Table of Contents

Introduction

「xshige's beta note」のblogをgitbookにしたもので、目次の上の検索欄でblogを検索できる。

2022/1/5
初版

dfu-util最新版インストール

dfu-util最新版インストール

概要

dfu-util最新版インストールについて記する。

インストール

以下の手順でインストールする：

```
# 古いものをアンインストールする
sudo apt remove dfu-util

# 最新ソールのダウンロード&インストール
cd ~/tools
git clone git://git.code.sf.net/p/dfu-util/dfu-util
cd dfu-util

sudo apt install libusb-1.0-0-dev

./autogen.sh
./configure
make
sudo make install

# 以上でインストールの完了
```

udev登録

dfu-utilで書き込むボードは、udev登録する必要がある。 それには、ボードのIDを調べるする必要がある。 ボードをDFUモードに以下のような手順で確認できる：

```
$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root
＜省略＞
Bus 001 Device 113: ID 1d50:607f OpenMoko, Inc. Spark Core
＜省略＞
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root
```

以上で、SparkCoreは「ID 1d50:607f」であることが分かる。

ParkCoreの登録例：

```
sudo gedit /etc/udev/rules.d/50-particle.rules
# 以下のurlの内容のものを作成する
https://gist.github.com/monkbroc/b283bb4da8c10228a61e
```

/etc/udev/rules.d/50-particle.rules

```
# UDEV Rules for Particle boards
#
# This will allow reflashing with DFU-util without using su
#
# The latest version of this file may be found at:
#   https://gist.github.com/monkbroc/b283bb4da8c10228a61e
#
# This file must be placed at:
#
# /etc/udev/rules.d/50-particle.rules    (preferred locatio
#
# To install, type this command in a terminal:
#   sudo cp 50-particle.rules /etc/udev/rules.d/50-particle
#
# After this file is installed, physically unplug and recon
# Particle device.
#
# Core
SUBSYSTEMS=="usb", ATTRS{idVendor}=="1d50", ATTRS{idProduct
# Photon/P1/Electron
SUBSYSTEMS=="usb", ATTRS{idVendor}=="2b04", ATTRS{idProduct
#
# If you share your linux system with other users, or just
# idea of write permission for everybody, you can replace M
# OWNER:="yourusername" to create the device owned by you,
# GROUP:="somegroupname" and mange access using standard un
#
#
# If using USB Serial you get a new device each time (Ubunt
# eg: /dev/ttyACM0, ttyACM1, ttyACM2, ttyACM3, ttyACM4, etc
#    apt-get remove --purge modemmanager     (reboot may be
#
# CREDITS:
#
# Edited by Julien Vanier
#
# This file is derived from the Teensy UDEV rules
# http://www.pjrc.com/teensy/49-teensy.rules
#
```

以下でudevを有効にする：

```
sudo udevadm control --reload-rules

sudo usermod -a -G dialout $USER
sudo usermod -a -G plugdev $USER
```

動作確認

ボードをUSB接続して、DFUモードにして以下を実行する。

```
$ dfu-util -l
dfu-util 0.11

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMo
Copyright 2010-2021 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANT
Please report bugs to http://sourceforge.net/p/dfu-util/tic

Found DFU: [1d50:607f] ver=0200, devnum=7, cfg=1, intf=0, p
Found DFU: [1d50:607f] ver=0200, devnum=7, cfg=1, intf=0, p
```

以上のように「Found DFU」が表示されれば、正常動作していることになる。 動作しない場合、再起動してから確認してみる。

参考資料

dfu-util - Device Firmware Upgrade Utilities

以上

2022/1/5
初版

gcc-arm-none-eabi最新版インストール

gcc-arm-none-eabi最新版インストール

概要

gcc-arm-none-eabiのバージョンが古くてビルドできない例があったので、
gcc-arm-none-eabi最新版インストールについて記する。

インストール

以下の手順でインストールする：

```
# 古いUbuntuパッケージをアンインストールする
sudo apt remove binutils-arm-none-eabi gcc-arm-none-eabi li

cd ~/tools
# 新しいものをダウンロードする
wget https://developer.arm.com/-/media/Files/downloads/gnu-
# 解凍する
tar xf gcc-arm-none-eabi-10.3-2021.10-x86_64-linux.tar.bz2


# パスを設定する
export PATH=$PATH:$HOME/tools/gcc-arm-none-eabi-10.3-2021.1
# .bashrcに上のexportと同じものを登録する

# パスが有効になっていることの確認
which arm-none-eabi-gcc
/home/USER/tools/gcc-arm-none-eabi-10.3-2021.10/bin//arm-nc
```

以上

2022/1/6
初版

st-flash最新版インストール

st-flash最新版インストール

概要

STM32ボードの書き込みツールであるst-flash(ST-Link utilities)のインストールについて記する。

インストール

以下の手順でインストールする：

```
# 関連ライブラリのインストール
sudo apt install git make cmake libusb-1.0-0-dev
sudo apt install gcc build-essential

cd ~/tools
# ソースのダウンロード
git clone https://github.com/stlink-org/stlink
cd stlink
# ビルド
cmake .
make

# インストール(関連ファイルのコピー)
cd bin
sudo cp st-* /usr/local/bin
cd ../lib
sudo cp *.so* /lib32

cd ..
sudo mkdir /usr/local/stlink/chips
sudo cp config/chips/*.* /usr/local/stlink/chips/

# udev登録
sudo cp config/udev/rules.d/49-stlinkv* /etc/udev/rules.d/
```

動作確認

```
# 接続確認
$ lsusb
＜省略＞
Bus 001 Device 021: ID 0483:374b STMicroelectronics ST-LINK
＜省略＞
#「ST-LINK/V2.x」があることを確認する

$ st-info --probe
Found 1 stlink programmers
  version:    V2J37S26
  serial:     066FFF525254667867254217
  flash:      0 (pagesize: 0)
  sram:       0
  chipid:     0x0000
  descr:      unknown device
# 「Found」があることを確認する

$ st-flash read dummy.bin 0 0xFFFF
st-flash 1.7.0-129-g7cc1fda
---------- old ------------
# Chip-ID file for F1xx Medium-density
#
chip_id 0x410
description F1xx Medium-density
flash_type  1
flash_size_reg 0x1ffff7e0
flash_pagesize 0x400
sram_size 0x5000
bootrom_base 0x1ffff000
bootrom_size 0x800
option_base 0x1ffff800
option_size 0x10
flags 2

---------- new ------------
# Chip-ID file for F1xx Medium-density
#
chip_id 0x410
description F1xx Medium-density
flash_type  1
flash_size_reg 0x0
flash_pagesize 0x400
sram_size 0x5000
bootrom_base 0x1ffff000
bootrom_size 0x800
option_base 0x1ffff800
option_size 0x10
```

```
flags 2

2022-01-05T22:52:18 INFO common.c: F1xx Medium-density: 20
2022-01-05T22:52:18 INFO common.c: read from address 000000
#----------------------------

# ST-LINKが故障している場合
$ st-flash read dummy.bin 0 0xFFFF
st-flash 1.7.0-129-g7cc1fda
2022-01-05T22:54:34 ERROR common.c: Can not connect to targ
Failed to connect to target
```

書き込み例

```
$ st-flash write .pio/build/genericSTM32F103CB/firmware.bin
st-flash 1.7.0-129-g7cc1fda
---------- old ------------
# Chip-ID file for F1xx Medium-density
#
chip_id 0x410
description F1xx Medium-density
flash_type  1
flash_size_reg 0x1ffff7e0
flash_pagesize 0x400
sram_size 0x5000
bootrom_base 0x1ffff000
bootrom_size 0x800
option_base 0x1ffff800
option_size 0x10
flags 2

---------- new ------------
# Chip-ID file for F1xx Medium-density
#
chip_id 0x410
description F1xx Medium-density
flash_type  1
flash_size_reg 0x0
flash_pagesize 0x400
sram_size 0x5000
bootrom_base 0x1ffff000
bootrom_size 0x800
option_base 0x1ffff800
option_size 0x10
flags 2

2022-01-05T23:09:41 INFO common.c: F1xx Medium-density: 20
file .pio/build/genericSTM32F103CB/firmware.bin md5 checksu
2022-01-05T23:09:41 INFO common.c: Attempting to write 1026
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Flash page at addr: 0x08
2022-01-05T23:09:41 INFO common.c: Finished erasing 11 page
```

```
2022-01-05T23:09:41 INFO common.c: Starting Flash write for
2022-01-05T23:09:41 INFO flash_loader.c: Successfully loade
2022-01-05T23:09:41 INFO flash_loader.c: Clear DFSR
 11/ 11 pages written
2022-01-05T23:09:42 INFO common.c: Starting verification of
2022-01-05T23:09:42 INFO common.c: Flash written and verifi
```

参考資料

Installing ST-Link v2 to flash STM32 targets on Linux
NucleoのST-Linkを使ったFirmware書き込み
stlink Tools Tutorial
flash a ST board with STLINK and Linux

以上

2022/2/7+
初版

new user for platformio

new user for platformio

概要

platformioのために新しいユーザーを設定する。
platformioを使用しているときに環境問題で動作が不良になったときがあり、
pythonの仮想環境をいったん消去してplatformioを再インストールしても 解決しなかったときがあった。
これを解決するために、新しいアカウントを作り、新しいユーザーを設定して、そこにログインして その環境にplatformioをインストールする。
ここでは、新しいアカウントを作る方法について記する。
(ubuntu-20.04を想定する)

新しいアカウントの作成方法

既存のログイン環境で以下を実行する：
(ここでは設定する新しいユーザー名をNEWUSERとする)

```
sudo su -
useradd -m NEWUSER
passwd NEWUSER
usermod -G sudo NEWUSER

ここで、いったんログアウトして
NEWUSERでログインする。

# デフォルトのshellがbashでないので
# bashに設定し直す

# 現在のshell
echo $SHELL
/usr/bin/sh

# bashのパスの確認
which bash
/usr/bin/bash

# shellのパスを上で確認したパスにする
chsh -s /usr/bin/bash

# 設定を有効にするため、いったんログアウトして、再ログインする

必要があれば既存の.bashrcの内容を
新しいログインの.bashrcにコピーする

https://beta-notes.way-nifty.com/blog/2021/02/post-2b331d.h
「arduinoフレームワーク用platformio.ini集」
の「PlatformIOのインストール」、「udev登録」を参照して
platformioを(再)インストールする。
```

参照情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

ディレクトリ関連：
「/bin」「/usr/bin」「/usr/local/bin」ディレクトリの使い分け
Ubuntu でホームディレクトリ内のディレクトリ名を英語表記に

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO

VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

以上

2021/2/21+
初版

platformio.ini for arduino

platformio.ini for arduino

概要

arduinoフレームワーク用platformio.ini集
本記事は。各ボード用のplatformio.iniを集め、platformioの使い方を説明したものである。
(ホストはubuntu20.04を想定している)

準備

開発ツールをインストールする前に 必要なものをインストールする：

```
sudo apt update

sudo apt install net-tools
sudo apt install git curl
sudo apt install gcc-arm-none-eabi build-essential

sudo apt install cmake
sudo apt install libusb-dev
sudo apt install libusb-1.0-0-dev

sudo apt install picocom

sudo apt install python3-distutils
sudo apt install python3-venv

# lib for Arduino IDE for microbit
sudo apt install libudev1:i386

# for Arduino IDE for ESP32/ESP8266
sudo apt install python-is-python3
pip install pyserial
pip uninstall serial
```

bt(bootterm)のインストール

```
git clone https://github.com/wtarreau/bootterm
cd bootterm
make
sudo make install
```

本ツールは、コマンドとして「bt」と入力すると最近有効になった
「/dev/ttyACMx または /dev/ttyUSBx」に接続する。
参照：Bootterm – a developer-friendly serial terminal program

PlatformIOのインストール

```
python3 -m venv pio_env
source pio_env/bin/activate

pip install platformio

インストール後も、このツールを使用する場合
同じディレクトリで以下を実行する：
source pio_env/bin/activate
# 「source」は、「.」でも良い
```

udev登録

以下を実行して、udevのrulesを登録する：

```
curl -fsSL https://raw.githubusercontent.com/platformio/pla

sudo udevadm control --reload-rules

sudo usermod -a -G dialout $USER
sudo usermod -a -G plugdev $USER
```

platformioの使い方

```
# プロジェクトのディレクトリを作る
# (一つのスケッチごとに一つのディレクトリ)
mkdir proj
cd proj

# ソースを置くディレクトリsrcを作る
mkdir src

# スケッチを作成する
# (テスト用には以降に出てくるsrc/ASCIItable.inoを使用する)
gedit src/xxxx.ino

# 使うボードに対応したplatformio.iniを作る
# (内容は以降に出てくるplatform.iniをそのままコピーする)
gedit platformio.ini

# スケッチをビルドする
# (最初の１回はライブラリ・ツールを自動的にダウンロードする)
pio run

# ボードをUSB接続して書き込む
pio run -t upload

# USBシリアルを使っているスケッチならば
# 以下のコマンドでシリアル接続して出力を表示させる
bt
# または
picocom -b115200 /dev/ttyACM0 (または/dev/ttyUSB0)
```

その他の使い方

```
# build結果をクリアする
pio run -t clean

# キャッシュをクリアする
# (ツールやライブラリがダウンロードし直しになるので注意のこと)
sudo rm -r .pio

# 環境を切り替えて書き込む
pio run -e f303 -t upload
pio run -e f103 -t upload
```

環境を切り替えて書き込む場合のplatformio.iniは
以下のように複数の環境[env:xxx]を持っていること：
platformio.ini

```
[env:f103]
platform = ststm32
board = nucleo_f103rb
framework = arduino
build_flags = -DNUCLEO_F103RB
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = stlink

#lib_deps =

[env:f303]
platform = ststm32
board = nucleo_f303k8
framework = arduino
build_flags = -DNUCLEO_F303K8
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = stlink

#lib_deps =
```

[env:xxxx]のxxxxの部分は環境名にあたり、ダブらない任意の名前であること。

テスト用スケッチ

src/ASCIItable.ino

```
/*
  ASCII table

  Prints out byte values in all possible formats:
  - as raw binary values
  - as ASCII-encoded decimal, hex, octal, and binary values

  For more on ASCII, see http://www.asciitable.com and http

  The circuit: No external hardware needed.

  created 2006
  by Nicholas Zambetti <http://www.zambetti.com>
  modified 9 Apr 2012
  by Tom Igoe

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/ASCIITable
*/

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native
  }

  // prints title with ending line break
  Serial.println("ASCII Table ~ Character Map");
}

// first visible ASCIIcharacter '!' is number 33:
int thisByte = 33;
// you can also write ASCII characters in single quotes.
// for example, '!' is the same as 33, so you could also us
// int thisByte = '!';

void loop() {
  // prints value unaltered, i.e. the raw binary version of
  // The Serial Monitor interprets all bytes as ASCII, so 3
  // will show up as '!'
  Serial.write(thisByte);

  Serial.print(", dec: ");
  // prints value as string as an ASCII-encoded decimal (ba
  // Decimal is the default format for Serial.print() and S
```

```
  // so no modifier is needed:
  Serial.print(thisByte);
  // But you can declare the modifier for decimal if you wa
  // this also works if you uncomment it:

  // Serial.print(thisByte, DEC);


  Serial.print(", hex: ");
  // prints value as string in hexadecimal (base 16):
  Serial.print(thisByte, HEX);

  Serial.print(", oct: ");
  // prints value as string in octal (base 8);
  Serial.print(thisByte, OCT);

  Serial.print(", bin: ");
  // prints value as string in binary (base 2) also prints
  Serial.println(thisByte, BIN);

  // if printed last visible character '~' or 126, stop:
  if (thisByte == 126) {    // you could also use if (thisB
    // This loop loops forever and does nothing
    while (true) {
      continue;
    }
  }
  // go on to the next character
  thisByte++;
}
```

