Warsaw, 28.05.2021

Linux for Embedded Systems - Laboratory ex. 5

Emilia Wróblewska 291674

Problem Description

Exercise 5 should be just a port of the system from exercise 4 to another Embedded Linux framework - Yocto or OpenWRT (I'd rather suggest OpenWRT).

The subject of the assignment 4 was the implementation with the emulated Virt64 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 of songs, to download and to remove the song.

Procedure to recreate the design from the archive

Firstly, run “prepare.sh” which builds the system image, downloads necessary OpenWRT components and starts the python HTTP server as the last command. Then go to “./QemuVirt64” and start OpenWRT with „run_openwrt”. After the start of the system, fix the network configuration by:

```
# vi /etc/config/network (and changing the file „config
interface lan” section as below)
config interface 'lan'
     option ifname 'eth0'
     option proto 'dhcp'
# /etc/init.d/network restart
```

Then, download the “prepare_openwrt.sh” script by

```
# wget http://10.0.2.2:8000/prepare_openwrt.sh
```

Next run it with:

```
# chmod 777 prepare_openwrt.sh
# ./prepare_openwrt.sh
```

Now power off the machine and start the system again with „run_before” and „run_openwrt”. The music player starts automatically and the server is available on localhost:2222. The controls work the same as described in the previous report.

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. I built the image of the system with the “build.sh” script.

After that I downloaded the necessary components of OpenWRT:

```
wget https://downloads.openwrt.org/releases/19.07.7/targets/armvirt/
64/openwrt19.07.7-armvirt-64-Image

wget https://downloads.openwrt.org/releases/19.07.7/targets/armvirt/
64/openwrt19.07.7-armvirt-64-Image-initramfs

wget https://downloads.openwrt.org/releases/19.07.7/targets/armvirt/
64/openwrt19.07.7-armvirt-64- root.ext4.gz

wget https://downloads.openwrt.org/releases/19.07.7/targets/armvirt/
64/openwrt-sdk-19.07.7-armvirt-64_gcc-7.5.0_musl.Linux-x86_64.tar.xz
```

Next, I unpacked the filesystem image and increased the size of root.ext4.

```
gzip -c -d openwrt-19.07.7-armvirt-64-root.ext4.gz > root.ext4
truncate -s \>512M root.ext4
/sbin/resize2fs root.ext4
```

After that, I unpacked the SDK. I downloaded and unpacked the examples from the lecture and I added the repositories with packages to the SDK.

```
tar -xJf openwrt-sdk-19.07.7-armvirt-64_gcc-7.5.0_musl.Linux-x86_64.tar.xz
tar xjf L8_examples.tar.bz2
cd openwrt-sdk-19.07.7-armvirt-64_gcc-7.5.0_musl.Linux-x86_64
```

Then, I adjusted the path at the end of the "feeds.conf.default" by adding "src-link lines /absolute/path/to/L8_examples" (in my case the path was "/home/mini/lab5/M"). Then I configured the SDK and compiled the GPIO driver.

```
export LANG=C
./scripts/feeds update lines
./scripts/feeds install -p lines -a
make menuconfig
#(In the menuconfig make sure that drv-mpc8xxx is selected as
# a package. Save configuration at the end.)
make package/drv-mpc8xxx/compile
```

Next step was to check if the gpio works correctly, so I ran the HTTP server in the "openwrt-sdk-19.07.7-....Linux-x86_64/bin/targets/armvirt/64/packages" directory to host the driver package locally. Then, in the "QemuVirt64" folder I started the GUI with the „run_before" script and I created the „run_openwrt" script to start openWRT (I adjusted the path for kernel).

```
#!/bin/bash
(
 ./buildroot-2021.02/output/host/bin/qemu-system-aarch64 -M virt \
 -m 1024M \
 -cpu cortex-a53 -nographic -smp 1 \
 -kernel ../openwrt-19.07.7-armvirt-64-Image \
 -append "rootwait root=/dev/vda console=ttyAMA0" \
 -netdev user,id=eth0,hostfwd=tcp::8888-:80,hostfwd=tcp::2222-:22 \
 -device virtio-net-device,netdev=eth0 \
 -drive file=../root.ext4,if=none,format=raw,id=hd0 \
 -device virtio-blk-device,drive=hd0 \
```

```
 -device virtio-rng \
 -soundhw hda -audiodev id=pa,driver=pa
)
```

