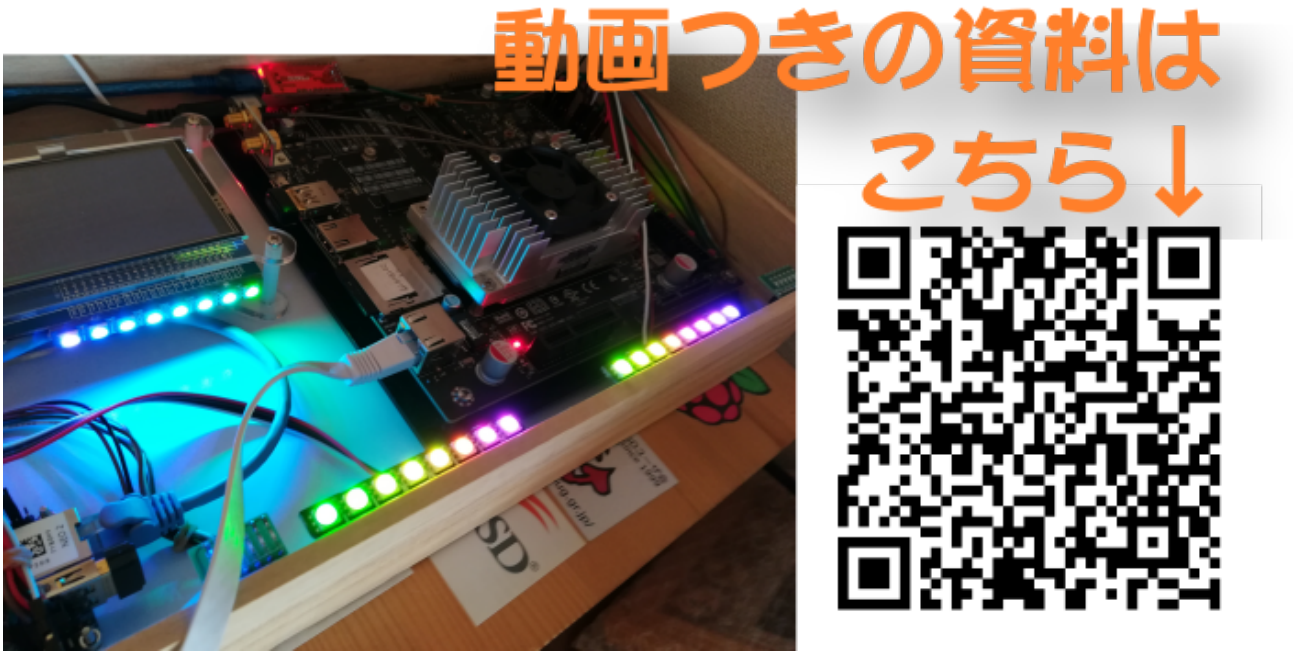
NanpPiとNetBSDとピカピカと

NanoPi Contest 2021

by むとうたけし(/610t/610t)