本スケッチはarduinoのサンプルそのもの(bpsのみ変更)である。

plaformio.ini

(1)platformio.iniの先頭にあたる以下はコメントにあたるので省略している：

```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra fla
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and exam
; https://docs.platformio.org/page/projectconf.html
```

(2)追加するライブラリがある場合は以下のように 「lib_dep= 」以降にライブラリのurlなどを追加する：

```
lib_deps =
    #https://github.com/arduino-libraries/NTPClient.git
    arduino-libraries/NTPClient@^3.1.0
```

(3)以下、各ボード用のplatformio.iniになる。

Arduino Uno

```
[env:uno]
platform = atmelavr
board = uno
framework = arduino
#build_flags = -Dxxxx
upload_protocol = arduino
monitor_speed = 115200
lib_ldf_mode = deep+

#lib_deps =
```

wio-temirnal

```
[env:seeed_wio_terminal]
platform = atmelsam
board = seeed_wio_terminal
framework = arduino
build_flags = -DWIO_TERMINAL
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    https://github.com/Seeed-Studio/Seeed_Arduino_mbedtls/a
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcUnifie
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcBLE/ar
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcWiFi/a
    https://github.com/Seeed-Studio/Seeed_Arduino_FreeRTOS/
    https://github.com/Seeed-Studio/Seeed_Arduino_FS/archiv
    https://github.com/Seeed-Studio/Seeed_Arduino_SFUD/arch
    #
    https://github.com/Seeed-Studio/Seeed_Arduino_LCD/archi
    # 551
    #https://github.com/arduino-libraries/NTPClient.git
    arduino-libraries/NTPClient@^3.1.0
```

XIAO

```
[env:seeed_xiao]
platform = atmelsam
board = seeed_xiao
framework = arduino
build_flags = -DXIAO
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    olikraus/U8g2 @ ^2.28.7
```

Feather M4

```
[env:adafruit_feather_m4]
platform = atmelsam
board = adafruit_feather_m4
framework = arduino
build_flags = -DFEATHER_M4
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

#lib_deps =
```

Grand Central M4

```
[env:adafruit_grandcentral_m4]
platform = atmelsam
board = adafruit_grandcentral_m4
framework = arduino
build_flags = -DGRANDCENTRAL_M4
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

#lib_deps =
```

M5Atom

```
[env:esp32dev]
platform = espressif32
#board = esp32dev
board = m5stick-c
framework = arduino
build_flags = -DM5ATOM
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    # use M5Atom lib
    3113
    # use "FastLED"
    126
```

M5StickC

```
[env:esp32dev]
platform = espressif32
board = m5stick-c
framework = arduino
build_flags = -DM5STICK_C
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    m5stack/M5StickC @ ^0.2.0
```

リセット方法：
(1)「M5」と書かれたボタンを正面に見て左側にあるボタンを6秒長押して電源を切る。
(2)そのボタンを再度、２秒長押しをして電源を入れる。（リセットになる）

M5StickCPlus

```
[env:esp32dev]
platform = espressif32
board = m5stick-c
framework = arduino
build_flags = -DM5STICK_CP
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    m5stack/M5StickCPlus @ ^0.0.1
```

リセット方法：
(1)「M5」と書かれたボタンを正面に見て左側にあるボタンを6秒長押して電源を切る。
(2)そのボタンを再度、２秒長押しをして電源を入れる。（リセットになる）

M5Stack Fire

```
[env:esp32dev]
platform = espressif32
board = m5stack-fire
framework = arduino
build_flags = -DM5STACK_FIRE
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    m5stack/M5Stack
    https://github.com/lovyan03/LovyanGFX.git
```

Power on/off:

- Power on: click the red power button on the left
- Power off: Quickly double-click the red power button on the left

ESP32

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
build_flags = -DESP32
monitor_speed = 115200
lib_ldf_mode = deep+

#lib_deps =
```

ESP8266

```
[env:huzzah]
platform = espressif8266
board = esp_wroom_02
framework = arduino
build_flags = -DESP8266
monitor_speed = 115200
lib_ldf_mode = deep+

#lib_deps =
```

micro:bit

```
[env:bbcmicrobit]
platform = nordicnrf51
board = bbcmicrobit
framework = arduino
build_flags = -DMICROBIT -DNRF51_S110
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = cmsis-dap

lib_deps =
    https://github.com/adafruit/Adafruit_BusIO/archive/mast
    https://github.com/sparkfun/SparkFun_MAG3110_Breakout_B
    https://cdn-learn.adafruit.com/assets/assets/000/046/21
    https://github.com/stm32duino/LSM303AGR/archive/master.
    https://github.com/adafruit/Adafruit-GFX-Library/archiv
    #
    https://github.com/sandeepmistry/arduino-BLEPeripheral/
    https://github.com/adafruit/Adafruit_Microbit/archive/m
    #
    https://github.com/ht-deko/microbit_Screen/archive/mast
```

Nucleo STM32F103RB

```
[env:nucleo_f103rb]
platform = ststm32
board = nucleo_f103rb
framework = arduino
build_flags = -DNUCLEO_F103RB
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = stlink

#lib_deps =
```

Nucleo STM32F303K8

```
[env:nucleo_f303k8]
platform = ststm32
board = nucleo_f303k8
framework = arduino
build_flags = -DNUCLEO_F303K8
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = stlink

#lib_deps =
```

TeensyLC

```
[env:teensylc]
platform = teensy
board = teensylc
framework = arduino
#build_flags = -Dxxxx
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = teensy-cli

#lib_deps =
```

Teensy3.6

```
[env:teensy36]
platform = teensy
board = teensy36
framework = arduino
#build_flags = -Dxxxx
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = teensy-cli

#lib_deps =
```

Teensy4.0

```
[env:teensy40]
platform = teensy
board = teensy40
framework = arduino
#build_flags = -Dxxxx
monitor_speed = 115200
lib_ldf_mode = deep+

upload_protocol = teensy-cli

#lib_deps =
```

ビルド・エラー

Arduino-IDEのコンパイラと異なり、platformioのコンパイラは、関数定義の後方参照を許さないので、 この場合、関数の未定義エラーになる。したがって、Arduino-IDEでビルドできているソースでもエラーになることがある。このときの対応方法は、未定義エラーになる関数定義をプロトタイプ宣言としてソースの先頭に置き、後方参照を解消する。(関数定義の本体の位置はそのままで移動させる必要はない)

参考情報

M5Stack関連：

- PlatformIO M5Stack開発の現状
- https://github.com/m5stack/M5StickC-Plus.git
- https://github.com/m5stack/M5StickC.git
- https://github.com/m5stack/M5Stack.git

micro:bit関連：

- micro:bit Arduino/MBED開発ツール(v2)(micro:bit-v2対応,linux版)
- micro:bit v2 で遊ぶ

platformio関連：

```
https://docs.platformio.org/en/latest/platforms/creating_bo

Installation
1.Create boards directory in core_dir if it doesn't exist.
2.Create myboard.json file in this boards directory.
3.Search available boards via pio boards command. You shoul
```

- PlatformIO Core (CLI)
- Arduino-IDEとPlatformioのコンパイラーの挙動の違いについて
- ubuntu20.04をインストールする

以上

2021/8/20
CrytalDiskInfoの結果を追加した。

2021/8/16
初版

SSD benchmark

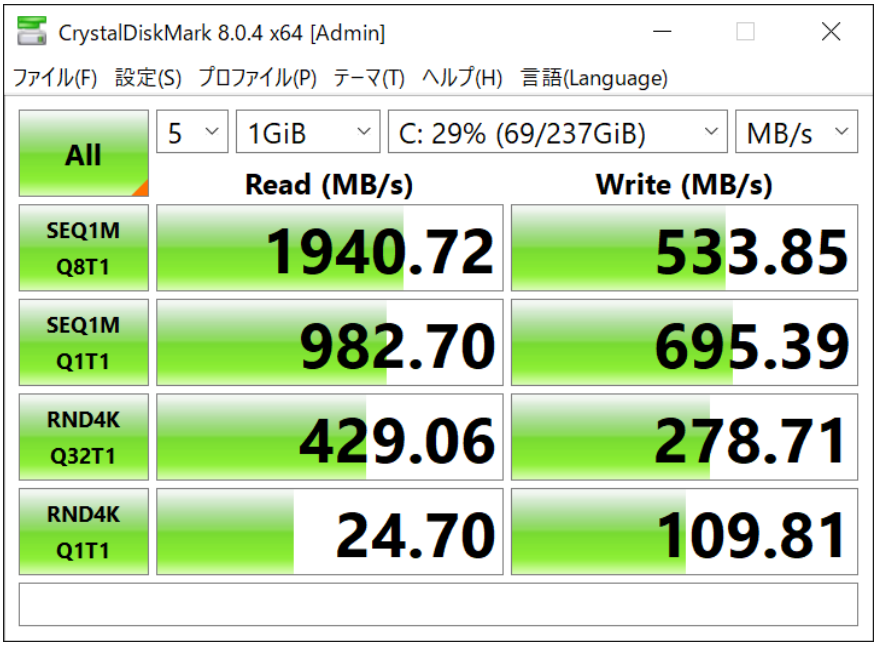
SSD benchmark

概要

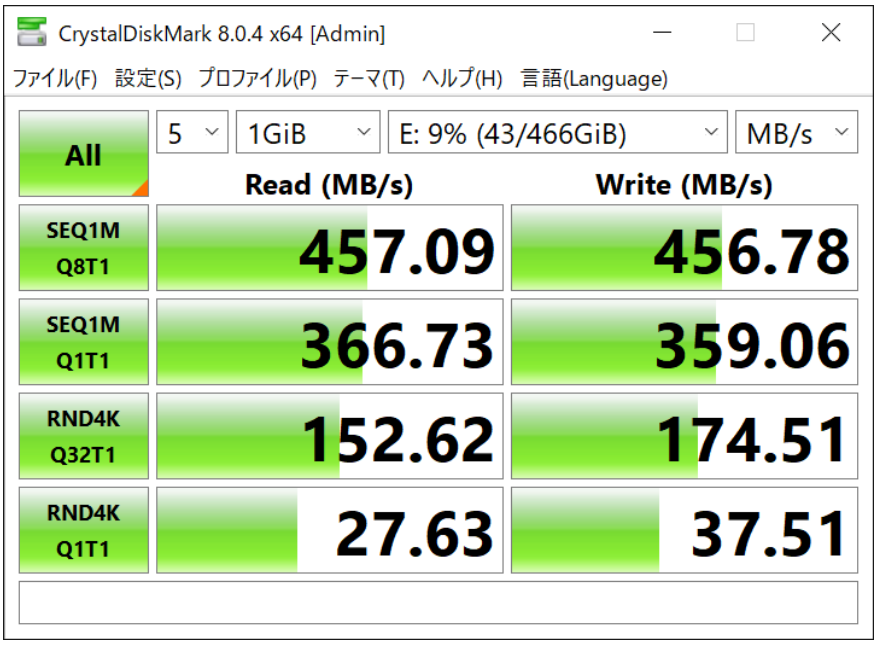
ノートPCのSSDベンチマークの結果についてのメモである。
使用ノートPCは「ThinkPad X1 Carbon(20HQ)」になる。
使用ベンチマークは「CrytalDiskMark8」になる。
CrytalDiskMark8ダウンロード

ベンチマーク結果

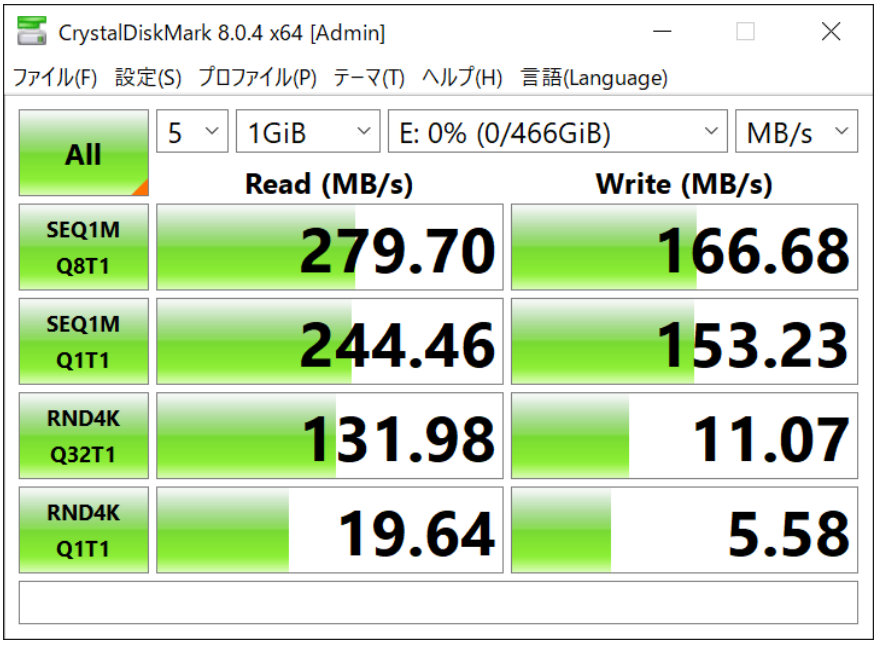
C:(内蔵NVMe)



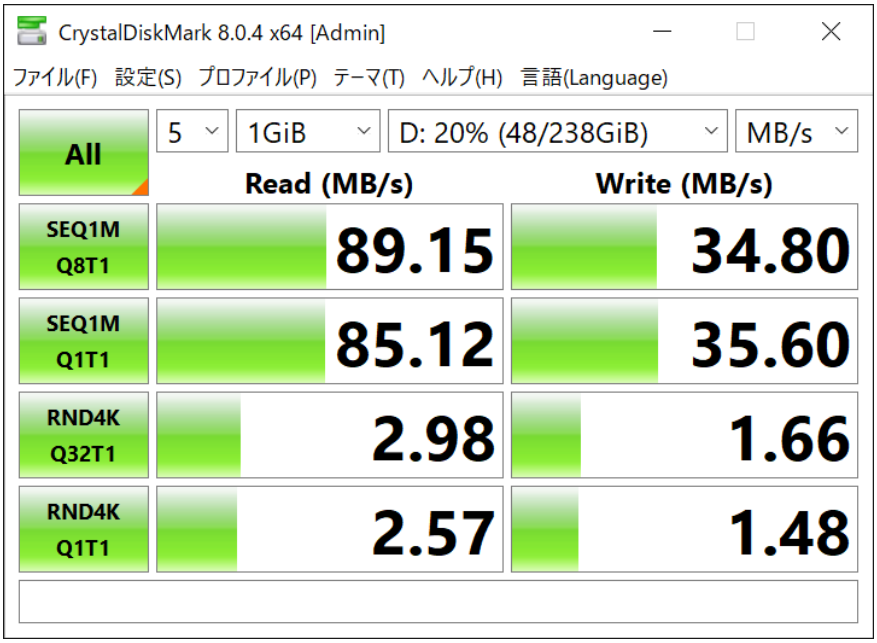
E:(外付けUSB3.0 NVMe)



E:(外付けUSB3.0 SATA-SSD)

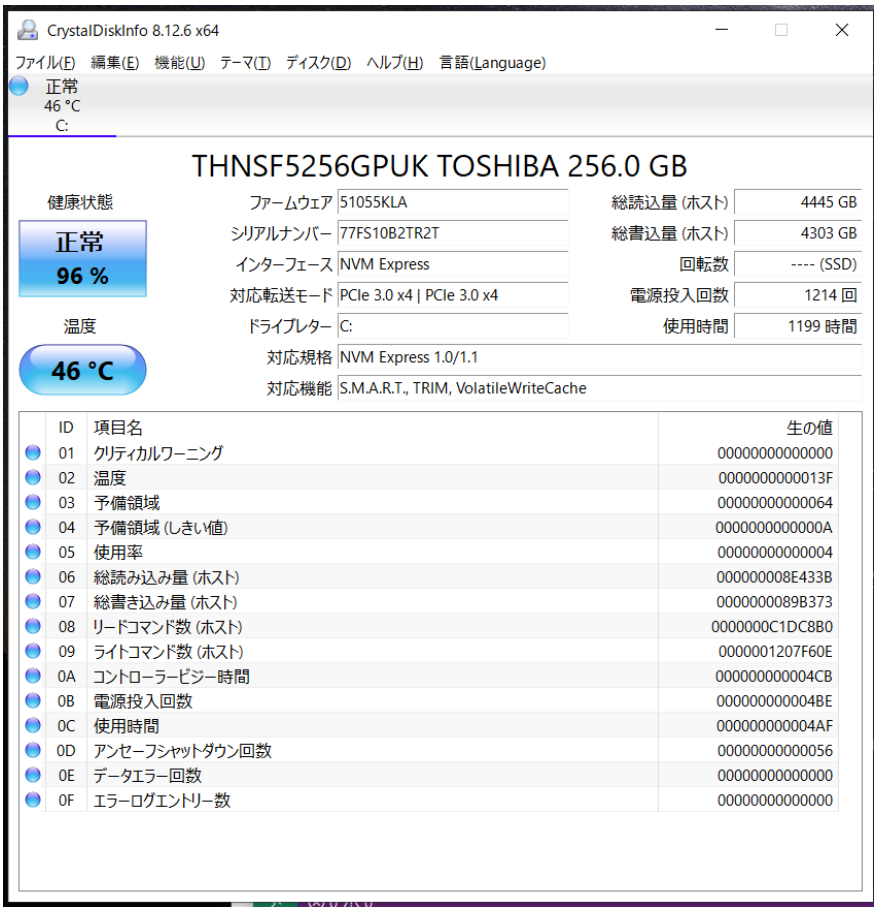


D:(SD)



CrytalDiskInfoの結果

C:(内蔵)



E:(外付けUSB3.0 NVMe 500GB)

E:(外付けUSB3.0 NVMe 1TB)

参照情報

ThinkPad X1 Carbonのマシンタイプの識別
ThinkPad X1 Carbon ハードウェア保守マニュアル
ThinkPad X1 Carbon ユーザー・ガイド

「CrystalDiskMark 8」がリリース！自作UIライブラリ「Project Priscilla」を採用
CrystalDiskInfo HDD/SSDの健康状態をチェック

SAMSUNG 500GB (980 MZ-V8V500B/IT)
SSDケース (HDE-13)
NVMe M.2 SSD + 外付けケースは安くて爆速！1GB/sの高速SSDストレージを低価格で！

WD 500GB SATA (WD Blue 3D NAND SATA WDS500G2B0A)
USB3.0接続 2.5型 SATA SSD/HDDケース

以上

2021/2/7+

MicroPython CircuitPython Performace Test (v2)

MicroPython CircuitPython Performace Test (v2)

概要

MicroPython/CircuitPython Performance Test この記事は「MicroPython Performance Test」を見直して新たな結果を追加した第2版にあたる。

pyborad/STM32用MicroPythonスクリプト

PerformaceTest.py

