

令和2年度「専修学校による地域産業中核的人材養成事業」

教員用講義資料

令和 2 年度「専修学校による地域産業中核的人材養成事業」

教員用講義資料

情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

目次

第 1 章 組込 IoT の位置づけ	1
第 2 章 マイクロ工場 (μ Factory) の開発	4
2.1 マイコンの開発実習	4
2.2 LED の制御	14
2.3 Button クラスの制御	29
2.4 シリアル通信の制御	34
2.5 WEB 経由の通信制御	48
2.6 サーボモータの回転角度制御	64
2.7 Bluetooth の活用	78
2.8 LED の動作確認	89
2.9 Button クラスの動作確認	94
2.10 シリアル通信計測	99
2.11 WEB 経由の通信計測	109
2.12 フルカラー LCD の計測	126
2.13 加速度センサの活用	158
2.14 カラーセンサの活用	176
2.15 IoT クラウドモデルの開発	192
2.16 自由なデータ分析	187

第1章 組込IoTの位置づけ

組込IoTの位置づけ

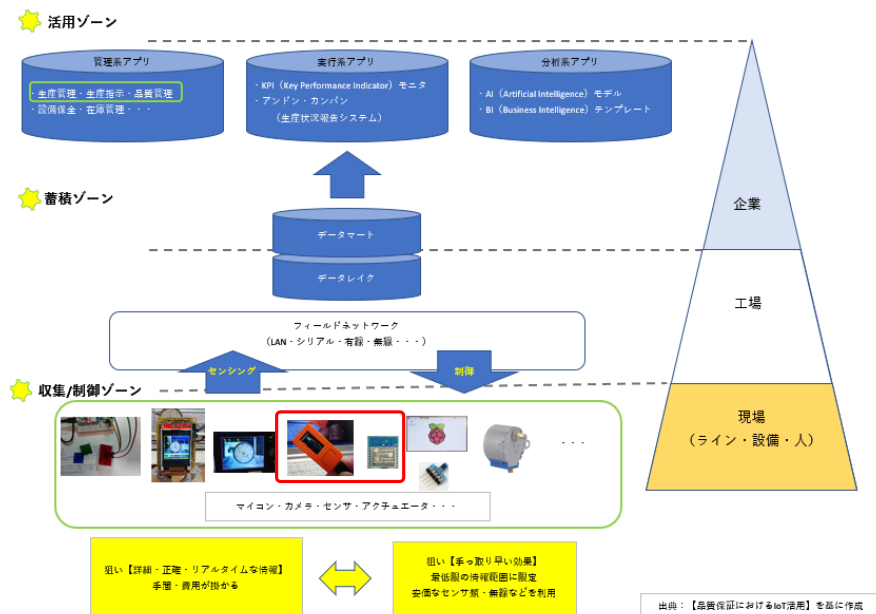
予備知識



一般社団法人全国専門学校情報教育協会

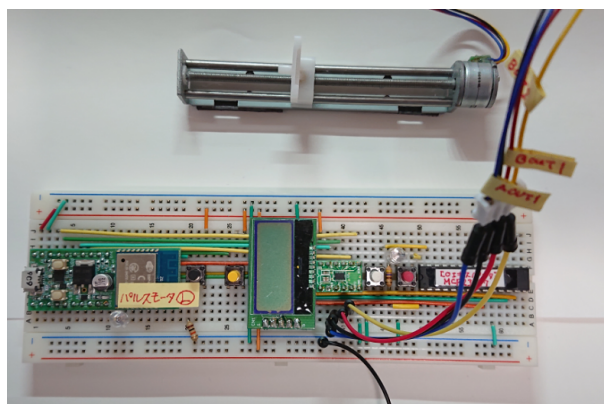
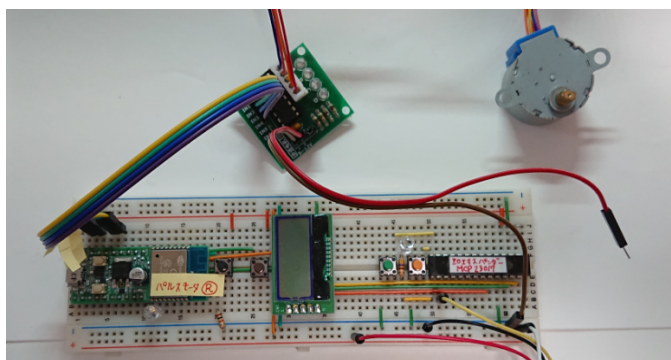
1

ファクトリー体系図



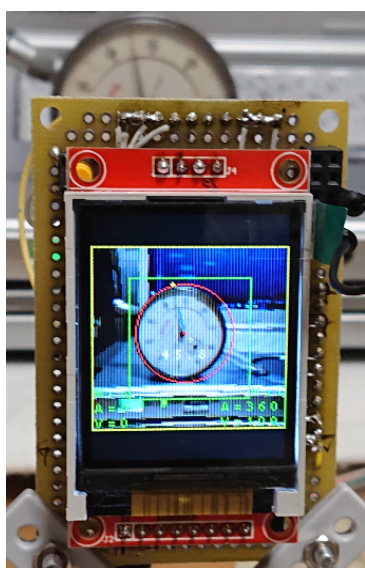
一般社団法人全国専門学校情報教育協会

ステッピングモータ制御例



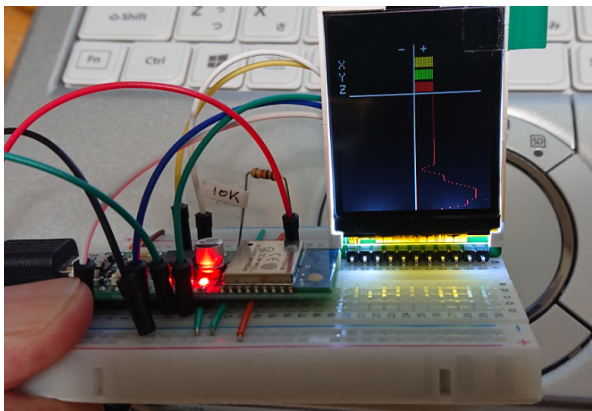
一般社団法人全国専門学校情報教育協会

アナログゲージ判読の例



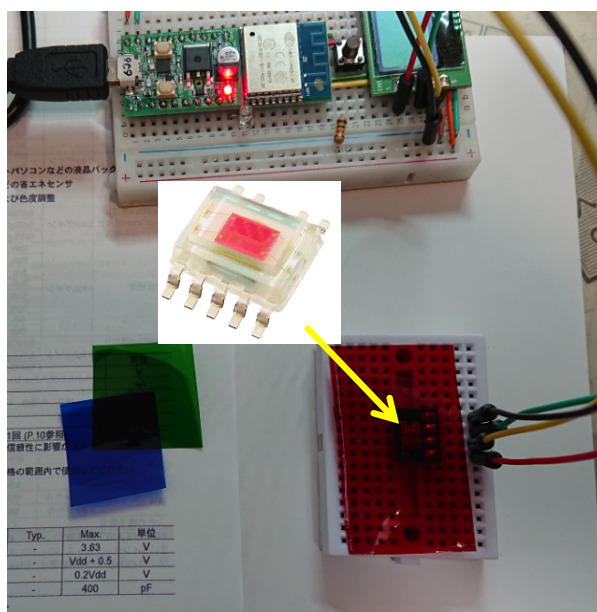
一般社団法人全国専門学校情報教育協会

加速度センサの例



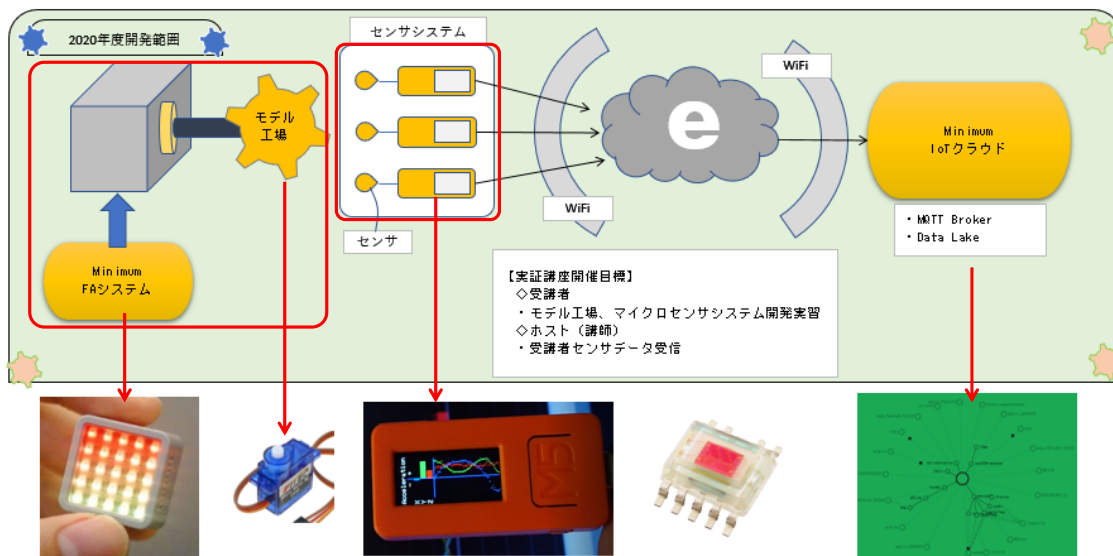
一般社団法人全国専門学校情報教育協会

色センサの例



一般社団法人全国専門学校情報教育協会

開発範囲



一般社団法人全国専門学校情報教育協会

第2章 マイクロ工場 (μ Factory) の開発 2.1 マイコンの開発実習

M5Atom vs M5Stick-C

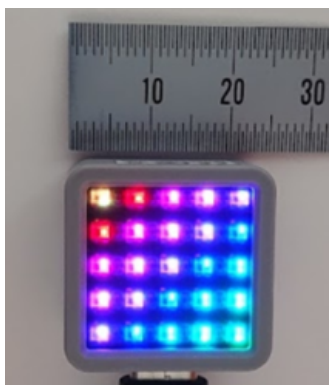
実習で用いるマイコン



一般社団法人全国専門学校情報教育協会

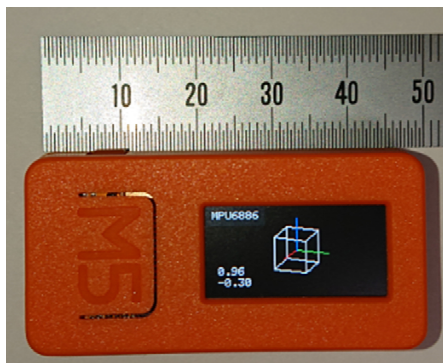
実習で用いるマイコン

制御用



M5ATOM Matrix

センシング用



M5Stick-C

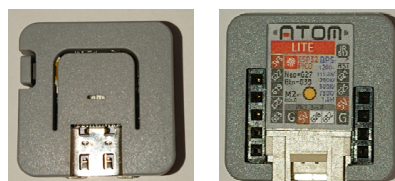


M5ATOM Matrix



- ◇組み込みデバイス開発に適する大きさ(24×24mm)
- ◇Wi-FiとBluetooth通信が可能
- ◇4 MBの内蔵SPIフラッシュメモリ
- ◇ESP32-PICO-D4チップを搭載
- ◇赤外線LED・5×5 RGBマトリクスLED
 - ・内蔵IMUセンサ(MPU6886)・Grove互換I/F
 - ・汎用SW (LEDマトリクス奥)・USB Type-C
 - ・基板取り付け用M2ネジ穴
- ◇電源:5VDC USBケーブルまたは背面ピンソケット給電

M5ATOM Lite



◇Single LEDのタイプ

M5Stick-C



- ◇0.96インチ80×160TFTカラー
- ◇Wi-FiとBluetooth通信が可能
- ◇80mAh LiPoバッテリー
- ◇電源: 5VDC USBまたはピンソケット給電
- ◇ESP32-PICO-D4チップを搭載
 - ※ M5ATOMと同一
- ◇4MB フラッシュ+520K RAM
- ◇・6軸IMU(MPU6886)・赤色LED
 - ・IRトランスミッタ・マイクロフォン
 - ・SW×2・電源SW×1
 - ・Grove I/F・RTC

◇TFTカラー液晶搭載
↓
多彩な表示が可能
↓
【センシング】に用いる

3

CPU ESP32-PICO-D4

- ◇M5ATOM、M5Stick-Cはいずれも同じCPU ESP32-PICO-D4チップを使用している
- ◇CPUメーカーから下に示すデータシートが公開されている (右は目次の一部)



Contents

- 1 Overview
- 2 Pin Definitions
 - 2.1 Pin Layout
 - 2.2 Pin Description
 - 2.3 Strapping Pins
- 3 Functional Descriptions
 - 3.1 CPU and Internal Memory
 - 3.2 External Flash and SRAM
 - 3.3 Crystal Oscillators
 - 3.4 RTC and Power Consumption
- 4 Peripherals and Sensors
- 5 Electrical Characteristics
 - 5.1 Absolute Maximum Ratings
 - 5.2 Recommended Operating Conditions
 - 5.3 DC Characteristics (3.3 V, 25 °C)
 - 5.4 Wi-Fi Radio
 - 5.5 BLE Radio
 - 5.5.1 Receiver
 - 5.5.2 Transmitter
 - 5.6 Reflow Profile

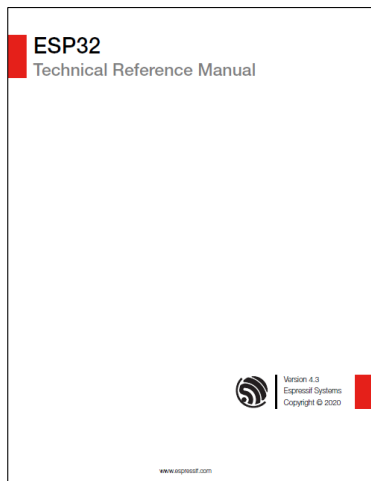
目次

1. 概要
 2. ピン配置
 3. 機能説明
 4. 入出力とセンサ
 5. 電気的特性
- ...etc

4

CPU ESP32

◇ESP32-PICO-D4チップの詳細は、このチップのベースになった【ESP32を参照せよ】と記述がある
※詳細は【ESP32 Technical Reference Manual】を参照する(右は目次の一部、700頁以上ある)