English version



はじめに

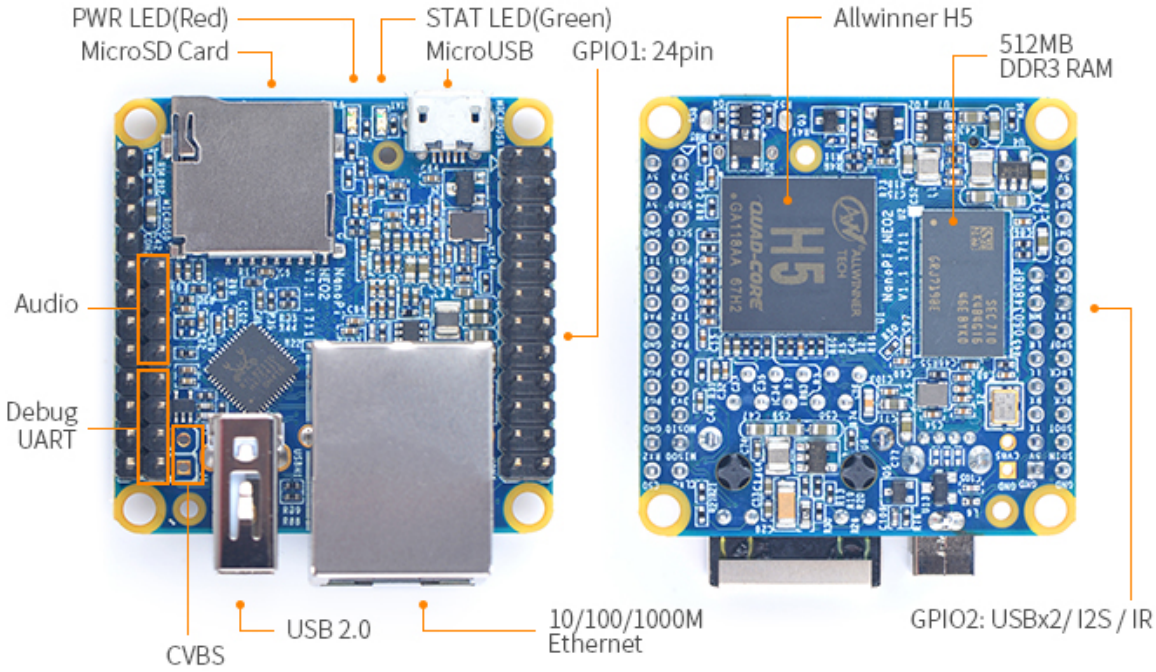
近年、ゲーミングパソコンなどを中心に、ピカピカ光るステキなパソコンが色々と出ています。
我らがマイコンでも、やっぱりピカピカしたいですよね?

NanoPiなら、それができます!!

では、実際にやってみましょう!!

