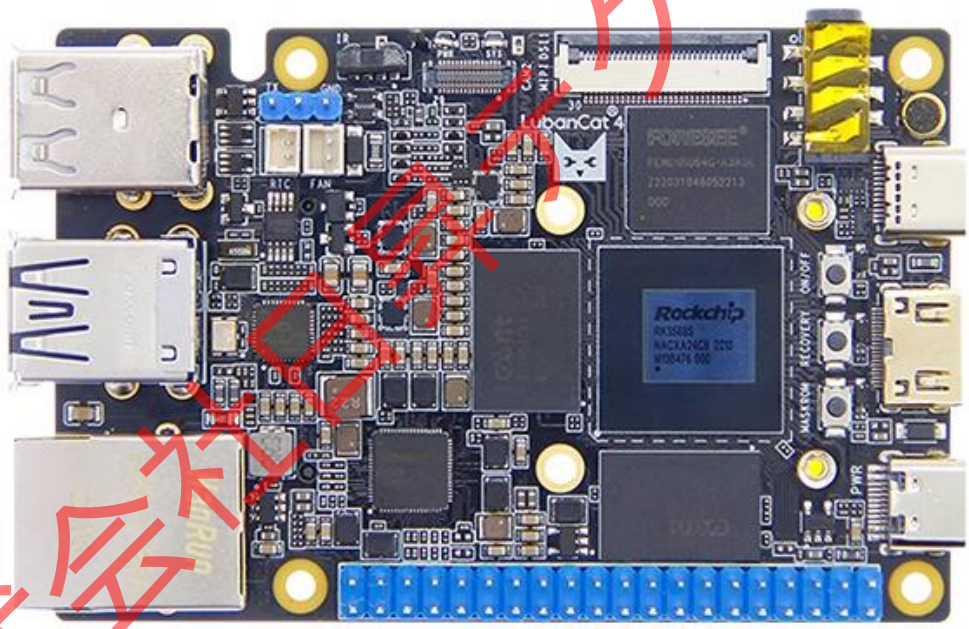
RK3588S 開発ボード
クイックスタート マニュアル

株式会社日昇テクノロジー
https://www.csun.co.jp
info@csun.co.jp

作成日 2024/06/28



- 修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2024/06/28

目　次

1　概述

本ボードは高性能と豊富な機能より、教育、商業応用、工業制御などの分野で、幅広い応用シーンを備えている：マイクロコンピュータ、Linux サーバー、スマートホームハブ、工業ボード等。

特徴：Ubuntu、Debian、Android などのシステムをサポートし、アプリ開発者及びシステム開発者向けの複数の教材を提供しています。組み込み業界の初心者でも、マニュアルに基づいて開発を完了できます。また、組み込みのベテランにとっては、製品の二次開発プロセスを加速させることができます。

2　準備

2.1　ハードウェア

① LubanCat4 ボード　1 台

② 5V/4A 電源アダプタ　1 台

③ 32GB TF カード　1 枚

④ カードリーダー　1 台

⑤ 5 インチ静電タッチパネル液晶或いは HDMI ディスプレイ　1 台

2.2　ソフトウェア

① イメージファイルダウンロード URL：

http://www.dragonwake.com/download/LubanCat4/3-LinuxIMG.zip
http://www.dragonwake.com/download/LubanCat4/4-AndroidIMG/Android12.zip
http://www.dragonwake.com/download/LubanCat4/4-AndroidIMG/Android13.zip
http://www.dragonwake.com/download/LubanCat4/4-AndroidIMG/AndroidTV.zip

② 関連ツールダウンロード URL：

http://www.dragonwake.com/download/LubanCat4/6-tools.zip

3　システムイメージファイルの書き込み

ボードを実行するには、システムの書き込み、システムの起動、システムへのログインの 3 つのステップが必要です。

LubanCat4 ボードは emmc 搭載しているので、出荷時にデフォルトでシステムイメージファイルが書き込まれています。新しいイメージファイルを書き込む必要がない場合は、このセクションをスキップしてください。

LubanCat4 ボードのシステムイメージファイルには、さまざまなルートファイルシステムが含まれており、必要に応じて選択してダウンロードすることができます。

注意：システムイメージファイルは LubanCat4 ボードのコアであり、linux カーネル、基本的なプログラムなどを含み、LubanCat4 ボードを実行するために必要なので、ボードを実行する前にボードのイメージファイルを書き込む必要があります

3.1　イメージファイルの取得

3.1.1　イメージファイル・ネーミング・ルール

lubancat-(プロセッサ・モデル)-(ルートファイルシステム)-(デスクトップ・タイプ)-(更新時間)_update

3.1.1.1　プロセッサー・モデル

rk3588s：rk3588s プロセッサーを使用したボード、例えば LubanCat-4 ボード

3.1.1.2　ルートファイルシステム

debian11：システムのルートファイルシステムは debian11 です

3.1.1.3　デスクトップ・タイプ

1. gnome：gnome スイートを使用したデスクトップ版イメージ
2. lite：コマンドラインバージョン、デスクトップなし

3.1.1.4　更新日

更新日のフォーマットは yyyymmdd_update(年月日_update)

3.1.1.5　例

LubanCat-rk3588 debian11 の汎用イメージを例に挙げます

例：　lubancat-rk3588-debian11-gnome-20230807_update

1.ボードの該当モデル：rk3588 プロセッサー搭載ボード

2.ルートファイルシステム：debian11

3.システムカテゴリ：gnome デスクトップ版

4.更新日：2023 年 08 月 07 日

3.2　Windows プラットフォームでイメージファイルを書き込む

3.2.1　eMMC への書き込み

3.2.1.1　ツールの取得とインストール

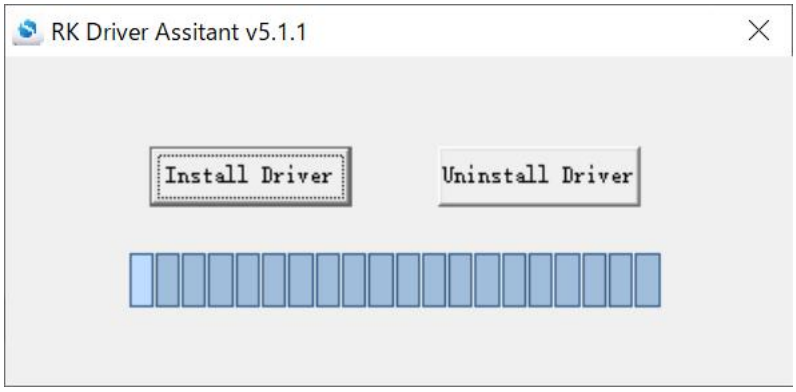
eMMC への書き込みには書き込みツール RKDevTool とドライバアプリ DriverAssitant が必要

フォルダ 6-tools

3.2.1.1.1 DriverAssitant のインストール

DriverAssitant ソフトウェア圧縮パッケージを解凍し、DriverInstall.exe をダブルクリックしてドライブインストール画面を開きます。

ドライブのインストールを開始するには、「Install Driver」をクリックします。以前に古いドライバがインストールされているかどうか不明な場合は、「Uninstall Driver」をクリックして古いドライバを削除してから、「Install Driver」をクリックします。

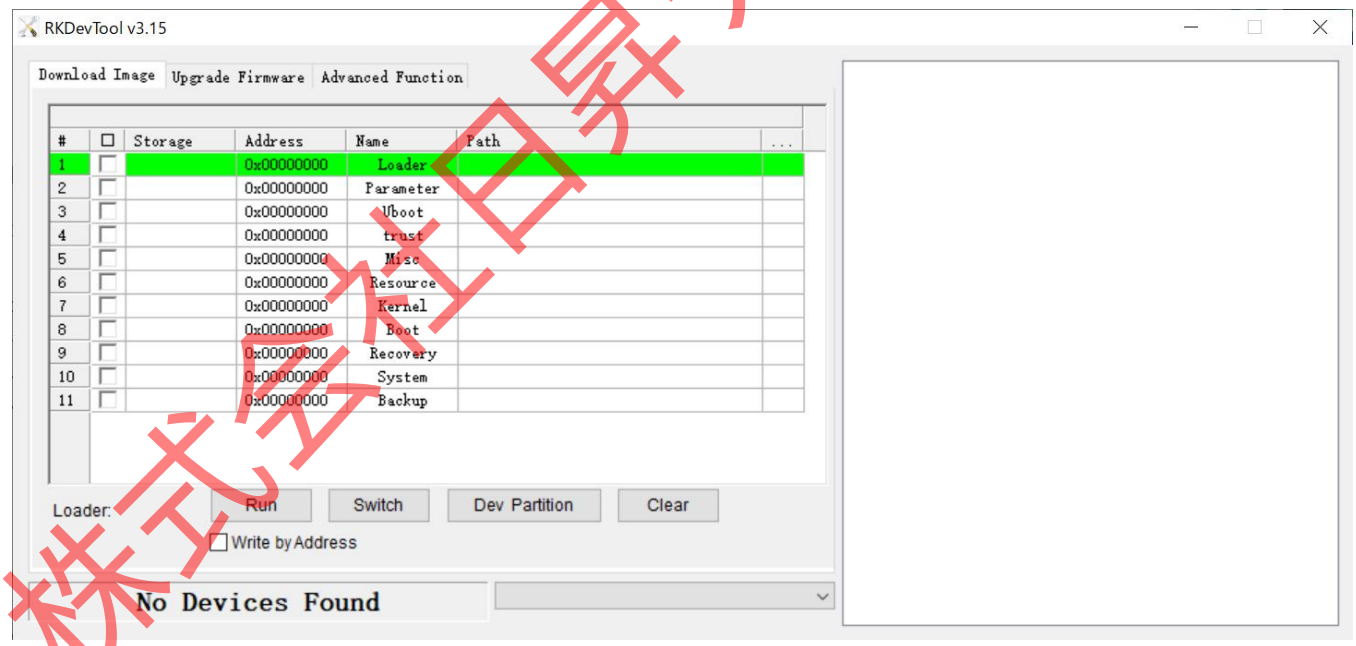


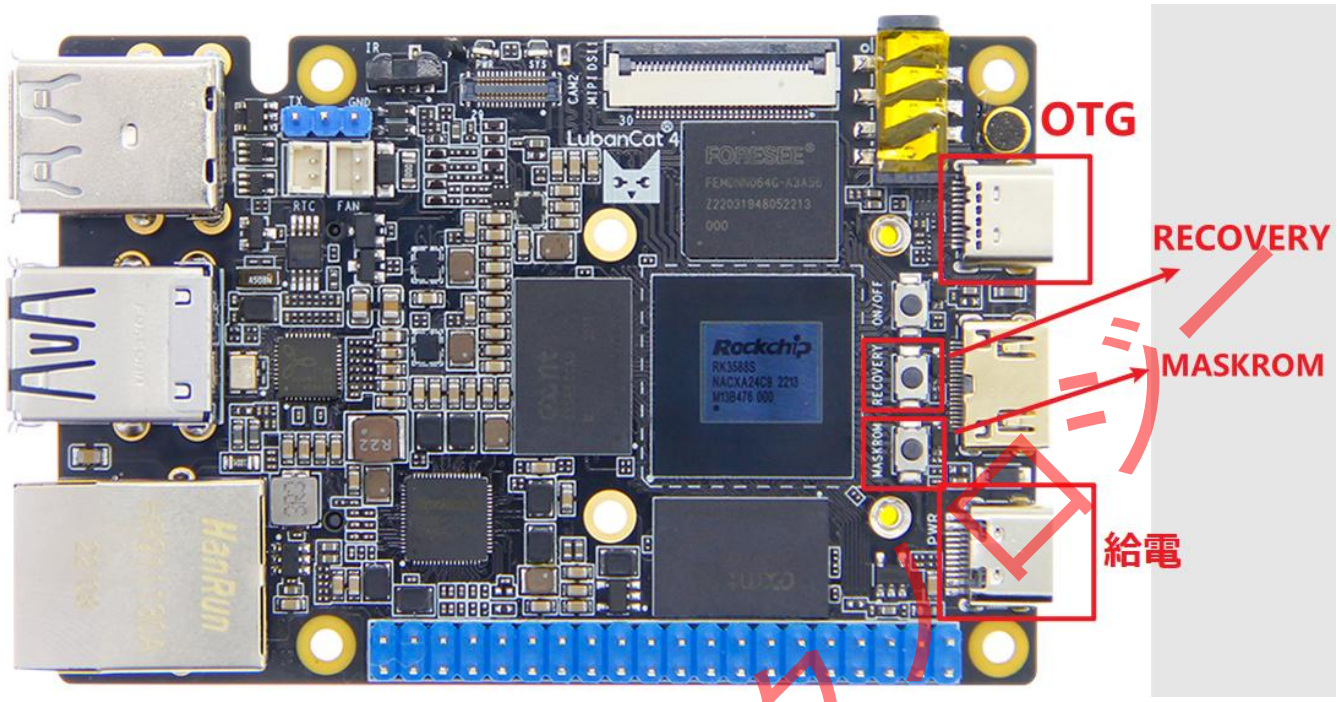
3.2.1.1.2 RKDevTool のインストール

Rockchip 専用 USB 書き込みツール、Windows で USB インターフェイスを使用してシステムイメージファイルをボードに書き込みすることができます。

圧縮されたパッケージを解凍すると、インストールせずに使用できます。RKDevTool.exe をダブルクリックしてアプリの画面を開きます。

アプリには、「Download Image」、「Upgrade Firmware」、「Advanced Function」3 つタブがあります。





3.2.1.2　Recovery モードによるイメージファイルの書き込み

このモードは、システムが正常に動作するボードに適しています。

RKDevTool 書き込みツールを開き、ボードを書き込みモードにします。

1.Type-C ケーブルを 2 本用意し、1 本はイメージファイル書き込み用、1 本は給電用

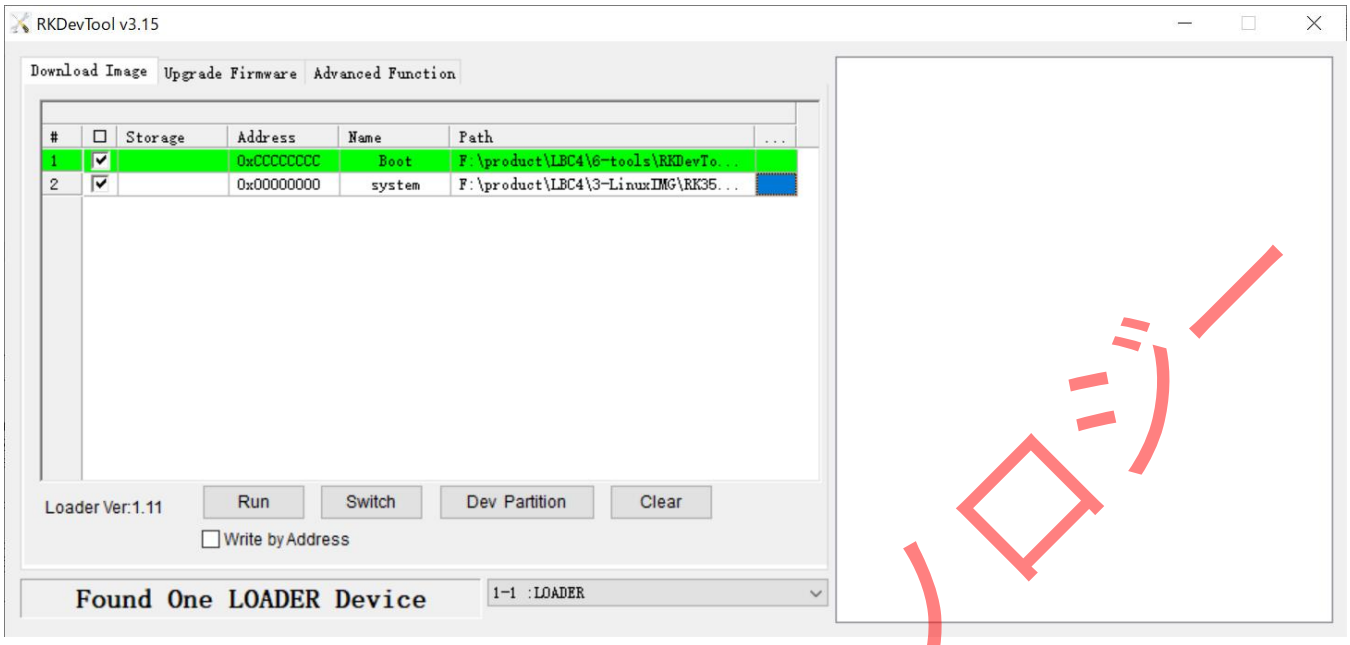
2.電源アダプタ、USB ケーブルなど、ボードに電力を供給する可能性のあるすべての配線を切断します

3.1 本の Type-C ケーブルを使って一端をボードの OTG 端子に接続し、もう一端をパソコンの usb 端子に接続して RKDevTool アプリを開く

4.RECOVERY キーを押したまま、もう 1 本の Type-C ケーブルを使用してボードに電源を入れます

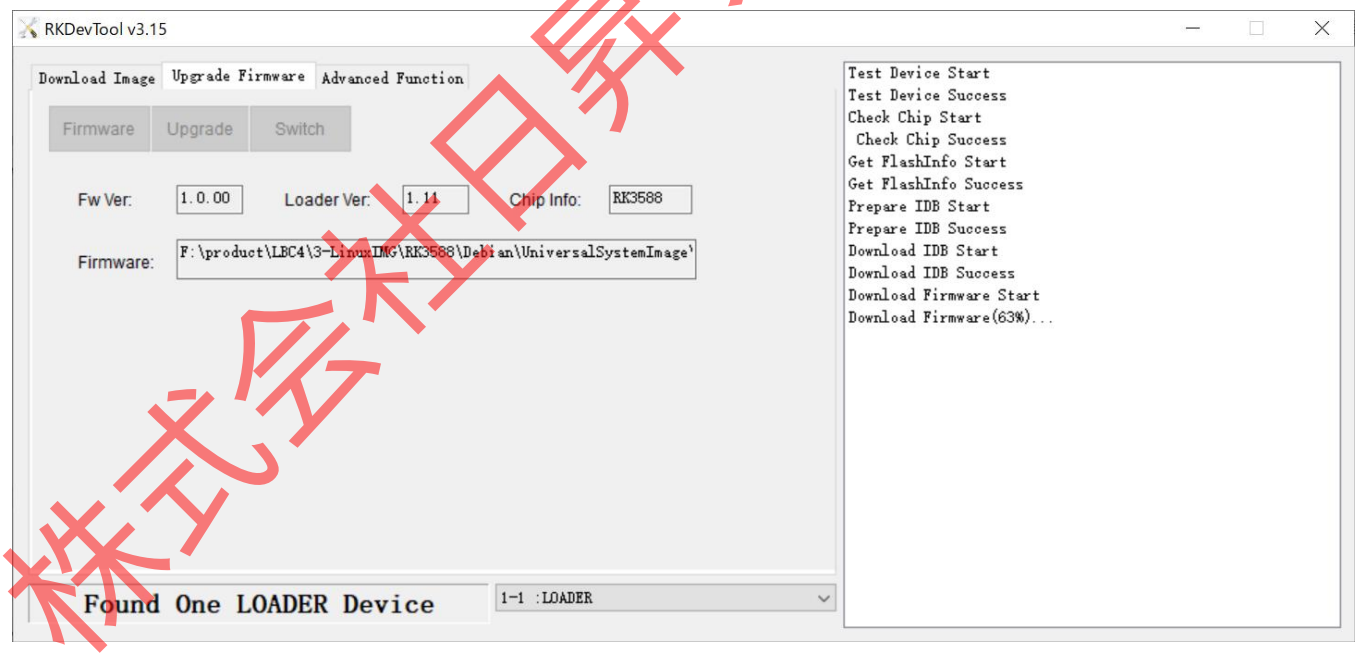
5.次の図に示すように、アプリ画面で LOADER デバイスを検出するように指示するのを待って、キーを放します

6.失敗した場合は、手順 2~5 を繰り返します。

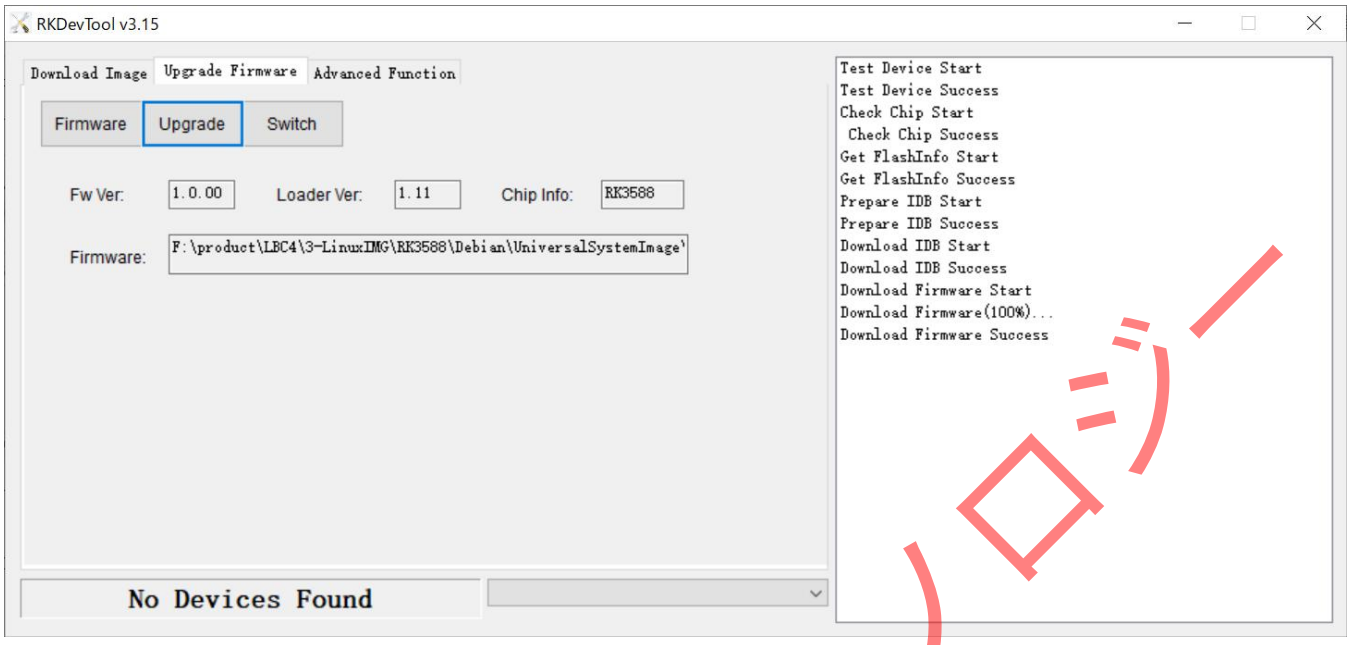


「Firmware」をクリックして書き込むイメージファイルを選択し、「Open」をクリックします

ファームウェアのロードが完了するまで待ち、「Upgrade」をクリックしてファームウェアの書き込みを開始します。



イメージファイルの書き込みに成功(次の図を参照)



3.2.1.3 MASKROM モードによるイメージファイルの書き込み

このモードは、ボードがシステムに書き込まれていないか、書き込まれたシステムが壊れて動作しない場合に適しています(すべての状況に対応)。

RKDevTool 書き込みツールを開き、ボードを書き込みモードにします。

1.Type-C ケーブルを 2 本用意し、1 本はイメージファイル書き込み用、1 本は給電用

2.電源アダプタ、USB ケーブルなど、ボードに電力を供給する可能性のあるすべての配線を切断します

3.1 本の Type-C ケーブルを使って一端をボードの OTG 端子に接続し、もう一端をパソコンの usb 端子に接続して RKDevTool アプリを開く

4.MASKROM キーを押したまま、もう 1 本の Type-C ケーブルを使用してボードに電源を入れます

5.次の図に示すように、アプリ画面で MASKROM デバイスを検出するように指示するまで待って、キーを放します

6.失敗した場合は、手順 2~5 を繰り返します。

「Firmware」をクリックすると、書き込みするイメージを選択できます。たとえば、update.img を選択して開くことができます

ファームウェアのロードが完了するまで待ち、「Upgrade」をクリックしてファームウェアの書き込みを開始します。

3.2.2 SD カードへの書き込み

イメージファイルを SD カードへの書き込みには専用の書き込みアプリと SD カードリーダーが必要

3.2.2.1 書き込みアプリのインストール

SD カードへのイメージファイルの書き込みには、書き込みツール SD_Firmware_Tool.exe が必要

圧縮されたパッケージ SDDiskTool を解凍してインストールせずに使用する。SD_Firmware_Tool.exe をダブルクリックしてアプリ画面を開きます。

3.2.2.2 イメージファイルの書き込み

SD_Firmware_Tool.exe を開き、SD カードを挿入します。

最初に書き込先の SD カードを選択して、次に機能モードを「SD Boot」を選択して、次に書き込むイメージファイルを選択して、最後に「Create」クリックして SD カードへの書き込み開始します。

SD カードの書き込みが完了するまで待ちます。イメージファイルが大きいほど、書き込みにかかる時間が長くなります。

注意：作成開始をクリックすると一定の確率で書き込みできないと表示されます。エラー画面を閉じて再び作成手順を行います

書き込みに成功したら、次の図のように示します。

3.3 FAQs

Q1：MASKROM モードの書き込みと Recovery モードの書き込みの違いは何ですか。

A1：Recovery モードでの書き込みは、uboot の起動中にボードのピンが押されたことを検

出して書き込みモードに入ります。Recovery モードでの書き込みは、イメージファイルが完全なことを前提としています(Android モードで使用されることが多い)。MASKROM モードの書き込みは、チップが起動時にこのピンが押されたことを検出してから、書き込みモードに入り、MASKROM モードの書き込みはほとんどのシーンに適しています。

4　システム起動とシステム登録

ボードを最初に実行するには、システムの書き込み、システムの起動、システムへのログインの 3 つのステップが必要です。

次はシステム起動とシステムログインについて説明します。

•システムを起動するには、ストレージデバイスを適切に接続し、電源投入などの操作を実行する必要があります。詳細については、この文書の次の節を参照してください。

•システムへのログインは、シリアルまたは SSH を使用してログインできます。LCD を接続した場合は、ディスプレイに表示される画面から直接デスクトップシステムにアクセスできます。

4.1　ボードの起動方法

LubanCat-RK3588 ボードは、さまざまな起動方法に対応しており、主に eMMC と SD カードを使用して起動します。

オンボード eMMC を搭載していないボードでは、SD カードの起動方法のみがサポートされています。

eMMC が搭載され、TF カードスロットが残されているボードについては、eMMC と SD カードの 2 つの起動方式に対応している。

1.SD カードにイメージファイルがある場合は、sd カードを優先して起動する

2.SD カードがイメージファイルを書き込まれず、emmc がイメージファイル書き込まれている場合は emmc のイメージファイルを用いて起動する

3.sd カードと emmc のどちらもイメージファイルを書き込まれていない場合は起動しない

4.2　起動時の注意

emmc または SD カードは Linux システムのイメージファイルを書き込み、ボードは電源を入れると自動的に実行されます。

ボードを起動する際の注意事項は次のとおりです：

1.ボード上の周辺機器を正しく接続して、ネジを使用してしっかりと固定して、特にホット

プラグをサポートしていないデバイス（mipi ディスプレイ、mipi カメラ、pcie-wi ネットワークカード、pcie-4G ネットワークカード、ハードディスクなど）、もし電源オンした状況で抜き差しすると、デバイスを壊す恐れがあって、デバイスを正常に動作させることができません。

2.ディスプレイの使用に注意すること：mipi ディスプレイはホットプラグをサポートしていない、HDMI インタフェースは 2 つの形態があり、1 つは標準の HDMI インタフェース、もう 1 つは micro-HDMI、2 つのインタフェースは直接接続できなくて、変換ケーブルを使用する必要があります。VGA、DP ポートのディスプレイの場合も、信号変換ケーブルを使うことで表示することもできます。

3.ボードはマウスとキーボードをサポートしており、USB インターフェイスを介してボードに接続する必要があります。

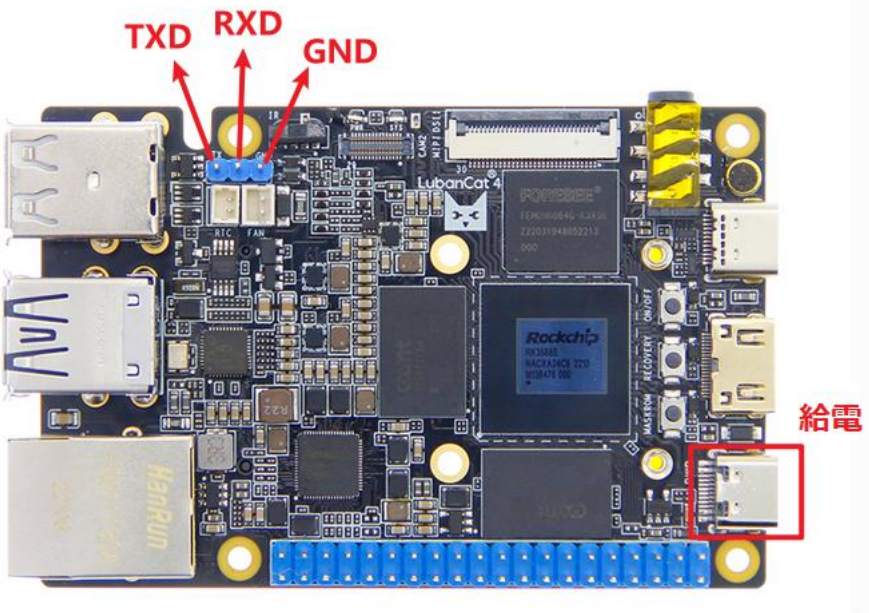
4.電源アダプタを使用してボードに電力を供給します。注意：表示されている電源の仕様は最小仕様です。周辺機器を多数使用する場合は、同等の電圧で大電力の電源を使用してください。

注：書き込みが完了した後、イメージファイルが最初に起動する時、ボードの設定を行います。電源投入から 1、2 分後に自動的に再起動して、システムが正常に動作するようになります。

4.3　シリアルポート端末ログイン

4.3.1　シリアルポート接続

デュポンケーブルで接続して使うことができます



リスト 1：ピン接続

```
#線対線
LubanCat4 ボード------USB から UART への変換モジュール
GND--------GND
TXD--------RXD
RXD--------TXD
```

4.3.2 ソフトウェアの準備

MobaXterm のインストールと使用

・ダウンロード先：https://mobaxterm.mobatek.net/download.html

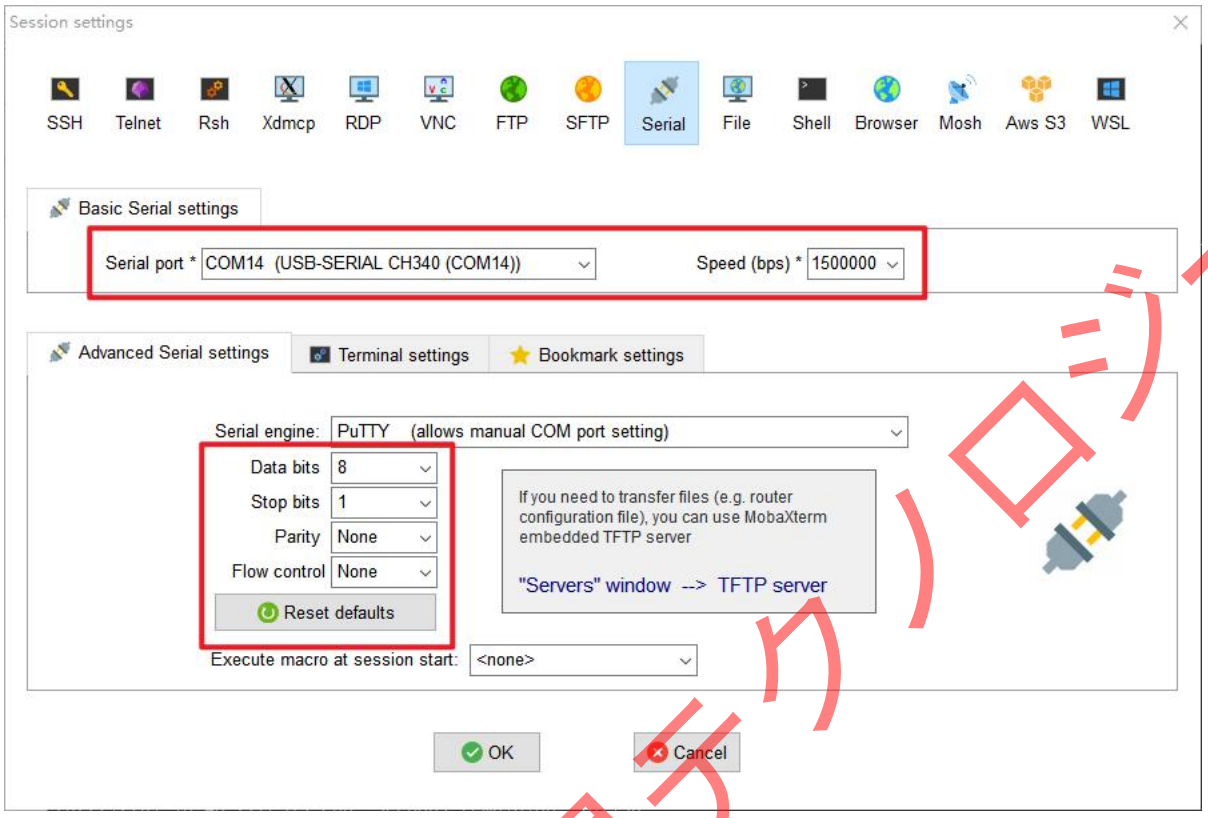
PC とシリアル・モジュールを繋いで、PC のデバイス・マネージャを開いてポート名を確認します。

自分のポートが COMxx を使用していることがわかります(ない場合はシリアルポートに対応するドライバを自分でインストールする必要があります)。そして MobaXterm アプリを開き、sessions アイコンをクリックすると session setting がポップアップし、Serial を選択します。

次の図に示すように、正しいシリアルポートを選択し、ボーレートを 1500000 に設定し、フロー制御をオフにします。

「OK」をクリックすると通信が始まりますが、まだ電源が入っていないので反応がありま

せん。



4.3.3 電源

表 1：ボード給電

電源タイプ	ボードモデル
Type-C（5V@2A 以上）	LubanCat4

注：LubanCat4 が正常に動作するには 5V@2A 以上の電力が必要です。出来れば 5V@4A がお勧めです。

4.3.4 電源オン

注：書き込み後の最初の起動では再起動が行われます。これは正常な現象です。ボードは設定を行っています。

電源を入れるとたくさんの情報が表示されます。これは正常です。システムが動作していることを示すものです。

```
[    1.516057] usb 5-1: New USB device found, idVendor=09da, idProduct=c10a, bcdDevice=94.06
[    1.516109] usb 5-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[    1.516123] usb 5-1: Product: USB Mouse
[    1.516135] usb 5-1: Manufacturer: A4Tech
[    1.537750] input: A4Tech USB Mouse Mouse as /devices/platform/usbhost/fd000000.dwc3/xhci-hcd.0.auto/usb5/5-1/5-1:1.0/0003:09DA:C10A.0001/input/input5
[    1.545495] devfreq fde60000.gpu: Couldn't update frequency transition information.
[    1.594654] input: A4Tech USB Mouse as /devices/platform/usbhost/fd000000.dwc3/xhci-hcd.0.auto/usb5/5-1/5-1:1.0/0003:09DA:C10A.0001/input/input6
[    1.596770] hid-generic 0003:09DA:C10A.0001: input,hiddev96,hidraw0: USB HID v1.10 Mouse [A4Tech USB Mouse] on usb-xhci-hcd.0.auto-1/input0
[    1.598024] ALSA device list:
[    1.598053]   #0: rockchip,rk809-codec
[    1.611157] EXT4-fs (mmcblk0p3): mounted filesystem with ordered data mode. Opts: (null)
[    1.611318] VFS: Mounted root (ext4 filesystem) on device 179:3.
[    1.612050] devtmpfs: mounted
[    1.616140] Freeing unused kernel memory: 1536K
[    1.637525] Run /sbin/init as init process
[    1.844211] systemd[1]: Failed to find module 'autofs4'

Welcome to Debian GNU/Linux 10 (buster)!

[    2.011430] systemd-fstab-generator[124]: Ignoring "noauto" for root device
[    2.144157] phy phy-fe8a0000.usb2-phy.1: charger = USB_DCP_CHARGER
[  OK  ] Listening on Journal Socket (/dev/log).
[  OK  ] Listening on Network Service Netlink Socket.
[  OK  ] Reached target Remote File Systems.
[  OK  ] Reached target Swap.
[  OK  ] Started Dispatch Password …ts to Console Directory Watch.
[  OK  ] Listening on Syslog Socket.
[  OK  ] Created slice system-getty.slice.
[  OK  ] Created slice system-systemd\x2dfsck.slice.
[  OK  ] Listening on initctl Compatibility Named Pipe.
[  OK  ] Listening on fsck to fsckd communication Socket.
[  OK  ] Started Forward Password R…uests to Wall Directory Watch.
[  OK  ] Reached target Local Encrypted Volumes.
[  OK  ] Reached target Remote Encrypted Volumes.
[  OK  ] Listening on udev Control Socket.
[  OK  ] Listening on udev Kernel Socket.
[  OK  ] Created slice system-serial\x2dgetty.slice.
[  OK  ] Listening on Journal Socket.
         Starting Nameserver information manager...
         Starting udev Coldplug all Devices...
         Starting Journal Service...
         Mounting POSIX Message Queue File System...
[    2.274194] rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x3
         Mounting /sys/kernel/debug...
```

しばらく待ってからログインできるようになりました（下の図のように）

```
[    6.697880] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[    6.698410] JL2101 Gigabit Ethernet stmmac-1:00: attached PHY driver [JL2101 Gigabit Ethernet] (mii_bus:phy_addr=stmmac-1:00, irq=POLL)
[    7.341107] rk-pcie 3c0800000.pcie: PCIe Linking... LTSSM is 0x0
[    7.341147] rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x3
[    7.997991] Freeing drm_logo memory: 4748K
[    8.221147] dwmac4: Master AXI performs any burst length
[    8.221242] rk_gmac-dwmac fe010000.ethernet eth1: No Safety Features support found
[    8.221277] rk_gmac-dwmac fe010000.ethernet eth1: IEEE 1588-2008 Advanced Timestamp supported
[    8.221827] rk_gmac-dwmac fe010000.ethernet eth1: registered PTP clock
[    8.222687] IPv6: ADDRCONF(NETDEV_UP): eth1: link is not ready
[  OK  ] Created slice User Slice of UID 1000.
         Starting User Runtime Directory /run/user/1000...
[  OK  ] Started User Runtime Directory /run/user/1000.
         Starting User Manager for UID 1000...
[    8.354395] rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x3
[    8.354398] rk-pcie 3c0800000.pcie: PCIe Linking... LTSSM is 0x0
[  OK  ] Started User Manager for UID 1000.
[  OK  ] Started Session c1 of user cat.
[    9.367877] rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x3
[    9.367931] rk-pcie 3c0800000.pcie: PCIe Linking... LTSSM is 0x0
[    9.842363] ttyFIQ ttyFIQ0: tty_port_close_start: tty->count = 1 port count = 2
[   10.381189] rk-pcie 3c0000000.pcie: PCIe Linking... LTSSM is 0x3
[   10.381192] rk-pcie 3c0800000.pcie: PCIe Linking... LTSSM is 0x0

Debian GNU/Linux 10 lubancat ttyFIQ0

[username:password] root:root cat:temppwd

Modify information : /etc/issue

lubancat login: [   11.394632] rk-pcie 3c0000000.pcie: PCIe Link Fail
[   11.394669] rk-pcie 3c0000000.pcie: failed to initialize host
[   11.394830] rk-pcie 3c0800000.pcie: PCIe Link Fail
[   11.394842] rk-pcie 3c0800000.pcie: failed to initialize host
[   12.275452] rk_gmac-dwmac fe010000.ethernet eth1: Link is Up - 1Gbps/Full - flow control off
[   12.275556] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

注：図に表示された文字を確認して、ログイン出来ます。時にはいくつかの誤りの情報があるかもしれないが、これは使用に影響しません。

ログインするにはユーザー名とパスワードが必要です。

ユーザータイプ----ユーザー名----パスワード
スーパーユーザー----root--------root
一般ユーザー--------cat---------temppwd

ユーザー名（大文字と小文字を区別）を入力した後にパスワードを入力します（パスワードを入力する際提示文書は表示されませんので、入力方法や大文字に注意が必要）

ユーザー名とパスワードが正しければ端末システムに入る

ユーザー名を変更するには usermod コマンドを使用し、パスワードを変更するには passwd コマンドを使用します。

4.4　自動ログイン設定

4.4.1　デスクトップ自動ログイン(cat ユーザー)

・Debian 系設定ファイル　/etc/gdm3/daemon.conf

・Ubuntu 系設定ファイル　/etc/gdm3/custom.conf

インストールされているシステムより設定ファイルを変更します

```
#自動ログインの設定
AutomaticLoginEnable=true
#自動ログインするユーザーの設定
AutomaticLogin=cat
```

サービスの再起動またはボードの再起動後有効になる

```
sudo systemctl restart gdm3.service
```

4.4.2　デスクトップ自動ログイン(root)

・Debian 系設定ファイル　/etc/gdm3/daemon.conf

・Ubuntu 系設定ファイル　/etc/gdm3/custom.conf

インストールされているシステムより設定ファイルを変更します

```
#自動ログインの設定
AutomaticLoginEnable=true

#自動ログインするユーザーの設定
AutomaticLogin=root
```

・/etc/pam.d/gdm-autologin を開く

•次の行を見つける

```
auth required pam_succeed_if.so user！=root quiet_success
```

この行の前に「#」を追加します。

```
#auth required pam_succeed_if.so user！=root quiet_success
```

サービスの再起動またはボードの再起動後有効になる

```
sudo systemctl restart gdm3.service
```

4.4.3　デスクトップ自動ログインをオフにする

・Debian 系設定ファイル　/etc/gdm3/daemon.conf
・Ubuntu 系設定ファイル　/etc/gdm3/custom.conf
インストールされているシステムより設定ファイルを変更します

```
#自動ログインをオフに設定する場合は、ファイルの内容を変更します
AutomaticLoginEnable=false
```

サービスの再起動またはボード再起動後有効になる

```
sudo systemctl restart gdm3.service
```

4.4.4　シリアル端末の自動ログインをオンにする

シリアル端末への自動ログインを行うには、電源投入時に root ユーザーが自動ログインする場合の例として、/lib/systemd/system/serial-getty@.service ファイルを変更するだけです。

```
#シリアルポートのサービスファイルを開く
vi/lib/systemd/system/serial-getty¥e.service
```

次の内容を探します。

```
#ExecStart=-/sbin/agetty--autologin root--noclear%I$TERM
ExecStart=-/sbin/agetty-o'-p--¥¥u'--keep-baud 115200,38400,9600%I<→$TERM
```

次のように変更します

```
ExecStart=-/sbin/agetty--autologin root--noclear%I$TERM
#ExecStart=-/sbin/agetty-o'-p--¥¥u'--keep-baud 115200,38400,9600%I<→$TERM
```

cat ユーザーとして自動的にログインするには、上記の「root」を「cat」に置き換えるだけでよい。

4.4.5　シリアル端末の自動ログインをオフにする

```
#シリアルポートのサービスファイルを開く
vi/lib/systemd/system/serial-getty¥e.service
```

次の内容を探します。

```
ExecStart=-/sbin/agetty--autologin root--noclear%I$TERM
#ExecStart=-/sbin/agetty-o'-p--¥¥u'--keep-baud 115200,38400,9600%I<→$TERM
```

次のように変更します

```
#ExecStart=-/sbin/agetty--autologin root--noclear%I$TERM
ExecStart=-/sbin/agetty-o'-p--¥¥u'--keep-baud 115200,38400,9600%I<→$TERM
```

変更が完了したら、再起動してください。

5　ネットワークの接続及び静的構成

LubanCat4 ボードにはイーサネットポート及び WIFI モジュール I/F が搭載されており、また 4G モジュール、PCIE からイーサネット変換モジュール、USB からイーサネット変換モジュールなどを接続することができます。このように、ボードを提供してネットに接続するだけではなく、組み合わせによって、様々なニーズに適応することができます。

5.1　ping コマンド

ping(Packet Internet Groper)は、パケットインターネットグローパーで、ネットワーク接続量をテストするためのプログラムです。これは、TCP/IP ネットワークにおいて、相手先ホストと通信できるか（導通）を確認するコマンドです。

ping の原理：指定された IP アドレスに一定の長さのパケットを送信し、指定された IP アドレスが存在すれば、同じ大きさのパケットを返すことが約束されているが、当然、指定された時間内に返されなければ「タイムアウト」となり、指定された IP アドレスが存在しないとみなされます。

5.1.1　ローカルエリア通信

一つの LAN（Local Area Network）にはゲートウェイが必須であり、ゲートウェイを ping することによって自分の ip が使えるかどうかを確認することができて、もしゲートウェイさえ ping が通じないならば、その LAN は接続されていないことを証明することができます。

#ping コマンド
sudo ping ip アドレス

ネットワーク接続に成功すると、次のような一連のデータが表示されます：

