

Software Defined Radio Server

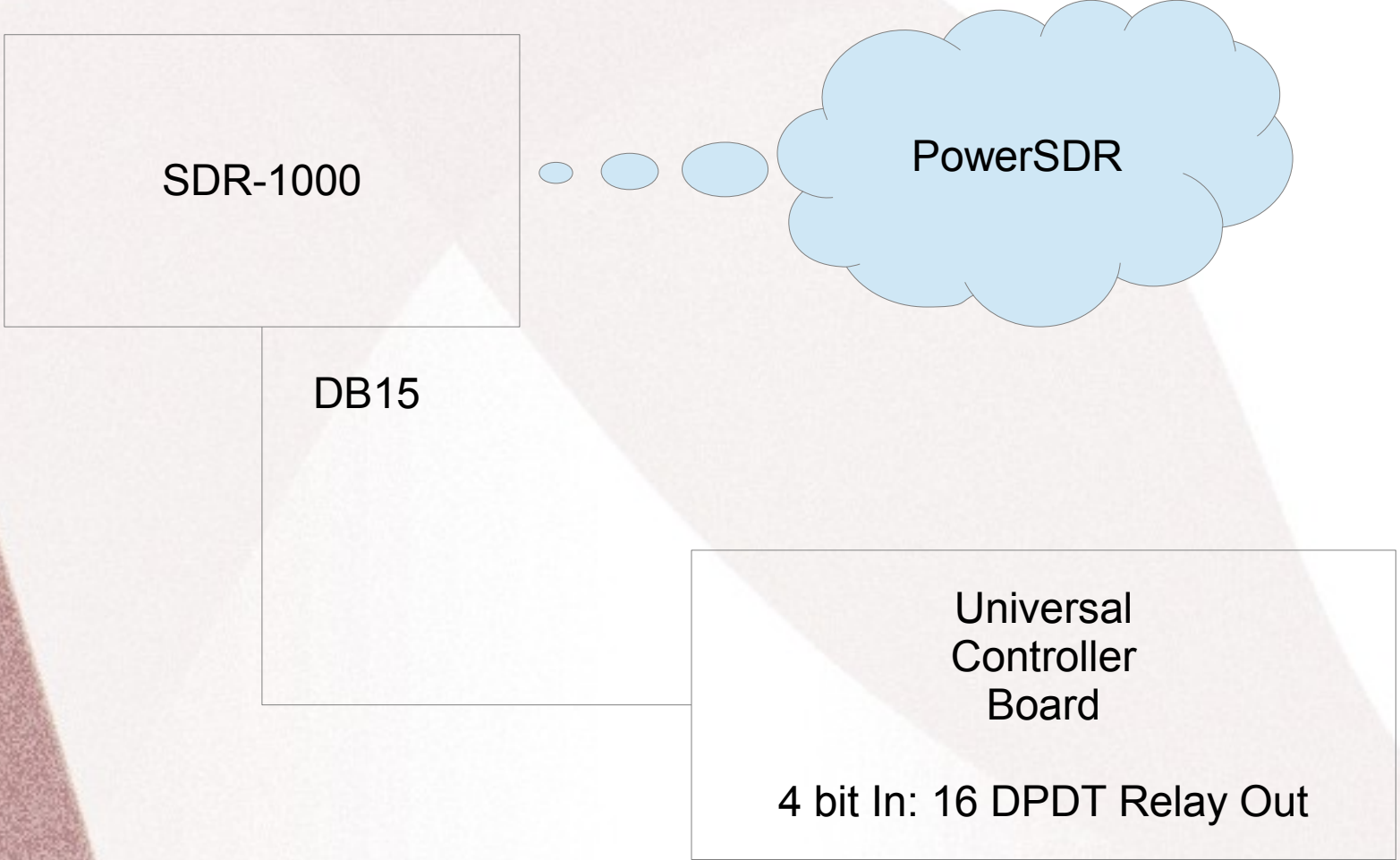
**“A Radio Server for VHF+ Contesting
And Weak Signal Work”**

