Lab 4 - QEMU system emulation and SDR software

SDR software and Emulation

This lab involves installing software defined radio (SDR) software on an emulated ARM processor. While not exactly the same as the hardware platform, the qemu emulation running Debian is very similar to the Beaglebone. It provides a way to work on the software without the hardware. We will also install quisk, a SDR application which will allow you to operate your radio (but it eventually needs to be executed on your Beaglebone).

After completing this lab, you should realise that you can use the sox record program to capture i/q samples from your radio, convert to the right format for the `k9an-wsprd` program, and decode wspr.

A few hints.

- Note that although this lab is not particularly difficult, In particular, SoapySDR (Part 3) takes a long time to compile in qemu.
- I used 8GB of memory and 16GB of disk for the Ubuntu VM in Part 1 (minimal installation option).
- Make copies of your intermediate qcow2 files so you don't lose everything if it crashes or hangs. You should first shutdown qemu (`sudo shutdown now`) before you make the backup.

1. Laboratory Experiment

Part 1 - Linux VM/Image (30%)

The first task is that you really need a Linux distribution to run qemu and all the tools properly. Most developers targetting embedded Linux devices use Linux hosts. The most straightforward way is to use a separate Linux partition on your computer or use the **Ubuntu live USB (https://ubuntu.com/tutorials/try-ubuntu-before-you-install#1-getting-started)** . Make sure you install Ubuntu 20.04 LTS as older versions use an out of date version of qemu.

It is possible to do this lab under Windows 10 using the native qemu, but your life will be much simpler if you use Linux as you will need other things like an X windows client. Better alternatives are to run Ubuntu in Hyper-V under Windows 10 Pro (but not Windows 10 home as it does not support Hyper-V). Other virtualisation software for both MacOS and Windows 10 include VMWare and VirtualBox. I have completed this lab using VirtualBox on a Mac.

The reason we use Ubuntu on the host for a Debian target is that Ubuntu has a more frequent update schedule and a more polished desktop environment. Internally they share the same package manager.

Part 2 - Running WSPR decoder under QEMU (40%)

The qemu files created through the process described in the lecture (but changed to aarch64) is available **here (https://drive.google.com/drive/folders/1uZm11EyFzMsVGCJ4XT9Qp1FZZPAhDKT6?usp=sharing)**. Note this this image is simply the installed Debian and does not include any other packages (such as the last step involving sudo and putting elec3607 in the sudo group). You will need to install qemu-system-arm and possibly other packages. First make a copy of the qemu image and execute qemu using the command below.

```
$ cp debian-3607-aarch64-initial.qcow2 debian-3607-aarch64.qcow2
$ qemu-system-aarch64 -M virt -cpu cortex-a53 -m 1G -initrd initrd.img-4.19.0-16-arm64 -kerne
l vmlinuz-4.19.0-16-arm64 \
   -append "root=/dev/vda2 console=ttyAMA0" -drive if=virtio,file=debian-3607-aarch64.qcow2,f
ormat=qcow2,id=hd \
   -net user,hostfwd=tcp::10022-:22 -net nic -nographic -device intel-hda -device hda-duplex
```

