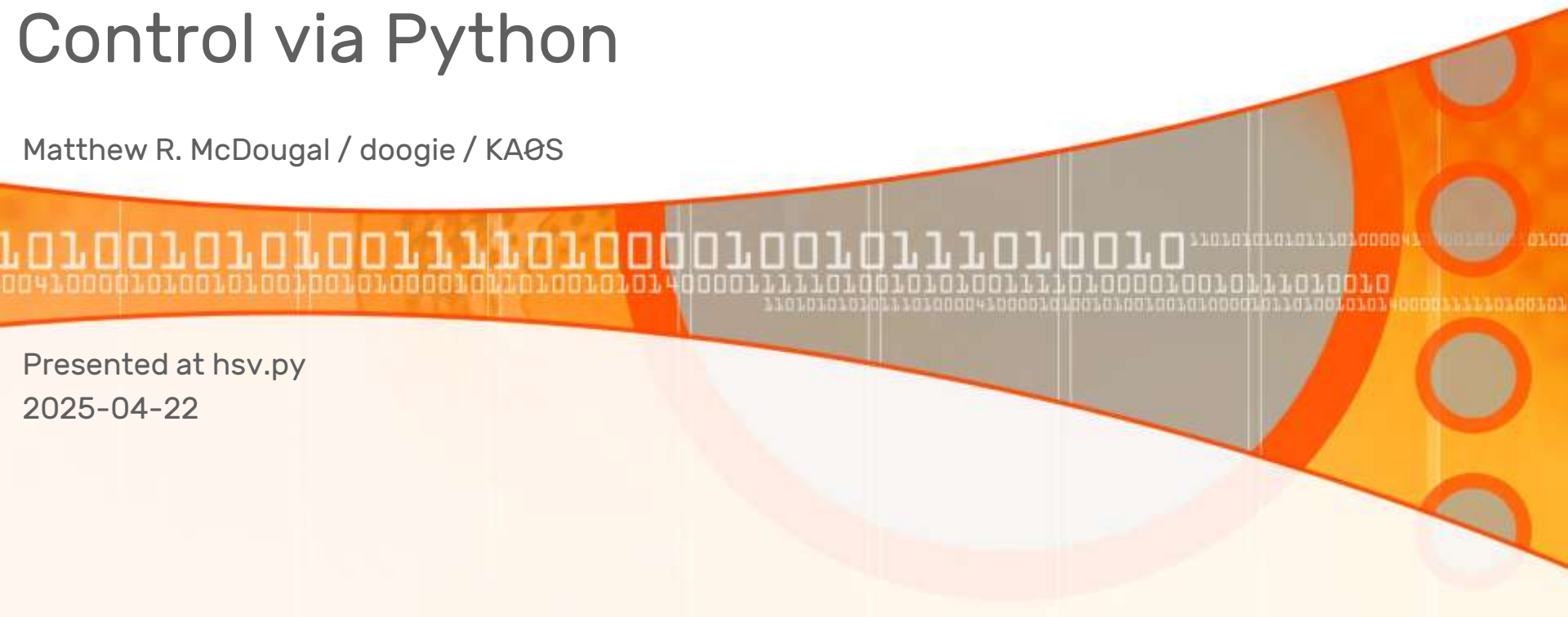
SunSDR2 Ham Radio Control via Python

Matthew R. McDougal / doogie / KA0S



Presented at hsv.py
2025-04-22

About Me

- Matt or doogie

- Extra class amateur radio operator
- Callsign KA0S
- Licensed since 2004

- Aerospace engineer by day
- Electronics hacker always

- Hobby is trying hobbies

Ham Radio

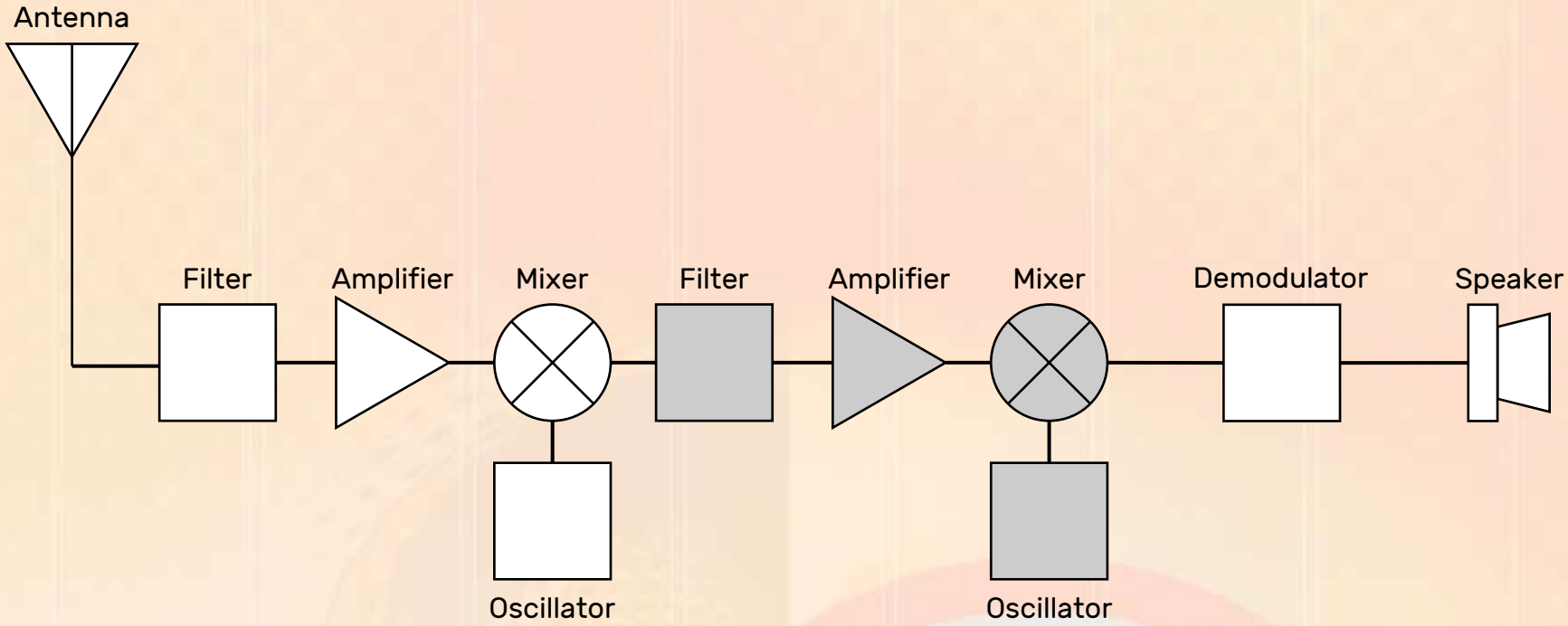
- Local & global communications with other amateur radio operators

- Many purposes:
  - Experimentation
  - Socialization
  - Emergency communications
  - Contests

- Has experimentation at its heart:
  - Trying new radios
  - Building antennas
  - Communications modes
  - Field operations
  - Effects of space weather

Ham Radio

Radio Architecture



Radio Architecture

- In standard radio architectures, the filters, mixers, oscillators, and modulation sections are all physical components with a signal chain fixed by the design of the radio
- If you want a narrower filter to reduce interference from a nearby radio, have to physically change out a module or individual components.
- With more powerful computers, digital signal processors, and microcontrollers, most of those operations can now be done digitally, giving significantly more flexibility

Software Defined Radio Advantages

- Software defined radios essentially directly digitize a large swath of radio signals which are then processed by computer software

- Visually see the radio signals in a given frequency range
- Click around to different signals
- Adjust filters
- Apply new noise reduction algorithms
- Experiment with new types of modulation

