

IoT Lab Exercise 01-2024

Hamza AHMAD, Armin ASSADZADEH, Numan DEMIR

Tiny URL: <http://tinyurl.com/iotlab2401>

Helpful resources:

Github repository: <https://github.com/RIOT-OS/RIOT>

API documentation: <https://doc.riot-os.org/>

Remember: the examples directory is very helpful - as well as the test directories :)

Riot-OS Setup

Let's get started; first we have to set up Riot-OS on our host:

https://wiki.elvis.science/index.php?title=Riot-OS_Setup

Check the output on the Moserial application (hello world example).

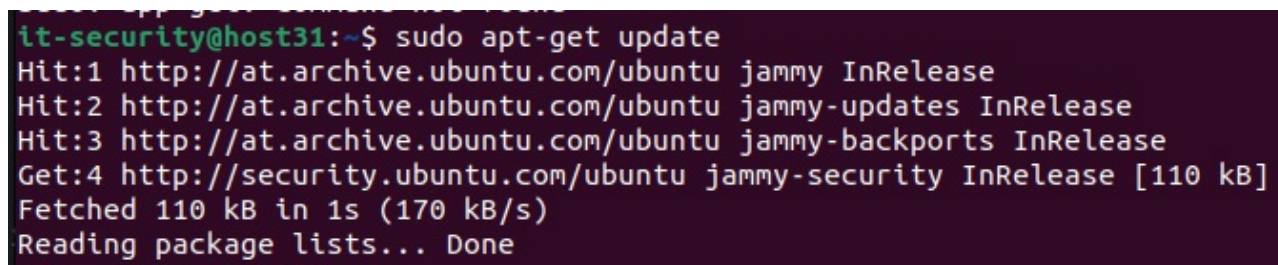
Step 1

We start by installing the following packages:

git, openocd, gcc-multilib, build-essential, python3-serial, libudev-dev, moserial

First, we update the package lists for upgrades and new package installations:

- `sudo apt-get update`



```
it-security@host31:~$ sudo apt-get update
Hit:1 http://at.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://at.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://at.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Fetched 110 kB in 1s (170 kB/s)
Reading package lists... Done
```

Next, we install the necessary packages:

- `sudo apt-get install {}`

```
it-security@host31:~$ sudo apt-get install git
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.34.1-1ubuntu1.10).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install openocd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openocd is already the newest version (0.11.0-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install gcc-multilib
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc-multilib is already the newest version (4:11.2.0-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install python3-serial
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3-serial is already the newest version (3.5-1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install libudev-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libudev-dev is already the newest version (249.11-0ubuntu3.12).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$ sudo apt-get install moserial
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
moserial is already the newest version (3.0.12-0ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
it-security@host31:~$
```

Step 2

Proceed with installing the toolchain:

- `sudo apt-get install gcc-arm-none-eabi`

```
it-security@host31:~$ sudo apt-get install gcc-arm-none-eabi
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc-arm-none-eabi is already the newest version (15:10.3-2021.07-4).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Moment of Uncertainty/Doubts

After finishing Step 2, we were unsure if we should continue to Step 3 or follow the additional instructions given on Wikipedia at Step 2. The information shown was:

"If you use a recent version of RIOT-OS (Release-2021.07) you can go on with Step 3. If you use an older RIOT-OS version, you will have to do the following steps:"

We weren't completely sure if we were using the latest version, but we checked on GitHub "<https://github.com/RIOT-OS/RIOT/releases>" and decided to go ahead with Step 3 because our version wasn't older than Release-2021.07. The version we're using is:



Step 3

First, we download RIOT-OS by cloning the repository:

- `git clone https://github.com/RIOT-OS/RIOT.git myRIOT`

```
it-security@host31:~$ git clone https://github.com/RIOT-OS/RIOT.git myRIOT
Cloning into 'myRIOT'...
remote: Enumerating objects: 354712, done.
remote: Counting objects: 100% (2874/2874), done.
remote: Compressing objects: 100% (1526/1526), done.
remote: Total 354712 (delta 1631), reused 2366 (delta 1323), pack-reused 351838
Receiving objects: 100% (354712/354712), 151.25 MiB | 24.95 MiB/s, done.
Resolving deltas: 100% (241606/241606), done.
Updating files: 100% (16221/16221), done.
```

Then, we navigate to the "Hello World"-Example directory:

- `cd myRIOT/examples/hello-world`

Finally, we build the application on "native":

- `make all term`


```

it-security@host31:~/myRIOT/examples/hello-world$ make all term
Building application "hello-world" for "native" with CPU "native".

