW POWER RC-OSC
SLEEP TIMER
DEBUG INTERFACE
CLOCK MUX & CALIBRATION
SLEEP MODE CONTROLLER
RESET_N
XOSC_Q2
XOSC_Q1
P2_4
P2_3
P2_2
P2_1
P2_0
DMA
8051 CPU CORE
MEMORY ARBITRATOR
32 KB FLASH
4 KB SRAM
P1_7
P1_6
P1_5
P1_4
P1_3
P1_2
P1_1
P1_0
I/O CONTROLLER
ADC AUDIO / DC 8 CHANNELS
AES ENCRYPTION & DECRYPTION
IRQ CTRL
FLASH WRITE
RADIO REGISTERS
RADIO DATA INTERFACE
CC1110Fx
P0_7
P0_6
P0_5
P0_4
P0_3
P0_2
P0_1
P0_0
USART 1
USART 2
TIMER 1 (16-bit) + Noise Shaper
TIMER 2 (MAC TIMER)
TIMER 3 (8-bit)
TIMER 4 (8-bit)
I2S
DEMODULATOR
AGC
MODULATOR
FRAME CONTROL
RECEIVE CHAIN
FREQUENCY SYNTHESIZER
TRANSMIT CHAIN
RF_P
RF_N
CC1111Fx
DP
DM
USB PHY
USB
1 KB FIFO SRAM

	Write		Read		
	Single Byte	Burst	Single Byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x00	IOCFG2				R/W configuration registers, burst access possible
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	FSTEST				
0x2A	PTEST				
0x2B	AGCTEST				
0x2C	TEST2				
0x2D	TEST1				
0x2E	TEST0				
0x2F					
0x30	SRES		SRES	PARTNUM	Command Strobes, Status registers (read only) and multi byte registers
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	PATABLE	PATABLE	PATABLE	PATABLE	
0x3F	TX FIFO	TX FIFO	RX FIFO	RX FIFO	

RTFDS

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):

- 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.

15.1 Data Whitening

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

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

(Read The Fine Data Sheet)

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

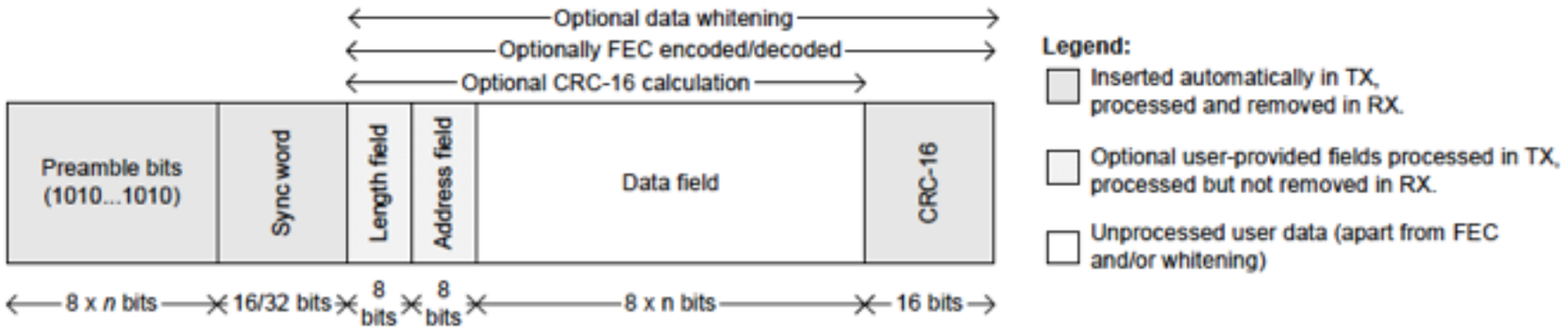


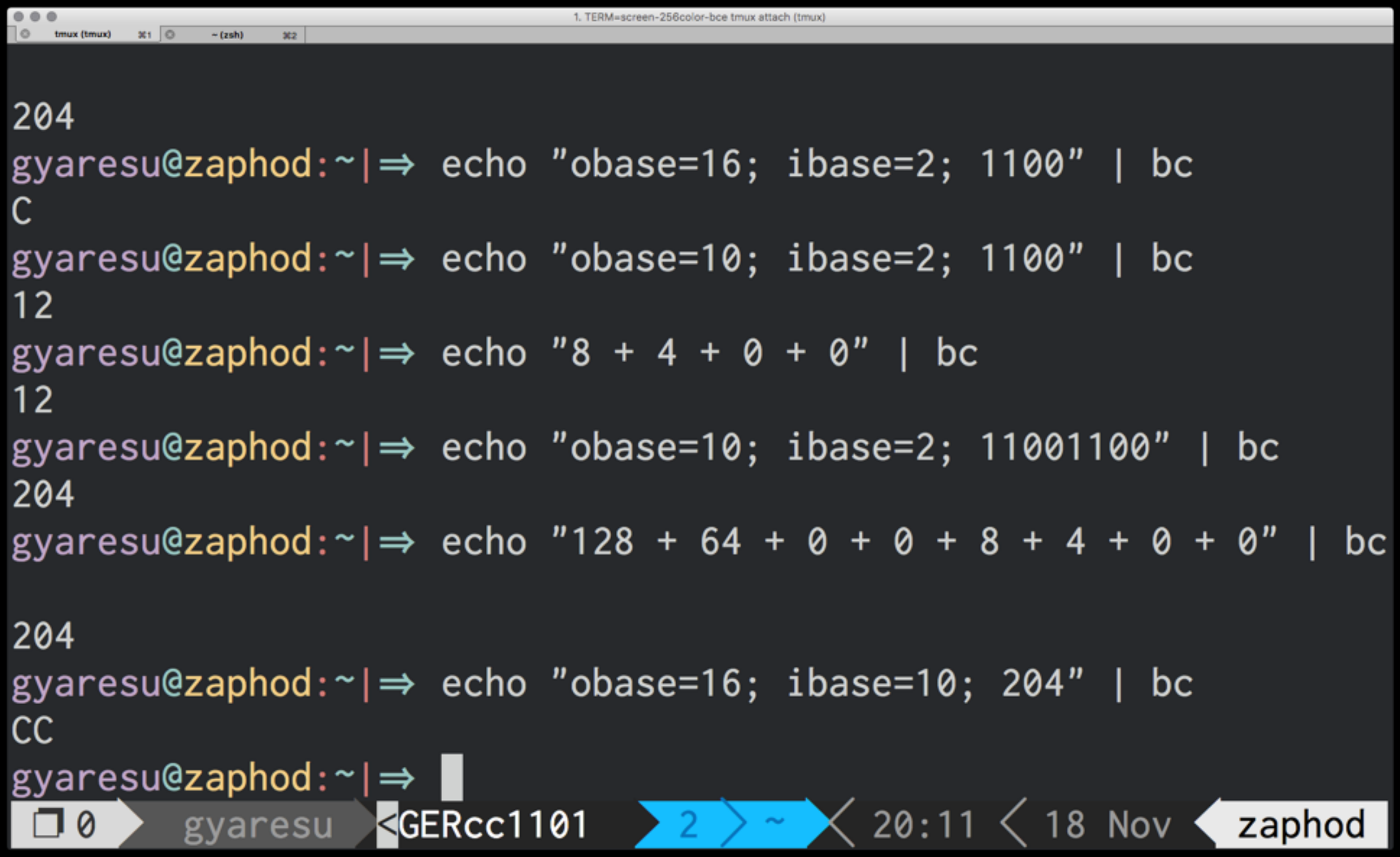
Figure 19: Packet Format

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.

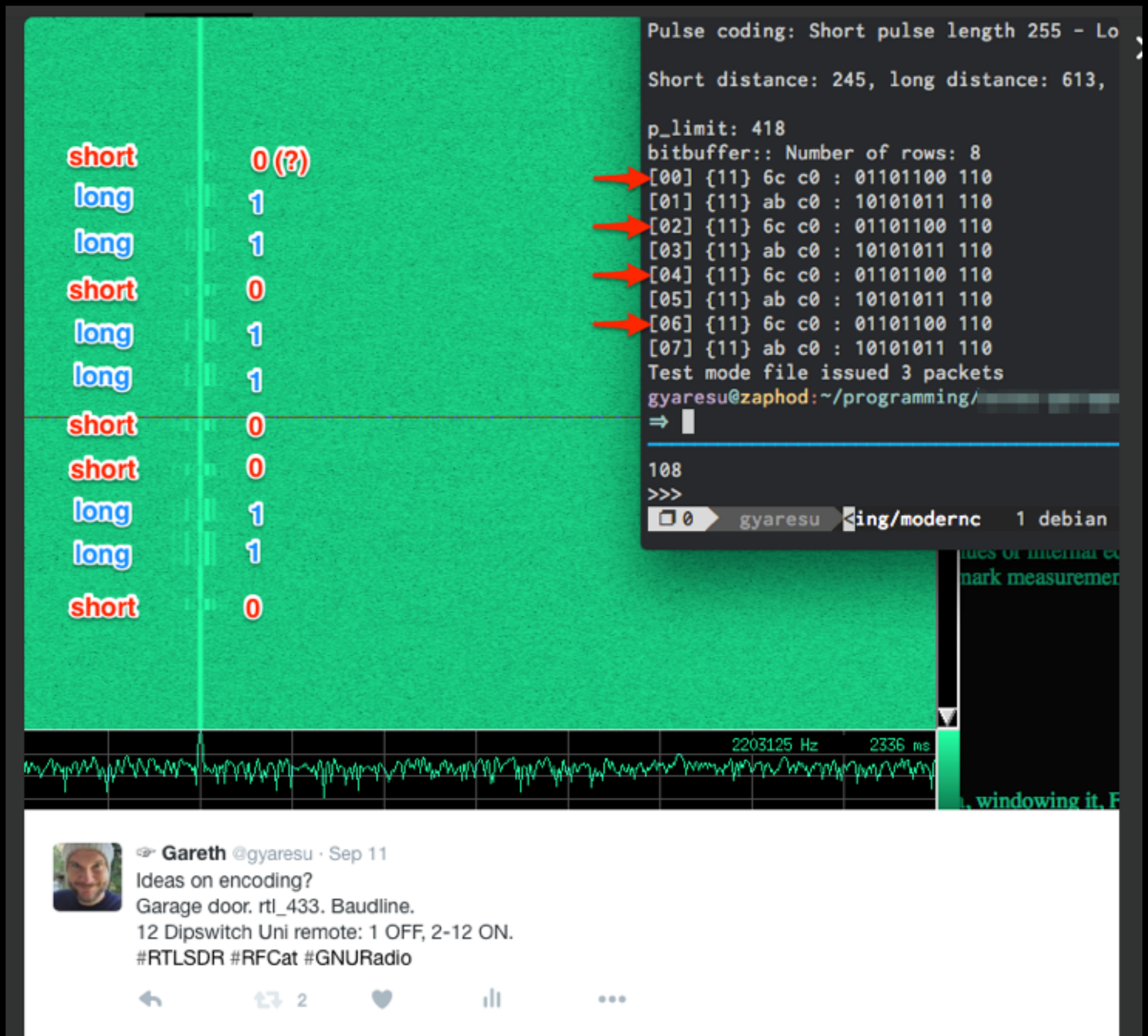
The synchronization word is a two-byte value set in the `SYNC1` and `SYNC0` registers. The sync word provides byte synchronization of the incoming packet. A one-byte sync word can be emulated by setting the `SYNC1` value to the preamble pattern. It is also possible to emulate a 32 bit sync word by setting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



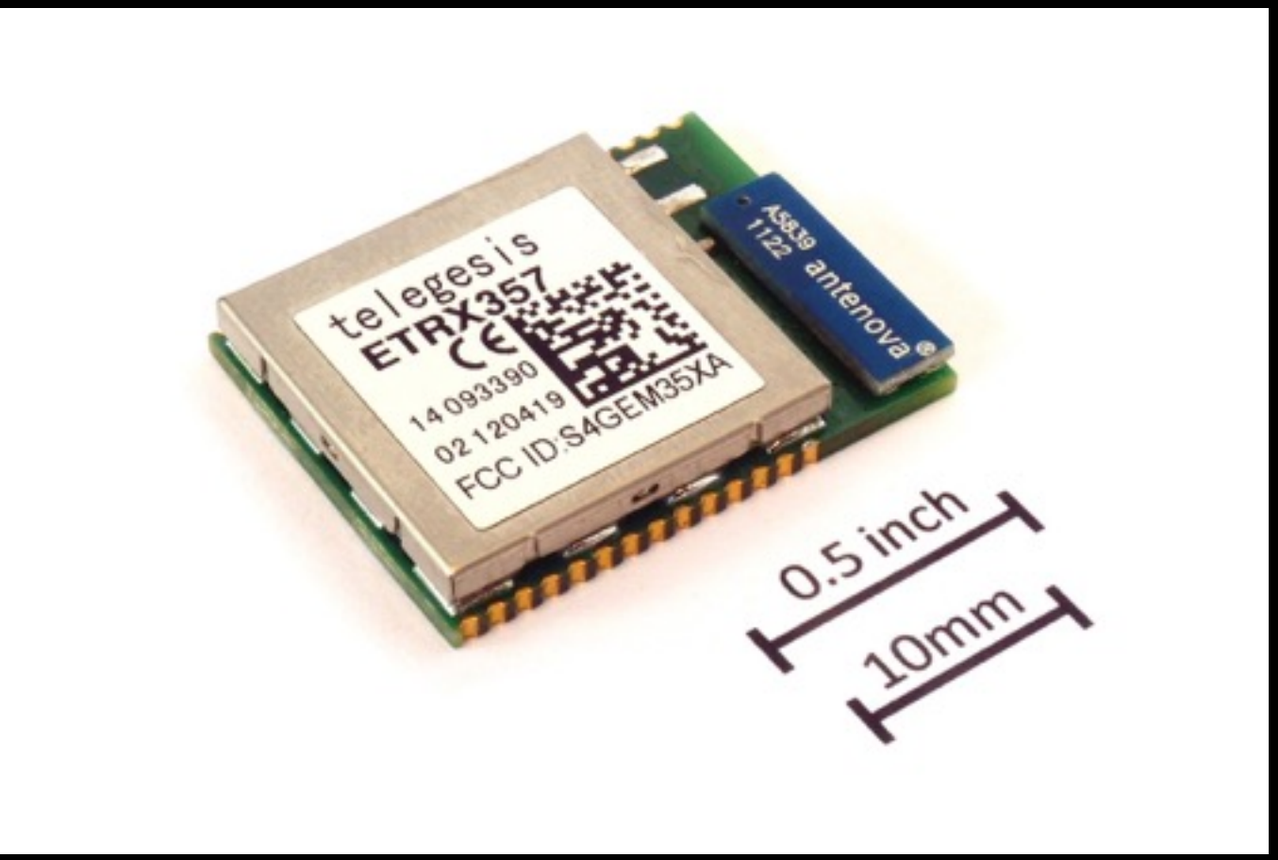
Packet identification



ZigBee <protocols> Z-Wave

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

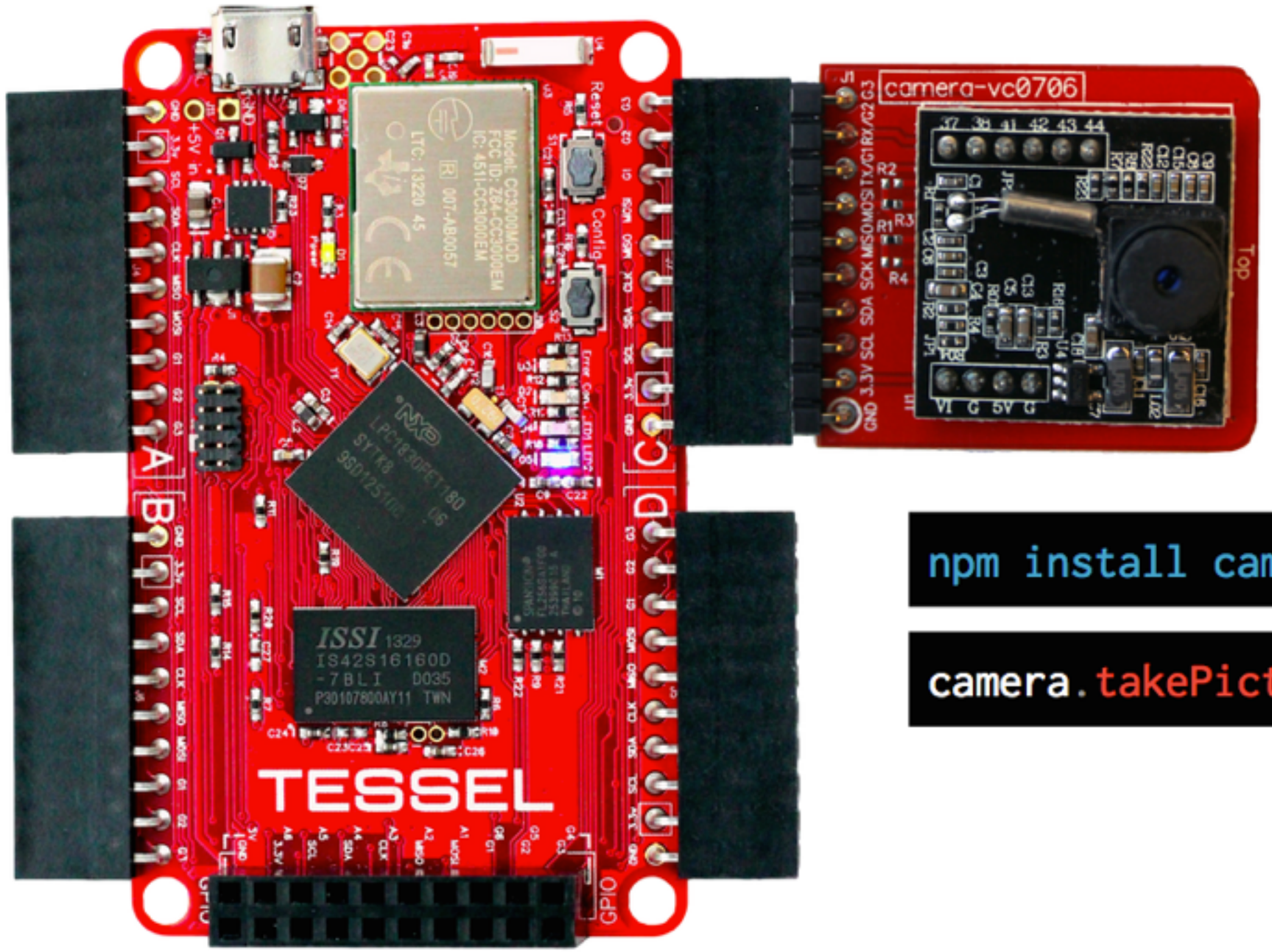
Residential.
~900MHz



Brand: XBee

Brand: Sigma Designs

Wait… how did JavaScript get in here?