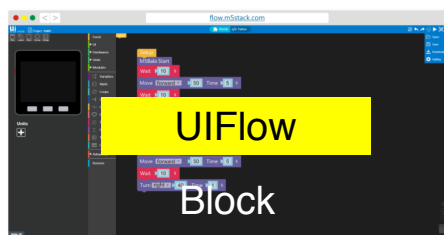


- > 1 Documentation Conventions
- > 2 System and Memory
- > 3 Interrupt Matrix
- > 4 Reset and Clock
- > 5 IO_MUX and GPIO Matrix
- > 6 DPort Register
- > 7 DMA Controller
- > 8 SPI
- > 9 SDIO Slave
- > 10 SD/MMC Host Controller
- > 11 Ethernet MAC
- > 12 I²C Controller
- > 13 I²S
- > 14 UART Controllers
- > 15 LED_PWM
- > 16 Remote Control Peripheral
- > 17 MCPWM
- > 18 PULSE_CNT
- > 19 64-bit Timers
- > 20 Watchdog Timers
- > 21 eFuse Controller
- > 22 TWAI
- > 23 AES Accelerator
- > 24 SHA Accelerator

ソフトウェア開発環境

◇開発プラットフォーム 現在3種類が提供されている
・UIFlow(専用IDE) ・Micro Python(シリアルターミナル) ・Arduino IDE(専用IDE)

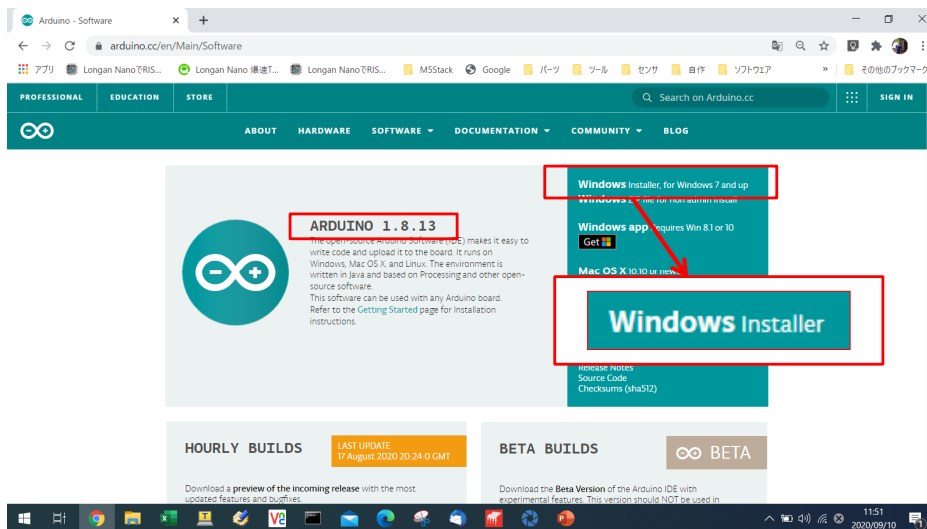
→ Arduino IDE(専用IDE)を用いる



Arduino IDEの Install → Arduino Home

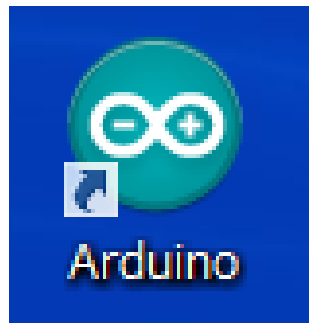


Installer Download

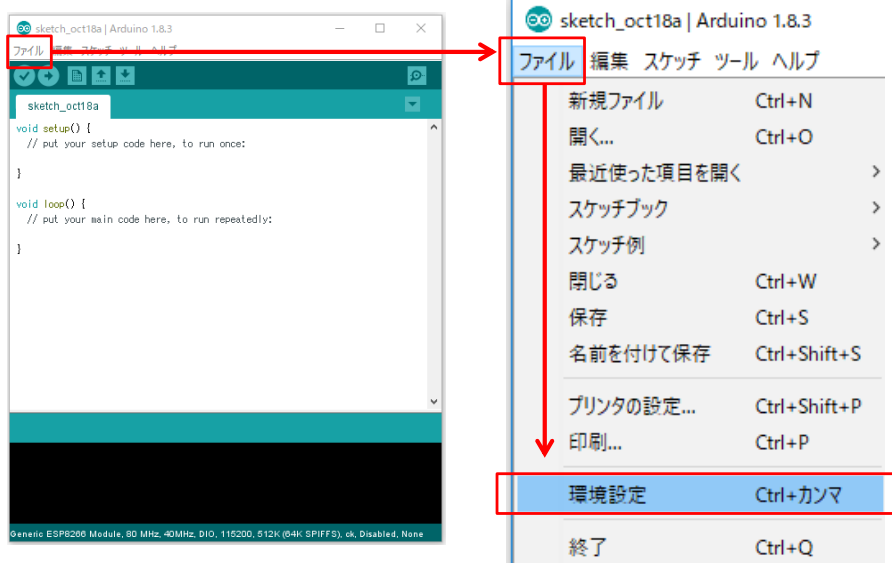


Arduino IDE

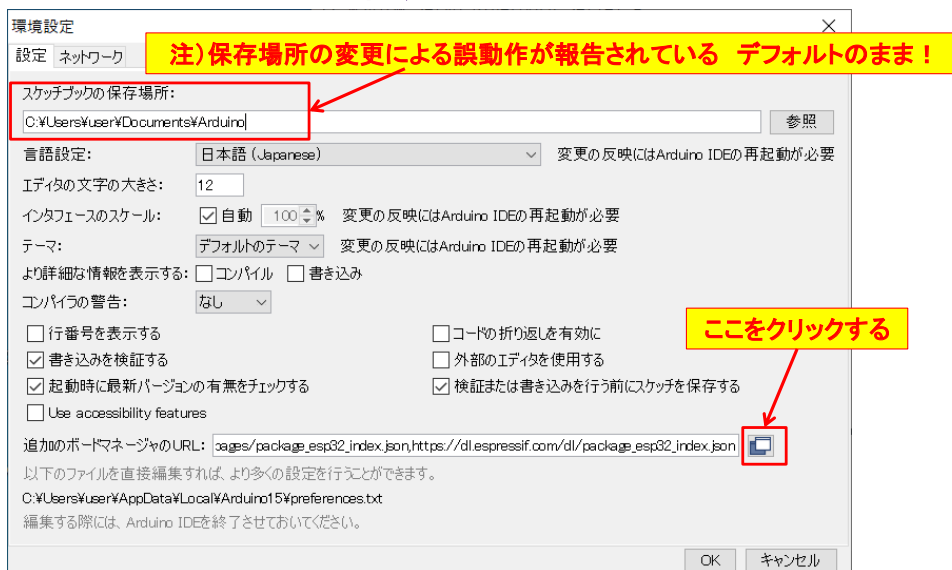
- ◇すべてをInstall → 眼鏡マークアイコンが作られる
- ◇IDEの中央白い部分にソースコードを記述する



環境設定



追加のボードマネージャのURL指定

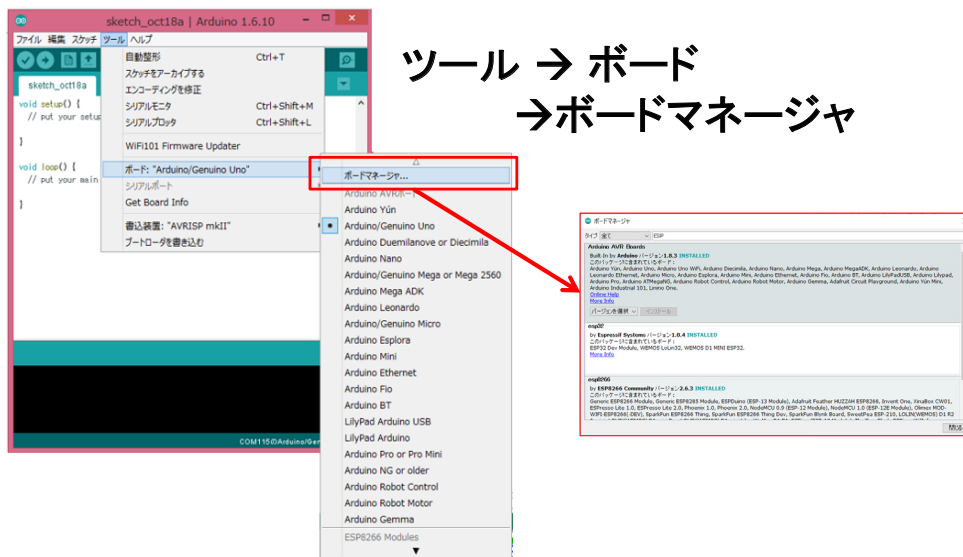


追加のボードマネージャのURL



https://dl.espressif.com/dl/package_esp32_index.json

ボードマネージャ



ツール → ボード
→ボードマネージャ

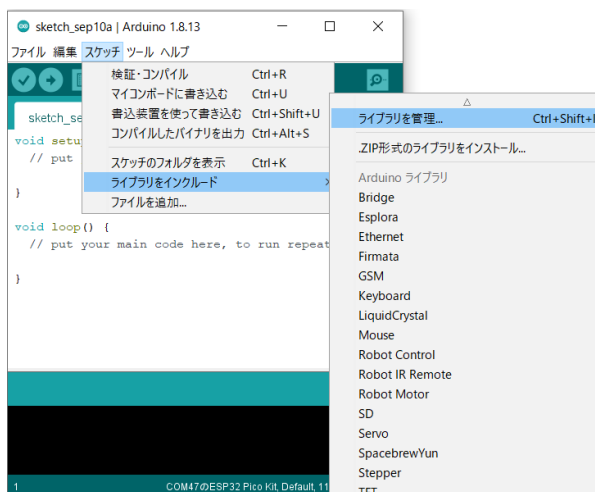
ESPマイコン用パッケージ

◇検索BOXに **ESP** と入力しESP32パッケージをインストール



対象マイコン用ライブラリの Install

◇スケッチ → ライブラリをインクルード → ライブラリを管理 とたどる



M5Atom・M5Stick-C・FastLEDライブラリの Install

◇ライブラリマネージャでM5Atom、FastLED、M5StickCを検索し、以下の3ライブラリをインストールする



LEDライブラリの修正

◇C:\Users\<user account>\Documents\Arduino\libraries\M5Atom\src\utility にある LED_Display.cpp にバグがある → 103行目を訂正する

```

97
98   offsetx = offsetx % xsize;+
99   offsety = offsety % ysize;+
100
101   int8_t setdatax = (offsetx < 0) ? (-offsetx) : (xsize - offsetx);+
102   int8_t setdatay = (offsety < 0) ? (-offsety) : (ysize - offsety);+
103   // xSemaphoreTake(_xSemaphore, portMAX_DELAY);+
104   xSemaphoreTake(_xSemaphore, 100); //<--- correct the source line!+
105   for (int x = 0; x < 5; x++)+
106   {+
107       for (int y = 0; y < 5; y++)+

```

マイコンボードの選択

◇IDEのツール → ボード → ESP32 Arduino → ESP Pico Kit または M5Stick-C を選択する

M5Atomの場合

- ESP32 Dev Module
- ESP32 Wrover Module
- ESP32 Pico Kit
- TinyPICO
- MagicBit
- Turta IoT Node
- TTGO LoRa32-OLED V1
- TTGO T1
- XinaBox CW02
- SparkFun ESP32 Thing
- u-blox NINA-W10 series (ESP32)
- Widora AIR
- Electronic SweetPeas - ESP320
- Nano32
- LOLIN D32
- LOLIN D32 PRO
- WEMOS LOLIN32
- Dongsen Tech Pocket 32
- WuMaker M5E8 Bluetooth Battery

M5Stick-Cの場合

- ThaiEasyElec's ESPino32
- M5Stack-Core-ESP32
- M5Stack-FIRE
- M5Stick-C
- ODROID ESP32
- Heltec WiFi Kit 32
- Heltec WiFi LoRa 32

この部分がない場合もある

M5Atomの場合、Upload Speedは115200にセットする
これより早いと書き込みに失敗する



COMポートの認識確認

- ◇マイコンをUSBケーブルでPCと接続し、COMポートが認識されることを確認しておく
- ◇Windowsのデバイスマネージャを開き、COMとLPTで新しいCOMポートが認識されない場合には、各マイコン用シリアルドライバ(CP210.x)をインストールする

※通常のインストールであれば、図の場所にシリアルドライバが格納されている

名前	更新日時	種類	サイ
amd64	2020/07/09 18:47	ファイル フォルダー	
CP210x_6.7	2020/07/09 18:47	ファイル フォルダー	
CP210x_6.7.4	2020/07/09 18:47	ファイル フォルダー	
FTDI USB Drivers	2020/07/09 18:47	ファイル フォルダー	

2.2 LEDの制御

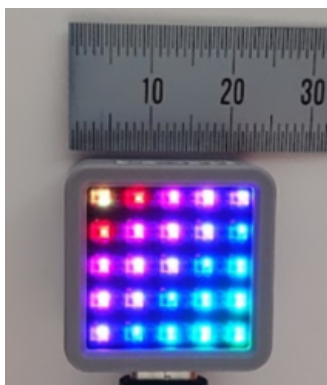
M5Atom

LED



実装しているLED

制御用



M5ATOM Matrix

- ◇M5ATOM Matrix には、5×5個のフルカラーLEDが実装されている
- ◇希望するLEDを点灯させることができれば、図のようにドットマトリクスで文字や図形パターン表示などが可能になる
- ◇このLEDは、信じられないかもしれないが1つずつ個別のマイコンを内蔵したWS2812C-2020というLEDである



LEDチップ(WS2812C-2020)データシート

General description

WS2812C-2020 is an intelligent control LED light source, its exterior adopts the latest MOLDING packaging technology, the control circuit and RGB chips are integrated in a package of 2020 component. Its internal includes intelligent digital port data latch and signal reshaping amplification drive circuit. Also include a precision internal oscillator and a voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.

The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel, the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.

RESET time >280μs , it won't cause wrong reset while interruption, it supports the lower frequency and inexpensive MCU.

Refresh Frequency updates to 2KHz, Low Frame Frequency and No Flicker appear in HD Video Camera, it improve excellent display effect.

LED with low driving voltage, environmental protection and energy saving, high brightness, large scattering angle, good consistency, low power, long life and other advantages. The control chip integrated in LED above becoming more simple circuit, small volume, convenient installation.