```
# Peformace Test

import pyb

def performanceTest():
    millis = pyb.millis
    endTime = millis() + 10000
    count = 0
    while millis() < endTime:
        count += 1
    print("Count: ", count)

performanceTest()
```

ESP32/ESP8266用MicroPythonスクリプト

PerformaceTestESP32.py

```
# Peformace Test for ESP32
from machine import RTC
rtc = RTC()
rtc.datetime((2020, 2, 9, 1, 12, 48, 0, 0))
# (year, month, day[, hour[, minute[, second[, microsecond[

def performanceTest():
    secs = rtc.datetime()[6]
    endTime = secs + 10
    count = 0
    while rtc.datetime()[6] < endTime:
        count += 1
    print("Count: ", count)

performanceTest()
```

PC/BareMetal-RpiZeo/Linux-Rpizero/Maixduino用スクリプト

PerformanceRpiZero.py

```
# Peformace Test RpiZero
import time
def performanceTest():
    msec = time.ticks_ms
    endTime = msec() + 10000
    count = 0
    while msec() < endTime:
        count += 1
    print("Count: ", count)

performanceTest()
```

CC3200用MicroPythonスクリプト

PerformanceTestCC3200.py

```
# Peformace Test for CC3200
from machine import RTC
rtc = RTC(datetime=(2020, 2, 9, 10, 11, 0, 0, None))

def performanceTest():
     secs = rtc.now()[5]
     endTime = secs + 10
     count = 0
     while rtc.now()[5] < endTime:
         count += 1
     print("Count: ", count)
performanceTest()
```

CircuitPython/Python3用スクリプト

performanceTestCPY.py

```
# Peformace Test CircuitPython
from time import monotonic_ns
def performanceTest():
    endTime = monotonic_ns() + 10000000000 # 10 sec
    count = 0
    while monotonic_ns() < endTime:
        count += 1
    print("Count: ", count)

performanceTest()
```

XIAO CircuitPython用スクリプト

PerformanceCircuitPython_XIAO.py

```
# Peformace Test XIAO CircuitPython
from time import monotonic
def performanceTest():
    endTime = monotonic() + 10.0 # 10 sec
    count = 0
    while monotonic() < endTime:
        count += 1
    print("Count: ", count)

performanceTest()
```

実行例

```
# pyboard/STM32の場合
$ ./pyboard.py --device /dev/ttyACM0 PerformanceTest.py
Count:  2825625

# ESP32/ESP866の場合
$ ./pyboard.py --device /dev/ttyUSB0 PerformanceTestESP32.p
Count:  39187
```

結果

ボード名	M/C/P	Count
F767ZI	M	5,676,566
F446RE	M	2,825,625
F4Disco	M	2,882,769
Maixduino	M	2,218,879
Pico-MP	M	1,507,516
F401RE	M	1,362,752
L476RG	M	1,089,347
PyPortal	C	474,734
Grand-Central-M4-Express	C	445,404
Feather-M4-Express	C	438,444
Teensy4.0	C	432,413
Pico-CPY	C	341,033
XIAO	C	174,388
Circuit-Playground-Express	C	121,800
ESP8266	M	64,049
ESP32	M	39,187
CC3200	M	5,529
BareMetal-RpiZero	M	680,525
Linux-RpiZero	P	6,370,022
PC(linux)	P	161,265,687
micropython-Rpi4	M	16,117,283
python3-Rpi4	P	11,359,199

M/C/Pは、pythonの種類を表し、
M:MicroPython、
C:CircuitPython、
P:Python3
を表す。

コメント

・Pico-MPは、RPI_Picoボードの MicroPython、Pico-CPYは、
RPI_PicoボードのCircuitPythonを意味する。

・BareMetal-RpiZeroは、RpiZeroでのBareMetalのmicropython、 Linux-RpiZeroは、linux上でのmicropython、 PC(Linux)は、PCのLinux上でのmicropythonを意味する。

・MaixduinoはMaixduinoのmicropythonであるMaixPyを意味する。

・Teensy4.0はTeensy4.0のCircuitPythonを意味する。

・python3-Rpi4は「python3 performanceCircuitPython.py」の数字、micropython-Rpi4は「micropython performanceCircuitPython.py」の数字を意味する。

なお、PC(linux)のCPUは、「Intel® Core™ i7-3520M CPU @ 2.90GHz × 4)」である。

・STM32系(F7xx/F4xx/L4xx)がESP32/ESP8266に比べてダントツに速いようだ。

・Linux-RpiZeroとBareMetal-RpiZeroの数字の違いはCPUキャッシュの有効/無効の差と思われる。

・CC3200は、他のMicroPythonボードに比べて極端に遅い結果だが、省電優先設計のせいだと思われる。

・Teensy4.0のCircuitPythonはmonotonic_ns()のオーバーヘッドが大きい気がする。

・XIAOとCircuit-Playground-Expressは、同じプロセッサATSAMD21(Cortex-M0+,48MHz)にもかかわらず、性能差が出ているのは実装の違いで、XIAOのほうは、floatで動いているのに対して、Circuit-Playground-Expressのほうは、long_intで動いていることによる差が出ていると考えている。

・Pico-MP,Pico-CPYはCortex-M0であるがクロックが他のM0より高いこと(最大133Mz)と浮動小数点ライブラリが最適化されていることでM4並みの性能が出ているようだ。MP,CPYの性能差は最適化が進んでいるかどうかの差と思われる。

参照情報

PicoボードにMicropython/CircuitPythonをインストールする

RaspberryPiのpython3でCircuitPythonのAPIを使用する

XIAOにCircuitPythonをインストールする

Circuit-Playground-ExpressにCircuitPythonをインストールする

Teensy4.0にCurcuitPythonをインストールする

BareMetalのMicropythonをRaspberryPi_Zeroにインストールしてみる

NUCLEO-F767ZIにMicropythonをインストールする(v2)
Nucleo-L476RGにMicroPythonをインストールする
Nucleo-F401REにMicroPythonをインストールする
STM32F4-Discovery」にMicroPythonをインストールする
NUCLEO-F446REにMicropythonをインストールする(v2)
NUCLEO F446RE MicroPython インストール方法

CC3200 MicroPython インストール

ESP32のMicroPythonのインストール方法
ESP-WROOM-02 MicroPython インストール方法

MicroPythonのツールとしてpyboad.pyを使う
The pyboard.py tool

ＥＳＰ３２－ＤｅｖＫｉｔＣ　ＥＳＰ－ＷＲＯＯＭ－３２開発ボード
MicroPython - Quick reference for the ESP32

ampyを用いたMicroPythonのファイル操作とプログラム実行

以上

2022/1/20
バッテリを交換したので、その状態を追加した。

2021/9/9
バッテリー状態を追加した。

2021/8/24
SATA_SSDの速度結果を追加した。

2021/8/23
初版

ThinkPad X1 Carbon(20HQ) linux install

ThinkPad X1 Carbon(20HQ) linux install

概要

ThinkPad X1 Carbon(20HQ)にlinuxをインストールした際のメモである。

準備

1.クリーンインストールするために、あらかじめ、M.2_NVMe_SSDを用意して NVM2用SSDアダプター(USB3.0)に組み込み、別のPCなどで、ext3/ext4にフォーマットしておく。
2.USB-DVDドライブの用意
3.linuxインストール用DVD(日経linuxの付録などでよい)の用意
実際には、ubuntu20.04を使用した。
4.biosのメニューが選べるようにwindows10の高速スタートアップ(高速起動)を無効化する
以下を参考にして高速起動を無効にする。
Windows 10で高速スタートアップ（高速起動）を無効化する方法

M.2_NVMe_SSDの交換

1.内蔵バッテリーの無効化
(1)バッテリー駆動にしてPCの電源をオンにする。ロゴ画面が表示されたら、すぐに F1 を押すと、ThinkPad Setup に入る。
(2)「Config」 → 「Power」を選択する。「Power」サブメニューが表示される。

(3)「Disable built-in battery」を選択し、Enter キーを押す。
(4)「Setup Confirmation」ウィンドウで、「Yes」を選択する。内蔵バッテリーが無効になり、PCの電源が自動的に切れる。

2.マイクロ SIM カードの取り外し
スマフォなどと同様にSIMカードの穴にピンを入れてカードを取り外す。

3.「ベース・カバー・アセンブリー」の取り外し
以下を参照してベースカバーを取り外す。
ベースカバー・アセンブリの取り外し - Thinkpad x1 Carbon
今回の件では無関係だが、バッテリーを交換する際の情報になるのでバッテリーの型番をメモしておいたほうが良い。
ちなみに、本機の場合、「FRU P/N 01AV494」だった。

4.M.2_NVM2_SSDの交換
以下の動画を参考に準備で用意したフォーマット済みM.2_NVM2_SSDに交換する。
取り付け、取り外し動画 - ThinkPad X1 Carbon Gen 5 (20HQ, 20HR, 20K3, 20K4),ThinkPad X1 Carbon Gen 6 (20KG, 20KH)
なお、交換の際、ヒートシンク用サーマルパッドが貼ってあるようなら、それをはがし新しいボードに張り替える。 新品の持ち合わせがあるようなら、それを使うほうが良いと思われる。

5.「ベース・カバー・アセンブリー」の取り付け
以上で作業は終わったのでカバーを取り付ける。

linuxのインストール

1.インストール用DVDを入れたUSB-DVDドライブをPCに接続する。
2.ACアダプターを接続してPCの電源をオンにする。ロゴ画面が表示されたら、すぐに F12 を連打する。しばらくすると、ブートデバイスを選択するメニューが表示されるので、USB-DVDを選択する。するとDVDからインストールプログラムが読み込まれるので、あとは、linuxのインストールを選択する。

linux起動直後の作業

1.ホームディレクトリ内のディレクトリ名を英語表記にするために 以下を実行する：

```
LANG=C xdg-user-dirs-gtk-update
```

ウィンドウが開くので、" Don't ask me this again " にチェックを入れて、 [Update Names] をクリックする。

2.googleなどでエラーを検索しやすいようにエラーメッセージを英語にするために以下を.bashrcに登録する：

```
export LC_ALL=en_US.UTF-8
```

その他のアプリなどのインストールについては以下を参照のこと。
ubuntu20.04をインストールする
arduinoフレームワーク用platformio.ini集

起動時のログ例

```
$ uname -a
Linux XXXX 5.11.0-27-generic #29~20.04.1-Ubuntu SMP Wed Aug

$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            7.6G     0  7.6G   0% /dev
tmpfs           1.6G  1.8M  1.6G   1% /run
/dev/nvme0n1p5  916G  8.6G  861G   1% /
tmpfs           7.6G     0  7.6G   0% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           7.6G     0  7.6G   0% /sys/fs/cgroup
/dev/loop1      256M  256M     0 100% /snap/gnome-3-34-1804
/dev/loop0       56M   56M     0 100% /snap/core18/1885
/dev/loop2       63M   63M     0 100% /snap/gtk-common-them
/dev/loop3       50M   50M     0 100% /snap/snap-store/467
/dev/loop4       30M   30M     0 100% /snap/snapd/8542
/dev/nvme0n1p1  511M  4.0K  511M   1% /boot/efi
tmpfs           1.6G   24K  1.6G   1% /run/user/1000

$ iwconfig
lo        no wireless extensions.

enp0s31f6  no wireless extensions.

wlp4s0    IEEE 802.11  ESSID:"elecom-xxxxxx"
          Mode:Managed  Frequency:5.18 GHz  Access Point: 0
          Bit Rate=520 Mb/s   Tx-Power=22 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:
          Power Management:on
          Link Quality=33/70  Signal level=-77 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid
          Tx excessive retries:3  Invalid misc:1   Missed b

wwan0     no wireless extensions.