NanoPi NEO2

Friendly ARM社のNanoPi NEO2は、小さいボディーに色々な機能を詰め込んだワンボードマイコンです。

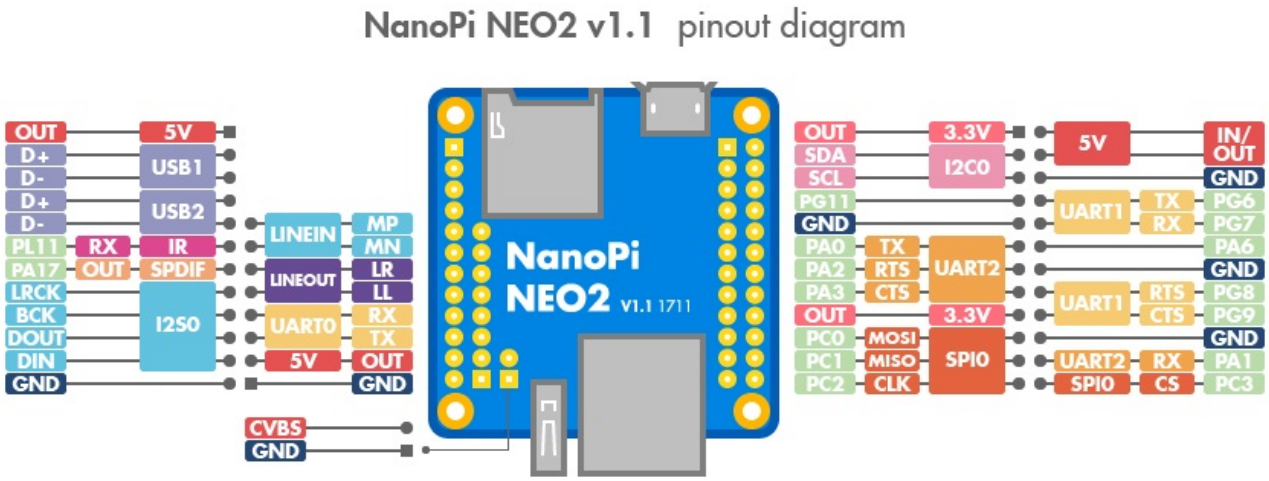


from FriendlyARM Wiki

NanoPi NEO2には、以下のような特徴があります。赤字は、特にオススメの機能です。

- SoC: **Allwinner's 64-bit H5 quad-core** (Cortex-A53)
- GPU: Mail450
- RAM: 512M DDR3
- I/O: microSD, **Ethernet 10/100/1000M**, USB, Serial, GPIO, Audio
- Size: **40 x 40mm, 13g**
- OS: UbuntuCore 16.04, Debian, **NetBSD**

NanoPi NEO2のピン配置は、以下の図のようになります。
たくさんのGPIOやシリアルなどがあることがわかります。
左側の24ピンのコネクタは、5V, 3.3V, GNDなどの配列がRaspberry Piの40ピンの上24ピンと同じになっています。



from FriendlyARM Wiki

OSとしては、公式が提供しているUbuntuCore以外にも、Armbianをはじめとして様々なOSで動作可能です。
普通はLinux系のOSで動かすことがほとんどだと思いますが、"*Of course it runs NetBSD.*"の精神で知られているNetBSDでももちろん動きます。今回は、OSとして、NetBSDを使うことにします。

NetBSDでGPIOを使う

NetBSDでは、GPIOデバイスは以下のように認識されます。GPIO用のデバイスファイルは、`/dev/gpio*` です。

dmesg

```
$ grep gpio /var/run/dmesg.boot
sunxigpio0 at simplebus1: PIO
gpio0 at sunxigpio0: 94 pins
sunxigpio0: interrupting on GIC irq 43
sunxigpio1 at simplebus1: PIO
gpio1 at sunxigpio1: 12 pins
sunxigpio1: interrupting on GIC irq 77
gpioleds0 at simplebus0: nanopi:green:pwr nanopi:blue:status
```

NetBSDでGPIOを使うためには、初期設定を行う必要があります。
はじめに、 `/etc/rc.conf` に `gpio=YES` を追加します。
更に、以下のように `/etc/gpio.conf` で、GPIOのモードを設定します(`gpio.conf` (5))。

この設定は、セキュリティー上の理由で、起動時にしか変更できないことに注意してください。
`OPTION insecure` を指定したカーネルを作成することで、この制限は無くなります。