After booting, you will see a login prompt. Use elec3607/elec3607 as the username/password. In the emulation, install git and clone **https://github.com/phwl/elec3607-labquestions.git (https://github.com/phwl/elec3607-labquestions.git)**. After you figure out all the packages that are needed and write them down in your lab book, you should be rewarded with the following output:

```
elec3607@debian:~$ cd elec3607-labquestions/labs/lab4-qemu-wspr/
elec3607@debian:~/elec3607-labquestions/labs/lab4-qemu-wspr$ make wspr
(cd wsprcan; make)
make[1]: Entering directory '/home/elec3607/elec3607-labquestions/labs/lab4-qemu-wspr/wsprcan'
gcc -c -o wsprd.o wsprd.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-math
gcc -c -o wsprsim_utils.o wsprsim_utils.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic
-O3 -ffast-math
gcc -c -o wsprd_utils.o wsprd_utils.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3
-ffast-math
wsprd_utils.c: In function ‘unpackpfx’:
```

```
wsprd_utils.c:162:21: warning: comparison is always true due to limited range of data type [-
Wtype-limits]
             if( (nc >= 0) & (nc <= 9) ) {
                     ^~
wsprd_utils.c:181:17: warning: comparison is always true due to limited range of data type [-
Wtype-limits]
         if( (nc >= 0) & (nc <= 9) ) {
                 ^~
wsprd_utils.c:175:9: warning: ‘strncat’ specified bound 1 equals source length -Wstringop-
overflow=]
         strncat(call,"/",1);
         ^~~~~~~~~~~~~~~~~~
wsprd_utils.c:184:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call,"/",1);
             ^~~~~~~~~~~~~~~~~~
wsprd_utils.c:190:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call,"/",1);
             ^~~~~~~~~~~~~~~~~~
wsprd_utils.c:197:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call,"/",1);
             ^~~~~~~~~~~~~~~~~~
wsprd_utils.c: In function ‘unpk_’:
wsprd_utils.c:274:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call_loc_pow," ",1);
             ^~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:276:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call_loc_pow," ",1);
             ^~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:291:13: warning: ‘strncat’ specified bound 1 equals source length [-Wstringop-
overflow=]
             strncat(call_loc_pow," ",1);
             ^~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:326:9: warning: ‘strncat’ specified bound 1 equals source length -Wstringop-
overflow=]
         strncat(call_loc_pow," ",1);
         ^~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:328:9: warning: ‘strncat’ specified bound 1 equals source length -Wstringop-
overflow=]
```

```
        strncat(call_loc_pow," ",1);
        ^~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c: In function ‘unpackpfx’:
wsprd_utils.c:198:13: warning: ‘strncat’ output may be truncated copying 2 bytes from a string
of length 3 [-Wstringop-truncation]
            strncat(call,pfx,2);
            ^~~~~~~~~~~~~~~~~~~
wsprd_utils.c:191:13: warning: ‘strncat’ output may be truncated copying 1 byte from a string of
length 3 [-Wstringop-truncation]
            strncat(call,pfx,1);
            ^~~~~~~~~~~~~~~~~~~
wsprd_utils.c:185:13: warning: ‘strncat’ output may be truncated copying 1 byte from a string of
length 3 [-Wstringop-truncation]
            strncat(call,pfx,1);
            ^~~~~~~~~~~~~~~~~~~
wsprd_utils.c:176:9: warning: ‘strncat’ output truncated before terminating nul copying as many
bytes from a string as its length [-Wstringop-truncation]
        strncat(call,tmpcall,strlen(tmpcall));
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c: In function ‘unpk_’:
wsprd_utils.c:325:9: warning: ‘strncat’ output truncated before terminating nul copying as many
bytes from a string as its length [-Wstringop-truncation]
        strncat(call_loc_pow,callsign,strlen(callsign));
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:327:9: warning: ‘strncat’ output truncated before terminating nul copying as many
bytes from a string as its length [-Wstringop-truncation]
        strncat(call_loc_pow,grid6,strlen(grid6));
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:329:9: warning: ‘strncat’ output may be truncated copying 2 bytes from a string of
length 2 [-Wstringop-truncation]
        strncat(call_loc_pow,cdbm,2);
        ^~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:290:13: warning: ‘strncat’ output truncated before terminating nul copying as many
bytes from a string as its length [-Wstringop-truncation]
            strncat(call_loc_pow,callsign,strlen(callsign));
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:292:13: warning: ‘strncat’ output may be truncated copying 2 bytes from a string
of length 2 [-Wstringop-truncation]
            strncat(call_loc_pow,cdbm,2);
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:273:13: warning: ‘strncat’ output truncated before terminating nul copying as many
bytes from a string as its length [-Wstringop-truncation]
            strncat(call_loc_pow,callsign,strlen(callsign));
```

```
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:275:13: warning: ‘strncat’ output may be truncated copying 4 bytes from a string
of length 4 [-Wstringop-truncation]
            strncat(call_loc_pow,grid,4);
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprd_utils.c:277:13: warning: ‘strncat’ output may be truncated copying 2 bytes from a string
of length 2 [-Wstringop-truncation]
            strncat(call_loc_pow,cdbm,2);
            ^~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -c -o tab.o tab.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-math
gcc -c -o fano.o fano.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-math
gcc -c -o jelinek.o jelinek.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-
math
gcc -c -o nhash.o nhash.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-math
nhash.c: In function ‘nhash’:
nhash.c:351:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
     case 12: c+=((uint32_t)k[11])<<24;
              ~^~~~~~~~~~~~~~~~~~~~~~~
nhash.c:352:5: note: here
     case 11: c+=((uint32_t)k[10])<<16;
     ^~~~
nhash.c:352:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
     case 11: c+=((uint32_t)k[10])<<16;
              ~^~~~~~~~~~~~~~~~~~~~~~~
nhash.c:353:5: note: here
     case 10: c+=((uint32_t)k[9])<<8;
     ^~~~
nhash.c:353:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
     case 10: c+=((uint32_t)k[9])<<8;
              ~^~~~~~~~~~~~~~~~~~~~
nhash.c:354:5: note: here
     case 9 : c+=k[8];
     ^~~~
nhash.c:354:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
     case 9 : c+=k[8];
              ~^~~~~~
nhash.c:355:5: note: here
     case 8 : b+=((uint32_t)k[7])<<24;
     ^~~~
nhash.c:355:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
     case 8 : b+=((uint32_t)k[7])<<24;
              ~^~~~~~~~~~~~~~~~~~~~~~~
nhash.c:356:5: note: here
```

```
    case 7 : b+=((uint32_t)k[6])<<16;
    ^~~~
nhash.c:356:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 7 : b+=((uint32_t)k[6])<<16;
             ~^~~~~~~~~~~~~~~~~~~~~~
nhash.c:357:5: note: here
    case 6 : b+=((uint32_t)k[5])<<8;
    ^~~~
nhash.c:357:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 6 : b+=((uint32_t)k[5])<<8;
             ~^~~~~~~~~~~~~~~~~~~~~
nhash.c:358:5: note: here
    case 5 : b+=k[4];
    ^~~~
nhash.c:358:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 5 : b+=k[4];
             ~^~~~~~
nhash.c:359:5: note: here
    case 4 : a+=((uint32_t)k[3])<<24;
    ^~~~
nhash.c:359:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 4 : a+=((uint32_t)k[3])<<24;
             ~^~~~~~~~~~~~~~~~~~~~~~
nhash.c:360:5: note: here
    case 3 : a+=((uint32_t)k[2])<<16;
    ^~~~
nhash.c:360:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 3 : a+=((uint32_t)k[2])<<16;
             ~^~~~~~~~~~~~~~~~~~~~~~
nhash.c:361:5: note: here
    case 2 : a+=((uint32_t)k[1])<<8;
    ^~~~
nhash.c:361:15: warning: this statement may fall through [-Wimplicit-fallthrough=]
    case 2 : a+=((uint32_t)k[1])<<8;
             ~^~~~~~~~~~~~~~~~~~~~~
nhash.c:362:5: note: here
    case 1 : a+=k[0];
    ^~~~
gcc -o k9an-wsprd wsprd.o wsprsim_utils.o wsprd_utils.o tab.o fano.o jelinek.o nhash.o -
I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-math -L/usr/local/lib  -lfftw3 -
lm
gcc -c -o wsprsim.o wsprsim.c -I/usr/local/include -Wall -Wextra -std=c99 -pedantic -O3 -ffast-
math
```

```
wsprsim.c: In function ‘main’:
wsprsim.c:171:9: warning: ‘strncpy’ specified bound 23 equals destination size -Wstringop-
truncation]
         strncpy(message,argv[optind],23);
         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
wsprsim.c:157:17: warning: ‘strncpy’ specified bound 16 equals destination size [-Wstringop-
truncation]
                 strncpy(c2filename,optarg,16);
                 ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~
gcc -o wsprsim wsprsim.o wsprsim_utils.o wsprd_utils.o tab.o fano.o nhash.o -I/usr/local/include
-Wall -Wextra -std=c99 -pedantic -O3 -ffast-math -L/usr/local/lib  -lfftw3 -lm
make[1]: Leaving directory '/home/elec3607/elec3607-labquestions/labs/lab4-qemu-wspr/wsprcan'
sox data/iq-16b.wav -c 1 -t wav -r 12000 -b 16 mono.wav
wsprcan/k9an-wsprd mono.wav
mono  -1 -1.3   0.001437 -1  VK2RG QF56 30
mono -19 -1.0   0.001455 -1  VK3GOD QF23 23
mono -20 -1.1   0.001478 -1  VK4YEH QG62 37
<DecodeFinished>
rm mono.wav
```

What this did was:

- First compile the program inside the wsprcan directory. Note all of the warnings, this is sloppy programming and they all should be corrected (in fact the program is not well written at all). The resulting binaries are `k9an-wsprd` and `wsprsim.`
- Use the sox program to translate the i/q outputs of the SDR (obtained by recording my ELEC3706-sdr into the file iq-16b.wav), into a mono file (`mono.wav`). Note that this results in a degradation of the signal to noise ratio of the audio output, instead one should use the **phasing method (https://www.dsprelated.com/showarticle/176.php)** to do SSB demodulation.
- Execute the `k9an-wspr` program on the `mono.wav` file.

Part 3 - Installing SoapySDR and quisk (30%)

Quisk is a software defined radio software program. It uses the SoapySDR library as the interface to the radio. In this part of the lab, don't use the `apt` installer as the excercise is to compile the code from source.

First install SoapySDR by following these instructions:
https://github.com/pothosware/SoapySDR/wiki#installation
(https://github.com/pothosware/SoapySDR/wiki#installation)

Then install the Python 3 version of quisk: **https://james.ahlstrom.name/quisk/ (https://james.ahlstrom.name/quisk/)** .

Once you have done this, you can ssh into your qemu virtual machine (from another terminal window on the host machine) and execute quisk. Under Linux desktop, a window with the quisk software should appear. Quisk will be a useful tool for debugging your radio.

```
ssh -Y vmuser@localhost -p10022
```

```
quisk
```

The display should be similar to below. Don't worry about the ALSA issues, they need to be addressed by installing and configuring libasound2 (which we will do later).