概要説明 General description

- ◇WS2812C-2020はインテリジェント(高機能)制御LED光源で、その外装は最近のモールドイングパッケージ技術を採用し、制御回路とRGBチップは2020コンポーネントのパッケージに集積されている。
- ◇高機能デジタルポートのデータラッチと信号整形アンプ駆動回路を内蔵している。
- ◇ピクセルが示す光色の高度な調和を効果的に確実にする内部発信回路と、電圧プログラム可能な電流制御部も含んでいる。
- ◇データ転送プロトコルは、単独の**NZR通信モード**を用いている。
- ◇ピクセルパワーオンリセット後、DIN(Digital IN)ポートはコントローラからデータを受け取り、1つ目のピクセルは最初の24bitデータを集めて、そのデータをラッチ(確保)し、内部信号再整形増幅回路により再整形される他のデータは、DO(Digital OUT)ポートを通じて、次の段のピクセルに送られる。
- ◇各ピクセル送信後、信号を24bitに減少する。
- ◇ピクセルは自動的に再整形送信技術を採用し、接続段数の作りは信号送信速度にだけ依存して信号送信を制限しない
- ◇280 μ sを超えるリセット時間は、割り込みの間の誤ったリセットは引き起こさず、低周波数で低価格なMCUをサポートする
- ◇リフレッシュ周期は2KHzで更新され、HDビデオカメラでもちらつきが現れず、優れた表示効果の改善をしている
- ◇低駆動電圧、環境保護そしてエネルギー保護、高輝度、大きな散乱角度、良好な整合性、低電力、長寿命かつ他の優位性を伴うLED
- ◇上記LED中に集積された制御チップは、シンプルで小型、設置が容易な回路になっている

NZR communication mode (NRZとは異なる)

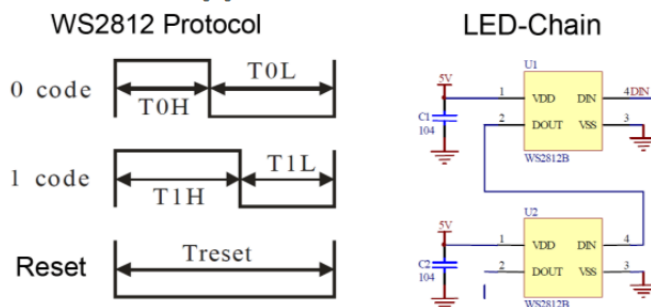
- ◇このモードの名称については、WEB上でも疑問が投げられている

What's the name of signal encoding used by WS2812 LEDs?

The WS2812b color LED uses some kind of duty cycle encoding to encode three states:

- one
 - zero
 - reset
- one zero reset** ← これが命名の元になっているらしい(分かり難い)

This can be seen in the following figure:

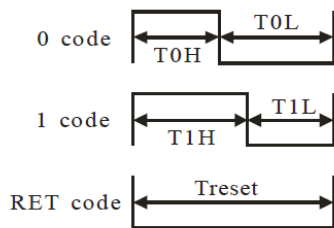


データ送信のタイミング

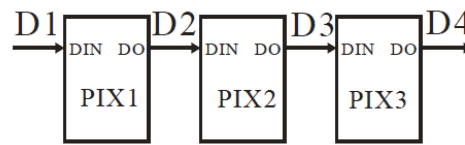
Data Transfer Time

T0H	0 code, high voltage time	220ns~380ns
T1H	1 code, high voltage time	580ns~1μs
T0L	0 code, low voltage time	580ns~1μs
T1L	1 code, low voltage time	580ns~1μs
RES	Frame unit, low voltage time	>280μs

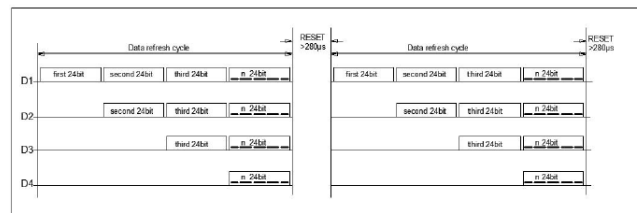
Sequence Chart



Cascade Method



データ送信方法 Data Transmission Method



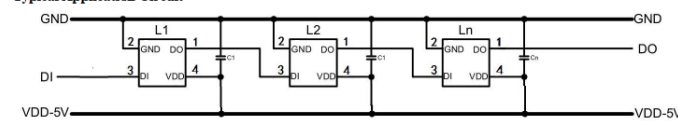
Note: The data of D1 is send by MCU, and D2, D3, D4 through pixel internal reshaping amplification to transmit.

Composition of 24bit Data

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Note: Data transmit in order of GRB, high bit data at first.

Typical Application Circuit

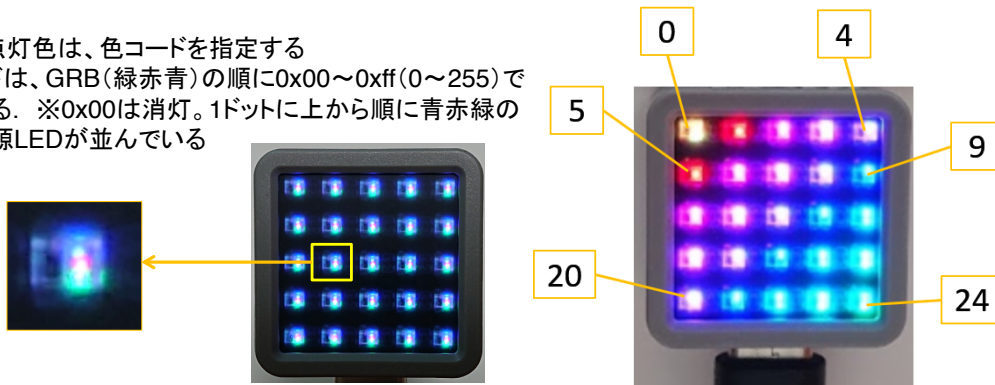


Remarks: C1 is the filter capacitor for VDD, its value of 100nF.

M5ATOMのLEDもこのように接続されている

LEDの番号

- ◇M5ATOM Matrix には、5×5個のフルカラーLEDが実装されている
- ◇希望するLEDを制御するには、LEDの番号と色を指定して次の関数を呼ぶ
M5.dis.drawpix(LED番号, 色)
- ◇LED番号はUSBコネクタを下に向けて、マトリクス左上から右に0,1,・・・, 4、下の段左から5,6,・・・と続き、右下最後のLEDは24番に割り当てられている
- ◇LEDの点灯色は、色コードを指定する
色コードは、GRB(緑赤青)の順に0x00~0xff(0~255)で指定する。 ※0x00は消灯。1ドットに上から順に青赤緑の順で光源LEDが並んでいる



一般社団法人全国専門学校情報教育協会

7



LED を1つずつG・R・Bで順に点灯してみよう！

LED巡回点灯

一般社団法人全国専門学校情報教育協会

8

システム構想

- ◇関数 M5.dis.drawpix(LED番号, 色) を呼び出す
- ①. LEDの番号は 0~24 で指定すれば良い
 - ②. 色番号は、0xff0000が緑、0x00ff00が赤、0x0000ffが青になっている
 - ③. 0~0xff(0~256)の範囲でLED各色の明るさを調整する
 - ④. 実際に関数に渡すときは、これらを3バイトにして

緑の場合 → 0xff0000

赤の場合 → 0x00ff00

青の場合 → 0x0000ff

として渡す(実際は4バイトで渡される)中間色の場合は、各色の明るさを次のようにする

紫の場合 → 0x00ffff (赤+青)

黄の場合 → 0xffff00 (赤+緑)

- ⑤. 渡す値を小さくすれば各色の明るさが変わる(ここでは0xff固定)
- ◇LEDを番号順に、緑→赤→青→消灯として1巡したら次のLEDに移る

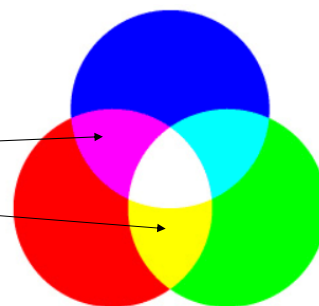
①. LED Indexを0~24で繰り返す

②. 色 Indexを0~3で繰り返す

③. 光色は、0x00ff0000の4バイトを(色 Index × 8bit)右シフトして渡す

※色Indexが0(最初)の時は緑。3回繰り返してIndexが3の時は24bit右シフトして0x00000000となりLEDは消灯(黒)になる

※このプログラムは、全LEDが正しく動くかどうか、確認するテストも兼ねている



光の三原色

ソースコード 1/2 (M5A_LED_1)

◇初期化部分までのソースコード(C++)

```
// M5ATOM LED 1
// LED 巡回点灯

#include <M5Atom.h> // マイコンボードライブラリ

uint8_t idx = 0; // LED番号
uint8_t col = 0; // 色番号 0:G 1:R 2:B 3:OFF

// 初期化
void setup() {
  M5.begin(true, false, true); // SerialEnable , I2CEnable , DisplayEnableの順
  delay(1000); // ここにWaitを入れないと、初めのLEDが点灯しない
  // --->搭載しているLEDには、マイコンが内蔵されているので、
  // そちらとの間の初期化時間も考慮する必要があるのかもしれない
}
```

TABキーで行頭をそろえる

ソースコード 2/2 (M5A_LED_1)

◇初期化部分までのソースコード(C++)

```
// 通常処理
void loop() {
  if (1==1) // このように記述するのは意味が無いように思えるが、
           // 近い将来役立つ
  {
    // LED番号 と 色番号を指定してLED点灯
    M5.dis.drawpix(idx, 0x00ff0000 >> (col*8));
    col++; // 色番号更新
    if (col >= 4){ // 色番号が4になったら、0に戻す
      col=0;
      idx++; // 3色点灯終わったときに、LED番号更新
      if (idx >= 25){ // LED番号が25になったら0に戻す
        idx = 0;
      }
    }
  }
  delay(500); // しばし待つ
}
```

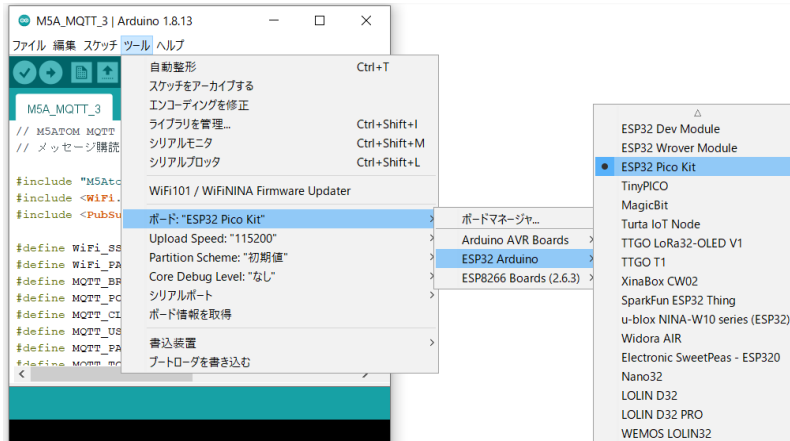
ライブラリ FastLEDのインストール

◇内部で使用している【FastLEDライブラリ】をインストールしておく

The image shows two screenshots of the Arduino IDE Library Manager. The top screenshot shows the search results for 'FastLED' by Daniel Garcia. The 'インストール' (Install) button is highlighted with a red box. A yellow arrow points down to the second screenshot, which shows the library status as 'INSTALLED'.

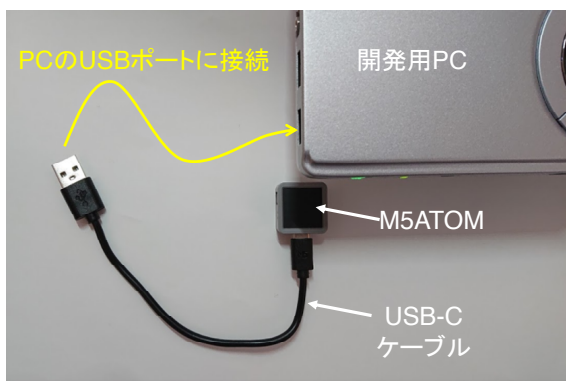
マイコンボードの選択

- ◇以下のように IDE で **ツール** → **ボード** → **…とたどり、ESP32 Pico Kit** を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、**M5ATOMを使用する場合は、必ずこの設定で行う**

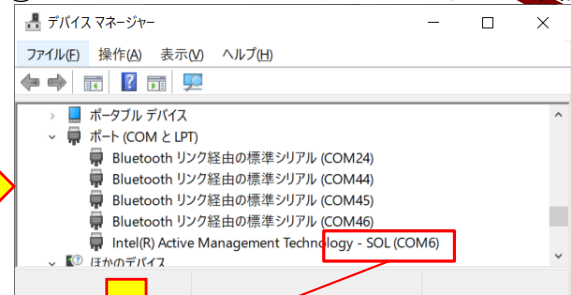


マイコンをPCと接続

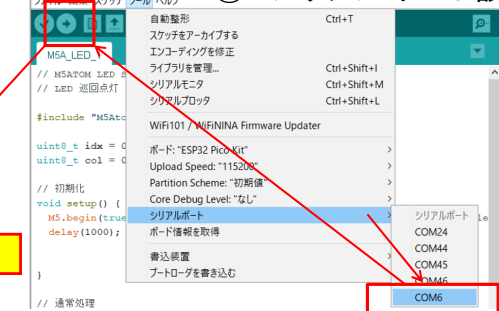
①. M5ATOMをPCと接続



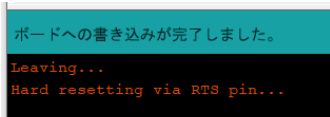
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