$ ls -l /dev/nvm*
crw------- 1 root root 240, 0  8月 22 14:16 /dev/nvme0
brw-rw---- 1 root disk 259, 0  8月 22 14:22 /dev/nvme0n1
brw-rw---- 1 root disk 259, 1  8月 22 14:22 /dev/nvme0n1p1
brw-rw---- 1 root disk 259, 2  8月 22 14:22 /dev/nvme0n1p2
brw-rw---- 1 root disk 259, 3  8月 22 14:22 /dev/nvme0n1p5

# nvmeはnvmeストレージを意味する
```

[参考]SSD速度

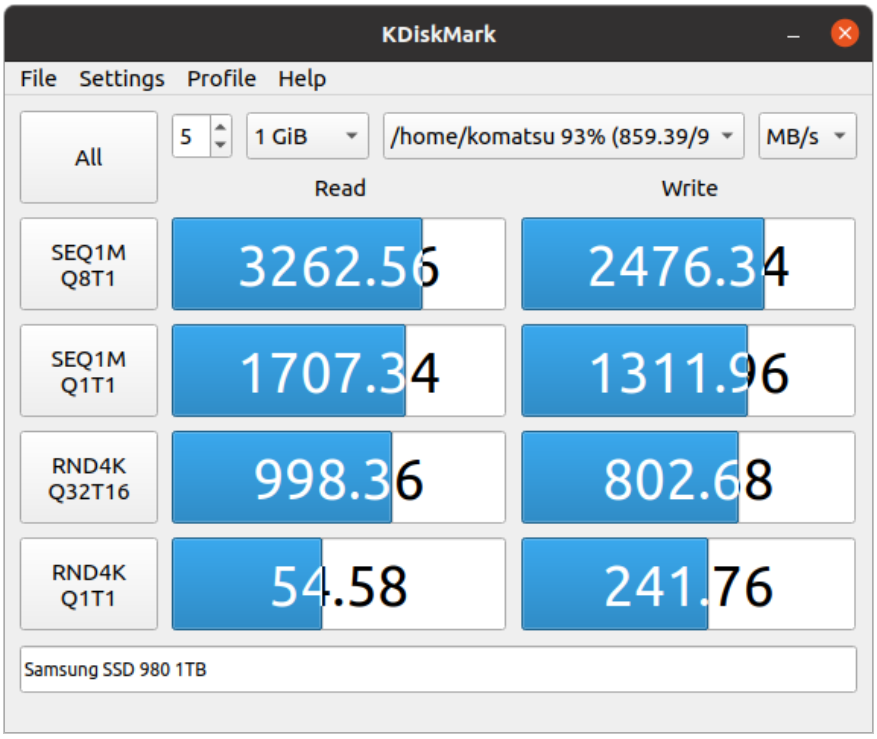
インストール：

```
#ツールをインストールする
sudo add-apt-repository ppa:jonmagon/kdiskmark
sudo apt install kdiskmark
```

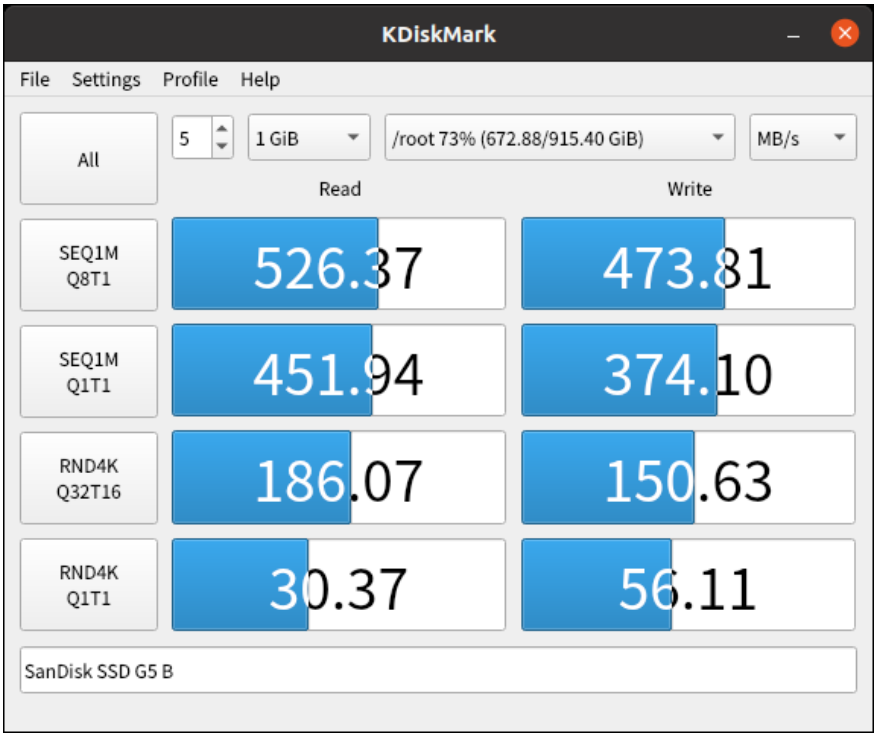
実行：

```
kdiskmark
#画面が表示されたら[All]をクリックすると測定が開始される。
#測定が終わるまで待つ
#測定が完了したら[Alt]+[PrtSc]を押す。
#画面の画像が[Pictures]に保存される。
#[File]/[Save]で測定データをテキストで保存できる。
```

結果：



比較(SATA_SSD)：



[参考]S.M.A.R.T.

インストール：

```
#S.M.A.R.T.情報を取得するために以下をインストールする：

sudo apt install smartmontools
```

実行：

```
#diskをスキャンする
sudo smartctl --scan
/dev/nvme0 -d nvme # /dev/nvme0, NVMe device

#デバイス情報を表示する
sudo smartctl -i /dev/nvme0
smartctl 7.1 2019-12-30 r5022 [x86_64-linux-5.11.0-27-gener
Copyright (C) 2002-19, Bruce Allen, Christian Franke, www.s

=== START OF INFORMATION SECTION ===
Model Number:                       Samsung SSD 980 1TB
Serial Number:                      S649NJ0R231646W
Firmware Version:                   1B4QFXO7
PCI Vendor/Subsystem ID:            0x144d
IEEE OUI Identifier:                0x002538
Total NVM Capacity:                 1,000,204,886,016 [1.00
Unallocated NVM Capacity:           0
Controller ID:                      5
Number of Namespaces:               1
Namespace 1 Size/Capacity:          1,000,204,886,016 [1.00
Namespace 1 Utilization:            28,024,627,200 [28.0 GB
Namespace 1 Formatted LBA Size:     512
Namespace 1 IEEE EUI-64:            002538 d21140cea4
Local Time is:                      Mon Aug 23 15:58:13 202

#S.M.A.R.T.情報を表示する
sudo smartctl -a /dev/nvme0
smartctl 7.1 2019-12-30 r5022 [x86_64-linux-5.11.0-27-gener
Copyright (C) 2002-19, Bruce Allen, Christian Franke, www.s

=== START OF INFORMATION SECTION ===
Model Number:                       Samsung SSD 980 1TB
Serial Number:                      S649NJ0R231646W
Firmware Version:                   1B4QFXO7
PCI Vendor/Subsystem ID:            0x144d
IEEE OUI Identifier:                0x002538
Total NVM Capacity:                 1,000,204,886,016 [1.00
Unallocated NVM Capacity:           0
Controller ID:                      5
Number of Namespaces:               1
Namespace 1 Size/Capacity:          1,000,204,886,016 [1.00
Namespace 1 Utilization:            28,024,823,808 [28.0 GB
Namespace 1 Formatted LBA Size:     512
Namespace 1 IEEE EUI-64:            002538 d21140cea4
Local Time is:                      Mon Aug 23 16:01:17 202
Firmware Updates (0x16):            3 Slots, no Reset requi
Optional Admin Commands (0x0017):   Security Format Frmw_DL
```

```
Optional NVM Commands (0x0055):     Comp DS_Mngmt Sav/Sel_F
Maximum Data Transfer Size:         512 Pages
Warning  Comp. Temp. Threshold:     82 Celsius
Critical Comp. Temp. Threshold:     85 Celsius
Namespace 1 Features (0x10):        *Other*

Supported Power States
St Op     Max   Active     Idle   RL RT WL WT  Ent_Lat  Ex_
 0 +     5.24W       -        -    0  0  0  0        0
 1 +     4.49W       -        -    1  1  1  1        0
 2 +     2.19W       -        -    2  2  2  2        0
 3 -   0.0500W       -        -    3  3  3  3      210    1
 4 -   0.0050W       -        -    4  4  4  4     1000    9

Supported LBA Sizes (NSID 0x1)
Id Fmt  Data  Metadt  Rel_Perf
 0 +     512       0         0

=== START OF SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED

SMART/Health Information (NVMe Log 0x02)
Critical Warning:                   0x00
Temperature:                        37 Celsius
Available Spare:                    100%
Available Spare Threshold:          10%
Percentage Used:                    0%
Data Units Read:                    223,023 [114 GB]
Data Units Written:                 286,175 [146 GB]
Host Read Commands:                 22,516,048
Host Write Commands:                22,648,134
Controller Busy Time:               2
Power Cycles:                       18
Power On Hours:                     1
Unsafe Shutdowns:                   7
Media and Data Integrity Errors:    0
Error Information Log Entries:      0
Warning  Comp. Temperature Time:    0
Critical Comp. Temperature Time:    0
Temperature Sensor 1:               37 Celsius
Temperature Sensor 2:               36 Celsius
Thermal Temp. 2 Transition Count:   2

Error Information (NVMe Log 0x01, max 64 entries)
No Errors Logged
```

[参考]バッテリー状態(バッテリーへたり確認)

tool install&help

```
sudo apt install acpi

$ acpi -h
Usage: acpi [OPTION]...
Shows information from the /proc filesystem, such as batter
thermal information.

  -b, --battery            battery information
  -i, --details            show additional details if avail
                             - battery capacity information
                             - temperature trip points
  -a, --ac-adapter         ac adapter information
  -t, --thermal            thermal information
  -c, --cooling            cooling information
  -V, --everything         show every device, overrides abo
  -s, --show-empty         show non-operational devices
  -f, --fahrenheit         use fahrenheit as the temperatur
  -k, --kelvin             use kelvin as the temperature un
  -d, --directory <dir>    path to ACPI info (/sys/class re
  -p, --proc               use old proc interface instead o
  -h, --help               display this help and exit
  -v, --version            output version information and e

By default, acpi displays information on installed system b
Non-operational devices, for example empty battery slots ar
The default unit of temperature is degrees celsius.

Report bugs to Michael Meskes <meskes@debian.org>.
```

フル充電状態：

```
$ acpi -V(大文字であることに注意)
Battery 0: Full, 100%
Battery 0: design capacity 4456 mAh, last full capacity 240
Adapter 0: on-line
Thermal 0: ok, 32.0 degrees C
Thermal 0: trip point 0 switches to mode critical at temper
Cooling 0: Processor 0 of 10
Cooling 1: intel_powerclamp no state information available
Cooling 2: Processor 0 of 10
Cooling 3: pch_skylake no state information available
Cooling 4: Processor 0 of 10
Cooling 5: x86_pkg_temp no state information available
Cooling 6: iwlwifi 0 of 20
Cooling 7: Processor 0 of 10
Cooling 8: iwlwifi_1 no state information available
```

放電中(ACアダプターを外している状態)：

```
$ acpi -V
Battery 0: Discharging, 99%, 06:57:56 remaining
Battery 0: design capacity 4519 mAh, last full capacity 243
Adapter 0: off-line
Thermal 0: ok, 36.0 degrees C
Thermal 0: trip point 0 switches to mode critical at temper
Cooling 0: Processor 0 of 10
Cooling 1: intel_powerclamp no state information available
Cooling 2: Processor 0 of 10
Cooling 3: pch_skylake no state information available
Cooling 4: Processor 0 of 10
Cooling 5: x86_pkg_temp no state information available
Cooling 6: iwlwifi 0 of 20
Cooling 7: Processor 0 of 10
Cooling 8: iwlwifi_1 no state information available
`
```

以上の数値により、バッテリー状態は、53%で、約7時間くらい使用可能ということになる。
50%を割ったらバッテリーの交換時期と考えると、そろそろ交換時期ということになる。
(それでも、約7時間もつようなので、現状、そのまま使用する予定でいる)

[参考]バッテリー状態(バッテリ交換後)

実際の使用においてバッテリが持たなくなったので交換した。
(2022/1/20)

ネットで「Lenovo レノボ ThinkPad X1 Carbon 交換用互換バッテリー 第5世代 (2017) 第6世代(2018) 向け 01AV429 01AV494 01AV430 01AV431 対応」で検索して、互換バッテリを入手した。

交換方法は「内蔵バッテリーの取り付け、取り外し - Thinkpad x1 Carbon
」を参照した。
バッテリ筐体の精度の問題だと思うが、PC本体のコネクタとの勘合が多少悪いようで、取り付けに多少手こずったが、なんとか固定できた。

充電中：

```
acpi -V
Battery 0: Charging, 99%, 00:00:05 until charged
Battery 0: design capacity 4343 mAh, last full capacity 434
Adapter 0: on-line
Thermal 0: ok, 30.0 degrees C
Thermal 0: trip point 0 switches to mode critical at temper
Cooling 0: Processor 0 of 10
Cooling 1: intel_powerclamp no state information available
Cooling 2: Processor 0 of 10
Cooling 3: pch_skylake no state information available
Cooling 4: Processor 0 of 10
Cooling 5: x86_pkg_temp no state information available
Cooling 6: iwlwifi 0 of 20

Cooling 7: TCC Offset 25 of 63
Cooling 8: Processor 0 of 10
Cooling 9: iwlwifi_1 no state information available
```

なぜか、「Charging, 100%」には、ならなかった。
「design capacity」と「last full capacity」のmAh値が同じなので、バッテリ交換が正常に行われたことを判断できる。

放電中(フル充電後、ACアダプターを外している状態)：

```
acpi -V
Battery 0: Discharging, 99%, 08:11:39 remaining
Battery 0: design capacity 4516 mAh, last full capacity 451
Adapter 0: off-line
Thermal 0: ok, 35.0 degrees C
Thermal 0: trip point 0 switches to mode critical at temper
Cooling 0: Processor 0 of 10
Cooling 1: intel_powerclamp no state information available
Cooling 2: Processor 0 of 10
Cooling 3: pch_skylake no state information available
Cooling 4: Processor 0 of 10
Cooling 5: x86_pkg_temp no state information available
Cooling 6: iwlwifi 0 of 20
Cooling 7: TCC Offset 25 of 63
Cooling 8: Processor 0 of 10
Cooling 9: iwlwifi_1 no state information available
`
```

「design capacity」のmAh値が測定のたびに変動するのは、なぜか？

とりあえず、バッテリが持つようになったので、良しとする。

参照情報

ThinkPad X1 Carbonのマシンタイプの識別
シリアル番号の探し方 - マシンタイプ-モデル（製品番号）、シリアル番号（製造番号）が記載されたラベル
ThinkPad X1 Carbon ハードウェア保守マニュアル
ThinkPad X1 Carbon ユーザー・ガイド
BIOS (Boot Menu) からブートデバイスを選択する - ideapad, ThinkPad, ThinkStation, ThinkCentre, ideacentre
内蔵バッテリーの取り付け、取り外し - Thinkpad x1 Carbon

取り付け、取り外し動画 - ThinkPad X1 Carbon Gen 5 (20HQ, 20HR, 20K3, 20K4),ThinkPad X1 Carbon Gen 6 (20KG, 20KH)

SAMSUNG 1TB (980 MZ-V8V1T0B/IT)
SAMSUNG 500GB (980 MZ-V8V500B/IT)
SSDケース (HDE-13)
NVMe M.2 SSD + 外付けケースは安くて爆速！1GB/sの高速SSDストレージを低価格で！

たぶん解決済みだと思うが参考まで：
ThinkPad X1 Carbon第5世代ノートパソコン無償点検・修理のお知らせ

CrystalDiskMark 代替の KDiskMark(Linux) と AmorphousDiskMark(macOS)
CrazyDiskInfo is an interactive TUI S.M.A.R.T viewer for Unix systems.
→ ubuntu20.04では正常動作しなかった。

「CrystalDiskMark 8」がリリース！自作UIライブラリ「Project Priscilla」を採用
CrystalDiskInfo HDD/SSDの健康状態をチェック

SSD benchmark(@ThinkPad X1 Carbon)

以上

2021/8/8 初版

Switching ADSL to Optical network memo

Switching ADSL to Optical network memo

概要

nifty.comのASDLのサービス終了に伴い光ネットワークに切り替えたが、 設定に試行錯誤が必要になり手間がかかったのでメモとしてまとめた。

工事

工事としては、引き込んでいるADSLのメタルワイヤを光ファイバに切り替え、ADSLモデムをONU(Optical Network Unit)に置き換えて、 屋内に引き込んだ光ファイバをONUに接続する。
ONUには有線LANポートと電話ポートが出ているので ひかり電話を契約している場合は電話ポートに電話機を接続する。
工事業者の仕事としては、電話の動作確認をしたところで 完了となる。

ONU設定

ADSLモデムのときはルーター機能がなかったので 別途、(WiFi)ルーターを接続して、ルーター経由で 接続先の設定を行っていたが、 今回のONUの場合、ルーター機能が内蔵されているので ONUの有線LANポートにPCを接続して設定する。

1. 接続後、PCのブラウザーで「 http://192.168.1.1/ 」に アクセスして、機種設定用パスワードの入力画面で パスワード(任意)を設定する。これはONUの設定を変更する際のパスワードになる。
2. ログイン画面で、ユーザー名をuser、上で設定したパスワードを入力して、ログインする。
3. 設定ウィザード画面で以下を入力する：
   接続先名:「IPv4 PPPoE」　(IPv4を契約している場合)
   接続先ユーザ名: @nifty_ID + @nifty.com
   (例：abc12345@nifty.com)
   接続パスワード: 接続用パスワード

以上でONUの設定としては完了になり、ONUのLANポートにPCなどを接続すると、そのままインターネットに接続できる。

なお、契約としてIPv6を選択した場合、PPPoPを使用しないので、上の設定無しで、インターネットに接続できるらしい。

WiFi設定

接続上位のONUがルーター機能を持っているので、それに接続する(下位の)WiFiルーターはアクセスポイントモード(ルーター機能が無効になる)に切り替えてONUのLANポートに接続する。
アクセスポイントモードへの切り替え方法はWiFiルータごとに異なるので取扱説明書などを参照すること。

参考情報

ホームゲートウェイ/ひかり電話ルータ (RX-600KI)

以上

2021/2/18:
第２版

pio nixie NTP Client v2

pio nixie NTP Client v2

概要

以下のnixieクロックにＮＴＰクライアントの機能を追加する(V2)

Wio nixie tube clock

本記事は「 nixieクロックにＮＴＰクライアントの機能を追加する 」の第２版にあたる。 これは、wio-terminalのWiFiファームウェアのアップグレードに対する対応になる。(wio-terminalのファームウェア・アップデートについて(v2)(linux版))

NTPクライアントの機能を追加しているので、RTCのハードを追加する必要はない。 (ホストPCとしてはubuntuを想定している)

プロジェクト wiot-nixie-NTP のディレクトリを作成する

