HOP **HACKING** HEDY

atlas, cutaway and Q

AUTHORS

- atlas
  - Code monkey
  - Magical debugging wizard
- cutaway
  - Radio-Fu
  - Supreme dongle overlord
- Q
  - Hardware-Fu
  - Creator of sexy slides

AGENDA

- **Prelude**
  - **Hedy and her Patent**
- **FHSS**WTF
  - Technical explanation
- **FHSS** Usage and Analysis
  - Case examples
    - Problems revolving around FHSS
    - Possible solutions
- **Project**
  - What we are developing
    - Hardware: Devices Utilized, Debugging and Wiring Instructions
    - Developing and Using the Code Base
  - Practical Applications for decoding hopping patterns

PRELUDE

- Explanation of the Title
  - What is a **HEDY**?
  - Surprisingly not a lot of people understood the title
    - That’s right, a focus group was held
  - So I must explain!
- Alternative Title
  - “3 Dudes and a Dongle”
  - Not sure we would have been accepted

PRELUDE

This is a Hedy ☺

PRELUDE

- Hedy Lamarr
  - Inventor
    - Supremely nerdy
    - nerd_chick++;
  - Actress
    - Less important here

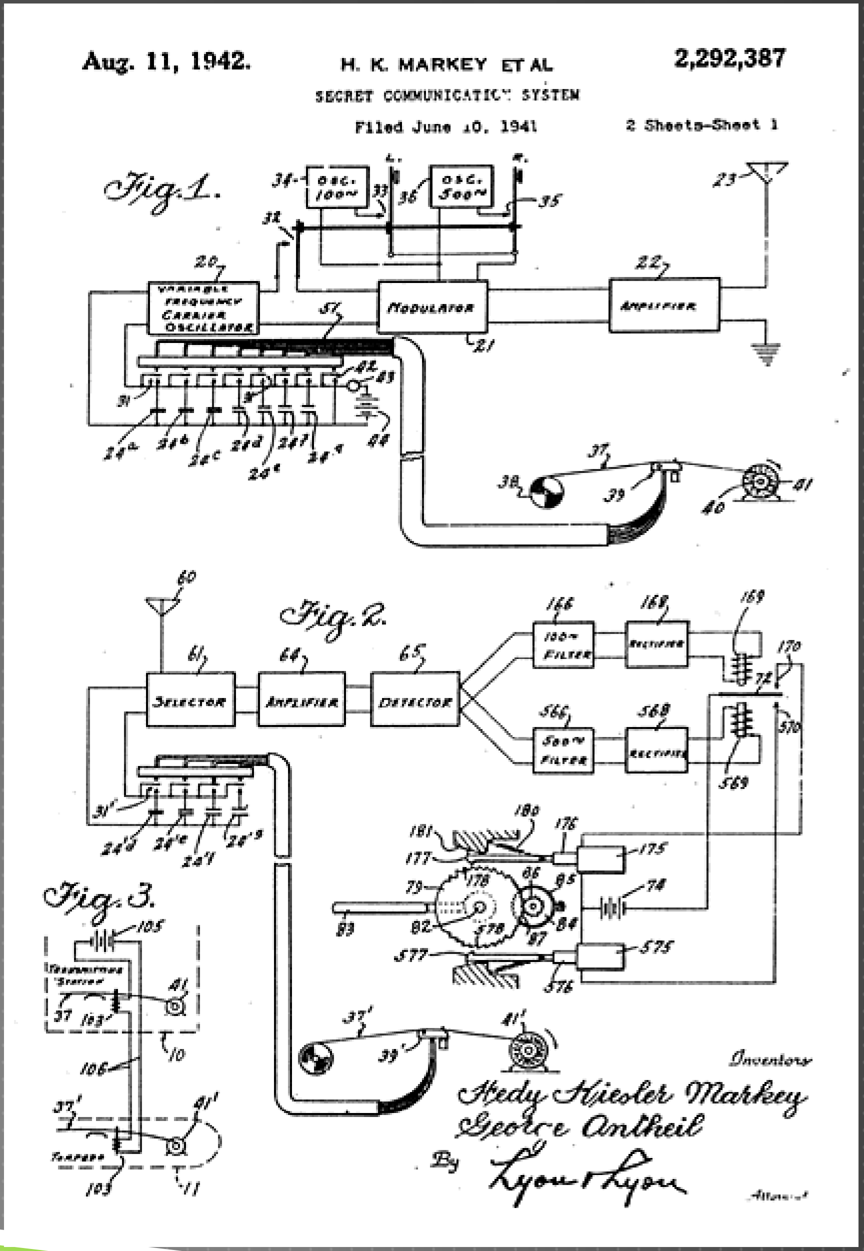
    Sidenote: Military used her celebrity to sell War Bonds instead of her brains to join a panel of scientists during the war. ☹

PRELUDE

- Inventor of “secret communication system”
  - Along with George Antheil
  - Patent: 2,292,387
  - Filed: June 10, 1941
- Recognized by EFF in 1997 for her achievements
  - Go EFF

PRELUDE

- Secret communication
  - **TX**
    - VFCO -> Modulator -> Amplifier
  - **RX**
    - Selector -> Amplifier -> Detector
      - Noise reduction (filtering)
- Ya, that’s a piano roll
  - Used to change between 88 freqs.
  - Ah, the days before digital
- End Result
  - Impressive for the days of Analog



Aug. 11, 1942. H. K. MARKEY ET AL 2,292,387

SECRET COMMUNICATIC'' SYSTEM

Filed June 10, 1941 2 Sheets-Sheet 1

Inventors

Hedy Kiesler Markey

George Antheil

By Lyon & Lyon

Attorneys

FHSSWTF

- **FHSS** - Frequency Hopping Spread Spectrum
  - Transmission method
    - Carrier is rapidly switched in the frequency domain
    - Multiple channels
      - Depends upon bandwidth
        - Bluetooth – 79 MHz
          - @ 1MHz channel spacing – 79 channels
        - Smart Meters – 26MHz
          - @ 500 kHz channel spacing – 52 channels

FHSSWTF

- Pseudo-random sequence to choose frequencies
  - Known by Transmitter and Receiver
    - Multiple sync methods
  - Stored algorithm within memory generates pattern
    - Creates spread code sequence tables
    - Possibly stored within a piano roll
      - If you’re Hedy Lamarr

FHSS USAGE

- Who uses them?
  - Military
    - SINCGARS
      - Single Channel Ground and Airborne Radio System
        - Voice-and-Data
        - 25 kHz Channels
        - 30-88 MHz
          - 120-352 channels
          - 100 hops/second
            - Very slow
            - Expected to be replaced by an SDR solution
              - Deployment issues

FHSS ANALYSIS

- SINCGARS
  - **Problem**: 500,000 units purchased[1] Until the 2008 improvement programs, how many radios were relying on FHSS for security?
    - How many **still** are?
  - **Solution:** Upgrades provide voice-encryption, though backwards compatible with old radio hardware.
    - **Does security still exist if you're supporting ancient hardware?**
    - **SDR solution provides field-programmable devices**
      - **Interesting!**
      - **See: JTRS Project**

[1] – Reference: http://defensesystems.com/microsites/2010-peo-c3t/not-your-father-radios.aspx

FHSS USAGE

- Commercial
  - Bluetooth
    - Bandwidth:79 MHz
    - Channel Spacing: 1 MHz
    - Total Channels 79
      - 3200 clock cycles
        - Hops every other
        - 1600 hops/second

FHSS ANALYSIS

- Bluetooth
  - **Problem:** With commercially available hardware (USRPv2) you can generate hopping patterns for an entire technology within 24 hours and utilize this for future patterns.
  - **Solution:** We can’t rely on FHSS as an implementation of a security model within our technology. We must do better.
    - **Encryption! Yes, please.**