```
root@lubancat:~# ping 192.168.103.254
PING 192.168.103.254 (192.168.103.254): 56 data bytes
64 bytes from 192.168.103.254: icmp_seq=0 ttl=64 time=0.635 ms
64 bytes from 192.168.103.254: icmp_seq=1 ttl=64 time=0.724 ms
64 bytes from 192.168.103.254: icmp_seq=2 ttl=64 time=0.536 ms
64 bytes from 192.168.103.254: icmp_seq=3 ttl=64 time=0.609 ms
64 bytes from 192.168.103.254: icmp_seq=4 ttl=64 time=0.746 ms
^C--- 192.168.103.254 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.536/0.650/0.746/0.077 ms
root@lubancat:~#
```

ネットワークに接続できなかった場合はエラーを報告します（下の図のように）。

```
root@lubancat:~# ping 192.168.103.254
PING 192.168.103.254 (192.168.103.254): 56 data bytes
ping: sending packet: Network is unreachable
root@lubancat:~#
```

5.1.2 インターネットへの接続

ボードがインターネットに接続されているかどうか確認する場合は、サイトに ping することで確認できます。

#ping コマンド
sudo ping xxx.com

- baidu.com で例に挙げます。
- ネットワーク接続に成功すると、次のような一連のデータが表示されます。

```
root@lubancat:~# ping baidu.com
PING baidu.com (39.156.66.10) 56(84) bytes of data.
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=1 ttl=49 time=40.3 ms
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=2 ttl=49 time=40.2 ms
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=3 ttl=49 time=39.4 ms
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=4 ttl=49 time=39.5 ms
64 bytes from 39.156.66.10 (39.156.66.10): icmp_seq=5 ttl=49 time=40.8 ms
^C
--- baidu.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 39.418/40.049/40.846/0.537 ms
root@lubancat:~#
```

- ネットワークに接続できなかった場合下の図のように表示されます。

```
root@lubancat:~# ping baidu.com
ping: unknown host
root@lubancat:~#
root@lubancat:~# ping baidu.com
ping: baidu.com: Temporary failure in name resolution
root@lubancat:~#
```

5.2　ネットワーク接続

インターネットに接続したり、LAN を構成したりするには、1 つの前提条件を満たす必要があります-デバイスは ip を取得する必要があります。ip はデバイスの名前と理解できます。LAN 内のデバイスは、LAN 内で ip を介して通信することができます。

#ip を確認する ifconfig

```
cat@lubancat:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.172  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::760:b10d:ac3c:6442  prefixlen 64  scopeid 0x20<link>
        ether c2:1d:00:8d:69:90  txqueuelen 1000  (Ethernet)
        RX packets 6848  bytes 529934 (517.5 KiB)
        RX errors 0  dropped 289  overruns 0  frame 0
        TX packets 8239  bytes 1379741 (1.3 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

cat@lubancat:~$
```

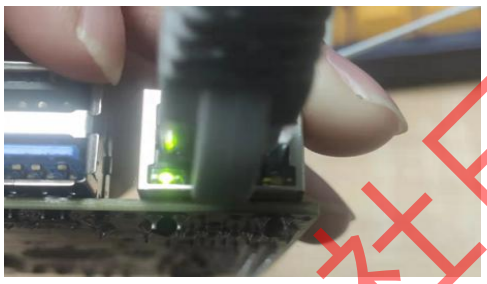
5.2.1　ネットワークポート接続

LBC4 のネットワークインタフェースは以下のとおりである。

ネットワークに接続するには、ネットワークケーブルをネットワークポートに差し込む必要があります。ボードは 10/100/1000M アダプティブをサポートしています。自分のネットワークに合わせて適切なネットワークケーブルを選択することができます。

カテゴリー 5 ケーブル ----- 100M
カテゴリー 5e ケーブル線 ----- 1000M （お勧め）
カテゴリー 6 ケーブル ----- 1000M （お勧め）
カテゴリー 6e ケーブル ----- 1000M （お勧め）

ギガビットに適したネットワークケーブルを使えば、接続に成功すると、ネットワークインタフェースのライトが点灯する。



5.2.2 ワイヤレスネットワーク接続

・USB ネットワークアダプタ対応

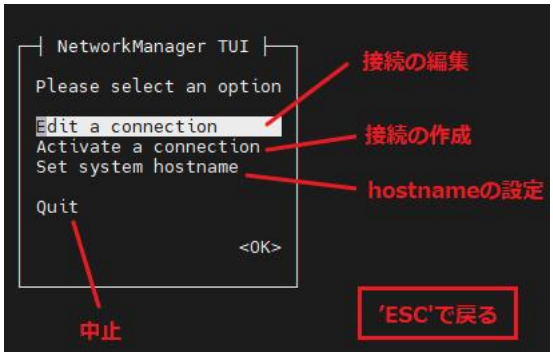
・PCIE ネットワークアダプタ対応

5.2.2.1 コマンドライングラフィックよる接続

・グラフィック構成へのアクセス

```
sudo nmtui
```

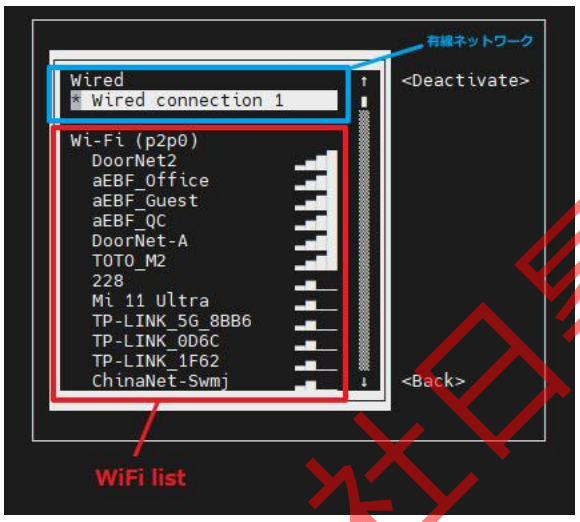
・キーボードの矢印キーで Active a connection に移動して Enter キーを押して wifi 設定に移行



・次に矢印キーを接続したい wifi に移動して Enter キーを押す

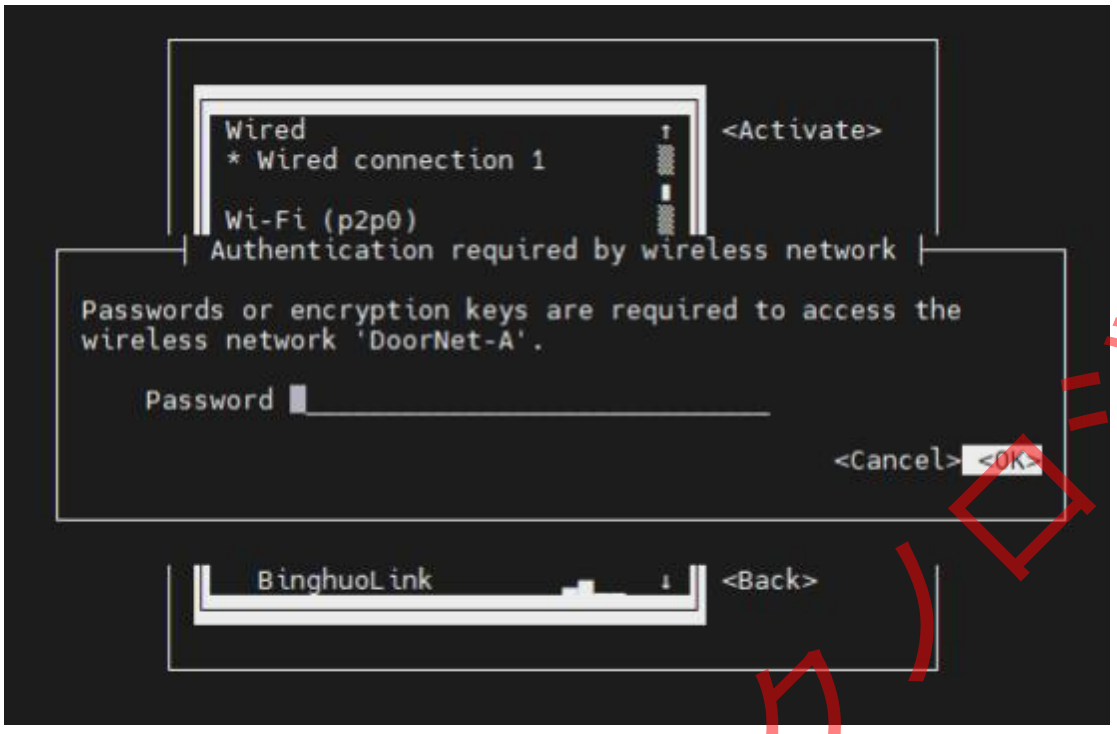
・接続されていないパスワード付きホットスポットに接続している場合はパスワード入力画面へ

・パスワードがない、または接続されているホットスポットの場合は、ホットスポットに接続するか、接続を解除する

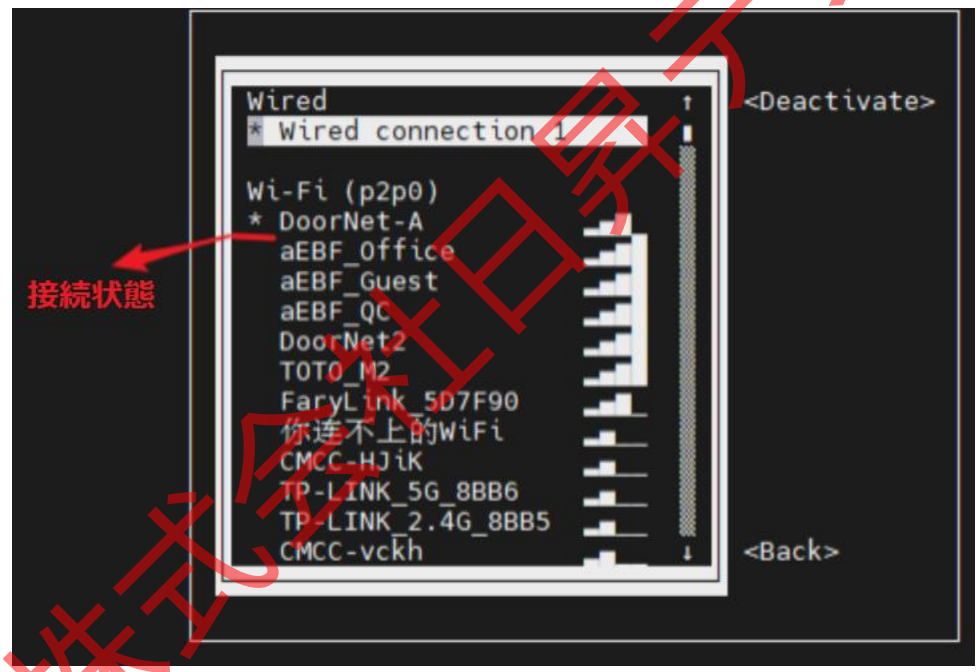


注解：一部のネットワークアダプタ(rtl8821cu など)を使用する場合、システムは 2 つのネットワークアクセスポイント p2p0 と wlan0 を生成する。Wifi はどちのノードで接続してもいいです。p2p0 を使用する場合は設定に便利し、wlan0 を使用する場合は比較的に多くのユーザーの習慣に合うことができます。

パスワードが設定されたホットスポットに初めて接続した場合は、次のようになります



• パスワードを入力すると、wifi が接続されていることがわかる

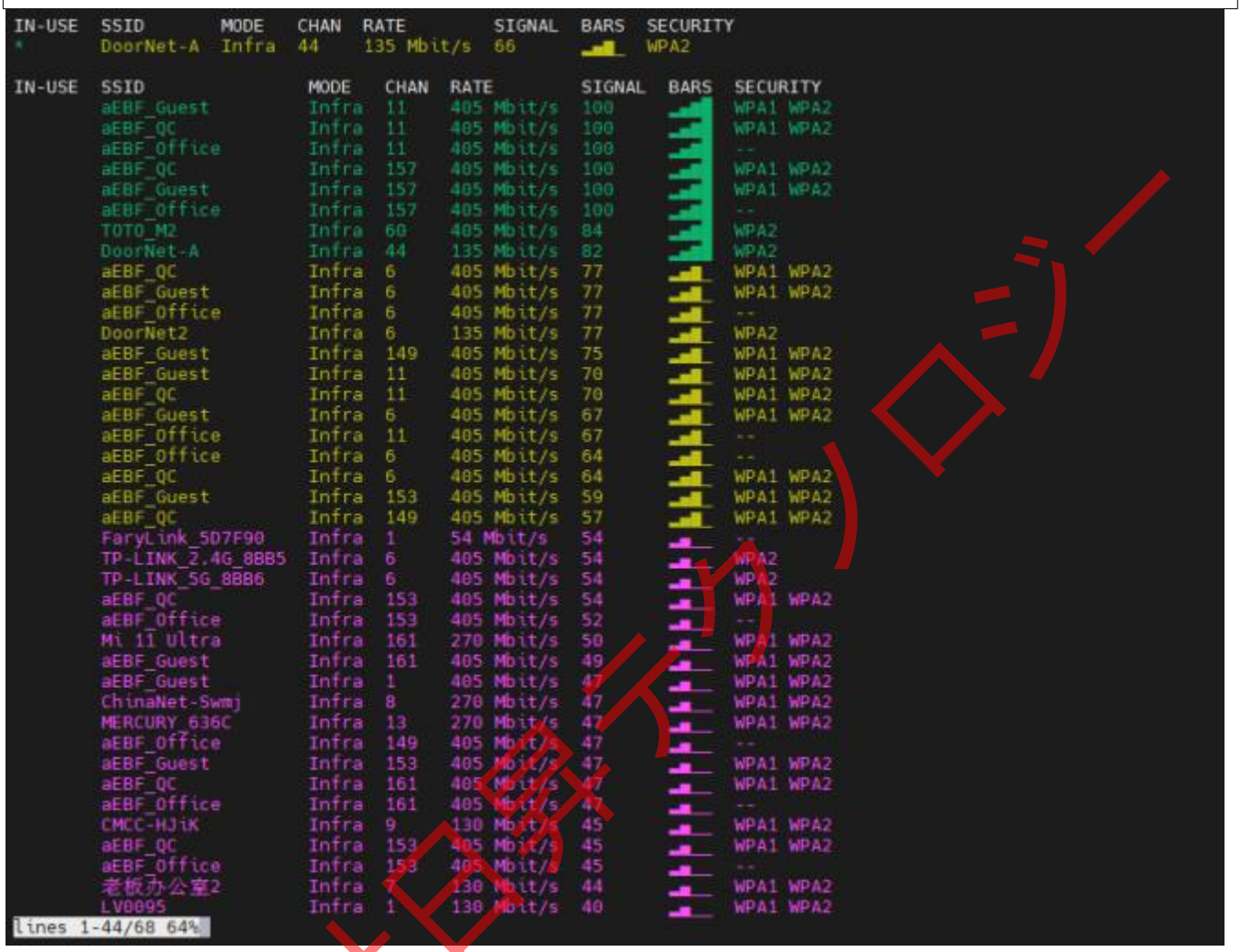


wifi を切断したい場合は、接続した状態で Enter キーを押すと切断できます

5.2.2.2 コマンドラインよる接続

・wifi リストを表示する

```
nmcli dev wifi list
```



矢印キーを押して wifi リストを検索し、q を押して終了

・wifi 接続

使用例：

ホットスポット：PPP

パスワード：00000000

インタフェース wlan0

#初めての接続

```
sudo nmcli dev wifi connect PPP password '00000000' ifname wlan0
```

#最初の接続が成功した後に再接続または WiFi 切り替え後

```
sudo nmcli dev wifi connect PPP
```

5.2.3　USB 共有ネットワーク

5.2.3.1 携帯電話の USB 共有ネットワーク

USB を使用してネットワークを共有する手順

注意：この機能を使用する前に、携帯電話が USB ネットワーク共有に対応しているかどうかを確認する必要があります

操作手順：

1.携帯電話はインターネット接続が必要

2.携帯電話とボードを USB でつなぐ

3.携帯電話で USB 共有の機能をオンにする

以上の手順でボードは携帯電話のネットワークを共有することができる。

5.2.3.2　コンピュータの USB 共有ネットワーク-Windows

注意：この機能をオンにするには、ボードの OTG インターフェイスを使用する必要があります。コンピュータは、ボードの正常な動作を維持するために十分な電流を提供する必要です。直接コンピュータの USB でボードに電源を供給することを試すことができますが、もしボードが再起動の現象が発生した場合は、一部の周辺機器を減らしてもう一度試してみましょう。もし正常に起動する場合は、この機能を使用することができます。

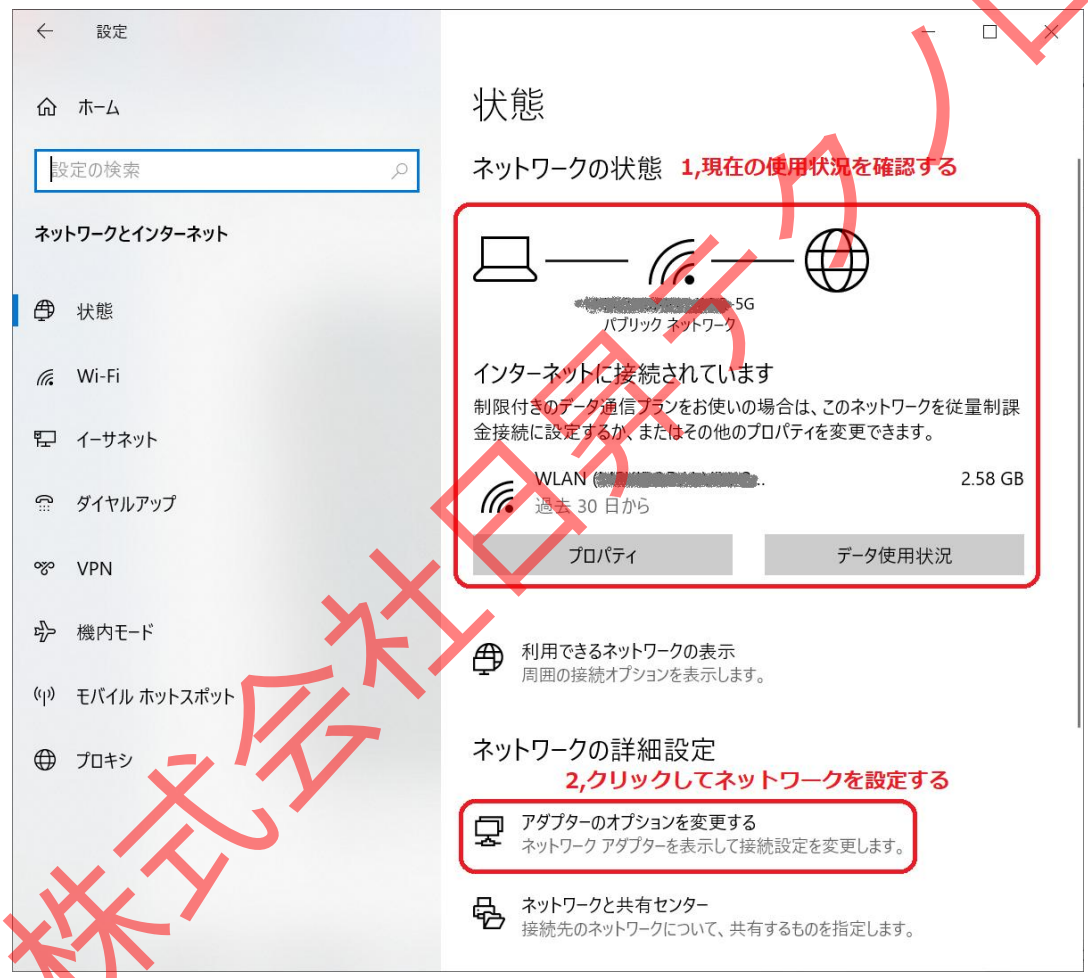
操作方法：

1.パソコンの USB ポートとボードの OTG ポート（一般的には給電ポート）をデータ伝送可能なケーブルで接続する

2.パソコンがネットワーク共有を開始する、具体的な手順は以下の通り

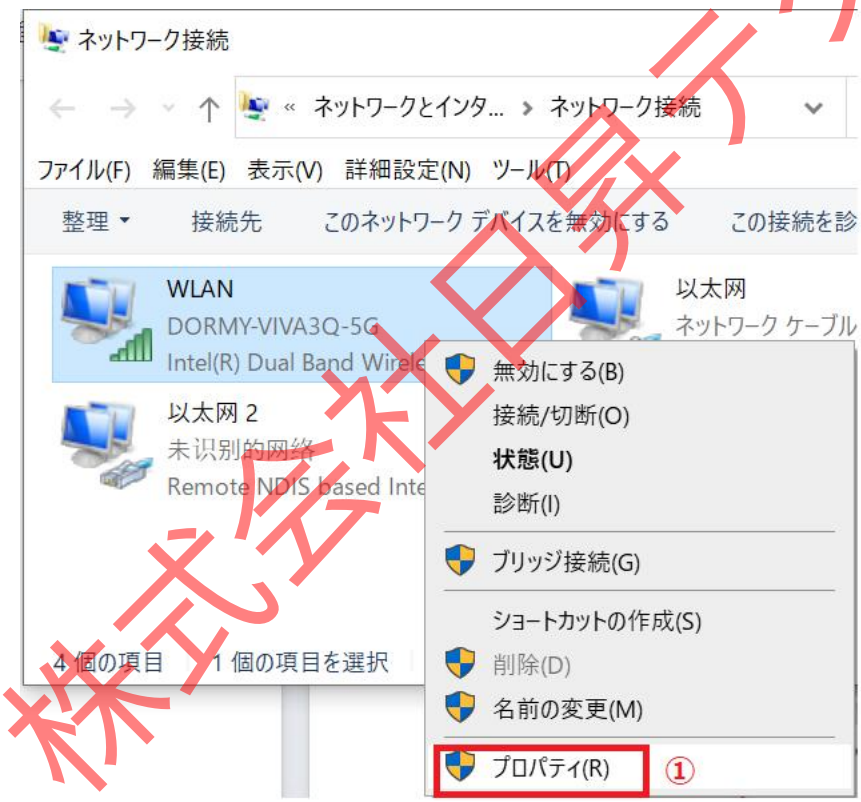
•ネットワーク設定を開く

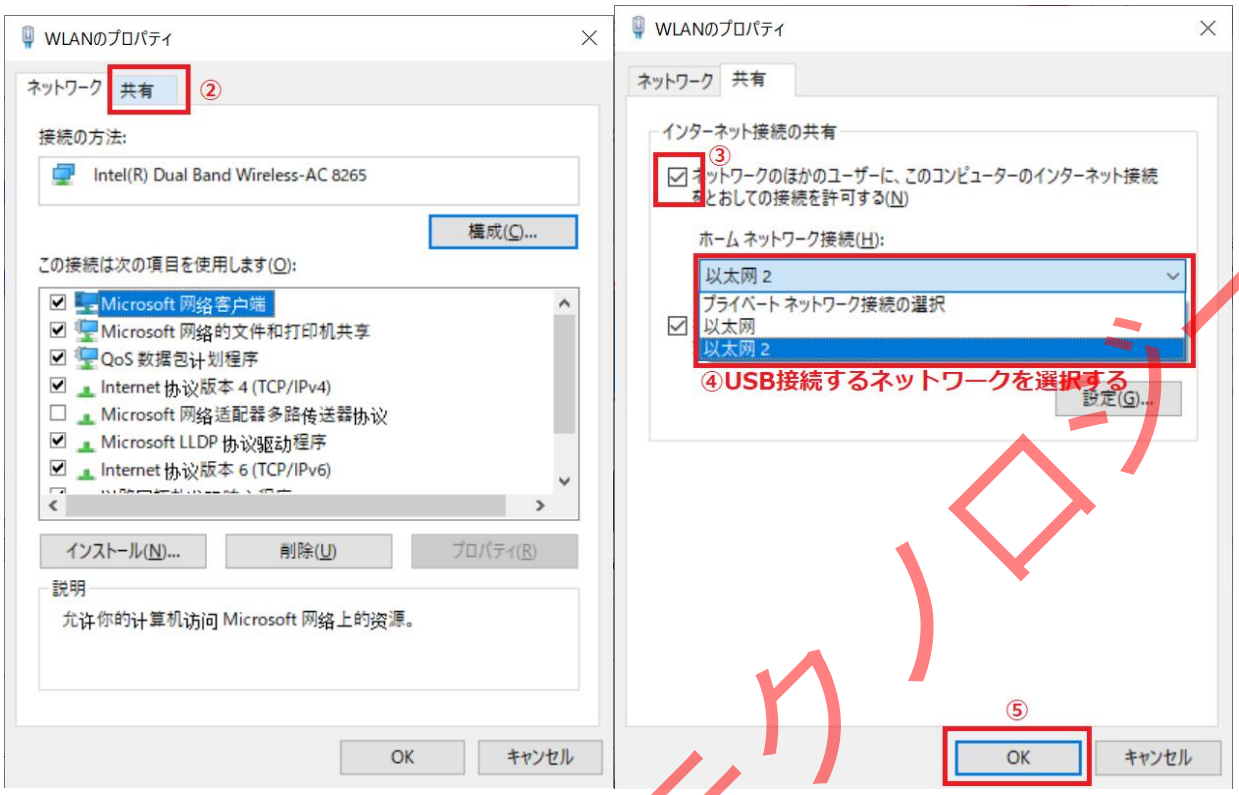
•現在のネットワーク状態の表示



•現在のすべてのネットワーク接続を表示

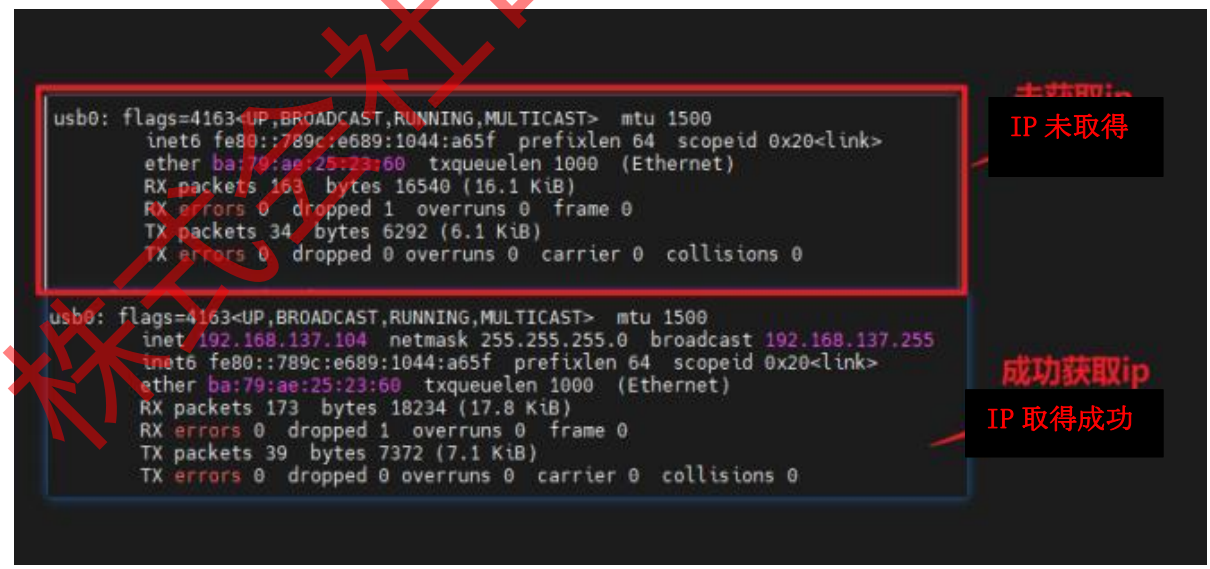
•ネットワーク共有の設定





・ip を自動的に取得するかどうかを確認する

```
# 数秒待つ
# ip を取得したかを確認
ifconfig
```



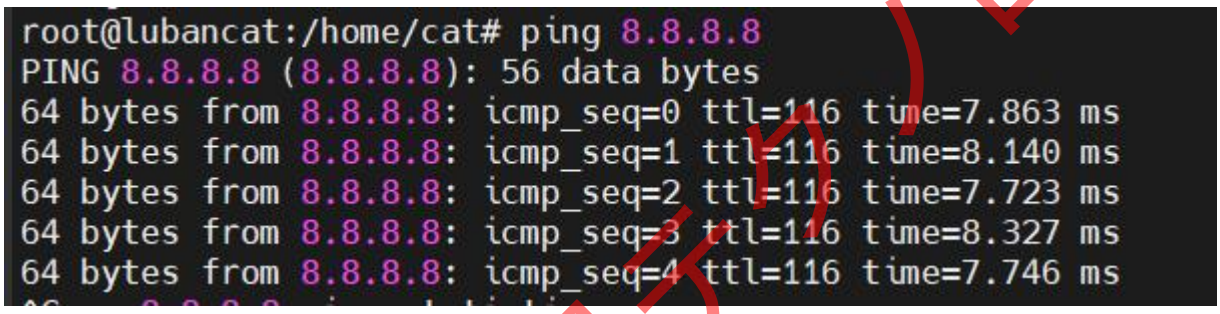
• パソコンでネットワーク共有をオンにしている場合、20 秒後に usb0 が ip を自動取得していない場合、手動で ip を取得する必要がある

```
#手動設定による ip の自動取得
dhclient usb0
```

•ip が正常に取得した後、下記のコマンドを使用して、ネットワークに正常に接続されているかどうかを確認できます。

```
ping 8888
```

成功した場合は次の図の様に表示する



5.2.3.3 コンピュータの USB 共有ネットワーク-Linux

注意：この機能をオンにするには、ボードの OTG インターフェイスを使用する必要があります。コンピュータは、ボードの正常な動作を維持するために十分な電流を提供する必要です。直接コンピュータの USB でボードに電源を供給することを試すことができますが、もしボードが再起動の現象が発生した場合は、一部の周辺機器を減らしてもう一度試してみましょう。もし正常に起動する場合は、この機能を使用することができます。

以下では Ubuntu20.04 を使ってデモを行っています。他の Linux ディストリビューションも同様の手順になります。

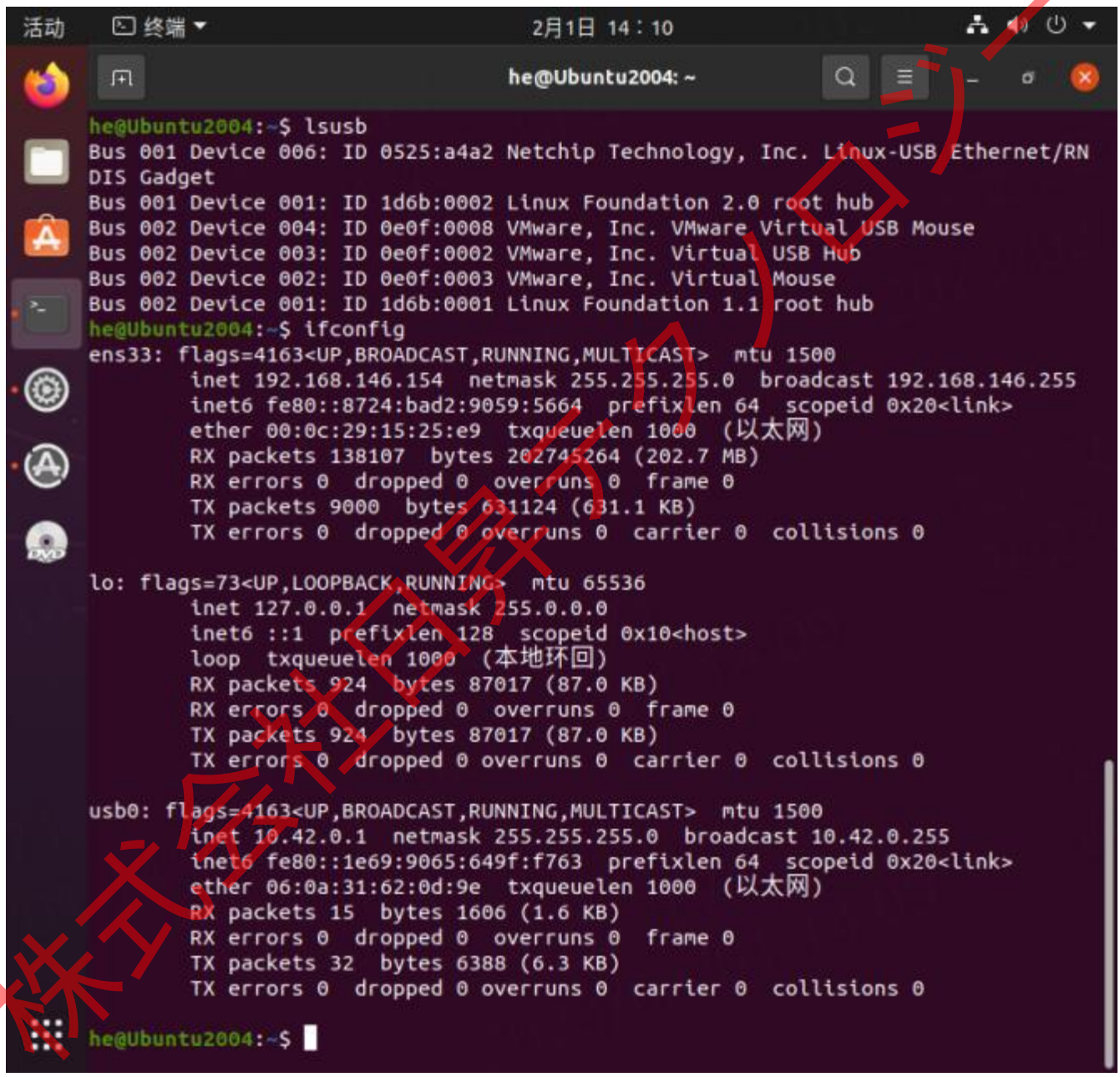
操作方法：

1.パソコンの USB ポートとボードの OTG ポート（一般的には給電ポート）をデータ伝送可能なケーブルで接続する

2.パソコンがネットワーク共有を開始する、具体的な方法は以下の通り

・ボードを OTG ポートを経由でパソコンに正しく接続

接続が完了すると、端末から lsusb コマンドを使用して Linux-USB Ethernet/RNDIS Gadget デバイスが検索され、ifconfig コマンドを使用して usb0 ネットワークアダプタが表示されます。



•ネットワーク設定を開き、新しいインタフェースを作成

端末で nm-connection-editor コマンドを使用してネットワークマネージャーを開き、Ethernet(Ethernet)接続を新規作成する。

• ### ネットワーク共有の設定

イーサネットのオプションタブの設定を変更し、デバイスを usb0、つまりボードに接続されているネットワークデバイスに設定します。•ネットワーク共有の設定

「Ethernet」タブの設定を変更し、デバイスを usb0 に設定します。ボードに接続されるネットワークデバイスである。

「IPv4 設定」タブを変更し、他のコンピュータと共有するようにメソッドを設定し、設定を保存します

・ip を自動的に取得するかどうかを確認する

ボードのターミナルで次のコマンドを実行します

```
# 数秒待つ
# ip を取得したかを確認
ifconfig
```

図に示すように、ボードが取得した ip は 10.42.0.12 である

```
usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.42.0.12  netmask 255.255.255.0  broadcast 10.42.0.255
        inet6 fe80::ad:e185:1404:5d76  prefixlen 64  scopeid 0x20<link>
        ether 42:ad:29:ac:e1:86  txqueuelen 1000  (Ethernet)
        RX packets 235  bytes 29283 (22.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36  bytes 5863 (5.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

・パソコンでネットワーク共有を開始してから 20 秒経過しても、USB0 が ip を自動的に取得しない場合は、USB を抜き差しするか、ip を手動で取得する必要がある

```
# ip を手動で取得
dhclient usb0
```

•ip が正常に取得されたら、下記のコマンドを使用して、ネットワークに正常に接続されているかどうかを確認できます

```
ping 8.8.8.8
```

成功した場合は次の図の様に表示する

```
root@lubancat:/home/cat# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=116 time=7.863 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=8.140 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=7.723 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=8.327 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=7.746 ms
```

5.3　静的ネットワーク構成

注：静的 ip の設定が正しくないと、ネットワークに正常に接続できない

注意：入門社の場合は、動的に取得した ip を静的に設定することをお勧めします

5.3.1 静的 ip がネットワークに接続できない問題のトラブルシューティング

この節は静的 ip を設定する時に、ネットに接続することができない問題を解決します。もし問題を解決することができないならば、コマンドライン出力の内容を google してみましょう。

静的 ip がネットワークに接続できない問題のトラブルシューティング
1.ネットワークに接続できない---->動的 ip 接続が正しく接続できるかどうかを確認する
可---->次へ、不可---->ネットワークリンクに問題がある
2.自分のゲートウェイが正しく設定されているかチェック------>動的接続時に自動的に設定されるゲートウェイを確認
同じ-->下の方へ、違う-->ゲートウェイを再設定してください
3.設定されている静的 ip アドレスが他のデバイスによって使用されているかを確認する---->ゲートウェイに ping を実行する
可-->次のページに移動します。いいえ-->固定 ip を変更するか、動的割り当て ip を

設定する

4.DNS サーバを設定---->114.114.114.114 または 8.8.8.8 に設定する

C:¥Users¥zhang¥OneDrive¥桌面¥翻译文档¥114.114.114.114C:¥Users¥zhang¥OneDrive¥桌面¥翻译文档¥8.8.8.8

可---->終了、不可---->debug 情報を元に geogle で捜索

5.3.2 nmtui

```
root@lubancat:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.142  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::e523:c785:434c:8695  prefixlen 64  scopeid 0x20<link>
        ether 52:43:ff:4f:80:e8  txqueuelen 1000  (Ethernet)
        RX packets 6932  bytes 622654 (608.0 KiB)
        RX errors 0  dropped 464  overruns 0  frame 0
        TX packets 965  bytes 81077 (79.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 38
```

・静的アドレスを設定する前の ip アドレス

静的アドレスの設定を成功させたい場合は、設定したい ip アドレスが他のデバイスによって占有されているかどうかに留意する必要があります。ping で静的アドレスを確認する。もし、戻り値があれば、ip が他のデバイスによって占有されている事が証明されます。

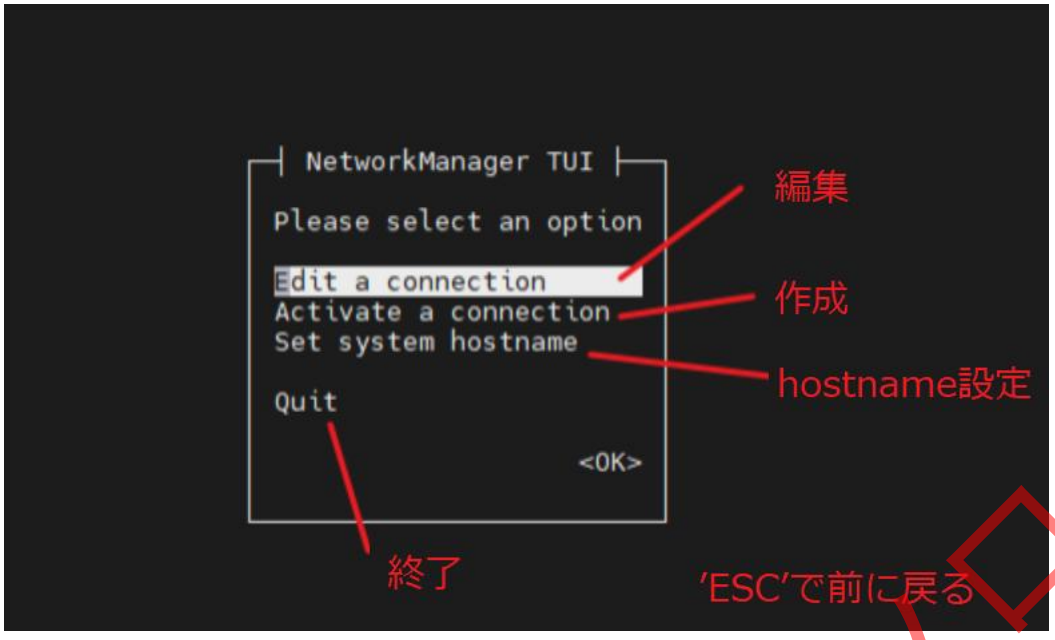
注意：初心者はネットワーク設定を把握してない場合は静的アドレスを動的接続時に取得した ip アドレスに設定することをお勧めします。

以下に詳細な手順を示す

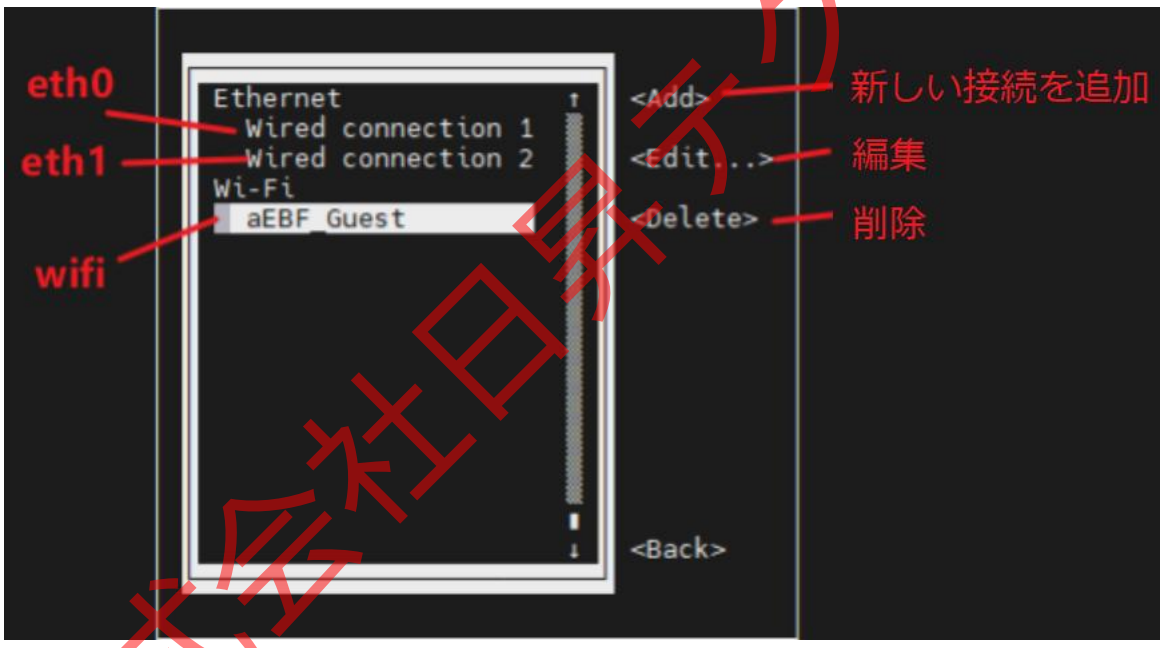
・グラフィック構成へアクセス

```
nmtui
```

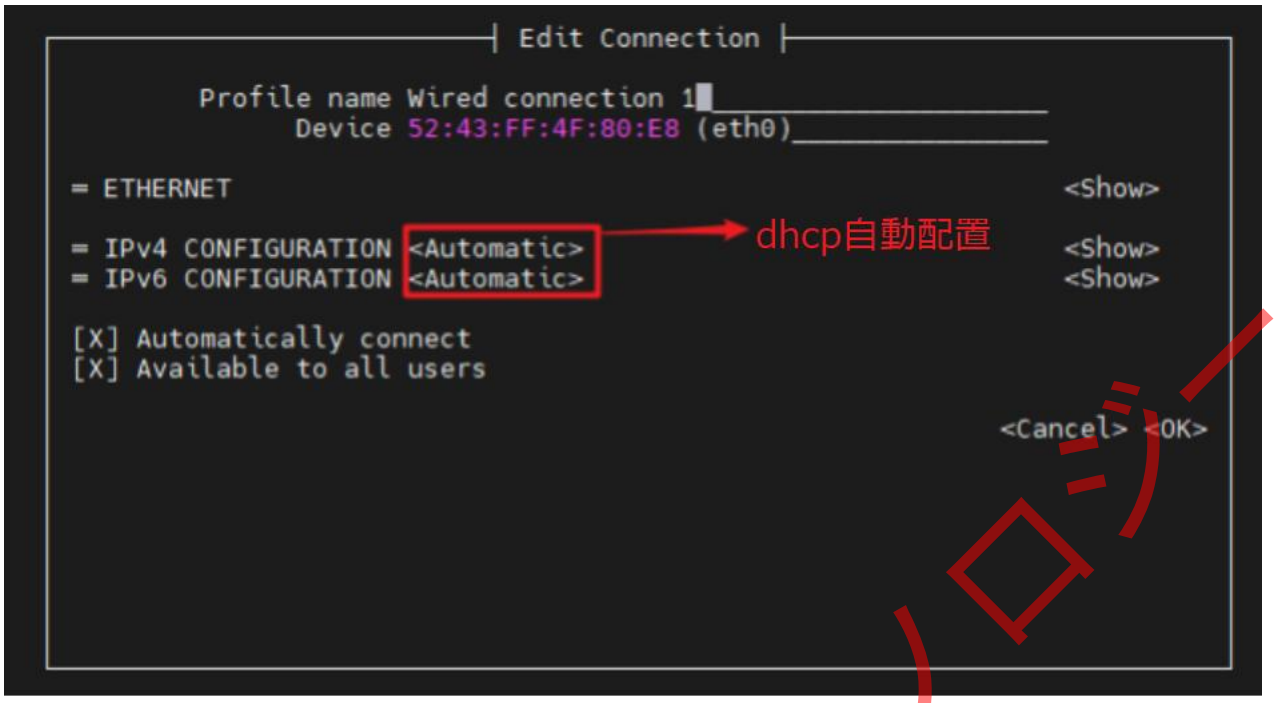
・キーボードの移動矢印キーを Edit a connection に移動 Enter キーを押して wifi 設定に移行