"make" -C /home/it-security/myRIOT/boards/common/init
"make" -C /home/it-security/myRIOT/boards/native
"make" -C /home/it-security/myRIOT/boards/native/drivers
"make" -C /home/it-security/myRIOT/core
"make" -C /home/it-security/myRIOT/core/lib
"make" -C /home/it-security/myRIOT/cpu/native
"make" -C /home/it-security/myRIOT/cpu/native/periph
"make" -C /home/it-security/myRIOT/cpu/native/stdio_native
"make" -C /home/it-security/myRIOT/drivers
"make" -C /home/it-security/myRIOT/drivers/periph_common
"make" -C /home/it-security/myRIOT/sys
"make" -C /home/it-security/myRIOT/sys/auto_init
"make" -C /home/it-security/myRIOT/sys/libc
"make" -C /home/it-security/myRIOT/sys/preprocessor
  text    data    bss    dec    hex filename
 27524    552   47840   75916  1288c /home/it-security/myRIOT/examples/hello-
world/bin/native/hello-world.elf
/home/it-security/myRIOT/dist/tools/pyterm/pyterm -ps /home/it-security/myRIOT/e
xamples/hello-world/bin/native/hello-world.elf --process-args tap0
Twisted not available, please install it if you want to use pyterm's JSON capabi
lities
Welcome to pyterm!
Type '/exit' to exit.
2024-03-05 11:32:30,910 # RIOT native interrupts/signals initialized.
2024-03-05 11:32:30,911 # RIOT native board initialized.
2024-03-05 11:32:30,911 # RIOT native hardware initialization complete.
2024-03-05 11:32:30,911 #
2024-03-05 11:32:30,911 # main(): This is RIOT! (Version: 2024.04-devel-361-g31d
23)
2024-03-05 11:32:30,911 # Hello World!
2024-03-05 11:32:30,911 # You are running RIOT on a(n) native board.
2024-03-05 11:32:30,911 # This board features a(n) native CPU.

```

Step 4

In this Step we want to build and run "Hello World" on the SAM R21 Board.

First, we set the rights for our username to be able to flash the application on the board:

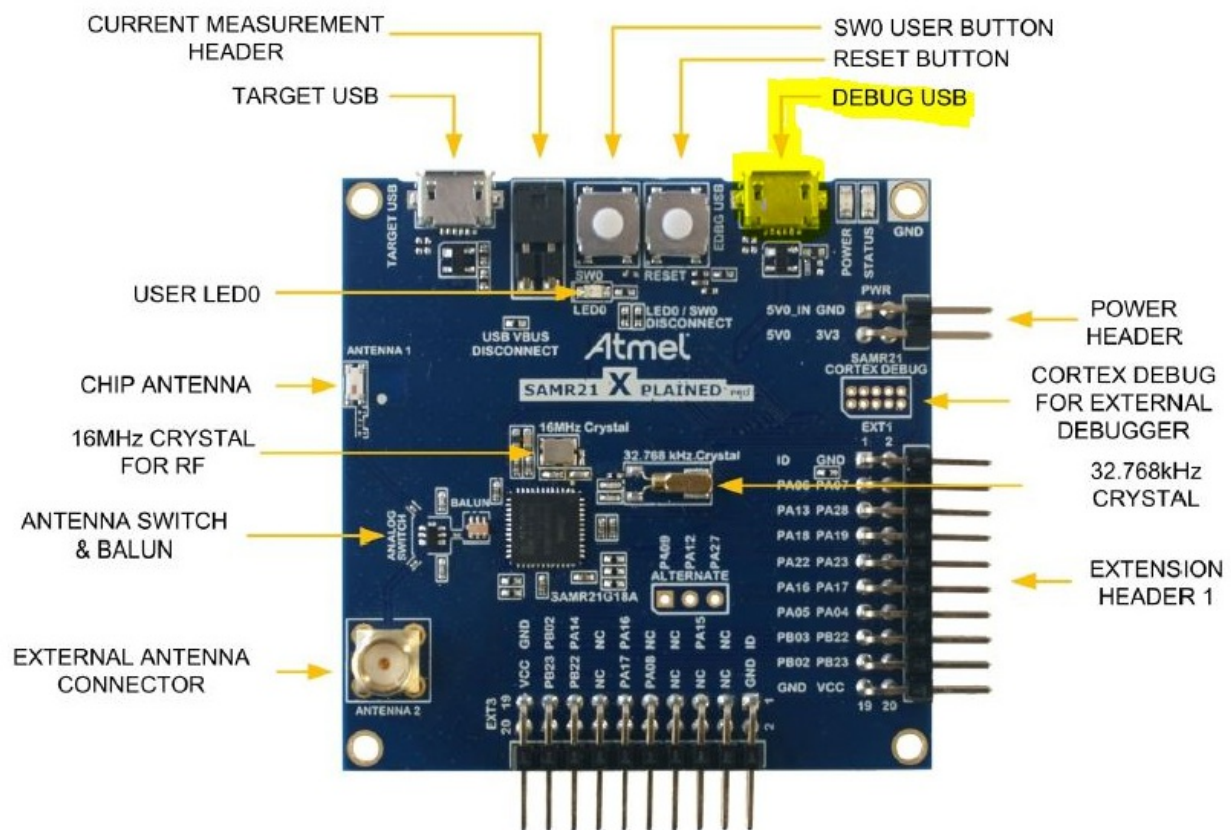
- `sudo usermod -a -G dialout <username>`

