How to ping6 an nRF51422 device by using the nRF51 IoT SDK

About

This is a simple step-by-step guide that walks you through setting up the nRF51 IoT SDK to ping an nRF51422 device. I followed these steps to get from a blank setup to a running environment where I could do a ping6 from a Raspberry Pi to the PCA10028 Development Kit. Ping is a very basic "hello world" for verifying a network link. Still, I find it extremely satisfactory to be able to ping an nRF51422 device from a Raspberry Pi, as an nRF51422 is the lowest cost and lowest energy device with an IP address that I have ever seen!

Most of this guide will be focused on setting up and preparing the Linux environment on a Raspberry Pi. This is a narrowed-down guide that worked for me and my setup. There are many other ways to do the setup; actually, this guide might not even work for you. This is an example for Windows and Keil. Carrying out the same procedure on Linux or OS X and GCC is possible as well, and the steps would not be too different.

Prerequisites

To follow this guide, you need specific hardware and software; see the dependencies below. Knowledge of the nRF51 IoT SDK documentation is also required, as is knowledge of *nix terminals and how to use Linux.

Software

- Windows desktop environment v7.0 or later
- Keil for ARM
- Win32 Disk Imager: http://sourceforge.net/projects/win32diskimager/
- Latest Raspbian image: http://www.raspberrypi.org/downloads/
- Terminal emulator for Windows with ssh capabilities. I use Cygwin, some use PuTTY, MobaXterm, ...
- nRFGo Studio: can be downloaded from MyPage
- nRF51 Tools: can be downloaded from MyPage
- nRF51 IoT SDK: can be downloaded from MyPage
- bluez_4.99-2_armhf.deb: http://mirrordirector.raspbian.org/raspbian/pool/main/b/bluez/bluez_4.99-2_armhf.deb
- libcap-ng0_0.6.6-2_armhf.deb: http://mirrordirector.raspbian.org/raspbian/pool/main/libc/libcap-ng/libcap-ng0_0.6.6-2_armhf.deb
- radvd_1.8.5-1_armhf.deb: http://mirrordirector.raspbian.org/raspbian/pool/main/r/radvd/radvd_1.8.5-1_armhf.deb
- Raspbian kernel with 6LoWPAN: can be downloaded from MyPage

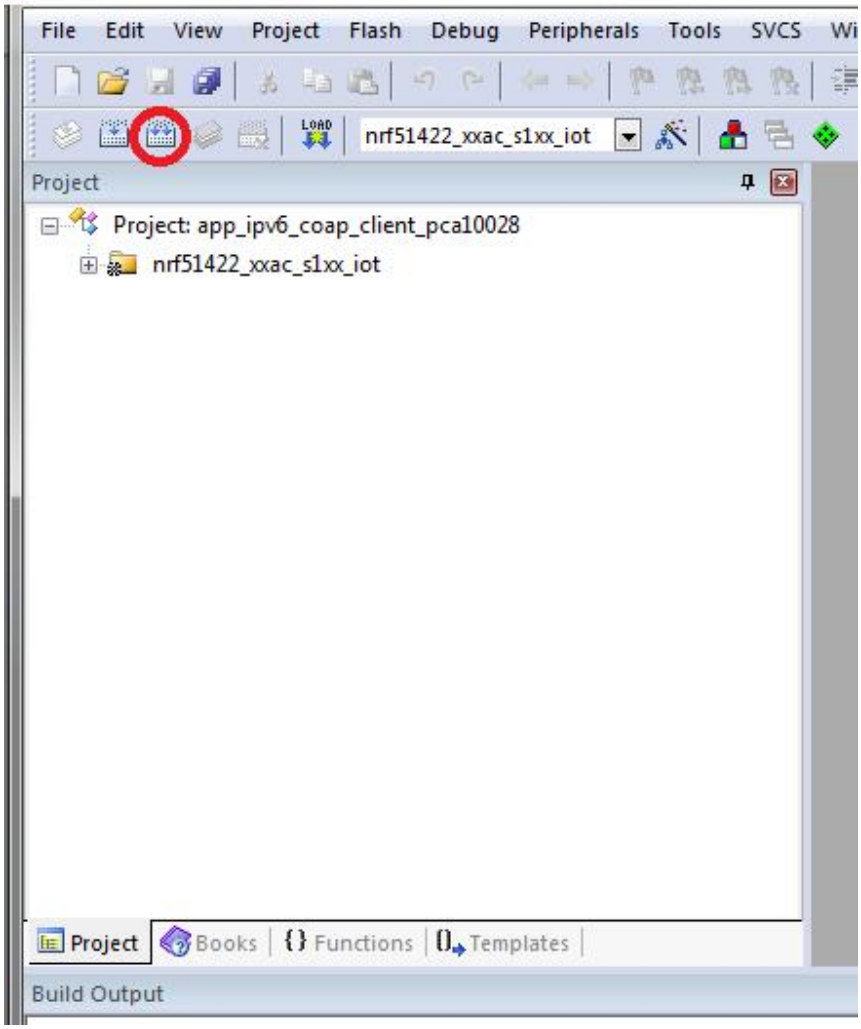
Hardware

- Raspberry Pi model B(+): http://www.raspberrypi.org/products/model-b-plus/
- SD card with at least 8 GB storage
- SD card reader
- Bluetooth v4.0 USB dongle, for example: http://www.dx.com/p/bluetooth-v4-0-csr4-0-usb-dongle-adapter-black-207993#.VIqcmdolTwo
- Desktop PC
- Ethernet switch
- Nordic nRF51 Development Kit (PCA10028): http://www.nordicsemi.com/eng/Products/nRF51-DK/(language)/eng-GB

