docs.computeblade.com

COMPUTE

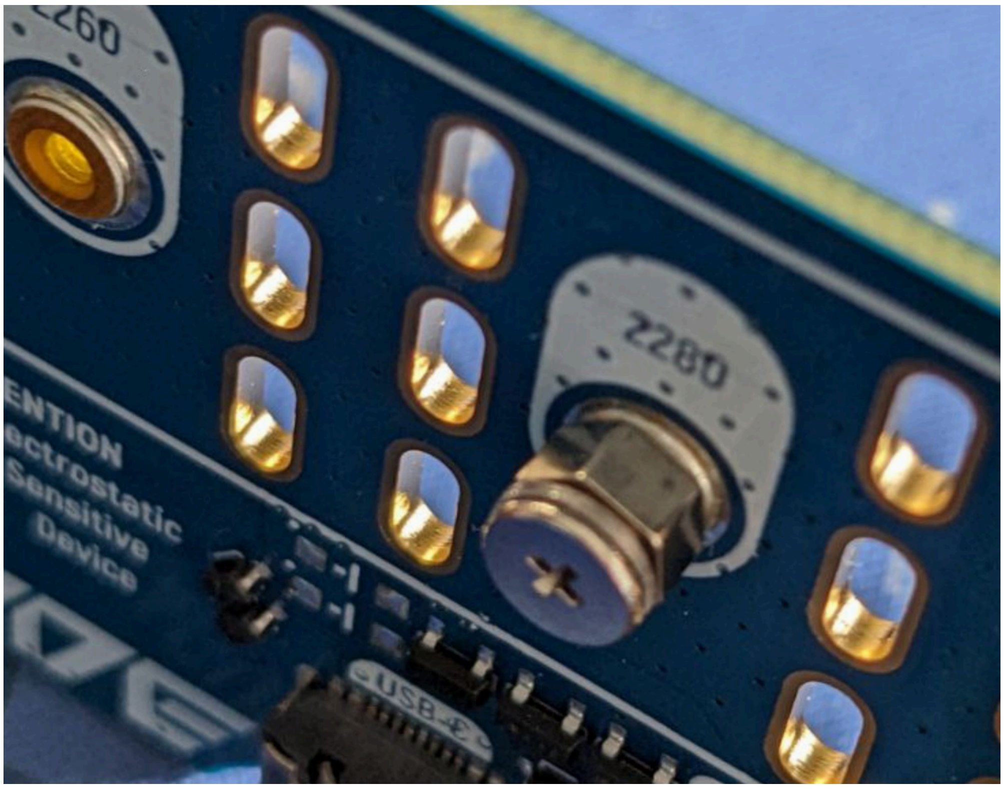
BLADE

Compute Blade RC2

Main Features

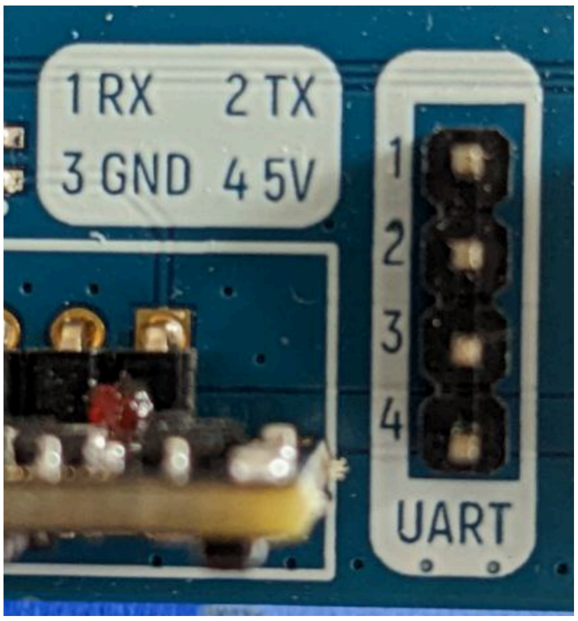
NVMe Boot

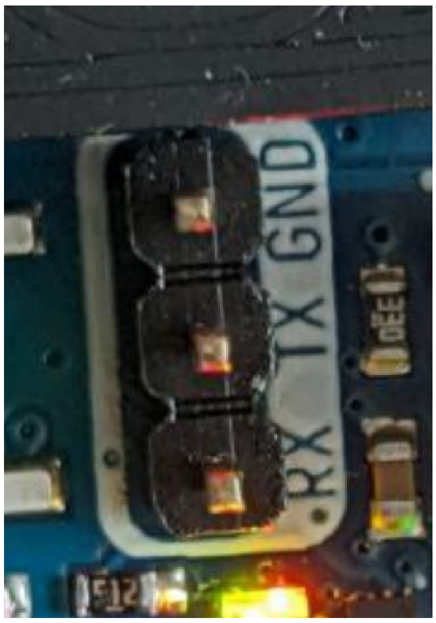
Works out of the box on the latest (after July 2021) Raspberry Pi CM4. If you're not using the latest Raspberry Pi CM4, you'll need to update the bootloader specifying the correct boot order.
Use standoff for the NVMe drive, 2mm high.



UART

This is the same UART0 port, but with the big one, you can use 5V in/out (in "in" mode ONLY if the Blade has no other power source)*
Connect "correctly":
RX -> TX
TX -> RX
GND -> ~~flower pot~~ GND



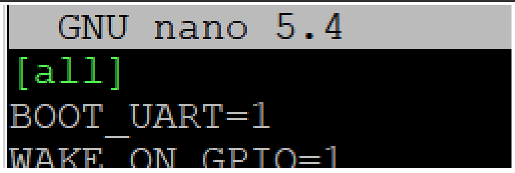


in config:

```
enable_uart=1
```

You can change the bootloader setting to enable UART from the boot.
Add this line to **boot.conf** and flash the bootloader:

```
BOOT_UART=1
```



```
  GNU nano 5.4
[all]
BOOT_UART=1
WAKE_ON_GPIO=1
```

Follow this [link]() to the Raspberry Pi Documentation BOOT_UART value

*it'll work and probably won't burn your house down, but it's not exactly FCC or CE compliant.

USB-A

To turn on the USB-A port to access the operating system on the Blade, add the following line to the config.txt:

```
dtoverlay=dwc2,dr_mode=host
```

Stealth mode

Stealth mode will turn off all but the Digital LEDs at once. It is implemented via a GPIO and a transistor, which means it won't harm the Blade functionality. It is possible to change the brightness using software PWM however in some cases, it may not work perfectly.

Compute Blade RC2 cannot turn off the LEDs on the Ethernet port this way, but they can be turned off in other ways. The [Digital LEDs]() will also not be turned off due to technical limitations, but since they are controlled from the GPIO this should not be a problem.

