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
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

<table>
<thead>
<tr>
<th></th>
<th>Pan Frank</th>
<th>Max BW</th>
<th>DAXIQ</th>
<th>SCU</th>
<th>ATU</th>
<th>GPSDO</th>
<th>Mic</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLEX-6300</td>
<td>2</td>
<td>7 MHz</td>
<td>2 up to</td>
<td>2</td>
<td>Opt</td>
<td>—</td>
<td>Unbal</td>
<td>160—6m</td>
</tr>
<tr>
<td>FLEX-6500</td>
<td>4</td>
<td>14 MHz</td>
<td>4 up to</td>
<td>4</td>
<td>YES</td>
<td>Opt</td>
<td>Unbal + Bal</td>
<td>160—4m</td>
</tr>
<tr>
<td>FLEX-6700</td>
<td>8</td>
<td>14 MHz</td>
<td>4 up to</td>
<td>8</td>
<td>YES</td>
<td>Opt</td>
<td>Unbal + Bal</td>
<td>160—2m</td>
</tr>
<tr>
<td>FLEX-6700R</td>
<td>8</td>
<td>14 MHz</td>
<td>4 up to</td>
<td>2</td>
<td>N/A</td>
<td>Opt</td>
<td>N/A</td>
<td>160—2m</td>
</tr>
</tbody>
</table>

FLEX-6000 HW System Architecture

![Diagram of FLEX-6000 hardware system architecture](image-url)
Talking to the Radio Server

SmartSDR Ethernet API Interfaces

- Streaming Data
  - WMALL DATA
  - FN DATA
  - METER DATA
  - RF IQ DATA
  - DISCOVERY

- Control & Status
  - CLIENT DATA

- Protocols
  - UDP
  - TCP

SmartSDR and the use of FlexLib

SmartSDR APIs

- SmartSDR
- FlexLib API
- Ethernet API
- Windows Computer
- UDP
- TCP
- FLEX-6XXX

SmartSDR-Windows

FlexLib
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

SmartSDR FlexLib API
SmartSDR Ethernet API
SmartSDR Waveform API

SmartSDR Client GUI
FlexLib
Windows Computer
UDP
TCP
SmartSDR
Embedded App
FLEX-6XXX

What I am doing

3rd Party App using Ethernet API

SmartSDR Ethernet API

Client Application
Windows Computer
UDP
TCP
SmartSDR

N4PY K6TU
AVAILAIBLE TODAY
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

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 (24-192kps)
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|500000002
- Status preface, handle, v-bar, status
  S67EF9A22|slice 0 freq=7.243
  S67EF9A22|slice 0 filter_lo=300 filter_hi=2700

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
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

Learning the Protocol

Sniffing TCP/IP API
Using Wireshark
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
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