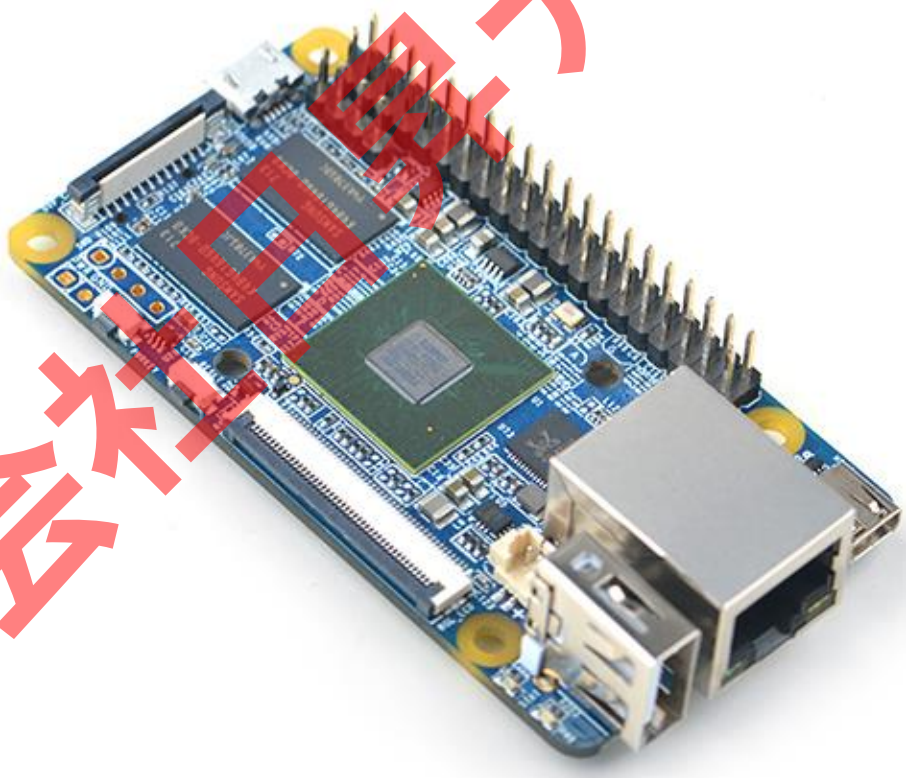
Cortex-A53 8コア S5P6818 ボード
NanoPi-Fire3 簡易マニュアル