```

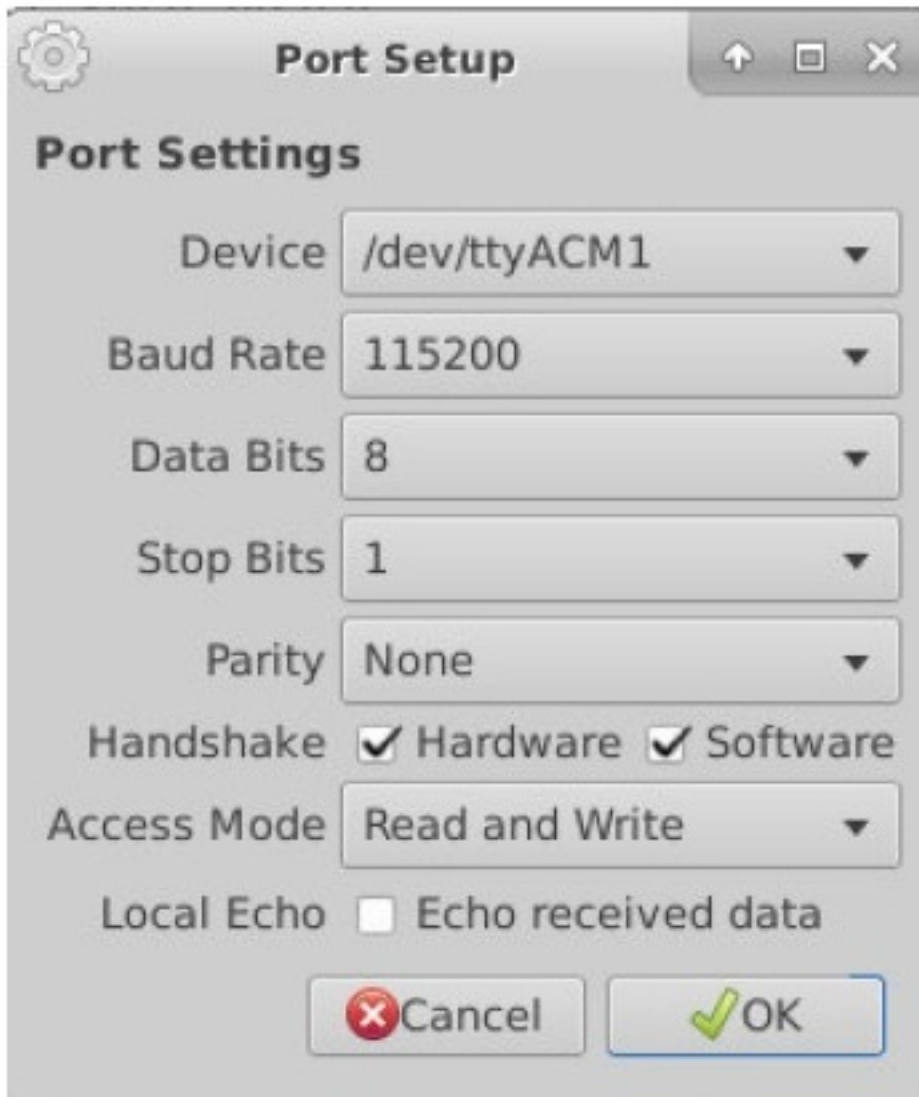
it-security@host31:~/myRIOT/examples/hello-world$ sudo usermod -a -G dialout it-
security

```

After rebooting the computer, we connect the board to our computer via micro USB cable (USB EDBG interface).



and connect via Moserial:

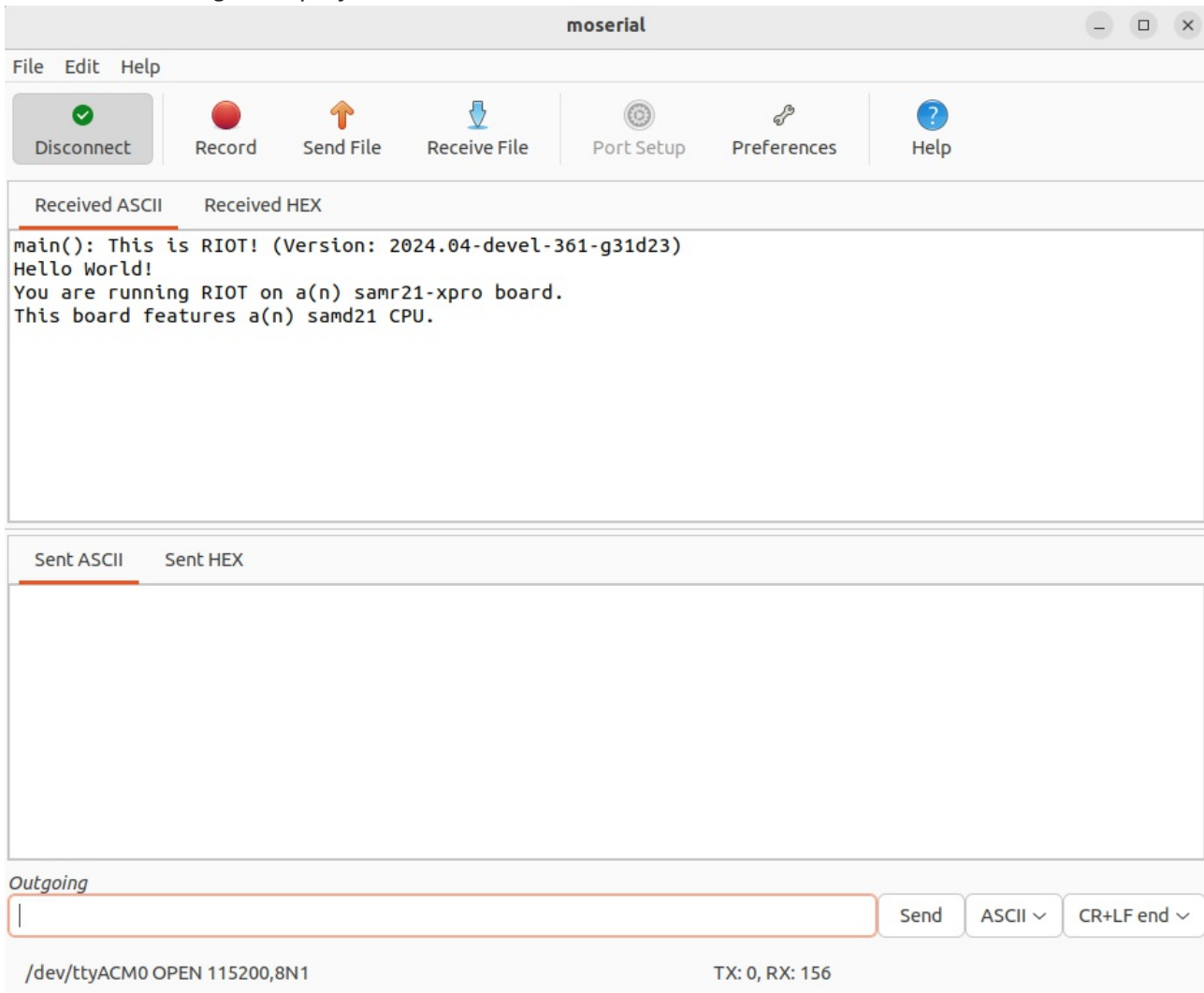


Build and flash the application.

To build and flash the application we use these commands:

- `cd myRIOT/examples/hello-world`
- `make BOARD=samr21-xpro`
- `make BOARD=samr21-xpro flash`

Now the following is displayed in Moserial:



Task 1

In the "Hello World" code change the output to "Hello Earth!".

To change the output we first navigat to the directory:

- `cd myRIOT/examples/hello-world`

Then, we change the text in the main.c file:

- `nano main.c`