Camera

```
npm install camera-vc0706
```

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

JavaScript is robots

J5

News API Examples Articles Platform Support

Fork me on GitHub

Johnny-Five

The JavaScript

Programming Framework

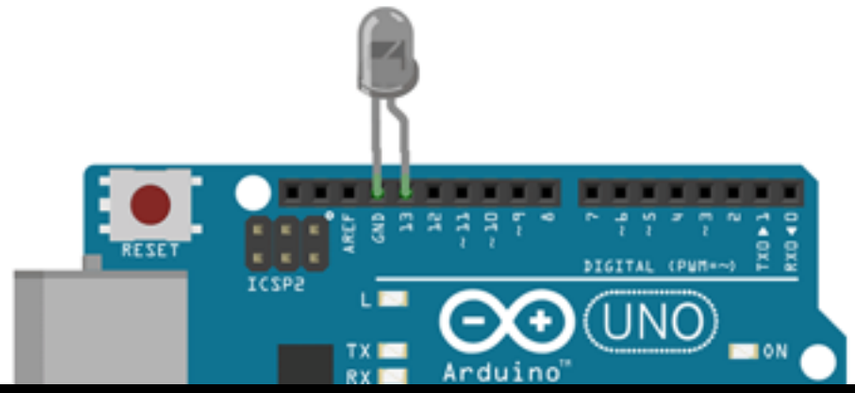
Johnny-Five is the original JavaScript Robotics programming framework. Released by Bocoup in 2012, Johnny-Five is maintained by a community of passionate software developers and hardware engineers. Over 75 developers have made contributions towards building a robust, extensible and composable ecosystem.

Star 4,838 Fork 645

Hello World!

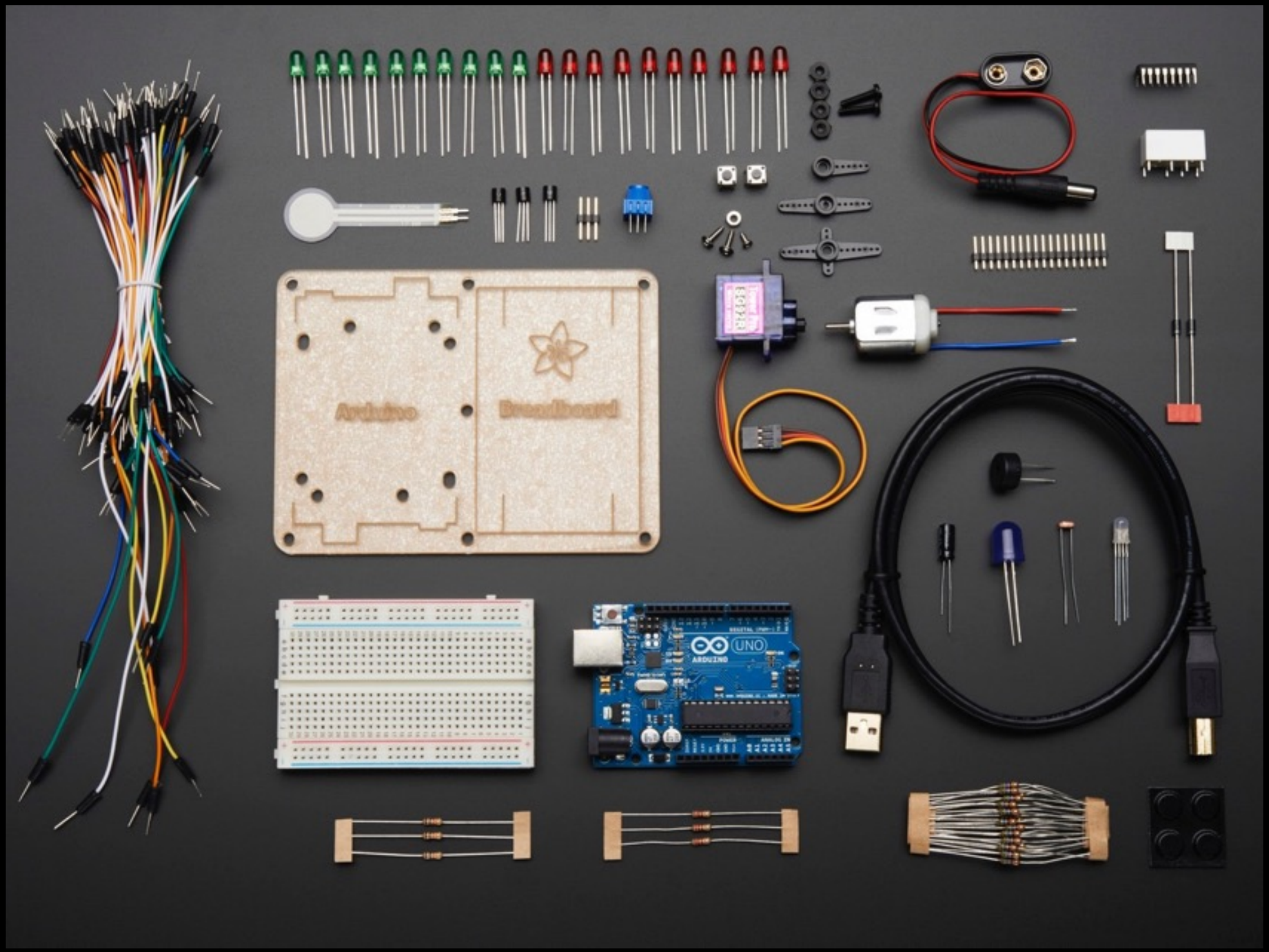
The ubiquitous "Hello World" program of the microcontroller and SoC world is "blink an LED!". The following code demonstrates how this is done using the Johnny-Five framework.