`/etc/gpio.conf`

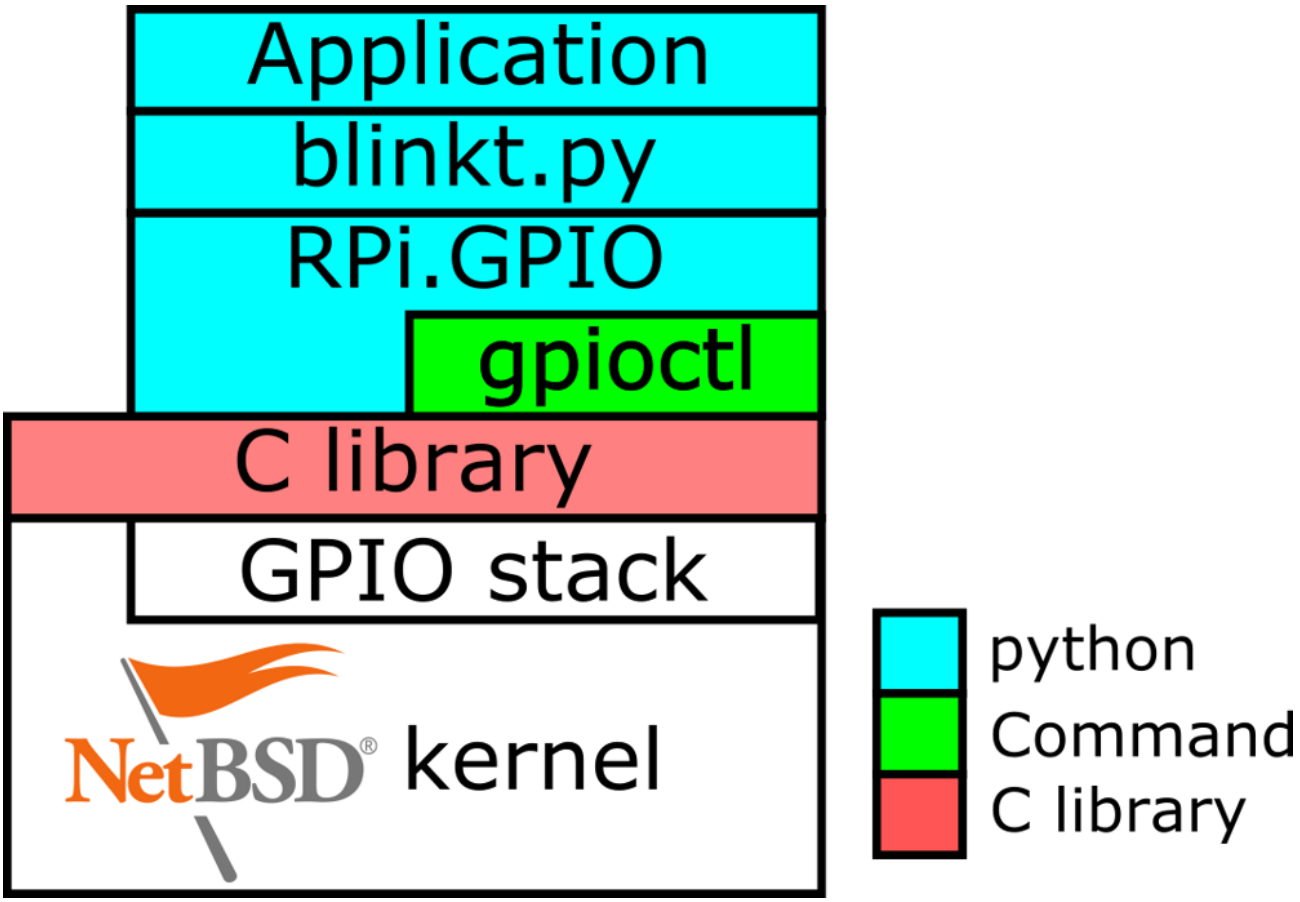
```
## GPIO
gpio0 0 set out PA0
gpio0 1 set out PA1
gpio0 2 set out PA2
gpio0 3 set out PA3
gpio0 6 set out PA6
gpio0 11 set out PA11
gpio0 12 set out PA12
gpio0 86 set out PG6
gpio0 87 set out PG7
## for my Blinkt!
gpio0 88 set out dat
gpio0 89 set out clk
# gpio0 88 set out PG8
# gpio0 89 set out PG9

## NanoPi OLED hat
# gpio0 0 set in b1
# gpio0 2 set in b2
# gpio0 3 set in b3
```

`gpioleds0` は。内蔵の緑LED(`nanopi:green:pwr`)と青色LED(`nanopi:blue:status`)になります。後述(Case 1)のように、システム関連変数を変更する `sysctl` (8)コマンドで、LEDの点灯/消灯を制御できます。

その他のGPIOに関しては、 `gpioctl` (8)コマンドから利用することができます。

C言語のライブラリも用意されており、 `open` と `ioctl` を使って操作することができます(`gpio` (4))。

NetBSDで、Blinkt!を使う場合のシステム階層は、以下のようになります。



Case 1:青色LEDによるロードアベレージの表示

では、はじめに、内蔵のLEDを使って、ロードアベレージを表示してみましょう。

`sysctl` でLEDの状態を書き込んであげることで、簡単に点滅することができます。

loadavg.sh

```
#!/bin/sh
while true
do
  INTERVAL=`uptime|awk '{print $(NF-2)}'|sed -e 's/,//'|sed -e 's/^/e(-/' -e
's/$/)/'|bc -l`
  echo ${INTERVAL}
  sysctl -qw hw.led.nanopi_blue_status=1
  sleep ${INTERVAL}
  sysctl -qw hw.led.nanopi_blue_status=0
  sleep ${INTERVAL}
done
```

動作の様子は、以下の動画のようになります。NanoPi NEO2は画面の右下にあります。

https://youtu.be/C2xbhGp5R9c

Case 2:Knight2000に挑戦!!