```
mkdir wiot-nixie-NTP
cd wiot-nixie-NTP
# 以下を実行して必要なファイルを作成する
pio init --board seeed_wio_terminal

# platformをupdateする
pio platform update

nano platformio.ini
以下にように編集する：
```

```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra fla
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and exam
; https://docs.platformio.org/page/projectconf.html

[env:seeed_wio_terminal]
platform = atmelsam
board = seeed_wio_terminal
framework = arduino
build_flags = -DWIO_TERMINAL
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    https://github.com/Seeed-Studio/Seeed_Arduino_mbedtls/a
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcUnifie
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcBLE/ar
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcWiFi/a
    https://github.com/Seeed-Studio/Seeed_Arduino_FreeRTOS/
    https://github.com/Seeed-Studio/Seeed_Arduino_FS/archiv
    https://github.com/Seeed-Studio/Seeed_Arduino_SFUD/arch
    #
    https://github.com/Seeed-Studio/Seeed_Arduino_LCD/archi
    # 551
    #https://github.com/arduino-libraries/NTPClient.git
    arduino-libraries/NTPClient@^3.1.0
```

該当スケッチのダウンロード

```
cd wiot-nixie-NTP

wget https://macsbug.files.wordpress.com/2020/05/wio_nixie_
mv wio_nixie_tube_clock.zip_-4.pdf wio_nixie_tube_clock.zip
unzip wio_nixie_tube_clock.zip
cp Wio_nixie_tube_clock/*.* src/
```

デモ・スケッチ

上でダウンロートしたスケッチにNTPクライアント機能を追加して以下のようなソースに編集する：

src/Wio_nixie_tube_clock_NTP.ino

```
// NTP Client part added on 2020/08/01 by xshige
//#include <AtWiFi.h>
#include <rpcWiFi.h>

#include <NTPClient.h>
#include <WiFiUdp.h>

const char *ssid     = "your_ssid";
const char *password = "your_passwd";

//<-----------------------------------------------------------
// derived/forked from http://mrkk.ciao.jp/memorandom/unixt
#define ARRAYSIZE(_arr) (sizeof(_arr) / sizeof(_arr[0]))
#define TIME_OFFSET 0
#define SECONDS_IN_A_DAY    (24*60*60)
#define EPOCH_DAY       (1969*365L + 1969/4 - 1969/100 + 19
#define UNIX_EPOCH_DAY  EPOCH_DAY
#define YEAR_ONE        365
#define YEAR_FOUR       (YEAR_ONE * 4 + 1)  //  it is YEAR_
#define YEAR_100        (YEAR_FOUR * 25 - 1)
#define YEAR_400        (YEAR_100*4 + 1)    //  it is YEAR_

void ConvertUnixTimeToLocalTime(uint64_t unixtime, uint32_t
    uint32_t unixday;
    uint16_t year = 0;
    uint8_t  leap = 0;
    uint32_t n;
    uint8_t month, day, weekday;
    uint8_t hour, minute, second;
    static const uint16_t monthday[] = { 0,31,61,92,122,153

    unixtime += TIME_OFFSET;
    second = unixtime % 60;
    minute = (unixtime / 60) % 60;
    hour = (unixtime / 3600) % 24;

    unixday = (uint32_t)(unixtime / SECONDS_IN_A_DAY);
    weekday = (uint8_t)((unixday + 3) % 7); //  because the
    unixday += UNIX_EPOCH_DAY;  //  days from 0000/03/01 to

    year += 400 * (unixday / YEAR_400);
    unixday %= YEAR_400;

    n = unixday / YEAR_100;
    year += n * 100;
    unixday %= YEAR_100;
```

```
    if (n == 4){
        leap = 1;
    } else {
        year += 4 * (unixday / YEAR_FOUR);
        unixday %= YEAR_FOUR;

        n = unixday / YEAR_ONE;
        year += n;
        unixday %= YEAR_ONE;
        if (n == 4) {
            leap = 1;
        }
    }
    if (leap != 0) {
        month = 2;
        day = 29;
    }
    else {
        month = (unixday * 5 + 2) / 153;
        day = unixday - monthday[month] + 1;    //
        month += 3;
        if (month > 12) {
            ++year;
            month -= 12;
        }
    }
    *psecond = second;
    *pminute = minute;
    *phour = hour;
    *pyear = year;
    *pmonth = month;
    *pday = day;
}
//>------------------------------------------------------------

char *weekday[] = {"Sun","Mon","Tue","Wed","Thu","Fri","Sat
uint32_t year;
uint8_t month, day, hour, minu, sec;

WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, "ntp.nict.jp", 3600*9, 60000);
// You can specify the time server pool and the offset, (in
// additionaly you can specify the update interval (in mill
// NTPClient timeClient(ntpUDP, "europe.pool.ntp.org", 3600

//============================================================
// added NTP Client feature: 2020.08.01 xshige
// Woi Terminal nixie tube clock : 2020.05.25 macsbug
```

```
//  https://macsbug.wordpress.com/2020/05/25/wio-nixie-tube
// M5Stack nixie tube clock      : 2019.06.16 macsbug
//  https://macsbug.wordpress.com/2019/06/16/m5stack-nixie-
// M5StickC Nixie tube Clock     : 2019.06.06 macsbug
//  https://macsbug.wordpress.com/2019/06/06/m5stickc-nixie
// RTC DS3231 : https://wiki.52pi.com/index.php/Raspberry_P
// RTClib : https://www.arduinolibraries.info/libraries/rt-
// mode controll : 5 way switch
// mode 1 : yyyy_mmdd_hhmmss
// mode 2 : mmdd_hh_mmss
// mode 3 : mmdd_ss_hhmm
// rtc    : DS3231, SDA1 = 2, SCL1= 3;
#include <SPI.h>
#include <TFT_eSPI.h>
TFT_eSPI tft = TFT_eSPI();
#include <Wire.h>
//#include "RTClib.h"
//RTC_DS3231 rtc;
#include "vfd_18x34.c"   // font 18px34
#include "vfd_35x67.c"   // font 35x67
#include "vfd_70x134.c"  // font 70px134
#include "apple_35x41.c" // icon 35px41
uint32_t targetTime = 0; // for next 1 second timeout
const uint8_t*n[] = { // vfd font 18x34
  vfd_18x34_0,vfd_18x34_1,vfd_18x34_2,vfd_18x34_3,vfd_18x34
  vfd_18x34_5,vfd_18x34_6,vfd_18x34_7,vfd_18x34_8,vfd_18x34
const uint8_t*m[] = { // vfd font 35x67
  vfd_35x67_0,vfd_35x67_1,vfd_35x67_2,vfd_35x67_3,vfd_35x67
  vfd_35x67_5,vfd_35x67_6,vfd_35x67_7,vfd_35x67_8,vfd_35x67
  vfd_35x67_q,vfd_35x67_n };
const uint8_t*b[] = { // vfd font 70x134
  vfd_70x134_0,vfd_70x134_1,vfd_70x134_2,vfd_70x134_3,vfd_7
  vfd_70x134_5,vfd_70x134_6,vfd_70x134_7,vfd_70x134_8,vfd_7
  vfd_70x134_q,vfd_70x134_n };
const char *monthName[12] = {
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };
uint16_t yy;
uint8_t mn, dd, hh, mm, ss;
uint8_t md = 2; // mode 1, 2, 3

void setup() {
//
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while ( WiFi.status() != WL_CONNECTED ) {
    delay ( 500 );
```

```
    Serial.print ( "." );
  }
  timeClient.begin();
//---------------------
  tft.init();
  tft.setRotation(3);
  tft.fillScreen(TFT_BLACK);
  pinMode(WIO_5S_PRESS, INPUT_PULLUP);
/****
  Wire.begin(SDA1,SCL1); delay(10);
  rtc.begin();
  rtc.adjust(DateTime(__DATE__, __TIME__)); // Set the PC t
  // esp_timer_init();
  // wifi_setup();
  rtc_setup();  // Read the value of RTC
****/
}

void loop() {
  timeClient.update();
  // if (targetTime < esp_timer_get_time()/1000 ){

  // display current clock on serial port
  ConvertUnixTimeToLocalTime(timeClient.getEpochTime(),
     &year, &month, &day, &hour, &minu, &sec);
  Serial.printf("%4d/%02d/%02d(%s): %02d:%02d:%02d\r\n",
     year, month, day, weekday[timeClient.getDay()], hour,

/****
  DateTime now = rtc.now();
  yy = now.year();
  mn = now.month();
  dd = now.day();
  hh = now.hour();
  mm = now.minute();
  ss = now.second();
****/
  yy = year;
  mn = month;
  dd = day;
  hh = hour;
  mm = minu;
  ss = sec;

  if(digitalRead(WIO_5S_PRESS) == LOW){         // mode cha
   if (md == 3){md = 1;tft.fillRect(1,1,317,236,TFT_BLACK);
   if (md == 2){md = 3;tft.fillRect(1,1,317,236,TFT_BLACK);
   if (md == 1){md = 2;tft.fillRect(1,1,317,236,TFT_BLACK);
```

```
  }
  if ( md == 3 ){ hhmm();}              // yyyy,mm,dd,ss,hh,
  if ( md == 2 ){ yyyy_mmdd_hhmmss();} // yyyy,mm,dd,hh,mm,
  if ( md == 1 ){ mmss();}              // mm,ss
  // periodic_ntp();
  delay(500);
}

/****
void rtc_setup(){
  DateTime now = rtc.now(); // time geting from RTC
  if (now.year() == 2165){  // rtc check
    tft.setCursor(20,200);tft.setTextColor(TFT_RED);
    tft.print("RTC not installed");delay(2000);
    tft.fillRect(1, 1, 317, 236, TFT_BLACK);
    }else{
    yy = now.year(); mn = now.month(); dd = now.day();
    hh = now.hour(); mm = now.minute(); ss = now.second();
    //Serial.printf("%d %d %d %d %d %d\n",yy,mn,dd,hh,mm,ss
    tft.setCursor(20,200);tft.setTextColor(TFT_BLUE);
    tft.print("SET UP RTC");delay(2000);
    tft.fillRect(1, 1, 317, 236, TFT_BLACK);
  }
}
****/

void yyyy_mmdd_hhmmss(){
  int y1 = (yy / 1000) % 10; int y2 = (yy / 100) % 10;
  int y3 = (yy /   10) % 10; int y4 = yy % 10;
  int ma = (mn /   10) % 10; int mb = mn % 10;
  int d1 = (dd /   10) % 10; int d2 = dd % 10;
  int h1 = (hh /   10) % 10; int h2 = hh % 10;
  int m1 = (mm /   10) % 10; int m2 = mm % 10;
  int s1 = (ss /   10) % 10; int s2 = ss % 10;

  //int p0 = 8; int x0 = 40; int t0 = 22; // icon
  //tft.pushImage( p0 + 0*x0, t0, 35,41, (uint16_t *)apple_

  int p1 = 80; int px1 = 40; int py1 = 5;
  tft.pushImage( p1 + 0*px1, py1, 35,67, (uint16_t *)m[y1])
  tft.pushImage( p1 + 1*px1, py1, 35,67, (uint16_t *)m[y2])
  tft.pushImage( p1 + 2*px1, py1, 35,67, (uint16_t *)m[y3])
  tft.pushImage( p1 + 3*px1, py1, 35,67, (uint16_t *)m[y4])

  int p2 = 80; int px2 = 40; int py2 = 76;
  tft.pushImage( p2 + 0*px2, py2, 35,67, (uint16_t *)m[ma])
  tft.pushImage( p2 + 1*px2, py2, 35,67, (uint16_t *)m[mb])
  //tft.drawPixel(118,13, ORANGE); tft.drawPixel(119,23,ORA
```

```
  tft.pushImage( p2 + 2*px2, py2, 35,67, (uint16_t *)m[d1])
  tft.pushImage( p2 + 3*px2, py2, 35,67, (uint16_t *)m[d2])

  int p3 = 2; int px3 = 40; int py3 = 150;
  tft.pushImage( p3 + 0*px3, py3, 35,67, (uint16_t *)m[h1])
  tft.pushImage( p3 + 1*px3, py3, 35,67, (uint16_t *)m[h2])
  tft.pushImage( p3 + 2*px3, py3, 35,67, (uint16_t *)m[10])
  tft.pushImage( p3 + 3*px3, py3, 35,67, (uint16_t *)m[m1])
  tft.pushImage( p3 + 4*px3, py3, 35,67, (uint16_t *)m[m2])
  tft.pushImage( p3 + 5*px3, py3, 35,67, (uint16_t *)m[10])
  tft.pushImage( p3 + 6*px3, py3, 35,67, (uint16_t *)m[s1])
  tft.pushImage( p3 + 7*px3, py3, 35,67, (uint16_t *)m[s2])

  if ( s1 == 0 && s2 == 0 ){ fade1();}
}

void mmss(){
  int ma = (mn / 10) % 10; int mb = mn % 10;
  int d1 = (dd / 10) % 10; int d2 = dd % 10;
  int h1 = (hh / 10) % 10; int h2 = hh % 10;
  int m1 = (mm / 10) % 10; int m2 = mm % 10;
  int s1 = (ss / 10) % 10; int s2 = ss % 10;

  int p0 = 8; int x0 = 40; int t0 = 22; // icon
  tft.pushImage( p0 + 0*x0, t0, 35,41, (uint16_t *)apple_35

  int p2 = 65; int px2 = 40; int py2 = 10;
  tft.pushImage( p2 + 0*px2, py2, 35,67, (uint16_t *)m[ma])
  tft.pushImage( p2 + 1*px2, py2, 35,67, (uint16_t *)m[mb])
  //tft.drawPixel(118,13, ORANGE); tft.drawPixel(119,23,ORA
  tft.pushImage( p2 + 2*px2, py2, 35,67, (uint16_t *)m[d1])
  tft.pushImage( p2 + 3*px2, py2, 35,67, (uint16_t *)m[d2])

  int p3 = 240; int px3 = 40; int py3 = 10;
  tft.pushImage( p3 + 0*px3, py3, 35,67, (uint16_t *)m[h1])
  tft.pushImage( p3 + 1*px3, py3, 35,67, (uint16_t *)m[h2])

  int p4 = 2; int px4 = 80; int py4 = 100;
  tft.pushImage( p4 + 0*px4   , py4, 70,134, (uint16_t *)b[
  tft.pushImage( p4 + 1*px4 -4, py4, 70,134, (uint16_t *)b[
  //tft.drawPixel(155,150, ORANGE); tft.drawPixel(155,190,C
  tft.fillCircle(156,151,3,TFT_ORANGE);tft.fillCircle(156,1
  tft.fillCircle(156,151,1,TFT_YELLOW);tft.fillCircle(156,1
  tft.pushImage( p4 + 2*px4 +4, py4, 70,134, (uint16_t *)b[
  tft.pushImage( p4 + 3*px4   , py4, 70,134, (uint16_t *)b[

  if ( m1 == 0 && m2 == 0 ){ fade2();}
}
```

```
void hhmm(){
  int ma = (mn / 10) % 10; int mb = mn % 10;
  int d1 = (dd / 10) % 10; int d2 = dd % 10;
  int h1 = (hh / 10) % 10; int h2 = hh % 10;
  int m1 = (mm / 10) % 10; int m2 = mm % 10;
  int s1 = (ss / 10) % 10; int s2 = ss % 10;

