Introduction

The öchìn CM4v2 it's a tiny carrier board for the Raspberry Pi Compute Module 4. It is designed for applications where a powerful machine with low consumption and small dimensions is required. The small form factor makes it interesting for all those applications where the space available is not much and containing the weight is important, such as in robotics, home automation and IOT.

The board is compatible with all Raspberry Pi CM4 modules equipped with eMMC. Depending on your needs, you can select a CM4 module with an SDRAM starting from 1GB up to 8GB and the eMMC from 8GB up to 32GB, with or without the Wi-Fi / BT4 connection.

Key features of the Raspberry Pi CM4 compatible with öchìn are as follows:

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- Small Footprint 55mm × 40mm × 4.7mm module
- H.265 (HEVC) (up to 4Kp60 decode), H.264 (up to 1080p60 decode, 1080p30 encode)
- OpenGL ES 3.0 graphics
- Options for 1GB, 2GB, 4GB or 8GB LPDDR4-3200 SDRAM with ECC
- Options for 8GB, 16GB, or 32GB eMMC Flash
  - Peak eMMC bandwidth 100MBytes/s
  - 2.4 GHz, 5.0 GHz IEEE 802.11 b/g/n/ac wireless
  - Bluetooth 5.0, BLE
  - On board electronic switch to select between PCB trace or external antenna
- 1 × USB 2.0 port (highspeed)
- MIPI CSI-2:
  - 1 × 2-lane MIPI CSI camera port, 1 × 4-lane MIPI CSI camera port
- 1 × USART, 3 × UART, 1 × I2C port, 1 × SPI port, Analog Video Output

Design

The board has the same dimensions as the RPi CM4 module, 55mm x 40mm with rounded corners.
Several interfaces like USARTs, USBs are addressable bye the SM06B-GHS and SM04B-GHS connectors, available on the sides of the board. Two FCC/FPC connector are available to connect two MIPI CSI-2 cameras. The front camera is the “camera 1” which is a MIPI CSI-2 4lane camera. The side camera is the “camera 0” which is a MIPI CSI-2 2lane camera. The two FCC/FPC connectors have contacts on the bottom, the interface is the same used on the Raspberry Pi Zero.
On top of the board there is a 14x1.27mm connector, that could be used to connect a custom board on top, replacing the top PCB cover.
On the bottom of the öchìn carrier board there are the DF40HC(3.0)-100DS-0.4v mating connectors for the CM4 module.

Applications

The powerful Quad core Cortex-A72 CPU and GPU, the big RAM size and the CODECs and Graphic Accelerators available on the CM4 makes the system very interesting for a huge number of applications. Are suitable all the applications related to audio / video manipulation and streaming, neural networks, machine learning, computer vision, etc.

There are several communication interfaces available on board that enables to connect the öchìn to external hardware to connect to the lower abstraction layers, closer to the physical world.

An example of use could be a drone, where a flight controller board sense the physical references and control the actuators and motors to guarantee the stability of the flight. It could handle the stability of the vehicle, collect the data from the sensors and from the remote control and send the telemetry data back to the ground station. This is a very common structure nowadays for a UAS. The öchìn board could be used to add a wide range of functionality to the system.

The öchìn could handles up two MIPI cameras and others connected to the USB interfaces. Other sensors could also be added to the öchìn using the UART, SPI, I2C interfaces. All those data could be processed by several threads running on the CM4.