**A Radio Server for HF, VHF+ Contesting, and
Weak Signal Work using a port 80 (browser
based) control approach**

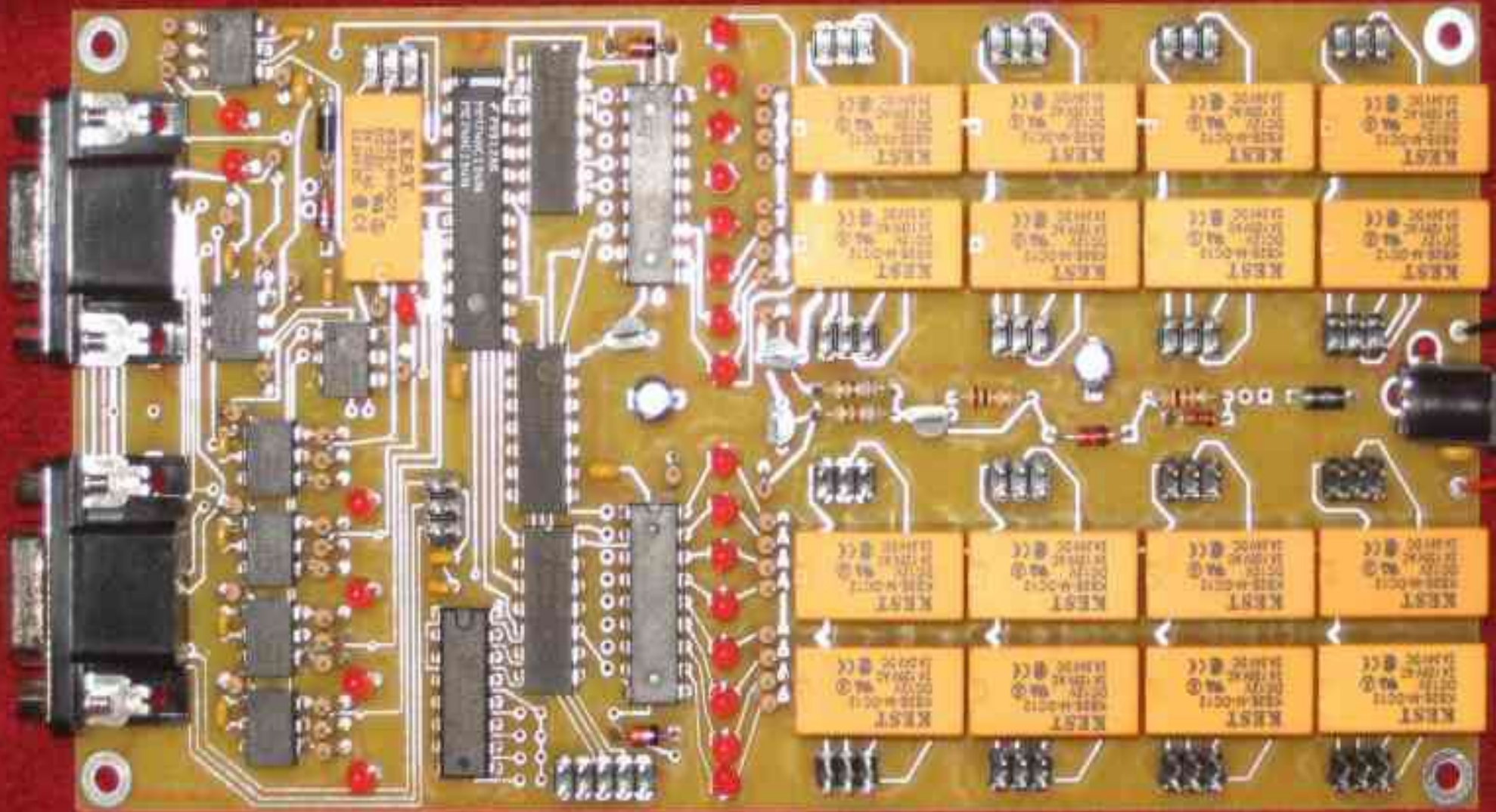
Phil Theis K3TUF

**Digital Communications Conference
October 10, 2015**

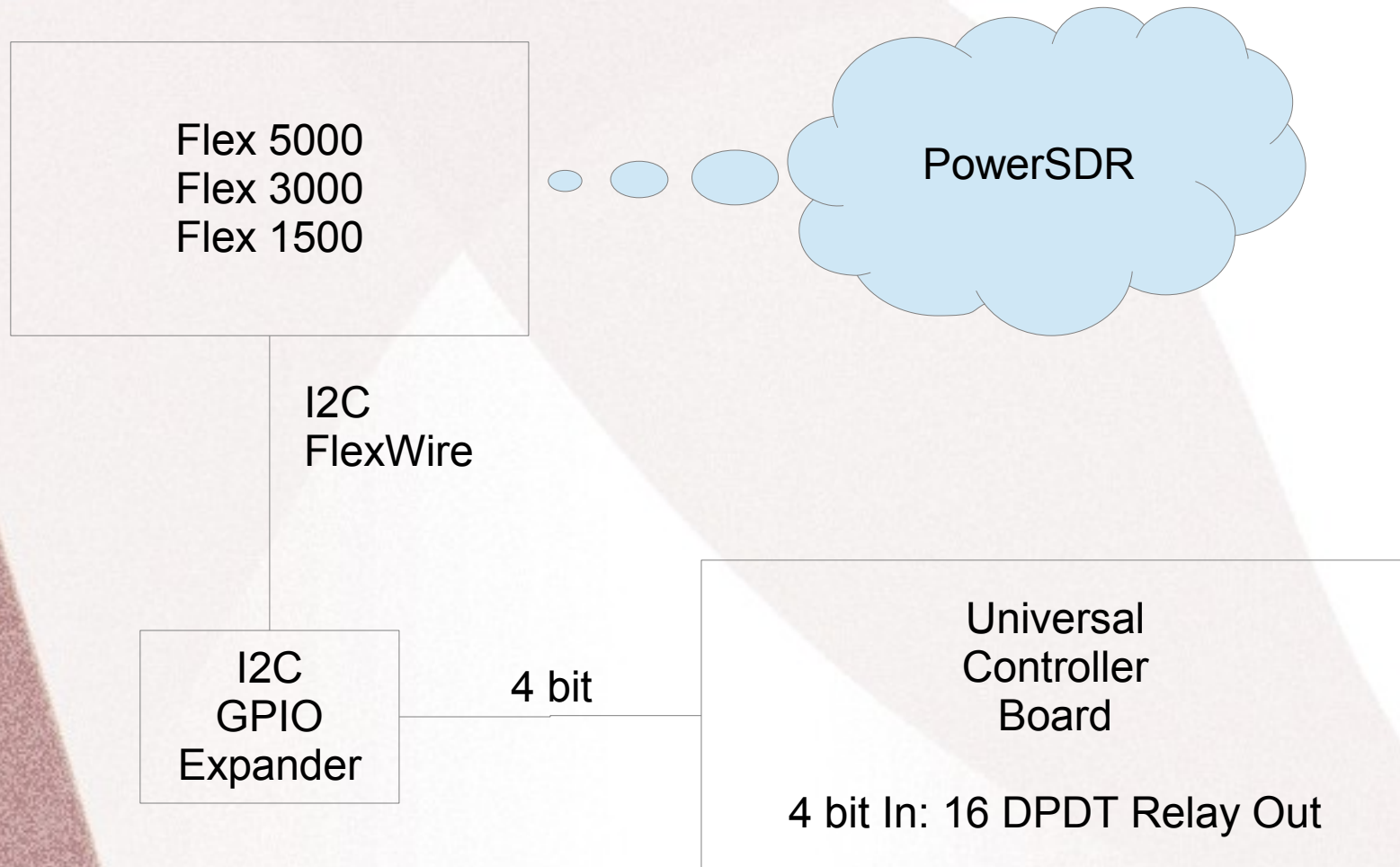
Background



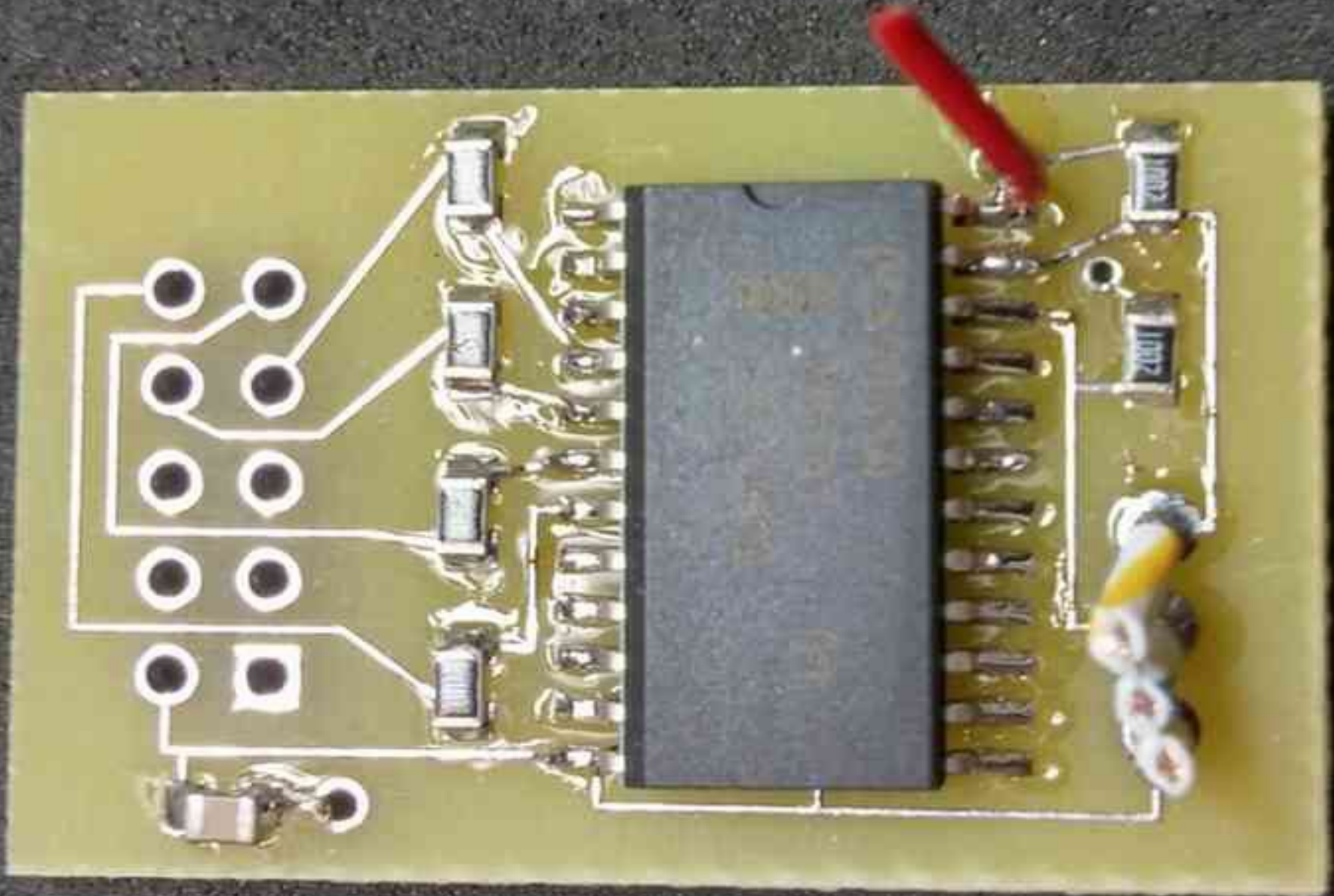
SDR1000 UCB



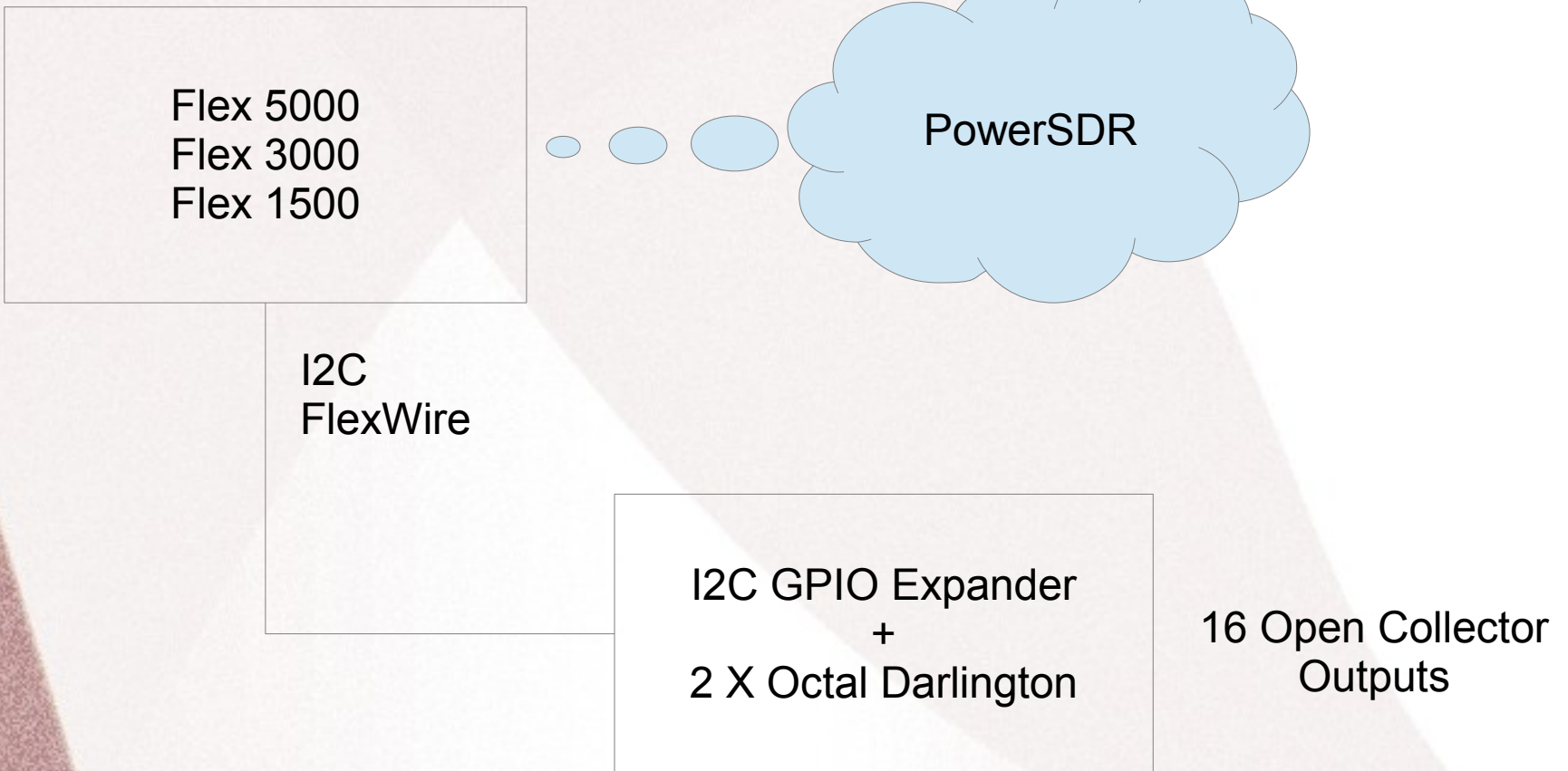
Background



UCB Daughter Board



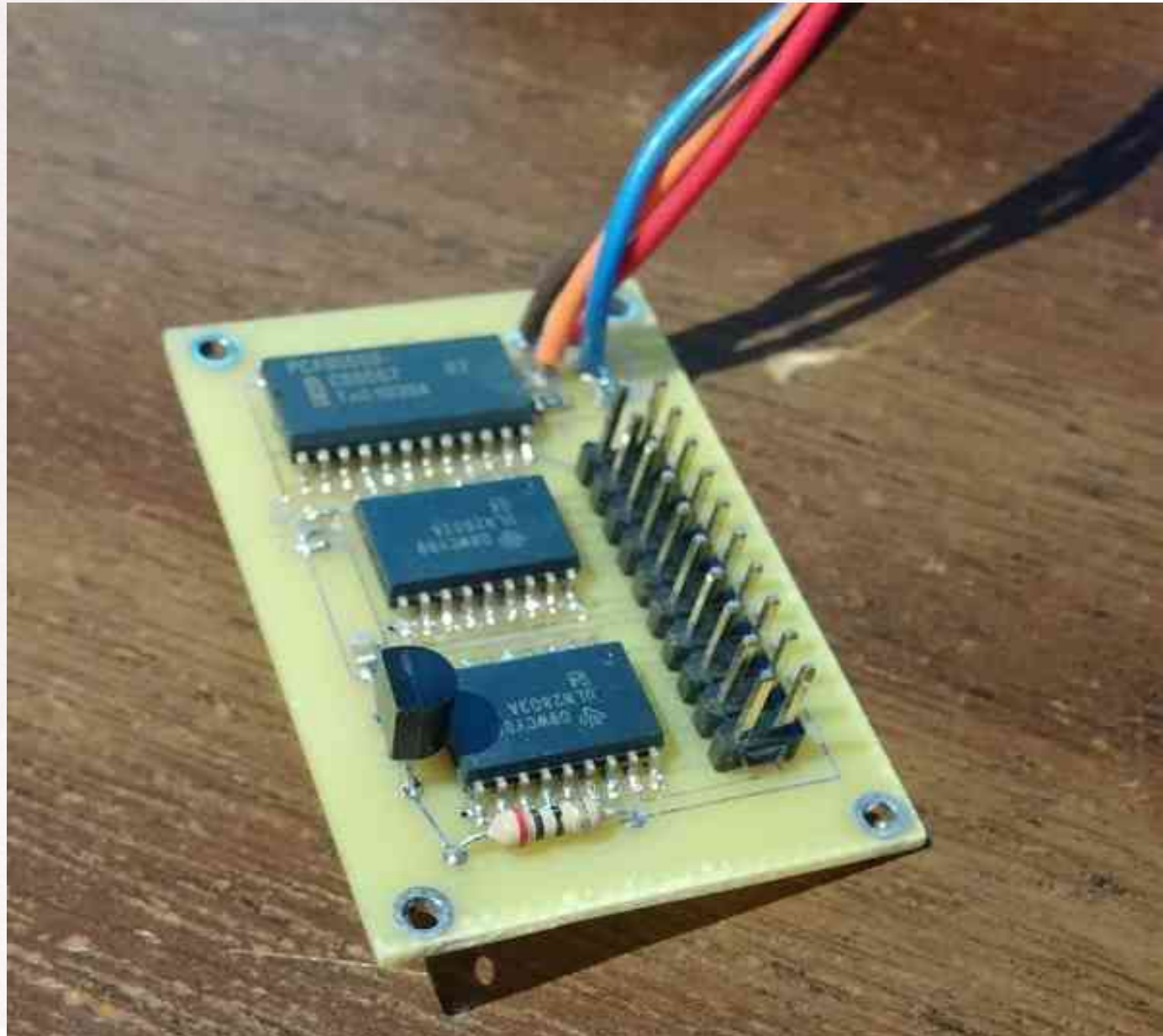
Background



FlexWire Board

Used with:

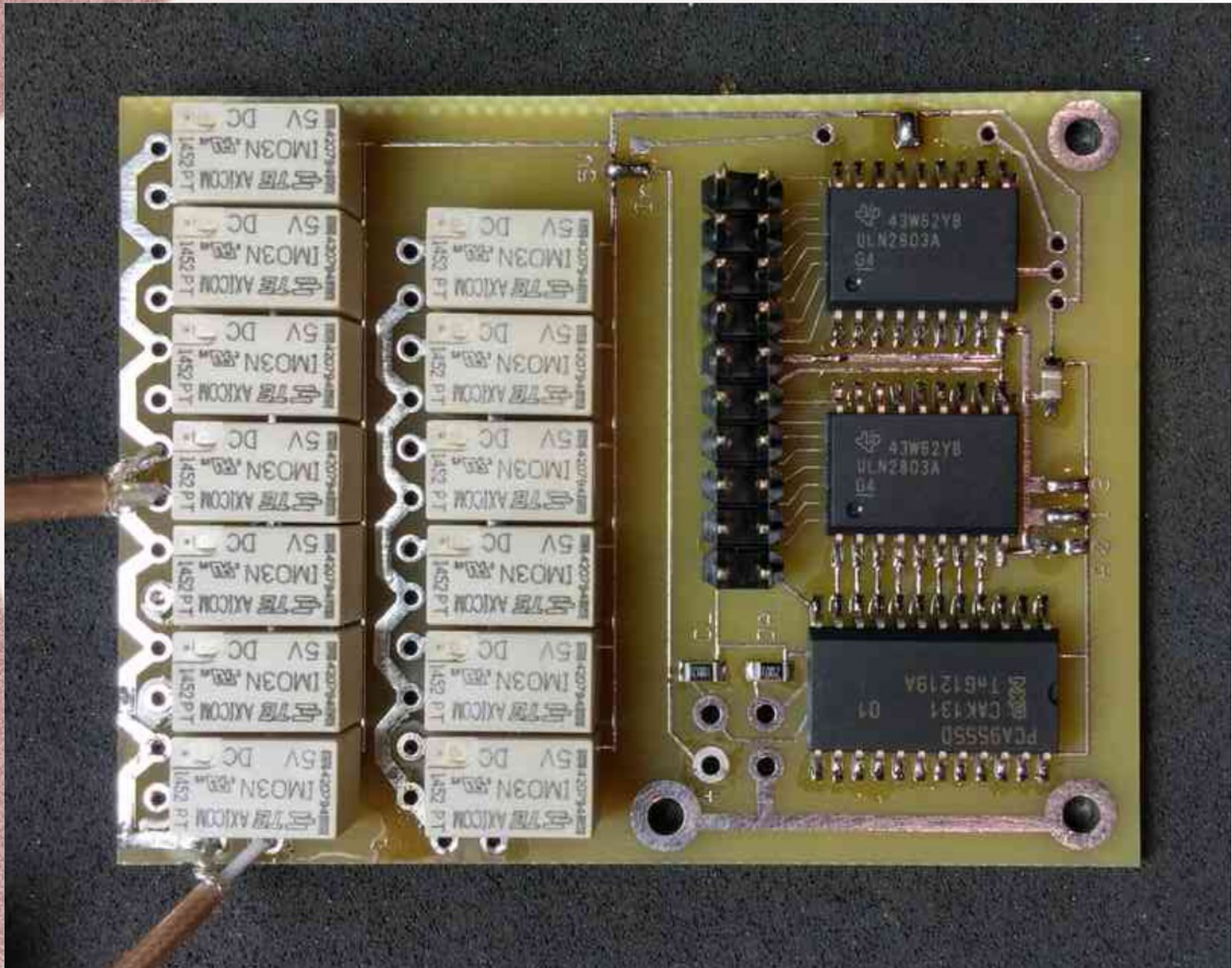
Flex 1500
Flex 3000
Flex 5000





Finding the right MultiPole Relay

FlexWire(I2C) with RF Relays



Flex 6000 series



Initial Plans

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

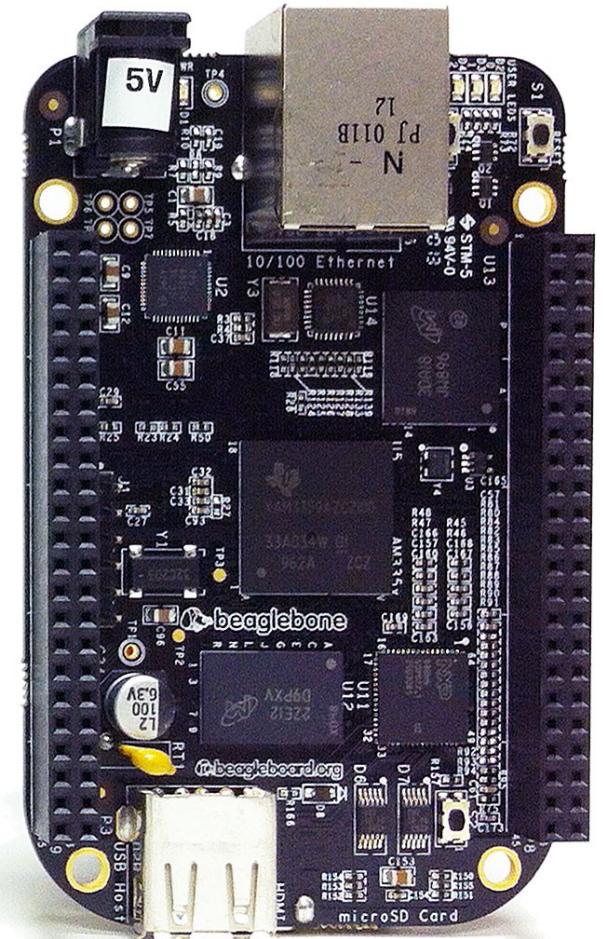
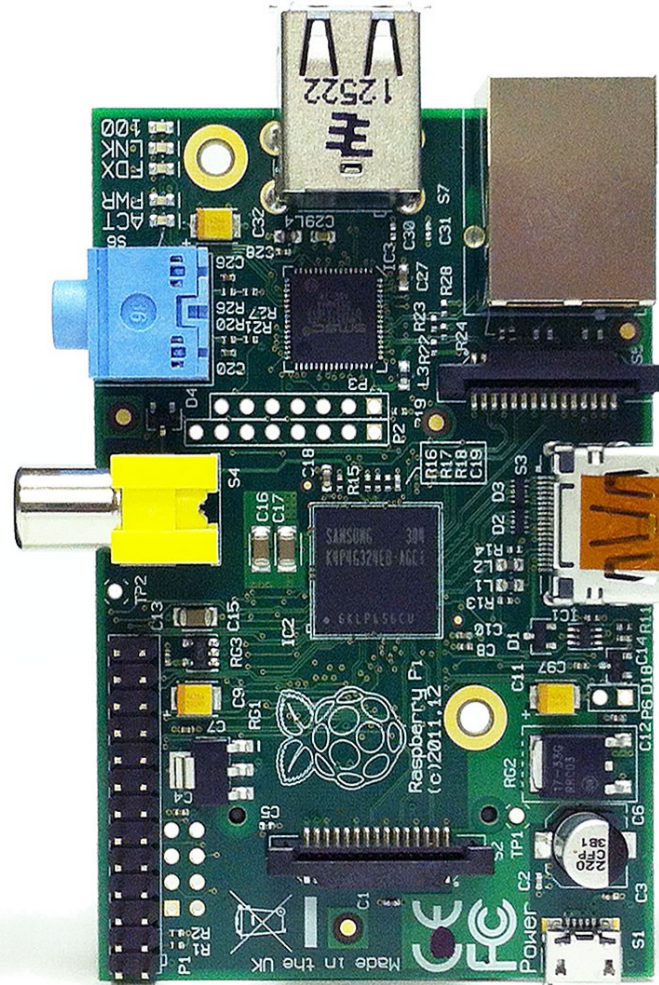
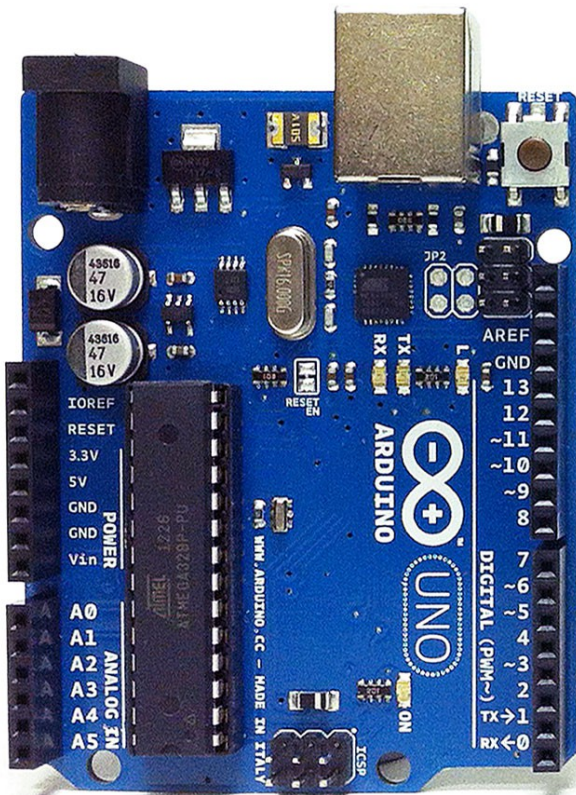
October 2014