株式会社日昇テクノロジー

https://www.csun.co.jp

info@csun.co.jp

作成日 2020/1/20



- **修正履歴**

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2020/1/20

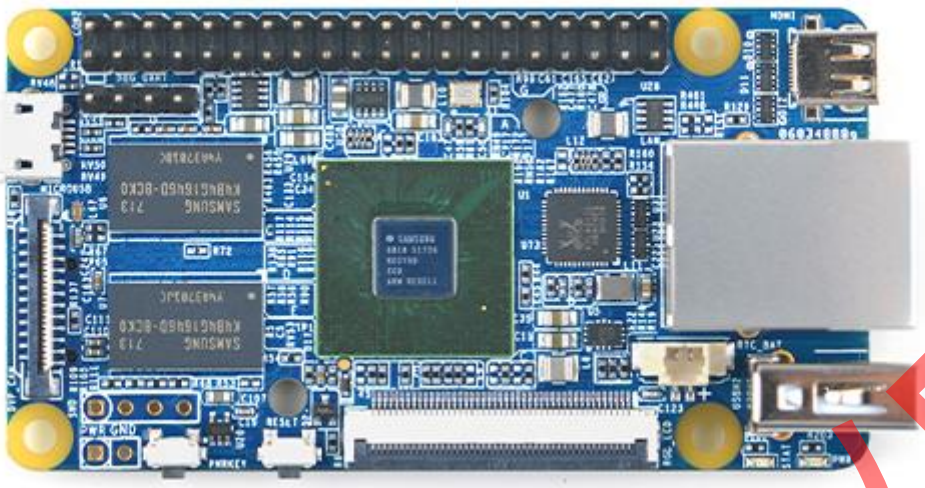
目　　次

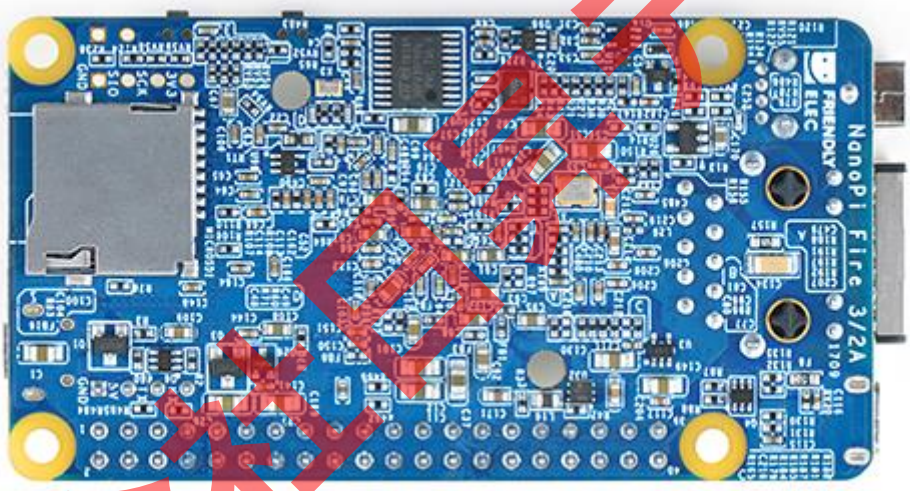
1 紹介

NanoPC-T3はSamsungのオクタコア・Cortex-A53・S5P6818 SoCを備えている。本ボードはCMOSカメラインターフェース、RGB LCD表示インターフェース、MicroHDMIビデオ出力ポートを備えており、USB2.0、MicroSD、ギガバイトイーサネットポートなどの一般的な標準インターフェースを備えている。またデバッグ シリアル ポート、40 Pinラズベリーパイ互換拡張ポートがある。RTCをサポートし、RTCのインターフェースピンを搭載。本ボードはFriendlyCore, Debian, AndroidのOSをサポートしている。Qt-5.9及びOpenGL ES1.1/2.0、ビデオハードウェアデコード等のグラフィックス処理ライブラリをサポートする。