・編集するネットワークを選択してください。ここでは eth0 の例で示す



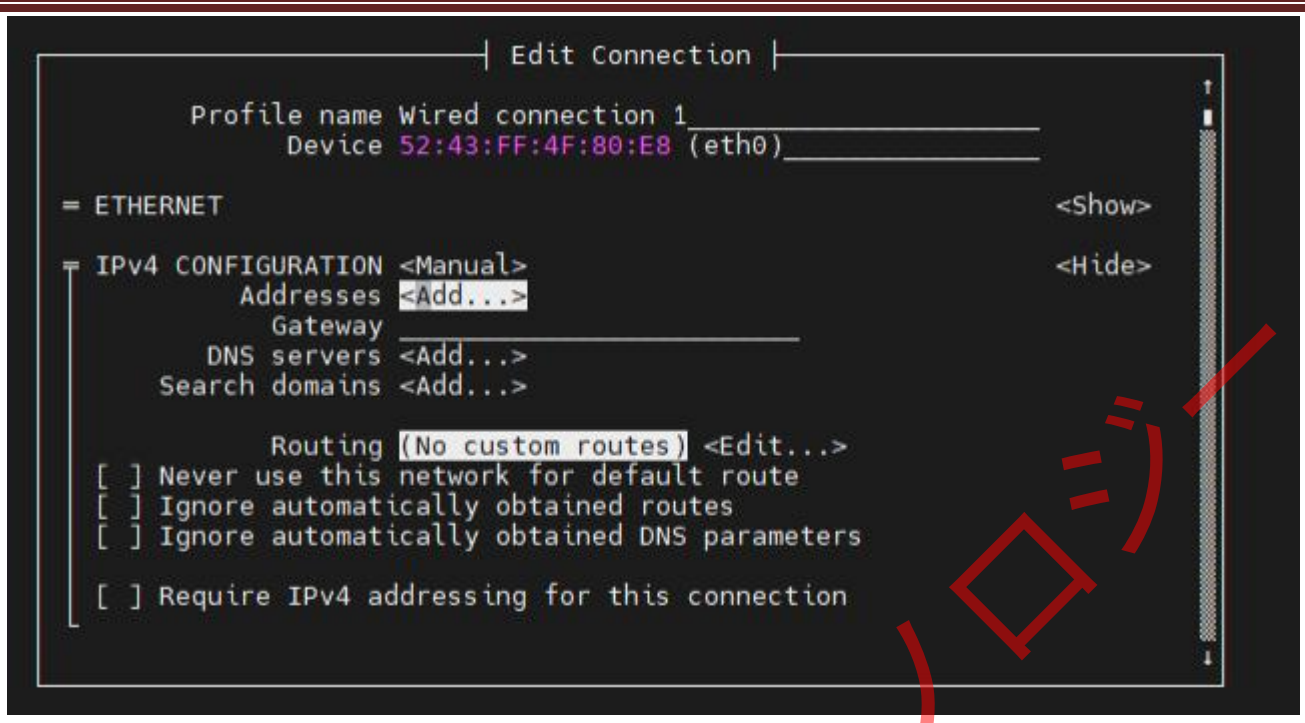
・ネットワークデフォルト設定が表示する



•次の手順で静的 ip を配置する

•IPV4 CONFIGURATION を Manual に設定する必要がある

・次にカーソルを show に移動して enter を押して詳細設定に進む



・カーソルを address<Add...>に移動して ip アドレスを追加

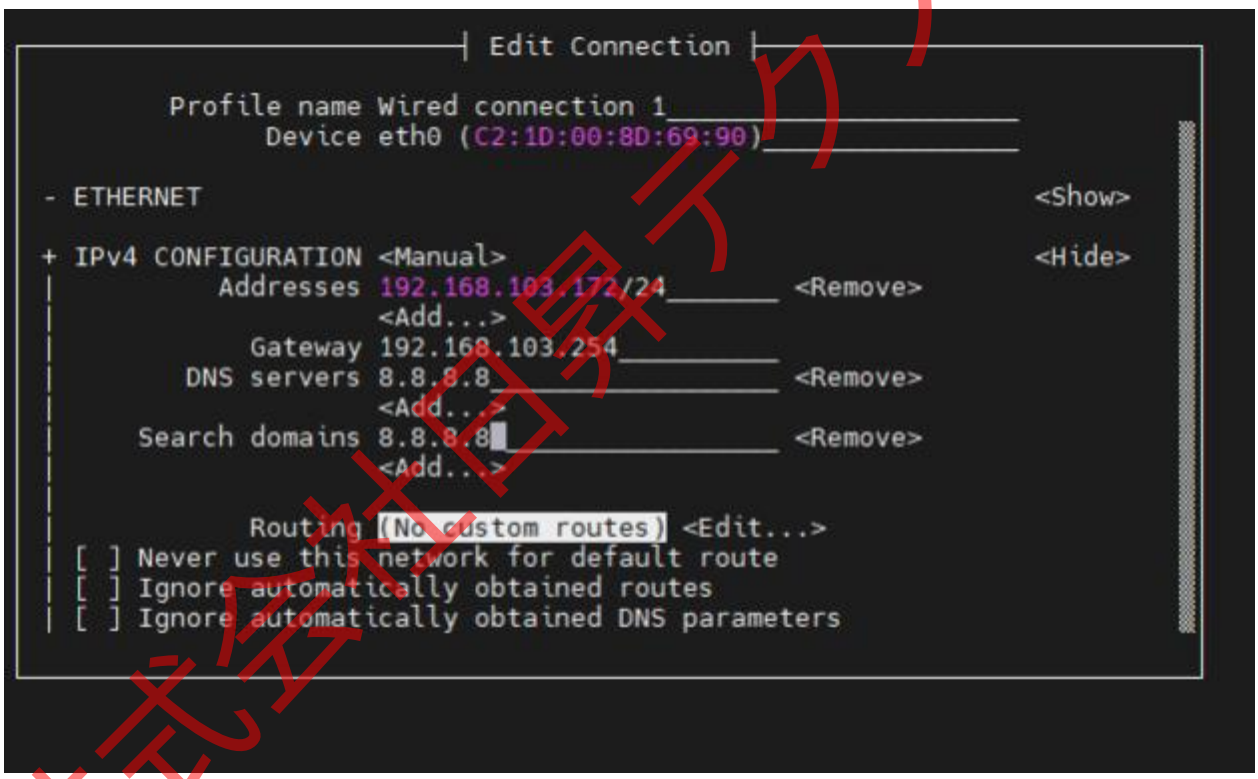
・ip アドレス 192.168.103.172、ゲートウェイ 192.168.103.254 を例に挙げる

注意：ip アドレスとゲートウェイは自分の実際のネットワーク状況に基づいて設定する必要があります、もしここの設定をそのままコピーする場合、ボードがネットに接続できない可能性があります。初心者は静的 ip を動的に取得した ip に修正することをお勧めます

・ゲートウェイの取得方法

```
#1.IP を自動的に取得するようにネットワークを設定する
#2.IP が正常に取得後、次のコマンドを使用します
 route

#結果
root@lubancat:~# route
Kernel IP routing table
Destination    Gateway         Genmask    Flags    Metric    Ref    Use  Iface
Default    _gateway     0.0.0.0  UG    100    0      0    eth0
Default    FusionWrt.lan 0.0.0.0  UG    600    0      0    wlan0

#3.Gateway。次のコマンドを使用します
```

```
route -n

#結果
root@lubancat:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use n Iface
0.0.0.0         192.168.103.254 0.0.0.0         UG    100    0        0 eth0
0.0.0.0         192.168.25.1    0.0.0.0         UG    600    0        0 wlan0

#ゲートウェイのアドレスが表示されます----192.168.103.254
```



・192.168.103.172/24 ここで、/24 はマスク 255.255.255.0 である。

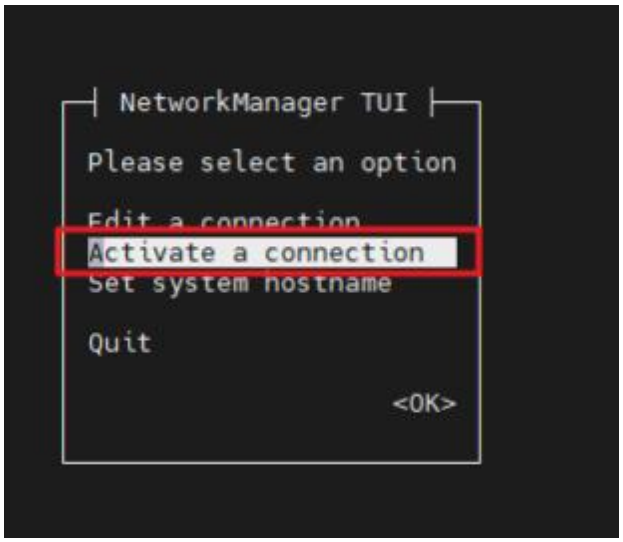
•DNS servers 世界共通 DNS–>8.8.8.8

•search domain は、DNS servers の設定をそのまま使用可能

・DNS servers と search domain は複数設定可能。

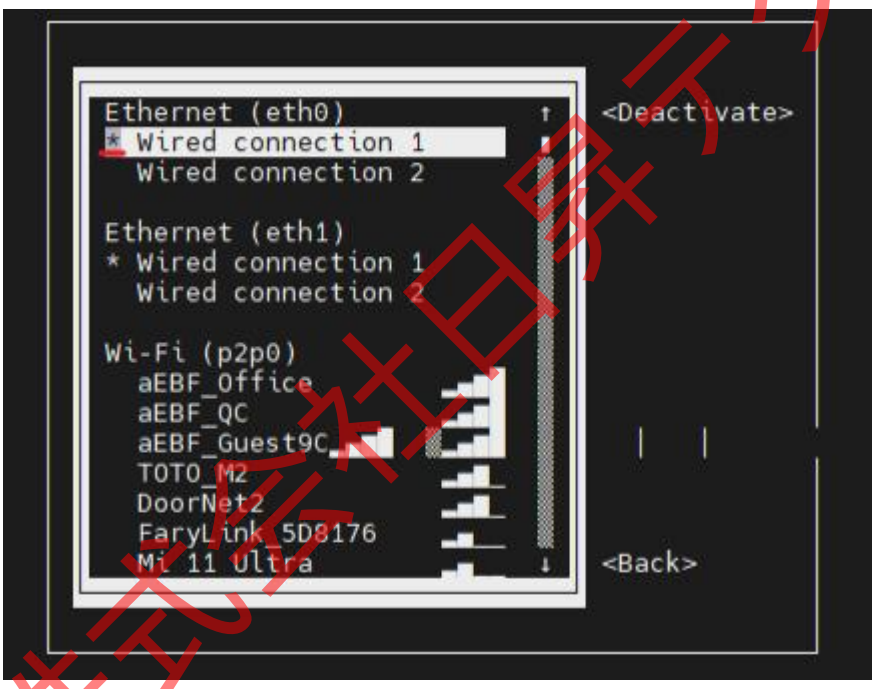
・設定が完了したら、後ろに移動して ok をクリックして設定完了

•設定が完了したら、ネットワークを有効にするには設定を有効にする必要



・Activate a connection をクリックして接続へ

•次の図のステータスは、ネットワークがアクティブになっています



・enter を 1 回押して接続を解除し、もう一度 enter キーを押して再接続する

・再接続後の IP は設定した IP になる

```
root@lubancat:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.156  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::e523:c785:434c:8695  prefixlen 64  scopeid 0x20<link>
        ether 52:43:ff:4f:80:e8  txqueuelen 1000  (Ethernet)
        RX packets 7075  bytes 636531 (621.6 KiB)
        RX errors 0  dropped 476  overruns 0  frame 0
        TX packets 1074  bytes 90251 (88.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 38
```

静的アドレスを設定後

また、ping baidu.com を使用して、外部ネットワークへの接続が成功したかどうかを確認することもできます

```
root@lubancat:~# ping mi.com
PING mi.com (58.83.160.156) 56(84) bytes of data.
64 bytes from 58.83.160.156 (58.83.160.156): icmp_seq=1 ttl=50 time=44.3 ms
64 bytes from 58.83.160.156 (58.83.160.156): icmp_seq=2 ttl=50 time=43.9 ms
64 bytes from 58.83.160.156 (58.83.160.156): icmp_seq=3 ttl=50 time=45.1 ms
64 bytes from 58.83.160.156 (58.83.160.156): icmp_seq=4 ttl=50 time=45.4 ms
64 bytes from 58.83.160.156 (58.83.160.156): icmp_seq=5 ttl=50 time=45.0 ms
^C
--- mi.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 11ms
rtt min/avg/max/mdev = 43.893/44.741/45.415/0.629 ms
root@lubancat:~#
```

5.3.3 nmcli

eth0 の場合、この操作は nmtui でのネットワーク編集と同様で、ただグラフィカルインタフェースからコマンドラインへの操作になっています。コマンドラインには様々なネーミングがあるので、ここでは部分的に紹介するので、興味のある方は自分で探索してみてください

```
#接続されている設定を最初にリストします。eth0 は Wired connection 1 に接続しています
root@lubancat:~# nmcli c s
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  35ecb023-3194-3edb-bf90-4198f82329a8  ethernet  eth0
Wired connection 2  45f86cca-f8df-376e-8aad-37e10d2a65ce  ethernet  --
aEBF_Guest          fb1d506e-6dc7-4c5b-a4ce-339f3349d26d  wifi      --
```

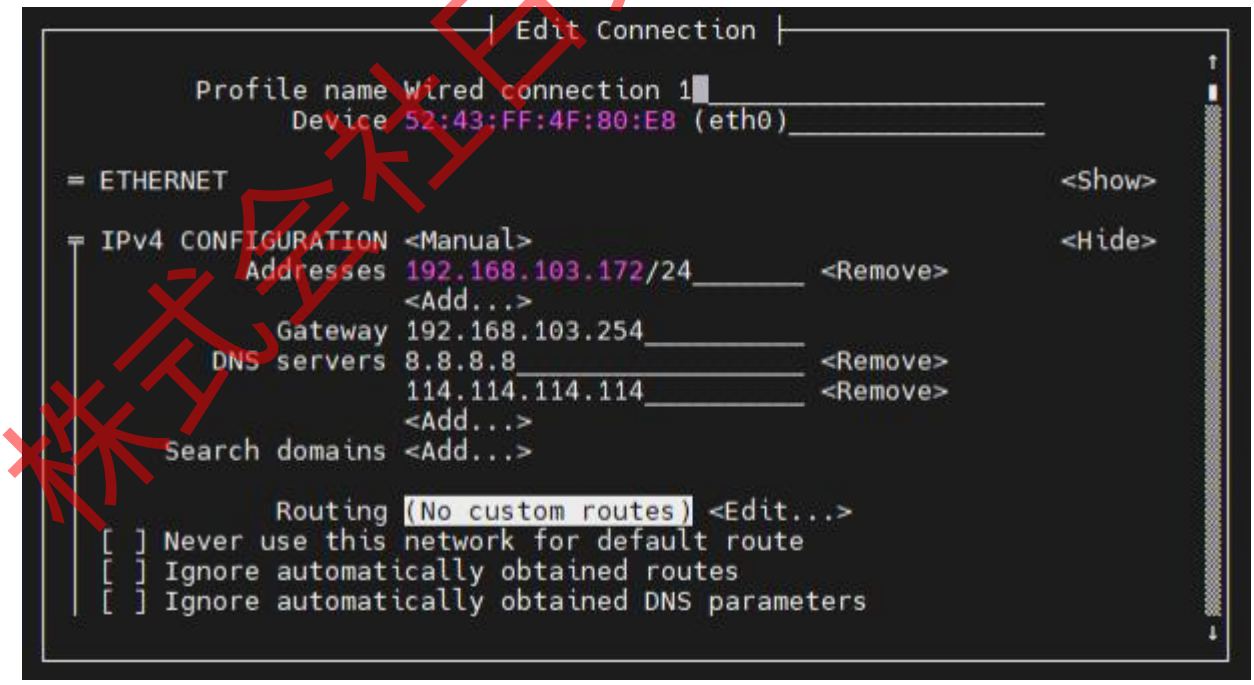
次に Wired connection 1 を修正します

```
#静的 ip 設定
sudo nmcli c modify'Wired connection 1'[+-]オプション　オプション値
#IP アドレスとサブネットマスクの変更
```

```
sudo nmcli c m 'Wired connection 1' ipv4.address 192.168.103.172/24
#静的設定に変更(デフォルトは auto)
sudo nmcli c m'Wired connection 1'ipv4.method manual
#ゲートウェイの変更
sudo nmcli c m 'Wired connection 1' ipv4.gateway 192.168.103.254
#DNS の変更
sudo nmcli c m 'Wired connection 1' ipv4.dns 8.8.8.8
#DNS の追加
sudo nmcli c m 'Wired connection 1' +ipv4.dns 114.114.114.114
#ipv6 の禁止
sudo nmcli c m 'Wired connection 1' ipv6.method disabled
#電源入れて自動接続
sudo nmcli c m 'Wired connection 1' connection.autoconnect yes
```

最初に ipv4.address を変更する必要があり、その後 ipv4.method を変更できることに注意してください。

オプションをデフォルト値に戻すには、オプション値を空の引用符 "" に置き換えます。



nmtui にアクセスして構成を表示できます。構成が変更されていますが、まだ有効になって

いません：

```
root@lubancat:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.156  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::e523:c785:434c:8695  prefixlen 64  scopeid 0x20<link>
        ether 52:43:ff:4f:80:e8  txqueuelen 1000  (Ethernet)
        RX packets 5722  bytes 514529 (502.4 KiB)
        RX errors 0  dropped 596  overruns 0  frame 0
        TX packets 416  bytes 31550 (30.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 38
```

```
#設定のアクティブ化
sudo nmcli c up ifname eth0
```

設定が完了すると ip が変更されます。

```
root@lubancat:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.172  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::e523:c785:434c:8695  prefixlen 64  scopeid 0x20<link>
        ether 52:43:ff:4f:80:e8  txqueuelen 1000  (Ethernet)
        RX packets 5928  bytes 532677 (520.1 KiB)
        RX errors 0  dropped 625  overruns 0  frame 0
        TX packets 459  bytes 34772 (33.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 38
```

6 SSH 端末登録及びネット共用

SSH は、リモートログインセッションや他のネットワークサービスにセキュリティを提供するための信頼性の高いプロトコルです。

注意: SSH 端末ログインの前提は、コンピュータとボードがネットワークを通じて接続および通信できることです。

インターネット接続とは異なり、SSH はインターネットを必要とせず、コンピュータとボードが小規模なネットワークを構成するだけで十分です。

6.1 ネットワークプラン

- もしルーターがあり、コンピュータがすでにルーターに接続されている場合、Wi-Fi または有線でルーターに接続することができます。

- デスクトップコンピュータを使用している場合、デュアルネットワークポートを備えたコ

ンピュータがあれば、1 つのネットワークポートをボードに接続し、もう 1 つのネットワークポートをインターネットに接続します（設定方法：後述のネットワーク共有）。

- コンピュータに USB ポートがあり、通信可能な TYPE-C ケーブルがある場合、コンピュータがネットワークに接続されていれば、USB 経由でネットワークを共有することができます。

- ノートパソコンを使用している場合、またはデスクトップコンピュータにワイヤレスネットワークカードがある場合：

1. Wi-Fi でインターネットに接続し、有線でボードに接続（設定方法：後述のネットワーク共有）。
2. 有線でネットワークに接続し、コンピュータのホットスポットを有効にしてボードに接続。
3. 2.4G および 5G 帯域の両方を持つワイヤレスネットワークカードがあれば、Wi-Fi でネットワークに接続し、ホットスポットを有効にしてボードに接続。

注意: 私たちのガイドの大部分はインターネット接続を必要とするため、可能であればボードをネットワークに接続し、魂を吹き込んでください。

具体的な操作方法は以下の「ネットワーク展開」を参照してください。

6.2 MobaXterm を使用した SSH 端末ログイン

MobaXterm を使用してボードに SSH ログインすることをお勧めします。このツールは SSH とシリアル機能だけでなく非常に強力です。

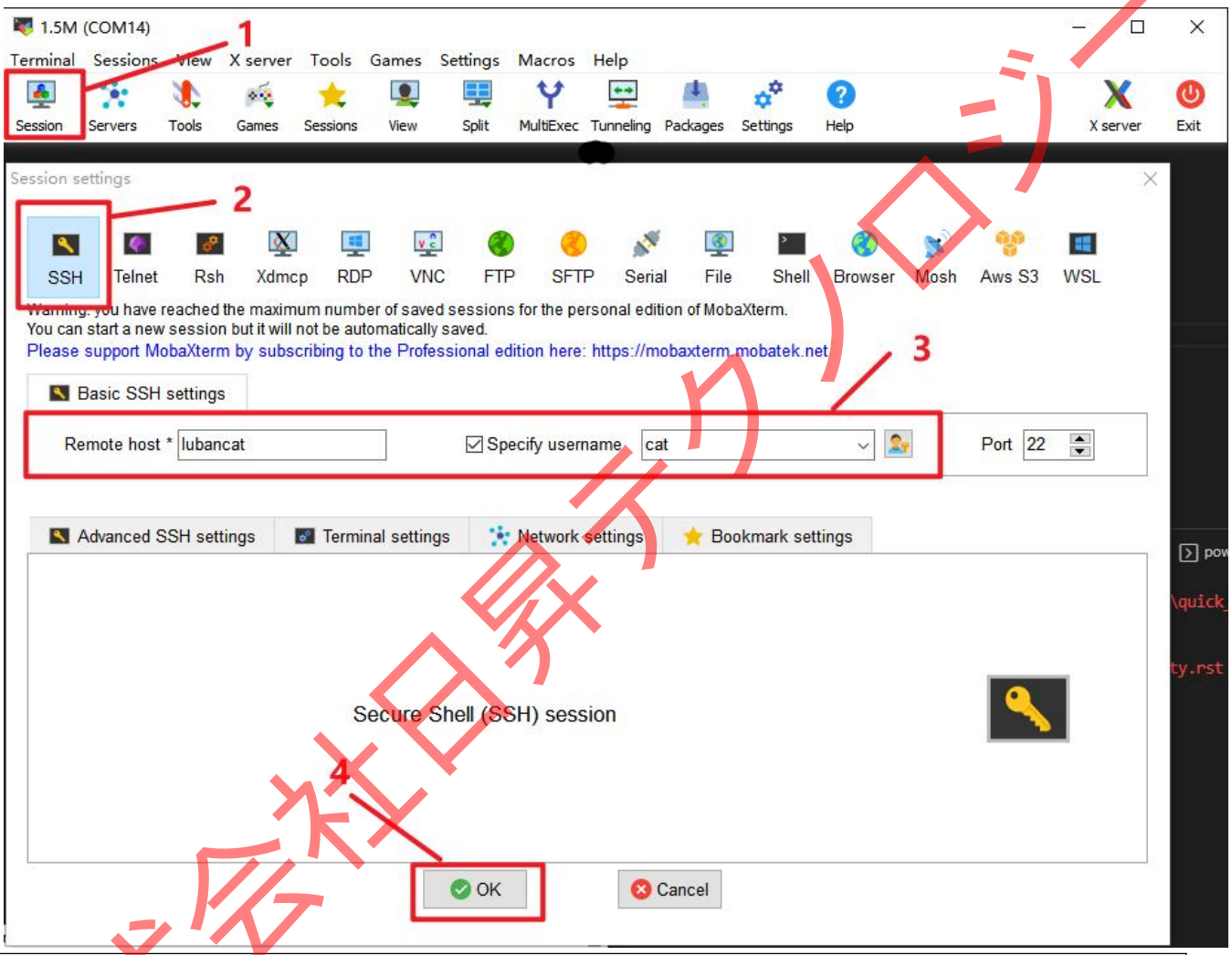
6.2.1 ホスト名を使用した SSH ログイン

この方法は、ボードの IP が不明な場合に適しています。

MobaXterm ソフトウェアを開き、セッションアイコンをクリックしてセッション設定を表示し、SSH を選択します。

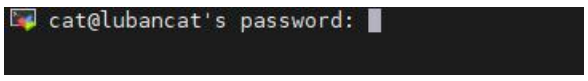
- ボードのホスト名は「lubancat」で、後の「Specify username（指定ユーザー）」は任意で記入できます。記入すると、後のログイン時にパスワードを入力するだけで済みます。

- この方法は、ネットワーク内に 1 台のボードがある場合にのみ適しています。複数のデバイスがある場合、ランダムに 1 つに接続されます。



1. ユーザー名：cat
2. ユーザーパスワード：temppwd

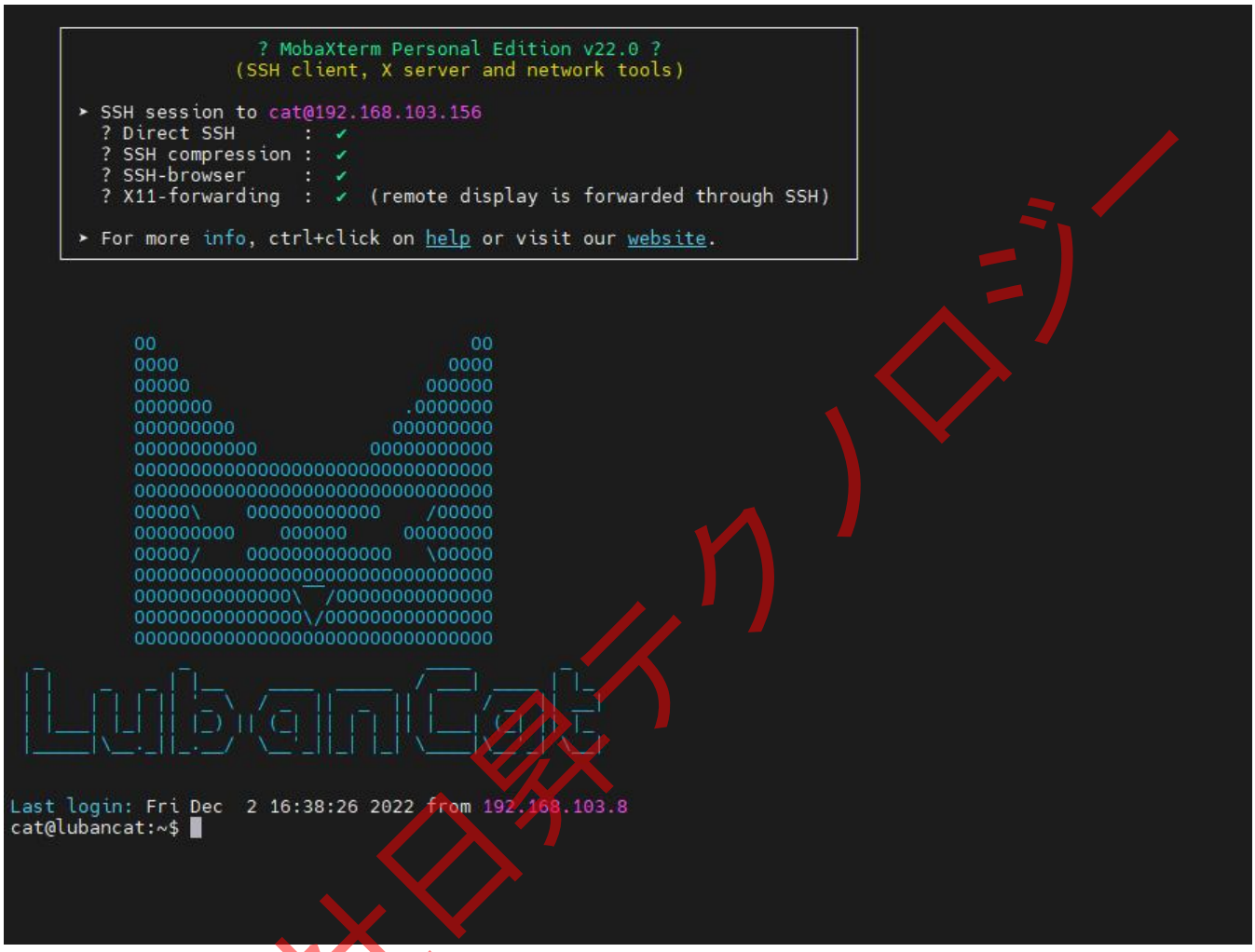
- 上記の手順に従った場合、パスワードを入力するだけ（明文表示されません）で、確認ボタンを押すと以下の画面が表示されます。



- 指定ユーザーがない場合、最初にユーザー名を入力する必要があります（明文表示されます）、次にパスワードを入力します（表示されません）。

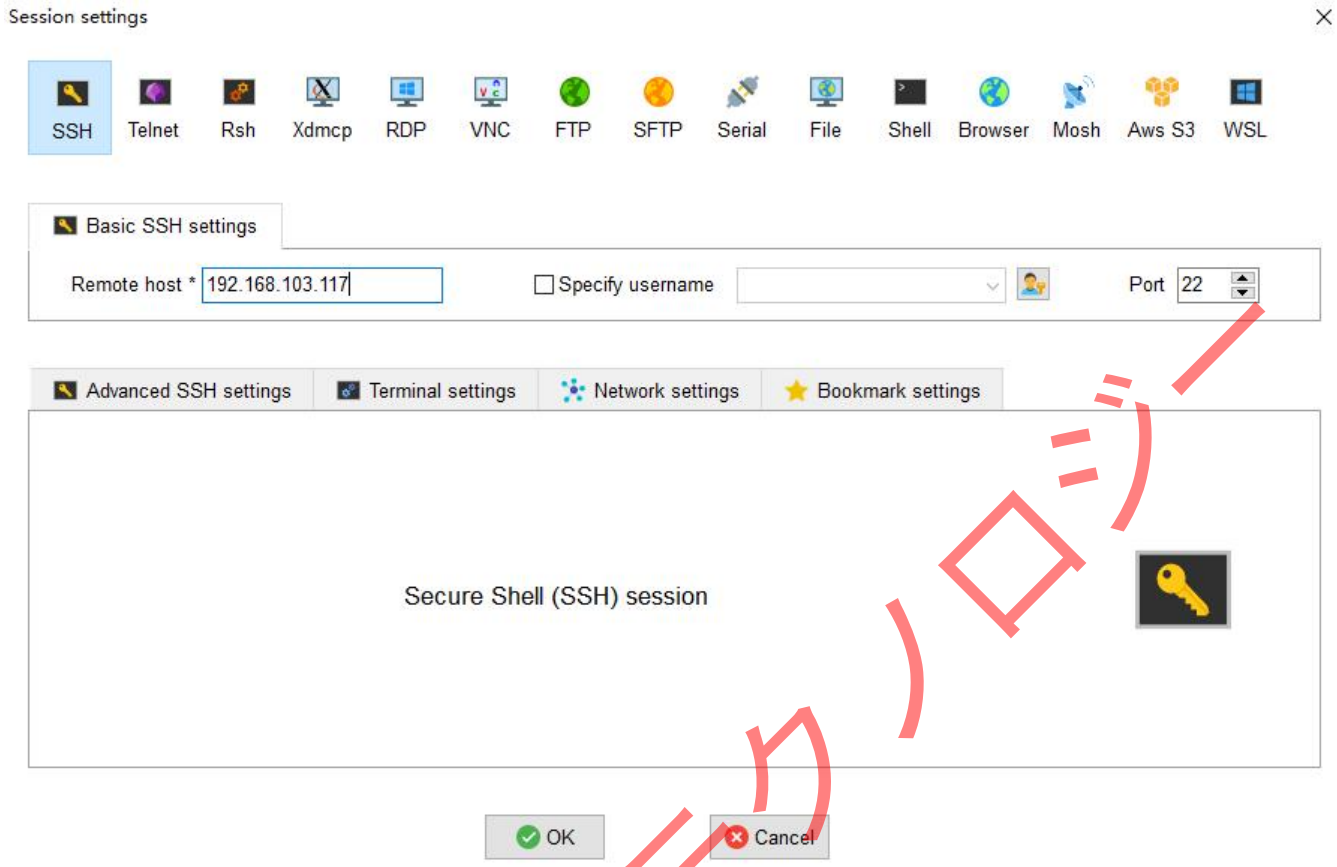
注意: 入力方法と大文字小文字に注意してください。

正しく入力すると、以下の画面が表示されます。

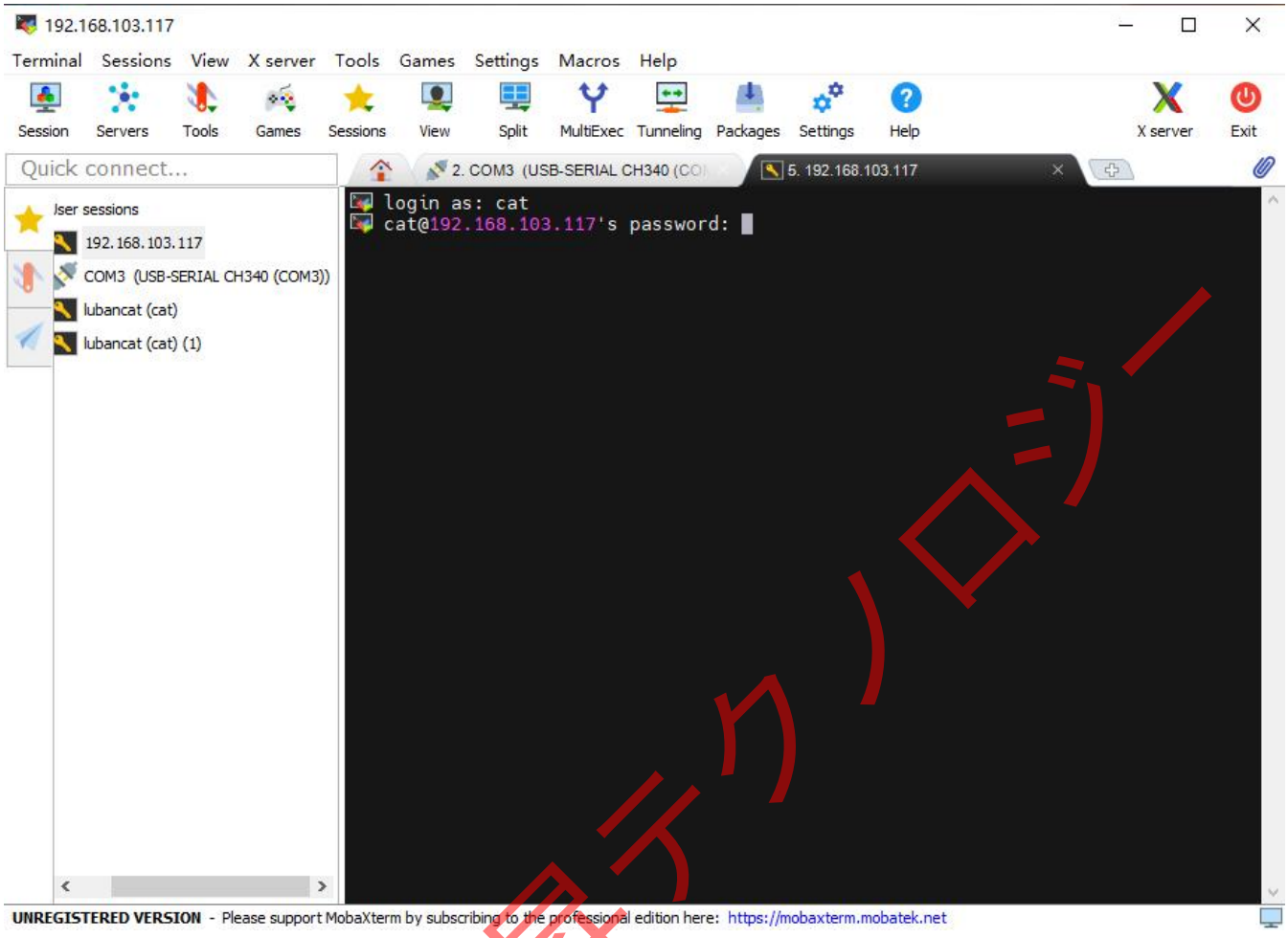


6.2.2 指定 IP - SSH ログイン

ボードの IP がわかっている場合は、指定 IP で SSH 端末にログインできます。この方法はボードの IP がわかっている場合に適しています。



指定ユーザー名がない場合は、ユーザー名とパスワードを入力する必要があります。事前にユーザー名を設定している場合は、パスワードだけを入力すればよいです。



ヒント: シリアル端末および SSH 端末でアカウントパスワードを入力するときは明文表示されません。完全に入力した後、Enter キーを押します。

6.2.3 root ユーザー – SSH ログイン

6.2.3.1 ログイン前の準備

LubanCat イメージの工場出荷時設定では、root ユーザーの SSH ログインはサポートされていません。SSH ログイン root ユーザーを設定するには、設定を変更する必要があります。

- 一般ユーザーおよび root ユーザーの変更方法

```
#1. ファイル /etc/ssh/sshd_config を編集
 sudo vi /etc/ssh/sshd_config
```

```
#2. ファイルの末尾に「PermitRootLogin yes」を追加
#3. ファイルを保存
#4. sshd サービスを再起動
systemctl restart sshd
```

注釈: ssh ログイン中に sshd サービスを再起動すると、パスワードを入力することで端末の使用を続行できます。

- root ユーザーの変更方法（一般ユーザーは su コマンドを使用して切り替え可能、パスワードは root）

```
#1. ファイル /etc/ssh/sshd_config を編集
echo "PermitRootLogin yes" >> /etc/ssh/sshd_config
#2. sshd サービスを再起動
systemctl restart sshd
```

注釈: ssh ログイン中に sshd サービスを再起動すると、パスワードを入力することで端末の使用を続行できます。

6.2.3.2 root ユーザーでのログイン

root ユーザーの SSH ログイン方法は、一般ユーザーのログイン方法と同じですが、ユーザー名を「cat」から「root」に変更し、パスワードを「temppwd」から「root」に変更します。

注釈: 詳細な操作方法については、上記の「ホスト名を使用した SSH ログイン」および「指定 IP - SSH ログイン」を参照してください。

7 apt 更新ソース

apt は、Ubuntu、Debian、および関連する Linux ディストリビューションで deb パッケージをインストール、更新、削除、および管理するためのコマンドラインユーティリティです。

普段使用するソフトウェアやコマンドは、apt からダウンロードして使用できます。

7.1 更新とアップグレード

ボードを最新の状態に保つことは非常に重要です。最初の、そしておそらく最も重要な理由はセキュリティです。Linux オペレーティングシステムを実行するデバイスは、何百万行ものコードに依存しています。時間が経つにつれて、これらのコードに既知の脆弱性が明らかになり、これらの脆弱性は公開されているデータベースに記録されているため、簡単に悪用される可能性があります。

また、一部のソフトウェアは最新のパッケージに依存しているため、更新とアップグレードによってより多くのソフトウェアと互換性を持つことができます。

注記: 新しいイメージを焼き込んだ場合や一部のソフトウェアがインストールできない場合、apt を使用して更新とアップグレードを行うことをお勧めします。

注意: apt コマンドを使用するにはネットワーク接続が必要です。

```
# 1. ソフトウェアパッケージデータベースを更新する
  sudo apt update
# 2. インストール済みのソフトウェアパッケージをアップグレードする
  sudo apt upgrade
```

注意: アップグレード中に update エラーが発生した場合は、コマンドを再実行してください。

以下のようなエラーが発生した場合、システム時間が正しくないことを意味し、ネットワークが時間を取得できていません。

```
cat@lubancat:~$ sudo apt update
Get:1 http://mirrors.ustc.edu.cn/debian buster InRelease [122 kB]
Get:2 http://mirrors.ustc.edu.cn/debian-security buster/updates InRelease [34.8
kB]
Get:3 http://mirrors.ustc.edu.cn/debian buster-updates InRelease [56.6 kB]
Reading package lists... Done
E: Release file for http://mirrors.ustc.edu.cn/debian/dists/buster/InRelease is
not valid yet (invalid for another 1304d 1h 17min 37s). Updates for this reposit
ory will not be applied.
E: Release file for http://mirrors.ustc.edu.cn/debian-security/dists/buster/upda
tes/InRelease is not valid yet (invalid for another 1354d 8h 50min 10s). Updates
 for this repository will not be applied.
E: Release file for http://mirrors.ustc.edu.cn/debian/dists/buster-updates/InRel
ease is not valid yet (invalid for another 1354d 16h 2min 51s). Updates for this
 repository will not be applied.
```

解決方法：

1. しばらく待って、ネットワークが時計を更新するのを待ちます。

2. コマンド date -s を使用して手動で時間を設定します。

7.2 apt の一般的なコマンド

7.2.1 apt を使用してパッケージデータベースを更新する

apt は実際には利用可能なパッケージのデータベース上で動作します。データベースが更新されていない場合、システムは利用可能な更新されたパッケージがあるかどうかを知ることができません。これが、Linux システムをインストールした後の最初の作業として apt データベースを更新する必要がある理由です。

```
sudo apt update
```

このコマンドを実行すると、さまざまなサーバーからパッケージ情報が取得されるのがわかります。

7.2.2 apt を使用してインストール済みソフトウェアパッケージをアップグレードする

ソフトウェアパッケージデータベースを更新した後、インストール済みのソフトウェアパッケージをアップグレードできます。最も便利な方法は、利用可能なすべての更新ソフトウェアパッケージをアップグレードすることです。

sudo apt upgrade

7.2.3 apt を使用してソフトウェアパッケージをインストールする

```
sudo apt install package_name
```

理由があって特定のソフトウェアパッケージをインストールしたいが、アップグレードしたくない場合、すでにインストールされている場合にのみアップグレードできます。

```
sudo apt install <package_name> --no-upgrade
```

ソフトウェアパッケージをインストールせずにアップグレードしたい場合は、以下のコマンドを使用してアップグレードできます。

```
sudo apt install <package_name> --only-upgrade
```

特定のバージョンのソフトウェアをインストールする場合

```
sudo apt install <package_name>=<version_number>
```

7.2.4 apt を使用してソフトウェアパッケージを削除する

インストール済みのソフトウェアパッケージを削除するには

```
sudo apt remove package_name
# 複数のパッケージを指定する場合は、スペースで区切ります
sudo apt remove package1 package2
```

remove コマンドは指定されたソフトウェアパッケージをアンインストールしますが、一部の構成ファイルは残る可能性があります。すべての構成ファイルを含むソフトウェアパッケージを削除するには、remove の代わりに purge を使用します。

```
sudo apt purge
```