```
GNU nano 6.2                                main.c *
/*
 * Copyright (C) 2014 Freie Universität Berlin
 *
 * This file is subject to the terms and conditions of the GNU Lesser
 * General Public License v2.1. See the file LICENSE in the top level
 * directory for more details.
 */

/**
 * @ingroup      examples
 * @{
 *
 * @file
 * @brief        Hello World application
 *
 * @author        Kaspar Schleiser <kaspar@schleiser.de>
 * @author        Ludwig Knüpfer <ludwig.knuepfer@fu-berlin.de>
 * @}
 */

#include <stdio.h>

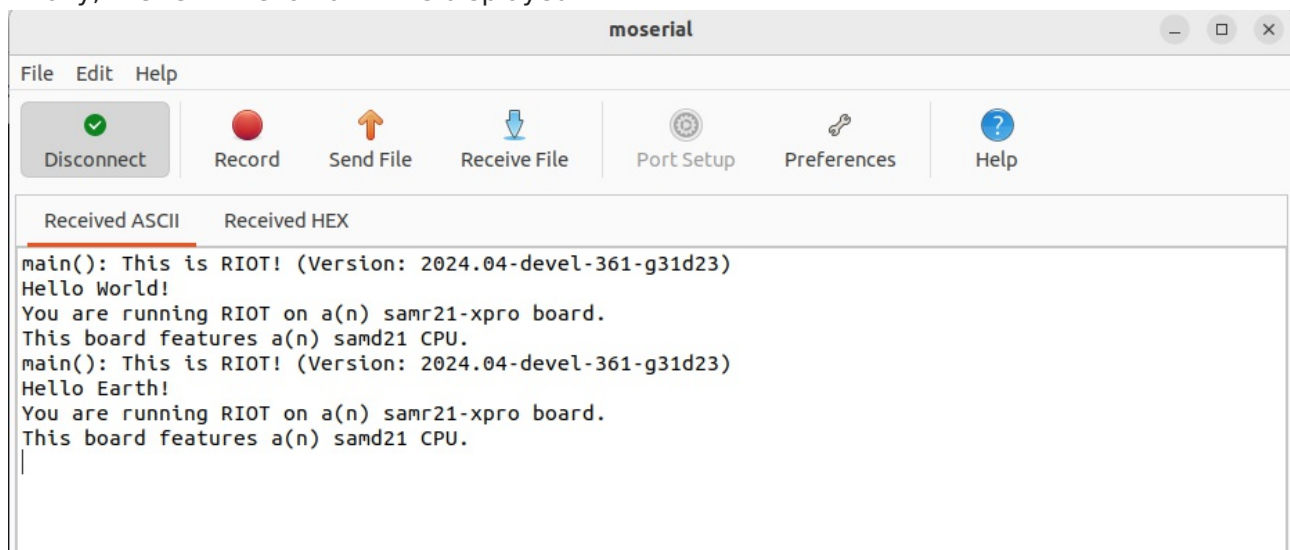
int main(void)
{
    puts("Hello Earth!");

    printf("You are running RIOT on a(n) %s board.\n", RIOT_BOARD);
    printf("This board features a(n) %s CPU.\n", RIOT_CPU);

    return 0;
}
```

After that we board again.

Finally, the text "Hello Earth!" is displayed:



The image shows a terminal window titled "moserial" with a menu bar (File, Edit, Help) and a toolbar with icons for Disconnect, Record, Send File, Receive File, Port Setup, Preferences, and Help. The main area displays the output of the application, with "Received ASCII" selected. The output shows two successful boardings, with the second one displaying "Hello Earth!".

```
main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
Hello World!
You are running RIOT on a(n) samr21-xpro board.
This board features a(n) samd21 CPU.
main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
Hello Earth!
You are running RIOT on a(n) samr21-xpro board.
This board features a(n) samd21 CPU.
```

Task 2

Setup the "gnrc_networking" example and start with "Connecting two RIOT instances" - show that you actually send UDP messages between 2 boards.

We followed these instructions:

https://github.com/RIOT-OS/RIOT/tree/master/examples/gnrc_networking

First, we switch to the "gnrc_networking" directory:

```
it-security@host31:~/myRIOT/examples$ cd gnrc_networking
it-security@host31:~/myRIOT/examples/gnrc_networking$
```

Then, create a tap0 interface:

- `sudo ip tuntap add tap0 mode tap user ${USER}`

```
it-security@host31:~/myRIOT/examples/gnrc_networking$ sudo ip tuntap add tap0 mode tap user
${USER}
[sudo] password for it-security:
it-security@host31:~/myRIOT/examples/gnrc_networking$
```

Now, we use "make term" to automatically connect to the tap0 interface:

```
it-security@host31:~/myRIOT/examples/gnrc_networking$ make term
/home/it-security/myRIOT/dist/tools/pyterm/pyterm -ps /home/it-security/myRIOT/examples/gnrc
_networking/bin/native/gnrc_networking.elf --process-args tap0
Twisted not available, please install it if you want to use pyterm's JSON capabilities
2024-03-05 12:02:31,327 # RIOT native interrupts/signals initialized.
2024-03-05 12:02:31,328 # RIOT native board initialized.
2024-03-05 12:02:31,328 # RIOT native hardware initialization complete.
Welcome to pyterm!
Type '/exit' to exit.
2024-03-05 12:02:31,328 #
2024-03-05 12:02:31,328 # NETOPT_TX_END_IRQ not implemented by driver
2024-03-05 12:02:31,328 # main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
2024-03-05 12:02:31,328 # RIOT network stack example application
2024-03-05 12:02:31,329 # All up, running the shell now
>
```

After that we build the application on the board:

```
cd myRIOT/examples/gnrc_networking
```

```
make BOARD=samr21-xpro
```

```
make BOARD=samr21-xpro flash
```

```
main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
RIOT network stack example application
All up, running the shell now
>
```

To get the IP address we use:

- `ifconfig`

```
main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
RIOT network stack example application
All up, running the shell now
> ifconfig
Iface 6 HWaddr: 43:B4 Channel: 26 NID: 0x23 PHY: 0-QPSK

    Long HWaddr: 00:04:25:19:18:01:C3:B4
    TX-Power: 0dBm State: IDLE max. Retrans.: 3 CSMA Retries: 4
    AUTOACK ACK_REQ CSMA L2-PDU:102 MTU:1280 HL:64 RTR
    RTR_ADV 6LO IPHC
    Source address length: 8
    Link type: wireless
    inet6 addr: fe80::204:2519:1801:c3b4 scope: link VAL
    inet6 group: ff02::2
    inet6 group: ff02::1
    inet6 group: ff02::1:ff01:c3b4
```

We start the udp server on port 8808:

- udp server start 8808

```
main(): This is RIOT! (Version: 2024.04-devel-361-g31d23)
RIOT network stack example application
All up, running the shell now
> udp server start 8808
Success: started UDP server on port 8808
>
>
```

We use this command on the other board:

- udp send fe80::204:2519:1801:c3b4 8808 HelloHamza

Finally, we received the message:

```
>
> PKTDUMP: data received:
~~ SNIP 0 - size: 10 byte, type: NETTYPE_UNDEF (0)
00000000 48 65 6C 6C 6F 48 61 6D 7A 61
~~ SNIP 1 - size: 8 byte, type: NETTYPE_UDP (4)
  src-port: 8808  dst-port: 8808
  length: 18  cksum: 0xb9a7
~~ SNIP 2 - size: 40 byte, type: NETTYPE_IPV6 (2)
traffic class: 0x00 (ECN: 0x0, DSCP: 0x00)
flow label: 0x00000
length: 18  next header: 17  hop limit: 64
source address: fe80::204:2519:1801:c276
destination address: fe80::204:2519:1801:c3b4
~~ SNIP 3 - size: 24 byte, type: NETTYPE_NETIF (-1)
if_pid: 6  rssi: -58  lqi: 255
flags: 0x0
src_l2addr: 00:04:25:19:18:01:C2:76
dst_l2addr: 00:04:25:19:18:01:C3:B4
~~ PKT      - 4 snips, total size: 82 byte
```

Hex to ASCII:

48 65 6C 6C 6F 48 61 6D 7A 61

Character encoding

ASCII

↻ Convert

✕ Reset

↑↓ Swap

HelloHamza

We encountered issues during the final step with receiving messages on the board. After some time, we discovered that the IPv6 address used was incorrect instead of fe80::204:2519:1801:c**3**b4, the other board send udp messages to fe80::204:2519:1801:c**2**b4. After correcting the IPv6 address, we successfully received the message from the other board as expected.

(Also shown above)

:)