Rig Control & Audio Interfaces

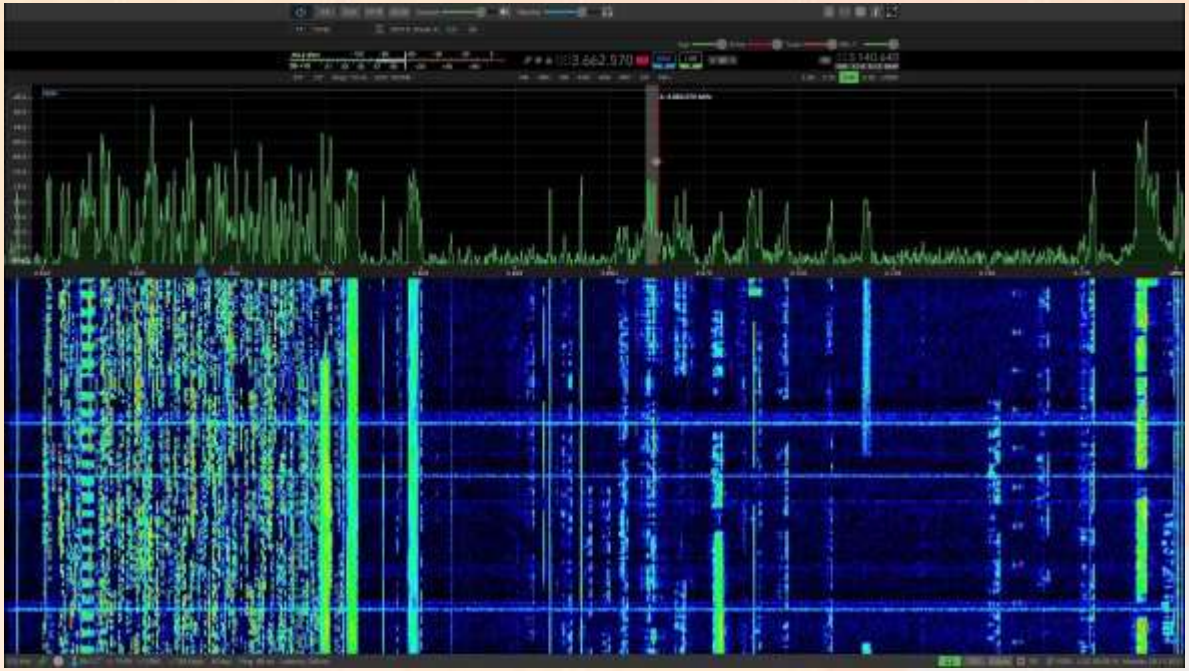
- Even on traditional radios, users often desire ways to control the tuning, filters, and other controls from a computer to assist with operating digital modes or automate various tasks

- Most modern radios have some form of control protocol
- Lots of limitations
  - Serial based
  - Manufacturer specific
  - Polling required
  - Not multi-application friendly
  - Inconsistent support for various controls

- Audio interfaces are even more cantankerous
  - Often at signal levels optimal for the radio but not soundcards (A few radios now have internal USB audio interfaces)
  - Virtual audio cables to pipe sound around

Expert Electronics SunSDR2

- Paired with ExpertSDR3 software
- New protocol for both transceiver control and audio interface – TCI



TCI – Transceiver Control Interface

- Openly documented: https://github.com/ExpertSDR3/TCI

- Based on modern WebSockets, facilitating remote and distributed control manipulation and audio transport
- Server-client architecture keeps all clients in sync

- For example:
  - Physical radio in a convenient location for antenna cables & grounding
  - Control software on a tucked away mini PC
  - Remote antenna switch or preamplifier placed on a tower
  - Macro pad or scroll wheel at a comfortable operating location
  - Logging software running on a laptop

- As long as these can all reach the control software via network, they can all interact seamlessly for their specific purpose!

WebSocket Protocol Basics

- Designed to be carried through normal HTTP/HTTPS transport
- Well supported by all modern web browsers
- Persistent, bi-directional connection

- Two types of frames: text and binary
- TCI uses text for control commands and binary for audio or IQ data

- Libraries available for most programming languages
- Generally implemented as event-driven using callbacks or asynchronous functions

- For Python:
  - https://pypi.org/project/websockets/
  - https://websockets.readthedocs.io/en/stable/

Python websockets & asyncio

- Python `websockets` library uses `asyncio` for event-driven programming
- Don't need to worry about `threading`, the GIL, etc.
- Only slight changes to overall code structure

- Two new keywords: `async` and `await`

```
import asyncio

async def func_1():
    await asyncio.sleep(1)
    print('Function 1 done')

async def func_2():
    await asyncio.sleep(2)
    print('Function 2 done')

async def main():
    await asyncio.gather(func_1(), func_2())

asyncio.run(main())
```

- Also get Queues, Locks, Events, Semaphores, etc.

