Rubus
(formerly known as Project Amber)

Document Type	Prototype Internal Manual
Project Codename:	Rubus_Large_1.0.0-RC2
Version	0.9.6
Author	Carlo Maria Curinga
State (Draft/Proposed/Approved)	Draft

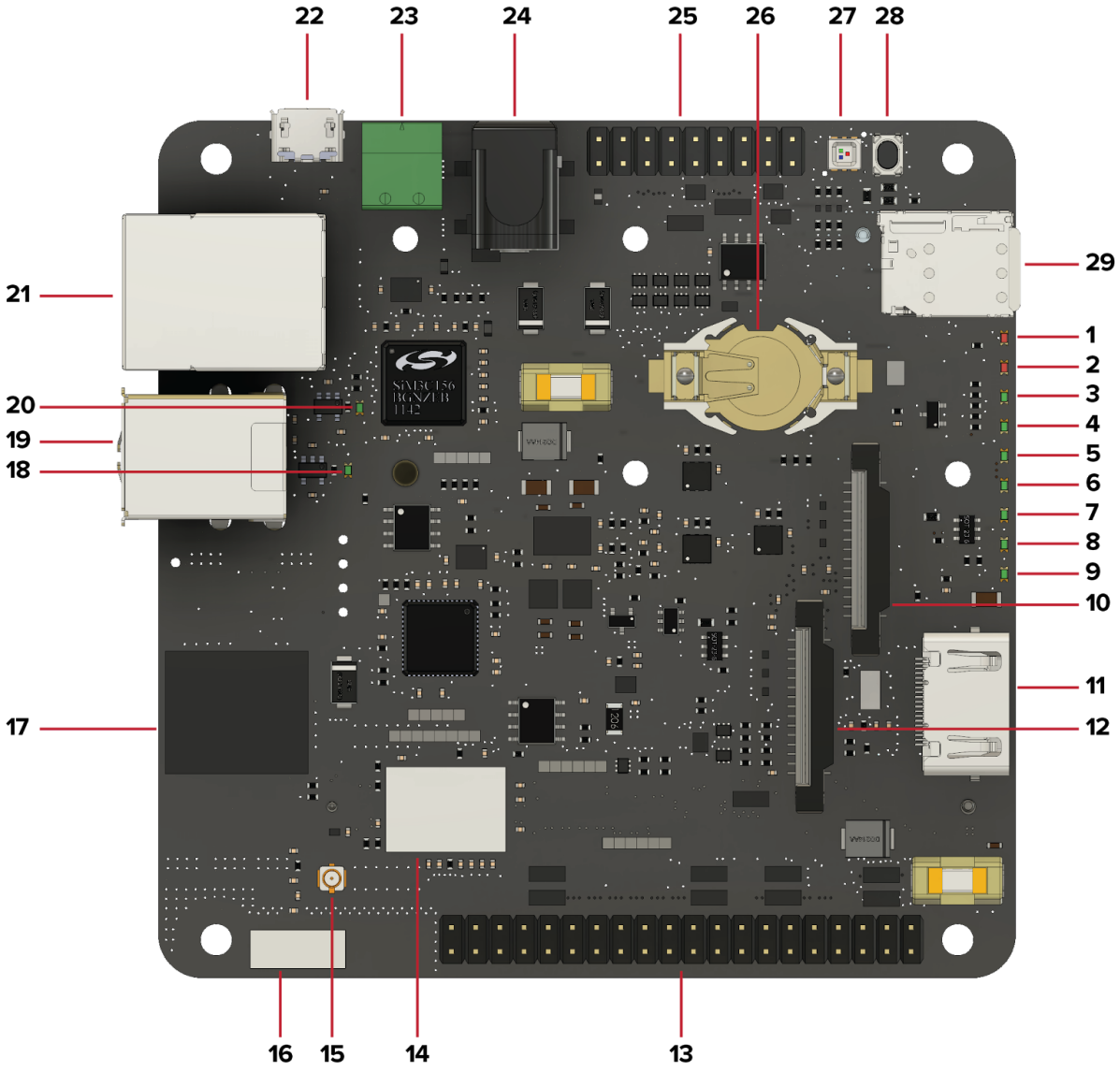
Note: the device covered by this manual is provided as an early evaluation unit. Both hardware and software are still in a "release candidate" stage and can be subject to change before a production release. Given the confidential nature of the early evaluation program, the reader is requested not to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