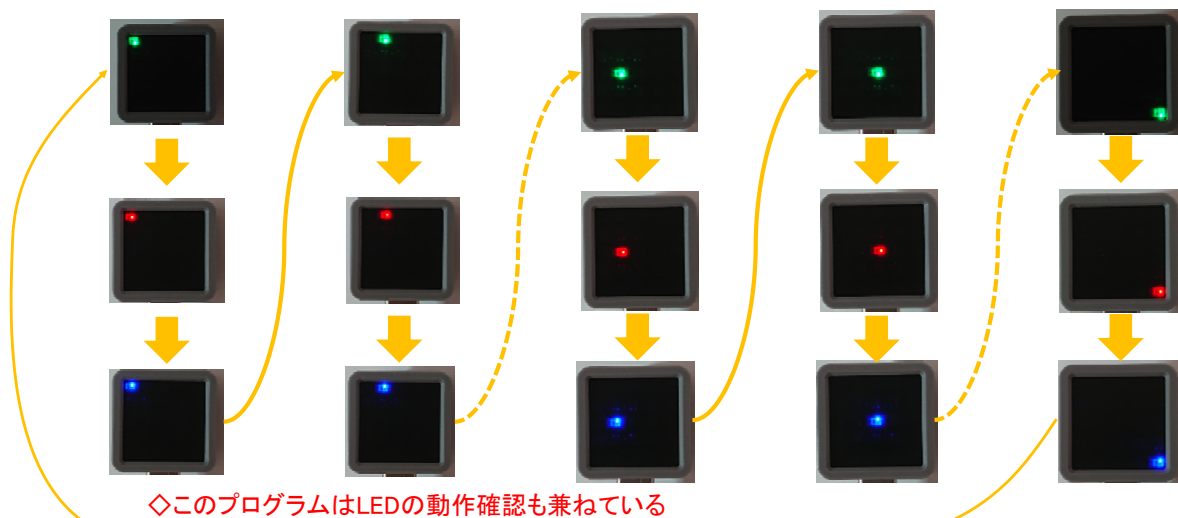


④. コンパイル



動作確認

- ◇新しいプログラムは書き込みが終了すると、自動的にスタートしている
- ◇LEDの点灯の様子を観察する中で、特に最初と最後のLEDの点灯は、注意深く見る



LEDを段階的に色を変化させて表示してみよう

LED GRADATION

ソースコード 1/2 (M5A_LED_4)

```
#include "M5Atom.h"

int ptr = 0; // 12色パターンを示すインデックス
CRGB colors[] = { // 12の色パターン あらかじめ中間の色も準備
  0xff0000,      //G
  0xff8000,
  0xffff00,      //G+R
  0x80ff00,
  0x00ff00,      //R
  0x00ff80,
  0x00ffff,      //R+B
  0x0080ff,
  0x0000ff,      //B
  0x8000ff,
  0xff00ff,      //G+B
  0xff0080
};

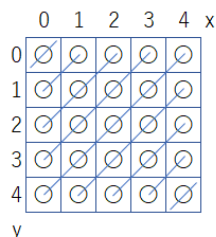
void setup() {
  M5.begin(true, false, true); // SerialEnable , I2CEnable , DisplayEnable
}
```

◇CRGB はFirstLED.hで#includeしている
pixcetypes.hで定義されている色指定構造体

ソースコード 2/2 (M5A_LED_4)

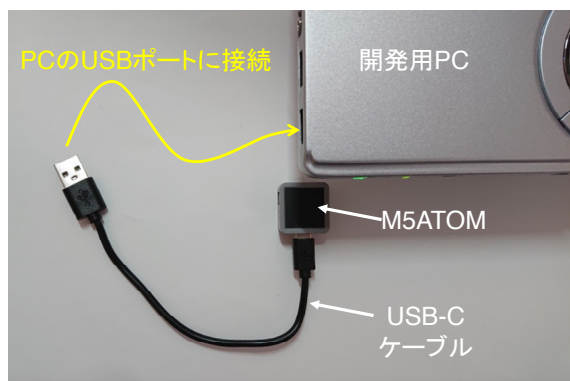
```
void loop() {
  int i, j, t, x, y, p;
  p = ptr;
  for (i = 0; i < 9; i++) {
    t = 4 - abs(4 - i);
    for (j = 0; j <= t; j++) {
      if (i <= 4) {
        x = j;
        y = t - j;
      }
      else {
        x = (4 - t) + j;
        y = 4 - j;
      }
      M5.dis.drawpix(x, y, colors[p]); // (x,y)座標指定表示
    }
    p++;
    p = (p == 12) ? 0 : p;
  }
  delay(100);
  ptr++;
  ptr = (ptr == 12) ? 0 : ptr;
}
```

◇この部分で図の斜線のように、LEDの座標を作り出している

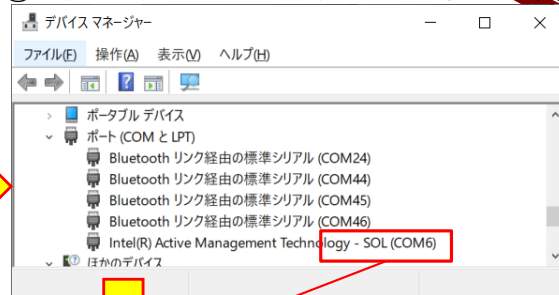


マイコンをPCと接続

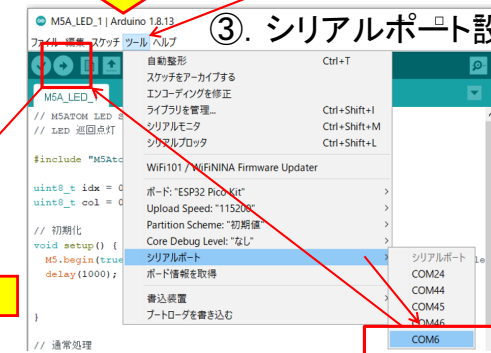
①. M5ATOMをPCと接続



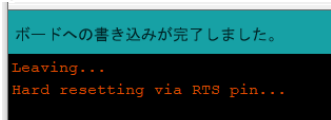
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル

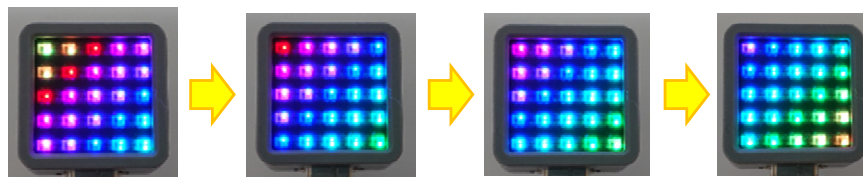


一般社団法人全国専門学校情報教育協会

19

動作確認

◇書き込みが完了するとマイコンがリセットされて、図のようにグラデーション表示が行われる



一般社団法人全国専門学校情報教育協会

20

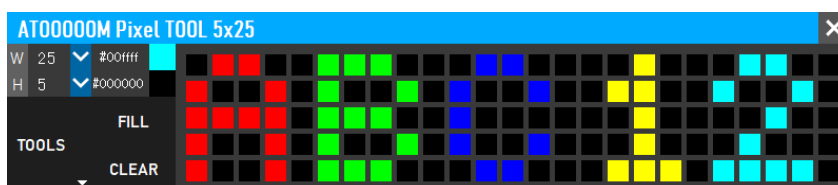


情報伝達に利用できる文字フォントのスクロール表示

LED MATRIX

ATOM PIXEL TOOL

- ◇1色8bitでGRBの並びの3バイトのデータがあればLEDを1ドット点灯させることができる
- ◇これを5×5ドットのフォントとして表示してみよう
- ◇表示にはM5ATOMのライブラリに含まれる `M5.dis.displaybuff()` という関数を使う
- ◇フォントデータは手作業で作ることもできるが、M5ATOM製造元が公開しているツールがある
 - ※ATOM PIXEL TOOLはexeファイルで提供されるのでインストールは不要



- ◇フォントデータのサイズと、色を決めてマウスでドットをクリックするとパターンができる
- ◇TOOLSの下の参画印をクリックして適当な名称で保存すれば、IDEで使用できるソースコードに変換される



フォントデータの例

◇文字列“ABC12”をフォントデータにすると下図のようになる

```
// Image Size: width=25,height=5+
// Data Size: 377 +
const unsigned char image_font[377]=
{
/* width 025 */ 0x19,+
/* height 005 */ 0x05,+
/* Line 000 */ 0x00,0x00,0x00, 0xff,0x00,0x00, 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0x00, 0x00,
0xff,0x00, 0x00,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0xff, 0x00,0x00,0xff, 0x00,0x00,
0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00,
0x00,0xff,0xff, 0x00,0xff,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, // +
/* Line 001 */ 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,
0x00, 0x00,0x00,0x00, 0x00,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,
0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0xff,0x00, 0xff,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0xff,
0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0xff, 0x00,0x00,0x00, // +
/* Line 002 */ 0xff,0x00,0x00, 0xff,0x00,0x00, 0xff,0x00,0x00, 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0x00, 0x00,
0xff,0x00, 0x00,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,
0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00,
0x00,0x00,0x00, 0x00,0xff,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, // +
/* Line 003 */ 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0x00, 0x00,
0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,
0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,
0x00, 0x00,0xff,0xff, 0x00,0x00,0x00, 0x00,0x00,0x00, // +
/* Line 004 */ 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0x00,0x00, 0x00,0x00,0x00, 0x00,0xff,0x00, 0x00,
0xff,0x00, 0x00,0xff,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0x00,0x00,0xff, 0x00,0x00,0xff, 0x00,0x00,
0x00, 0x00,0x00,0x00, 0x00,0x00,0x00, 0xff,0xff,0x00, 0xff,0xff,0x00, 0xff,0xff,0x00, 0x00,0x00,0x00, 0x00,0xff,0xff,
0x00,0xff,0xff, 0x00,0xff,0xff, 0x00,0xff,0xff, 0x00,0x00,0x00, // +
};+

```

ソースコード 1/2 (M5A_LED_6)

◇初期化部分までのソースコード(C++)

```
#include "M5Atom.h" ◇この部分はATOM PIXEL TOOLの出力ファイルの内容そのまま

// Image Size: width=25,height=5
// Data Size: 377
const unsigned char image_font[377]=
{
/* width 025 */ 0x19, // 全体の横幅(5ドット×文字数)
/* height 005 */ 0x05, // 一文字の高さ
/* Line 000 */ 0x00,0x00,0x00,....., 0x00,0x00,0x00, // ...はデータ省略
/* Line 001 */ 0xff,0x00,0x00,....., 0x00,0x00,0x00, // ...はデータ省略
/* Line 002 */ 0xff,0x00,0x00,....., 0x00,0x00,0x00, // ...はデータ省略
/* Line 003 */ 0xff,0x00,0x00,....., 0x00,0x00,0x00, // ...はデータ省略
/* Line 004 */ 0xff,0x00,0x00,....., 0x00,0x00,0x00, // ...はデータ省略
};
```

ソースコード 2/2 (M5A_LED_6)

◇初期化および通常の処理部分

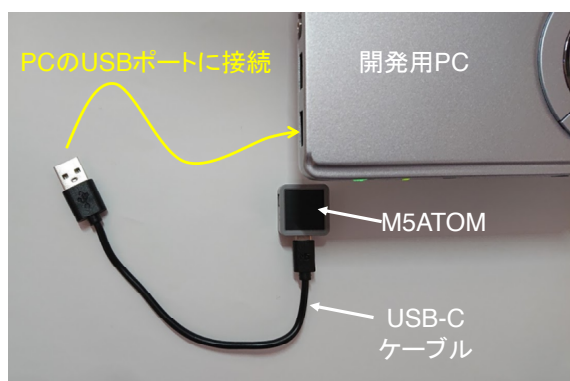
```
void setup()
{
  M5.begin(true, false, true);
}

void loop()
{
  int i; // 変数 i は、スクロールの向きとドット数を表す
        // マイナス→左向き プラス→右向き

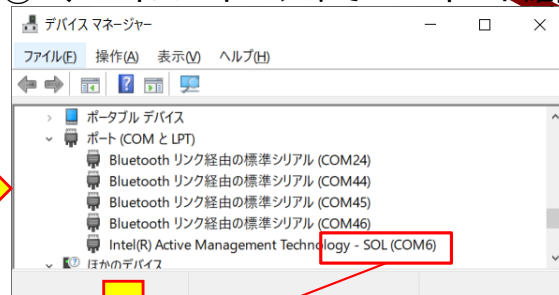
  for(i=0; i > -25; i--){
    M5.dis.displaybuff((uint8_t*)image_font, i, 0); // ツールが出力したデータの名称に合わせる
    delay(500);
  }
}
```

マイコンをPCと接続

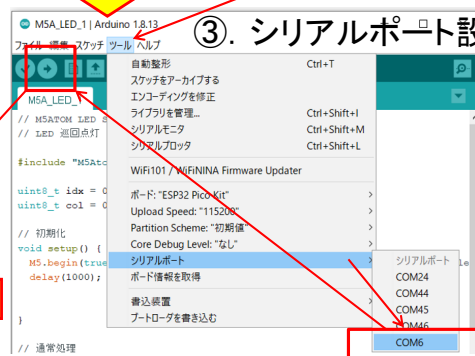
①. M5ATOMをPCと接続



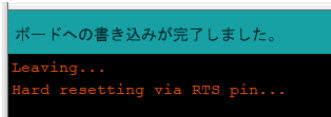
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



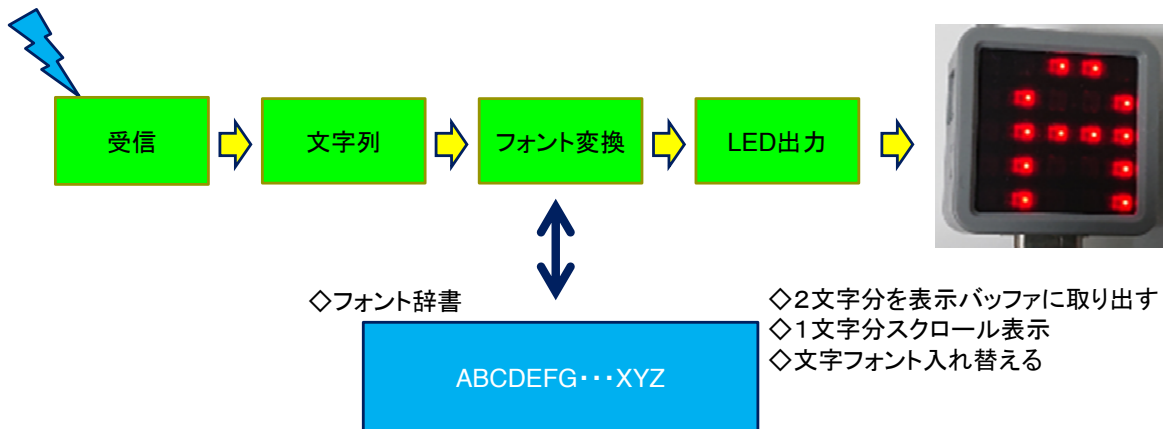
動作確認

◇設定した文字フォントが繰り返しスクロール表示される



応用開発

◇フォントパターンを埋め込んだデータは変わらないので、このままでは表示内容は変化しない
◇外部から受け取った文字列を自在に表示できるようにするにはどうすればよいか？