GPIOにつながったLEDがたくさんある場合、最初に作るのはKnight2000のアレです。

今回接続したピンの配列は、以下のとおりです。

pin assign

Pin#	Name	gpio0 dev#	LEDの場所
3	PA12	12	左から1番目
5	PA11	11	左から2番目
12	PA6	6	左から3番目
15	PA3	3	左から4番目
22	PA1	1	左から5番目

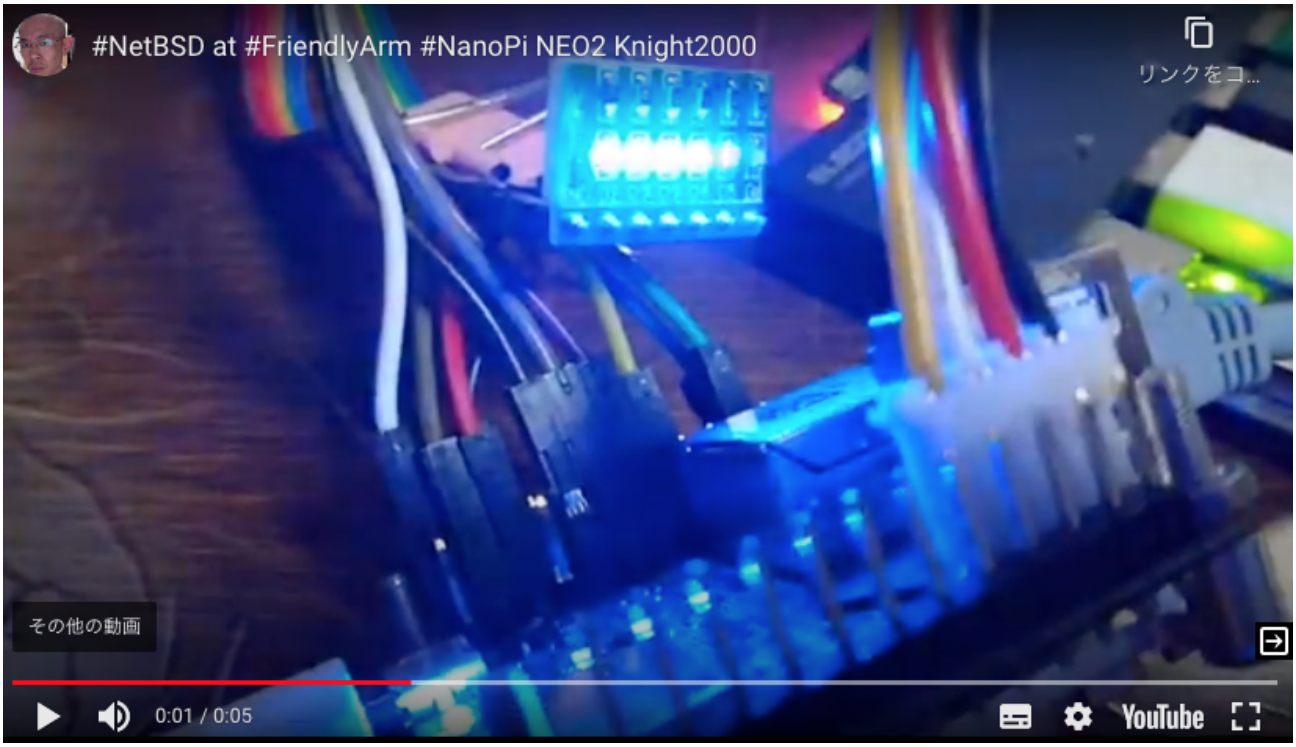
以下のようなシェルスクリプトで、5つの青色LEDが一定方向に点滅を繰り返します。