FHSS IS NOT ENCRYPTION

- “In FHSS, the frequencies to be used in the hopping sequence may be selected by the user. In the unlicensed band, any group of 26 frequencies or more (out of the 79 available) is legal. To “tune in”, a listener should know the number of frequencies selected in the system, the actual frequencies, the hopping sequence, as well as the dwell time! The FHSS modulation acts as a layer 1 encryption process. There could be no need for application level encryption!”
  - *Reference: “FHSS vs DSSS in Broadband Wireless Access and WLAN”*
    - Sorin M. Schwartz

FHSS ANALYSIS

- Why do we keep using it?
  - Helps prevent overuse in unlicensed bands
    - Especially with adaptive technology
- Attack Vectors
  - Must have the hopping pattern to jam or receive entire data stream
    - Break the PRNG associated with the algorithm
      - Obtain spread codes
    - Analyze channel data in time domain fast enough to catch the hops until repeats start to occur
    - Generate the entire pattern for all clock values
      - *Reference: Ossmann/Spill Shmoo 2009*

SOLUTIONS

- Learn from our experience
  - We have seen this issue in the Smart Grid industry
    - Some 1st generation devices relied on FHSS as primary prevention of eavesdropping
    - 2nd and 3rd generation introduced encryption and signing of keys
  - Do not rely on a single point of failure

PROJECT

- Inspiration
  - False assumptions that speed and pseudo random sequence creates a secure transmission
    - Simply untrue with the powerful technology we have available
  - Let’s build some devices that
    - Can be configured for known ISM bands
    - Automatically analyze channel spacing
    - Can Decode FHSS Hopping Patterns
    - Utilize a custom code-base with far-reaching capabilities to get people started
  - That is the goal of our project …

PROJECT

- In 1942 Hedy Lamarr and George Antheil helped develop a system to assist in the prevention of jamming American radio-controlled torpedoes.

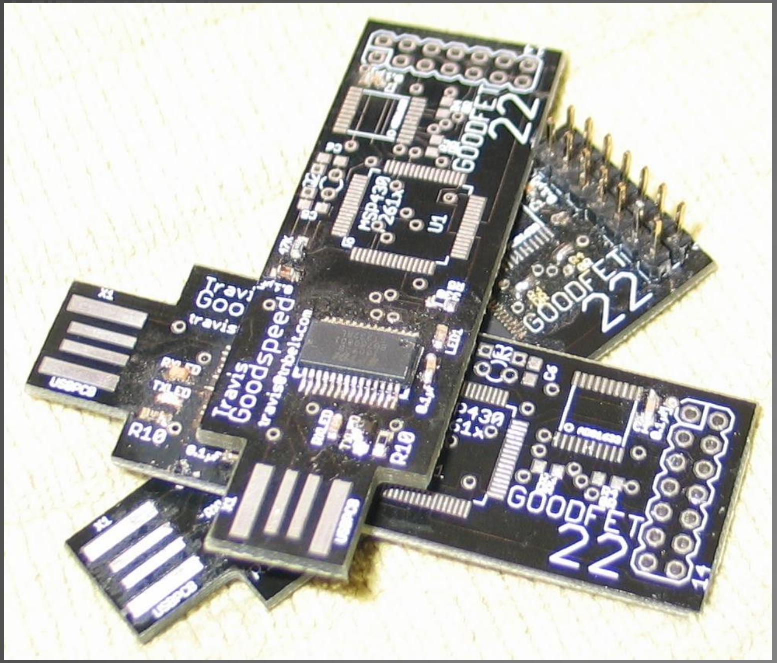
- 69 years later, it’s time to upgrade …

H4RDW4RE

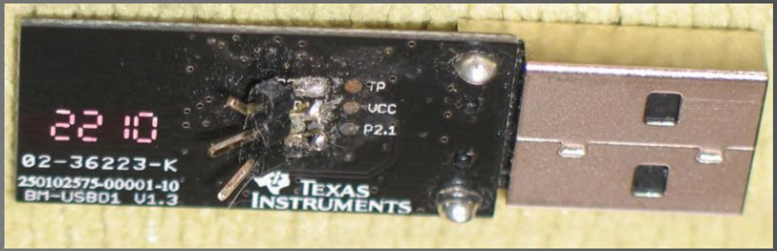
- Cutaway’s Initial Interest
  - Started out as bootstrap for hardware and firmware interaction
  - Reading is one thing, but you don’t really know anything until you have destroyed something……that belongs to atlas
  - Get to eat Sushi…at ShmooCon
  - Get Tuna to buy me more drinks…at ShmooCon

H4RDW4RE

Goodfet Graveyard



eZ430-Chronos Dongle

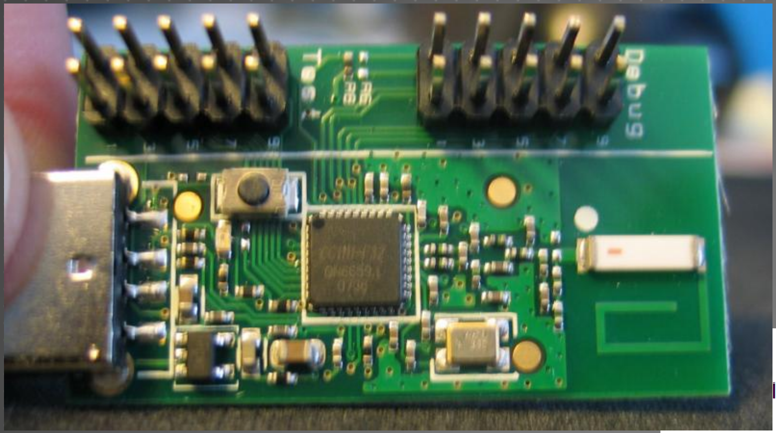


H4RDW4RE

- GoodME Project
  - Cons
    - Already being worked by atlas, Q, Travis, and Mike
    - Pink might be pretty but some people just don’t get it
  - Pros
    - CC1110-based
    - Great place to start
    - Actively being worked

H4RDW4RE

- CC1111EMK868-915 Evaluation Module Kit
  - CC1111-based
  - All pins broken out
  - Programmable via Goodfet
  - Goodfet.cc interacts via Data Debug



l for a minimum connector layout on external target

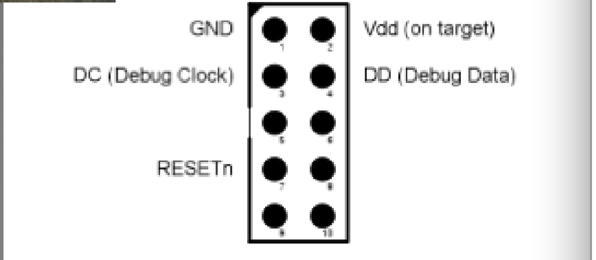
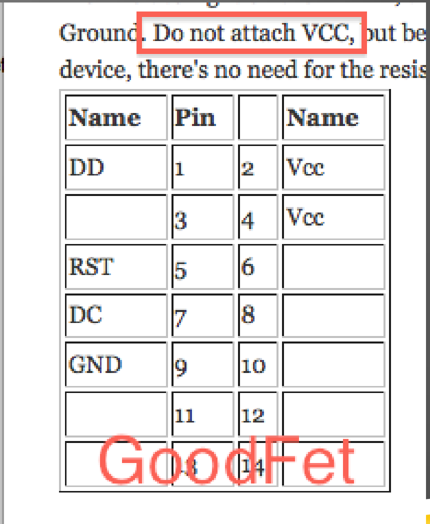


Figure 5. Minimum Debug Connector Pinout (top view)

