Design Your Own Raspberry Pi Compute Module PCB

by magkopian

If you've never heard of the Raspberry Pi Compute Module before, it's basically a fully fledged Linux computer with the form factor a laptop RAM stick!

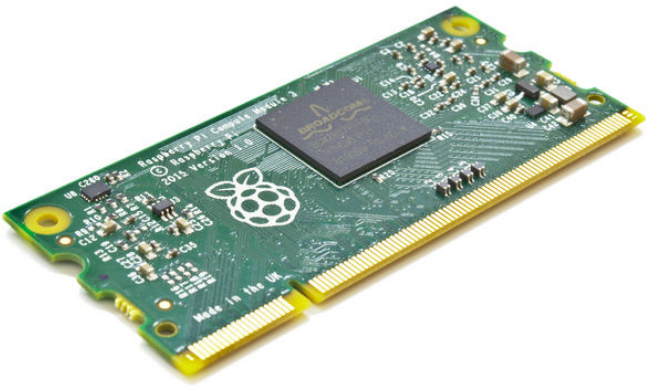
With it becomes possible to design your own custom boards where the Raspberry Pi is just another component. That gives you an enormous amount of flexibility as it allows you to have access to a much greater amount of IO pins, while the same time you get to choose exactly what hardware you want on your board. The on-board eMMC also eliminates the need for an external micro SD card, which makes the Compute Module perfect for designing Raspberry Pi based products.

Unfortunately, while the Compute Module allows you to do all this it still appears to be lacking in terms of popularity compared to the traditional Raspberry Pi Model A and B. As a result, there aren't many open source hardware projects out there based on it. And for anyone who might want to get started with designing their own boards the amount of resources they have is rather limited.

When I first got started with the Raspberry Pi Compute Module a few months ago, that was exactly the issue I was faced with. So, I decided to do something about it. I decided to design an open source PCB based on the Compute Module, that is going to have all the basic features that make the Raspberry Pi great. That includes a camera connector, USB host, audio output, HDMI and of course a GPIO header compatible with the regular Raspberry Pi boards.

The goal of this project is to provide an open source design for a Compute Module based board, that anyone will be able to use as a starting point for designing their own custom board. The board was designed on KiCAD, an open source and cross platform EDA software package, in order to allow as many people as possible to take advantage of it.

Simply grab the design files, adapt them to your needs and spin your own custom board for your project.



Step 1: Parts and Tools

To get started with the Raspberry Pi Compute Module you are going to need the following parts:

1 x **Raspberry Pi Compute Module 3** - I highly recommend getting the regular version which includes the on-board eMMC and **not** the Lite version. If you want to use the Lite version in your project you'll have to make a few changes to the design, and that includes adding a micro SD card connector. Finally, I have only tested the board with the CM3 and I cannot guarantee that it will work with the first CM version that was released back in 2014.

Update 29/1/2018:
It appears that the Foundation has just released the Compute Module 3+ and not only that, but now it also comes with the option for an 8GB, 16GB or 32GB eMMC! According to the datasheet, it appears that the CM3+ is **electrical identical** to the CM3 which means that it's basically a drop in replacement for the CM3.

1 x **Compute Module IO Board** - My design was intended to serve as a starting point for designing your own custom board based on it, not to be replacement for the Compute Module IO board. So, to make your life easier I highly recommend getting your hands to an IO board and use that for development before moving to a custom board. Apart from giving you access to every single pin of the CM plus a variety of connectors, the IO board is also needed for flashing the on-board eMMC. Which is something that you cannot do with my board, unless you do some changes to the design first.

1 x **Raspberry Pi Zero Camera Cable** or **Compute Module Camera Adapter** - On my design I'm using a very similar camera connector to the one used by the Compute Module IO Board and the Raspberry Pi Zero. So, in order to attach a camera you're either going to need an adapter cable designed for the Pi Zero or the camera adapter board that comes together with the Compute Module Development Kit. As far as I know, purchasing the adapter board separately is quite expensive. So, if you like me decided to buy your CM and IO Board separately to save some money, I advise you to get the camera adapter cable designed for the Pi Zero instead.

1 x **Raspberry Pi Camera Module** - I have only tested the board with the original 5MP camera module and not the newer 8MP version. But since the former appears to be working just fine I see no reason the later wouldn't as it is supposed to be backwards compatible. Either way, the 5MP version can be found for less than 5€ on eBay nowadays which is why I'd recommend getting one.

4 x **Female to Female Jumper Wires** - You are going to need at least 4 for configuring the camera connector on the IO board, you'll likely want to get more though. They aren't needed for the custom board but can be useful if you plan attaching any external hardware via the GPIO header.

