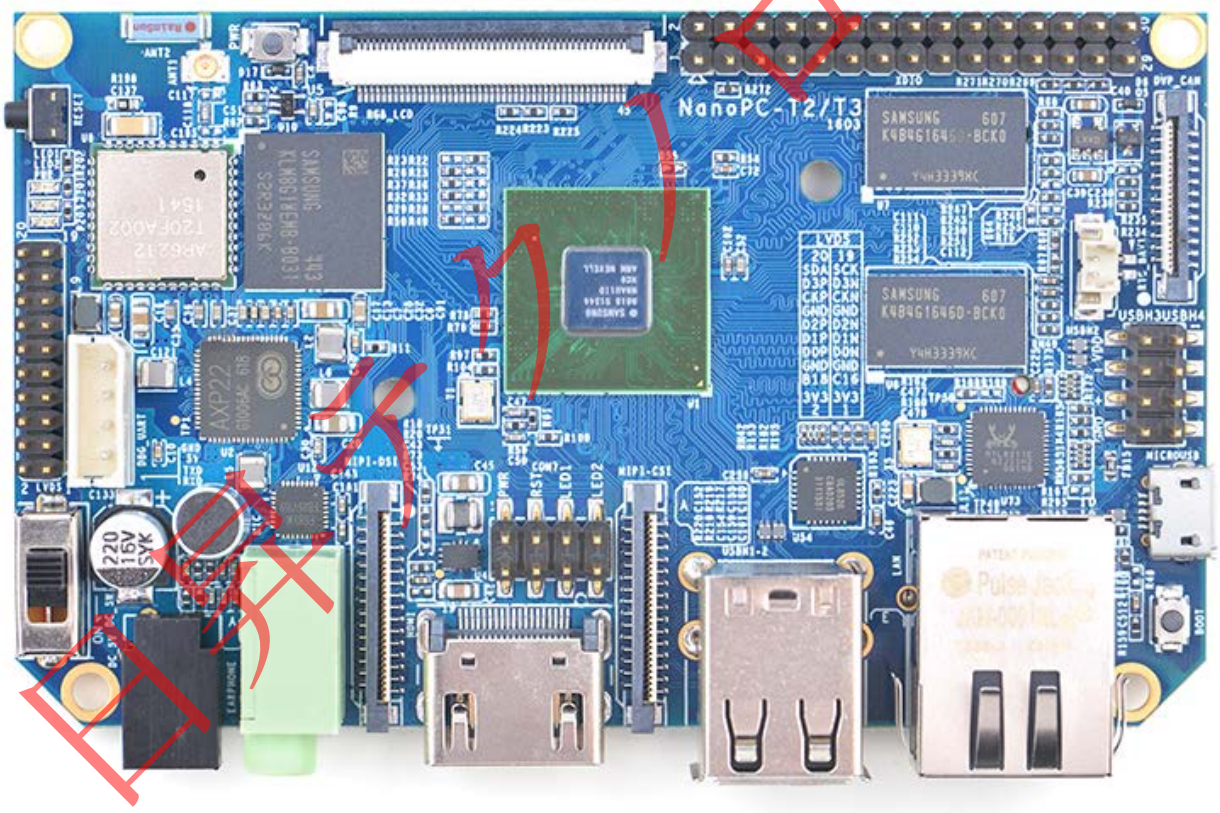
Cortex-A53 8 コア S5P6818 ボード
NanoPC-T3 簡易マニュアル

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

作成日 2016/6/9



- **修正履歴**

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2016/6/9

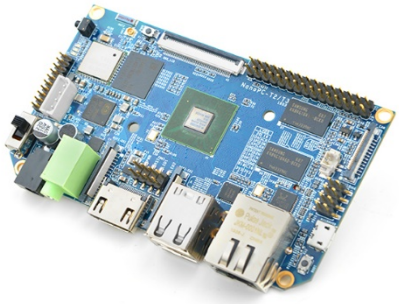
目　次

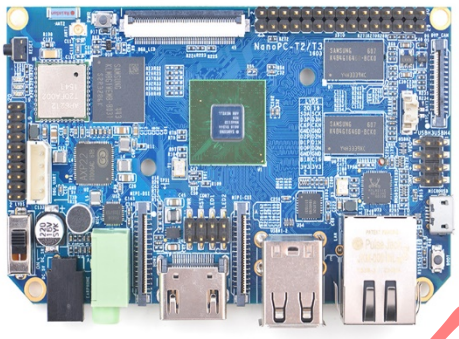
1 紹介

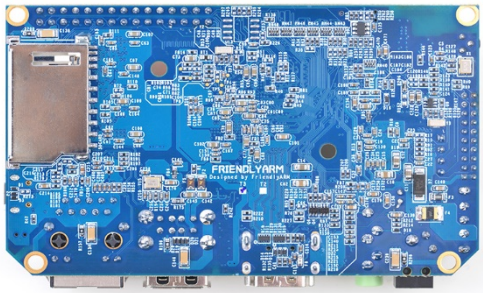
NanoPC-T3はSamsungのオクタコア・Cortex-A53・S5P6818 SoCを備えている。既存の4418ベースのボードNanoPC-T2に比べて、NanoPC-T3は、T2の全てのインターフェースとポートが搭載されているだけでなく更に強力なSOCも備えている。そのダイナミックな周波数スケールは400Mから1.4GHzまでである。NanoPC-T3は8G eMMCを搭載し、また、オーディオジャック、ビデオ入力/出力インターフェース、内蔵WIFI、ブルートゥース、Gbps イーサネット・ポートが装備されている。また、NanoPC-T3は ボード上に磁器アンテナとシリアルデバッグポートのパワーマネージメント機能を備えている。オーバーヒートの問題を避けるために、NanoPC-T3は取り付け穴付きヒートシンクを備えている。NanoPC-T3は2つのカメラインターフェース：DVPとMPI-CSIを備えている。また4つのビデオ・インターフェース：HDMI 1.4A、LVDS、バラレルRGB—LCD、MPI-DSIを備えている。RTCをサポートし、RTCのインターフェースピンを搭載。2つのAタイプポートと2つの2.54ミリピッチピンヘッダをもつ4つのUSBポートを備えている。NanoPC-T3はAndroid 5.1、 DebianとUbuntoCore+Qtのmultiple OSシステムをサポートしている。それは豊富なインターフェースとポートを備えたオープンソースプロジェクトです。