1. Install Node.js or io.js.
2. Setup your board.
3. `npm install johnny-five`

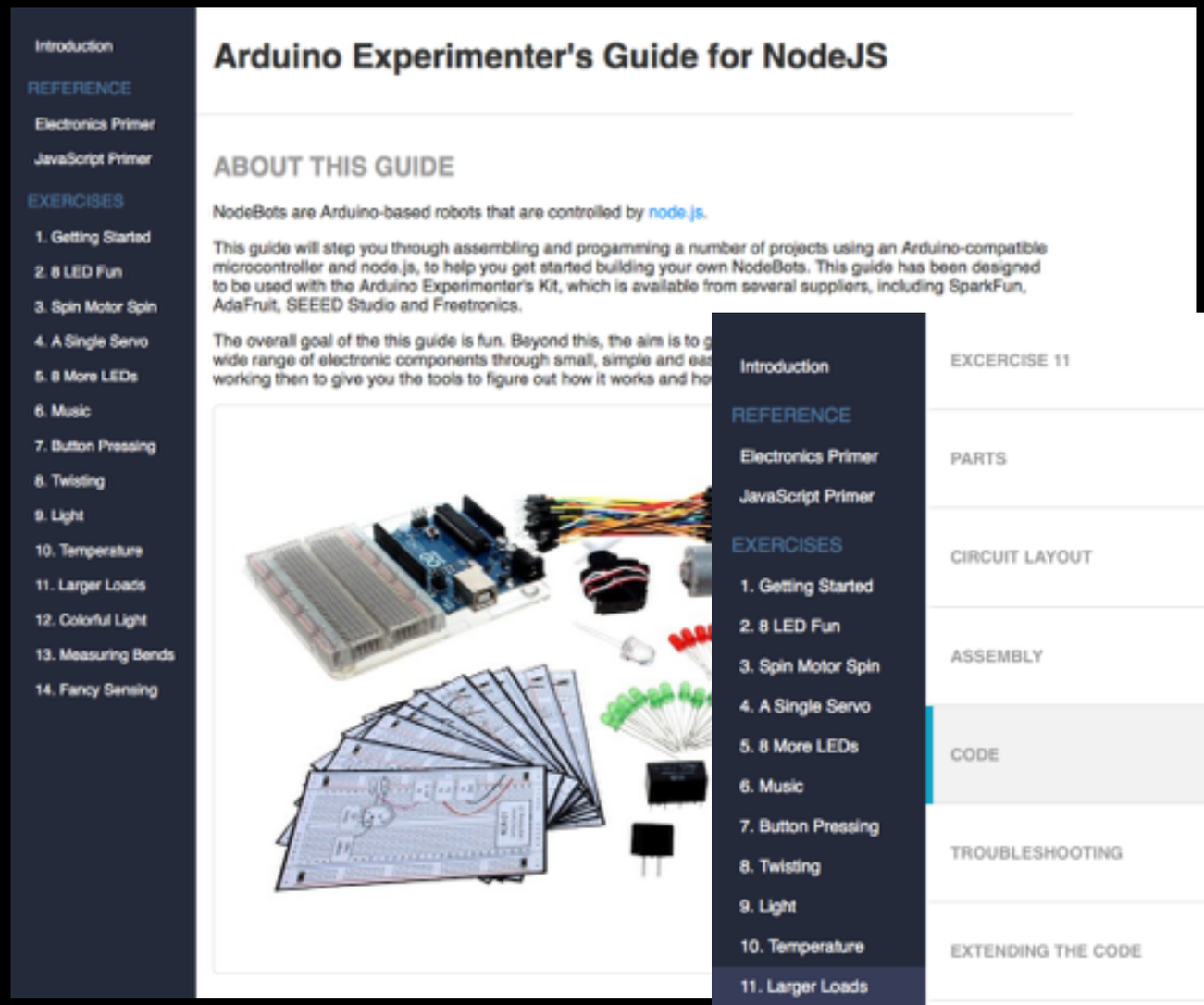


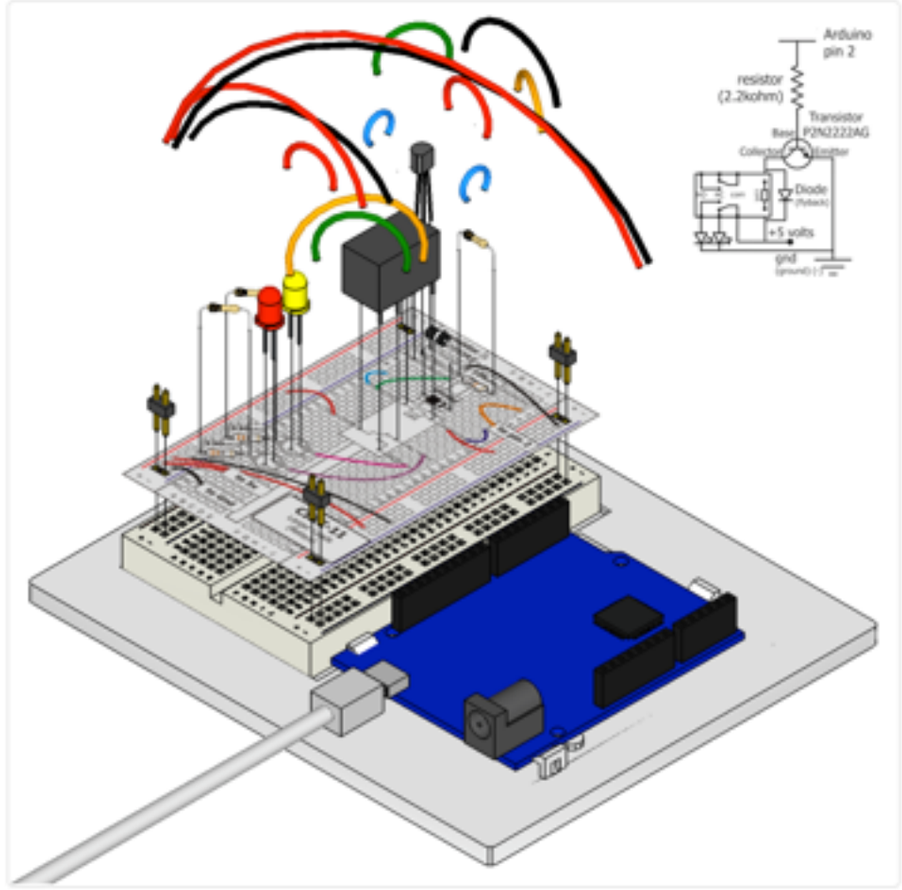
ARDX

Arduino Experimenter's Kit



node-ardx.org





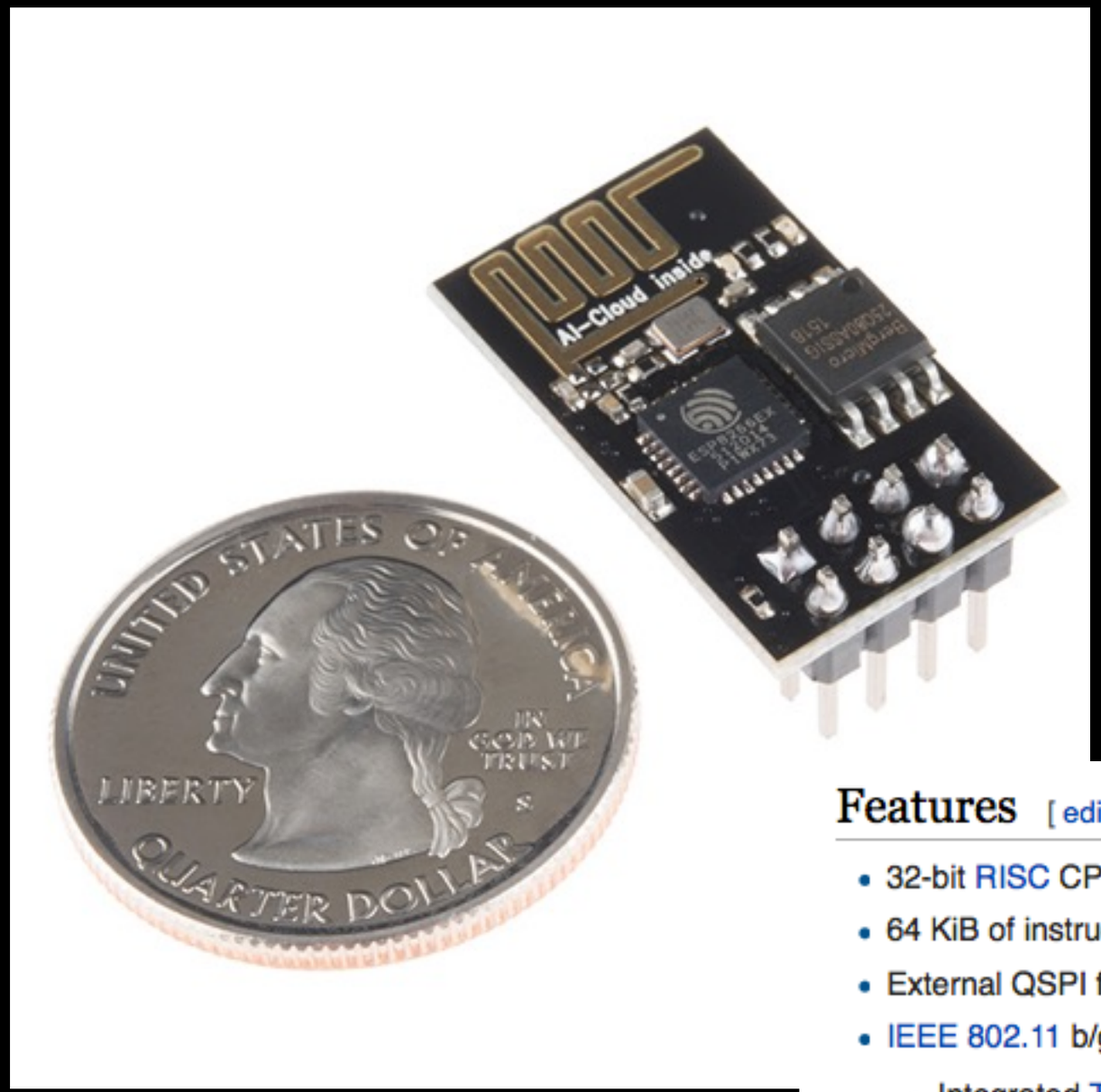
Assembly video: http://ardx.org/VIDE11

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();
```

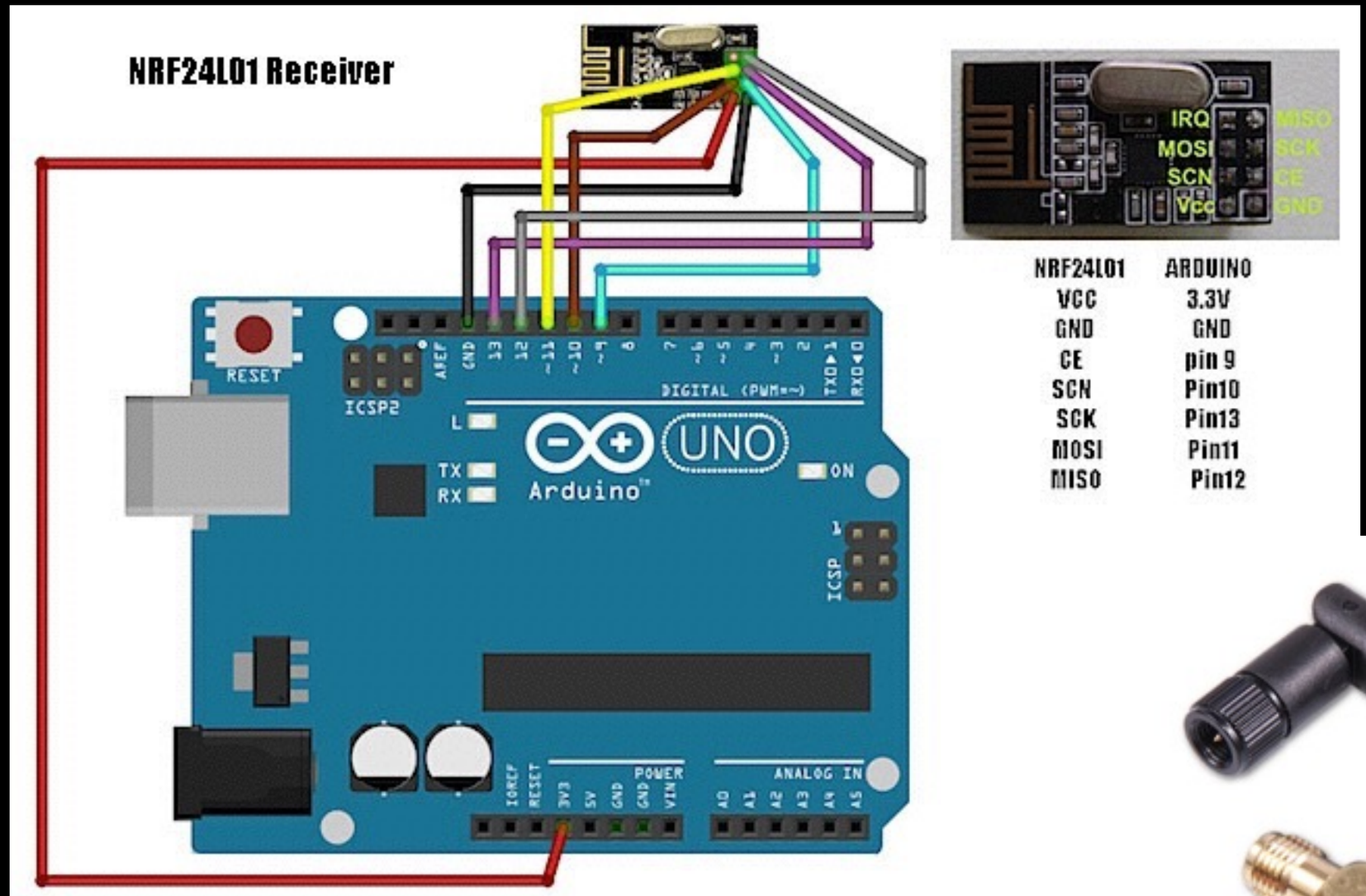
ESP8266 (2.4GHz WiFi™)



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™)



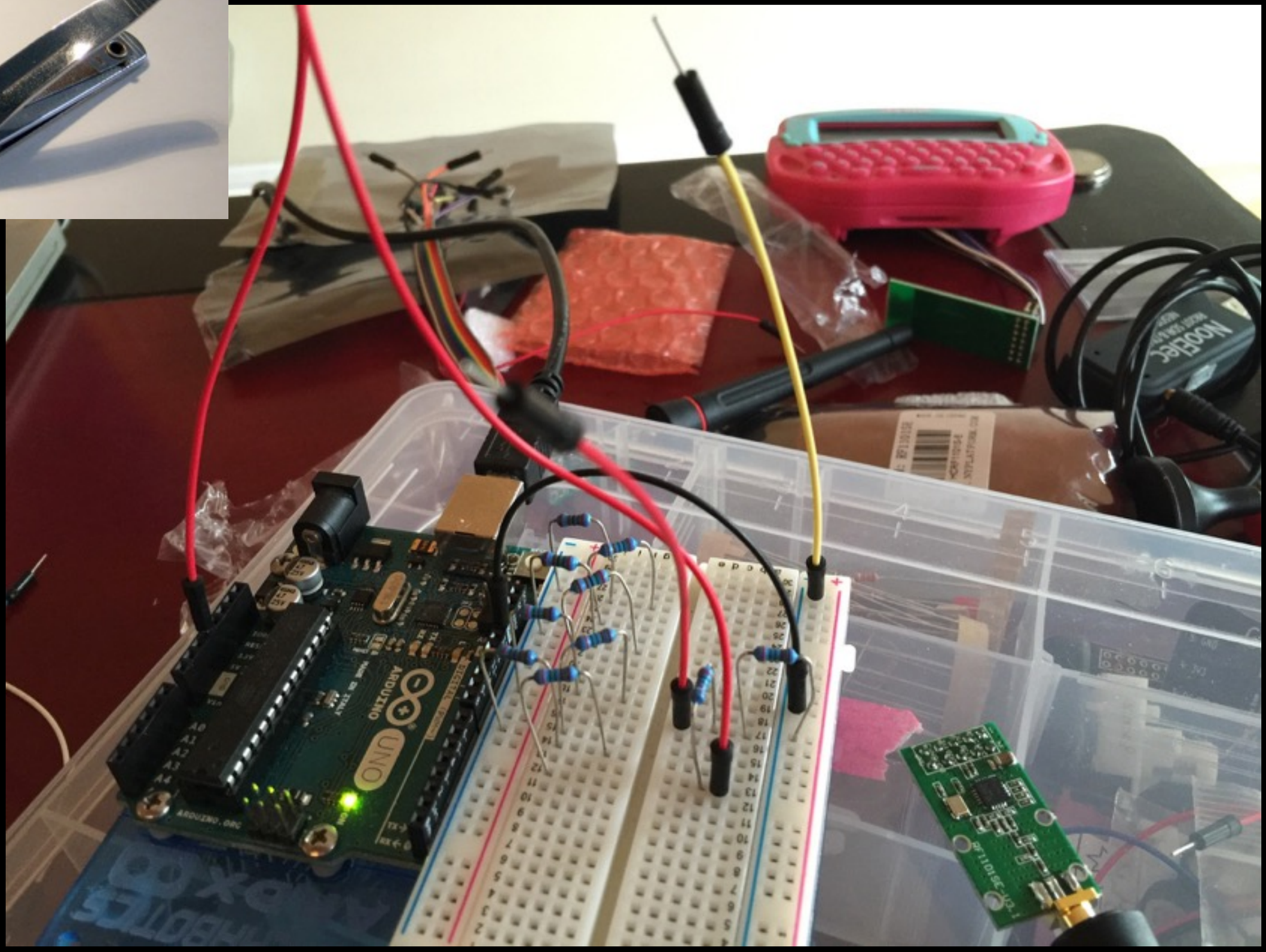
NRF24L01	ARDUINO
VCC	3.3V
GND	GND
CE	pin 9
SCN	Pin10
SCK	Pin13
MOSI	Pin11
MISO	Pin12

GFSK (Gaussian FSK)

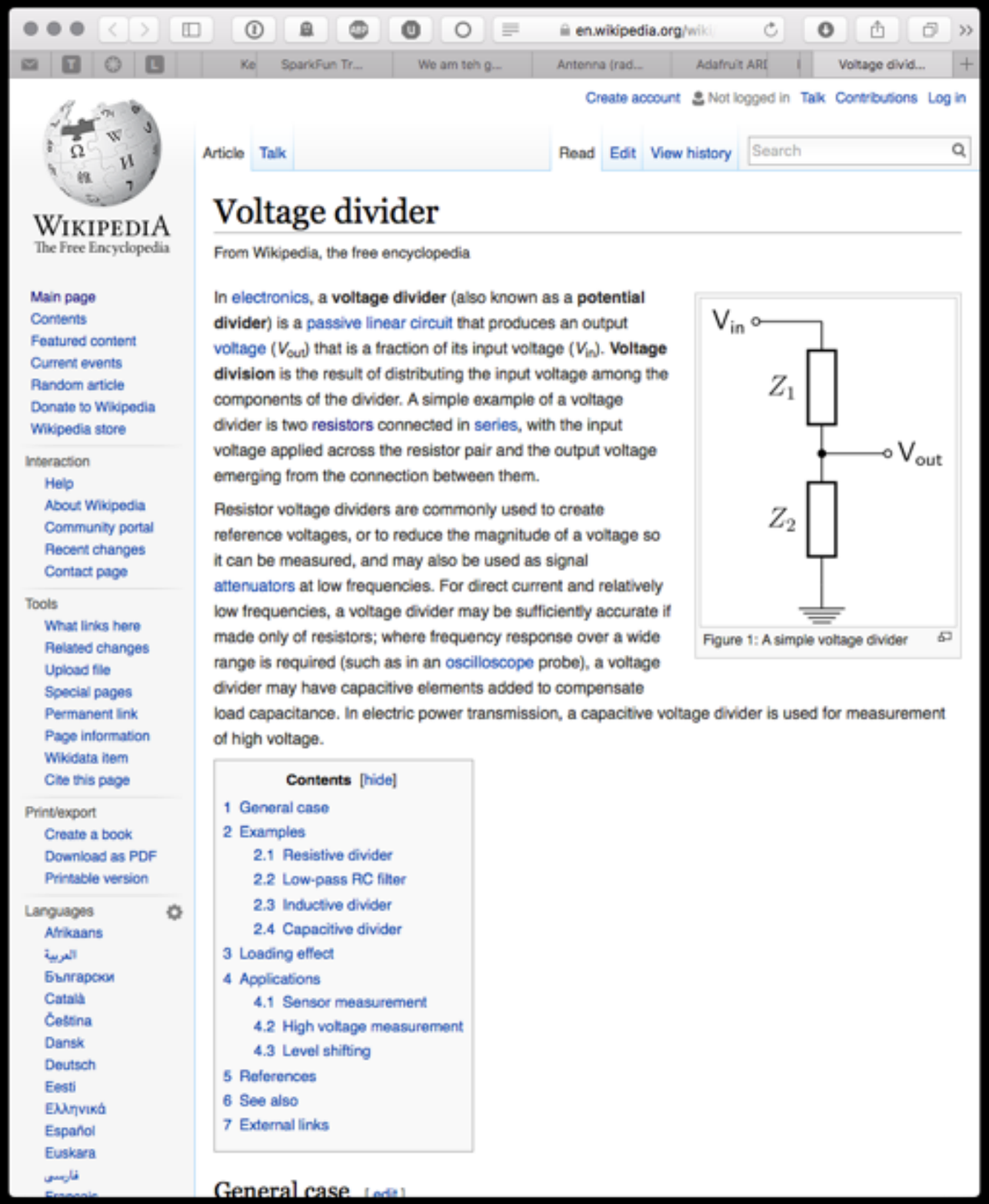
+1, -1

+1, +0.98, +0.95… -0.95, -0.98, =1

Lesson #1: Plug it in



Voltage Dividers (WAT?)



en.wikipedia.org/wiki/Voltage_divider

Create account · Not logged in · Talk · Contributions · Log in

Article · Talk · Read · Edit · View history · Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Afrikaans
العربية
Български
Català
Čeština
Dansk
Deutsch
Eesti
Ελληνικά
Español
Euskara
فارسی

Voltage divider

From Wikipedia, the free encyclopedia

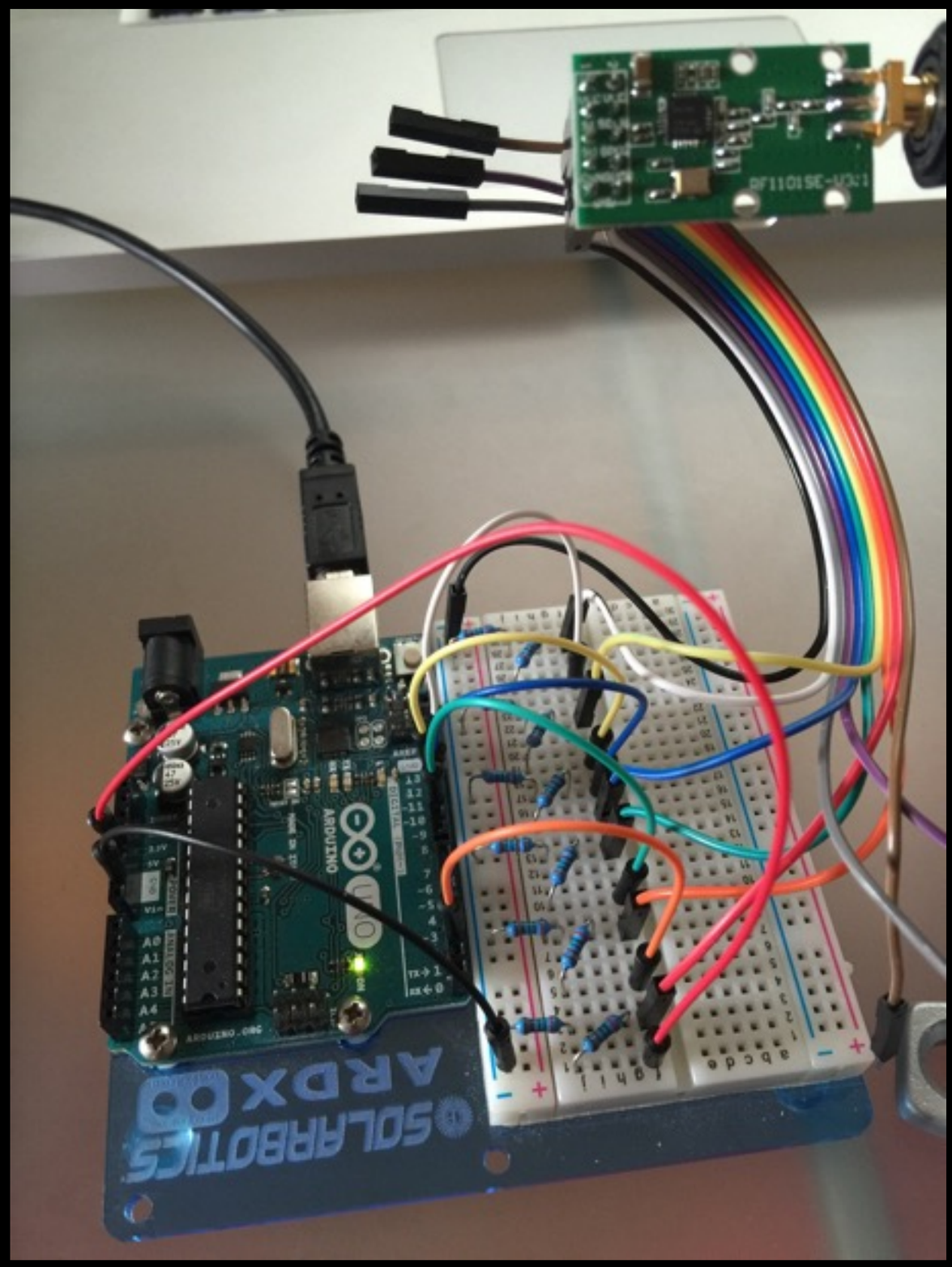
In electronics, a **voltage divider** (also known as a **potential divider**) is a passive linear circuit that produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). **Voltage division** is the result of distributing the input voltage among the components of the divider. 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.

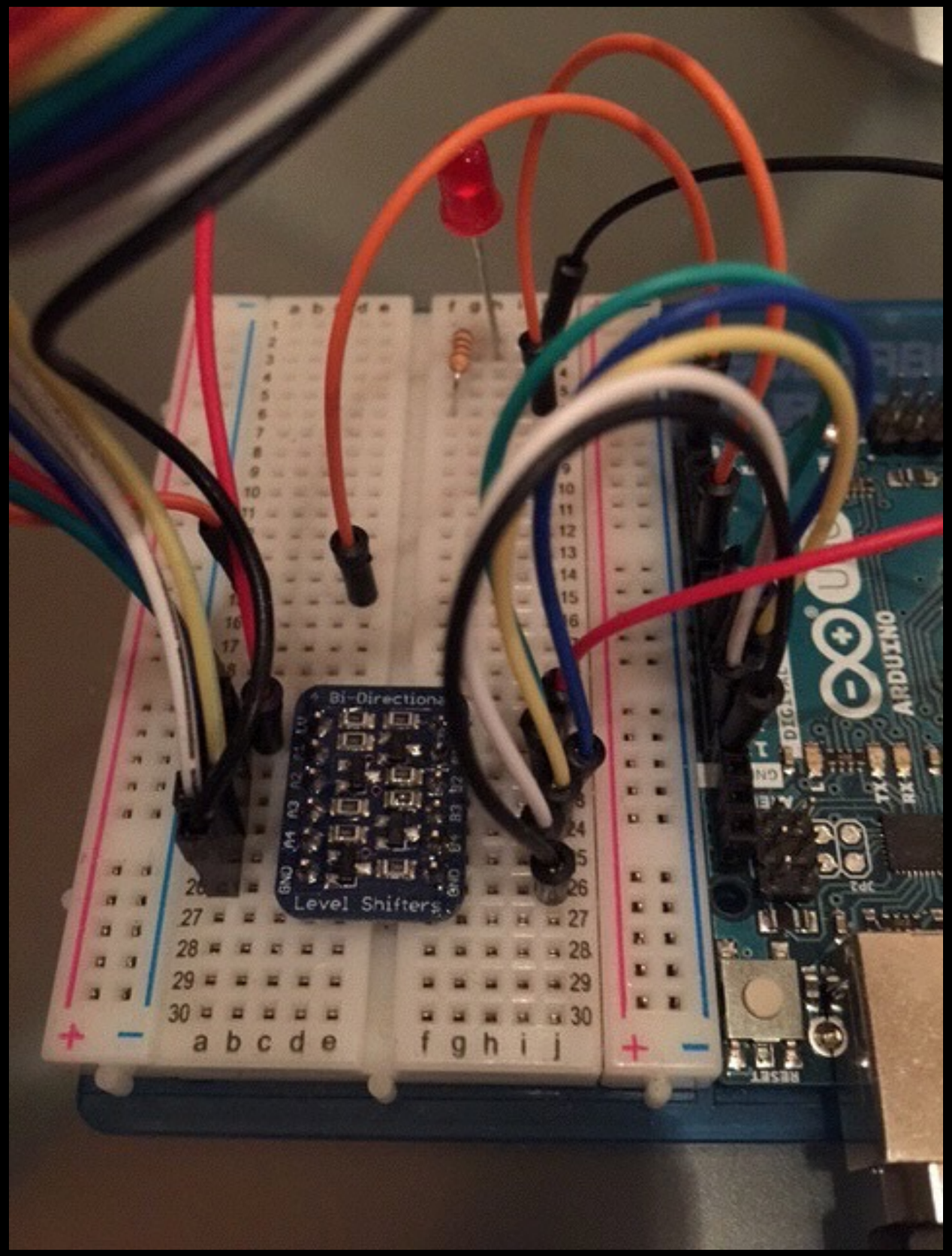
Figure 1: A simple voltage divider

Contents [hide]

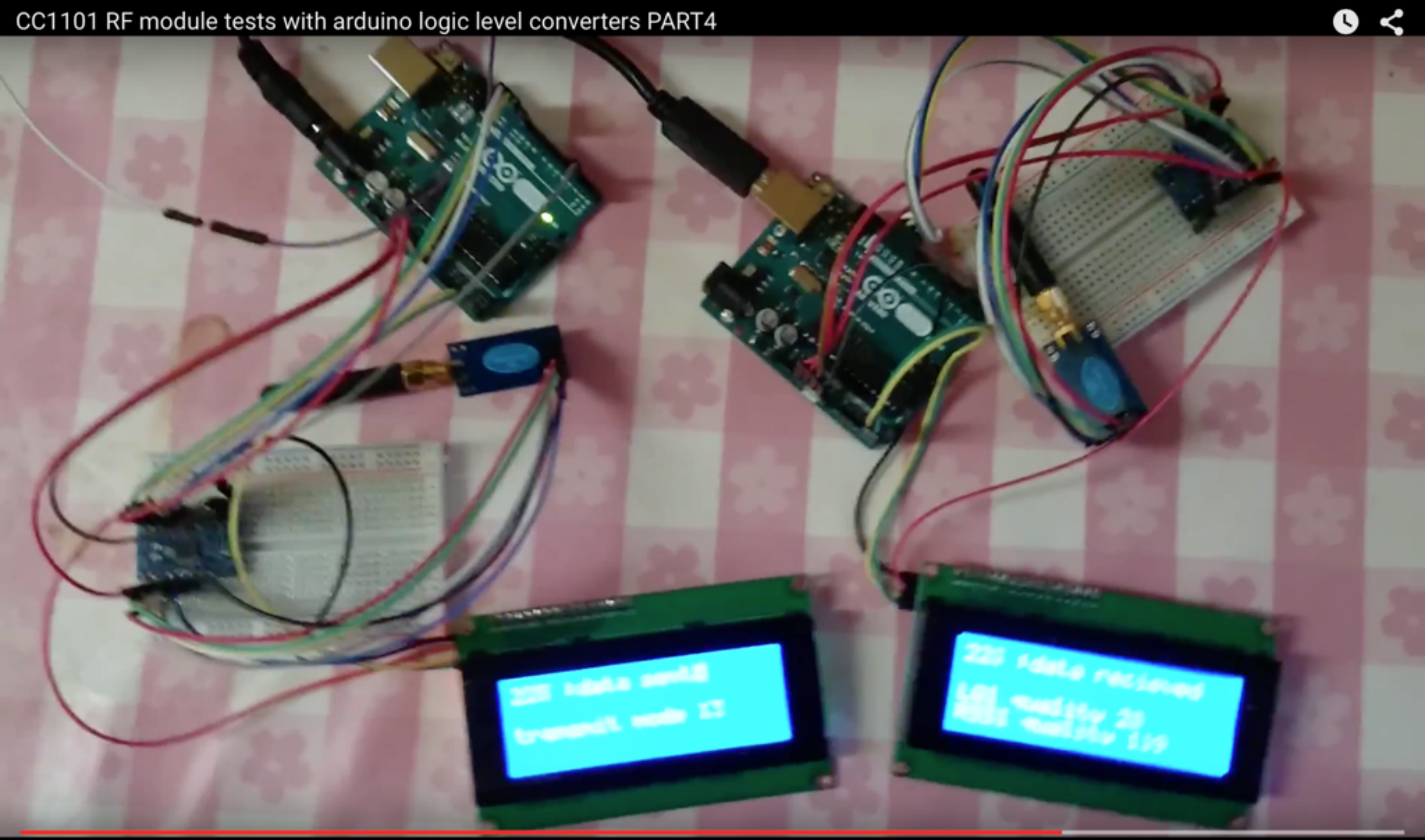
General case [edit]



Logic Level Converters



It’s alive!


CC1101 RF module tests with arduino logic level converters PART4
0:57 / 1:16

rf1101.cpp

1. TERM=screen-256color-bce tmux attach (tmux)

tmux (tmux) ⌘1 ~ (zsh) ⌘2