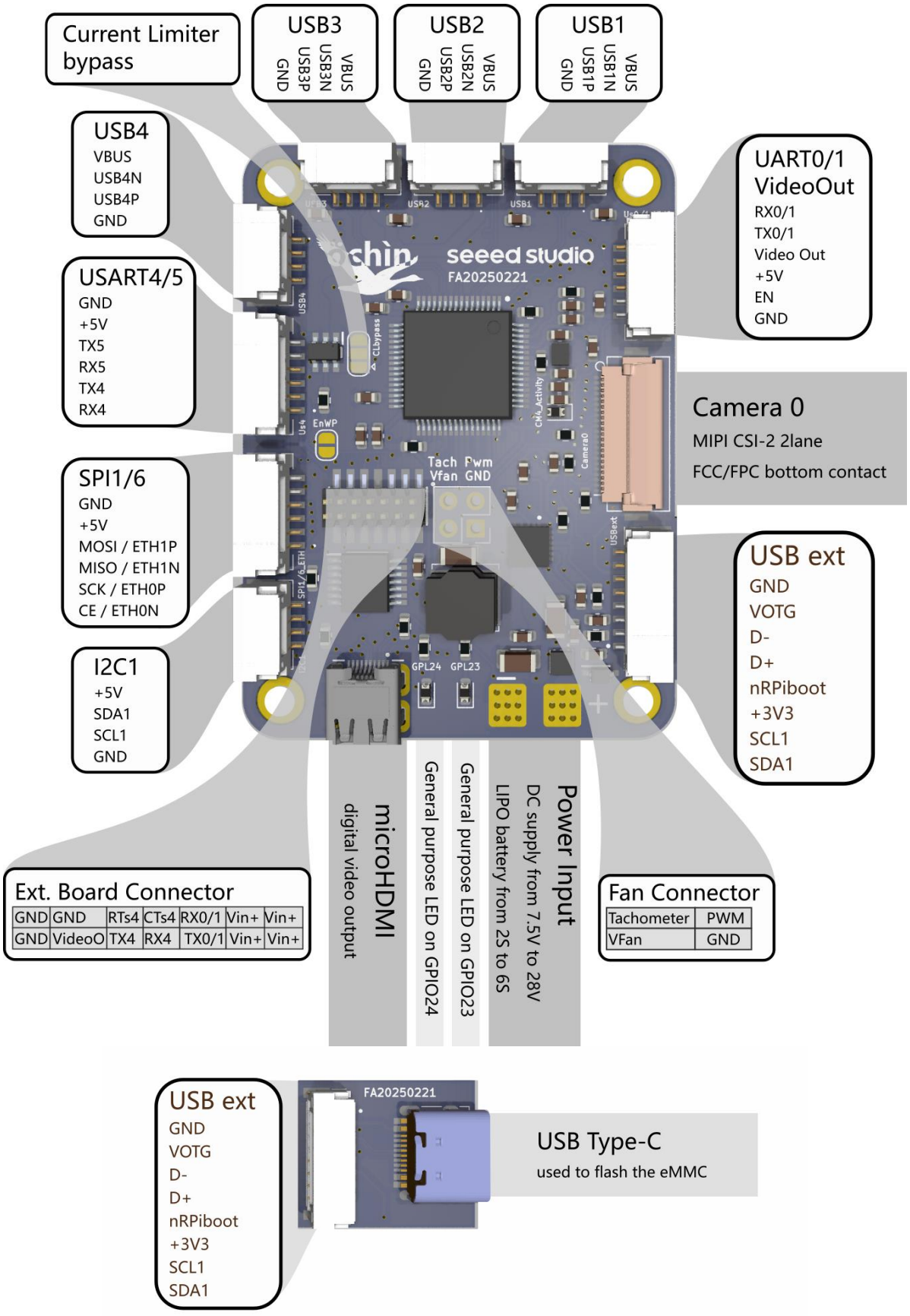
The software infrastructure available within the CM4 enables to code with C/C++, python, MATLAB, and several other languages. There is also a wide number of frameworks that can be used to develop algorithms with Deep Learning, Neural Networks or Computer Vision technique, such as OpenCV, Caffe, PyTorch, TensorFlow etc. The öchìn board also provide the analog video output from the CM4. It means that it's possible to create a video output from the whole set of data, including images and its elaboration, to transmit like a common analog video out. This is very convenient because it gives the chance to see the output of the processing in real time without taking care of creating a digital wireless bridge.

This is only an example, a similar structure could be though for rovers, submarines, airplanes, and robots in general. Of course, the öchìn board it's very interesting for the IOT world also, where the size and weight is important. In this scenario, the AI is relevant and the tendency to miniaturize IOT devices and make them increasingly intelligent requires adequate hardware.