Preparing the nRF51422 device for setup

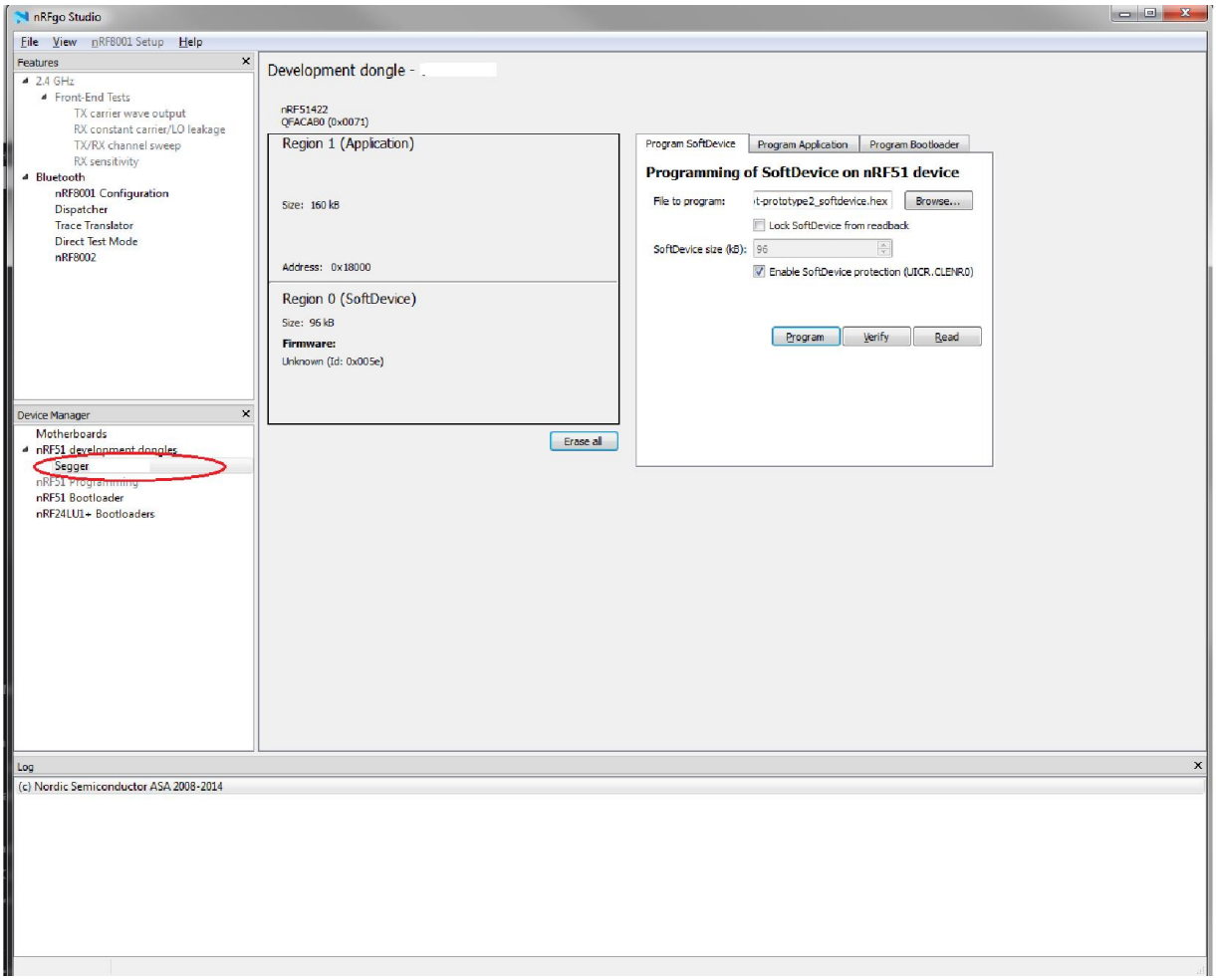
1. Download and install nRFGo Studio.
2. Download and install Keil v4.x.
3. Download and install the latest nRF51 Tools.
4. Download and unzip the nRF51 IoT SDK to any path on your computer.
5. Start Keil and open the nRF51 IoT example `app_ipv6_coap_client_pca10028.uvproj`, which is located in the following folder: `\Nordic\nrf51\examples\iot\ipv6_coap_client\boards\pca10028\arm`