```
 * reset
 *
 * Reset RF1101
 */
void RF1101::reset(void)
{
  RF1101_Deselect();                    // Deselect RF1101
  delayMicroseconds(5);
  RF1101_Select();                      // Select RF1101
  delayMicroseconds(10);
  RF1101_Deselect();                    // Deselect RF1101
  delayMicroseconds(41);
  RF1101_Select();                      // Select RF1101

  wait_Miso();                          // Wait until MISO goes low
  spi.send(RF1101_SRES);                // Send reset command strobe
  wait_Miso();                          // Wait until MISO goes low

  RF1101_Deselect();                    // Deselect RF1101

  setDefaultRegs();                     // Reconfigure RF1101
  setRegsFromEeprom();                  // Take user settings from
EEPROM
}

/**
 * setDefaultRegs
 *
 * Configure RF1101 registers
 */
void RF1101::setDefaultRegs(void)
{
  writeReg(RF1101_IOCFG2,   RF1101_DEFVAL_IOCFG2);
  writeReg(RF1101_IOCFG1,   RF1101_DEFVAL_IOCFG1);
  writeReg(RF1101_IOCFG0,   RF1101_DEFVAL_IOCFG0);
```

NORMAL rf1101.cpp unix < utf-8 < cpp 34% ln 192:23

```
 * 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
 * Settings optimized for low current consumption
 */
//#define RF1101_DEFVAL_IOCFG2     0x29       // GDO2 Output Pin
Configuration
#define RF1101_DEFVAL_IOCFG2     0x2E       // GDO2 Output Pin
Configuration
#define RF1101_DEFVAL_IOCFG1     0x2E       // GDO1 Output Pin
Configuration
#define RF1101_DEFVAL_IOCFG0     0x06       // GDO0 Output Pin
Configuration
#define RF1101_DEFVAL_FIFOTHR    0x07       // RX FIFO and TX FIFO
Thresholds
```

NORMAL rf1101.h unix < utf-8 < cpp 41% ln 221:11