2 主な仕様

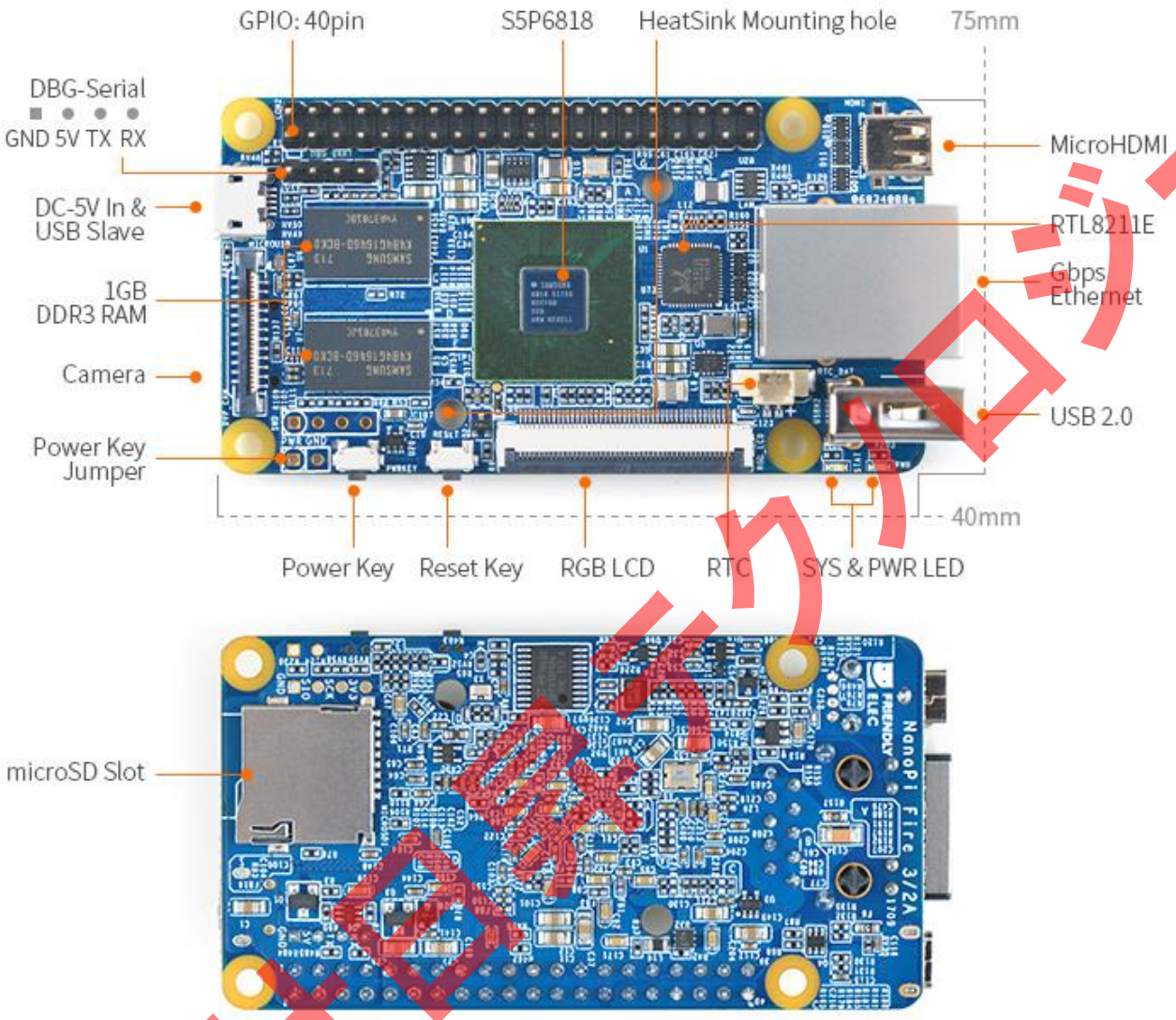
・SoC： Samsung S5P6818オクタコアのCortex-A53、400MHz～1.4GHz
・電力マネージメントユニット： ARM® Cortex®-M0を搭載して、ソフトウェアのパワーオフとウェイクアップをサポートする。
・システムメモリ：１GB 32ビットDDR3 RAM
・USB2.0ホスト：１×Aタイプ
・MicroUSB：１×MicroUSB2.0クライアント
・ストレージ：１×MicroSD カードソケット
・イーサネット： Gbit イーサネットポート (RTL8211E)
・LCDインターフェイス：0.5 mmピッチFPCコネクタ、フルカラー（RGB：8-8-8）
・HDMI：1.4A、１×MicroHDMI(Type-D)、1080P
・DVPカメラ ： １×DVP カメラインターフェース（0.5mmピッチFPCコネクタ）
・GPIO：2.54ミリピッチ、40ピンヘッダー、ラズベリーパイのGPIOとコンパチ、UART, SPI, I2C, PWM, IO等含む
・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：パワー、リセット
・LED: 電源LED×1、システム指示LED×1
・RTCバッテリー：RTCバッテリーシート
・ヒートシンク：２×ヒートシンク取付け穴付き
・給電：DC 5V/2A
・PCBサイズ：75mm×40mm
・OS/ソフトウェア: Android、Debian、UbuntuCore+Qt