6. Build the example:



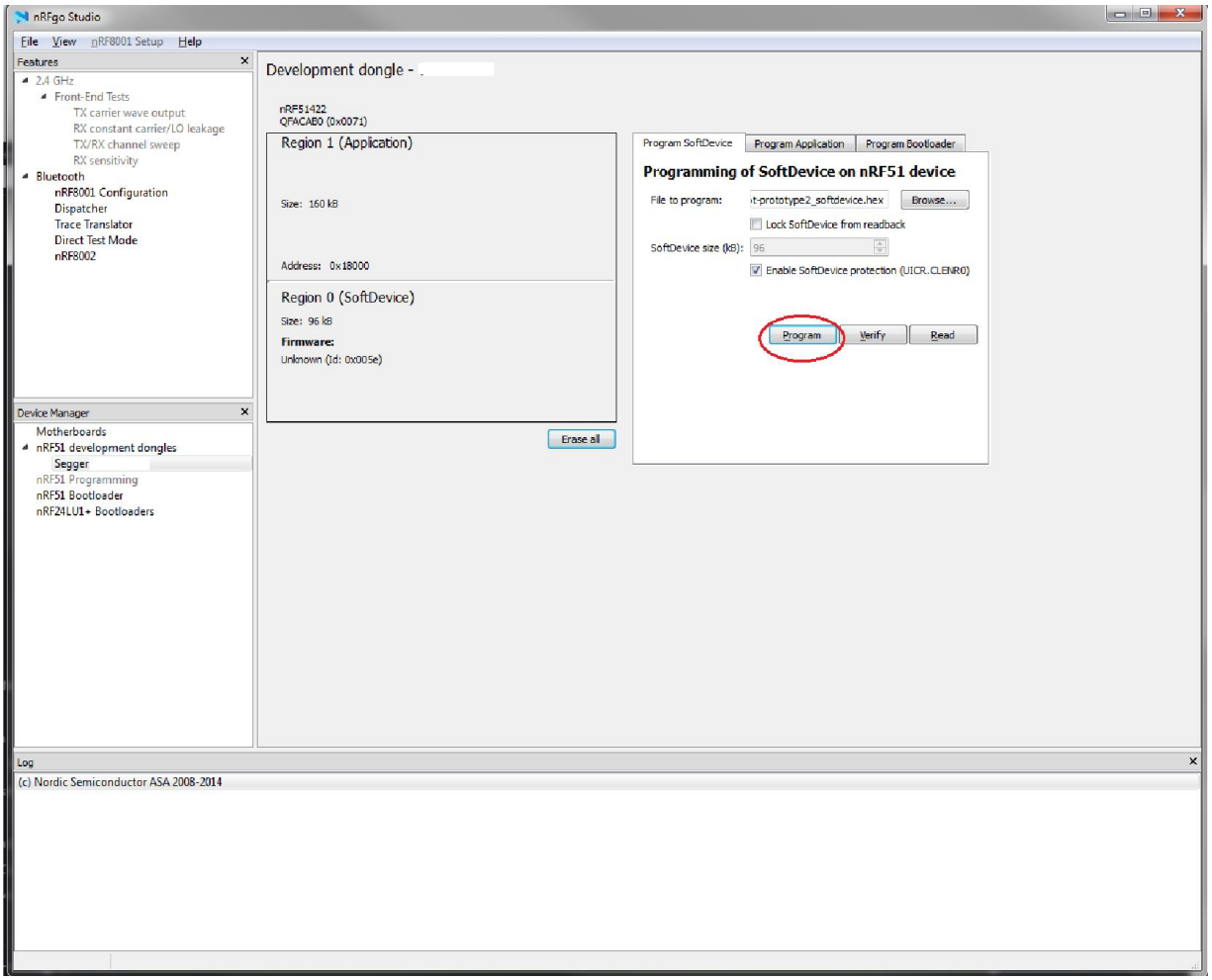
7. Connect your nRF51 Development Kit to the computer.
8. Start nRFGo Studio.