◇通信は後付けて、メモリ上の変数領域から文字列を表示できればよい(フォント辞書を根気よく作る)

2.3 Button クラスの制御

M5Atom

Button(押しボタンスイッチ)



一般社団法人全国専門学校情報教育協会

M5ATOM Buttonクラス

- ◇M5ATOMには、ライブラリ中にButtonというSW(スイッチ)を取り扱うクラスがある
- ◇その中でボタンの状態を検出できる関数が定義されている
- ◇IDEのデフォルトのスケッチ保存先の以下のパスにヘッダファイルがある
C:¥Users¥user¥Documents¥Arduino¥libraries¥M5Atom¥src¥utility¥Button.h
- ◇ヘッダファイル内のButtonクラス定義部分を下に示す

```
class Button {  
public:  
    Button(uint8_t pin, uint8_t invert, uint32_t dbTime);  
    uint8_t read();  
    uint8_t isPressed();  
    uint8_t isReleased();  
    uint8_t wasPressed();  
    uint8_t wasReleased();  
    uint8_t pressedFor(uint32_t ms);  
    uint8_t releasedFor(uint32_t ms);  
    uint8_t wasReleasefor(uint32_t ms);  
    uint32_t lastChange();  
  
    ...  
};
```

一般社団法人全国専門学校情報教育協会

1

Buttonクラスの機能

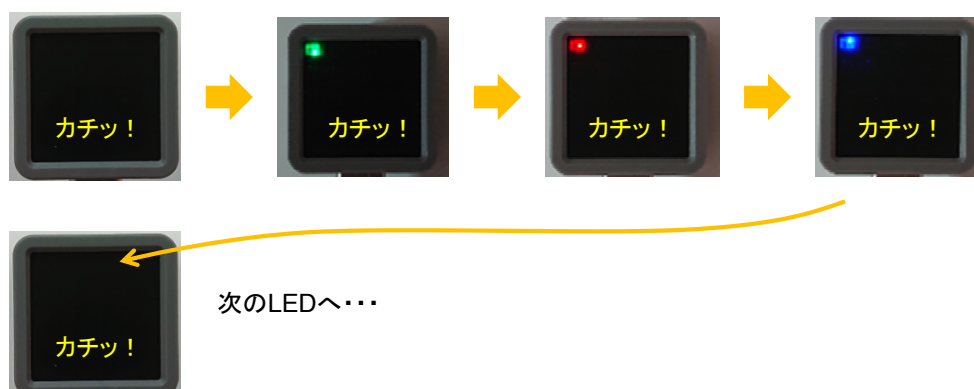
- ◇Buttonの関数は、以下の機能がある
- ◇ここでは、wasPressed() を用いてボタンの押下状態を調べる
- ◇ボタンの状態は M5.update() を用いて更新される

関数	機能
M5.update()	ボタンの状態を更新する関数 ※loop()関数内で必ず実行する
isPressed()	ボタンを押しているかどうかを返す ※ボタンを押している間は常にTRUEが戻る
isReleased()	ボタンを離しているかどうかを返す ※ボタンを押していない間は常にTRUEが戻る
wasPressed()	ボタンを押してから最初に呼び出した時だけ、TRUEを返す
wasReleased()	ボタンを押して、離してから最初に呼び出した時だけTRUEを返す
pressedFor(ms)	ボタンを指定時間以上押している場合にTRUEが返される
releasedFor(ms)	ボタンを離してから指定時間以上経過している場合にTRUEが返れる
wasReleasefor(ms)	指定時間以上ボタンを押し、離してから最初に呼び出した時だけTRUEを返す
lastChange()	最後にボタンの状態が変更された時の millis() の値が返却される 現在のmillis()からの差分が経過時間になる

システム構想

- ◇ボタン(LED表示部)を押下するたびに、次の動作を繰り返す

消灯 → G → R → B → 次のLEDへ



ソースコード 1/2 (M5A_Button_1)

◇初期化部分までのソースコード
※M5A_LED_1 と同じ

```
#include <M5Atom.h> // マイコンボードライブラリ

uint8_t idx = 0; // LED番号
uint8_t col = 0; // 色番号 0:G 1:R 2:B 3:OFF

// 初期化
void setup() {
  M5.begin(true, false, true); // SerialEnable, I2CEnable, DisplayEnable
  delay(1000); // ここにWaitを入れないと、初めのLEDが点灯しない
                // --->搭載しているLEDには、マイコンが内蔵されているので、
                // そちらとの間の初期化時間も考慮する必要があるらしい
}
```

ソースコード 2/2 (M5A_Button_1)

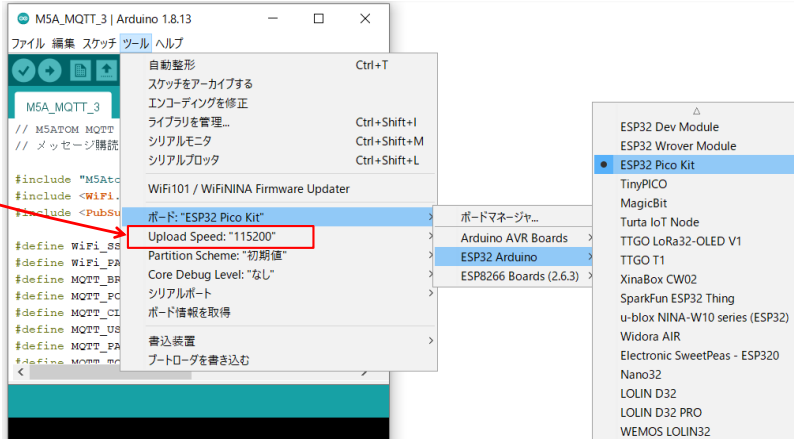
◇通常処理部分
※ボタンの押下状態を調べるために `M5.Btn.wasPressed()` を呼ぶ
また、ボタン状態の更新のために `M5.update()` を呼ぶ (他は M5A_LED_1 と同じ)

```
void loop() {
  M5.update(); // ボタン状態を更新する
  if (M5.Btn.wasPressed()) // ボタンの押下状態を調べる
  {
    // ボタンが押されていたらTrue
    // LED番号 と 色番号を指定してLED点灯
    M5.dis.drawpix(idx, 0x00ff0000 >> (col*8));
    col++; // 色番号更新
    if (col >= 4){ // 色番号が4になったら、0に戻す
      col=0;
      idx++; // 4色点灯終わったときに、LED番号更新
    }
    if (idx >= 25){ // LED番号が25になったら0に戻す
      idx = 0;
    }
  }
}
}
```

※【近い将来役立つ】部分の意味がここで分かる

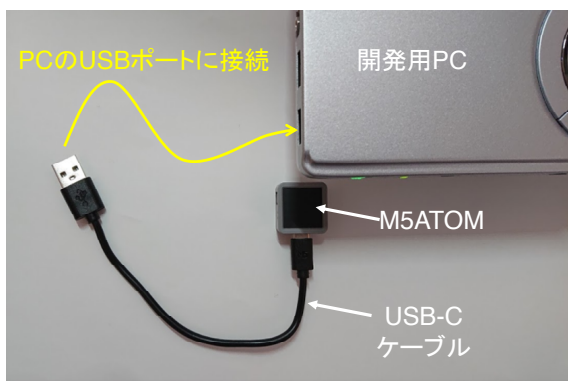
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う

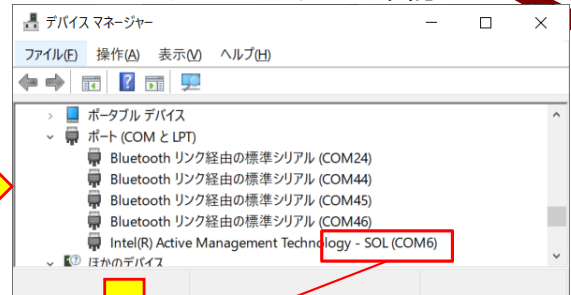


マイコンをPCと接続

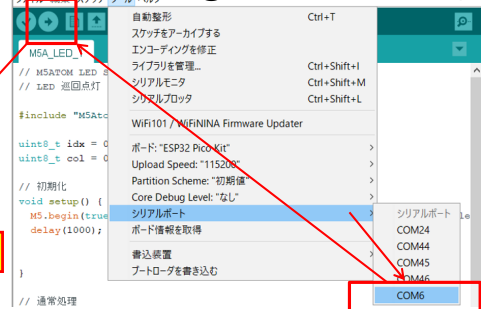
①. M5ATOMをPCと接続



②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

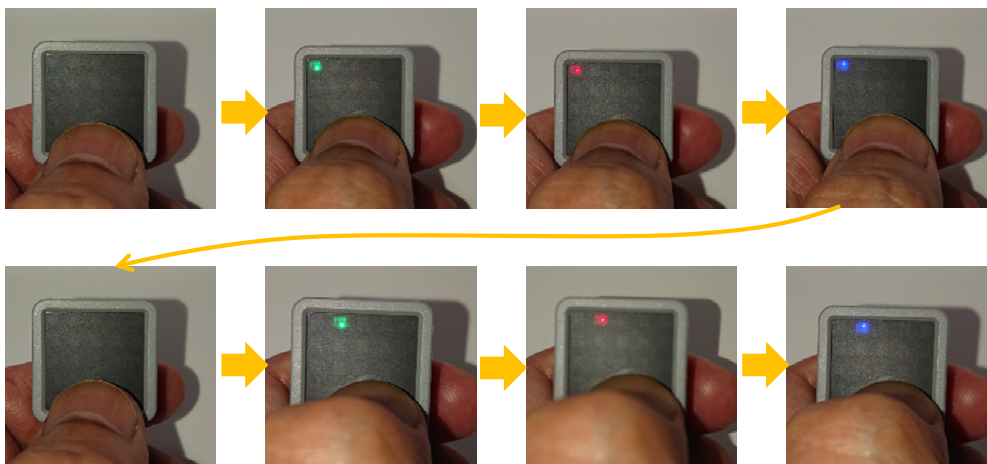


④. コンパイル



動作確認

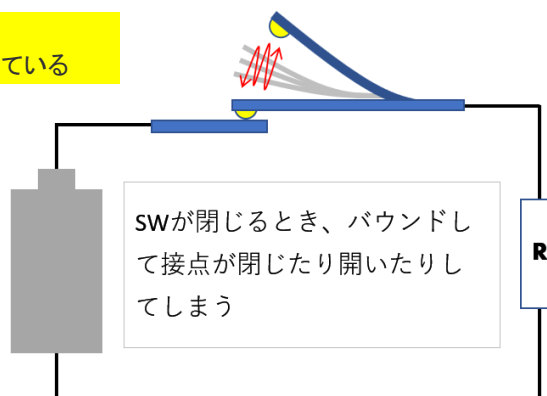
- ◇新しいプログラムは書き込みが終了すると、自動的にスタートしている
- ◇ボタンを押下するたびに、次の動作を繰り返すことを確認しよう！
消灯→G→R→B→次のLEDへ



チャタリング

- ◇機械式SWで、SWの接点が閉じる際に、1度で開閉せず、バウンドしてON→OFF→ON…を何度か繰り返してしまう現象が発生する(時間にして数ms~10ms) 下図
- ◇何も対策しなければ、機械の誤動作などを招くこともあり、危険
- ◇対応策は、回路の工夫でもできるが、一般的にソフトウェアで対策することが多い
→ 数msの間隔において複数回調べて同じ状態が続くことを確認する、等

◇Buttonクラスを用いれば、
クラス内部でチャタリング対策も行われている

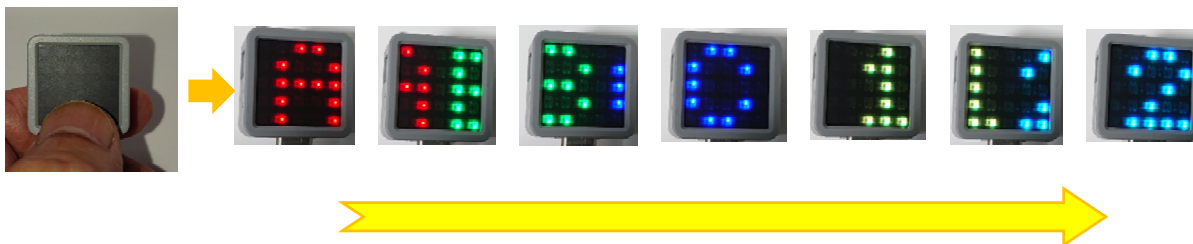




応用開発

◇LEDの文字フォントスクロール表示をボタンを押すたびに1回行うようにするにはどうすればよいか？
※このようなプログラムが開発できれば、来訪者が受付前に立った時に【ようこそ】と表示ができるし、通信でスクロール表示をトリガーすることもできる！！

◇ボタンを押下 → スクロール表示 …



2.4 シリアル通信の制御

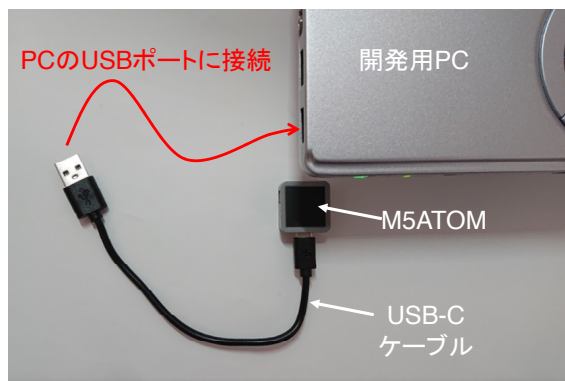
M5Atom

シリアル通信



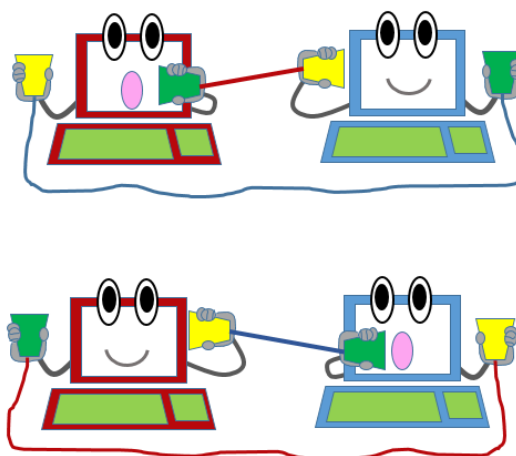
M5ATOMはPCとUSB接続される

- ◇M5ATOMは、プログラム書き込みの際にはPCとUSB接続します
- ◇マイコンはUSB経由のシリアル通信によって受け取ったプログラムをフラッシュメモリに書き込みます
- ◇今回は、このシリアル通信を利用して、PCとの間で通信を行います