Initialization in the terminal:

```
echo "21" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio21/direction
```

To turn off the LEDs:

```
echo "1" > /sys/class/gpio/gpio21/value
```

To turn on the LEDs:

```
echo "0" > /sys/class/gpio/gpio21/value
```

To do this immediately after booting the OS, you can add these commands to /etc/rc.local:

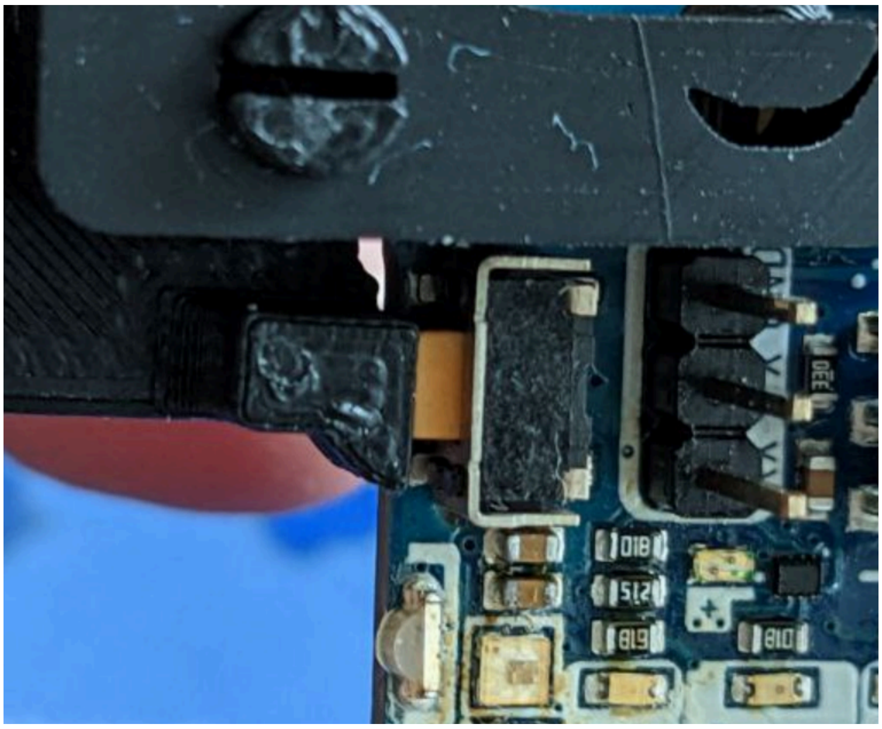
```
sudo nano /etc/rc.local
```

```
#!/bin/sh -e
#
# rc.local
#

echo "21" > /sys/class/gpio/export
sleep 0.1
echo "out" > /sys/class/gpio/gpio21/direction
echo "1" > /sys/class/gpio/gpio21/value
sleep 0.1

exit 0
```

Disable digital LED:

```
from time import sleep
import board
import neopixel
import psutil
import RPi.GPIO as GPIO

g,r,b = 0,0,0
LEDbrightness = 0.2

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

pixels = neopixel.NeoPixel(
    board.D18, 2, brightness=LEDbrightness, auto_write=False,
pixel_order=neop>
)

pixels[0] = (0, 0, 0)
pixels[1] = (0, 0, 0)
pixels.show()
```

Edge button



A button is connected to GPIO20 in the Rasberry PI compute module and can be customized as you like.

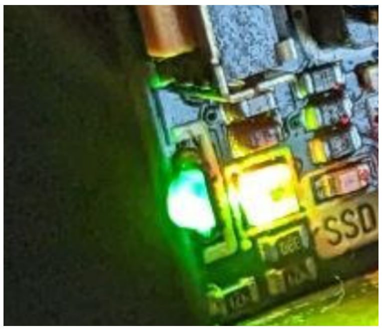
Python example:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_UP) #Button to GPIO20

try:
    while True:
         button_state = GPIO.input(20)
         if button_state == False:
             print('Button Pressed...')
             time.sleep(0.2)
```

```
except:
    GPIO.cleanup()
```

Digital LEDs

The blade has two digital LEDs mounted in series. A square yellow LED is located on the top of the board, and a green LED is located on edge. Both LEDS are connected to GPIO18 on the compute module.



Preparation:
You'll need to install the Adafruit_Blinka library that provides CircuitPython support in Python:
https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi

And then NeoPixel Library:
https://learn.adafruit.com/neopixels-on-raspberry-pi/python-usage

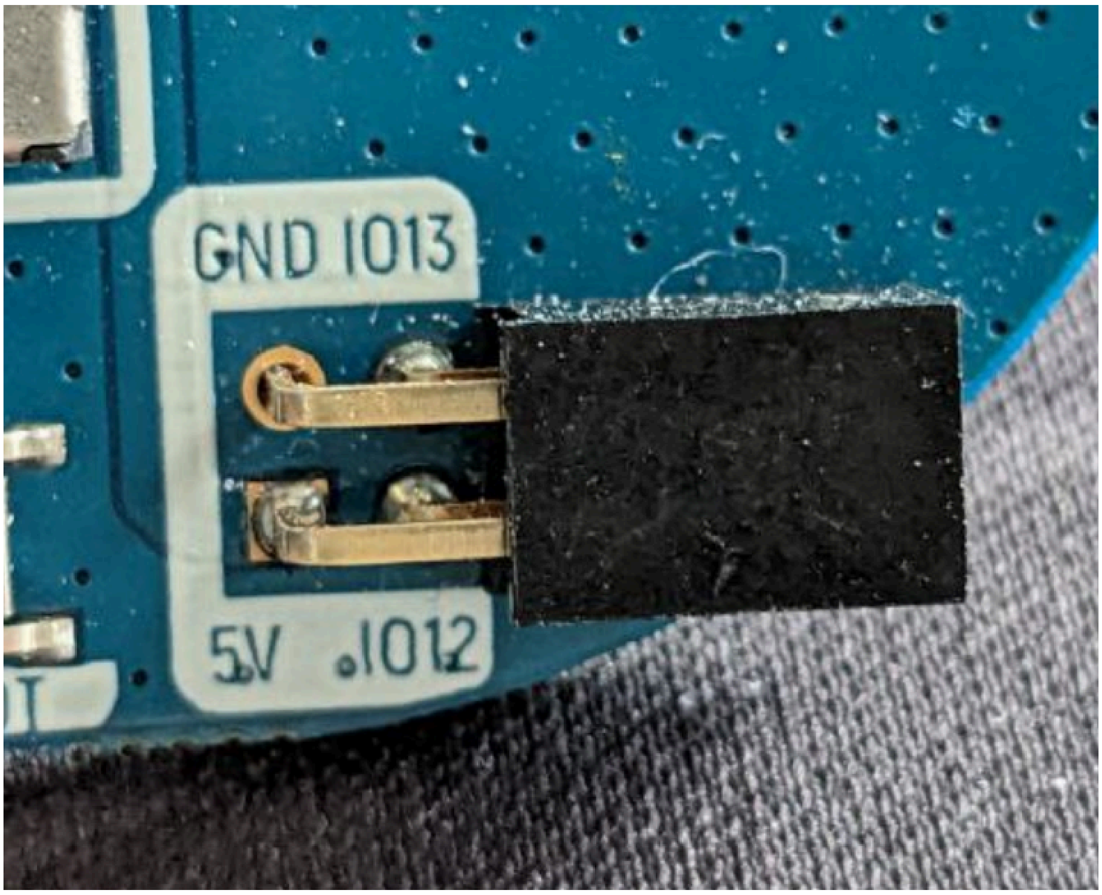
Newer CM4 Revisions aren't included in rpi-ws281x yet (e.g. Compute Module 4 Rev 1.1). If you get "RuntimeError: ws2811_init failed with code -3 (Hardware revision is not supported)", you need to follow these steps
https://github.com/rpi-ws281x/rpi-ws281x-python/issues/56#issuecomment-753320723

Here is an example Python script that will show you the CPU temperature on the square LED and the CPU load on edge LED: And use the button for blade identification:
Demo

```
from time import sleep
import board
import neopixel
import psutil
import RPi.GPIO as GPIO

g,r,b = 0,0,0
mintemp = 40 #Temperature Boundaries. Closer to this is green
maxtemp = 70 #Temperature Boundaries. Closer to this is red
```