7.2.5 apt を使用して未使用のソフトウェアパッケージを削除する

新しいソフトウェアパッケージをシステムにインストールする際、そのパッケージが依存する他のソフトウェアパッケージもインストールされます。パッケージを削除すると、依存関係がシステムに残ります。これらの残りのパッケージは、他の何物にも使用されないため、削除することができます。

```
sudo apt autoremove
```

7.2.6 apt を使用してソフトウェアパッケージのリストを生成する

list コマンドを使用すると、利用可能、インストール済み、アップグレード可能なソフトウェアパッケージをリストアップできます。

```
sudo apt list
# このコマンドは、パッケージのバージョンやアーキテクチャに関する情報を含むすべてのパッケージのリストを出力します。特定のソフトウェアパッケージがインストールされているかどうかを確認するには、grep コマンドを使用して出力をフィルタリングできます。
```

```
sudo apt list | grep package_name
# インストール済みのソフトウェアパッケージのみをリストアップするには
sudo apt list --installed
# 実際にソフトウェアパッケージをアップグレードする前に、アップグレード可能なソフトウェアパッケージのリストを取得することが有用です。
sudo apt list --upgradeable
```

7.2.7 apt を使用してソフトウェアパッケージを検索する

このコマンドを使用すると、利用可能なソフトウェアパッケージリストから特定のソフトウェアパッケージを検索できます。

```
sudo apt search package_name
```

7.2.8 apt を使用してソフトウェアパッケージの情報を表示する

新しいソフトウェアパッケージをインストールまたは削除する前に、パッケージの依存関係、インストールサイズ、ソフトウェアパッケージのソースなどの情報が有用です。

```
sudo apt show package_name
```

7.2.9 apt を使用してダウンロードファイルのアーカイブをクリーンアップする

```
sudo apt-get clean
```

7.2.10 apt を使用してソフトウェアソースコードをダウンロードする

```
sudo apt-get source <packages>
```

7.2.11 apt を使用してソフトウェア依存関係を確認する

```
sudo apt-cache depends <packages>
```

7.2.12 apt を使用してソフトウェア依存関係をチェックする

```
sudo apt-get check
```

8 パッケージの更新

システムの安定稼働を保証するために、定期的にカーネルのアップグレードパッケージをリリースしています。アップグレードパッケージの更新内容は更新記録ファイルに記録されているので、必要に応じてシステムを更新することができます。

アップグレードパッケージはカーネル関連のファイルのみを置き換えるため、カーネル関連でないフォルダに保存されているファイルは変更されません。したがって、システムが稼働している状態でアップグレードパッケージをインストールし、インストール後に再起動す

るだけでアップグレードが完了し、ファイルも消失しません。

8.1 自動アップグレード

アップグレードパッケージはウェブサイトにも同期されるので、以下のコマンドで直接アップグレードできます。

8.1.1 注意事項

カーネルパッケージを更新すると、uEnv.txt ファイルが元の状態に戻るため、再度設定が必要です。

「sudo apt upgrade」を使用してカーネルを更新したくない場合は、以下の操作を行うことができます。

```
# カーネルの自動更新を無効にする
  sudo apt-mark hold linux-image-5.10.160 linux-headers-5.10.160
# 上記の操作を解除する場合
  sudo apt-mark unhold linux-image-5.10.160 linux-headers-5.10.160
```

注記: デフォルトでは自動更新は無効になっており、upgrade ではカーネルを更新できません。

8.1.2 手動で更新

```
# ソフトウェアソースを更新する
  sudo apt update
# ソフトウェアソース内のカーネルイメージパッケージを検索する
  sudo apt search 5.10.160
```

- ソフトウェアソースに Linux カーネルの更新パッケージがあることが確認できます。

```
# 通常のカーネルをインストールする
  sudo apt install linux-image-5.10.160 linux-headers-5.10.160
# インストール完了後、再起動する
  sudo reboot
```

8.2 手動でアップグレードパッケージをインストールする

自分でカーネルをコンパイルする場合、SDK 上でコンパイルした deb パッケージを手動でインストールできます。ファイル転送を通じてボードに転送します。

アップグレードパッケージは全部で 4 つあり、バージョンによって名前が異なります。以下のように表示されます。

```
cat@lubancat:~$ ls *.deb
linux-headers-5.10.160_5.10.160-1_arm64.deb linux-image-5.10.160_5.10.160-1_arm64.deb
linux-image-5.10.160-dbg_5.10.160-1_arm64.deb linux-libc-dev_5.10.160-1_arm64.deb
cat@lubancat:~$
# インストール方法
# パッケージをインストールする
  sudo dpkg -i *.deb
# インストール後、再起動する
```

8.3 アップグレードパッケージの分析

- linux-headers-5.10.160_5.10.160-1_arm64.deb**

このパッケージはカーネルのソースコードであり、インストール後に/usr/src/linux-headers-5.10.160 にコピーされます。

- linux-libc-dev_5.10.160-1_arm64.deb**

このパッケージは libc のヘッダーファイルであり、インストールは必須ではありません。

- linux-image-5.10.160-dbg_5.10.160-1_arm64.deb**

このパッケージはカーネルのデバッグ用カーネルおよびドライバであり、システムのエラーやクラッシュ時にデバッグに使用されます（インストールは必須ではありません）。

- linux-image-5.10.160_5.10.160-1_arm64.deb**

このパッケージにはカーネル、ドライバ、デバイスツリー、およびアップグレードスクリプトが含まれています。

ここでは、各アップグレードパッケージの簡単な説明のみを行いましたが、詳細な分析は読者に委ねます。deb パッケージの内容を確認するためのコマンドを提供します。

```
sudo dpkg-deb -c xxx.deb
```

9 ファイル転送と NFS ネットワークファイルシステム

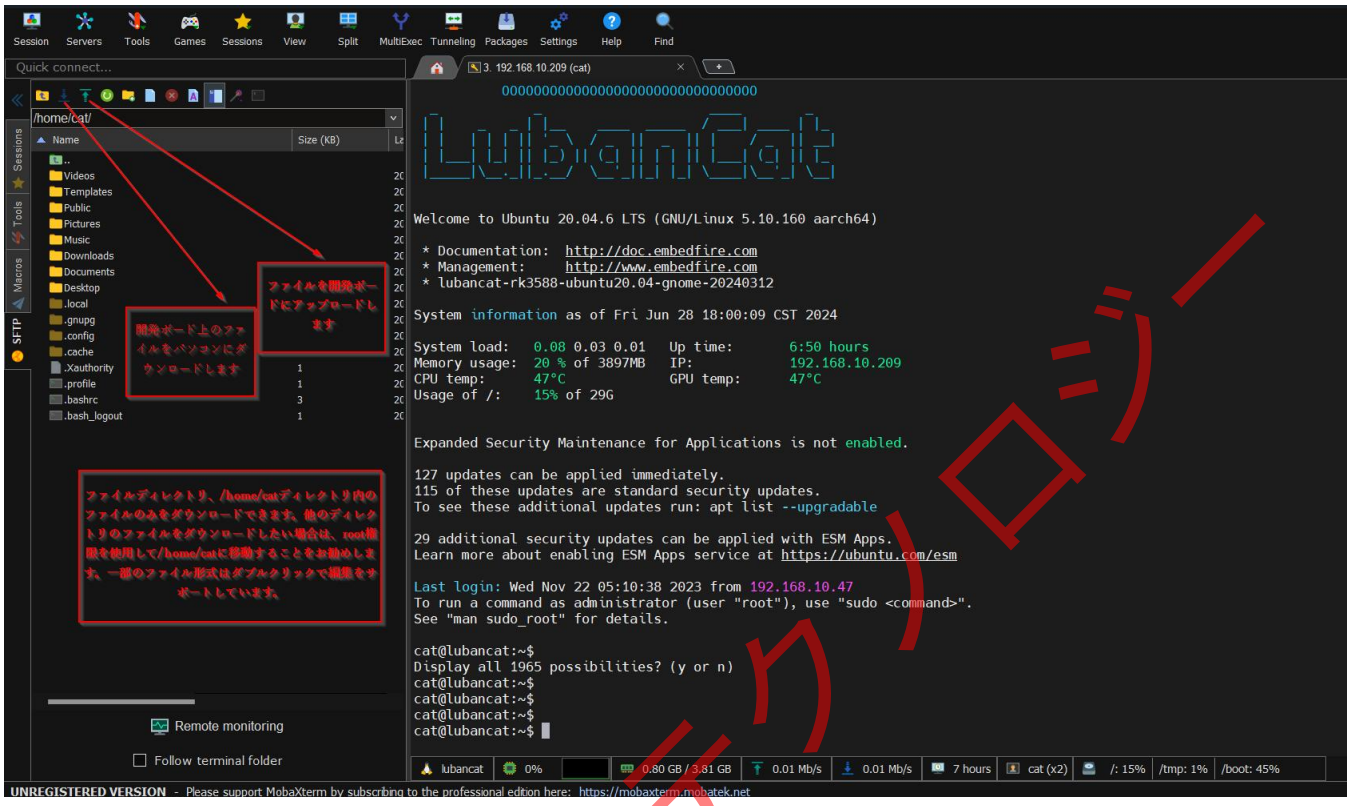
この章では、PC ホストとボード間でファイルを転送する方法について説明します。

この章の内容はネットワークを介した転送に依存しており、ボードのネットワーク環境を設定する必要があります。具体的な設定方法については、以下の章「SSH ログイン」を参照してください。

本章では、以下の方法を中心に紹介します。

- MobaXterm
- FileZilla
- NFS
- TFTP

9.1 MobaXterm



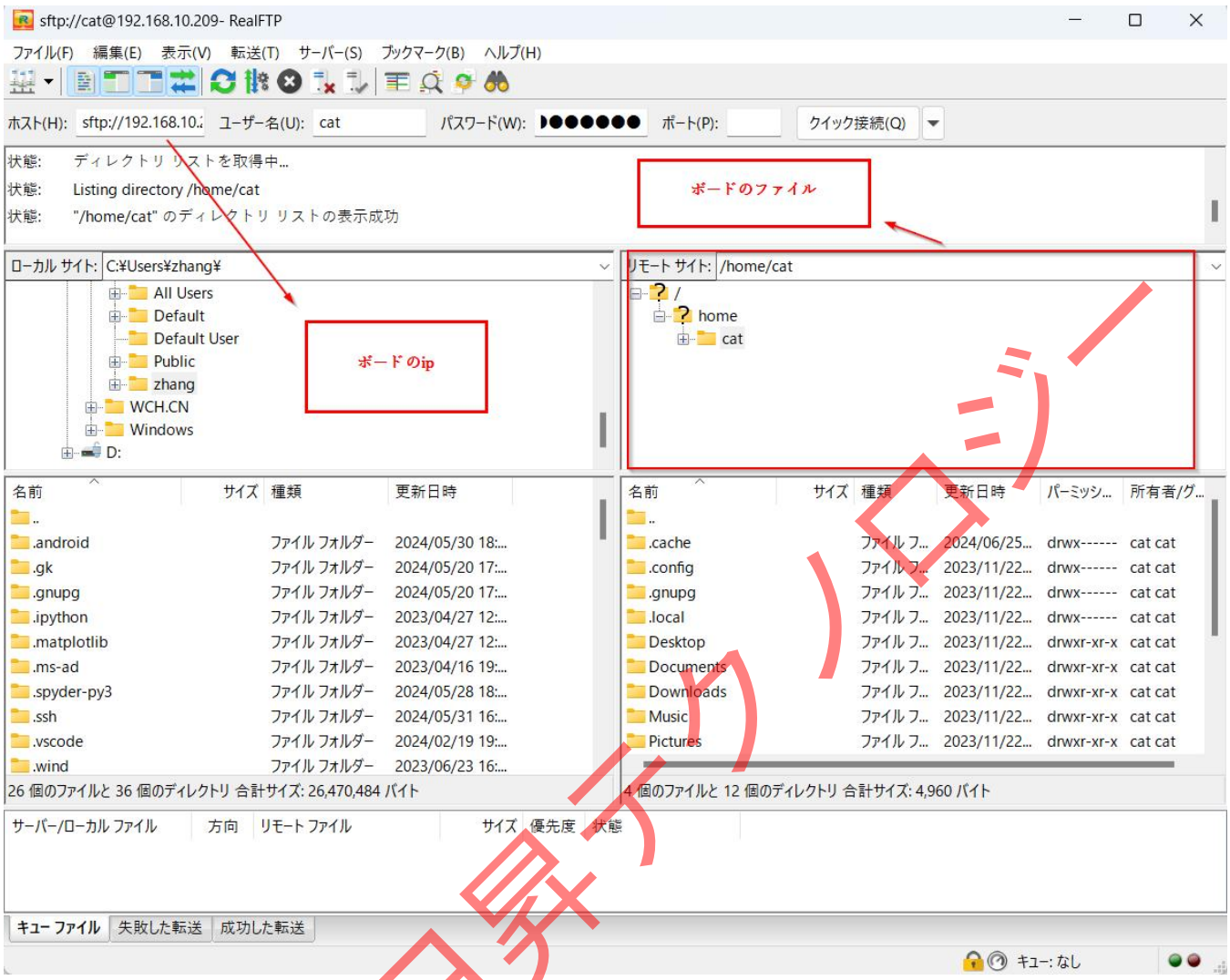
この方法では、MobaXterm ソフトウェアをダウンロードする必要があります。

- メリット: Linux を操作しながら、ファイルの転送も簡単に行えます。

- デメリット: /home/cat ディレクトリ以下のフォルダのみ転送可能で、リンクファイルの正確な転送ができません。ソースコードの移植には支障があります。

9.2 FileZilla

FileZilla をインストール後、FileZilla を開きます。



- メリット: より詳細なディレクトリ構造が提供され、ファイル転送がより便利になります。

- デメリット: /home/cat ディレクトリ以下のフォルダのみ転送可能で、リンクファイルの正確な転送ができません。ソースコードの移植には支障があります。

9.3 NFS ファイルシステム

9.3.1 NFS 環境の構築

次に、NFS を利用して上記の環境を構築する方法について説明します。主にネットワーク接続、ホストでの NFS サービスの開始、およびボードでのファイルシステムのマウントの 3 つのステップに分かれます。

目標は、開発ホストとボードの以下のディレクトリをマッピングすることです：

- 開発ホストの共有ディレクトリ: /home/zerok/workdir

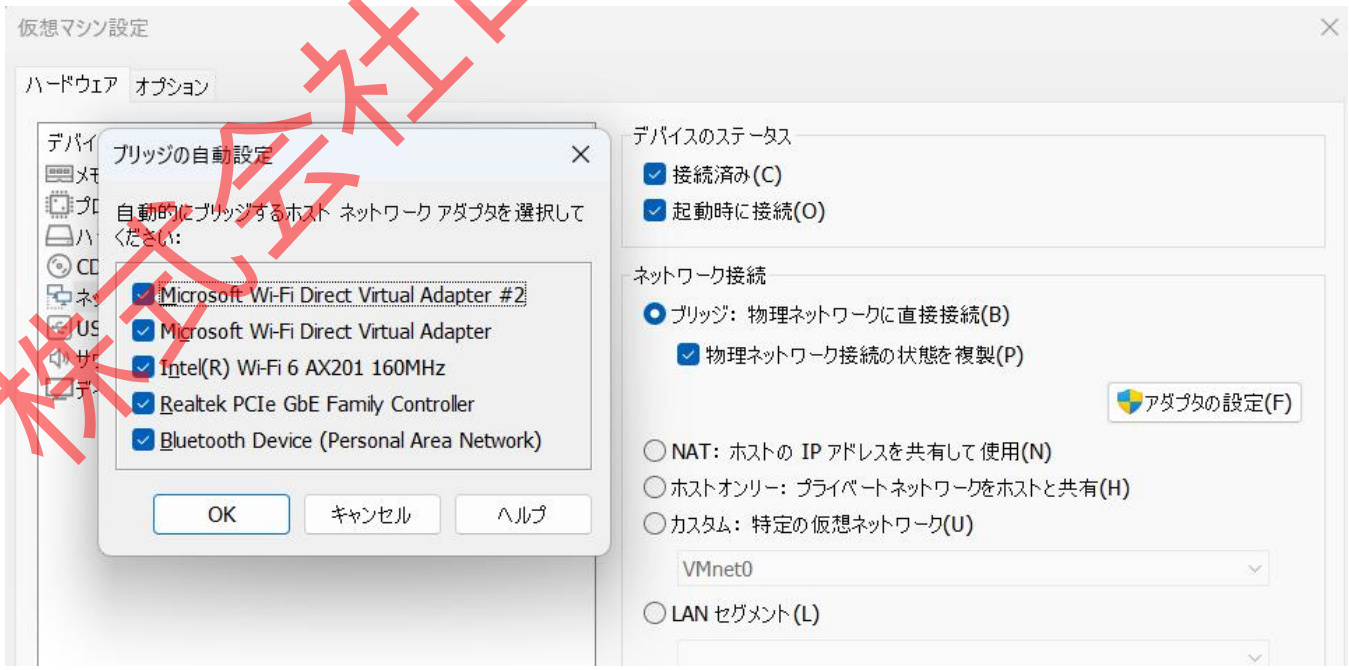
- ボードのマウントディレクトリ: /mnt

9.3.1.1 ローカルネットワークへの接続

9. 3. 1. 1. 1 ネットワーク設定

このシナリオでは、開発ホストとボードがネットワークを介して相互にアクセスする必要があります。また、NFS ファイルシステムを公衆ネットワークに公開することは多くのセキュリティ問題を引き起こすため、操作を簡素化するために、開発ホストとボードをローカルネットワークに接続します。つまり、両方とも同じスイッチ（ルーター）に有線で接続します。

もし開発ホストが仮想マシン上にインストールされている場合は、VirtualBox で仮想マシンのネットワーク設定を「ブリッジアダプタ」モードに変更する必要があります。設定を変更した場合は、仮想マシンを再起動して設定を有効にします。



9.3.1.1.2 相互の ping テスト

ifconfig コマンドを使用して、それぞれの IP アドレスとサブネットマスクを確認できます。

既知のホスト IP アドレスは: 192.168.103.132

ボードのアドレスは: 192.168.103.156

注意: 開発ホストで ifconfig を使用してコマンドが見つからない場合は、以下のコマンドを使用してインストールします。

```
sudo apt install net-tools
```

開発ホストの IP アドレスとサブネットマスクに基づいて、開発ホストが 192.168.100.*のネットワークセグメントにあることがわかります。同じローカルネットワークセグメント内にいることを確認するだけで、相互に通信できます。

もし ifconfig コマンドで IP アドレスが表示されない場合や、IP アドレスが開発ホストの IP と同じネットワークセグメントにない場合は、ネットワーク接続を確認してください。

開発ホストからボードへの ping

```
[redacted]:~$ ping 192.168.103.156
PING 192.168.103.156 (192.168.103.156) 56(84) bytes of data.
64 bytes from 192.168.103.156: icmp_seq=1 ttl=64 time=0.895 ms
64 bytes from 192.168.103.156: icmp_seq=2 ttl=64 time=1.14 ms
64 bytes from 192.168.103.156: icmp_seq=3 ttl=64 time=0.878 ms
64 bytes from 192.168.103.156: icmp_seq=4 ttl=64 time=1.05 ms
64 bytes from 192.168.103.156: icmp_seq=5 ttl=64 time=0.896 ms
64 bytes from 192.168.103.156: icmp_seq=6 ttl=64 time=0.959 ms
64 bytes from 192.168.103.156: icmp_seq=7 ttl=64 time=0.867 ms
64 bytes from 192.168.103.156: icmp_seq=8 ttl=64 time=1.14 ms
64 bytes from 192.168.103.156: icmp_seq=9 ttl=64 time=0.857 ms
64 bytes from 192.168.103.156: icmp_seq=10 ttl=64 time=0.872 ms
^C
--- 192.168.103.156 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9110ms
rtt min/avg/max/mdev = 0.857/0.954/1.135/0.105 ms
```

ボードからホストへの ping

```
cat@lubancat:~$ ping 192.168.103.132
PING 192.168.103.132 (192.168.103.132): 56 data bytes
64 bytes from 192.168.103.132: icmp_seq=0 ttl=64 time=0.901 ms
64 bytes from 192.168.103.132: icmp_seq=1 ttl=64 time=1.070 ms
64 bytes from 192.168.103.132: icmp_seq=2 ttl=64 time=1.208 ms
64 bytes from 192.168.103.132: icmp_seq=3 ttl=64 time=1.113 ms
64 bytes from 192.168.103.132: icmp_seq=4 ttl=64 time=0.870 ms
64 bytes from 192.168.103.132: icmp_seq=5 ttl=64 time=0.888 ms
64 bytes from 192.168.103.132: icmp_seq=6 ttl=64 time=0.921 ms
64 bytes from 192.168.103.132: icmp_seq=7 ttl=64 time=1.308 ms
64 bytes from 192.168.103.132: icmp_seq=8 ttl=64 time=1.274 ms
^C--- 192.168.103.132 ping statistics ---
9 packets transmitted, 9 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.870/1.061/1.308/0.164 ms
cat@lubancat:~$
```

9.3.1.2 开发ホストで NFS サービスを開始

ネットワークを確認し、ローカルネットワークの IP アドレス情報を把握した後、開発ホストの NFS サービスを設定できます。以下のステップはすべて開発ホスト上で行います。

9.3.1.2.1 NFS サービスのインストール

Ubuntu システムにはデフォルトで NFS サービスがインストールされていないため、以下のコマンドで NFS サーバーソフトウェアをインストールする必要があります。

```
sudo apt install nfs-kernel-server
```

9.3.1.2.2 ユーザー ID の確認

NFS を設定する際には、ユーザー UID とグループ GID を使用します。id コマンドで確認できます。開発ホストのターミナルで以下のコマンドを入力します。

```
id
```

上図で id コマンドを使用してユーザー ID とグループ ID を確認できます。

この例では、開発ホストのユーザー UID とグループ GID が 998 であることがわかります。自分の開発ホストの関連 ID を確認して、以下の設定ファイルで使用します。

9.3.1.2.3 NFS の設定

NFS サービスをインストールした後、/etc/exports ファイル（/etc ディレクトリ内の exports という名前のファイル）が追加されます。NFS サービスはこのファイルの設定に基づいて動作します。

vim を使用して/etc/exports ファイルを開くコマンドは以下の通りです。

```
sudo vim /etc/exports
```

/etc/exports ファイルの末尾に以下の内容を追加して保存します。以下の内容はすべて同じ行に記述します。

```
/home/zerok/workdir 192.168.103.0/24(rw,sync,all_squash,anonuid=1000,anongid=1000,no_subtree_check)
```

具体的な設定は自分の実験環境に基づいて調整する必要があります。以下の説明を理解して、自分の環境に合わせて修正してください：

- /home/zerok/workdir: 共有する開発ホストのディレクトリ。後ろの設定とスペースで区切ります。

- 192.168.103.0/24: アクセスを許可する範囲を指定します。/24 はマスクで、24 個の 1、つまり 11111111.11111111.11111111.00000000 を表し、マスクは 255.255.255.0 です。

192.168.103.0 は、192.168.103.*の IP アドレスを持つホストがこのディレクトリにアクセスできることを意味します。

- rw: クライアントの権限を指定します。rw は読み書き可能を意味し、具体的な権限はファイルシステムの rwx およびユーザーのアイデンティティに依存します。

- sync: データをメモリとハードディスクに同期して書き込みます。

- anonuid=1000: クライアントのユーザーを指定のローカルユーザー ID にマッピングします。ここでは 1000 が開発ホスト zerok ユーザーの UID です。具体的なホストユーザーの UID に基づいて設定してください。

- anongid=1000: クライアントのユーザーを指定のローカルユーザーグループ ID にマッピングします。ここでは 1000 が開発ホスト embedfire ユーザーグループの GID です。具体的なホストユーザーグループの GID に基づいて設定してください。

- no_subtree_check: サブディレクトリの権限をチェックしないデフォルト設定です。

この設定では、クライアントのユーザーをローカル UID/GID が 1000 のユーザー（開発ホストの zerok）にマッピングします。したがって、ボード上で開発ホストと異なるユーザーを使用して NFS 共有ディレクトリにアクセスすると、すべて zerok の権限を持つことになります。これにより、相互アクセスが簡単になります。たとえば、ボード上の root ユーザーがファイルを作成すると、開発ホスト上では zerok が作成したと見なされます。開発ホスト上で zerok しか読み書きできないファイルも、ボード上の root または他のユーザーから読み書きできます（zerok として見なされるため）。この設定は安全なアクセス方式ではありませんが、開発作業には非常に便利です。

9.3.1.2.4 共有ディレクトリの作成

共有設定を有効にするためには、共有ディレクトリを作成する必要があります。この例で

は、共有ディレクトリは/home/zerok/workdir です。

以下のコマンドを使用してディレクトリを作成します。実験環境に合わせて共有ディレクトリを設定してください。

```
mkdir /home/zerok/workdir
```

9.3.1.2.5 exports 設定の更新

/etc/exports ファイルを編集して保存した後、exportfs コマンドを使用して設定を更新できます。

```
sudo exportfs -arv
```

このコマンドのパラメータの説明は以下の通りです：

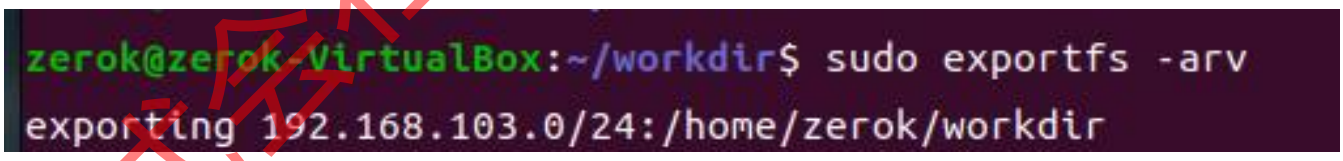
- **-a**: /etc/exports ファイル内のすべての内容をマウントまたはアンマウントします。

- **-r**: /etc/exports ファイル内の共有内容を再マウントします。

- **-u**: ディレクトリをアンマウントします。

- **-v**: exportfs 時に詳細情報を画面に出力します。

設定が正常であれば、このコマンド実行後に共有ディレクトリの項目がリストされます。この例の実行結果は以下の通りです。



9.3.1.2.6 NFS の共有状況の確認

showmount –e コマンドを使用して現在の NFS サーバーのロード状況を確認できます。

```
showmount -e
```

9.3.1.3 NFS クライアントのインストール

開発ホストで NFS サービスを開始した後、ボードで NFS クライアントをインストールして NFS サービスを使用できるようにする必要があります。

NFS クライアントのインストールコマンドを実行します。

```
sudo apt install nfs-common -y
```

NFS サーバーの共有ディレクトリを確認

ボード上で「showmount -e + "NFS サーバー IP"」コマンドを実行します。ネットワーク環境によって NFS サーバーの IP が異なる場合があるので、実際の状況に応じて使用します。

```
showmount -e 192.168.103.132
```

```
cat@lubancat:~$
cat@lubancat:~$ showmount -e 192.168.103.132
Export list for 192.168.103.132:
/home/zerok/workdir 192.168.103.0/24
```

9.3.1.3.1 NFS ファイルシステムの一時マウント

mount コマンドを使用して、NFS サーバーの共有ディレクトリをボードの/mnt ディレクトリにマウントします。

注意: 以下の 192.168.103.132 をユーザーの実際のネットワーク環境の NFS サーバー IP に設定してください。

```
sudo mount -t nfs 192.168.103.132:/home/zerok/workdir /mnt
```

このコマンドで使用される各パラメータの説明は以下の通りです：

- **-t nfs**: マウントするファイルシステムの形式を nfs に指定します。

- **192.168.103.132**: NFS サーバーの IP アドレスを指定します。

- **/home/zerok/workdir**: NFS サーバーの共有ディレクトリを指定します。

- **/mnt**: ローカルのマウントディレクトリ。NFS サーバーの共有ディレクトリをボードの/mnt ディレクトリにマッピングします。

マウントが成功すると、ターミナルには何も表示されません。Linux の哲学では「何もメッセージがないことが良いニュース」です。

この方法でマウントされたディレクトリは一時的なものであり、ボードを再起動すると再度マウントが必要です。

9.3.1.3.2 NFS 共有ディレクトリのテスト

マウントに成功した後、NFS サーバーの共有ディレクトリで「sudo touch hello_world.txt」コマンドを入力すると、共有ディレクトリに hello_world.txt ファイルが作成されます。

```
zerok@zerok-VirtualBox:~/workdir$ sudo touch hello_world.txt
zerok@zerok-VirtualBox:~/workdir$ ls
hello_world.txt
zerok@zerok-VirtualBox:~/workdir$
```

ボードの/mnt ディレクトリに移動すると、NFS サーバーの共有ディレクトリ内の hello_world.txt ファイルが確認できます。

```
cat@lubancat:~$ ls /mnt/
hello_world.txt
cat@lubancat:~$
```

9.3.1.3.3 アンマウント

クライアントがネットワーク上で NFS 共有ディレクトリを見つけられない場合（例えば、開発ホストがシャットダウンした場合）、NFS クライアントのターミナルにはいくつかのメッセージが表示されることがあります。また、ls コマンドを使用して共有ディレクトリを確認すると長時間待機することがあります。この場合、umount コマンドを使用してディレクトリをアンマウントできます。例は以下の通りです。

```
sudo umount /mnt
```

このコマンドを使用する際には、アンマウントするディレクトリをパラメータとして指定

します。出力がなければ正常に実行されています。現在マウントされているディレクトリで umount 操作を行うと「デバイスがビジーです」と表示されることがあります。アンマウントする際には、まずホームディレクトリ「~」に切り替えてから umount 操作を行うことをお勧めします。

- メリット: ソフトリンク付きファイルの転送が可能で、リムーバブルハードディスクのように使える。拡張性があり、使いやすい。

- デメリット: グラフィカルユーザーインターフェースがなく、Linux 間でのみファイル転送が可能。

9.4 TFTP

この方法では、特別なソフトウェアをダウンロードする必要はなく、ファイルエクスプローラーを使用するだけで済みます。

フォルダを開いて以下を入力します。

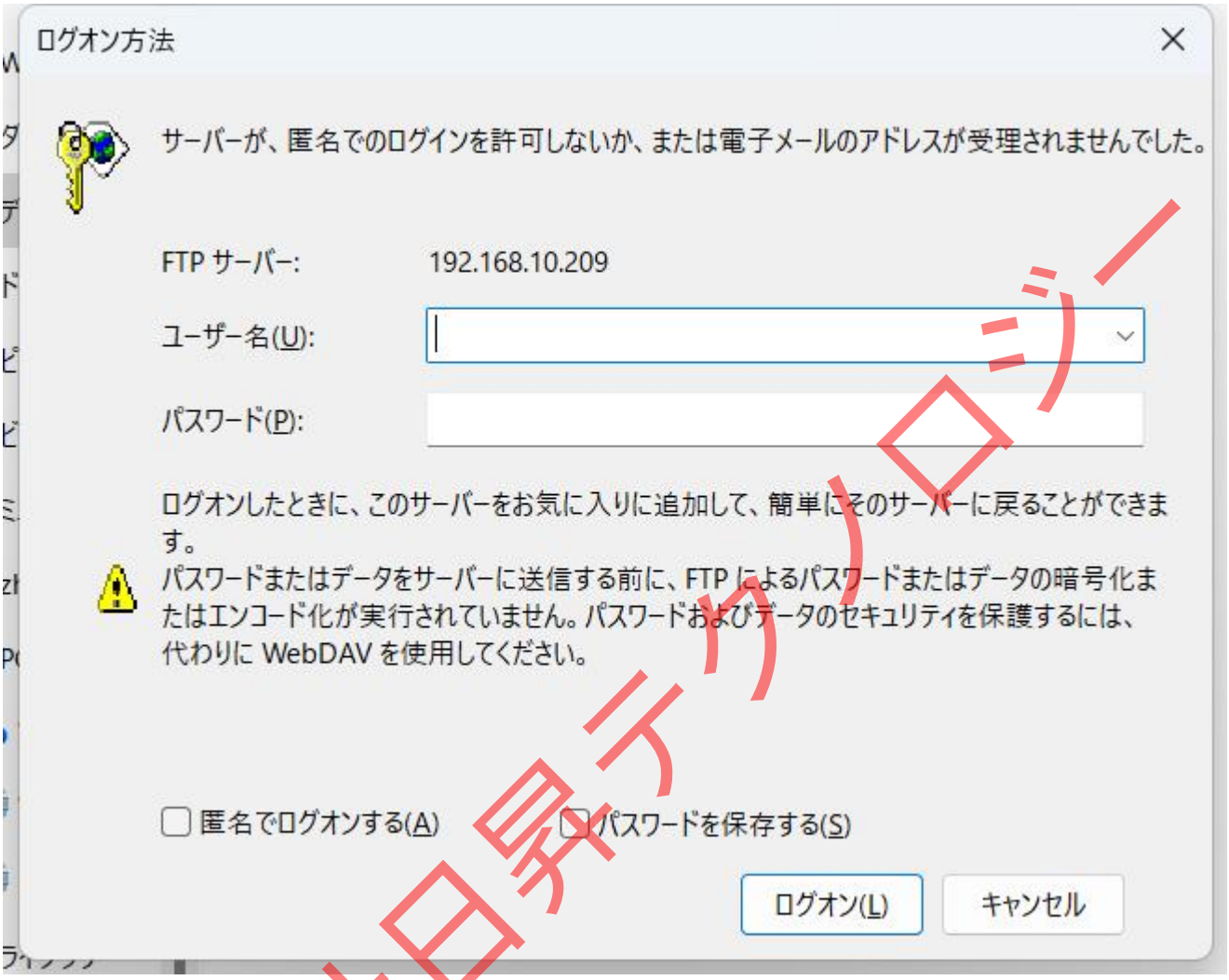
```
# TCP 接続: この方法を推奨します。
  tftp://192.168.103.156/
# UDP 接続
  ftp://192.168.103.156/
```

次に、ユーザー名とパスワードを入力します。これで成功です。

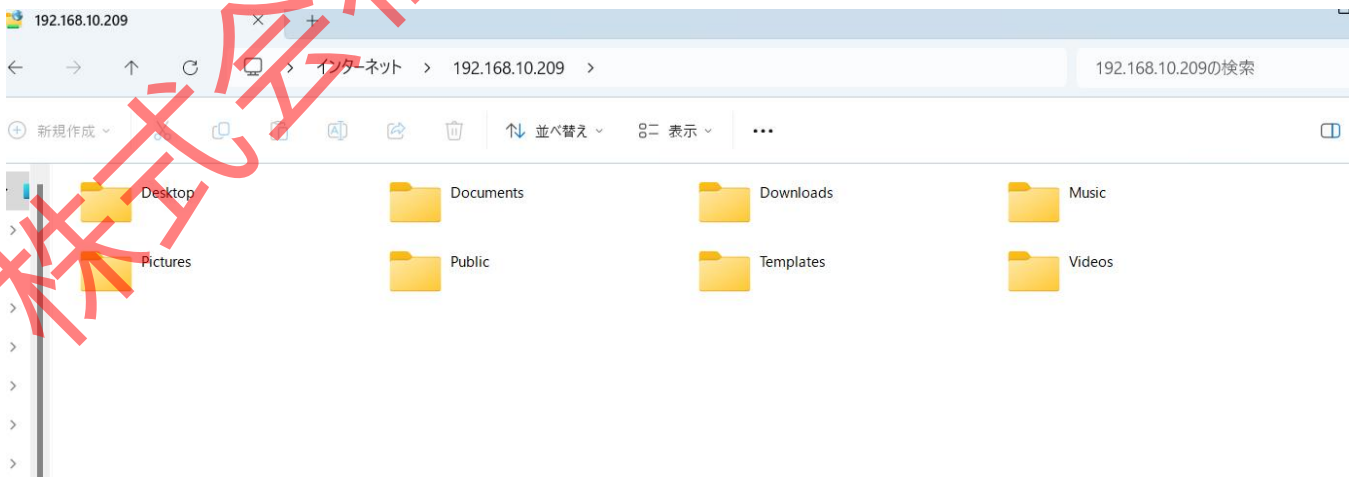
- メリット: システムに直接組み込まれています。

- デメリット: /home/cat ディレクトリ以下のフォルダのみ転送可能で、リンクファイルの正確な転送ができません。ソースコードの移植には支障があります。

ID とパスワードを入力



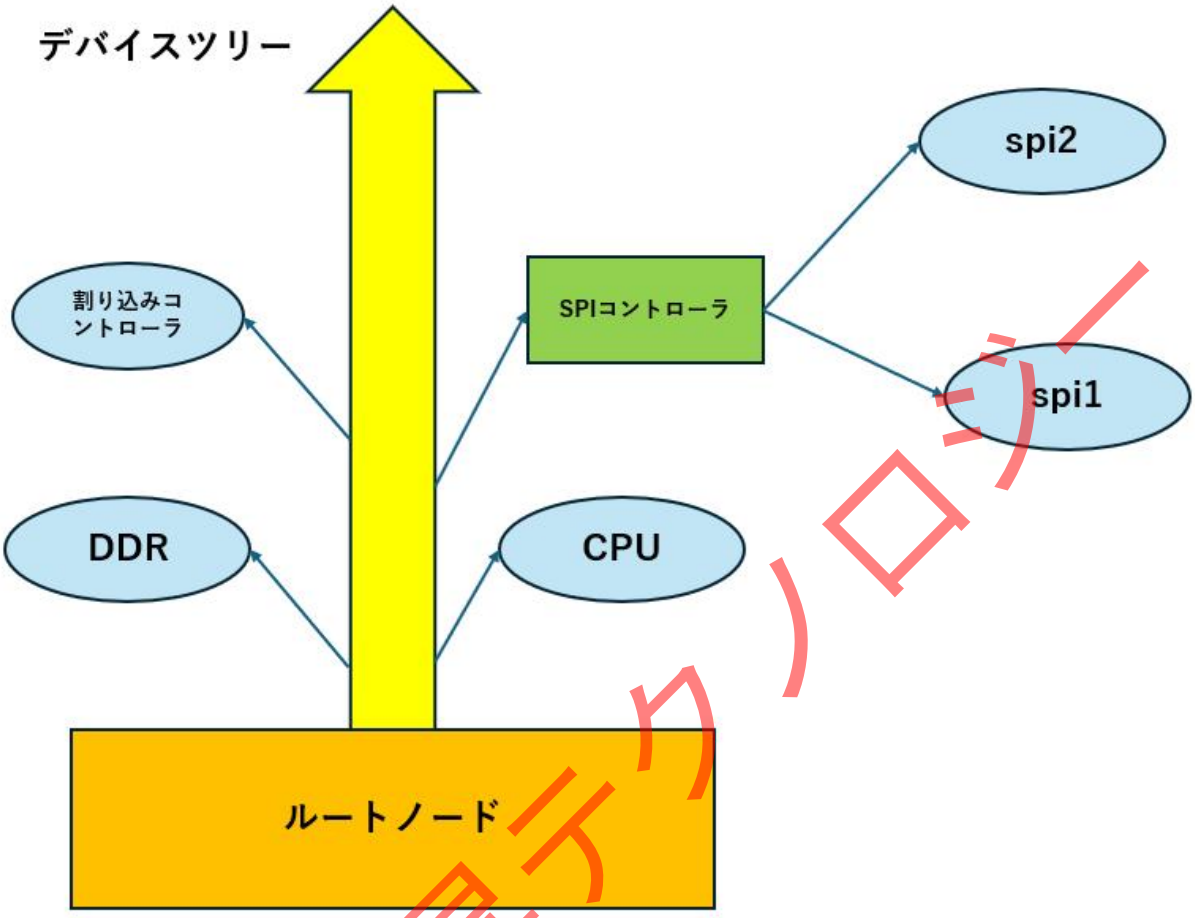
成功



10 プロファイル、デバイスツリー、デバイスツリー プラグイン

10.1 デバイスツリー

Linux カーネルはバージョン 3.x からデバイスツリーの概念を導入し、ドライバコードとデバイス情報を分離しました。デバイスツリーはハードウェアプラットフォームのハードウェアリソースを記述するために使用されます。この「デバイスツリー」はブートローダ（uboot）によってカーネルに渡され、カーネルはデバイスツリーからハードウェア情報を取得できます。デバイスツリーをカスタマイズすることで、i2c、spi、mipi、mini-pcie、i2s などのインターフェースを操作するために必要なハードウェアリソースを設定して有効化できます。



デバイスツリーは「ツリー構造」でハードウェアリソースを記述します。たとえば、ローカルバスがツリーの「幹」であり、デバイスツリーでは「ルートノード」と呼ばれます。ローカルバスに接続された spi バスはツリーの「枝」であり、デバイスツリーでは「ルートノードの子ノード」と呼ばれます。spi バスの下には複数の spi デバイスがあり、これらの「枝」はさらに分かれます。

10.1.1 デバイスツリーの使用

ボードでは、専用の主デバイスツリーがあり、/boot/dtb に配置されています。以下の図のように表示されます：

```
root@lubancat:~# ls /boot/dtb/
overlay                  rk3588-lubancat-5io.dtb          rk3588s-lubancat-4.dtb
rk3588-lubancat-5.dtb    rk3588-lubancat-rk_series.dtb    rk3588s-lubancat-4-v1.dtb
root@lubancat:~# ls /boot/
```

共通イメージは多くの異なるボードをサポートする必要があるため、起動時にデバイス ID を読み取り、デバイスツリーを変更して異なるボードに適応させます。

ボードのデバイスツリーは/boot/rk-kernel.dtb というファイルで指定されます。以下の図のように表示されます：

```
root@lubancat:/boot# ls -l /boot
总用量 56428
-rw-rw-r-- 1 root root     1537  8月  7 08:52 boot.cmd
-rw-r--r-- 1 root root        0  6月 18 22:56 boot_init
-rw-rw-r-- 1 root root     1609  8月  7 08:52 boot.scr
-rw-r--r-- 1 root root   198337  8月  4 10:01 config-5.10.160
drwxrwxr-x 3 root root     4096  6月 18 22:56 dtb
drwxrwxr-x 2 root root     4096  8月  7 08:52 extlinux
-rw-r--r-- 1 root root 35389952  8月  4 10:01 Image-5.10.160
-rw-rw-r-- 1 root root  7069963  8月  7 08:52 initrd-5.10.160
-rw-r--r-- 1 root root  5755545  6月 18 22:56 initrd.img-5.10.160
drwxrwxr-x 2 root root     4096  6月 18 22:56 kerneldeb
-rw-rw-r-- 1 root root  1215954  6月 18 22:56 logo.bmp
-rw-rw-r-- 1 root root  1215954  8月  7 08:53 logo_kernel.bmp
drwx------ 2 root root    16384  8月  7 08:53 lost+found
lrwxrwxrwx 1 root root       26  6月 18 22:56 rk-kernel.dtb → dtb/rk3588s-lubancat-4.dtb
-rw-r--r-- 1 root root  6028870  8月  4 10:01 System.map-5.10.160
drwxrwxr-x 2 root root     4096  6月 18 22:56 uEnv
root@lubancat:/boot#
```

rk-kernel.dtb は、dtb/rk3588s-lubancat-4.dtb というデバイスツリーにソフトリンクされています。したがって、システム起動時にこのデバイスツリーが使用されます。

10.2 設定ファイル

ボードでは、設定ファイルが非常に重要な機能を持っています。設定ファイルはデバイスツリープラグインの有効化と無効化、カーネルの起動パラメータ、起動するカーネルの選択を設定できます。設定ファイルは/boot/uEnv にあります。

```
root@lubancat:~# ls /boot/uEnv/
uEnv.txt                  uEnvLubanCat4.txt    uEnvLubanCatA2IO.txt
uEnvLubanCat-series.txt   uEnvLubanCat5.txt
uEnvLubanCat4-V1.txt      uEnvLubanCat5IO.txt
root@lubancat:~#
```

設定ファイルは複数あります。uEnvLubanCat4.txt を使ってください。

注記: 新バージョンのイメージでは、uEnv.txt は対応するボードの設定ファイルに自動的にソフトリンクされます。ls -l /boot/uEnv/uEnv.txt または ls -l /boot/uEnv を使用して、実際