CC1111EM

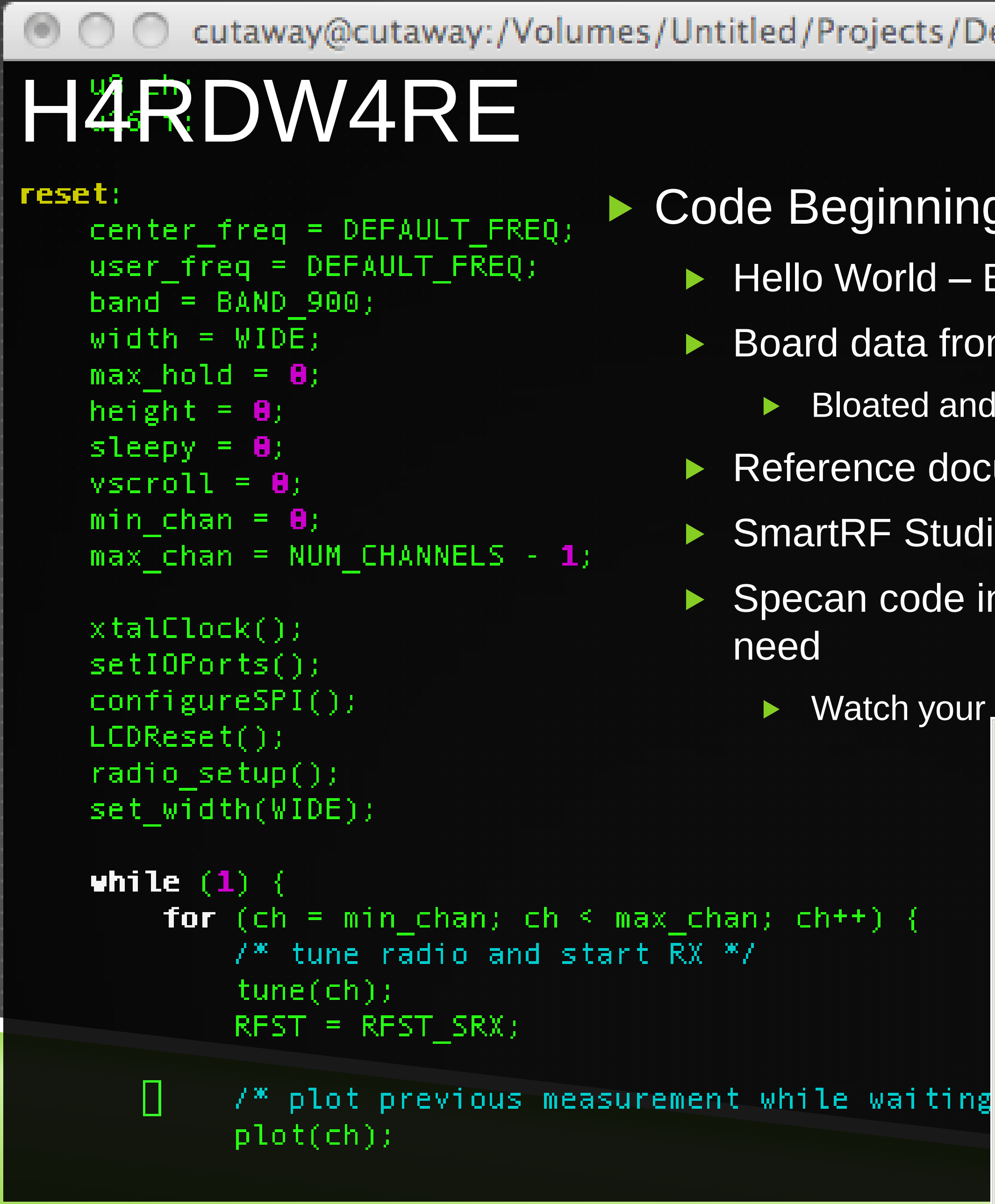


Ground. Do not attach VCC, but be
device, there's no need for the resis

Name	Pin		Name
DD	1	2	Vcc
	3	4	Vcc
RST	5	6	
DC	7	8	
GND	9	10	
	11	12	
	13	14	

GoodFet

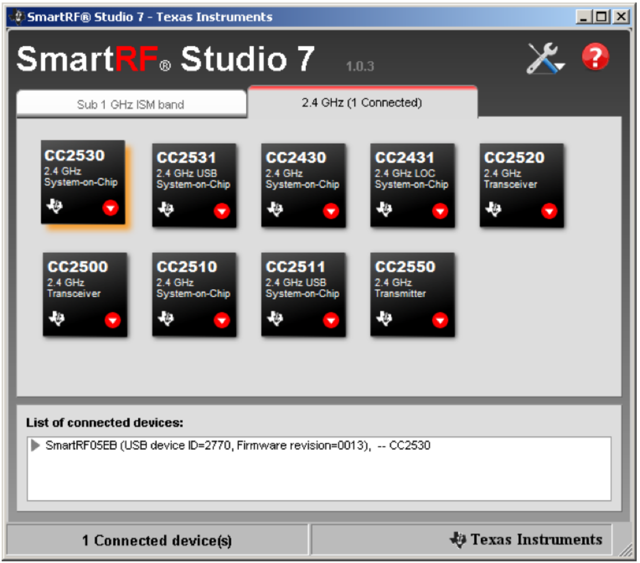
H4RDW4RE



cutaway@cutaway:/Volumes/Untitled/Projects/Dev/cc1111/Freq_Hopping/s...

```
reset:
    center_freq = DEFAULT_FREQ;
    user_freq = DEFAULT_FREQ;
    band = BAND_900;
    width = WIDE;
    max_hold = 0;
    height = 0;
    sleepy = 0;
    vscroll = 0;
    min_chan = 0;
    max_chan = NUM_CHANNELS - 1;

    xtalClock();
    setIOPorts();
    configureSPI();
    LCDReset();
    radio_setup();
    set_width(WIDE);

    while (1) {
        for (ch = min_chan; ch < max_chan; ch++) {
            /* tune radio and start RX */
            tune(ch);
            RFST = RFST_SRX;

            /* plot previous measurement while waiting
            plot(ch);
```

- Code Beginnings
  - Hello World – Blinking LED
  - Board data from TI’s IAR Demo
    - Bloated and Complicated
  - Reference documents for CC111*
  - SmartRF Studio – provides radio data
  - Specan code includes everything you need
    - Watch your clocks and pins