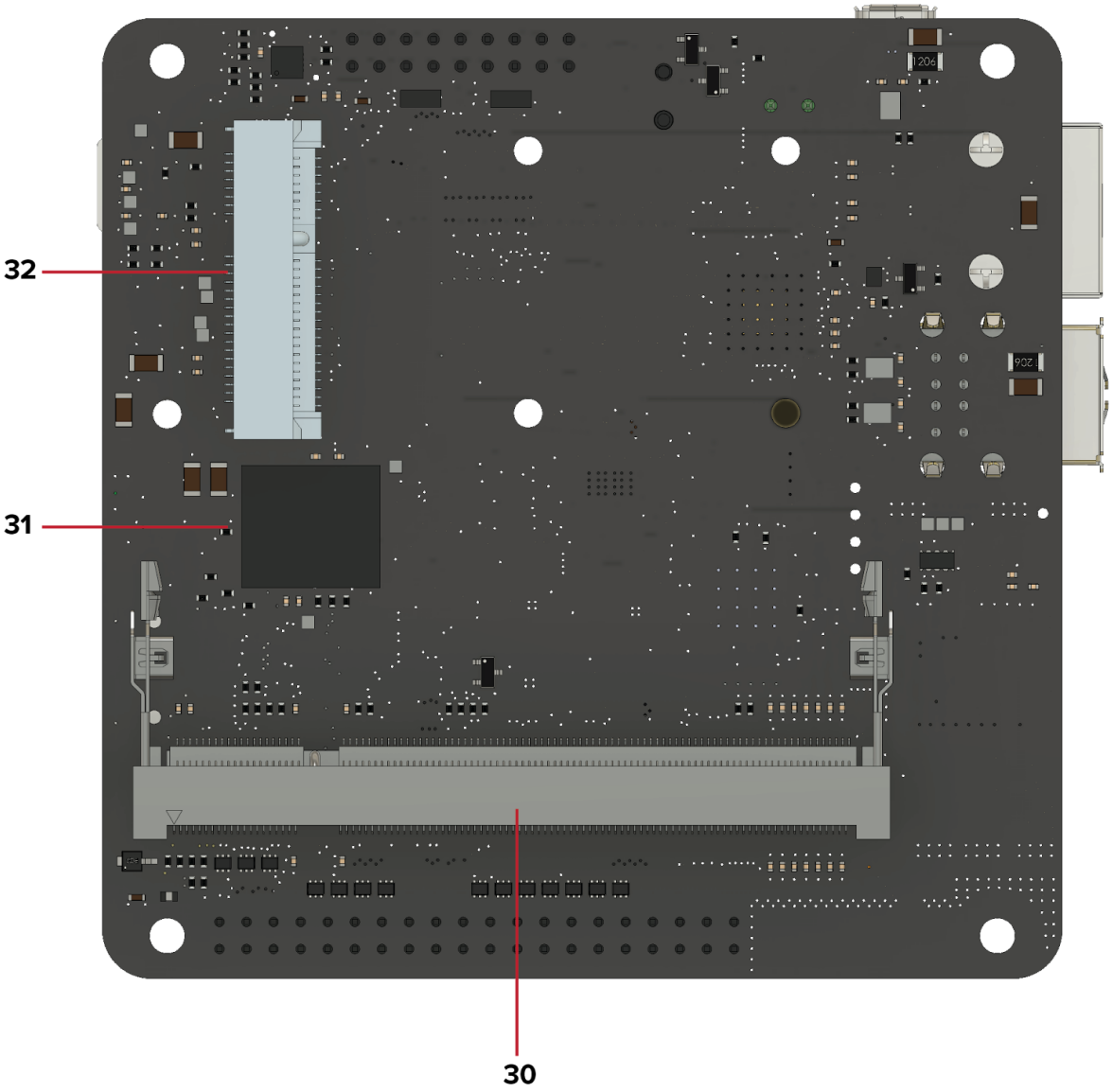
Specs

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)	802.11ac/a/b/g/n 2.4 & 5GHz WiFi + Bluetooth 4.2
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, which 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 nano-SIM card reader)
Power	6 to 30V (5V @2.5A via HAT)
RTC	Via I2C chip, with dedicated coin-cell Battery (positive polarity of battery facing upwards)
Ethernet	1 x 10/100 RJ45

MIPI	1 x Raspberry Pi camera connector 1 x Raspberry Pi display connector
HDMI	1 x HDMI Type A with CEC
User feedback	1 x RGB LED, 9 x Status LEDs
Temperature range	-25 to 70 celsius degrees

Device mapping





#	Name	Notes/Description
1	5V Status LED	Indicates 5V current flow.
2	3V3 Status LED	Indicates 3.3V current flow. This is the same as the red LED on the Raspberry Pi 3 Model B
3	LNK Status LED	Ethernet Link/Activity LED.
4	SPD Status LED	Ethernet Speed LED. off when in 10-Mbps mode, on when in 100-Mbps mode
5	FDX Status LED	Ethernet Full-Duplex indicator
6	ACT Status LED	CM3L Activity LED. This is the same as the green LED on the Raspberry Pi 3 Model B
7	PAN Status LED	If supported by the mPCIE (32) card connected, indicates PAN network activity
8	LAN Status LED	If supported by the mPCIE (32) card connected, indicates LAN network activity
9	WAN Status LED	If supported by the mPCIE (32) card connected, indicates WAN network activity
10	DSI connector	Standard full-size Raspberry Pi Display connector
11	HDMI	Full-size HDMI Type A with CEC support
12	CSI connector	Standard full-size Raspberry Pi Camera connector
13	HAT connector	40-pin Raspberry Pi HAT (Hardware Attached on Top) standard connector
14	WiFi/BT combo chip	802.11ac/a/b/g/n 2.4 & 5GHz WiFi + Bluetooth 4.2
15	WiFi/BT uFL antenna connector	If the RF switch is set on the external position, the antenna attached to this connector will become the main Radio antenna for the WiFi/BT combo chip (14)