1 x **HDMI Cable** - I decided to use a full size HDMI connector on my board to eliminate the need for adapters. Of course, if you prefer using a mini or even a micro HDMI connector feel free to adapt the design to your needs.

1 x **5V Micro USB Power Supply** - Your phone charger should probably do just fine for most cases as long as it can provide at least 1A. Keep in mind that this is just a general value, your actually power requirements are going to depend on the hardware you decide to include on your custom board.

in fact the **1473005-4**. Don't make the same mistake I did and get the mirrored version, these connectors aren't cheap.

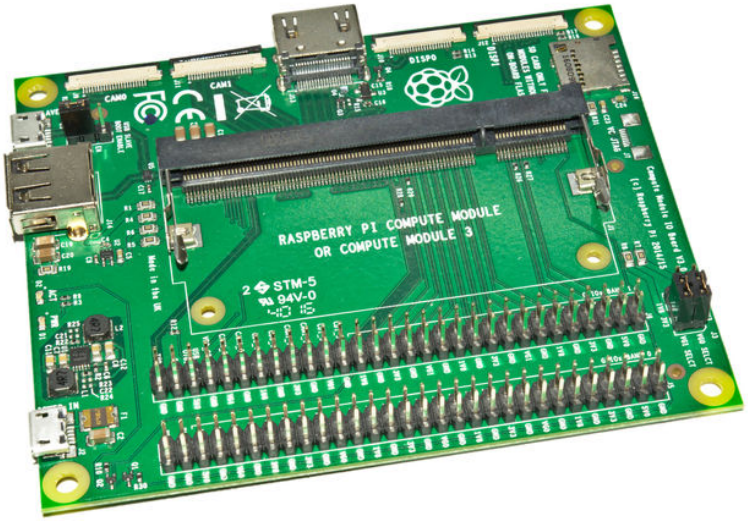
For the rest of the parts that I choose to include on the board you can take a look at the BOM to get more information, I tried to include links to the datasheets for most of them.

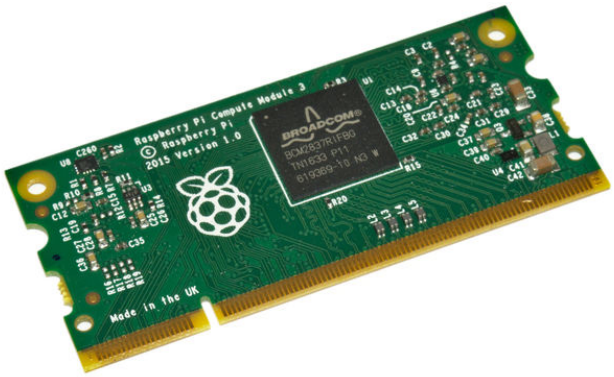
Soldering Equipment - The smallest components on the board are the 0402 decoupling capacitors, but the HDMI as well as the camera and the SODIMM connectors can also be a bit challenging without any kind of magnification. If you have good experience with SMD soldering thought it shouldn't be a big issue. Either way, if you happen to have access to a microscope I highly recommend it.

1 x **USB Ethernet Adapter** - If you plan installing or updating pretty much any package on your system, you're going to need at least temporary Internet access. A 2-in-1 Ethernet adapter plus USB hub is probably a good combo as you only have one USB port available. Personally I use the Edimax EU-4208 which works out of the box with the Pi and doesn't require external power, but it doesn't have a USB hub built in. If you are looking into buying a USB Ethernet adapter here you can find a list with ones that have been tested with the Raspberry Pi.

If you want to add more USB ports and even Etherent directly on your custom board, I'd suggest having a look at the **LAN9512** from Microchip. It is the same chip used by the original Raspberry Pi Model B and is going to give you 2 USB ports and 1 Ethernet port. Alternatively, if you need 4 USB ports consider having a look at its cousin **LAN9514**.

1 x **DDR2 SODIMM RAM Connector** - This is probably the most important component of the entire board and likely the only one that can't be easily substituted. To save you from the trouble the part that you should get is the **TE CONNECTIVITY 1473005-4**. It is available from most major suppliers including T ME, Mouser and Digikey, so you should have no problem finding it. Be very careful though, double check and make sure that the part you're ordering is





Step 2: Flashing the EMMC

First thing you need to do before you start using your Compute Module is flashing the latest Raspbian Lite image on the eMMC. The official Raspberry Pi documentation is very well written and describes the whole process in great detail for both Linux and Windows. For that reason I'm only going to describe the steps you need to take very briefly on Linux, so they can serve as a quick reference.

