

## ***Software Defined Radio Server***

“A Radio Server for VHF+ Contesting  
And Weak Signal Work”

Phil Theis K3TUF

Digital Communications Conference  
October 10, 2015

## ***Initial Plans***

- Need Band Data
- Switch Transverters
- 6700 is Great Radio (#1 on Sherwood Engineering List)
- No way to change uW bands
- Of HF bands for that matter

## *Put an Embedded Device to work*

- Select Device
- Use Rapid Development Tools
  - Python
- Get on the air
- End of Story ?

## *Python in Action*



## ***Elegance and Simplicity***

- Integrated Development Environment
- Built In – Off the Shelf
  - Beagle Bone Black
  - Immediate Bone Script
  - Python
  - Ethernet or USB

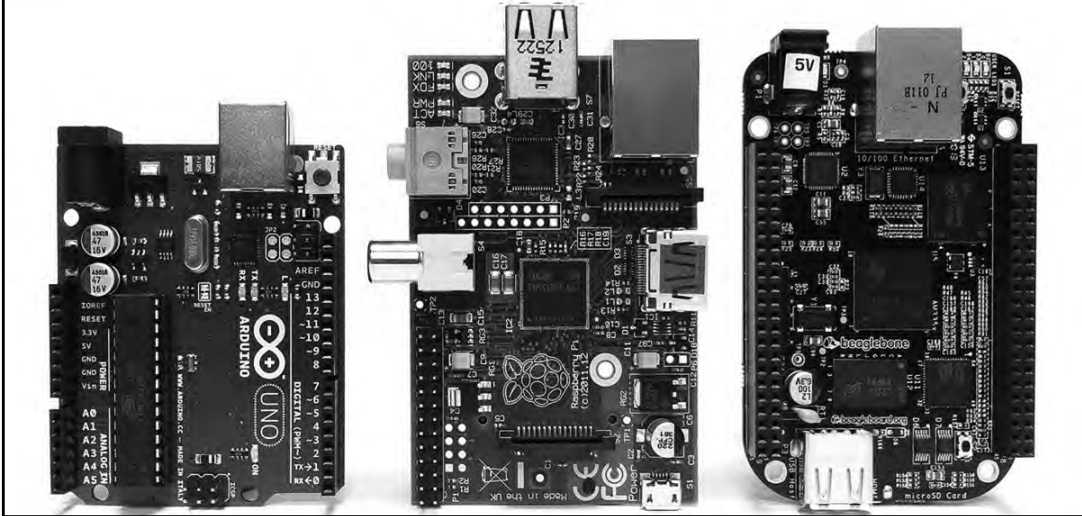
*October 2014*

## ***Talk Today***

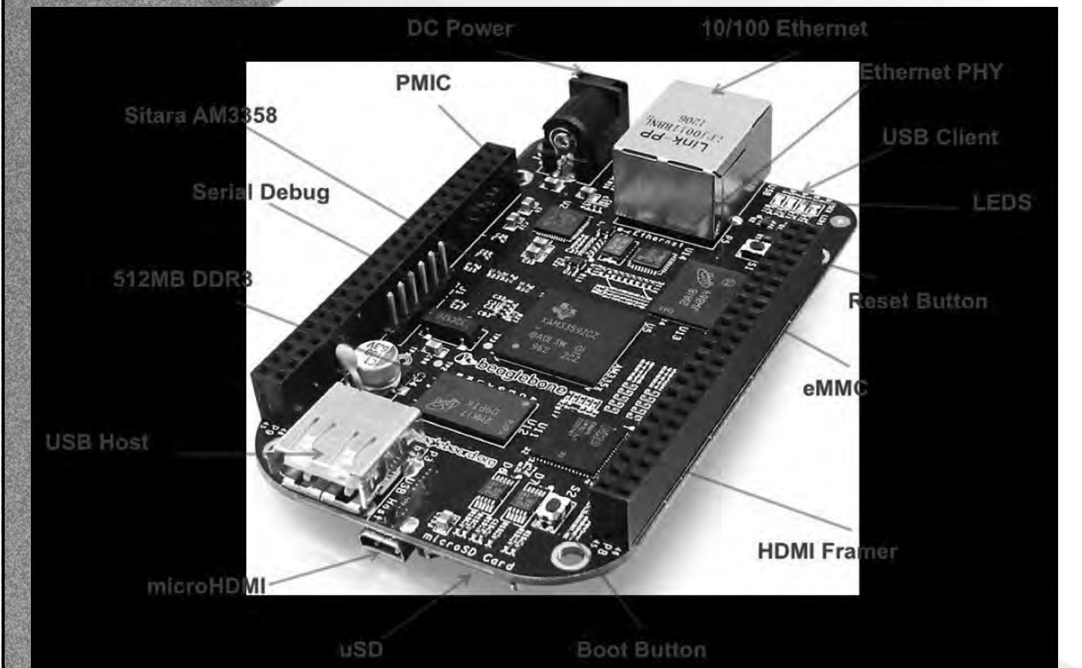
- Take you through the Process
- See what I learned along the way
- Much more that can happen
  - Transverter Control
  - Remote Control of 6K radios
  - Tasks around the Shack
- All Via Ethernet