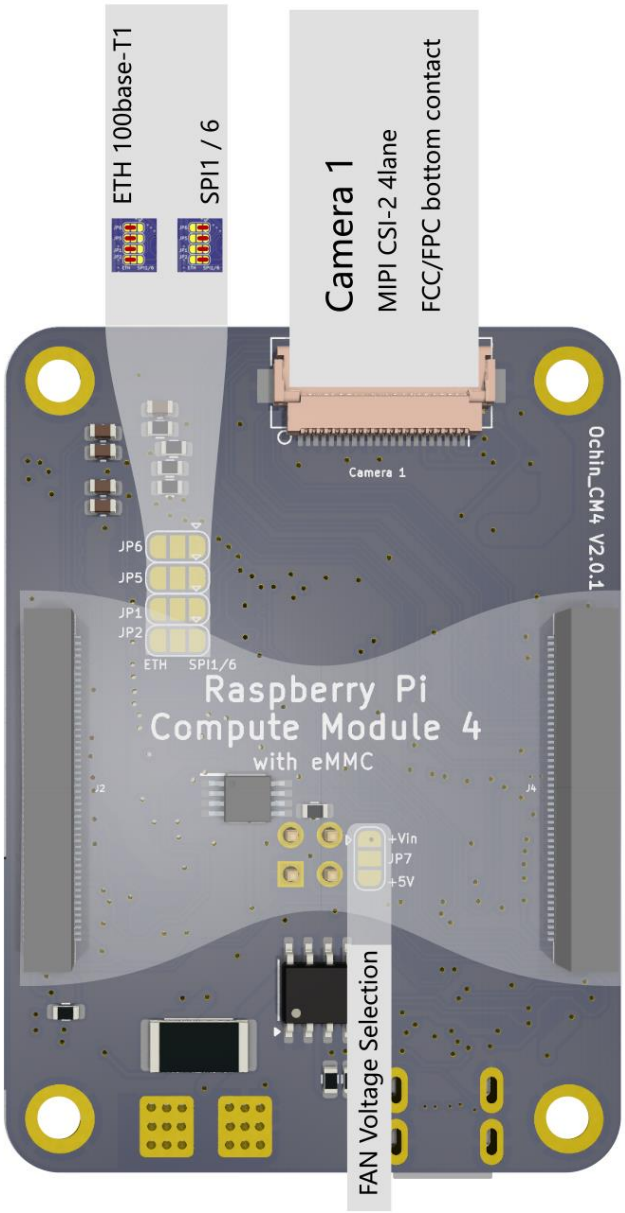
Hardware interfaces

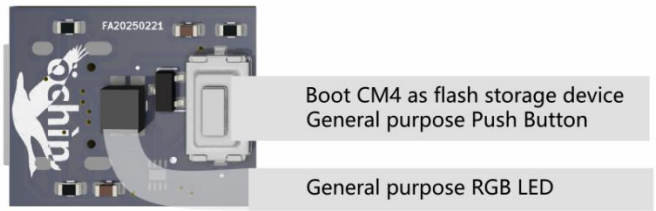
The öchìn CM4v2 board provides the following interfaces on the side's connectors:

- 4x USB 2.0 480Mbps (4x SM04B-GHS-TB(LF)(SN) connectors)
- 1x USB Type-C (for flashing eMMC)
- 2x CSI camera (2x FH12-22S-0.5SH (55) connectors)
- I2C1 (SM04B-GHS-TB(LF)(SN) connector)
- SPI1 / 6 (SM06B-GHS-TB(LF)(SN) connector)
- UART0 / 1 + Video Out (SM06B-GHS-TB(LF)(SN) connector)
- UART4 / UART5 (SM06B-GHS-TB(LF)(SN) connector)
- 1x Ethernet transformerless 100Base-T
- 1x microHDMI
- 2x general purpose LEDs
- 1x RGB LED on external tiny board
- 1x general purpose button on external tiny board
- 1x 4-Wire Fan controller (pwm, tacho, supply voltage selectable, Vin or 5V)


Current Limiter bypass
USB3
VBUS
USB3N
USB3P
GND
USB2
VBUS
USB2N
USB2P
GND
USB1
VBUS
USB1N
USB1P
GND
USB4
VBUS
USB4N
USB4P
GND
UART0/1 VideoOut
RX0/1
TX0/1
Video Out
+5V
EN
GND
USART4/5
GND
+5V
TX5
RX5
TX4
RX4
Camera 0
MIPI CSI-2 2lane
FCC/FPC bottom contact
SPI1/6
GND
+5V
MOSI / ETH1P
MISO / ETH1N
SCK / ETH0P
CE / ETH0N
USB ext
GND
VOTG
D-
D+
nRPiboot
+3V3
SCL1
SDA1
I2C1
+5V
SDA1
SCL1
GND
seeed studio
FA20250221
Tach Pwm
Vfan GND
GPL24 GPL23
microHDMI
digital video output
General purpose LED on GPIO24
General purpose LED on GPIO23
Power Input
DC supply from 7.5V to 28V
LIPO battery from 2S to 6S
Ext. Board Connector
GND GND RTs4 CTs4 RX0/1 Vin+ Vin+
GND VideoO TX4 RX4 TX0/1 Vin+ Vin+
Fan Connector
Tachometer PWM
VFan GND
USB ext
GND
VOTG
D-
D+
nRPiboot
+3V3
SCL1
SDA1
FA20250221
USB Type-C
used to flash the eMMC

Top view


ETH 100base-T1
SPI1 / 6
Camera 1
MIPI CSI-2 4lane
FCC/FPC bottom contact
Camera 1
Öchin_CM4 V2.0.1
JP6
JP5
JP1
JP2
ETH
SPI1/6
Raspberry Pi
Compute Module 4
with eMMC
+Vin
JP7
+5V
FAN Voltage Selection


FA20250221
Boot CM4 as flash storage device
General purpose Push Button
General purpose RGB LED

Bot view

USB

The Raspberry Pi CM4 module is equipped with a single USB2.0 interface. In order to connect multiple USB devices to the CM4 module, the ochin_CM4v2 board is equipped with a 4xUSB2.0 HUB. The four USB ports are accessible via the four SM04B-GHS connectors. We recommend that you read the document 'Wiring and Suggestion' to make the connections correctly.