First of all, you need to make sure that you have your IO board set to **programming mode** and the Compute Module is inserted to the SODIMM connector. To set the board to programming mode move the **J4** jumper to the **EN** position.

Next, you are going to need to build the **rpiboot** tool on your system so you can use it to obtain access to the eMMC. To do so, you need a copy of the usbboot repository which can be obtained easily using **git** as follows,

```
git clone --depth=1 https://github.com/raspberrypi/usbboot && cd usbboot
```

Now, in order to build rpiboot you need to make sure that both **libusb-1.0-0-dev** and **make** packages are installed on your system. So, assuming you're on a Debian based distro such as Ubuntu run,

```
sudo apt update && sudo apt install libusb-1.0-0-dev make
```

If you don't use a Debian based distro the name of the **libusb-1.0.0-dev** package might be different, so make sure to find how it is called in your case. Once the build dependencies are installed you can build the **rpiboot** binary simply by running,

```
make
```

After the built is complete run **rpiboot** as root and it will start to wait for a connection,

```
sudo ./rpiboot
```

Now plug the IO board to your computer by connecting a micro USB cable to its **USB SLAVE** port and then apply power to the **POWER IN** port. After a few seconds the **rpiboot** should be able to detect the Compute Module and allow you access to the eMMC. That should result to a new block device appearing under /**dev**. You can use the **fdisk** program to help you find the name of the device,

```
sudo fdisk -l
```

```
Disk /dev/sdi: 3.7 GiB, 3909091328 bytes, 7634944 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x8e3a9721</p><p>Device     Boot  Start     End Sectors  Size Id Type
/dev/sdi1          8192  137215  129024   63M  c W95 FAT32 (LBA)
/dev/sdi2        137216 7634943 7497728  3.6G 83 Linux
```

In my case it was /**dev**/**sdi** as I have quite a few drives already attached on my system, but yours will most definitely vary.

After you've been absolutely sure you have found the **correct** device name, you can use **dd** in order to burn the Raspbian Lite image to the eMMC. Before doing that though, make sure that there aren't any partition of the eMMC already mounted on your system.

```
df -h
```

If you find any **unmount** them as follows,

```
sudo umount /dev/sdXY
```

Now be extremely careful, using the wrong device name with **dd** can potentially **destroy your system** and **cause data loss**. Do not proceed with the next step unless you're completely sure that you know what you're doing. If you need any more information please have a look at the documentation regarding this.

```
sudo dd if=<date>-raspbian-stretch-lite.img of=/dev/sdX bs=4M && sync
```

Once the **dd** and **sync** commands finish, you should be able to unplug the IO board from your computer. Finally, don't forget to move the **J4** jumper back to the **DIS** position and your Compute Module should be ready for its first boot.

Step 3: First Boot

Before booting for the first time make sure to plug a **USB keyboard** and an **HDMI monitor** to your IO board. If everything goes as expected and your Pi finishes booting, having them attached will allow you to interact with it.

When you're prompted to login use "**pi**" for the username and "**raspberry**" for the password as these are the default login credentials. You can now run some commands to make sure that everything is working as expected as you'd normally do on any Raspberry Pi, but don't attempt installing anything yet as you still don't have an Internet connection.

An important thing you need to do before shutting your Pi down is enabling **SSH**, so you can connect to it from your computer after the next boot. You can do that very easily using the **raspi-config** command,

```
sudo raspi-config
```

To enable SSH go to **Interfacing Options**, select **SSH**, choose **YES**, **OK** and **Finish**. In case you're asked if you want to reboot decline. After you are done shutdown your Pi and once it finishes remove the power.

```
sudo shutdown -h now
```

Next, you need to establish an **Internet connection** using the **USB Ethernet adapter** that you should already have. If your adapter also features a USB hub you can use it to plug your keyboard if you like, otherwise you can just connect to your Pi over SSH. Either way, keep the HDMI monitor **plugged** at least for now, to make sure that the boot process finishes as expected.

Also, near the end it should also show you the **IP address** that your Pi got from the DHCP server. Try using this to connect to your Pi via SSH.

```
ssh pi@<ip-address-of-pi>
```

After successfully connecting to your Pi over SSH you no longer need the monitor and the keyboard plugged in, so feel free to unplug them if you like. At this point you should also have access to the Internet from your Pi, you can try pinging something like google.com to verify it. After making sure that you have access to the Internet it's a good idea to update the system by running,

```
sudo apt update && sudo apt upgrade
```

Step 4: Configuring the Camera

The biggest difference between a regular Raspberry Pi board and the Compute Module is that in the case of the later apart from just enabling the camera by using **raspi-config**, you also need a custom **device tree file**.