シリアル通信のイメージ

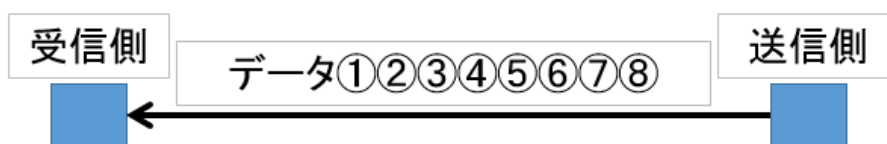
- ◇シリアル通信のイメージは【糸電話】



シリアル通信は・・・

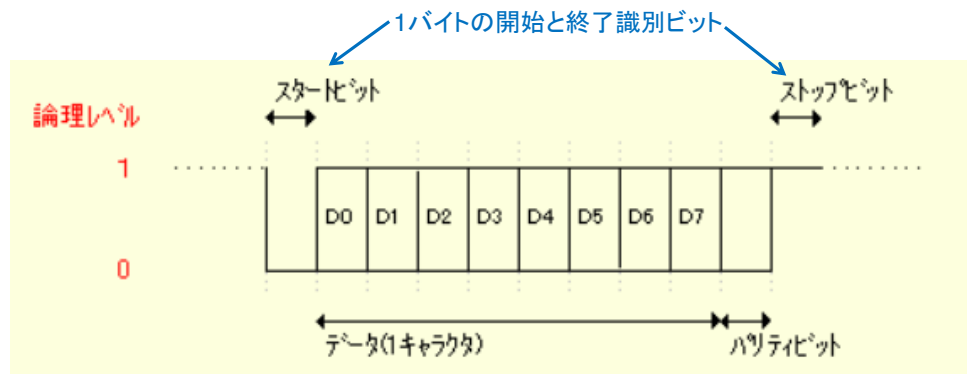
◇最も多用されている通信方式です。

- ✓ PC間、PC・マイコン・装置間など、多く利用される
- ✓ 1本の信号線で1ビットずつデータを伝送する
- ✓ 基本的に1対1の通信方式



信号線の中のデータ

- ◇1バイトのデータ通信(調歩同期式通信を示す)
- ※左に向かってデータが送られる様子
- 複数バイトの場合でも、以下が繰り返される



- ◇誤り検出のためパリティビットが付加できる

文字 A(0x41)のデータ

◇半角アルファベット「A」(ASCIIコードで0x41)のデータを偶数パリティで送信すると、図のようになる

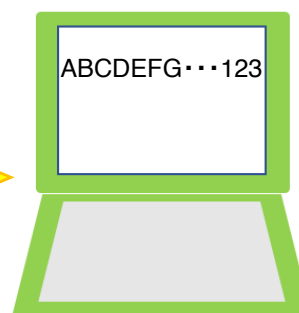
D0	D1	D2	D3	D4	D5	D6	D7	P
1	0	0	0	0	0	1	0	0

‘A’のデータ(偶数パリティ)

※下位(D0)のビットから送られる

実際のデータはD7が上位なので、(パリティ+0x41)となる

※偶数パリティ → パリティビットも含めて全ビットをビット加算した結果が0になるようにパリティビットをセットする



固定のメッセージをPCに向けて送信する

送信

システム構想

- ◇単純なメッセージ送信を行う
- ◇メッセージは固定
- ◇既にLEDとボタンが使えるようになっている
 - ボタン押下に同期して【LED点灯制御+メッセージ送信】を行う

◇以下のライブラリ関数が準備されている

- ①. ボタン押下 → M5.update() と M5.Btn.wasPressed()
- ②. LED点灯制御 → M5.dis.drawpix(LED番号, 色コード)
- ③. メッセージ送信 → Serial.print() または Serial.println()

◇過去の資産(ボタンのプログラム)の流用を考えて進める

シリアル通信のデフォルト速度

- ◇M5ATOMは、シリアルポートの初期化処理も、デフォルトでライブラリが処理をする
- ◇C:\Users\user\Documents\Arduino\libraries\M5Atom\src\M5Atom.cpp の該当部分を示す

```
void M5Atom::begin(bool SerialEnable , bool I2CEnable , bool DisplayEnable )+
{+
    if( _isInited ) return;+
+
    _isInited = true;+
+
    if( I2CEnable )+
    {+
        Wire.begin(25,21,10000);+
    }+
    if( SerialEnable )+
    {+
        Serial.begin(115200);+
        Serial.flush();+
        delay(50);+
        Serial.print("M5Atom initializing..");+
    }+
}
```

※この部分でシリアルポートの初期化が行われている Serial.begin()のパラメータが通信速度
初期メッセージは改行コードを出力していないので Serial.println() に変更しても良い

ソースコード 1/2 (M5A_Serial_1)

◇初期化処理まで (この部分は、ボタンのプログラムと同じ)

```
#include <M5Atom.h> // マイコンボードライブラリ

uint8_t idx = 0; // LED番号
uint8_t col = 0; // 色番号 0:G 1:R 2:B 3:OFF

// 初期化
void setup() {
  M5.begin(true, false, true); // SerialEnable, I2CEnable, DisplayEnable
  delay(1000); // ここにWaitを入れないと、初めのLEDが点灯しない
               // --->搭載しているLEDには、マイコンが内蔵されているので、
               //   そちらとの間の初期化時間も考慮する必要があるらしい
}
```

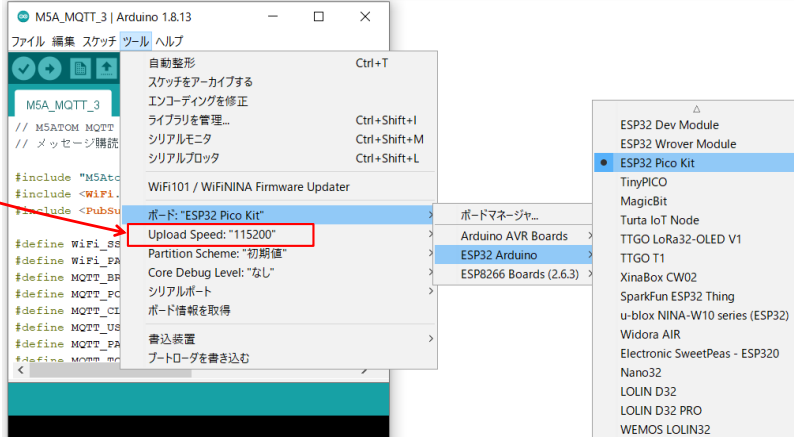
ソースコード 2/2 (M5A_Serial_1)

◇通常処理では `Serial.println("ABC123abc!");` でメッセージを送信している

```
// 通常処理
void loop() {
  M5.update(); // ボタン状態を更新する
  if (M5.Btn.wasPressed()) { // ボタンの押下状態→押されていたらTrue
    Serial.println("ABC123abc!"); // メッセージ送信 改行コード付き
    M5.dis.drawpix(idx, 0x00ff0000 >> (col*8)); // LED番号と色番号を指定してLED点灯
    col++; // 色番号更新
    if (col >= 4){ // 色番号が4になったら、0に戻す
      col=0;
      idx++; // 3色点灯終わったときに、LED番号更新
      if (idx >= 25){ // LED番号が25になったら0に戻す
        idx = 0;
      }
    }
  }
}
```

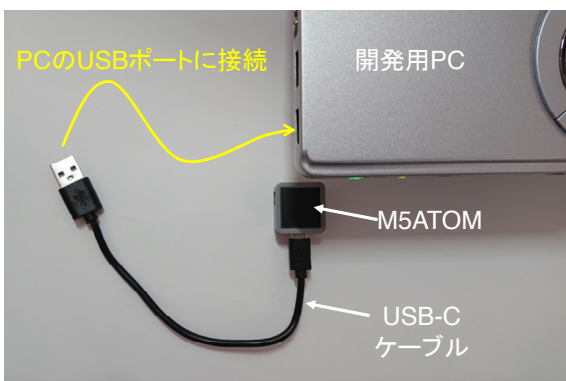
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う

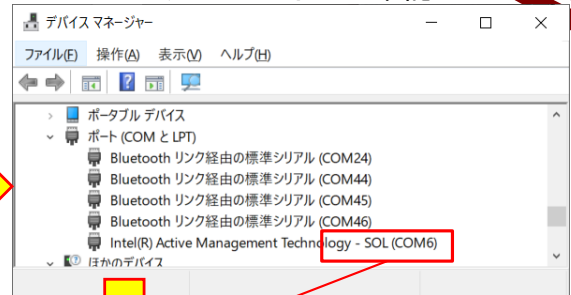


マイコンをPCと接続

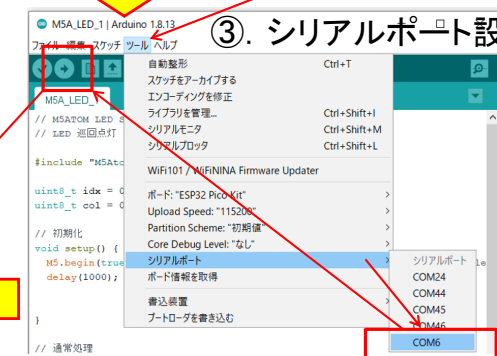
①. M5ATOMをPCと接続



②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

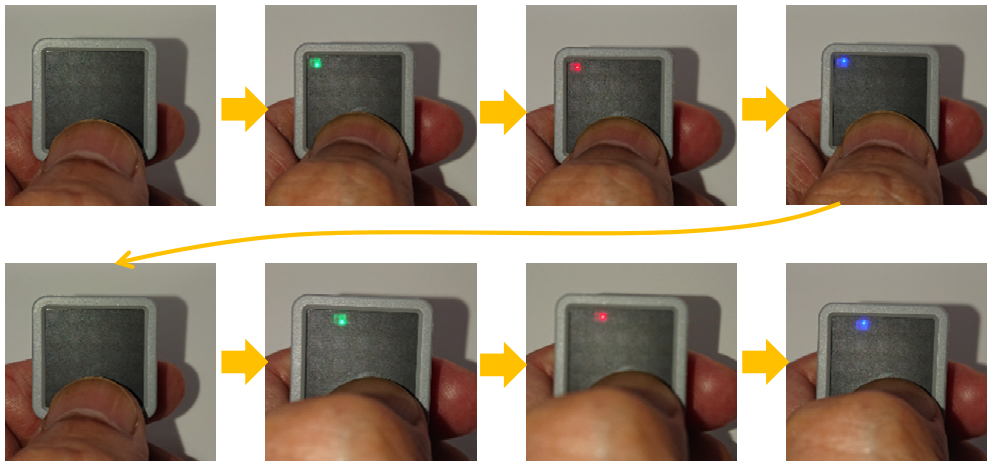


④. コンパイル



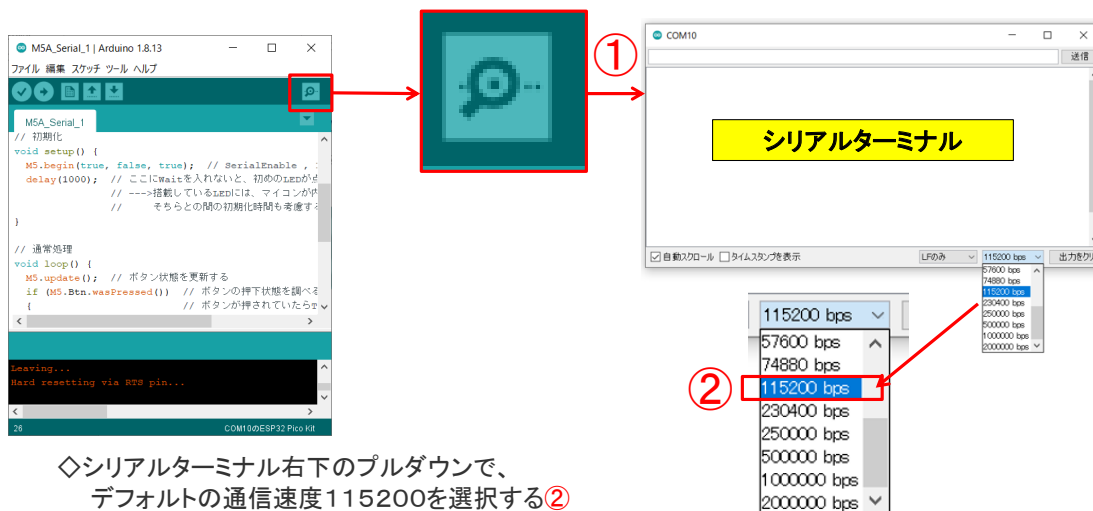
動作確認 1/3

- ◇新しいプログラムは書き込みが終了すると、自動的にスタートしている
- ◇ボタンを押下するたびに、次の動作を繰り返すことを確認しよう！
消灯→G→R→B→次のLEDへ



動作確認 2/3 メッセージ送信の確認準備

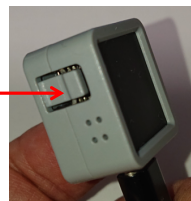
- ◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①



- ◇シリアルターミナル右下のプルダウンで、
デフォルトの通信速度115200を選択する②

動作確認 3/3 メッセージ送信の確認

- ◇M5ATOMの左側にある **RESET SW** を押す
 - ※シリアルターミナルに**リセット時メッセージ**が表示される
- ◇M5ATOMのLED部分を押し下るとLEDが順次点灯しながら**メッセージを受信する様子が確認できる**



COM10

```

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_F
configsip: 188777542, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_c
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
M5Atom initializing...
ABC123abc!
ABC123abc!
ABC123abc!
            
```

} 固定メッセージ

自動スクロール タイムスタンプを表示

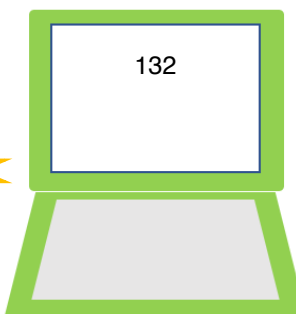
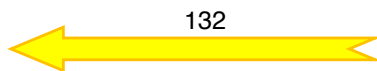
COM10

```

ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST
configsip: 188777542, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
M5Atom initializing...
            
```