Talk Today

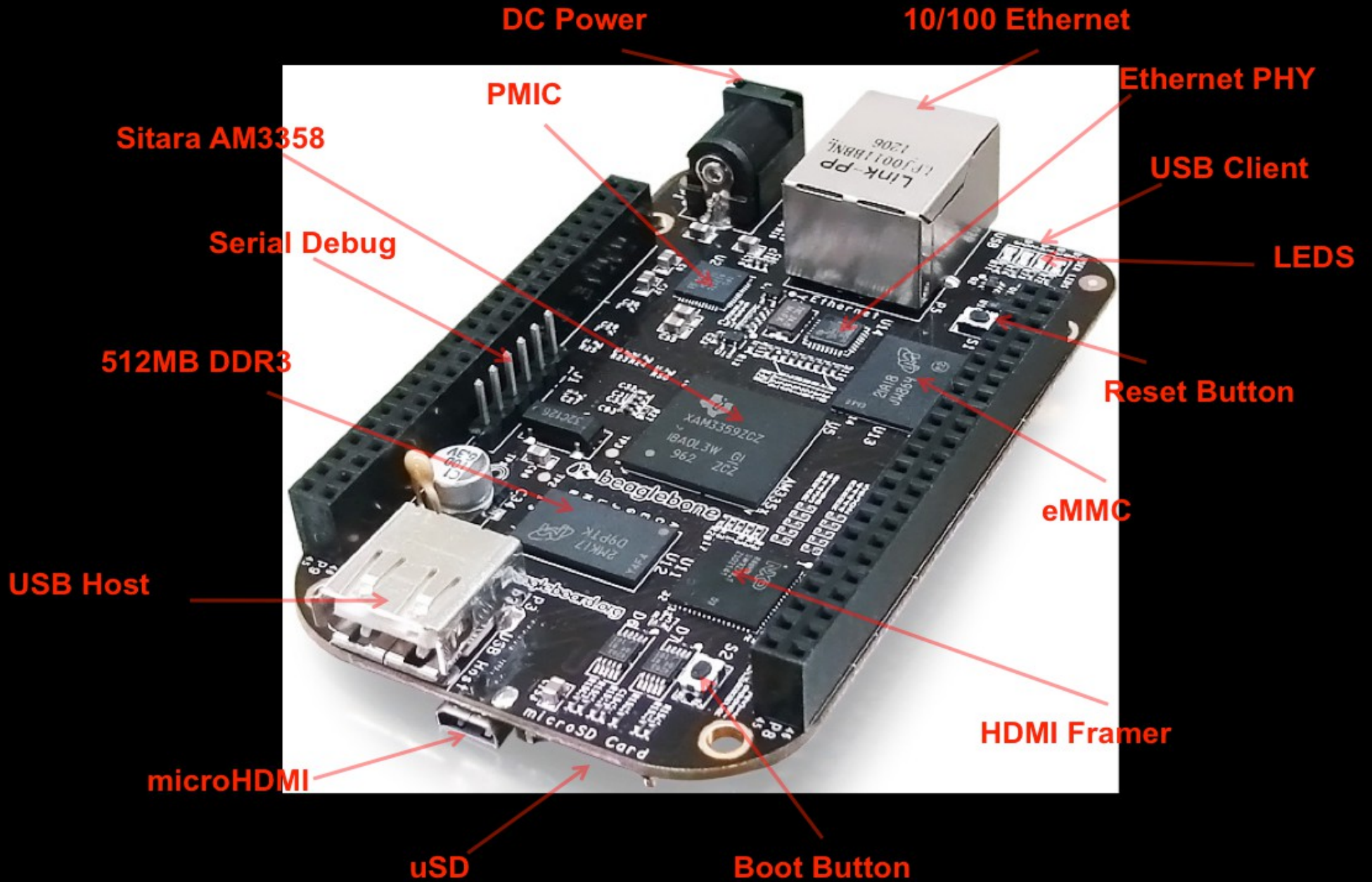
- Take you through the Process
- See what I learned along the way
- Much more that can happen
 - Transverter Control
 - Remote Control of 6K radios
 - Contest Mode Control
 - Tasks around the Shack
 - Monitoring
- All Via Ethernet

Device Choices

- Arduino – Raspberry Pi – Beagle Bone



Beagle Bone Black

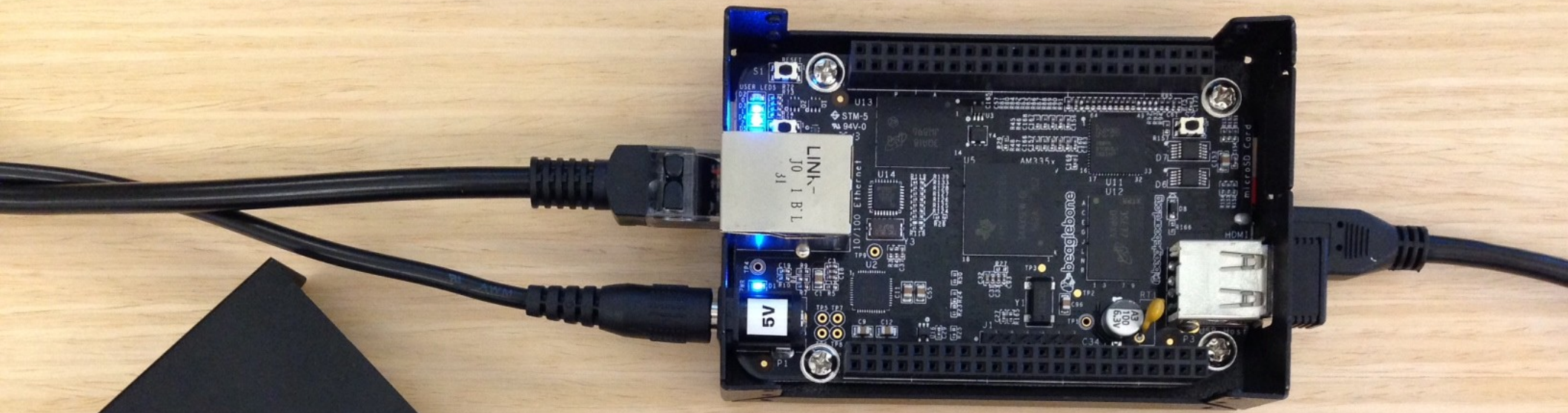


GPIO pins

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_BUT	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.