You can find more information regarding the configuration of the Compute Module for use with a camera in the doc umentation. But in general, the camera connector among the others also features 4 **control pins**, which need to be connected to 4 **GPIO pins** on the Compute Module, and it is up to you to decide which ones while designing your custom board.

In my case, while designing the board I choose **CD1_SDA** to go to **GPIO28**, **CD1_SCL** to **GPIO29**, **CAM1_IO1** to **GPIO30** and **CAM1_IO0** to **GPIO31**. I choose these particular GPIO pins as I wanted to have a 40 pin GPIO header on my board, that also maintains compatibility with the GPIO connector of the regular Raspberry Pi boards. And for that reason I had to make sure that the GPIO pins I'm using for the camera don't also appear in GPIO header.

So, unless you decide to make changes on the wiring of the camera connector, you need a /**boot**/**dt-blob.bin** that tells your Pi to configure **GPIO28-31** as described above. And in order to generate a **dt-blob.bin**, which is a binary file, you need a **dt-blob.dts** to compile. To make things easy I'm going to provide my own **dt-blob.dts** for you to use which you can then adapt to your needs if you have to.

To compile the device tree file use the device tree compiler as follows,

```
dtc -I dts -O dtb -o dt-blob.bin dt-blob.dts
```

I'm not sure why but the above should result to quite a few warnings, but as long as the **dt-blob.bin** has been generated successfully everything should be fine. Now, move the **dt-blob.bin** you just generated to /**boot** by executing,

```
sudo mv dt-blob.bin /boot/dt-blob.bin
```

The above will probably give you the following warning,

```
mv: failed to preserve ownership for '/boot/dt-blob.bin': Operation not permitted
```

This is just mv complaining that it can't preserve the file ownership as /**boot** is a FAT partition which is to be expected. You may have noticed that /**boot**/**dt-blob.bin** doesn't exist by default, this is because the Pi uses a built in device tree instead. Adding your own inside /**boot** though overrides the built in one and allows you to configure the function of its pin the way you like. You can find more about the device tree in the documentation.

After that is done you need to enable the camera,

```
sudo raspi-config
```

Go to **Interfacing Options**, select **Camera**, choose **YES**, **OK** and **Finish**. In case you're asked if you want to reboot decline. Now, shutdown your Pi and remove the power.

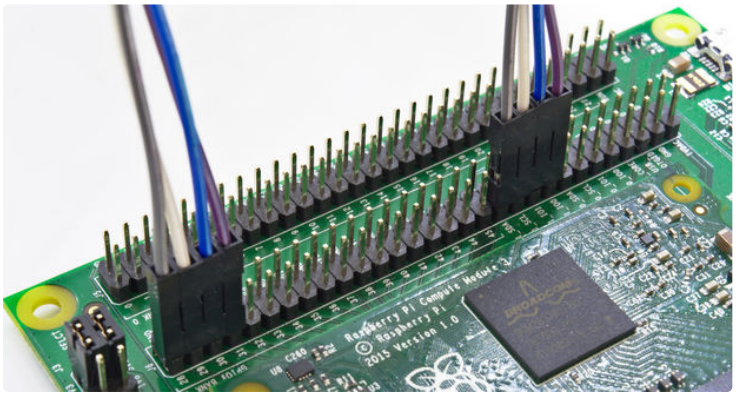
After the power has been removed from the IO board, using 4 female to female **jumper wires** connect the pins for **GPIO28** to **CD1_SDA**, **GPIO29** to **CD1_SCL**, **GPIO30** to **CAM1_IO1** and **GPIO31** to **CAM1_IO0**. Finally, attach your camera module to the **CAM1** connector using the camera adapter board or a camera cable designed for the Raspberry Pi Zero and apply power.

If everything worked as expected after the Pi boots you should be able to utilize the camera. To try to take a picture after connecting to your Pi via SSH run,

```
raspistill -o test.jpeg
```

If the command finishes with no errors and a **test.jpeg** gets created it means that it worked. If you want to have a look at the picture you just took you can connect to your Pi via SFTP and trasfer it to your computer.

```
sftp pi@<ip-address-of-pi>
sftp> get test.jpeg
sftp> exit
```



https://www.instructabl...