2 主な仕様

・SoC： Samsung S5P6818オクタコアのCortex-A53、400MHz～1.4GHz
・電力マネージメントユニット： AXP228 PMU ソフトウェアのパワーオフとウェイクアップをサポートする。
・システムメモリ：1GB 32ビットDDR3 RAM
・SDIO WiFi/Bluetoothモジュール
・USB ホスト：4×USB2.0ホスト、2つのAタイプポート、2つの2.54ミリピッチピンヘッダ
・MicroUSB：1×MicroUSB2.0クライアント、タイプA
・eMMC： 8GB
・ストレージ： 1 x SD カードソケット
・イーサネット： Gbit イーサネットポート（RTL8211E）
・WiFi： 802.11b/g/n
・Bluetooth： 4.0 デュアルモード
・アンテナ：磁器アンテナ IPXインターフェイス
・ビデオ入力: DVP カメラ/MIPI-CSI（2つのカメラインターフェース）
・ビデオ出力：HDMIタイプA/LVDS/LCD/MIPI DSI(4つのビデオ出力インターフェイス）
・オーディオ：3.5ミリオーディオジャック/HDMI経由
・マイク：オンボードマイクx1
・USB：USB2.0ホスト×4、標準タイプAポート×2と2.54ミリピッチのピンヘッダ×2
・LCDインターフェイス：0.5 mmピッチ45ピンFPCシート、フルカラー（RGB：8-8-8）
・マイクロUSB：1×マイクロUSB2.0クライアント、タイプA
・HDMI：1.4A、タイプA、1080P
・DVPカメラ： 0.5ミリピッチ、24ピン、FPC用シート
・GPIO：2.54ミリピッチ、30ピンヘッダー
・シリアルデバッグポート：2.54ミリピッチ、4ピンヘッダー
・ユーザーキー：K1(パワー)、リセット
・LED: 電源LED×1、GPIO LEDx2
・その他のリソース：CPUの内部TMU
・RTCバッテリー：RTCバッテリーシート
・ヒートシンク：1×ヒートシンク取付け穴付き
・PCB：6レイヤ
・PCBサイズ：100mm×60mm
・OS/ソフトウェア:U ブート、Android、Debian

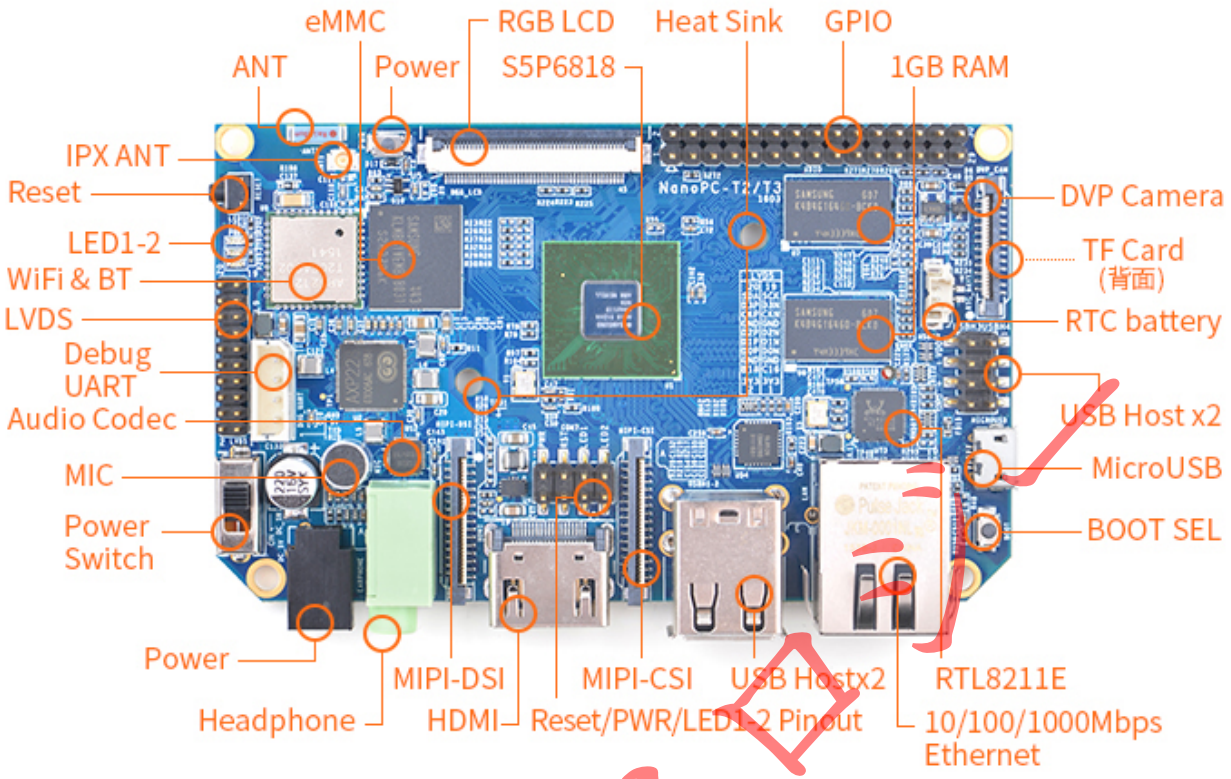






3 インターフェースの配置及びサイズ

3.1 インターフェースの配置



3.1.1 30Pin GPIO ピン定義

ピン	名前	ピン	名前
1	SYS_3.3V	2	DGND
3	UART2_TX/GPIOD20	4	UART2_RX/GPIOD16
5	I2C0_SCL	6	I2C0_SDA
7	SPI0_MOSI/GPIOC31	8	SPI0_MISO/GPIOD0
9	SPI0_CLK/GPIOC29	10	SPI0_CS/GPIOC30
11	UART3_TX/GPIOD21	12	UART3_RX/GPIOD17

13	UART4_TX/GPIOB29	14	UART4_RX/GPIOB28
15	GPIOB31	16	GPIOB30
17	GPIOC4	18	GPIOC7
19	GPIOC8	20	GPIOC24
21	GPIOC28	22	GPIOB26
23	GPIOD1/PWM0	24	GPIOD8/PPM
25	GPIOC13/PWM1	26	AliveGPIO3
27	GPIOC14/PWM2	28	AliveGPIO5
29	VDD_5V	30	DGND

3.1.2 20 ピン LVDS インターフェイス ピン定義

ピン	名前	ピン	名前
1	SYS_3.3V	2	SYS_3.3V
3	GPIOC16	4	GPIOB18
5	DGND	6	DGND
7	LVDS_D0−	8	LVDS_D0+

9	LVDS_D1−	10	LVDS_D1+
11	LVDS_D2−	12	LVDS_D2+
13	DGND	14	DGND
15	LVDS_CLK−	16	LVDS_CLK+
17	LVDS_D3−	18	LVDS_D3+
19	I2C2_SCL	20	I2C2_SDA