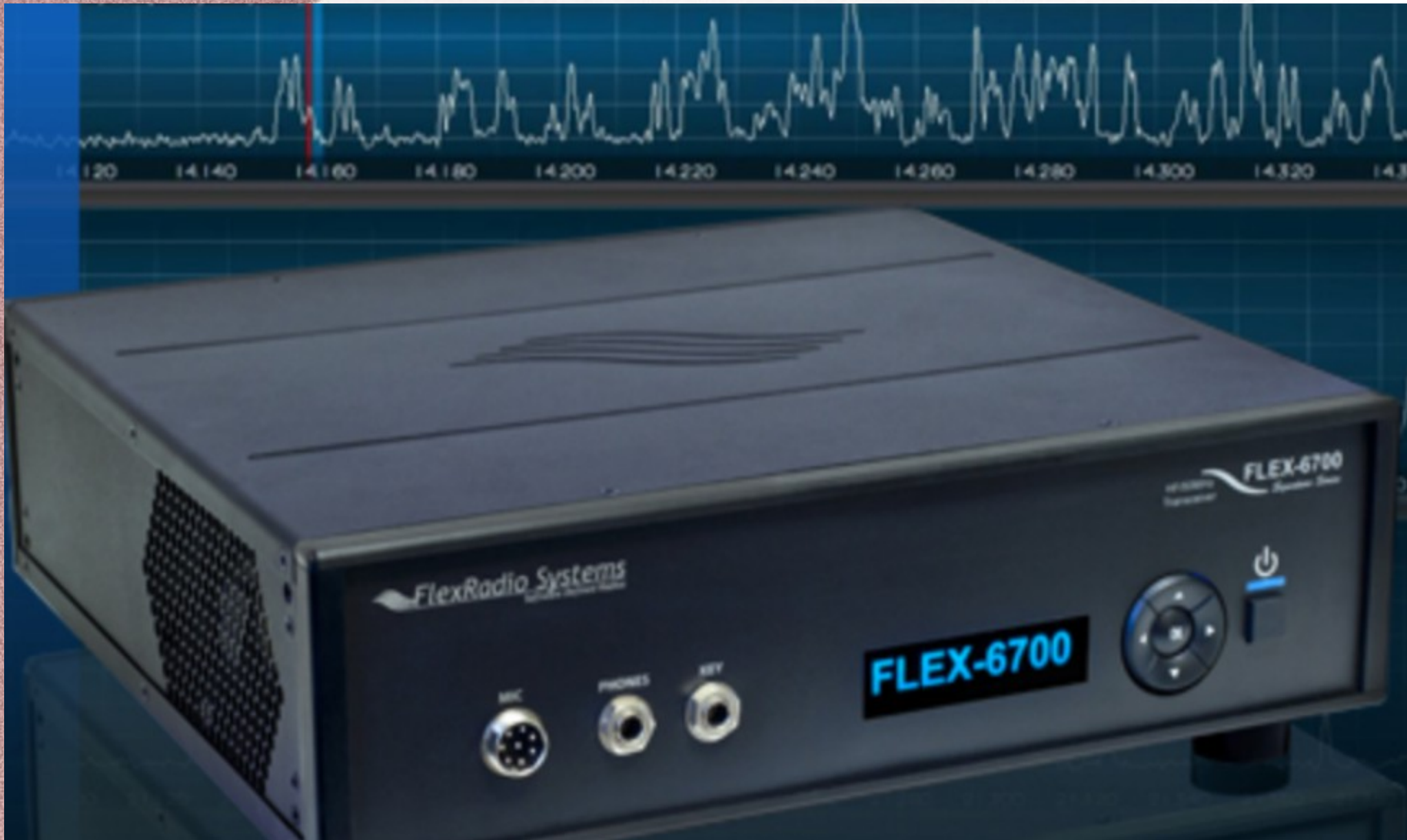
Apache Web Server



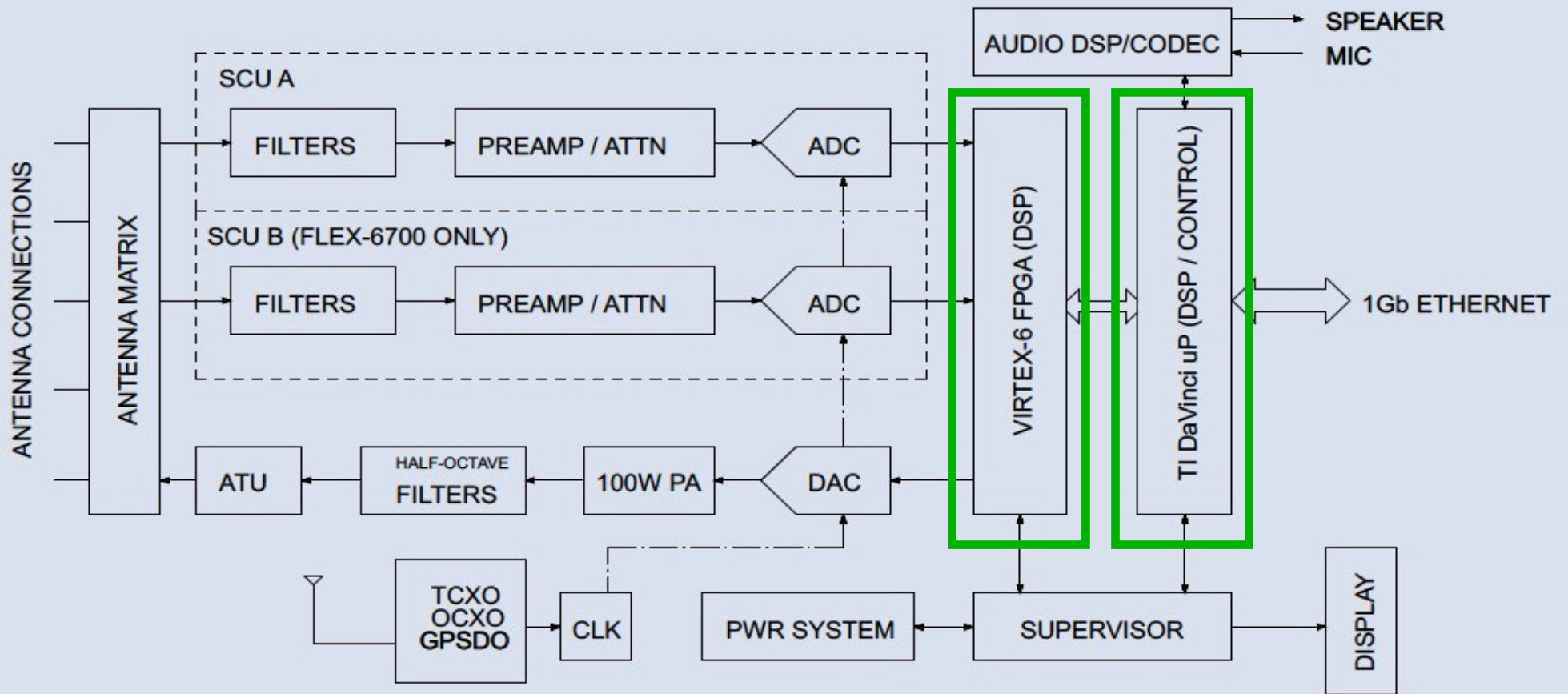
- Port 80
- PHP
- Available to any Device