knight_rider.sh

```
#!/bin/sh
PINS="12 11 6 3 1"
while true
do
  for l in $PINS
  do
    sudo gpioctl gpio0 $l 1
  done
```

```
    for l in $PINS
    do
      sudo gpioctl gpio0 $l 0
    done
  done
```

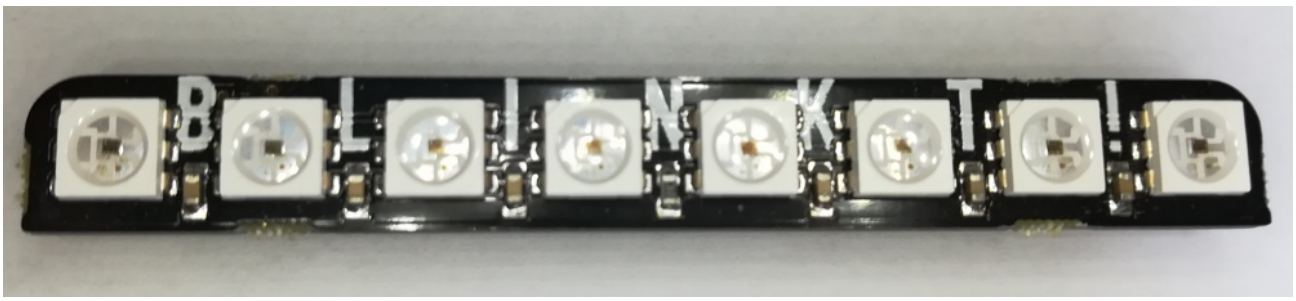
動作の様子は、以下の動画のようになります。



https://youtu.be/YL3kHWc577A

フルカラー8連LED: Blinkt!

Blinkt!は、Raspberry Piの40ピンGPIOに刺してつかえる、8連のフルカラーLEDです。2つのGPIOだけを使って、8連のフルカラーLEDの操作が可能です。



Blinkt!:ラズベリーパイ40pin GPIO用8連フルカラーLED(APA102)

販売: switch-science , pimoroni

Pythonライブラリソース at github https://github.com/pimoroni/blinkt

ピン接続図: https://pinout.xyz/pinout/blinkt

Blinkt!で使われる2つのGPIOは、DATA(16pin)とCLOCK(18pin)で、それぞれLEDに送るデータと、データを送るタイミングを指定するクロックになります。
クロック(青)の立ち下がりのデータ(オレンジ)の値を送ることになっています。