16	WiFi/BT embedded antenna	Embedded high-performance SMD antenna covering both 2.4 and 5GHz frequencies. It is the default antenna selected for the WiFi/BT combo chip (14)
17	Co-processor	Artik020 MCU
18	USB1 FLT Status LED	The green LED stays on as long as there is enough current flowing on the top USB port. When this LED is off, it means a fault or under-voltage is happening on the top USB port

19	USB	2 x USB Type-A
20	USB2 FLT Status LED	The green LED stays on as long as there is enough current flowing on the bottom USB port. When this LED is off, it means a fault or under-voltage is happening on the bottom USB port
21	Ethernet	10/100 ethernet RJ45 connector
22	DBG - Programming port	micro-USB connector that allows to flash the eMMC from a host computer using etcher.io or usbboot. If the device is powered on while there is a cable connected to this port, it will enter a programming mode exposing its eMMC as mass-storage to a host computer (via etcher.io or usbboot)
23	2-POS Phoenix connector	6-30V input power. This is a Industry standard connector. *Use either this or the Barrel Jack connector (24) - never both at the same time!*
24	2.1 / 5.5 mm Barrel Jack connector	6-30V input power. *Use either this or the Phoenix connector (23) - never both at the same time!*
25	Co-Processor I/O connector	8 x GPIO / ADC 1 x SPI 1 x I2C 1 x Debug UART
26	CR122 RTC coin-cell battery socket	This allows the embedded RTC to keep track of time while the device is powered off.
27	RGB LED	
28	RESET push-button	When pressed (and released) DS1307 reboots the CM3L (30)
29	nano-SIM socket	This allows the use of a wide portfolio of cellular Modems via the mPCIE socket (32)
30	CM3L socket	SODIMM-200 socket for the Raspberry Pi Compute Module 3 Lite
31	eMMC	8GB class 5.1 industrial eMMC - main storage for the CM3L (30)
32	mPCIE	Mini PCI express socket

Developer Instructions

Flashing CM3L on Rubus

Rubus supports the Raspberry Pi Compute Module 3 **lite** (CM3L).