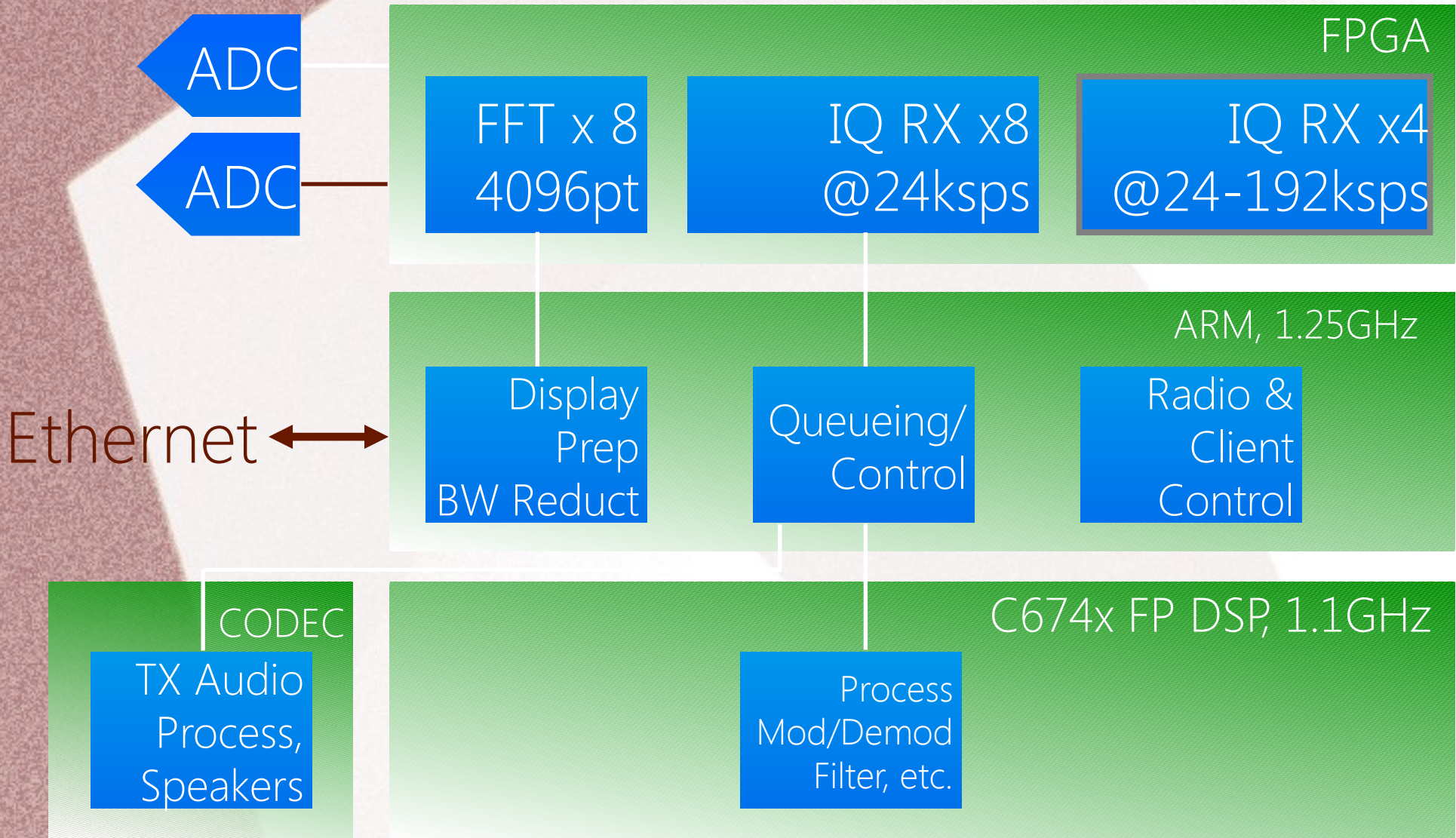
The Radio Server



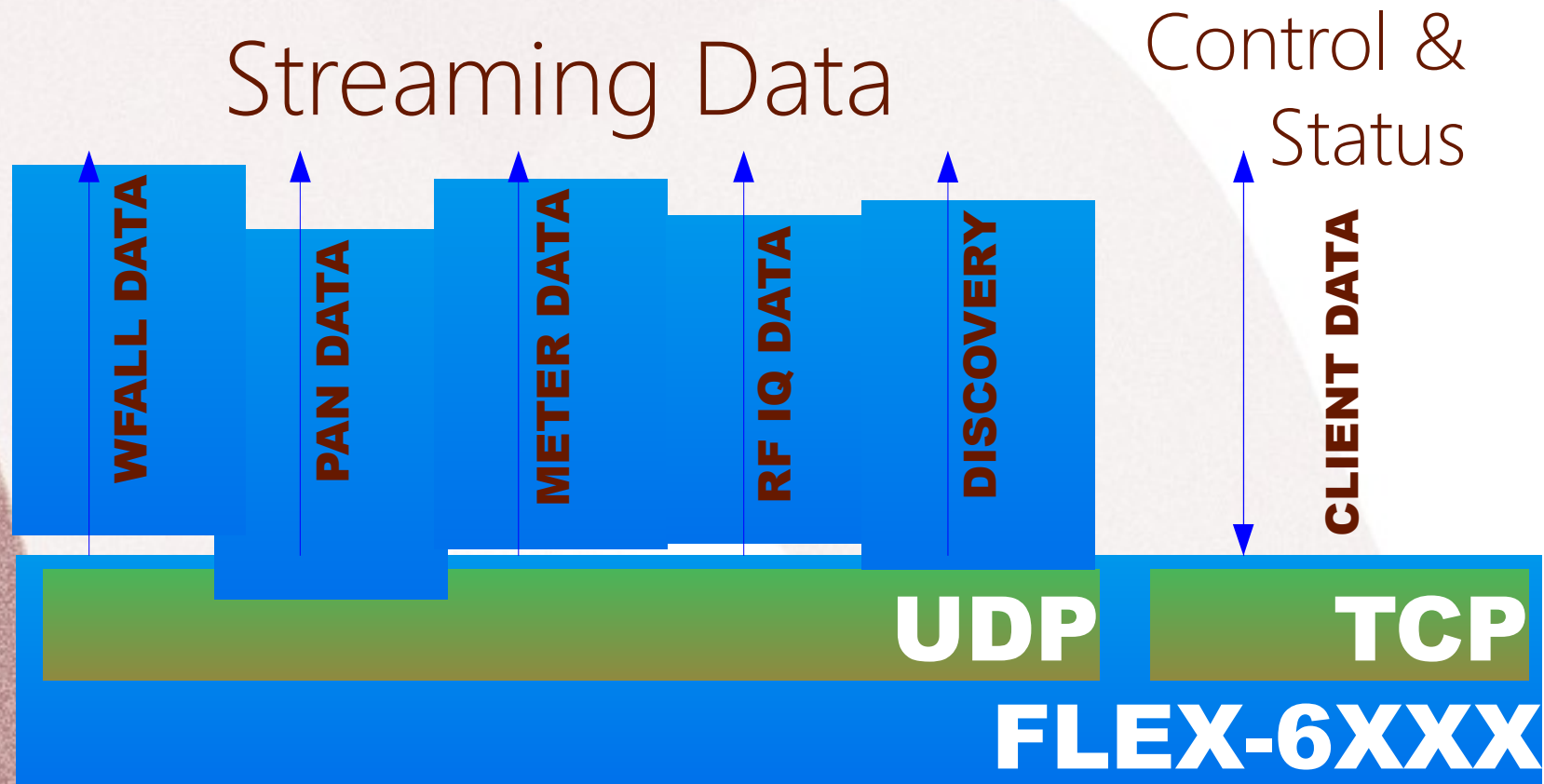
FLEX-6000 HW System Architecture



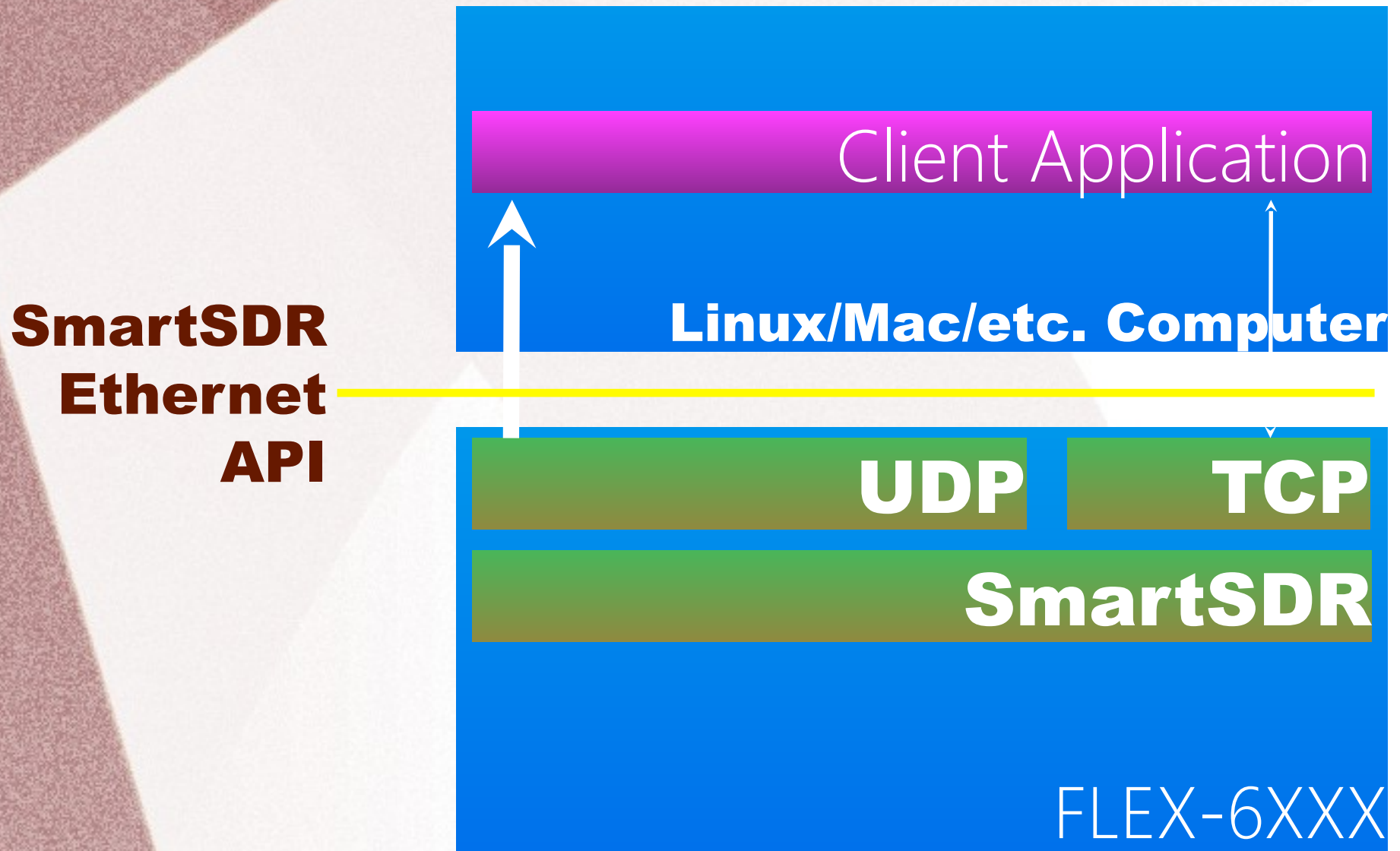
SmartSDR SW Architecture



SmartSDR Ethernet API Interfaces



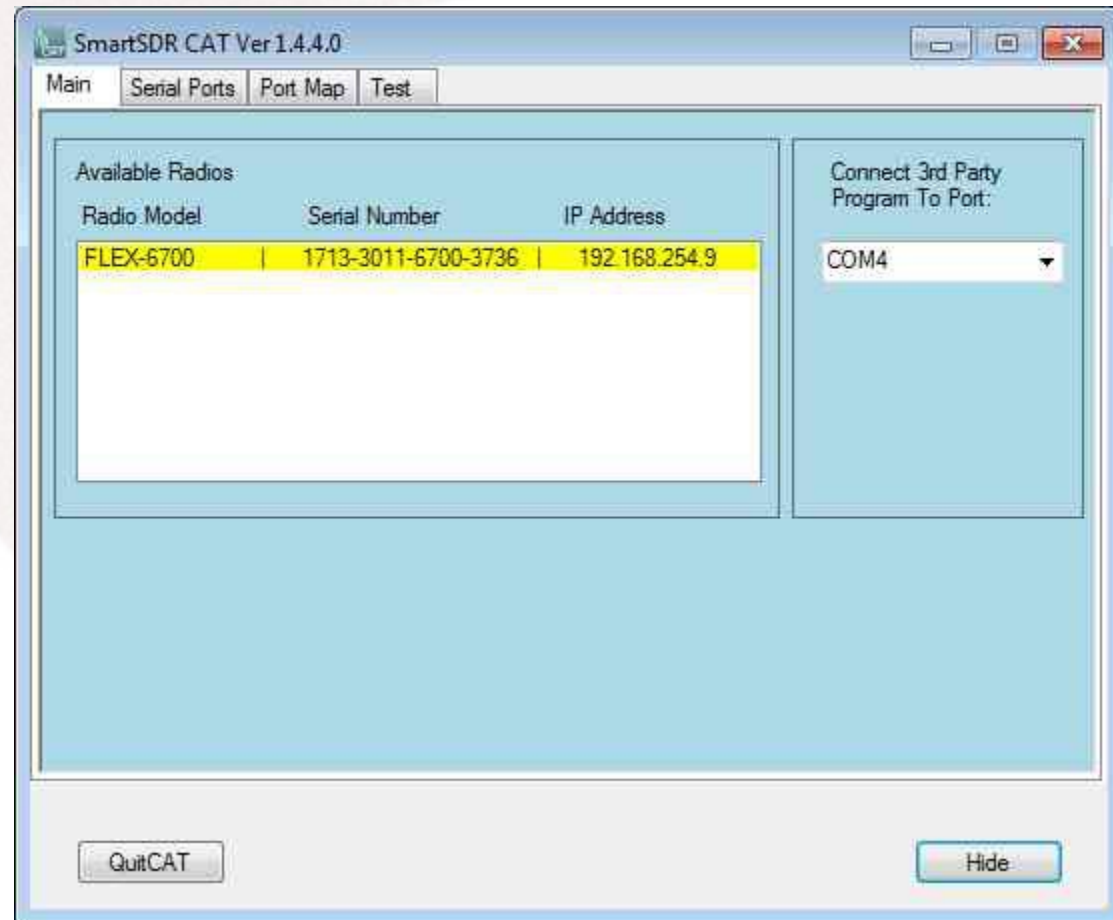
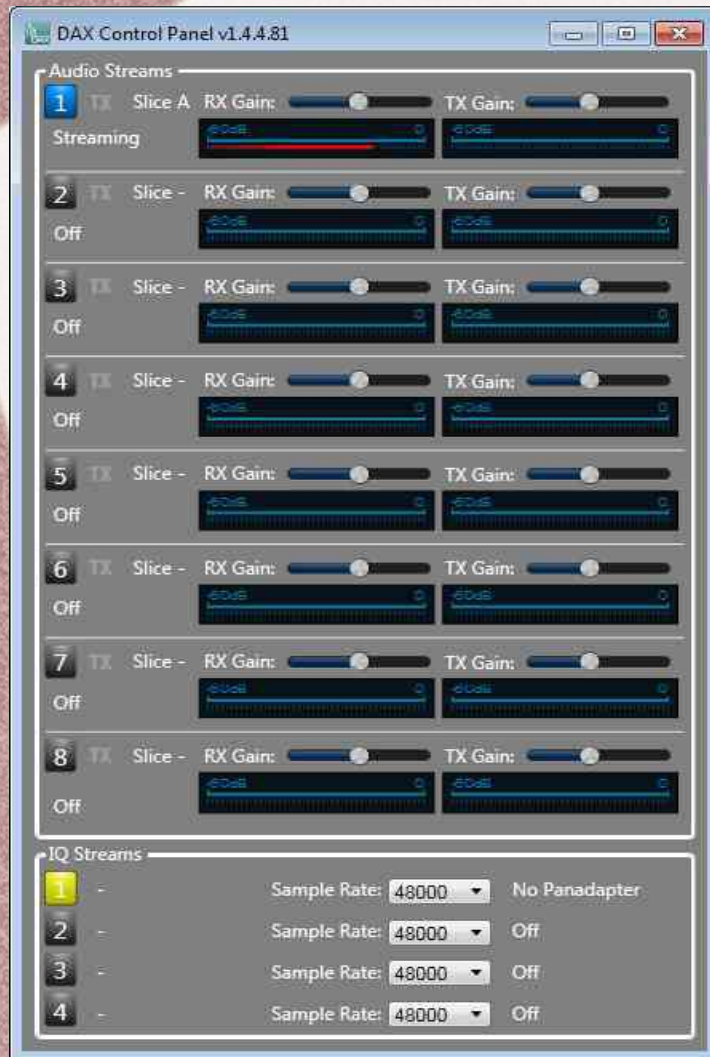
3rd Party App using Ethernet API



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

DAX & SmartCAT



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

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)

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

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

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
```

```
R45|0
```

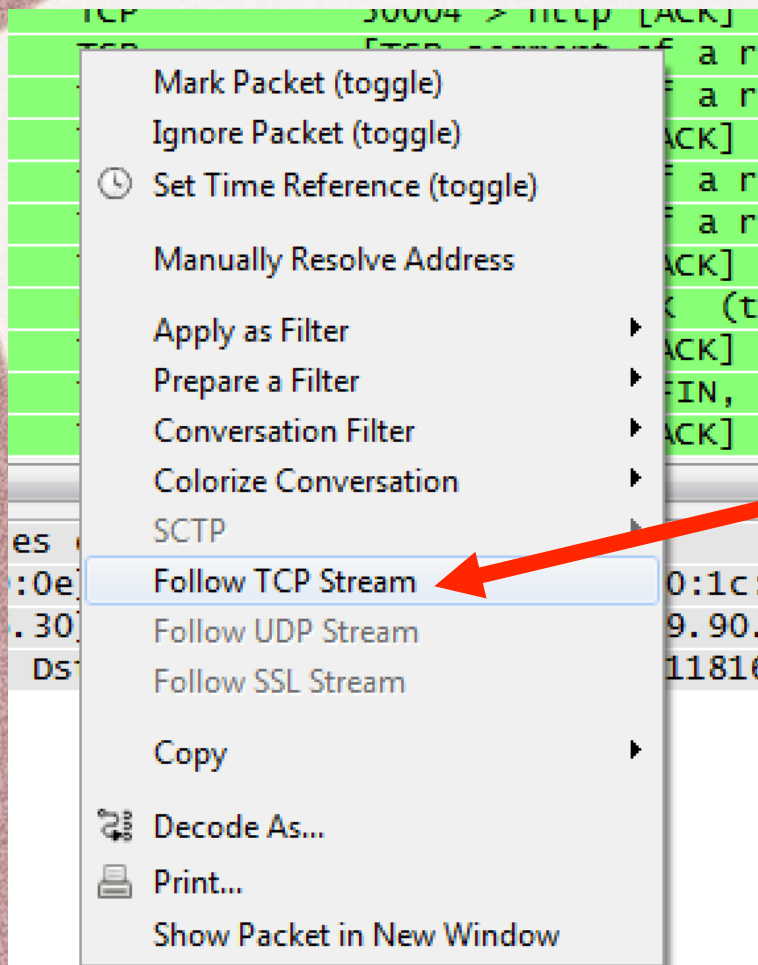
- ▶ Change slice receiver settings

