Linux Driver Development for Embedded Processors

Raspberry Pi 4 Practical Labs Setup

Building a Linux Embedded System for the Raspberry Pi 4 Model B

Raspberry Pi 4 Model B is the latest product in the popular Raspberry Pi range of computers. It offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation Raspberry Pi 3 Model B+, while retaining backwards compatibility and similar power consumption. For the end user, Raspberry Pi 4 Model B provides desktop performance comparable to entry-level x86 PC systems.

This product's key features include the high-performance Broadcom BCM2711, 64-bit quad-core Cortex-A72 (ARM v8) processor, dual-display support at resolutions up to 4K via a pair of micro-HDMI ports, hardware video decode at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability (via a separate PoE HAT add-on).

You can see Raspberry Pi 4 Tech Specs at https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/

Raspbian

Raspbian is the recommended operating system for normal use on a Raspberry Pi. Raspbian is a free operating system based on Debian, optimised for the Raspberry Pi hardware. Raspbian comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi. Raspbian is a community project under active development, with an emphasis on improving the stability and performance of as many Debian packages as possible.

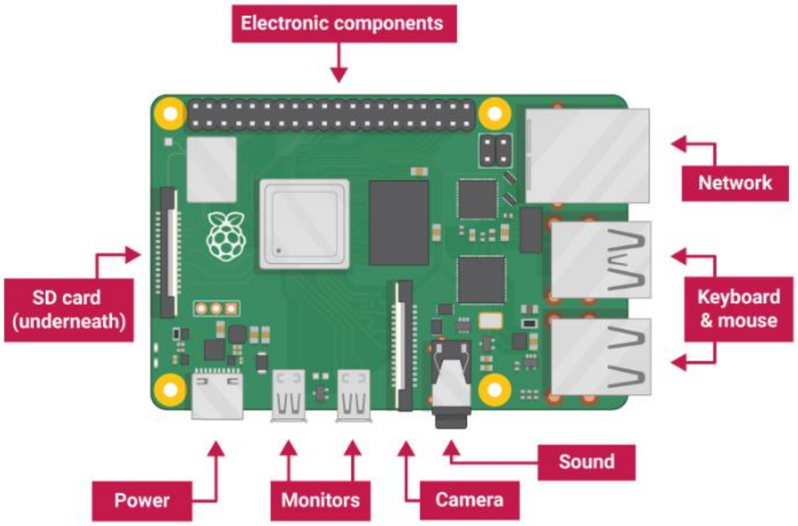
You will install in a SD a **Raspbian** image based on **kernel 4.19.y.** Go to https://downloads.raspberrypi.org/raspbian/images/raspbian-2019-09-30/ and download 2019-09-26-raspbian-buster.zip image.

To write the compressed image on the SD card, you will download and install **Etcher**. This tool, which is an Open Source software, is useful since it allows to get a compressed image as input. More information and extra help is available on the Etcher website at https://etcher.io/

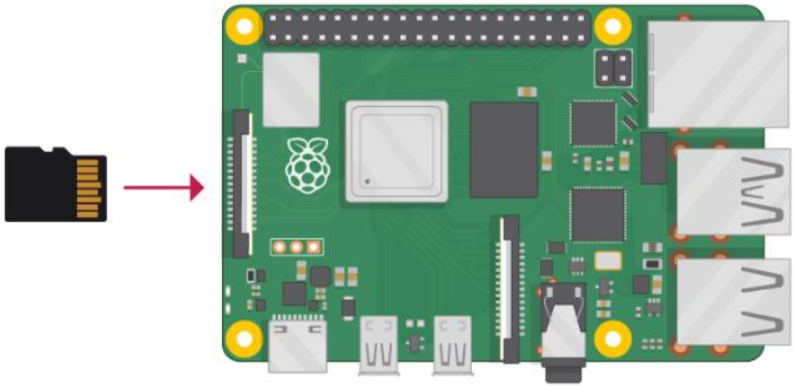
Follow the steps of the Writing an image to the SD card section at https://www.raspberrypi.org/documentation/installation/installing-images/README.md