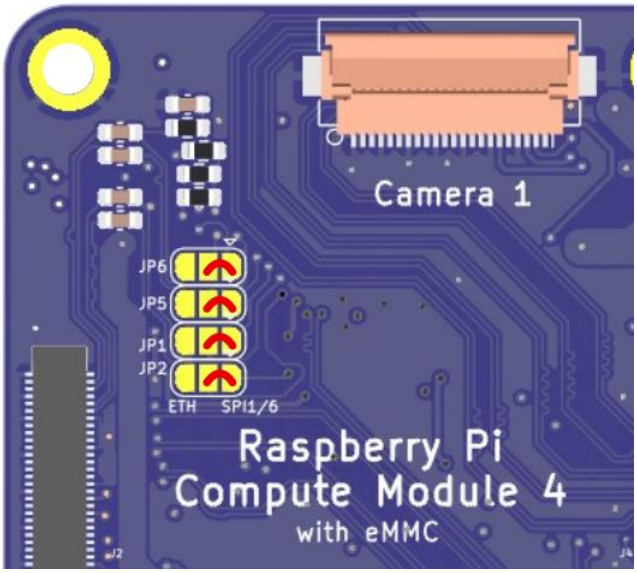
It should be noted that the same USB port on the CM4 is also shared with the USB Type-C connector, which is needed to write the OS in flash. This switching occurs when connecting USB Type-C to a computer. In fact, the VBus provided by the computer switches the USB from the HUB to the Type-C port to allow the module to be flashed.

It is therefore important to bear in mind that none of the 4 USB ports will work as long as the ochin board is connected via a USB Type-C cable to a computer.

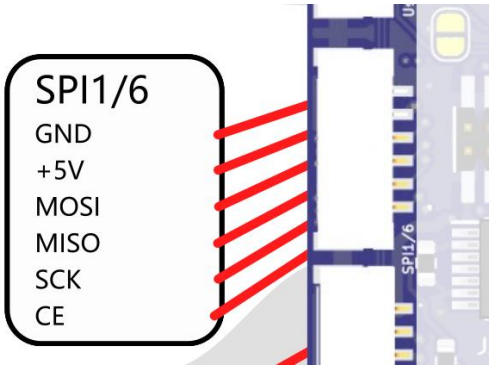
SPI1/Ethernet

The connector, named SPI1/6, can be used to connect to either the SPI1/6 or the Ethernet port. In order to allow either connection, there are four smd switches (normally open) on the back of the ochin_CM4v2 board, which must be soldered appropriately.

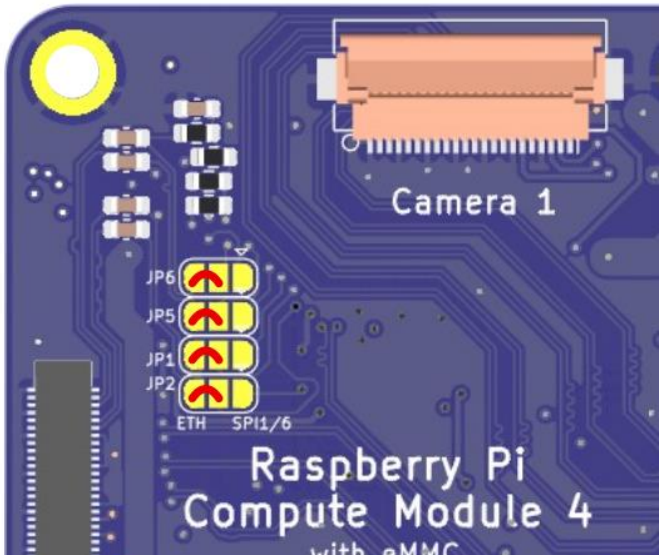
In order to use the connector as SPI1/6, it is necessary to solder the jumpers between the central pad and the pad on the right (indicated by the word SPI1/6) of JP6,5,1,2 as shown in the following figure:



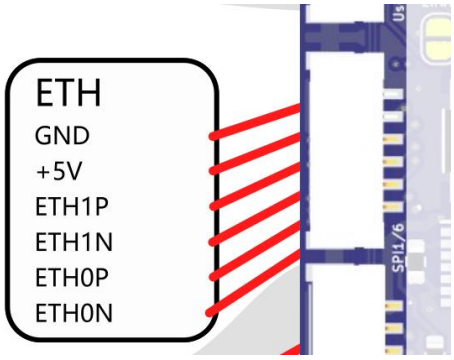
The order of the signals on the SPI1/6 connector is as follows



In the case of connecting to the Ethernet port of the RPI CM4, the smd switches JP6,5,1,2 should be soldered as follows:



In this case, the order of the signals on the SPI1/6 connector will be as follows:

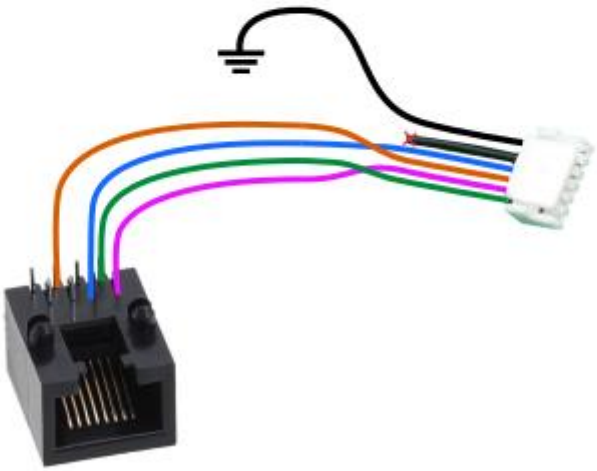


An important note must be made regarding the Ethernet connection. In order to allow Ethernet connectivity and without affecting the size of the ochin_CM4v2 board, it was decided to adopt a transformerless configuration, like the one adopted and described

by Texas Instruments in Application Note AN-1519 for their DP83848 Ethernet devices. It is therefore advisable to study and understand the AP-1519 before deciding to use the Ethernet port in a transformerless configuration.

The following image represents the connection diagram for connecting to the ochin_CM4v2 using a standard RJ45 cable:

Signal Name	JST GHR-6V Pin Number	RJ45 Pin Number
ETH0_N	1	2
ETH0_P	2	1
ETH1_N	3	4
ETH1_P	4	3
		nc
GND	6	nc
		nc
		nc

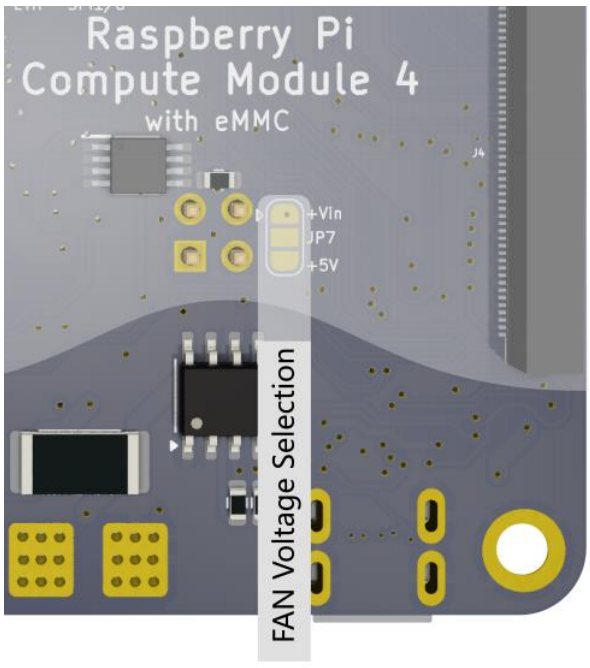


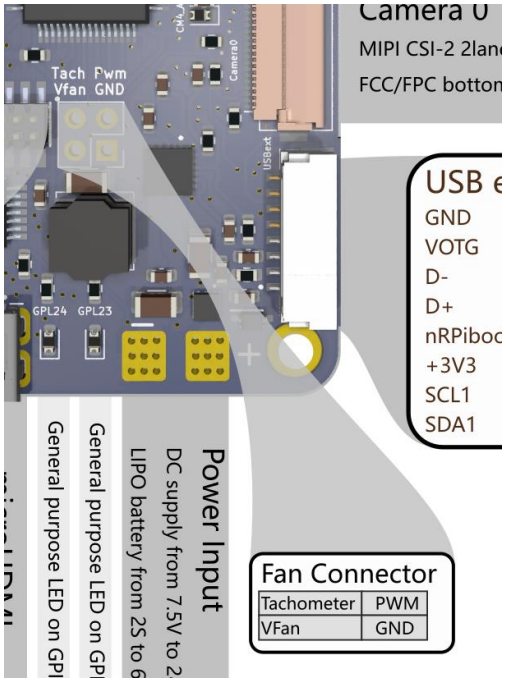
The Fan Controller

he Ochin board features an integrated EMC2301 fan controller connected to the I²C1 bus. To support external fans operating at 5V, 12V, or higher voltages, a normally open SMD switch is provided on the bottom side of the board and must be soldered manually to enable fan power routing.

By default, if the EMC2301 is not actively configured via software, the fan will operate at maximum speed. When properly controlled, the EMC2301 allows for PWM-based speed regulation of both 3-wire (open-loop) and 4-wire (tacho-sensed) fans.

A basic Python test script is included in the documentation to verify functionality. For accurate integration and usage, please refer to the EMC2301 datasheet.