```python
LEDbrightness = 0.2 #Yes, you can change the brightness here
Button = 20
flag = 0

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(Button,GPIO.IN,pull_up_down=GPIO.PUD_UP)
pixels = neopixel.NeoPixel(
    board.D18, 2, brightness=LEDbrightness, auto_write=False,
pixel_order=neopixel.GRB
)

def show(percent, pixel):
    a = int(round((510 / 100) * percent))
    if a > 255:
        r = 255
        g = 510 - a
    else:
        r = a
        g = 255
    pixels[pixel] = (g, r, b)
    pixels.show()
#    return print('showed', a, 'on', pixel)

while True:
    temp =
int(round(psutil.sensors_temperatures()["cpu_thermal"][0].current))
#    print('temp = ', temp)
    button_state = GPIO.input(Button)
    if mintemp <= temp <= maxtemp:
        temp = temp - mintemp
        mtemp = maxtemp - mintemp
        c = int(round((temp / mtemp) * 100))
        show(c, 0)
    elif temp < mintemp:
        pixels[0] = (255, 0, 0)
    else:
        pixels[0] = (0, 255, 0)

    sleep(0.1)

    load = psutil.cpu_percent()
 #   print('load = ', load)
    show(load, 1)
    sleep(0.1)
    if button_state == 0:
        sleep(0.5)
        if flag == 0:
            flag = 1
        else:
            flag = 0
    if flag == 1:
        print('Blade ID is active')
        pixels[1] = (0, 0, 255)

    else:
        print('Blade ID isn\'t active')
```

Square connector at the rear of the Blade



The square connector on the Compute Blade's main purpose is to connect a fan unit. The output ifs 5V and should work with any fan that supports 5V. The speed of the fan can be controlled with PWM.

But it can be used to control any 5v PWM fan. Or as an additional UART port (UART5 on CM4)
Warning: Take care when wiring into this connector!

GND	GPIO13
5V	GPIO12

UART connection (RP2040 Fan Unit)

On the Blade
Add to Config.txt

```
dtoverlay=uart5
```

Console:

```
sudo apt-get install cu
sudo chmod 666 /dev/ttyAMA1
sudo cu -l /dev/ttyAMA1 -s 115200
```

Fan Unit code:
https://github.com/merocle/FanUnit

Notes:

```
~.
```

to close **cu** session

```
jobs -l
```

to check if session in the background (in case **"cu: /dev/ttyAMA1: Line in use**" error)

```
fg
```

to return to current opened session

PWM fan (non-smart Fan Unit)

When using a dumb Fan Unit
the fan is controlled from the left blade in the pair

Examples:

```
wget
https://raw.githubusercontent.com/DriftKingTW/Raspberry-Pi-PWM-Fan-Contr
ol/master/read_fan_speed.py
python read_fan_speed.py
```

You need to change in the script:
TACH = 13

```
wget
https://raw.githubusercontent.com/DriftKingTW/Raspberry-Pi-PWM-Fan-Contr
ol/master/fan_control.py
python fan_control.py
```

You need to change in the script:
FAN_PIN = 12

to run the script in the background quickly, run `nohup python fan_control.py`

Links:
https://blog.driftking.tw/en/2019/11/Using-Raspberry-Pi-to-Control-a-PWM-Fan-and-Monitor-its-Speed/

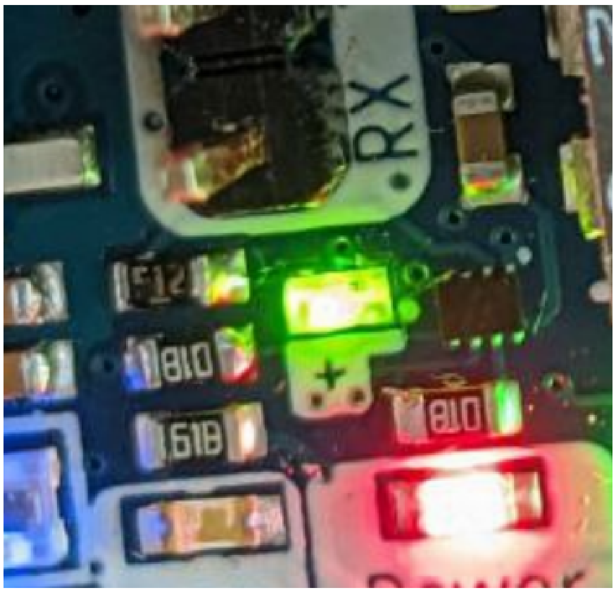
Ethernet with PoE

1Gbit with Power over Ethernet
802.3at Type 2, PoE+ (up to 30W)
Compatible with 802.3af (802.3at Type 1), PoE (up to 15W)

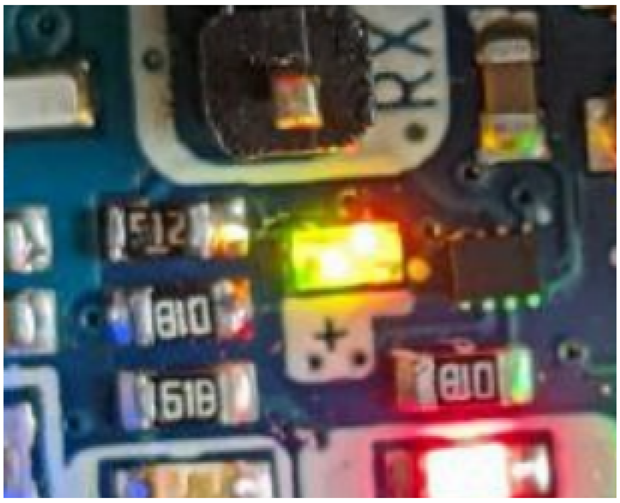
PoE determining

You can use it to monitor power supply failures or limit yourself in overclocking
LED with "+" and GPIO23 indicate the operating mode:

Green - 5-volt present, PoE or USB-C or 5v direct
GPIO23 Low

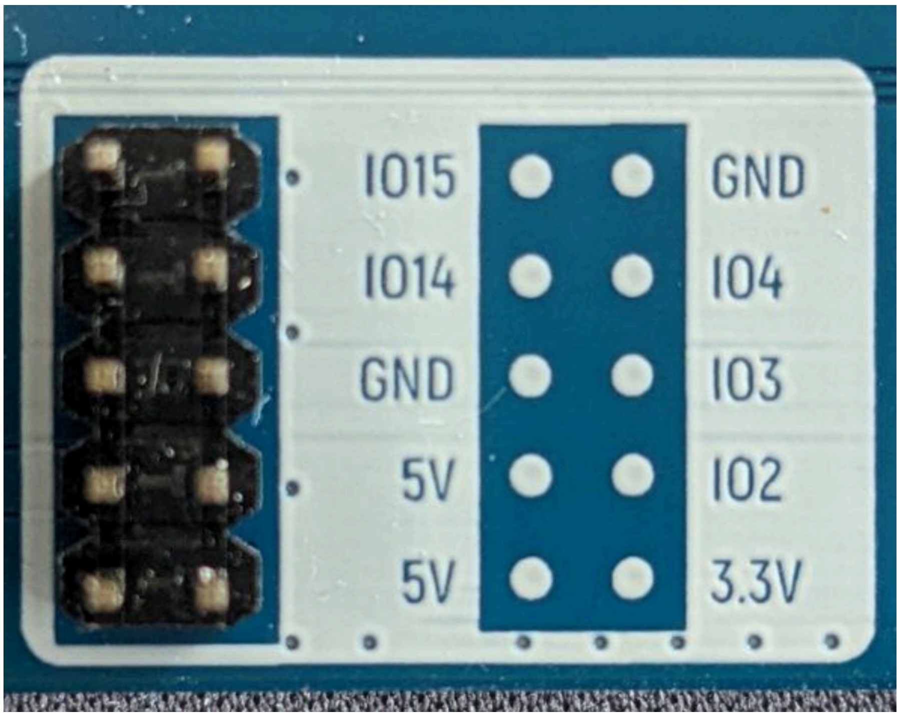


Orange (Green and Red together) - 802.3at Type 2 operating mode, PoE+
GPIO23 High
Blade can take up to 30W power (22W for RC2)

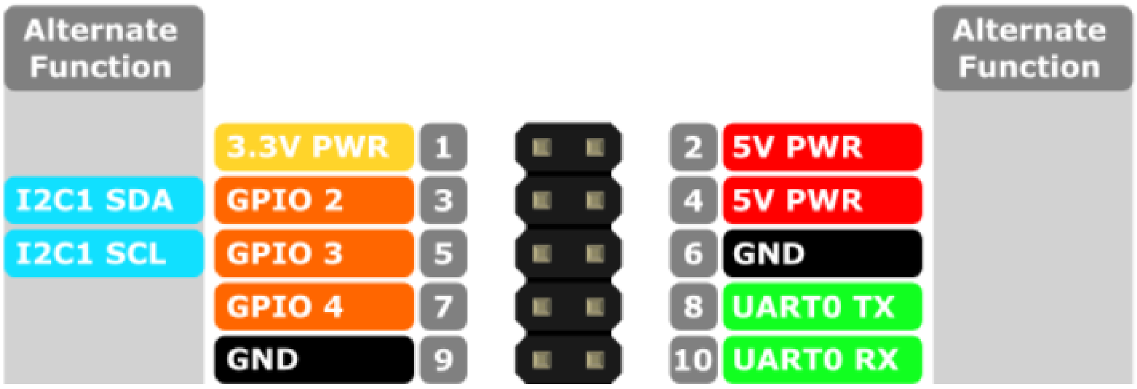


Compute blade headers

You can use Real-Time Clocks (RTCs) for the Raspberry Pi and other compatible modules.
The section will be updated



This is the top part of the standard GPIO connector of the Raspberry Pi (upside down)



picture source

RTC

in config:

```
dtoverlay=i2c-rtc,ds3231
dtparam=i2c_arm=on
```

https://spellfoundry.com/sleepy-pi/setting-up-the-real-time-clock-on-raspbian-jessie/

in terminal:

```
sudo i2cdetect -l|sort
sudo i2cdetect -y 1
sudo apt-get -y remove fake-hwclock
sudo update-rc.d -f fake-hwclock remove
sudo systemctl disable fake-hwclock
```

edit the file:

```
sudo nano /lib/udev/hwclock-set
```

```
#!/bin/sh
# Reset the System Clock to UTC if the hardware clock from which it
# was copied by the kernel was in localtime.

dev=$1

#if [ -e /run/systemd/system ] ; then
#    exit 0
#fi

if [ -e /run/udev/hwclock-set ]; then
    exit 0
fi

if [ -f /etc/default/rcS ] ; then
    . /etc/default/rcS
fi

# These defaults are user-overridable in /etc/default/hwclock
BADYEAR=no
HWCLOCKACCESS=yes
HWCLOCKPARS=
HCTOSYS_DEVICE=rtc0
if [ -f /etc/default/hwclock ] ; then
    . /etc/default/hwclock
fi

if [ yes = "$BADYEAR" ] ; then
#    /sbin/hwclock --rtc=$dev --systz --badyear
    /sbin/hwclock --rtc=$dev --hctosys --badyear
else
#    /sbin/hwclock --rtc=$dev --systz
    /sbin/hwclock --rtc=$dev --hctosys
fi

# Note 'touch' may not be available in initramfs
> /run/udev/hwclock-set
```

Check:

```
timedatectl status
```

Read:

```
sudo hwclock -r
```

Write:

```
sudo hwclock -w
sudo hwclock --verbose
```

ZYMKEY 4i

https://docs.zymbit.com/getting-started/zymkey4/quickstart/

Compute Blade TPM and Dev

TPM

In config:

```
dtparam=spi=on
dtoverlay=tpm-slb9670
```

To check
In terminal:

```
mkdir infineon-tpm
git clone https://github.com/infineon/eltt2
cd eltt2
make
sudo ./eltt2 -g
cd ..
```

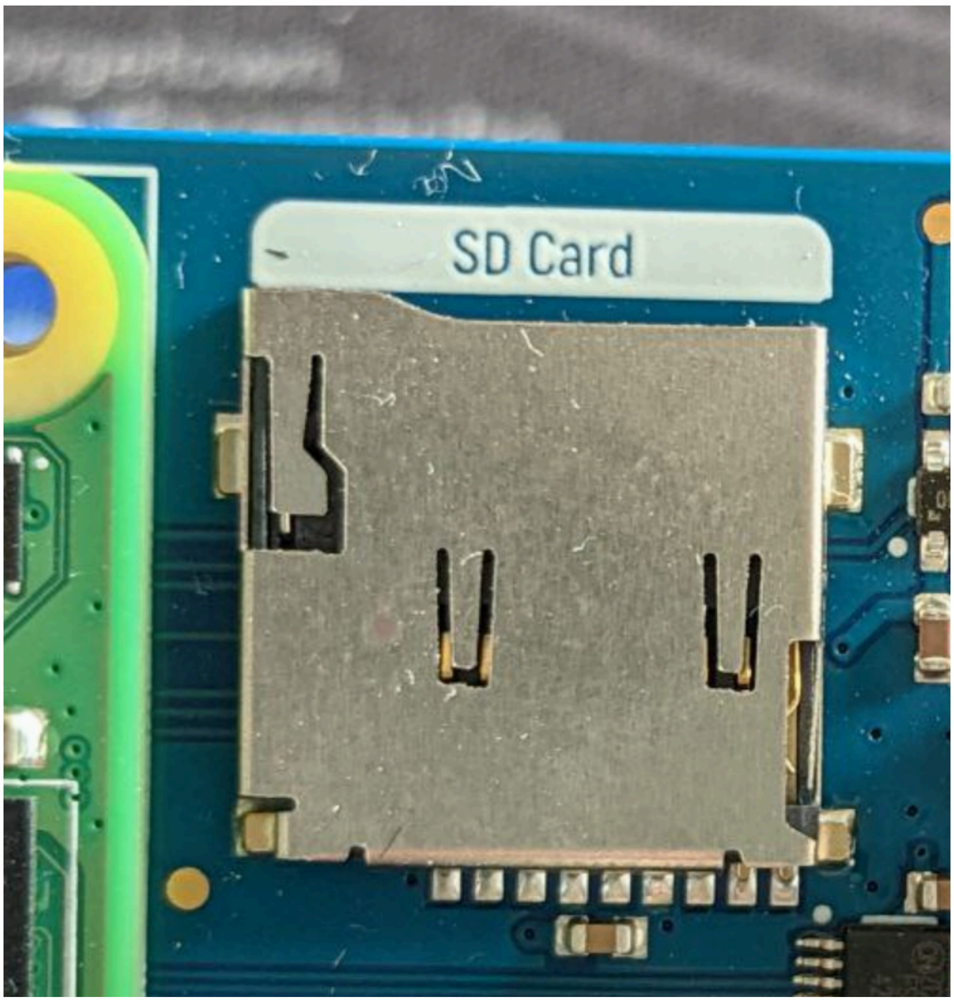
For more information, please see these links:
https://www.infineon.com/dgdl/Infineon-OPTIGA-TPM-Quick-Start-Guide-AdditionalProductInformation-v03_00-EN.pdf?fileId=5546d4626cb27db2016d05dfaac31284

https://aws.amazon.com/ru/blogs/iot/using-a-trusted-platform-module-for-endpoint-device-security-in-aws-iot-greengrass/

Compute Blade Dev additional features

MicroSD card slot

The MicroSD card slot only works with the Raspberry Pi CM4 lite

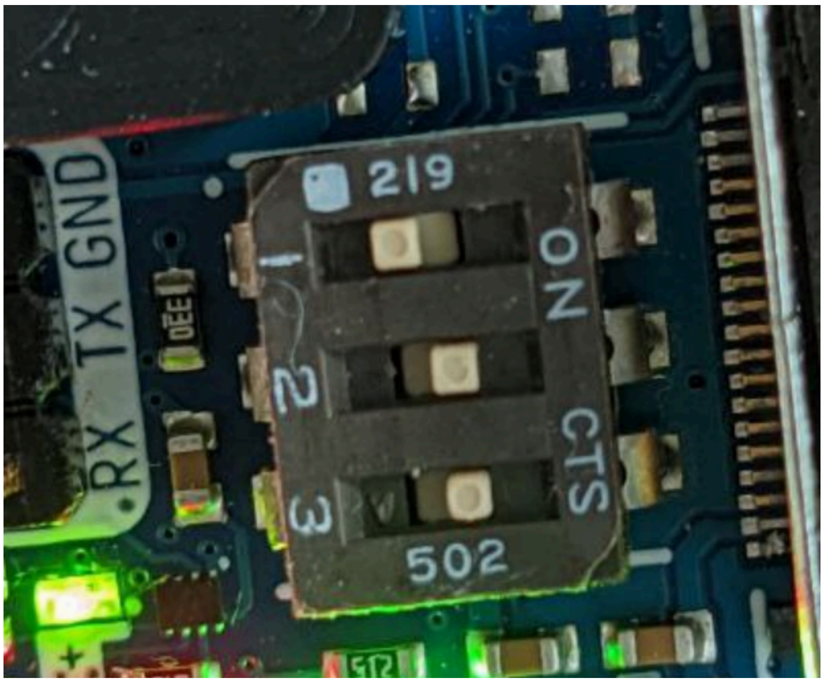


HDMI

The HDMI port is mostly designed for debugging. The port is fragile, so be careful. If possible, use a small flexible cable. If it does break off take care to ensure the board isn't shorted.

Write Protection, Wireless, and Bluetooth Switch

You are able to turn off wireless, Bluetooth and enable the bootloader write protection



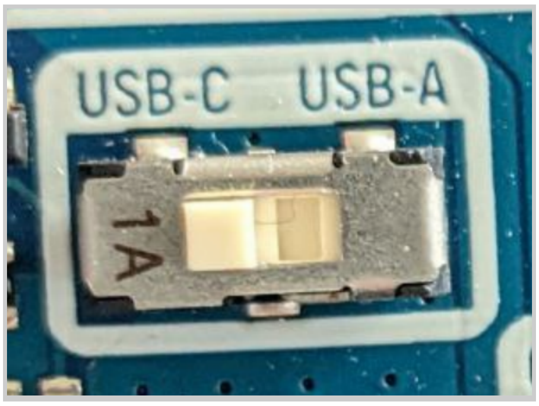
The following pins are:
1 - Write Protection (left - disabled)
2 - Wi-Fi (left - enabled)
3 - Bluetooth (left - enabled)

Switch only when the Blade is off

To activate write protection, you also need to flash the bootloader. Just follow the steps in the official documentation. Link

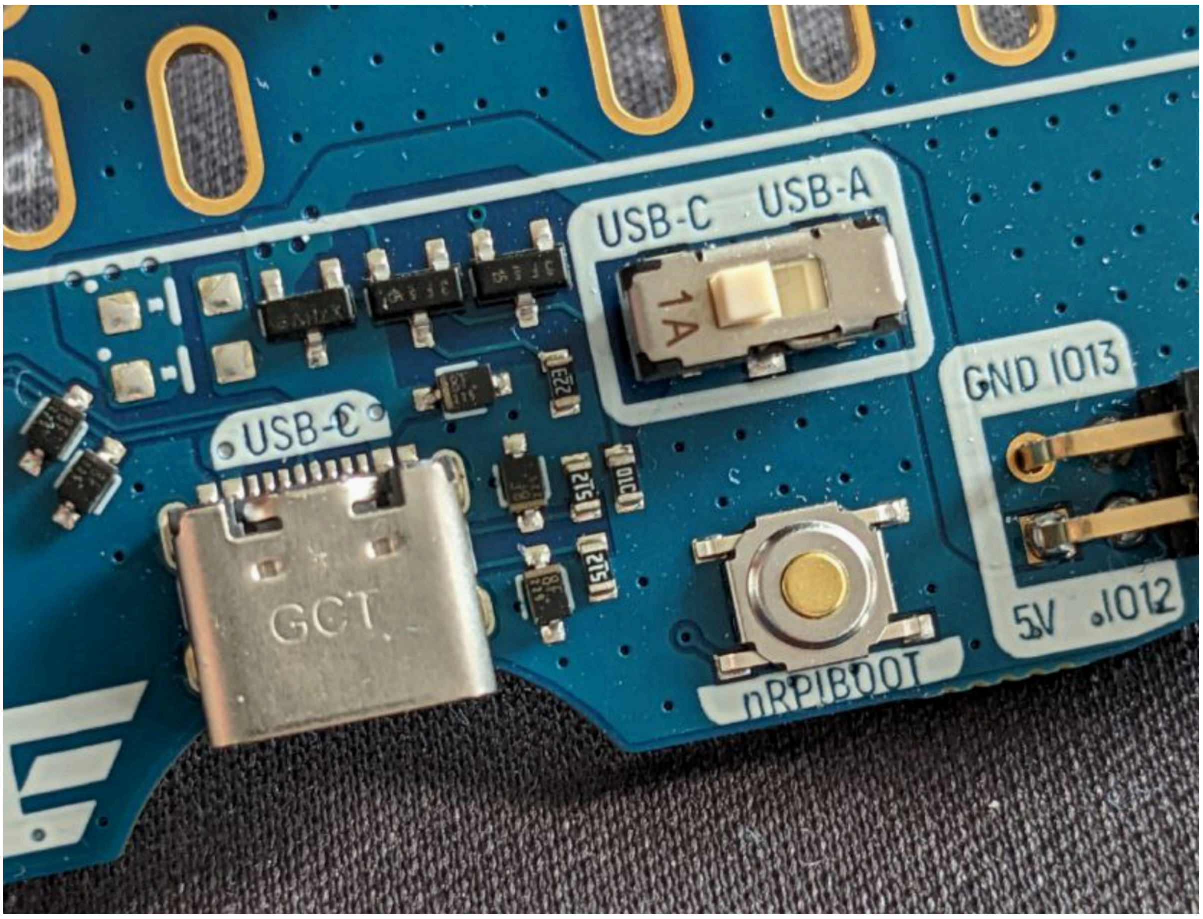
USB-C

The compute module can only operate one USB port at a time. There is a switch to enable which port you would like to utilize.



Flashing the bootloader

To flash the bootloader, you need to turn on the Blade with the nRPIBOOT button pressed (a second after turning the Blade on, you can already release it).
You can turn it on from USB-C or PoE; it doesn't matter. To flash, you need to plug in a USB-C and switch the switch to USB-C (the switch can be switched during operation).



On a computer with Linux to which you will connect the Blade:

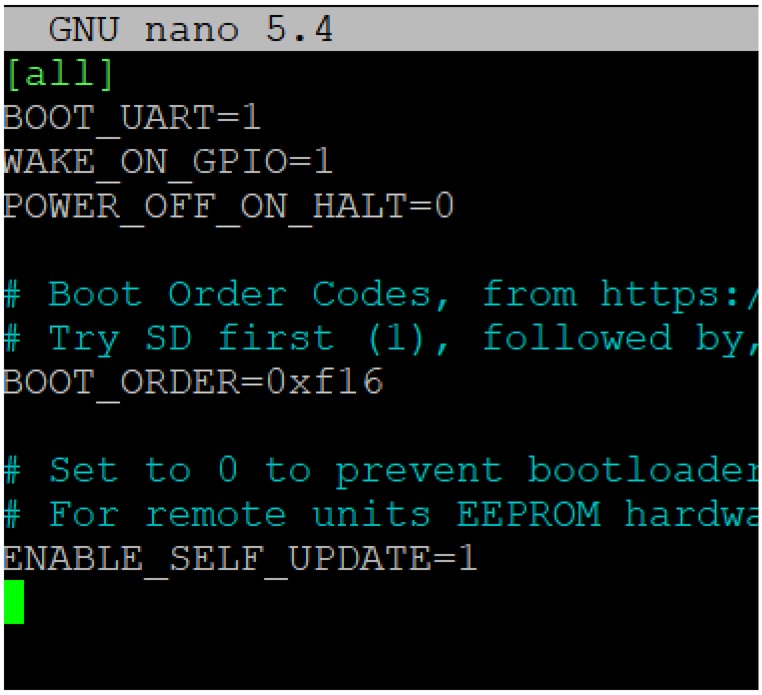
```
sudo apt install libusb-1.0-0-dev
git clone --depth=1 https://github.com/raspberrypi/usbboot
cd usbboot
make
cd recovery
nano boot.conf
```

change the line