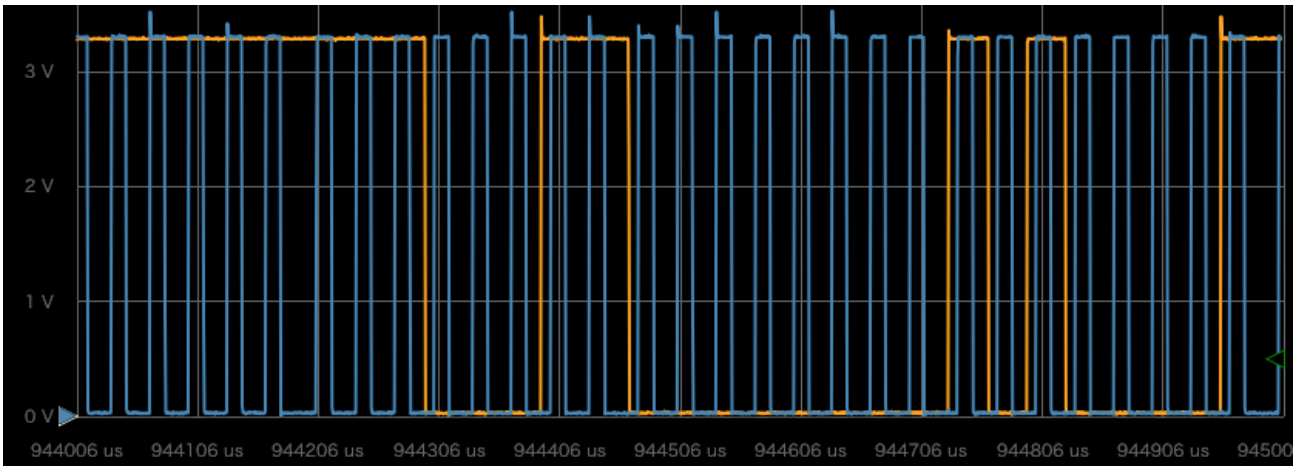
以下の順で1つのLEDあたり8bit x 4のデータを送信します。

blightness(有効5bit/8bit)

b

g

r



Case 3:Blinkt!でピカピカ

{Free,Net,Open}BSD用のRPi.GPIOは、githubで https://github.com/610t/BSD.RPi.GPIO として公開しています。
更に、Blynkt!用Pythonライブラリソース at githubが必要です。

ここには、 `gpioctl` を使って書かれた `GPIO.py` と、C言語で書かれたpythonモジュール版の `GPIO.c` があります。
前者は毎回ユーザランドコマンドを実行するため、動作速度がとても遅くなります。
ということで、後者を使ってピカピカします。

NanoPi NEO2で使うGPIOポートは、任意の二つを選ぶことができます。
今回は、以下のようにピンを割り当てました。この割り当てを使うと、Blynkt!を24ピンGPIO端子に直挿しして動くようになります。

pin assign

	Pin#	Name	gpio0 dev#
DAT	16	PG8	88
CLK	18	PG9	89

実際に動作させるためには、 `blinkt.py` の中で `DAT` と `CLK` としてGPIOポート番号をハードコーディングで指定しているため、このポート番号をBlinkt!を接続したポート番号に変更することが必要です。

今回の場合、具体的には以下のようなパッチを使います。このパッチでは、NanoPi NEO2のGPIOポートのPG8を `DAT` に、PG9を `CLK` に割り当てています。

blinkt.py.diff

```
% diff -u library/blinkt.py{.org,}
--- library/blinkt.py.org       2021-09-04 19:02:16.000000000 +0900
+++ library/blinkt.py   2021-09-04 19:02:30.000000000 +0900
@@ -7,8 +7,8 @@

 __version__ = '0.1.2'

-DAT = 23
-CLK = 24
+DAT = 88
+CLK = 89
 NUM_PIXELS = 8
 BRIGHTNESS = 7
```

動作の様子は、以下の動画のようになります。

https://youtu.be/c4HuA0kCc2I

さらに、複数のBlynkt!を他のGPIOポートに接続して、 `blinkt.py` を各GPIOポートに変更したものに変えることで、複数のBlynkt!を動かすこともできます。

おわりに

いかがでしたか?

これで、あなたのNanoPiが、

ナウでヤングなピカピカパソコン

として利用できるようになりました。

本当は、I2Cを使って、もっと色々なデバイスで遊びたかったのですが、残念ながら現在のNanoPi NEO2用のNetBSDはI2Cを認識しないため、GPIOで遊ぶだけになってしまいました。
また機会があれば、I2Cでピカピカしたいと思います。

参考文献

むとうの以前の発表など

- NetBSD de Blinkt
- Blinkt!でみる*BSDのGPIO: または、翼よあれがaarch64の灯だ

Nano Pi公式

- NanoPi NEO2
- NanoHat OLED
- NanoHat Hub

#NanoPi_NEO2 #NetBSD #GPIO #Blinkt!