EMC2301_test.py

```python
#sudo apt-get install python3-smbus
#dtparam=i2c_arm=on,i2c_arm_baudrate=400000
from smbus import SMBus
import time

i2cbus = SMBus(1)  # Create a new I2C bus
EMC2301_ADDRESS = 0x2f
Fan1Setting = 0x30
TACH_LSB_REG = 0x3E  # Reg LSB
TACH_MSB_REG = 0x3F  # Reg MSB

def count2RPM(count):
    EDGES_PER_REV = 2
    rpm = 3932160*2/count
    return rpm

def readTacho():
    lsb = i2cbus.read_byte_data(EMC2301_ADDRESS, TACH_LSB_REG)
    msb = i2cbus.read_byte_data(EMC2301_ADDRESS, TACH_MSB_REG)
    tach_count = (msb << 8) | lsb
    return count2RPM(tach_count)

def main():

    i2cbus.write_byte_data(EMC2301_ADDRESS, Fan1Setting, 0x00)
    time.sleep(4)
    print("tacho: "+"%.2f" % readTacho())
    i2cbus.write_byte_data(EMC2301_ADDRESS, Fan1Setting, 0x64)
    time.sleep(4)
    print("tacho: "+"%.2f" % readTacho())
    i2cbus.write_byte_data(EMC2301_ADDRESS, Fan1Setting, 0xff)
    time.sleep(4)
    print("tacho: "+"%.2f" % readTacho())
    i2cbus.write_byte_data(EMC2301_ADDRESS, Fan1Setting, 0x64)
    time.sleep(4)
    print("tacho: "+"%.2f" % readTacho())


if __name__ == "__main__":
    main()
```

The Current Limiter

The IC4 - AP22615AWU-7 is a single-channel current-limited high-side power switch with output OVP optimized for USB and other hot-swap applications. On the ochin_CM4 board, the current threshold of the AP22615AWU-7 is set to 3A. If the IC4 it's triggered, it will cut the VBUS on the USBs to keep the CM4 supplied while an overcurrent condition happens. There are four possible condition that triggers the current limiter:
(from the AP22615 datasheet Document number: DS41022 Rev. 5 – 2):

Overcurrent and Short-Circuit Protection
An internal-sensing FET is employed to check for overcurrent conditions. Unlike current-sense resistors, sense FETs do not increase the series resistance of the current path. When an overcurrent condition is detected, the device maintains a constant output current and reduces the output voltage accordingly. Complete shutdown occurs only if the fault stays long enough to activate thermal limiting.

Three possible overload conditions can occur. In the first condition, the output has been shorted to GND before the device is enabled or before V_{IN} has been applied. The AP22815/AP22615 senses the short-circuit and immediately clamps output current to a certain safe level.

In the second condition, an output short or an overload occurs while the device is enabled. At the instance the overload occurs, higher current can flow for a very short period of time before the current limit function can react. After the current limit function has tripped, the device switches into current limiting mode, and the current is clamped at I_{LIMIT} or I_{SHORT}.

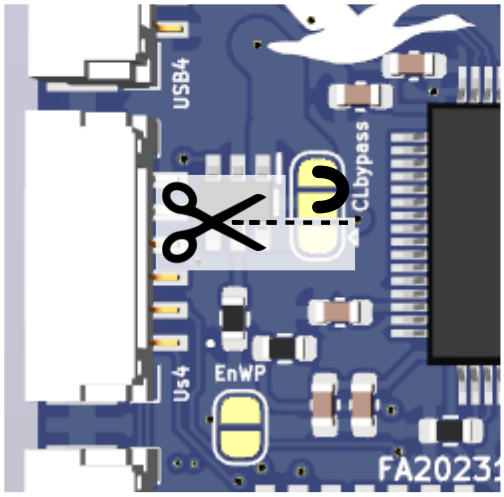
In the third condition, the load is gradually increased beyond the recommended operating current. The current is permitted to rise until the current-limit threshold (I_{TRIG}) is reached or until the thermal limit of the device is exceeded. The AP22815/AP22615 is capable of delivering current up to the current-limit threshold without damaging the device. Once the threshold is reached, the device switches into its current limiting mode and is set at I_{LIMIT}.

Thermal Protection
Thermal protection prevents the IC from damage when heavy-overload or short-circuit faults are present for extended periods of time. The AP22815/AP22615 implements a thermal sensing to monitor the operating junction temperature of the power distribution switch. Once the die temperature rises to approximately +140°C due to excessive power dissipation in an overcurrent or short-circuit condition the internal thermal sense circuitry turns the power switch off, thus preventing the power switch from damage. Hysteresis is built into the thermal sense circuit allowing the device to cool down approximately +35°C before the switch turns back on. The switch continues to cycle in this manner until the load fault or input power is removed. The FLG open-drain output is asserted when an overtemperature shutdown or overcurrent occurs with 7ms deglitch.

When V_{IN} operates below 4V, the lower V_{IN} voltage results in higher equivalent R_{ON} and might potentially cause the chip to enter thermal cycling condition by higher output current.