のリンク先の設定ファイルを確認できます。

LubanCat-4：

```
root@lubancat:/home/cat# ls -l /boot/uEnv/
total 24
lrwxrwxrwx 1 root root   17 Nov 22  2023 uEnv.txt -> uEnvLubanCat4.txt
-rw-rw-r-- 1 root root  204 Mar 12 13:44 uEnvLubanCat-series.txt
-rw-rw-r-- 1 root root 2736 Mar 12 13:44 uEnvLubanCat4-V1.txt
-rw-rw-r-- 1 root root 2718 Mar 12 13:44 uEnvLubanCat4.txt
-rw-rw-r-- 1 root root 1930 Mar 12 13:44 uEnvLubanCat5.txt
-rw-rw-r-- 1 root root  322 Mar 12 13:44 uEnvLubanCat5IO.txt
-rw-rw-r-- 1 root root  192 Mar 12 13:44 uEnvLubanCatA2IO.txt
root@lubancat:/home/cat#
```

```
root@lubancat:/home/cat# cat /boot/uEnv/uEnvLubanCat4.txt
uname_r=5.10.160
size=0x1000000
cmdline="earlyprintk console=ttyFIQ0 console=tty1 consoleblank=0 logleve
l=7 rootwait rw rootfstype=ext4"

enable_uboot_overlays=1
#overlay_start

#dtoverlay=/dtb/overlay/rk3588-lubancat-i2c2-m4-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-i2c3-m1-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-i2c5-m3-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-i2c6-m3-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-i2c8-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-pwm3-ir-m3-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-pwm10-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-pwm11-ir-m3-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-pwm14-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-pwm15-ir-m3-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-spi0-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart0-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart4-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart6-m1-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart7-m1-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart7-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-uart9-m2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-can0-m0-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588-lubancat-can2-m0-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-overlay.dtbo
```

```
## vp0
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-hdmi0-8k-overlay.dtbo
## vp1
#
## vp2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dp0-in-vp2-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
## vp3
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-1080p-overlay.dtbo
```

デバイスプラグイン

- cmdline を変更すると、カーネル起動パラメータを設定できます。詳細は省略します。

- デバイスツリープラグインの行頭に#がない場合、デバイスツリープラグインが有効です。

- デバイスツリープラグインの行頭に#がある場合、デバイスツリープラグインが無効です。

- dtoverlay=dtb/overlay/xxx.dtbo の後ろにデバイスツリープラグインのパスとファイル名を指定します。直接そのパスを確認してもそのフォルダは存在しないため、ベースディレクトリ/boot を挿入する必要があります。デバイスツリープラグインのアクセス範囲は/boot ディレクトリ内に限定されています。デバイスツリープラグインを追加する場合は、/boot ディレクトリに配置し、正しいパスを設定してください。

注意: 設定ファイルを変更した後、再起動して設定ファイルを有効にする必要があります。電源の抜き差しで再起動しないでください。ファイルが完全にストレージに書き込まれていない可能性があり、設定ファイルの変更が成功しないことがあります。どうしても電源の抜き差しで再起動する必要がある場合は、抜き差し前に`sync`コマンドを実行し、設定ファイルをストレージに書き込んでください

10.3 デバイスツリープラグイン

Linux 4.4 以降、ダイナミックデバイスツリー（Dynamic DeviceTree）が導入されました。デバイスツリープラグインは、メインデバイスツリーの「パッチ」として理解できます。プラグインは動的にシステムにロードされ、カーネルによって認識されます。

デバイスツリープラグインは、メインデバイスツリーを「剪定」および「接ぎ木」するためのものです。

- 剪定: 不要なデバイスを無効にする

- 接ぎ木: メインデバイスに新しいデバイスノードを追加する

10.3.1 デバイスツリープラグインの使用

ボードのデバイスツリープラグインは/boot/dtb/overlay にあります。

以下のように表示されます：

```
1 root@lubancat:/boot/dtb/overlay# ls
2 rk3588-lubancat-5-cam0-imx415-overlay.dtbo
3 rk3588-lubancat-5-cam1-imx415-overlay.dtbo
4 rk3588-lubancat-5-cam2-ov8858-overlay.dtbo
5 rk3588-lubancat-5-cam3-ov8858-overlay.dtbo
6 rk3588-lubancat-5-cam4-ov8858-overlay.dtbo
7 rk3588-lubancat-5-cam5-ov8858-overlay.dtbo
8 rk3588-lubancat-5-dp0-disabled-overlay.dtbo
9 rk3588-lubancat-5-dsi0-1024x600-overlay.dtbo
10 rk3588-lubancat-5-dsi0-1080p-overlay.dtbo
11 rk3588-lubancat-5-dsi0-800x1280-overlay.dtbo
12 rk3588-lubancat-5-dsi1-1024x600-overlay.dtbo
13 rk3588-lubancat-5-dsi1-1080p-overlay.dtbo
14 rk3588-lubancat-5-dsi1-800x1280-overlay.dtbo
15 rk3588-lubancat-5-edp-overlay.dtbo
16 rk3588-lubancat-5-hdmi0-overlay.dtbo
17 rk3588-lubancat-5-hdmi1-vp0-overlay.dtbo
18 rk3588-lubancat-5-hdmi1-vp2-overlay.dtbo
19 rk3588-lubancat-5-hdmi-8k-overlay.dtbo
20 rk3588-lubancat-5-hdmi-rx-overlay.dtbo
21 rk3588-lubancat-5io-hdmi-8k-overlay.dtbo
22 rk3588-lubancat-5io-hdmi-rx-overlay.dtbo
23 rk3588-lubancat-can0-m0-overlay.dtbo
```

24 rk3588-lubancat-can2-m0-overlay.dtbo
25 rk3588-lubancat-i2c2-m4-overlay.dtbo
26 rk3588-lubancat-i2c3-m0-overlay.dtbo
27 rk3588-lubancat-i2c3-m1-overlay.dtbo
28 rk3588-lubancat-i2c4-m3-overlay.dtbo
29 rk3588-lubancat-i2c5-m3-overlay.dtbo
30 rk3588-lubancat-i2c6-m3-overlay.dtbo
31 rk3588-lubancat-i2c8-m2-overlay.dtbo
32 rk3588-lubancat-msata-overlay.dtbo
33 rk3588-lubancat-pwm0-m2-overlay.dtbo
34 rk3588-lubancat-pwm10-m2-overlay.dtbo
35 rk3588-lubancat-pwm11-ir-m3-overlay.dtbo
36 rk3588-lubancat-pwm13-m1-overlay.dtbo
37 rk3588-lubancat-pwm13-m2-overlay.dtbo
38 rk3588-lubancat-pwm14-m1-overlay.dtbo
39 rk3588-lubancat-pwm14-m2-overlay.dtbo
40 rk3588-lubancat-pwm15-ir-m3-overlay.dtbo
41 rk3588-lubancat-pwm1-m2-overlay.dtbo
42 rk3588-lubancat-pwm3-ir-m3-overlay.dtbo
43 rk3588-lubancat-spi0-m1-overlay.dtbo
44 rk3588-lubancat-spi0-m2-overlay.dtbo
45 rk3588-lubancat-uart0-m2-overlay.dtbo
46 rk3588-lubancat-uart1-m1-overlay.dtbo
47 rk3588-lubancat-uart3-m0-overlay.dtbo
48 rk3588-lubancat-uart4-m2-overlay.dtbo
49 rk3588-lubancat-uart6-m1-overlay.dtbo
50 rk3588-lubancat-uart7-m1-overlay.dtbo
51 rk3588-lubancat-uart7-m2-overlay.dtbo
52 rk3588-lubancat-uart9-m2-overlay.dtbo
53 rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
54 rk3588s-lubancat-4-cam0-ov8858-overlay.dtbo
55 rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
56 rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
57 rk3588s-lubancat-4-dp0-in-vp2-overlay.dtbo
58 rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo

```
59 rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
60 rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
61 rk3588s-lubancat-4-dsi1-1024x600-overlay.dtbo
62 rk3588s-lubancat-4-dsi1-1080p-overlay.dtbo
63 rk3588s-lubancat-4-dsi1-800x1280-overlay.dtbo
64 rk3588s-lubancat-4-hdmi0-8k-overlay.dtbo
65 rk3588s-lubancat-4-v1-dsi0-1024x600-overlay.dtbo
66 rk3588s-lubancat-4-v1-dsi0-1080p-overlay.dtbo
67 rk3588s-lubancat-4-v1-dsi0-800x1280-overlay.dtbo
68 rk3588s-lubancat-4-v1-dsi1-1024x600-overlay.dtbo
69 rk3588s-lubancat-4-v1-dsi1-1080p-overlay.dtbo
70 rk3588s-lubancat-4-v1-dsi1-800x1280-overlay.dtbo
71# これらは LubanCat-RK シリーズのすべてのボードのデバイスツリープラグインです。
```

デバイスツリープラグインを設定する方法は、設定ファイルを変更することです。設定ファイルは/boot/uEnv 内にあります。前のセクションで説明した内容に従ってください。

10.3.1.1 設定ファイルの変更

```
1 # ボードの設定ファイルを変更する
2 sudo vi /boot/uEnv/uEnv.txt
3
4 # または（xxx はボードの実際の設定ファイル）
5 sudo vi /boot/uEnv/uEnvxxx.txt
6
7 # vim で設定ファイルを開いた内容は以下の通りです
8 uname_r=5.10.110
9 size=0x1000000
10 cmdline="earlyprintk console=ttyFIQ0 console=tty1 consoleblank=0 loglevel=7 rootwait rw rootfstype=ext4"
11 enable_uboot_overlays=1
12
13 #overlay_start
14
15 dtoverlay=/dtb/overlay/rk3588-lubancat-i2c2-m4-overlay.dtbo
```

```
16 #dtoverlay=/dtb/overlay/rk3588-lubancat-i2c3-m1-overlay.dtbo
17 dtoverlay=/dtb/overlay/rk3588-lubancat-i2c5-m3-overlay.dtbo
18 #dtoverlay=/dtb/overlay/rk3588-lubancat-i2c6-m3-overlay.dtbo
19 dtoverlay=/dtb/overlay/rk3588-lubancat-i2c8-m2-overlay.dtbo
20
21 #overlay_end
```

- デバイスツリープラグインを有効にするには、対応するデバイスツリープラグインのコメントを削除します（#を削除）。

- デバイスツリープラグインを無効にするには、対応するデバイスツリープラグインにコメントを追加します（行頭に#を追加）。

- 上記の設定ファイルでは、i2c2-m4、i2c5-m3、i2c8-m2 はコメントされていないため、有効な状態です。i2c3-m1、i2c6-m3 はコメントされているため、無効な状態です。

デバイスツリーおよびデバイスツリープラグインの詳細については、システム構築マニュアルを参照してください。

11 GCC コンパイラ

```
# コードの場所
lubancat_rk_code_storage/quick_start/hello
```

早期はプロセッサの性能が低く、ストレージ容量やコンパイル能力も不足していたため、交差コンパイルが一般的でした。しかし、交差コンパイルにはオフラインコンパイルができない、操作が煩雑、環境設定が複雑などの欠点があります。プロセッサは高性能で、開発ボード上でプログラムをコンパイルする時間が短いため、ボードに統合された GCC ソフトウェアを直接使用してコンパイルすることをお勧めします。これにより、ファイル転送にかかる時間を大幅に削減できます。

GCC コンパイラが付属しており、以下のコマンドで GCC のバージョンを確認できます。

```
1 # gcc コマンドを確認する
2 gcc -v
3
4 # gcc のインストールパスを確認する
5 which gcc
6
7 # gcc がない場合、インストールする
8 sudo apt update
9 sudo apt install gcc -y
```

以下の図のように、使用している gcc のバージョンは 9.4.0 です。gcc のバージョンはイメージやシステムによって異なる場合がありますが、コンパイルには影響しません。

```
cat@lubancat:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/aarch64-linux-gnu/9/lto-wrapper
Target: aarch64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 9.4.0-1ubuntu1~20.04.1' --with-bugur
l=file:///usr/share/doc/gcc-9/README.Bugs --enable-languages=c,ada,c++,go,d,fortran,objc,obj-c++,gm
2 --prefix=/usr --with-gcc-major-version-only --program-suffix=-9 --program-prefix=aarch64-linux-gn
u- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enab
le-threads=posix --libdir=/usr/lib --enable-nls --enable-clocale=gnu --enable-libstdcxx-debug --ena
ble-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disable-libqua
dmath --disable-libquadmath-support --enable-plugin --enable-default-pie --with-system-zlib --with-
target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --enable-fix-cortex-a53-843419 --d
isable-werror --enable-checking=release --build=aarch64-linux-gnu --host=aarch64-linux-gnu --target
=aarch64-linux-gnu
Thread model: posix
gcc version 9.4.0 (Ubuntu 9.4.0-1ubuntu1~20.04.1)
cat@lubancat:~$
```

11.1 Hello World!

コードの場所:

```
quick_start/hello/hello.c
```

コード:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("Hello World!¥n");
```

```
6    printf("meow!¥n");
7    return 0;
8 }
```

ソースコード hello.c:

```
1 # vi を使用して hello.c ファイルを作成する
2 vi hello.c

3 # 'i'または'a'を押して編集モードに入る
4 # コードを vim エディタにコピーする
5 # 'Esc'キーを押す
6 # ":wq"を入力して保存して終了する

7 # ソースコードを直接ボードにダウンロードしてからコンパイルすることもできます
8 # 以下のコマンドを入力してコンパイルする
9 gcc -o hello hello.c

10 # プログラムを実行する
11 ./hello
```

以下の図のように

11.2 VSCode での簡便なデバッグ開発

直接ボード上でデバッグやコンパイルを行う以外に、VSCode を使用してリモートでボードに SSH ログインし、開発を行う方法も提供します。

この記事は「SSH ログイン」を前提としています。SSH ログインができない場合、以下の VSCode リモート開発は正常に機能しません。

11.2.1 環境設定

まず、環境を設定します。静的 IP とパスワードなしでのログインを設定しておくと、後の開発で迅速に接続でき、開発効率が大幅に向上します。

11.2.1.1 VSCode の使用

VSCode のダウンロードは[公式サイト](https://code.visualstudio.com/)から行います。

VSCode の初心者向けチュートリアルは[こちら](https://blog.csdn.net/weixin_52212950/article/details/124693906)。

11.2.1.2 ボードの IP アドレスを確認する

```
1 # コマンドを実行して IP を確認する
2 ifconfig
```

以下の図のように、192.168.103.156 は LubanCat の IP アドレスです。

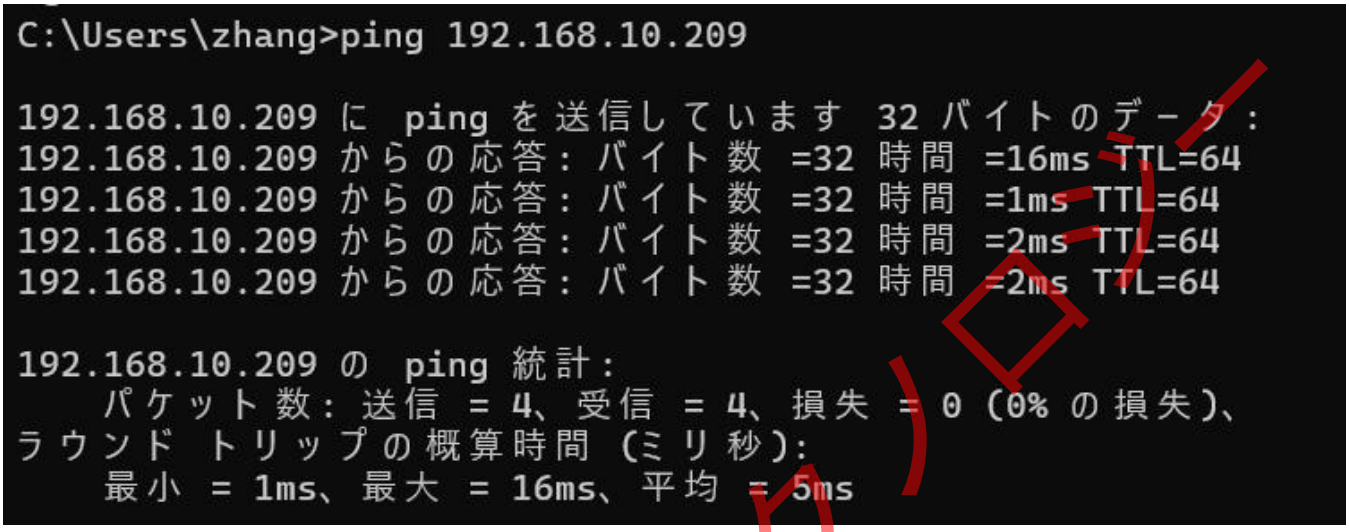
```
cat@lubancat:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.103.156  netmask 255.255.255.0  broadcast 192.168.103.255
        inet6 fe80::a093:53ff:fe6d:3c69  prefixlen 64  scopeid 0x20<link>
        ether a2:93:53:6d:3c:69  txqueuelen 1000  (Ethernet)
        RX packets 818406  bytes 133075412 (133.0 MB)
        RX errors 0  dropped 13533  overruns 0  frame 0
        TX packets 996749  bytes 1086875595 (1.0 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 39

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 179775  bytes 6457137183 (6.4 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 179775  bytes 6457137183 (6.4 GB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

cat@lubancat:~$
```

11.2.1.3 VSCode でボードに接続する

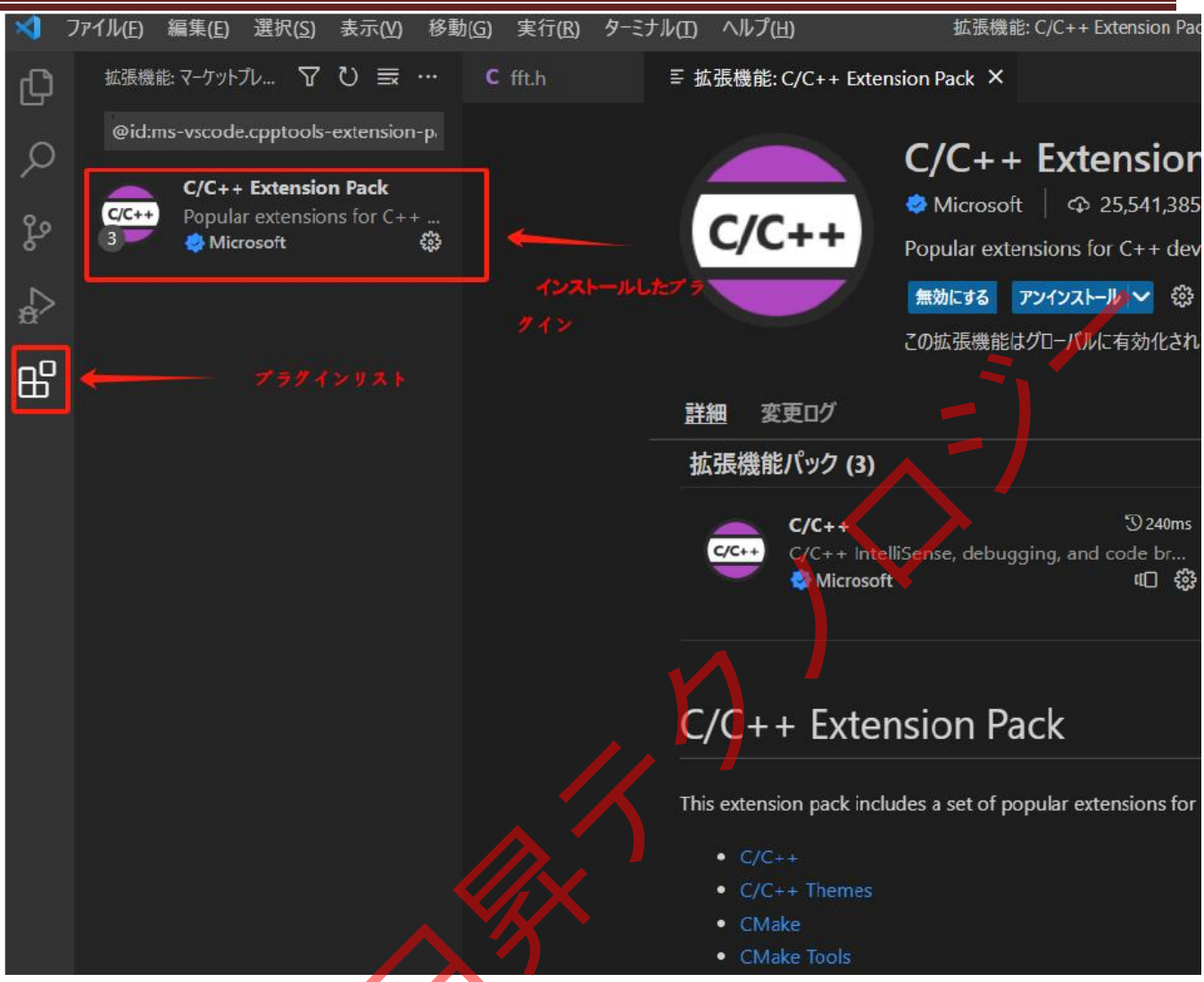
ボードに接続する前に、ボードとホスト間のネットワークが正常に通信できることを確認します。PC でコマンドプロンプトまたは PowerShell を開きます。

1 # ボードに ping を実行する
2 ping <ip>
3 # <ip>にはボードの IP アドレスを入力します

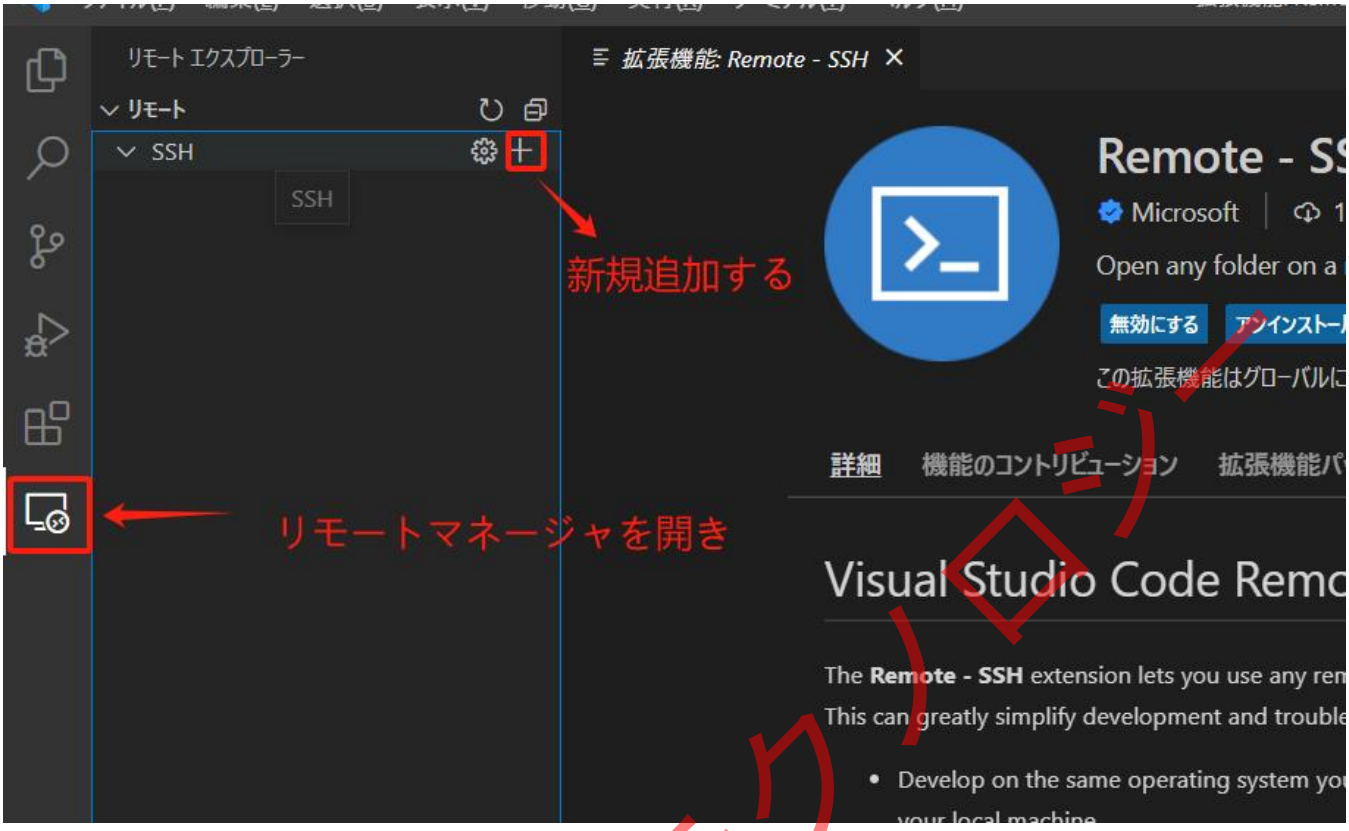
以下の図のように、ボードとホスト間で正常に通信できます。



次に、インストールした VSCode を開き、基本操作を設定した後、左側の拡張機能をクリックし、Remote-SSH プラグインをダウンロードします。



以下の図のように、インストールが完了したら、SSH リモート接続でボードに接続できます。



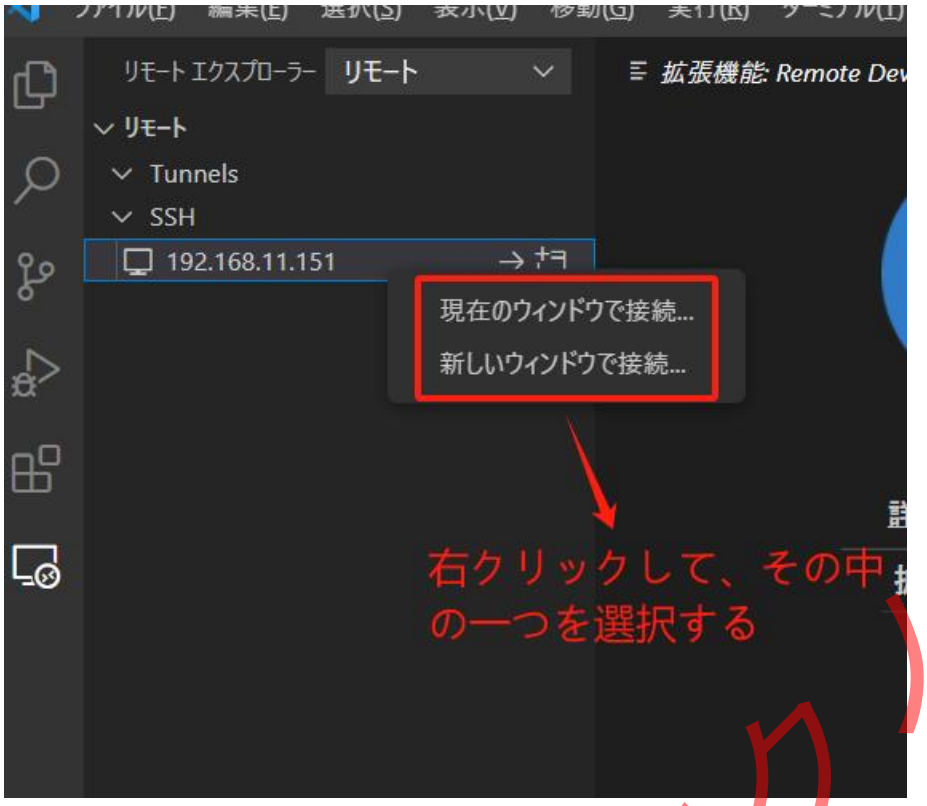
以下の図のように、上部に入力を求める内容が表示されます。

```
1 # 192.168.xxx.xxx はボードの IP アドレス
2 ssh cat@192.168.xxx.xxx

3 # Enter キーを押して接続する

4 # 最初のオプションを選んで設定を保存する
```

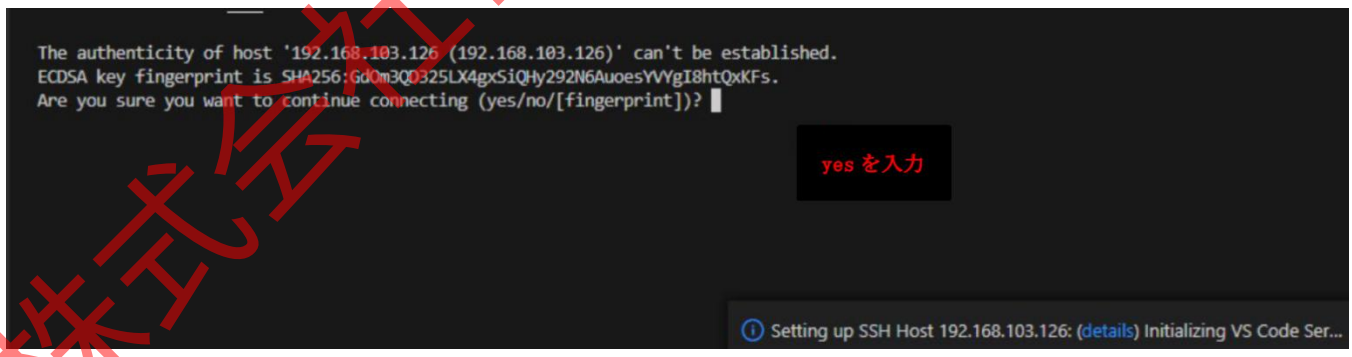
注意: IP アドレスでの直接接続以外に、ホスト名でボードに接続することもできます。同じローカルネットワーク内にボードが 1 台だけの場合、以下のコマンドを使用して接続できます。

```
ssh cat@lubancat
```

以下の図のように、ボードを開きます。



次に、VSCode は Linux、Windows、Mac を選択するように求めます。ここでは Linux を選択します。パスワードの入力が必要な場合は、パスワードを入力します。初めてログインする場合は、VSCode がログインを確認するように求めますので、ターミナルで「yes」と入力します。次に、ファイルを開いてプロジェクトを選択するか、他の操作を行います。



11.2.1.4 パスワードなしログイン

VSCode で毎回ログインするのが面倒な場合は、以下の方法を使用して解決できます。

```
# PC のコマンドプロンプトまたは PowerShell を開く
# 以下を入力
 ssh-keygen -t rsa
```

Enter キーを連続して押して終了する

.ssh フォルダを見つけて、id_rsa_pub ファイルを開く

以下の図のように、メモ帳で開くことができます。開いたら内容をコピーします。

ボードに接続する
cat ユーザーとしてログインする
コマンドを入力
ssh-keygen -t rsa

Enter キーを連続して押して終了する
/home/cat ディレクトリに.ssh フォルダが生成される

```
cat@lubancat:~$ ls -ag
total 56
drwxr-xr-x 6 cat  4096 Aug 19 13:41  .
drwxr-xr-x 3 root 4096 Aug 10 14:16  ..
-rw------- 1 cat    54 Apr 21 20:55  .Xauthority
-rw-r--r-- 1 cat  1600 Apr  9  2020  .Xdefaults
-rw-r--r-- 1 cat   220 Feb 25  2020  .bash_logout
-rw-r--r-- 1 cat  3771 Feb 25  2020  .bashrc
drwx------ 2 cat  4096 Apr 21 20:55  .cache
drwxr-xr-x 5 cat  4096 Apr 21 20:55  .config
-rw-r--r-- 1 cat   807 Feb 25  2020  .profile
drwx------ 2 cat  4096 Aug 19 13:41  .ssh
-rw------- 1 cat   870 Aug 19 13:41  .viminfo
drwxrwxr-x 5 cat  4096 Aug 19 12:00  .vscode-server
-rw-rw-r-- 1 cat   183 Aug 19 12:00  .wget-hsts
-rw-r--r-- 1 cat    14 Apr  9  2020  .xscreensaver
-rw-rw-r-- 1 cat     0 Aug 18 15:48 '01'$'\344\270\203\351\207\214\351\246\231''.mp3'
-rw-rw-r-- 1 cat     0 Aug 18 15:48 '02'$'\344\270\203\351\207\214\351\246\231''.wav'
-rw-rw-r-- 1 cat     0 Aug 18 15:48 '03'$'\344\270\203\351\207\214\351\246\231''.flac'
cat@lubancat:~$
```

これでパスワードなしログインが可能になります。

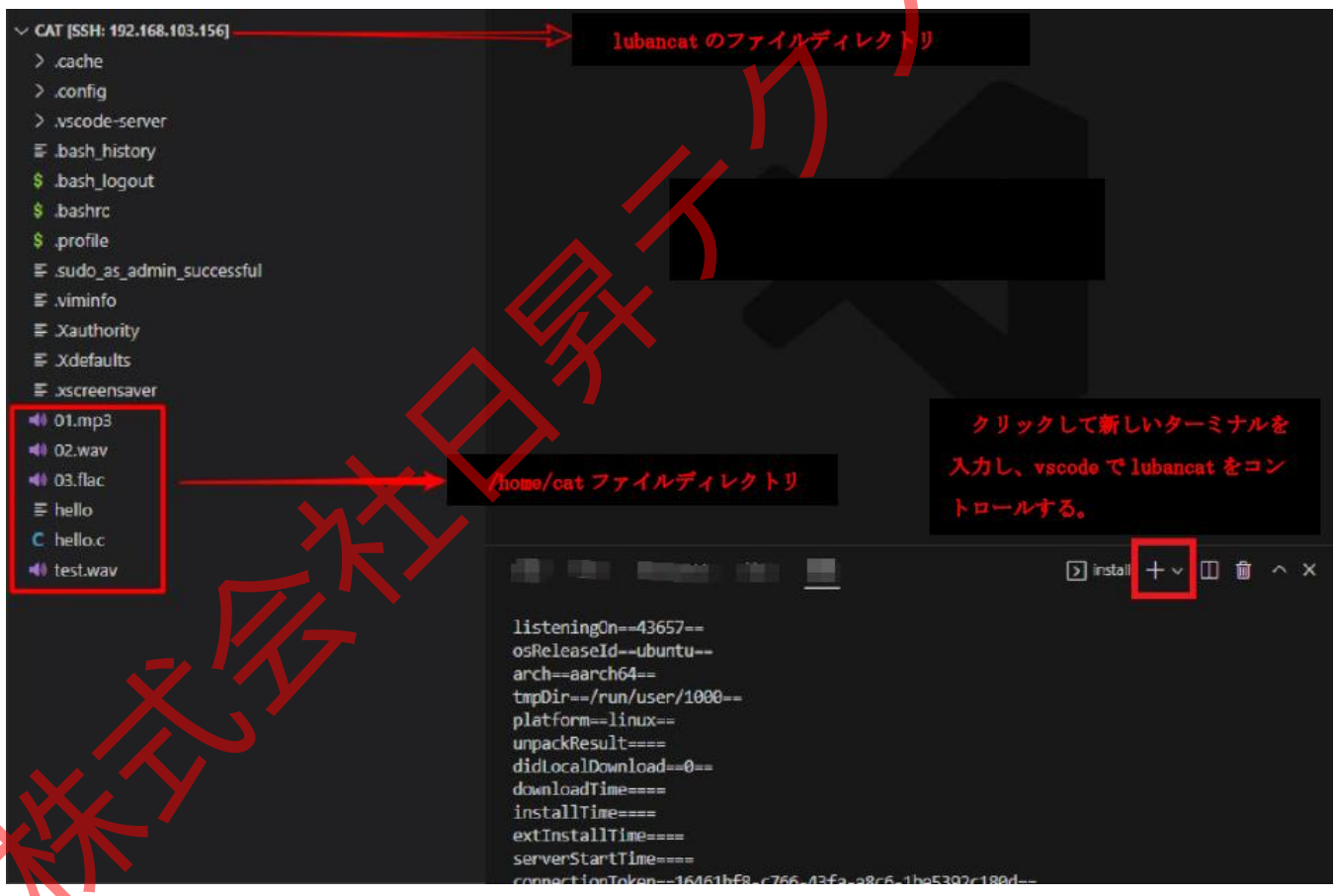
```
cat@lubancat:~/.ssh$ ls -ag
total 16
drwx------ 2 cat 4096 Aug 19 13:58 .
drwxr-xr-x 6 cat 4096 Aug 19 13:58 ..
-rw------- 1 cat 2602 Aug 19 13:40 id_rsa
-rw-r--r-- 1 cat  566 Aug 19 13:40 id_rsa.pub
```

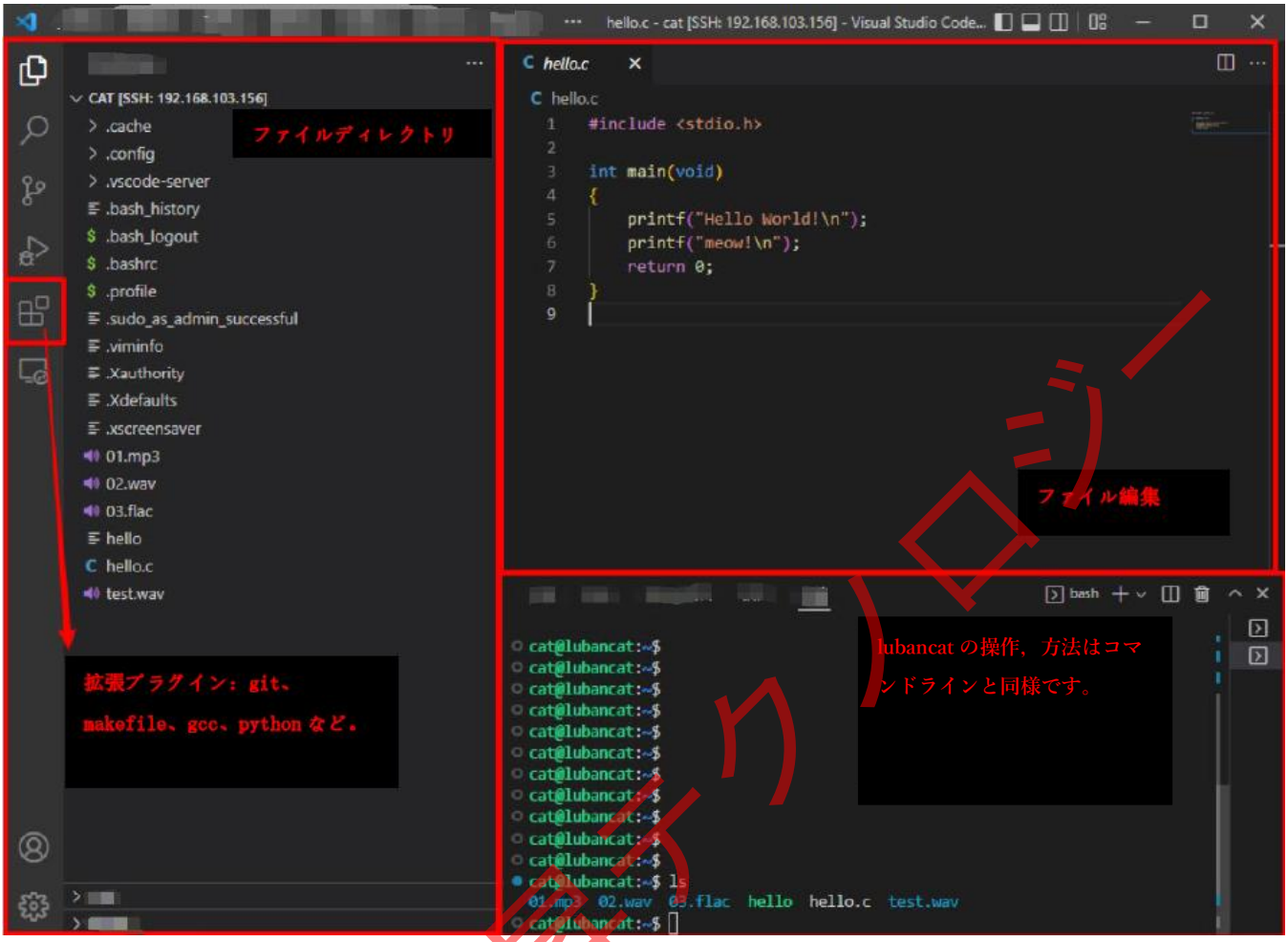
authorized_keys ファイルを作成する
vi authorized_keys

メモ帳からコピーした内容を貼り付けて保存して終了する

11.2.1.5 ボードに接続する

機能の紹介は以下の図の通りです。コンパイルと実行の操作を行います。





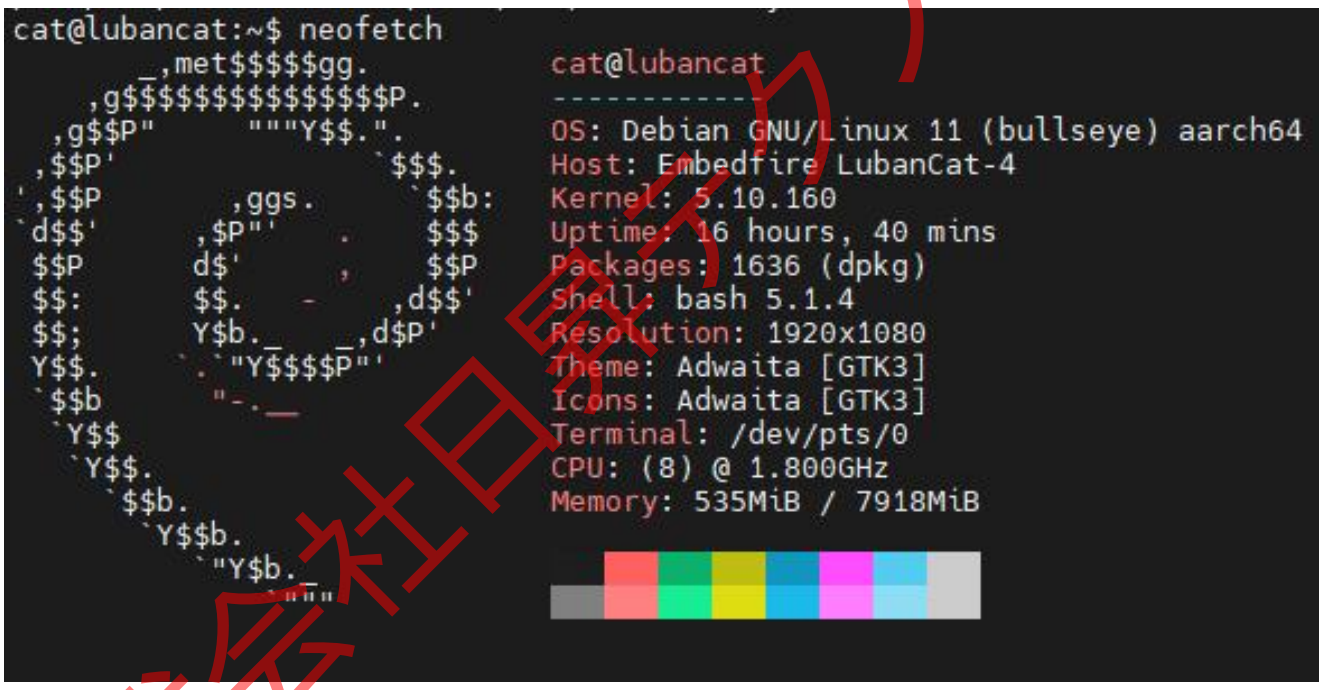
上記の内容は、VSCode を使用した便利な開発チュートリアルです。C 言語、C++、Python、Java、Go などのさまざまな言語に対応するプラグインが VSCode に用意されており、開発をサポートします。VSCode には、さらに多くの機能がありますので、ぜひ探索してみてください。

12 ボード情報の確認

12.1 総合情報の確認

```
# ツール neofetch のインストール
  sudo apt update
  sudo apt install neofetch

コマンドを実行
  neofetch
```

- LubanCat-4 Debian11：



上記の図から、現在のシステム状況を分析できます（LubanCat-4 を例にとります）。

1. **OS**：aarch64 アーキテクチャの Debian11 のルートファイルシステム

2. **Host**：ボード名は LubanCat-4

3. **Kernel**：5.10.160 の Linux カーネル

4. **Uptime**：稼働時間は 16 時間 40 分

5. **Packages**：1636 個のインストール済みパッケージ

6. **Shell**：bash 5.1.4 を使用

7. **Resolution**：画面解像度は 1920x1080

8. **CPU**：8 コア（big.LITTLE アーキテクチャのため、実際の CPU クロックは表示されません）

9. **Memory**：合計 7918MiB（8GB）、現在使用中 535MiB

12.2 ファイルシステムの確認

