Warsaw, 14.05.2021

Linux for Embedded Systems - Laboratory ex. 4

Emilia Wróblewska 291674

Problem Description

The subject of the assignment 4 is the implementation with the emulated Virt 64 board, a device equipped with two forms of user interface:

> **1)** simple buttons and LEDs should be used to control basic functions;
>
> **2)** web interface (or other network interface) should be used to control advanced functions;
>
> **3)** The design should include support for additional equipment connected to the board. In the simplest version it can be an emulated sound card. In a better version it can be an USB sound card, camera or other device forwarded from the host. Please remember that QEMU allows you to share USB devices with the emulated board.

My chosen topic:

> **E)** Using the emulated or forwarded sound card, please build the music player. The local buttons and LEDs should be used to switch on/off the player, to select the song and to control volume. The web interface should be used to configure the list songs, to download and to remove the song.

Procedure to recreate the design from the archive

Firstly, navigate into the “QemuVirt64” directory and run “build.sh” script to build the image of the system. Now to start the solution, firstly run the “run_before” script to launch the GUI and then in another terminal “runme” script to start the machine.
The player starts automatically and the web server is available on the localhost:2222.

For the controls:

Buttons in GUI:

12 – Previous song
13 – Pause/Play (shining LED 24 indicates that the song is paused)
14 – Next song
15 – Volume down
16 - Volume up

Web server:

- "Upload song" redirects the user to a page for uploading files. This way one can add a song to the playlist.
- Panel with buttons which have exactly the same functionalities as GUI buttons.
- Input field where one should enter the whole name of the song file. When the input is filled with appropriate name, one can click:
    - "Download" to download this song.
    - "Move Up" to move this song one place up the playlist.
    - "Move Down" to move this song one place down the playlist.
    - "Delete" to delete this song from the playlist.

(Note: The buttons in GUI react rather slowly, so sometimes the buttons should be pressed a little longer - for about 0.5 second.)

Description of the solution

I used the project from the "QemuVirt64" folder downloaded from https://github.com/wzab/BR_Internet_Radio/tree/gpio_simple_2021.02 as a starting point. To the system I added the following packages:

Target packages → Audio and video applications → alsa-utils → all packages
Target packages → Audio and video applications → mpd
Target packages → Audio and video applications → mpd → All converter plugins
Target packages → Audio and video applications → mpd → All decoder plugins
Target packages → Audio and video applications → mpd → Input plugins - curl
Target packages → Audio and video applications → mpd → Output plugins - alsa
Target packages → Audio and video applications → mpd → Output plugins - tcp sockets
Target packages → Audio and video applications → mpd-mpc
Target packages → Interpreter languages and scripting → python3 → External python modules → python-flask
Target packages → Interpreter languages and scripting → python3 → core python3 modules → readline, ssl, sqlite, xml, xz, zlib
Target packages → Shell and utilities → screen

Now in the “overlay/opt” folder I added my scripts: “music_player.py” which handles interaction with GUI buttons and “webservice.py” which implements the python server. I based my code on my previous solutions, so the transformation I made from the C functions handling “bounce effect” to python may cause them to not be completely reliable. I observed that one fast click on a GUI button is too fast for my algorithms, the buttons should be held for around 0.5 second to properly react.

Then I added a “S99musicplayer” script into “/etc/init.d” directory to automatically launch the player and web server at the start of the machine:

```
#!/bin/sh
alsactl init
sleep 3

mpc update
mpc clear
mpc add /
mpc repeat 1
screen -m -d /usr/bin/python3 /opt/music_player.py
screen -m -d /usr/bin/python3 /opt/webservice.py
```

The “sleep 3” command is added to delay slightly the execution of the music_player.py, since without it the mpd wasn’t fully initialized before the scripts were trying to connect to the mdp client. This caused me the error: “MPD error: Connection refused”.

Finally, I added some mp3 samples into “overlay/var/lib/mpd/music”.

The last step was to modify the “runme” script to forward an emulated sound card to the QEMU:

```
#!/bin/bash
(
 cd buildroot-2021.02
 ./output/host/bin/qemu-system-aarch64 \
 -M virt -cpu cortex-a53 -nographic -smp 1 \
```

```
 -m 1024M \
 -kernel output/images/Image -append "rootwait root=/dev/vda console=ttyAMA0" \
 -netdev user,id=eth0,hostfwd=tcp::8888-:8810,hostfwd=tcp::2222-:22 \
 -device virtio-net-device,netdev=eth0 \
 -drive file=output/images/rootfs.ext2,if=none,format=raw,id=hd0 \
 -device virtio-blk-device,drive=hd0 \
 -soundhw hda -audiodev id=pa,driver=pa \
 -device virtio-rng-device
)
```

Now after executing make, I can start my system using “run_before” and “runme” and the music player will start after boot. I can access my server at localhost:2222. (I used the old syntax for the sound card because for some unknown reasons my machine couldn’t find the sound card when I added new syntax to the script - I got the message that “No sound card found”.)