```

/**
 * RF1101 configuration registers
 */
#define RF1101_IOCFG2            0x00       // GDO2 Output Pin
Configuration
#define RF1101_IOCFG1            0x01       // GDO1 Output Pin
Configuration
#define RF1101_IOCFG0            0x02       // GDO0 Output Pin
Configuration
#define RF1101_FIFOTHR           0x03       // RX FIFO and TX FIFO
Thresholds
#define RF1101_SYNC1             0x04       // Sync Word, High Byte
#define RF1101_SYNC0             0x05       // Sync Word, Low Byte
#define RF1101_PKTLEN            0x06       // Packet Length
```

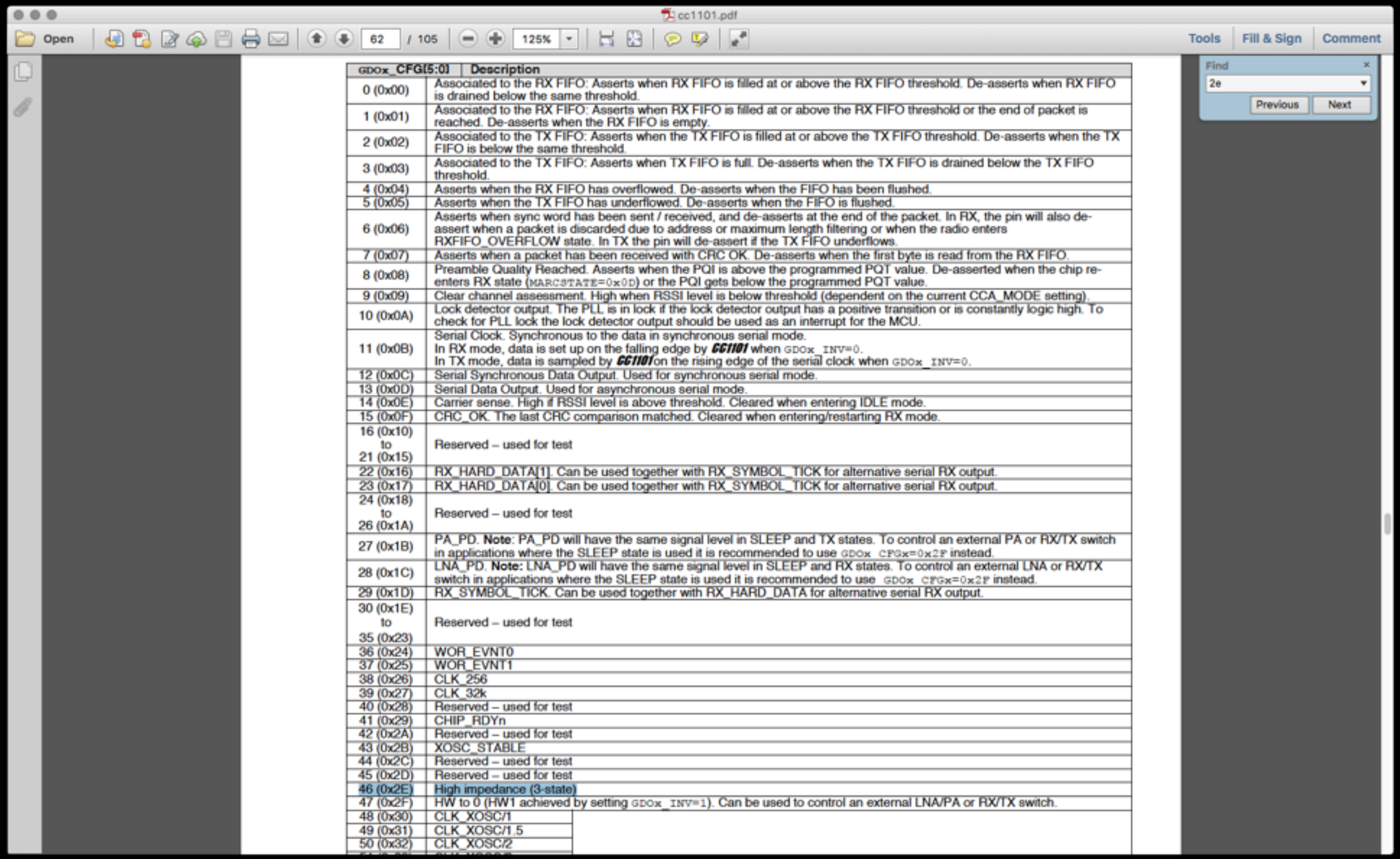
NORMAL rf1101.h unix < utf-8 < cpp 22% ln 121:16

0 gyaresu 0 shodan 1 ..noPAGERcc1101 2 vi < 09:56 < 19 Nov zaphod

Wat?

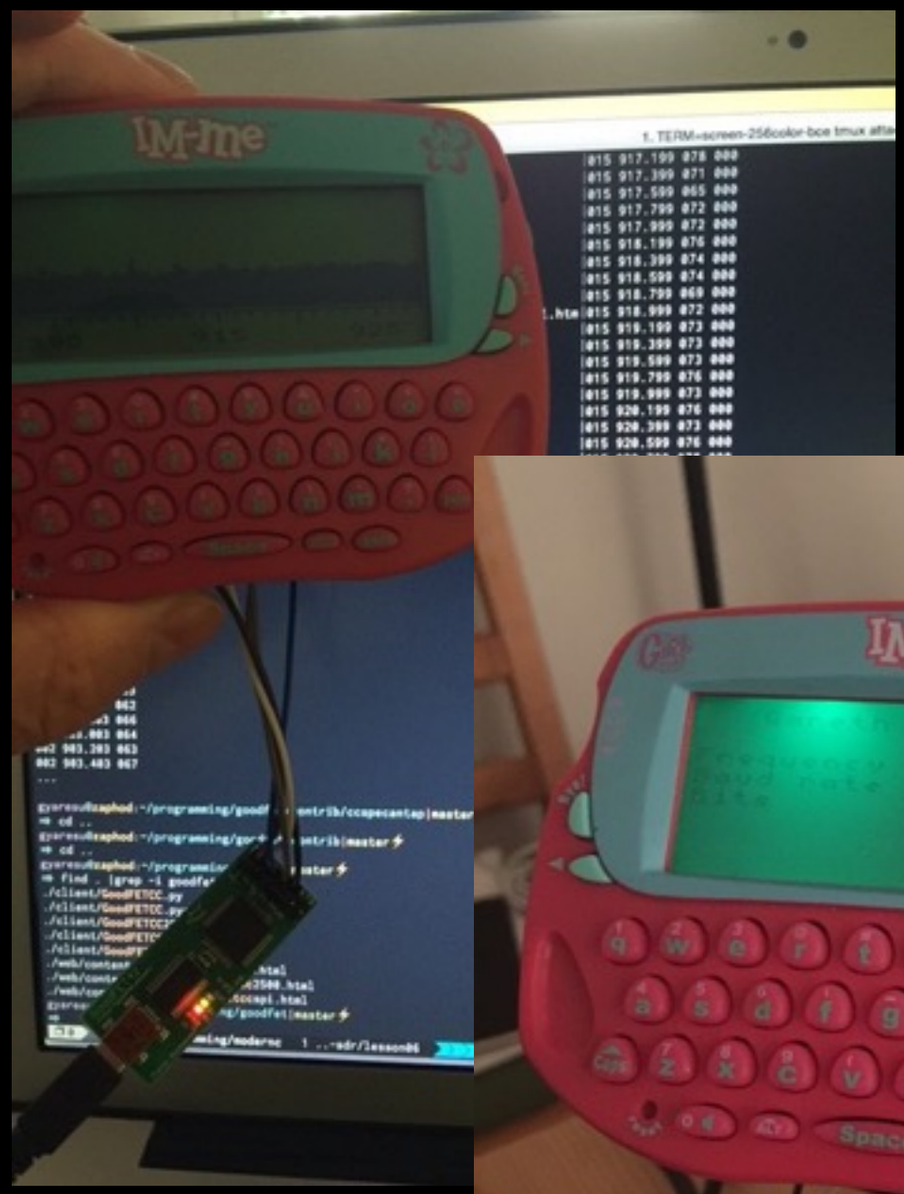
ABC

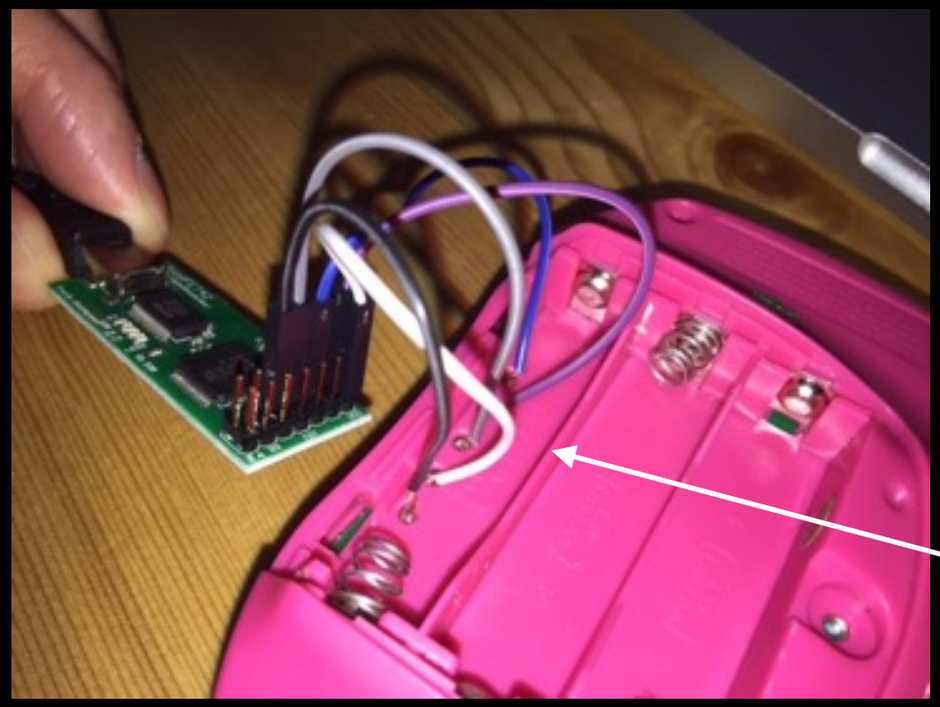
Always Be Checking the damn data sheet



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 (MARCSTATE=0x0D) 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 ***CC1101*** when GDOx_INV=0. In TX mode, data is sampled by ***CC1101*** on the rising edge of the serial clock when GDOx_INV=0.
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 GDOx_CFGx=0x2F 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 GDOx_CFGx=0x2F 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 GDOx_INV=1). 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

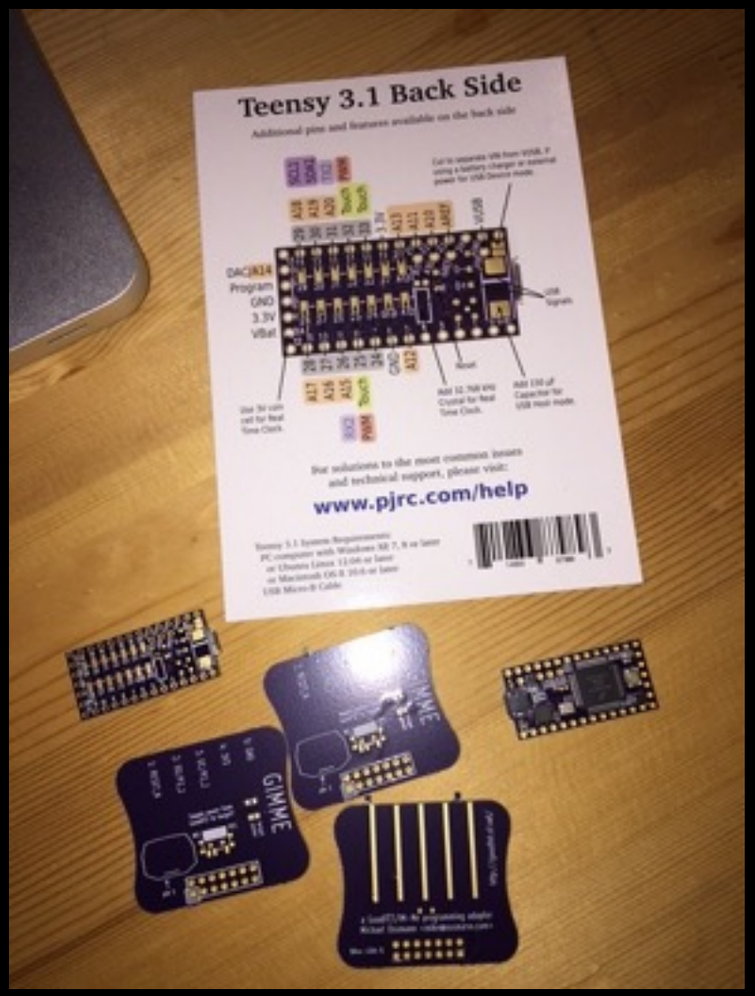
Real men hack pink





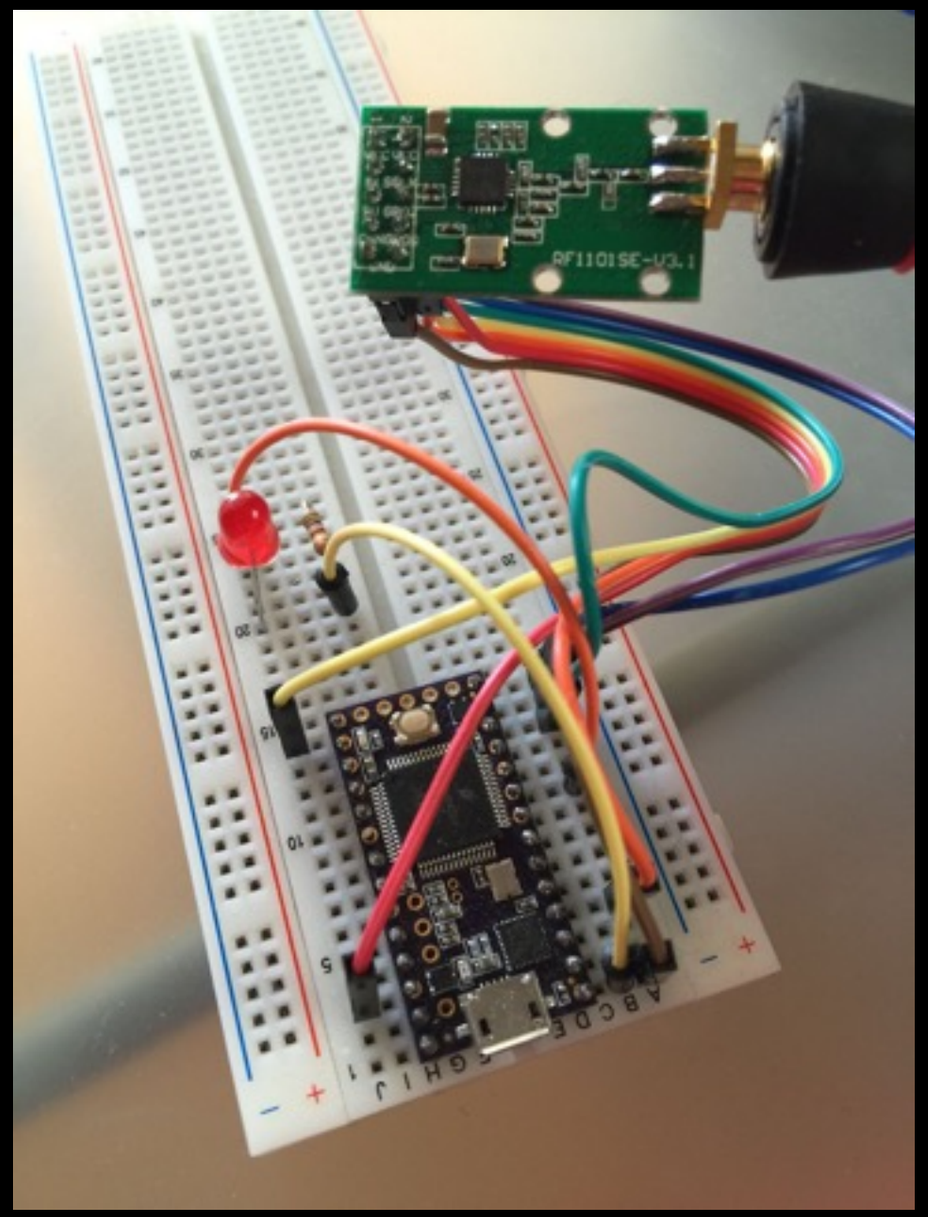
GIMME

Teensy don’t care



pjrc.com/teensy

OSHWPark.com



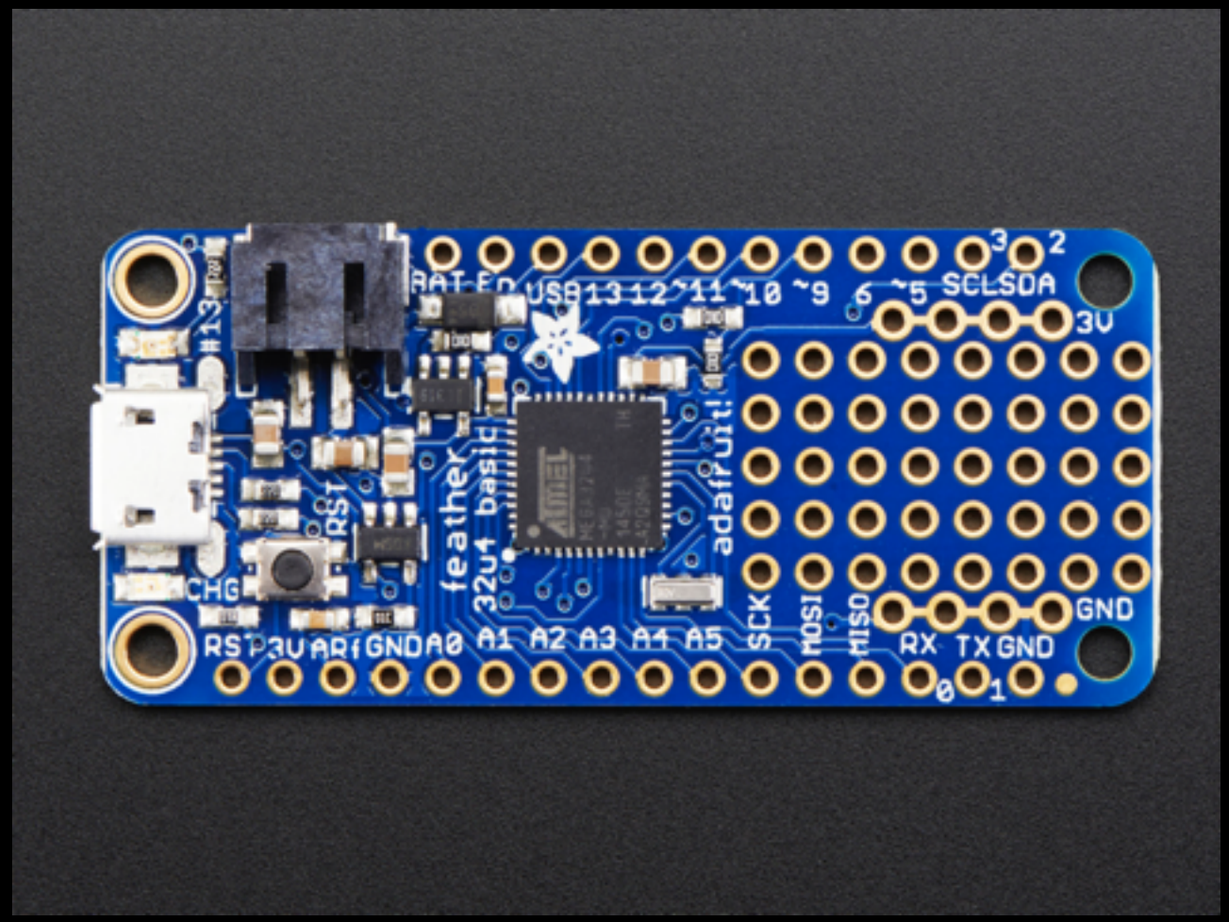
96MHz
256 KB Flash
64 KB OF RAM!

YARD Stick One

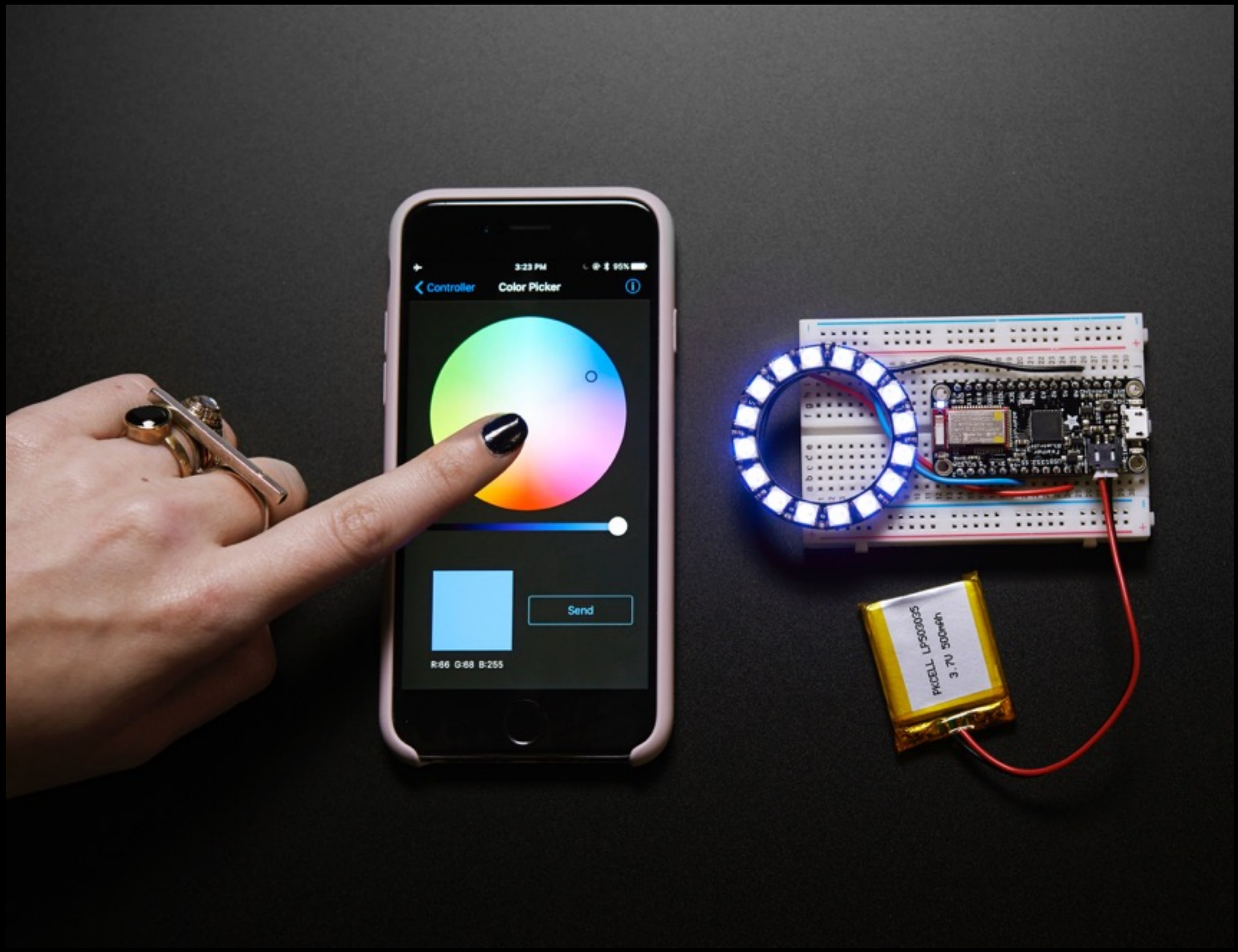
- SMA connector for external antennas such as ANT500
- receive amplifier for improved sensitivity
- transmit amplifier for higher output power
- strong RF performance across the entire operating frequency range
- low pass filter for elimination of harmonics when operating in the 800 and 900 MHz bands
- 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

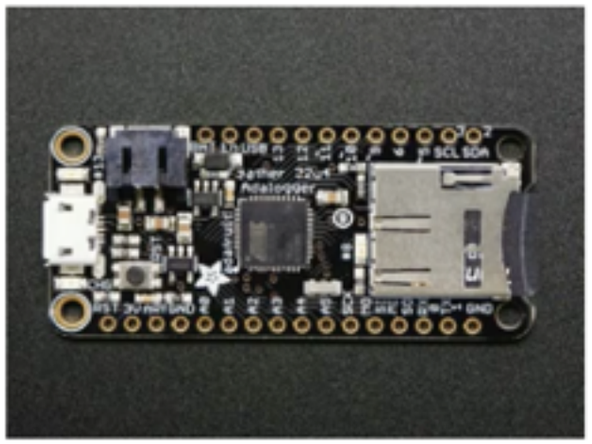


Adafruit Feather 32u4 Bluefruit LE



Adafruit's pretty great

BOARDS / FEATHER



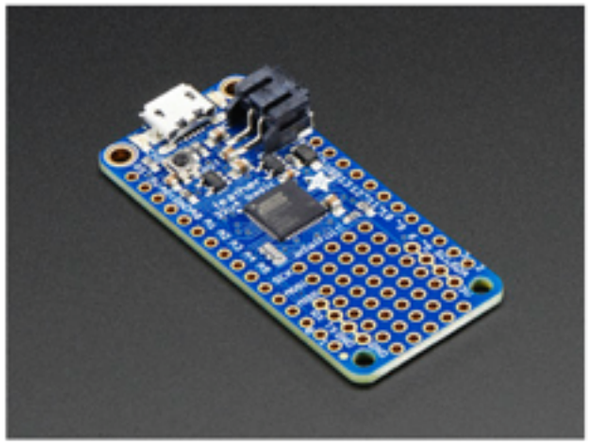
Adafruit Feather 32u4 Adalogger

PRODUCT ID: 2795

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

NOTIFY ME

