Outline

General context: we wish to design an instrument in which the data are collected by the Raspberry Pi 4, under control of the PC, to be transferred to the PC for processing and display.

GNU Radio on Raspberry Pi 4
1. making sure GNU Radio is properly installed \(^1\): accessing GNU Radio blocks and playing a sound
2. first demonstration with RTL-SDR dongle: FM receiver
3. from RPi4 to PC used as sound card: Zero-MQ publish/subscribe
4. from PC to RPi4: TCP/IP server running as a Python thread

Objective: a FM radio receiver running on the RPi4, streaming sound from the RPi4 to the PC, whose carrier frequency is controlled from the PC

\(^1\)raspberrypi4_64_gnuradio_defconfig Buildroot configuration file with GNU Radio support (and dependencies) at https://github.com/oscimp/oscimp_br2_external/tree/master/configs
**GNU Radio 3.8 on the PC**

- Older distributions still provide GNU Radio 3.7 as package (obsolete since Jul. 2019) – current Debian/stable is 3.8
- pybombs allows for multiple version of GNU Radio to be installed on a same computer (including 3.9 and 3.10)
- We will launch `gnuradio-companion` on the PC:
  1. check the packaged version of GNU Radio: `apt-cache policy gnuradio`
  2. if the version is 3.8, install the packaged version
  3. if the version is 3.7, use PyBOMBS (Python Build Overlay Managed Bundle System) as described at [https://github.com/gnuradio/pybombs](https://github.com/gnuradio/pybombs)
     3.1 `sudo apt-get install python3-pip`
     3.2 `sudo pip3 install pybombs`
     3.3 `pybombs auto-config`
     3.4 `pybombs recipes add-defaults`
     3.5 `pybombs prefix init ~/prefix-3.8 -R gnuradio-default` assuming the installation directory is in `$HOME/prefix-3.8`
     3.6 `source ~/prefix-3.8/setup_env.sh` (will have to be repeated in each terminal launching GNU Radio Companion)
     3.7 `gnuradio-companion`
  4. if GNU Radio Companion has started: install osmosdr, either from the binary package `apt` if GNU Radio 3.8 is packaged, or with PyBOMBS (`pybombs install gr-osmosdr`) if this installation system was used
  5. close GNU Radio Companion and launch again (or refresh package list) to access Osmocom Source: the links between blocks must be curves (≥3.8) and not lines at right angles (≤3.7).
GNU Radio on Raspberry Pi4

1. Check that GNU Radio is properly installed: on the RPi4,
   
   ```python3
   import gnuradio
   ```
   
   must return with a prompt and no warning/error

2. basics of GNU Radio flowcharts: one source, digital processing blocks, and sinks with consistent datarate along the processing path (samp_rate/decimation factors)

3. no graphical output in the Raspberry Pi4: launch gnuradio-companion on the PC and select Options → Generate Options → No GUI

4. the Id defines the name of the output Python script

5. Run → Generate to convert the flowgraph in a Python script (see console for output file & path)

6. copy (scp ² the Python script from the PC to the Raspberry Pi4

7. on the Raspberry Pi4, execute with python3 my_script.py

²on the PC: scp my_script.py root@pi4_IP_address:/root
GNU Radio on Raspberry Pi4

1. The trivial flowchart generated on the PC with GNU Radio Companion (3.8 – check the curved connections)
   - 2 audio outputs (GNU Radio inputs) for stereo
   - 48 kS/s sampling rate
   - Sine wave signal source
   - Run on the RPi4 with `python3 pgm.py`

   ⇒ must output a tone on the audio jack

2. FM radio receiver to check proper operation of DVB-T dongle
Streaming from RPi4 to PC

- An instrument collects the data and sends them to a PC for processing
- the PC might not process all data but only segments
- UDP-like Zero-MQ stream: Publish-Subscribe mechanism (supported by Python, GNU/Octave, C, C++ ...)
- demonstration: stream the FM demodulated sound to the PC used as sound card.

