

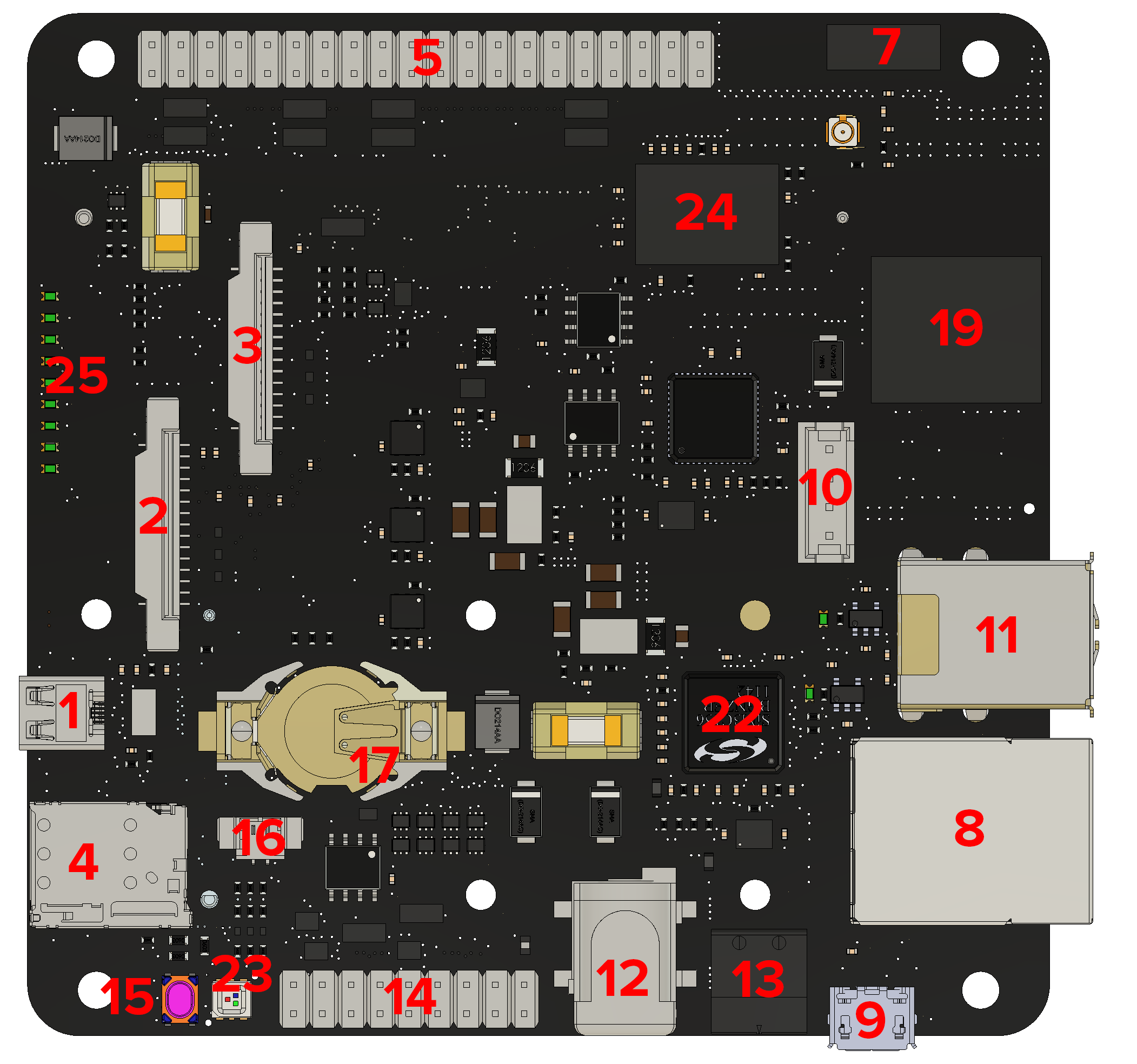
Project Amber

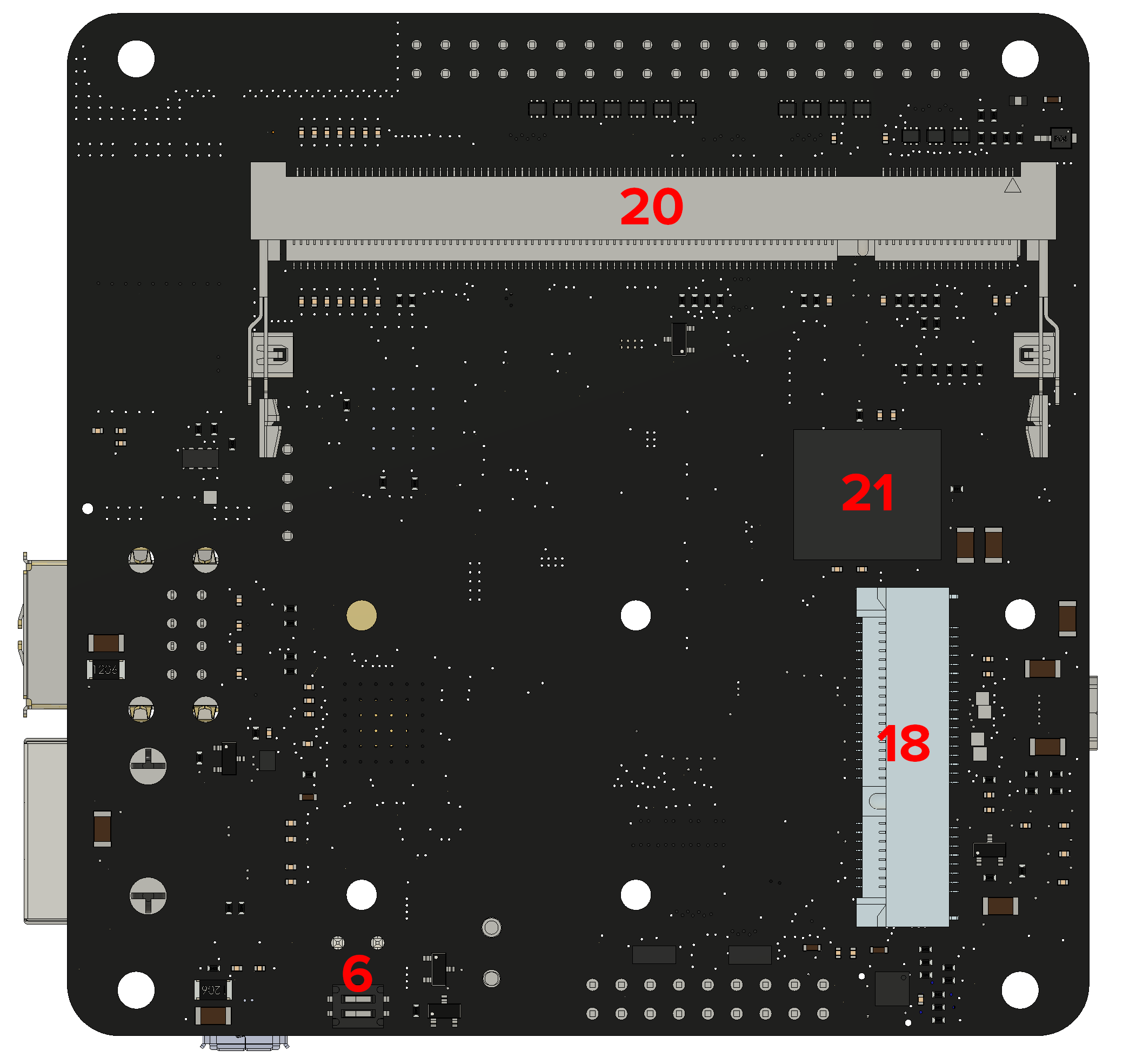
|  |  |
| --- | --- |
| **Document Type** | Prototype Internal Manual |
| **Project Codename:** | Amber\_Large\_1.0.0-RC1 |
| **Version** | 0.7.0 |
| **Author** | Carlo Maria Curinga |
| **State (Draft/Proposed/Approved)** | Draft |

*Note: the device covered by this manual is provided as an early evaluation unit. Both hardware and software are still in a “release candidate” stage and can be subject to change before a production release. Given the confidential nature of the early evaluation program, the reader is requested not to publicly disclose information about this project. Feedback to resin.io is highly encouraged!*