My Library – eesdr-tci

- `asyncio` & `websockets` implementation of TCI protocol

- https://pypi.org/project/eesdr-tci/
- https://github.com/ars-ka0s/eesdr-tci

```
import asyncio
from eesdr_tci.listener import Listener

async def print_params(name, rx, subrx, params):
    print(f'{name.ljust(40)} {" " if rx is None else rx} {" " if subrx is None else subrx}   {params}')

async def main():
    tci_listener = Listener('ws://localhost:50001')
    tci_listener.add_param_listener('*', print_params)
    await tci_listener.start()
    await tci_listener.ready()
    await tci_listener.wait()

asyncio.run(main())
```

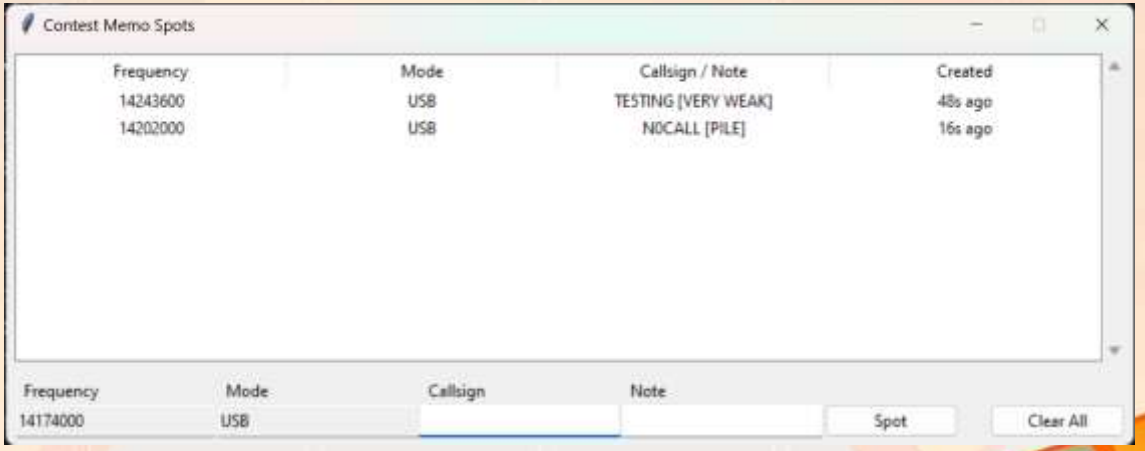
```
PROTOCOL                                 ['ExpertSDR3', 2.0]
DEVICE                                   SunSDR2DX
RECEIVE_ONLY                             False
TRX_COUNT                                2
CHANNELS_COUNT                           2
VFO_LIMITS                               [0, 160000000]
IF_LIMITS                                [-156250, 156250]
...
READY                                    None
...
APP_FOCUS                                True
IF                                  0 0  -14000
VFO                                 0 0  14236000
```

Simple Examples Included

- json_dump: outputs startup parameters as JSON dictionary
- param_listener: prints out all parameter changes
- receive_audio: receives audio stream which can be piped to other utilities
- spot_saved_stations: repeatedly spots a list of stations which places a clickable tag for that station on the EESDR interface
- scanner: moves between a list of stations and pauses if squelch is broken

Additional Examples Included

- ctcss_decode: listens for sub-audible tones in receiver audio and prints possible matches
- direwolf_interface: sends and receives audio via the direwolf packet radio soundmodem
- cw_macro_keyer: CW macro keyer with freeform text box and speed adjustment
- contest_memo: use the spotting feature to leave notes during contests



Separate Projects

- https://github.com/ars-ka0s/midi-to-tci
- Use a MIDI controller to manipulate numerous TCI parameters

Separate Projects

- https://github.com/ars-ka0s/eesdr-owrx-connector
- Use SunSDR2 as a signal source for OpenWebRX+, a web-accessible receive-only SDR interface with numerous built-in decoders