```
# コマンドを使用
   df -h

# 実行結果
   root@lubancat:~# df -h
   ファイルシステム 容量 使用 可用 使用% マウントポイント
   udev 3.9G 8.0K 3.9G 1% /dev
   tmpfs 792M 1.8M 791M 1% /run
   /dev/mmcblk0p3 58G 4.2G 51G 8% /
   tmpfs 3.9G 0 3.9G 0% /dev/shm
   tmpfs 5.0M 4.0K 5.0M 1% /run/lock
   tmpfs 3.9G 8.0K 3.9G 1% /tmp
   /dev/mmcblk0p2 124M 56M 62M 48% /boot
   tmpfs 792M 60K 792M 1% /run/user/1000
   tmpfs 792M 44K 792M 1% /run/user/0
   root@lubancat:~#
```

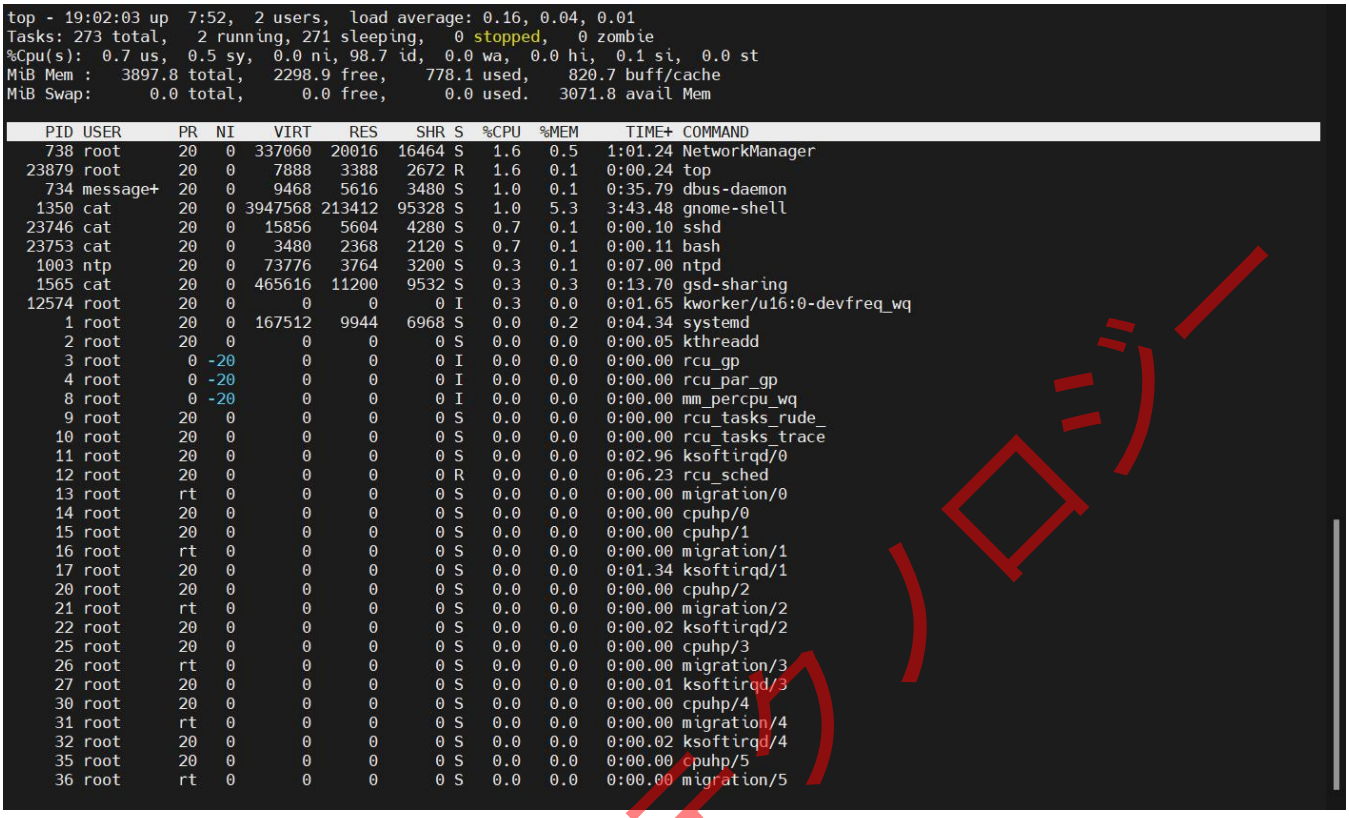
12.3 監視ツール

12.3.1 top

ボードに付属しているプロセス監視ツールで、基本的なプロセス情報を確認できます。

```
# コマンドを使用
```

```
top
```

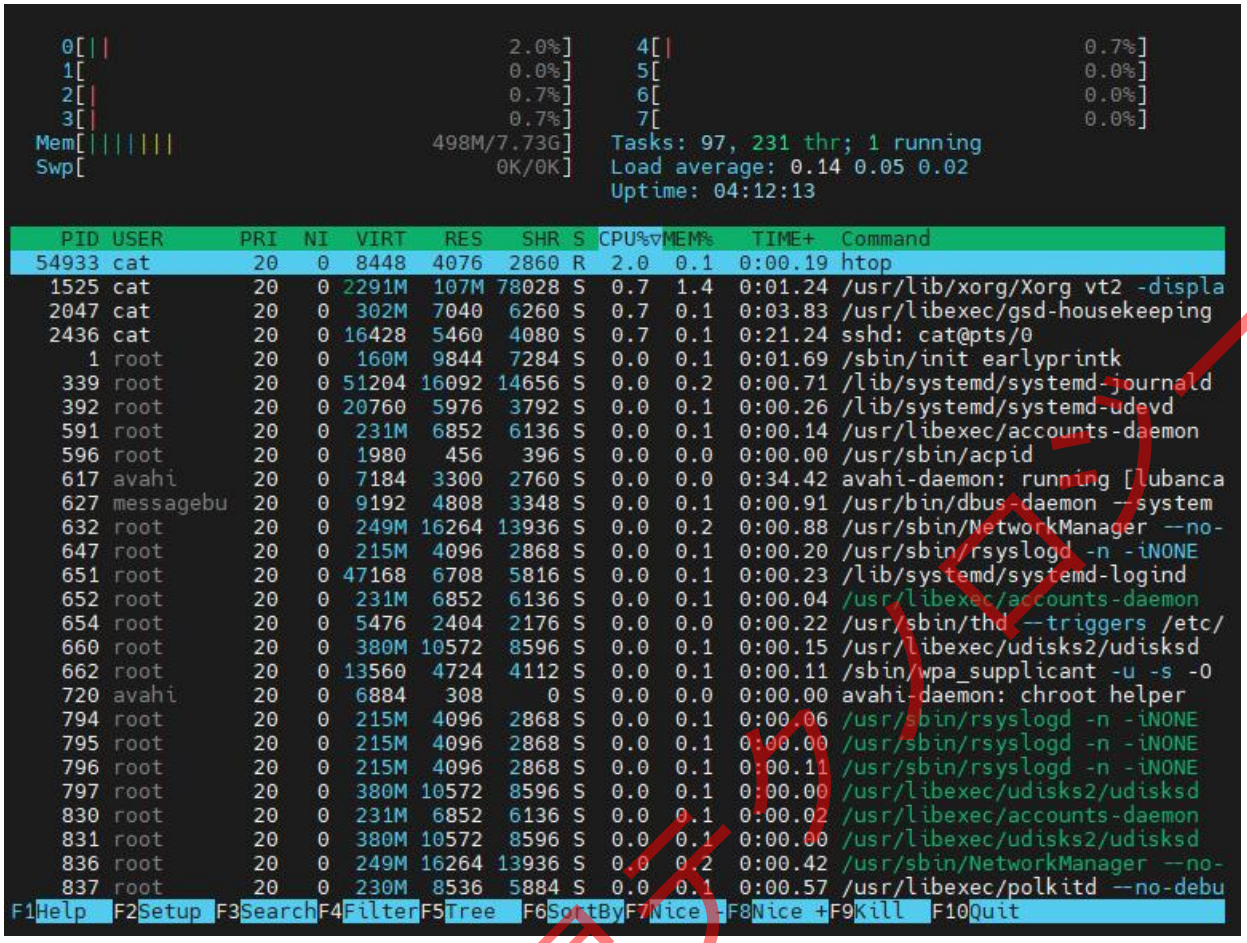


```
top - 19:02:03 up  7:52,  2 users,  load average: 0.16, 0.04, 0.01
Tasks: 273 total,   2 running, 271 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.5 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :   3897.8 total,   2298.9 free,    778.1 used,    820.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used.   3071.8 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    738 root      20   0  337060  20016  16464 S   1.6   0.5   1:01.24 NetworkManager
  23879 root      20   0    7888   3388   2672 R   1.6   0.1   0:00.24 top
    734 message+  20   0    9468   5616   3480 S   1.0   0.1   0:35.79 dbus-daemon
   1350 cat       20   0 3947568 213412  95328 S   1.0   5.3   3:43.48 gnome-shell
  23746 cat       20   0   15856   5604   4280 S   0.7   0.1   0:00.10 sshd
  23753 cat       20   0    3480   2368   2120 S   0.7   0.1   0:00.11 bash
   1003 ntp       20   0   73776   3764   3200 S   0.3   0.1   0:07.00 ntpd
   1565 cat       20   0  465616  11200   9532 S   0.3   0.3   0:13.70 gsd-sharing
  12574 root      20   0       0      0      0 I   0.3   0.0   0:01.65 kworker/u16:0-devfreq_wq
      1 root      20   0  167512   9944   6968 S   0.0   0.2   0:04.34 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.05 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
      9 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rude_
     10 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_trace
     11 root      20   0       0      0      0 S   0.0   0.0   0:02.96 ksoftirqd/0
     12 root      20   0       0      0      0 R   0.0   0.0   0:06.23 rcu_sched
     13 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     16 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/1
     17 root      20   0       0      0      0 S   0.0   0.0   0:01.34 ksoftirqd/1
     20 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/2
     21 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/2
     22 root      20   0       0      0      0 S   0.0   0.0   0:00.02 ksoftirqd/2
     25 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/3
     26 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/3
     27 root      20   0       0      0      0 S   0.0   0.0   0:00.01 ksoftirqd/3
     30 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/4
     31 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/4
     32 root      20   0       0      0      0 S   0.0   0.0   0:00.02 ksoftirqd/4
     35 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/5
     36 root      rt   0       0      0      0 S   0.0   0.0   0:00.00 migration/5
```

12.3.2 htop

htop は Linux システムのインタラクティブなプロセスビューアで、テキストモードアプリケーションです。top よりもユーザーフレンドリーで、インタラクティブな操作やカラーテーマ、プロセスリストの横方向または縦方向のスクロールをサポートし、マウス操作が可能です。

```
# コマンドを使用
  htop
```



12.3.3 btop

btop はクールなパフォーマンステスト監視分析ツールです。機能一覧には次のものがあります。

- ゲーム風のメニューシステム
- フルマウスサポート
- 矢印キーでプロセスを選択
- 選択したプロセスの詳細な統計情報
- プロセスフィルター
- カテゴリ間の簡単な切り替え
- 選択したプロセスに信号を送信
- メニュー経由での設定

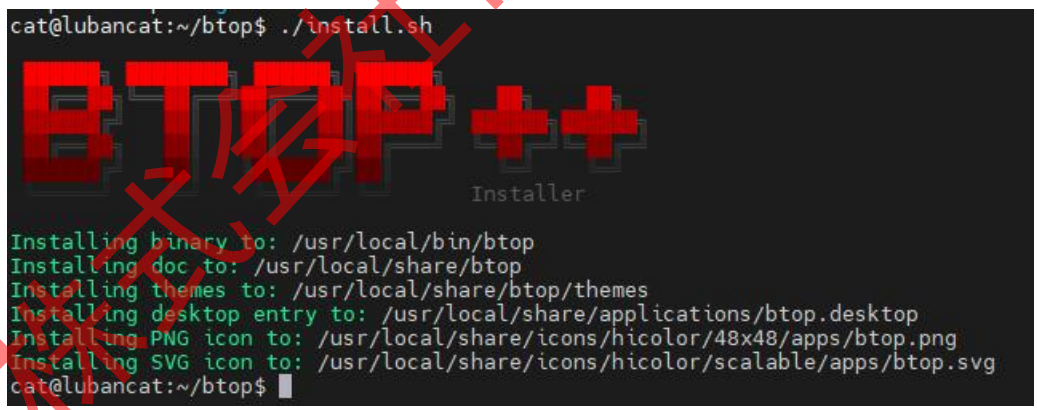
- ネットワーク I/O の自動スケーリンググラフ

- ディスクの IO 活動/速度の表示

- バッテリーメーター

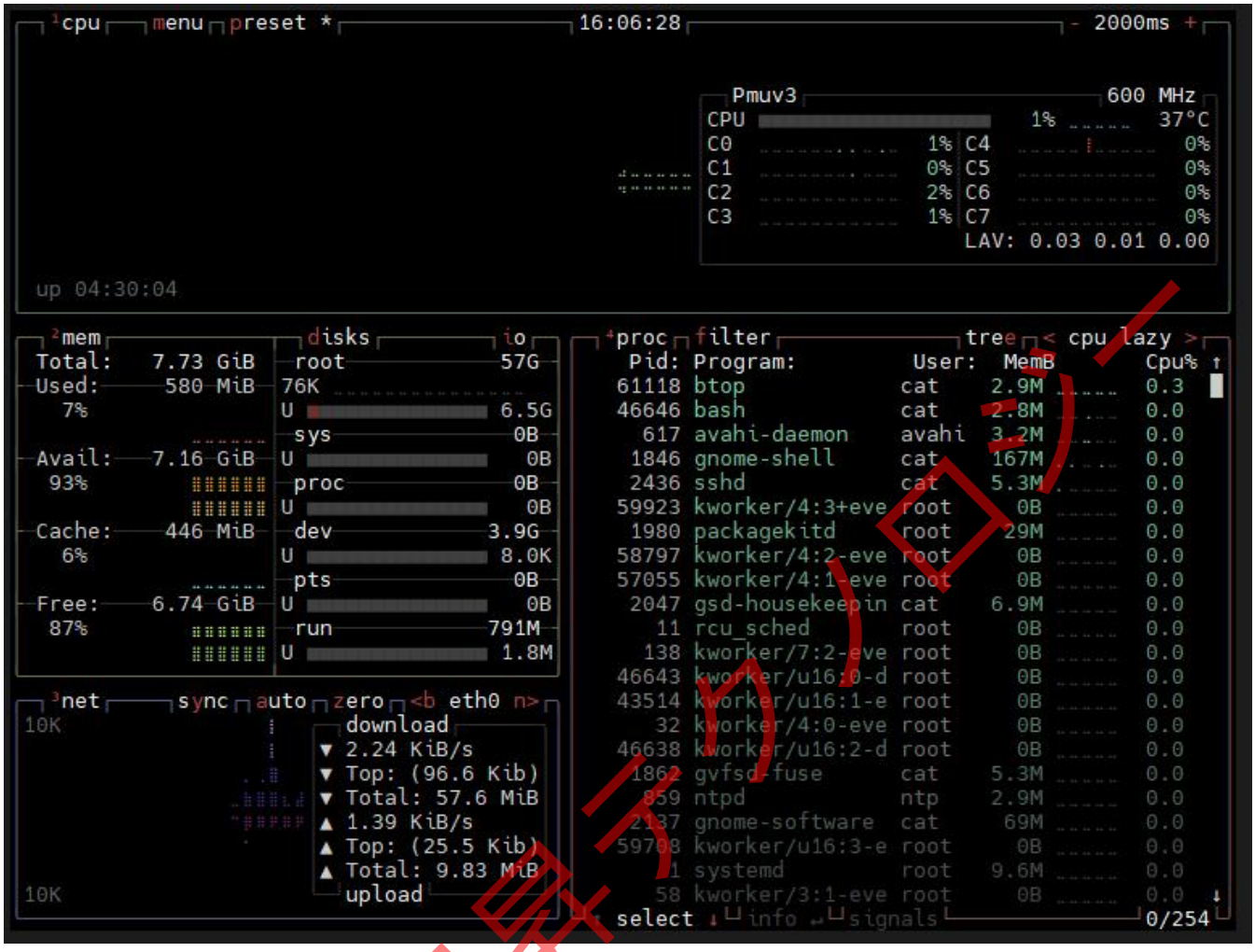
- カスタムプリセット

```
# btop のインストール

# ソースパッケージをダウンロード
wget https://github.com/aristocratos/btop/releases/download/v1.2.13/btop-aarch64-linux-musl.tbz

# ソースパッケージを解凍
  tar xf btop-aarch64-linux-musl.tbz

# ディレクトリを切り替え
  cd btop

# btop をインストール
  ./install.sh
```



注意: btop の実行には、最小で幅 80 文字、高さ 24 文字のターミナルまたはウィンドウが必要です。

```
# コマンドを使用
  btop
```



12.4 CPU 情報の確認

/proc/cpuinfo ファイルには CPU の情報が保存されています。以下のコマンドで確認できます。

```
# コマンドを使用
   cat /proc/cpuinfo
# 実行結果
```

```
cat@lubancat:~$ cat /proc/cpuinfo
processor : 0
BogoMIPS : 48.00
Features : fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant : 0x2
CPU part : 0xd05
CPU revision : 0

Serial : cee3557a7891ec2b
```

```
//ARM_CPU_PART の定義**
#define ARM_CPU_PART_AEM_V8 0xD0F
#define ARM_CPU_PART_FOUNDATION 0xD00
#define ARM_CPU_PART_CORTEX_A57 0xD07
#define ARM_CPU_PART_CORTEX_A72 0xD08
#define ARM_CPU_PART_CORTEX_A53 0xD03
#define ARM_CPU_PART_CORTEX_A73 0xD09
#define ARM_CPU_PART_CORTEX_A75 0xD0A
#define ARM_CPU_PART_CORTEX_A35 0xD04
#define ARM_CPU_PART_CORTEX_A55 0xD05
#define ARM_CPU_PART_CORTEX_A76 0xD0B
#define ARM_CPU_PART_NEOVERSE_N1 0xD0C
#define ARM_CPU_PART_CORTEX_A77 0xD0D
#define ARM_CPU_PART_NEOVERSE_V1 0xD40
#define ARM_CPU_PART_CORTEX_A78 0xD41
#define ARM_CPU_PART_CORTEX_A78AE 0xD42
#define ARM_CPU_PART_CORTEX_X1 0xD44
#define ARM_CPU_PART_CORTEX_A510 0xD46
#define ARM_CPU_PART_CORTEX_A710 0xD47
#define ARM_CPU_PART_CORTEX_X2 0xD48
#define ARM_CPU_PART_NEOVERSE_N2 0xD49
```

```
#define ARM_CPU_PART_CORTEX_A78C 0xD4B
```

上記の情報から、LubanCat-4 の CPU には 8 つのコアがあることがわかります。

- コア 0-3 は 0xD05 で、CORTEX_A55 であることがわかります。

- コア 4-7 は 0xD0B で、CORTEX_A76 であることがわかります。

- RK3588 シリーズのチップは 4xCORTEX_A76 + 4xCORTEX_A55 で構成されています。

12.5 SOC クロックの確認

CPU の周波数を確認する

デフォルトでは可変周波数モードになっており、周波数は現在のアプリケーションの使用状況に応じて変化します。

```
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_cur_freq

# 実行結果（小コア 1.2GHz、大コア 408MHz）

cat@lubancat:~$ cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_cur_freq
1200000
1200000
1200000
1200000
408000
408000
408000
408000
cat@lubancat:~$
```

GPU クロックの確認

デフォルトでは可変周波数モードになっており、周波数は現在のアプリケーションの使用状況に応じて変化します。

```
cat /sys/class/devfreq/fb000000.gpu/cur_freq
```

```
# 実行結果（GPU のクロックは 300MHz）
cat@lubancat:~$ cat /sys/class/devfreq/fb000000.gpu/cur_freq
300000000
cat@lubancat:~$
```

NPU クロックの確認

デフォルトでは可変周波数モードになっており、周波数は現在のアプリケーションの使用状況に応じて変化します。

```
sudo cat /sys/kernel/debug/clk/clk_summary | grep clk_npu_dsu0

# 実行結果（NPU のクロックは 750MHz）
cat@lubancat:~$ sudo cat /sys/kernel/debug/clk/clk_summary | grep clk_npu_dsu0
clk_npu_dsu0 0 3 0 750000000 0
cat@lubancat:~$
```

DDR クロックの確認

デフォルトでは可変周波数モードになっており、周波数は現在のアプリケーションの使用状況に応じて変化します。

```
cat /sys/class/devfreq/dmc/cur_freq
# 実行結果（DDR のクロックは 528MHz）
cat@lubancat:~$ cat /sys/class/devfreq/dmc/cur_freq
528000000
cat@lubancat:~$
```

12.6 SOC 温度の確認

LubanCat-4 は Thermal を使用してチップの温度を監視します。

```
# thermal のすべてのデバイスをリストする
ls /sys/class/thermal

# 実行結果
```

```
root@lubancat:~# ls /sys/class/thermal/
cooling_device0 cooling_device3 thermal_zone2 thermal_zone5
cooling_device1 thermal_zone0 thermal_zone3 thermal_zone6
cooling_device2 thermal_zone1 thermal_zone4
```

cooling_device は、SOC が温度制御に達したとき、SOC が自身のデバイスの消費電力を減少させるために使用されます。

1. cooling_device0：SOC の 4 つの小コアのクロックダウンデバイス
2. cooling_device1：SOC の 2 つの大コアのクロックダウンデバイス
3. cooling_device2：SOC の別の 2 つの大コアのクロックダウンデバイス
4. cooling_device3：SOC の GPU のクロックダウンデバイス

thermal_zone は SOC 内部の温度監視ポイントです。

1. thermal_zone0：SOC の主要な温度監視。cooling_device の起動と停止はこのデバイスに依存します。
2. thermal_zone1：SOC の 2 つの大コアの温度監視
3. thermal_zone2：SOC の別の 2 つの大コアの温度監視
4. thermal_zone3：SOC の 4 つの小コアの温度監視
5. thermal_zone4：チップの中心温度監視
6. thermal_zone5：SOC の GPU の温度監視
7. thermal_zone6：SOC の NPU の温度監視

SOC の主要な温度監視を確認**

```
cat /sys/class/thermal/thermal_zone0/temp
```

他の温度監視ポイントも同様に確認できます。

12.7 SOC スケジューリング

12.7.1 CPU スケジューリングポリシー（クラスター）

ここでは、LubanCat-4 のクラスターのスケジューリングポリシーを説明します。

LubanCat-4 の CPU スケジューリングポリシー（クラスター）は、各コアを個別に制御するのではなく、8 つのコアを 3 つのクラスターに分けて制御します。

```
# クラスターを表示
ls /sys/devices/system/cpu/cpufreq/
policy0 policy4 policy6
```

- 4 つの CORTEX_A55 コア – policy0

- 2 つの CORTEX_A76 コア – policy4

- 2 つの CORTEX_A76 コア – policy6

これにより、クラスターを制御することで SOC の全体的なパフォーマンスを制御できます。

クラスターの固定周波数設定は簡単です。

```
# 現在の SOC のスケジューリングポリシーを確認
  cat /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
# 実行結果（現在の SOC の調頻ポリシーは ondemand）
    cat@lubancat:~$ cat /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
    ondemand
    cat@lubancat:~$


# SOC のすべてのスケジューリングポリシーを確認
  cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_governors


  # 実行結果（7 種類）
```

```
cat@lubancat:~$ cat
/sys/devices/system/cpu/cpufreq/policy0/scaling_available_governors
interactive conservative ondemand userspace powersave performance schedutil
cat@lubancat:~$
```

- conservative：CPU 負荷に応じて動的にクロックを調整し、一定の割合で滑らかにクロックを上げたり下げたりします。

- ondemand：CPU 負荷に応じて動的にクロックを調整し、大きな幅で調整します。最大クロックまたは最小クロックに直接設定します。

- interactive：CPU 負荷に応じて動的にクロックを調整し、ondemand よりも応答時間が速く、設定項目が多いため柔軟に設定できます。

- userspace：ユーザー空間のアプリケーションがクロックを調整するためのインターフェースを提供します。

- powersave：消費電力優先で、常にクロックを最低値に設定します。

- performance：パフォーマンス優先で、常にクロックを最高値に設定します。

- schedutil：EAS（Energy Aware Scheduling）を使用する governor です。EAS は次世代のタスクスケジューリングポリシーで、CPUFreq と CPUIdle のポリシーを組み合わせ、タスクの実行 CPU を選択する際にパフォーマンスと消費電力の両方を考慮します。Schedutil は EAS 用の CPU 調頻ポリシーです。

クラスターのスケジューリングポリシー設定（policy0 を例として）**

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# サポートされているスケジューリングポリシーを確認
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_governors
```

```
# 必要なスケジューリングポリシーに設定（例として performance）
echo performance > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor
```

クラスターの固定周波数ポリシー設定（policy0 を例として）

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# ポリシーを "userspace" に変更
echo userspace > /sys/devices/system/cpu/cpufreq/policy0/scaling_governor

# CPU の利用可能なクロックを確認
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies

# 実行結果（8 種類）
cat@lubancat:~$ cat
/sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
408000 600000 816000 1008000 1200000 1416000 1608000 1800000
cat@lubancat:~$

# クロックを設定（600MHz に設定）
echo 600000 > /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
```

12.7.2 GPU スケジューリングポリシー

```
# 現在の GPU のスケジューリングポリシーを確認
  cat /sys/class/devfreq/fb000000.gpu/governor

# サポートされている GPU のスケジューリングポリシーを確認
  cat /sys/class/devfreq/fb000000.gpu/available_governors
```

```
# 実行結果（5 種類）
    cat@lubancat:~$ cat /sys/class/devfreq/fb000000.gpu/available_governors
    dmc_ondemand userspace powersave performance simple_ondemand
    cat@lubancat:~$
```

- simple_ondemand：負荷に応じて動的にクロックを調整。

- userspace：ユーザー空間のアプリケーションがクロックを調整するためのインターフェースを提供。

- powersave：消費電力優先で、常にクロックを最低値に設定。

- performance：パフォーマンス優先で、常にクロックを最高値に設定。

- dmc_ondemand：simple_ondemand を基に、シーンベースの変頻をサポートし、DDR 変頻専用。

GPU のスケジューリングポリシー設定

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# 必要なスケジューリングポリシーに設定（例として performance）
echo performance > /sys/class/devfreq/fb000000.gpu/governor
```

GPU の固定周波数ポリシー設定

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# ポリシーを "userspace" に変更
echo userspace > /sys/class/devfreq/fb000000.gpu/governor

# GPU の利用可能なクロックを確認
cat /sys/class/devfreq/fb000000.gpu/available_frequencies
```

```
# 実行結果（8 種類）
cat@lubancat:~$ cat /sys/class/devfreq/fb000000.gpu/available_frequencies
1000000000 900000000 800000000 700000000 600000000 500000000 400000000
300000000
cat@lubancat:~$

# クロックを設定（400MHz に設定）
echo 400000000 > /sys/class/devfreq/fb000000.gpu/userspace/set_freq
```

12.7.3 DDR スケジューリング

```
# 現在の DDR のスケジューリングモードを確認
  cat /sys/class/devfreq/dmc/governor

# 実行結果
    cat@lubancat:~$ cat /sys/class/devfreq/dmc/governor
    dmc_ondemand
    cat@lubancat:~$

# DDR のサポートされているスケジューリングモードを確認**
  cat /sys/class/devfreq/dmc/available_governors

# 実行結果
    cat@lubancat:~$ cat /sys/class/devfreq/dmc/available_governors
    dmc_ondemand userspace powersave performance simple_ondemand
    cat@lubancat:~$
```

- simple_ondemand：負荷に応じて動的にクロックを調整。

- userspace：ユーザー空間のアプリケーションがクロックを調整するためのインターフェースを提供。

- powersave：消費電力優先で、常にクロックを最低値に設定。

- performance：パフォーマンス優先で、常にクロックを最高値に設定。

- dmc_ondemand：simple_ondemand を基に、シーンベースの変頻をサポートし、DDR 変頻専用。

DDR のスケジューリングポリシー設定

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# 必要なスケジューリングポリシーに設定（例として performance）
echo performance > /sys/class/devfreq/dmc/governor
```

DDR の固定周波数ポリシー設定

```
# root ユーザーで操作します
su root
# パスワードを入力（デフォルトは "root"）

# ポリシーを "userspace" に変更
echo userspace > /sys/class/devfreq/dmc/governor

# CPU の利用可能なクロックを確認
cat /sys/class/devfreq/dmc/available_frequencies

# 実行結果（4 種類）
cat@lubancat:~$ cat /sys/class/devfreq/dmc/available_frequencies
528000000 1068000000 1560000000 2112000000
cat@lubancat:~$

# クロックを設定（1560MHz に設定）
echo 1560000000 > /sys/class/devfreq/dmc/userspace/set_freq
```

12.7.4 NPU の固定周波数

手で userspace モードに切り替え

```
echo userspace > /sys/class/devfreq/fdab0000.npu/governor

# クロックを 1GHz に設定
echo 1000000000 > /sys/class/devfreq/fdab0000.npu/userspace/set_freq

# クロックが正常に設定されたか確認
cat /sys/class/devfreq/fdab0000.npu/cur_freq
```

注意: 上記の設定は一時的に有効であり、再起動すると無効になります。永久に固定する場合は、/etc/init.d/boot_init.sh の末尾に上記のコマンドを追加します。

13 RT-Linux

RT-Linux は、Linux カーネルをベースにしたリアルタイムオペレーティングシステムです。これにより、Linux の汎用性とリアルタイム性を兼ね備えたプラットフォームが提供され、開発者はリアルタイムアプリケーションを開発するための統一された環境を利用できます。

RT-Linux の核心は、Linux カーネルに対するリアルタイム拡張です。これにより、リアルタイムタスクに必要なスケジューリングメカニズムと時間管理が提供されます。RT-Linux では、優先度の高いリアルタイムタスクが優先度の低いタスクを中断できるプリエンプティブスケジューリングポリシーが採用されています。これにより、リアルタイムタスクがタイムリーに応答することが保証されます。RT-Linux はタスクのスケジューリングと割り込み処理を改良し、タスクが予定された時間内に実行されるようにしています。

RT-Linux は、時間に敏感なアプリケーション分野、例えば産業オートメーション、ロボット制御、航空宇宙システムなどに適しています。精密なタスクスケジューリングと高速な応答時間を提供することで、システムがリアルタイムタスクの要求に応じて正常に動作することを保証します。従来の Linux カーネルと比較して、RT-Linux はリアルタイム性能が向上していますが、完全なハードリアルタイムシステムではなく、タスクの実行時間が絶対に正確であることを保証するわけではありません。非常に高い時間的要求のあるアプリケーションには、より専門的なリアルタイムオペレーティングシステムが必要な場合があります。

RT-Linux は Linux オペレーティングシステムの汎用性を持ち、Linux エコシステムのさまざまなツールやライブラリを利用して開発を行うことができます。開発者は広範な開発ツールやリソースを使用してリアルタイムアプリケーションを構築することができ、開発効率が向上します。

注記: ボードのカーネルも RT-Linux をサポートしており、PREEMPT_RT パッチを使用してカーネルを変更しています。

13.1 RT-Linux のインストール方法

13.1.1 カーネルのオンライン更新

旧カーネルをアンインストールします。旧カーネルは RT-Linux と互換性がないため、アンインストールする必要があります。

```
# 旧カーネルをアンインストール
  sudo apt remove -y -f linux-headers-5.10.160 linux-image-5.10.160
```

RT-Linux カーネルを取得

```
sudo apt install linux-headers-5.10.160-rt89 linux-image-5.10.160-rt89
```

カーネルを有効にするために再起動

```
reboot
```

13.2 RT-Linux のテスト

13.2.1 cyclictest

Cyclictest は、Linux システムのリアルタイム性能をテストするツールであり、システムの応答時間とクロック精度を測定できます。Cyclictest は周期的な負荷を作成することで、リアルタイムタスク処理におけるシステムの能力を評価します。

注意: このテストは負荷が大きく、SSH ログインでは制御台にアクセスできなくなる可能性があるため、シリアルポートでの root ログインを推奨します。以下のコマンドはすべてシリアルポートでの root ログインを前提としています。

```
# ソフトウェアパッケージの更新
    apt update

# ツールのインストール
    apt install -y python git

# Cyclictest のソースコードをダウンロード
    git clone https://github.com/jlelli/rt-tests.git

# ソースコードディレクトリに移動
    cd rt-tests

# ソースコードをコンパイル（依存関係エラーが発生した場合は、必要な依存関係をインストールしてください）
    make -j4
```

```
# 負荷テスト（rt-tests ディレクトリ内で実行する必要があります）**
  ./cyclictest -t 4 -p 99 -n -m -d 0 & ./hackbench -l -1 -g 15 -f 25 -P &

# 空負荷テスト（rt-tests ディレクトリ内で実行する必要があります）**
  ./cyclictest -t 4 -p 99 -n -m -d 0

# /usr/bin にインストール（オプション）
  make install

# 負荷テスト（オプション、他のディレクトリでも実行可能）
  cyclictest -t 4 -p 99 -n -m -d 0 & hackbench -l -1 -g 15 -f 25 -P &

# 空負荷テスト（オプション、他のディレクトリでも実行可能）**
  cyclictest -t 4 -p 99 -n -m -d 0
```

hackbench -l -1 -g 15 -f 25 -P は負荷を作成するためのコマンドです。

- **-l -1**：hackbench が無限ループで実行され、自動的に終了しないことを示します。

- **-g 15**：各プロセスグループのプロセス数を 15 に設定します。各プロセスグループは通信を行うためのプロセスを生成します。

- **-f 25**：各プロセスグループ間の親子関係の数を 25 に設定します。これにより、プロセスグループ間の通信方法が決定されます。

- **-P**：複数の CPU コアを使用してテストを実行し、マルチコアシステムのシナリオをシミュレートします。

cyclictest -t 4 -p 99 -n -m -d 0

- **-t 4**：4 つのスレッドを使用してテストを実行します。各スレッドは周期的な負荷を作成します。

- **-p 99**：スレッドの優先度を 99 に設定します。これは最高のリアルタイム優先度です。最高優先度を使用することで、システムのリアルタイム性能をよりよくテストできます。

- **-n**：テスト期間中にスリープを禁止します。これにより、システムがアクティブな状態を保つことができます。

- **-m**：テストを実行する前にメモリをロックします。これにより、テスト中のメモリページングエラーを防ぎ、より一貫したテスト結果が得られます。

- **-d 0**：遅延出力を無効にします。これにより、各スレッドの詳細な遅延データが表示されず、テストの全体的な統計情報のみが表示されます。

14 40PIN 引出し PIN 紹介

LUBANCAT RK PIN 機能			
機能	PIN		機能
3.3V	1	2	5V
I2C3_SDA	3	4	5V
I2C3_SCL	5	6	GND
GPIO	7	8	UART_TX
GND	9	10	UART_RX
GPIO	11	12	PWM
GPIO	13	14	GND
GPIO	15	16	GPIO
3.3V	17	18	GPIO
MOSI	19	20	GND
MISO	21	22	GPIO
SCLK	23	24	CSO
GND	25	26	CSI
I2C5_SDA	27	28	I2C5_SCL
GPIO	29	30	GND
GPIO	31	32	PWM
PWM	33	34	GND
PWM	35	36	GPIO
GPIO	37	38	GPIO
GND	39	40	GPIO

LUBANCAT4 の PIN 設定

					3.3V	1	2	5V					
			I2C5_SDA_M3	47	GPIO1_B7	3	4	5V					
			I2C5_SCL_M3	46	GPIO1_B6	5	6	GND					
	I2C2_SDA_M4	UART6_RX_M1		32	GPIO1_A0	7	8	GPIO4_A3	131	UART0_TX_M2			
					GND	9	10	GPIO4_A4	132	UART0_RX_M2			
	I2C2_SCL_M4	UART6_TX_M1		33	GPIO1_A1	11	12	GPIO1_D6	62	PWM14_M2	I2C8_SCL_M2		
	PDM1_SDI0_M1	PWM3_IR_M3		39	GPIO1_A7	13	14	GND					
	PDM1_SDI1_M1			40	GPIO1_B0	15	16	GPIO3_C1	113			UART7_RX_M1	SPI1_CLK_M1
					3.3V	17	18	GPIO3_D2	122				UART9_RTSN_M2
	PDM1_SDI3_M1	UART4_RX_M2	SPI0_MOSI_M2	42	GPIO1_B2	19	20	GND					
	PDM1_SDI2_M1		SPI0_MISO_M2	41	GPIO1_B1	21	22	GPIO3_D4	124		UART9_RX_M2		
	PDM1_CLK1_M1	UART4_TX_M2	SPI0_CLK_M2	43	GPIO1_B3	23	24	GPIO1_B4	44	SPI0_CS0_M2	UART7_RX_M2	PDM1_CLK0_M1	
					GND	25	26	GPIO1_B5	45	SPI0_CS1_M2	UART7_TX_M2		
			I2C6_SDA_M3	136	GPIO4_B0	27	28	GPIO4_B1	137	I2C6_SCL_M3			
				102	GPIO3_A6	29	30	GND					
SPI1_MOSI_M1		I2C3_SCL_M1		111	GPIO3_B7	31	32	GPIO1_D7	63	PWM15_IR_M3	I2C8_SDA_M2		
UART9_CTSN_M2			PWM10_M2	123	GPIO3_D3	33	34	GND					
		UART9_TX_M2	PWM11_IR_M3	125	GPIO3_D5	35	36	GPIO4_A0	128		SPI0_MISO_M1		UART9_RTSN_M1
SPI1_MISO_M1	UART7_TX_M1	I2C3_SDA_M1		112	GPIO3_C0	37	38	GPIO4_A1	129		SPI0_MOSI_M1		UART9_CTSN_M1
					GND	39	40	GPIO4_A2	130		SPI0_CLK_M1		

15 GPIO 制御

GPIO とは、General Purpose I/O の略で、汎用入出力端子を指します。簡単に言うと、MCU/CPU が制御できるピンのことです。これらのピンは通常、複数の機能を持ち、基本的には高低電圧の入力検出と出力を行います。一部のピンは、シリアルポート、I2C、ネットワーク、電圧検出の通信ピンとして、メインコントローラのオンチップペリフェラルに接続されます。

Linux は GPIO サブシステムドライバフレームワークを提供しており、このドライバフレームワークを使用してボード上の GPIO を柔軟に制御できます。

15.1 GPIO の命名

Rockchip のピンの ID は、コントローラ（bank）＋ポート（port）＋インデックス番号（pin）で構成されます。

- コントローラと GPIO コントローラの数は一致します。

- ポートは固定で A、B、C、D があり、各ポートには 8 つのインデックス番号があります（a=0、b=1、c=2、d=3）。

- インデックス番号は 0、1、2、3、4、5、6、7 です。

15.2 GPIO sysfs インターフェースを使用して IO を制御

Linux では、最も一般的な GPIO の読み書き方法は GPIO sysfs インターフェースを使用することです。これは、/sys/class/gpio ディレクトリ内の export、unexport、gpio{N}/direction、gpio{N}/value（{N}は実際のピン番号）ファイルを操作することで実現されます。この方法は多くのシェルスクリプトで使用されます。カーネル 4.8 以降では libgpiod のサポートが追加されましたが、従来の sysfs ベースのアクセス方法は徐々に廃止される予定です。

例：GPIO の計算

PIN	コントローラ	ポート番号	インデックス番号	計算結果
GPIO1_C4	1	C	4	52 (32 x 1 + 8 x 2 + 4)
GPIO3_B2	3	B	2	106 (32 x 3 + 8 x 1 + 2)
GPIO0_D6	0	D	6	30 (32 x 0 + 8 x 3 + 6)

```
# 操作例（管理者権限が必要）
# GPIO1_C4 ピンを有効にする
echo 52 > /sys/class/gpio/export

# ピンを入力モードに設定
echo in > /sys/class/gpio/gpio52/direction

# ピンの値を読み取る
cat /sys/class/gpio/gpio52/value

# ピンを出力モードに設定
```

```
echo out > /sys/class/gpio/gpio52/direction

# ピンを低電圧に設定
echo 0 > /sys/class/gpio/gpio52/value

# ピンを高電圧に設定
echo 1 > /sys/class/gpio/gpio52/value

# ピンをリセット
echo 52 > /sys/class/gpio/unexport
```

15.3 libgpiod を使用して IO を制御

libgpiod はキャラクタデバイスインターフェースであり、GPIO のアクセス制御はキャラクタデバイスファイル（例：/dev/gpiodchip0）を操作することで実現されます。libgpiod は、いくつかのコマンドツール、C ライブラリ、Python ラッパーを提供します。

```
# gpiod コマンドラインツールのインストール
sudo apt install gpiod
```

gpiod ツールの使用方法は sysfs インターフェースとは異なり、gpiod はコントローラ単位で操作し、ポート番号とインデックス番号を指定します。以下に例を示します。

例：GPIO の計算

PIN	コントローラ	ポート番号	インデックス番号	gpiod の結果
GPIO1_C4	1	C	4	1 20(8 x 2 + 4)
GPIO3_B2	3	B	2	3 10(8 x 1 + 2)
GPIO0_D6	0	D	6	0 30(8 x 3 + 6)

一般的なコマンドラインは以下の通りです。-h オプションを使用してコマンドの使用方法を確認できます（例として GPIO1_C4 を使用）。

libgpiod コマンド

コマンド	機能	使用例	説明
gpiodetect	全 GPIO コントローラをリスト	gpiodetect (パラメータなし)	全 GPIO コントローラをリスト
gpioinfo	GPIO コントローラのピン状況をリスト	gpioinfo 1	第 1 グループのコントローラのピン状況をリスト
gpioset	GPIO を設定	gpioset 1 20=0	第 1 グループのコントローラのピン番号 20 を低電圧に設定
gpioget	GPIO ピンの状態を取得	gpioget 1 20	第 1 グループのコントローラのピン番号 20 の状態を取得
gpiomon	GPIO の状態を監視	gpiomon 1 20	第 1 グループのコントローラのピン番号 20 の状態を監視

重要: Rockchip のピン ID はコントローラ（bank）+ポート（port）+インデックス番号（pin）で構成されます。ポート番号とインデックス番号は 1 つの数値として gpiod に渡されます。すべてのピンが libgpiod で制御できるわけではありません。例えば、LED などの一部のピンは既に使用されているため、デバイスがビジー状態であるため使用できません。

15.4 FAQ

Q1: GPIO を使用しているときに「gpioset: error setting the GPIO line values: Device or resource busy」と表示される場合**

A1: GPIO が使用中であることを示しています。使用中の原因は、デバイスツリーでそのピンが GPIO または他の多重機能として使用されているためです。

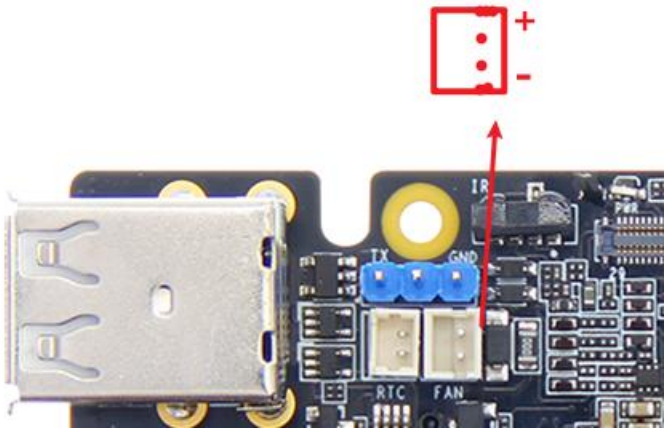
16 FAN インタフェース

16.1 インターフェースモデルと接続

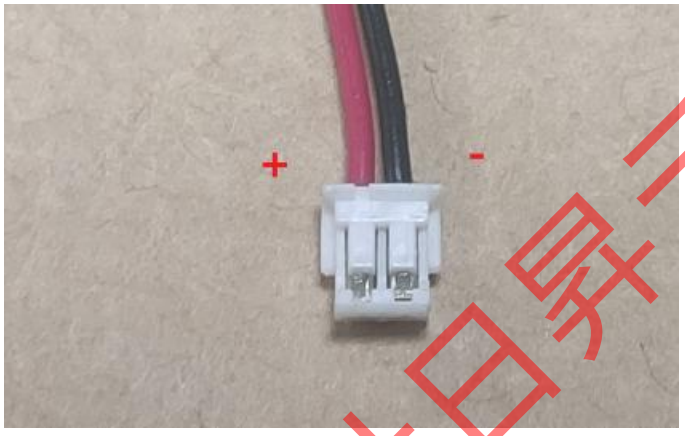
- パッケージは A1501WV-2P（1.5mm 1x2P スルーホール）

- 5V 電源のファンのみ対応

出力の正負極は、下図のようになります（ボードの裏側に詳細な表示があります）。



使用するファンのインターフェースは下図のようになります。



注：もしファンのインターフェースの正負極が上図と逆である場合、改線方法を用いて、ピンセットで 2 つの線を交換することができます。

16.2 ファンの多段階調速

表 1: 多段階調速

ファン回転レベル	トリガ温度	ファンの出力
0	CPU 温度が 50 度未満	ファン停止
1	CPU 温度が 50 度超、55 度未満	最大回転数の 39%
3	CPU 温度が 55 度超、60 度未満	最大回転数の 59%
4	CPU 温度が 60 度超、65 度未満	最大回転数の 78%
5	CPU 温度が 65 度超	最大出力