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 40Pin GPIO ピン定義

ピン	名前	ピン	名前
1	SYS_3.3V	2	VDD_5V
3	I2C0_SDA	4	VDD_5V

5	I2C0_SCL	6	DGND
7	GPIOD8/PPM	8	UART3_TXD/GPIOD21
9	DGND	10	UART3_RXD/GPIOD17
11	UART4_TX/GPIOB29	12	GPIOD1/PWM0
13	GPIOB30	14	DGND
15	GPIOB31	16	GPIOC14/PWM2
17	SYS_3.3V	18	GPIOB27
19	SPI0_MOSI/GPIOC31	20	DGND
21	SPI0_MISO/GPIOD0	22	UART4_RX/GPIOB28
23	SPI0_CLK/GPIOC29	24	SPI0_CS/GPIOC30
25	DGND	26	GPIOB26
27	I2C1_SDA	28	I2C1_SCL
29	GPIOC8	30	DGND
31	GPIOC7	32	GPIOC28
33	GPIOC13/PWM1	34	DGND

35	SPI2_MISO/GPIOC11	36	SPI2_CS/GPIOC10
37	AliveGPIO3	38	SPI2_MOSI/GPIOC12
39	DGND	40	SPI2_CLK/GPIOC9

3.1.2 4 ピン DebugPort（UART0）インターフェイス　ピン定義

ピン	名前	ピン	名前
1	DGND	2	VDD_5V
3	UART_TXD0	4	UART_RXD0

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	GPIOC13/PWM1
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	Blue LSB to MSB	RGB のブルー信号
12-19	Green LSB to MSB	RGB のグリーン信号
21-28	Red LSB to MSB	RGB の赤信号

30	GPIOB25	汎用 GPIO、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信号、システム専用、ユーザーは設定できない。
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 Power Key

Power Key Jumper より外部キーに接続して、Power Key 機能を実現可能。

3.1.6 RTC

RTC バックアップ用バッテリーインタフェースを搭載、3V ボタン電池を直結可能。電流は約 3.35uA。外部電源接続してない時だけ RTC はボタン電池で給電される。