3.1.3 DVP カメラ IF ピン定義

ピン	名前
1、2	SYS_3.3V
7,9,13,15,24	DGND
3	I2C0_SCL
4	I2C0_SDA
5	GPIOB14
6	GPIOB16
8,10	NC

11	VSYNC
12	HREF
14	PCLK
16-23	データ bit7-0

3.1.4 RGB LCD IF ピン定義

ピン番号	名前	説明
1, 2	VDD_5V	5V 出力 — LCD モジュールに電源を供給するために使うことが可能
11, 20, 29, 37, 38, 39, 40, 45	DGND	グランド
3-10	ブルーの LSB から MSB へ	RGB のブルー信号
12-19	グリーンの LSB から MSB へ	RGB のグリーン信号
21-28	赤の LSB から MSB へ	RGB の赤信号
30	GPIOB25	ユーザーがコントロール可能
31	GPIOC15	LCD モジュールと制御バックライトを認識し、抵抗式タッチを実現するワンワイヤ技術はユーザーに適用できない。

32	XnRSTOUT Form CPU	システムがリセットされる時、出力のレベルは低い
33	VDEN	RGB 信号が有効な信号であることを示す
34	VSYNC	垂直同期
35	HSYNC	水平同期
36	LCDCLK	LCD クロック，ピクセル周波数
41	I2C2_SCL	静電容量式タッチのデータ伝送のための I2C2 クロック信号
42	I2C2_SDA	静電容量式タッチのデータ伝送のための I2C2 データ信号
43	GPIOC16	静電容量式タッチ用のピンを中断・I2C2 と使用される
44	NC	接続されていない

3.1.5 MIPI-DSI インタフェース ピン定義

ピン番号	名前
1、2、3	VDD_5V
4	DGND
5	I2C2_SDA
6	I2C2_SCL

7	DGND
8	GPIOC0
9	DGND
10	GPIOC1
11	DGND
12	GPIOA28
13	nRESETOUT
14、15	DGND
16	MIPIDSI_DN3
17	MIPIDSI_DP3
18	DGND
19	MIPIDSI_DN2
20	MIPIDSI_DP2
21	DGND
22	MIPIDSI_DN1

23	MIPIDSI_DP1
24	DGND
25	MIPIDSI_DN0
26	MIPIDSI_DP0
27	DGND
28	MIPIDSI_DNCLK
29	MIPIDSI_DPCLK
30	DGND

3.1.6 MIPI-CSI インタフェース ピン定義

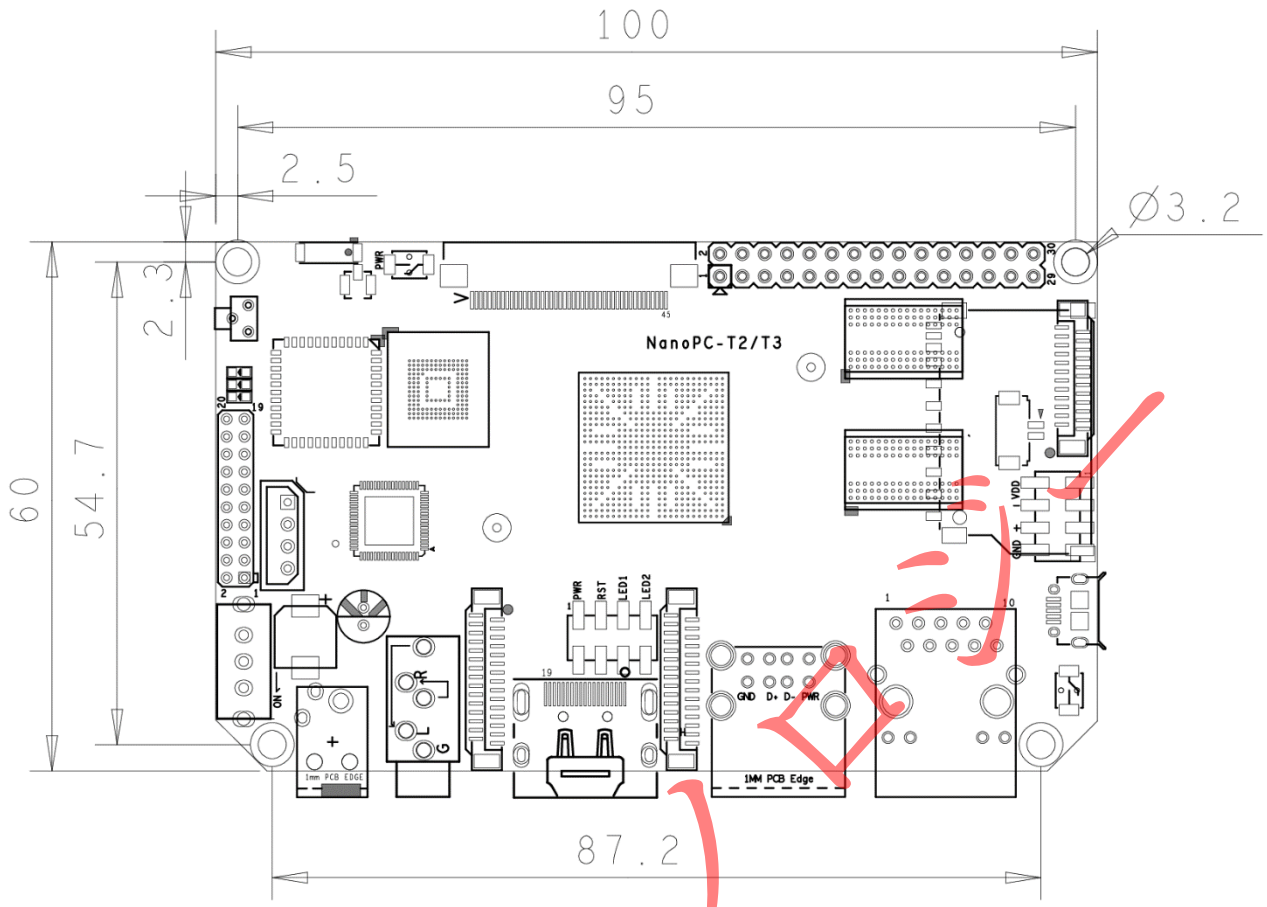
Pin#	Name
1, 2	SYS_3.3V
3	DGND
4	I2C0_SDA
5	I2C0_SCL
6	DGND

7	SPI2_MOSI/GPIOC12
8	SPI2_MISO/GPIOC11
9	SPI2_CS/GPIOC10
10	SPI2_CLK/GPIOC9
11	DGND
12	GPIOB9
13	GPIOC2
14, 15	DGND
16	MIPICSI_DN3
17	MIPICSI_DP3
18	DGND
19	MIPICSI_DN2
20	MIPICSI_DP2
21	DGND
22	MIPICSI_DN1