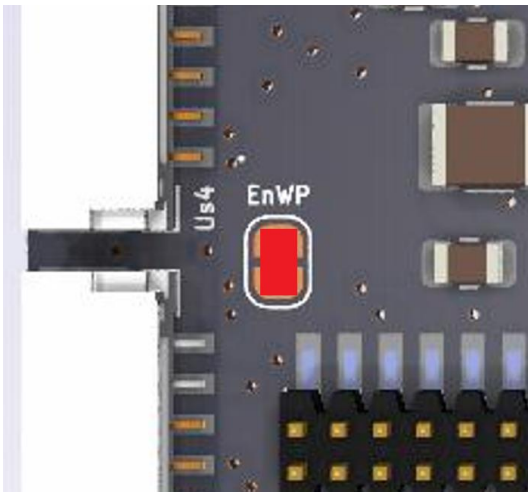
In case more current is needed on the USBs, it is necessary to short JP4 (CLbypass). Before shorting JP4, it is important to cut the "normally closed" trace, between pads 1 and 2 of JP4 (pad 1 is the one indicated by the triangular marker). This is done by using a sharp blade and making sure that you only cut the connection between the two pads, and if necessary, test it with a multimeter. If the current limiter is bypassed, the USB power supply is connected directly to the output of the +5V switching regulator, the same one that powers the CM4. The power supply can provide up to 8A, leaving 2A for the CM4 (the manual says 1.4A but it is best to be cautious); 6A remains available for USB and external peripherals. That said, before removing the limiter, it is important to

know how much current the external peripherals will draw, so leave current for the CM4, otherwise it will reboot.



eMMC write protection

To write protect the eMMC it's necessary to short the jumper named EnWp:



The current sensor INA219

In order to measure the current drawn by the ochin_CM4v2 board and all its peripherals, it was equipped with an INA219 current sensor. The chip is located on the bottom of the pcb and works by measuring the potential drop at the ends of a shunt resistor. The chip is connected to the CM4 module via the I2C1 bus (addr. 0x40) by means of which it is possible to read the input voltage measurement, the shunt voltage and after calibration of the chip also power and current drawn. To calculate the current from the shunt voltage, it is necessary to know the shunt resistance, which in the case of the ochin_CM4v2 is 100mOhms. For details on how to read, write and calibrate the INA219 chip, it is necessary to study its datasheet, provided by Texas Instruments.

The LEDs GPL23 and GPL24

The anodes of the two LEDs GPL23 and GPL24 are connected to the GPIO23 and GPIO24 pins of the Raspberry Pi CM4, respectively. They can be used for debugging purposes or for status signalling. It is possible to manage the two LEDs by means of the Raspberry Pi's GPIO management libraries.

Below is an example of managing the two LEDs using a script in Python:

```python
from RPi import GPIO
import time

#CPU Ref Number
GPIO.setmode(GPIO.BCM)

#set GPIO23 and GPIO24 as output
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)

#endless loop
while (True):
    GPIO.output(23, GPIO.HIGH)
    GPIO.output(24, GPIO.LOW)
    time.sleep(1)
    GPIO.output(23, GPIO.LOW)
    GPIO.output(24, GPIO.HIGH)
    time.sleep(1)
```

The S1 button

The S1 button on the small external board has two functions. The first is to put the CM4 module in 'flash mode' to allow the user to access the flash on the module itself. This procedure is explained in more detail in the "Software Configuration" section. If the CM4 module is started up in normal mode, the S1 button also performs the function of a "user button". This is possible because in addition to being connected to the "nRPIBOOT" pin, it is also connected to the GPIO4 pin. The button can therefore be used during normal operation to interact with the sw running on the CM4 module. To know the status of the button, i.e. whether it is pressed or not, simply configure the GPIO4 as an input and monitor its status. Since GPIO4 is "active low", it is to be expected that it will go to "false" if S1 is pressed and be "true" the rest of the time.

Below is an example of managing the two LEDs using a script in Python:

```
from RPi import GPIO
import time

#CPU Ref Number
GPIO.setmode(GPIO.BCM)

#set GPIO23 and GPIO24 as output
GPIO.setup(23, GPIO.OUT)
GPIO.setup(24, GPIO.OUT)
#set the GPIO4, pin 54 as input
GPIO.setup(4, GPIO.IN)

#endless loop
while (True):
    if GPIO.input(4) == 0:
        GPIO.output(23, GPIO.HIGH)
        GPIO.output(24, GPIO.LOW)
    else:
        GPIO.output(23, GPIO.LOW)
        GPIO.output(24, GPIO.HIGH)
```

The RGB led

On the external board, next to the S1 button is an RGB LED. This LED is present for the purpose of being used at will by the user, to communicate any information by means of all possible colour tones, intensities and flashes. Management of the RGB LED is entrusted to a chip called KTD2026, which is connected to the CM4 module via the I2C1 interface.

Using the I2C libraries, it is possible to configure the chip to vary the intensity of each of the three basic colours and their respective transitions. For more precise and detailed information on the handling of the chip, it is necessary to study the datasheet of the KTD2026, available on the website of the manufacturer, Kinetic Technologies (https://www.kinet-ic.com).