```
BOOT_ORDER=0xf16
```

Optionally, I also recommend correcting 0 to 1 on this line to enable UART on boot

```
BOOT_UART=1
```



```
  GNU nano 5.4
[all]
BOOT_UART=1
WAKE_ON_GPIO=1
POWER_OFF_ON_HALT=0

# Boot Order Codes, from https:/
# Try SD first (1), followed by,
BOOT_ORDER=0xf16

# Set to 0 to prevent bootloader
# For remote units EEPROM hardwa
ENABLE_SELF_UPDATE=1
```

That means:
Try NVMe first, followed by SD then repeat.
You can find more about boot order by following this link.

[optional]
to update to the latest bootloader version use repo and the right link:

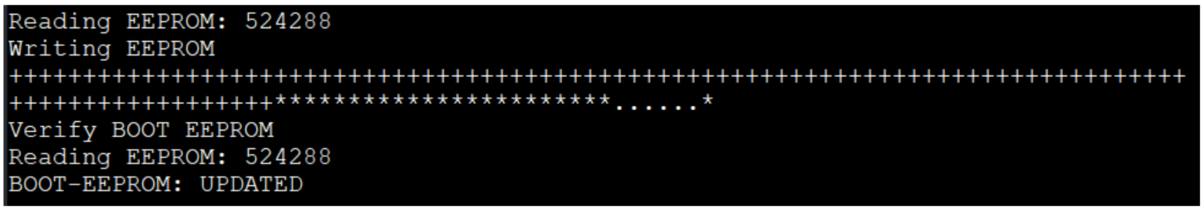
```
rm -f pieeprom.original.bin
curl -L -o pieeprom.original.bin
https://github.com/raspberrypi/rpi-eeprom/raw/4c5aebdb200bc9a2ffd2a0158e
fffb9603c33be7/firmware-2711/latest/pieeprom-2024-04-15.bin
```

[optional]
cd

```
./update-pieeprom.sh
cd ../
sudo ./rpiboot -d recovery
```

If UART is connected, you will see the firmware status. If the HDMI port is connected, the screen will turn green.

on UART (you should use BOOT_UART=1 in boot.conf):

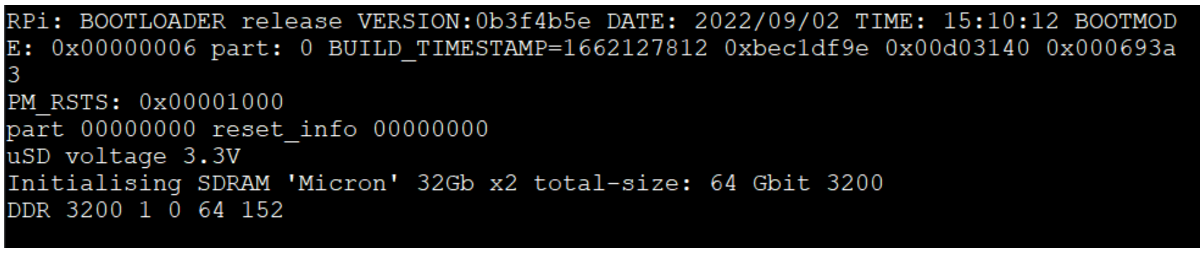


```
Reading EEPROM: 524288
Writing EEPROM
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++***********************......*
Verify BOOT EEPROM
Reading EEPROM: 524288
BOOT-EEPROM: UPDATED
```

Check the bootloader version:

```
vcgencmd bootloader_version
```

Also, on the UART when booting:



```
RPi: BOOTLOADER release VERSION:0b3f4b5e DATE: 2022/09/02 TIME: 15:10:12 BOOTMOD
E: 0x00000006 part: 0 BUILD_TIMESTAMP=1662127812 0xbec1df9e 0x00d03140 0x000693a
3
PM_RSTS: 0x00001000
part 00000000 reset_info 00000000
uSD voltage 3.3V
Initialising SDRAM 'Micron' 32Gb x2 total-size: 64 Gbit 3200
DDR 3200 1 0 64 152
```

If the screen is red and in the UART during flashing:

```
Bad signature pieeprom.bin
FATAL error-code 24
```

You are probably using the wrong pieeprom.original.bin image. The right one should be 512k in size.
pieeprom.original.bin
In my case, the problem was due to an incorrect link:
~~https://github.com/raspberrypi/rpi-eeprom/blob/master/firmware/stable/pieeprom-2022-09-02.bin~~

The correct one:
https://github.com/raspberrypi/rpi-eeprom/raw/master/firmware/stable/pieeprom-2022-09-02.bin

Flashing OS to eMMC / accessing eMMC

In short, it's like this:

- Slide the USB switch to USB-C
  Press Boot (nRPIBOOT) on the Blade and connect it to your Linux/Mac with USB-C cable

- On a computer with Linux to which you will connect the Blade