$21.95
OUT OF STOCK



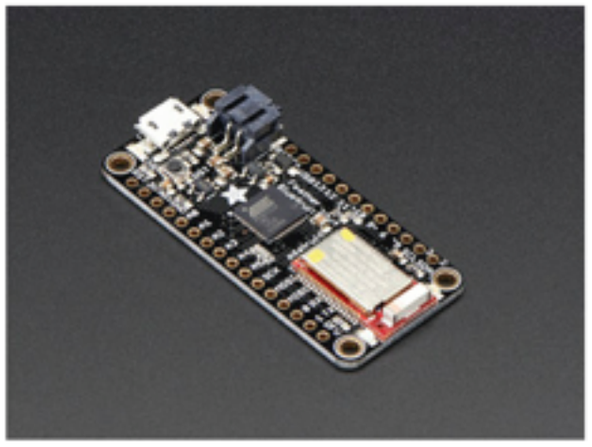
Adafruit Feather 32u4 Basic Proto

PRODUCT ID: 2771

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

ADD TO CART

$19.95
39 IN STOCK



Adafruit Feather 32u4 Bluefruit LE

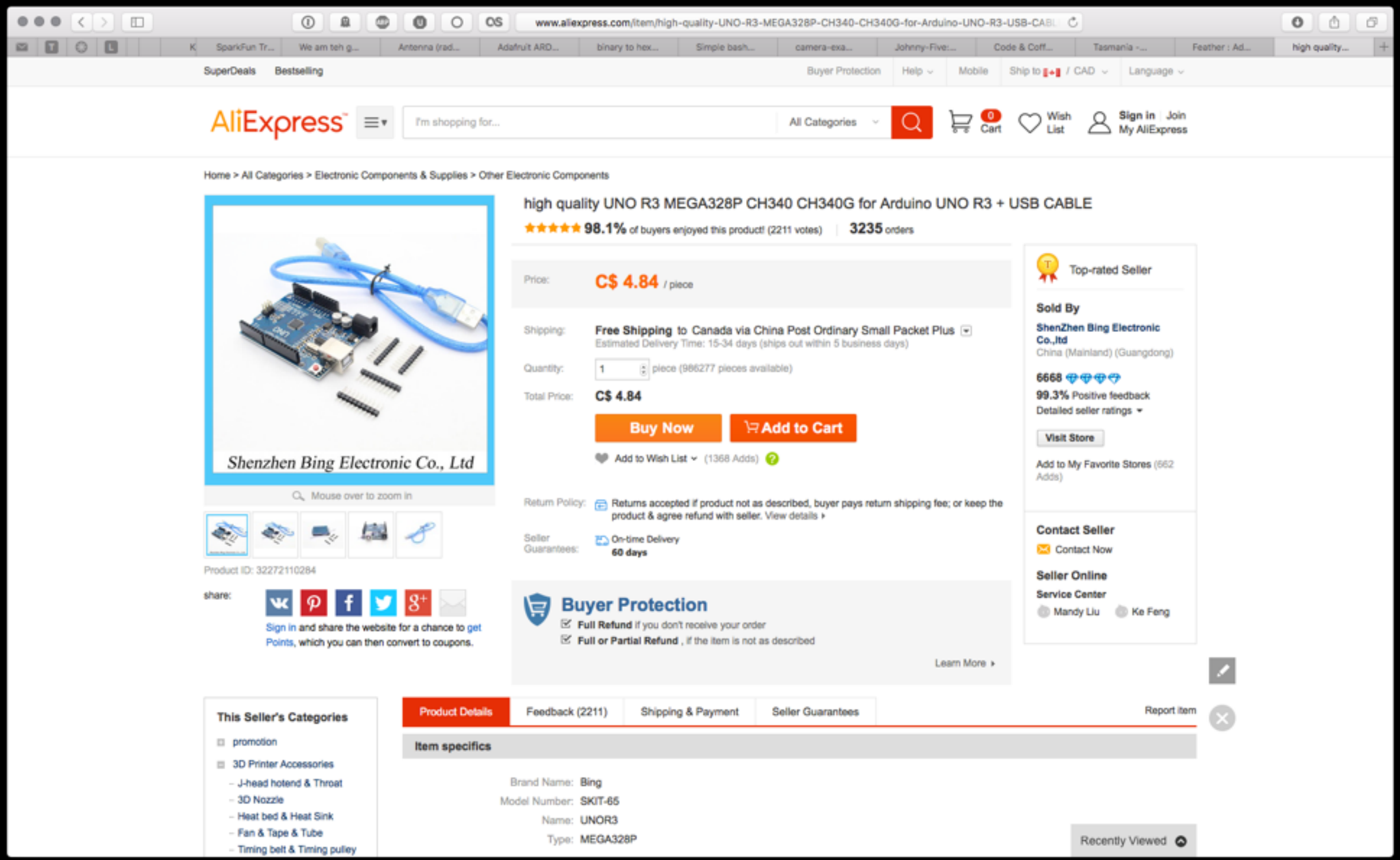
PRODUCT ID: 2829

Feather is the new development board from Adafruit, and like it's namesake it is thin, light, and lets yo
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 BTL

ADD TO CART

$29.95
IN STOCK

Cheap!

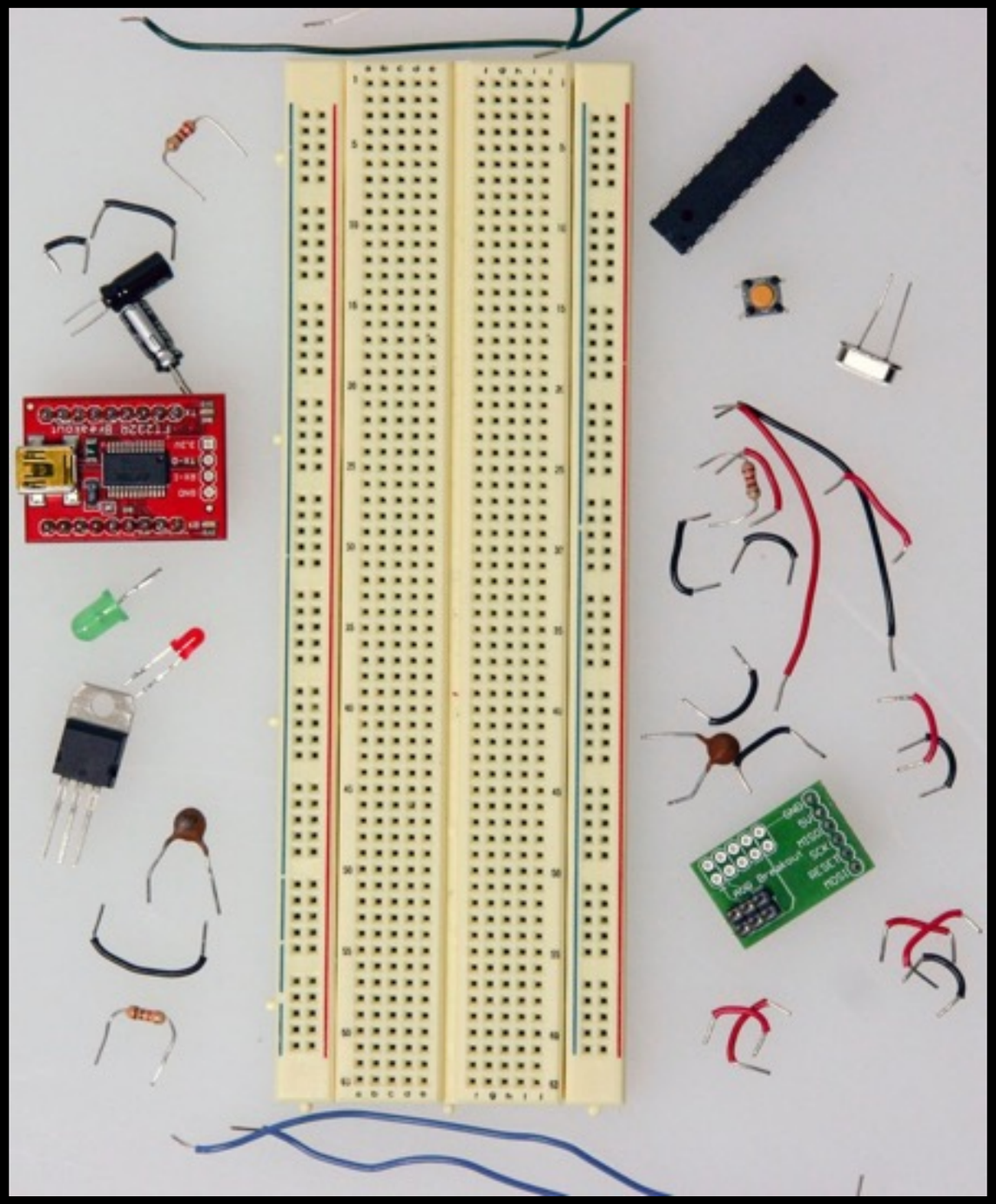

www.aliexpress.com/item/high-quality-UNO-R3-MEGA328P-CH340-CH340G-for-Arduino-UNO-R3-USB-CABL
SuperDeals
Bestselling
Buyer Protection
Help
Mobile
Ship to / CAD
Language
AliExpress
I'm shopping for...
All Categories
0 Cart
Wish List
Sign in | Join
My AliExpress
Home > All Categories > Electronic Components & Supplies > Other Electronic Components
high quality UNO R3 MEGA328P CH340 CH340G for Arduino UNO R3 + USB CABLE
98.1% of buyers enjoyed this product! (2211 votes)
3235 orders
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
Buy Now
Add to Cart
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
Shenzhen Bing Electronic Co., Ltd
Mouse over to zoom in
Product ID: 32272110284
share:
Sign in and share the website for a chance to get Points, which you can then convert to coupons.
Buyer Protection
Full Refund if you don't receive your order
Full or Partial Refund , if the item is not as described
Learn More
Top-rated Seller
Sold By
ShenZhen Bing Electronic Co.,ltd
China (Mainland) (Guangdong)
6668
99.3% Positive feedback
Detailed seller ratings
Visit Store
Add to My Favorite Stores (662 Adds)
Contact Seller
Contact Now
Seller Online
Service Center
Mandy Liu
Ke Feng
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)

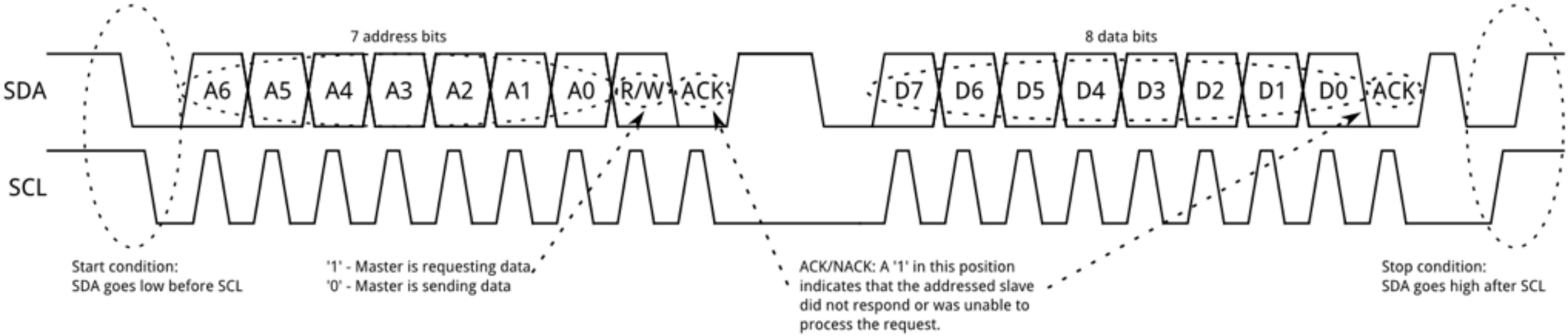
sudo nvram boot-args="kext-dev-mode=1"

Arduino from scratch



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

I2C (Inter-Integrated Circuit)



SPI (Serial Peripheral Interface)

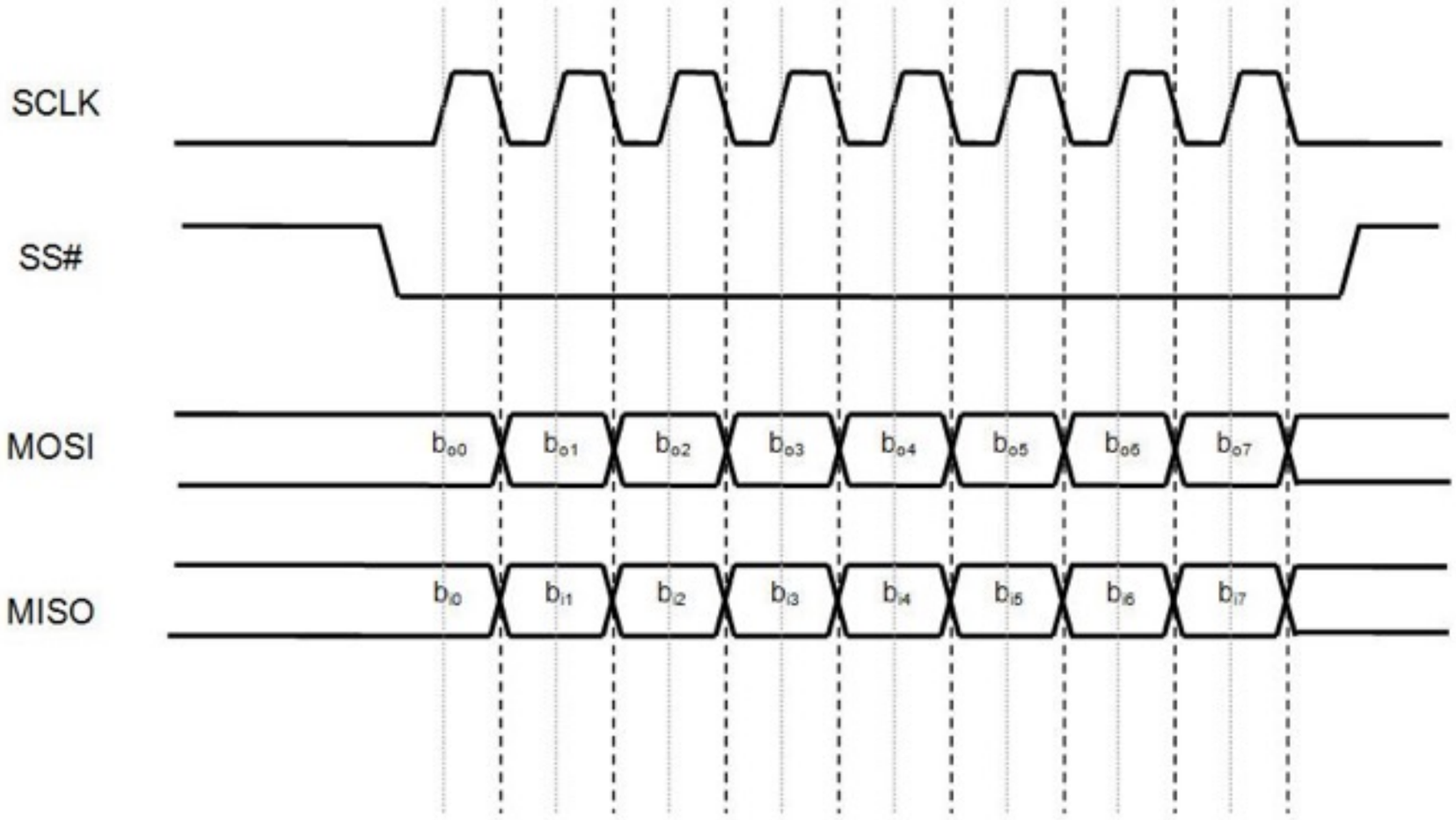
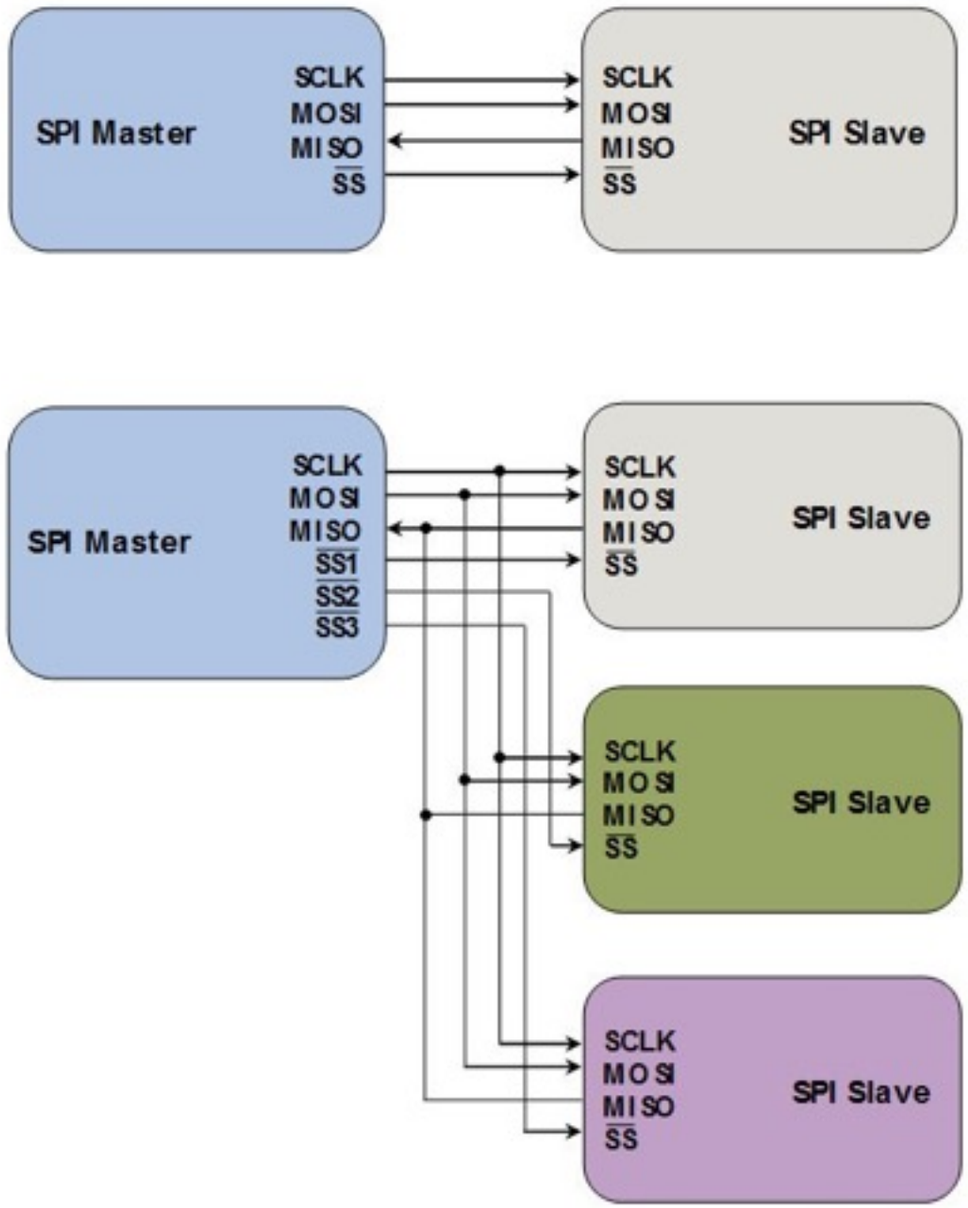


Figure 2 : A simple SPI communication. Data bits on MOSI and MISO toggle on the SCLK falling edge and are sampled on the SCLK rising edge. The SPI mode defines which SCLK edge is used for toggling data and which SCLK edge is used for sampling data.

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

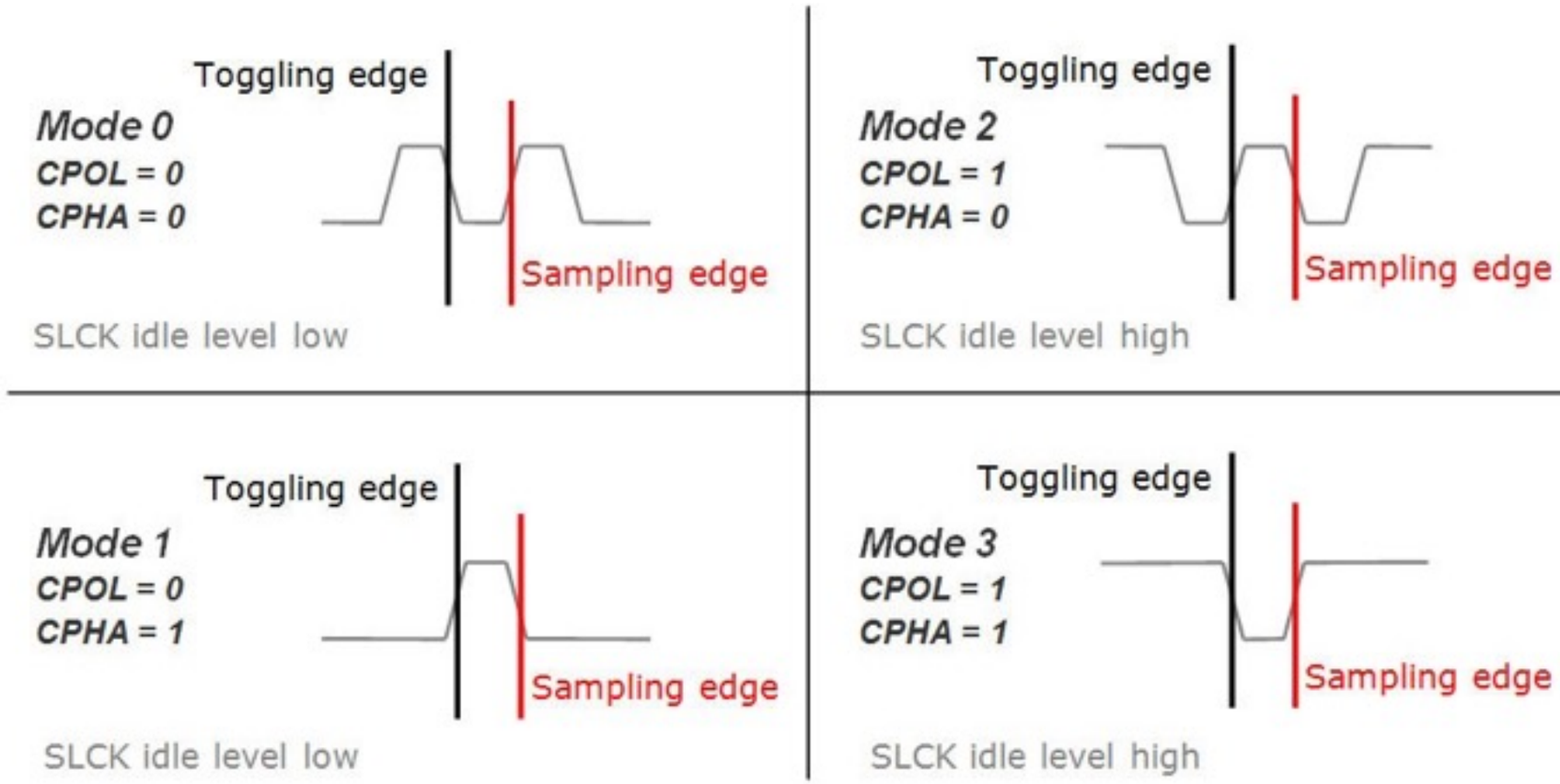
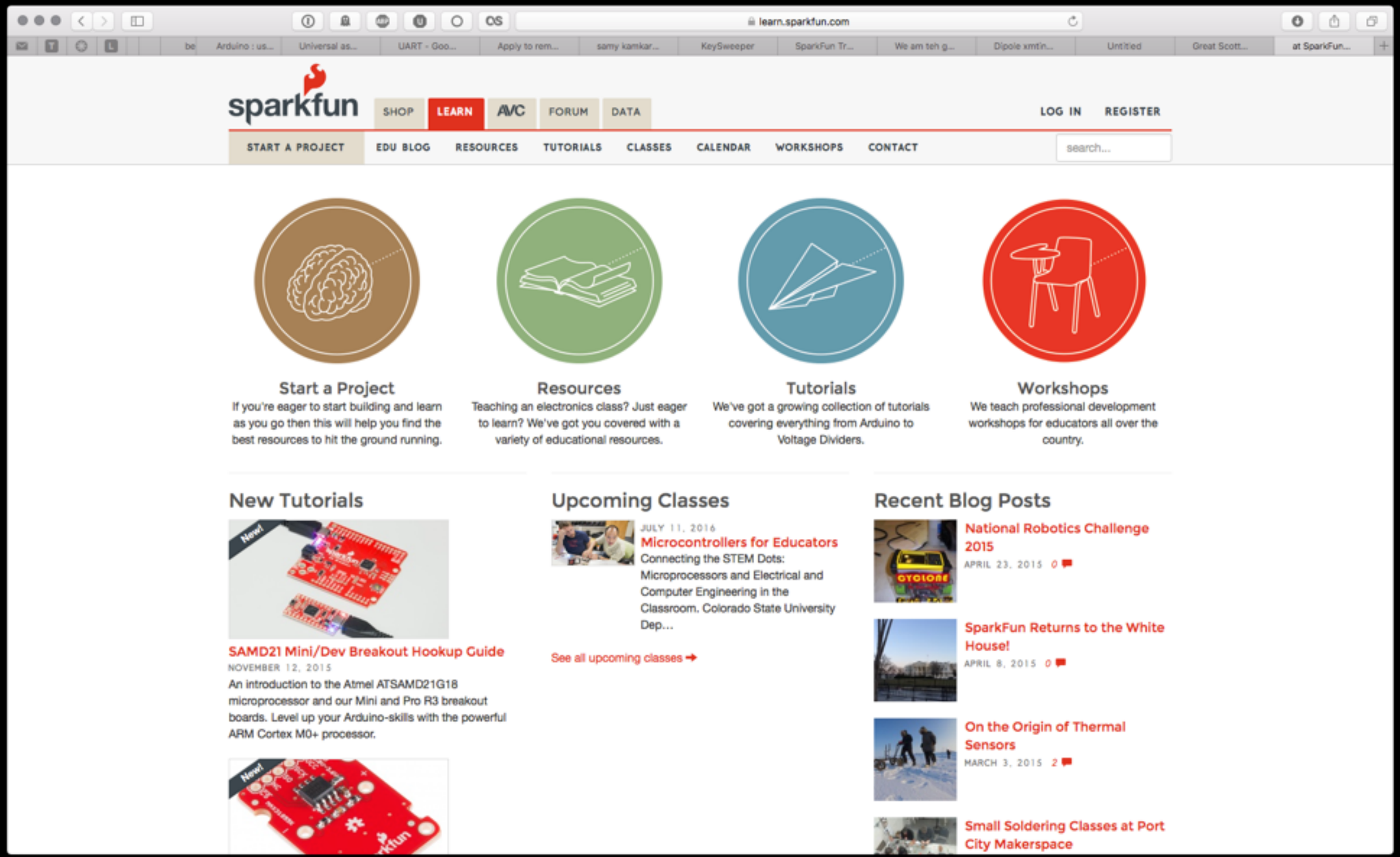
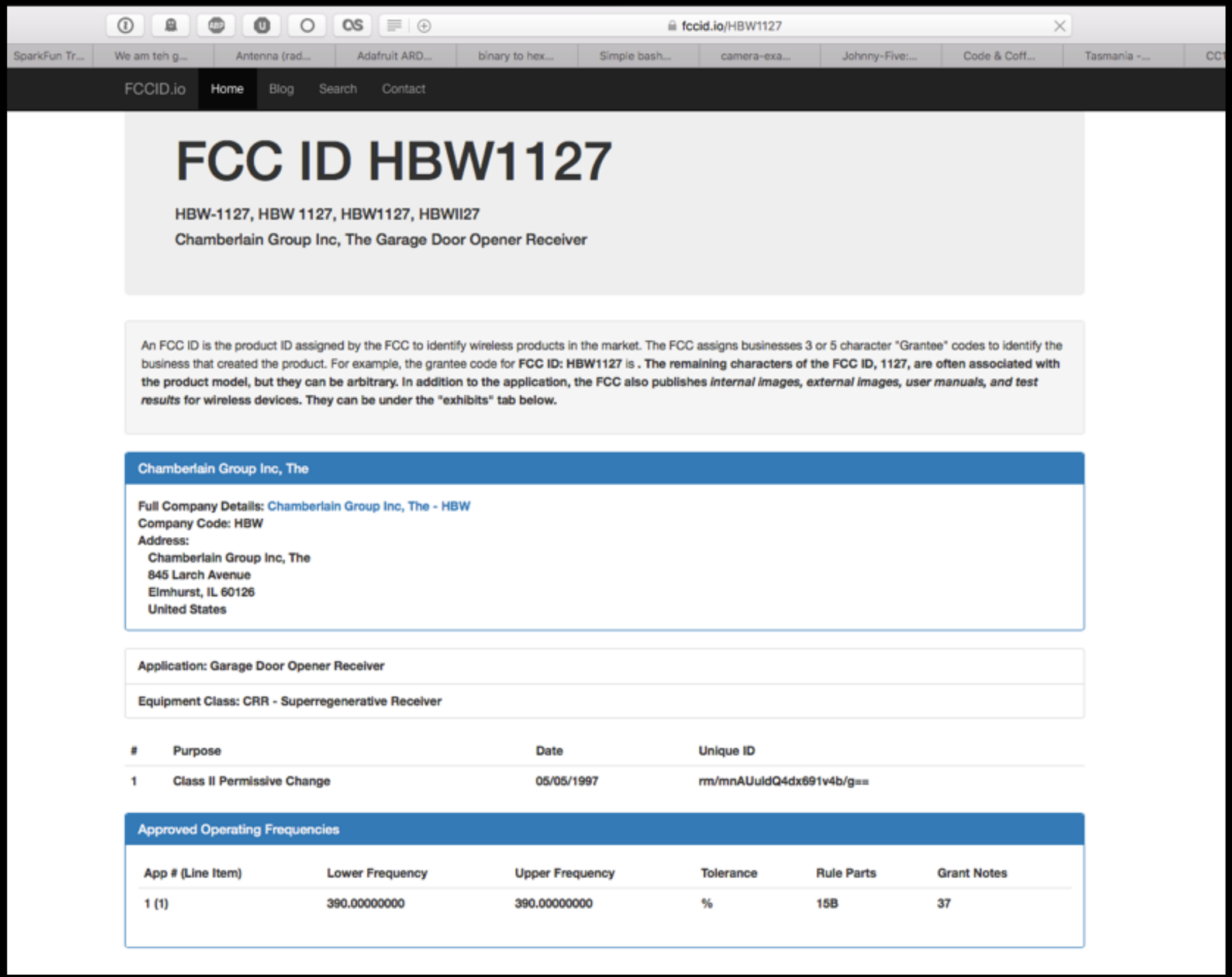


Figure 3 : SPI modes are defined with the parameters 'CPOL' – clock polarity and 'CPHA' – clock phase, which explicitly define 3 parameters: the edges used for data sampling and data toggling and the SCL clock signal idle level – that is the conventional level SCLK is set at when the bus is not in communication.

learn.sparkfun.com



fccid.io 390MHz



Garage door operating frequencies

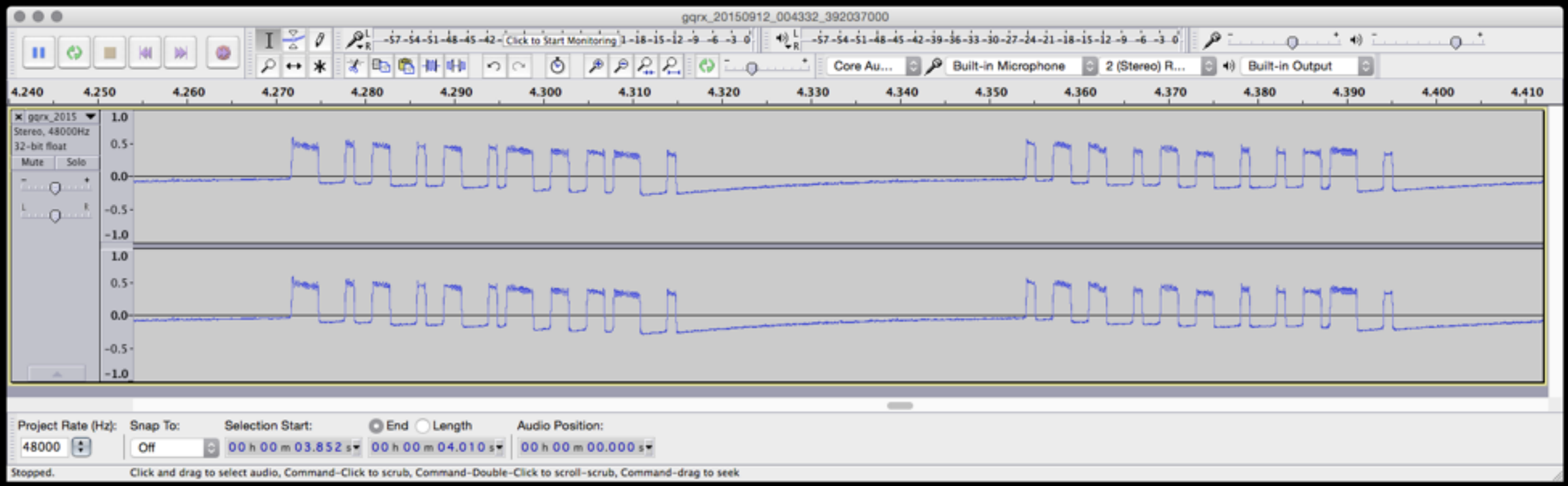
3.2 Operational Mode

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

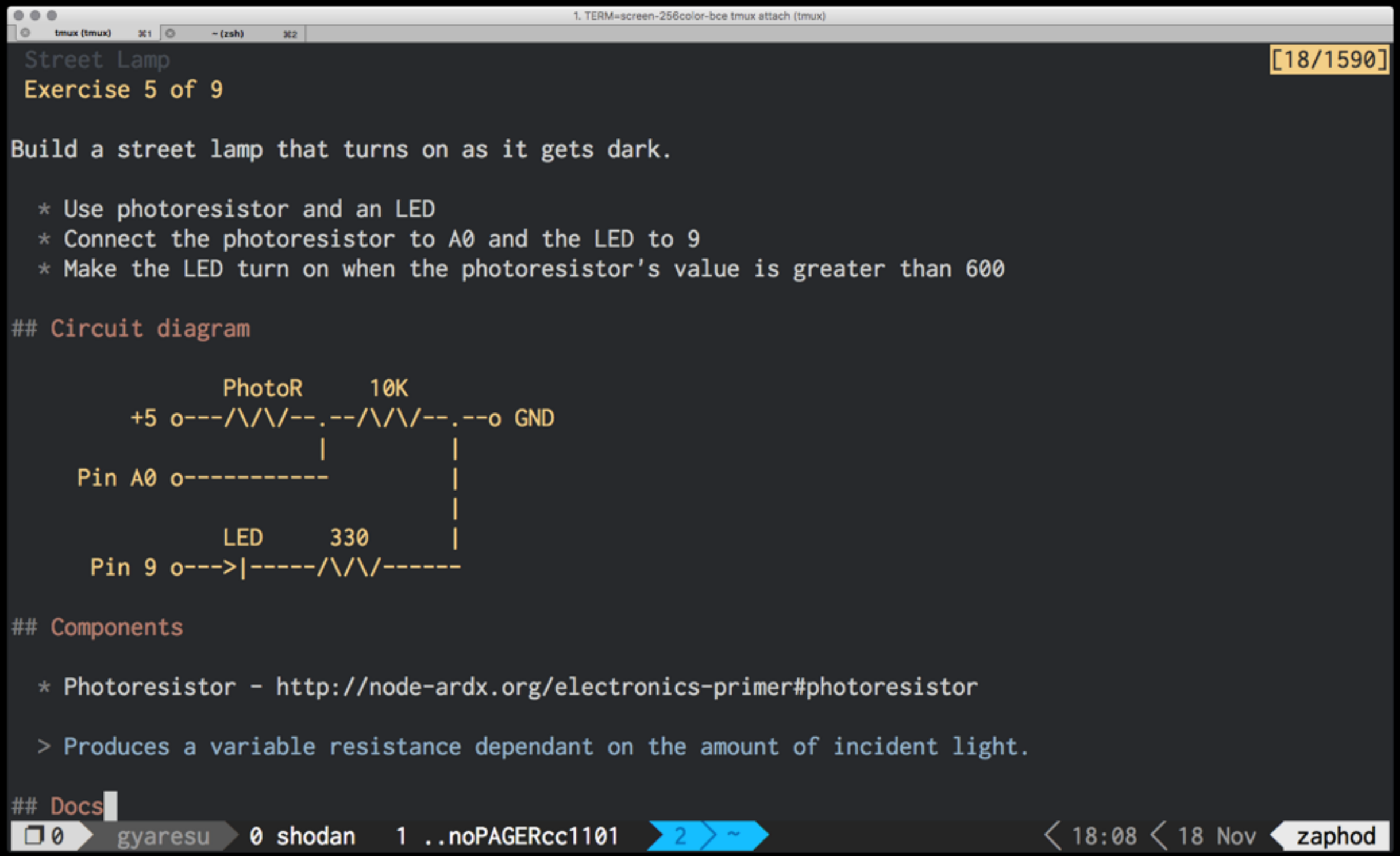
Thanks random data sheet from the internet!

Woot!

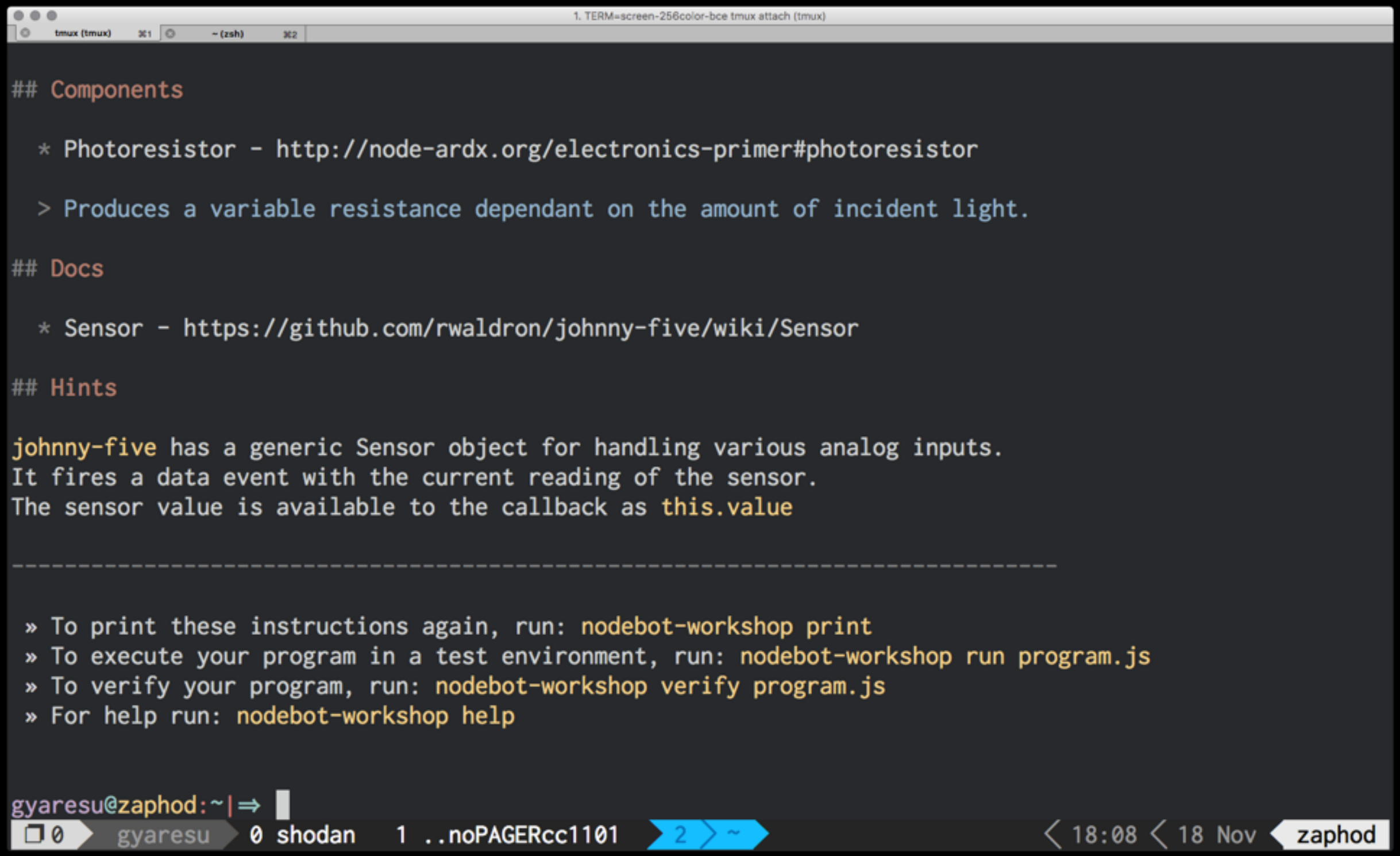

gqrx_20150912_004332_392037000
Click to Start Monitoring
Core Au...
Built-in Microphone
2 (Stereo) R...
Built-in Output
4.240 4.250 4.260 4.270 4.280 4.290 4.300 4.310 4.320 4.330 4.340 4.350 4.360 4.370 4.380 4.390 4.400 4.410
gqrx_2015
Stereo, 48000Hz
32-bit float
Mute
Solo
1.0
0.5
0.0
-0.5
-1.0
Project Rate (Hz):
48000
Snap To:
Off
Selection Start:
End
Length
Audio Position:
00 h 00 m 03.852 s
00 h 00 m 04.010 s
00 h 00 m 00.000 s
Stopped.
Click and drag to select audio, Command-Click to scrub, Command-Double-Click to scroll-scrub, Command-drag to seek

NodeSchool.io/vancouver
@nodeschoolyvr

Fun and instructional



Works? Now plug it into an Arduino!



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

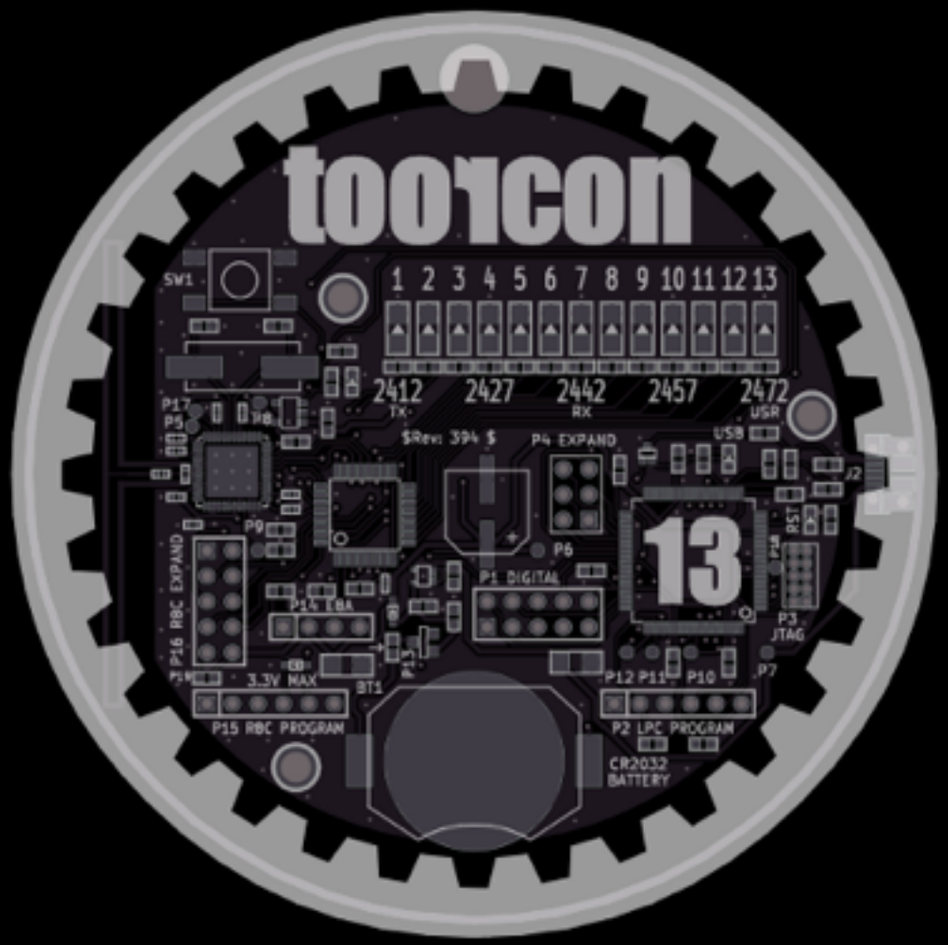
Products

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

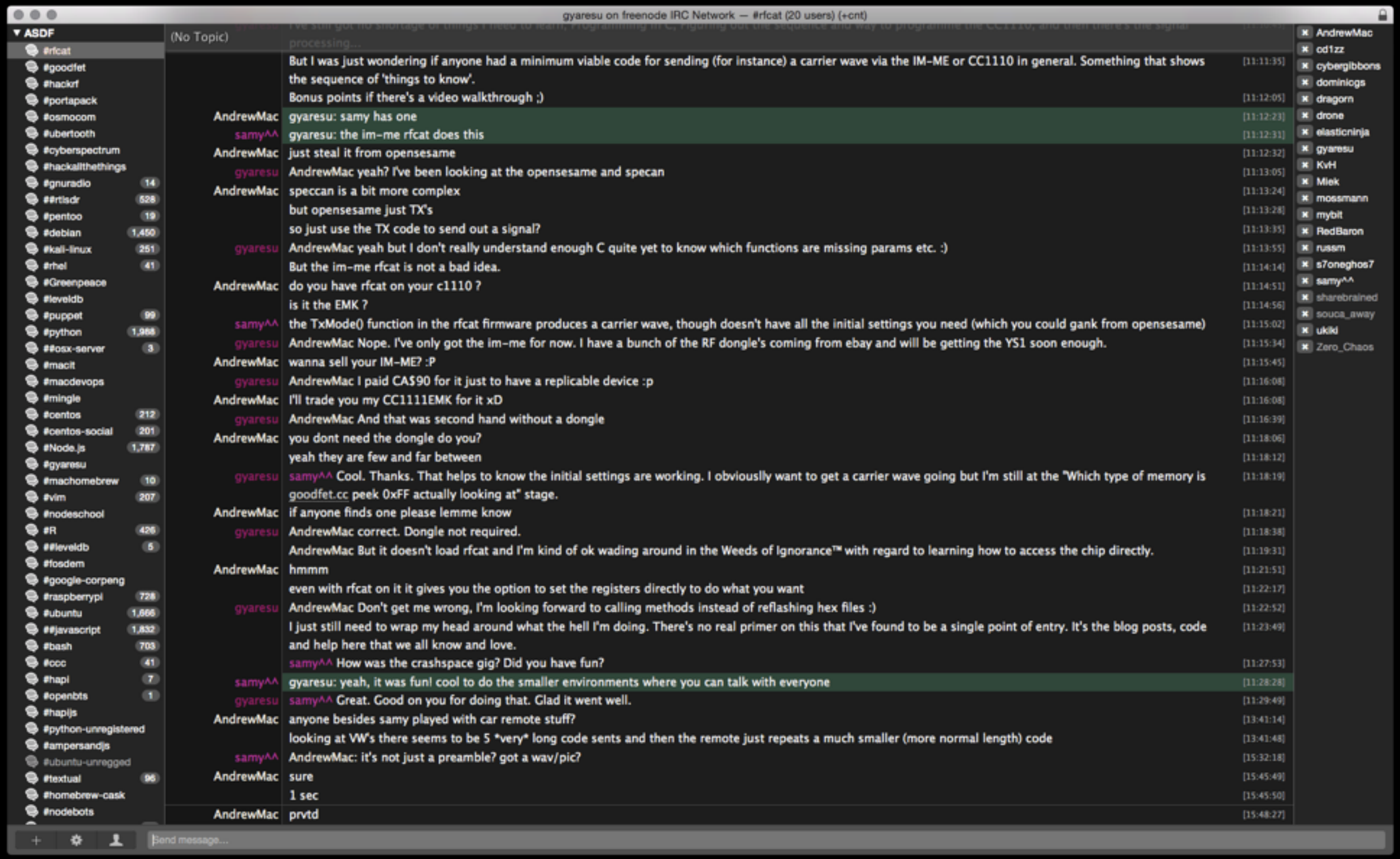
Menstrual Fluid Volume

Menstrual Fluid Color

Safe,100% Medical Grade Silicone

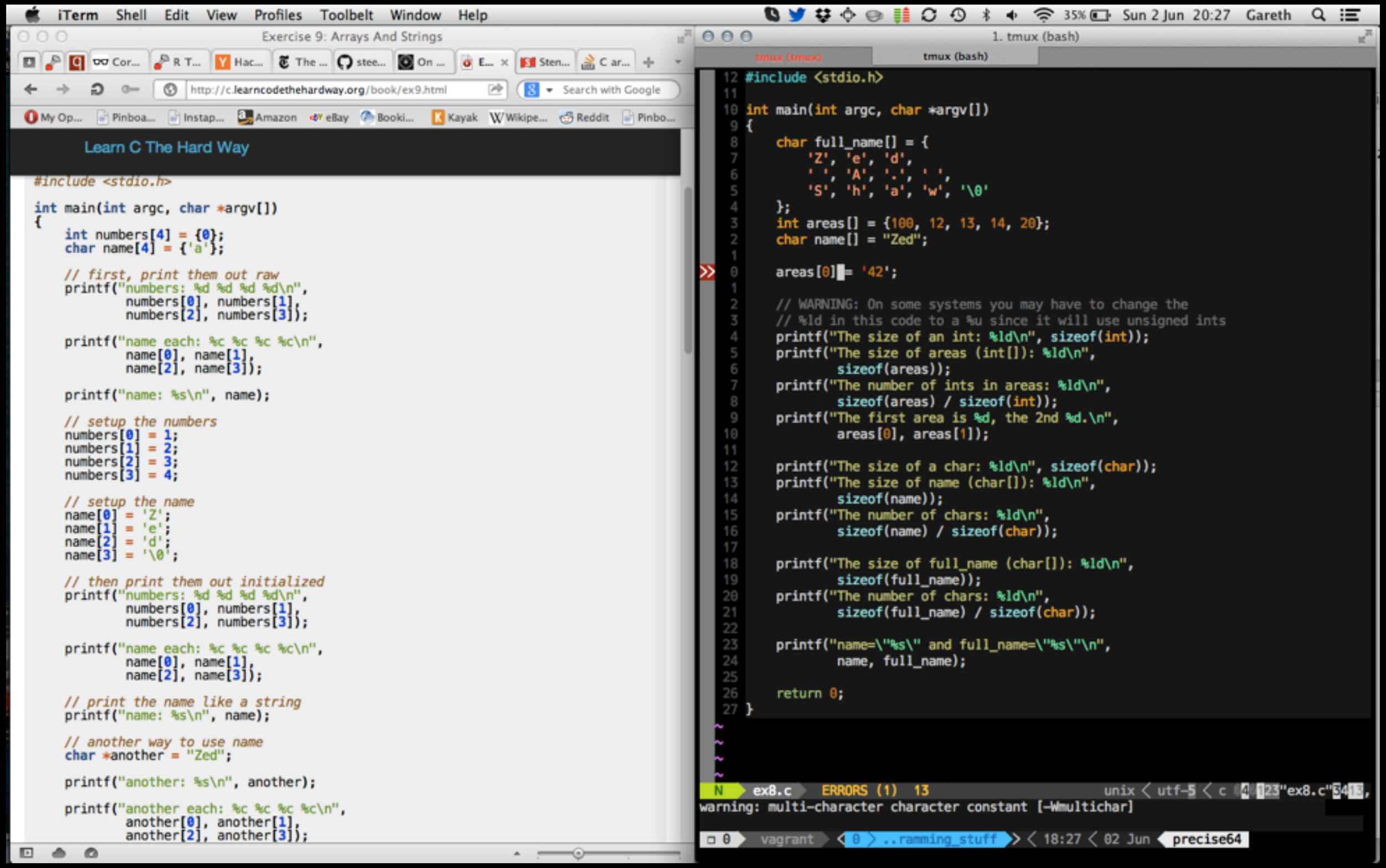
Safe BLE 4.0

Internet Relay Chat (Freenode)

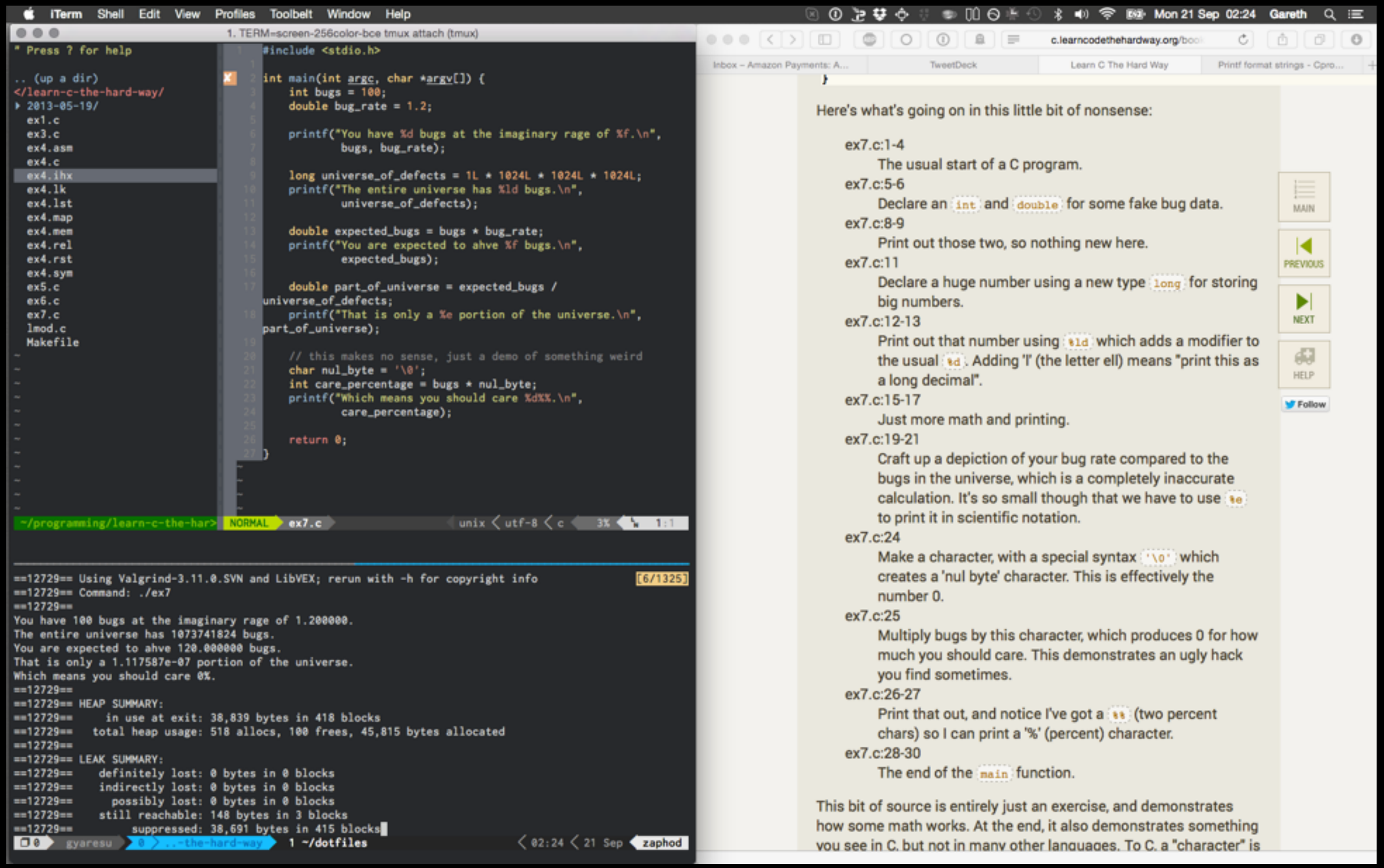


People are amazing

2013 LCTHW


iTerm Shell Edit View Profiles Toolbelt Window Help
35% Sun 2 Jun 20:27 Gareth
Exercise 9: Arrays And Strings
http://c.learncodethehardway.org/book/ex9.html
Search with Google
Learn C The Hard Way
#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]);
1. tmux (bash)
tmux (tmux)
tmux (bash)
#include <stdio.h>