23	MIPICSI_DP1
24	DGND
25	MIPICSI_DN0
26	MIPICSI_DP0
27	DGND
28	MIPICSI_DNCLK
29	MIPICSI_DPCLK
30	DGND

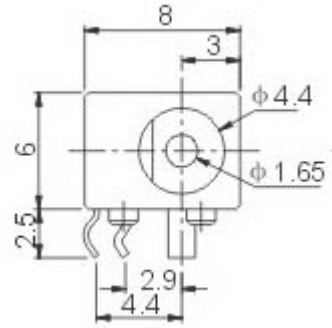
1. VDD_SYS_3.3V：3.3V電源の出力
2. VDD_5V：5 V電源入力/出力。電圧がMicroUSBより高い場合、ボードに給電、そうでない場合、ボードはMicroUSBから電源を取る。入力範囲：4.7～5.6V。
3. 更に詳しい情報については回路図をチェックしてください。

3.2 PCB サイズ


100
95
2.5
Ø3.2
2.3
60
54.7
87.2
NanoPC-T2/T3
PWR
RST
LED1
LED2
GND D+ D- PWR
1MM PCB Edge
1mm PCB EDGE
ON
OFF

詳細についてはドキュメントを参照してください。

- **電源ジャック**
  - DC 4.7～5.6V IN、4.0＊1.7 ミリメートル電源ジャック



EEPROM について

・このボードはユニークな MAC 搭載の EEPROM を装備している。この EEPROM は I2C0 に接続され、そのアドレスは 0×51 であるため、EEPROM の中には I2C0 に接続できないものもある。接続してしまうと、アドレスの衝突の原因になる。

・弊社のテストにおいて、EEPROM のチップは I2C0：24C04、24C08、24C16 に接続不可である。我々テストしたこれらのチップは I2C0: 24C01、 24C02、 24C256 接続可能である。

・EEPROM アドレス発行についての詳細は以下のサイトを参照。

http://www.onsemi.com/pub_link/Collateral/CAT24C01-D.PDF

4 クイックスタート

4.1 ハードウェアの準備

・NanoPC-T3ボード
・SDカード/ TFカード：Class10以上の8GBのSDHCカードが必要
・DCインタフェースの外部電源、5V/2A
・HDMIモニタまたはLCD
・USBキーボード、USBマウス、同時に使う場合はUSB HUB（またはシリアルボードへのTTL）
・Ubuntu 14.04 64ビットシステムを推奨する。

4.2 実行システムを持つ microSD カードを作成する

4.2.1 SD カードから NanoPC-T3 を高速でスタート

次のファイルをHPからダウンロードしてください。

・4G SDHCカードを取得し、そのデータをバックアップする。

LCD または HDMI 出力の場合は、次のファイルを使用してください：	
s5p6818-debian-sd4g-20160426.img	Debian のイメージファイル

s5p6818-android-sd4g-20160426.img	Android のイメージファイル
s5p6818-core-qte-sd4g-20160426.img	Ubuntu Core + QT のイメージファイル
フラッシュユーティリティ:	
win32diskimager.rar	Windows ユーティリティ。Linux でユーザーは“dd”を使用できる。

・これらのファイルを解凍する。SDカードをWindowsのPCに挿入し、win32diskimager ユーティリティを管理者として起動させる。ユーティリティのメインウィンドウでSDカードのドライブとイメージファイルを選択し、SDカードを点滅させるために【Write】をクリックする。
・このカードをNanoPC-T3のブートソケットに挿入し、ブートキーを押して保持し、電源をオンにする（5V/2Aの電源）。PWR LEDがオンでLED1が点滅している場合、NanoPC-T3が正常に起動していることを示します。

4.2.2 eMMC から NanoPC-T3-を起動する

・RAW Image をダウンロードする
イメージファイル【s5p6818-eflasher-sd4g.img.zip】とウィンドウズツール【win32diskimager.zip】を取得する。

・RAWImageをSDカードにフラッシュする
WindowsPCにSDカード（4G以上）を挿入し、アドミニストレータとしてwin32diskimagerツールを起動する。メイン画面で、あなたのSDカードのドライブとイメージファイルを選択し、 [Write]をクリックする。

・RAWImageを準備する
AndroidとDebianのイメージファイル（System-image-files-for-eMMC）をダウンロードする。ダウンロードした後、 ″.tgz″ 圧縮フォルダを解凍し、それをSDカードにコピーする。

OS	Image	Files	コピー先
Android 5.1	android-lollipop-images.tgz android-lollipop-images.tgz.hash.md5	boot.img system.img userdata.img cache.img partmap.txt	images¥android
Debian (Jessie)	debian-jessie-images.tgz debian-jessie-images.tgz.hash.md5	boot.img rootfs.img	images¥debian

		partmap.txt	
Ubuntu Core + QT	core-qte-images.tgz core-qte-images.tgz.hash.md5	boot.img rootfs.img partmap.txt	images¥core-qte

・OSを指定する
デフォルトではSDカードの設定フィル[images¥FriendlyARM.ini]はAndroidがEMMCにフラッシュされるように指定する。Debian をインストールしたい場合は次のように変更する。

```
OS = Debian
```

[#]はコメント
・NanoPC-T3のEMMCにImageをフラッシュする
このカードをあなたのNanoPC-T3のeMMCに挿入する。ボードをHDMIまたはLCDに接続し、ブートキーを押したままにし、インストールを始めるためにボードの電源をオンにする（5V/2A電源）。HDMIまたはLCDからイントールの全過程をみることができる。次のメッセージがポップアップされた場合、インストールは成功したことになる。

```
Android is fused successfully.
All done.
```

インストールが完了したら、eMMCからのボードを起動するために、[リセット]あるいはボードの電源をオフにする。

LEDのステータスをチェックしてインストールのプロセスもモニターできる。

LED ステータス	インストールステータス
LED1 が連続で2回点滅 LED2 はオフ	電源がオンの正常な状態 インストールが実行されてない
LED1 と LED2 は 0.3s ごとに交互に点滅	インストール実行中
LED1 と LED2 が 1.2 秒ごとに交互に点滅	インストール成功
LED1 と LED2 が同時に点滅	インストール失敗

4.2.3 Linux Desktop 環境での作成

1)microSDをUbuntuのパソコンに挿入　以下のコマンドでSDカードのデバイス名をチェックする

```
dmesg | tail
```

dmesgが「sdc：sdc1 sdc2」と類似した情報を出力する時、SDカード対応デバイス名は/dev/sdcになる。コマンドcat /proc/partitionsでも確認できる。

2)Linuxのスクリプトをダウンロードする

```
git clone https://github.com/friendlyarm/sd-fuse_nanopi3.git
cd sd-fuse_nanopi3
```