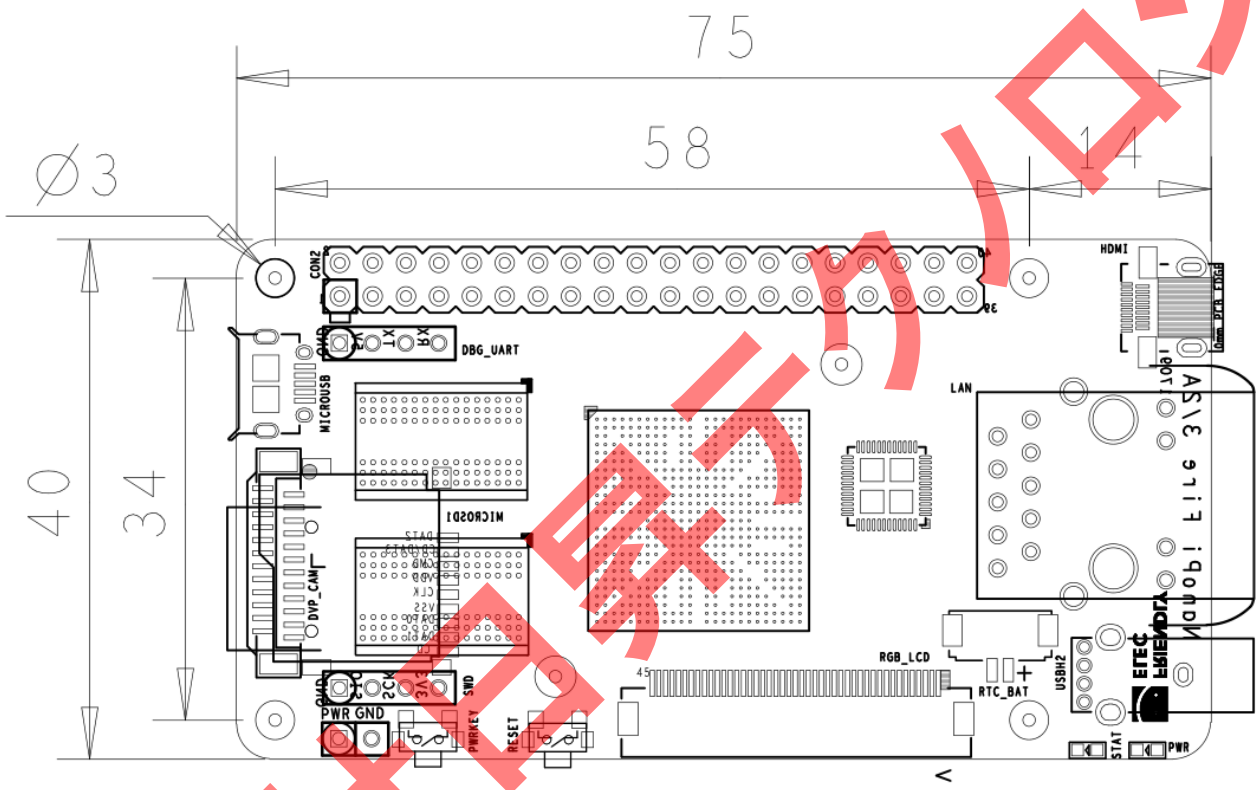
3.1.7 USB 2.0 HOST

1A 電流制限保護

3.1.8 その他

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



詳細については DXF ファイルをご参照してください。

4 クイックスタート

4.1 ハードウェアの準備

・NanoPi-Fire3ボード
・microSDカード/ TFカード：Class10以上の8GBのSDHCカードが必要
・microUSB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 SD カードから起動

次のファイルをHPからダウンロードしてください。
・8G以上SDHCカードを用意し、そのデータをバックアップする。

Image Files	
s5p6818-sd-friendlycore-xenial-4.4-armhf-YYYYMMDD.img.zip	32bit FriendlyCore システムイメージ（Qt 5.10.0 を内蔵), Ubuntu core を基づく
s5p6818-sd-friendlycore-xenial-4.4-arm64-YYYYMMDD.img.zip	64 bit FriendlyCore システムイメージ（Qt 5.10.0 を内蔵), Ubuntu core を基づく
s5p6818-sd-lubuntu-desktop-xenial-4.4-armhf-YYYYMMDD.img.zip	LUbuntu システムイメージ,X Window の GUI 付き
s5p6818-sd-friendlywrt-4.4-YYYYMMDD.img.zip	FriendlyWrt システムイメージ (OpenWrt よりカスタマイズ)
s5p6818-sd-android7-YYYYMMDD.img.zip	Android7 システムイメージ（4G LTE をサポート）
s5p6818-sd-android-lollipop-YYYYMMDD.img.zip	Android5.1 システムイメージ
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。

4.3 Linux Desktop 環境で起動カードの作成

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_s5p6818.git
cd sd-fuse_s5p6818
```

3)下記コマンドでLubuntu desktopを起動するSDカードを作成する

```
sudo ./fusing.sh /dev/sdx lubuntu
```

（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)イメージファイルだけ生成する場合：

```
sudo ./mkimage.sh lubuntu
```

4.4 TF カードパーティションの追加

Debian/Ubuntu システムでは自動で SD カードパーティションを追加する。初めて起動の際、パーティションとルートファイルシステムを自動的に生成する。
Android システムでは PC で下記コマンドを実行する：

```
sudo umount /dev/sdx?
sudo parted /dev/sdx unit % resizepart 4 100 resizepart 7 100 unit MB print
sudo resize2fs -f /dev/sdx7
```

4.5 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。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。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に設定できる。

4.6 パソコンで SD カード上の起動パラメータの変更

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。Kernel Command Lineのパラメータを更新したい場合は、「fw_setenv」ツールを使用することができる。

現在のカーネルの起動パラメータを確認する。