Connect and Set Up Hardware

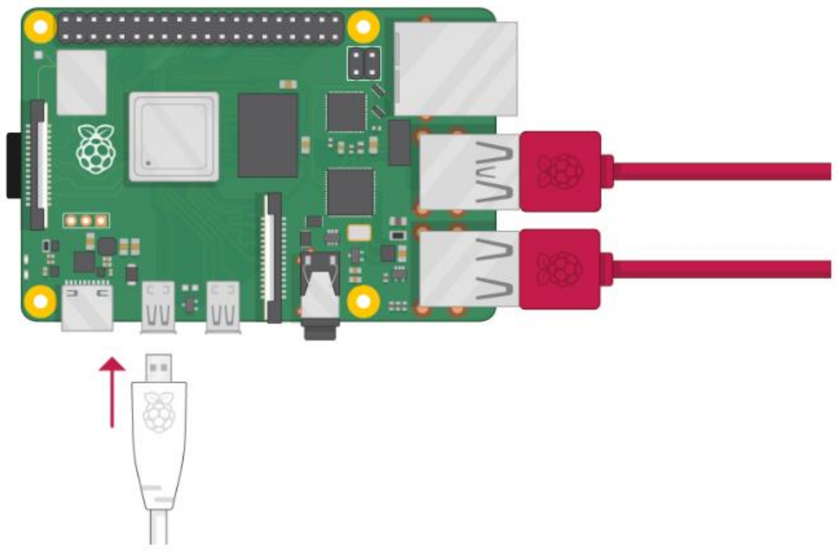
Now get everything connected to your Raspberry Pi 4. It's important to do this in the right order, so that all your components are safe.



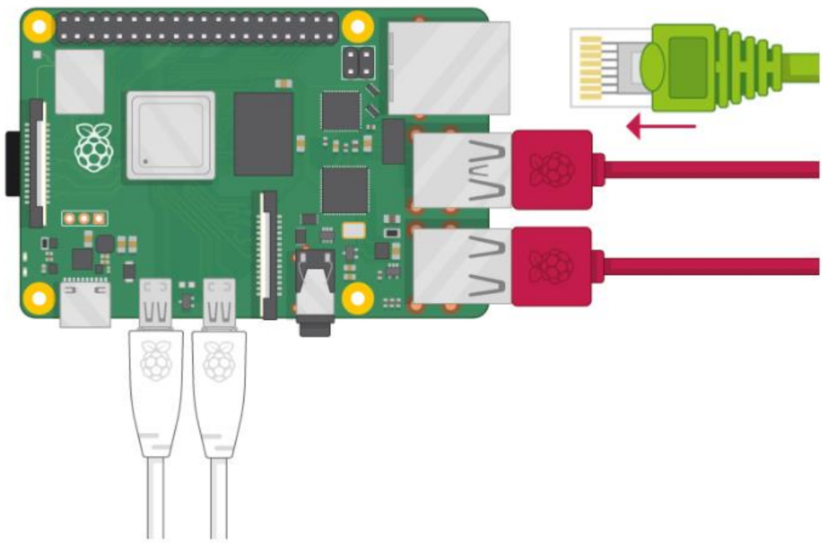
Insert the SD card you've set up with Raspbian into the microSD card slot on the underside of your Raspberry Pi 4.