3)AndroidのSDカードを作成する

```
su
./fusing.sh /dev/sdx
```

（注：/dev/sdxを実際のSDカードのデバイスファイル名に変えてください）

初めて使う際、ダウンロードするか確認が必要。Yを押してダウンロードし、N或いは10秒間入力無い場合は取り消しする。

4)DebianのSDカードを作成する

```
./fusing.sh /dev/sdx debian
```

4.2.4 LCD/HDMI の解像度

システムが起動すると、ubootがLCDに接続されているかをチェックする。ubootがLCDを認識した場合には、その解像度を設定する。デフォルトでは、ubootはHDMI720Pへのディスプレイを設定する。LCDの解像度をリセットしたい場合は、カーネル内の[arch/arm/plat-s5p6818/nanopi3/lcds.c] ファイルを修正し、再コンパイルできる。NanoPC-T3がHDMIモニターに接続され、Androidを起動した場合、「EDID」をチェックすることで自動的に適切なHDMIモードに解像度を設定する。NanoPC-T2がHDMIモニターに接続され、Debianを起動した場合、デフォルトでHDMI720Pへの解像度をセットする。この場合、カーネルの設定を変更することで1080Pまでセットできる。

4.3 パソコンで SD カード上のイメージファイルの更新

システムを実行する前に、少し変更したい場合は、本節の内容をご参照ください。

作成したmicroSDカードをLinuxのパソコンに挿入して、SDカードのboot、rootfsをマウントして内容を変更できる。下記の場合変更が必要：

1）カーネルのコマンドラインのパラメータを更新したい場合は、[sd-fuse_nanopi3/tools」の下にある、「fw_setenv」ツールを使用することができる。

現在のコマンドラインを確認する。

```
cd sd-fuse_nanopi2/tools
./fw_printenv /dev/sdc | grep bootargs
```

現在の Android 5.1.1_r6 により SELinux が有効になる。デフォルトモードは enforcing となり、Command Line を通して変更することが可能。

```
./fw_setenv /dev/sdc bootargs XXX androidboot.selinux=permissive
```

直ぐに、permissive モードに変更でき、[XXX]は元の bootargs に置き換える必要がある。

2)カーネルの更新

新バージョンのUbootが起動時にLCDを認識した場合、SDカードのブートパーティションのuImage.hdmiを読み取る。

Androidにおいては、同じファイルであるため、直接新しいコンパイラのuImageで、SDカードのブートパーティションのファイルに交換する。

Debianにおいては、2つのファイルが異なるため、新しいコンパイラをサポートするLCD uImageで、直接SDカードのブートパーティションのファイルに交換する。HDMIのカーネルをサポートする場合は、uImage.hdmiに交換する。

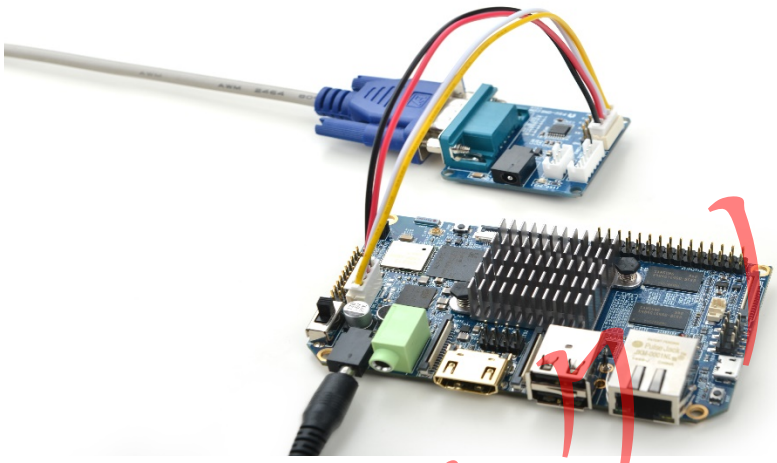
4.4 Android または Debian を実行する

microSDカードをNanoPC-T3に挿入し、HDMIモニターと接続して、電源（5V/2A）に接続すると、NanoPC-T3はSDカードから起動する。PWRLEDとLED1が点滅でシステムが起動していることが確認でき、またHDMIモニターにはAndroidとDebainが確認できる。

1） NanoPC-T3をHDMIモニターに接続したい場合、USBマウス、キーボードが必要である。もしLCDと接続していれば、タッチパネルで操作可能。

2）カーネルを開発する場合、シリアルデバッグポートに接続すれば、端末からNanoPC-T3を操作できる。

シリアルケーブル経由でUbuntuとMinicomoの実行しているPCにNanoPC-T3を接続する場合は次のようになります。



Debianでの[root]のパスワードは[fa]である。

5 Debain システム

5.1 イーサネットに接続する。

NanoPC-T3 が電源を入れる前にイーサネット経由でネット接続された場合、電源をいれた後、PC-T3 は自動的に IP を取得する。

5.2 無線ネットワークに接続する

[Vi]または[gedit]で、ファイル[/etc/wpa_supplicant/wpa_supplicant.conf]を開き、以下の行を追加する。

```
network = {
        SSID = "YourWiFiESSID"
        PSK = "YourWiFiPassword"
}
}
```

[YourWiFiESSID]と[YourWiFiPassword]は実際のESSIDとパスワードと置き換える必要がある。

保存＆終了し、次のコマンドを起動すると、ボードは指定されたWIFIに接続される。以下の方法を参照。

```
ifdownwlan0
 ifupwlan0
```

もし、WIFIのパスワードに特殊文字が含まれているか、パスワードをプレーンテキストとして保存したくない場合、WIFIパスワードのPSKを生成するために、「wpa_passphrase」を使用することができる。

```
wpa_passphrase YourWiFiESSID
```

次にあなたのパスワードのプロンプトタイプを示す。
のファイルを開く場合は、パスワードが更新されていることがわかり、クリアテキストのパスワードを削除できる。

システムのWIFI APモードがオンになっている場合は、検索してワイヤレスルーターに接続することはできない。以下の手順に従い、WIFI APモードをオフにする必要がある。

```
su
turn-wifi-into-apmode no
```

5.3 Wi-Fi 無線ホットスポットの配置

WiFiホットスポットの配置を以下の手順で行う。

```
turn-wifi-into-apmode yes
```

プロンプトに従って、システムを再起動する。デフォルトのホットスポット名は[nanopi2-wifiap]で、パスワードは123456789。PCホストから[nanopi2-wifiap]に接続可能になる。接続が成功すれば、SSHをを介して192.168.8.1でNanoC-T3に登録できる。

```
ssh root@192.168.8.1
```

パスワードは[fa]である。次のコマンドWiFiモードを確認できる。

sshののログインをスムーズ、且つ速くするために次のコマンドを起動し、WiFi無線のパワーセービングモードをオフにする。

```
iwconfig wlan0 power off
```