9. Select the connected board:

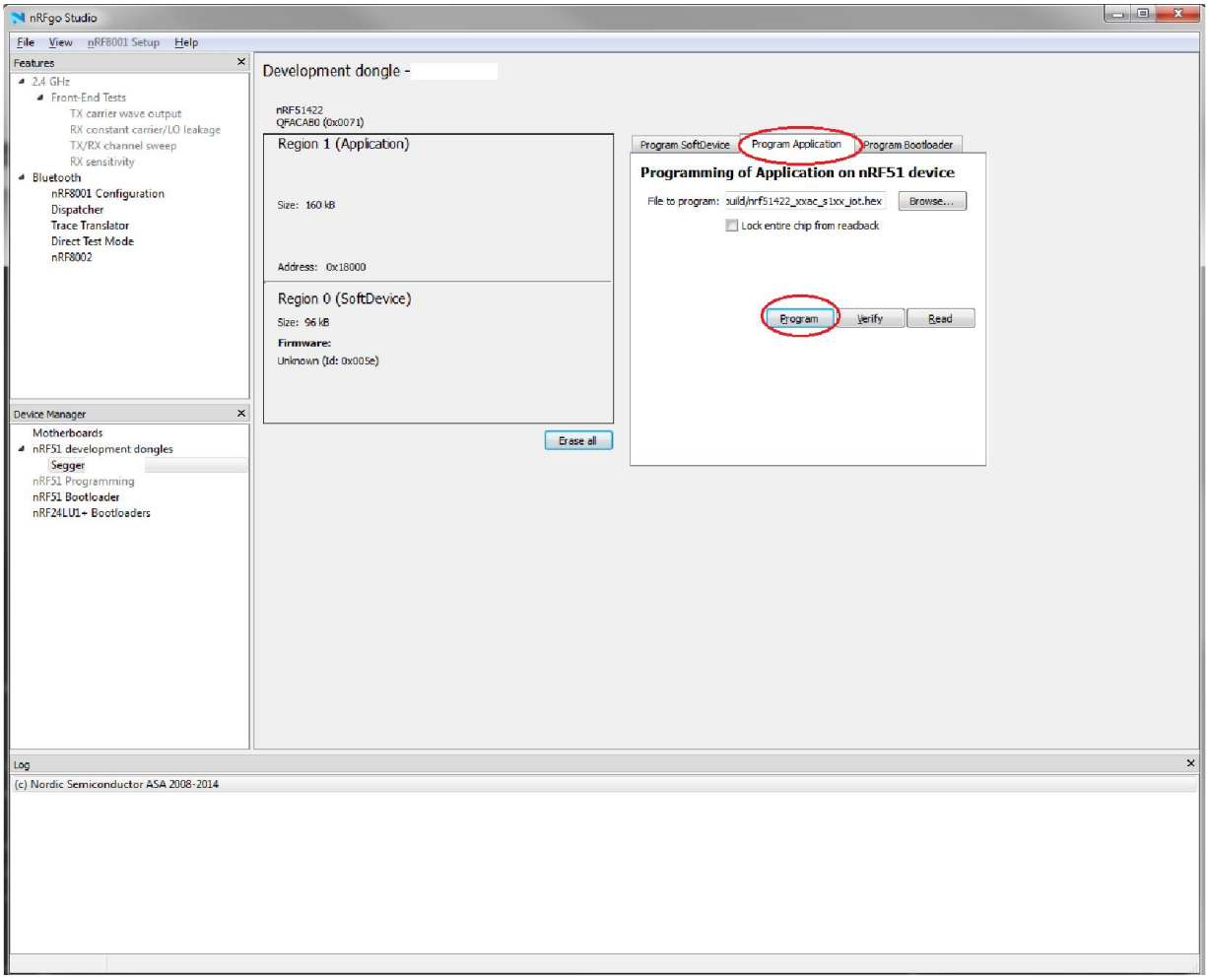


10. On the “Program SoftDevice” tab, click “Browse” and select the SoftDevice that is located in the following folder in the IoT SDK:
`\Nordic\nrf51\components\softdevice\s1xx_iot\s1xx-iot-prototype2_softdevice.hex`

Click “Program”:



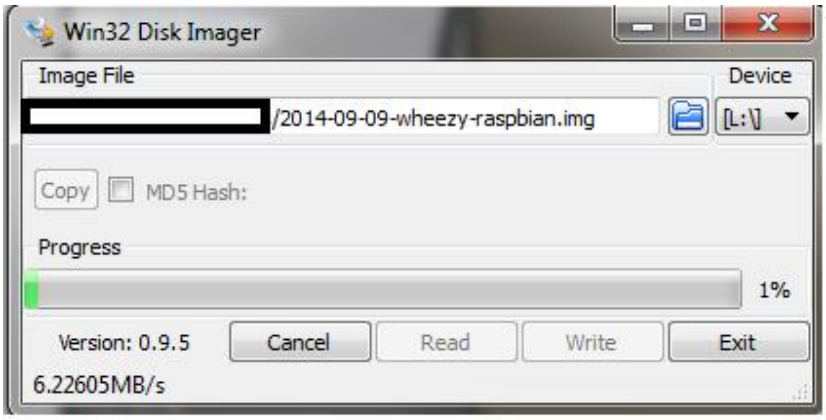
11. Select the “Program Application” tab, click “Browse”, and select the example that you just compiled in Keil: `\Nordic\nrf51\examples\iot\ipv6_coap_client\boards\pca10028\arm\_build\nrf51422_xxac_s1xx_iot.hex`.
    Click “Program” to program this example:



12. LED 1 on the PCA10028 board should now be blinking, which means that the device is advertising. You have successfully prepared the Development Kit.

Setting up the Raspberry Pi and finding its IP address

1. Download and unzip the Raspbian image: http://www.raspberrypi.org/downloads/
2. Insert an SD card into an SD card reader and write the unzipped image to your card. I used Win32 Disk Imager:

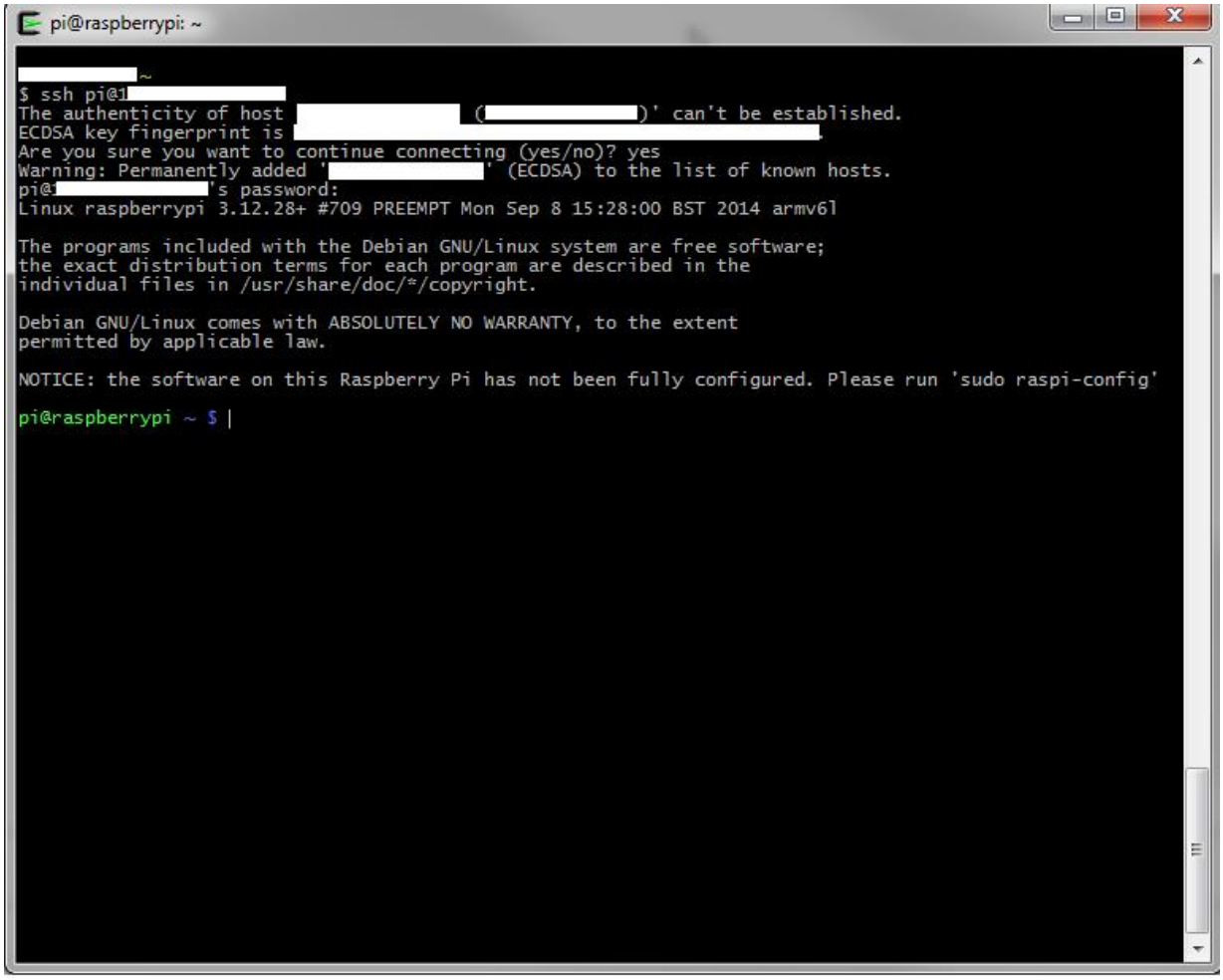


   See http://www.raspberrypi.org/documentation/installation/installing-images/README.md for alternative methods.
3. Now it's time to boot the Raspberry Pi:
   a. Insert the SD card.
   b. Connect a network cable.
   c. Insert the Bluetooth 4 dongle.
   d. Connect a micro USB power cable.
4. Find the IP address of your Raspberry Pi: http://www.raspberrypi.org/documentation/troubleshooting/hardware/networking/ip-address.md

Installing required packages on the Raspberry Pi

The following part is almost 100% run from a terminal accessing the Raspberry Pi, thus it requires some command line skills. But I will try to keep it as simple and step-by-step as possible.

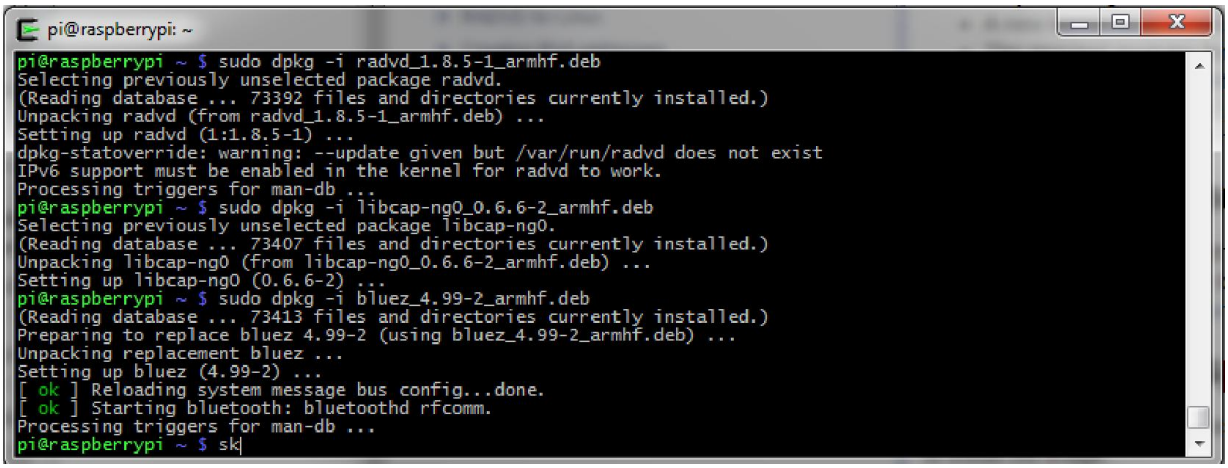
1. Log on to your Raspberry Pi with an ssh terminal using **ssh username@ip-address**. The default username is “pi”, and the default password is “raspberry”.



2. Run **sudo raspi-config** as suggested in the welcome prompt. Select “Expand Filesystem” from the raspi-config menu.
3. Select “Finish” and reboot the Raspberry Pi. Then log in again.
4. Download the required packages to the Raspberry Pi by using the following commands:
    a. **wget http://mirrordirector.raspbian.org/raspbian/pool/main/b/bluez/bluez_4.99-2_armhf.deb**
    b. **wget http://mirrordirector.raspbian.org/raspbian/pool/main/libc/libcap-ng/libcap-ng0_0.6.6-2_armhf.deb**
    c. **wget http://mirrordirector.raspbian.org/raspbian/pool/main/r/radvd/radvd_1.8.5-1_armhf.deb**

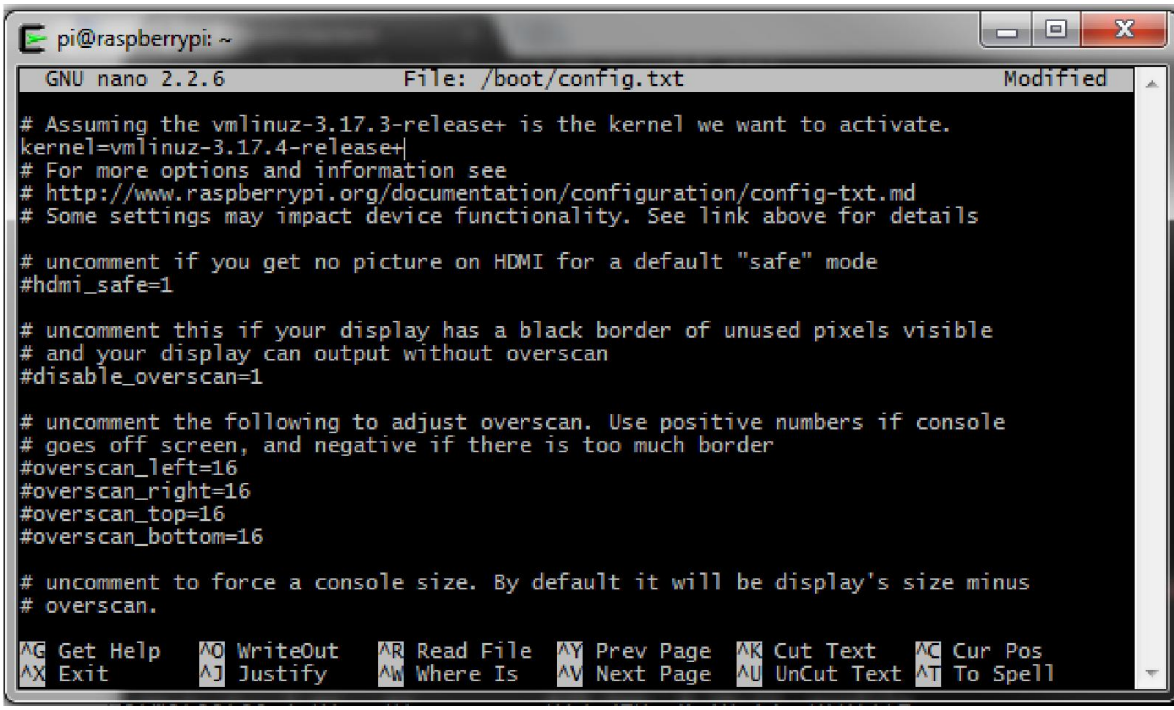
d. **wget -O kernel.zip**
http://www.nordicsemi.com/eng/nordic/download_resource/41602/5/28710770

5. Unzip the kernel package: **unzip kernel.zip**
6. Install the downloaded packages:
   a. **sudo dpkg -i radvd_1.8.5-1_armhf.deb**
   b. **sudo dpkg -i libcap-ng0_0.6.6-2_armhf.deb**
   c. **sudo dpkg -i bluez_4.99-2_armhf.deb**
   d. **sudo dpkg -i linux-image-3.17.4-release+_1_armhf.deb**
   e. **sudo dpkg -i linux-headers-3.17.4-release+_1_armhf.deb**

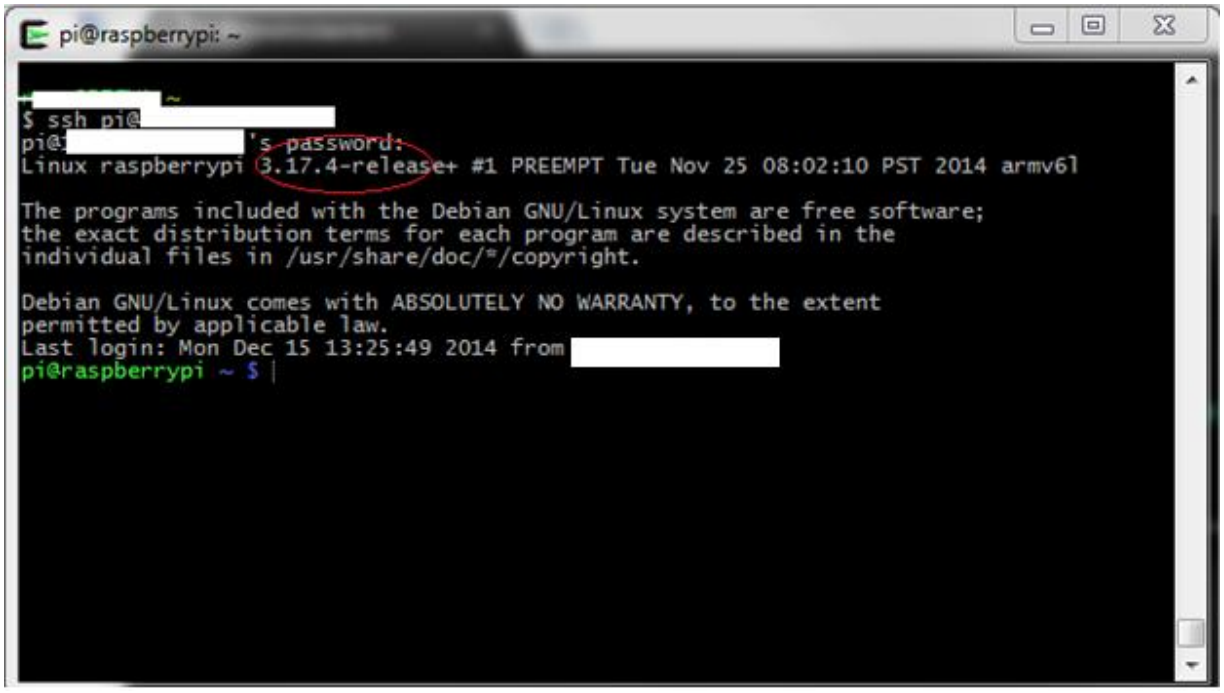


7. Modify Raspbian to boot to the new kernel that you just installed by editing `/boot/config.txt`:
   a. **sudo nano /boot/config.txt**
   b. Add the following line:
      `kernel=vmlinuz-3.17.4-release+ to config.txt`
   c. Save the file and exit.

d. Reboot to the new kernel:
sudo reboot



8. The new kernel is now installed. You can verify this by checking the welcome text that is displayed when you log on to the Raspberry Pi:

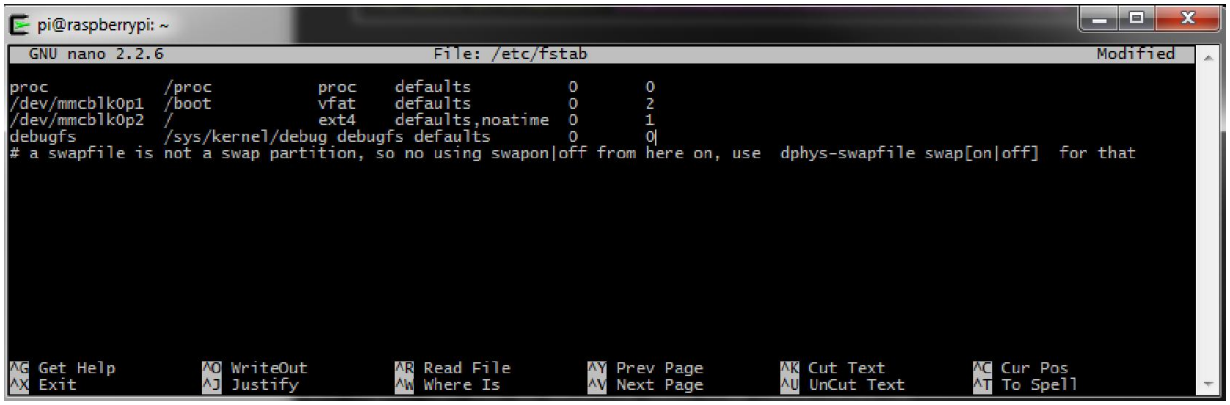


Great! Now we have all software dependencies installed on the Raspberry Pi router. Let's move on to the configuration.

Configuring your Raspberry Pi

Some configuration is needed to set up the Raspberry Pi correctly. The following steps are carried out with root privileges, so ensure that you are exactly following the steps.

1. Log in as root: **sudo su**
2. Edit fstab: **nano /etc/fstab**
3. Configure fstab to mount debugfs automatically when starting up. Add the following line:
   `debugfs /sys/kernel/debug debugfs defaults 0 0`



4. Save the file and exit.
5. Reboot, or mount debugfs manually this time (it will be mounted automatically during startup from now on): **mount –a**
6. Enable the 6LoWPAN module for the kernel: **modprobe bluetooth_6lowpan**
7. Set the PSM number: **echo 35 > /sys/kernel/debug/bluetooth/6lowpan_psm**
8. Set IPv6 forwarding: **echo 1 > /proc/sys/net/ipv6/conf/all/forwarding**
9. Create a radvd configuration file: **nano /etc/radvd.conf**
   a. Insert the following text into `radvd.conf`:

```
interface bt0
{
AdvSendAdvert on;
prefix 2001:db8::/64
 {
  AdvOnLink off;
  AdvAutonomous on;
  AdvRouterAddr on;
 };
};
```

   b. Save the file and exit.