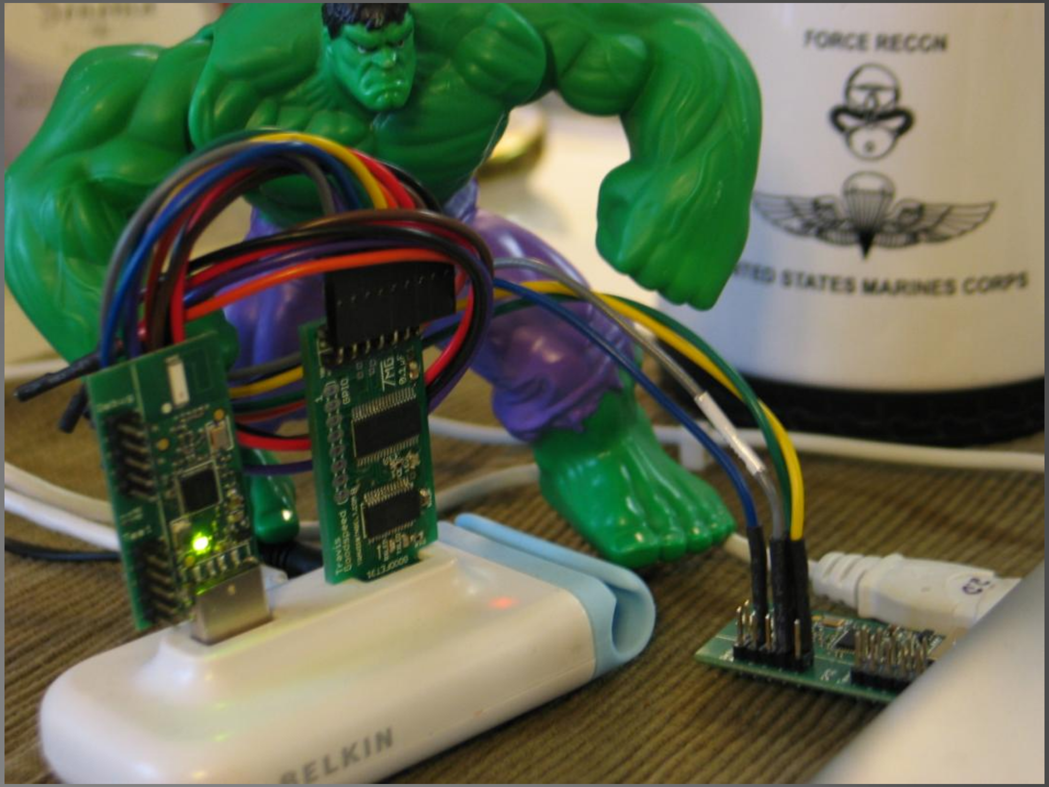
SmartRF® Studio 7 - Texas Instruments
SmartRF® Studio 7
1.0.3
Sub 1 GHz ISM band
2.4 GHz (1 Connected)
CC2530 2.4 GHz System-on-Chip
CC2531 2.4 GHz USB System-on-Chip
CC2430 2.4 GHz System-on-Chip
CC2431 2.4 GHz LOC System-on-Chip
CC2520 2.4 GHz Transceiver
CC2500 2.4 GHz Transceiver
CC2510 2.4 GHz System-on-Chip
CC2511 2.4 GHz USB System-on-Chip
CC2550 2.4 GHz Transmitter
1 Connected device(s)
Texas Instruments

List of connected devices:

SmartRF05EB (USB device ID=2770, Firmware revision=0013), -- CC2530

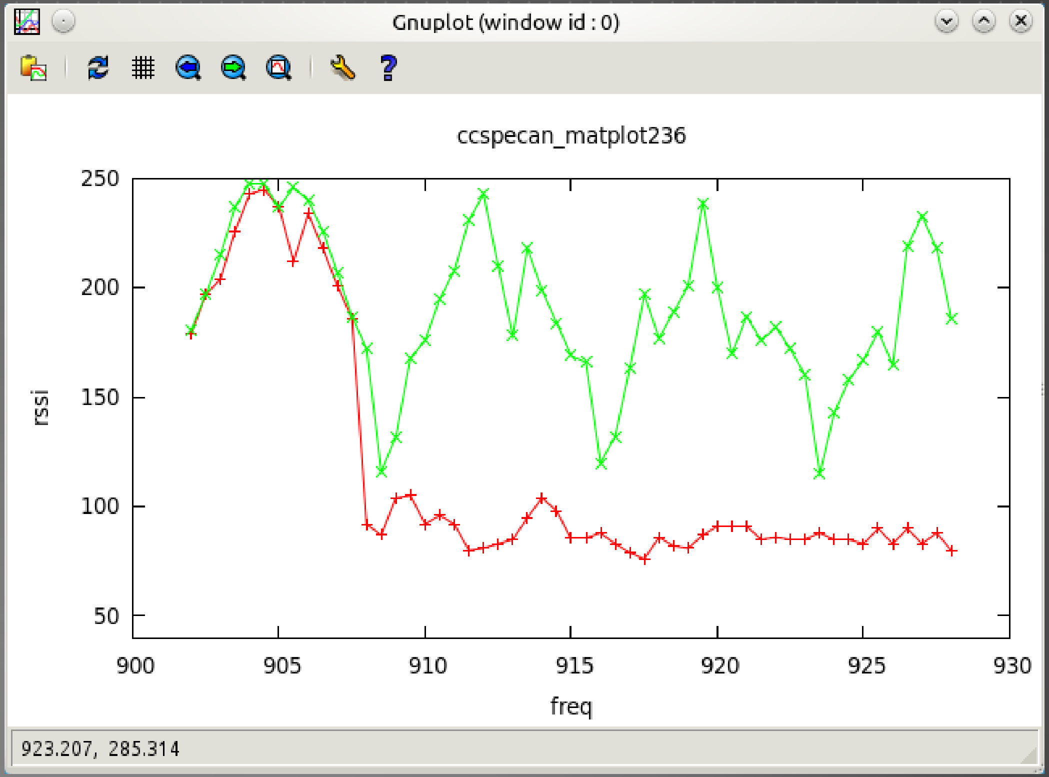
H4RDW4RE

- Code Advances
  - Prerequisites
    - 902 - 928 MHz Range
    - Fast scanning
  - Strip Specan Firmware Code
    - Remove display (but compensate for delay)
    - Shrink frequency range
  - Leverage Goodfet
    - Data Debug dumping
    - Python scripts for displaying data
    - Halting CPU – affects results