“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, Rubus can expose variable storage sizes via its embedded eMMC wired to the CM3L via the SODIMM-200 pins.

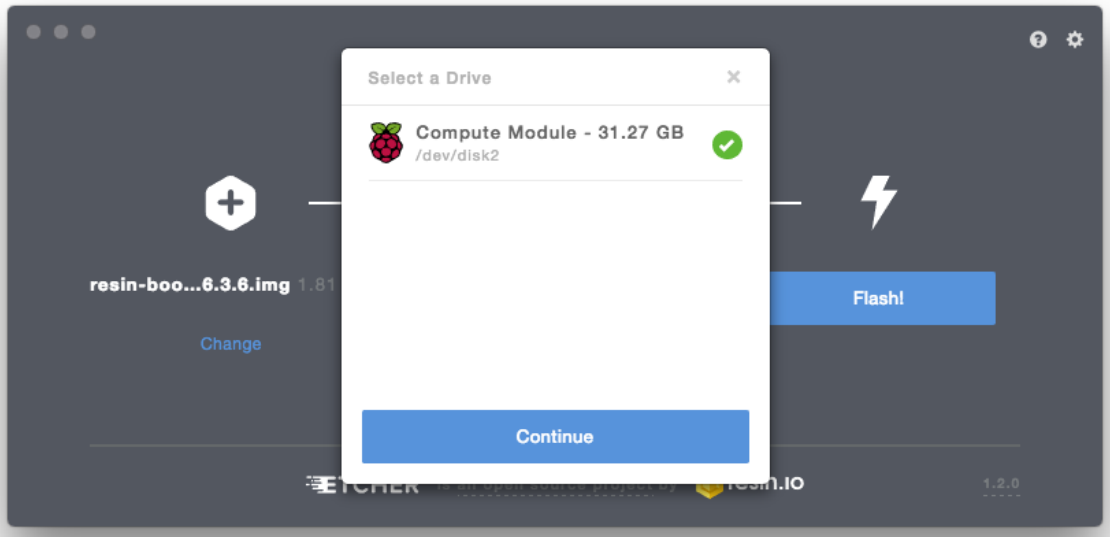
NOTE: Any recent OS distribution (2017+) for the Raspberry Pi 3 Model B should be compatible with Rubus. This includes Raspbian and resinOS 2.x targeting the “raspberrypi3” device-type.

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 by 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 into Etcher in an upcoming release).

If you want to use the tool provided by the Raspberry Pi Foundation instead, please follow their instructions here.

Once you have everything set up on your computer, run Etcher on your computer, connect Rubus via CM3L Debug port (9) and power it from the POWER IN port (12).
NOTE: a CM3L module needs to be inserted in Rubus!

After a couple of seconds, the Rubus 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, select the image you want to write, and press the “Flash!” button.



After flash is complete, power off Rubus and unplug the DEBUG micro-USB cable. Powering Rubus on will now 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 shown on the ACT status LED

Installing specific Hardware support

At the time of writing this manual, OS integration is still under development; a manual operation is required to enable full hardware support:

Generic instructions for Raspberry Pi 3 distributions:

- Copy this file in the boot partition
  Note: since version 0.9.5 of this manual, this file also allows automatic configuration loading via EEPROM for the HAT specification
  You can access the boot partition of your Rubus by simply attaching it to your computer with the DBG micro USB cable, power it up and open Etcher. It will detect and mount the partition for you!
- Copy this dtb in the ***overlays*** folder in the boot partition
  - Add the following line to your config.txt

```
dtoverlay=rubus-rc2,poll_once=off,sdio_overclock=35,gpiopin=40,active_low
```