Specs

|  |  |
| --- | --- |
| **Main processor** | Raspberry Pi Compute Module 3 Lite - BCM2837 quad-core ARM Cortex A53 (ArmV8) at 1.2GHz |
| **RAM ( Main processor )** | 1GB |
| **Memory ( Main processor )** | 8/16/32/64 GB eMMC 5.1 |
| **Co-processor** | Artik020 - 32-bit ARM® Cortex M4 core at 40 MHz |
| **RAM ( Co-processor )** | 32 kB |
| **Memory ( Co-processor )** | 256 kB |
| **Radio connectivity ( Main processor )** | Dual band (2.4 & 5 GHz) WiFi  Bluetooth 4.1 |
| **Radio connectivity ( Co-processor )** | Bluetooth 4.2 Smart |
| **I/O ( Main Processor )** | Raspberry Pi 40-pin HAT |
| **I/O ( Co-processor )** | I2C, SPI, UART, GPIO, ADC |
| **Deep Sleep State management** | The Main processor along with its interfaces can be programmatically shut down and spawned back up via the co-processor, that can access the RTC chip when the main processor is OFF for time-based operations |
| **USB** | 2 x 2.0 Type-A |
| **PCI** | Mini PCI Express subset socket (USB,  UART and I2C with SIM card reader) |
| **Power** | 6 to 18V |
| **RTC** | Via I2C chip, with dedicated coin-cell  battery |
| **Ethernet** | 1 x 10/100 RJ45 |
| **MIPI** | 1 x Raspberry Pi camera connector  1 x Raspberry Pi display connector |
| **HDMI** | 1 x microHDMI |
| **User feedback** | 1 x RGB LED, 9 x Status LEDs |
| **Temperature range** | -25 to 70 celsius degrees |

Device Mapping



|  |  |
| --- | --- |
| **1** | Micro HDMI port |
| **2** | Raspberry Pi Display connector.  *NOTE: As for any Raspberry Pi Compute Module carrier board, a binary file with the pinout description is required in order to use the interface. At the time of writing, OS integration is still under development, so in the meantime a manual operation is needed: put* [*this file*](https://drive.google.com/open?id=0BwMoybaHFhYKWmxoSldKbnpsQkE) *in the boot partition (resin-boot in case of resinOS). This operation is required only once and will work for both Display (2) and Camera (3) interfaces.* |
| **3** | Raspberry Pi Camera connector.  *NOTE: As for any Raspberry Pi Compute Module carrier board, a binary file with the pinout description is required in order to use the interface. At the time of writing, OS integration is still under development, so in the meantime a manual operation is needed: put* [*this file*](https://drive.google.com/open?id=0BwMoybaHFhYKWmxoSldKbnpsQkE) *in the boot partition (resin-boot in case of resinOS). This operation is required only once and will work for both Display (2) and Camera (3) interfaces.* |
| **4** | nanoSIM socket |
| **5** | Raspberry Pi HAT connector |
| **6** | USB SLAVE BOOT ENABLE selector via 2-POS DIP SWITCH.   1,2 OFF means boot from eMMC  1,2 ON means expose eMMC as mass storage via CM3L debug port (9)  *NOTE: The next revision will drop this selector in favour of an automatic detection mechanism that won’t require any action from the user apart from plugging the micro-USB cable in the CM3L debug port (9)* |
| **7** | 2.4GHz & 5GHz embedded aerial (WiFi and Bluetooth). There is also a uFL connector for an external antenna. |
| **8** | RJ45 10/100 Ethernet port. This port is connected to the Raspberry Pi Compute Module 3 aka CM3L (20) via its USB controller. |
| **9** | CM3L debug port (DBG). When USB SLAVE BOOT ENABLE (6) is in flash mode, it allows to plug Amber to a host computer and flash its eMMC as a mass storage device. |
| **10** | Artik020 FTDI JTAG-USB controller with additional debug port. |
| **11** | Dual USB type-A female port. This port is connected to the Raspberry Pi Compute Module 3 aka CM3L (20) via its USB controller. |
| **12** | POWER IN port 1. Accepts Voltage input between 6V and 18V. Barrel Jack 5.5/2.1 |
| **13** | POWER IN port 2. Accepts Voltage input between 6V and 18V. 2-PIN Phoenix connector |
| **14** | MCU I/O pins. This pins expose the Artik020 (19) I/O. |
| **15** | RST tactile switch button. This button is connected to the START pins of the CM3L (23). Circuit is closed by default so the device will start on power input, unless the button is pressed which will prevent the CM3L (20) boot until release. Performing this operation allows to hard-reset the CM3L (20). |
| **16** | RST external connector. This connector allows to wire an external button for moving outside the board the RST (15) function. |
| **17** | I2C RTC chip with CR1225 coin-cell battery |
| **18** | Mini PCIe subset socket. This socket exposes as much protocols as possible that match the PCIe standard. USB, UART and I2C are supported, while PCI bus is not due to missing controller on the CM3L (23). USB based cards like SDcard readers, USB ports are supported. USB/Serial based cards like RS232, RS485, zigbee adapters are supported. USB+I2C cards like GPRS/3G/4G adapters, LoRa WAN adapters are supported.  *NOTE: we are currently performing compatibility tests on a wide range of cards, and will publish a list of known working devices.  NOTE: The device design allows to enable and disable (therefore also reset) programmatically this port via a GPIO pin exposed to the CM3L (20) via an I/O expander. At the time of writing, software support for it is being developed* |
| **19** | Artik020 MCU with embedded Bluetooth 4.2 and aerial. |
| **20** | SODIMM-200 socket for the CM3L |
| **21** | 8GB eMMC. Wired to the CM3L (20) SDIO as main internal storage, this will be also offered in 16GB, 32GB and 64GB in the next revision |
| **22** | USB Hub and Ethernet controller. |
| **23** | RGB LED. This LED is connected to the CM3L (20) via an I2C I/O Expander |
| **24** | Dual band Wifi+BT 4.1 embedded chip.  *NOTE: At the time of writing, development of the driver for this module is still ongoing. This module won’t work until software support is released by resin.io. In order to get WiFi/BT connectivity, a USB Dongle is temporarily required.* |
| **25** | Status LEDs |