On the Raspberry Pi4, fetch samples, demodulate and send

```
48000*24
input rate = samp_rate/6
```

cutoff = samp_rate/12

Adapt IP @ to you embedded board network configuration

---

**Options**
- Title: Not titled yet
- Output Language: Python
- Generate Options: No GUI
- Run Options: Prompt for Exit

**Variable**
- Id: samp_rate
- Value: 1.152M

---

**osmocom Source**
- Sync: Unknown PPS
- Number Channels: 1
- Sample Rate (sps): 1.152M
- Ch0: Frequency (Hz): 96.9M
- Ch0: Frequency Correction (ppm): 0
- Ch0: DC Offset Mode: 0
- Ch0: IQ Balance Mode: 0
- Ch0: Gain Mode: False
- Ch0: RF Gain (dB): 10
- Ch0: IF Gain (dB): 20
- Ch0: BB Gain (dB): 20

---

**Low Pass Filter**
- Decimation: 6
- Gain: 1
- Sample Rate: 1.152M
- Cutoff Freq: 96k
- Transition Width: 48k
- Window: Hamming
- Beta: 6.76

---

**WBFM Receive**
- Quadrature Rate: 192k
- Audio Decimation: 4

---

**ZMQ PUB Sink**
- Address: tcp://1...1.200:5555
- Timeout (msec): 100
- Pass Tags: No

---

Publish Sink: the address tcp://192.168.x.y:5555 is the Raspberry Pi4 Ethernet address (listening to incoming connection requests). The port is a random value > 1024, here 5555.
Streaming from RPi4 to PC

On the PC:

Options
Title: Not titled yet
Output Language: Python
Generate Options: No GUI
Run Options: Prompt for Exit

Variable
Id: samp_rate
Value: 48k

ZMQ SUB Source
Address: tcp://1...1.200:5555
Timeout (msec): 100
Pass Tags: No

Audio Sink
Sample Rate: 48k

IP is the embedded board network address

set the Subscribe Source to the tcp://192.168.x.y:5555 IP address of the Raspberry Pi4, same port as before.
Commands from PC to RPi4

Multithreaded Python script approach

- GNU Radio Companion is a Python script generator
- GNU Radio Companion 3.8 allows for inserting additional Python commands in its initialization code: **Python Snippets**
- GNU Radio Companion 3.8 allows for adding Python functions: **Python Module**
- Launch a separate thread running a TCP (connected mode) server
- Receive commands from the PC running a TCP client (**telnet**)
- Tune the GNU Radio flowgraph variables by calling the callback function associated with the modified variable
- Alternate ZeroMQ solution: REQ/REP (Request/Reply)

**What is a thread?**

- function run in parallel to the main program but sharing the same memory space

```python
import threading
import time

def jmf1(argument):
    while True:
        print(argument)
        time.sleep(1)

threading.Thread(target=jmf1, args=(1,)).start()
threading.Thread(target=jmf1, args=(2,)).start()
threading.Thread(target=jmf1, args=(3,)).start()
```

- make sure to kill/quit the thread before leaving the Python script (**self.my_status=...**)
What is a server?

Definition: a server waits for a connection, a client connects to the server when it needs information.

```python
import socket
import string
while True:
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(('127.0.0.1', 4242))
    print("Waiting for connection")
    sock.listen(1)
    conn, addr = sock.accept()
    with conn:
        print(\'connected from \', addr)
        while True:
            data = conn.recv(1)
            if data:
                data = data.decode()
                print(data)
                if \'q\' in data:
                    sock.shutdown(socket.SHUT_RDWR)
                    sock.close()
                    break

Run python3 my_server in one terminal
Run telnet localhost 4242 in another terminal
Enjoy ... quit by sending \'q\'
```

³ availability of the selected port can be checked using nmap localhost which lists ports used by running services
Putting it all together ...

Python Snippet executes the thread including the Python Module running the TCP server controlling the GNU Radio execution by tuning parameters with the associated callback function.

---

J.-M. Friedt, W. Feng. Analyse et réalisation d’un RADAR à synthèse d’ouverture (SAR) par radio logicielle (2/3), GNU/Linux Magazine France 242 (Nov. 2020)
Commands from PC to RPi4

- Demonstrate how you modify the previous flowchart, streaming the output of the FM demodulator to the PC, to tune the broadcast station frequency on the Raspberry Pi4 from the PC.
- Provide a graphical user interface allowing to enter the FM radio frequency and transferring the information to the RPi4.

Why GNU Radio > 3.8?

Link between Python and C++:

- ≤3.8: SWIG (Simplified Wrapper and Interface Generator) – runtime errors (dynamic library linking issues)
- ≥3.9: C++(11/14) bindings through Pybind11 – compilation errors

---

5D. Estévez, GNU Radio 3.9 in Buildroot at https://destevez.net/2021/10/gnu-radio-3-9-in-buildroot/