16.3 FAQs

Q1: ファンの多段階調速をどうやって変更するのですか？

A1: この機能は、デバイスツリーを変更することで多段階調速の設定を変更する必要があります。

17 赤外信号

17.1 カスタム赤外線リモコン

```
# 出力レベルを設定する
  echo '7 4 1 7'> /proc/sys/kernel/printk
# ログ出力を有効にする
  echo 1 > /sys/module/rockchip_pwm_remotectl/parameters/code_print

# 赤外線リモコンのボタンを押す

# ユーザーコードとキーコードを取得する

  [ 2094.260433] USERCODE=0xff00
  [ 2094.287460] RMC_GETDATA=bb
  [ 2094.751505] USERCODE=0xff00
  [ 2094.778527] RMC_GETDATA=bb
  [ 2095.054456] USERCODE=0xff00
  [ 2095.081495] RMC_GETDATA=bb
  [ 2095.343114] USERCODE=0xff00
  [ 2095.370167] RMC_GETDATA=bb

   USERCODE がユーザーコード
  RMC_GETDATA がキーコード
ユーザーコードに基づいて、デバイスツリーを設定し、キーコードに対応する機能をキーにマッピングする
```

```
&pwm7 {
    compatible = "rockchip,remotectl-pwm";
    pinctrl-names = "default";
    pinctrl-0 = <&pwm7m0_pins>;
    remote_pwm_id = <3>;
    handle_cpu_id = <1>;
    remote_support_psci = <0>;
    status = "okay";

    ir_key_lubancat{
    rockchip,usercode = <0xff00>;
    rockchip,key_table =
        <0xeb   KEY_POWER>,
        <0xec   KEY_MENU>,
        <0xfe   KEY_BACK>,
        <0xb7   KEY_HOME>,
        <0xa3   KEY_WWW>,
        <0xf4   KEY_VOLUMEUP>,
        <0xa7   KEY_VOLUMEDOWN>,
        <0xf8   KEY_REPLY>,
        <0xfc   KEY_UP>,
        <0xfd   KEY_DOWN>,
        <0xf1   KEY_LEFT>,
        <0xe5   KEY_RIGHT>;
    };
```

18 HDD のマウント

本章はボードのハードディスクマウントについて説明します

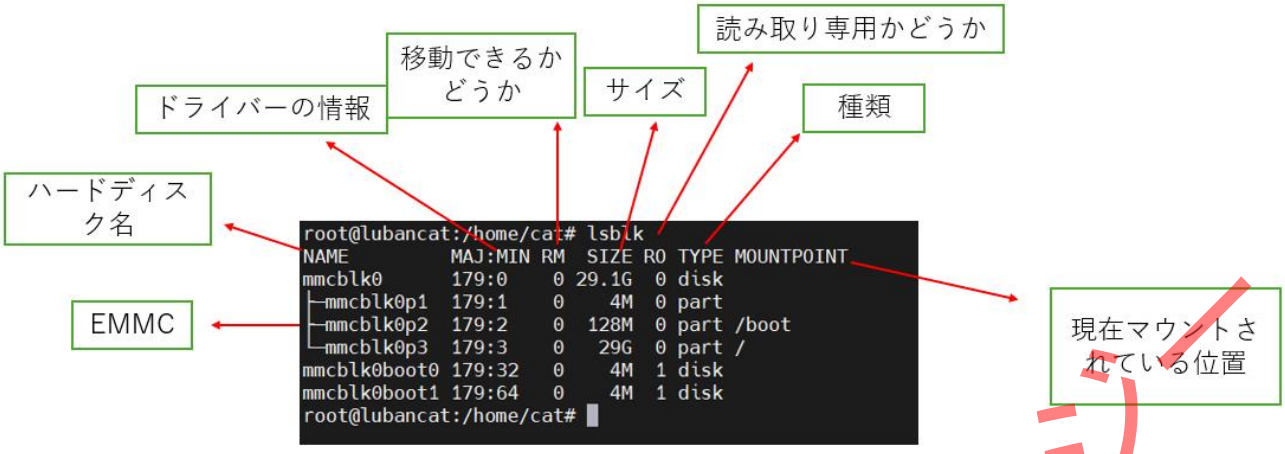
18.1 ハードディスクの確認

18.1.1 lsblk

すべての利用可能なブロックデバイスの情報をリストし、それらの依存関係を表示します。

lsblk

以下の図のように：



左側にすべての利用可能なブロックデバイスの情報がリストされ、右側には現在のハードディスクのマウント位置が表示されます。

- 現在利用可能なブロックデバイスは 5 つあり、それぞれ sda、sdb、mmcblk0、mmcblk0boot0、mmcblk0boot1 です。

- mmcblk0boot0 と mmcblk0boot1 は mmcblk0 の一部の情報であり、lsblk では読み取り専用であるため、これを利用可能なブロックとして使用する必要はありません。

- sda は msata ハードディスクで、ホットプラグをサポートしておらず、RM=0 でサイズは 16G、システム上では 14.8G として認識され、読み書き可能であり、唯一のパーティション sda1 が/root/test_blk にマウントされています。

- sdb は USB メモリで、ホットプラグをサポートしており、RM=1 でサイズは 32G、システム上では 28.7G として認識され、読み書き可能であり、唯一のパーティション sdb1 はマウントされていません。

- mmcblk0 はオンボードの eMMC で、ホットスワップをサポートしておらず、RM=0、サイズは 32G です。システム上では 29.1G として認識され、読み書きが可能です。

3 つのパーティションがあります：mmcblk0p1 は uboot およびシステムブート関連のパーティションで、サイズは 4M です。mmcblk0p2 はブートパーティションで、カーネ

ル、デバイスツリー、設定ファイルなどを格納するパーティションで、/boot パーティションにマウントされています。mmcblk0p3 はルートファイルのパーティションで、ルートファイルシステムを格納し、/（ルートディレクトリ）にマウントされています。

18.1.2 fdisk

fdisk はハードディスクの基本情報を確認するために使用できます。

```
# ハードディスクまたはパーティションの基本情報をリストする
fdisk [options] -l [<disk>]
# ハードディスクまたはパーティションの情報を変更する（詳細は自分で調べてください）
    fdisk [options] [<disk>]

    例：以下の図のように

    root@lubancat:~/rt-tests# fdisk -l /dev/mmcblk0
    Disk /dev/mmcblk0: 29.13 GiB, 31272730624 bytes, 61079552 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: gpt
    Disk identifier: D9500000-0000-4945-8000-362C000010D3

    Device       Start    End Sectors Size Type
    /dev/mmcblk0p1 16384   24575    8192   4M unknown
    /dev/mmcblk0p2 24576  286719  262144 128M unknown
    /dev/mmcblk0p3 286720 61079487 60792768  29G unknown
    root@lubancat:~/rt-tests#
```

- emmc の基本情報を確認できます。

- fdisk はハードディスクの設定を変更することもできますが、詳細な方法はここでは説明しませんので、自分で調べてください。

18.1.3 SD カード

SD カードの情報は emmc とほぼ同じです。同じタイプのドライバを使用しています。ボードのシステムでは、SD カードのデフォルトデバイスは`/dev/mmcblk1`です。SD カードはシステムイメージとしてボードを起動できるため、`lsblk`では非移動デバイスとして表示されます。

パーティション番号は`/dev/mmcblk1p#`（#は 1, 2, 3, 4, 5, 6, 7）です。

18.1.4 USB メモリ

USB メモリはボードのシステムではデフォルトデバイスは/dev/sdx（x は a, b, c, d, e…）です。デバイスの数に応じてデバイス番号が決まります。lsblk では移動可能デバイスとして表示されます。

パーティションデバイス番号は/dev/sdx#（x は a, b, c, d, e…、#は 1, 2, 3, 4, 5, 6, 7）です。

18.1.5 SATA ハードディスク

SATA ハードディスクはボードのシステムではデフォルトデバイスは/dev/sdx（x は a, b, c, d, e…）です。デバイスの数に応じてデバイス番号が決まります。

パーティションデバイス番号は/dev/sdx#（x は a, b, c, d, e…、#は 1, 2, 3, 4, 5, 6, 7）です。

18.2 手動でのマウント

18.2.1 ハードディスクのマウント

基本的なコマンド：

```
# マウントするフォルダを作成する（既存のフォルダがある場合は新たに作成する必要はありません）
    mkdir /mnt/mydir

# ハードディスクをマウントする
    mount /dev/xxx /mnt/mydir
```

- /dev/xxx はハードディスクのパーティションデバイス
- /mnt/mydir は、あなたがマウントしたいフォルダです。一般的なフォルダのマウント場所は/mnt または/media なので、ここでは/mnt/mydir にハードディスクをマウントします。

mount の機能は非常に豊富で、ここでは基本的な使用方法について簡単に説明します。必要に応じて、mount の詳細な使用方法を自分で調べることができます。

18.2.2 ハードディスクのアンマウント

```
umount /mnt/mydir
```

- /mnt/mydir はマウントされたフォルダで、lsblk で確認できます。

18.2.3 fstab による自動マウント（デバイスパーティション）

注：この方法は単一のストレージデバイスで、ハードディスクのデバイス番号が変わらな

い場合に適しています。

```
# /etc/fstab ファイルを編集する
  vi /etc/fstab
```

```
# 以下の内容を追加する
  /dev/sdx /mnt/mydir auto defaults 0 2
```

- /dev/sdx はマウントするハードディスクのパーティション

- /mnt/mydir はマウントするフォルダ

- auto はハードディスクパーティションのファイルシステム形式を自動判別する設定。他には ext4 や ntfs などがあります。

- defaults はデフォルトの設定。他には mode=1777 や rw,noauto などがあります。

- 0 と 2 はそれぞれバックアップとチェックの順序です。

18.2.4 fstab による自動マウント（UUID）

注：この方法はストレージデバイスが多く、デバイス番号が変わる可能性がある場合に適しています。

UUID は「Universally Unique Identifier」（汎用一意識別子）の略です。これはコンピュータシステム内のさまざまなリソース（例：ディスク、パーティション、ファイル、ディレクトリなど）を一意に識別するための識別子です。

UUID は 128 ビットの数値識別子で、通常は 16 進数で表されます。その目的は、異なるシステムや環境で一意性を保ち、競合や重複を避けることです。各 UUID は異なるコンピュータやネットワーク内でも一意である必要があります。

```
# 方法 1：ハードディスクパーティションの UUID を一覧表示する
```

```
    lsblk -o NAME,UUID

# 方法 2：ハードディスクパーティションの UUID を一覧表示する
    blkid
```

- ハードディスクパーティションの UUID を取得する

```
# /etc/fstab ファイルを編集する
    vi /etc/fstab
```

```
# 以下の内容を追加する

    UUID=xxxxx /mnt/mydir auto defaults 0 2
```

注：UUID のほかに PARTUUID も UUID と同じ機能がありますが、ここでは詳述しません。

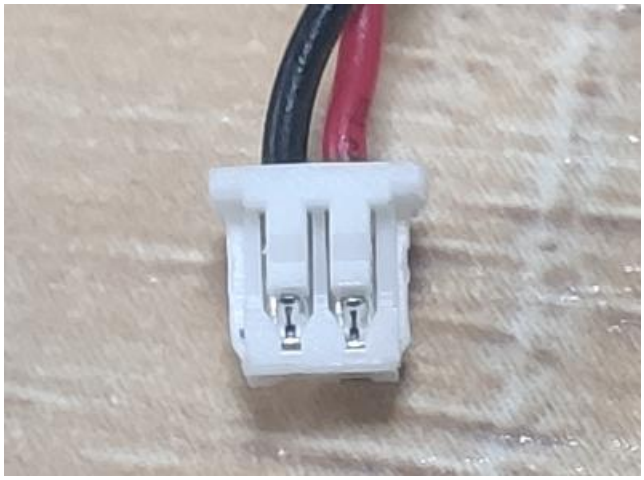
19 RTC

RTC（Real Time Clock）は、時間を記録するための専用ハードウェアデバイスです。通常は SoC 内部に統合されているか、外部 I2C 通信を介して接続されます。

なぜ RTC が必要なのかというと、Linux のシステム時間（いわゆる wall time）はシステムが稼働しているときにしか利用できず、システムがシャットダウンすると失われてしまうためです。RTC は、システムがシャットダウンした後も、外部電池や他の電源を頼りに動作し続け、時間を保存します。

ボードでは、RTC 機能を有効にするには RTC バッテリーが必要です。RTC バッテリーの選択肢は以下の通りです：

接続の正負極は以下の通りです：



もし配線の順序が間違っている場合、ピンセットを使って RTC バッテリーのインターフェースを調整できます。

19.1 使用方法

Linux は 3 つのユーザースペース呼び出しインターフェースを提供しています。ボードでの対応パスは以下の通りです：

- SYSFS インターフェース： /sys/class/rtc/rtc0/

- PROCFS インターフェース： /proc/driver/rtc

- IOCTL インターフェース： /dev/rtc0

SYSFS インターフェース：

```
root@lubancat:~# cat /sys/class/rtc/rtc0/date
2023-06-19
root@lubancat:~# cat /sys/class/rtc/rtc0/time
03:15:22
```

PROCFS インターフェース：

```
root@lubancat:~# cat /proc/driver/rtc
rtc_time        : 07:06:56
rtc_date        : 2023-06-19
alrm_time       : 03:18:50
alrm_date       : 2023-06-19
alarm_IRQ       : no
alrm_pending    : no
update IRQ enabled : no
periodic IRQ enabled : no
periodic IRQ frequency : 1
max user IRQ frequency : 64
24hr            : yes
root@lubancat:~#
```

IOCTL インターフェース：

```
カーネルドキュメントを参考にしてください。
tools/testing/selftests/timers/rtctest.c
```

19.2 よく使われるコマンド

```
date // システムクロックを変更する、具体的な使用方法は`man`で確認してください。
hwclock -s // ハードウェア時間をシステム時間に同期する。
hwclock -w // システム時間をハードウェア時間に同期する。
timedatectl // システム時間などを表示する。
```

20 カメラ

20.1 使用カメラ

本ボードは現在、最大 3 つの imx415 カメラを同時にサポートしています。以下は Debian 11 システムを用いた説明です。

20.1.1 単一カメラ

CAM0、CAM1、CAM2 はそれぞれ独立してカメラを接続できます。カメラはデフォルトでオフになっており、`/boot/uEnv/uEnv.txt`ファイルで設定します。

20.1.1.1 カメラの有効化

ここでは CAM0 を有効にする例を示します。CAM1 や CAM2 も同様の方法で有効にできます。

```
# 設定ファイルを開く
vi /boot/uEnv/uEnv.txt
```

- CAM0 のデバイスツリープラグインを見つけ、その前の#を削除してファイルを保存し、再起動します。

```
# CAM0
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.1.2 カメラの無効化

ここでは CAM0 を無効にする例を示します。CAM1 や CAM2 も同様の方法で無効にできます。

```
# 設定ファイルを開く
  vi /boot/uEnv/uEnv.txt
```

- CAM0 のデバイスツリープラグインを見つけ、その前に#を追加してファイルを保存し、再起動します。

```
# CAM0
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.1.3 カメラノード

単一カメラのノードは/dev/video11 です。

カメラプレビューコマンド:

```
gst-launch-1.0 v4l2src device=/dev/video11 ! video/x-raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink
```

20.1.2 デュアルカメラ

CAM0、CAM1、CAM2 はそれぞれ独立してカメラを接続できます。カメラはデフォルトで

オフになっており、`/boot/uEnv/uEnv.txt`ファイルで設定します。デュアルカメラは任意の組み合わせで使用できます。

- CAM0 と CAM1

- CAM0 と CAM2

- CAM1 と CAM2

20.1.2.1 カメラの有効化

ここでは CAM0 と CAM1 を有効にする例を示します。他の組み合わせも同様の方法で有効にできます。

```
# 設定ファイルを開く
  vi /boot/uEnv/uEnv.txt
```

- CAM0 と CAM1 のデバイスツリープラグインを見つけ、その前の#を削除してファイルを保存し、再起動します。

```
# CAM0
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.2.2 カメラの無効化

ここでは CAM0 と CAM1 を無効にする例を示します。他の組み合わせも同様の方法で無効にできます。

```
# 設定ファイルを開く
  vi /boot/uEnv/uEnv.txt
```

- CAM0 と CAM1 のデバイスツリープラグインを見つけ、その前に#を追加してファイ

ルを保存し、再起動します。

```
# CAM0
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.2.3 カメラノード

デュアルカメラのノードは/dev/video22 と/dev/video31 です。CAM0 と CAM1 を有効にした場合、それぞれのノードは/dev/video22 と/dev/video31 になります。

デュアルカメラプレビューコマンド

```
gst-launch-1.0 v4l2src device=/dev/video22 ! video/x-raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink &
gst-launch-1.0 v4l2src device=/dev/video31 ! video/x-raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink &
```

20.1.3 トリプルカメラ

CAM0、CAM1、CAM2 はそれぞれ独立してカメラを接続できます。カメラはデフォルトでオフになっており、/boot/uEnv/uEnv.txt ファイルで設定します。

20.1.3.1 カメラの有効化

ここでは CAM0、CAM1、CAM2 を有効にする例を示します。他の組み合わせも同様の方法で有効にできます。

```
# 設定ファイルを開く
vi /boot/uEnv/uEnv.txt
```

- CAM0、CAM1、CAM2 のデバイスツリープラグインを見つけ、その前の#を削除して

ファイルを保存し、再起動します。

```
# CAM0
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.3.2 カメラの無効化

ここでは CAM0、CAM1、CAM2 を無効にする例を示します。他の組み合わせも同様の方法で無効にできます。

```
# 設定ファイルを開く
  vi /boot/uEnv/uEnv.txt
```

- CAM0、CAM1、CAM2 のデバイスツリープラグインを見つけ、その前に#を追加してファイルを保存し、再起動します。

```
# CAM0
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam0-imx415-overlay.dtbo
# CAM1
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam1-imx415-overlay.dtbo
# CAM2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-cam2-imx415-overlay.dtbo
```

20.1.3.3 カメラノード

トリプルカメラのノードは/dev/video33、/dev/video42、/dev/video51 です。CAM0、CAM1、CAM2 を有効にした場合、それぞれのノードは/dev/video33、/dev/video42、/dev/video51 になります。

トリプルカメラプレビューコマンド

```
  gst-launch-1.0 v4l2src device=/dev/video33 ! video/x-raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink &
  gst-launch-1.0 v4l2src device=/dev/video42 ! video/x-raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink
```

```
&
   gst-launch-1.0 v4l2src device=/dev/video51 ! video/x-
raw,format=NV12,width=3840,height=2160,framerate=30/1 ! videoconvert ! autovideosink
&
```

21 MIPI LCD

注記: ボードのイメージは、デフォルトで MIPI ディスプレイがオフになっています。MIPI ディスプレイを有効にするには追加の設定が必要です。

ボードの MIPI ディスプレイ対応リスト

1. 5.5 インチ MIPI ディスプレイ - 1080x1920@60Hz

3. 7 インチ MIPI ディスプレイ - 1024x600@60Hz

3. 10.1 インチ MIPI ディスプレイ - 800x1280@60Hz

その他のディスプレイを適合させる場合

- dsi0 インターフェースは最大 3840x2160@60Hz をサポート

- dsi1 インターフェースは最大 2048x1080@60Hz をサポート

21.1 MIPI DSI インターフェース

本ボードには 2 つの MIPI DSI インターフェースがあり、これらのインターフェースを他のインターフェースと組み合わせてマルチディスプレイを実現できます。

インターフェースの詳細情報

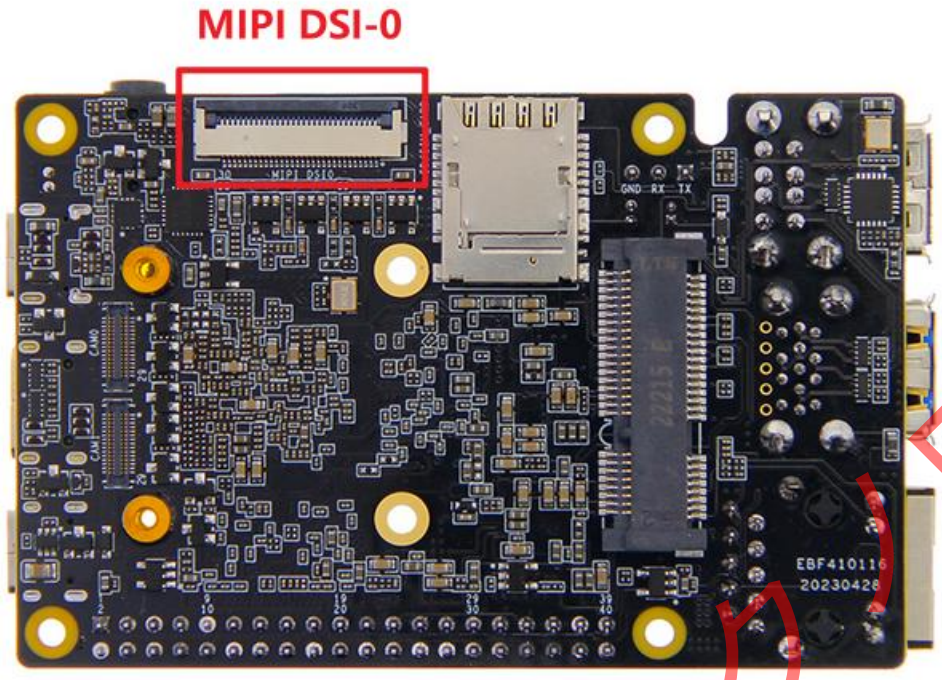
- FPC コネクタ

- ピッチ：0.5mm

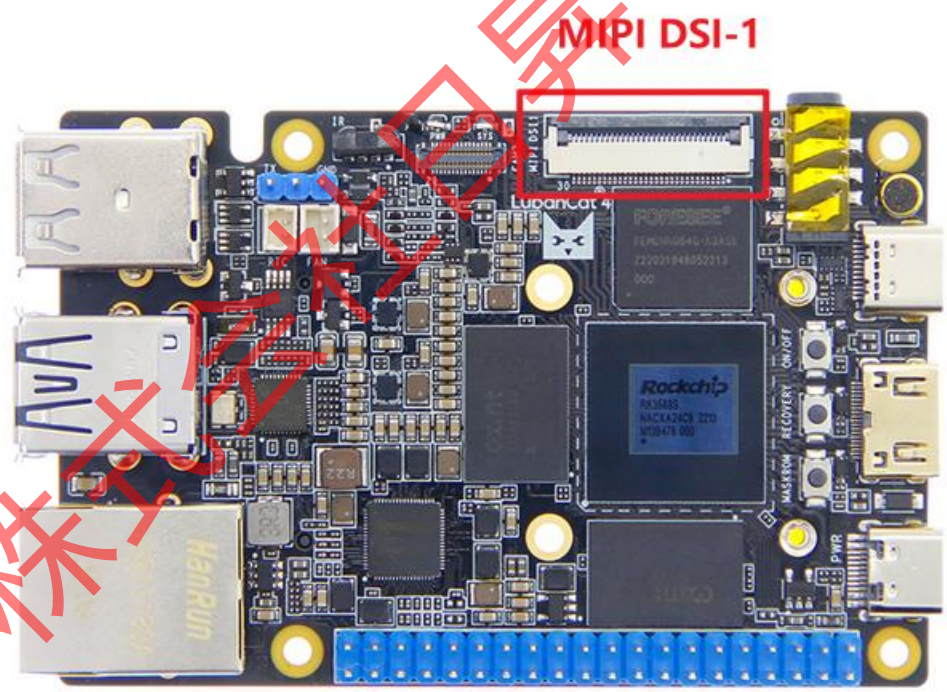
- 30 ピン

- 上向きフリップ接続

- MIPI DSI 0 (ボードの背面にあります)



- MIPI DSI 1 (ボードの前面にあります)



MIPI ディスプレイに接続する際の FPC ケーブル要件：

- 30 ピン

- ピッチ 0.5mm

- 反対面接続

21.3 MIPI ディスプレイの使用方法

MIPI ディスプレイのオン/オフは、設定ファイルで行います。設定ファイルおよびデバイスツリー&デバイスツリープラグインに移動して設定を変更します。

```
# 設定ファイルの確認方法
cat /boot/uEnv/uEnv.txt
```

```
# Display
#dsi0 in vp2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
#dsi1 in vp3
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi1-1080p-overlay.dtbo
```

設定ファイルの# Display の下に、#dsi0 in vp2 と#dsi1 in vp3 という 2 つのエリアがあります。

- #dsi0 in vp2 の下のエリアのデバイスツリープラグインは dsi0 インターフェースに対応しています。

- #dsi1 in vp3 の下のエリアのデバイスツリープラグインは dsi1 インターフェースに対応しています。

これらのエリアの下には、各ディスプレイに対応する複数のデバイスツリープラグインがあります。

- rk3588s-lubancat-4-xxxx-800x1280-overlay.dtbo は 10.8 インチ MIPI ディスプレイに対応

- rk3588s-lubancat-4-xxxx-1024x600-overlay.dtbo は 7 インチ MIPI ディスプレイに対応

- rk3588s-lubancat-4-xxxx-1080p-overlay.dtbo は 5.5 インチ MIPI ディスプレイに対応（

21.3.1 MIPI ディスプレイの有効化

例として、MIPI dsi0 インターフェースの 5.5 インチディスプレイを有効にする方法を示します。

```
# 設定ファイルを編集
  sudo vi /boot/uEnv/uEnv.txt
```

```
#dsi0 in vp2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
```

- #dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo の行の前の#を削除します。
- 本質的には、#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo のコメントを解除します。

```
#dsi0 in vp2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
```

ファイルを修正した後、設定を有効にするために再起動できます。

21.3.2 MIPI ディスプレイの無効化

例として、MIPI dsi0 インターフェースの 7 インチディスプレイを無効にする方法を示します。

```
# 設定ファイルを編集
  sudo vi /boot/uEnv/uEnv.txt
```

```
#dsi0 in vp2
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
#dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
```

• dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo の行の最初に#を追加します。

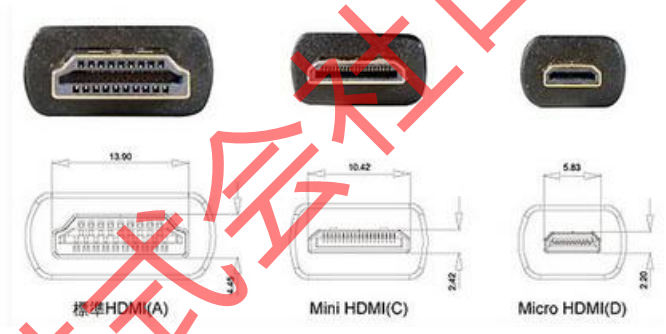
• 本質的には、コメントされていない dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo をコメントアウトします。

```
1 #dsi0 in vp2
2 #dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-800x1280-overlay.dtbo
3 #dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1024x600-overlay.dtbo
4 #dtoverlay=/dtb/overlay/rk3588s-lubancat-4-dsi0-1080p-overlay.dtbo
```

ファイルを修正した後、設定を有効にするために再起動できます。

22 HDMI

三種の一般的な HDMI インターフェースの比較図

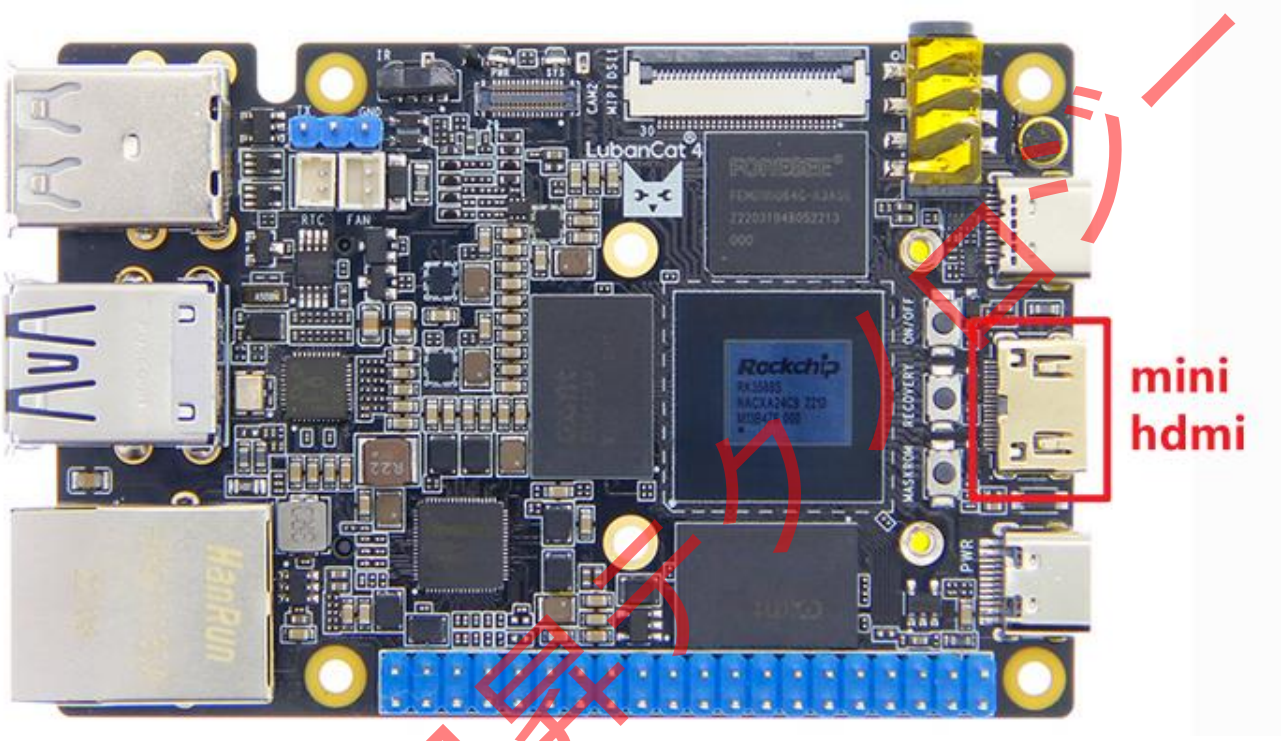


本ボードで使用される HDMI インターフェースは MINI-HDMI で、ホットスワップに対応しています。

注意:HDMI と MINI-HDMI のインターフェースのサイズと形状は大きく異なるため、直接接続することはできません。HDMI と MINI-HDMI を接続して使用する場合は、HDMI to MINI-HDMI 変換アダプターを購入する必要があります。

22.1 表示の説明

22.1.1 表示インターフェース説明



サポートされるビデオ出力パラメータ（デフォルト状態）：3840x2160@60Hz

- 本ボードの HDMI インターフェースはデフォルトで有効になっており、HDMI インターフェースを接続し、ディスプレイとペアリングすると表示されます。

- 本ボードの HDMI インターフェースは理論上最大で 7680x4320@60Hz を出力できますが、システムデフォルトでは最大で 3840x2160@60Hz までサポートされています。これは、LubanCat-4 が 4 画面の異なる表示を考慮しているためです。最大出力を 7680x4320@60Hz に変更すると、1 つの画面の異なる表示を失い、最大 3 画面の異なる表示しかサポートされなくなります。これを総合的に考慮した結果、4 画面の異なる表示の方式を採用しました。

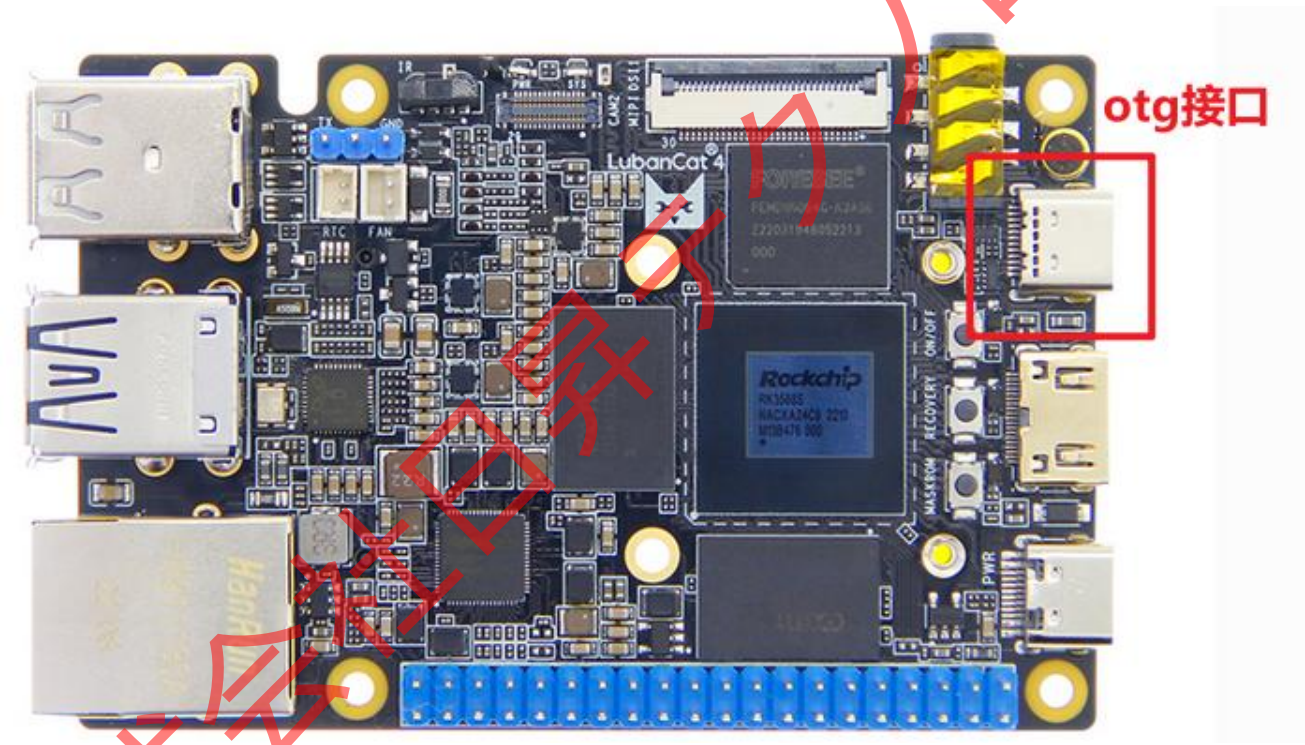
注: HDMI インターフェースを 7680x4320@60Hz に設定する必要がある場合は、資料を参考にしてデバイスツリーを変更するか、お問い合わせください。

23 DP

23.1 表示インターフェースの説明

サポートされるビデオ出力パラメータ（デフォルト状態）：3840x2160@60Hz

本ボードのの Type-C OTG インターフェースは基本的な USB 機能のほかに、DP ビデオ出力もサポートしています。



- 本ボードの Type-C インターフェースの DP 機能はデフォルトで有効になっており、Type-C インターフェースを接続し、ディスプレイとペアリングすると表示されます。

- 本ボードの Type-C インターフェースは理論上最大で 7680x4320@60Hz を出力できますが、システムデフォルトでは最大で 3840x2160@60Hz までサポートされています。これは、本ボードが 4 画面の異なる表示を考慮しているためです。最大出力を 7680x4320@60Hz に変更すると、1 つの画面の異なる表示を失い、最大 3 画面の異なる表

示しかサポートされなくなります。これを総合的に考慮した結果、4 画面の異なる表示の方式を採用しました。

注: DP インターフェースを 7680x4320@60Hz に設定する必要がある場合は、資料を参考にしてデバイスツリーを変更するか、お問い合わせください。

24 ビデオコーデック-mpp ライブラリに基づく

24.1 MPP について

Rockchip が提供するメディア処理プラットフォーム（Media Process Platform、略称 MPP）は、Rockchip シリーズのチップに適した汎用メディア処理ソフトウェアプラットフォームです。このプラットフォームは、アプリケーションソフトウェアからチップ関連の複雑な低層処理を隠蔽し、異なるチップの差異を隠すことを目的としています。ユーザーに統一されたビデオメディア処理インターフェース（Media Process Interface、略称 MPI）を提供します。

MPP の提供する機能には次のものがあります：

ビデオデコード	H.265 / H.264 / H.263 / VP9 / VP8 / MPEG-4 / MPEG-2 / MPEG-1 / VC1 / MJPEG / AV1
ビデオエンコード	H.265 / H.264 / VP8 / MJPEG
ビデオ処理	ビデオコピー、スケーリング、色空間変換、フィールドビデオデインターレース（Deinterlace）

24.2 RKMPP ライブラリの取得とコンパイル

24.2.1 テスト環境

- テストボード：Lubancat4

- テスト OS：Ubuntu 20.04

- カーネルバージョン：Linux 4.19.232

- RK 公式 MPP ライブラリのアドレス：[GitHub リンク](https://github.com/rockchip-linux/mpp)

24.2.2 関連ツールのインストール

```
sudo apt update
sudo apt install -y git cmake
```

24.2.3 RK 公式 MPP リポジトリのクローン

```
git clone https://github.com/rockchip-linux/mpp.git
```

24.2.4 コンパイル

aarch64 の適切なビルドディレクトリに移動します。

```
cd mpp/build/linux/aarch64/
```

クロスコンパイル設定ファイルを編集し、コンパイラ gcc と g++を指定します（通常はデフォルトのままで問題ありません）。

```
vim arm.linux.cross.cmake
```

```
cmake_minimum_required( VERSION 2.6.3 )

SET(CMAKE_SYSTEM_NAME Linux)
SET(CMAKE_C_COMPILER "aarch64-linux-gnu-gcc")
SET(CMAKE_CXX_COMPILER "aarch64-linux-gnu-g++")
#SET(CMAKE_SYSTEM_PROCESSOR "armv7-a")
SET(CMAKE_SYSTEM_PROCESSOR "armv8-a")

add_definitions(-fPIC)
add_definitions(-DARMLINUX)
add_definitions(-Dlinux)
~
~
~
~
~
```

bash スクリプトを実行してコンパイルを開始します（コンパイルには約 15 分かかります）。

```
./make-Makefiles.bash
make
```

コンパイルが終了すると、多くのファイルがディレクトリに生成されます。

```
root@lubancat:~/mpp/build/linux/aarch64# ls
arm.linux.cross.cmake  compile_commands.json  osal              utils
CMakeCache.txt         Makefile               rockchip_mpp.pc
CMakeFiles             make-Makefiles.bash    rockchip_vpu.pc
cmake_install.cmake    mpp                    test
```

test ディレクトリに移動すると、生成されたテストプログラムが見つかります。

```
root@lubancat:~/mpp/build/linux/aarch64# cd test/
root@lubancat:~/mpp/build/linux/aarch64/test# ls
CMakeFiles             mpi_dec_mt_test     mpi_dec_test     mpi_rc2_test
cmake_install.cmake    mpi_dec_multi_test  mpi_enc_mt_test  mpp_info_test
Makefile               mpi_dec_nt_test     mpi_enc_test     vpu_api_test
```

24.3 ビデオデコード

デコーダーデモは mpi_dec_test シリーズプログラムで、decode_put_packet と

decode_get_frame インターフェースを使用する単一スレッドの mpi_dec_test、多重スレッドの mpi_dec_mt_test、および多インスタンスの mpi_dec_multi_test が含まれます。

24.3.1 テスト環境

- テストボード：Lubancat4
- テスト OS：Ubuntu 20.04
- カーネルバージョン：Linux 4.19.232

24.3.2 mpi_dec_test のコマンドパラメータ

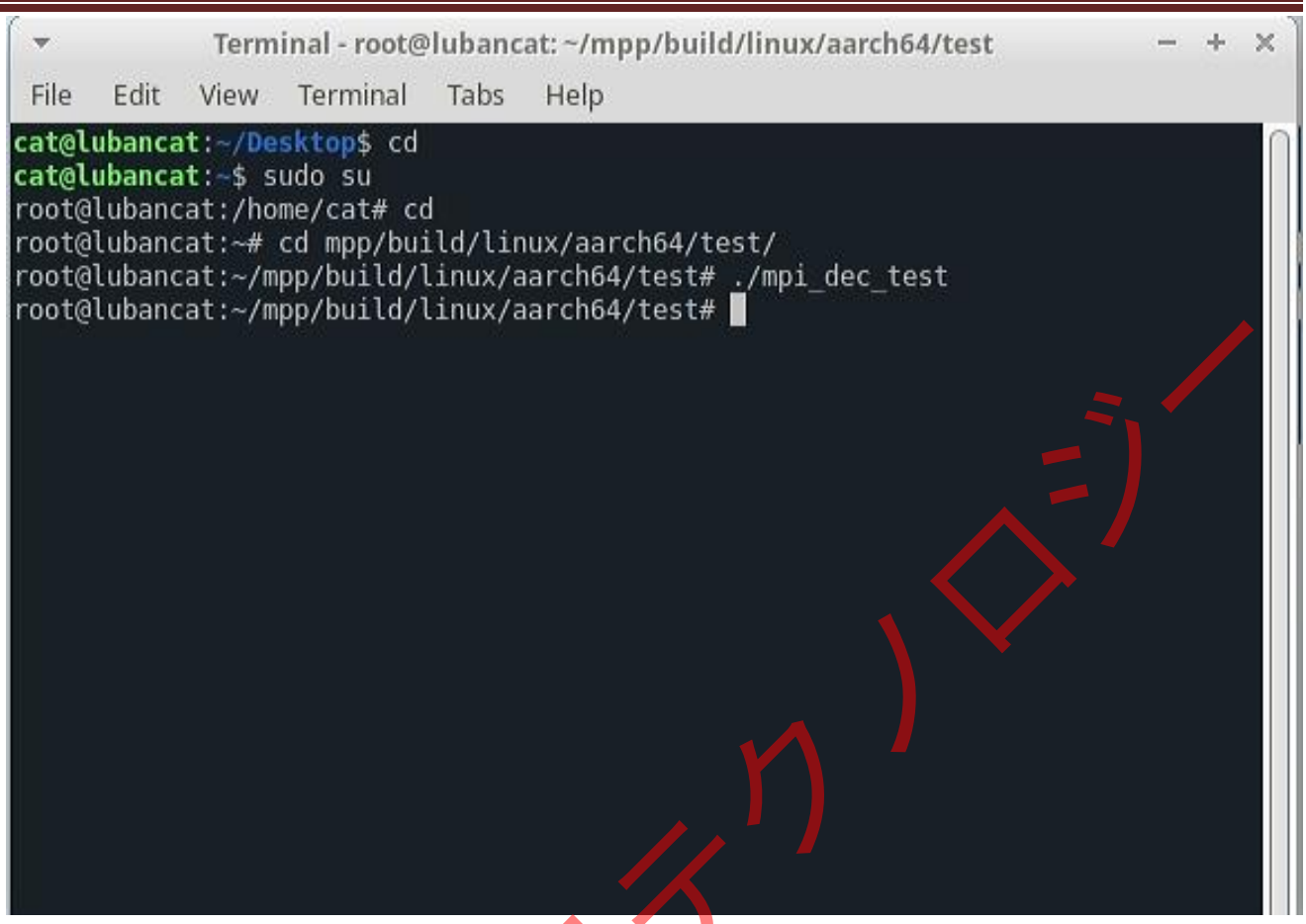
24.3.2.1 ターミナルで mpi_dec_test のコマンドパラメータを確認

2 つのターミナルを開き、1 つのターミナルに以下のコマンドを入力してログ出力を監視します。