Developer Instructions

# Flashing CM3L on Amber

Amber supports the Raspberry Pi Compute Module 3 **lite** (CM3L), and future versions.

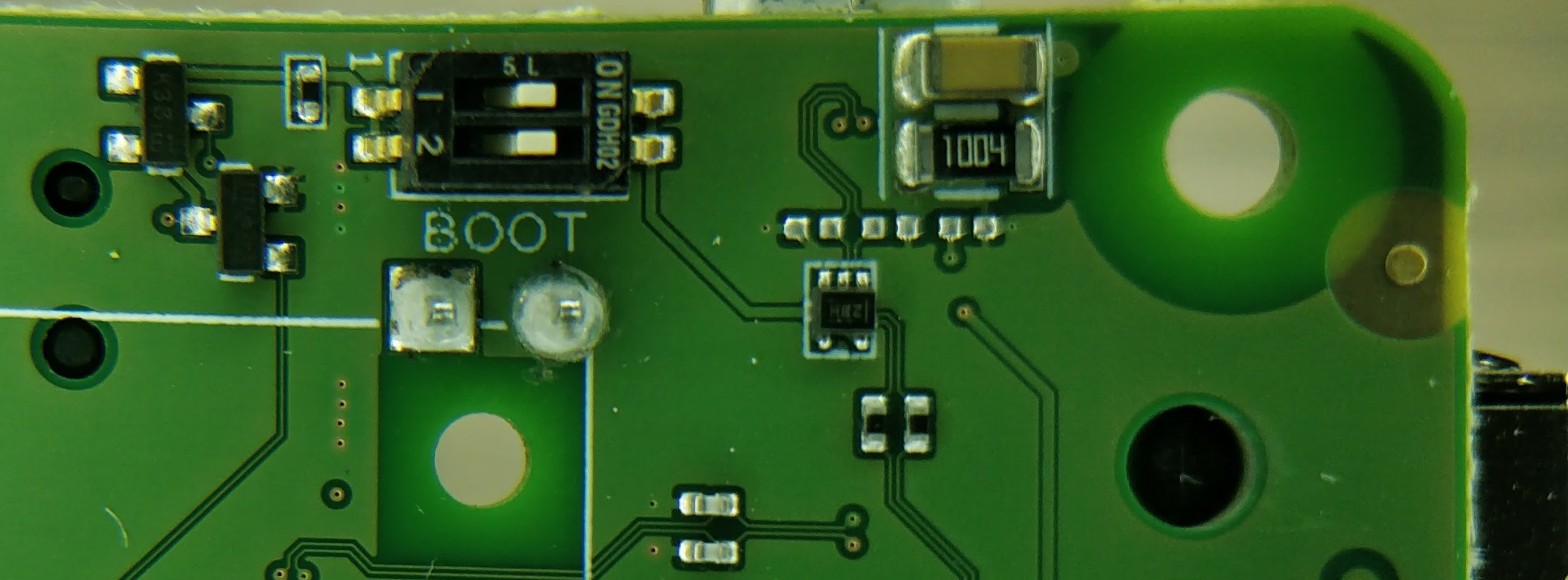
“Lite” means that the module itself has the eMMC socket unpopulated and the traces are exposed via SODIMM-200. This is very important since the standard CM3 has a fixed 4GB eMMC. Instead, with the CM3L, Amber can expose variable storage sizes via its embedded eMMC wired to the CM3L via the SODIMM-200 pins.