int main(int argc, char *argv[])
{
char full_name[] = {
'Z', 'e', 'd',
' ', 'A', '.', ' ',
'S', 'h', 'a', 'w', '\0'
};
int areas[] = {100, 12, 13, 14, 20};
char name[] = "Zed";
areas[0] = '42';
// WARNING: On some systems you may have to change the
// %ld in this code to a %u since it will use unsigned ints
printf("The size of an int: %ld\n", sizeof(int));
printf("The size of areas (int[]): %ld\n",
sizeof(areas));
printf("The number of ints in areas: %ld\n",
sizeof(areas) / sizeof(int));
printf("The first area is %d, the 2nd %d.\n",
areas[0], areas[1]);
printf("The size of a char: %ld\n", sizeof(char));
printf("The size of name (char[]): %ld\n",
sizeof(name));
printf("The number of chars: %ld\n",
sizeof(name) / sizeof(char));
printf("The size of full_name (char[]): %ld\n",
sizeof(full_name));
printf("The number of chars: %ld\n",
sizeof(full_name) / sizeof(char));
printf("name=\"%s\" and full_name=\"%s\"\n",
name, full_name);
return 0;
}
N ex8.c ERRORS (1) 13 unix < utf-5 < c "ex8.c"
warning: multi-character character constant [-Wmultichar]
0 vagrant 0 ..ramming_stuff 18:27 02 Jun precise64

2015 LCTHW


iTerm Shell Edit View Profiles Toolbelt Window Help
Mon 21 Sep 02:24 Gareth
1. TERM=screen-256color-bce tmux attach (tmux)
" 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
#include <stdio.h>
int main(int argc, char *argv[]) {
int bugs = 100;
double bug_rate = 1.2;
printf("You have %d bugs at the imaginary rage of %f.\n",
bugs, bug_rate);
long universe_of_defects = 1L * 1024L * 1024L * 1024L;
printf("The entire universe has %ld bugs.\n",
universe_of_defects);
double expected_bugs = bugs * bug_rate;
printf("You are expected to ahve %f bugs.\n",
expected_bugs);
double part_of_universe = expected_bugs /
universe_of_defects;
printf("That is only a %e portion of the universe.\n",
part_of_universe);
// this makes no sense, just a demo of something weird
char nul_byte = '\0';
int care_percentage = bugs * nul_byte;
printf("Which means you should care %d%%.\n",
care_percentage);
return 0;
}
~/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
gyaresu 0 ../-the-hard-way 1 ~/dotfiles 02:24 21 Sep zaphod
c.learncodethehardway.org/boo
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
MAIN
PREVIOUS
NEXT
HELP
Follow

Gareth
me@gareth.codes

twitter.com
github.com

@gyaresu