# 共通情報

## マイコンの仕組み

### CISCとRISC

CISC；Complex Instruction Set Computer

複雑命令セットコンピュータ。一つの命令で複雑な処理を実現できる。

RISC；Reduce Instruction Set Computer

縮小命令セットコンピュータ。―つの命令がシンプルで、動作速度が速い。

複雑な処理はワイヤードロジック（物理的に結線された論理回路）によって実現される。

### バスコントローラー

割込みコントローラ、タイマ、A/D変換など内部バスであるperipheral equipment（周辺機器；ペリフェラルとも呼ばれる）に対してCPUを介した外側と窓口。

### DMA

CPUを介さずにROM/RAM/周辺機器への読書きを行う。

通常マザーボードやマイコンチップなどにDMAC（専用の制御IC）が搭載されCPUとの通信を制御する。

### PIO

ソフトウェアによるROM/RAM/周辺機器への読書き。DMACを介して通信する為DMA方式より通信負荷が大きい。

### SIO；Serial I/O

文字通りだが、シリアル通信を行う為の調歩同期などを行う。

SIO ICは概ねCMOSなので、232C, 422等として使うためにはレベル変換ICが必要。

CAN；Controller Area Network0

PPG；Pulse Pattern Generator

UDC；Up Down Counter

### RTC

通常I2Cで、マイコンの起動時などに時刻情報を取得する。

LinuxOSの場合、Devicetree Overlayでi2c-rtcという仕組みがあり、ブート時に自動的に読み込んでくれる（らしい）。

## メモリの領域

### スタック領域

メソッド（関数）呼び出しに関する情報を格納するメモリ領域。

A，B，Cの順に確保したメモリは，C，B，Aの順番で解放される。

この領域の大きさを stuck size と言う。

ヒープ領域

malloc やnew 等で動的に確保されるメモリ領域。

静的領域

グローバル変数、（定数？）などの静的変数が置かれる。

テキスト領域

機械語（ニーモニック）に翻訳されたプログラムが格納される領域。

## ディスプレイ

### モノクロディスプレイ

GPEG：KS008A4B

## 交流電流測定

家庭内の使用電力をデータベースで

電流センサはこの辺りが安い

<https://akizukidenshi.com/catalog/g/gP-08960/>

交流の実効値を取る場合、たかだか60Hzではあるがpythonでは少しきついらしい。C言語で書く必要がある。

（情報元）

<https://qiita.com/osamasao/items/a38707465e8f8a9948ed>

<https://www.ishikawa-lab.com/RasPi_wattmeter.html>

## 通信

### 無線232C

#### 概要

ブルートゥースにはSPP；Serial Port Profile と言う通信規格があり，これはRS-232Cを仮想化する為の通信規格。

ようするにRS-232Cのシリアル通信を無線化できる。

（補足）この通信規格の為に232Cは無線LAN化するよりも，ブルートゥース化する方がはるかに容易。

また、SPP通信は受信先（例えばパソコン）でCOMポートとして認識されるらしい。

#### 変換基板を自作する方法

20.06.30やった事はないのでテストが必要

情報１：<https://ameblo.jp/ikkotei/entry-12323853292.html>

情報２：<https://www.elp.co.jp/staffblog/?p=7678>

##### 必要な物

秋月　Bluetooth無線モジュール評価キット \2,400

<http://akizukidenshi.com/catalog/g/gK-07378/> ※これはusb給電が可能

（RN42-I/RMという通信基盤が載った評価キット）

または

SparkFun　WRL-12582 \3,056

[https://www.sengoku.co.jp/mod/sgk\_cart/detail.php?code=EEHD-4Z8F#](https://www.sengoku.co.jp/mod/sgk_cart/detail.php?code=EEHD-4Z8F)

（こちらもRN42-I/RMを使用している）

⇒ ※但しこのどちらも5V系らしく，電圧変換が必要（RS232Cは±7VをHigh信号とみなす）

ＲＳ２３２レベル変換基板 \500：<http://akizukidenshi.com/catalog/g/gK-06464/>

9pin　Dサブコネクタ 約 \50

無線基板への電源アダプタ（５V）約 \500 通信基板は50mAとかであれば十分らしい

#### 補足情報

・販品で232C ↔ ブルートゥースのアダプタは販売されている，が高価（２万くらい）

RS422/485変換するデバイスも販売されている。がこれも高価（２万）。

<http://intersolutionmarketing.jp/store/products/detail.php?product_id=73>

#### ラズベリーパイ

ブルートゥースは基本的に1対1通信を想定しているので，複数機器とブルートゥース通信が必要なら，機器群との通信を総括する端末としてラズベリーパイのようなものを使った方が早いかも。

ラズパイなら232Cも422/485対応していて，ブルートゥース通信も可能。

### Bluetooth関連

#### Web Bluetoothについて

https://houwa-js.co.jp/2021/03/20210316/

Google Chartsと合わせるとWebブラウザだけでセンサの見える化ができる。

## 電源問題

### 電池種類

リチウム電池

長所

短所

高温環境で使えない

鉛蓄電池

長所

短所

高温環境で使えない

電気二重層コンデンサ

長所

短所

太陽電池

長所

短所

ボタン電池

長所

短所

容量が少ない。以下が比較的大容量。

CR2354：3V/560mA、CR2450：3V/610mA、CR3032：3V/500mA

### 外部電源駆動

・ダイオードで保護？

・理想ダイオードICを使う？ →高い（800円位）

参考：https://akizukidenshi.com/catalog/g/gK-07919/

・開発ボードの回路を参考にする

## アイソレータ 基板

# microbit

## 概要

電源 1.8-3.6V 情報元不明だが，6Vまでは大丈夫と言う情報もある。

ROM 256KB

RAM 16kB

Core variant: ARM Cortex-M0+

Speed 16Mhz (crystal) 48MHz (max)

USB current 120mA max

ボタンスイッチ × 2個 加速度センサ × 1個 磁力センサ × 1個

温度センサー？

ポートは４つに見えるが、アナログ３つ、デジタル１７有る。I2C, SPI通信もできる。

Wifiは無いが、ブルートゥースがある。（但しBLE；範囲は5Mほど）

アナログ出力は驚いた事に音声も出せる。

プログラムファイルの拡張子は.hex

## microbit v2

新旧比較

|  |  |  |
| --- | --- | --- |
| 旧 (v1.5) | 機能 | 最新 (v2) |
| Nordic Semiconductor nRF51822 | プロセッサ | Nordic Semiconductor nRF52833 |
| 256kB Flash 16kB RAM | メモリ | 512kB Flash 128kB RAM |
| NXP KL26Z, 16kB RAM | インターフェース | NXP KL27Z, 32kB RAM |
| 未対応 | マイク機能 | LED付きMEMSマイク |
| 未対応 | スピーカー機能 | オンボードスピーカ |
| 未対応 | タッチセンサ | タッチセンサ対応ロゴ |
| 25ピン。専用GPIO×3、I2C、SPPI、外部電源。 ワニ口プラグ/バナナプラグ接続用リングピン×3。接続用リングピン×3。 | エッジコネクタ | 25ピン、専用GPIO×4、PWM、I2C、SPI、外部電源。ワニ口プラグ/バナナプラグ接続用リングピン×3(接続を容易にするためにノッチ加工) |
| 共有 I2C | I2C | 専用 I2C bus |
| 2.4GHz micro:bit無線/Bluetooth4.0 | 無線 | 2.4GHz micro:bit無線/Bluetooth5.0 |
| microUSBポート経由で5V、エッジコネクタまたはバッテリー経由で3V | 電源 | microUSBポート経由で5V、エッジコネクタまたはバッテリー経由で3V。(LED電源インジケータ付き、電源ボタンを押し続ける電源オフ) |
| 拡張ボード用に最大 90mA | 電力供給 | 拡張ボード用に最大 200mA |
| ST LSM 303 | モーションセンサ | ST LSM 303 |
| C++, Makecode, Python, Scratch | 対応言語 | C++, Makecode, Python, Scratch |
| 50mm(W) x 40mm(H) | サイズ | 50mm(W) x 40mm(H) |

## 外部デバイス

### バッテリー

・使い捨てでも良ければ、ボタン電池が一番楽

・大容量コンデンサで対応する？但し別途充電用の回路基板を作る必要がある。

<http://akizukidenshi.com/catalog/g/gP-04300/>

・カメラ用に3Vバッテリーが普通に販売されている。但し専用充電器を買う必要がある。

・他にもボタン型充電電池<http://akizukidenshi.com/catalog/g/gB-08345/>

・通常の充電電池×３で3.6Vだして落とした方が良いかも？

注意点

入力電源が3.6Vを超えると、micro:bit内のデバイスを壊す可能性がある。

電源コネクタ(PHコネクタ)から供給

逆流防止用のダイオードBAT60を経由して他のデバイスに供給されますので、供給される電圧は0.1～0.2V電圧降下します。

## 開発環境

### PCとWebブラウザで行う

対応ブラウザ：Microsoft Edge，Google Chrome，Mozilla Firefox，Safari

開発用Webサイト：<https://makecode.microbit.org/>

⇒ 新しいプロジェクト　をクリック

microbitとPCはUSBで接続する。

ブロック（独自言語），Java，Pythonから選ぶ

デメリット：

Web接続したPCが無いと開発できない。

インテリセンスが使えない。

### PCと専用エディタで行う

Muエディタダウンロード：<https://codewith.mu/>

補足）Win10であればMicrosoft Store で専用アプリがある。

### スマホとBluetoothで行う

BLE(Bluetooth 4.0) 対応しているスマホでmicro:bit とペアリング＋専用アプリを用いる事で開発ができる。

※但しAndroid のmicro:bitアプリは「通信できない」と言う苦情が非常に多く見られた。

## 外部キットなど

### おもちゃ系

#### Yahboom

LEGOを使ったラジコンカー

<https://sanuki-tech.net/micro-bit/appendix-products/yahboom-running-bit/>

複合キット；なんとヘリコプターまである。

<https://category.yahboom.net/products/buildingbit-super-kit>

# meowbit

micro:bitと接続する事ができる、ゲーム用途でのマイコン。

Microsoft Makecode Acradeに対応した携帯ゲーム機。

Makecodeでゲームを作成できる？

# Arduino

## 概要

ラズパイに比べると純然たるマイコンに近い。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | プロセッサ | クロック |  | 価格 | その他 |
| Uno R3 | ATmega32 8p | 16MHz |  | ¥3,300 | PWN pin6本 |
|  |  |  |  |  |  |
| ESP8266 |  |  |  | ¥1,000 | wifi搭載 |
| ESP32 | Tensilica LX6 | 32kHz |  | ¥1,100 | Arduino core for the ESP32  wifi搭載  **厳密にはArduinoではない** |
| Nano RP2040 |  |  |  | ¥1200 | Arduino Studio で開発できるRaspberry Pi Pico(RP2040)ボード。 |

ItsyBitsy

## チュートリアル

### インストール

ダウンロード元

<https://www.arduino.cc/en/software>

Linux AppImage 64 bits (X86-64)

Linux ZIP file 64 bits (X86-64)

補足：

公式（Arduino IDE） C/C++ を使う。 オープンソース

Eagleファイル：（CC-BY-SA）著作権表示、元作品のライセンスを継承。商用利用は可能。

Arduino　IDE（GPLライセンス）

（参考）<https://www.arduino.cc/en/Main/CopyrightNotice>

lsusbでポート番号を確認

## Arduino IDE

### ライブラリの追加

スケッチ → ライブラリをインクルード

### コンパイル・書き込み

スケッチ → 検証・コンパイル

マイコンボードに書き込む PCから直接接続している場合？

書込装置を使って書き込む デバッガなど別のデバイスを介して書き込む場合？

### デバッグ

シリアルモニタでの値の確認：

まずsetupメソッドでシリアルモニタの通信の開始を記述する必要がある

void setup() {

Serial.begin(115200); // 指定のbuadrateでシリアルモニタを開始する。

// Serial.beginは省略可能だが、その場合Serial.print が使えなくなる。

}

次にツール→シリアルモニタ でシリアルモニタを開き、右下のコンボボックス（右図参照）でbaudrateをSerial.begin() で指定したものと同じにする。

あとは必要な時にSerial.print() メソッドを実行する。

void loop() {

if(value==1){

Serial.print("hoge"); // console画面に出力

}

}

### setup（詳細）

void IRAM\_ATTR onTimer(){

}

void IRAM\_ATTR io\_callback(){

}

void setup() {

attachInterrupt(32, io\_callback, FALLING); // 割り込み登録　(pin#, メソッド, フラグ)

timerAttachInterrupt(timer1, &onTimer, true);

}

attachinterruptのフラグ

|  |  |
| --- | --- |
| LOW | LOWのとき発生 |
| CHANGE | 状態が変化したときに発生 |
| RISING | LOW → HIGH（立上がり）で発生 |
| FALLING | HIGH → LOW（立下がり）で発生 |

### データ型

|  |  |
| --- | --- |
|  |  |
| boolean | TrueまたはFalse |
| char | -128～127までの整数 |
| unsigned char | 0～255までの整数 |
| byte | 0～255までの整数 |
| int | -32768～32767までの整数 |
| unsigned int | 0～65535までの整数 |
| word | 0～65535までの整数 |
| long | -2147483648～2147483647までの整数 |
| unsigned long | 0～4294967295までの整数 |
| float | -3.4028235E+38～3.4028235E+38 |
| double | -3.4028235E+38～3.4028235E+38 |
| int8\_t | 8ビットの整数を格納 |
| uint8\_t | 8ビットの非負整数を格納 |
| int16\_t | 16ビットの整数を格納 |
| uint16\_t | 16ビットの非負整数を格納 |
| int32\_t | 32ビットの整数を格納 |
| uint32\_t | 32ビットの非負整数を格納 |
| int64\_t | 64ビットの整数を格納 |
| uint64\_t | 64ビットの非負整数を格納 |

型変換

（公式情報）<https://www.arduino.cc/en/Reference/Cast>

### 割込み

#### 概要

タイマーに使われるのはCPUに使われるクロックでは無く，peripheral周波数と言うもの。

#### 基本プログラム例（タイマー）

hw\_timer\_t \*timer = NULL; // NULLが無いとエラーになった 2021-08-09

portMUX\_TYPE timerMtutex = portMUX\_INITIALIZER\_UNLOCKED;

volatile int counter=0; // 割込み中にアクセスする変数にはvolatileを付ける

void IRAM\_ATTR onTimer(){ // IRAM\_ATTR　高速RAMへの配置を指定する

portENTER\_CRITICAL\_ISR(&timerMtutex); // 割込み禁止区間　開始

counter+=1; // 割込みハンドルのメソッドでは最低限の処理のみを行う

portEXIT\_CRITICAL\_ISR(&timerMtutex); // 割込み禁止区間　終了

xSemaphoreGiveFromISR(timerSemaphore, NULL); // セマフォに許可を与える

}

void setup() {

timer = timerBegin(0, 80, true); // (タイマー#，divider，countUp)

timerAttachInterrupt(timer, &onTimer, true); // (タイマーハンドル，割込みハンドラ，エッジ割込み)

timerAlarmWrite(timer, 1000000, true); // (タイマーハンドル，タイマーカウント数，継続タイマ)

timerAlarmEnable(timer); // タイマーアラーム（割込み）を有効

}

タイマー分周の考え方

分周は英語ではdividerと言い，　peripheral周波数÷divider = タイマー動作周波数　と，文字通り割り算でタイマー動作周波数を決める。ESP32ではperipheral周波数が80M(80 \* 106)Hzなのでdivierを80にするとタイマー動作周波数は (80 \* 106)÷80= 1M(106)Hz（=1μsec毎） となる。

これを「タイマクロックは80MHzを80分周」などと表現する。

なお，getApbFrequency()/1000000(=106) でどんなCPUでも1μSecにできる。

タイマーのアラーム（≒割込み）はtimerAlarmWriteの第２引数で指定したカウント数で動作する為，　割り込み発生タイミング ＝ タイマー動作周波数 × カウント数　となり，この場合，

1μ(1/106)Sec×1,000,000(=106) = 1sec となり，1秒毎に割込みが発生する。

（参考）<http://marchan.e5.valueserver.jp/cabin/comp/jbox/arc202/doc21105.html>

補足

dividerはプリスケーラ―とも呼ばれる

#### 注意点

・割込み処理を行うメソッド（以後ISRとする）は必要最低限の事だけを行う。

・ISRで待ち時間が発生する通信処理などを行ってはいけない。

・割込みの検知と具体的処理はメインループ（loop）で行う。

・ISRにはIRAM\_ATTR属性を付ける。これは高速メモリに配置される事を意味する。

・ISRは，戻り値を持たないvoid 型のメソッドで無ければならない。

・割込み処理中にアクセスする変数にはvolatile を付ける。（ROMでなく，RAMに配置される？）

・サフィックス（接尾辞）がISRになっているメソッドがあれば，ISR内ではそちらを使う

・高度な処理，時間のかかる処理は排他制御で保護し，別タスクで実行する 意味不明

#### ライブラリ（割り込み）

IRAM\_ATTR　このメソッドを高速なIRAMの上に置く事を指定する。

getApbFrequency()

APB bus clock；APBはAdvanced Periheral Busの意で，peripheral周波数を返す関数。

timerAttachInterrupt

ISR（関数）を割込処理に結びつける

構文：void timerAttachInterrupt (hw\_timer\_t \* timer, void (\* fn)(void), bool edge);

hw\_timer\_t： タイマーのハンドラへのポインタ

fn： ISR（関数）のアドレス

edge： true：エッジ割込指定（タイマーの場合こちら）　false：レベルトリガー

timerAlarmWrite

構文：void timerAlarmWrite(hw\_timer\_t \*timer, uint64\_t alarm\_value, bool autoreload);

timer タイマーのハンドラへのポインタ

alarm\_value ISR（関数）を呼び出すまでのタイマのカウンタ数

autoreload true：連続動作 false：ワンショット

timerAlarmEnable

指定したタイマーを有効化する。

(opp) timerAlarmDisable

portENTER\_CRITICAL()　portEXIT\_CRITICAL()

この２つの関数に囲まれている場所はcritical sectionと呼ばれ，割込みが発生しない。

割込み中に変更されては困る変数が有る場合に，この２つの関数で囲う。

portMUX\_TYPE（型）

loop とISR（割り込みをハンドルするメソッド）間で共通変数へアクセスを管理する。

portENTER\_CRITICAL()で使用する。割込み中に変更されては困る変数が有る場合に使う。

muxtex；Mutual Exclusion相互排除。

値

portMUX\_INITIALIZER\_UNLOCKED 排他制御されていない状態（初期値）

xSemaphoreTake

セマフォを取得する

構文：xSemaphoreTake(timerSemaphore, timeout)

引数

timerSemaphore SemaphoreHandle\_tオブジェクト。

timeout　 セマフォを取得できなかった場合のタイムアウト時間（単位？）

値：portMAX\_DELAY 取得できるまで待ち続ける。

戻り値

pdTRUE 取得成功

xSemaphoreCreateBinary

構文：SemaphoreHandle\_t semahore xSemaphoreCreateBinary()

一人だけ取得できる「バイナリセマフォ」を作成。ただし、デフォルトは誰もセマフォを取得できず，誰かが許可した場合にだけ取得可能。

xSemaphoreGiveFromISR

セマフォに許可を与える

xSemaphoreGiveFromISR(timerSemaphore, NULL);

第二引数は本来なら以下の様に処理する必要がある

static signed portBASE\_TYPE xHigherPriorityTaskWoken = pdFALSE;

xSemaphoreGiveFromISR( xSemaphore, &xHigherPriorityTaskWoken );

### ライブラリ（抜粋）

参考資料

<http://www.musashinodenpa.com/arduino/ref/index.php>

<https://garretlab.web.fc2.com/arduino_reference/language/functions/index.html>

<https://garretlab.web.fc2.com/arduino/esp32/reference/libraries/>

#### 行き先不明

#define LED\_BUILTIN 4 // プリプロセッサでピン番号を指定（GPIOの番号では無い）

※グローバル変数は volatile が必要？

※sqrtの戻り値はdoubleらしいが， (float)sqrt(mean); としなくても型変換してくれるらしい。

（公式情報）<https://www.arduino.cc/reference/en/language/functions/math/sqrt/>

#### デジタルIO

##### pinMode

構文：pinPode(pin, mode)

pin:

mode: INPUT, OUTPUT, INPUT\_PULLUP,

pinMode(13, OUTPUT);

（参考）<https://www.arduino.cc/reference/en/language/functions/digital-io/pinmode/>

digitalWrite/Read

digitalWrite(1, 1) // (pin#, 状態) 0:LOW, 1:HI

digitalRead(1) // (pin#)

String は const char\* として定義されている

“c” はconst char\*(String) とみなされ，’c’ はcharとみなされる

#### IO応用

##### pulseIn

構文：pulseIn(pin, value [, timeout])

pin: パルスを読み取るピン番号

value: パルスとして認識する電圧。HIGH, LOW

timeout 単位μSec。型：unsigned long

戻り値 型：unsigned long

タイムアウトした場合=0。パルス入力があった場合はμSec値が返る

void setup()

{

pinMode(7, INPUT);

}

void loop()

{

unsigned long duration = pulseIn(pin, HIGH);

}

参考：<https://cdn.arduino.cc/reference/jp/language/functions/advanced-io/pulsein/>

#### 定義済関数

##### millis

MCUがプログラムを開始した時から現在までの経過時間をmSecで返す。

構文：millis()

##### sqrt

平方根を返す。

構文：double sqrt(x)

#### クラス

##### Servo

#include <Servo.h>

Servo servo; //Servoオブジェクトを作成

void setup() {

servo.attach(15); //PWMの制御線をD15以外に接続している場合はここの数字を変更する

}

void loop() {

servo.write(20); //サーボモータの角度を20度に設定

delay(2000); //2秒待機

servo.write(160); //サーボモータの角度を160度に設定

delay(2000); //2秒待機

}

情報元<https://entotsu44.hatenablog.com/entry/2021/05/04/090000>

##### Serial

説明：シリアルモニタ（確認用のコンソール画面）への出力を管理する。

print() 文字列をシリアルモニタに表示する。

printLn() 改行コード付きで文字列をシリアルモニタに表示する。

write()

##### Ticker

##### WiFi

モジュール：WiFi.h

###### status()

|  |  |
| --- | --- |
| 値 | 説明 |
| WL\_CONNECTED | WiFiネットワークに接続完了。 |
| WL\_NO\_SHIELD | WiFiシールドが存在しない。 |
| WL\_IDLE\_STATUS | WiFi.begin()の呼び出しから、接続完了、または、接続失敗までのアイドル状態。 |
| WL\_NO\_SSID\_AVAIL | 利用できるSSIDが存在しない。 |
| WL\_SCAN\_COMPLETED | ネットワークスキャン完了。 |
| WL\_CONNECT\_FAILED | WiFiネットワークに接続失敗。 |
| WL\_CONNECTION\_LOST | WiFiネットワークの接続ロスト。 |
| WL\_DISCONNECTED | WiFiネットワークから切断。 |

（情報元）<https://www.arduinolibraries.info/architectures/esp32>

onEvent()

##### WiFiClient

Webクライアントとの接続や情報を管理する。

メソッド

available

connected

構文：bool connected( [host] , [port] );

read

データを読み込む。引数を省略した際は1byteを読み込む。

構文：int read([uint8\_t \*buf], [size\_t size]);

readStringUntil()

構文　String readStringUntil()

flush

print

println

文字列をバイトにして送信する。

##### WiFiServer

簡易的なhtmlサーバーを提供する。

（参考資料）<http://microtechnica.xyz/?p=271>

begin

用法：void begin(uint16\_t port = 0, [ int reuse\_enable ]);

port サーバが利用するポート番号。省略時は、コンストラクタで指定した値を利用。

reuse\_enable ローカルアドレスの再利用を許可の指定。

##### xSemaphoreCreateBinary

バイナリセマフォを提供？

# ESP32

## 概要

Arduino IDEで開発が可能（※１）なマイコンだが、Espressif Systemsという会社が作っている別のマイコン。

Arduinoと共に，制御プログラムは「スケッチ」と呼ばれる。

ESP8266

（参考）<https://tech-blog.cerevo.com/archives/859/>

こんなものもある

<https://akizukidenshi.com/catalog/g/gK-09758/>

（※１）Arduino core for the ESP32というものを使っている。

シリーズの違い（ESP-WROOM）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 02 | 02D | 02U | 32D | 32 |
| 値段（単体） | \250 | \360 | \350 | \460 | \550 |
| CPU | ESP8266EX | ESP8266EX | ESP8266 | ESP32-D0WD | Tensillica LX6 |
| クロック | 80MHz | 80MHz | 80MHz | 40MHz | 160MHz |
| SRAM | 36KB | 50KB? | 50KB? | 520KB | 520KB |
| フラッシュ | 2MB | 2MB | 2MB | 4MB | 4MB |
| 動作電圧 | 3.0~3.6V | 2.7~3.6V | 2.7~3.6V | 2.7~3.6V | 2.2~3.6V |
| GPIO | 8本 | 8本 | 8本 | 21本 | 21本 |
| Wifi | b/g/n | b/g/n | b/g/n | b/g/n | b/g/n |
| Bluetooth | なし | なし | なし | v4.2 | v4.2 |
| 消費電力 | 80mA | 80mA | 80mA | 80mA | 80mA |
| その他 | ADC1本 | 02の後継 | アンテナが付けれる | 32Uもある | ADC16本  DACが2本 |

※Arduinoではフラッシュが32KBほどしかないらしい。

※ESP02よりもESP32の方が消費電力は少ないらしい（Deep Sleep時）

ＥＳＰ３２－ＷＲＯＶＥＲ－Ｅ

2020年末に発売された新機種。SRAMのバグなどが修正されている。

ESP32 ECO V3というリビジョンのチップを搭載。

## 詳細情報

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/>

各ピンの特徴を書くのは膨大になってしまうため，注意事項のみを箇条書きにする。

### PIN

ESP32

GPIO34～39では内部pull-up，pull-down抵抗は使えない

ＥＮ イネーブル端子 10kΩでプルアップ。通常は HI（使用可能）。

GND に接続する事で LOW になりアンイネーブル状態になる

※ESP32 の場合はこれをリセットボタンとしている

PGM ボタン

別名：BOOT, IO0

書込みモード／プログラム実効モード を切り替える。

PGM が HIGH（ボタン開放）の状態で RST ボタンが押されると、実行モード。

PGM が LOW（ボタン押下）の状態で RST ボタンが押されると、書込みモード

公式：<https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>

### 遠隔操作

外部サービスを使う

ngrok

Blynk

Netlify

ESP32をWifi経由で動かす方法：<https://qiita.com/uhey22e/items/c285edf79893db14bfa6>

### Deep Sleep

タイマー（一番使いやすい）

GPIO立ち上がり割り込み

下記サイトがわかりやすい

<https://lang-ship.com/blog/work/esp32-deep-sleep/>

wifiを利用する時は通信時間≒エネルギー消費量となる。短くするのが吉。

⇒ 必然的にクライアントとしてネットワークに接続する事になる。

### 赤外線リモコン

ツール→ライブラリを管理→検索ボックスにIRremoteESP　IRremoteESP8266が表示されるのでインストール

その後、ファイル→スケッチ例→IRremoteESP8266→IRrecDumpV2

・使用するピンは4番 (D2)が推奨らしい

Espressif ESP-EYE

ESP32-CAM 技適が通っていない？

M5Stack ESP32 Camera

いづれも日本での入手経路は不明。

## 開発環境

### ESP-IDF

公式の総合開発環境。初心者はArduino IDEから入り，限界を感じてきたらこちらに移行すると良いらしい。

4.1までは簡単に環境を作れる。4.2以降はちょっとめんどくさいらしい。

開発ボード

ENがリセットボタン

### Arduino IDE

シリアル通信のドライバをインストール

Windows

ダウンロード先：https://jp.silabs.com/developers/usb-to-uart-bridge-vcp-drivers

右図はインストール後のデバイスマネージャー

Linux

Linuxでは大体のシリアルICのドライバが予め入っている。但し，Arduino Studio内部でシリアル通信にpython(2)のpyserialを使用しているのでこちらをインストールしておく必要がある。

sudo pip insatall pyserial # python(2) にpyserialをインストール

sudo chmod o+wr /dev/ttyUSB0 # USBケーブルを抜き差しするごとに書き込み権限の設定が必要

（ここまでの情報参考元）[https://qiita.com/ycrack/items/bbd7f9f95376cca53241#windows](https://qiita.com/ycrack/items/bbd7f9f95376cca53241" \l "windowsの場合)の場合

Arduino IDE本体のダウンロード：<https://www.arduino.cc/en/software>

インストール方法：

Windows

arduino-\*.\*.\*-windows.exeを実行するだけ

Linux

tar xJfv arduino-\*.\*.\*-linux64.tar.xz # 展開＋解凍　　\*.\*.\* はバージョン

sudo ./arduino-\*.\*.\*/install.sh # インストール用スクリプトの実行

#source ./arduino-\*.\*.\*/install.sh # このやり方ではうまく行かなかった

～Arduino IDE　つづき

#### Arduino IDEの設定

ボードの追加

2023-09-30追記

（Board Manager） → 検索 → esp32 → INSTALL

ファイル → 環境設定 → 追加のボードマネージャのURLを追加（下の表を参照）→ OK。

|  |  |  |
| --- | --- | --- |
|  | URL | キーワード |
| ESP32 | <https://dl.espressif.com/dl/package_esp32_index.json> | ESP32 |
| ESP02 | <http://arduino.esp8266.com/stable/package_esp8266com_index.json> | esp8266 |

ボードの選択

ツール → ボード → ボードマネージャ →  にESP32と入力 → インストール

#### サンプルプログラム

ファイル → スケッチ例 → 01.Basics → Blink；いわゆる「Lチカ」のソース

補足：#define LED\_BUILTIN 4　としておく

void setup() { // 起動時に実行される、お決まりのメソッドらしい。

Serial.begin(115200); // Serial.println()を行う為に必要

pinMode(4, OUTPUT); // ここでのピン番号はIO番号ではない。基板に書いてある番号。

}

void loop() { // 同じく，while true{} されるお決まりメソッドらしい。

digitalWrite(4, HIGH);

delay(3000);

digitalWrite(4, LOW);

delay(3000);

}

その他のサンプル

Wifi：ファイル → スケッチ例 → Wifi（ドロップダウンの下の方） → SimpleWifiServerなど

その他の項目は[Arduino：Arduino IDE](#_Arduino_IDE)を参照。

### MicroPython

<https://micropython-docs-ja.readthedocs.io/ja/latest/esp32/tutorial/intro.html>

### その他

VS Code 公式がExtensionを出しているらしい。

Visual Studio

Eclipse

### ビルドツール

## 関連商品

### TinyPICO

### ESP32-PICO-KIT

## 関連ライブラリなど

### Tasmota

ESP32で開発する際のソフトウェア部分をGUIで簡単にできるようにするツール。

## 具体例

### ライタの自作

#### 必要な物

##### USB to シリアル

CP2102, FT234X, FT231XS, FT232RL などを積んだもの。

※製品はいくらでもあるが，500mA取れる3.3V出力のある製品は少ない！

##### 電源

5.0VDC → 3.3VDC

500mA以上が必要。開発ボードでは1A程度の電源回路が載っている事が多い

・3端子レギュレータかDC-DC（降圧）を使う。DC-DCの方がはるかに楽だが，値段差も大きい。

（参考）DCDCは450円程度。レギュレータを使う場合は1回路に付き，計100円程度。

・3端子レギュレータを使う際は場合**パスコン（バイパスコンデンサ）が必要**。

⇒ 基本的に各製品の説明書に書いてある容量のコンデンサを使えば大体問題は無いらしい

・できれば静電気対策などの安全回路が欲しいが，無くても動作はする。

##### タクトスイッチ

タクトスイッチはそれぞれプルアップ(10kΩ程度)で接続

チャタリング防止の0.1μF程度のコンデンサもあると良い。（ついでにノイズ対策にもなる）

RSTボタン

別名：RES, EN

プログラムを頭からやり直す。

※ENはEnableの意で，基板の有効無効を切り替えだが，これでリセットとする基板もある。

PGMボタン

別名：BOOT, IO0

書込みモード／プログラム実効モード　を切り替える。

PGMがHIGH（ボタン開放）の状態でRSTボタンが押されると、実行モード。

PGMがLOW（ボタン押下）の状態でRSTボタンが押されると、書込みモード。

補足：チャタリングにはCRの方が効くらしい。

情報元：https://www.marutsu.co.jp/pc/static/large\_order/1405\_311\_ph

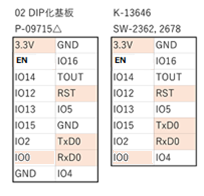
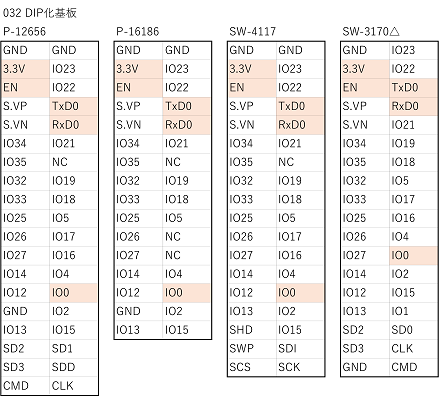
##### 変換基板

何種類かあるので注意。ランドピッチ（長辺）は全て2.54。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | コード | 製品 | 値段 | ピッチ | その他 |
| 秋月 | P-09715 | 02基板のみ | \120 | 7.62 |  |
| 秋月 | K-09758 | 02 DIP化 | \650 | 7.62 | P-09715の基盤 |
| 秋月 | K-13646 | 02 DIP化 | \580 | 27.94 | DEV: K-12236 ピッチ15.24 |
| 秋月 | P-16186 | 32基盤のみ | \120 | 22.86 |  |
| 秋月 | P-12656 | 32基盤のみ | \240 | 7.62 | コンデンサ付き |
| 秋月 | K-11755 | 32DIP化 | \900 | 7.62 | P-12656の基盤 |
| switch science | 2678 | 02基板のみ | \165 | 7.62 |  |
| switch science | 2362 | 02基板のみ | \165 | 27.94 |  |
| switch science | 2347 | 02DIP化 | \926 | 27.94 |  |
| switch science | 3170 | 32DIP化 | \1100 | 25.4 | 3210, 6364がDev　よしさん |
| switch science | 4117 | 32基盤のみ | \286 | 22.86 | U字型のものもある4116 |

7.62幅（×3）　22.86幅（×9）　27.94幅（×11）※ブレッドボードの端から端

ピン配置



説明の必要なピン

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ＥＮ | イネーブル端子 | 10kΩでプルアップ。通常はHI（使用可能）。  GNDに接続する事でLOWになりアンイネーブル状態になる。  ※ESP32の場合はこれをリセットボタンとしている |
|  |  |  |

#### 注意点など

・「USB to シリアル」で3.3V，1A 出ている製品が有れば良かったが，市販品で3.3Vを出力している物が少ない。あっても350mAの電流量制限。（ESPは起動時に？500mA必要と書いてある。）

・自前で電源回路を作るにも，ほとんどの「USB to シリアル」の製品には出力制限の無い5Vポートが無い。制限電流量も350mA。（この350mAという数字は何かお決まりの数値なの？）

・USBポートの金具を買ってきて5Vピンを使えないかと考えたが，市販されているUSBポートの金具にはDIP化された製品が無い為，実質難しい。

（おまけにこの方法ではシリアル通信ICを買ってきて，自分で回路を作る事になるので出来れば避けたい。）

⇒ 結果的に[これ](https://www.switch-science.com/catalog/2499/)しかなかった。良い製品が売っていればもっと楽に作れるはず。2021-06-26

⇒ これは3.3V出てるが電流量不明。使えるかも？　<https://akizukidenshi.com/catalog/g/gK-06503/>

#### 参考資料

開発ボードに載っている電子素子

TPS2065 パワー・ディストリビューション・スイッチ。出力電流の調整。

AP7361-33E-13

LESD5D5.0CT1G パワーコネクタ → バリスタらしい

BAT760-7 ショットキーダイオード

MICROSMD035F-2 リセッタブルヒューズ トリップ電流750mA(20℃)

BLM18PG600SN1 フェライトビーズ・インダクタ

RSX101VA-30 ショットキーバリアダイオード

AMS1117-3.3 ３端子電源レギュレーター

LXE11DAA2-137 USB電装ライン用フィルタ

LXDC DCDCコンバータ（村田製作所）

情報

（公式）<https://docs.espressif.com/projects/esp-idf/en/stable/index.html>

（参考）：[https://qiita.com/ycrack/items/bbd7f9f95376cca53241#windows](https://qiita.com/ycrack/items/bbd7f9f95376cca53241" \l "windowsの場合)の場合

メモ

EN ESP02の時に使用し、ENにつなぐ。（プルアップで常に3V）

RST ESP02の場合はRSTに、ESP32の場合はENにつなぐ。

PGM ESP32の場合に使用し、IO0につなぐ。

### MQTT

inclueライブラリ

→ArduinoStudioのスケッチ→ライブラリをインクルード→ツール.zip形式のライブラリをインストール

・PubSubClient.h MQTTメッセージ向きのクライアントライブラリ

（DL）<https://www.arduino.cc/reference/en/libraries/pubsubclient/>からダウンロード

・Cooperative Multitasking

（DL）https://www.arduino.cc/reference/en/libraries/cooperative-multitasking/

・MQTT Client

（DL）https://www.arduino.cc/reference/en/libraries/mqtt-client/

/\* MQTT割込： MQTT接続 \*/

void onMqttConnect(bool sessionPresent) {

}

/\* MQTT割込： MQTT切断 \*/

void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {

}

/\* MQTT割込： サブスクライブ \*/

void onMqttSubscribe(uint16\_t packetId, uint8\_t qos) {

}

/\* MQTT割込： パブリッシュ \*/

void onMqttPublish(uint16\_t packetId) {

Serial.println(packetId);

}

/\* MQTT割込： メッセージ受信 \*/

void onMqttMessage(char\* topic, char\* payload,

AsyncMqttClientMessageProperties properties, size\_t len, size\_t index, size\_t total) {

}

### 赤外線リモコン

IRremoteEPS8266

（Arduino Unoなど向きの「IRremote」の類似品）

（DL）https://www.arduino.cc/reference/en/libraries/irremoteesp8266/

### MP3再生

概要

・フラッシュRAM（4MBまで）にmp3を配置

SPIFFS領域 書き込むことのできる領域

内部DACを使用（2chのDACがある；GPIO25，26）

ファイルアップロード

（参考）<https://www.mgo-tec.com/blog-entry-spiffs-uploader-plugin-arduino-esp32.html>

前準備

音声再生ライブラリ？

・ESP8266Audioというライブラリを利用

インストール方法

cd ~/Arduino/libraries

git clone https://github.com/earlephilhower/ESP8266Audio

git clone https://github.com/Gianbacchio/ESP8266\_Spiram

公式：[https://github.com/earlephilhower/ESP8266Audio#installation](https://github.com/earlephilhower/ESP8266Audio" \l "installation)

~/Arduino/myproj/data　というディレクトリを作成し、そこにｍｐ３ファイルを配置

参考：<https://blog.techlab-xe.net/esp32-play-mp3sound/>

参考：<https://wak-tech.com/archives/923>

副次的な情報

音声の入手先 ニコニ・コモンズ

I2S接続のアンプボード MAX98357A

<https://www.marutsu.co.jp/pc/i/829728/>

・ESPMQTT Client

https://www.arduino.cc/reference/en/libraries/espmqttclient/

#include <MQTTClient.h>

#include "EspMQTTClient.h"

EspMQTTClient mqttClient(

"YOUR\_SSID",

"YOUR\_PASSWORD",

"MQTT\_BROKER\_NAME\_OR\_IP"

);

void setup() {

Serial.begin(115200);

delay(50); // Serial Init Wait

}

void onConnectionEstablished() {

Serial.println("onConnectionEstablished.");

mqttClient.subscribe("mytopic", [] (const String &payload) {

Serial.println(payload);

});

mqttClient.publish("mytopic", "This is a message");

}

void loop() {

mqttClient.loop();

}

（ソース参考）https://github.com/techiesms/ESP32-MQTT/blob/master/ESP32-MQTT.ino

（ソース参考）<https://qiita.com/dzonesasaki/items/358cfbac489258627cee>

（公式チュートリアル）<https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device?queryID=8372974954f48e26c3197023a447c2ac>

## 資料

APIリファレンス１：<http://www.musashinodenpa.com/arduino/ref/index.php>（基礎部分もある）

APIリファレンス２：<https://garretlab.web.fc2.com/arduino/esp32/reference/>（広く，詳しい）

## トラブルシューティング

### 共通

‘LED\_BUITIN’ was not declared in this scope

⇒ LED\_BUILTINを定義していない為のエラー。

#define LED\_BUILTIN 4 // ピン番号はこのように指定できる

もしくはLED\_BUILTINの部分を全て具体的な数値（4とか）に変える。

### Linux

exec: “python”: executable file not found in $PATH

⇒ pythonの実行ファイルのパスを通す。

which python // このコマンドの実行結果を確認

⇒表示されない：python(2)がインストールされていないので apt install python-is-python2

⇒表示される：export でそのパスを追加する　例：export PATH=$PATH:/usr/bin/python

ImportError: No module named serial

⇒ python(2) にpyserialが入っていない。Arduino Studio内部ではシリアル通信にpyserialを使っているらしい。しかもpython3では無い。

sudo pip -m install pyserial

could not open port /dev/ttyUSB0 permission denied

⇒ /dev/ttyUSB0 に読み書き許可が無い

ls /dev/ttyUSB0 -l # 現在の権限を確認

sudo chmod o+rw /dev/ttyUSB0 # その他ユーザーに読書き権限を付与（所有者はroot）

sudo chmod 777 /dev/ttyUSB0 # めんどうならばこれでも良い

# Google Coral

## 概要

機械学習(ML推論)用ASICのEdge TPUを搭載したシングルボードコンピュータ。

※TPU(TensorFlow Processing Unit)「TensorFlow」に最適化させたカスタムASIC(application specific integrated circuit、特定用途向け集積回路）

軽量の独自Linux、Mendel Linux を使用。

8GB以上のSDカード推奨（最大何Gまで？128Gは可能らしいが）

Arm NXP i.MX 8M SOC(4コア)　サポートOS：Debian

公式：<https://coral.ai/products/>

NVIDEAの対抗馬として開発。NVIDEAよりも小さい環境、モバイル機に入れるとか、衛星に入れるとか・・・Nvidiaが参入できない場面に切り込む様な商品

## チュートリアル

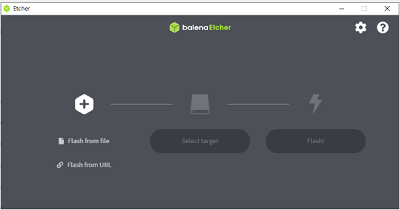
### イメージの書込み

以下からイメージをダウンロード

[https://coral.ai/software/#mendel-linux](https://coral.ai/software/" \l "mendel-linux)

<https://mendel-linux.org/images/enterprise/eagle/enterprise-eagle-flashcard-20211117215217.zip>

⇒ DDやbalena Etcherで書き込む。

Etcher

ddを使う場合

[ terminal @開発機 ]

lsblk # 先にデバイス名を確認する /dev/sdbとかdev/sdcとか行った具合

# 解凍しながら書き込み

# 注意：bsは指定が無いと恐ろしく遅くなる事がある為、ほぼ必須オプション。

unzip -p enterprise-eagle-flashcard-\*\*\*.zip | sudo dd of=/dev/sdc bs=1M status=progress

補足

| パイプ（Linux独自；コマンドの処理を別のコマンドに渡す）

ddコマンドの補足

of：書き込み先 bs：書き込み処理１回のブロックサイズ

status：=progressとする事で書き込み状況を表示できる

eMMCへの書込み

coralボードのdipスイッチを操作し、SDカードモードへ変更する。

SDカードモード【 1:on　2:off　3:on　4:on 】　 eMMCモード【 1:on　2:off　3:off　4:off 】

↓

次に、microSDを差し込み、coral boardに電源を供給。赤色LEDが点灯し、eMMCへの書込みが始まる。(5~10分ほどかかる) eMMCへ書込みが終わると、赤色LEDが消灯する。

↓

電源とmicroSDを抜き、eMMCモードに切り替える。

補足

・eMMC

≒基盤に組み込まれたSDカード。SDカードとeMMCの関係は携帯電話のSIMカードとeSIMに似ている。

・microSDが必要なのは最初の書き込み時だけ。内部に8GBのeMMCがあり、そこにbootファイルなどが保存される。それ以降のSDカードは、FAT32で初期化してファイル保存用のストレージなどに使うと良い。

### 最初のログイン

coralボードの初回起動は時間が掛かる（3～5分ほど）

その間にMDTを用意（開発機）

[ terminal @開発機 ]

python3 -m pip install mendel-development-tool # 全ユーザー向きにするなら--user をつける

mdt devices # 現在接続しているデバイスの確認

# > green-snail (192.168.100.2)など

mdt shell # shellを起動

mdt shell green-snail # 複数台使用している場合

> mendel@green-snail:~$ # ログイン成功した場合

補足：仮想環境を作って、仮想環境のpipにインストールするのが良い。

ログインついでに色々な確認する（任意）

[ terminal @coralボード ]

cat /etc/debian\_version # Debianのバージョン（任意）

cat /etc/lsb-release # ディストリビューションの詳細情報（任意）

systemctl get-default # 起動モードの確認（任意）

補足

coralボードへのログイン方法は３つある。

・mdt shellコマンドを用いる（上記）【USB type-C接続；パスワード不要】

・sshを用いる（後述）

・シリアルコンソールを用いる（後述）【USB Micro-B接続】

補足

・MDTの利用には、開発機側でバージョン3.7以降のpythonをインストールする必要がある。[2022.06.21現在]

・ここで使われている「192.168.100.2」というIPアドレスは、USB type-C接続時のみ利用可能。

・coralボードのホスト名（green-snailなど）はイメージの書込み枚に無作為に選択される。

・ルーター等を経由したLAN接続したい場合にはIPアドレスの設定が必要。

### インターネットへの接続

[ terminal @coralボード ]

nmtui # 設定ツールを起動

> Activate a connection（接続をアクティベートする）

# 補足：以下のようにコマンドで設定する事もできる（らしい）

nmcli dev wifi connect SSID password PASSWORD ifname wlan0

[ terminal @coralボード ]

# 試しにaptをupdate

sudo apt update

# could not ge lock /var/lib/dpkg/lock が出た場合

sudo rm /var/lib/apt/lists/lock # lockファイルを削除する

exit # ここまで出来たら一度ログアウトしておく。

sudo shutdown now # 電源を落としたい場合

補足

・nmtuiのついでにcoralデバイスのホスト名を変更しておいても良い。

・Mendel Linuxでは /sbin にしかshutdownコマンドが無い。その為shutdownコマンドには常にsudo が必須。

### ssh

#### 概要

ssh通信ができるとVSCodeでcoral機のコーディングができるようになる為、強く推奨。

sshで接続するには以下の２つの方法がある。

・鍵認証

・coralの設定を変更し、パスワード認証を許可する。

（google coral（Mendel Linux）はデフォルトではパスワード認証によるssh接続を禁止している。）

#### ネットワーク設定の確認

sshによる接続に先立ち、wifi経由での通信が有効かどうか確認する。

開発機とcoralボードの通信に必要な条件は以下の通り

・coralボードのwifi設定が終わっている。（又はcoralボードは有線LAN接続している）

・coralボードと開発機は同一ネットワーク内にいる。またはルーターでルート設定がされている。

・mDNS（BonjourやAvahi）が有効かつcoralボードのホスト名が分かっている。またはcoralボードのIPアドレスが分かっている。

⇒ 良く分からない場合は同一のルーターか同一のスマホ（テザリング）に接続する

[ terminal @開発機 ]

# IPアドレスが分かっている場合

ping 192.168.1.1 #通信可能かどうか確認

# mDNSが有効でホスト名が分かっている場合

ping green-snail.local # hostname.local

#### パスワード認証

パスワード認証の許可

USB type-Cで接続し、mdt shellでログイン。

（coral標準の鍵認証を使ったsshでログインしても良い）

[ terminal @coral ]

sudo nano /etc/ssh/sshd\_config # テキストエディタ「nano」を使う。（viでも良い）

[ /etc/ssh/sshd\_config ]

PasswordAuthentication yes # 56行目あたり

補足：nanoの使い方（抜粋）

Ctrl + X で編集を終了 → Y 保存；N 保存しない；C　終了のキャンセル

範囲を選んでからCtrl+K → 切り取り ；範囲選ばずにCtrl+K → 一行切り取り ；ペースト Ctrl + U

[ terminal @coral ]

sudo systemctl restart sshd # sshd（sshサービス）再起動

exit # ログアウト

⇒ USB type-Cを抜く

補足：sshdの最後のdはdaemonの意で≒サービス（バックグラウンドアプリ）

蛇足：悪魔のつづりはdemon。demonとは別。

sshでcoralボードに接続

[ terminal @開発機 ]

# ホスト名で接続する場合

ssh mendel@green-snail.local

# > パスワードを入力。パスワードは「**mendel**」

ここまでできたら次は[configファイル](#_configファイル)へ。

#### 鍵認証（coral標準）

鍵ファイル

「最初のログイン」で示した「mdt shellでのログイン」を行うと、自動的に非公開鍵(private key) が開発機に保存される。

鍵ファイルの保存先パス：~/.config/mdt/keys/mdt.key

# 鍵ファイルの確認（または、中身を見てみたい人は以下を実行）

cat ~/.config/mdt/keys/mdt.key

補足：mvコマンドはファイルの移動コマンドだが、ファイル名変更にも使える。

接続

[ terminal @開発機 ]

# -i オプションで非公開鍵を指定する。

ssh mendel@192.168.100.2 –i ~/.config/mdt/keys/mdt.key

鍵ファイルの再作成

秘密鍵を失くしたり、上書きしてしまった場合の復帰方法。

まずは[シリアルコンソール](#_シリアルコンソール)でcoralボードに接続。

[ シリアルコンソール@coral ボード ]

rm /home/mendel/.ssh/authorized\_keys # 公開鍵を削除

sudo service mdt-keymaster restart # 鍵管理サービス（デーモン）を再起動

exit # 一度coralボードからログアウト

⇒ USB Micro-Bケーブルを抜く → screen（シリアルコンソール）実行前の画面に戻れる

⇒ USB typeCケーブルで開発機とcoralボードを接続

[ terminal@開発機 ]

rm ~/.config/mdt/keys/mdt.key # 念のため、秘密鍵を削除（存在する場合）

mdt shell # 再度、mdt shellコマンドでログイン

# ⇒ Looks like you don’t have a private keyなどと表示され、秘密鍵が作成される

# ⇒ coralボードに接続したらすぐ、exitでログアウトする

ssh mendel@192.168.100.2 –i ~/.config/mdt/keys/mdt.key # sshにてログイン

（参考）[https://coral.ai/docs/dev-board/mdt/#recover-mdt-access](https://coral.ai/docs/dev-board/mdt/" \l "recover-mdt-access)

問題点

USB type-Cでの接続の場合、IPアドレスを**192.168.100.2から変更できない**。

複数のcoralボードを管理したい場合にこれでは「WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!」が発生する為に困ってしまう。

⇒ パスワード認証かssh-keygenによる鍵認証が良い

2022-07-11 IPアドレスの変更方法ある？

#### 鍵認証（ssh-keygen）

[ terminal @coral ]

ssh-keygen # 公開鍵(id\_rsa.pub)と秘密鍵(id\_rsa)のペアが作成される

補足：パスフレーズは空白を入力する事で「パスフレーズ無し」とする事も可能。但し、10文字程度のパスフレーズを設けた方がより安全ではある。

公開鍵(id\_rsa.pub)

通常サーバー側に置いておくファイルで、慣例的にauthorized\_keysというファイル名に変更して用いる。

このテキストファイルは複数行で構成されていて、各行が別々の「鍵情報」を保持する

⇒ つまり、ssh-keygen後の公開鍵（id\_rsa.pub）の処理は以下のうちのいずれかとなる：

・id\_rsa.pub（公開鍵）をauthorized\_keysに名前を変更（上書き）

[ terminal @coralボード ]

mv ~/.ssh/id\_rsa.pub ~/.ssh/authorized\_keys

・今回作成された公開鍵をauthorized\_keysに追加

[ terminal @coralボード ]

cat ~/.ssh/id\_rsa.pub # 内容を表示 → コピー

nano ~/.ssh/authorized\_keys # 下の行に今回の公開鍵情報を張り付けて保存する

⇒いずれの方法でも、一度ログアウト

[ terminal @coralボード ]

exit

↓

[ terminal @開発機 ]

# scpを使ってファイルをcoralボード⇒開発機にコピー

scp mendel@192.168.100.2:~/.ssh/id\_rsa ~/.ssh/id\_rsa.key –i ~/.config/mdt/keys/my\_rsa.key

# ホスト名を使う場合

scp mendel@green-snail.local:~/.ssh/id\_rsa ~/.ssh/id\_rsa.key –i ~/.config/mdt/keys/my\_rsa.key

# configファイル（後述）を作成している場合は以下のようにしてもよい

scp mendel@mycoral:~/.ssh/id\_rsa ~/.ssh/my\_rsa.key

補足：scp sshが有効な時に使えるファイル転送ソフト

USB type-Cを抜く

ping green-snail.local # WiFiを使った通信が可能かどうか確認する

# >応答がない場合：もう一度USB type-Cを接続し、coralボードにログインし、shutdown→電源を入れ直す。

# >応答があったら

ssh mendel@green-snail.local -i ~/.ssh/my\_rsa.key # sshでログインし直す

ここまでできたら次は[configファイル](#_configファイル)へ。

#### configファイル（ssh接続設定ファイル）

VSCodeでの接続に先立ち、sshの設定ファイル（configファイル）を作成する必要がある。

注意！configファイルを作る場所は決まっている

# LinuxやiOS

touch ~/.ssh/config

# Windows

New-Item C:\Users\UserName\.ssh\config

[ ~/.ssh/config @開発機 ]（例）

Host test # この設定に付ける名前。なんでも良い。

HostName green-snail.local # ホスト名.local。

User mendel # ユーザー名。coralボード初期値：mendel

# 鍵認証を行う場合は、鍵ファイルへのパスを記述。パスワード認証の場合は、この項目は不要

IdentityFile ~/.config/mdt/keys/mdt.key

補足：IPアドレスで接続しても全く問題はない

HostName 192.168.100.2 # 接続先のIPアドレス。

このファイルを作成しておくと、以下の様にしてsshログインが可能。

[ terminal @開発機 ]

ssh test # 設定ファイルを使ったsshログイン

また、VSCodeでRemoteDevelpmentの機能を使った開発が可能。

#### VSCodeからの接続

必要な拡張機能のインストール

拡張機能を表示： 又はCtrl+Shift+X → **Remote Development**  を選択してインストール。

CONTAINERS  のアイコンが追加される。（これでリモートデバイスの選択などを行う）

接続

をクリックしてSSH Targetsを開く

→ configで設定した名前（上の例だとtest）の左にある新しいウィンドウで開く  をクリック

（サーバー[接続先]OSの種類を問われる事がある）

右下に と表示（初回は3～5分くらいかかる）

画面の左下が  →  となる。

### 公式カメラの接続方法

coralの電源を落とす

公式チュートリアル：[https://coral.ai/docs/dev-board/camera/#connect-the-coral-camera](https://coral.ai/docs/dev-board/camera/" \l "connect-the-coral-camera)

### 物体識別のデモ

公式のGithubで物体識別のテストができる

# 作業用のディレクトリの作成＋移動

mkdir google-coral

cd google-coral

# Githubのリポジトリをクローン（ダウンロード）

git clone https://github.com/google-coral/examples-camera.git --depth 1

# モデルダウンロードのスクリプトが存在するディレクトリへ移動＋実行

cd examples-camera

sh download\_models.sh

# OpenCVインストールのスクリプトが存在するディレクトリへ移動＋実行

cd opencv

bash install\_requirements.sh

参考：[https://github.com/google-coral/examples-camera/tree/master/opencv#set-up-your-device](https://github.com/google-coral/examples-camera/tree/master/opencv" \l "set-up-your-device)

注意点

USBカメラの場合、detect.pyに下記の修正が必要（カメラデバイス番号を1にする必要がある。）

[ examples-camera/opencv/detect.py ]

# 61行目

cap = cv2.VideoCapture(args.camera\_idx)

↓

cap = cv2.VideoCapture(1)

実行

[ terminal@coral ]

cap = cv2.VideoCapture(1)

補足

公式が用意しているデモも存在する。

edgetpu\_demo # フルパス：/usr/bin/edgetpu\_demo

bashで書かれたスクリプトで、内部で/usr/bin/edgetpu\_detectというpythonスクリプトを実行している。

環境変数でモデルファイルなども選択できるようなので、これで物体識別を行っても良い？

### モデル作成

#### 共通情報

・Edge TPUは8bitで量子化される為、モデルも8bitで作成される必要がある。

以下のようにTensorFlowのモデルからエキスポートされる。

TensorFlowモデル(32bit float) → Frozen graph(.pbファイル) → TensorFlow Lite(8bit) → Edge Tpu model(.tflite file) → Coral機器

対応モデル

classifier（画像分類）

MobileNet V2 入門編にはよさそう。V3が存在するらしい？

MobileNet V1 (quant-aware)

detector（画像認識、物体検知）

EfficientDet-Lite 小型化で高精度を謳ったモデル。20以上のクラス分類には非対応。

SSDLite MobileDet 再学習らしいので、時間が短くて済むはず。

SSD MobileNet v1 (quant-aware)

補足

quant-aware

量子化（bit数変換）処理の内、誤差などを自動的に修正して最適化する手法

公式資料：<https://www.tensorflow.org/model_optimization/guide/quantization/training?hl=ja>

結論：EfficientDet-Lite、SSDLite MobileDet、SSD MobileNet v1、MobileNet v2、の順で行う。

公式情報：<https://coral.ai/docs/edgetpu/models-intro/>

#### 通常の方法

公式チュートリアルが存在する

<https://www.tensorflow.org/lite/tutorials/model_maker_image_classification?hl=ja>

画像は最低でも数百枚くらい必要らしい。公式に載っているのはアーカイブされたファイルを利用する手法。

image\_path = tf.keras.utils.get\_file(

'flower\_photos.tgz',

'https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz',

extract=True)

image\_path = os.path.join(os.path.dirname(image\_path), 'flower\_photos')

補足：tgz=tar+gzらしい。

画像データは以下のような構成になる（らしい）

Image\_photo\_data

├ class1

│ ├ class1\_image001.jpg

│ ├ class1\_image002.jpg

│ …

├ class2

…

import os

import numpy as np

import tensorflow as tf

from tflite\_model\_maker import model\_spec

from tflite\_model\_maker.image\_classifier import DataLoader

# 画像ファイルの読み込み

image\_path = tf.keras.utils.get\_file(

'flower\_photos.tgz',

'https://storage.googleapis.com/download.tensorflow.org/example\_images/flower\_photos.tgz',

extract=True)

image\_path = os.path.join(os.path.dirname(image\_path), 'flower\_photos')

# 学習データとテストデータを作成

data = DataLoader.from\_folder(image\_path)

train\_data, test\_data = data.split(0.9)

# ここでモデル作成（GPUを使用する）

model = image\_classifier.create(train\_data)

#### EfficientDet-Lite

##### 概要

・EfficientDet-LiteのデータセットはPascal VOC XMLの書式で作成する。

・labelImgなどの一般のアノテーションツールでも作成ができる。

・EfficientDet-Lite は20以上のクラスを含むモデル作成には向かない。

公式：<https://colab.research.google.com/github/google-coral/tutorials/blob/master/retrain_efficientdet_model_maker_tf2.ipynb>

##### labelimgを利用する場合

１　データ作成

labelimg作業ディレクトリを以下のように構成しlabelimgを起動

OpenDirでdataset\imgasesを指定。Change Save Dirでdataset\annotationsを指定。

dataset\

├ images 画像ファイルを格納するディレクトリ

│ ├ P0001.jpg

│ ├ P0002.jpg

├ annotations xmlファイルを格納するディレクトリ。最初は空っぽで良い。

↓　作業終了後のファイル

dataset\

├ images

│ ├ P0001.jpg

│ ├ P0002.jpg

├ annotations

│ ├ P0001.xml

│ ├ P0002.xml

補足：imagesファイルにxmlファイルを格納しても良いらしい（colab上で設定必要）

補足：画像ファイルの拡張子は.jpgに限定される。（？完全に判明したわけではない2022-08-31）

２　datasetディレクトリを丸ごとzip圧縮する。（ファイル名は「dataset.zip」とする）

３　ここ<https://coral.ai/docs/edgetpu/models-intro/>からgoogle colaboratoryのRetrain EfficientDet-Liteを開く。

セッションストレージにアップロード、またはドラッグ＆ドロップで１で作成した.zipファイルをアップロード。

４　use\_custom\_dataset:True　dataset\_is\_split:False で(Optional) Load your own Pascal VOC datasetを実行

５　len(画像ファイル) >= batch\_size とする必要がある

メモ

CorabのGPUの使用

メニュー：ランタイム > ランタイムのタイプを変更 > ハードウェアアクセレレータを「GPU」に。

次ページに続く

～EfficientDet-Litのデータセット作成　labelimgを利用する場合　続き

補足：本来であれば以下のようなディレクトリ構成にすべき

dataset\

├ train 学習データ

│ ├ images

│ ├ annotations

├ validation ニューラルネットワークのパラメータの良し悪し確認の検証データ

│ ├ 同様にannotations、imagesディレクトリを格納

├ test 学習後に汎化性能を確かめる為に使う

│ ├ 同様にannotations、imagesディレクトリを格納

##### 識別

１枚の画像を識別

python3 detect\_image.py \

--model efficientdet-lite-salads\_edgetpu.tflite \

--labels salad-labels.txt \

--input salad.jpg \

--output salad\_result.jpg

##### 詳細

###### split\_datasetメソッド

split\_dataset(

images\_path= , images\_in images\_in = 'dataset/images'

annotations\_path = annotations\_in, annotations\_in = 'dataset/annotations'

val\_split = 0.2, 0~1

test\_split = 0.2, 0~1

out\_path = 'split-dataset'):

### バックアップ

#### シリアルコンソール

内部eMMCのバックアップに先立ち、シリアルコンソールでcoralボードにログインする



Micro-BのUSBと開発機PCを接続。（右図参照）

↓

[ terminal@開発機 ]

sudo apt list --installed | grep screen # screenが入っているか確認

sudo apt install screen # screenが入っていない場合はインストール。

sudo usermod -aG plugdev,dialout <username> # 必要な補助グループに追加。（この処理、不要かもしれない）

reboot # （開発機の）再起動

# ここでcoral ボードに電源を投入

dmesg | grep ttyUSB # USBのデバイス名を確認

screen /dev/ttyUSB0 115200 # シリアルコンソールで接続

[ terminal@coral シリアルコンソール ]

# > 最初は空白画面が表示される→Enterを押す

# ユーザー名を入力する画面になる。⇒ ユーザー名もパスワードもmendel

# 終了する時

sudo shutdown now

# > reboot: Power down と表示されたら、Micro-BのUSBを抜くと元の画面に戻れる

トラブルシューティング

u-boot ⇒ U-Bootモードに入ってしまった。再起動した方が早い。

補足：起動処理初期にキー入力を行うとu-bootモードに入る

補足：ums 0 mmc 0 とするらしい？

[screen is terminating] ⇒ Micro-BのUSBを一回外して付け直す

資料

（公式；serial-console）<https://coral.ai/docs/dev-board/serial-console/>

（参考）<https://stackoverflow.com/questions/64176284/backing-up-and-restoring-the-emmc>

#### バックアップ

バックアップ

補足：シリアルコンソールで行う。

[ terminal@coral シリアルコンソール ]

# 8Gb以上のSDカードを挿入

lsblk # 現在接続している外部記憶デバイスを調べる

# > 内部eMMCが「mmcblk0」 SDカードがmmcblk1 であったとする

# DDを使ってバックアップ

sudo dd if=dev/mmcblk0 of=/dev/mmcblk1 bs=4M status=progress

（補足）USBメモリにバックアップしても問題ない

復元

補足：こちらもシリアルコンソールで行う。

[ terminal@coral シリアルコンソール ]

lsblk

# > 内部eMMCが「mmcblk0」 SDカードがmmcblk1 であったとする

# coralの内部eMMCにコピー

sudo dd if=/dev/mmcblk1 of=/dev/mmcblk0 bs=4M status=progress

#### Demo

Webブラウザ版

edgetpu\_demo --stream

開発PCのWebブラウザで192.168.100.2:4664にアクセスする。

cannot open display: と出たら

sudo systemctl restart weston

#### 補足情報

設定ツールfastbootをインストール（不要かも？

sudo apt-get install fastboot

Macの場合、Android SDK platform toolsを利用する。

Windowsの場合、GitBashターミナルの利用を推奨

その際、pythonの直接実行ができないため、以下のコマンドを実行する必要がある。

echo "alias python3='winpty python3.exe'" >> ~/.bash\_profile

source ~/.bash\_profile

#### トラブルシューティング

##### ロック /var/lib/dpkg/lock が取得できませんでした

lockファイルを削除する

rm /var/lib/apt/lists/lock

##### SOMEONE IS DOING SOMETHING NASTY!

sshでの接続時に過去のホスト情報と違う場合に発生する。

~/.ssh/knownhostsを編集する。分からなければ、このファイルを一度削除する。

## その他情報

### Tensorflow Liteのインストール

$ wget https://dl.google.com/coral/python/tflite\_runtime-1.14.0-cp37-cp37m-linux\_aarch64.whl

$ pip3 install tflite\_runtime-1.14.0-cp37-cp37m-linux\_aarch64.whl

### カメラの接続

USBカメラの接続

v4l2-ctl --list-formats-ext --device /dev/video1

## 詳細

### 標準のパーティション構成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| パーティション | フォーマット | サイズ | シンボル | 補足 |
| mmcblk0p1 | ext2 | 128MiB | boot |  |
| mmcblk0p2 | 不明 | 1.00MiB | misc |  |
| mmcblk0p3 | ext4 | 2GiB | home |  |
| mmcblk0p4 | ext4 | 5.13GiB | rootfs |  |

### python3.9のインストール

pythonのソースコードは以下のWebストレージでダウンロード可能

<https://www.python.org/ftp/python/>

#### インストール

[ terminal ]

# 必要パッケージのインストール

sudo apt update

sudo apt install python3-dev python3-pip zlib1g-dev libssl-dev

# python3.9 のダウンロードと展開

wget https://www.python.org/ftp/python/3.9.2/Python-3.9.2.tar.xz

tar xvf Python-3.9.2.tar.xz

# コンパイルとインストール

cd Python-3.9.2 # ダウンロードしたディレクトリへ移動

./configure # 環境の確認とmakefileの作成

make –j4 # コンパイル　–jオプション；ジョブの数を指定

sudo make altinstall # インストール

/usr/local/bin/python3.9 にインストールされる

補足

なお、この手順は新バージョンのpythonをインストールする時にはお決まりの手順（らしい）

#### pip3.9のインストール

get-pip.pyというスクリプトを使う。見つからない場合は以下のどちらかで入手

# bootstrapから入手

Wget https://bootstrap.pypa.io/get-pip.py

# PypaのGithubリポジトリで入手

wget https://github.com/pypa/get-pip/raw/master/get-pip.py

# pip3.9のインストール

sudo python3.9 get-pip.py

# 使い方例：python-venvのインストール

sudo pip3.9 install python3.9-venv

#### 標準のpythonを変更

例えばpython3.9を新しくインストールしたとする

sudo unlink /usr/bin/python3 # 不要かも 2023-03-24

sudo rm /usr/bin/python3 # 不要かも 2023-03-24

sudo ln -sf /usr/bin/python3.9 /usr/bin/python3

補足

ln 書式： target\_path ← link 引数は対象が先、リンクが後

オプション：-s シンボリックリンクを作成；-f既存のファイルを強制的に上書き

こんな方法がある？

sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.9

# update-alternatives: --install needs <link> <name> <path> <priority>

#### トラブルシューティング

# make実行時のエラーメッセージを確認

sudo apt make --debug

補足： python3-venvのインストールがpython3.9のインストールより早いと

# make実行時のエラーメッセージを確認

make --debug

# 不足するパッケージの追加インストール

sudo apt install <lib\_name>-dev

zlib not available

# zlibをインストール

sudo apt install zlib1g-dev

/usr/include/zlib.h　が本体らしい

### SDカード /home/ディレクトリ化

内部eMMCは8Gしか容量が無いので対応が必要となる。色々方法はあるが、/homeディレクトリをSDカードのパーティションに移動するのが良策。

以下、具体的な手順：

# パーティションとファイルシステムを作成（フォーマット）

# 補足：マウント先のシステムのterminalでやったほうが無難

sudo fdisk /dev/mmcblk1 # 対話形式でパーティション作成

Command (m for help): d → 1 # パーティション番号1を削除

Command (m for help): n　→ p → 1 # primary のパーティション1を再作成

# First sectorとLast sector は空白（default）で良い

Command (m for help): p # 念のため変更内容を確認

Command (m for help): w # 変更内容の適用（fdiskが終了）

# フォーマットからマウントまで

sudo mkfs.ext4 /dev/mmcblk1p1 # フォーマット

sudo mount /dev/mmcblk1p1 /mnt # マウント

# ファイルがホームディレクトリにすでに存在する場合は、SDカードにコピーする

sudo rsync -aX --exclude='/\*/.gvfs' /home /mnt # コピー

sudo diff -r /home /mnt/home -x ".gvfs/\*" # コピーが正しく終わった事を確認

# もとの/homeのファイルを削除

sudo rm -rf /home/\*

# SDカードを一度アンマウントして、homeディレクトリとしてマウントし直す

sudo umount /dev/mmcblk1p1

sudo mount /dev/mmcblk1p1 /home/

# SDカードのUUIDを取得

lsblk --output NAME,UUID,FSTYPE

/etc/fstabを編集して自動マウントするようにする

[ /etc/fstab ]

# この行を削除しないとうまく行かないかも

#/dev/mmcblk0p3 /home ext4 noatime,nosuid,nodev,defaults 0 3

# この行を追加する

UUID={ID} /home ext4 defaults 0 3

補足

・mmcblk1とmmcblk1p1では意味が異なるので注意！

資料

mendelのホームディレクトリが無くなってしまった場合

usermod -d /home/mendel –m mendel # ホームディレクトリを再設定

sudo chown -R mendel:mendel /home/mendel # 所有者の変更

Web資料では元のホームディレクトリとのデータ同期を以下のようにやっていた

# ファイルがホームディレクトリにすでに存在する場合は、SDカードにコピーする

sudo rsync -aXS --exclude='/\*/.gvfs' /home /mnt # コピー

sudo diff -r /home /mnt/home -x ".gvfs/\*" # コピーが正しく終わった事を確認

sudo chown -R mendel:mendel /mnt/home/mendel # 所有者の変更

主たる情報源１：<https://github.com/f0cal/google-coral/issues/61>

主たる情報源２：[https://ja.linux-console.net/?p=2430#gsc.tab=0](https://ja.linux-console.net/?p=2430" \l "gsc.tab=0)

### swap領域を増やす

概要

Webアプリなどを作成するとkswapd0のCPU使用量が上がってくる

これはスワップ管理をするデーモンなので、スワップ領域を増やす事で解決する（？）

公式によるとRAMが2GB未満の場合：RAMの2倍という事なのでここでは2GBにする。

（google coralのRAMは1GB）

具体例

スワップ用パーティションの作成では無く、スワップファイルでの対応

# 現在のスワップ領域の設定を確認する

swapon -a

# スワップファイルを作成する

# 補足 ddでは　bs×count でファイルをコピーする

sudo dd if=/dev/zero of=/swapfile bs=1M count=4096 # /(root) に直接作成

chmod 600 /swapfile # 権限の変更

# フォーマットしてをスワップを有効化

sudo mkswap /swapfile

sudo swapon /swapfile # 取り消す時はswapoff /swapfile

次回ログイン時にも有効になるように下記の設定ファイルを変更

[ /etc/fstab ]

/swapfile swap swap sw 0 0 0 # システムがswapと認識できる

補足：fallocateを使った方法を示すWebサイトがあるが、fallocateは非推奨

無事作成されたかどうかを確認する

[ terminal ]

free -mh

資料

推奨スワップ領域

RAMが2GB未満の場合：RAMの2倍

RAMが2GB～8GBの場合：RAMと同量

RAMが8GB～64GBの場合：4GBからRAMの0.5倍

RAMが64GB以上：負荷に依存するが，少なくとも4GB

fstabの他の設定方法

/swapfile swap swap sw defaults 0 0 # optionをdefaultに

/swapfile none swap sw 0 0 0 # マウントポイントをnoneに　何のために？

### X window

* + 1. Xorgが最初から入っている
    2. DISPLAY環境変数が :10.0なので:0.0に変更する必要がある。

補足

* + 1. weston/wayland based display

### メンテナンス関連

OSのバージョン確認

cat /etc/mendel\_version

### 公式の対応モデル

classifier

画像分類

移った画像を何らかの主題に基づき分類する。1枚の画像に対しDog98%といった結果を得られる。

MobileNet V2 入門編にはよさそう。V3が存在するらしい？

MobileNet V1 (quant-aware)

detector

画像認識、物体検知

マルチクラスの物体検出を行う。

SSDLite MobileDet 再学習らしいので、時間が短くて済むはず。

EfficientDet-Lite 元々、小型化で高精度を謳ったモデル

SSD MobileNet (quant-aware)

quant-awareの意味

<https://www.tensorflow.org/model_optimization/guide/quantization/training>

post-training quantization

quantization aware training

LSTM；Long Short Term Memory

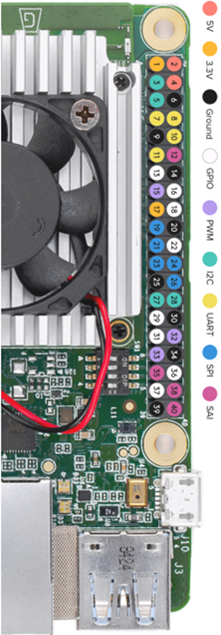
### その他モデル

#### YOLO

<https://github.com/guichristmann/edge-tpu-tiny-yolo>

### Github

### GPIO

公式資料：[https://coral.ai/docs/dev-board/gpio/#header-pinout](https://coral.ai/docs/dev-board/gpio/" \l "header-pinout)

#### 概要

ピン配置はラズパイとほとんど同じ。

違う点

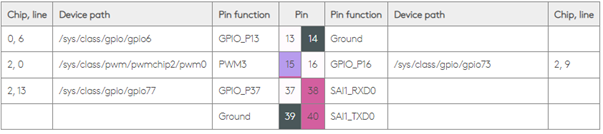
・PWM対応ピン

制御はpythonのperipheryというライブラリで行う。

<https://pypi.org/project/python-periphery/>

<https://python-periphery.readthedocs.io/en/latest/index.html>

[公式資料](https://coral.ai/docs/dev-board/gpio/" \l "program-with-python-periphery)（中略）のChip番号、line番号が重要らしい



[ python ]

from periphery import GPIO

gpio\_p16 = GPIO("/dev/gpiochip2", 9, "out")

gpio\_p37 = GPIO("/dev/gpiochip2", 13, "out")

[ bash ]

ls /dev/ | grep gpio

# gpiochip0 ~ gpiochip4 が表示される

具体例

[ python ]

from periphery import GPIO

led = GPIO("/dev/gpiochip2", 13, "out") # pin 37

button = GPIO("/dev/gpiochip4", 13, "in") # pin 36

try:

while True:

led.write(button.read())

finally:

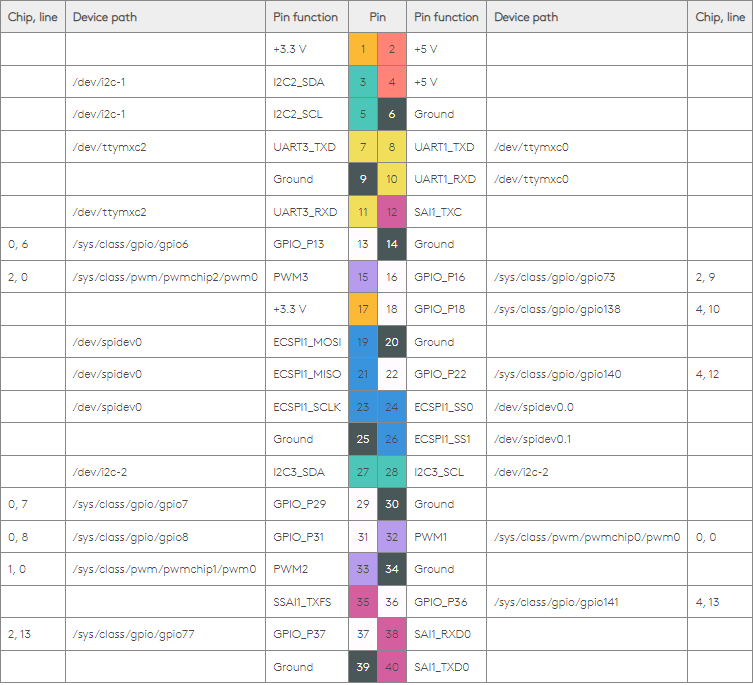
led.write(False)

led.close()

button.close()

#### I2C

公式：[https://coral.ai/docs/dev-board/gpio/#i2c](https://coral.ai/docs/dev-board/gpio/" \l "i2c)



### Devicetree Overlay

dtsファイルは以下の以下公式資料を参考にする（というか、コピペ）

dtsファイルをdtbにコンパイルする方法も以下公式資料を参考にする

<https://coral.googlesource.com/docs/+/refs/heads/master/DeviceTreeOverlays.md>

<https://github.com/armbian/sunxi-DT-overlays/blob/master/examples/gpio-button.dts>

[ terminal ]

sudo mv sample.dtbo /boot

[ /boot/overlays.txt ]

# List of device tree overlays to load. Format: overlay=<dtbo name, no extenstion> <dtbo2> ...

overlay= sample

参考：https://stackoverflow.com/questions/63928580/how-do-i-rebuild-coral-mendel-with-a-devicetree-modification

2023-01-06うまくいかず

## opencv

<https://github.com/google-coral/examples-camera>

<https://github.com/google-coral/examples-camera/tree/master/opencv>

カメラデバイスの確認方法

v4l2-ctl --list-devices

sudo apt install python3-opencv

容量不足対策

<https://qiita.com/rhene/items/4419ef08b85e697fe8c0>

## 管理

## リファレンス

### MDTコマンド

[https://coral.ai/docs/dev-board/mdt/#mdt-commands](https://coral.ai/docs/dev-board/mdt/" \l "mdt-commands)

## 資料

Mendel Linuxのソースコード

https://coral.googlesource.com/docs/+/refs/heads/master/GettingStarted.md

<https://coral.ai/models/image-classification/>

・RaspberryPIOSでは、/boot/overlays/ 下に様々なdtboファイルがあった

・一部のサブシステムでは、devicetree / bindings /の下に概要バインディングの説明がある

## メモ

・sdカードは「/dev/mmcblk」として認識される模様。/dev/mmcblk1が1つ目のパーティション。

# Jetson

## 基本情報

### 概要

NVIDIA（ビデオボードのメーカー；GPUが得意）が展開しているシングルボードコンピュータ。

GPUを多数積んでおり、画像認識などのAI演算の用途で用いる。（アクセラレータ）。

※GPU；Graphics Processing Unit　画像処理に特化したCPU

用途

・USB、または専用のカメラコネクタにカメラを接続した画像認識用途が一般的。

RaspberryPiの純正カメラが流用できる。

・TensorFlow、Pytorchなどがインストール可能。

・ROS；Robot Operating Systemをインストールする事もできるらしい。

（参考）<https://qiita.com/karaage0703/items/aa54e086f0a2f165d3e9>

情報

（モジュール比較）<https://developer.nvidia.com/embedded/jetson-modules>

（公式）<https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/>

（公式チュートリアル）<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit>

（参考）<https://www.pref.saitama.lg.jp/documents/201701/20210719_jetson.pdf>

### tensorRT

NvidiaのGPUを効率的に使えるようにしたライブラリ。

通常JetPackに含まれるのでインストールの必要はない。

場所：/usr/src/tensorrt

インストール方法

sudo apt install nvidia-tensorrt

参考：<https://qiita.com/tsutof/items/f81d3900fa77d954ef39>

### JetPack

#### バージョン確認

まずはデバイスのリリースバージョンを確認

cat /etc/nv\_tegra\_release

R\*とREVISIONの部分を足したもの　例：R32 (release), REVISION: 7.3 → 32.7.3

以下でもよいらしい

dpkg-query --show nvidia-l4t-core

デバイスのリリースバージョンからJetPackのバージョンが分かる公式：<https://developer.nvidia.com/embedded/jetpack-archive>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| デバイス | Ubuntu | JetPack | Cuda | Tensorrt | 対象機器 | 備考 |
| L4T 32.7.3 | 18.4 | 4.6.3 | 10.2 | 8.2.1.9-1 | Xavier NX Series,TX2 Series, AGX Xavier Series, Nano, TX1 |  |
|  | 20.4 |  | 11.8 |  |  |  |

アップグレード

sudo apt show nvidia-jetpack

<https://developer.nvidia.com/deepstream-getting-started>

#### インストールする場合

[ /etc/apt/sources.list.d/nvidia-l4t-apt-source.list ]（確認）

deb https://repo.download.nvidia.com/jetson/common r32.7 main

deb https://repo.download.nvidia.com/jetson/t210 r32.7 main

sudo apt install nvidia-jetpack

## 製品

### AGX Orin

24万円程度

GPU 2048 基の NVIDIA® CUDA® コアと 64 基の Tensor コア搭載

CPU 12 コア Arm Cortex-A78AE v8.2 64 ビット CPU　3 MB L2 + 6 MB L3

メモリー 32 GB 256 ビット LPDDR5 204.8 GB/秒

ストレージ 64 GB eMMC 5.1

開発者キットあり

### AGX Xavier

15万円程度

GPU 512-core NVIDIA Volta™ GPU with 64 Tensor Cores

CPU 8-core NVIDIA Carmel Arm®v8.2 64-bit CPU　8MB L2 + 4MB L3

Memory 32 GB 256-bit LPDDR4x　136.5GB/s

Storage 32 GB eMMC 5.1

開発者キットあり

### TX2 NX

5万円程度

GPU 256 Pascal cores (最大1.3GHz)

CPU Dual Denver and quad A57 CPU (最大2GHz)

メモリー 4 GB 128-bit LPDDR4 51.2GB/s

ストレージ 16 GB eMMC

wifiのボードが最初からついている

参考

<https://www.macnica.co.jp/business/semiconductor/articles/nvidia/133162/>

次ページに続く

### Nano

4GB　2万円程度

2GB　1万円程度

開発者キット

GPU 128-core Maxwell

CPU Quad-core ARM A57 @ 1.43 GHz

Memory 4 GB 64-bit LPDDR4 25.6 GB/s

Storage microSD (not included)

Jetson nano Moduleの場合はeMMC(64GB)

Wi-Fiは別途カードが必要

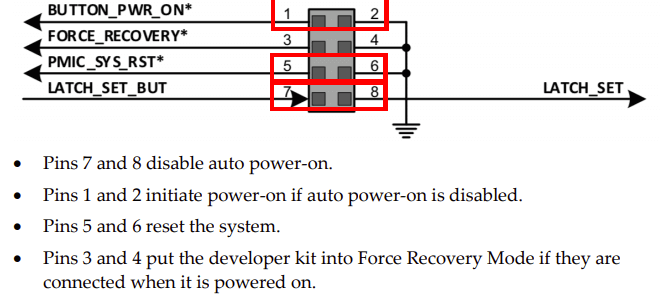
USBのアンテナが楽で良い。

PCIeカードの場合はIntel Dual Band Wireless-AC 8265 ワイヤレス　など、Intel系が良いらしい。

### Xavier NX

６万円程度（販売終了？）

参考：<https://www.macnica.co.jp/business/semiconductor/articles/nvidia/133162/>

電源ボタン

## 周辺機器

### 筐体

#### MIC-710AIL-00B1

Nano/ Xavier NX込み

11万くらい

#### Jetson Xavier NX用アルミケース

マルツ、スイッチサイエンスなどで販売

Nano用のものも存在する

#### Jetson Mate

Jetson Nano/NXを4台まで収納できる

秋月、マルツで販売

23,480円

#### Jetson Nano Metal Case/Enclosure

秋月、マルツで販売

2000円くらい

<https://www.marutsu.co.jp/pc/i/35107380/>

### 一般情報

microUSB type B、またはACアダプタから給電。

5V/3.0~3.5Aが推奨。

### 冷却装置

### カメラ

### カメラ付きキット

#### Baumer　AXシリーズ

46万くらい

Sony IMX265/Sony IMX250 を使用

CMOSセンサとJetson 間のデータのボトルネックを無くして高速化？

（公式；英語）<https://www.baumer.com/us/en/product-overview/industrial-cameras-image-processing/industrial-cameras/ax-series/c/43083>

#### ADLINK　 NEON-2000-JNX Series

40万くらい

Jetson TX2

IP67

<https://www.adlinktech.com/Products/Machine_Vision/SmartCamera/NEON-2000-JNX_Series?lang=ja>

#### ADVANTECH

AMV-RN01 詳細不明

#### ミルキューブ

詳細不明

[](https://milcube.com/)https://milcube.com/

## チュートリアル(Jetson Nano Dev Kit)

JETSON SDKをダウンロードしてEtcherやDDで書き込む。

「JetPack」という情報もあるが？

### 必要機器

・wifiアンテナ

wifiボード（wifiはG; 2.4GHzしか対応していない）

Intel 8265NGW M.2（エムドットツー；Eキー）

NGFF M.2 アンテナ ピグテール ケーブル

⇒ アマゾンくらいでしか売ってない？

USBタイプ

USBタイプでも良い。WDC-867DU3S2、WDB-867DU3Sなど；

USB3.0対応、2.4/5GHz両方対応が無難

<https://www.elecom.co.jp/category/cat_wifi-adapter/?category=client-personal>

・電源； 5VDC；2A以上

typeC又はアダプタ

アダプタ：<https://akizukidenshi.com/catalog/g/gM-06841/>

USB：<https://akizukidenshi.com/catalog/g/gM-17092/>

・SDカード 64GB以上、USH-1以上を推奨

・HDMI(2.0)ケーブル

eDP 1.4でも良いらしいが、普通にHDMIで良い

参考：<https://rt-net.github.io/tutorials/jnmouse/driver/install.html>

### ハード設定

ACアダプタを使う場合 J48のジャンパーピンのDC EN とD63を短絡する

USB給電を行う場合 J48のジャンパーピンを取り外す。

### OS設定

#### インストール

公式：<https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit>

イメージのダウンロード

<https://developer.nvidia.com/jetson-nano-sd-card-image>

#### 初期設定（ヘッドレス）

手順

１　先に開発機を起動しておき、USB type b(mini) のポートと開発機を接続する。

２　SDカードを挿入し、Jetsonに電源を入れる

→ 正しく接続された場合はL4T-READMEというデバイスが開発機にマウントされる。

上手く行かない場合は以下を試す

- USBケーブルを差し直す

補足：この状態は有線LAN（実際にはusb仮想）としても接続している状態で、IPは192.168.55.1

補足：USBの無線LANレシーバーは、初期ログイン時は抜いておいた方が良いかも

以下のコマンドでボーレートなどが確認できる

[ terminal@開発機 ]

dmesg | grep tty

注意点

・空白画面のまま変わらない場合はまだ起動中な可能性があるので少し待つ。

・Jetsonが完全に起動し終わってUSBを接続すると、ずっと白い画面のままになってしまい何も出来ない？

次ページに続く

～初期設定（ヘッドレス）　続き

３　L4T-READMEが認識されたらシリアルコンソールでログイン

[ terminal@開発機 ]

# 接続

sudo screen /dev/ttyACM0 115200 # なぜかsudoが必要

# screenが入っていない場合

sudo apt install screen

sudo chmod o+wr/dev/ttyACM0 # 権限が必要な場合があるかも

# 他にUSB機器を接続していない場合はこれでも良いらしいが？

ls /dev/cu.usbmodem\*

・**空白画面が表示されたらEnterを押す**。設定ウィザードが表示されるので、それに従って設定する。

・設定ウィザードでネットワークを設定するとうまく行かないので、ネットワーク設定は飛ばして先に進む。

（dummyを選択 ⇒ Do not configure the network at this time）

⇒ 一度ログアウトされる。（L4T-READMEがアンマウントされ、再度マウントされる）

⇒ 再度シリアルコンソールでログインする

４　ネットワークの設定

後述

#### ネットワークの設定

※USBの無線LANレシーバーの場合はここで取り付ける

[ terminal（シリアルコンソール）]

nmcli r wifi on # wifi をオン

nmcli d wifi list # 有効な接続の確認 nmcli device wifi listでもよい？

sudo nmcli d wifi connect YOUR\_SSID password XXXX # 接続

⇒補足：sshでログインできるのでssh user\_name@jetson.localなどとしてログインする。

補足：nmtuiを用いても良いが、シリアルコンソール接続中だと画面ズレが生じて文面が見づらい。

参考：https://zenn.dev/tetsu\_koba/articles/b4495387488df5

補足

　usbシリアルで接続中はssh nvidia@192.168.55.1でログインできる。（ただし初期のユーザー名はnvidiaではないらしい；2023-04-06）

jetson@192.168.55.1という情報もあったが、接続できず 2023-05-28

2023-04-06 接続できる事は確認できたが、うまくいかない場合はホストマシンを192.168.55.100に設定する。

#### ホスト名変更

/etc/hostsと/etc/hostnameの中身を変更すれば良いが、少し煩雑であったり、失敗する事もあるのでhostnamectlコマンドがお勧め。

[ terminal ]

hostname # 現在のホスト名を確認

hostnamectl set-hostname [変更後のホスト名] # ホスト名変更；パスワードを求められる

reboot

#### ユーザー名の変更（任意）

場合によっては自動ログインの解除が必要

GUI：設定 > 詳細 > ユーザ

ラズパイを参考に

[ terminal @Jetson（sshまたはシリアルコンソール）

sudo useradd -M tmp # -M オプション；ホームディレクトリを作らない

sudo gpasswd -a tmp sudo # tmpユーザをsudoグループに追加

sudo passwd tmp # パスワード変更　※パスワードの入力文字は表示されないが気にしなくて良い

exit # 今作ったtmpユーザーでラズパイにログインする為にログアウト。

# tmp でログインし直す

sudo usermod -l **newjetson jetson** # usermod -lでユーザ名を変更

sudo usermod -d /home/newjetson -m newjetson # usermod -dでホームディレクトリ変更

sudo groupmod -n newjetson jetson # groupmod -nでグループを変更

sudo passwd newjetson # newpiユーザーのパスワードを変更（後でも可）

# newpi でログインしなおす # 新パスワード，またはpiユーザーのパスワード

sudo userdel tmp # tmpユーザーの削除

資料

[ terminal @Jetson（sshまたはシリアルコンソール）　但しログイン中

# 名前の変更

usermod -l newjetson jetson

# ホームディレクトリの変更

usermod -d /home/ newjetson -m newjetson

# コメントの変更

usermod -c newjetson newjetson

### Ubuntuのアップデート

python3.8を使う場合にはUbuntuをアップデートした方が良い

2時間以上かかるらしいので覚悟してやる事

#### Ubuntu20のイメージ

有志が作ったイメージがあるらしい。多分これが一番早い

URL：[https://github.com/Qengineering/Jetson-Nano-Ubuntu-20-image#installation](https://github.com/Qengineering/Jetson-Nano-Ubuntu-20-image" \l "installation)

のour Syncのところ。

各種バージョン情報

|  |  |  |  |
| --- | --- | --- | --- |
|  | バージョン |  | バージョン |
| CUDA | 10.2.300 | tensorflow | 2.4.1 |
| cuDNN： | 8.2.132 | torch | 1.13.0a0+git7c98e70 |
| TensorRT： | 8.0.1.6 | torchvision | 0.14.0a0+5ce4506 |
| VPI | 1.1.15 | OpenCV | 4.8.0 |
| python3 | 3.8.10 |  |  |

書き込み

[ terminal（開発機）]

xzcat JetsonNanoUb20\_3.img.xz | sudo dd of=/dev/sdb bs=1M status=progress

補足：初回は自動ログインできるらしい

ID情報

user jetson

pass jetson

先にやる事

・wifi接続

・ホスト名の変更

・ユーザー名の変更（任意）

他にやった方が良い事

・python3 –m pip install –U pip

#### アップデート

自力でアップデートする場合

[ terminal ]

# Chromiumを削除する必要があるらしい

sudo apt remove –y --purge chromium-browser chromium-browser-l10n

[ /etc/update-manager/release-upgrades ]（書き込みには管理者権限が必要）

prompt=normal

[ terminal ]

sudo apt update

sudo apt dist-upgrade

sudo reboot

再起動後

[ terminal]

sudo do-release-upgrade

補足：SSH経由でのアップグレードは非推奨らしい。直接ログインが一番無難。

補足：途中で出る選択肢は基本Y(yes)でアップデートする

補足：標準pythonを変更して、apt\_pkg？というライブラリが無いと失敗した。その場合は3.6に戻す。

#### アップデート直後

sudo apt autoremove

sudo apt update

# 再度chromiumをアンインストール

sudo apt remove --purge chromium-browser chromium-browser-l10n

sudo apt --fix-broken install

sudo apt upgreade

# libopencvの更新？

sudo dpkg --purge --force-all libopencv-dev

sudo apt --fix-broken install

# 必要ライブラリorcaのインストール

sudo apt install orca -y

sudo apt autoremove

# ここで再起動

sudo reboot

言語の変更

GUI

Settings > Region & Language > Manage Installed Languages

torch

あやしくない？<https://github.com/Qengineering/Jetson-Nano-Ubuntu-20-image>

#### 資料

情報元１　<https://qiita.com/rikupo/items/cc807c45e46166b422b6>

情報元２　<https://qiita.com/Manyan3/items/c1d583dbfef443536cef>（おおもとの情報元は情報１と同じ）

Jetpackなど：<https://qiita.com/dandelion1124/items/58210e472b3057af97f5>

情報元　おおもと（英語）<https://qengineering.eu/install-ubuntu-20.04-on-jetson-nano.html>

### 基本設定

#### cuda

##### 概要

バージョン確認

/usr/local/cuda/bin/nvcc -V

tensorflowとcudaの対応表

[https://www.tensorflow.org/install/source?hl=ja#gpu\_support\_2](https://www.tensorflow.org/install/source?hl=ja" \l "gpu_support_2)

##### アップデート

CUDAパッケージのアンインストール

sudo apt-get --purge remove cuda

sudo apt-get --purge remove libcudnn8

sudo apt-get --purge remove libcudnn8-dev

# NVIDIAドライバーのアンインストール

sudo apt-get --purge remove nvidia-\*

システムを再起動する

新しいCUDAバージョンのダウンロード

NVIDIAの開発者向けページから、新しいCUDAバージョンをダウンロードします。以下のリンクからCUDA 11.2をダウンロードできます。

cudaのリポジトリ（Jetsonはarm互換のCPU）

<http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/arm64/>

補足（通常のレポジトリ

<http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/>

インストール

sudo dpkg -i cuda-repo-ubuntu1804-11-2-local\_11.2.0-460.32.03-1\_amd64.deb

sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub

sudo apt-get update

sudo apt-get install cuda

参考：https://note.com/kuwakado/n/nef079b46b689

参考：https://qiita.com/yukoba/items/4733e8602fa4acabcc35

#### nano

標準ではviしか入ってないっぽい。

#### python

##### venv

python-venvは入っていないのでインストールが必要

sudo apt install python3-venv

##### アップグレード

aptでpython3.7と3.8がインストールできる。

ただし、

python3.8 Ubuntu 20.04を入れた方が色々と良い

sudo apt install python3.7 python3.7-venv

* + - 1. メモ：デフォルトのpythonの変更

sudo unlink /usr/bin/python3 # 不要かも 2023-03-24

sudo rm /usr/bin/python3 # 不要かも 2023-03-24

sudo ln -sf /usr/bin/python3.7 /usr/bin/python3

tensorrtのアップグレードが必要？

* + - * 1. python3.9をインストールする

#### swap領域

スワップファイルでの対応する場合

# スワップファイルを作成する

# 補足 ddでは　bs×count でファイルをコピーする

sudo dd if=/dev/zero of=/swapfile bs=1M count=6144 # /(rootディレクトリ) に直接swapfileを作成

sudo chmod 600 /swapfile # 権限の変更

# フォーマットしてをスワップを有効化

sudo mkswap /swapfile

sudo swapon /swapfile # 取り消す時はswapoff /swapfile

次回ログイン時にも有効になるように下記の設定ファイルを変更

[ /etc/fstab ]

/swapfile swap swap sw 0 0 0 # システムがswapと認識できる

補足：fallocateを使った方法を示すWebサイトがあるが、fallocateは非推奨

無事作成されたかどうかを確認する

[ terminal ]

free -mh

非公式の設定スクリプトがあるらしい

git clone https://github.com/JetsonHacksNano/installSwapfile

cd installSwapfile

./installSwapfile.sh

[installSwapfile.sh]

# You can use 1 or 2 if need be

SWAPSIZE=6

補足１

最初から/dev/zram0 ~ /dev/zram3 までで合計2GBのスワップ領域があり、新しく6GBのスワップ領域を作成する為（SWAPSIZE=6の時）、合計で8GBのスワップ領域となる

補足２

このスクリプトではfallocateを使っているらしいが、スワップ領域はddで作成する事が推奨されている。

#### VNC

JetsonはデフォルトでVNCサーバーが入っていない。

Jetson Nanoにはvinoが良いらしい。

vinoのインストール

sudo apt -y install vino

gsettings set org.gnome.Vino prompt.enabled false

gsettings set org.gnome.Vino require-encryptsion false

### OpenCV

wget https://github.com/Qengineering/Install-OpenCV-Jetson-Nano/raw/main/OpenCV-4-5-5.sh

sudo chmod 755 ./OpenCV-4-5-5.sh

./OpenCV-4-5-5.sh

### tensorflow

#### インストール

##### .whファイル

JetPackのバージョンに応じたtensorflowの.whファイルをダウンロード

<https://developer.download.nvidia.com/compute/redist/jp/>

<https://forums.developer.nvidia.com/t/official-tensorflow-for-jetson-nano/71770>

##### コンパイル

### pytorch

yoloxを使うため

#### インストール

##### 公式

https://docs.nvidia.com/deeplearning/frameworks/install-pytorch-jetson-platform/index.html

ただしJetson Nanoは非対応

##### .whファイル

https://forums.developer.nvidia.com/t/pytorch-for-jetson/72048

(3.6) https://nvidia.box.com/shared/static/fjtbno0vpo676a25cgvuqc1wty0fkkg6.whl

もしくは

https://developer.download.nvidia.com/compute/redist/jp/v461/pytorch/

> 1.10はインストールできた2023-05-25

sudo apt -y install python3-pip # pipをインストール

python3 -m pip install Cython # numpyのインストールに必要

python3 -m pip install numpy==1.19.4 # numpyが入っていないっぽい

sudo apt install libopenblas-base libopenblas-dev libopenmpi-dev # torchに必要なライブラリ

python3 -m pip install torch-1.10.0-cp36-cp36m-linux\_aarch64.whl

補足：

・2023-05-25時点ではバージョン指定しないとnumpy==1.19.5が入る。

・1.19.5ではうまく動かない。”Illegal instruction (core dumped)”

テストソース

import torch

print(torch.\_\_version\_\_)

print(torch.cuda.is\_available())

##### torch visionのインストール

# torch visionのインストール

git clone --branch v0.11.1 https://github.com/pytorch/vision torchvision

cd torchvision

export BUILD\_VERSION=0.x.0

python3 setup.py install

checkout方式

git clone https://github.com/pytorch/vision torchvision

git checkout v0.11.1

export BUILD\_VERSION=0.11.1

python3 setup.py install

sudo python3 setup.py install # sudo がないとだめ？

torchのバージョンによってインストールするtorchvisionのバージョンが変わってくるので注意

参考；公式のバージョン表：<https://pypi.org/project/torchvision/>

|  |  |  |
| --- | --- | --- |
| torch | torchvision |  |
| 1.10.0 | 0.11.1 |  |

##### 資料

https://murasan-net.com/index.php/2022/11/01/jetson-nano-yolov5-setup/

参考：https://qiita.com/spa15332/items/ddb449cccb1da6f0047f

参考：<https://qiita.com/ez_tera/items/56a7f97c5d98dd914fd9>

##### ビルド

3.7以上でtorchを使う場合

Ubuntuが20.4である必要があるらしい

参考：<https://i7y.org/pytorch-build-on-jetson-nano/>

OpenCV

sudo wget https://github.com/Qengineering/Install-OpenCV-Jetson-Nano/raw/main/OpenCV-4-5-0.sh

情報元：[https://murasan-net.com/index.php/2022/11/01/jetson-nano-yolov5-setup/#index\_id4](https://murasan-net.com/index.php/2022/11/01/jetson-nano-yolov5-setup/" \l "index_id4)

### YOLO

#### v8のインストール

# pip3 install torch # pytorch はインストールしてある前提で

pip3 install ultralytics

#### v5のインストール

git clone https://github.com/ultralytics/yolov5.git

cd yolov5/

pip3 install -r requirements.txt # sudoなし

注意点

requirements.txtには以下の記述もある。個別にインストールしたものがある場合はコメントアウトする

#opencv-python>=4.1.2

#torch>=1.7.0

#tensorboard>=2.2

#torchvision>=0.8.1

# 以下、意味あるか分からないが

# matplotlib

# numpy # ==1.19.5 はバグがあるらしい

# requests

# Pillow

# tqdm

# pandas

# 間違いなくinstallが必要なもの

gitpython

thop

psutil

pyyaml

ultralytics

2023-07-12 メモ：ultralyticsはpython3.7以降が必要

python3 detect.py # 初回は .ptファイルのダウンロードがある

NMS ；Non-Maximum Suppression

オーバーラップするバウンディングボックスの内、もっとも可能性の高いバウンディングボックスのみを残す処理の事（らしい）

#### 実行結果（単位：mSec）

2023-08-02 実行

[ v8n ] 　.ptファイルのサイズ： 6.5MB [ v8s ] 　.ptファイルのサイズ： 22.6MB

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| Image | **61.1** | **104.4** |
| Preprocess | 33.5 | 4.2 |
| Inference | 61.1 | 104.4 |
| postprocess/img | 158.6 | 6.5 |

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| Image | **83.4** | **175.9** |
| preprocess | 83.4 | 6.5 |
| inference | 85.0 | 175.9 |
| postprocess/img | 263.0 | 7.7 |

[ v5nu ] .ptファイルのサイズ：5.5MB [ v5su ]　.ptファイルのサイズ： 18.5MB

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| Image | **69.8** | **104.5** |
| Preprocess | 53.3 | 4.1 |
| Inference | 69.8 | 104.5 |
| postprocess/img | 9.9 | 6.9 |

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| image | **75.3** | **73.7** |
| preprocess | 62.0 | 4.6 |
| inference | 75.3 | 73.7 |
| postprocess/img | 96.6 | 8.4 |

2023-08-02 v6の.ptファイルのダウンロード先が分からなかった

[ v6n ] [ v6s ]

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| Image |  |  |
| preprocess |  |  |
| inference |  |  |
| postprocess/img |  |  |

|  |  |  |
| --- | --- | --- |
|  | bus.jpg | zidane.jpg |
| image |  |  |
| preprocess |  |  |
| inference |  |  |
| postprocess/img |  |  |

#### ncnn版

参考資料

資料

#### 資料

##### 公式チュートリアル

https://github.com/dusty-nv/jetson-inference

https://developer.nvidia.com/embedded/learn/tutorials

YOLOv5 v6.0（2023年7月確認版）：<https://qiita.com/ez_tera/items/56a7f97c5d98dd914fd9>

YOLOv5(Orin, Nano)：<https://qiita.com/ez_tera/items/861505675ed33acd9a95>

YOLOv4 ：<https://qiita.com/ShimoyamaAi/items/7e44d07bc4b98d7ee055>

### dockerイメージ

tensorflow https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-tensorflow

pytorch https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-pytorch

ML全般 https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-ml

# tensorflow

sudo docker pull nvcr.io/nvidia/l4t-tensorflow:r32.7.1-tf2.7-py3

# pytorch

sudo docker pull nvcr.io/nvidia/ l4t-pytorch:r32.7.1-pth1.10-py3

docker run -it -d --name my\_image\_name my\_image # コンテナの作成

docker run --runtime nvidia -it --net host nvcr.io/nvidia/l4t-pytorch:r32.5.0-pth1.7-py3 bash

情報元：<https://wooolwoool.hatenablog.com/entry/2021/03/10/235851>

### その他初期設定関連

#### [Jetson Easy setup configurator](https://github.com/rbonghi/jetson_easy)

<https://github.com/rbonghi/jetson_easy>

### カメラデバイスの確認

#

sudo apt install v4l-utils

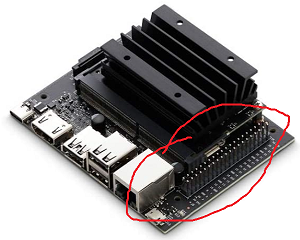
#

v4l2-ctl --list-devices

## 詳細

### GPIO（信号入出力）

以下のモデルには入出力のピンが標準で付いている。



Jetson AGX Orin

Jetson Xavier NX 開発者キット

Jetson Nano開発者キット

### devicetree の設定スクリプト？

sudo /opt/nvidia/jetson-io/jetson-io.py

情報元：<https://qiita.com/MakotoA67/items/8d2118e9a34eccb1cc6f>

### dockerを使う

sudo docker pull nvcr.io/nvidia/l4t-ml:r32.6.1-py3

### 関連パッケージ

#### jetson-stats

jtopなどを含む

sudo -H pip3 install jetson-stats

reboot

### 独自ファイル／ディレクトリなど

jetson\_release

## 情報

ポリテク

中部E312 マイコン制御として。Deep Learningも多少やる様子。

2022年現在空きあり

・AI認定プログラム「Jetson AI Specialist」というNVIDIAのベンダー資格がある。

# obniz

## 概要

・「obnizクラウド」経由で対応マイコン（※１）がリモート動作する（※２）システム。

・標準言語がJavaScript（※３）とマイコンとしてはユニーク。Python、ブロックプログラムでも可。

・開発環境はWebブラウザ上で動作する。

（参考）<https://qiita.com/wicket/items/546c2900bda7c8c85a60>

補足

※１　ESP32（ESP32-WROOM32）搭載機

※２　対応マイコン＋obnizクラウド＋クラウドに指令を出すコンピュータ（またはスマホ）という構成になる。

※３　Node.jsなども使えるらしい

公式デバイスで利用する場合

【 obnizクラウドの（無期限）利用料込み。 】

|  |  |  |
| --- | --- | --- |
| 名前 | 値段 |  |
| obniz Board | ¥6,090 | BLE/UART/AD/Displayなど |
| obniz Board 1Y | ¥6,930 | obniz Boardにスリープ機能搭載 |
| ESP32-DevkitC | ¥5,390 | 単体だと¥1,480（秋月） |
| obniz PLC | ¥？ | フォトプラ＋リレーで24V対応したマイコン？  詳細不明 |

<https://obniz.com/ja/products>

ESP32（ESP32-WROOM32）搭載の非公式（自作）デバイスの場合

・165円/月・台

・最初の１台は無料

・Biz（商用）プラン有り（詳細不明）

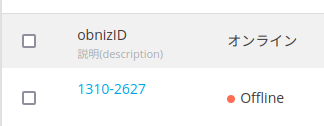
<https://obniz.com/ja/pricing>

## チュートリアル

アカウントの作成

<https://obniz.com/ja/console> →

デバイスキーの取得

開発者コンソール→管理→デバイス→Add Device→IDのみオンラインで発行

→ 「デバイスキー」 の Download → ダウンロードされたファイルをテキストエディタで開く。

python3 pip install -m obniz\_cli # npm 使わずともインストールできる？

# 公式情報なのに失敗した

obniz-cli os:flash-create -p /dev/ttyUSB0 --description "最初の１台" --config ./wifi-config.json

# こちらが成功 2022-03-20

obniz-cli flashos -p /dev/ttyUSB0

DeviceKey

Input text >> # デバイスキーを入力

Select Interface

0:Wireless LAN(default) # Wireless 以外の選択肢があるのが疑問 2022-03-20

1:Wired LAN

Enter SSID-Pass # 自分のSSIDを数字で指定する

--- select Network ---

0:DHCP(default)

1:Static IP

--- Proxy Setting ---

0:Proxy disable(default)

1:Proxy enable

> 問題がなければ Online と表示される

もう一度

https://obniz.com/ja/console → 管理→デバイス　を表示すると、今追加したデバイスがオンラインになっている。

「Lチカ」のcathodeは0番で良いらしい

cathode

蛇足情報

wifi-config.jsonの書き方？

{

"wifi\_channel": "14", # 14 日本　"11": North America　"13": EU and Almost all the World

"net": "wirelesslan",

"wifi": [

{

"ssid": "your ssid",

"pass": "your password"

}

]

}

情報元：https://obniz.com/ja/doc/reference/obnizos-for-esp32/settings/setting-json

sudo apt install npm

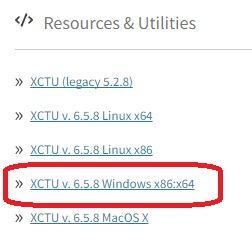
npm i obniz-cli -g

https://obniz.com/ja/doc/reference/obnizos-for-esp32/quick-start/

# XBEE

## 概要

## XCTU

少しだけ分かりにくい（右図参照）

ダウンロードサイト：

<https://hub.digi.com/support/products/xctu/>

# Lapis

ラピスセミコンダクタ株式会社。横浜に本社を置く日本の企業。

CT Sensor Shield 2

Lazurite 920J \4,620

# M5Stack

## 概要

特徴

・ESP32が内蔵されている。（bluetooth, wifi付）

・液晶ディスプレイ付き。

・押しボタン3つ。

・microSDのポート有り

・Groove（Seeed Technology社から製造・販売されている製品群）対応

開発環境

・Arduino Studio

## 種類

### 概要

メインシリーズ

BASIC 最もスタンダードなモデル

Gray BASICの上位モデル、9軸IMUを搭載

CORE2 Grayの上位モデル、PSRAM・タッチスクリーンを搭載

FIRE Grayの上位モデル、LEGOと繋げられる

特別シリーズ

Faces M5Stack Grayをゲーム機風にしたポケットコンピュータ型モデル

M5GO LEGOと繋げたりする学習キット。

Tough 防水性・耐久性のあるモデル。

カメラ

UnitV2 Linux OSを搭載し、単独で動くAIカメラ

PoECAM PoEを搭載したカメラ

スティック

M5StickC スティックタイプの最もスタンダードなモデル

M5StickC Plus M5StickCの後継機、画面サイズがアップ

M5StickV カメラ搭載モデル

M5StickT 赤外線サーモグラフィーカメラ搭載モデル

ATOM（小サイズ）

ATOM Matrix 液晶・バッテリー非搭載。代わりにLEDマトリックスを使う。

ATOM Lite フルカラーLEDが１つだけ搭載

ATOM Echo ATOMサイズにマイク・スピーカーを搭載したモデル（Amazon）

ATOM U USB-Aコネクタを搭載したモデル。

Stamp（極小サイズ）

M5Stamp Pico USBシリアル変換ICすらついていない。

M5Stamp C3 M5Stamp PicoにUSBシリアル変換ICを搭載

M5Stamp C3U メインチップでシリアル変換を行う。

M5Stamp S3

M5 Paper

M5Paper タッチ操作可能な電子ペーパーを搭載したモデル

M5Stack CoreInk M5Paperの小型モデル

参考：<https://www.ekit-tech.com/?p=3900>

### ATOM

5 V / 500 mA

#### ATOM Lite

CPU ESP32-PICO-D4（ESP32 PICOベース）

動作電源 5 V / 500 mA

### Stamp

#### M5Stamp Pico

CPU ESP32-PICO-D4

動作電源 5V / 500mA deep sleep 5V / 0.35mA

~MATE ピンなどがセットになって販売

### UnitV2

ARM Cortex-A7デュアルコア 1.2 GHz（Neon、FPU内蔵）

128 MB DDR3 メモリ

512 MB NANDフラッシュメモリ

GC2145 1080 P カラーセンサ

マイクロフォン、microSDカードスロット、UARTポート、USB-C

## 関連デバイス

### 赤外線送受信ユニット

<https://www.switch-science.com/catalog/5699/>

# BeagleBone Black

Texas Instrumentsが開発・販売している小型マイコンボードのシリーズ。

有線だがEthernetなどもあり，Raspberry Pi に寄せた仕様になっているらしい。

# HiFive

RISC-Vチップを搭載した評価ボード

ArduinoのIDEを使ってプログラムの開発を行う事ができる。

# ルネサスR8

## 仕様

全般

供給電圧(VCC) 2.2V～5.5V

発振子

20MHz 水晶（X2）

32.768kHz 水晶(X1)

ROM 16kバイト＋2kバイト RAM 1kバイト

RDタイマ（2本）にインプットキャプチャかoutput compare が選択できる。

8bitプリスケーラ

通信

I2C，UART

PORT(GPIO)

IO電圧最大 供給電圧±0.3V

入力H 0.8×VCC以上，VCC未満

入力L 0.2×VCC以下

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | L電流 | H電流 | ピン合計 |  |
| P2\_0～P2\_7 | 20mA (40) | -20mA (-40) | 80mA (160mA)  Hの時は -80(-160) |  |
| それ以外 | 5mA (10) | -5mA (-10) |  |

※() 内はピーク

### 配線

電源

AE-R8C25開発ボードへの電源について：

CN2の1,2 に+5Vを印加する。3,4をGNDと同電位にする。

その際、オンボードのレギュレータ(CN4；5.5～20V) を使っても良いらしい。

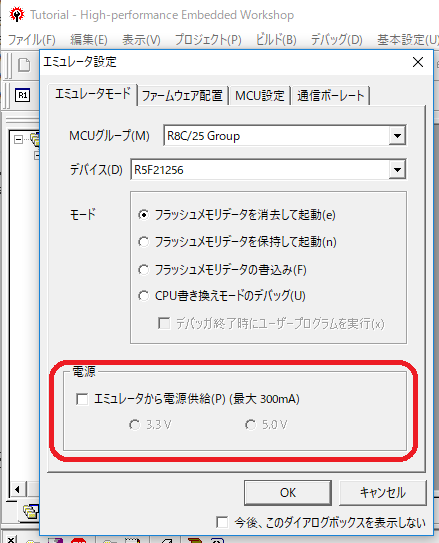
但しエミュレータ接続時はCN4からの電源を使ってはいけない。むしろ両方接続すると、どちらかが壊れる。

エミュレータからの電圧について

E8aエミュレータマニュアルp25より：

(b) VCC LED：ユーザシステムの電源状態を示します。

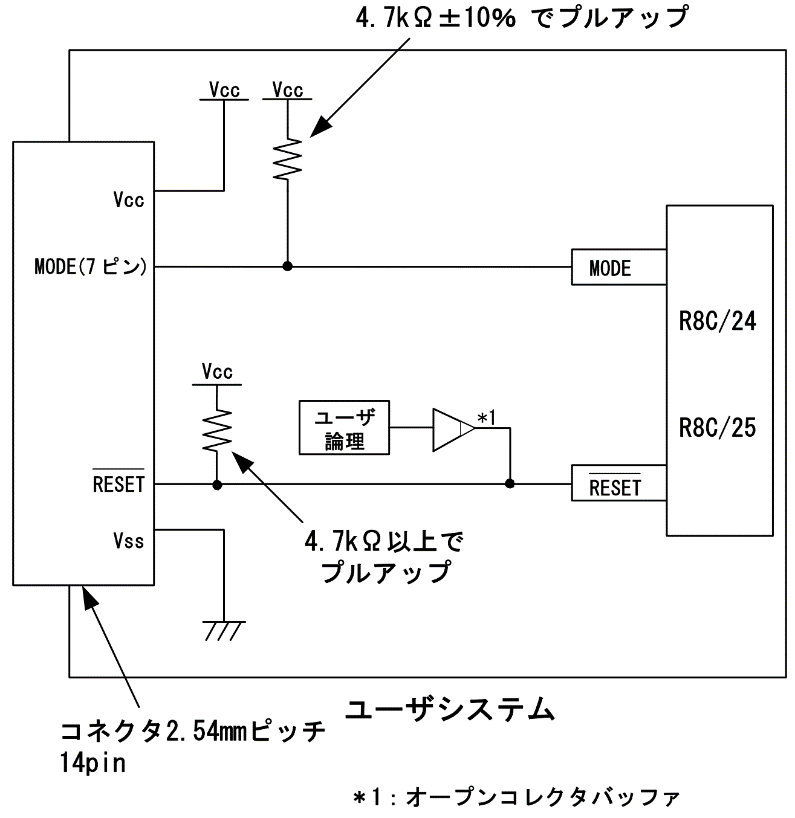
E8aエミュレータからユーザシステムに電源を供給した場合、VCC LEDが**橙色**に点灯します。 ユーザシステムからE8aエミュレータに電源を供給した場合、VCC LEDが**黄緑色**に点灯します。



E8aエミュレータから電源を取る事ができる。　（右図参照）

※但し他に電源をつないでいる時にエミュレータからも電源を供給しない事。どちらかが壊れる。

エミュレータと開発ボード（ユーザーインターフェイス）との信号の配線



エミュレータのマニュアル**別冊**の

５．E8aの接続例

を確認。（右図）

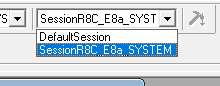
## アプリ

Flash Development Toolkit

フラッシュ書き込みツール

High-performance Embedded Workshop

総合開発環境

デバッグ専用のpinがマイコンボードにあるらしい？

エミュレータ設定が表示されない場合は右上の「ソース」を

「SessionR8C E8a SYSTEM」にする。

疑問

MCU ；Micro Computer Unit?

その他

5.5～20Vの電源を使用可能。（オンボードレギュレータ：最大出力電流150mA）

基本命令数89命令　←何のこと？20.06.01

開発ツールFlash Development Tool Kit について [20.05.29]

箱に入っていたCDのものはバージョンが4.07。Win10はV.4.09 Release03でないと対応していない。

Renesasのユーザー登録が必要だが、アップデート版をダウンロードできる。（そもそも無償評価版）

## High-performance Embedded Workshop

### 概要

コードファイルを保管する場所を「ワークスペース」と呼ぶ。デフォルトでは C:\WorkSpace になっている。

WorkSpace\ Tutorial にサンプルプログラムがある。

ファイルの単位は Workspace - Project - 各ファイル　となっている。

### 使い方

 停止ボタン：「一時停止」の意らしい。なお、一時停止中はF9でブレークポイントを置ける。

 現在のメモリの中身が確認できる。

### 注意点

関数内変数は，関数の最上部で宣言しないとコンパイルエラーになる。（理由は不明20.06.22）

### 新規プロジェクトのチュートリアル

いわゆる「Lチカ」を行う為のチュートリアル、という視点でのメモ書き。

新規作成

プロジェクトタイプ

大きく、以下の２つがある

|  |  |
| --- | --- |
| C source startup Application | C 言語で記述されたスタートアッププログラムを作成するためのプロジェクト  Tutorial　はこちらで記述してあるのでこちらの方が良いかも。 |
| Application | C/C++言語で記述された初期ルーチンファイルを含む |

CPI種別：

M16C R32C/M32C/M16C，R8C/Tiny

※R8C/Tiny シリーズにはR8C/2xシリーズなどが含まれる。

注意点。OK押した後のダイアログで「完了」を押してしまいそうになるが，「次へ」を押す事で色々な設定ができる。

I/Oライブラリ使用 標準入出力ライブラリが使える

ヒープメモリ使用 ヒープ領域管理関数sbrk() が使える

特にプロジェクト作成ダイアログの最後でメインプログラムなどの名前を変更できる。

メインプログラムはmain.c の方が分かりやすい。

デバッグ設定

メニュー：デバッグ → デバッグの設定

ターゲット：R8C E8a SYSTEM

デバッグフォーマット：IEEE695\_RENESAS → OKを押す

→ 「ターゲットを変更すると…」ダイアログが表示されるので はい

※初回でないと、この「ターゲットを変更すると…」ダイアログが表示されない。

デバッグセッション？？？

メニュー：デバッグ → デバッグセッション： R8C\_E8a\_SYSTEM を選択

ダウンロードモジュール

PC上でデバッグをする為に，マイコンのプログラムをダウンロードする（？）

なお、プロジェクトタイプがApplicationの時は初期設定で自動的に作成される。　←作成されなかったが？

Tutorialの設定が $(CONFIGDIR)\$(PROJECTNAME).x30 になっていたのでそれに合わせる。

コンフィギュレーションは：Debug\_R8C\_E8a\_SYSTEM

#### ファイル構成

project名

　├─ Debug

　├─ Release

　├─ Source

　│ ├ project名.c main関数がある，スタートポイント。個人的にはmain.c の方が分かりやすいと思う。

　│ ├ cstartdef.h スタックサイズの定義

　│ ├ typedefine.h 文字通り、型に関するdefine を記述する。（scalar types と表現されている）

　│ ├ resetprg.c(.h) C言語向き初期化。reset される時に実行されるプログラム？

　│ │ **エントリポイントの記述**がある。

　│ ├ initsct.c(.h) それぞれのセクションの初期化（？）

　│ ├ fvector.c 固定のベクトルテーブルの定義（？）　FixedVECTORの意らしい。

　│ ├ intprg.c 割込みの定義（トップアドレス？）

　│ ├ sfr\_r825.inc(.h) アセンブリ言語でのsfrレジスタの定義

　│ │ 名前はiodefine.h とかsfr\_define.hの方が分かりやすくない？

　│ ├ heap.c ヒープメモリのサイズの定義

　├　project名.hwp プロジェクトの設定ファイル。iniファイル形式になっている。

　├

※HEWの自動作成ファイルやTutorialを参考にしたファイル構成。

必ずしも使い勝手は良くないように感じたので慣れてきたら別バージョンで構成を練り直す。20.06.17

ファイルの追加は「プロジェクト名」のツリー部分を右クリックするとできる。

#### プログラムの流れ

スタックサイズの設定

#pragma を使っている例もあるが，基本的に同じ事を行っている

cstartdef.h

#define \_\_STACKSIZE\_\_ 0x80 // ユーザースタック(USP) のサイズ

#define \_\_ISTACKSIZE\_\_ 0x80 // 割込みスタック(ISP) のサイズ

#define \_\_HEAPSIZE\_\_ 0x0 // heap size

#define \_\_STANDARD\_IO\_\_ 0 // use standard I/O (ex printf)

#define \_\_WATCH\_DOG\_\_ 0 // When watchdog is made effective after reset, here is set to 1.

CPUレジスタの設定

RAMの初期化

ヒープ領域の設定

main();

while(1);

start

initsct

\_init

main

exit

定義へジャンプ　について

検索対象は現在開いているファイルのみ。つまり，ジャンプしたい場合は.hファイルは全て開いた方が良い。

#### クロックについて

クロックに関連するレジスタはCM0(0006h)とCM1(0007h)。更にFRA0～FRA2。

強いて言いうとOCD。

※XCIN-XCOUTは\_X1(32.768kHz)で，XIN-XOUTはX2(20MHz) のタイマ　X Clock の意？

|  |  |  |  |
| --- | --- | --- | --- |
| シンボル | 初期値 | 意味 | 備考 |
| CM02 | 0 | waitモードで周辺機能クロック停止なし |  |
| CM03 | 1 | XCIN-OUTドライブキャパシティ = High | INクロックが安定すると再書き込み？？ |
| CM04 | 0 | port4-3, 4-4 を選択 | 1:XCIN-XCOUT を選択 |
| CM05 | 1 | XINクロックを停止(OFF) | 0:XINクロックが発振(ON) |
| CM06 | 1 | Divide-by-8モード | 0の時はCM16,17のdivider値になる。 |
| CM07 | 0 | システムクロックを選択 | 0:XINクロックを選択 |
| CM10 | 0 | クロック稼働 | 1:全クロック停止（停止モード） |
| CM11 | 0 | XIN-XOUTオンチップフィードバック有効 |  |
| CM12 | 0 | XCIN-XCOUT オンチップフィードバック有効 |  |
| CM13 | 0 | port4-6, 4-7を選択 | 1:XIN-XOUTを選択 |
| CM14 | 0 | 低速オンチップ発振子を使用 |  |
| CM15 | 1 | XIN-OUT　ドライブキャパシティ = High |  |
| CM16 | 0 | システムクロックのdivier値選択 | b7, b6の順で  00：なし 01：by2 10：by4 11：by16 |
| CM17 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| シンボル | 初期値 | 意味 | 備考 |
| FRA00 | 0 | 高速オンチップ発振子をオフ | 1:オン |
| FRA01 | 0 | 低速オンチップ発振子を選択 | 1:高速オンチップを選択 |

また，開発ボードのジャンパJ1-4も関連。

J1,2：32.768kHz発信子(XC [X1])用 J3,4,：20MHz発振子(X [X2])用

1-3を接続、2-3をカット：外部発振子XC (X)を使用

1-3をカット、2-3を接続：ポートとして使用。（内部発振子を使う）

※ジャンパ配線は外部発振子を使うようになっているが、レジスタ(CM04,CM13)がOFFなので結局内部発振子を選択している。

⇒ まとめると、デフォルトでは低速オンチップ発振子をdiver無し（生値のまま）で使っている為、CPUクロックは125kHzと言う事になる。

ドライブキャパシティ

オンチップフィードバック

#### ポート関連のレジスタ

|  |  |
| --- | --- |
| PD0(0x00E2h) | ポート０方向レジスタ。各ビットの値と設定は　0:入力モード　1:出力モード  ポート1は00E3h，ポート2は00E6h，ポート3は00E7h，ポート4は00Ah，ポート5は00EEh |
| P2DRP(0x00F4h) | ポート２駆動能力レジスタ　　　※これができるのはポート２だけらしい  出力トランジスタの駆動能力設定  .b0がP2\_0　.b1がport2\_1　　0:Low 1:High |
| PUR0(0x00FCh) | .b0がport0\_0～0\_3プルアップ有無。  .b1がport0\_4～0\_7プルアップ有無。  .b2-3がport1用， .b4-5がport2用 .6-7がport3用  0x00FDh がport4-6用 |
| PMR(0x00F8h).b4 | ポートCLK1/TXD1/RXD1切り替えビット |
| 〃 .b7 | SSU／I2Cバス切り替えビット　　　0：SSU　　1:I2C |

未使用ポートの処理

P0～P3 入力モードに設定し、端子ごとに抵抗を介してVSSにプルダウン接続

または端子ごとに抵抗を介してVCCにプルアップ接続

P4\_3～P4\_5、P6 出力モードに設定し、端子を開放(注1、2)

暴走やノイズを考慮して、一定時間毎に方向レジスタの内容を再書き込みするプログラムにする。

pd2

trdoer1(013BH)

trdfcr(013AH)

表記

#### タイマーについて

##### RA

Lチカする為にはタイマーRAで十分なので，RAの使い方をまとめる。

（⇒ 但し、最終的な目的はPWMなので，余裕があったらRDの使い方も習得する 20.06.22）

特徴：

カウントダウン（アンダーフロー時に割込みが入る）

使用方法

・モードレジスタ(0x0102h).b4-b6 の3ビットで「カウントソース」を選択する。

・RAレジスタと(0x0104h) とプリスケーラレジスタ(0x0103h) に初期値を設定する。

・RAコントロール(0x0100h).b0 を1にするとカウント開始。（0にすると停止。）

⇒ RAレジスタとプリスケーラがカウントダウンする。プリスケーラが下の位となっており，プリスケーラがアンダーフローする瞬間に上の位であるRAレジスタの値が１つ減る。同時にプリスケーラは先ほど設定した初期に戻る。

RAレジスタがアンダーフローする時に割込みが発生する。つまり、タイマー割り込みの間隔は：

タイマー周期(mSec) = CPU周期(mSec) × カウントソース × プリスケーラ初期値 × RAレジスタ値

##### 割込み

タイマーの割り込みはソフトウェア割込みに分類される。（portによる割込みではない、と言う事？）

デフォルトでは割込み禁止になっている為，アセンブラ(asm) で割込み許可命令を書く必要がある。

asm(" fset I "); /\* 全体の割り込み許可 \*/

asm(" fclr I "); /\* 割り込み禁止\*/

また，デフォルトではTRAIC割込み制御レジスタの割り込み優先レベル(0x0056h.b0-b2) が０になっている。

レベル0では割込みが発生しない（禁止）なので，適当にレベルを上げておく。

割込みメソッドは以下の様に記述する

#pragma interrupt tra\_interrupted (vect=22) // 割り込み番号(vect=)はハードウェアマニュアルのベクタテーブルを確認する。

void tra\_interrupted(void)

{

int2ic\_addr.bits.b3 = 0: // 割り込み要求(0x055h.b3) をオフしないと，再度割込みが発生する。

// 処理

}

※但しintprg.c で割込み関数が実装されているので重複に注意。

割込みの優先度は割り込み制御レジスタ(0x056h).b0～b2 の3ビットで決める。

・アンダーフロー無し ってどうゆう事？ 20.06.22

<http://j-mcr.net/tech/r8cm12a/main008.html>

タイマーRA, RBにはPrescaler がある。RD, REにはない。

つまりCPUのクロック周期を分周期（Divider）で調整して，その周期時間×RDのカウント数でタイマーを使う？

タイマーRDには以下の５つのモードがある。（英語のdatasheetの直訳）

タイマーモード インプットキャプチャ、アウトプットコンペア。

PWMモード あらゆる幅の断続的なPWM

リセット同期PWM のこぎり派と短絡防止時間無しの三相波形(6本)

相補PWMモード デッドタイム付の三角波の三相波形(6本) ※complementary

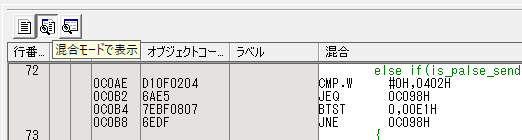
PWM3モード 一定間隔でのPWM波形(2本)

タイマーRDに関連するレジスタ

0x0137～0x015F まで。

fC4から生成された信号を使って1秒シグナルを作れる ⇒ Real-Time Clock Mode

#### サイクル数に関して

HEWでコンパイル → ダウンロードした後に

「混合モードで表示」をするとマシン語（ニーモニック）が表示される。

各MCUのユーザーズマニュアル　ソフトウェア編にコードサイズが載っているので，それでC言語のソースから実際に何サイクル必要なのかが分かる。

但しDIV（割り算）に関してはこの限りで無い。

##### アウトプットコンペア

タイマーのカウントアップ値が指定の値に達した時に割込みを入れてくれる機能。対応しているタイマーはRD。

（REも対応しているようではあるが？）

※最初思っていた，IOポートのOn/Off 回数をカウントするものではないらしい

概要

例えばTRD0(0x0146-0147h)とTRDGRA0(0x0148h-0149h)の値を比較する。

カウントアップ式で，値が一致すると割り込みが発生する。（オーバーフローした際にも割込みは発生する。）

値が一致すると自動的にTRDiの値をゼロに戻す。

設定すれば，一致した時にポート（trdioa0など）をOnにする事もできる。

関連レジスタ

|  |  |
| --- | --- |
| trd0ic(0x0048h).b0-b2 | TRD0割込み優先レベル選択　　　0x0049h:TRD1用 |
| trdstr(0x0137h).b0 | TRD0のカウント開始。（.b1はTRD1用） |
| 〃　　　　　　.b2 | [ CSEL ] TRD0カウント動作選択 （.b3はTRD1用）  0: TRDGRAレジスタとのコンペア一致でカウント停止 1:一致後もカウント継続 |
| trdmr(0x0138h).b0 | 0:TRD0とTRD1は独立　　　1: TRD0とTRD1は同期 |
| 〃　　　　　　.b4 | TRDGRC0レジスタ機能選択　　※意味がわからない20.06.26  0:ジェネラルレジスタ 1:TRDGRA0のバッファレジスタ |
| trdoer(0x013Bh) | TRDIOji port出力禁止　　0:出力許可　　1:出力禁止 |
| trdcr0(0x0140h).b0-b2 | カウントソース選択　　TRD1用は0x0150h |
| 〃　　　　　　.b5-b7 | カウンタクリア選択　　　　000：クリア禁止  001：TRDGRAiと一致でクリア　010：TRDGRBiと一致でクリア　など |
| trdcr0(0x0141h).b0-b1 | コンペア一致でIOポート出力したい時に設定が必要。  0x0141.b0-3がTRDA用。b4-b7がTRDB用。0x0142 がTRDC, TRDD用 |
| 〃　　　　　　.b4 | オーバーフロー [OVF] アンダーフローは.b5 [UDF] |
| trdier0(0x0144h) | タイマRD0割込み許可　　　　0x0154h：RD1用  .b0：TRDTRAと一致した割込み[IMFA]を有効　0:割込み禁止　　1:割込み許可  .b1-3：B-D　　.b4：オーバーフローの割り込み　　.b5：アンダーフローの割り込み |
| trdgra0(0x0148h) | タイマRDジェネラルレジスタA0  B0：0x014Ah　　C0：0x014Ch　　D0：0x014Eh |
| trdiora0(0x0151h) | タイマRD I/O制御レジスタA1  C1：0x0152h |

補足

trdiorc0(0x0142h).b3=1 にすると，TRDGRCとタイマーRDが一致した時に， TRDIOのC0ではなく，A0ポートをONにする事ができる。 ⇒ そんな事に意味があるのだろうか？

##### PWM

特徴

カウントアップ式

カウントソース：f1、f2、f4、f8、f32、fOCO40M，TRDCLKに接続した外部信号

fOCO40M　高速

FRA00ビットを1にすると供給。

wait 状態でも停止しない。

TRDIO**B**～TRDIO**D**を使用

関連レジスタ

|  |  |
| --- | --- |
| trdstr(0x0137h).b0 | TRD0のカウント開始。（.b1はTRD1用） |
| 〃　　　　　　.b2 | 0: TRDGRAレジスタとのコンペア一致でカウント停止 1:一致後もカウント継続 |
| trdmr(0x0138h).b0 | 0:TRD0とTRD1は独立　　　1: TRD0とTRD1は同期 |
| trdcr0(0x0140h).b0-b2 | カウントソース選択 |
| TRD0(0x0146h) | TRDのカウンタ [2byte] |
| TRDGRA0(0x0148h) | コンペアするカウンタ数（＝パルス数）　この値に一致すると割り込み発生  この値に達するまではportをonし続ける為，これがパルス幅になる。  割り込み番号　RD0：8　RD1：9  　TRD0とTRD0A（等）が一致する度に割込みが発生するらしい  　⇒　割り込み先で，一致したのはTRD0i のうちどれ？という処理が必要と言う事？ |
| TRDGRji(0x？？？) | 1/fk×この値がOnの時のパルス幅となる。jはA～D。 |
| TRDGRji () |  |

タイマー自体は同周期で動き，コンペア数（＝パルス数）を変更すれば良い。

### プログラム詳細

#### レジスタ

PRCR：重要なレジスタの保護(=1)／解除(=0)

アドレス：000Ah

.b0 [CM0, CM1] システムクロック；[OCD]オシレーション停止；[FRA0 FRA1]高速オンチップオシレータ

.b1 [PM0, PM1] プロセッサモード

.b2 [PD0] ポート０のディレクション

.b3 VCA2電圧ディレクション, VW0C, VW1C, VW2C電圧モニタ

.b4-5 予約　常に0

.b6-7

P[0-6]D：ポート入出力設定

アドレス：00E0h ※P0Dの場合

.b0-7 =0：各ポートを入力で使う。 =1：各ポート（ビット）を出力で使う

#### RENESAS特有の pragma

#pragma ENTRY 初期化関数（エントリポイント）の指定。

構文：#pragma entry *name\_of\_entry\_point\_method*

#pragma ADDRESS # 自動的にvolatile

変数を宣言する時のメモリアドレスを指定する

［用法］

#pragma address [ (　*arg*=絶対アドレス [, ...] ) ]

［例］

#pragma address (io=0x0ffe00)

int io; // 変数ioは0x0ffe00番地に生成される。

#pragma CREG　C言語における変数名を指定

構文：#pragma CREG *arg\_name\_on\_C cpuレジスタ名*

#pragma INTERRUPT 割込み関数を定義する

構文：#pragma INTERRUPT *method\_ name* (vect *=num*)

#pragma section program interrupt

#pragma ASM

アセンブリ言語を記述する。これは括弧の開始になっていて、　#pragma ENDASM　で括弧を閉じる。

#### アセンブラ言語（ニーモニック；抜粋）

AND.B レジスタをビット単位で書き換える

（例）AND.B #値, アドレス番地。

FCLR フラグレジスタのビットクリア

SHL 左シフト

### 特筆事項

0xF0000 ～ 0FFFFF の4k バイトの領域を near と呼ぶ。

用語

データフラッシュ領域

ユーザフラッシュ領域

ベクタ領域 プログラムの開始番号を可変にするCPUで、プログラムのジャンプ先ばかりを格納した領域。

### 参考資料

RENESASコンパイラ言語仕様

<http://tool-support.renesas.com/autoupdate/support/onlinehelp/ja-JP/csp/V4.01.00/CS+.chm/Compiler-CCRL.chm/Output/cd_EXP_LANG.html>

# SPRESENSE

## 概要

Sonyが出しているシングルボードコンピュータ

（参考）<https://qiita.com/TaroYoshino/items/e9ec0c6dabacb6074fbb>

Neural Network Console

ドラッグ＆ドロップで簡単ニューラルネットワーク設計

Windowsアプリ版、クラウド（Webアプリ）版がある。※Linuxは無い様子。

<https://dl.sony.com/ja/>

# NORDIC Semiconductor

nRF52 DK

個人的にはBluetoothに特化したマイコンに思える。

Adafruit Feather nRF52 Bluefruit LE

https://www.switch-science.com/catalog/3277/

nRF52832 MDBT42Q 評価ボード

https://www.switch-science.com/catalog/5526/

# TinkerBoard

# 市販マイコンの仕様まとめ

## 環境センサ系

### BME280

IIRフィルター 圧力値を突然の変化から守るためのフィルター。

・仕様まとめ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 項目 | アドレス | calib# | 分解能 |  |
| 温度 | 0x88～0x8D | 0～5 | 16+(osrs\_h - 1) |  |
| 圧力 | 0x8E～0x9F | 6～23 | 16+(osrs\_p - 1) |  |
| 湿度 | 0xA1,0xE1～0xE7 | 25～32 | 16bit固定 |  |

# 関連するソフトウェア

## circuitpython

Adafruitが開発したpythonコンパイラ。対応マイコンのハードウェアで動作する。

MicroPythonから派生。オープンソースのプログラミング言語。

## Mongoose OS

https://mongoose-os.com/docs/mongoose-os/quickstart/setup.md

## Mosquitto

MQTTのブローカー

## Node RED

フローベースでハードウェア制御のプログラムを行う開発ツール。

対応システム

OPTO 22, groov EPIC / groov RIO

Raspberry Pi

Intel, IoT Gateway

Siemens, SIMATIC IOT 2040

など

## RTOS

大きくFreeRTOSとeT-Kernelがあるらしい。

|  |  |  |  |
| --- | --- | --- | --- |
|  | FreeRTOS | eT-Kernel | μTron |
|  |  |  |  |
| 料金など | フリー | 業者から買う  ※その代わりサポートがある |  |

MCOS？？

古くはITRON，VxWorksなどがあった。

### Linuxとの併用

|  |  |  |
| --- | --- | --- |
|  | リアルタイムOS | Linux |
| 起動時間 | 速い | 遅い |
| リアルタイム処理 | 可 | 不可  ※リアルタイム処理に対応可能な製品も一部あり |
| ソースコード公開義務 | なし | あり |
| ソフトウェア資産 | 充実度でLinuxより劣る | 豊富 |
| ソフトウェア技術者数 | 組込み分野に限定される | PCベースを含めると非常に多い |
| 信頼性・堅牢性 | 機能安全認証取得済み製品あり | 機能安全認証への対応が難しい |

大きくネイティブ（OpenAMP）方式とハイパーバイザー方式がある。

|  |  |  |
| --- | --- | --- |
|  | ネイティブ方式（OpenAMP） | ハイパーバイザー方式 |
| ヘテロコア対応 | 対応 | 未対応  ※同一命令セットの構成は対応可能な場合あり |
| オーバーヘッド | 無 | 有 |
| OS の分割・割当単位 | コア単位 | 制限無し |
| 応答性 | ◎ | △ |
| 開発コスト | 〇 | △ |
| デバッグ | 〇 | △ |
| 構成の自由度 | 〇 | 〇 |

# 用語

MSYS2

Minimal SYStem。Windows上で動作する最小限Unixシステム。

（関連）MinGW

MSYS2

MinGW GCCなどの開発ツールをまとめたツールチェイン

REPL；Read-Eval-Print Loop 入力・評価・出力のループ

エッジデバイス

ネットワークに接続され「末端」に位置するデバイスの事。

GNSS

Global Navigation Satellite System / 全球測位衛星システム

米国（GPS）、日本（QZSS）、ロシア（GLONASS）、欧州連合（Galileo）等の衛星測位システムの総称

IMU；Inertial Measurement Unit

慣性計測装置。加速度、回転、位置変化といった物理運動の検出・計測を行うセンサ。

3軸のジャイロと3方向の加速度計によって、3次元の角速度と加速度が求められる

# Memo

## スマートリモコン

### IODATA　HS-IRR01

サンワサプライ　400-SSA004

### スイッチボット　SwitchBot Hub Mini

python

pathlib.Path.cwd()

pyenv

/usr/binpython のシンボリックリンクの向かう先を変えるだけでよいらしい