Connecting to your kit

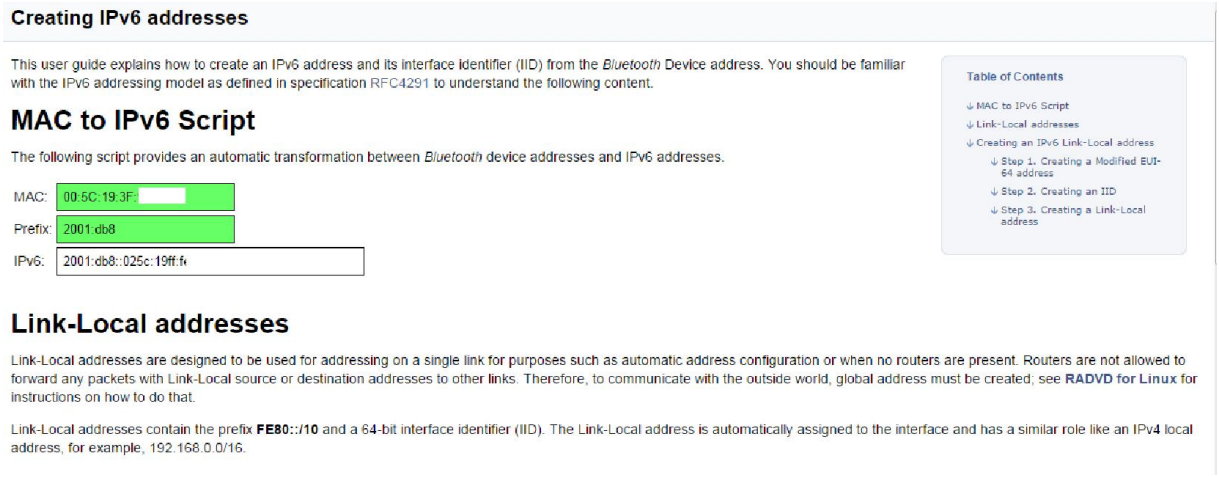
The following steps are carried out with root privileges, so ensure that you are exactly following the steps.

1. Ensure that your kit is up and running, with LED1 blinking.
2. Identify the kit by running the following command: **hcitool lescan**
3. Copy the MAC address from the device that is advertising as nCoap_Client, for example: 00:aa:bb:cc:dd:ee
4. Connect to the nCoap_client:
   echo "connect <MAC address> 1" > /sys/kernel/debug/bluetooth/6lowpan_control
5. Verify that the LED on your board has stopped blinking.
6. Verify that you have established a connection: **hci con**
7. Add the IP prefix to the *Bluetooth* interface bt0: **ifconfig bt0 add 2001:db8::1/64**
8. Restart the radvd service: **service radvd restart**

Pinging your kit

We are now almost ready to interact with the kit over IP.

1. Convert the BLE MAC address that you identified in a previous step to an IPv6 address by using the script in the IoT SDK documentation at **User Guides > Creating IPv6 addresses** (a00034.html):



Creating IPv6 addresses

This user guide explains how to create an IPv6 address and its interface identifier (IID) from the *Bluetooth* Device address. You should be familiar with the IPv6 addressing model as defined in specification RFC4291 to understand the following content.

Table of Contents

- MAC to IPv6 Script
- Link-Local addresses
- Creating an IPv6 Link-Local address
  - Step 1. Creating a Modified EUI-64 address
  - Step 2. Creating an IID
  - Step 3. Creating a Link-Local address

MAC to IPv6 Script

The following script provides an automatic transformation between *Bluetooth* device addresses and IPv6 addresses.

MAC: 00:5C:19:3F:

Prefix: 2001:db8

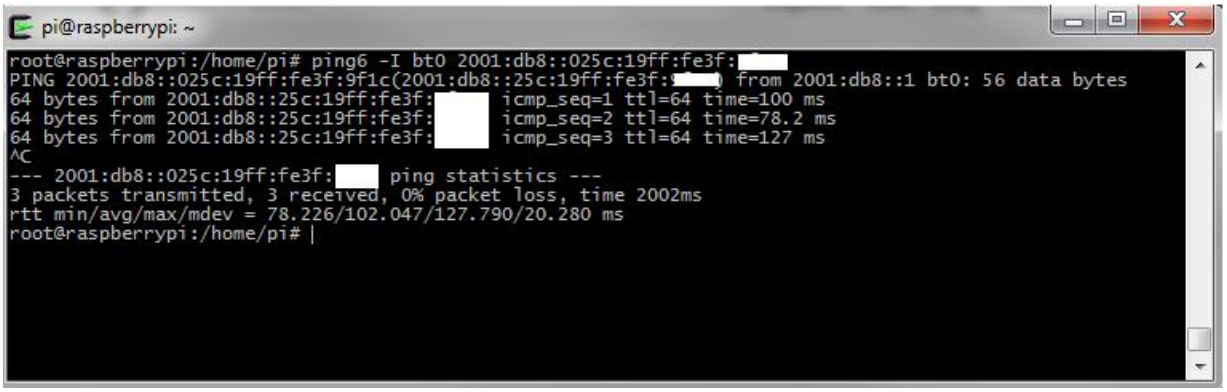
IPv6: 2001:db8::025c:19ff:fe

Link-Local addresses

Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration or when no routers are present. Routers are not allowed to forward any packets with Link-Local source or destination addresses to other links. Therefore, to communicate with the outside world, global address must be created; see **RADVD for Linux** for instructions on how to do that.

Link-Local addresses contain the prefix **FE80::/10** and a 64-bit interface identifier (IID). The Link-Local address is automatically assigned to the interface and has a similar role like an IPv4 local address, for example, 192.168.0.0/16.

2. Ping6 your device: **ping6 -I bt0 <ipv6-address>**



Final words and tips

Congratulations! You now have a simple ping6 up and running for a Raspberry Pi and a PCA10028 device. All the fun starts now!

You probably want to ensure that some of the previous steps are done automatically at each startup. There are many ways to do so; I found rc.local to be useful for this purpose: http://www.raspberrypi.org/documentation/linux/usage/rc-local.md

Please share your findings and recommendations on devzone!