Download

Step 5: Moving From the IO Board to a Custom PCB

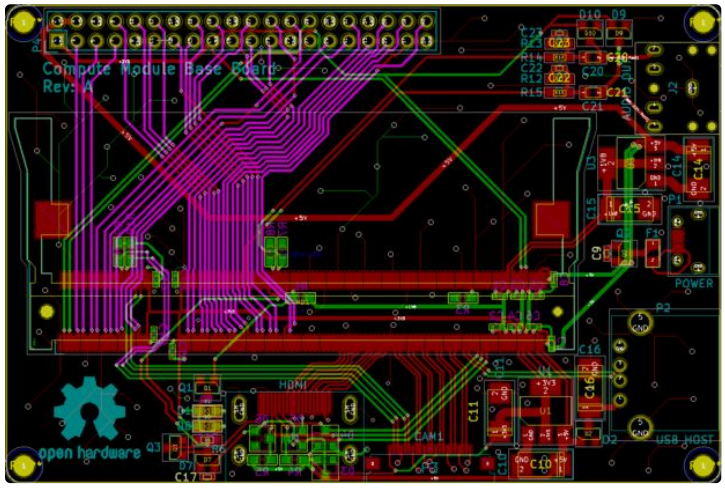
Now that you are done with all the basic configuration you can move to designing your own custom board based on the Compute Module. Since this is going to be your first project, I highly encourage you to grab my design and extend it to include any additional hardware you like.

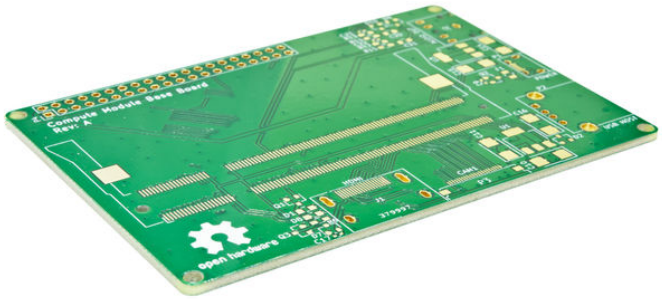
The back of the board has plenty of space for adding your own components and for relatively small projects you likely don't even have to increase the dimensions of the board. Also, in case this is a standalone project and you don't need a physical GPIO header on your board, you can easily get rid of it and save some space on the top side of the PCB. The GPIO header is also the only component that is routed through the second inner layer and removing it frees it up completely.

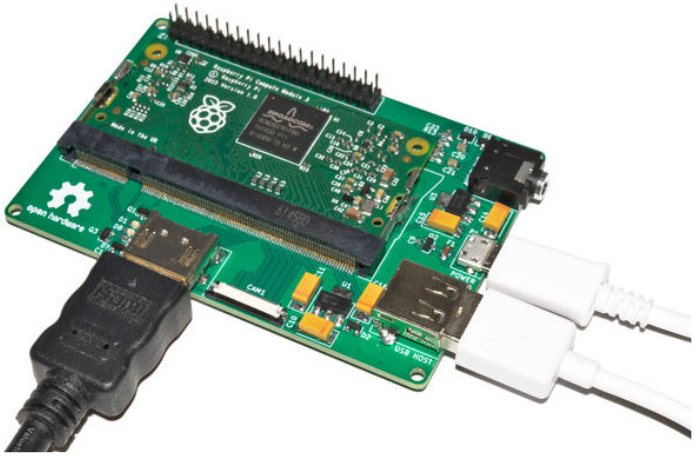
I should point out that I have successfully assembled and tested one of the boards myself, and I have verified that everything including the camera and the HDMI output appears to be working as expected. So, as long as you don't make any huge changes to the way I've routed everything you shouldn't have any issues.

In case you have to do some big layout changes though, keep in mind that most of the traces that go to the **HDMI** and **camera** connectors are routed as **100 Ohm differential pairs**. This means that you have to take this into account in case you have to move them around the board. Also, it means that even if you drop the GPIO header from your design, which means that now the internal layers won't contain any traces, you still need a **4 layer PCB** in order to achieve a differential impedance close to 100 Ohm. If you are not going to make use of the HDMI output and the camera though, you should be able to go with a 2 layer board by getting rid of them and cut down the cost of the boards a bit.

Just for reference, the boards were ordered from ALLPCB with a total thickness of **1.6mm** and I didn't ask for impedance control, as it would likely raise the cost quite a bit and I also wanted to see if it would matter. I also selected immersion gold finish to make hand soldering of the connectors easier as it guarantees that all pads are going to be nice and flat.


Compute Module Base Board
Rev: A
open hardware
HDMI
CAM1
POWER
USB HOST


Compute Module Base Board
Rev: A
open hardware


open hardware
CAM1
POWER
USB HOST