次のコマンドでWiFiモードをチェックできる

```
cat /sys/module/bcmdhd/parameters/op_mode
```

出力する数字が2であれば、現在のWiFi無線ホットスポットモードとして機能していることを示す。
ステーションモードに切り替えたい場合、以下のコマンドを入力する。:

```
turn-wifi-into-apmode no
```

5.4 Bluetooth を使ってファイルを転送する

GUIの右上にあるBuluetoothのアイコンをクリックすると、メニューが表示される。[Make Discoverable]によってNanoPC-T3 が他のBuluetoothデバイスから検出可能になる。Devices...は検索画面を開き、近くのBluetoothデバイスを検索する（[Make Discoverable]はを先に有効にする必要がある）。[Send Files to Divices]でNanoPC－T3が別のBuluetoothデバイス（NanoPC-T3とペア）にファイルを送ることができる。

5.5 Debian のパッケージソフトをインストールする

提供している画像は標準的なDebian jessieシステムである。apt-getなどのコマンドでパッケージソフトをインストールすることができる。初めてインストールする場合、まず以下のコマンドでパッケージソフトリストを更新する必要がある。

```
apt-get update
```

その後、パッケージソフトをインストールすることができる。例えばFTPサーバーをインストールするには以下のコマンドを使用する。

```
apt-get install vsftpd
```

/etc/apt/sources.listを編集することで、ダウンロードサーバーを変更することができる
http://www.debian.org/mirror/listから全てのサーバーリストが取得可能。 [armhf]が付くリストを選択することが必要。

6 システムのコンパイル方法

6.1 クロスコンパイラをインストールする

先ず、コンパイラをダウンロードして解凍する。

```
git clone https://github.com/friendlyarm/prebuilts.git
sudo mkdir -p /opt/FriendlyARM/toolchain
sudo tar xf prebuilts/gcc-x64/arm-cortexa9-linux-gnueabihf-4.9.3.tar.xz -C /opt/FriendlyARM/toolchain
```

コンパイラのパスをPATHに追加する。viでvi ~/.bashrcを実行して、末尾に以下の内容を追加する。

```
export PATH=/opt/FriendlyARM/toolchain/4.9.3/bin:$PATH
export GCC_COLORS=auto
```

~/.bashrcスクリプトを実行してカレントshellで有効にする。"."の後ろにスペースがある。

```
. ~/.bashrc
```

コンパイラは64ビットのため、32ビットのLinuxでは実行できない。
インストールの完了後、インストールが成功したかを確認できる。

```
arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolchain/4.9.3/libexec/gcc/arm-cortexa9-linux-gnueabihf/4.9.3/lto-wrapper
Target: arm-cortexa9-linux-gnueabihf
Configured with: /work/toolchain/build/src/gcc-4.9.3/configure --build=x86_64-build_pc-linux-gnu
--host=x86_64-build_pc-linux-gnu --target=arm-cortexa9-linux-gnueabihf
--prefix=/opt/FriendlyARM/toolchain/4.9.3
--with-sysroot=/opt/FriendlyARM/toolchain/4.9.3/arm-cortexa9-linux-gnueabihf/sys-root
--enable-languages=c,c++
--with-arch=armv7-a --with-tune=cortex-a9 --with-fpu=vfpv3 --with-float=hard
...
Thread model: posix
gcc version 4.9.3 (ctng-1.21.0-229g-FA)
```

6.2 U-Boot のコンパイル

U-Bootソースコードをダウンロードし、コンパイルする。ブランチは[nanopi2-lollipop-mr1]であることに注意する。

```
git clone https://github.com/friendlyarm/uboot_nanopi2.git
cd uboot_nanopi2
git checkout nanopi2-lollipop-mr1
make s5p6618_nanopi3_config
make CROSS_COMPILE=arm-linux-
```

コンパイルに成功した後、u-boot.binを取得する。Fastbootで、SDカードのUbootを更新する。手順は下記の通り：
1）PCでコマンド [sudo apt-get install android-tools-fastboot]でfastbootツールをインストールする。

2）シリアルデバッグセットでNanoPC-T3とPCを接続する。起動後2秒以内、シリアル端末でEnterを押して、u-bootのコマンドラインモードに入る。

3）u-bootのコマンドラインモードでfastbootコマンドを入力し、Enterを押してfastbootモードに入る。

4）microUSBケーブルでNanoPC-T3とPCを接続する。PC側で下記コマンドを入力してu-boot.binを書き込む。

```
fastboot flash bootloader u-boot.bin
```

注意点：直接ddコマンドでSDカードを更新することはできない。このコマンドは、PC-T3を起動するときに故障の原因になる。

6.3 mkimage を用意する

カーネルをコンパイルするには u-boot の mkimage ツールが必要。因って、カーネル uImage をコンパイルする前に、PC 側で実行できることの確認が必要。

直接 sudo apt-get install u-boot-tools コマンドでインストールできる。或いは自分でコンパイルしてインストールする。

```
cd uboot_nanopi2
make CROSS_COMPILE=arm-linux- tools
sudo mkdir -p /usr/local/sbin && sudo cp -v tools/mkimage /usr/local/sbin
```

6.4 Linux kernel のコンパイル

6.4.1 カーネルのコンパイル

1） カーネルのソースコードをダウンロードする。

NanoPC-T2のカーネルのソースコードは[nanopi2-lollipop-mr1]ブランチにある。

```
git clone https://github.com/friendlyarm/linux-3.4.y.git
cd linux-3.4.y
git checkout nanopi2-lollipop-mr1
```

2) Androidカーネルをコンパイルする。

```
make nanopi3_android_defconfig
touch .scmversion
make uImage
```

3) Debianカーネルをコンパイルする。

```
make nanopi3_linux_defconfig
touch .scmversion
make uImage
```

コンパイル成功後、新しく生成したファイルはarch/arm/boot/uImage、HDMI出力をサポートする。SDカードのbootセクションにある同じファイル名のファイルと置き換えれば良い。

LCD表示をサポートするイメージファイルを作成するには設定を変更する必要がある。

```
touch .scmversion
make nanopi3_linux_defconfig
make menuconfig
  Device Drivers -->
    Graphics support -->
```

```
        Nexell Graphics --->
    [ ] LCD
        [*] HDMI
        (0)    Display In    [0=Display 0, 1=Display 1]
                    Resolution (1920 * 1080p)    --->
make uImage
```

コンパイル成功後、uImageはHDMI　1080P用に生成される。既存のuImageに替えることができます。

6.4.2 新しく生成されたカーネルの使用方法

・SD カード上のカーネルファイルを更新

Android を起動するために SD カードを使用する場合、SD カードのブートセクション(e.g. section 1 /dev/sdX1)に生成された uImage ファイルをコピーすることができる。SD カードを Debian に使用し、HDMI モニター用の uImage を生成した場合、ブートセクションで uImage ファイルを置き換えるために、その uImage を使うことができる。