```
sudo apt install libusb-1.0-0-dev
git clone --depth=1 https://github.com/raspberrypi/usbboot
cd usbboot
make
cd recovery
sudo ./rpiboot
```

If you have a Mac OS you need Homebrew

```
brew install pkgconfig libusb
```

- After that you will be able to mount the Boot partition if present. (it usually happens automatically). Note a stock CM4 has a completely empty eMMC.
  You can fix your files/configs or install the OS using the Raspberry Pi Imager

  For complete instructions, see Jeff Geerling's website: instructions

Notes

Tests

Get CPU temp:

```
vcgencmd measure_temp
```

Get throttling state:

```
vcgencmd get_throttled
```

Display the CPU temperature with an update once per second:

```
watch -n 1 vcgencmd measure_temp
```

Display throttling state with an update once per second:

```
watch -n 1 vcgencmd get_throttled
```

Display CPU frequency with an update once per second:

```
watch -n 1 vcgencmd measure_clock arm
```

Stress test 1:

```
sudo apt install stress-ng mesa-utils -y
stress-ng --cpu 0 --cpu-method fft
```

Stress test 2:

```
sudo apt-get install sysbench
sysbench --test=cpu --cpu-max-prime=20000 run
sysbench --test=cpu --num-threads=8 --cpu-max-prime=20000 run
```

best OC so far:
over_voltage=9
arm_freq=2350

watch -n 1 "vcgencmd measure_clock arm && echo && vcgencmd measure_temp && echo && vcgencmd measure_volts && echo && vcgencmd get_throttled && echo && free -h && echo && vcgencmd pmic_read_adc"

Fan Unit current version

The Smart Fan Unit is a Raspberry Rp2040 based device, it can control the fan itself using two temperature sensors on board (airflow temperature). And it communicates with both blades via UART. It also has a button, two digital LEDs, two regular LEDs (to show which blade is connected or both).

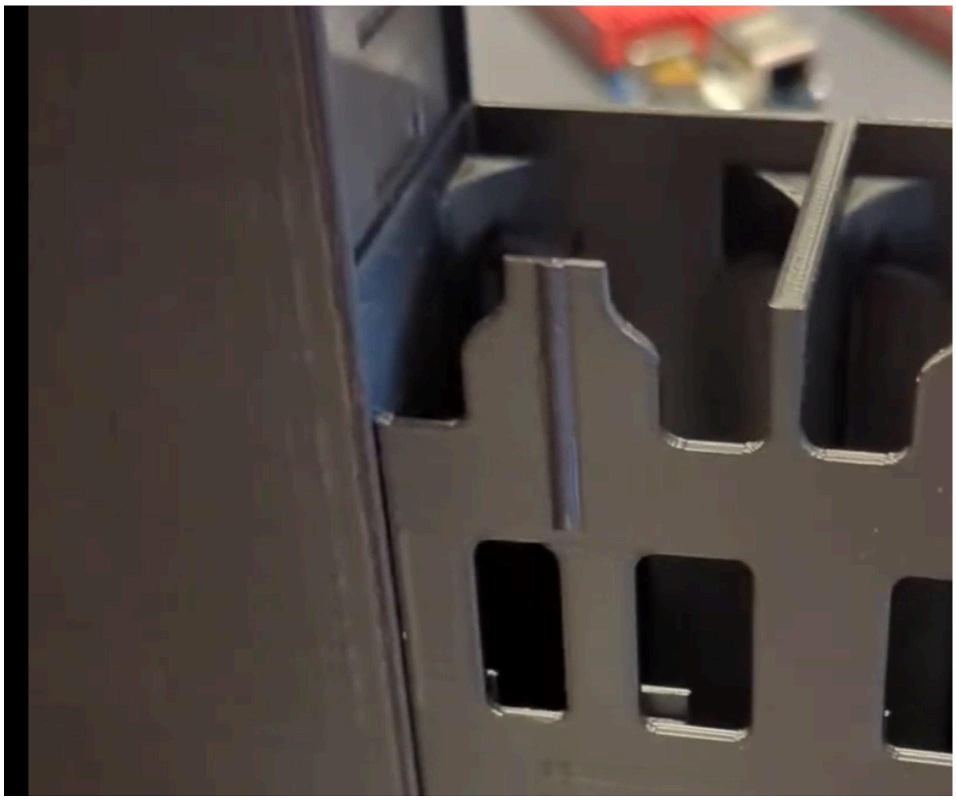
The version without Smart is connected to the left blade in a pair with 4 pins, can measure speed and control the fan.

The non-smart version is currently powered only by the left blade. The version I sent to Jeff is the newest test version powered by both blades.

