Linux Driver Development with Raspberry Pi®

Practical Labs

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

Raspberry Pi OS

Raspberry Pi OS is the recommended operating system for normal use on a Raspberry Pi. Raspberry Pi OS is a free operating system based on Debian, optimised for the Raspberry Pi hardware. Raspberry Pi OS comes with over 35,000 packages: precompiled software bundled in a nice format for easy installation on your Raspberry Pi. Raspberry Pi OS 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 on a uSD a **Raspberry Pi OS** image based on **kernel 5.4.y**. Go to https://downloads.raspberrypi.org/raspios_full_armhf/images/raspios_full_armhf-2020-12-04/ and download the 2020-12-02-raspios-buster-armhf-full.zip image.

To write the compressed image on the uSD 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/.

Once the image is installed on the uSD card, you will insert the uSD into the SD card reader on your host PC, then you will enable UART, SPI and I2C peripherals in the programmed uSD. Open a terminal application on your host PC, and type the following commands:

```
~$ lsblk
~$ mkdir ~/mnt
~$ mkdir ~/mnt/fat32
~$ mkdir ~/mnt/ext4
~$ sudo mount /dev/mmcblk0p1 ~/mnt/fat32
```

See the files in the fat32 partition. Check that config.txt is included:

```
~$ ls -l ~/mnt/fat32/
```

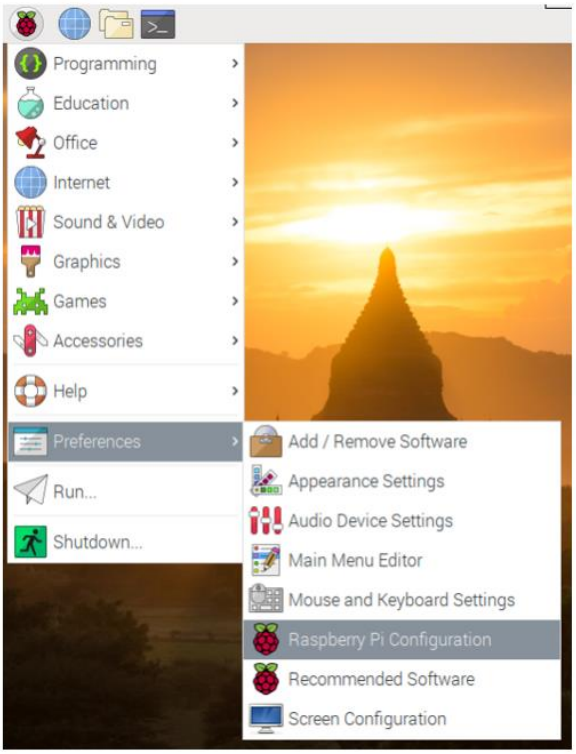
Update the config.txt file, adding the next values:

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

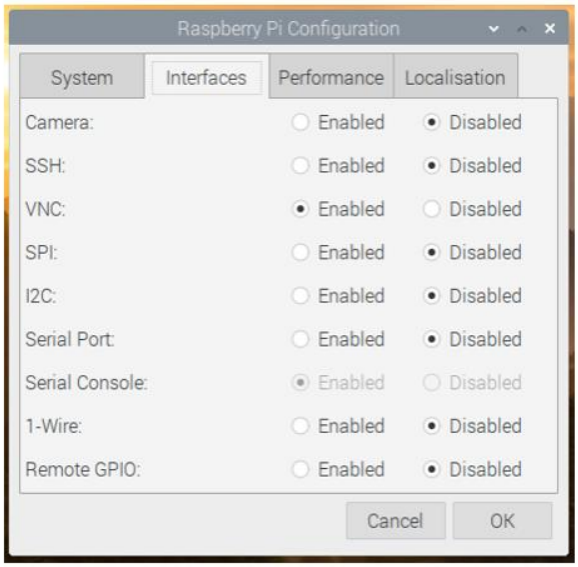
```
dtparam=i2c_arm=on
dtparam=spi=on
dtoverlay=spi0-cs
# Enable UART
enable_uart=1
kernel=kernel7l.img
```

```
~$ sudo umount ~/mnt/fat32
```

You can also update previous settings (after booting the Raspberry Pi 4 board) through the Raspberry Pi 4 Configuration application found in Preferences on the menu.

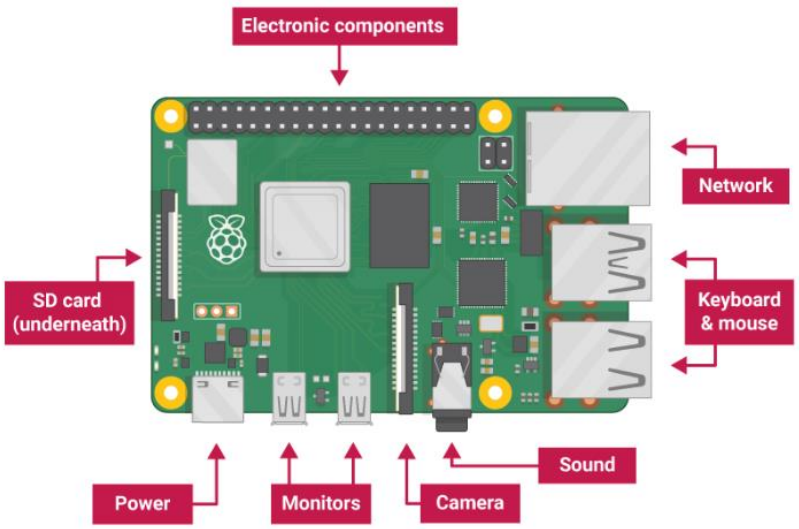


The Interfaces tab is where you turn these different connections on or off, so that the Pi recognizes that you've linked something to it via a particular type of connection:

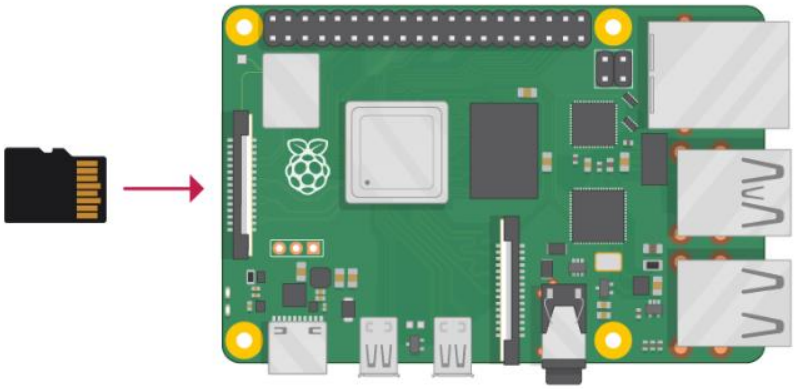


Connect and set up hardware

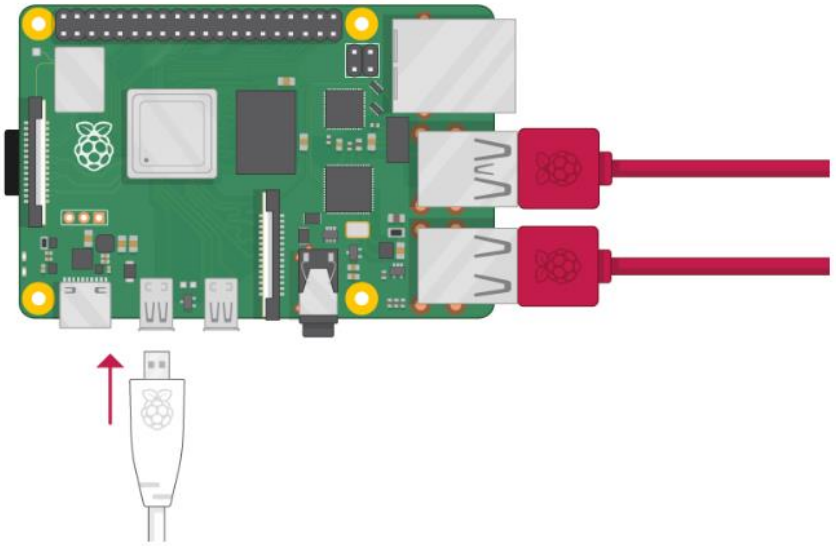
Now get everything connected to your Raspberry Pi 4. It's important to do this in the right order, so that all your components are safe.



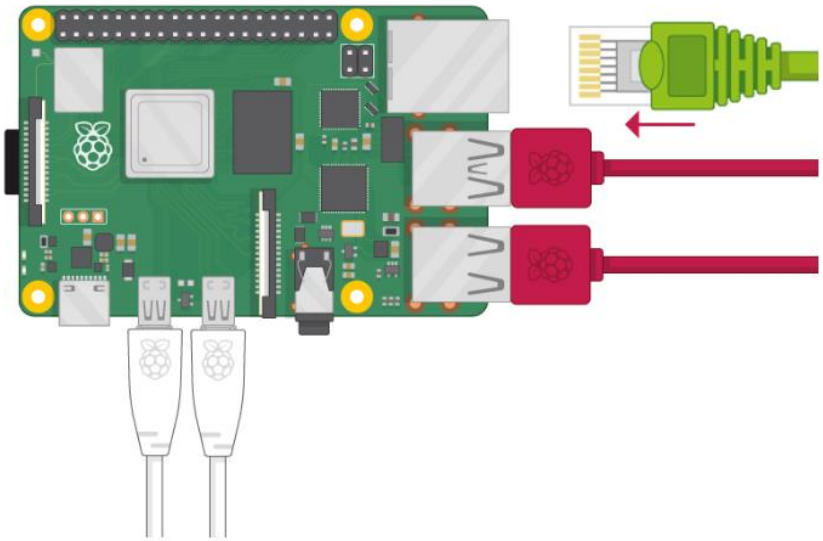
Insert the uSD card you've set up with **Raspberry Pi OS** into the microSD card slot on the underside of your Raspberry Pi 4.



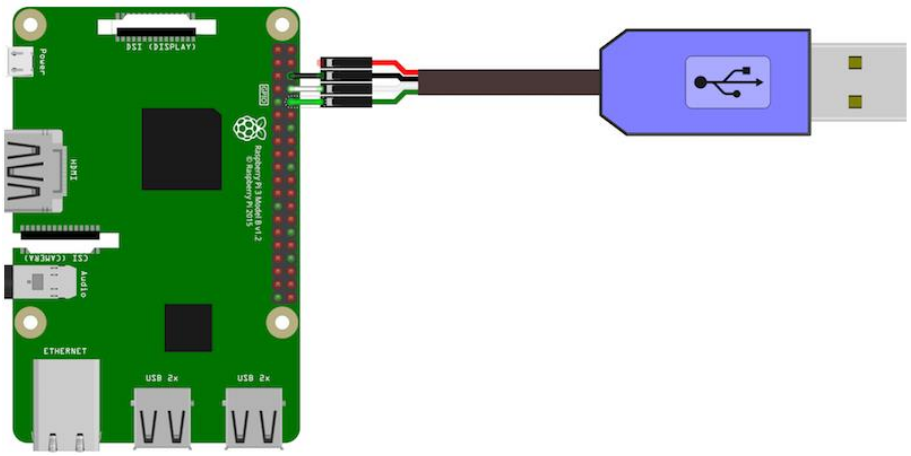
Connect your screen to the first of Raspberry Pi 4's HDMI ports, labelled HDMI0. You can also connect a mouse to a USB port and keyboard in the same way.



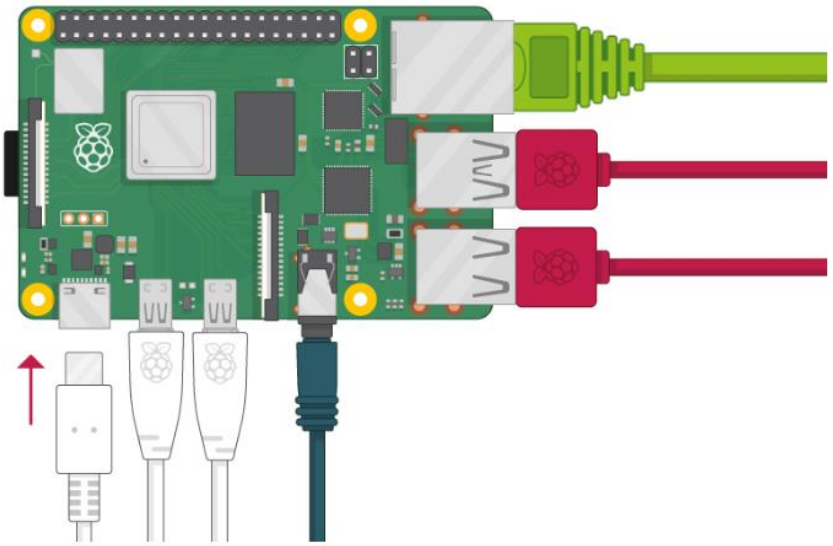
Connect your Raspberry Pi 4 to the internet via Ethernet, use an Ethernet cable to connect the Ethernet port on Raspberry Pi 4 to an Ethernet socket on the host PC.



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 Raspberry Pi OS Desktop will appear.

For the official Raspberry Pi OS, the default user name is **pi**, with password **raspberry**.

To find out the version of the booted kernel, run `uname -r` on the Raspberry Pi terminal:

```
pi@raspberrypi:~$ uname -r
5.4.79-v7+
```

Reset the board. You can disconnect your screen from the Raspberry Pi's HDMI port during the development of the labs (you will only need to connect a screen in the LAB 10.3).

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

To see Linux boot messages on the console, add `loglevel=8` in the file `cmdline.txt` under `/boot`.

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

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
```

Now, every kernel message 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 the following kernel.printk value to the /etc/sysctl.conf file, then reboot the processor:

```
kernel.printk = 8 4 1 3
pi@raspberrypi:~$ sudo nano /etc/sysctl.conf
pi@raspberrypi:~$ sudo reboot
```

Setting up ethernet communication

Connect an Ethernet cable between your host PC and the 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 the Raspberry Pi 4 is properly set up and connected. Configure the eth0 interface with IP address 10.0.0.10:

```
pi@raspberrypi:~$ sudo ifconfig eth0 10.0.0.10 netmask 255.255.255.0
```

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** (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 make libc6-dev libncurses5-dev
```

On your host PC, create the linux_rpi4 folder, where you are going to download the kernel sources:

```
~$ mkdir linux_rpi4
~$ cd linux_rpi4/
```

Get the kernel sources. The git clone command below will download the current active branch without any history. Omitting the --depth=1 will download the entire repository, including the full history of all branches, but this takes much longer and occupies much more storage:

```
~/linux_rpi4$ git clone --depth=1 -b rpi-5.4.y
https://github.com/raspberrypi/linux
```

Install the 32-bit toolchain for a 32-bit kernel:

```
~$ sudo apt install crossbuild-essential-armhf
```

Compile the kernel, modules and device tree files. First, apply the default configuration:

```
~/linux_rpi4/linux$ KERNEL=kernel7l

~/linux_rpi4/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-
bcm2711_defconfig
```

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

```
~/linux_rpi4/linux$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

Device drivers >
```

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

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
```

For the LAB 12.1, you will need the functions that enable the triggered buffer support. If they are not defined accidentally by another driver, there's an error thrown out while linking. To solve this problem, you can select, for example, the HTS221 driver, which includes this triggered buffer support:

```
Device drivers >
   <*> Industrial I/O support  > Humidity sensors --->
          <*> STMicroelectronics HTS221 sensor Driver
```

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 device and also install the modules. Insert the previously programmed uSD into the SD card reader on your host PC, and execute the following commands:

```
~$ lsblk
~$ mkdir ~/mnt
~$ mkdir ~/mnt/fat32
~$ mkdir ~/mnt/ext4
~$ sudo mount /dev/sdd1 ~/mnt/fat32/
~$ sudo mount /dev/sdd2 ~/mnt/ext4/
```

```
~/linux_rpi4/linux$ sudo env PATH=$PATH make ARCH=arm CROSS_COMPILE=arm-linux-
gnueabihf- INSTALL_MOD_PATH=~/mnt/ext4 modules_install
```

Finally, update kernel, device tree files and modules:

```
~/linux_rpi4/linux$ sudo cp ~/mnt/fat32/kernel7l.img ~/mnt/fat32/kernel7l-
backup.img
~/linux_rpi4/linux$ sudo cp arch/arm/boot/zImage ~/mnt/fat32/kernel7l.img
~/linux_rpi4/linux$ sudo cp arch/arm/boot/dts/*.dtb ~/mnt/fat32/
~/linux_rpi4/linux$ sudo cp arch/arm/boot/dts/overlays/*.dtb*
~/mnt/fat32/overlays/
~/linux_rpi4/linux$ sudo cp arch/arm/boot/dts/overlays/README
~/mnt/fat32/overlays/
~$ sudo umount ~/mnt/fat32
~$ sudo umount ~/mnt/ext4
```

To find out the version of your new kernel, boot the system and run `uname -r`:

```
pi@raspberrypi:~$ uname -r
5.4.83-v7+
```

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

```
~/linux_rpi4/linux$ scp arch/arm/boot/zImage root@10.0.0.10:/boot/kernel7l.img
~/linux_rpi4/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 of the Raspberry Pi 3 labs developed through this book.