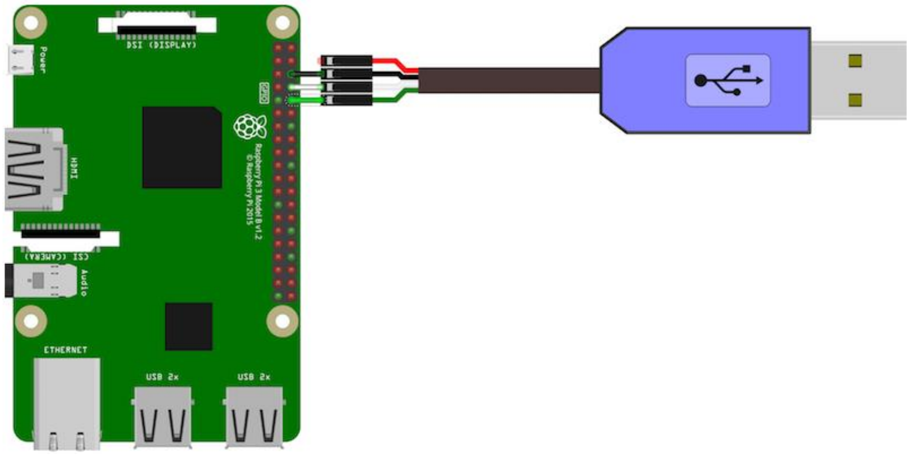
Connect your screen to the first of Raspberry Pi 4's HDMI ports, labelled HDMI0. You can also connect a mouse to an USB port and keyboard in the same way.



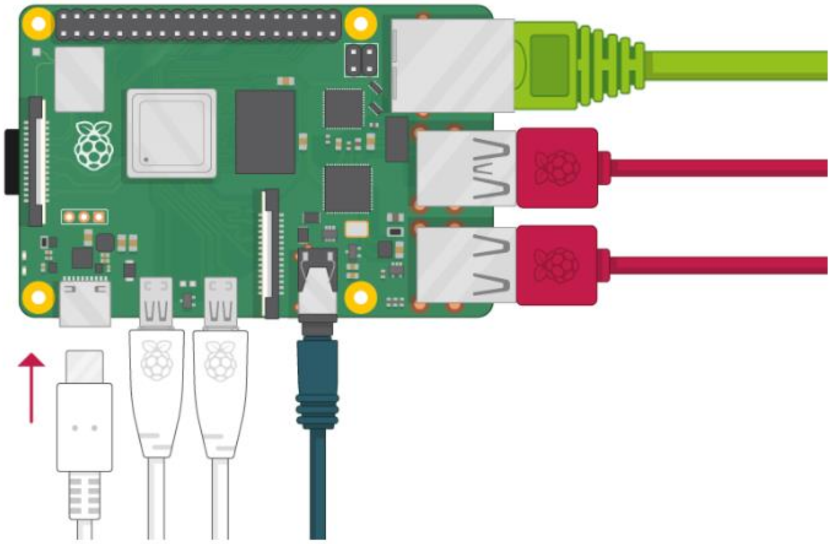
Connect your Raspberry Pi 4 to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on Raspberry Pi 4 to an Ethernet socket on the host PC.



The serial console is a helpful tool for debugging your board and reviewing system log information. To access the serial console, connect a USB to TTL Serial Cable to the device UART pins as shown below.



Plug the USB power supply into a socket and connect it to your Raspberry Pi's power port.



You should see a red LED light up on the Raspberry Pi 4, which indicates that Raspberry Pi 4 is connected to power. As it starts up , you will see raspberries appear in the top left-hand corner of your screen. After a few seconds the Raspbian Desktop will appear.

Launch a terminal on the host Linux PC by clicking on the Terminal icon. Type dmesg at the command prompt:

```
~$ dmesg
```

In the log message you can see that the new USB device is found and installed, for example **ttyUSB0**.

Launch and configure a serial console, for example **minicom** in your host to see the booting of the system. Through this console, you can access and control the Linux based system on the Raspberry Pi 4 Model B. Set the following configuration: **“115.2 kbaud, 8 data bits, 1 stop bit, no parity”**.

Reset the board:

```
pi@raspberrypi:~$ sudo reboot
```

To see Linux boot messages on the console change the loglevel to **8** in the file cmdline.txt under /boot

```
pi@raspberrypi:~$ sudo sudo nano /boot/cmdline.txt // loglevel=8
```

To change your current console_loglevel simply write to this file:

```
pi@raspberrypi:~$ echo <loglevel> > /proc/sys/kernel/printk
```

For example:

```
pi@raspberrypi:~$ echo 8 > /proc/sys/kernel/printk
```

In that case, every kernel messages will appear on your console, as all priority higher than 8 (lower loglevel values) will be displayed. Please note that after reboot, this configuration is reset. To keep the configuration permanently just append following line to /etc/sysctl.conf file in the Raspberry Pi 4:

```
kernel.printk = 8 4 1 3
```

```
pi@raspberrypi:~$ sudo nano /etc/sysctl.conf
```

Setting up Ethernet Communication

Connect an Ethernet cable between your host PC and your Raspberry Pi 4 Model B board. Set up the Linux host PC´s IP Address:

1. On the host side, click on the Network Manager tasklet on your desktop, and select Edit Connections. Choose "Wired connection 1" and click "Edit".
2. Choose the "IPv4 Settings" tab, and select Method as "Manual" to make the interface use a static IP address, like 10.0.0.1. Click "Add", and set the IP address, the Netmask and Gateway as follow:

   Address: 10.0.0.1
   Netmask: 255.255.255.0
   Gateway: none or 0.0.0.0

Finally, click the "Save" button.

3. Click on "Wired connection 1" to activate this network interface.

Copying Files to your Raspberry Pi

You can access the command line of a Raspberry Pi 4 remotely from another computer or device on the same network using SSH. Make sure your Raspberry Pi 4 is properly set up and connected. Configure the eth0 interface with IP address 10.0.0.10:

```
pi@raspberrypi:~$ sudo nano /etc/dhcpcd.conf
# Example static IP configuration:
interface eth0
static ip_address=10.0.0.10/24
```

```
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
static routers=255.255.255.0
```

Save the file typing "Ctrl+O" and exit typing "Ctrl+X".

Raspbian has the SSH server disabled by default. You have to start the service:

```
pi@raspberrypi:~# sudo /etc/init.d/ssh restart
```

Now verify that you can ping your Linux host machine from the Raspberry Pi 4 Model B. Exit the ping command by typing "Ctrl-c".

```
pi@raspberrypi:~# ping 10.0.0.1
```

You can also ping from Linux host machine to the target. Exit the ping command by typing "Ctrl-c".

```
~$ ping 10.0.0.10
```

By default the root account is disabled, but you can enable it by using this command and giving it a password:

```
pi@raspberrypi:~$ sudo passwd root /* set for instance password to "pi" */
```

Now you can log into your pi as the root user. Open the sshd_config file and change **PermitRootLogin** to **yes** (also comment the line out). After editing the file type "Ctrl+x", then type "yes" and press "enter" to exit.

```
pi@raspberrypi:~$ sudo nano /etc/ssh/sshd_config
```

Building the Linux Kernel

There are two main methods for building the kernel. You can build locally on the Raspberry Pi 4, which will take a long time; or you can cross-compile, which is much quicker, but requires more setup. You will use the second method.

Install Git and the build dependencies:

```
~$ sudo apt install git bc bison flex libssl-dev
```

Get the kernel sources:

```
~$ git clone --branch rpi-4.19.y https://github.com/raspberrypi/linux
```

Know what version of RPi you´re running:

```
pi@raspberrypi:~$ uname -a
Linux raspberrypi 4.19.75-v7l+ #1270 SMP Tue Sep 24 18:51:41 BST 2019 armv7l GNU/Linux
```

That means that the date this kernel was build was **24.09.2019**. Then in the kernel repo to to your workstation, run this:

```
~/linux$ git tag
```

And you will get a list of tags with dates. In your case the 'tag: raspberrypi-kernel_1.20190925-1' seems to be the correct one, so check out to this one:

```
~/linux$ git checkout raspberrypi-kernel_1.20190925-1
```