```
sudo tail -f /var/log/syslog
```

もう 1 つのターミナルで mpi_dec_test テストプログラムを実行します。

```
mpi_dec_test
```



テストプログラムを実行すると、ログに以下のヘルプドキュメントが表示されます。

```
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: usage: ./mpi_dec_test [options]
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -i    input_file   input bitstream file
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -o    output_file  output decoded frame file
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -w    width        the width of input bitstream
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -h    height       the height of input bitstream
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -t    type         input stream coding type
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -f    format       output frame format type
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -n    frame_number max output frame number
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -s    instance_nb  number of instances
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -v    trace option q - quiet f - show fps
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -slt  slt file     slt verify data file
Nov 20 16:13:54 lubancat mpp[2361]: mpi_dec_utils: -help help         show help
Nov 20 16:13:54 lubancat mpp[2361]: mpi: mpp coding type support list:
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: mpeg2          id 2
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: mpeg4          id 4
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: h.263          id 3
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: h.264/AVC      id 7
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: h.265/HEVC     id 16777220
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: vp8            id 9
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: VP9            id 10
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: avs            id 16777222
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: avs+           id 16777221
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: avs2           id 16777223
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: jpeg           id 8
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: dec id 0 coding: av1            id 16777224
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: enc id 1 coding: h.264/AVC      id 7
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: enc id 1 coding: jpeg           id 8
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: enc id 1 coding: h265           id 16777220
Nov 20 16:13:54 lubancat mpp[2361]: mpi: type: enc id 1 coding: vp8            id 9
```

ヘルプドキュメントは 2 つの部分に分かれています：mpi_dec_test のコマンドパラメータ

説明と、ストリームファイルのプロトコルタイプの説明です。

24.3.2.2 mpi_dec_test のコマンドパラメータの説明

-i	入力ストリームファイル
-o	出力画像ファイル
-w	画像の幅（ピクセル単位）
-h	画像の高さ（ピクセル単位）
-t	ストリームファイルのプロトコルタイプ
-f	画像の色空間フォーマットおよびメモリ配置方式（デフォルトは NV12）
-n	最大デコードフレーム数（ストリームが長い場合、テスト時には最初の n フレームのみ出力できます）
-s	MPP インスタンス数（デフォルトは 1）
-v	ログオプション（q はサイレントモード、f は fps 表示モード）
-slt	出力フレームに対応するチェックサムファイル
-help	ヘルプドキュメントを表示

Tips:

1. mpi_dec_test のコマンドパラメータで、入力ファイル（i）とストリームタイプ（t）は必須です。その他のパラメータ（出力ファイル（o）、画像幅（w）、画像高さ（h）、デコードフレーム数（n）など）はオプションで、テストのニーズに応じて設定できます。

2. mpi_dec_test のコマンドパラメータで、出力フレームに対応するチェックサムファイル（slt）は、出力フレームデータを対応する循環冗長検査コードに変換します（具体的なロジックは utils/utils.c を参照）。チェックサムファイルのサイズは通常数キロバイト程度で、チップの slt テストでは出力フレームファイルの比較をチェックサムファイルの比較に変換することで、テストサイクルを大幅に短縮できます。

24.3.2.3 MPP デコード対応のストリームファイルプロトコルタイプの説明

MPP が対応するデコードタイプ：

```
lubancat mpp[2457]: mpi: type: dec id 0 coding: mpeg2          id 2
lubancat mpp[2457]: mpi: type: dec id 0 coding: mpeg4          id 4
lubancat mpp[2457]: mpi: type: dec id 0 coding: h.263          id 3
lubancat mpp[2457]: mpi: type: dec id 0 coding: h.264/AVC      id 7
lubancat mpp[2457]: mpi: type: dec id 0 coding: h.265/HEVC     id 16777220
lubancat mpp[2457]: mpi: type: dec id 0 coding: vp8            id 9
lubancat mpp[2457]: mpi: type: dec id 0 coding: VP9            id 10
lubancat mpp[2457]: mpi: type: dec id 0 coding: avs            id 16777222
lubancat mpp[2457]: mpi: type: dec id 0 coding: avs+           id 16777221
lubancat mpp[2457]: mpi: type: dec id 0 coding: avs2           id 16777223
lubancat mpp[2457]: mpi: type: dec id 0 coding: jpeg           id 8
lubancat mpp[2457]: mpi: type: dec id 0 coding: av1            id 16777224
```

Tips:

1. MPP ライブラリが対応する入力ファイルのエンコードフォーマット（t）は MPEG2/4、H.263/4/5、VP8/9、および JPEG などで、id 後の数字は異なるエンコードフォーマットに対応するパラメータ値です。パラメータ値は OMX の定義に由来し、HEVC および AVS フォーマットのパラメータ値は他のフォーマットとは顕著に異なります。

24.3.3 デコードデモ

デコードプロセスを説明ます。

Tips:mp4 デコードプロセスは主に 2 つのステップに分かれます。最初のステップは、mp4 を mpp ライブラリが対応する純粋なビデオタイプ（例：h264）に変換すること、2 番目のステップは mpp ライブラリを使用して変換後のビデオをデコードすることです。

24.3.3.1 mp4 から h264 への変換

mp4 から h264 への変換には FFmpeg ツールを使用します。FFmpeg ツールの他の使用方法については対応する章を参照してください。

```
sudo apt update && sudo apt install -y ffmpeg # FFmpeg ツールのインストール
ffmpeg -i test.mp4 -c:v libx264 01.h264
```

ここで、test.mp4 は変換するソースファイル名で、01.h264 は出力ファイル名です。

24.3.3.2 h264 のデコード

ここでは、01.h264 ファイルをデコードします。2 つのターミナルを開き、1 つのターミナルに以下のコマンドを入力してログ出力を監視します。

```
sudo tail -f /var/log/syslog
```

もう 1 つのターミナルでデコードプログラムを実行します。

```
mpi_dec_test -i 01.h264 -t 7 -n 60 -o 01.yuv
```

ヒント：上記のコマンドは、01.h264 をデコードし、01.yuv として保存します。ここで、-i は入力ファイル、-t 7 は入力ストリームファイルのプロトコルタイプが H.264 であること、-n 60 は 60 フレームをデコードすること、-o は出力ファイルを表します。

```
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: input file 01.h264 size 23432070
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: cmd parse result:
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: input file name: 01.h264
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: output file name:
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: width      :    0
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: height     :    0
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: type       :    7
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_utils: max frames :   60
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: mpi_dec_test start
Nov 21 09:23:41 lubancat mpp[2032]: mpp_info: mpp version: e34f0dd1 author: Herman Chen   2023-07-17 [hal_vp8e]: Fix crash on unsupport input format
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 mpi_dec_test decoder test start w 0 h 0 type 7
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode_get_frame get info changed found
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decoder require buffer w:h [1920:1080] stride [1920:1088] buf_size 4177920
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 0
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 1
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 2
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 3
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 4
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 5
Nov 21 09:23:41 lubancat mpp[2032]: mpi_dec_test: 0x55b9a482e0 decode get frame 6
```

24.4 ビデオエンコード

エンコーダーデモは mpi_enc_test シリーズプログラムで、単一スレッドの mpi_enc_test および多インスタンスの mpi_enc_multi_test が含まれます。

24.4.1 テスト環境

- テストボード：Lubancat4

- テスト OS：Ubuntu 20.04

- カーネルバージョン：Linux 4.19.232

24.4.2 mpi_enc_test のコマンドパラメータ

24.4.2.1 ターミナルで mpi_enc_test のコマンドパラメータを確認

2 つのターミナルを開き、1 つのターミナルに以下のコマンドを入力してログ出力を監視します。

```
sudo tail -f /var/log/syslog
```

もう 1 つのターミナルで mpi_enc_test テストプログラムを実行します。

```
mpi_enc_test
```

テストプログラムを実行すると、ログに以下のヘルプドキュメントが表示されます。

```
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: usage: mpi_enc_test [options]
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -i       input_file         input frame file
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -o       output_file        output encoded bitstream file
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -w       width              the width of input picture
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -h       height             the height of input picture
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -hstride hor_stride         the horizontal stride of input picture
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -vstride ver_stride         the vertical stride of input picture
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -f       format             the format of input picture
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -t       type               output stream coding type
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -tsrc    source type        input file source coding type
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -n       max frame number   max encoding frame number
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -g       gop reference mode gop_mode:gop_len:vi_len
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -rc      rate control mode  set rc_mode, 0:vbr 1:cbr 2:fixqp 3:avbr
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -bps     bps target:min:max set tareget:min:max bps
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -fps     in/output fps      set input and output frame rate
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -qc      quality control    set qp_init:min:max:min_i:max_i
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -s       instance_nb        number of instances
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -v       trace option       q - quiet f - show fps
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -l       loop count         loop encoding times for each frame
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -ini     ini file           encoder extra ini config file
Nov 21 09:58:53 lubancat mpp[2445]: mpi_enc_utils: -slt     slt file           slt verify data file
Nov 21 09:58:53 lubancat mpp[2445]: mpi: mpp coding type support list:
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: mpeg2            id 2
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: mpeg4            id 4
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: h.263            id 3
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: h.264/AVC        id 7
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: h.265/HEVC       id 16777226
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: vp8              id 9
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: VP9              id 10
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: avs              id 16777222
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: avs+             id 16777221
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: avs2             id 16777223
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: jpeg             id 8
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: dec id 0 coding: av1              id 16777224
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: enc id 1 coding: h.264/AVC        id 7
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: enc id 1 coding: jpeg             id 8
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: enc id 1 coding: h265             id 16777220
Nov 21 09:58:53 lubancat mpp[2445]: mpi: type: enc id 1 coding: vp8              id 9
Nov 21 09:58:53 lubancat mpp[2445]: mpi: mpp color support list:
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 0     0x00000 YUV420SP,      NV12
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 1     0x00001 YUV420SP-10bit
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 2     0x00002 YUV422SP,      NV24
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 3     0x00003 YUV422SP-10bit
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 4     0x00004 YUV420P,       I420
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 5     0x00005 YUV420SP,      NV21
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 6     0x00006 YUV422P,       422P
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 7     0x00007 YUV422SP,      NV42
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 8     0x00008 YUV422-YUYV,   YUY2
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 10    0x0000a YUV422-UYVY,   UYVY
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 12    0x0000c YUV400-Y8,     Y800
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 15    0x0000f YUV444SP
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 16    0x00010 YUV444P
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65536 0x10000 RGB565
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65537 0x10001 BGR565
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65538 0x10002 RGB555
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65539 0x10003 BGR555
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65542 0x10006 RGB888
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65543 0x10007 BGR888
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65546 0x1000a ARGB8888
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65547 0x1000b ABGR8888
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65548 0x1000c BGRA8888
Nov 21 09:58:53 lubancat mpp[2445]: mpi: color: id 65549 0x1000d RGBA8888
```

ヘルプドキュメントは 3 つの部分に分かれています：mpi_enc_test のコマンドパラメータの説明、ストリームファイルのプロトコルタイプの説明、および画像の色空間フォーマットおよびメモリ配置方式の説明です。

24.4.2.2 mpi_enc_test のコマンドパラメータの説明

パラメータ	説明

-i	入力画像ファイル
-o	出力ストリームファイル
-w	画像の幅（ピクセル単位）
-h	画像の高さ（ピクセル単位）
-hstride	垂直方向の隣接する 2 行間の距離（バイト単位）
-vstride	画像成分間の行数の間隔数（単位は 1）
-f	画像の色空間フォーマットおよびメモリ配置方式（デフォルトは NV12）
-t	ストリームファイルのプロトコルタイプ
-tsrc	ソースストリームフォーマット（全体のエンコードデコードパフォーマンステストでのみ使用）
-n	最大デコードフレーム数（ストリームが長い場合、テスト時には最初の n フレームのみ出力できます）
-g	GOP 参照モード（異なる TSVC ストリームに対応）
-rc	ビットレート制御モード（0: VBR; 1: CBR; 2: FIXQP; 3: AVBR）
-bps	ビットレート制約パラメータ（コマンドフォーマット：bps_target:bps_min:bps_max）
-fps	入力/出力フレームレート制御（デフォルトは 30）。このコマンドパラメータは入力フレームレートと出力フレームレートの比率を示し、実際のフレームレートには関係ありません。
-qc	品質制御
-s	MPP インスタンス数（デフォルトは 1）
-v	ログオプション（q はサイレントモード、f は fps 表示モード）
-ini	追加のエンコード設定ファイル ini（未使用）
-slt	出力ストリームに対応するチェックサムファイル

Tips:

1. mpi_enc_test のコマンドパラメータで、画像幅（w）、画像高さ（h）、およびストリームタイプ（t）は必須です。その他のパラメータ（入力ファイル（i）、出力ファイル（o）、エンコードフレーム数（n）、色空間フォーマットおよびメモリ配置方式（f）など）はオプションで、指定しない場合、mpi_enc_test はデフォルトのカラーバー画像を生成してエンコードします。

2. mpi_enc_test のコマンドパラメータは多様なビットレート制御オプションを提供し、ユ

ーザーはビットレート制御モード（rc）とビットレート制約パラメータ（bps）を使用して出力ストリームのビットレートを制御できます。ビットレート制御モード（rc）は、可変ビットレートモード（VBR）、固定ビットレートモード（CBR）、qp 修正ビットレートモード（FIXQP）、および適応ビットレートモード（AVBR）に分かれ、デフォルトモードは VBR です。ビットレート制約パラメータ（bps）は、MPP 内部でビットレート境界を設定するための参考として使用されます。

3. mpi_enc_test のコマンドパラメータで、ログオプション（v）は q の場合、MPP のデイリーログが無効になります。ログオプション（v）は f の場合、毎秒平均フレームレートと現在のフレームレートが表示されます。

mpi_enc_test のコマンドパラメータのフレームレート制御（fps）のフォーマットは以下の通りです：

```
-fps fps_in_num:fps_in_den:fps_in_flex/fps_out_num:fps_out_den:fps_out_flex
```

Tips:ここで、in/out はそれぞれ入力/出力を示し、num は分子、den は分母、flex は 0 で固定フレームレート、1 で可変フレームレートを示します。入力と出力のデフォルトの num と den はそれぞれ 30 と 1 で、デフォルトの入力/出力フレームレートは 30 です。このコマンドパラメータは入力フレームレートと出力フレームレートの比率を示し、実際のフレームレートには関係ありません。

mpi_enc_test のコマンドパラメータで、品質制御（qc）は出力ストリームフォーマットが H.264、H.265、VP8、および JPEG の場合にのみ有効で、コマンドフォーマットは以下の通りです：

```
-qc qp_init/min/max/min_i/max_i
```

24.4.2.3 MPP エンコード対応のストリームファイルプロトコルタイプの説明

MPP が対応するエンコードタイプ：

```
lubancat mpp[2457]: mpi: type: enc id 1 coding: h.264/AVC          id 7
lubancat mpp[2457]: mpi: type: enc id 1 coding: jpeg               id 8
lubancat mpp[2457]: mpi: type: enc id 1 coding: h265               id 16777220
lubancat mpp[2457]: mpi: type: enc id 1 coding: vp8                id 9
```

Tips:1. MPP ライブラリが対応する入力ファイルのエンコードフォーマット（t）は H.265、H.264、VP8、MJPEG などで、id 後の数字は異なるエンコードフォーマットに対応するパラメータ値です。

24.4.2.4 画像の色空間フォーマットおよびメモリ配置方式の説明

画像の色空間フォーマットは YUV と RGB の 2 種類に分かれます。MPP は多くのメモリ配置方式（f）をサポートしており、id 後の数字は異なるメモリ配置方式に対応するパラメータ値です。注意すべきは、YUV と RGB フォーマットのパラメータ値には顕著な違いがあることです。

24.4.3 エンコードデモ

ここでは、上記でデコードされた`01.yuv`ファイルのエンコードを例に、エンコードプロセスを示します。

Tips:ここでは、mpp ライブラリを使用して yuv ファイルを h265 にエンコードする方法、および ffmpeg を使用して h265 を mp4 に変換する方法を示します。

24.4.3.1 yuv を h265 にエンコード

2 つのターミナルを開き、1 つのターミナルに以下のコマンドを入力してログ出力を監視

します。

```
sudo tail -f /var/log/syslog
```

もう 1 つのターミナルでエンコードプログラムを実行します。

```
mpi_enc_test -i 01.yuv -w 1920 -h 1080 -t 16777220 -o 01.h265 -n 20
```

ヒント：上記のコマンドは、01.yuv をエンコードし、01.h265 として保存します。ここで、-i は入力ファイル、-w 1920 は幅を 1920 ピクセルに指定、-h 1080 は高さを 1080 ピクセルに指定、-t 16777220 は出力ストリームファイルのプロトコルタイプが H.265 であること、-n 60 は 60 フレームをエンコードすること、-o は出力ファイルを表します。

```
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: cmd parse result:
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: input  file name: 01.yuv
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: output file name: 01.h265
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: width      : 1920
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: height     : 1080
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: format     : 0
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_utils: type       : 16777220
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: mpi_enc_test start
Nov 21 10:40:59 lubancat mpp[2534]: mpp_info: mpp version: e34f0dd1 author: Herman Chen   2023-07-17 [hal_vp8e]: Fix crash on unsupport input format
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: 0x7f94001960 encoder test start w 1920 h 1080 type 16777220
Nov 21 10:40:59 lubancat mpp[2534]: mpp_enc: MPP_ENC_SET_RC_CFG bps 7776000 [486000 : 8262000] fps [30:30] gop 60
Nov 21 10:40:59 lubancat mpp[2534]: h265e_api: h265e_proc_prep_cfg MPP_ENC_SET_PREP_CFG w:h [1920:1080] stride [1920:1080]
Nov 21 10:40:59 lubancat mpp[2534]: mpp_enc: mode vbr bps [486000:7776000:8262000] fps fix [30/1] -> fix [30/1] gop i [60] v [0]
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 0    size 81279   tid 0 qp 13
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 1    size 1802    tid 0 qp 13
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 2    size 397     tid 0 qp 13
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 3    size 851     tid 0 qp 12
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 4    size 831     tid 0 qp 12
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 5    size 1510    tid 0 qp 11
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 6    size 410     tid 0 qp 11
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 7    size 318     tid 0 qp 11
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 8    size 463     tid 0 qp 11
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 9    size 2249    tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 10   size 402     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 11   size 257     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 12   size 323     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 13   size 3817    tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 14   size 314     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 15   size 236     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 16   size 5320    tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 17   size 713     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 18   size 6245    tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encoded frame 19   size 716     tid 0 qp 10
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: chn 0 encode 20 frames time 355 ms delay  14 ms fps 56.19 bps 1301436
Nov 21 10:40:59 lubancat mpp[2534]: mpi_enc_test: mpi_enc_test average frame rate 56.19
```

24.4.3.2 h265 を mp4 に変換

h265 を mp4 に変換するには、FFmpeg ツールを使用します。

```
sudo apt update && sudo apt install -y ffmpeg # FFmpeg ツールのインストール
ffmpeg -i 01.h265 -c:v libx265 -c:a aac -f mp4 0101.mp4
```

ヒント：ここで、01.h265 は変換する H.265 ビデオファイル名で、0101.mp4 は変換後の MP4 ファイル名です。-c:v libx265 はビデオを H.265 フォーマットでエンコードし、-c:a aac はオーディオを AAC フォーマットでエンコードし、-f mp4 は出力フォーマットを MP4 に指定します。

24.5 実用ツール

MPP は、ソフトウェアおよびハードウェアプラットフォームや MPP ライブラリ自体をテストするためのユニットテスト用ツールプログラムをいくつか提供しています。

mpp_info_test	MPP ライブラリのバージョン情報を読み取り、表示します。問題を報告する際には、表示された情報を添付できます。
mpp_buffer_test	カーネルのメモリアロケータが正常かどうかをテストします。
mpp_mem_test	C ライブラリのメモリアロケータが正常かどうかをテストします。
mpp_platform_test	いくつかのソフトウェアおよびハードウェアのランタイム環境が正常かどうかをテストします。
mpp_runtime_test	チッププラットフォーム情報の読み取りとテストを行います。

25 キーボードとマウスの共有——Synergy

25.1 Synergy について

Synergy は、ネットワークを介して複数のコンピュータ間でキーボードとマウスを共有できるソフトウェアで、Windows、Linux、Mac などのシステムをサポートしています。

Synergy のコードリポジトリ：https://github.com/symless/synergy-core

Synergy のコンパイルガイド：https://github.com/symless/synergy-core/wiki/Compiling

25.2 準備

- ボードと PC は同じローカルネットワーク内にあり、インターネットに接続できること
- ボードのシステムは Ubuntu または Debian であること

25.3 ボードに Synergy をインストールする

Synergy のソースコードを修正およびコンパイルし、起動時に自動で起動する機能を追加しました。この修正版はのソフトウェアリポジトリにアップロードされています（ボードの

オープンソースミラーにはデフォルトで搭載されています）。ユーザーは自身のシステムに対応するバージョンの Synergy をインストールするだけです。

Debian システムユーザーの場合：

```
sudo apt install synergy-debian
```

Ubuntu システムユーザーの場合：

```
sudo apt install synergy-ubuntu
```

ヒント：インストールが完了したら、ボードを再起動してソフトウェアを起動できます。Ubuntu システムではデスクトップに自動ログインする設定が必要です。

25.4 PC に Synergy をインストールする

25.4.1 リソースの取得

PC 用の Synergy リソースは以下のリンクから取得できます：

https://github.com/FLZeng/synergy-core/releases -

-

https://www.luoxx.top/archives/synergy-free-share

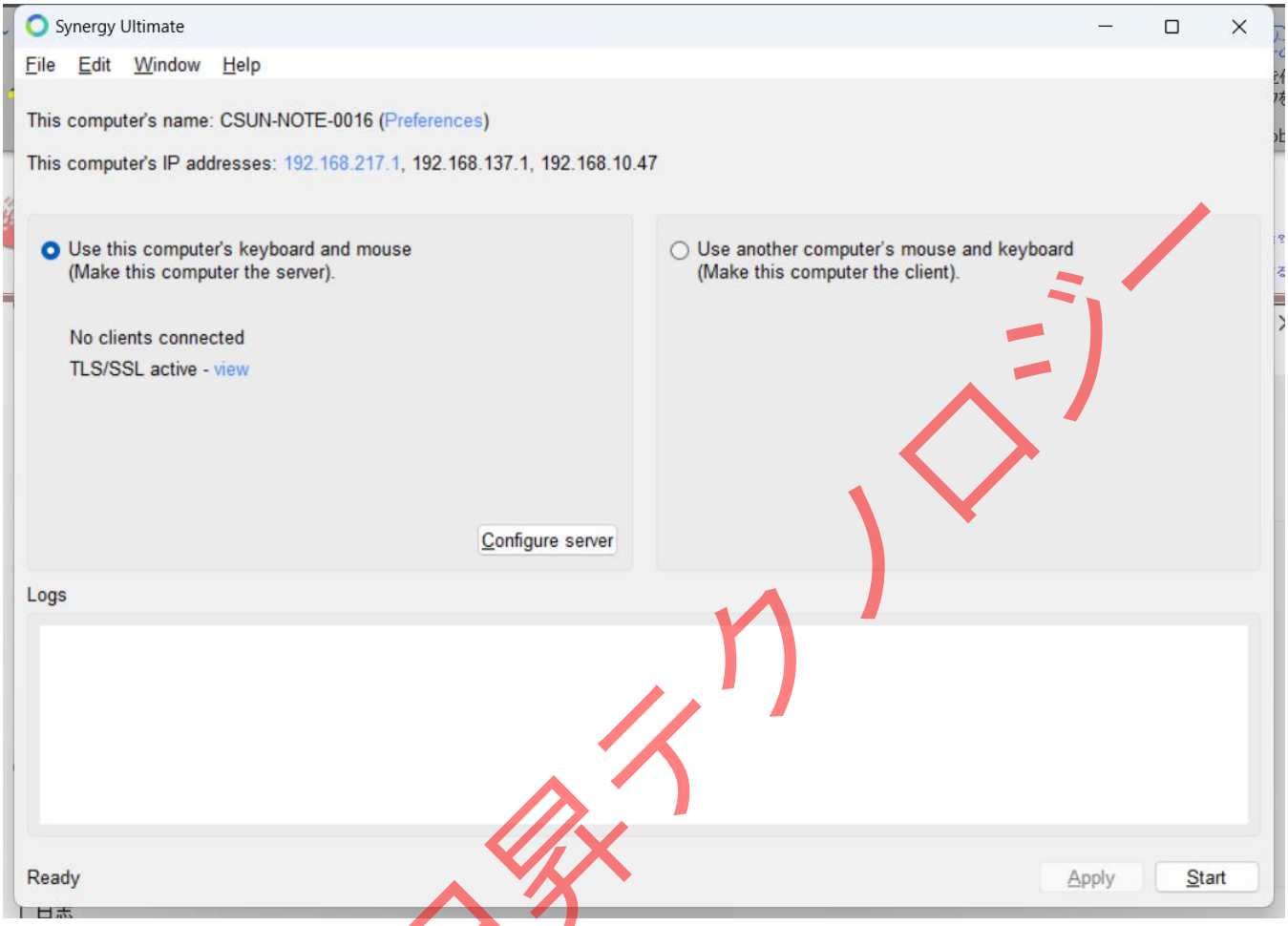
25.4.2 インストール手順

Windows システムユーザーの場合、インストーラ不要のバージョンをダウンロードした場合は、解凍後に synergy.exe を直接実行します。msi インストーラをダウンロードした場合は、msi ファイルを実行してインストールします。

macOS システムユーザーの場合、対応する dmg インストーラを実行してインストールします。

インストールが完了したら、Synergy を起動します。Synergy の画面は以下のようになりま

す：



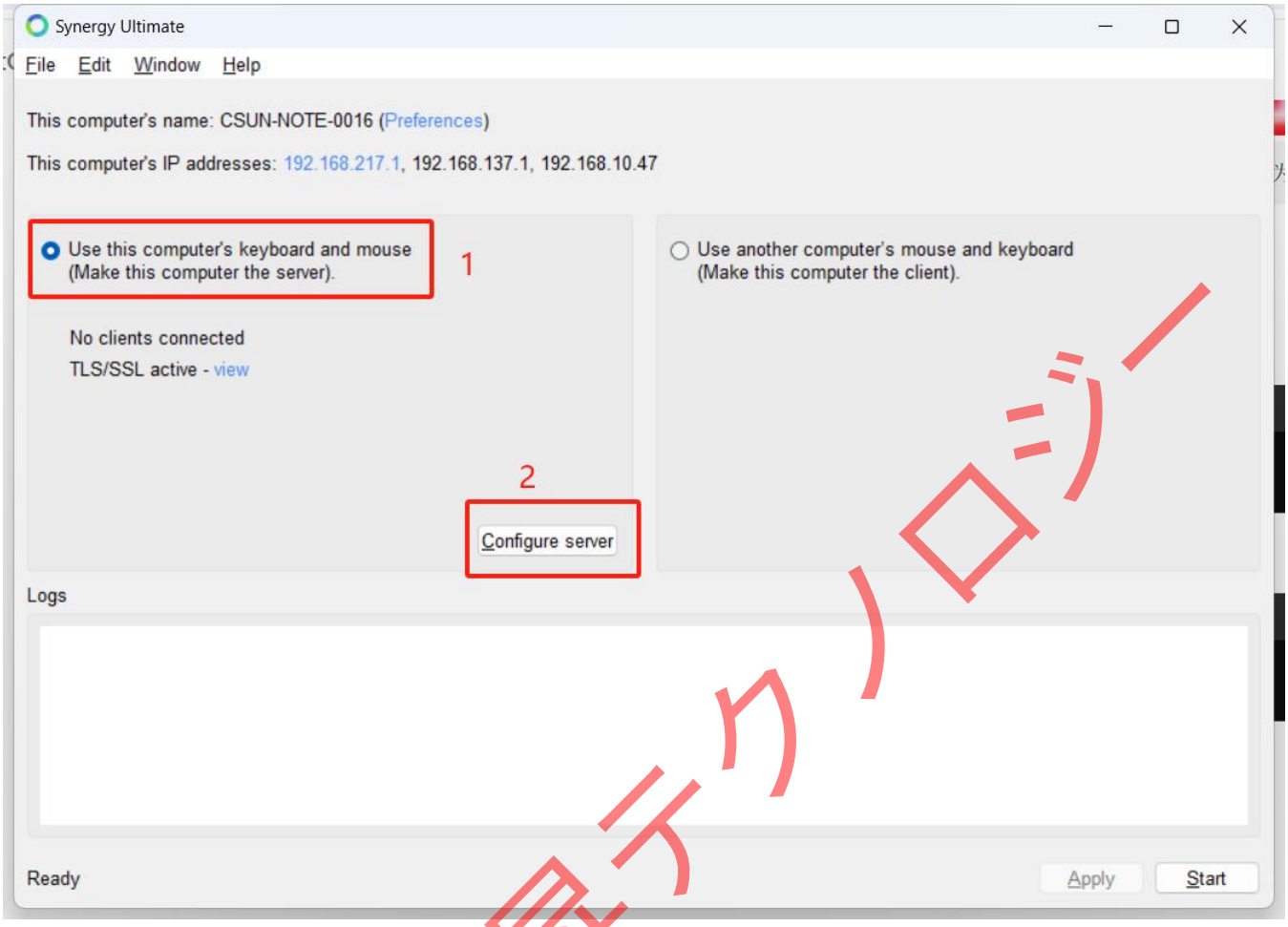
ヒント：PC の IP アドレスを確認し、ボードと接続する際に使用します。

25.5 キーボードとマウスの共有方法

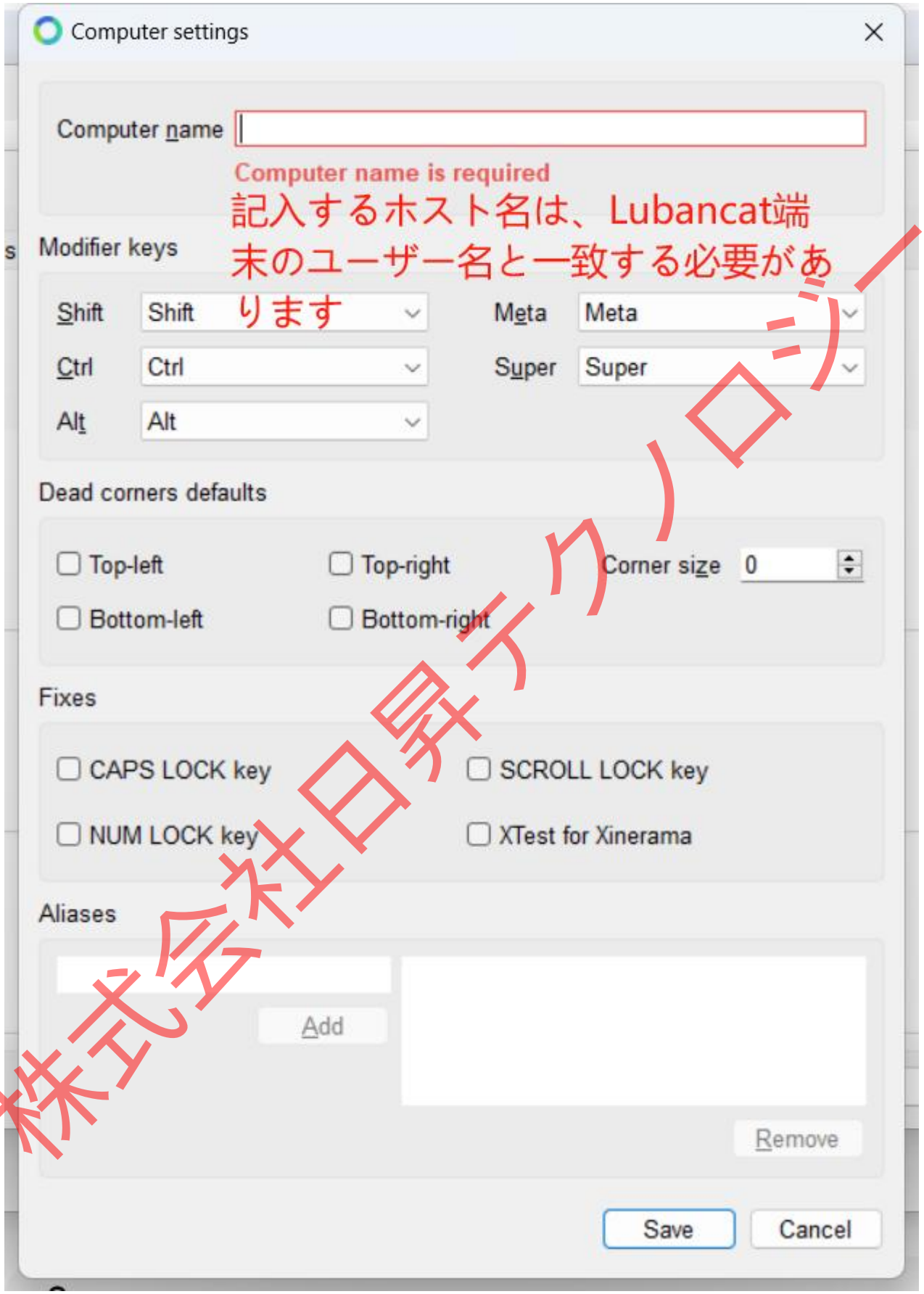
25.5.1 PC 側（サーバー側）の操作

ヒント：以下の手順は Windows システムを例に示します。

1. Synergy ソフトウェアを開き、サーバー側の設定に入ります。

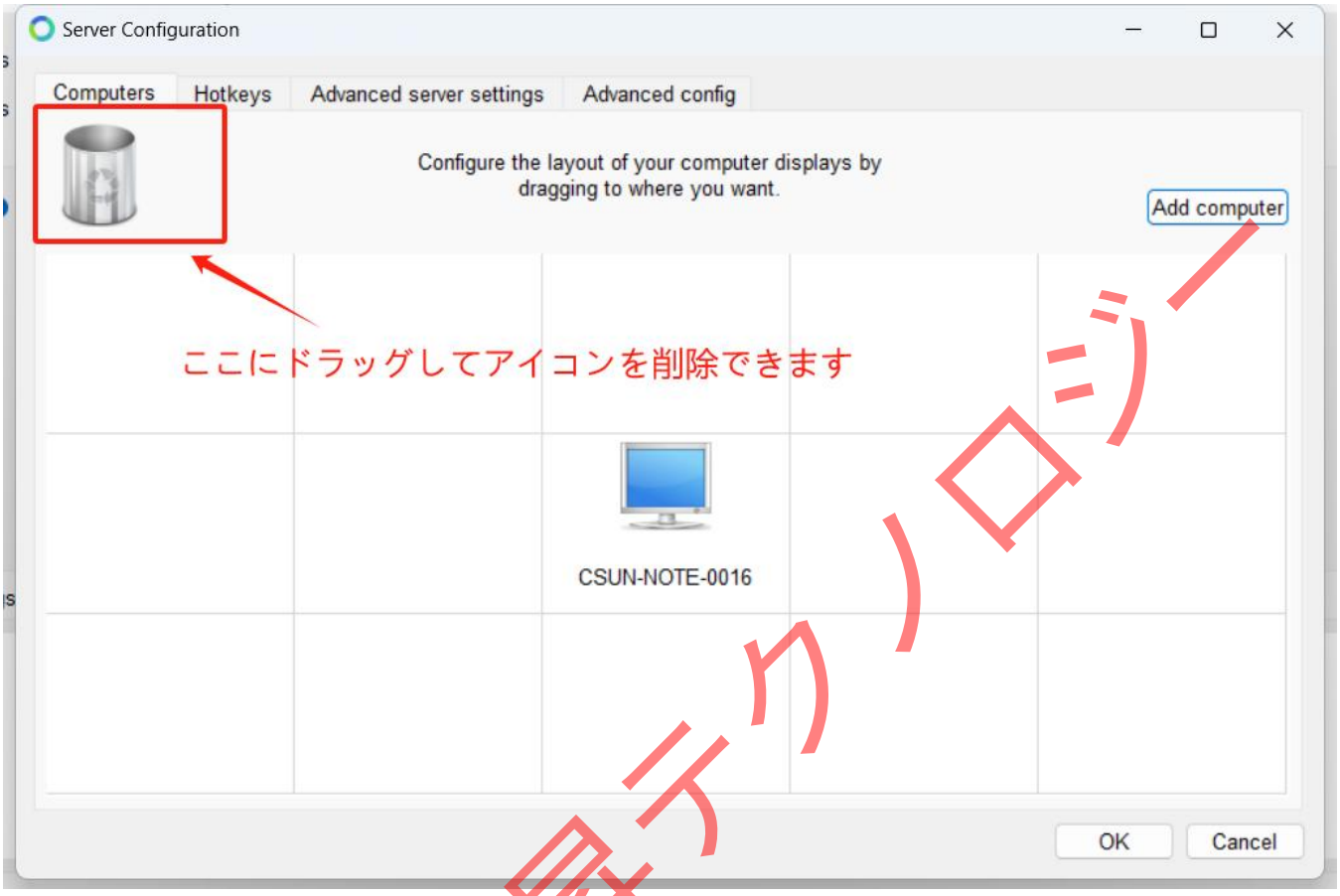


2. 設定画面に入ったら、右上の「コンピュータを追加」をクリックします。



3. アイコンをドラッグして、ボードデスクトップと PC の位置を調整し、右下の「OK」

をクリックして確定します。

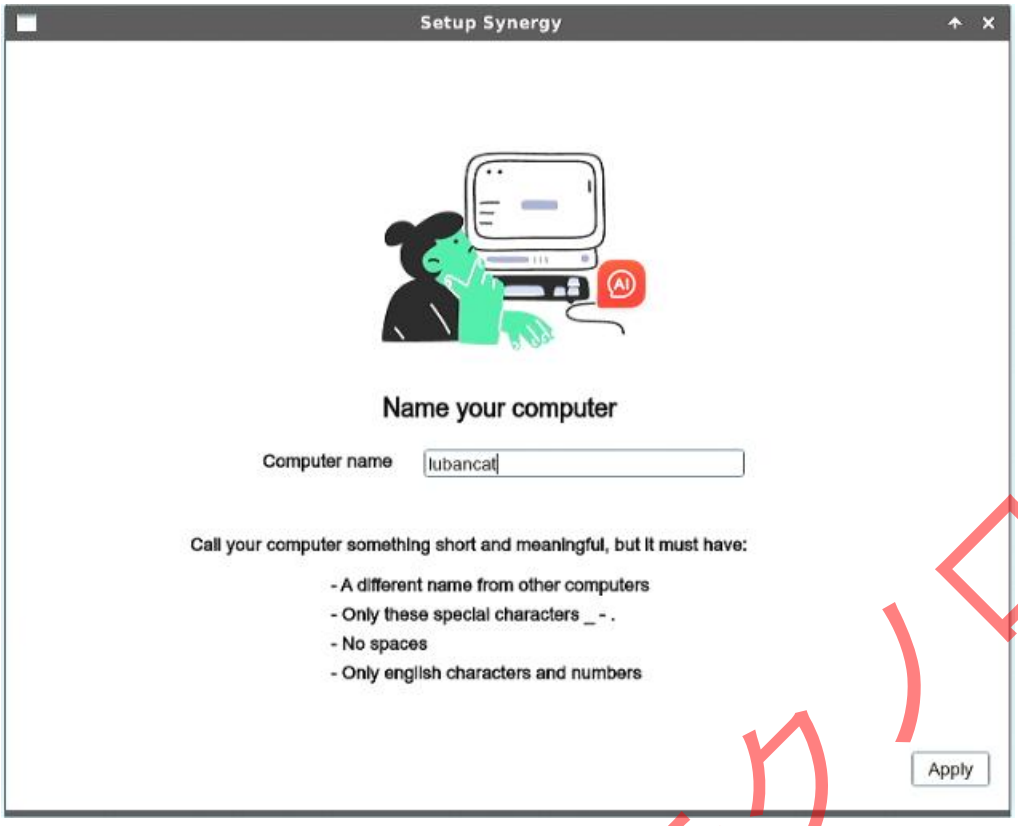


4. メイン画面に戻り、右下の開始ボタンをクリックして起動します。

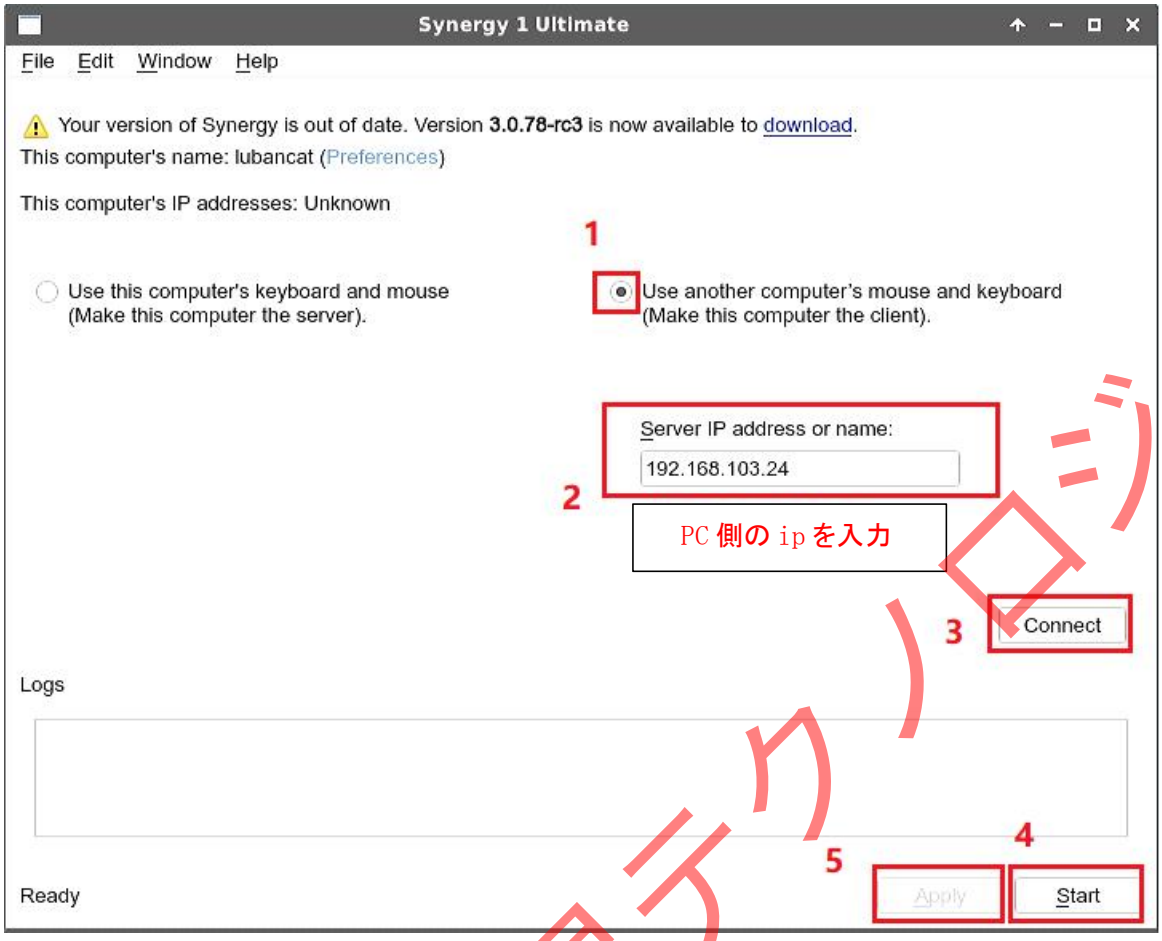
25.5.2 ボード側（クライアント側）の操作

ヒント：以下の手順は Debian システムを例に示します。初回使用時にはキーボードとマウスをボードに接続して設定を行います。

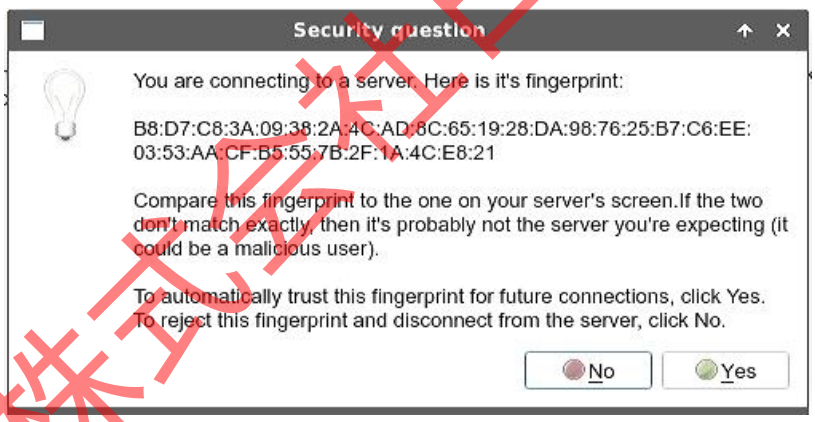
1. ソフトウェアのインストール後、再起動すると Synergy の画面に自動で入ります。初回使用時にはコンピュータ名を設定する必要があります。ここでは「lubancat」と設定します（このコンピュータ名は PC のサーバー側設定と一致させる必要があります）。



2. 設定が完了したらメイン画面に入り、以下の順序で操作を行います。ステップ 2 では PC の Synergy 画面に表示される IP を入力します（PC 側の Synergy 画面に表示されるホスト名を使用しないでください、大抵の場合接続できません）。



3. 初回接続時には、以下のような接続認証が表示される場合がありますが、「Yes」をクリックしてください。



4. 接続が完了すると、左下に「Synergy is connected （with TLSv1.3）」と表示されます。この時、PC 側の画面にも「Synergy is connected with TLSv1.3」と表示され、PC とボードがキーボードとマウスを共有していることが示されます。

ヒント：PC 側のキーボードとマウスを使用してボードを操作することができ、クリップボ

ードの内容も共有することができます。

以上。