・EMMC 上の Android のカーネルファイルを更新

EMMC でカーネルファイルを更新したい場合は、まずボードを起動し、eMMC のブートセクションをマウントし、セクションのカーネルファイルを新しく生成されたものと交換する。次に、新しいカーネルを実行するために再起動する。ご自分のボードを eMMC から起動したい場合は次の手順だカーネルファイルを更新することができる。

1）Android をロードした後、次のコマンドで eMMC のブートセクションをマウントする（例：eMMC のデバイス名は /dev/mmcblk0p1）。

```
su
mount -t ext4 /dev/block/mmcblk0p1 /mnt/media_rw/sdcard1/
```

2） Ubunto を使用し、ホスト PC にボードを接続する。次のコマンドを実行し、eMMC のブートセクションに uImge がいすをコピーする。

```
adb push uImage /mnt/media_rw/sdcard1/
```

3） 外部ストレージカード（SD カードまたは USB カード等）に新しく生成されたカーネルファイルをコピーすることがで、そのストレージカードにボードを接続し、そのカードからのファイルを eMMC のブートセクションに移動することができる。

4） 更新終了後、reboot を入力し、Android を再起動する。

・EMMC 上の Debian のカーネルファイルを更新

eMMC からボードを起動する場合、次の手順でカーネルを更新することができる。

1）Debian がロードされている場合は、eMMC のブートセクションは自動的にマントされる。それを確認するために、[mount]を使うことができる。

2）イーサネットを介してホスト PC にボードに接続し、eMMC のブートセクションに SCP/FTP 経由で、生成された uImage ファイルをコピーして、既存のファイルを置き換える

3）または、生成されたカーネルファイルを外部ストレージカードにコピーし、そのカードからのファイルを eMMC のブートセクションに移動することができる。

4）更新終了後、reboot を入力し、Android を再起動する。

・boot.img を生成する

直接 eMMC に書き込みし、イメージファイルを生成したい場合は、boot.img のファイルを生成し、

そのファイルをインストール SD カードにコピーする必要がある。

Andoroid においては、uImage ファイルを Android のソースコードの[device/friendly-arm/nanopi3/boot/]にコピーし、このすべての Android ソースコードをコンパイルする。

Debian においては、boot.img ファイルを生成するために、次の手順に従う。

1)Debian_nanopi2 をダウンロード

```
git clone https://github.com/friendlyarm/debian_nanopi2.git
```

2）HDMI モニター用のイメージファイルをコピーし、それを[debian_nanopi2/boot/uImage.hdmi]ファイルに置き換える。LCD 用のイメージファイルをコピーし、[debian_nanopi2/boot/uImage]ファイルに置き換える。

3）Debian の boot.img を生成する。

```
cd debian_nanopi2
mkdir rootfs
./build.sh
```

新しく生成された boot.img は[debian_nanopi2/sd-fuse_nanopi2/debian]ディレクトリ内に表示される。

コマンド[mkdir rootfs]は the build.sh script が起動するように ワーキングディレクトリを作成する。

6.4.3 カーネルモジュールのコンパイル

Androidはカーネルモジュールを含んでいる。場所はsystemセクションの/lib/modules/である。新しいカーネルモジュール或いはカーネルモジュールの設定が変更した場合、再コンパイルが必要である。

先ず、カーネルソースのモジュールをコンパイルする。

```
cd linux-3.4.y
make CROSS_COMPILE=arm-linux- modules
```

またAndroidのソースに2つのカーネルモジュールのソースがある。下記コマンドでコンパイルする：

```
cd /opt/FriendlyARM/s5p4418/android
./vendor/friendly-arm/build/common/build-modules.sh
```

“/opt/FriendlyARM/s5p4418/android”はAndroidのソースのTOPフォルダである、[-h]パラメータでヘルプ内容を確認できる。

コンパイル成功した後、生成したカーネルモジュールが表示される。

6.5 Andriodシステムのコンパイル

6.5.1 コンパイル環境の構築

64ビットのUbuntu 14.04を推奨する。必要なパッケージをインストールすれば良い。

```
sudo apt-get install bison g++-multilib git gperf libxml2-utils make python-networkx zip
sudo apt-get install flex libncurses5-dev zlib1g-dev gawk minicom
```

詳細内容は下記 URL をご参照ください。

https://source.android.com/source/initializing.html

6.5.2 ソースコードをダウンロードする

Android のソースコードをダウンロードするには repo が必要、インストール方法及び使用方法は下記 URL をご参照ください。https://source.android.com/source/downloading.html

```
mkdir android && cd android
repo init -u https://github.com/friendlyarm/android_manifest.git -b nanopi2-lollipop-mr1
repo sync
```

上記の“android”はワークフォルダーのことである。

6.5.3 システムをコンパイルする

```
source build/envsetup.sh
lunch aosp_nanopi2-userdebug
make -j8
```

コンパイル終了後、out/target/product/nanopi3/のフォルダにイメージファイルが生成される。

filename	partition	Description
boot.img	boot	-
cache.img	cache	-
userdata.img	userdata	-
system.img	system	-
partmap.txt	-	partition file

6.5.4 Image を SD カードに書き込む

SD カードからボードを起動したい場合は、生成されたイメージファイルを[sd-fuse_nanopi3/android/]ディレクトリにコピーし、それをスクリプトを使用し、SD カードに書き込む必要がある。詳細は http://wiki.friendlyarm.com/wiki/index.php/NanoPCT3#_Make_an_Installation_SD_Card_under_Linux_Desktop を参照。

6.5.5 Image を eMMC に書き込む

Android のコンパイルが成功したら、それを下記のいずれかの手順で eMMC に書き込むことができる。

1)Fastboot:NanoPC-T2 が eMMC から起動されたら、Uboot のコマンドラインモードに入るためにいずれかのキーを押し、「FASTBOOT」と入力する。ボードを Uboot が起動している USB ケーブル付きのホスト PC に接続し、PC のターミナルで次のコマンドを実行する。

```
  cd out/target/product/nanopi3
sudo fastboot flash boot boot.img
sudo fastboot flash cache cache.img
sudo fastboot flash userdata userdata.img
sudo fastboot flash system system.img
  sudo fastboot reboot
```

2）SD カードを使う

boot.img, cache.img, userdata.img, system.img, partmap.txt from the out/target/product/nanopi3 directory to your installation SD card's images/android のファイルをコピーする。この SD カードで eMMC に画像を書き込むことができる。