  //int p0 = 8; int x0 = 40; int t0 = 22; // icon
  //tft.pushImage( p0 + 0*x0, t0, 35,41, (uint16_t *)apple_

  int p2 = 65; int px2 = 40; int py2 = 10;
  tft.pushImage( p2 + 0*px2, py2, 35,67, (uint16_t *)m[ma])
  tft.pushImage( p2 + 1*px2, py2, 35,67, (uint16_t *)m[mb])
  //tft.drawPixel(118,13, ORANGE); tft.drawPixel(119,23,ORA
  tft.pushImage( p2 + 2*px2, py2, 35,67, (uint16_t *)m[d1])
  tft.pushImage( p2 + 3*px2, py2, 35,67, (uint16_t *)m[d2])

  int p3 = 240; int px3 = 40; int py3 = 10;
  tft.pushImage( p3 + 0*px3, py3, 35,67, (uint16_t *)m[s1])
  tft.pushImage( p3 + 1*px3, py3, 35,67, (uint16_t *)m[s2])

  int p4 = 2; int px4 = 80; int py4 = 100;
  tft.pushImage( p4 + 0*px4   , py4, 70,134, (uint16_t *)b[
  tft.pushImage( p4 + 1*px4 -4, py4, 70,134, (uint16_t *)b[
  //tft.drawPixel( 155,150, ORANGE); tft.drawPixel(155,190,
  tft.fillCircle(156,151,3,TFT_ORANGE);tft.fillCircle(156,1
  tft.fillCircle(156,151,1,TFT_YELLOW);tft.fillCircle(156,1
  tft.pushImage( p4 + 2*px4 +4, py4, 70,134, (uint16_t *)b[
  tft.pushImage( p4 + 3*px4   , py4, 70,134, (uint16_t *)b[

  if ( h1 == 0 && h2 == 0 ){ fade2();}
}

void fade1(){
  int p3 = 2; int px3 = 40; int py3 = 150;
  for ( int i = 0; i < 2; i++ ){
  tft.pushImage( p3 + 2*px3, py3, 35,67, (uint16_t *)m[11])
  tft.pushImage( p3 + 5*px3, py3, 35,67, (uint16_t *)m[11])
  delay(25);
  tft.pushImage( p3 + 2*px3, py3, 35,67, (uint16_t *)m[10])
  tft.pushImage( p3 + 5*px3, py3, 35,67, (uint16_t *)m[10])
  delay(25);
  }
}

void fade2(){
  int p3 = 2; int px3 = 40; int py3 = 150;
```

```
  for ( int i = 0; i < 2; i++ ){
  tft.fillCircle(156,151,3,TFT_BLACK);tft.fillCircle(156,19
  delay(25);
  tft.fillCircle(156,151,3,TFT_ORANGE);tft.fillCircle(156,1
  delay(25);
  }
}
```

以下の変更がWiFiファームウェア・アップグレードの対応にあたる：

```
//#include <AtWiFi.h>
#include <rpcWiFi.h>
```

以下については、自分の環境に合わせて変更すること：

```
const char *ssid     = "your_ssid";
const char *password = "your_passwd";
```

書き込み後に「picocom /dev/ttyACM0 -b115200」で通信ソフトを起動すると以下のような出力が表示される：

```
$ picocom /dev/ttyACM0 -b115200

Terminal ready
2020/08/01(Sat): 22:44:44
2020/08/01(Sat): 22:44:44
2020/08/01(Sat): 22:44:45
2020/08/01(Sat): 22:44:46
2020/08/01(Sat): 22:44:46
2020/08/01(Sat): 22:44:47
2020/08/01(Sat): 22:44:48
2020/08/01(Sat): 22:44:48
2020/08/01(Sat): 22:44:49
2020/08/01(Sat): 22:44:49
2020/08/01(Sat): 22:44:50
2020/08/01(Sat): 22:44:50
2020/08/01(Sat): 22:44:51
2020/08/01(Sat): 22:44:51
2020/08/01(Sat): 22:44:52
2020/08/01(Sat): 22:44:53
2020/08/01(Sat): 22:44:53
2020/08/01(Sat): 22:44:54
```

また、起動後、ニキシー管の時計が表示され、ボタンで表示画面を切り換えられる。

参考情報

Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)

PlatformIO Core (CLI)

以上

2022/1/30
初版

LovyanGFX sketch build by platformio

LovyanGFX sketch build by platformio

概要

「LovyanGFX - Display (LCD / OLED / EPD) graphics library」ライブラリを使用したスケッチを platformioでビルドする。
arduinoのサンプル・スケッチとして提供されているものだが、
plaformioビルド環境では、エラーを出てビルドできないことがある。
それを解決したplatformio.iniを提供する。
対応ボードは、M5Core2,M5Stack-Fire,Wio-Terminalとなる。
スケッチのソースは共通で、platformio.iniの内容で、対応ボードを切り替えることになる。

arduino_misaki.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir arduino_misaki
cd arduino_misaki

mkdir src
cd src
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m

cd ..

# platformioの設定ファイルをダウンロードする(M5Stack系のみ)
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

efont.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir efont
cd efont

mkdir src
cd src
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m

cd ..

# platformioの設定ファイルをダウンロードする(M5Stack系のみ)
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

LongTextScroll.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir LongTextScroll
cd LongTextScroll

mkdir src
cd src
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m

cd ..

# platformioの設定ファイルをダウンロードする(M5Stack系のみ)
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

5_images.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir 5_images
cd 5_images

mkdir src
cd src
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m

cd ..

# platformioの設定ファイルをダウンロードする(M5Stack系のみ)
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

ClockSample.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir ClockSample
cd ClockSample

mkdir src
cd src
wget https://raw.githubusercontent.com/lovyan03/LovyanGFX/m

gedit ClockSample.ino
# 以下のようにパッチをかける
修正前：
#define LGFX_USE_V1
#include <LovyanGFX.hpp>
→
修正後：
#define LGFX_AUTODETECT
//#define LGFX_USE_V1
#include <LovyanGFX.hpp>


cd ..

# platformioの設定ファイルをダウンロードする(M5Stack系のみ)
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

platformio.ini

上のスケッチをビルドするには、ボードに応じて、
以下のplatformio.iniを使用する：

M5Core2用：

```
[env:m5core2]
platform = espressif32
board = m5stack-core2
framework = arduino
upload_speed = 2000000
monitor_speed = 115200
board_build.partitions = default_16MB.csv

build_flags =
    -DM5C2
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    m5stack/M5Core2@^0.1.0
    ;https://github.com/m5stack/M5Core2.git
    https://github.com/FastLED/FastLED
    ;https://github.com/m5stack/M5Core2/blob/master/example
    ;tobozo/ESP32-Chimera-Core@^1.4.0
    lovyan03/LovyanGFX@^0.4.13
    https://github.com/tanakamasayuki/efont.git
    https://github.com/Tamakichi/Arduino-misakiUTF16.git
    #
    adafruit/Adafruit BMP280 Library @ ^2.5.0
    adafruit/Adafruit Unified Sensor @ ^1.1.4
    m5stack/UNIT_ENV @ ^0.0.2
    https://github.com/earlephilhower/ESP8266Audio.git
    tobozo/M5Stack-SD-Updater@^1.1.8

;lib_ldf_mode = deep+
lib_ldf_mode = deep
```

重要：
通常の設定と異なり「lib_ldf_mode = deep」(末尾に「+」が無い)にしないとビルドエラーになる。

M5Stack-Fire用：

```
[env:m5stack-fire]
platform = espressif32
board = m5stack-fire
framework = arduino
monitor_speed = 115200

upload_speed = 2000000
upload_protocol = esptool
board_build.partitions = default_16MB.csv

build_flags =
    -DM5STACK_FIRE
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    m5stack/M5Stack@^0.3.9
    ;tobozo/ESP32-Chimera-Core@^1.4.0
    lovyan03/LovyanGFX@^0.4.13
    https://github.com/tanakamasayuki/efont.git
    https://github.com/Tamakichi/Arduino-misakiUTF16.git
    #
    adafruit/Adafruit BMP280 Library @ ^2.5.0
    adafruit/Adafruit Unified Sensor @ ^1.1.4
    m5stack/UNIT_ENV @ ^0.0.2
    https://github.com/earlephilhower/ESP8266Audio.git
    tobozo/M5Stack-SD-Updater@^1.1.8

;lib_ldf_mode = deep+
lib_ldf_mode = deep
```

重要：
通常の設定と異なり「lib_ldf_mode = deep」(末尾に「+」が無い)にしないとビルドエラーになる。

Wio-Terminal用：

```
[env:seeed_wio_terminal]
platform = atmelsam
board = seeed_wio_terminal
framework = arduino
build_flags = -DWIO_TERMINAL
upload_protocol = sam-ba
monitor_speed = 115200

lib_deps =
    https://github.com/Seeed-Studio/Seeed_Arduino_mbedtls/a
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcUnifie
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcBLE/ar
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcWiFi/a
    https://github.com/Seeed-Studio/Seeed_Arduino_FreeRTOS/
    https://github.com/Seeed-Studio/Seeed_Arduino_FS/archiv
    https://github.com/Seeed-Studio/Seeed_Arduino_SFUD/arch
    #
    # RTC
    adafruit/RTClib@^2.0.2
    ;neironx/RTCLib@^1.6.0
    ;powerbroker2/SafeString@^4.1.14
    ;seeed-studio/Seeed Arduino RTC@^2.0.0
    #
    # display libs (* Select Only ONE *)
    ;https://github.com/Seeed-Studio/Seeed_Arduino_LCD/arch
    ;moononournation/GFX Library for Arduino@^1.1.8
    lovyan03/LovyanGFX@^0.4.13
    https://github.com/tanakamasayuki/efont.git
    https://github.com/Tamakichi/Arduino-misakiUTF16.git
    #
    # network related
    ;arduino-libraries/NTPClient@^3.1.0
    ;knolleary/PubSubClient@^2.8
    #
    # sensor related
    adafruit/Adafruit Unified Sensor@^1.1.4
    adafruit/Adafruit BME280 Library@^2.2.2
    adafruit/Adafruit DPS310@^1.1.1
    ;wemos/WEMOS SHT3x@^1.0.0
    https://github.com/Seeed-Studio/Grove_SHT31_Temp_Humi_S
    #

lib_ldf_mode = deep+
```

参照情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

LovyanGFX関連：
LovyanGFX - Display (LCD / OLED / EPD) graphics library (for ESP32 SPI, I2C, 8bitParallel / ESP8266 SPI, I2C / ATSAMD51 SPI) - M5Stack / M5StickC / TTGO T-Watch / ODROID-GO / ESP-WROVER-KIT / WioTerminal / and more...
LovyanGFX LCD Graphics driver

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

M5Core2関連：
M5Stack Core2を使ってみました。

M5Stackファミリ関連：
M5Core2 Arduino Install
M5Paper Arduino Install
M5CoreInk Arduino Install
M5Stamp-PICO Arduino Install
M5Stamp-C3 Arduino Install
Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(STARWARS編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(MQTT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(REST-API編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(OSC編)

以上

2022/1/1
初版

M5CAMERA CameraWebServer

M5CAMERA CameraWebServer

概要

「M5Camera」でCameraWebServerを動かす。
Arduinoのスケッチとしては、提供されているサンプルそのものものになるが、パッチが必要なので、それについて説明する。
また、ビルド環境としては、platformioを使用する。

ソース入手

以下の手順でソースを入手する：

```
cd ~/Downloads
git clone https://github.com/espressif/arduino-esp32.git

# platformioの環境に入る
mkdir CameraWebSerVer
cd CameraWebSerVer
mkdir src

cd src
cp ~/Downloads/arduino-esp32/libraries/ESP32/examples/Camer

cd ..
# platformioのビルドに必要な設定ファイルを入手する
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 次節のplatformio.iniの内容を作成する：
```

パッチ

src/CameraWebServer.ino

```
#10行あたりを以下のように変更する：
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM

#23行あたりを自分のWiFi環境に合わせて変更する：
const char* ssid = "*********";
const char* password = "*********";
```

src/camera_pins.h

```
#44行あたりを以下のように変更する：
#define SIOD_GPIO_NUM     22 //25

#55行あたりを以下のように変更する：
#define VSYNC_GPIO_NUM    25 //22
```

platformio.ini

M5Camera用

```
[env:esp-wrover-kit]
platform = espressif32
board = esp-wrover-kit
framework = arduino
upload_speed = 921600
monitor_speed = 115200
board_build.partitions = huge_app.csv
board_build.f_flash= 80000000L
board_build.flash_mode = qio

build_flags =
    -DM5CAMERA
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    m5stack/Timer-CAM@^0.0.1

lib_ldf_mode = deep+
```

動作

シリアル出力にCameraWebServerのIPアドレスが出力されるので、そのIPアドレスでWebブラウザーからアクセスする。

参考情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

M5Camera関連：
M5Cameraのサンプルコードを実行してみる(w/ M5Camera datasheet)
ESP32 Camera Demo
M5Stack社のカメラの選び方(2021年1月)
M5Cameraの使い方
M5Camera をレビューしてみた。分解したり、Arduino IDE でスマホに映したりする実験
M5Camera のHTTP stream を 動画としてキャプチャ

platformio関連：
Support ESP32 Wrover Module?
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

M5Stackファミリ関連：
M5Core2 Arduino Install
M5Paper Arduino Install
M5CoreInk Arduino Install
M5Stamp-PICO Arduino Install
M5Stamp-C3 Arduino Install
Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(STARWARS編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(MQTT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動か

す(v2)(REST-API編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(OSC編)

以上

2022/1/29
誤記修正：

```
修正前：
wget https://github.com/espressif/arduino-esp32/blob/master

→

修正後：
wget https://raw.githubusercontent.com/espressif/arduino-es
```

2021/12/22
初版

M5Core2 Arduino Install

M5Core2 Arduino Install

概要

platformioベースのArduinoフレームワークを使って
「M5Stack Core2 IoT開発キット」を動かす。
ホスト環境は、ubuntu20.04とする。
なお、platformioがインストール済みのものとする。

linuxのplatformioのインストールについては以下を参照のこと。
・arduinoフレームワーク用platformio.ini集

電源操作：
・電源オン：ボタン(左側面)のを押す。
・電源オフ：6秒間ボタン(左側面)を長押しする。
・リスタート：底にあるリセットボタンを押す。

M5Core2_FactoryTestの書き込み方法

まず、FactoryTestの書き込み方法について説明する。
以下を実行する：

```
# ソースをダウンロードする
cd Downloads/
git clone https://github.com/m5stack/M5-ProductExampleCodes

cd pio
# platformio環境に入る
source pio_env/bin/activate

mkdir core2
cd core2
mkdir FactoryTest
cd FactoryTest

# default_16MB.csvをダウンロードする
wget https://raw.githubusercontent.com/espressif/arduino-es

# ソースをコピーする
cp -r ~/Downloads/M5-ProductExampleCodes/Core/M5Core2/Ardui

gedit platformio.ini
# 次節の内容のplatformio.iniを作成する

# ビルド環境のクリーンアップ
pio -t clean

# M5CoreとPCをUSBで接続する
# ビルド&書き込む
pio -t upload
```

以上で書き込みが完了する。

なお、シリアルデバイスは通常「/dev/ttyACM0」になる。

platformio.ini

M5Core2のスケッチのビルドには
必要なライブラリの追加はあり得るが
基本的には以下のplatformio.iniを使用する。
platformio.ini