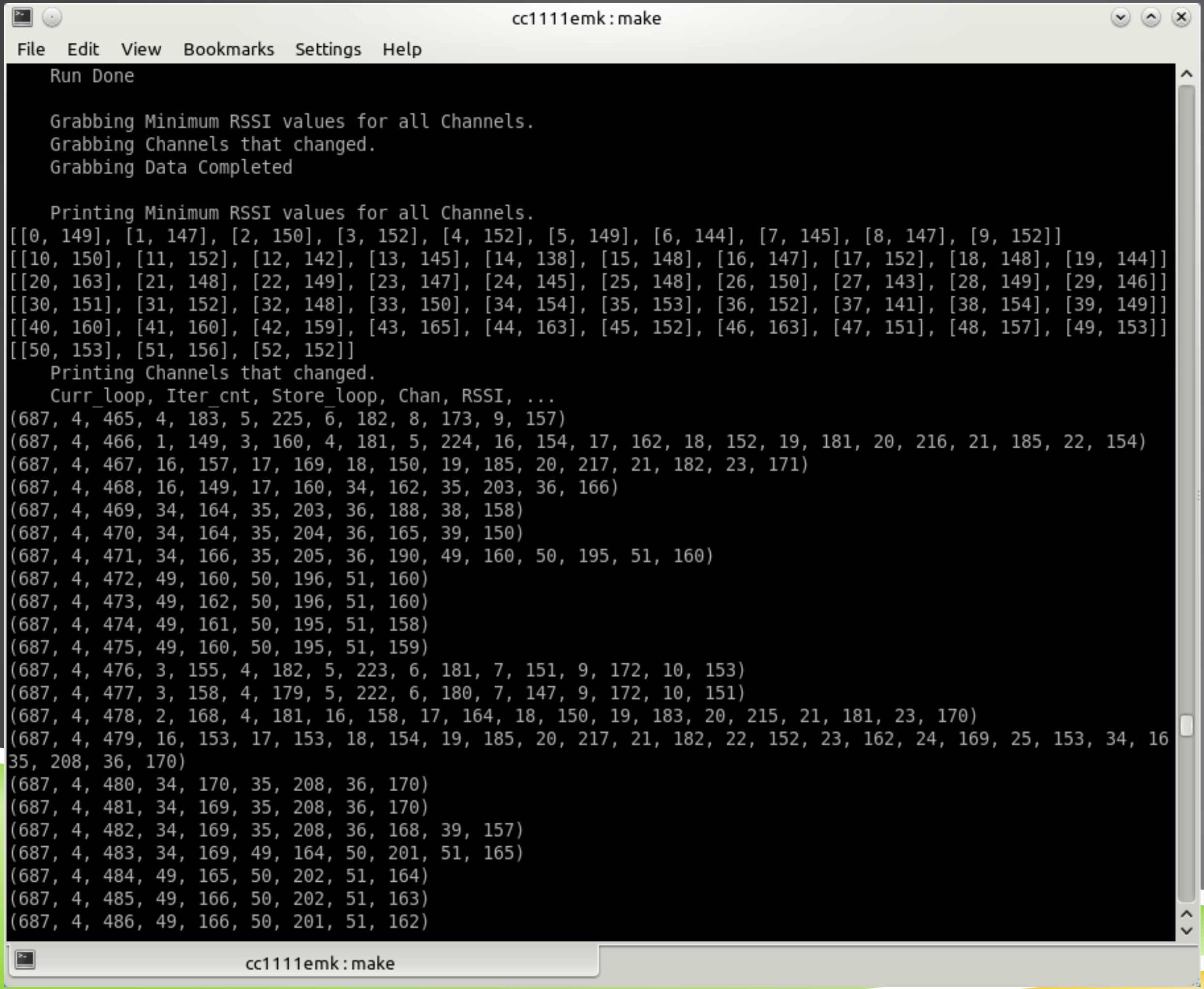
H4RDW4RE

- Resulting Firmware
  - maxscan – spectrum analyzer
  - hoptrans – create a carrier wave
    - Number of channels is known
    - Channel spacing is known
    - Hop timing is known



H4RDW4RE

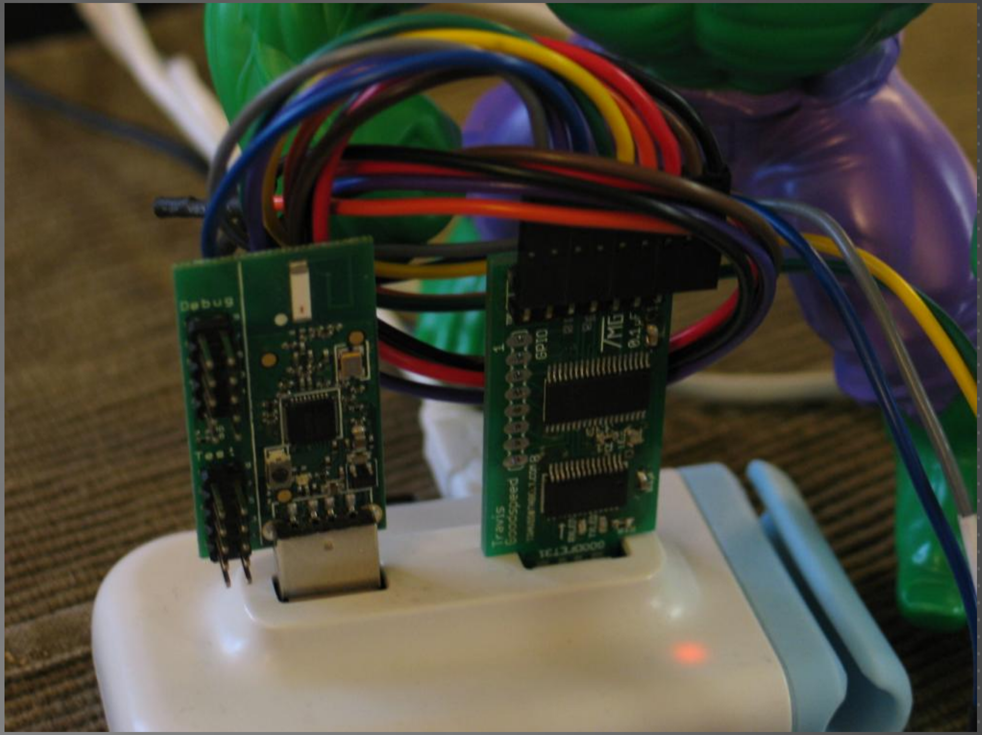
- Resulting Firmware (2)
  - Minscan – detects channel hops
    - Initializes frequencies
    - Scans frequencies for minimum RSSI
    - Monitors jumps in RSSI
    - Stores detected spikes
    - Data dump via Goodfet
    - Data is analyzed offline



```
    Run Done

    Grabbing Minimum RSSI values for all Channels.
    Grabbing Channels that changed.
    Grabbing Data Completed

    Printing Minimum RSSI values for all Channels.
[[0, 149], [1, 147], [2, 150], [3, 152], [4, 152], [5, 149], [6, 144], [7, 145], [8, 147], [9, 152]]
[[10, 150], [11, 152], [12, 142], [13, 145], [14, 138], [15, 148], [16, 147], [17, 152], [18, 148], [19, 144]]
[[20, 163], [21, 148], [22, 149], [23, 147], [24, 145], [25, 148], [26, 150], [27, 143], [28, 149], [29, 146]]
[[30, 151], [31, 152], [32, 148], [33, 150], [34, 154], [35, 153], [36, 152], [37, 141], [38, 154], [39, 149]]
[[40, 160], [41, 160], [42, 159], [43, 165], [44, 163], [45, 152], [46, 163], [47, 151], [48, 157], [49, 153]]
[[50, 153], [51, 156], [52, 152]]
    Printing Channels that changed.
    Curr_loop, Iter_cnt, Store_loop, Chan, RSSI, ...
(687, 4, 465, 4, 183, 5, 225, 6, 182, 8, 173, 9, 157)
(687, 4, 466, 1, 149, 3, 160, 4, 181, 5, 224, 16, 154, 17, 162, 18, 152, 19, 181, 20, 216, 21, 185, 22, 154)
(687, 4, 467, 16, 157, 17, 169, 18, 150, 19, 185, 20, 217, 21, 182, 23, 171)
(687, 4, 468, 16, 149, 17, 160, 34, 162, 35, 203, 36, 166)
(687, 4, 469, 34, 164, 35, 203, 36, 188, 38, 158)
(687, 4, 470, 34, 164, 35, 204, 36, 165, 39, 150)
(687, 4, 471, 34, 166, 35, 205, 36, 190, 49, 160, 50, 195, 51, 160)
(687, 4, 472, 49, 160, 50, 196, 51, 160)
(687, 4, 473, 49, 162, 50, 196, 51, 160)
(687, 4, 474, 49, 161, 50, 195, 51, 158)
(687, 4, 475, 49, 160, 50, 195, 51, 159)
(687, 4, 476, 3, 155, 4, 182, 5, 223, 6, 181, 7, 151, 9, 172, 10, 153)
(687, 4, 477, 3, 158, 4, 179, 5, 222, 6, 180, 7, 147, 9, 172, 10, 151)
(687, 4, 478, 2, 168, 4, 181, 16, 158, 17, 164, 18, 150, 19, 183, 20, 215, 21, 181, 23, 170)
(687, 4, 479, 16, 153, 17, 153, 18, 154, 19, 185, 20, 217, 21, 182, 22, 152, 23, 162, 24, 169, 25, 153, 34, 16
35, 208, 36, 170)
(687, 4, 480, 34, 170, 35, 208, 36, 170)
(687, 4, 481, 34, 169, 35, 208, 36, 170)
(687, 4, 482, 34, 169, 35, 208, 36, 168, 39, 157)
(687, 4, 483, 34, 169, 49, 164, 50, 201, 51, 165)
(687, 4, 484, 49, 165, 50, 202, 51, 164)
(687, 4, 485, 49, 166, 50, 202, 51, 163)
(687, 4, 486, 49, 166, 50, 201, 51, 162)
```

H4RDW4RE

- Issues
  - Data Debug dump via Goodfet is slow
  - Pausing CPU creates gaps in monitoring
  - Memory on CC1111EMK is too small
  - UART would work, but USB would be better
  - CC1111EMK can do USB but must be managed via firmware