自動スクロール タイムスタンプを表示

15



PCからメッセージを受信する

受信

16

システム構想

- ◇次は受信を行う
- ◇受信メッセージでLEDを制御しよう
- ◇メッセージ設計と処理設計
 - ①. メッセージ長 : 3バイト+改行コード
 - ②. 改行コードまで受信して、メッセージの解析と対応する処理を行う
 - ③. 受信バッファは、文字型配列で10バイト程度確保する(誤った長いメッセージに対応)
 - ④. メッセージの内容
 - 0文字目:LED X座標 0~4
 - 1文字目:LED Y座標 0~4
 - 2文字目:色 1:青 2:赤 3:緑 4:消灯 0:全LED消灯

◇以下のライブラリ関数を使用する

- ①. Serial.available() → 受信バッファにデータがあるか調べる
- ②. Serial.read() → 1文字受信する
- ③. M5.dis.drawpix(LED番号, 色コード) → LED番号と色を指定してLED点灯
- ④. M5.dis.drawpix(X座標, Y座標, 色コード) → LED座標と色を指定してLED点灯

1

ソースコード 1/5 (M5A_Serial_2)

◇初期化処理まで

```
#include <M5Atom.h> // マイコンボードライブラリ

int i; // 1文字受信時のバッファインデックス
char buf[10]; // メッセージバッファ

void fill_led(unsigned long col){ // LEDをすべて同じ色にする 全消灯に利用する
  for(int i=0; i<25; i++){
    M5.dis.drawpix(i, col); // LED番号を0~24まで指定して色を設定する
  }
}

void setup(){ // 初期化
  M5.begin(true, false, true); // SerialEnable, I2CEnable, DisplayEnable
  delay(1000); // ここにWaitを入れないと、初めのLEDが点灯しない
  // --->搭載しているLEDには、マイコンが内蔵されているので、
  // そちらとの間の初期化時間も考慮する必要があるらしい
  i=0; // バッファインデックス初期化
}
```

ソースコード 2/5 (M5A_Serial_2)

◇通常処理 最も外側部分

```
void loop() { // 通常処理
  char c;    // 受信データ 1文字分
  int x;     // LED X座標
  int y;     // LED Y座標
  int col;   // LED 色指定
  unsigned long color; // 色コード

  // 受信処理
  //座標・色コード計算
  //エラーチェック
  //色コード計算・LED点灯
}
```

ソースコード 3/5 (M5A_Serial_2)

◇通常処理 受信処理・電文解析・座標計算

```
if(Serial.available()){ // 受信データあり?
  c = Serial.read();    // 1文字 Read
  buf[i++] = c;         // 受信した文字を格納しインデックスを+1する } // 受信処理

if(c == '\n'){         // 改行ならば電文の終端
  i=0;                 // バッファインデックス初期化
  x = buf[0]-'0';      // X座標計算
  y = buf[1]-'0';      // Y座標計算
  col = buf[2]-'0';    // 色番号計算 } // 座標・色番号計算

  //エラーチェック

  //色コード計算・LED点灯
}
```

ソースコード 4/5 (M5A_Serial_2)

◇エラーチェック部

```
if(x<0 || x>4){ // X座標チェック
  Serial.println("Invalid Position X!!");
  return; // エラーメッセージを送信してOSに戻る
}

if(y<0 || y>4){ // Y座標チェック
  Serial.println("Invalid Position Y!!");
  return; // エラーメッセージを送信してOSに戻る
}

if(col<0 || col>4){ // 色コードチェック
  Serial.println("Invalid Color!!");
  return; // エラーメッセージを送信してOSに戻る
}
```

ソースコード 5/5 (M5A_Serial_2)

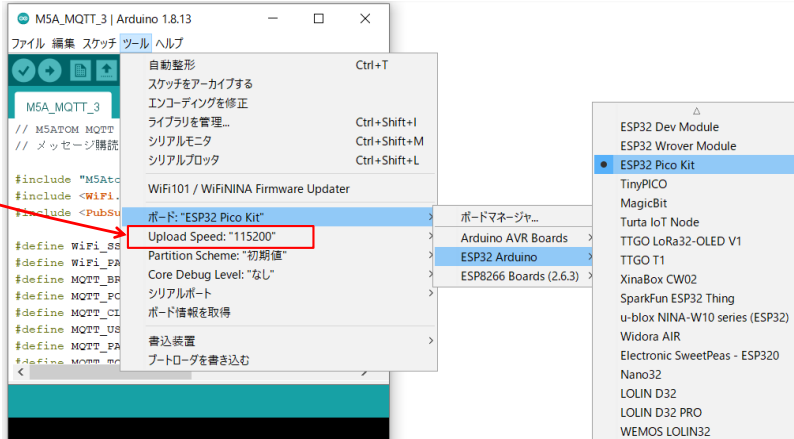
◇色コード計算・LED点灯処理

```
if(col == 0){
  fill_led(color = 0x000000); // ← 冒頭で宣言した関数をCallして、LED全消灯する
  return; // LEDをすべて消灯
} else if(col == 4) {
  color = 0x000000; // 色コード 黒 = 消灯
} else {
  color = 0x0000ff << (col-1)*8; // 他の色コード計算
}

M5.dis.drawpix(x, y, color); // X・Y座標・色コード指定LED点灯
```

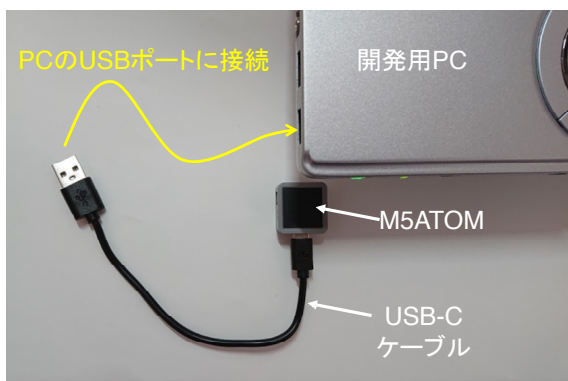

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う

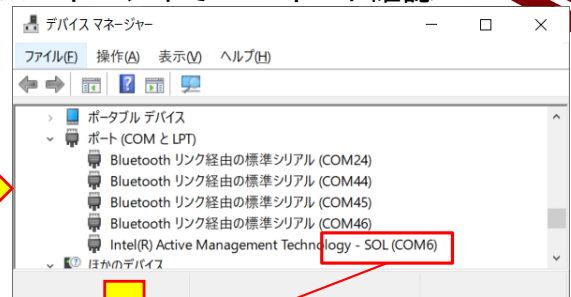


マイコンをPCと接続

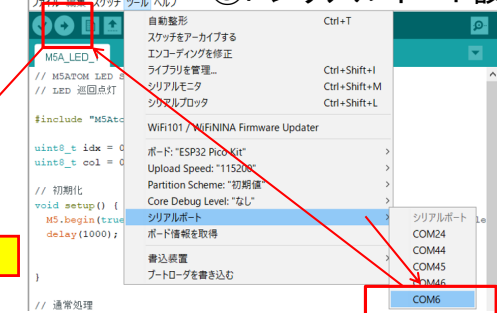
①. M5ATOMをPCと接続



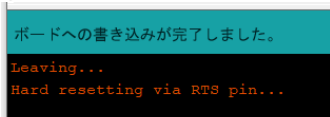
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

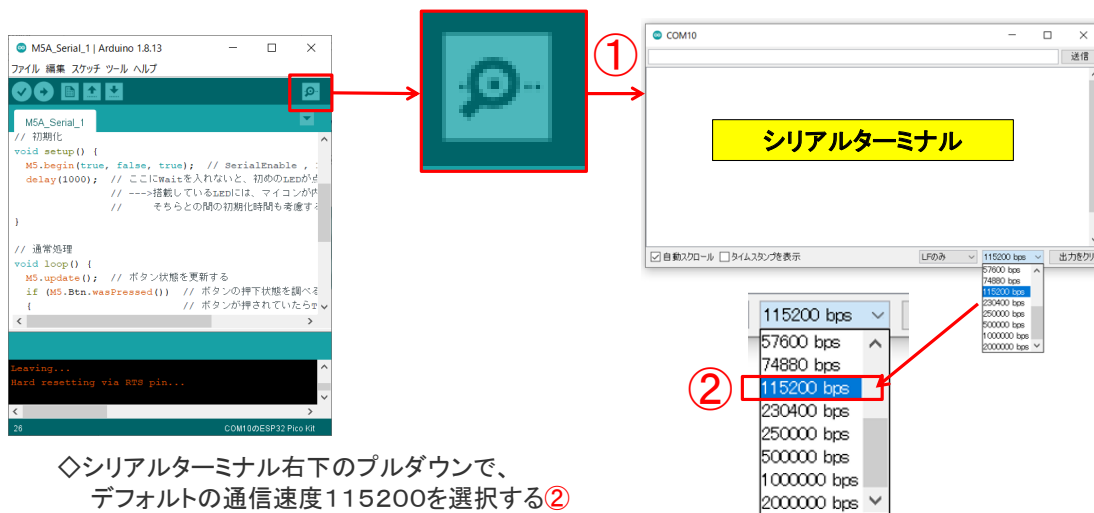


④. コンパイル



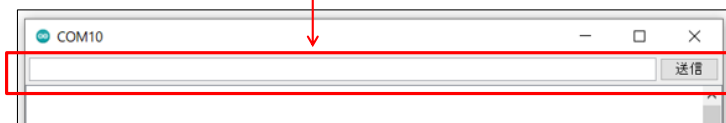
動作確認 1/2 メッセージ受信の確認準備

◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①



動作確認 2/2

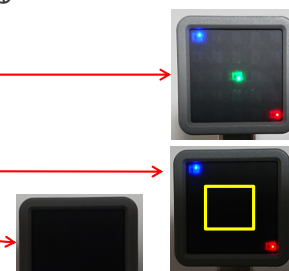
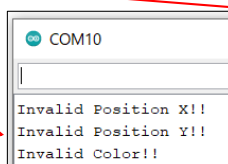
◇シリアルターミナルの送信BOXに以下の電文を記述して ENTER Key を押下する
または 送信ボタンをクリックする



◇電文を X座標 Y座標 色番号 の順にそれぞれ1桁の半角数字で入力する

- ①. 001 ENTER → 左上のLEDが青
- ②. 442 ENTER → 右下のLEDが赤
- ③. 223 ENTER → 中央のLEDが緑
- ④. 224 ENTER → 中央のLEDが消灯
- ⑤. 000 ENTER → 全消灯

⑥. 誤った座標や色番号を指定するとシリアルターミナルにメッセージが表示される



2.5 WEB 経由の通信制御

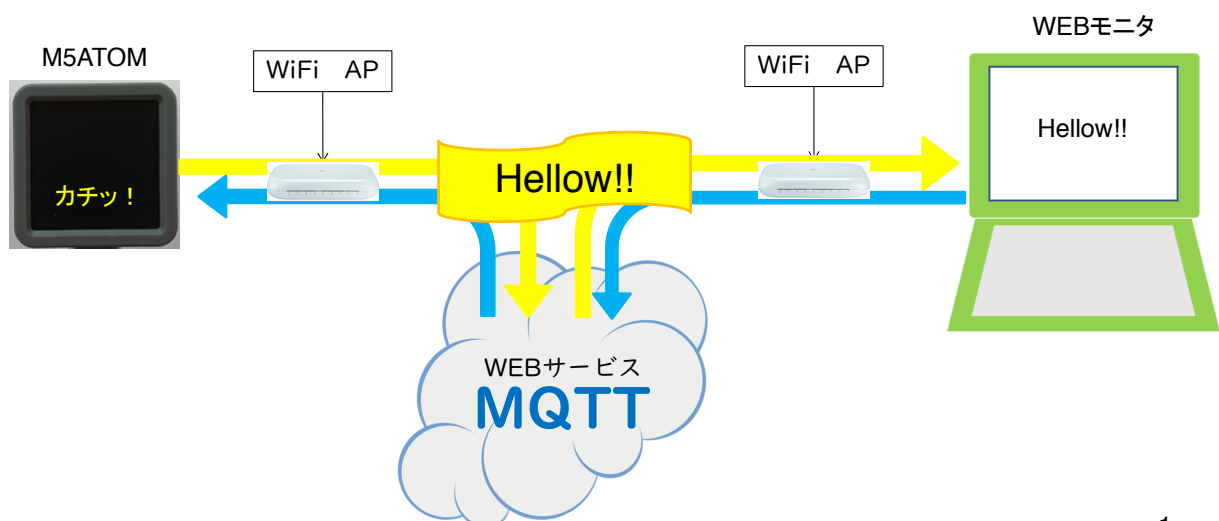
M5Atom

MQTT (WEB経由通信)



一般社団法人全国専門学校情報教育協会

目論見 → WEB経由メッセージ交換



一般社団法人全国専門学校情報教育協会

1

MQTTサービス

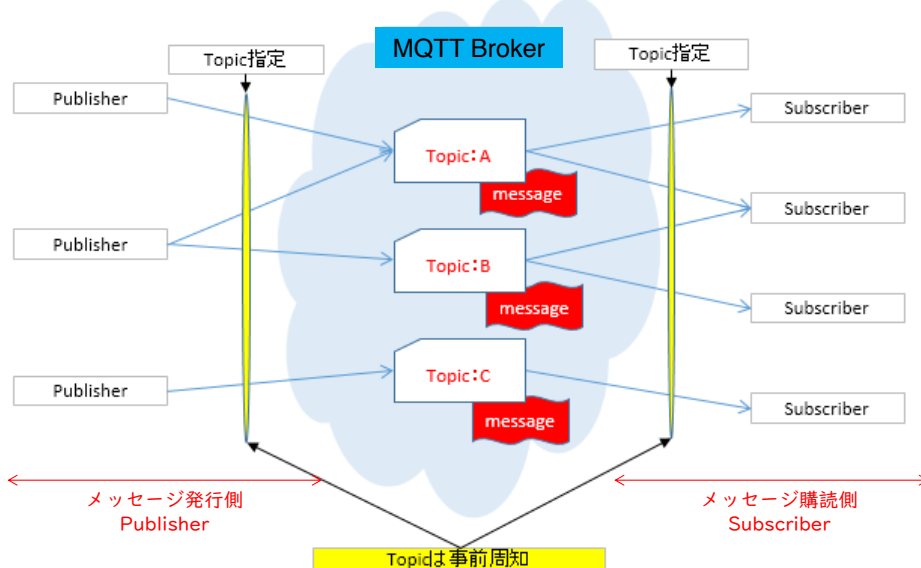
- ◇MQTTは、Message Queue Telemetry Transport の頭文字
- ◇サービスプロバイダ(MQTT Broker)が世界中にある
- ◇短いメッセージ交換に特化したWEBサービス
- ◇登録不要で即利用できる Broker が多い