*NOTE: Any recent OS distribution (2017+) for the Raspberry Pi 3 Model B should be compatible with Amber. This includes Raspbian and resinOS 2.x targeting the “raspberrypi3” device-type.*

The Raspberry Pi Foundation provides a tool that allows the Compute Module to expose the eMMC as a mass storage device that can be flashed like any other media. We implemented this feature directly in Etcher ( developing our own native nodeJS implementation which is currently up to 3 times faster). This feature is available on etcher starting from version 1.2.1 for OSX and Windows (Windows still needs the Raspberry Pi Foundation usbboot drivers installed, but we are working on integrating the drivers in etcher in an upcoming release).

*If you want to use the tool provided by the Raspberry Pi Foundation instead, Please follow their instructions* [*here*](https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md)*.*

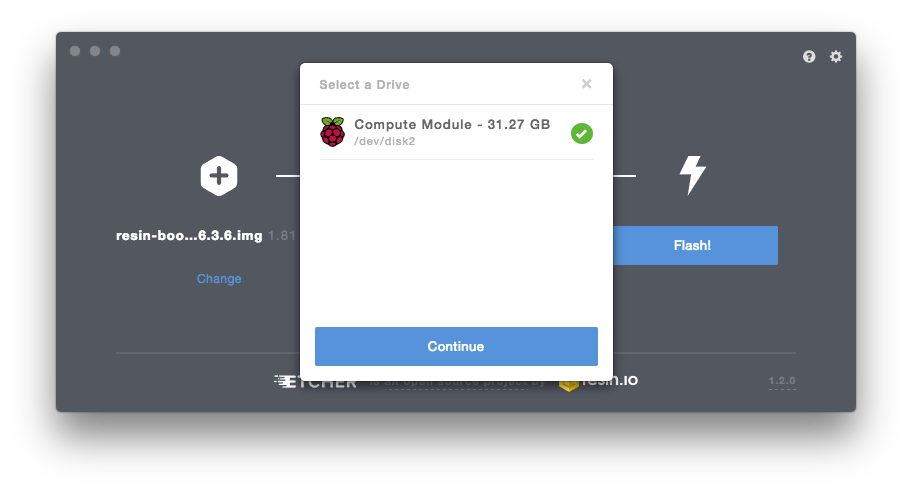
Once you have everything set up on your computer, set the USB SLAVE BOOT ENABLE selector (6) on the flash position: 1,2 ON