With the prepared script, I ran the OpenWRT system and firstly corrected the network configuration by doing:

 # vi /etc/config/network (and changing the file „config interface lan" section as below - I simply commented all other options)

```
config interface 'lan'
     option ifname 'eth0'
     option proto 'dhcp'
```

(To save & exit I clicked ESC, typed ":x" and pressed ENTER)

 # /etc/init.d/network restart

Next, I downloaded and installed the GPIO driver from the host and tested whether the configuration works correctly.

```
wget http://10.0.2.2:8000/kmod-drv-mpc8xxx_4.14.221-1_aarch64
_generic.ipk
opkg install kmod-drv-mpc8xxx_4.14.221-1_aarch64_generic.ipk
modprobe gpio-mpc8xxx
opkg update
opkg install libgpiod
opkg install gpiod-tools
gpiodetect
gpioinfo
gpioset gpiochip1 24=1
gpioset gpiochip1 24=1
```

To create a proper working music player I needed the python interpreter, so based on the email "[LINES][LINSW] libgpiod in OpenWRT" I added python to the OpenWRT. In „openwrt-sdk-19.07.7-armvirt-64_gcc7.5.0_musl.Linux-x86_64/" directory I executed:

```
./scripts/feeds update -a
./scripts/feeds install python3
./scripts/feeds install python3-flask
make menuconfig (I unchecked the "Cryptographically sign package
lists")
make V=s
```

and added libgiod package:

```
./scripts/feeds update
./scripts/feeds install python3 libgpiod
```

Here I replaced the Makefile from the libgiod directory ("package/feeds/packages/libgpiod/Makefile") with the Makefile sent by the teacher, compiled the package and hosted it locally:

```
make package/libgpiod/compile
cd bin/packages/aarch64_generic/packages
python3 -m http.server
```

And in OpenWRT:

```
opkg update
opkg install python
wget http://10.0.2.2:8000/python3-gpiod_1.3-1_aarch64_generic.ipk
opkg install python3-gpiod_1.3-1_aarch64_generic.ipk
```

Then I had to install soundcard drivers.

```
opkg install alsa-utils
opkg install pciutils
opkg install kmod-sound-hda-intel
alsactl init
```

Then with the HTTP python server I uploaded my "webservice.py" and "music_player.py" to OpenWRT. But after the run I had a problem with the MPD package.

```
opkg install mpd-full
opkg install mpc
```

After that, mpc played the music but my script still returned the error. To resolve the problem I changed the „/etc/mpd.conf" file by uncommenting the necessary commands and I created the directories which were written in the file. Then restarted the daemon „/etc/ init.d/mpd restart" and with „mpc update" I checked if everything works correctly. At the moment, the webservice.py and music_player.py worked but not at the same time. So I installed the screen package.

```
opkg install screen
```

The last step of the lab was about starting the application automatically after the start of the system. I found that my script should have at least two functions „start()” and „restart()”. Final version of my script looks as follows:

```
#!/bin/sh /etc/rc.common

START=99

start()
{
    echo "---- MUSIC PLAYER START ----"
    alsactl init
    sleep 3
    modprobe gpio-mpc8xxx
    mpc update
    mpc clear
    mpc add /
    mpc repeat 1
    screen -m -d python3 webservice.py
    screen -m -d python3 music_player.py
}

restart()
{
    echo "---- MUSIC PLAYER START ----"
    alsactl init
    sleep 3
    modprobe gpio-mpc8xxx
    mpc update
    mpc clear
    mpc add /
    mpc repeat 1
    screen -m -d python3 webservice.py
    screen -m -d python3 music_player.py
}
```

I created the script in “etc/init.d” and enabled it with “/etc/init.d/start_script enable”. In the end, I couldn’t do a better debouncing function. Moreover, although my music_player starts without fail - the music plays right after the start of the system and it can be controlled with GUI buttons, my webserver is not functional. It launches at localhost:2222, but due to some issue with character encoding, which I wasn’t able to fix, the layout is completely unreadable and it loads for eternity.