```
[env:m5core2]
platform = espressif32
board = m5stack-core2
framework = arduino
upload_speed = 2000000
monitor_speed = 115200
board_build.partitions = default_16MB.csv

build_flags =
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    https://github.com/m5stack/M5Core2.git
    https://github.com/FastLED/FastLED
    ;https://github.com/m5stack/M5Core2/blob/master/example
    lovyan03/LovyanGFX
    adafruit/Adafruit BME280 Library@^2.1.2
    adafruit/Adafruit Unified Sensor@^1.1
    m5stack/UNIT_ENV @ ^0.0.2

lib_ldf_mode = deep+
```

新たなプロジェクトをビルドしたい場合、 FactoryTestのディレクトリを、 まるごとコピーして、ディレクトリ名と src配下のスケッチを変更すれば良い。

sketch#1(UNIT_ENV_III)

別のサンプルとして「M5Stack用温湿度気圧センサユニット Ver.3（ENV Ⅲ）」をPORT-Aに接続して温度、湿度、気圧をシリアルに出力するスケッチを動かす。

src/UNIT_ENV_III.ino

```
#include "Wire.h"
#include "UNIT_ENV.h"

SHT3X sht30;
QMP6988 qmp6988;

void setup() {
  Serial.begin(115200);
  Wire.begin(32, 33); // Core2 PORT-A
  qmp6988.init();
}

void loop() {
  if (sht30.get() != 0) {
    return;
  }
  Serial.printf("Temperature: %2.2f℃  Humidity: %0.2f%%  P
  delay(1);//000);
}
```

M5Stack系ボードで汎用的に使用できるようにシリアル出力のみにしている。 たぶん、依存している部分は「Wire.begin(32, 33);」と思われる。

platformio.iniは上のものが、そのまま使用できる。
(必要なライブラリは登録済み)

sketch#2(b3NTP)

b3NTP.ino

```
// select board
////#define WIO_TERMINAL
////#define ESP8266
////#define ESP32
////#define M5ATOM
//------------------

// NTP Client for Wio-Terminal/ESP8266/ESP32
#ifdef M5ATOM
#include "M5Atom.h"
#define ESP32
#endif
#ifdef WIO_TERMAL
//#include <AtWiFi.h>
#include <rpcWiFi.h>
#endif
#ifdef ESP8266
#include <ESP8266WiFi.h>
#endif
#ifdef ESP32
#include <WiFi.h>
#endif

#include <time.h>

#define WIFI_SSID   "your_wifi_ssid"
#define WIFI_PASSWORD   "your_wifi_password"

void setup() {
  Serial.begin(115200);
  delay(100);
  Serial.print("\r\n\nReset:\r\n");

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  while(WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(500);
  }
  Serial.println();
  Serial.printf("Connected, IP address: ");
  Serial.println(WiFi.localIP());

  configTzTime("JST-9", "ntp.nict.jp", "ntp.jst.mfeed.ad.jp
}

void loop() {
  time_t t;
```

```
  struct tm *tm;
  static const char *wd[7] = {"Sun","Mon","Tue","Wed","Thr'

  t = time(NULL);
  tm = localtime(&t);
/*****
  Serial.printf("ESP8266/Arduino ver%s :  %04d/%02d/%02d(%s
        __STR(ARDUINO_ESP8266_GIT_DESC),
        tm->tm_year+1900, tm->tm_mon+1, tm->tm_mday,
        wd[tm->tm_wday],
        tm->tm_hour, tm->tm_min, tm->tm_sec);
****/
  Serial.printf("Arduino NTP:  %04d/%02d/%02d(%s) %02d:%02d
        tm->tm_year+1900, tm->tm_mon+1, tm->tm_mday,
        wd[tm->tm_wday],
        tm->tm_hour, tm->tm_min, tm->tm_sec);
  delay(1000);
}
```

以下はwifi環境に応じて変更すること：

```
#define WIFI_SSID   "your_wifi_ssid"
#define WIFI_PASSWORD   "your_wifi_password"
```

本ソースは、WIO_TERMINAL、ESP8266、ESP32、M5ATOMで共通になっており、platformio.iniの内容でターゲットを切り替えることができる。

platformio.iniは、上のものが、そのまま使用できる。

実行時のシリアル出力(/dev/ttyACM0)：

```
........
Connected, IP address: 192.168.1.15
Arduino NTP:  1970/01/01(Thr) 09:00:04
Arduino NTP:  1970/01/01(Thr) 09:00:05
Arduino NTP:  1970/01/01(Thr) 09:00:06
Arduino NTP:  1970/01/01(Thr) 09:00:07
Arduino NTP:  1970/01/01(Thr) 09:00:08
Arduino NTP:  2021/12/11(Sat) 16:55:59 ← ここでNTPによって時刻
Arduino NTP:  2021/12/11(Sat) 16:56:00
Arduino NTP:  2021/12/11(Sat) 16:56:01
```

参考情報

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

M5Stackファミリ関連：
M5Paper Arduino Install
M5CoreInk Arduino Install
M5Stamp-PICO Arduino Install
M5Stamp-C3 Arduino Install
Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(STARWARS編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(MQTT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(REST-API編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(OSC編)

その他：
スルーホール用テストワイヤ　ＴＴ－２００　（１０本入）

以上

2022/1/24
初版

WiFiAnalyzer for WioT/M5Core2

WiFiAnalyzer for WioT/M5Core2

概要

Wio-Terminal用とM5Core2用のWiFiAnallyzerを紹介する。
Arduinoのスケッチになっているが、platformioでビルドする。

Wio-Terminal用

以下の手順でビルト＆書き込みする：

```
# platformio環境に入る

mkdir WioWiFiAnalyzer
cd WioWiFiAnalyzer
mkdir src
cd src
#  ソースをダウンロードする
wget https://raw.githubusercontent.com/moononournation/Ardu

cd ..

gedit platormio.ini
# 次節のplatformio.iniの内容を作成する
```

platformio.ini(wio-terminal用)

Wio-Terminal用

```
[env:seeed_wio_terminal]
platform = atmelsam
board = seeed_wio_terminal
framework = arduino
;build_flags = -DWIO_TERMINAL
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    https://github.com/Seeed-Studio/Seeed_Arduino_mbedtls/a
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcUnifie
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcBLE/ar
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcWiFi/a
    https://github.com/Seeed-Studio/Seeed_Arduino_FreeRTOS/
    https://github.com/Seeed-Studio/Seeed_Arduino_FS/archiv
    https://github.com/Seeed-Studio/Seeed_Arduino_SFUD/arch
    #
    # RTC
    adafruit/RTClib@^2.0.2
    ;powerbroker2/SafeString@^4.1.14
    ;seeed-studio/Seeed Arduino RTC@^2.0.0
    #
    # display libs (* Select Only ONE *)
    ;https://github.com/Seeed-Studio/Seeed_Arduino_LCD/arch
    moononournation/GFX Library for Arduino@^1.1.8
    ;lovyan03/LovyanGFX@^0.4.10
    https://github.com/tanakamasayuki/efont.git
    https://github.com/Tamakichi/Arduino-misakiUTF16.git
    #
    # network related
    ;arduino-libraries/NTPClient@^3.1.0
    ;knolleary/PubSubClient@^2.8
    #
    # sensor related
    adafruit/Adafruit Unified Sensor@^1.1.4
    adafruit/Adafruit BME280 Library@^2.2.2
    adafruit/Adafruit DPS310@^1.1.1
    ;wemos/WEMOS SHT3x@^1.0.0
    https://github.com/Seeed-Studio/Grove_SHT31_Temp_Humi_S
    #
```

注意：
display系のライブラリが複数登録されているとビルトでエラーが出なくても正常に実行されない現象があった。

ヘッダーが異なっているので問題がないと思っていたが、必要なライブラリ以外はコメントアウトする必要がるようだ。
ダウンロードした(ローカルな)ライブラリをいったんクリアするためには「sudo rm -r .pio」を実行する。
その後、「pio run -t upload」でビルト＆書き込みを行なう。

M5Core2用

以下の手順でビルド＆書き込みを行なう：

```
# platformio環境に入る

mkdir Core2WiFiAnalyzer
cd Core2WiFiAnalyzer
mkdir src
cd src

#  ソースをダウンロードする
gedit main.ino
以下のurlからソースコードをコピー＆ペーストする：
https://macsbug.wordpress.com/2018/04/30/m5stack-wifianalyz
M5STACK WiFiAnalyzer

cd ..
# platformioの設定ファイルをダウンロードする
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platormio.ini
# 次節のplatformio.iniの内容を作成する
```

パッチ

M5Stack用をM5Core2用に変更するためにソースを以下のように変更する。

src/main.ino

```
#9行当たりを以下のように変更する：(M5StackUpdaterは使用しない)
//#include <M5Stack.h>
#include <M5Core2.h>
//#include "M5StackUpdater.h"                          // S

#27行当たりをコメントアウトする：(M5StackUpdaterは使用しない)
//  if(digitalRead(BUTTON_A_PIN) == 0){                // S
//     updateFromFS(SD); ESP.restart();                // S
//  }                                                  // S
```

platformio.ini(M5Core2用)

M5Core2用

```
[env:m5core2]
platform = espressif32
board = m5stack-core2
framework = arduino
upload_speed = 2000000
monitor_speed = 115200
board_build.partitions = default_16MB.csv

build_flags =
    -DM5C2
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM

    -mfix-esp32-psram-cache-issue

lib_deps =
    https://github.com/m5stack/M5Core2.git
    https://github.com/FastLED/FastLED
    ;https://github.com/m5stack/M5Core2/blob/master/example
    ;lovyan03/LovyanGFX
    adafruit/Adafruit BMP280 Library @ ^2.5.0
    adafruit/Adafruit Unified Sensor @ ^1.1.4
    m5stack/UNIT_ENV @ ^0.0.2
    https://github.com/earlephilhower/ESP8266Audio.git
    ;tobozo/M5Stack-SD-Updater@^1.1.8

lib_ldf_mode = deep+
```

参考情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

WiFiAnalyzer関連：
Dual Band WiFi Analyzer with Wio Terminal
M5STACK WiFiAnalyzer

M5Core2関連：
M5Stack Core2を使ってみました。

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

M5Stackファミリ関連：
M5Core2 Arduino Install
M5Paper Arduino Install
M5CoreInk Arduino Install
M5Stamp-PICO Arduino Install
M5Stamp-C3 Arduino Install
Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(STARWARS編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(MQTT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(REST-API編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(OSC編)

以上

2022/1/30
初版

uncannyeyes sketch build by platformio

uncannyeyes sketch build by platformio

概要

「M5Stack Electronic Animated Eyes」、「M5StickC Electronic Animated Eyes」のuncannyeyesスケッチをplatformioでビルドする。
両目バージョンは、M5Core2/M5Stack-Fireで共通ソース化する。
片目バージョンは、M5StickC/M5stickCPlusで共通ソース化する。
platformio.iniの内容で、対応ボードを切り替えることになる。

両目バージョン

M5Stack-Fireは、オリジナルのソースで動作するが、
M5Core2でも動作するように共通化する。

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir M5Stack_uncannyEyes
cd M5Stack_uncannyEyes

wget https://macsbug.files.wordpress.com/2019/05/m5stack_ur
mv m5stack_uncannyeyes.zip_-1.pdf m5stack_uncannyeyes.zip
unzip m5stack_uncannyeyes.zip
mv M5Stack_uncannyEyes src
cd src
mv screenshotToConsole.ino screenshotToConsole.ino.bu

gedit M5Stack_uncannyEyes.ino
# M5Core2のために以下のパッチをかける
```

パッチ内容：

```
#56行目付近：
#include <M5Stack.h>
→
#ifndef M5C2
#include <M5Stack.h>
#endif
#ifdef M5C2
#include <M5Core2.h>
#endif
-------------------------------------

#88行目付近：
#define WINK_L_PIN     39 // Pin for LEFT eye wink button
#define BLINK_PIN      38 // Pin for blink button (BOTH eye
#define WINK_R_PIN     37 // Pin for RIGHT eye wink button
→
#ifndef M5C2
#define WINK_L_PIN     39 // Pin for LEFT eye wink button
#define BLINK_PIN      38 // Pin for blink button (BOTH eye
#define WINK_R_PIN     37 // Pin for RIGHT eye wink button
#endif

#ifdef M5C2
// M5Core2
#define WINK_L_PIN     32 // Pin for LEFT eye wink button
#define BLINK_PIN      32 // Pin for blink button (BOTH eye
#define WINK_R_PIN     33 // Pin for RIGHT eye wink button
// for eyes enable status
int eyesStatus = 3; // 1:left 2:right 3:both
#endif
-------------------------------------

#122行目：
  M5.Speaker.begin();
→
#ifndef M5C2
  M5.Speaker.begin();
#endif

#129行目付近：
  pinMode(37, INPUT);pinMode(38, INPUT);pinMode(39, INPUT);
→
#ifndef M5C2
  pinMode(37, INPUT);pinMode(38, INPUT);pinMode(39, INPUT);
#endif
#ifdef M5C2
  pinMode(32, INPUT);pinMode(33, INPUT);
```

```
#endif
-------------------------------------

#153行目付近：
  if (e == 0){ M5.Lcd.setAddrWindow (  1,0,128,128);} // se
  if (e == 1){ M5.Lcd.setAddrWindow (191,0,128,128);} // se
→
#ifndef M5C2
  if (e == 0){ M5.Lcd.setAddrWindow (  1,0,128,128);} // se
  if (e == 1){ M5.Lcd.setAddrWindow (191,0,128,128);} // se
#endif
#ifdef M5C2
switch(eyesStatus) {
  case 3: // both
    if (e == 0){ M5.Lcd.setAddrWindow (  1,0,128,128);} //
    if (e == 1){ M5.Lcd.setAddrWindow (191,0,128,128);} //
    break;
  case 1: // left
    M5.Lcd.fillRect(191,0,128,128,TFT_BLACK); // clear opos
    if (e == 0){ M5.Lcd.setAddrWindow (  1,0,128,128);} //
    break;
  case 2: // right
    M5.Lcd.fillRect(1,0,128,128,TFT_BLACK); // clear oposit
    if (e == 1){ M5.Lcd.setAddrWindow (191,0,128,128);} //
    break;
  default:
    break;
};
#endif
-------------------------------------

#423行付近：
//screenshotToConsole();
}
→
//screenshotToConsole();
#ifdef M5C2
//<<=========================================
// touch button
  int keyCode = 0;

  TouchPoint_t pos = M5.Touch.getPressPoint();
    if (pos.y > 240) {
      // Yが240以上はボタンのエリア
      if (pos.x < 109) {
        keyCode = 1; // Left Button
      } else if (pos.x > 218) {
        keyCode = 3; // Mid Button
```

```
      } else if (pos.x >= 109 && pos.x <= 218) {
        keyCode = 2; // Right Button
      }
      if (keyCode != 0) {
        // ボタンを押した場合には75ミリ秒振動モーターを動かす
        M5.Axp.SetLDOEnable(3, true);
        delay(75);
        M5.Axp.SetLDOEnable(3, false);
        //Serial.println(keyCode);  // for Debug
      }
      if (keyCode == 1) eyesStatus = 1; // Left Button
      if (keyCode == 3) eyesStatus = 2; // Mid Button
      if (keyCode == 2) eyesStatus = 3; // Right Button
    }
    //Serial.println(eyesStatus); // for Debug
//>>=========================================
#endif
}
-------------------------------------
```