## Device Choices

- Arduino – Raspberry PI – Beagle Bone



## Beagle Bone Black

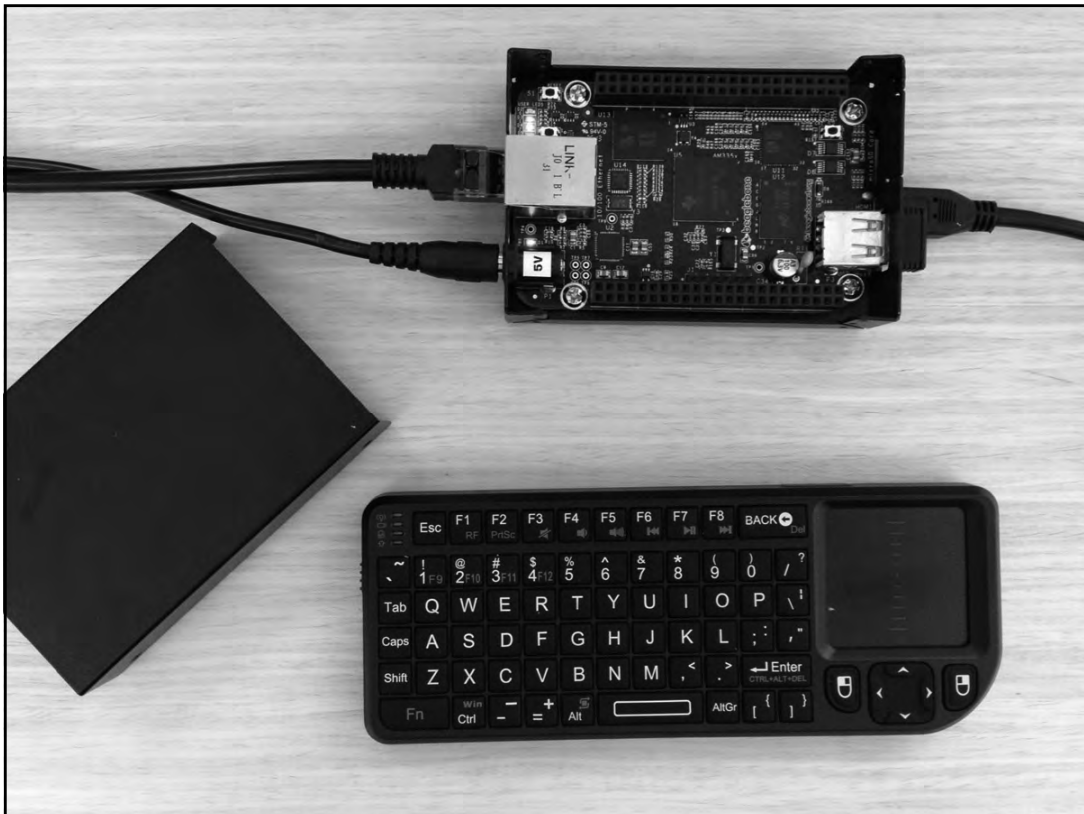


# GPIO pins

## 65 possible digital I/Os

P9				P8			
DGND	1	2	DGND	DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3	GPIO_38	3	4	GPIO_39
VDD_5V	5	6	VDD_5V	GPIO_34	5	6	GPIO_35
SYS_5V	7	8	SYS_5V	GPIO_66	7	8	GPIO_67
PWR_BUTTON	9	10	SYS_RESETN	GPIO_69	9	10	GPIO_68
GPIO_30	11	12	GPIO_60	GPIO_45	11	12	GPIO_44
GPIO_31	13	14	GPIO_40	GPIO_23	13	14	GPIO_26
GPIO_48	15	16	GPIO_51	GPIO_47	15	16	GPIO_46
GPIO_4	17	18	GPIO_5	GPIO_27	17	18	GPIO_65
I2C2_SCL	19	20	I2C2_SDA	GPIO_22	19	20	GPIO_63
GPIO_3	21	22	GPIO_2	GPIO_62	21	22	GPIO_37
GPIO_49	23	24	GPIO_15	GPIO_36	23	24	GPIO_33
GPIO_117	25	26	GPIO_14	GPIO_32	25	26	GPIO_61
GPIO_125	27	28	GPIO_123	GPIO_86	27	28	GPIO_88
GPIO_121	29	30	GPIO_122	GPIO_87	29	30	GPIO_89
GPIO_120	31	32	VDD_ADC	GPIO_10	31	32	GPIO_11
AIN4	33	34	GNDA_ADC	GPIO_9	33	34	GPIO_81
AIN6	35	36	AIN5	GPIO_8	35	36	GPIO_80
AIN2	37	38	AIN3	GPIO_78	37	38	GPIO_79
AIN0	39	40	AIN1	GPIO_76	39	40	GPIO_77
GPIO_20	41	42	GPIO_7	GPIO_74	41	42	GPIO_75
DGND	43	44	DGND	GPIO_72	43	44	GPIO_73
DGND	45	46	DGND	GPIO_70	45	46	GPIO_71

In GPIO mode, each digital I/O can produce interrupts.



## *Apache Web Server*



- Port 80
- PHP
- Available to any Device



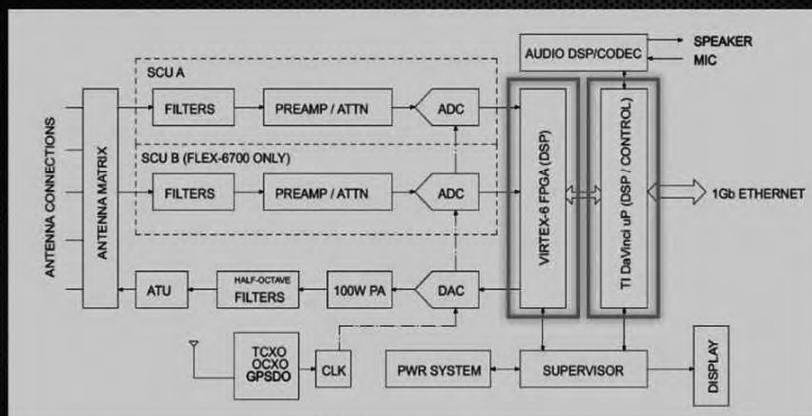
## *The Radio Server*



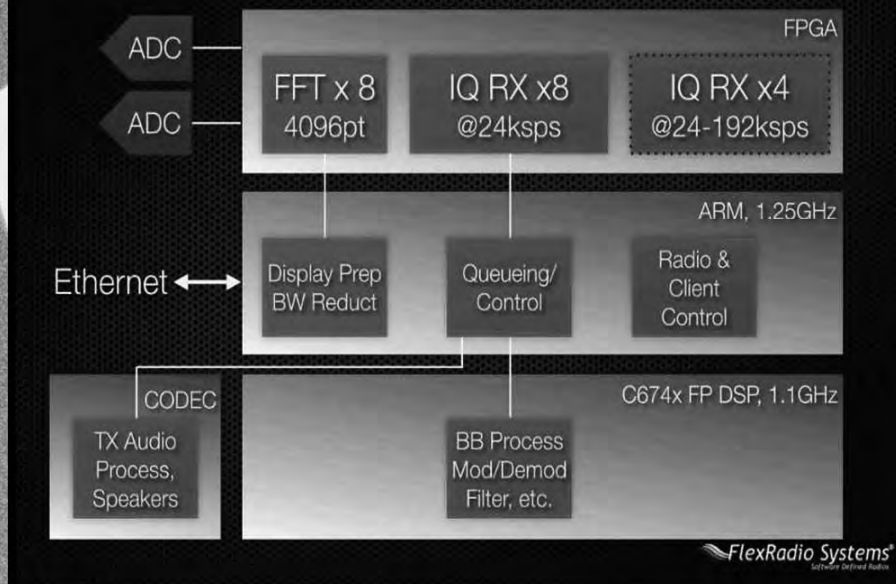
## FLEX-6000 Signature Series Family

	Pan /RX	Max BW	DAXIQ	SCU	ATU	GPSDO	Mic	Freq
FLEX-6300	2	7 MHz	2 up to 96kHz	1	Opt	—	Unbal	160—6m
FLEX-6500	4	14 MHz	4 up to 192kHz	1	YES	Opt	Unbal + Bal	160—4m
FLEX-6700	8	14 MHz	4 up to 192kHz	2	YES	Opt	Unbal + Bal	160—2m
FLEX-6700R	8	14 MHz	4 up to 192kHz	2	N/A	Opt	N/A	160—2m

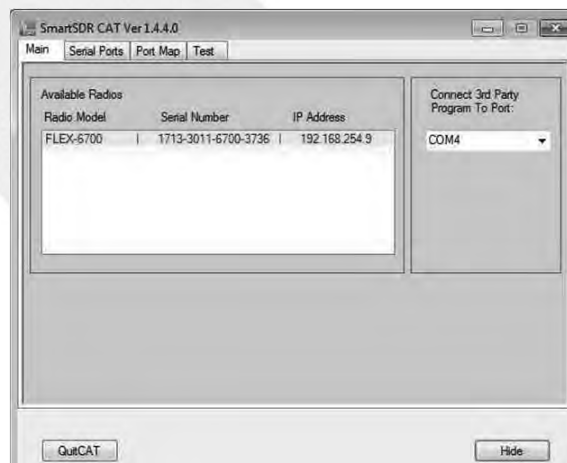
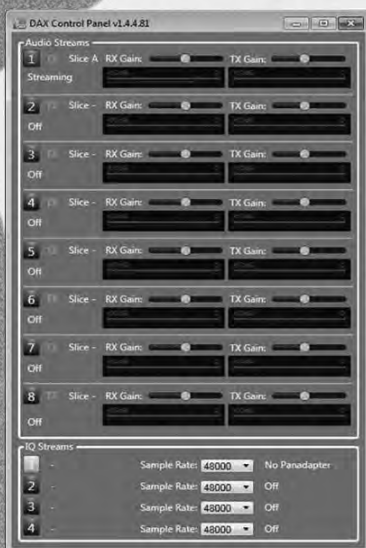
## FLEX-6000 HW System Architecture



# SmartSDR SW Architecture



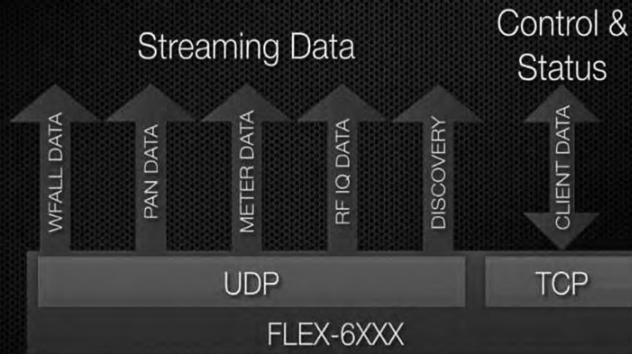
## DAX & SmartCAT





# Talking to the Radio Server

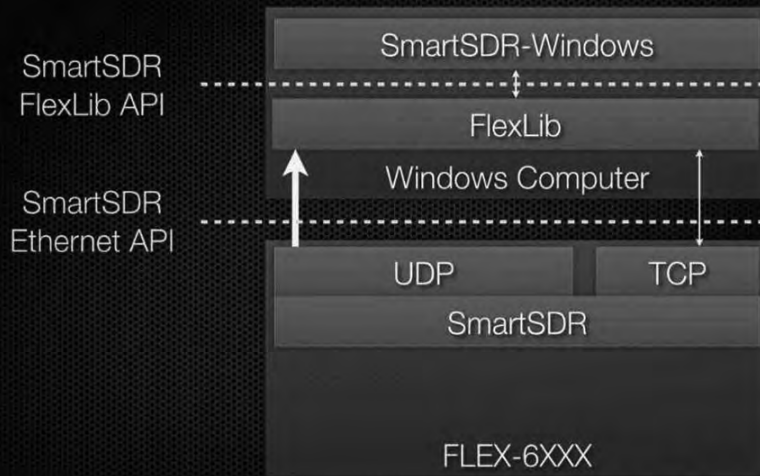
## SmartSDR Ethernet API Interfaces



FlexRadio Systems®  
SOFTWARE DEFINED RADIO

# SmartSDR and the use of FlexLib

## SmartSDR APIs



FlexRadio Systems®  
SOFTWARE DEFINED RADIO

## ***Flex Uses the API***

- SmartSDR Windows client rests on FlexLib which rests on the internet API
- CAT and DAX also use FlexLib
- You can do anything done in SmartSDR
- Unprecedented control over a Radio Server

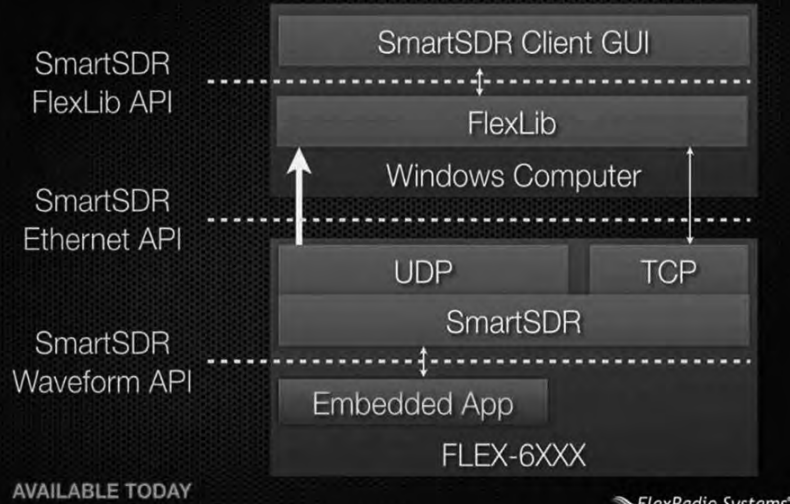
## ***FlexLib***

### **FlexLib - SmartSDR in .NET**

- ▶ FlexLib is a .NET 4.0 DLL that provides .NET style access to the SmartSDR Internet API
- ▶ Simplifies interoperation with the radio in .NET environment - Object Oriented, Events, etc.
- ▶ Provided at no charge on the FlexRadio website

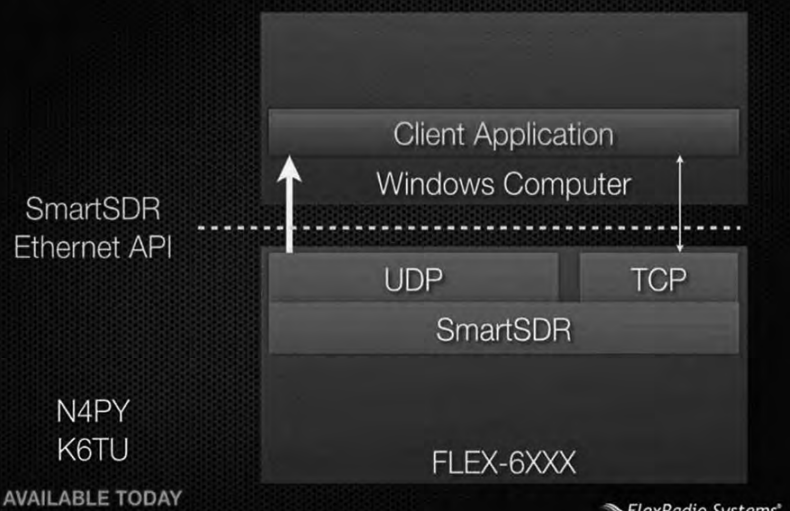
## Installing App in Radio

### 3rd Party Embedded App



## What I am doing

### 3rd Party App using Ethernet API



## **API Objectives**

### SmartSDR API Objectives

- ▶ Provide a common interface for FlexRadio products
- ▶ Support the building of an ecosystem around SmartSDR for the benefit of customers, developers and FlexRadio
- ▶ Provide a way to use a FLEX-6000 in a variety of applications, even ones that may not be mainstream

 FlexRadio Systems<sup>®</sup>  
Software Defined Radio

## **How to talk to the API**

### API Standards

- ▶ Radio control is a TCP/IP socket with simple commands (no standard known):  

```
slice create freq=14.1 ant=ANT1 mode=USB  
slice tune 0 14.105
```
- ▶ Streaming Panadapter/Waterfall/Meter/Discovery data are VITA-49 Extension
- ▶ I/Q and Real IF is VITA-49 IF Data (24-192ksps)

 FlexRadio Systems<sup>®</sup>  
Software Defined Radio

## API Commands

### SmartSDR TCP/UDP API Command Format

- ▶ Command preface, sequence, v-bar, command  
`C134|slice create freq=7.243`
- ▶ Response preface, sequence, v-bar, response  
`R134|50000002`
- ▶ Status preface, handle, v-bar, status  
`S67EF9A22|slice 0 freq=7.243`  
`S67EF9A22|slice 0 filter_lo=300 filter_hi=2700`

 FlexRadio Systems  
Software Defined Radio

## Establishing Connection

### SmartSDR TCP/UDP API Connecting to radio

- ▶ TCP/IP socket connection to port 4992
- ▶ API provides API version and a “handle”  
`V1.1.0.0`  
`H35E61405`
- ▶ Send commands!
- ▶ Interface is asynchronous, commands are non-blocking

 FlexRadio Systems  
Software Defined Radio

## Slice Exchange

### Slice Receivers, example

- ▶ Create a slice receiver

```
slice create [freq=<MHz>] [ant=<antenna>] [mode=<mode>]  
C34|slice create freq=14.236 mode=FDV  
R34|0
```

- ▶ Tune a slice receiver

```
slice tune 0 [freq=<MHz>] [ant=<antenna>] [mode=<mode>]  
C45|slice 0 freq=14.236
```

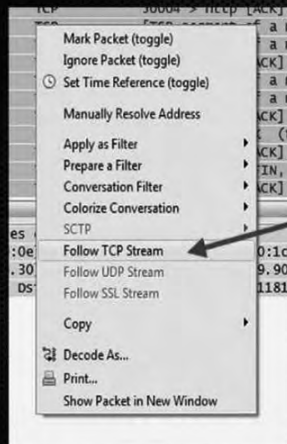
- ▶ Change slice receiver settings

```
slice set <slice> [<feature>=<value>]  
C71|slice set 0 diversity=1 tx=0 record=1  
R71|0
```

FlexRadio Systems®  
Software Defined Radio

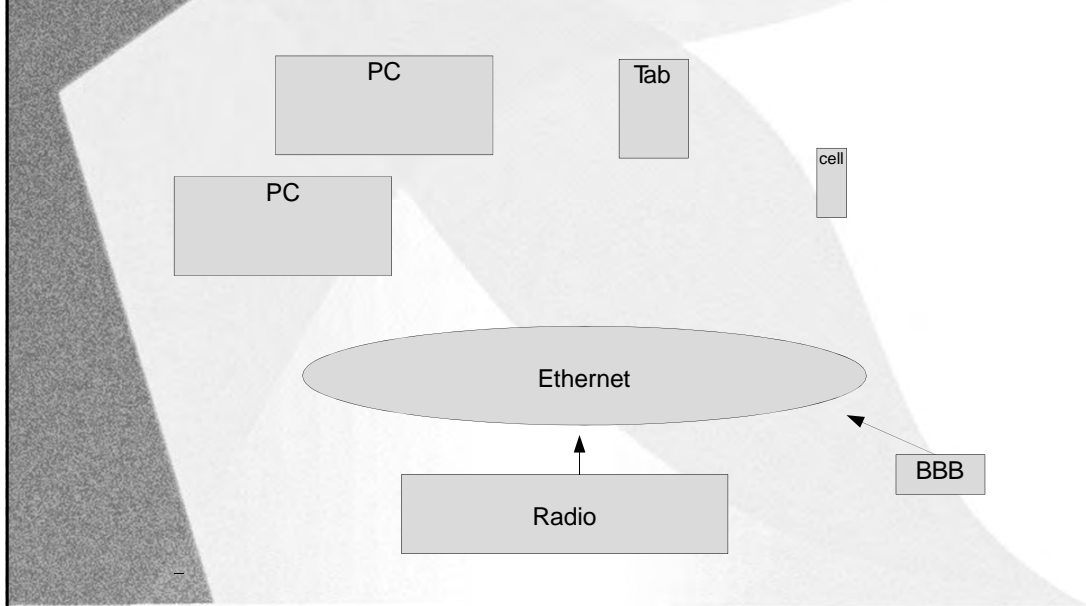
## Learning the Protocol

### Sniffing TCP/IP API Using Wireshark



FlexRadio Systems®  
Software Defined Radio

## *My Port 80 Plan*



## *Technology: Languages*

- HTML Hyper Text Markup Language
- AJAX Asynchronous JavaScript and XML
- DOM The Document Object Model is a platform and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents
- Apache / PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language

