Course notes, module 3 week 7
LEO1 kit, embedded linux, networking continued.

Version v2022-02-18, Kjeld Jensen kjen@sdu.dk

Agenda

1. Recap of the exercises from previous module
2. Preparation for next module
3. Exercises
4. Optional exercises

1. Recap of exercises from previous module

In the previous module we performed a basic setup of the RPi. This included ethernet network configuration for communicating with your linux desktop environment and routing traffic from the RPi to internet though your linux desktop environment. Finally we installed the Apache2 webserver.

We will briefly review the results and findings of the exercises and the optional exercises in class.

2. Preparation for next module

In module 4 we will focus on interfacing to sensors and actuators. No preparation is needed for this, however please make sure that you have completed all exercises from modules 1-3.

In particular you will need to have the RPi configured as a wifi access point like in exercise 3.3 in this module. This will enable all team members to connect to and work on the RPi at the same time.

Please remember to bring the full LEO1 kit and the hardware prerequisites to all remaining modules of the course.

3. Exercises

The exercises for this module focus on wifi networking, traffic routing, configuring a RPi wifi access point and running basic web services.

3.1 Connect the RPi to a Wifi network

The purpose of this exercise is to connect the RPi as a wifi client to an existing wifi access point and to enable internet access from the RPi via this wifi connection.

First ensure that wifi is active on the RPi. It may be blocked by rfkill if you haven't set the country to Denmark. The reason for this is that the RPi configuration thus doesn't know which regulations apply with regards to frequency spectrum management:

```
$ sudo raspi-config
```

Then set the country to DK under `Localization options`

Now configure the RPi to connect as client to a wifi access point. Please notice that the below configuration will work with most wifi access points but not Eduroam, so activate internet sharing on your phone and use the SSID and password from this wifi access point in you RPi configuration.

Wifi client settings can be configured via raspi-config, but this won't work if you wish to add the configuration to the MicroSD card before inserting it into the RPi. We will therefore edit the configuration file manually.

You can either choose to edit the configuration file by mounting the MicroSD card on your linux desktop environment, or you can ssh to the RPi via the ethernet connection configured in module 2. The below example assumes a ssh connection to the RPi:

```
$ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Add the below configuration to the bottom of the file:

```
network ={
     ssid ="SSID"
     psk ="PASSWORD"
     scan_ssid =1
}
```

Replace SSID and PASSWORD with the configuration for your wifi access point.

When you reboot you should observe that the RPi connects to the configured wifi access point. You can check this by running:

```
$ iwconfig
```

And you can see the IP address assigned to the RPi by the wifi access point by running:

```
$ ifconfig
```

Now if you try to ping a server such as dr.dk you may notice that this is not possible. The reason is likely that we now have configured both the RPi wifi interface and the RPi ethernet interface, thus the RPi doesn't know which interface to use for internet access and will try to use the RPi ethernet interface. Try to run:

```
$ ip route show
```

Which should produce an output similar to this:

```
default via 10.0.0.1 dev eth0 src 10.0.0.10 metric 202
default via 192.168.0.1 dev wlan0 proto dhcp src 192.168.0.150 metric 303
10.0.0.0/24 dev eth0 proto dhcp scope link src 10.0.0.10 metric 202
```

```
192.168.0.0/24 dev wlan0 proto dhcp scope link src 192.168.0.150 metric 303
```

Here the ethernet interface (eth0) is configured to route via `10.0.0.1` and the wifi interface (wlan0) is configured to route via `192.168.0.1`

Which gateway the RPi use to route external (internet) traffic to is determined by the `metric` value. The one with the lowest value is used. So to make the RPi route via the wifi interface we need to make sure, that it has the lowest `metric` value. To change this value we need to delete the route and then create it again:

```
$ sudo ip route del default via 192.168.0.1
$ sudo ip route add default via 192.168.0.1 dev wlan0 proto static metric 10
```

Now run:

```
$ ip route show
```

and observe the change in the routing table. The wireless interface should now have the lowest `metric` value, and you should now be able to ping dr.dk

3.2 Setting up the RPi as a Wifi access point

The purpose of this exercise is to configure the RPi as a wifi acces point that you can connect to using a wifi client such as your phone[1].

This configuration is obviously conflicting with the configuration in exercise 3.1 where the RPi acted as a wifi client: with only one wifi interface it is not possible to act as both an access point and a client at the same time.

For the RPi wifi access point we will use the subnet `192.168.10.0/24`

First we configure the RPi IP address on the wifi device. This IP address will then become the routing address (gateway) for the connected wifi clients:

```
$ sudo nano /etc/dhcpcd.conf
```

Add this configuration at the bottom of the file i.e. below our eth0 configuration:

```
interface wlan0
    static ip_address=192.168.10.1/24
    nohook wpa_supplicant
```

We then install and configure the access point (AP) package:

```
$ sudo apt install hostapd
```

Edit the configuration file (new file):

```
$ sudo nano /etc/hostapd/hostapd.conf
```

1Inspiration for this exercise has been found here: https://www.raspberrypi.com/documentation/computers/configuration.html#setting-up-a-routed-wireless-access-point

Add the configuration below, please repleace `XX` by your team number.

You may also change the `wpa_passphrase` however please observe that it must be at least 8 characters.

```
country_code=DK
interface=wlan0
ssid=LEO1_TEAM_XX
hw_mode=a
channel=40
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=embeddedlinux
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

The above configuration is for 5 GHz wifi. If you wish to use 2.4 GHz instead, then use
`hw_mode=g`

For 2.4 GHz the channels 1-13 are available in Denmark[2]. They are however overlapping and not all devices accept channel 12 and 13. You should therefore only use the channels: 1, 6, 11.

For 5 GHz the channels do not have the same problem of overlapping as on 2.4 GHz. There are quite a few channels available, but most of them require Dynamic Frequency Selection (DFS) to mitigate interference with 5 GHz radars. For now to make things simple use the channels[3]: 36, 40, 44, 48.

The above configuration will make the RPi broadcast the wifi network SSID and allow connections. However the wifi clients will usually not connect successfully unless an IP address is issued to the wifi client via the Dynamic Host Configuration Protocol (DHCP). To configure a DHCP service we install and configure the dnsmasq package:

```
$ sudo apt install dnsmasq
```

This service has a quite long default configuration file. For now we just make a backup and create a new empty file:

```
$ sudo mv /etc/dnsmasq.conf /etc/gnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Add the configuration below:

```
interface=wlan0
dhcp-range=192.168.10.150,192.168.10.199,255.255.255.0,1h
domain=wlan
address=/leo1_ap.wlan/192.168.10.1
```

2 For more information about wifi thannels please look at this reference:
https://en.wikipedia.org/wiki/List_of_WLAN_channels
3 These channels belong to the UNII-1 spectrum

Reboot the RPi for the configuration to take effect:

```
$ sudo reboot
```

After the RPi has rebooted, the wifi access point should become visible to wifi clients. Try to connect to the RPi wifi access point and observe that your wifi client by the DHCP is issued an IP address within the range `192.168.10.150-192.168.10.199`

You can test if everything works by using ssh to connect to the RPi wia wifi:

```
$ ssh pi@192.160.10.1
```

Or you can access the webserver that we installed on the RPi in module 2 via wifi:

```
http://192.168.10.1
```

3.3 Routing RPi wifi access point traffic

The purpose of this exercise is to enable the RPi wifi access point to route traffic from a connected wifi client to the internet via the ethernet port.

This exercise assumes that the RPi has internet access via the ethernet port like in module 2 exercise 6.6.

We will use iptables to handle the routing. iptables is not installed by default under Raspberry Pi OS lite so it must be installed:

```
$ sudo apt install iptables
```

We then use the same configuration method as in module 2 exercise 6.6, only this time we do this on the RPi and we route traffic from wifi clients connected to wlan0 onto the eth0 interface:

```
$ sudo sysctl -w net.ipv4.ip_forward=1
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Make sure your RPi is connected to the internet via the ethernet interface, which will probably be via one of your computers which routes TCP/IP traffic from the RPi onto a wireless connection like Eduroam.

Your RPi wifi access point should now provide internet access to the connected wifi clients. Try to connect a wifi client such as a phone or laptop without any other internet access to the RPi wifi access point. You should then be able to access the internet via the RPi.

3.4 RPi startup script

The iptables commands in exercise 3.3 need to be executed after each boot of the RPi. This is cumbersome and for this and many other purposes it is very practical to create a "boot" shell script that is executed each time the RPi has booted.