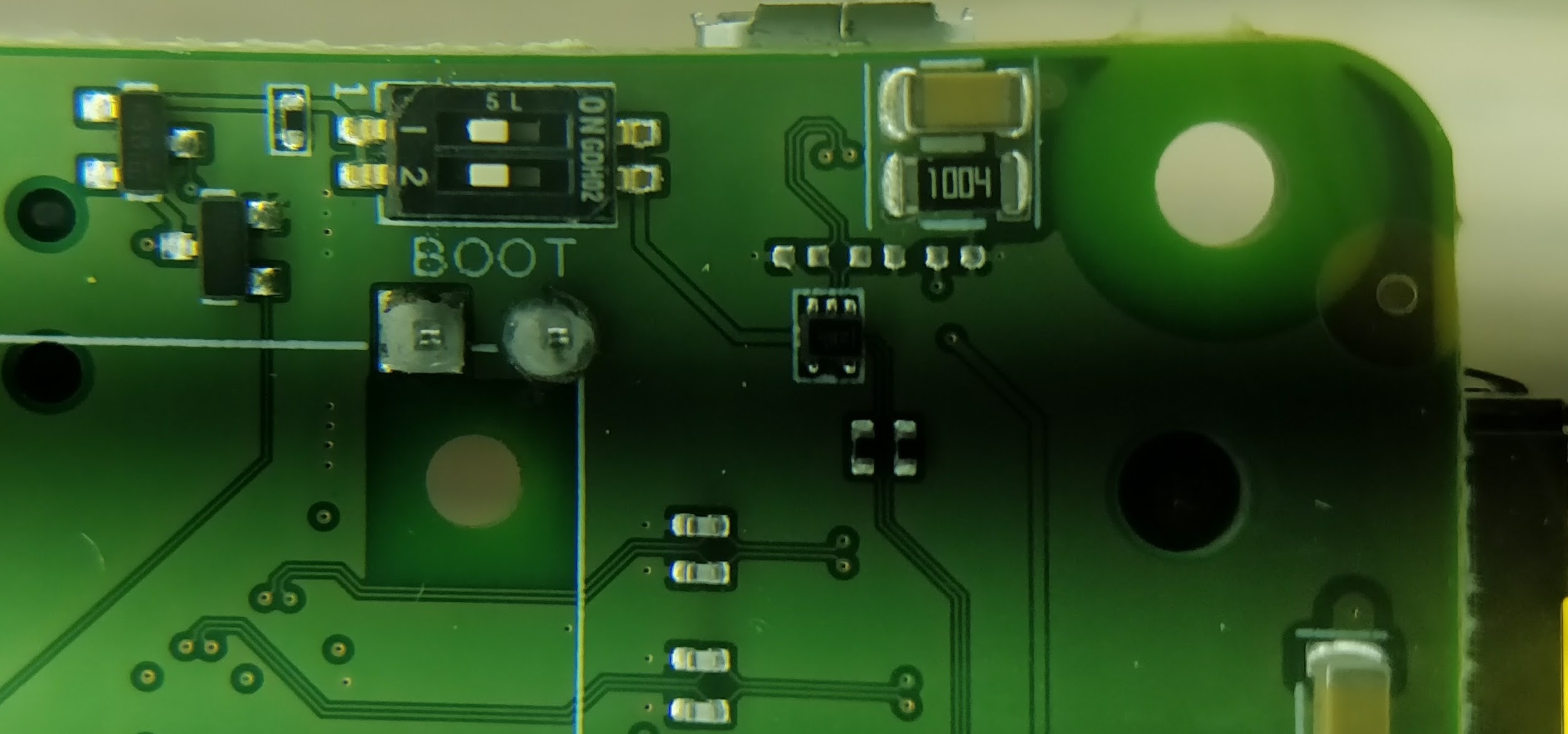
Then run Etcher on your computer, connect Amber via CM3L Debug port (9) and power it from the POWER IN port (12).

*NOTE: a CM3L module needs to be inserted in Amber!*

After a couple of seconds, the Amber eMMC should be detected on your computer by Etcher, which will initialize and list the board as a Compute Module based device (the name might change in the future). Proceed as usual selecting the image you want to write and press the “Flash!” button.



After flash is complete, power off Amber, put the USB SLAVE BOOT ENABLE selector (6) on the boot position: 1,2 OFF. Powering Amber on will result in the device booting from the freshly-written eMMC.