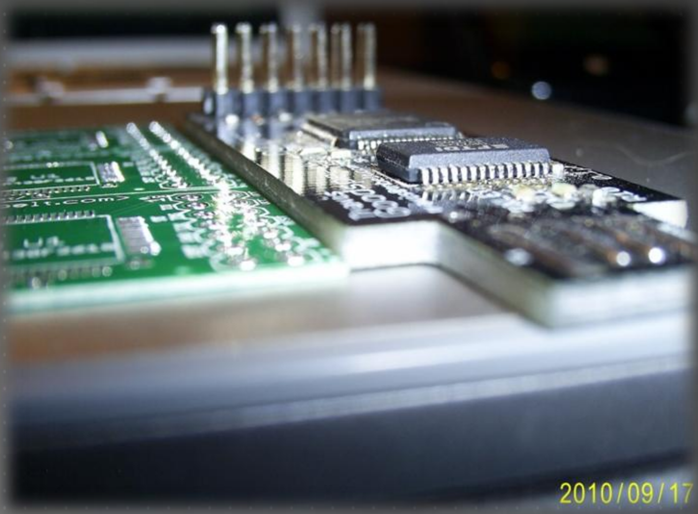


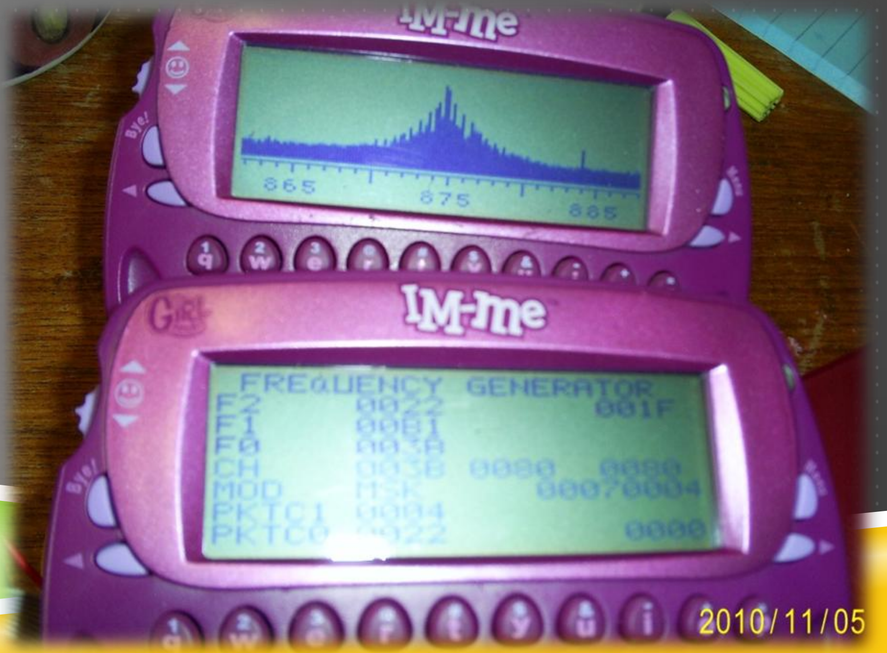
H4RDW4RE

- Lessons Learned
  - Hardware is HARD
    - When you start
    - You WILL fail, drive on
  - Hardware documentation is CONFUSING
    - You get better with time
    - The documentation DOES NOT
  - Radios are complex
    - Hardware radios ARE complicated
    - SmartRF HELPS
    - IM-ME code is an EXCELLENT example
  - We CAN see good results in test data
    - This IS step in a right direction

C0DE

- Overview
  - CC1111
  - Merging Code
  - Screen Shots of Code Running
  - GNU Plot
  - Stages of Analysis
  - Goals of Code
  - Where we are Currently





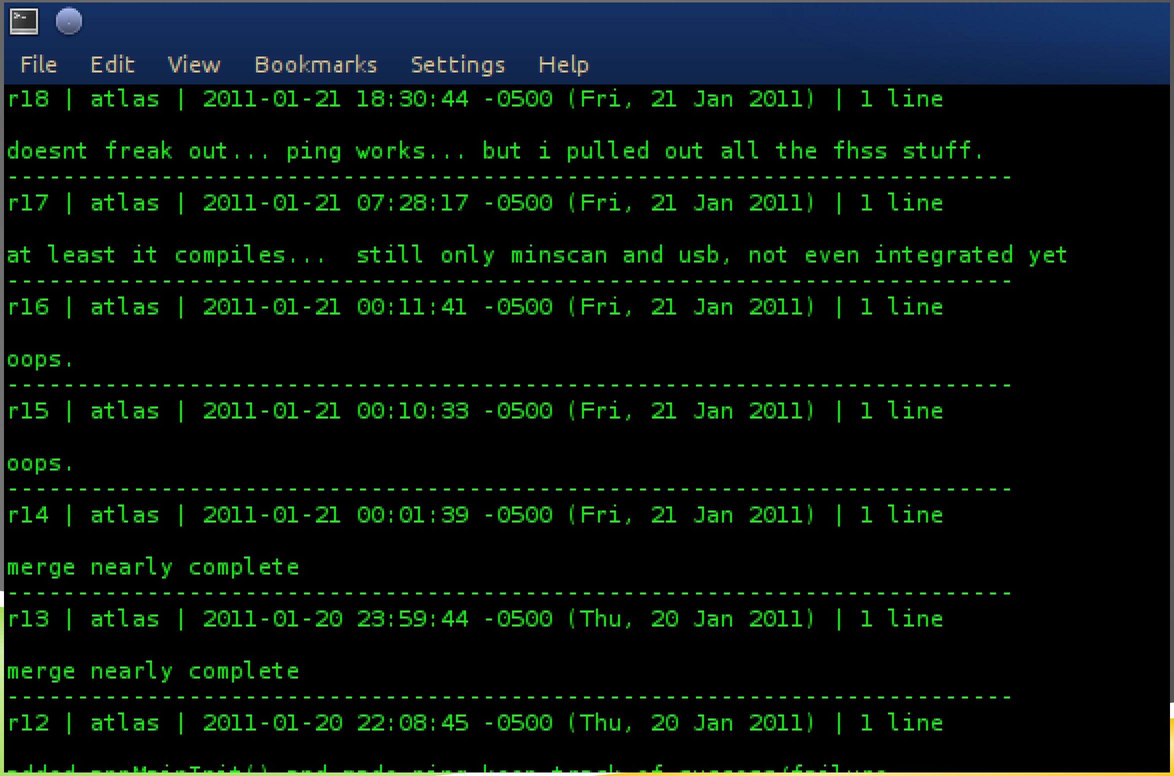
```
FREQUENCY GENERATOR
F2     0022           001F
F1     00B1
F0     AA3A
CH     0038   0000    0000
MOD    MSK        00070004
PKTC1  0004
PKTC0  0022           0000
```

CC1111

- USB-enabled version of TI’s most popular <1GHz radio/mcu
- Same radio used in the majority of today’s Smart Meters
- “easy” usb! Yeah right. YOU MUST READ “USB IN A NUTSHELL”
- Separate cc1111usb project soon to be released
  - GPL
  - Hacker-written (good or bad?)
  - Contained in one .c and one .h