Download the toolchain to the home folder:

```
~$ git clone https://github.com/raspberrypi/tools ~/tools
~$ export PATH=~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-
x64/bin:$PATH
~$ export TOOLCHAIN=~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-
x64/
~$ export CROSS_COMPILE=arm-linux-gnueabihf-
~$ export ARCH=arm
```

Updating the $PATH environment variable makes the system aware of file locations needed for cross-compilation. If you are on a 64-bit host system, you should use:

```
~$ echo PATH=\$PATH:~/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf-raspbian-
x64/bin >> ~/.bashrc
~$ source ~/.bashrc
```

Compile the kernel, modules and device tree files. First execute make mrproper in the kernel source directory:

```
~/linux$ make mrproper
```

Copy config.gz from the Raspberry Pi 4 to the kernel source directory and unpack it:

```
~/linux$ scp pi@10.0.0.10:/proc/config.gz .
~/linux$ gunzip -c config.gz > .config
```

If the config.gz file is missing, run the following command on the Raspberry Pi 4 to load the module that provides it:

```
pi@raspberrypi:~$ sudo modprobe configs
```

Run the following command in the Linux kernel source directory to bootstrap the new kernel configuration from the .config file obtained from Raspberry Pi 4 :

```
~/linux$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make oldconfig
```

Configure the following kernel settings that will be needed during the development of the labs:

```
~/linux$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make menuconfig

Device drivers >
```

```
   [*] SPI support  --->
          <*>   User mode SPI device driver support

Device drivers >
   [*] LED Support  --->
           <*>   LED Class Support
           -*-   LED Trigger support  --->
                        <*>   LED Timer Trigger
                        <*>   LED Heartbeat Trigger

Device drivers >
   <*> Industrial I/O support  --->
          -*-   Enable buffer support within IIO
          -*-   Industrial I/O buffering based on kfifo
          <*>  Enable IIO configuration via configfs
          -*-   Enable triggered sampling support
          <*>   Enable software IIO device support
          <*>   Enable software triggers support
                 Triggers - standalone  --->
                         <*> High resolution timer trigger
                         <*> SYSFS trigger

Device drivers >
    <*> Userspace I/O drivers  --->
          <*>   Userspace I/O platform driver with generic IRQ handling

Device drivers >
   Input device support  --->
          -*- Generic input layer (needed for keyboard, mouse, ...)
          <*>   Polled input device skeleton
```

Save the configuration and exit from menuconfig.

Compile kernel, device tree files and modules in a single step:

```
~/linux$ make -j4 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs
```

Having built the kernel, you need to copy it onto your Raspberry Pi and install the modules; insert a uSD into a SD card reader:

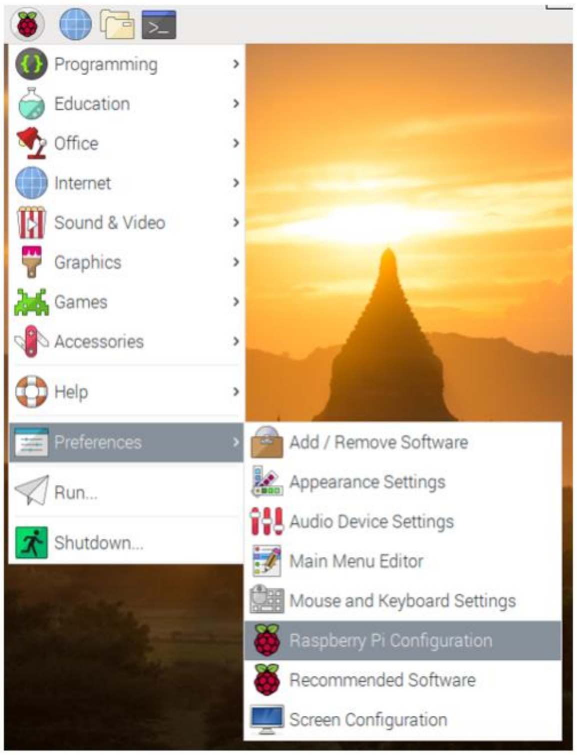
```
~$ lsblk
~$ mkdir ~/mnt
~$ mkdir ~/mnt/fat32
~$ mkdir ~/mnt/ext4
~$ sudo mount /dev/mmcblk0p1 ~/mnt/fat32
~$ sudo mount /dev/mmcblk0p2 ~/mnt/ext4/
~$ ls -l ~/mnt/fat32/ /* see the files in the fat32 partition, check that
config.txt is included */
```

Update the config.txt file adding the next values:

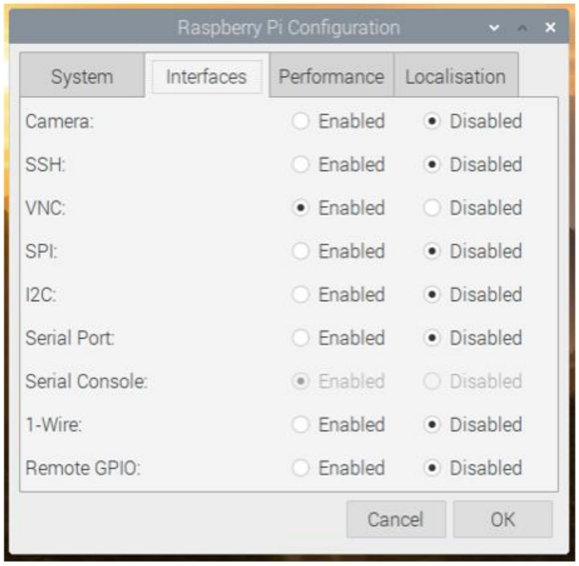
```
~$ cd mnt/fat32/
~/mnt/fat32$ sudo gedit config.txt
```

```
dtparam=i2c_arm=on
dtparam=spi=on
dtoverlay=spi0-cs
# Enable UART
enable_uart=1
```

You can also update previous settings through the Raspberry Pi 4 Configuration application found in Preferences on the menu:



The Interfaces tab is where you turn these different connections on or off, so that the Pi recognizes that you've linked something to it via a particular type of connection:


Raspberry Pi Configuration

Camera: Enabled [✓ Disabled]
SSH: Enabled [✓ Disabled]
VNC: [✓ Enabled] Disabled
SPI: Enabled [✓ Disabled]
I2C: Enabled [✓ Disabled]
Serial Port: Enabled [✓ Disabled]
Serial Console: [✓ Enabled] Disabled
1-Wire: Enabled [✓ Disabled]
Remote GPIO: Enabled [✓ Disabled]

Finally, update kernel, device tree files and modules:

```
~/linux$ sudo cp arch/arm/boot/zImage ~/mnt/fat32/$KERNEL.img
~/linux$ sudo cp arch/arm/boot/dts/*.dtb ~/mnt/fat32/
~/linux$ sudo cp arch/arm/boot/dts/overlays/*.dtb* ~/mnt/fat32/overlays/
~/linux$ sudo cp arch/arm/boot/dts/overlays/README ~/mnt/fat32/overlays/
~/linux$ sudo env PATH=$PATH make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
INSTALL_MOD_PATH=mnt/ext4 modules_install
~$ sudo umount ~/mnt/fat32
~$ sudo umount ~/mnt/ext4
```

If you modify later kernel or device tree files, you can copy them to the Raspberry Pi 4 remotely using SSH:

```
~/linux$ scp arch/arm/boot/zImage root@10.0.0.10:/boot/kernel7l.img
~/linux$ scp arch/arm/boot/dts/bcm2711-rpi-4-b.dtb root@10.0.0.10:/boot/
```

Hardware Descriptions for the Raspberry Pi 4 Model B Labs

Use the same hardware descriptions used for the Raspberry Pi 3 Model B labs.

Software Descriptions for the Raspberry Pi 4 Model B Labs

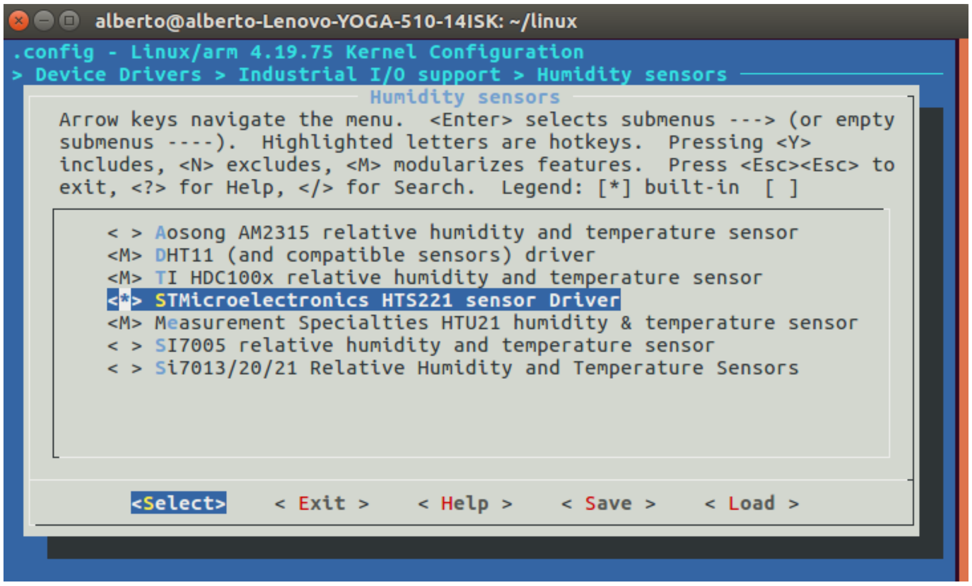
LAB 5.2 Software Description

Change the peripheral base address from 0x3F000000 (Raspberry Pi 3 Model B) to 0xfe000000 (Raspberry Pi 4 Model B).

LAB 12.1 Software Description

Functions for triggered buffer support are needed by this module. If they are not defined accidentally by another driver, there's an error thrown out while linking. To solve this problem, you can recompile the kernel selecting for example the HTS221 driver that includes this triggered buffer support.

```
~/linux$ ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- make menuconfig
```



```
alberto@alberto-Lenovo-YOGA-510-14ISK: ~/linux
.config - Linux/arm 4.19.75 Kernel Configuration
> Device Drivers > Industrial I/O support > Humidity sensors
                        Humidity sensors
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      < > Aosong AM2315 relative humidity and temperature sensor
      <M> DHT11 (and compatible sensors) driver
      <M> TI HDC100x relative humidity and temperature sensor
      <*> STMicroelectronics HTS221 sensor Driver
      <M> Measurement Specialties HTU21 humidity & temperature sensor
      < > SI7005 relative humidity and temperature sensor
      < > Si7013/20/21 Relative Humidity and Temperature Sensors

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

```
~/linux$ make -j4 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage

~/linux$ scp arch/arm/boot/zImage root@10.0.0.10:/boot/kernel7l.img
```

In the Host build the IIO tools:

```
~/linux$ cd tools/iio/
~/linux/tools/iio$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
```

```
    ~/linux/tools/iio$ scp iio_generic_buffer pi@10.0.0.10:
    ~/linux/tools/iio$ scp iio_event_monitor pi@10.0.0.10:
```

The kernel 4.19 modules developed for the Raspberry Pi 4 Model B board are included in the linux_4.19_rpi4_drivers.zip file and can be downloaded from the GitHub repository at https://github.com/ALIBERA/linux_book_2nd_edition.