What do you need to pay attention

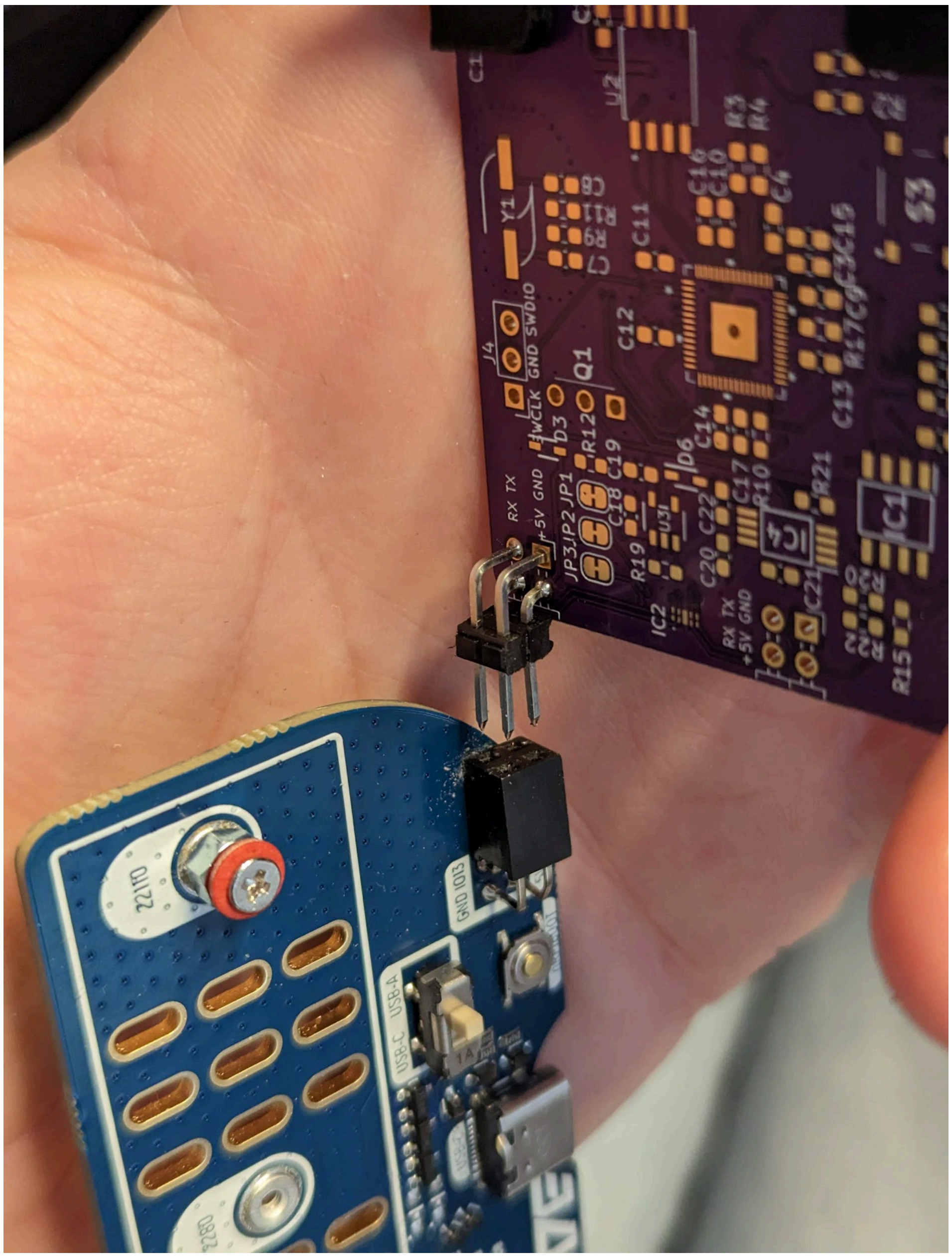
1.

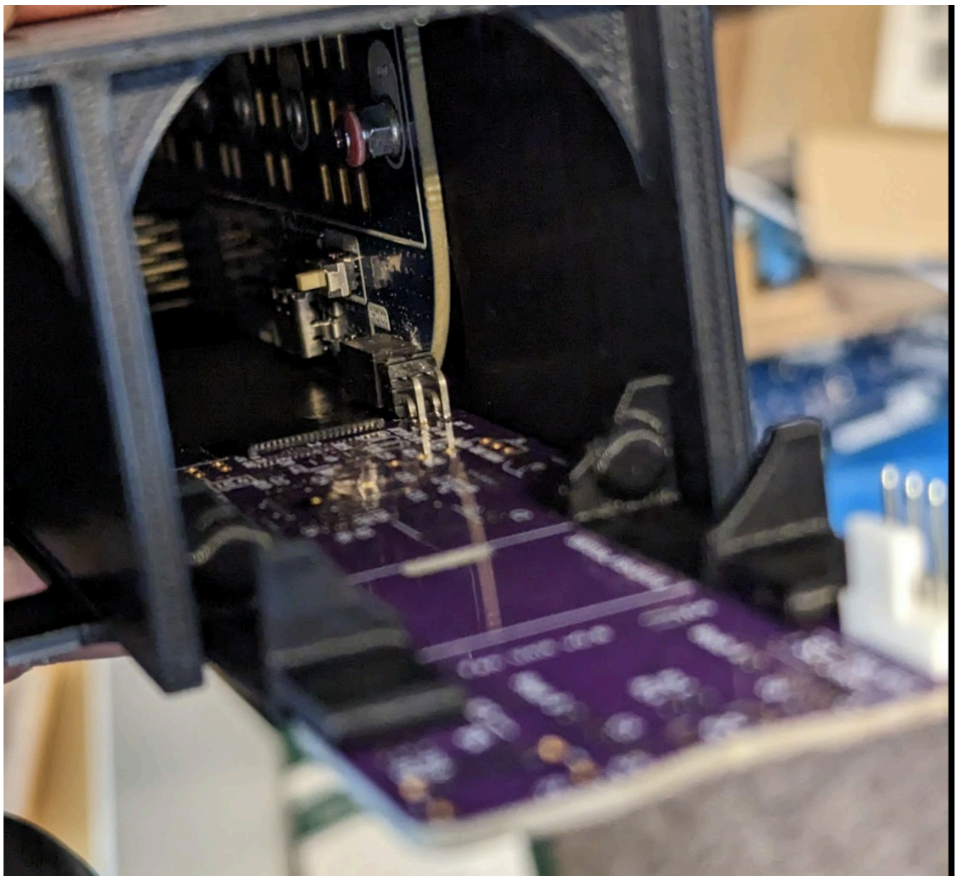
The current iteration of the 3D-printed enclosure has a weakness.
I broke one of the Fan Unit latches by inattention, but in general you just need to not over bend or put the chassis standing on it on the table.



2.

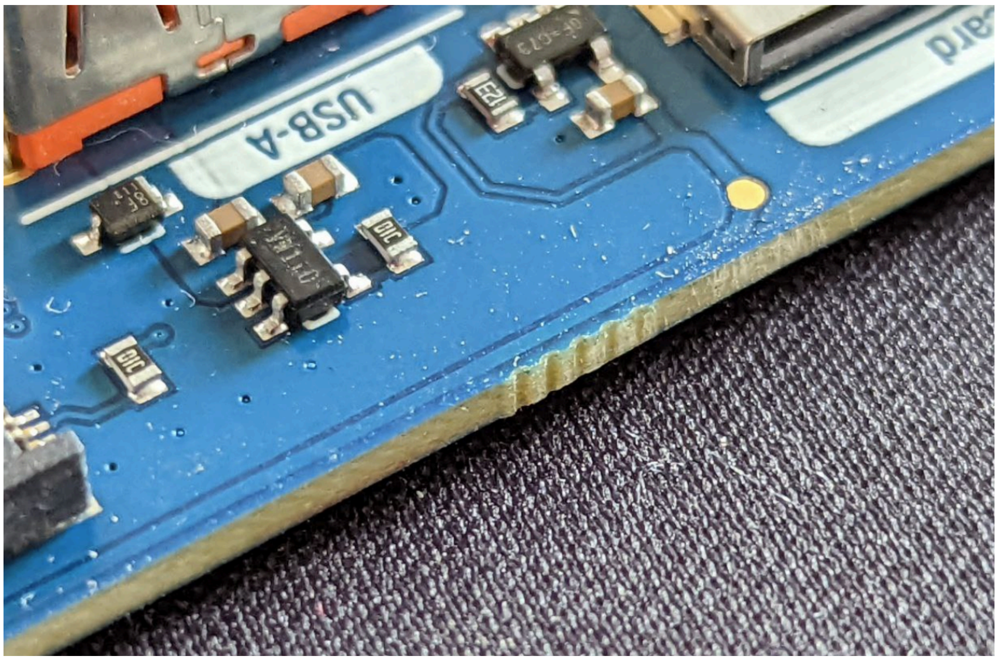
The second thing to look at is the contact of the Fan Unit assembly.
This may not be the most reliable option, because if the connector or pins are slightly bent, they will not fit together or may break. But this is the cheapest and easiest to replace option. Just make sure the first time that they are flat and fit together. You can start with a fanless assembly to check the case


RX TX
+5V GND
SWDIO GND SWCLK
J4
Y1
Q1
JP3 JP2 JP1
IC2
U2
GND IO13
USB-A
USB-C



3.

Processing traces on the edges of the RC2 PCB



Some RC2 blades have irregularities on the borders of the circuit board (the places of separation of the plates), this will be corrected in version 1, but in the meantime need to finalize it with knife or any other tool (incidentally on some blades, I have already done it)

4.

If you printed the case yourself, note that the blade should go in without difficulty. The model should be printed 1in1 with minimal error.
When printing with the Cura slicer, I use a “horizontal expansion” setting of “-0.08”
This is the challenge for the 3D printer. Best of luck!

Known problems

With NetGear switches
GS324TP v1
QS110TPP v1

there is a problem with PoE mode negotiation when using FTP twisted pair. I recommend using UTP in this case.
But maybe this is just my problem. No one else has been able to reproduce it yet.

Disclaimer

I spent a lot of time looking for good and free or cheap documentation solutions. But I'd rather do it here (Google Docs) and move it afterward than spend more time looking for a better solution :)
Moreover, here you can easily give comments or suggestions to help improve the documentation right at the start of Compute Blade v1.
This is the way.