- Put this file in `/lib/firmware/mrvl/` and reboot

resinOS / resin.io instructions:

- Download a 2.9.6 or newer version of resinOS from your application dashboard
- Copy this file in the boot partition (`resin-boot` when mounted on your computer, from within the booted device is `/mnt/boot/`)
  Note: since version 0.9.5 of this manual, this file also allows automatic configuration loading via EEPROM for the HAT specification.
  You can access the boot partition of your Rubus by simply attaching it to your computer with the DBG micro USB cable, power it up and open Etcher. It will detect and mount the partition for you!
- Download and put this file in the `overlays` folder of boot partition of the image
- Add a configuration variable in the dashboard called `RESIN_HOST_CONFIG_dtoverlay` and set its value to `rubus-rc2,poll_once=off,sdio_overclock=35,gpiopin=40,active_low`
- Flash and boot your device.

Flashing Artik020 on Rubus

The Artik020 SWDIO interface is exposed to the CM3L via a FTDI FT2232H-56Q JTAG USB controller. This means that the Artik020 can be flashed from the CM3L via, for example, openOCD. A reference can be found here.

Using the RTC from CM3L

Rubus 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

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, of course, any kind of language and library, like 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);
```

Toggling the Status LED bank

This device sports 9 status LEDs varying from power, eMMC, ethernet, WiFi, WAN, etc. There is a switch which allows to toggle them all on and off via software.

- `echo 511 > /sys/class/gpio/export`
- `echo "out" > /sys/class/gpio/gpio511/direction`
- `echo 0 > /sys/class/gpio/gpio511/value # turn off`
- `echo 1 > /sys/class/gpio/gpio511/value # turn on`

The above is an example using just standard sysfs. You can use any kind of language and library, like pi-pins for NodeJS:

```
const ledsToggle = require("pi-pins").connect(511);
ledsToggle.mode('out');
ledsToggle.value(0); // LEDs bank off
ledsToggle.value(1); // LEDs bank on
```

Toggling the WiFi/BT antenna mode[1]

This device sports both an integrated high-performance SMD antenna and a uFL connector for WIfi/BT radio. There is an RF switch which allows the toggle between the 2 via software.

- `echo 510 > /sys/class/gpio/export`
- `echo "out" > /sys/class/gpio/gpio510/direction`
- `echo 1 > /sys/class/gpio/gpio510/value # Internal antenna`
- `echo 0 > /sys/class/gpio/gpio510/value # External antenna`

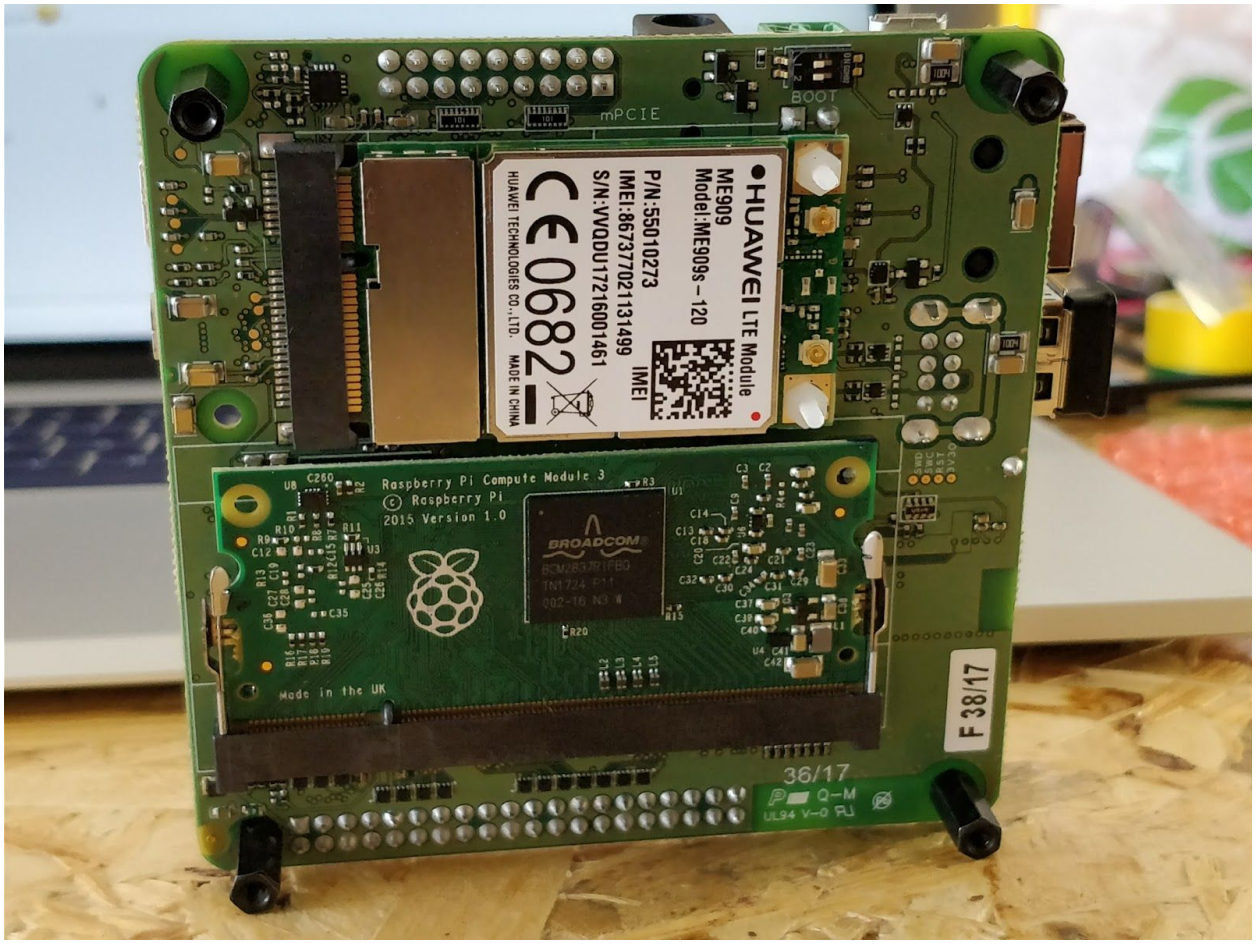
The above is an example using just standard sysfs. You can use any kind of language and library, like pi-pins for NodeJS:

```
const ledsToggle = require("pi-pins").connect(511);
ledsToggle.mode('out');
ledsToggle.value(1); // Internal antenna
ledsToggle.value(0); // External antenna
```

[1] *note: this will be replaced by a switch on the board in the next design revision*

Cellular via mPCIe card

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: 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.



Disabling RF activity on mPCIe radio cards

- `echo 508 > /sys/class/gpio/export`
- `echo "out" > /sys/class/gpio/gpio508/direction`
- `echo 1 > /sys/class/gpio/gpio508/value # mPCIe on`
- `echo 0 > /sys/class/gpio/gpio508/value # mPCIe off`

The above is an example using just standard sysfs. You can use any kind of language and library, like pi-pins for NodeJS:

```
const mpcieToggle = require("pi-pins").connect(508);
mpcieToggle.mode('out');
mpcieToggle.value(1); // mPCIe on
mpcieToggle.value(0); // mPCIe off
```

Power saving

- 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`)
- Turn off the STATUS LEDs bank
- Programmatically disable RF activity on mPCIe when not required (assumes a cellular modem is being used)

Known revision issues

- The current revision has an issue that prevents the board from being powered by the HAT (5V @2.5A). This issue has already been fixed in the upcoming 1.0.0-RC3 but in the meantime you can work it around by connecting one of the 5V pins of the HAT connector to the **+** pin of the phoenix power connector as illustrated below: