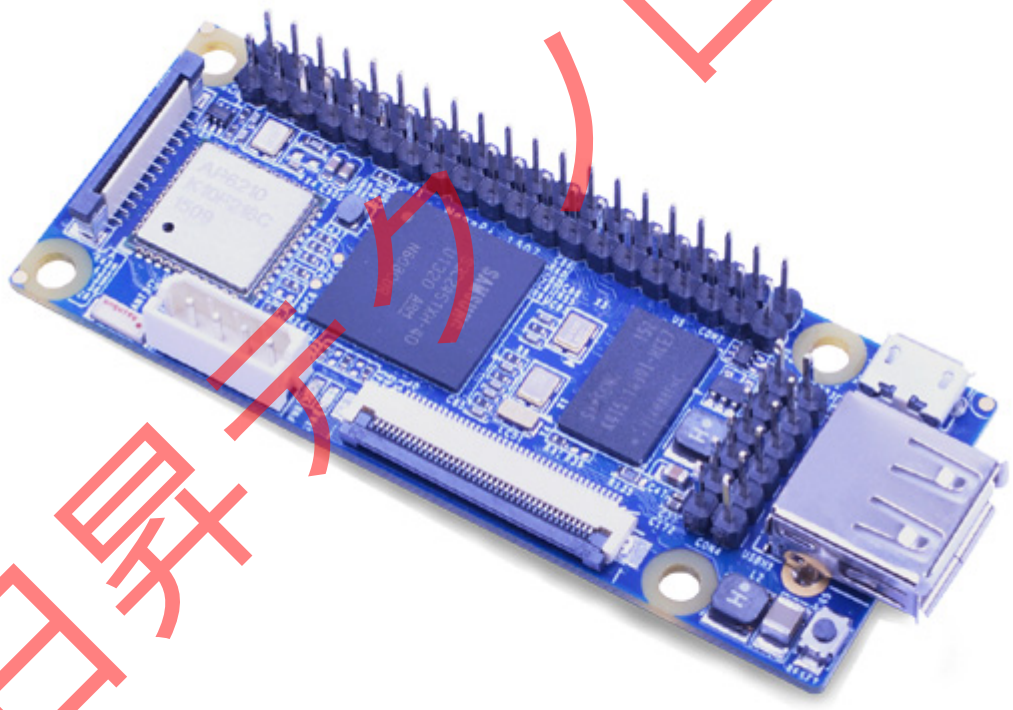
ARM9 コア S3C2451 ボード NanoPi 簡易マニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

作成日 2015/11/11



・修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2015/10/22
2	Ver1.1	Windows 環境で実行用 SD カード作成手順を追加	2015/11/11
3	Ver1.2	4.6 Wi - Fi ネットに接続の節の内容を修正	2015/11/20

目次

1 紹介

NanoPiは組み込みLinux利用者のために設計された低消費電力のARMマスタコントロールボードである。そのサイズはRaspberry Pi（RPi）の半分のサイズで、RPiのGPIOインターフェースと互換性がある。NanoPiはワイヤレス無線WiFiとBluetooth4.0モジュールを統合し、パラレルカメラインタフェースとカラーLCDインターフェースを搭載、TFカードからLinux / Debianシステムを実行することができる。IOT、ワイヤレススマートカー、ロボット、画像認識、HMIなどのアプリケーションの開発に最適である。

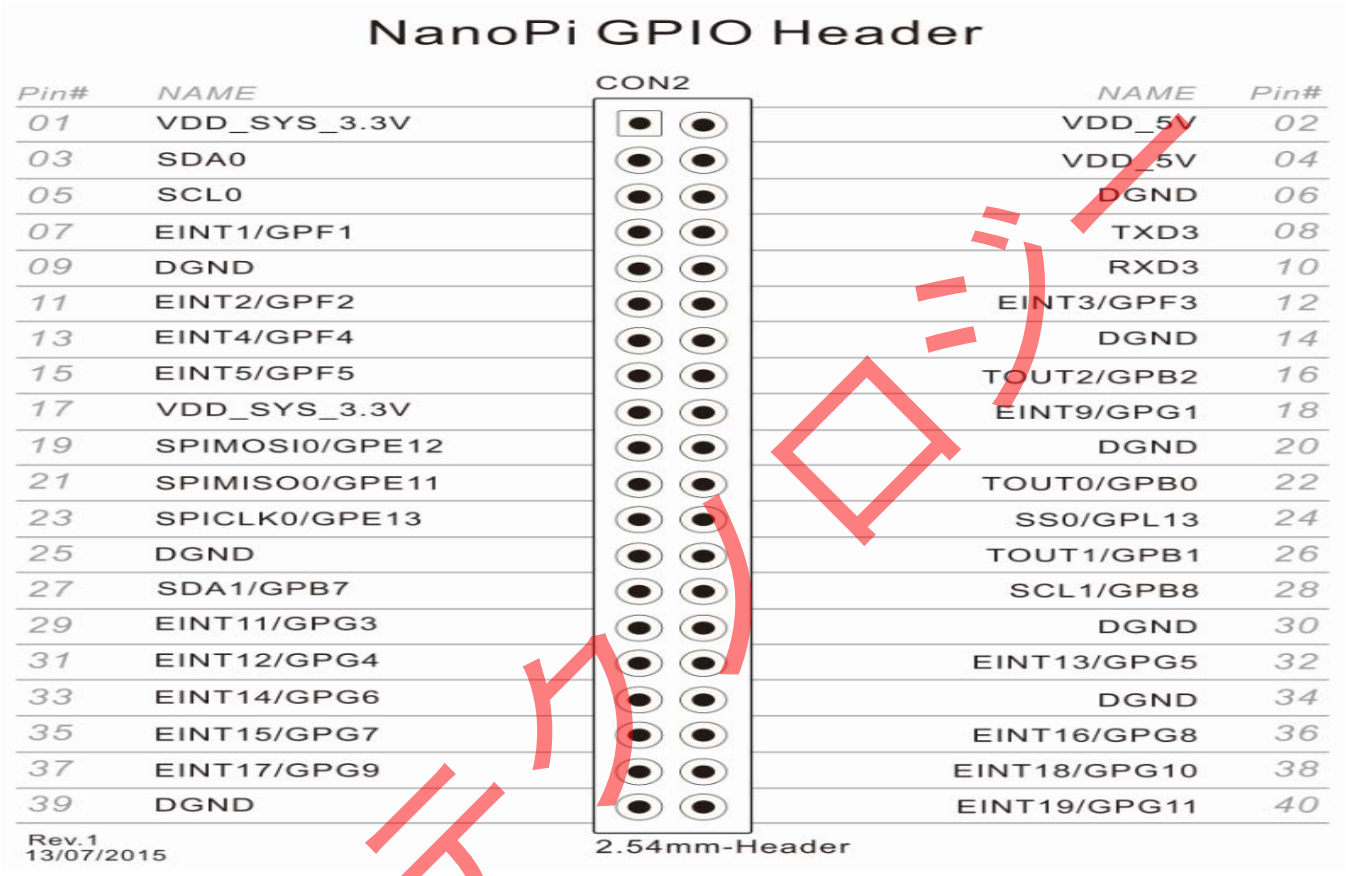
2 主な仕様

CPU：Samsung S3C2451、動作周波数　400Mhz
RAM：64M DDR2
統合SDIO WiFi Bluetoothモジュール
USBタイプA x1
デバッグ用シリアルポートx1
microSD Slot x1
microUSB x1：給電とデータの伝送をサポート、シリアル或いはイーサネットとして使用可
LCD I/F：0．5 mmピッチSMT FPCコネクタ、フルカラーLCDをサポート（RGB：8-8-8）
DVPカメラ I/F： 0.5mmピッチ省スペースタイプFPCソケット、ITU-R BT 601/6568ビット、I2CおよびIOを含む。
GPIO1：2.54mmピッチ、40ピン、RasberryRPiのGPIOと互換性がある 。UART、SPI、I2C、IOなどを含む。
GPIO2：2.54mmピッチ、12ピン、I2S、I2C、UARTなどピンを含む
PCBサイズ：75 x 30 mm
電源：DC 5V
OS/Software：u-boot、Linux-4.1、Debian8 jessie、Rabbit linux

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 GPIO1 ピン定義



3.1.2 GPIO2 ピン定義

Pin#	Name	Pin#	Name
1	VDD_5V	2	VDD_SYS_3.3V
3	TXD2	4	RXD2
5	SDA0	6	SCL0

7	IISSDOO	8	IISSDIO
9	IISSCLKO	10	IISLRCKO
11	IISCDCLKO	12	DGND

3.1.3 Debug Port CON1 (UART0)

Pin#	Name
1	DGND
2	VDD_5V
3	TXD0
4	RXD0

3.1.4 DVP Camera IF ピン定義

Pin#	Name
1, 2	VDD_SYS_3.3V
7,9,13,15,24	DGND
3	SCL0
4	SDA0

5	GPH13
6	GPJ12
8	XCLK
10	NC
11	VSYNC
12	HREF
14	PCLK
16-23	Data bit7-0

3.1.5 RGB LCD IF ピン定義

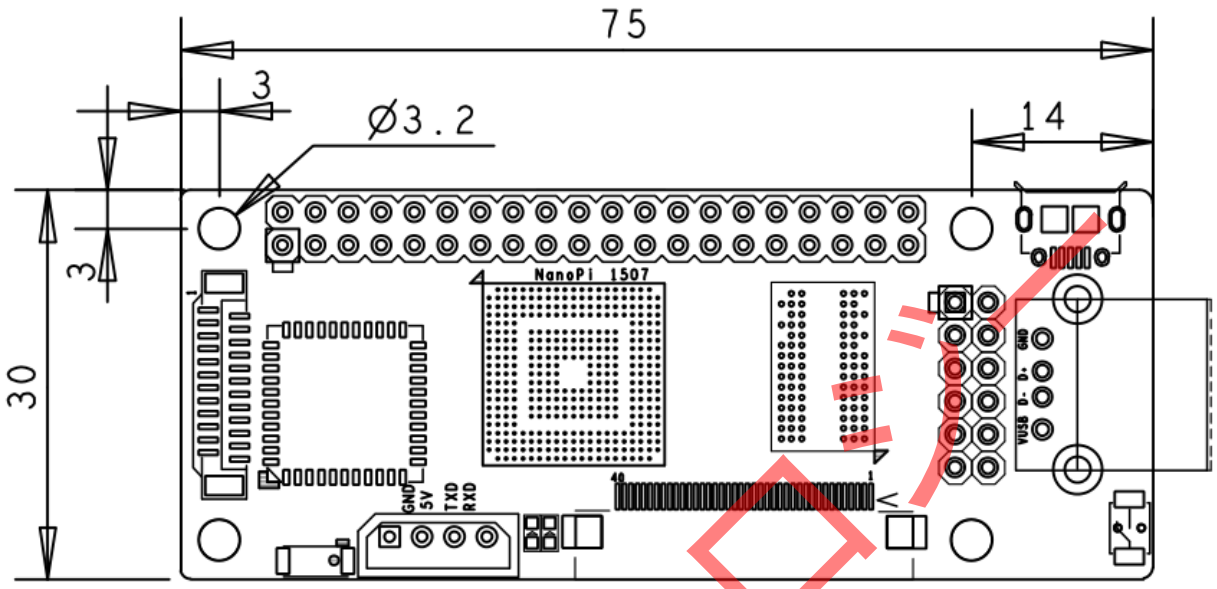
Pin#	Name
1, 2	VDD_5V
11, 20, 29	DGND
3-10	Blue LSB to MSB
12-19	Green LSB to MSB
21-28	Red LSB to MSB

30	GPG12
31	GPG2
32	XnRSTOUT Form CPU
33	VDEN
34	VSYNC
35	HSYNC
36	LCDCLK
37, 38, 39, 40	XM, XP, YM, YP

説明

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：NanoPi-1507-Schematic.pdf　をチェックしてください。

3.2 PCB サイズ



更に詳しいサイズについては　NanoPi-1507-Dimesions．zip　をダウンロードしてください。

4 クイックスタート

4.1 ハードウェアの準備

・NanoPiボード
・microSDカード/ TFカード：最小システムは64Mが必要
・microUSBケーブル
・Linuxを実行するコンピュータ１台、オンラインネットワーキング、Debian jessie 64ビットシステムの使用を推奨

4.2 実行システムを持つ microSD カードを作成する

1） microSDをUbuntuのパソコンに挿入　以下のコマンドであなたの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） ファームウェアをダウンロードし、microSDカードを作成する。

```
git clone https://github.com/friendlyarm/sd-fuse_nanopi.git
czd sd-fuse_nanopi
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）

もしgithubから取得できない場合、下記URLからダウンロードできる。：
http://wiki.friendlyarm.com/NanoPi/download/sd-fuse_nanopi.tgz 。
もしfusing.shスクリプトでカード作成時に問題があれば、WindowsシステムでNanoPi-fuser-win32.zipファームウェアパッケージをダウンロードして、WindowsシステムでSDカードを作成することができる。

4.3 実行システム

作成したmicroSDカードをNanoPiに挿入し、NanoPi microUSB線でコンピュータに繋げる。NanoPiは自動的に電源がオンになり、ボード上の青色LED点滅が確認できると、システムはすでに正常に実行されていることになる。デフォルトでインストールされているのはDebianシステムである。

4.4 microUSB を通じて NanoPi を登録する

microUSBケーブルでNanoPiをコンピュータに接続し、コンピュータ上でコマンド：dmesgを入力すると、以下の出力情報により接続成功の表示が見られる。

```
「12601.100339 ] usb 2-1.7：Product：FriendlyARM Gadget v2.4
「12601.100343 ] usb 2-1.7：Manufacturer：Linux 4.1.2-FriendlyARM with s3c-hsudc
「12601.103192」cdc_acm 2-1.7：2．0：This device cannotドcalls on、そのown．It isないa modem。
「12601.103368」cdc_acm 2-1.7：2． 0：ttyACM0：USB ACM device
「12601.105300」cdc_ether 2-1.7：2．2 usb0：register ' cdc_ether ' at usb-0000：00：1d.0-1.7、
CDC Ethernetデバイス、46：a 1：e7：6d：5c：32
```

パソコンにコマンド：ifconfigを入力すると、usb0のネットワーク機器が１つ増えたことが表示される。このとき、ｓｓｈを通して192.168.100.1というアドレスにアクセスし、NanoPiを登録する：

```
ssh  root@192.168.100.1
```

パスワードの入力を要求された時、デフォルトのパスワードfaを入力するとログインできる。

4.5 Wi - Fi を通じて NanoPi を登録する

コンピュータ上でnanopi-wifiapという無線ホット（パスワードは123456789）検索し、接続する。接続成功後、ｓｓｈを通じて192.168.100.1というアドレスまでNanoPiを登録できる：

```
ssh   root@192.168.8.1
```

パスワードの入力を要求された時、デフォルトのパスワードfaを入力するとログインできる。

4.6 Wi - Fi ネットに接続

NanoPiの出荷時、WiFiは無線ホットモードに設定されています。WiFiネットに接続するにはまずホットモードを終了します。以下のコマンドで終了する。

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

ボードを再起動する。

そして、ｓｓｈを使ってNanoPiに接続し、まず以下のコマンドを入力しWiFiのネットワークインターフェースを検索する。先頭の文字列がwlanのものがWiFiである。：

```
ifconfig -a
```

デフォルトの場合はwlan0で、/etc/network/interfaces.d/ディレクトリの下にネットワークインターフェイスと同名のプロファイルを新規作成する。wlan0を例として、コマンド：viで下のファイルを新規作成する。：

```
vi /etc/network/interfaces.d/wlan0
```

wlan0ファイルの内容は次の通りになる。：

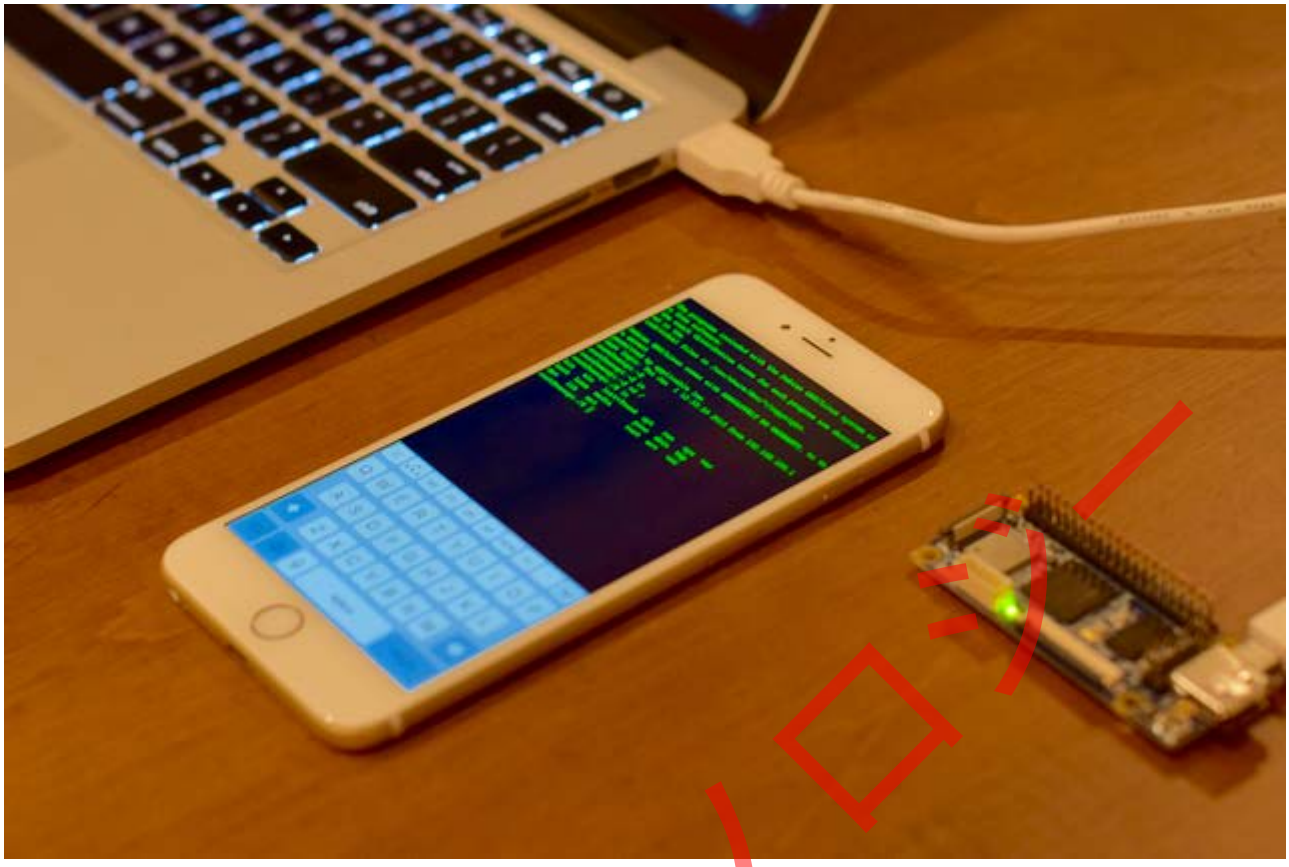
```
auto lo
iface lo inet loopback
auto wlan0
iface wlan0 inet dhcp
wpa-driver wext
wpa-ssid YourWiFiESSID
wpa-ap-scan 1
wpa-proto RSN
wpa-pairwise CCMP
wpa-group CCMP
wpa-key-mgmt WPA-PSK
wpa-psk YourWiFiPassword
```

上記の中で、YourWiFiESSIDとYourWiFiPasswordをあなたが接続したい無線AP名とパスワードに置き換えてください。上記ファイルを保存してクローズする。下記コマンドでWiFi接続する。

```
/etc/init.d/networking restart
```

ボードを再起動すると、自動的にWiFiと接続する。

注意しなければならないのは、TFカードが多数のボード上で実行されれば、WiFiのネットワークインターフェースがwlan1、wlan2などにリネームされる可能性がある。このファイル内容をクリアにし、再起動することで、デフォルト値：/etc/udev/rules.d/70-persistent-net.rulesに復元させることができる。



WiFi接続後、携帯側のクライアントでNanoPiを登録する。

4.7 Wi-Fi 無線ホットスポットの配置

NanoPiは出荷時に、既にWiFiを無線ホットスポットモードに設定した状態である。デフォルトのホットスポット名はnanopi-wifiapで、パスワードは123456789。

WiFi仕事モードは以下のコマンドで調査できる：

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

出力する数字が2ならば、当面は無線ホットスポットモードと示す。

もし今のWiFiが無線ホットスポットモードではない場合、以下のコマンドでオンにすることができる：

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

無線ホットスポット名とパスワードは/etc/hostapd/hostapd.conf編集ファイルから修正できる。

4.8 Bluetooth を使ってファイルを転送する

Debian jessieシステムでBuluetoothに関する必要なパッケージ（ブルートゥース、bluez、obexftpなど）を既にプレインストールした。

次に、コマンドラインでNanoPiと携帯電話の間にBuluetoothを通してファイルをアップロード/ダウンロードすることをデモンストレーションする。

この例では、私が使用するテスト機器はMX4携帯電話1台で、テストの前にまず携帯電話のBuluetooth機能を有効にし、検出可能なモデルに設定します。その後NanoPiに以下のコマンドを入力し周辺のBuluetoothデバイスを検索する。：

```
hcitool scan
```

```
Scanning…
8C:BE:BE:C5:2C:C7 MX4
```

上に挙げた結果は私のテスト用のMX4携帯電話を見つけることに成功した。覚えておく住所は8C：ｂｅ：ｂｅ：C５－２C：C7。次に、Sdptoolコマンドで、それがどのようなプロトコルをサポートしているかどうかを確認することができる。

```
sdptool browse 8C:BE:BE:C5:2C:C7
```

我々がテストしたいのはBuluetoothのファイル転送機能であるため、デバイスがOBEX File Transfer　プロトコルをサポートするか否かにのみ関心がある。このプロトコルのサポートがあれば、次の操作が可能である。：

```
Service Name: OBEX File Transfer
Service RecHandle: 0x1000c
Service Class ID List:
"OBEX File Transfer" (0x1106)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 11
"OBEX" (0x0008)
Language Base Attr List:
code_ISO639: 0x454e
encoding: 0x6a
base_offset: 0x100
Profile Descriptor List:
"OBEX File Transfer" (0x1106)
Version: 0x0100
```

携帯がOBEX File Transferプロトコルをサポートしていることがわかる。obexftpコマンドで操作でき、以下のコマンドで、携帯電話のディレクトリ下のファイルをリストになる。：

```
obexftp -b 8C:BE:BE:C5:2C:C7 -c / -l
```

以下のコマンドで携帯電話の/ Adnroid / djaof.dllファイルをNanoPiにダウンロードする。：

```
obexftp -b 8C:BE:BE:C5:2C:C7 -c /Android -g djaof.dll
```

以下のコマンドでローカルファイルhello.txtを携帯の/ Androidディレクトリにアップロードする。：

```
obexftp -b 8C:BE:BE:C5:2C:C7 -c /Android -p hello.txt
```

4.9 iBeaconトランスミッターを設定する

iBeacon技術は主に室内の距離を測るためにに使われる。以下のコマンドを入力すると、NanoPiをiBeaconトランスミッターに設定することができる。：

```
hciconfig hci0 up
hciconfig hci0 leadv 3
hciconfig hci0 noscan
hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 63 6F 3F 8F 64 91 4B EE 95 F7 D8 CC 64 A8 63 B5 00 00 00 00 C8
```

完成後NanoPiのBuluetoothは途切れなくラジオニュースを送る。その時、必要なことはAndroidやiPhone携帯にLocate Beaconというアプリケーションをインストールするだけである。そのアプリを通じて、携帯電話からNanoPiまでの距離を測ることができる。もし室内での位置情報を取得する必要があるならば、複数のiBeaconトランスミッターを設定する必要がある。ウェブで関連した詳細情報を得ることができる。

4.10 BLE を通じてモバイル通信をする

この機能はNanoPiでBLEサービスプログラムを搭載し、サービスプログラムを実行することをで、NanoPiがBLEの週辺機器になる。携帯電話はBluetoothを介してNanoPiに接続、データ通信できる。

BLEサービスプログラムはOpen sourceで、以下のルートで取得できる。

```
git clone https://github.com/friendlyarm/ble-peripheral-service-demo.git
```

スクリプトのコンパイルを実行して、ファイル名がnanopi_ble_serveという実行可能なファイルをコンパイルする。

```
./build.sh
```

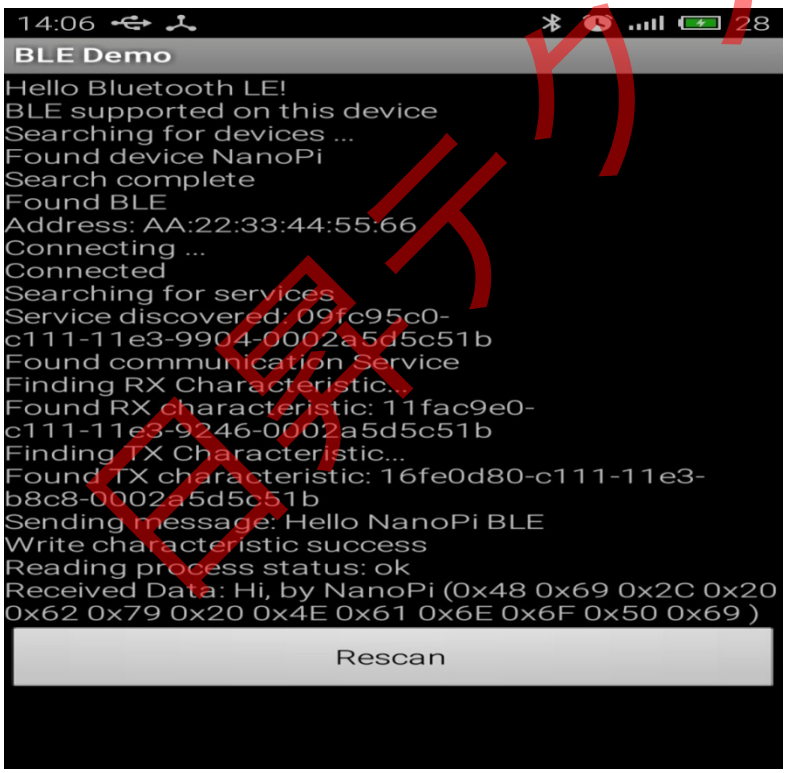
このスクリプトはあなたのクロスコンパイラが以下のパスにインストールすると仮定する。

/opt/FriendlyARM/toolschain/4.5.1/bin/arm-linux-gcc

もしこのパスではないときは、ご自身でスクリプトを変更してください。コンパイルが完成した後は、nanopi_ble_serverをNanoPiにコピーし、以下のコマンドで起動してください：

```
hciconfig hci0 down
service bluetooth stop
chmod 755 nanopi_ble_server
./nanopi_ble_server
```

最後に、あなたのAndroid携帯にBLE Scannerというアプリをインストールするか、或いはiPhoneにLightblueというアプリをインストールする。Buluetoothで検索し、NanoPiに接続する。我々はまたオープンソースのAndroid Demoを提供しており、android / BLETestディレクトリにある。これはeclipseのプロジェクトで、このAndroid Demoは携帯電話でどのようにBLEを通じてNanoPiに接続することを実演し、データの送受信をする。携帯端末Demoの実行インターフェースは次のようになる。



NanoPi端のサービスプログラム出力は次のようになる。

```
root@nanopi:~# ./nanopi_ble_server
2015/01/01 13:16:51 dev: hci0 up
2015/01/01 13:16:51 dev: hci0 down
2015/01/01 13:16:51 dev: hci0 opened
State: PoweredOn
2015/01/01 13:16:51 Generating attribute table:
2015/01/01 13:16:51 handle      type    props   secure  pvt     value
2015/01/01 13:16:51 0x0001      0x2800  0x02    0x00    *gatt.Service   [ 1B C5 D5 A]
2015/01/01 13:16:51 0x0002      0x2803  0x02    0x00    *gatt.Characteristic    [ 02]
2015/01/01 13:16:51 0x0003      0x11fac9e0c11111e392460002a5d5c51b      0x02    0x00]
2015/01/01 13:16:51 0x0004      0x2803  0x0C    0x00    *gatt.Characteristic    [ 0C]
2015/01/01 13:16:51 0x0005      0x16fe0d80c11111e3b8c80002a5d5c51b      0x0C    0x00]
Connect:  21:ae:bd:6c:ab:c1
2015/01/01 13:17:13 ignore l2cap signal:[ 06 00 05 00 13 02 02 00 00 00 ]
Recv data from device: 0 48 65 6c 6c 6f 20 4e 61 6e 6f 50 69 20 42 4c 45
Send data to device: Hi, by NanoPi
```

4.11 Debian のパッケージソフトをインストールする

私たちが提供するのは標準的なDebian jessieシステムである。ユーザーは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。ボードがは初めて実行される場合、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
```

もしパッケージソフトのダウンロードスピードが理想的な速度でなければ、/etc/apt/sources.listを編集し、より速いソースサーバーに交換するとよい。http://www.debian.org/mirror/listに完全なミラーサーバーリストがにある。“armel”がついてるのを選択してください。

5 システムのコンパイル方法

5.1 クロスコンパイラをインストールする

まず、コンパイラをダウンロードし解凍する。

```
git clone https://github.com/friendlyarm/prebuilts.git
tar xvzf prebuilts/gcc/arm-linux-gcc-4.4.3.tar.gz -C /
```

そしてコンパイラのパスをPATHに追加し、viでvi～/.bashrcを編集し、末尾に以下の内容を追加します。

```
export PATH=/opt/FriendlyARM/toolschain/4.4.3/bin/:$PATH
```

～/ . bashrcスクリプトを実行し、shellウィンドウで設定を直ぐに有効にする。"."の後ろにスペースがあることに注意する。

```
. ~/.bashrc
```

コンパイラは32ビットであるため、もし使用するパソコンにインストールされているLinuxシステムが６４ビットであれば、いくつかのパッケージソフトをインストールしてから、このコンパイラを実行する。例えばDebian8 jessie desktop 64bitシステムの下で、以下のコマンドを入力し、インストールする。：

```
dpkg -- add-architecture i386
```

```
apt-get update
apt-get install build-essential gcc-multilib rpm libstdc++6：i386 libgcc1：i386 zlib1g：i386 libncurses5：i386
```

5.2 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nanopiであることに注意する。

```
git clone https://github.com/friendlyarm/uboot_nanopi.git
cd uboot_nanopi
git checkout nanopi
make nanopi_config
make
```

コンパイルに成功した後、u-boot.binを取得する。直ぐにu-bootをテストしたい場合は、スクリプトfusing.shを使い、新しいu-boot をSDカードに書き込む。もしあなたのSDカード対応の設定名が/ dev / sddであれば、rootで以下のコマンドを実行する。

```
su
./fusing.sh /dev/sdd
```

注意：以上の操作によって、SDカードのデータが破壊する可能性がある。事前にバックアップする。

5.3 Linux kernel のコンパイル

5.3.1 カーネルのコンパイル

カーネルのソースコードをダウンロードしコンパイルする。

```
git clone https://github.com/friendlyarm/linux-4.x.y.git
cd linux-4.x.y
git checkout nanopi-v4.1.y
make nanopi_defconfig
touch .scmversion
make
```

NanoPiカーネルが所属するブランチはnanopi-v4.1.y、コンパイルが始まる前にまずブランチの切替をする。コンパイルが成功した後、新しく生成されるカーネル書込みファイルはarch/arm/boot/zImageとなる。

5.3.2 カーネルモジュールのコンパイル

現在のカーネルの構成はカーネルモジュールをコンパイラし生成する。例えばipv6、netfilter、通常カーネルをコンパイルする時、カーネルモジュール（. ko）は既にコンパイルした状態である。もし新しいカーネルモジュールやカーネルの配置に変化があれば、カーネルモジュールをコンパイル、インストールし、kernel-modules.tgzをパッケージ化する。そしてRootfs下のbasefs / kernel-modules.tgzを替えます。Rootのユーザーとして、以下のコマンドを実行し、ko を/ tmp / nanopi-modulesにインストールする。

```
make INSTALL_MOD_PATH=/tmp/nanopi-modules modules_install
```

次はカーネルモジュールにstripをする。そして圧縮バッグを作成する。

```
cd /tmp/nanopi-modules/lib/
find . -name ¥*.ko | xargs arm-linux-strip --strip-unneeded
tar czvf kernel-modules.tgz modules/
```

新しくコンパイルをするカーネルモジュールテストしたい場合は、既に作成したNanoPiを起動できるSDカ

ードのrootfs下の/ libディレクトリに圧縮バッグを解凍するか、あるいはmodulesをコピーする。
SDカードのrootfsは既に/media/fa/NANOPIにmountをしています。Rootで以下のコマンドを実行する。：

```
rm -rf /media/fa/NANOPI/lib/modules/
tar xzvf kernel-modules.tgz -C /media/fa/NANOPI/lib/
```

5.4 Debian システムの作成

5.4.1 Debian ファイルシステム

プレインストールされたDebianシステムが使用するのは公式にコンパイルされたパッケージソフトである。このファイルシステムは以下で見つけられる。：

```
git clone https://github.com/friendlyarm/sd-fuse_nanopi.git
cd sd-fuse_nanopi/prebuilt/
ls -l rootfs.tgz
```

Debianシステムを作り直したい場合、rootfs.tgzを解凍し、編集する。完成後、tarで再圧縮してよい。たとえば、ダウンロードしたdebバッグ一つをファイルシステムにプレインストールする。以下のコマンドを使用する。：

```
tar xzf rootfs.tgz
dpkg -i --force-all --root=./rootfs /tmp/qtembedded-4.8.5_armel.deb
tar czf rootfs.tgz rootfs
```

5.5 自分がコンパイルしたファイルで SD カードを作成する

5.5.1 SD カードシステムを作り直す

SDカードが書き込むサドルバッグをダウンロードする。相応するmasterブランチに切り替える。：

```
git clone https://github.com/friendlyarm/sd-fuse_nanopi.git
cd sd-fuse_nanopi
git checkout master
```

sd-fuse_nanopiにprebuiltがあり、SDシステムの実行に必要なバイナリファイルを保存します。：
Bootloader: u-boot.bin
カーネルコマンドラインパラメータ：sdenv.raw
Linuxカーネル：zImage
ファイルシステムの圧縮ファイル：rootfs.tgz
これらを、先にコンパイラ生成されたファイルと交換する。コンパイルのシステムのテストを実行する。ファイル置換後、sd-fuse_nanopiディレクトリで以下のコマンド
を実行しSDカードを作成する。

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

（注：/dev/sdxは実際のSDカードデバイスファイル名に替えてください）。
書込みが完成した後、SDカードをNanoPiに入れて実行してみましょう。

5.5.2 U-Boot 環境変数の更新

NanoPi起動後、以下のコマンドで、U-Bootコマンドの環境変数を見ることができる。：

```
fw_printenv bootargs
```

環境変数を変更する必要がある場合、例えばlcdの追加はfw_setenvコマンドを使う。呼び出しのサンプルは以下の通り：

```
fw_setenv bootargs root=/dev/mmcblk0p2 rootfstype=ext4 init=/sbin/init console=ttySAC0,115200 lcd=S70
```

SDカードを作成したい時、すぐ新しい環境変数を持つことができる。SDカードをパソコンに挿入し、コンピュータ上で以下のコマンドを実行し、環境変数を探し出す。Prebuiltディレクトリ下のsdenv.rawを交換し、再度SDカードの書き込みをする。

```
cd sd-fuse_nanopi
./readenv.sh /dev/sdd
cp sdenv.raw prebuilt/
```

5.5.3 SD カードの RAW ファイルについて

CPU　S3C2451 iROMのため

SDカードの後部からBootloaderを読み取る。更に、普通のSDカードとSDHCカードの位置は異なる。異なるブランドまたは異なる容量のSDカードの大きさはまた違うため、異なるカードのRAWファイルを作成できない。

大きさが全く同じSDカードをいくつか持っている場合、Linuxのddのようなツールを使い、すでに作成したSDカードのすべてのデータを読み取り、RAWファイルに保存し、このファイルを同じサイズのSDカードに書き込むことができる。

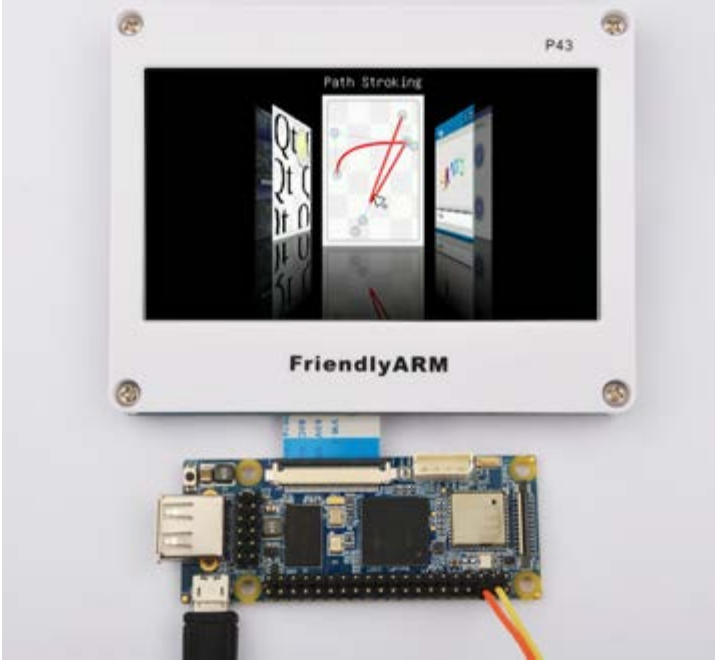
また、今のSDカードの容量は通常少し大きい。8 GBのSDカードでは、RAWファイルの書き込みに長時間かかり、また、現在のrootfsは実質～23 MBしかない。このため、直接スクリプトを使用してSDカードを作る方が速い。

6 拡張接続

6.1 カメラモジュールを接続する

現時点まだ対応しておりません。今後追加する予定です。

6.2 LCD を接続し Qt4 を実行する



NanoPiは以下のタイプのLCDと接続できる。- H43、S70、A70、W50、W35、P43、P35、TD35。
LCDの作動を開始させるには更に2つのステップを行う必要がある。:
1)miniUSBをコンピュータに接続すると、その給電はLCDの要求に満たない可能性がある。5V 2Aの電源の外付けが必要であり、GPIOのVDD_5VとDGNDピンに接続して良い。
2) U-bootの環境変数を変更し、lcdモデルパラメータを追加する必要がある。例えばS70、ボード上で実行後、以下のコマンドを入力し修正する。:
まず既存の環境変数を確認する。:

```
fw_printenv bootargs
```

出力は次の通り：

```
bootargs=root=/dev/mmcblk0p2 rootfstype=ext4 init=/sbin/init console=ttySAC0,115200
```

bootargs＝後ろのパラメーターの内容にlcd＝S70を加え、更にfw_setenvコマンドで新たに設定する。

```
fw_setenv bootargs root＝/ dev / mmcblk0p2 rootfstype＝ext4 init＝/ sbin / init console＝ttySAC0、115200 lcd＝S70
```

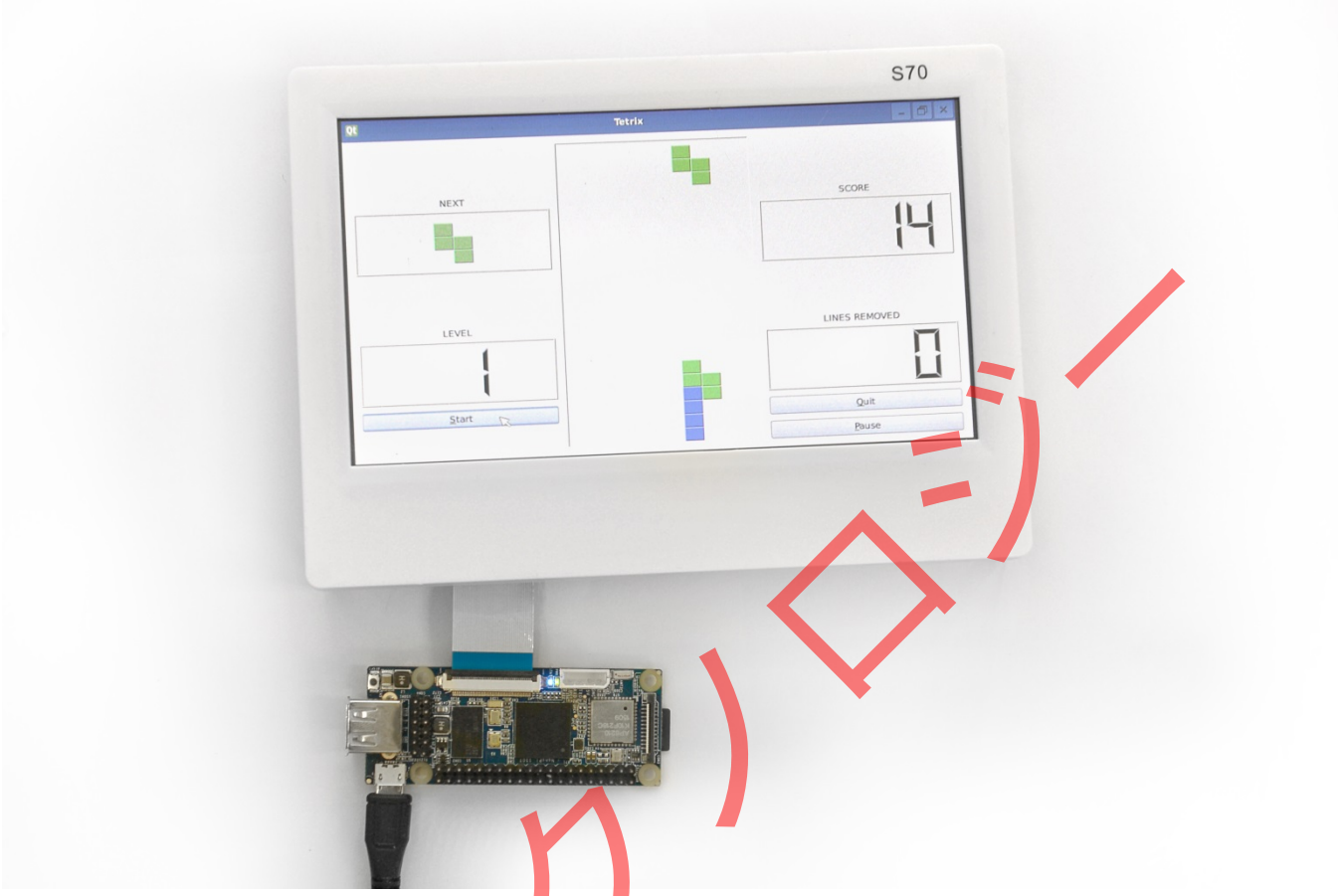
再起動後、LCDはDebianのコマンドラインが表示され、インターフェースを登録し、USBキーボード接続し登録の操作をして良い。
Qtとtslibパッケージのダウンロードⅰ

```
apt-get update
apt-get install qtembedded
apt-get install friendlyarm-tslib
```

qtemebddedはQt4関連のライブラリで、friendlyarm-tslibはタッチパネルの校正に使う。
Qtサンプルプログラムを実行する。：

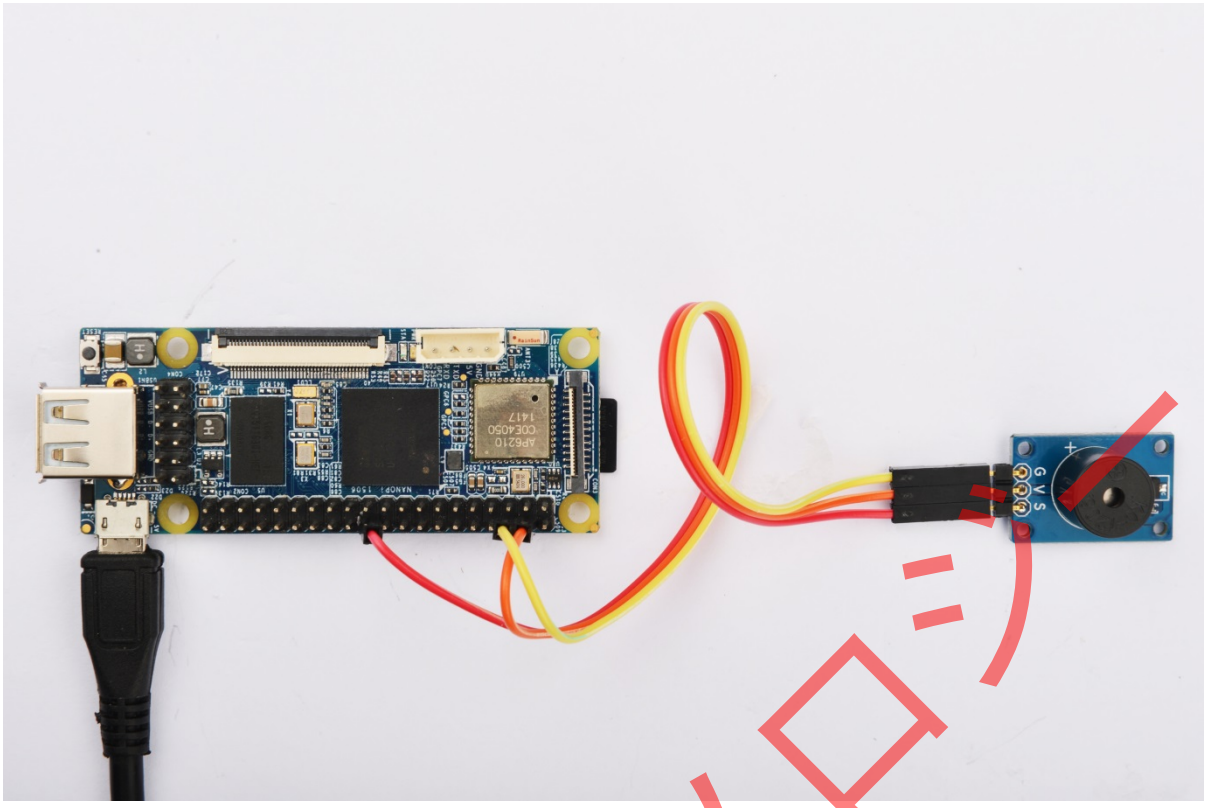
```
. /usr/bin/setqt4env
/usr/local/Trolltech/QtEmbedded-4.8.5-arm/examples/tetrix -qws
```

スクリプトsetqt4envはQtとtslib関連の環境変数の設定に使用する。初めてこのスクリプトを実行する場合は校正の画面が出てくる。5ポイントをクリックして校正を完成する。
tetrixはQt4が作成した小さいゲームテトリスで、その実行効果は次の通り。



6.3 Matrix 入門 DIY キットの接続と使用

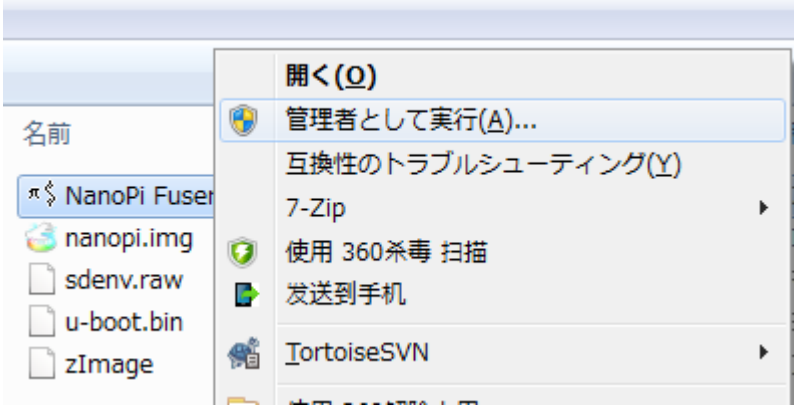
ハードウェア接続：

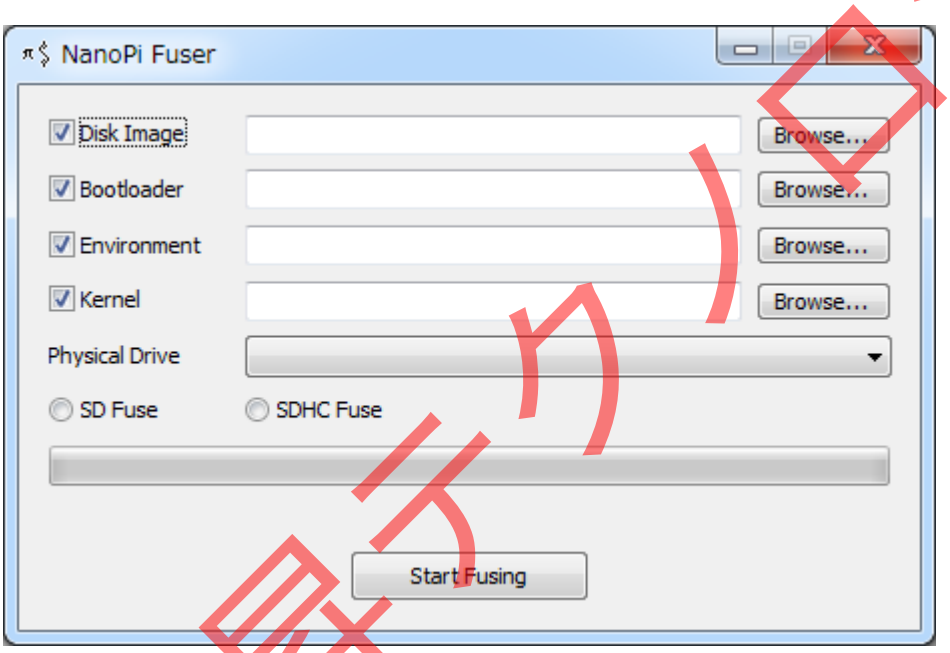


詳細内容は下記 URL をご参照ください：
http://wiki.friendlyarm.com/wiki/index.php/Matrix_-_Buzzer#NanoPi

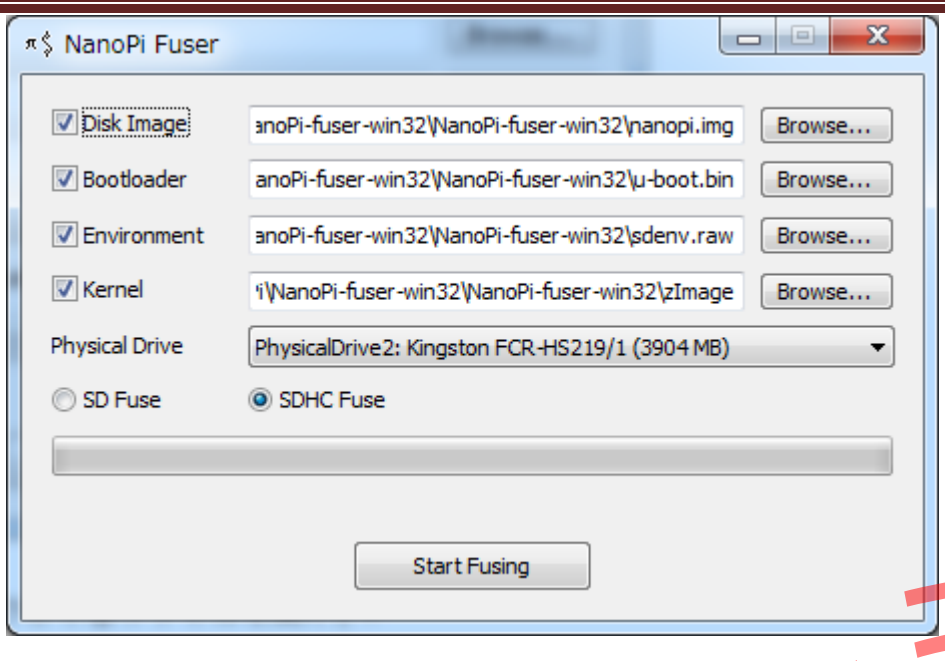
7 Windows 環境で実行用 SD カードの作成

1、NanoPi-fuser-win32.zip をダウンロードして解凍する。
2、NanoPi Fuser.exe を右クリックして「管理者として実行（A）…　」をクリックする。

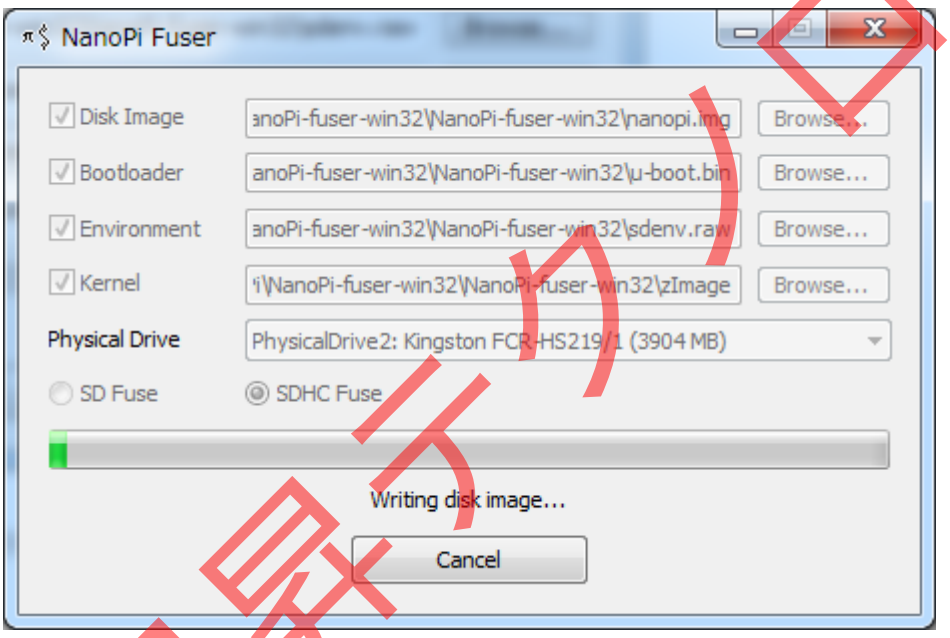




3、「Browse…」をクリックして各ファイルを開く。



４、「Start Fusing」をクリックする。



以上。