*NOTE: On managed resinOS, if you configured networking correctly on the resin.io dashboard while downloading the image to flash, this is the point when the device provisions and shows up in your application dashboard. If the device fails to connect to the configured WiFi SSID, the usual 4-blinks pattern will be showed on the ACT status LED*

# Flashing Artik020 on Amber

The Artik020 SWDIO interface is exposed to the CM3L via a FTDI FT2232H-56Q JTAG USB controller. For debugging purposes, a JTAG USB connector (10) is exposed on this revision (and will still be present, but not populated, on future revisions). This means that the Artik020 can be flashed from the CM3L via, for example, openOCD. A reference can be found [here](https://github.com/resin-io-modules/FT2232H-56Q-openocd)

# Using the RTC from CM3L

Amber sports a very common I2C RTC module that is well known, supported and documented within the Raspberry Pi ecosystem: the ***DS1307.***

There are plenty of guides on how to interact with the chip, including the following:

* <https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi/set-rtc-time>
* <https://thepihut.com/blogs/raspberry-pi-tutorials/17209332-adding-a-real-time-clock-to-your-raspberry-pi>

# Using the I/O Expander from CM3L

* Copy [this dtb](https://drive.google.com/open?id=1tPp2yRudpU9T9jzVARjPuYHAP3d5FsVh) in the ***overlays*** folder in the boot partition
* Add the following line to your config.txt

dtoverlay=amber-gpio-expander

* reboot

#### Controlling the RGB LED

* echo 504 > /sys/class/gpio/export # (Red)
* echo 505 > /sys/class/gpio/export # (Green)
* echo 506 > /sys/class/gpio/export # (Blue)
* echo "out" > /sys/class/gpio/gpio504/direction
* echo "out" > /sys/class/gpio/gpio505/direction
* echo "out" > /sys/class/gpio/gpio506/direction

Now that you have the 3 GPIO LED pins ready, you can define your target color by setting high or low each LED. For example, to turn the RGB off:

* echo 0 > /sys/class/gpio/gpio504/value
* echo 0 > /sys/class/gpio/gpio505/value
* echo 0 > /sys/class/gpio/gpio506/value

The above is an example using just standard sysfs. You can use, ofcourse, any kind of language and library, like [pi-pins](https://www.npmjs.com/package/pi-pins) for NodeJS:

const red = require("pi-pins").connect(504),

green = require("pi-pins").connect(505),

blue = require("pi-pins").connect(506);

red.mode('out');

green.mode('out');

blue.mode('out');