続き：

```


cd ..


# platformioの設定ファイルをダウンロードする
wget https://raw.githubusercontent.com/espressif/arduino-es

gedit platformio.ini
# 後節にあるplatformio.iniの内容をビルドするボードに応じて作成する

# ビルド＆書き込み
pio run -t upload
```

動作説明：
1.オリジナル版(M5Stack版)
オリジナルのものは、左ボタン、右ボタンを押すと、それぞれの目を瞑る(消える)。
中央ボタンを押すと両目を瞑る(消える)。
2.M5Core2版
オリジナルと同じ操作性は実装が難しかったので、
下の３つのタッチボタンで表示モードの切り替えにした。
左ボタン、右ボタンを押すとそれぞれの片目を表示するモードになり
中央ボタンは両目を表示するモードになる。

片目バージョン

M5StickCは、オリジナルのソースで動作するが、
M5StickCPlusでも動作するように共通化する。

以下の手順をダウンロード/ビルド/書き込みする：

```
以下のダウンロードurlがあるが、NOT_FOUNDになり使用できない。
https://macsbug.files.wordpress.com/2019/05/m5stickc_uncanr
https://macsbug.files.wordpress.com/2019/05/m5stickc_uncanr

そこで両目バージョンのプロジェクトを流用する：
(1)両目バージョンのプロジェクト・ディレクトリをコピーして
  片目バージョンのプロジェクト・ディレクトリ「M5SCx_uncannyEyes」を
(2)「M5StickC Electronic Animated Eyes」(https://macsbug.wo
プロジェクト・ディレクトリのM5Stack_uncannyEyes.inoの内容を差し替え
(3)以下のようにパッチをかける：
変更前：
#include <M5StickC.h>
→
変更後：
#ifdef M5STICK_C
#include <M5StickC.h>
#endif
#ifdef M5STICK_CP
#include <M5StickCPlus.h>
#endif

(4)platformio.iniをM5StickC,M5StickCPlus用に差し替える。
(5)「pio run -t upload」でビルド＆書き込みする。
```

動作説明：
正面の[M5]ボタンを押すと目を瞑る(消える)

platformio.ini

上のスケッチをビルドするには、ボードに応じて、
以下のplatformio.iniを使用する：

M5Core2用：

```
[env:m5core2]
platform = espressif32
board = m5stack-core2
framework = arduino
upload_speed = 2000000
monitor_speed = 115200
board_build.partitions = default_16MB.csv

build_flags =
    -DM5C2
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    https://github.com/m5stack/M5Core2.git
    https://github.com/FastLED/FastLED
    ;https://github.com/m5stack/M5Core2/blob/master/example
    ;lovyan03/LovyanGFX
    adafruit/Adafruit BMP280 Library @ ^2.5.0
    adafruit/Adafruit Unified Sensor @ ^1.1.4
    m5stack/UNIT_ENV @ ^0.0.2
    https://github.com/earlephilhower/ESP8266Audio.git
    tobozo/M5Stack-SD-Updater@^1.1.8

lib_ldf_mode = deep+
```

M5Stack-Fire用：

```
[env:m5stack-fire]
platform = espressif32
board = m5stack-fire
framework = arduino
monitor_speed = 115200
lib_ldf_mode = deep+

upload_speed = 2000000
upload_protocol = esptool
board_build.partitions = default_16MB.csv

build_flags =
    -DM5STACK_FIRE
    -DCORE_DEBUG_LEVEL=4
    -DBOARD_HAS_PSRAM
    -mfix-esp32-psram-cache-issue

lib_deps =
    m5stack/M5Stack@^0.3.9
    ;lovyan03/LovyanGFX
    adafruit/Adafruit BMP280 Library @ ^2.5.0
    adafruit/Adafruit Unified Sensor @ ^1.1.4
    m5stack/UNIT_ENV @ ^0.0.2
    https://github.com/earlephilhower/ESP8266Audio.git
    tobozo/M5Stack-SD-Updater@^1.1.8
```

M5StickC用：

```
[env:M5StickC]
platform = espressif32
board = m5stick-c
framework = arduino
build_flags = -DM5STICK_C
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    m5stack/M5StickC@^0.2.5
    ;m5stack/M5StickCPlus @ ^0.0.5
    tinyu-zhao/FFT @ ^0.0.1
    ;https://github.com/earlephilhower/ESP8266Audio.git
    earlephilhower/ESP8266Audio@^1.9.5
```

M5StickCPlus用：

```
[env:M5StickCPlus]
platform = espressif32
board = m5stick-c
framework = arduino
build_flags = -DM5STICK_CP
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    ;m5stack/M5StickC@^0.2.5
    m5stack/M5StickCPlus @ ^0.0.5
    tinyu-zhao/FFT @ ^0.0.1
    ;https://github.com/earlephilhower/ESP8266Audio.git
    earlephilhower/ESP8266Audio@^1.9.5
```

参照情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

M5Core2関連：
M5Stack Core2のタッチパネル研究
M5Stack Core2のGPIO調査
M5Stack Core2を使ってみました。

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

M5Stackファミリ関連：
M5Core2 Arduino Install
M5Paper Arduino Install
M5CoreInk Arduino Install
M5Stamp-PICO Arduino Install
M5Stamp-C3 Arduino Install
Wio-Terminal/ESP8622/ESP32ボードを共通のスケッチで動かす(NTP-CLIENT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動か

す(v2)(STARWARS編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(MQTT編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(REST-API編)
Wio-Terminal/M5Atom/ESP8622/ESP32ボードを共通のスケッチで動かす(v2)(OSC編)

その他；
Wio Electronic Animated Eyes

以上

2022/2/3
初版

Wio_ExtFlashLoad sketch build by platformio

Wio_ExtFlashLoad sketch build by platformio

概要

「Wio Ext Flash Load」、「External Flash Loader library for Wio Terminal」にあるWio_ExtFlashLoad(WriteSampleMenu.ino)スケッチをplatformioでビルドする。
arduinoのサンプル・スケッチとして提供されているものだが、
plaformioビルド環境でビルドする。
Wio_ExtFlashLoadは、アプリを起動するメニューになっており、
SDカードにビルドした.binを起き、それを選択して起動する。

WriteSampleMenu.ino

以下の手順をダウンロード/ビルド/書き込みする：

```
# platformioの環境に入る

# プロジェクト・ディレクトリを作成する
mkdir Wio_ExtFlashLoad
cd Wio_ExtFlashLoad

mkdir src
cd src
wget https://raw.githubusercontent.com/ciniml/ExtFlashLoade
wget https://raw.githubusercontent.com/ciniml/ExtFlashLoade


cd ..

gedit platformio.ini
# 後節にあるplatformio.iniの内容を作成する

# ビルド＆書き込み
pio run -t upload
```

以上で、実行すると、メニューが表示されるようになる。

platformio.ini

wio-terminal用：

```
[env:seeed_wio_terminal]
platform = atmelsam
board = seeed_wio_terminal
framework = arduino
build_flags = -DWIO_TERMINAL
upload_protocol = sam-ba
monitor_speed = 115200
lib_ldf_mode = deep+

lib_deps =
    https://github.com/Seeed-Studio/Seeed_Arduino_mbedtls/a
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcUnifie
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcBLE/ar
    https://github.com/Seeed-Studio/Seeed_Arduino_rpcWiFi/a
    https://github.com/Seeed-Studio/Seeed_Arduino_FreeRTOS/
    https://github.com/Seeed-Studio/Seeed_Arduino_FS/archiv
    https://github.com/Seeed-Studio/Seeed_Arduino_SFUD/arch
    #
    https://github.com/Seeed-Studio/Seeed_Arduino_LCD/archi
    #
    arduino-libraries/NTPClient@^3.1.0
    #
    ciniml/ExtFlashLoader@^0.1.2
```

アプリ用.binを作成する

アプリ用SDの詳細は「External Flash Loader library for Wio Terminal」に記述されているので、その中に入れる.binの作成方法について説明する。

「pio run」でビルドすると
以下に.binができる：

```
.pio/build/seeed_wio_terminal/firmware.bin
```

これをfirmware.binをapp.binにリネームしてSDにコピーする。

これだけでメニューから起動できる。
ただ、このままだとアプリ起動後、
メニューに戻ることができないので
setup()にボタンAを押しながらリセットしたときに
メニューに戻るコードを埋め込む。

(1)TFT_eSPI対応アプリの場合、「External Flash Loader library for Wio Terminal」の「アプリケーションのメニューアプリ対応」を参照のこと。

(2)LGFX対応アプリの場合は
以下のコードを埋め込む：

以下のヘッダーを追加する：

```
////////////////////////////////////////
// application menu header
#include <cstdint>
#include <ExtFlashLoader.h>
////////////////////////////////////////
```

変更前：

```
void setup() {

  lcd.init();

-------------------
```

→

変更後：

```
void setup() {
//// goto application menu ///////////////
 lcd.init();
 pinMode(WIO_KEY_A, INPUT_PULLUP);
 if( digitalRead(WIO_KEY_A) == LOW) {
    lcd.print("Launching QSPI application\r\n"); // for LGF

    ExtFlashLoader::ExtFlashLoader loader;
 }
////////////////////////////////////////
//
  //lcd.init();
```

platformio.iniには
以下のライブラリを追加する：

```
lib_deps =
    ...
        <省略>
    ...
    #
    ciniml/ExtFlashLoader@^0.1.2
```

参照情報

terminal関連:
Bootterm – a developer-friendly serial terminal program

LovyanGFX関連：
LovyanGFX - Display (LCD / OLED / EPD) graphics library (for ESP32 SPI, I2C, 8bitParallel / ESP8266 SPI, I2C / ATSAMD51 SPI) - M5Stack / M5StickC / TTGO T-Watch / ODROID-GO / ESP-WROVER-KIT / WioTerminal / and more...
LovyanGFX LCD Graphics driver
LovyanGFX-Display ライブラリを使用したスケッチをplatformioでビルドする

platformio関連：
arduinoフレームワーク用platformio.ini集
Building Core2 FactoryDemo in PlatformIO
VSCodeとPlatformIOでM5Stack Core2開発
M5Stack Core2とVSCode + PlatformIOとでM5Stackプログラミングを始めてみた。

Arduino-IDE関連：
Arduino IDE environment - M5Paper
Arduino IDEのインストールと設定 (Windows, Mac, Linux対応)

以上

2021/1/3
初版

MicroPython(F767ZI) Network Samples

MicroPython(F767ZI) Network Samples

概要

MicroPython(F767ZI) Network Samples

この記事は「 MicroPython(F767ZI)でStartwars(AsciiArt)を動かす 」の続編にあたり、NetworkアクセスSamplesを記述している。ネットワークの接続が完了すれば、それ以降はESP8266のmicropythonとほぼ同じようだ。
(ホストはubuntuを想定している)

sample1

http_get.py

```
# ネットワークを有効化する
import network
nic = network.LAN()
nic.active(1)

# 以下はESP8266のサンプルと全く同じ

def http_get(url):
    import socket
    _, _, host, path = url.split('/', 3)
    addr = socket.getaddrinfo(host, 80)[0][-1]
    s = socket.socket()
    s.connect(addr)
    s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n' % (
    while True:
        data = s.recv(100)
        if data:
            print(str(data, 'utf8'), end='')
        else:
            break
    s.close()

print(http_get('http://micropython.org/ks/test.html'))
```

REPL実行例：

```
paste mode; Ctrl-C to cancel, Ctrl-D to finish
=== import network
=== nic = network.LAN()
=== nic.active(1)
===
===
>>> nic.ifconfig()
('192.168.0.18', '255.255.255.0', '192.168.0.4', '192.168.0
>>>
paste mode; Ctrl-C to cancel, Ctrl-D to finish
===
=== def http_get(url):
===     import socket
===     _, _, host, path = url.split('/', 3)
===     addr = socket.getaddrinfo(host, 80)[0][-1]
===     s = socket.socket()
===     s.connect(addr)
===     s.send(bytes('GET /%s HTTP/1.0\r\nHost: %s\r\n\r\n'
===     while True:
===         data = s.recv(100)
===         if data:
===             print(str(data, 'utf8'), end='')
===         else:
===             break
===     s.close()
===
=== print(http_get('http://micropython.org/ks/test.html'))
===
===
```

REPL出力結果：

```
HTTP/1.1 200 OK
Server: nginx/1.10.3
Date: Sun, 03 Jan 2021 09:54:59 GMT
Content-Type: text/html
Content-Length: 180
Last-Modified: Tue, 03 Dec 2013 00:16:26 GMT
Connection: close
Vary: Accept-Encoding
ETag: "529d22da-b4"
Accept-Ranges: bytes

<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Test</title>
    </head>
    <body>
        <h1>Test</h1>
        It's working if you can read this!
    </body>
</html>
None
```

sample2

目的のプログラム実行前に
以下を実行して ネットワークの有効化と
IPアドレスの確認を行なう。

```

import network
nic = network.LAN()
nic.active(1)

nic.ifconfig()
```

httpserver.py

```
# 以下はESP8266のサンプルを差分のあるmachine関係をコメントアウトした

#import machine
#pins = [machine.Pin(i, machine.Pin.IN) for i in (0, 2, 4,

html = """<!DOCTYPE html>
<html>
    <head> <title>ESP8266 Pins</title> </head>
    <body> <h1>ESP8266 Pins</h1>
        <table border="1"> <tr><th>Pin</th><th>Value</th></
    </body>
</html>
"""

import socket
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]

s = socket.socket()
s.bind(addr)
s.listen(1)

print('listening on', addr)

while True:
    cl, addr = s.accept()
    print('client connected from', addr)
    cl_file = cl.makefile('rwb', 0)
    while True:
        line = cl_file.readline()
        if not line or line == b'\r\n':
            break

#rows = ['<tr><td>%s</td><td>%d</td></tr>' % (str(p), p.val
    rows = ['<tr><td>ABC</td><td>123</td></tr>']
    response = html % '\n'.join(rows)
    cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\r
    cl.send(response)
    cl.close()
```

REPL実行例：

```
>>> nic.active()
True
>>> nic.ifconfig()
('192.168.0.18', '255.255.255.0', '192.168.0.4', '192.168.0
>>>
# 上の192.168.0.18がサーバーアドレスになるので覚えておいてサーバー起

paste mode; Ctrl-C to cancel, Ctrl-D to finish
=== #import machine
=== #pins = [machine.Pin(i, machine.Pin.IN) for i in (0, 2,
===
=== html = """<!DOCTYPE html>
=== <html>
===     <head> <title>ESP8266 Pins</title> </head>
===     <body> <h1>ESP8266 Pins</h1>
===         <table border="1"> <tr><th>Pin</th><th>Value</t
===     </body>
=== </html>
=== """
===
=== import socket
=== addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
===
=== s = socket.socket()
=== s.bind(addr)
=== s.listen(1)
===
=== print('listening on', addr)
===
=== while True:
===     cl, addr = s.accept()
===     print('client connected from', addr)
===     cl_file = cl.makefile('rwb', 0)
===     while True:
===         line = cl_file.readline()
===         if not line or line == b'\r\n':
===             break
===
=== #rows = ['<tr><td>%s</td><td>%d</td></tr>' % (str(p), p
===     rows = ['<tr><td>ABC</td><td>123</td></tr>']
===     response = html % '\n'.join(rows)
===     cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html
===     cl.send(response)
===     cl.close()
===
===
```

上でサーバーを起動したら 確認したIPアドレス(192.128.0.xxx:80)で ブラウザー・アクセスする。

REPL出力例：

```
listening on ('0.0.0.0', 80)
client connected from ('192.168.0.2', 56016)
44
229
client connected from ('192.168.0.2', 56018)
44
229
```

ウェブ画面表示例：

ESP8266 Pins

Pin	Value
ABC	123

参考情報

https://docs.micropython.org/en/latest/esp8266/tutorial/network_basics.html
https://docs.micropython.org/en/latest/esp8266/tutorial/network_tcp.html

- M5Stack MicroPython API調査記録　ネットワーク編
- ESP32/ESP8266のMicroPythonでMQTTを使ってみた
- ESP32のMicroPythonでOSC(Open Sound Control)で通信する

以上