Warsaw 13.06.2018

Linux for Embedded Systems – Laboratory ex. 5

The student(s): Patryk Walczak

Description of the assignment

Exercise 5 is just recreation of the system from Exercise 4 in OpenWRT.

The subject of the assignment 5 is the implementation with the emulated Vexpress A9 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 a built-in audio output. In a better version it can be an external USB audio card or other device. Please remember that QEMU allows you to share USB devices with the emulated board.

I chosed the suggetion:
D) Using the standard audio output of the RPi, 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 cofigure the list songs, to download and to remove the song.

Procedure to recreate the design from the attached archive

Firstly, run „reproduce.sh” which starts python HTTP server as the last command. Then go to /BR_Internet_Radio/QemuVirt64 and start OpenWRT with „run_script”. After the start of the system, fix the network configuration by:

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

And download the reproduce_OpenWRT.sh by

```
wget http://10.0.2.2:8000/reproduce_OpenWRT.sh
```

Next run it with:

```
chmod 777 reproduce_OpenWRT.sh
./reproduce_OpenWRT.sh
```

Quit from the system and start „run_before” and „run_script” again.

Description of the solution

Firstly basing on the mail „[LINES] How to use our emulated GPIOs with OpenWRT?” I started from downloading the git repository BR_Internet_Radio. To speed up the building process I changed the build.sh by writing „make host-qemu’ instead of „make”.

```
 git clone https://github.com/wzab/BR_Internet_Radio
 cd BR_Internet_Radio
 git checkout gpio_simple_USB
 cd QemuVirt64
 ./build.sh
```

Then I started downloading the necessary components of OpenWRT.

```
wget https://downloads.openwrt.org/releases/19.07.2/targets/armvirt/64/openwrt-19.07.2-armvirt-64-Image
wget https://downloads.openwrt.org/releases/19.07.2/targets/armvirt/64/openwrt-19.07.2-armvirt-64-root.ext4.gz
wget https://downloads.openwrt.org/releases/19.07.2/targets/armvirt/64/openwrt-sdk-19.07.2-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-19.07.2-armvirt-64-root.ext4.gz > root.ext4
truncate -s \>512M root.ext4
/sbin/resize2fs root.ext4
```

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

```
tar -xJf openwrt-sdk-19.07.2-armvirt-64_gcc-7.5.0_musl.Linux-x86_64.tar.xz
wget http://koral.ise.pw.edu.pl/~wzab/LINES/L8_examples.tar.bz2
tar xjf L8_examples.tar.bz2
cd openwrt-sdk-19.07.2-armvirt-64_gcc-7.5.0_musl.Linux-x86_64
```

And I adjusted the path at the end of of the feeds.conf.default after adding „src-link mini /absolute/path/to/W8”. Then I configured SKD and I compiled the GPIO driver.

```
export LANG=C
scripts/feeds update mini
scripts/feeds install -a -p mini
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
```

Then I ran the HTTP server in the „bin/targets/armvirt/64/packages” and in the „BR_Internet_Radio/QemuVirt64” location I started GUI with „run_before” script and I created „run_script” to start openWRT (I adjusted the path from the mail). In the system firstly I corrected the network configuration.

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

Next, download and install the GPIO driver from the host and test whether the configuration works correctly.

```
wget http://10.0.2.2:8000/kmod-drv-mpc8xxx_4.14.171-1_aarch64_generic.ipk
opkg install kmod-drv-mpc8xxx_4.14.171-1_aarch64_generic.ipk
modprobe gpio-mpc8xxx
opkg update
opkg install libgpiod
opkg installgpiod-tools
gpiodetect
gpioset 1 24=1
gpioset 1 24=1
```

To create proper working music player I needed the python interpreter, so basing on „[LINES] Re: Exercise 5 and gpiod in Python / Ćwiczenie 5 i gpiod w Pythonie” I added python to the OpenWRT.In „openwrt-sdk-19.07.2-armvirt-64_gcc-7.5.0_musl.Linux-x86_64/” file:

```
./scripts/feeds update -a
./scripts/feeds install python3
./scripts/feeds install python3-flask
make menuconfig (check if the python package is selected and uncheck the "Cryptographically sign package lists")
make
```

and added libgiod package:

```
scripts/feeds update
scripts/feeds install python3 libgpiod
vim package/feeds/packages/libgpiod/Makefile
```

Here I changed the Makefile from the libgiod directory with the Makefile from the mail.

```
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. With „[LINES] Problem with sound in OpenWRT?” mail I had no problems with the step.

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

Then with the HTTP python sever 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. One of my friends fixed the problem with Mr.Zabołotny. 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 deamon „/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” and I prepared the „start.sh” script:

```
echo "---- MY APP START ----"
alsactl init
modprobe gpio-mpc8xxx
screen -m -d python3 webservice.py
python3 music_player.py
```

The last step of the lab was about starting the application automatically after the start of the system. After some troubles, I found that my script should have at least two functions „start()” and „restart()”. Finial version of my script looks like:

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

START=99

start()
{
echo "---- MY APP START ----"
alsactl init
modprobe gpio-mpc8xxx
screen -m -d python3 webservice.py
screen -m -d python3 music_player.py
}

restart()
{
echo "---- MY APP START ----"
alsactl init
modprobe gpio-mpc8xxx
```

```
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/run_after_start enable”. In the end, I fixed the errors from the previous laboratory. I added the redirecting buttons, but I cannot do a better debouncing function.