```
git clone https://github.com/friendlyarm/sd-fuse_s5p6818.git
cd sd-fuse_s5p6818/tools ./fw_printenv /dev/sdx | grep bootargs
```

例えば Android の SELinux を変更したい場合

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

上記コマンドで permissive モードに変更できる。[XXX]は元の bootargs に置き換える必要がある。

5 Android システム

5.1 Android7 で 4G モジュール EC20 を使用する

USB to miniPCIe 変換ボードを経由して EC20 モジュールを本ボードの USB Host I/F と繋ぐ。
他に設定する必要なく、Android を起動するだけで 4G でインターネットに接続する。

5.2 Android 起動時の Logo の変更

下記フォルダにあるlogo.bmpを交換する

```
/opt/FriendlyARM/smart4418/android/device/friendlyelec/nanopi3/boot/logo.bmp
/opt/FriendlyARM/smart4418/android/device/friendlyelec/nanopi2/boot/logo.bmp
```

ソースコードをコンパイルする。

6 システムのコンパイル

6.1 クロスコンパイラをインストールする

aarch64-linux-gcc 6.4 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 -b master --depth 1
cd prebuilts/gcc-x64
cat toolchain-6.4-aarch64.tar.gz* | sudo tar xz -C /
```

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/6.4-aarch64/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。

```
aarch64-linux-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolchain/6.4-aarch64/libexec/gcc/aarch64-cortexa53-linux-gnu/6.4.0/lto-wrapper
Target: aarch64-cortexa53-linux-gnu
Configured with: /work/toolchain/build/aarch64-cortexa53-linux-gnu/build/src/gcc/configure
--build=x86_64-build_pc-linux-gnu
--host=x86_64-build_pc-linux-gnu --target=aarch64-cortexa53-linux-gnu
--prefix=/opt/FriendlyARM/toolchain/6.4-aarch64
--with-sysroot=/opt/FriendlyARM/toolchain/6.4-aarch64/aarch64-cortexa53-linux-gnu/sysroot
--enable-languages=c,c++
--enable-fix-cortex-a53-835769 --enable-fix-cortex-a53-843419 --with-cpu=cortex-a53
...
Thread model: posix
gcc version 6.4.0 (ctng-1.23.0-150g-FA)
```

6.2 FriendlyCore/Lubuntu/EFlasher のカーネルのソースのコンパイル

ソースコードをダウンロードする。

```
git clone https://github.com/friendlyarm/linux.git -b nanopi2-v4.4.y --depth 1
cd linux
```

ブランチは[nanopi2-v4.4.y]であることに注意する。

Ubuntuカーネルをコンパイルする。

```
touch .scmversion
make ARCH=arm64 nanopi3_linux_defconfig
```

```
make ARCH=arm64
```

コンパイルに成功した後、arch/arm64/boot/Imageイメージファイルが生成する。arch/arm64/boot/dts/nexell/フォルダに新しいDTBファイル(s5p6818-nanopi3-rev*.dtb)も含まれる。SDカードのbootパーティションのファイルを切り替える。

6.3 Android7 kernel のコンパイル

Android 7.1.2ソースコードにカーネル済みのを含まれているが、カスタマイズ必要な場合、下記方法でコンパイルできる。

```
git clone https://github.com/friendlyarm/linux.git -b nanopi2-v4.4.y --depth 1
cd linux
touch .scmversion
make ARCH=arm64 nanopi3_nougat_defconfig
make ARCH=arm64
```

コンパイル後arch/arm64/boot/Imageが生成される。arch/arm64/boot/dts/nexell/フォルダに新しいDTBファイル(s5p6818-nanopi3-rev*.dtb)も含まれる。カーネルをデバッグするだけの場合、adbで素早く更新できる。

```
adb root; adb shell mkdir /storage/sdcard1/; adb shell mount -t ext4 /dev/block/mmcblk0p1 /storage/sdcard1/;
adb push arch/arm64/boot/Image arch/arm64/boot/dts/nexell/s5p6818-nanopi3-rev*.dtb /storage/sdcard1/
```