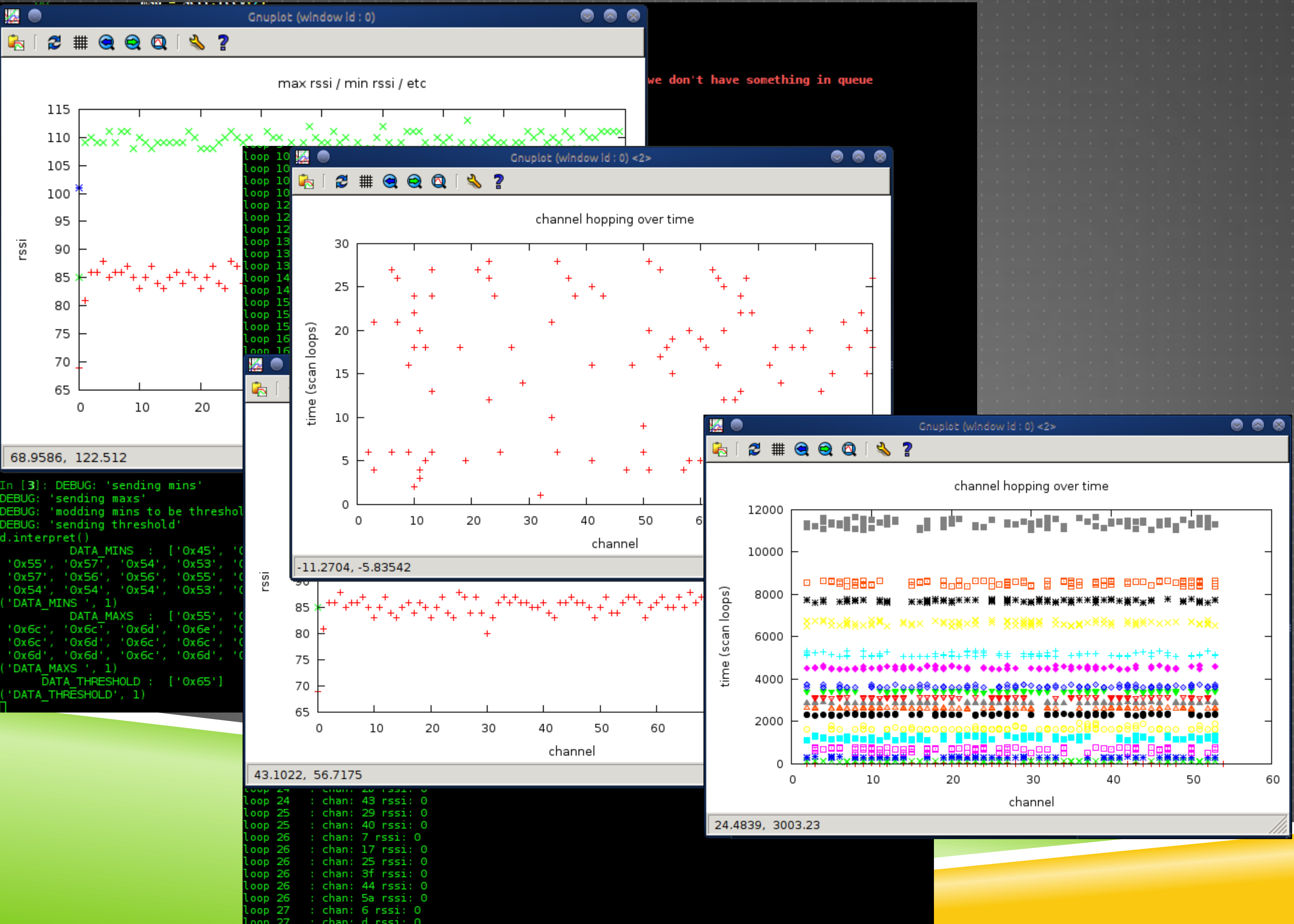
MERGING CODE

- “Messing up code”
- Still buggy, although somewhat less thrashing
- USB code was *really* a mess when we started



```
r18 | atlas | 2011-01-21 18:30:44 -0500 (Fri, 21 Jan 2011) | 1 line

doesnt freak out... ping works... but i pulled out all the fhss stuff.
------------------------------------------------------------------------
r17 | atlas | 2011-01-21 07:28:17 -0500 (Fri, 21 Jan 2011) | 1 line

at least it compiles...  still only minscan and usb, not even integrated yet
------------------------------------------------------------------------
r16 | atlas | 2011-01-21 00:11:41 -0500 (Fri, 21 Jan 2011) | 1 line

oops.
------------------------------------------------------------------------
r15 | atlas | 2011-01-21 00:10:33 -0500 (Fri, 21 Jan 2011) | 1 line

oops.
------------------------------------------------------------------------
r14 | atlas | 2011-01-21 00:01:39 -0500 (Fri, 21 Jan 2011) | 1 line

merge nearly complete
------------------------------------------------------------------------
r13 | atlas | 2011-01-20 23:59:44 -0500 (Thu, 20 Jan 2011) | 1 line

merge nearly complete
------------------------------------------------------------------------
r12 | atlas | 2011-01-20 22:08:45 -0500 (Thu, 20 Jan 2011) | 1 line

added proMainInit() and made ping keep track of success/failure
```

SCREEN SHOTS OF CODE RUNNING



```
we don't have something in queue
```

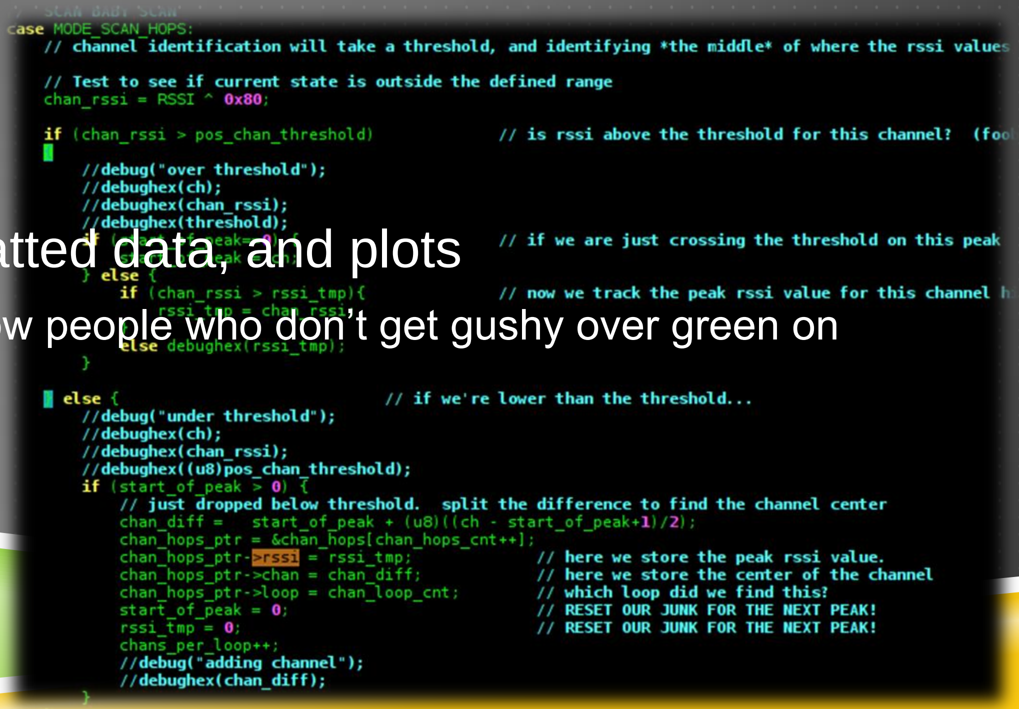
```
loop 10
loop 10
loop 10
loop 10
loop 12
loop 12
loop 12
loop 13
loop 13
loop 13
loop 14
loop 14
loop 15
loop 15
loop 15
loop 16
loop 16
```

```
In [3]: DEBUG: 'sending mins'
DEBUG: 'sending maxs'
DEBUG: 'modding mins to be threshol
DEBUG: 'sending threshold'
d.interpret()
          DATA_MINS  :  ['0x45', '0
 '0x55', '0x57', '0x54', '0x53', '0
 '0x57', '0x56', '0x56', '0x55', '0
 '0x54', '0x54', '0x54', '0x53', '0
('DATA_MINS ', 1)
          DATA_MAXS  :  ['0x55', '0
 '0x6c', '0x6c', '0x6d', '0x6e', '0
 '0x6c', '0x6d', '0x6c', '0x6c', '0
 '0x6d', '0x6d', '0x6c', '0x6d', '0
('DATA_MAXS ', 1)
      DATA_THRESHOLD :  ['0x65']
('DATA_THRESHOLD', 1)
```

```
loop 24   : chan: 43 rssi: 0
loop 25   : chan: 29 rssi: 0
loop 25   : chan: 40 rssi: 0
loop 26   : chan: 7 rssi: 0
loop 26   : chan: 17 rssi: 0
loop 26   : chan: 25 rssi: 0
loop 26   : chan: 3f rssi: 0
loop 26   : chan: 44 rssi: 0
loop 26   : chan: 5a rssi: 0
loop 27   : chan: 6 rssi: 0
loop 27   : chan: d rssi: 0
```