## Technology: Languages

- C Programming Language for the server
- JavaScript is a dynamic computer programming language. It is most commonly used as part of Web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed
- JSON JavaScript Object Notation
- Python for early proof of concept

## Eclipse Development Environment

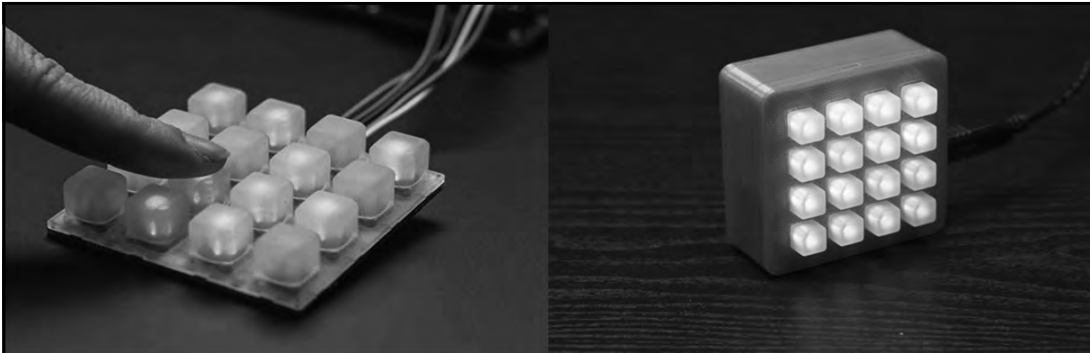
The screenshot displays the Eclipse IDE interface. The main editor window shows a C program with a switch statement. The code is as follows:

```
194     else printf("%s\n", "we now have 3456");
195     currentband = 3456;
196     sendband('6');
197     break;
198
199     case 5768 :
200     if (currentband == 5768) break;
201     else printf("%s\n", "we now have 5768");
202     currentband = 5768;
203     sendband('8');
204     break;
205
206     case 10368 :
207     if (currentband == 10368) break;
208     else printf("%s\n", "we now have 10368");
209     currentband = 10368;
210     sendband('a');
211     break;
212
213     case 24192 :
214     if (currentband == 24192) break;
215     else printf("%s\n", "we now have 24192");
216     currentband = 24192;
217     sendband('b');
218     break;
219
220     case 47088 :
221     if (currentband == 47088) break;
222     else printf("%s\n", "we now have 47088");
223     currentband = 47088;
224     sendband('c');
225     break;
226
227     default :
228     if (currentband == 28) break; // 28 represents all of HP for now
229     else printf("%s\n", "must be HP");
230     currentband = 28;
231     sendband('9'); //No kramavsecc
232
233 }
234
```

The terminal window at the bottom shows the following output:

```
Last login: Mon Apr 13 02:22:10 2015 from 192.168.254.18
root@e1488-1:~# cd /usr/src
root@e1488-1:~/usr/src# ls
0970.h  bndchg.c  file1  filetest1.c  gpio.c  gpiodef1.c  makeLED.c  patternmatchpointers  patternmatchtest.c  sockettest2.c  sockettest4  strtoctest
bndchg  devmem2.c  file2  filetest2.c  gpio.h  i2c  makeLEDs  patternmatchpointers.c  sockettest  sockettest3  sockettest4.c  strtoctest.c
bndchg1.c  devmem2.c  filetest1  filetest2.c  gpiodef1.c  makeLED  makeLEDs.c  patternmatchtest  sockettest2  sockettest3.c  sockettest5.c
root@e1488-1:~/usr/src#
```





- Instantaneous Re-Configuration
- Liaison to Run
- Split Audio
- No Loss of Focus
- Complete Control of Radio
- LED Feedback

## Future Tasks

- Monitor Temperatures
- Control Power Supplies
- Turn Antennas / Switch Antennas
- Multiple Locations with Distributed Computing
- Beacon Monitoring: Propagation Notification
- Performance of Beacons: Real Time Status
- Dayton Demonstration