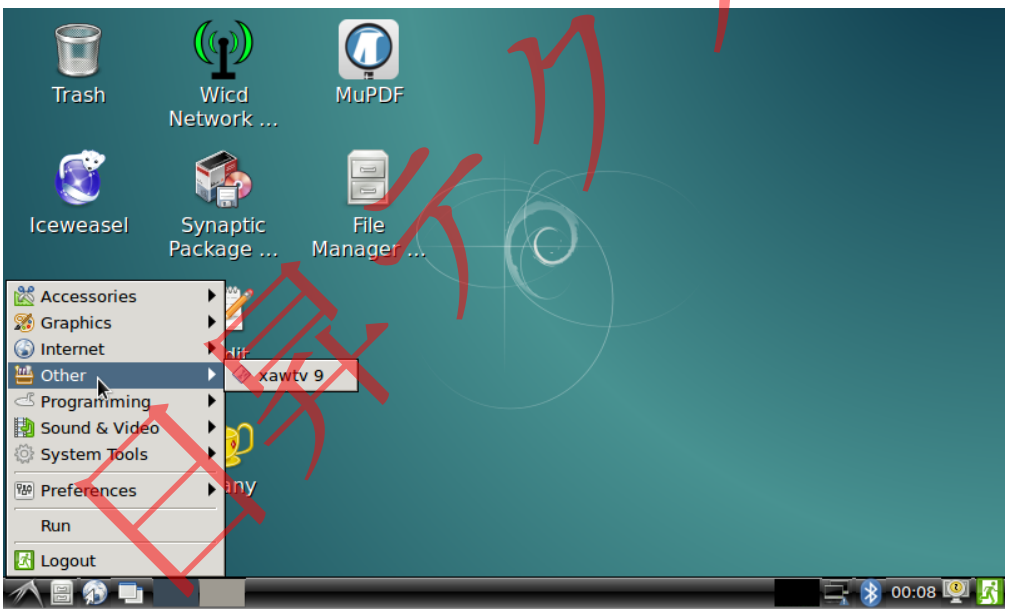
7 拡張モジュール

7.1 USB(FA-CAM202)200 万画素カメラモジュール

1、Debain システムを起動する。

2、左下のメニューから"Other"->"xawtv9"をクリックする。

3、"Welcome to xawtv9!"ダイアログから"OK"をクリックすると画像摂取が出来る。



7.2 OV5640 CMOS 500 万画素カメラモジュール

1、Android システムを起動する。

2、"Camera"をクリックする。

7.3 OpenCV を使用して USB カメラにアクセスする

OpenCV はオープンソースのコンピュータ向けライブラリ であり、クロスプラットフォーム・ビジョンライブラリである。NanoPC-T3 が実行されると、Debian ユーザーは USB カメラデバイスにアクセスするために OpenCV の API を使用することができる。

1. 次に紹介しているのは NanoPC-T3 に C++で OpenCV を使用する方法についてのガイドラインである。

・先ず、NanoPC-T3 がシリアル端末または SSH 経由で internet.Login に接続されていることを確認する必要がある。ログイン後、ユーザーネーム（root）とパスワード（fa）を入力する。

・次のコマンドを実行する。

```
apt-get update
apt-get install libcv-dev libopencv-dev
```

2. USB カメラが NanoPC-T3 で作動していることを確認する。NanoPC-T3 のカメラテストプログラムを使用し、カメラのテストができます。
3. 使用しているカメラのデバイスを確認する。

```
ls /dev/video*
```

注意： video0 が USB カメラデバイス。

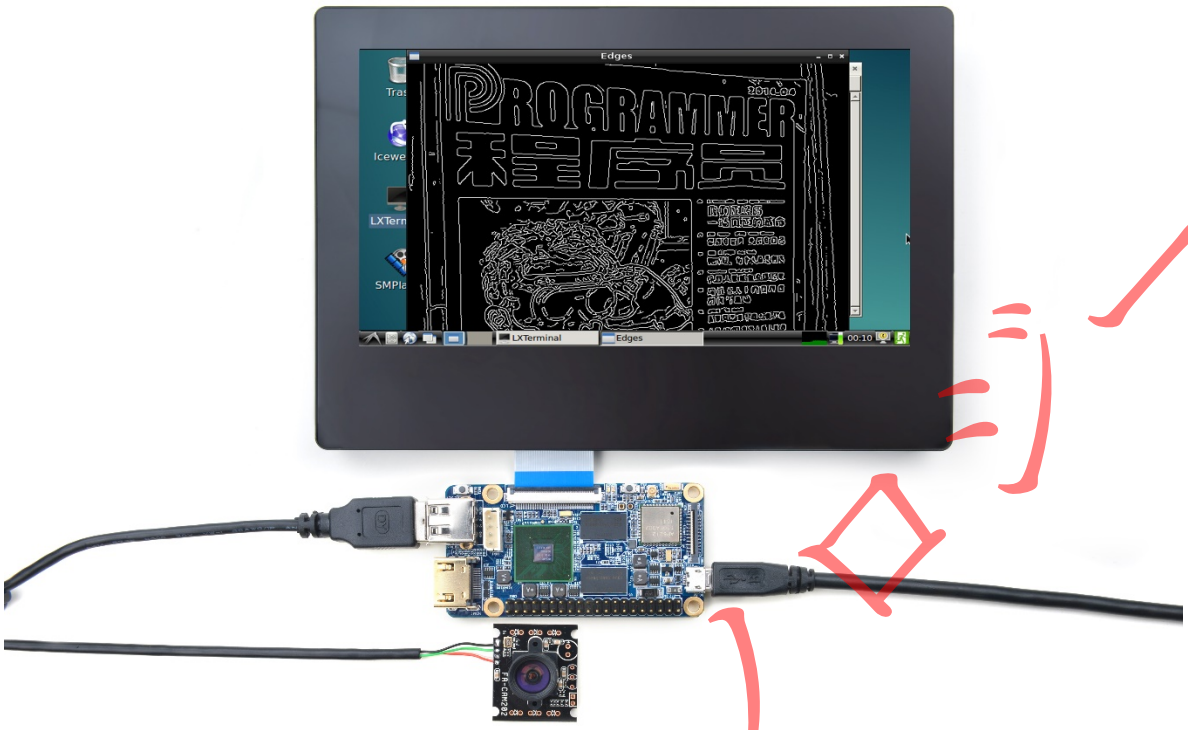
4. OpenCV のコードサンプル（C++における公式コード）は/home/fa/Documents/opencv-demo の下にある。次のコマンドでコードサンプルをコンパイルする。

```
cd /home/fa/Documents/opencv-demo
make
```

正常にコンパイルが完了した[demo] 実行ファイルが生成される。

5. NanoPC-T3 に USB キーボードを接続し、次のコマンドを実行する。

```
./demo
```



以上。