First create a boot script:

```
$ mkdir -p /home/pi/bin
$ nano /home/pi/bin/boot.sh
```

Add the lines below from exercise 3.3:

```
#!/bin/bash
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Now make the script executable:

```
$ chmod 755 /home/pi/bin/boot.sh
```

To make the script run during boot we add it to /etc/rc.local

```
$ sudo nano /etc/rc.local
```

Add the below text before the exit line at the end of the file:

```
sudo sh /home/pi/bin/boot.sh
```

Now reboot your RPi. After boot it should serve as wifi access point and route traffic from wifi clients to the ethernet interface without you having to ssh to the RPi and execute the iptables commands manually.

3.5 Apache2 webserver pages
With all set concerning the RPi wifi access point we turn to the Apache2 webserver already installed on the RPi. The purpose of this exercise is to be able to show basic output on a webpage so that you can connect to the RPi wifi access point and retrieve information about the access point or in the future any connected sensor/actuator etc.

First we install php which adds the php scipting language to the Apache2 webserver:

```
$ sudo apt install php
```

The apache webserver html root directory is located at `/var/www/html`

After a fresh installation of Apache2 a aingle file named `index.html` is located in this directory. This file contains the "Apache2 Debian Default Page" that you see when you visit `http://192.168.10.1`

First the user `pi` needs to be able to write files in the webserver directory. This is achieved by making the webserver user `www-data` the owner of those files and then granting the user `pi` access to the files owned by `www-data`:

```
$ cd /var/www
$ sudo chown -R www-data.www-data html
$ sudo chmod -R g+w html
$ sudo usermod -a -G www-data pi
```

For the usermod command to take effect you need to logout and then login.

Rename the `index.html` file to `index.html.orig` and create a new file named `index.php` which contains the text "Hello LEO1 World".

Now visit `http://192.168.1.10` and observe that you have created a simple web page with this text.

Now we move onto displaying information from a linux program on a web page. As an example the linux command uptime shows the accumulated time since boot of the computer:

```
$ uptime
```

In `/var/www/html` create a file named `uptime.php` Add the below php script to the file:

```
<?php
     $output = shell_exec("uptime");
     echo ('The linux uptime is:<br/>'.$output);
?>
```

Now visit `http://192.168.1.10/uptime.php` and observe that the webpage returns the result of this script.

If you are viewing the web page on a smartphone you may notice that the text is really small and difficult to read. In this case try rewriting `uptime.php` with the below html code encapsulating the php script. This provides an easy readable page:

```
<!DOCTYPE html>
<html>
<head>
<title>LEO1</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0">
<meta name="apple-mobile-web-app-capable" content="yes">
</head>
<body>
<p style="font-family: "Lucida Grande", Verdana, Geneva; font-size: 12px;">
<b>LEO1 Uptime test</b><br/>
<?php
$output = shell_exec("uptime");
echo ('The linux uptime is:<br/>'.$output);
?>
</p>
</body>
</html>
```

4. Optional exercises

The below exercises are not required but higly recommended. You may choose freely between them.

For exercises that result in a script or a method or the like please share your results on itslearning by starting a new thread at the itslearning forum with your team name as Subject (please use the format "Team_xx"). Then post your results and findings in this thread. Then other teams may then seek inspiration there.

4.1 RPi health monitoring scripts

The purpose of this exercise is to scripts that returns relevant data about the RPi state and health. The scripts should comply with the main Unix tenets.

Create a shell script that returns:

- the avilable disk space on the RPi in percent
- the currently available RAM on the RPi in percent
- the current CPU load in percent
- the RPi CPU temperature in degrees Celcius
- the number of bytes transferred for a given network interface such as eth0 and wlan0

4.2 RPi health data logging

Create a shell script that utilize one of the scripts from exercise 4.1 to log the data in to a text file. Use /etc/crontab to run the scripts with an interval of 1, 5 or 15 minutes depending on relevance. Save the values as comma separated values with the time as first column and the measured value as the second column.

Use the Unix Epoch (the number of seconds that have elapsed since January 1, 1970 at midnight UTC time minus the leap seconds) as the time:

```
$ date +%s
```

4.3 RPi health data web access

Create a PHP script under the webserver which provides access to the 10 latest logged health values. When presenting the values convert the Unix Epoch seconds into a date-time stamp.

4.4 Install a database on the RPi

Install a SQL database such as MySQL, MariaDB, PostgreSQL on the RPi.

Create a database, under this create a table that stores the Epoch time, a value ident and value.

Modify the logging from exercise 4.2 to save data from multiple scripts into the database instead of text files. Each health value should be assigned a value ident to allow search queries for a specific value type.