カーネル開発/デバッグ完了後、書き込み用のboot.imgを生成したい場合、カーネルImage及びDTBファイルをAndroid7ソースコードフォルダdevice/friendlyelec/nanopi3/bootにコピーして、Android7をコンパイルする。

6.4 Android7/FriendlyCore/Lubuntu/EFlasher の U-Boot ソースコードをコンパイル

U-Boot v2016.01 ソースコードをダウンロードしてコンパイルする。ブランチは nanopi2-v2016.01。

```
git clone https://github.com/friendlyarm/u-boot.git
cd u-boot
git checkout nanopi2-v2016.01
make s5p6818_nanopi3_config
make CROSS_COMPILE=aarch64-linux-
```

コンパイル後、fip-nonsecure.img が生成される。

For Android7: fip-nonsecure.img を Android7 ソースコードフォルダ device/friendlyelec/nanopi3/boot にコピーし、Android7 をコンパイルする。

6.5 Andriod7.1.2 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 16.04を推奨する。下記パッケージをインストール必要。

```
sudo apt-get install bison g++-multilib git gperf libxml2-utils make python-networkx zip
sudo apt-get install flex curl libncurses5-dev libssl-dev zlib1g-dev gawk minicom
sudo apt-get install openjdk-8-jdk
```

```
sudo apt-get install exfat-fuse exfat-utils device-tree-compiler liblz4-tool
```

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。http://download.friendlyarm.com/NanoPiFire3

```
tar xvf /path/to/netdisk/sources/s5pxx18-android-7.git-YYYYMMDD.tar
cd s5pxx18-android-7
./sync.sh
```

Git からコピーする。

```
git clone https://gitlab.com/friendlyelec/s5pxx18-android-7.git -b master
```