Below is an example of managing the two LEDs using a script in Python:

```python
#sudo apt-get install python3-smbus
#dtparam=i2c_arm=on,i2c_arm_baudrate=400000
from smbus import SMBus
from RPi import GPIO
import time
#CPU Ref Number
GPIO.setmode(GPIO.BCM)
#set the GPIO4, pin 54 as input
GPIO.setup(4, GPIO.IN)
def main():
    # Define registers values from datasheet
    ENRST = 0x00  #
    FlashPer1 = 0x01  #
    FlashOn1 = 0x02  #
    FlashOn2 = 0x03  #
    ChCTRL = 0x04  #
    RampRate = 0x05  #ramp-up/down transitions from 0% to 100%
    Led1Iout = 0x06  #1 led brightness
    Led2Iout = 0x07  #2 led brightness
    Led3Iout = 0x08  #3 led brightness
    Led4Iout = 0x09  #4 led brightness(for KTD2027)
    i2cbus = SMBus(1)  # Create a new I2C bus
    time.sleep(1)
    i2caddress = 0x32  # Address of KTD2026 device
    try:
        #ENRSTstat = i2cbus.read_byte_data(i2caddress, ENRST)  # Read ENRST
        #print(ENRSTstat) # print the value ENRST
        ENRSTstat = i2cbus.write_byte_data(i2caddress, ENRST, 0x07)  # chip reset
        print("chip reset done")
    except:
        print("chip reset failed")
        time.sleep(0.5)
        i2cbus.write_byte_data(i2caddress, ChCTRL, 0x15) #led 1,2,3 ON
        print("LEDs 1,2,3 always on")
```

```
    print("Blue")
    i2cbus.write_byte_data(i2caddress, Led1Iout, 0xff)
    i2cbus.write_byte_data(i2caddress, Led2Iout, 0x00)
    i2cbus.write_byte_data(i2caddress, Led3Iout, 0x00)
    time.sleep(2)
    print("Green")
    i2cbus.write_byte_data(i2caddress, Led1Iout, 0x00)
    i2cbus.write_byte_data(i2caddress, Led2Iout, 0xff)
    i2cbus.write_byte_data(i2caddress, Led3Iout, 0x00)
    time.sleep(2)
    print("Red")
    i2cbus.write_byte_data(i2caddress, Led1Iout, 0x00)
    i2cbus.write_byte_data(i2caddress, Led2Iout, 0x00)
    i2cbus.write_byte_data(i2caddress, Led3Iout, 0xff)
    time.sleep(2)

    while (True):
        if GPIO.input(4) == 0:
            print("Blue")
            i2cbus.write_byte_data(i2caddress, Led1Iout, 0xff)
            i2cbus.write_byte_data(i2caddress, Led2Iout, 0x00)
            i2cbus.write_byte_data(i2caddress, Led3Iout, 0x00)
        else:
            print("Green")
            i2cbus.write_byte_data(i2caddress, Led1Iout, 0x00)
            i2cbus.write_byte_data(i2caddress, Led2Iout, 0xff)
            i2cbus.write_byte_data(i2caddress, Led3Iout, 0x00)


if __name__ == "__main__":
    main()
```

Software Configuration

The öchìn board is only compatible with CM4 module with eMMC installed, because we decided to not mount the SD card slot to save space. Having the eMMC instead of SD also gives several advantages:

1. Faster than SD with speed up to 100Mbytes/s
2. More robust to physical corruption
3. Longer lifetime
4. More rugged

Having to write the ISO image directly to the eMMC is a bit more complicated than flash an SD card, but not so much thanks to the great work done by Raspberry Pi.

The procedure to flash the CM4 module is well explained in the Raspberry Pi Documentation:

https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md

The procedure it's straightforward, what you need to do in synthesis is the following:

1. Power up the board with boot configuration as "mass storage device".
    - To do so you need to power off the board (no Vin connected)
    - Press the "nRPiboot" button on the external tiny board and, keeping it pressed power up the board using the "Vin" Connector.
    - Connect the board to your PC using the USB Type-C port

2. Run the "RPiboot" software, downloaded from the Raspberry Pi website. After the software starts, the PC will see the partition of the eMMC like if it would an SD into the SDcard reader.
3. Flash the eMMC as you normally do with the SD. You could use Win32DiskImager on Windows or "dd" command on Linux.

Once you have the system running for the first time, the USB interface will not work because on CM4 it is disabled by default. To use the USBs, you need to enable them in the "config.txt" file, adding the following line at the end of the file:

dtoverlay=dwc2,dr_mode=host

The "config.txt" file could be found within the "boot" partition. To be noted is that the "boot" partition is also accessible when you connect the CM4 to the PC booted in MSD mode, since it is formatted as FAT32.

NB

The Type-C VBUS is not connected to the power supply of the ochin board. This is to not supply everything you may have attached to the HUB (Wi-Fi dongle, cameras etc..), the power provided by the PC may not be sufficient. It means that if you only plug the USB Type-C nothing is going to happens... To power on the board the supply on the Vin connector is needed.

The Type-C VBUS it is used to switch the CM4 usb from the HUB to the Type-C port, this is to be able to flash the eMMC. For this reason, during the normal operations, the Type-C connector shouldn't be connected to anything.