Software Defined Modem basic setup guide

Authors: *Filippo Campagnaro, Antonio Montanari, Roberto Francescon, Emanuele Coccolo*

Wireless and More srl and Dept. of Information Engineering, University of Padova, Italy
`https://www.wirelessandmore.it`

Contents

1 Overview

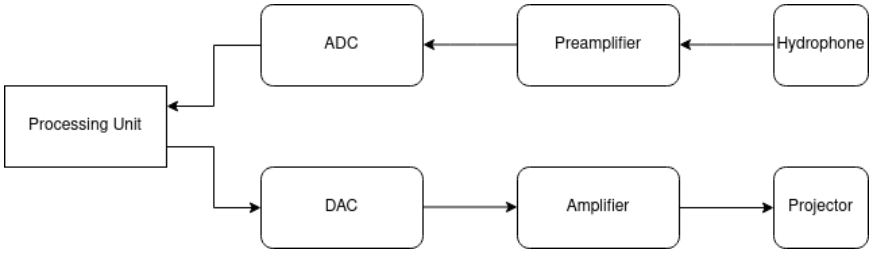

Figure 1: Hardware block diagram.

The general hardware configuration needed for this Software Defined Modem (SDM) is depicted in Figure 1. A processing unit performs the signal processing and transmits the modulated signal through a digital to analog converter (DAC) to a signal amplifier, that is connected with an acoustic projector. The same processing unit samples the pre-amplified signal received by a hydrophone using an analog to digital converter (ADC). All signal processing is performed in software using a combination of C and C++ libraries, hence developing a modular Software Defined Modem. With this approach, the signal

processing unit and DAC/ADC can be changed without any need to adapt the code: for instance, in Figure 2 a first transmission was performed using a Linux PC as the processing unit and its 192 kHz internal sound card DAC and ADC, while in Figure 3 the same transmission was performed using the Linux PC on Board (PCB) Raspberry Pi 4 with a high quality sound card. Alternatively, for better performing modems at the cost of more expensive equipment, the hardware suggested at these links can be used: `https://www.benthowave.com/products/BII-8080TR.html` and `https://www.benthowave.com/products/BII-2100TRSwitch.html`. In this case the only needed device is the processing unit (e.g., a laptop).

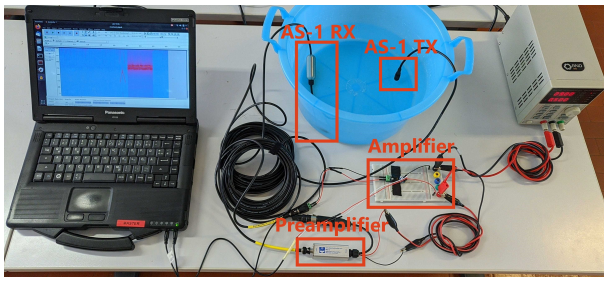

Figure 2: Test components with a PC running a software defined modem using Hi-Fi integrated sound cards as ADC and DAC.

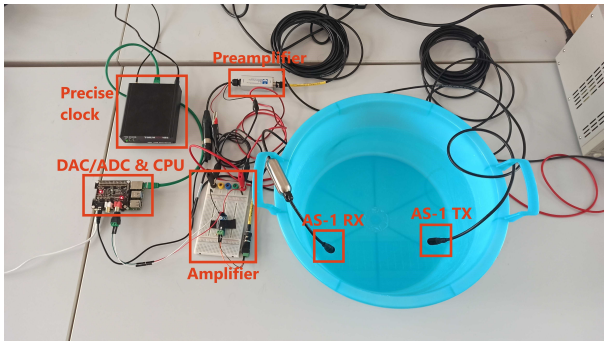

Figure 3: Test components of the modems with a Raspberry Pi 4-B running a software defined modem using HiFiBerry DAC+ ADC Pro and a TM2000 OCXO-based time server.

In this guide we present the Raspberry Pi 4 model B configuration, being it the cheapest we tested in house, and identifying it as the simplest to setup for Do It Yourself (DIY) practitioners and University projects.

In general, the following is the list of the hardware components needed.

- **Processing unit** The choice of the processing unit is crucial: enough computational power is necessary for reaching a given target performance. For our development, we chose the **Raspberry Pi 4 Model B** [1], with 8 GB of memory. Its powerful Quad core Cortex-A72 (ARM v8) processor allows to run quite all the most common modulation schemes and Forward Error Correction (FEC) algorithms found in the literature. Other processing boards can be employed such as other Raspberry Pi 4 Model B versions. The Raspberry Pi 3+, Tinker Board S or similar PCB should work, but were not tested. This choice affects (and is affected by) also the HiFi audio board: default compatibility will make the work that follows a lot easier.

- **HiFi audio board** Responsible for both ADC and DAC, the choice of the audio board can enable the possibility to implement more advanced settings in terms of complexity and performance. Given our choice of the Raspberry Pi 4 as processing board, we opted for the **HiFiBerry DAC+ ADC**

Pro [2] audio processing board which is by default compatible with the Raspberry Pi and easily integrated. Other Raspberry Pi sound cards HAT, with 192 kHz and cut-down frequency above 70 kHz, could be employed but were not tested.

- **Pre-amplifier** The pre-amplifiers **Aquarian PA-4** [3] with external power or **Aquarian PA-6** [4] with phantom power adapter to external power can be employed. Be aware that the adapter to 3.5 mm audio jack is needed.
- **Amplifier** you can use whatever audio amplifier with no cutoff freq above 20kHz. We used a LM3886TF-based class AB audio amplifier DIY kit available here: `https://www.ebay.it/itm/184253045896?hash=item2ae6565c88:g:1-sAAOSw37VelX5m`.
- **Transducers** For converting the electrical signal to sound wave, we chose to deploy the **Aquarian Scientific AS-1** hydrophone [5]. You can chose other transducers, depending on the frequencies you want your modem to operate. We opted for a high frequency modem operating at carrier above $f_c = 45$ kHz. Other transducers can be used as well, with different frequency responses, in the case lower transmission frequencies are desired.

2 Tested setup

The components we tested are described in following.

- **2 Raspberry Pi 4 Model B** [1]: we used the 8 GB version. Also the other Raspberry Pi 4 Model B versions should be OK and with Raspberry Pi 3+, Tinker Board S or similar PCB should work, but were not tested.
- **2 HiFiBerry DAC+ ADC Pro** [2] or other Raspberry Pi sound card HAT with 192 kHz and cut-down frequency above 70 kHz (not tested with other Raspberry Pi sound card HAT).
- **2 Hydrophone AS-1** [5] for high frequency transmissions (above 45 kHz): other transducers can be used as well in the case lower transmission frequencies are desired (e.g., `https://www.piezohannas.com/38kHz-Hydrophone-Communication-Underwater-Transducer-for-Positioning-Transponder-pd42763969.html` and `https://docs.unavlab.com/underwater_acoustic_antennas_en.html`).
- **Preamplifier PA-4** [3] with external power or **Preamplifier PA-6** [4] with phantom power adapter to external power. Adapter to 3.5 mm audio jack is needed.
- **Power amplifier** We used a LM3886TF-based class AB audio amplifier DIY kit available here: `https://www.ebay.it/itm/184253045896?hash=item2ae6565c88:g:1-sAAOSw37VelX5m`.
- **Precise clock (optional)** in the case one-way travel time ranging (OWTT) features is desired, a precise clock such as a miniature atomic clock or an oven-controlled crystal oscillator (OCXO) need to be acquired, the former suitable to expensive deployments where the ranging precision is crucial and need to be kept for long time (months), and the latter suitable for short (up to a few days) and lower cost deployment. To simplify the integration, an OCXO-based NTP server can be used: `https://timemachinescorp.com/product/gps-ntpptp-network-time-server-tm2000` or `https://timemachinescorp.com/product/gps-ntpptp-network-time-server-10mz-output-tm2500`.

Note: the external sound card HAT and specific amplifiers and preamplifiers are needed only because the AS-1 transducer operates in the bandwidth between 45 kHz to 100 kHz, hence to at least be able to transmit in the bandwidth from 45 to almost 90 kHz we need a sound card with at least 192 kHz sampling rate. In the case a transducer operating in the audible bandwidth (i.e., with maximum frequency below

20 kHz) is used, a normal sound card and whatever audio amplifier are enough. The modem can also be tested in air using audible frequencies with normal speakers and microphones.

In the reminder of this user-guide we describe the Raspberry Pi 4 configuration with AS-1 Hydrophones and HifiBerry DAC+ ADC Pro.

2.1 Setup Procedure

1. connect the HiFiBerry board on top of the Rasp HAT using the provided spacers
2. prepare two cables to connect the HiFiBerry to the TX/RX devices with the following connectors:
   (a) RCA to be connected to RAC red for TX;
   (b) mini-JACK 3.5mm to be connected on microphone port for RX
3. RX amplifier: the PA-4 amplifier has a BNC input directly connected to the hydrophone, the XLR output is connected via an adapter to the 3.5mm mini-jack connector of the HiFiBerry while the power supply cable can be connected to a 9 or 12V battery for power supply.
4. the TX amplifier has three ports that have to be connected respectively:
   - input: to a RCA output of the HiFiBerry
   - output: to the hydrophone
   - power supply: to a dual (-12V - 0V - 12V) power supply which can be a series of two 12V batteries with the center tap connected to the GND pin.
5. (optional) precise clock: connect to PPS output port of the clock to GPIO4 and a ground reference to one GROUND GPIO (e.g. bottom right)

3 Software setup

In the following section we describe the software tools that are necessary to run the SDM along with some useful tools that can be employed to get more advanced features. We used a Raspberry Pi 4 with RaspberryPiOS Operative System [6]. The system will be using ALSA as audio interface.

3.1 ALSA Installation

We are using the Advanced Linux Sound Architecture (ALSA) [7] as audio interface, to send audio samples to the audio output. To install it, execute the following command

```
sudo apt install libasound2-dev
```

3.2 FFTW3 Installation (optimized FFT library)

We are making use of advanced library for optimized Fourier Transform, installable with the following command

```
sudo apt install libfftw3-dev
```

3.3 FEC (KA9Q's optimized library for Convolutional and RS coding scheme, and dotproduct

This package provides a set of functions that implement several popular forward error correction (FEC) algorithms and several low-level routines useful in modems implemented with digital signal processing (DSP).

```
git clone https://github.com/quiet/libfec.git
cd libfec/
./configure
make
sudo make install
```

3.4 PPS-Client (synchronizer for internal clock using PPS signal)

This operation is needed only in the case you want to keep the clock of the Raspberry synchronized with an external high precision clock that provides 1PPS interface to, for instance, perform OWTT or use a time division multiple access (TDMA) Multiple Access Control (MAC) scheme. If this is not the case, you can skip this point.

```
sudo nano /boot/config.txt
```

Enable the GPIO4 on the Rasp, adding this to the bottom of the configuration (uncommenting the entry or adding this to the existing entry, **under [pi4] section (!!!)**)

```
dtoverlay=pps-gpio,gpiopin=4
```

(if not already installed)

```
sudo apt install git pps-tools build-essential

git clone https://github.com/rascol/PPS-Client.git
cd PPS-Client
./configure
./create-backups
make
sudo make install
sudo reboot
```

3.5 ALSA Configuration

To enable and configure the ALSA library for the HiFiBerry audio card, you need to setup the following files. If the files does not exist, proceed to create it.

```
sudo touch /etc/asound.conf
sudo nano /etc/asound.conf
```

The following is an example of configuration:

```
pcm.!default {
 rate 192000
 type hw
 card 0
 device 0
```

```
}
ctl.!default {
 type hw card 0
}
```

3.6 HiFiBerry Configuration

To enable and configure the HiFiBerry audio card, the following file needs to be modified:

```
sudo nano /boot/config.txt
```

Comment the following line:

```
dtparam=audio=on
```

In [pi4] section, add this overlay:

```
dtoverlay=vc4-fkms-v3d
```

In [all] section, add these lines:

```
dtoverlay=hifiberry-dacplusadcpro
force_eeprom_read=0
```

and comment (if present)

```
dtoverlay=vc4-fk
```

Reboot the board.
To test if the configuration is complete, execute the command

```
aplay -l
```

And the output should be:

```
**** List of PLAYBACK Hardware Devices ****
card 0: sndrpihifiberry [snd_rpi_hifiberry_dacplusadcpro], device 0:
    HiFiBerry DAC+ADC Pro HiFi multicodec-0 [HiFiBerry DAC+ADC Pro
   HiFi multicodec-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
```

Check whether all the related interfaces are present (e.g. `hw`, `plughw`, ...) by executing the following command

```
aplay -L
```

3.7 ALSAMIXER configuration for HiFiBerry DAC+ ADC Pro

On a terminal, type

```
alsamixer
```

F6 (to change card) and select `snd_rpi_hifiberry_dacplusadcpro`

Navigate with left/right arrow keys and change parameters with up/down arrow keys, F3 for Playback tab and F4 for Capture tab.

In Playback(F3) check that the following parameters are set:

- ADC Left: [VINL1[SE]]

- ADC Right: [VINR1[SE]]
- Analogue [db gain: 0.00, 0.00] (bar to 100)
- Analogue Playback Boost [db gain: 0.00, 0.00]

In Capture(F4) check that the following parameters are set:

- ADC [db gain: 0.00, 0.00] (bar to 9)
- Analogue Playback Boost [db gain: 0.00, 0.00]
- Max Overlcock DAC: 0
- Max Overlcock DSP: 0
- Max Overlcock PLL: 0

References

[1] Raspberry Pi, *Buy a Raspberry Pi 4 Model B*, Accessed January, 2023. `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/`.

[2] HiFiBerry, *DAC+ ADC Pro*, Accessed January, 2023. `https://www.hifiberry.com/shop/boards/hifiberry-dac-adc-pro/`.

[3] Aquarian Hydrophones, *PA-4*, Accessed January, 2023. `https://www.aquarianaudio.com/pa4.html`.

[4] Aquarian Hydrophones, *PA-6*, Accessed January, 2023. `https://www.aquarianaudio.com/pa6.html`.

[5] Aquarian Hydrophones, *AS-1*, Accessed January, 2023. `https://www.aquarianaudio.com/as-1-hydrophone.html`.

[6] Raspberry Pi, *Raspberry Pi OS*, Accessed January, 2023. `https://www.raspberrypi.com/software/`.

[7] Advanced Linux Sound Architecture (ALSA), *Project Homepage*, Accessed January, 2023. `https://alsa-project.org/wiki/Main_Page`.