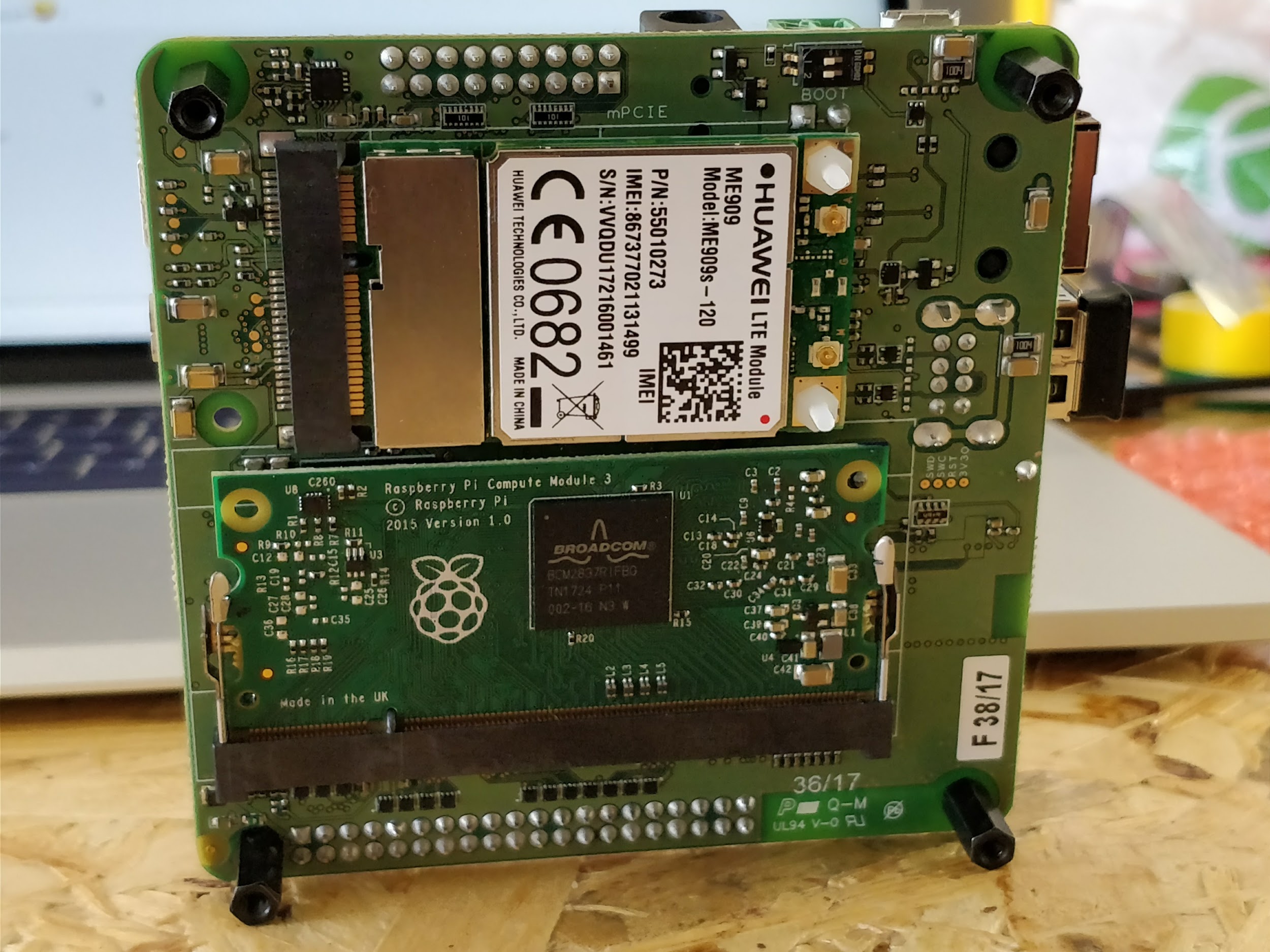
red.value(0);

green.value(0);

blue.value(0);

# Cellular via mPCIe card

As already mentioned on point 18 of the device mapping section, we are working on identifying and documenting cards known to work out of the box on the board. If you plan on adding LTE capability to the device, we suggest the [Huawei ME909s-120](https://techship.com/products/huawei-me909s-120-mpcie/): the card is known to work out of the box hence only APN configuration is required. On ResinOS (2.0.0+) you do so by adding a NetworkManager profile in the boot partition under the “system-connections” folder. You can find more info about this on our docs at <https://docs.resin.io/deployment/network/2.0.0/#cellular-modem-setup>



# HDMI output

There is the possibility that a 1.0.0-RC1 device would not output HDMI out of the box. If you encounter this issue, please be aware that it is related to GPIO41 (HDMI\_ENABLE) having a wrong default state at boot. The issue has been already resolved on the upcoming 1.0.0-RC2 version, but in the meantime you can simply set that pin state to HIGH and achieve full HDMI functionality. Here is an example: <https://gist.github.com/curcuz/00771654b31c834e548bd016a8a47a1c>

# Power saving tips\*

* If your system is headless you can turn off the HDMI port with:

*sudo /opt/vc/bin/tvservice -o*

(to turn it back on: *sudo /opt/vc/bin/tvservice -p*  )

This command will save you around 30-40mA.

* Turn off the RGB LED via the gpio-pca953x driver. The expander pin IO0 is Red, IO1 is Green and IO2 is Blue. Set all 3 low, the LED will turn off as result.

*\*Power optimization and low power consumption are key features for this project. The upcoming 1.0.0-RC2 revision will allow to selectively disable the RGB LED, the STATUS LEDS row, 3v3 and 5v peripherials and the CM3L itself via the co-processor. This will effectively lower the power consumption to the operating/idle absorption of the co-processor. API and documentation for 1.0.0-RC2 are being developed and will be released as soon as the evaluation kit upgrade will be sent to you.*

# Status LEDs on AmberIMG_20171003_164454.jpg