Android7 のソースコードのサイズ(8.2GB)が大きいので、暫く待つ必要。

6.5.3 システムをコンパイルする

```
cd s5pxx18-android-7
source build/envsetup.sh
lunch aosp_nanopi3-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
bl1-mmcboot.bin	raw	boot firmware
fip-loader.img	raw	boot firmware
fip-secure.img	raw	boot firmware
fip-nonsecure.img	raw	uboot-v2016.01
env.conf	-	Uboot 環境変数、Android カーネルコマンドパラメータを含む

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

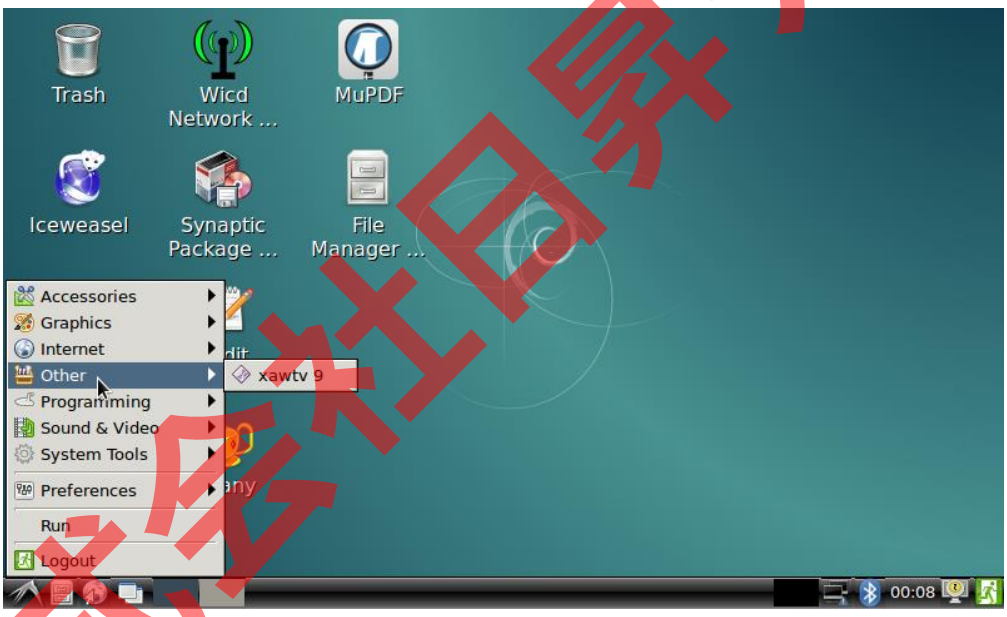
7 拡張モジュール

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"->"xawtv"をクリックする。

３、"Welcome to xawtv!"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、Android5.1 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。
NanoPi-Fire3 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 NanoPi-Fire3 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、NanoPi-Fire3 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 NanoPi-Fire3 で作動していることを確認する。NanoPi-Fire3 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。

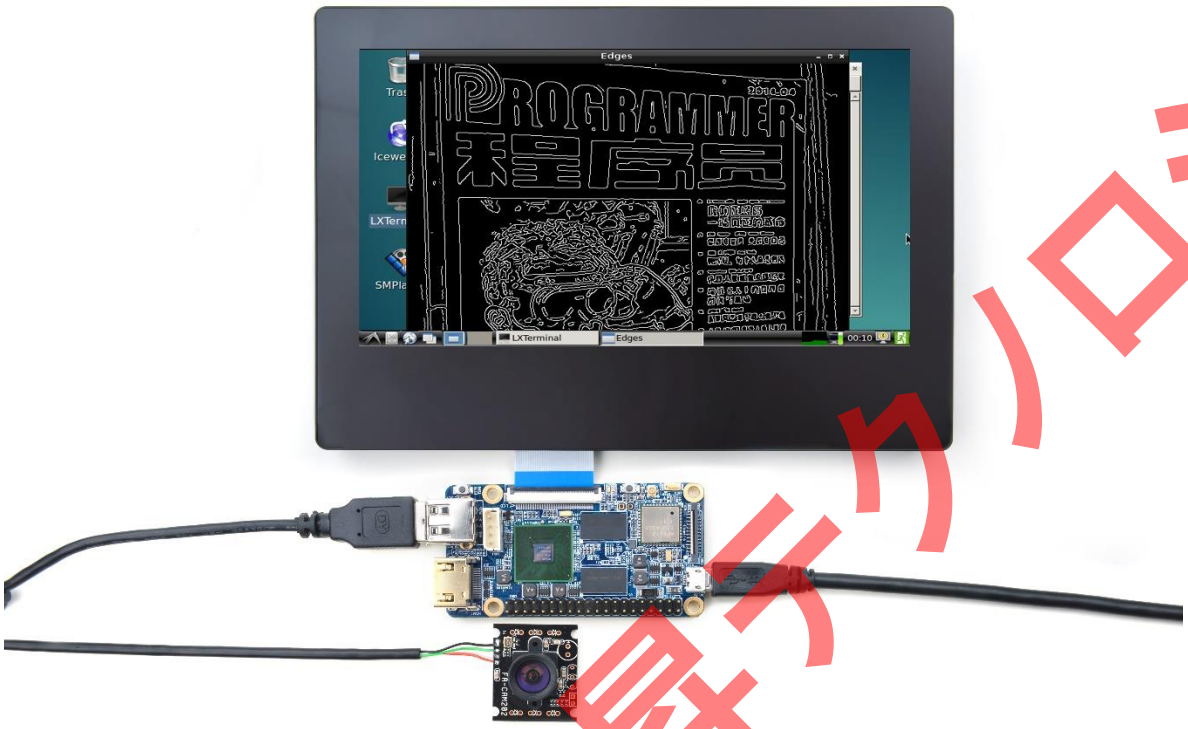
```
cd /home/fa/Documents/opencv-demo
```

```
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. NanoPi-Fire3 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。