```
slice set <slice> [<feature>=<value>]
```

```
C71|slice set 0 diversity=1 tx=0 record=1
```

```
R71|0
```


Sniffing TCP/IP / Using Wireshark



The Magic of the API

```
192.168.254.11 - PuTTY
V1.2.0.0
H80785E43
C5|sub slice all
R5|0|
M10000001|Client connected from IP 192.168.254.18
S80785E43|radio slices=7 panadapters=7 lineout_gain=46 lineout_mute=1 headphone_gain=45 headphone_mute=0 remote_on_enabled=0 pll_done=1 freq_error_ppb=-263 cal_freq=15.000 tnf_enabled=1 snap_tune_enabled=1 nickname=144-6700 callsign=K3TUF-1 binaural_rx=1 full_duplex_enabled=0
S80785E43|interlock timeout=0 acc_txreq_enable=0 rca_txreq_enable=0 acc_txreq_polarity=0 rca_txreq_polarity=0 tx1_enabled=1 tx1_delay=0 tx2_enabled=1 tx2_delay=0 tx3_enabled=1 tx3_delay=0 acc_tx_enabled=1 acc_tx_delay=0 tx_delay=0
S80785E43|slice 0 in_use=1 RF_frequency=28.450000 rit_on=0 rit_freq=0 xit_on=0 xit_freq=0 rxant=ANT1 mode=USB wide=0 filter_lo=100 filter_hi=2800 step=50 step_list=1,10,50,100,500,1000,2000,3000 agc_mode=med agc_threshold=70 agc_off_level=10 pan=0x40000000 txant=ANT1 loopa=0 loopb=0 qsk=0 dax=1 dax_clients=2 lock=0 tx=1 dax_tx=0 active=1 audio_gain=50 audio_pan=50 audio_mute=0 record=0 play=disabled record_time=0.0 anf=0 anf_level=0 nr=0 nr_level=0 nb=0 nb_level=0 wnb=0 wnb_level=0 apf=0 apf_level=0 squelch=1 squelch_level=20 diversity=0 diversity_parent=0 diversity_child=0 diversity_index=1342177293 ant_list=ANT1,ANT2,RX_A,RX_B,XVTR mode_list=LSB,USB,AM,CW,DIGL,DIGU,SAM,FM,NFM,DFM,RTTY fm_tone_mode=OFF fm_tone_value=67.0 fm_repeater_offset_freq=0.000000 tx_offset_freq=0.000000 repeater_offset_dir=SIMPLEX fm_tone_burst=0 fm_deviation=5000 dfm_pre_de_emphasis=0 post_demod_low=300 post_demod_high=3300 rtty_mark=2125 rtty_shift=170 digl_offset=2210 digu_offset=1500 post_demod_bypass=0 rfgain=0
S80785E43|slice 0 audio_gain=50 audio_pan=50 audio_mute=0
S80785E43|waveform installed_list=
SAFB2BE37|slice 0 RF_frequency=28.450050 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.450100 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.450150 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.450100 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.450050 wide=0 lock=0
SAFB2BE37|radio lineout_gain=46 lineout_mute=0 headphone_gain=45 headphone_mute=0
SAFB2BE37|slice 0 RF_frequency=28.453247 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.453150 wide=0 lock=0
SAFB2BE37|slice 0 RF_frequency=28.453100 wide=0 lock=0
S0|interlock state=PTT_REQUESTED source=5W tx_allowed=1
S0|interlock state=TRANSMITTING source=5W tx_allowed=1
S0|interlock state=UNKEY_REQUESTED tx_allowed=0
S0|interlock state=READY tx_allowed=1
SAFB2BE37|slice 0 pan=0x40000000 mode=USB qsk=0 tx=0
S0|interlock state=RECEIVE tx_allowed=0
SAFB2BE37|slice 0 pan=0x40000000 mode=USB qsk=0 tx=1
SAFB2BE37|interlock timeout=0 acc_txreq_enable=0 rca_txreq_enable=0 acc_txreq_polarity=0 rca_txreq_polarity=0 tx1_enabled=1 tx1_delay=0 tx2_enabled=1 tx2_delay=0 tx3_enabled=1 tx3_delay=0 acc_tx_enabled=1 acc_tx_delay=0 tx_delay=0
```

Eclipse Development Environment

The screenshot displays the Eclipse IDE interface. The main editor window shows a C program named `bndchg2.c` with the following code:

```
194     else printf("%s\n", "we now have 3456");
195     currentband = 3456;
196     sendband('8');
197     break;
198
199     case 5760 :
200     if (currentband == 5760) break;
201     else printf("%s\n", "we now have 5760");
202     currentband = 5760;
203     sendband('9');
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 HF for now
229     else printf (" %s\n", "must be HF");
230     currentband = 28;
231     sendband('0'); //No transverters
232
233 }
234 }
```

The left sidebar shows a file explorer with the following structure:

- Files
- My Home
- Root
- /
- bin
- boot
- dev
- etc
- home
- lib
- lost+found
- media
- mnt
- opt
- proc
- root
- run
- sbin
- selinux
- srv
- sys
- tmp
- usr
- var
 - backups
 - cache
 - lib
 - local
 - lock
 - log
 - mail
 - opt
 - run
 - spool
 - tmp
 - www
 - file1
 - file2
 - file3
 - filexectest1
 - index.php

The right sidebar shows a list of header files and functions:

- stdio.h
- stdlib.h
- unistd.h
- string.h
- sys/types.h
- sys/socket.h
- netinet/in.h
- netdb.h
- error(const char*) : void
- open_file(char*, char*) : FILE*
- sendband(char) : int
- main(int, char*[]) : int

The bottom panel shows a terminal window with the following output:

```
192.168.254.36
Last login: Mon Apr 13 02:22:18 2015 from 192.168.254.10
root@e14BB-1:~# cd /usr/src
root@e14BB-1:~/src# ls
GPIO.h      bndchg2.c  file1      filetest1.c  gpio.cpp      gpiofiletest1.c  makeLED.c  patternmatchpointers  patternmatchtest.c  socktest2.c  socktest4  strtoktest
bndchg     devmem2.c  file2      filetest2.c  gpio.h        i2c              makeLEDs   patternmatchpointers.c  socktest3          socktest3.c  socktest4.c  strtoktest.c
bndchg1.c  devmem2.c  filetest1  filetest2.c  gpiofiletest1  makeLED          makeLEDs.cpp  patternmatchtest      socktest2          socktest3.c  socktest5.c
root@e14BB-1:~/src#
```

The bottom status bar shows the following information:

- Writable
- Smart Insert
- 60 : 34

Programming Finally

- Program written in GNU 'C'
- Subscribes to Slice information in radio
- Parses the responses
- Watches for Frequency to change
- Sends signal to Band change output
 - Either GPIO or I2C
- Expands to additional needs
 - Active Slice
 - Active TX

Flex Web Interface

FLEX Web Interface	
Band	50MHz
Frequency	<input type="text"/> <input type="button" value="Change Frequency"/>
<input type="button" value="Mute"/>	Mode <input type="button" value="USB"/> <input type="button" value="LSB"/> <input type="button" value="CW"/> <input type="button" value="FM"/>
Brought to you by K3TUF	

- Apache HTTP Server
- Show Radio Status
- Send commands to Radio
- Perhaps display Panadapter
- Waterfall?

Technology: Languages

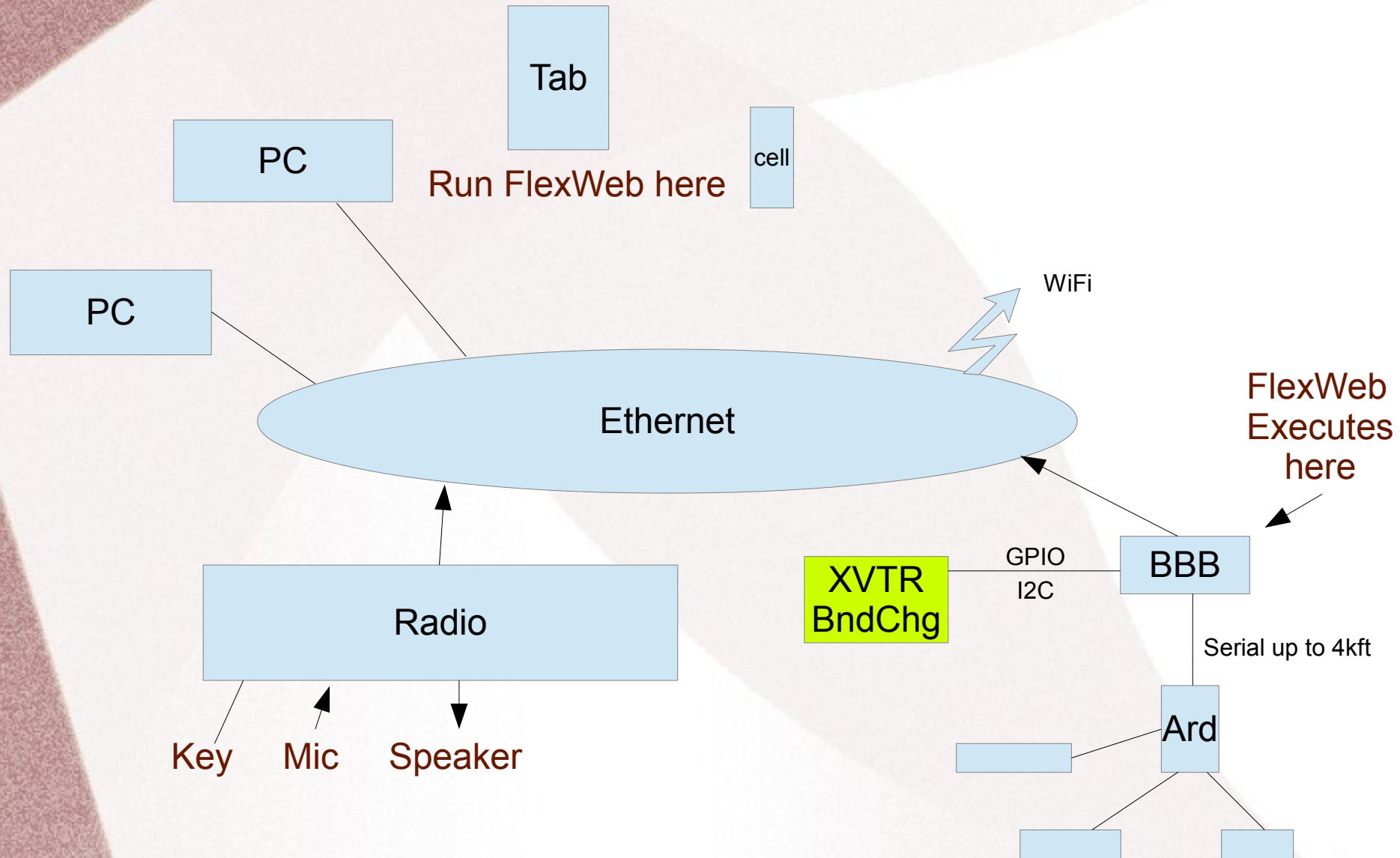
- HTML Hyper Text Markup Language
- AJAX Asynchronous JavaScript and XML
- DOM The Document Object Model is a platform and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents
- Apache / PHP is a server-side scripting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

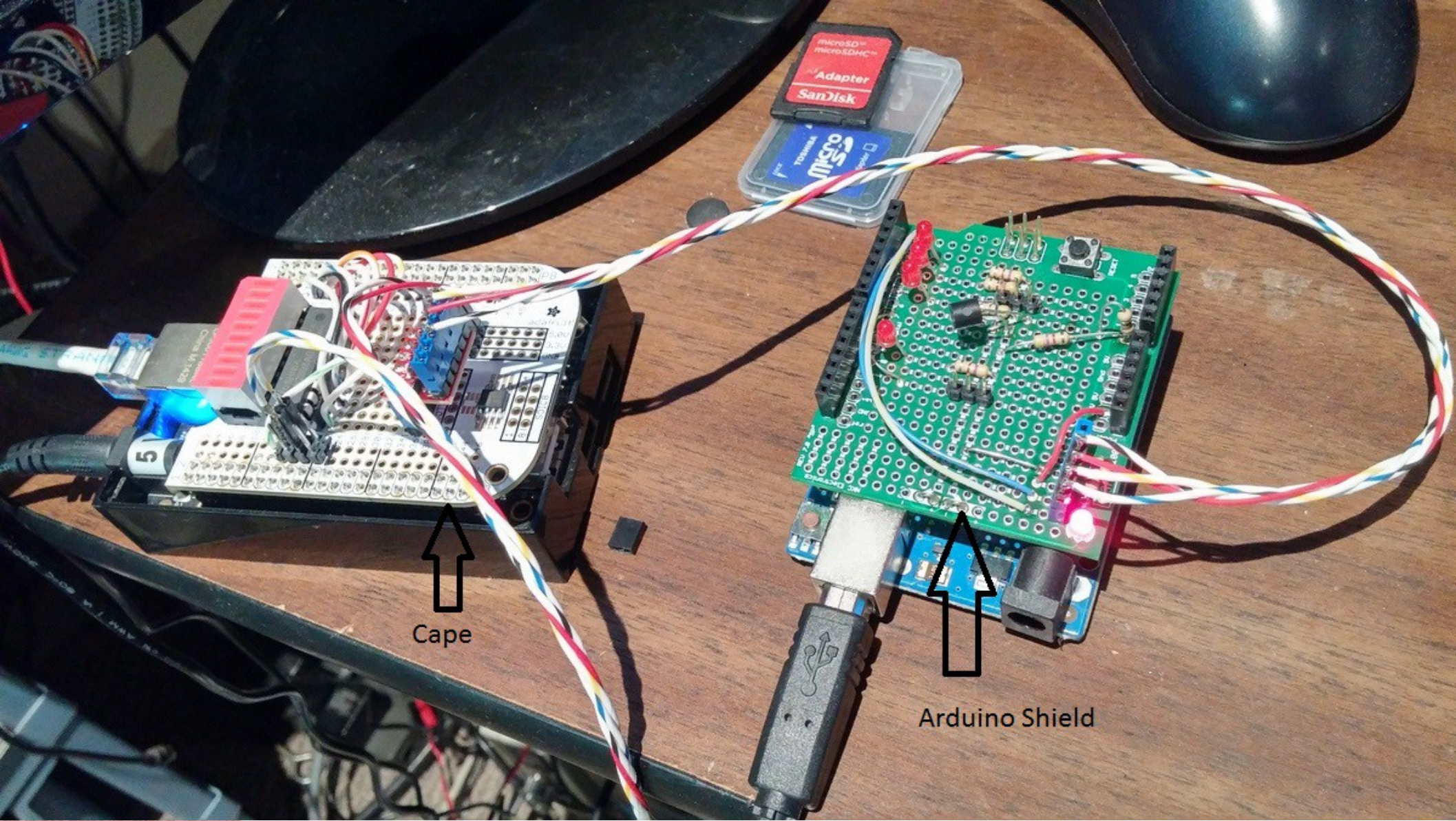
Technology: Languages cont'd

- 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

My Port 80 Plan

(Web Access)

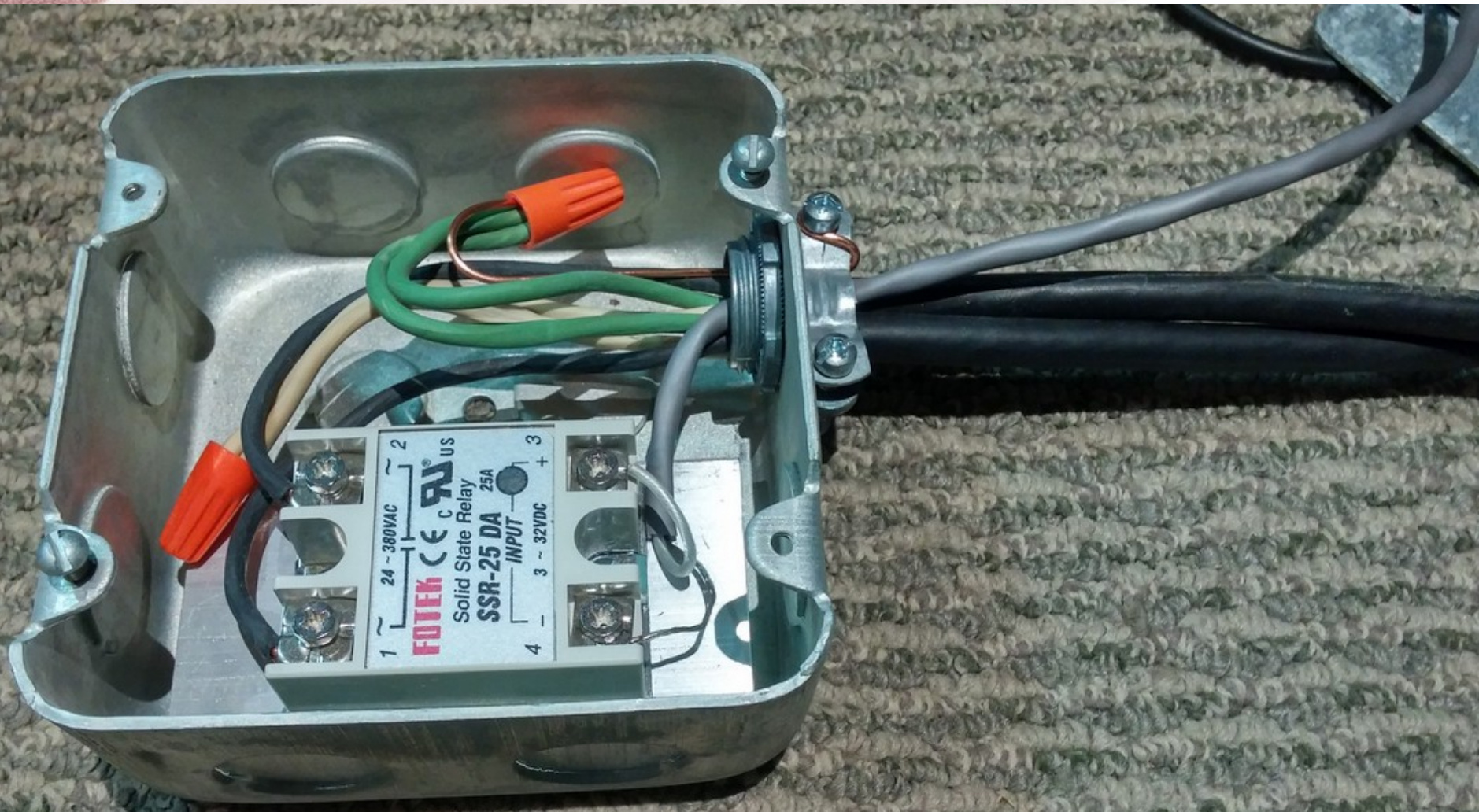


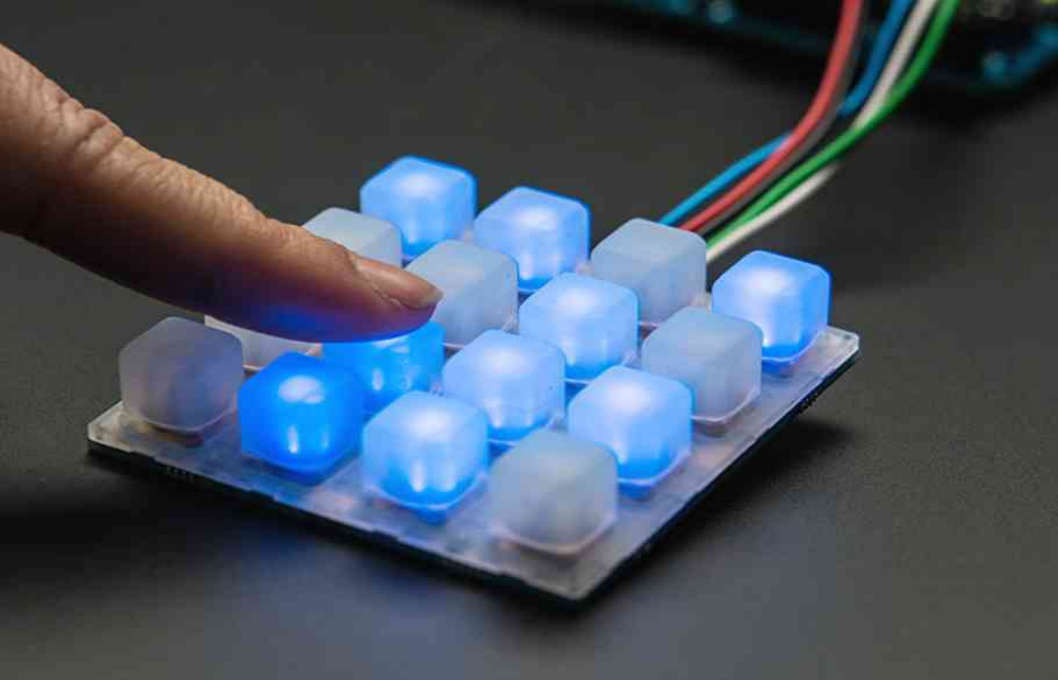


Cape

Arduino Shield

Hi Current Control





- 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
- Round out the Remote Experience
- Multiple Locations with Distributed Computing
- Beacon Monitoring: Propagation Notification
- Performance of Beacons: Real Time Status
- Operate Station from FL in Winter

Thank You

Phil Theis, K3TUF
phil@k3tuf.com