INTRO TO CODE

- Python Client using libusb (requires admin access)
- d.dump*() – grab large chunks of data from dongle
- d.get*() – grab incidental settings
- d.set*() – set incidental settings (threshold, channel info)
- d.doFloorAndCeiling()
- d.doChannelIdent()
- d.doHopTracking()
- d.interpret() – spits out formatted data, and plots
  - GNUplot - Really fun way to wow people who don’t get gushy over green on black…



```
    // SCAN BABY SCAN
case MODE_SCAN_HOPS:
    // channel identification will take a threshold, and identifying *the middle* of where the rssi values

    // Test to see if current state is outside the defined range
    chan_rssi = RSSI ^ 0x80;

    if (chan_rssi > pos_chan_threshold)             // is rssi above the threshold for this channel?  (foo
    {
        //debug("over threshold");
        //debughex(ch);
        //debughex(chan_rssi);
        //debughex(threshold);
        if (start_of_peak==0) {                     // if we are just crossing the threshold on this peak
            start_of_peak = ch;
        } else {
            if (chan_rssi > rssi_tmp){              // now we track the peak rssi value for this channel h
                rssi_tmp = chan_rssi;
            }
            else debughex(rssi_tmp);
        }

    } else {                            // if we're lower than the threshold...
        //debug("under threshold");
        //debughex(ch);
        //debughex(chan_rssi);
        //debughex((u8)pos_chan_threshold);
        if (start_of_peak > 0) {
            // just dropped below threshold.  split the difference to find the channel center
            chan_diff =   start_of_peak + (u8)((ch - start_of_peak+1)/2);
            chan_hops_ptr = &chan_hops[chan_hops_cnt++];
            chan_hops_ptr->rssi = rssi_tmp;             // here we store the peak rssi value.
            chan_hops_ptr->chan = chan_diff;            // here we store the center of the channel
            chan_hops_ptr->loop = chan_loop_cnt;        // which loop did we find this?
            start_of_peak = 0;                          // RESET OUR JUNK FOR THE NEXT PEAK!
            rssi_tmp = 0;                               // RESET OUR JUNK FOR THE NEXT PEAK!
            chans_per_loop++;
            //debug("adding channel");
            //debughex(chan_diff);
        }
```

STEPS OF ANALYSIS

- Floor and Ceiling – mins and maxes
- Channel threshold – how do we know when we’ve found a channel?
- Channel Identification and Spacing – currently broke
- Hopping pattern – this is the shizzle
  - Predictive Analysis (future)
- Sync Word – radios know when data is coming (future, easy)
- Data Rate – duh, like a modem (future, limited possibilities, often documented)

GOALS OF CODE

- Dispell myths of FHSS security (obviously)
- Weaponization and Automation
- Network Sniffing
- Network Participation
- Reduce time for real security testing

WHERE WE'RE AT

12/07/2010

- USB-controlled radio – great platform to play
- Calibration and Speed
- Channel identification – broken but close
- Hopping identification – some bugs in data storage and dumping
- GNUplotting for hollywood enjoyment (hey hollywood)

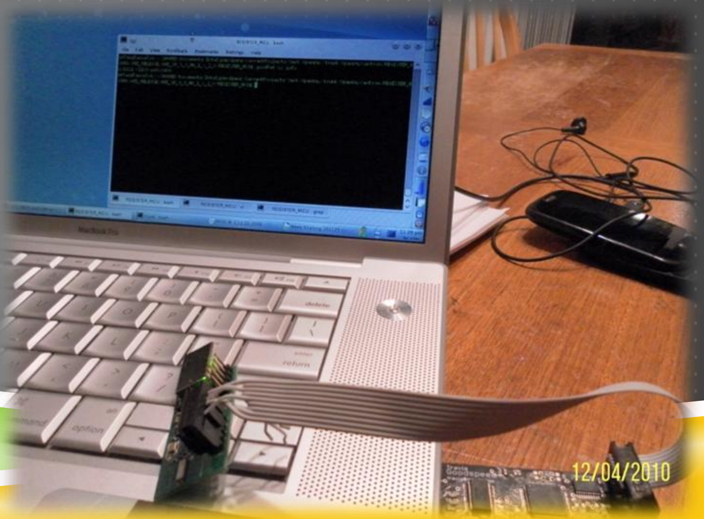
- Still need to analyze and coalesce final data better
- Want to port it to the CC1110 of the IM-ME Dongle

12/07/2010

11/30/2010

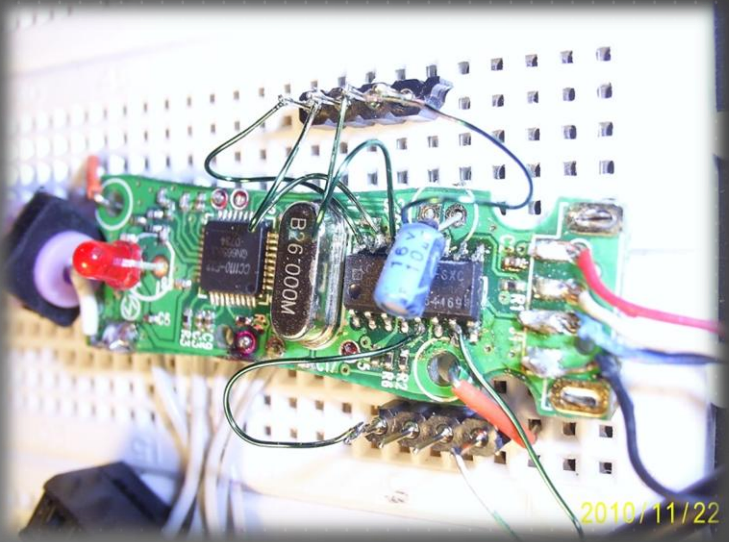
ACKNOWLEDGEMENTS

- Travis Goodspeed
  - Not only is he a master of the belt buckle but a supreme commander of all things Good and FET
  - Check out the GoodFET project
    - http://goodfet.sourceforge.net/
- “Michael Ossmann Is My Hero”
  - He has T-Shirts that say so …
  - Creator of neighborly spectrum analysis software
    - Developed at the last SHMOOCON!
    - http://www.ossmann.com/
- Bill Gates
- Brett and Jemain, and Bagettes



PROJECT INFORMATION

- Google Code Page
  - http://code.google.com/p/hedyattack/
- Member Information
  - atlas
    - atlas@r4780y.com
  - Don C. Weber
    - don@inguardians.com
  - Andrew Righter
    - andrew@inguardians.com



QUESTIONS & ANSWERS

- if time_remaining != 0:
  - answer_questions( );
- else:
  - print(“sorry!”);