- ◇WEBを経由するメッセージ交換 → インターネットに接続できれば場所を選ばない
- ◇マイコン⇄マイコン間、PC⇄PC間、マイコン⇄PC間でのメッセージ交換が可能
- ◇言語依存しない(マイコン向けC++・MicroPython・PC用Python・同C++・・・etc)
ライブラリが必要 → 本講座では PubSubClient を利用するのでIDEにインストールする

- ◇ROS(Robot OS)でnode間通信にも採用されている仕組み
- ◇ルールが簡単

MQTTの仕組み

◇トピック名付きメッセージを発行すると、購読登録しているクライアントに通知される



MQTT Broker

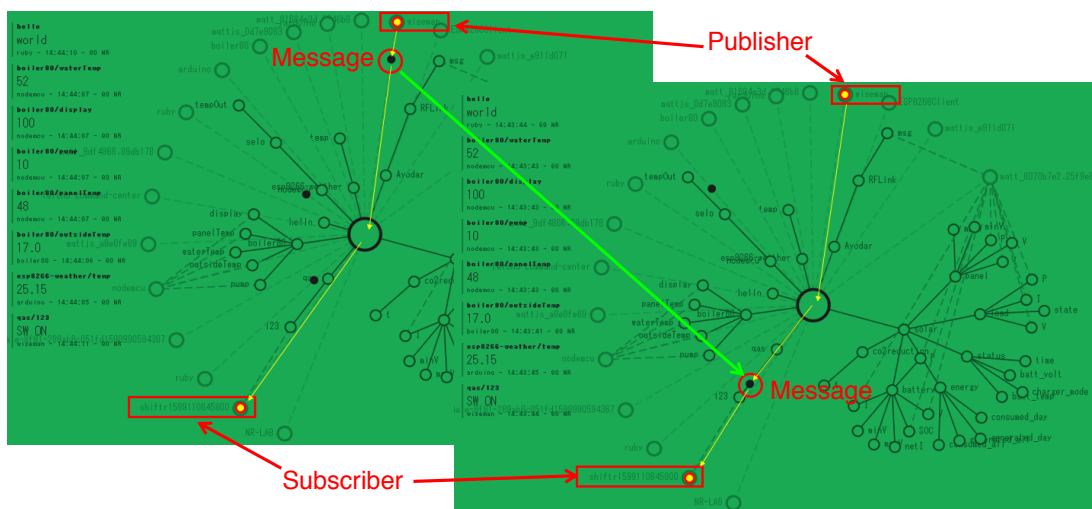
◇サービスを提供している MQTT Broker は世界中に沢山ある

- test.mosquitto.org
- broker.hivemq.com
- broker.shiftr.io ← 今回利用する
- broker.mqttdashboard.com
- iot.eclipse.org

... etc

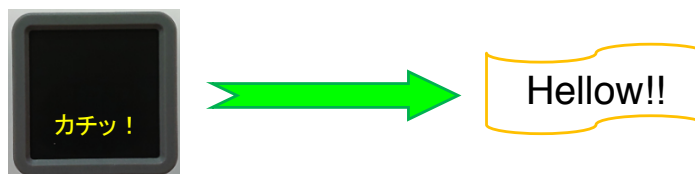
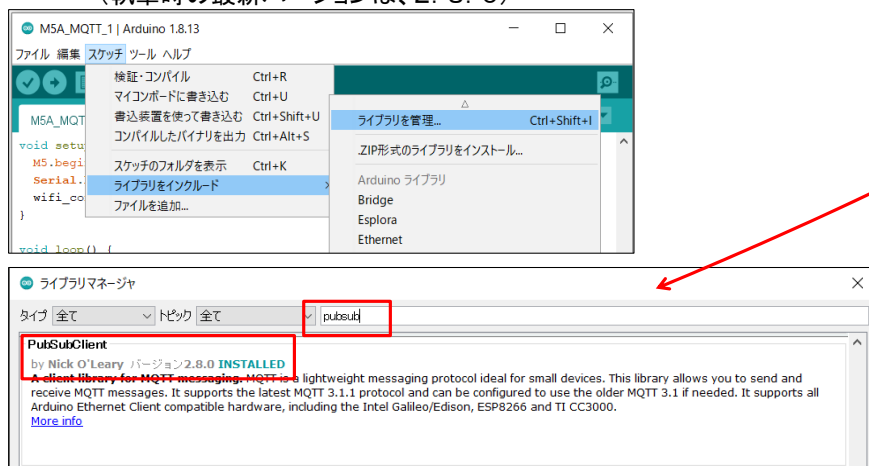
broker.shiftr.io

◇shiftr.io は、オープンMQTTの利用が認められており、図のようにPC上でメッセージの動きが見える
◇赤丸○で示しているのが発行したメッセージの流れである 中央の大きな丸○はBrokerを意味する



ライブラリの準備

- ◇IDEで **スケッチ** → **ライブラリをインクルード** → **ライブラリを管理** → **ライブラリマネージャ**を開く
- ◇検索窓に【**pubsub**】と入力して表示される一覧から **PubSubClient by Nick O'Leary** を選択し、インストールする → **Installed** と表示される
- ※同様のライブラリが数多く存在するので他のものと混同しないように！！
(執筆時の最新バージョンは、2. 8. 0)

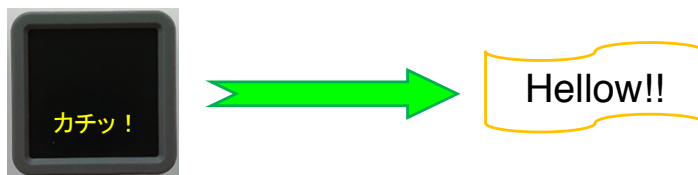


M5ATOMでメッセージを発行してみよう

MQTT PUBLISH MESSAGES

システム構想

- ◇ボタン押下に同期してWEB経由でメッセージを発行する（固定メッセージ）
- ◇M5ATOMは、WiFi接続機能を持っている → WiFi 経由でWEBサービスを利用する
 - ・ 接続先アクセスポイントのSSIDとPASSWORDを調査する
SSID = “ * * * * * ”
PASSWORD = “ * * * * * ”
- ◇MQTT Broker として、broker.shiftr.io の公開MQTTサービスを利用する 接続情報は下記
 - ・ 接続先URL = broker.shiftr.io
 - ・ 接続ポート番号 = 1883（固定）
 - ・ MQTTクライアント名 = ”M5ATOM”
 - ・ MQTTユーザーID = ”try”
 - ・ MQTTパスワード = ”try”
- ◇トピック名 → “gas/123” とする



ソースコード 1/3 (M5A_MQTT_1)

◇#include #define 部分

```
#include <M5Atom.h>
#include <WiFi.h> // WiFi接続ライブラリ
#include <PubSubClient.h> // MQTTクライアントライブラリ

#define WiFi_SSID "SSIDPASS" // WiFiアクセスポイントSSID
#define WiFi_PASS "1234567890" // 同 パスワード
#define MQTT_BROKER "broker.shiftr.io" // MQTT Broker URL
#define MQTT_PORT 1883 // MQTT BROKER PORT (固定)
#define MQTT_CLIENT_NAME "M5Atom" // MQTTブローカ接続時のクライアント名
#define MQTT_USER "try" // 公開MQTTユーザーID(固定)
#define MQTT_PASS "try" // 公開MQTTパスワード(固定)
#define MQTT_TOPIC "gas/123" // TOPIC名 (PublicsherとSubscriber間で決定)

WiFiClient espClient; // WiFiクライアントオブジェクト
PubSubClient client(espClient); // MQTTクライアントオブジェクト
```

ソースコード 2/3 (M5A_MQTT_1)

◇WiFiアクセスポイント接続関数 と 初期化処理部分

```
void wifi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時...表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n---> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}

void setup() { // 初期化処理
  M5.begin(true, false, true); // SerialEnable , I2CEnable , DisplayEnable
  wifi_connect(); // WiFiアクセスポイント接続
}
```

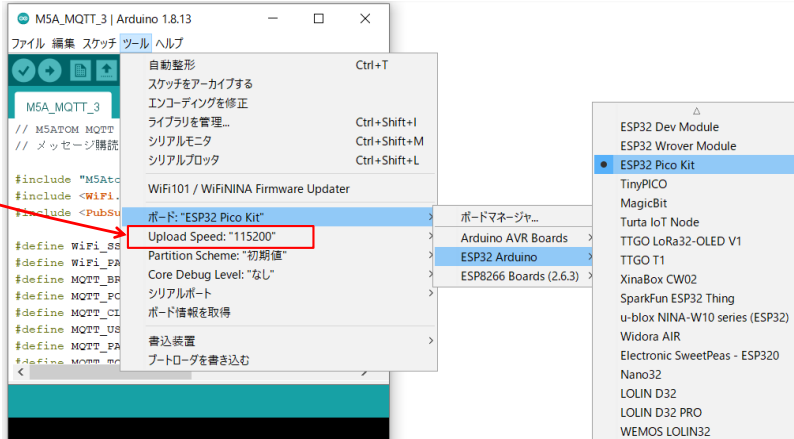
ソースコード 3/3 (M5A_MQTT_1)

```
void loop() { // 通常処理
  client.loop(); // MQTT接続状況更新
  while(!client.connected()){ // MQTT接続
    Serial.println("Mqtt Reconnecting"); // MQTT接続 試みているメッセージ
    if( client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS) ){
      Serial.println("Mqtt Connected"); // MQTT接続成功メッセージ
      break;
    }
    delay(3000); // しばし待ってから再接続
  }

  M5.update(); // M5ATOMボタン状況更新
  if (M5.Btn.wasPressed()){ // ボタンが押されていたか?
    client.publish(MQTT_TOPIC, "Button was pressed !!"); // ここでボタン押下メッセージを送信
    Serial.println("Send message"); // シリアルターミナルにも出力(動作確認に必須)
  }
}
```

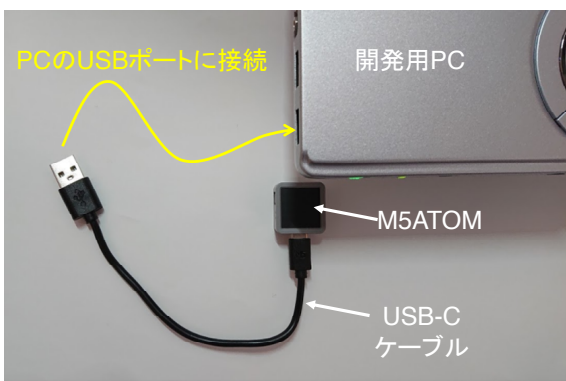

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う

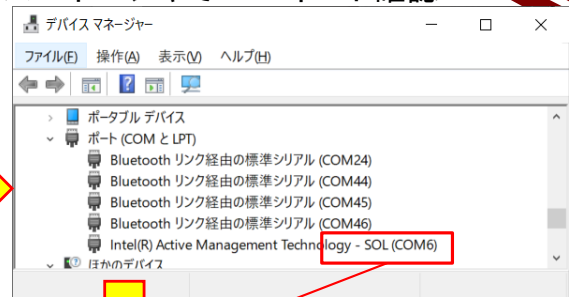


マイコンをPCと接続

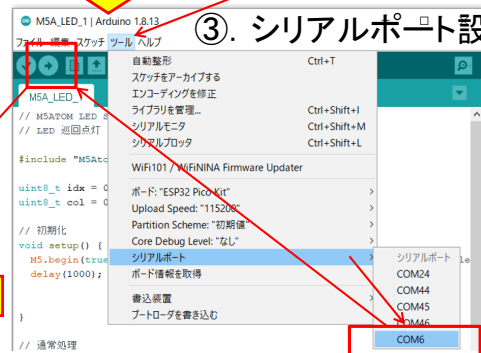
①. M5ATOMをPCと接続



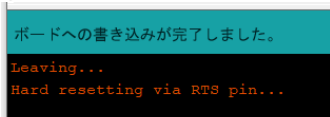
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

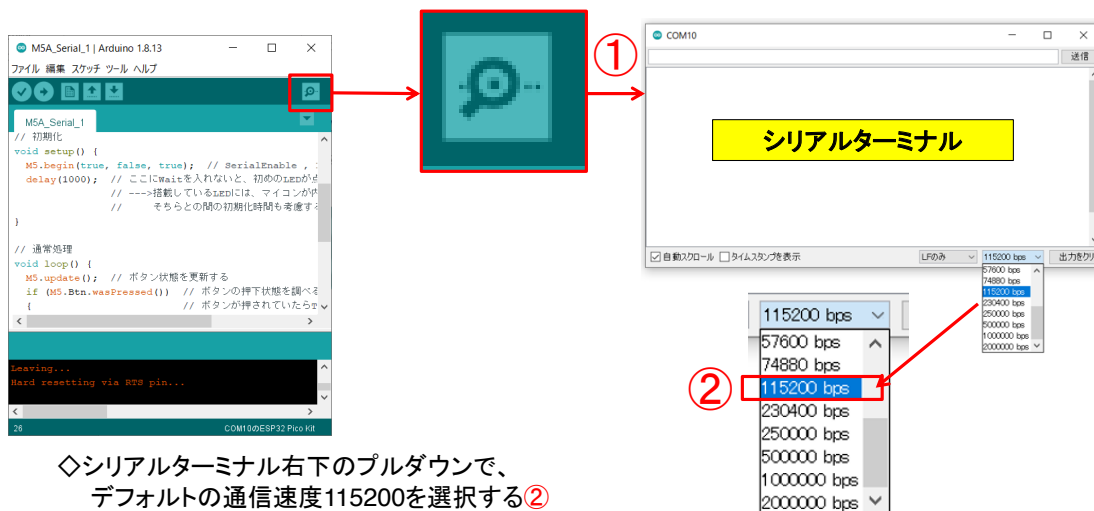


④. コンパイル



動作確認 1/5 シリアルターミナルの準備

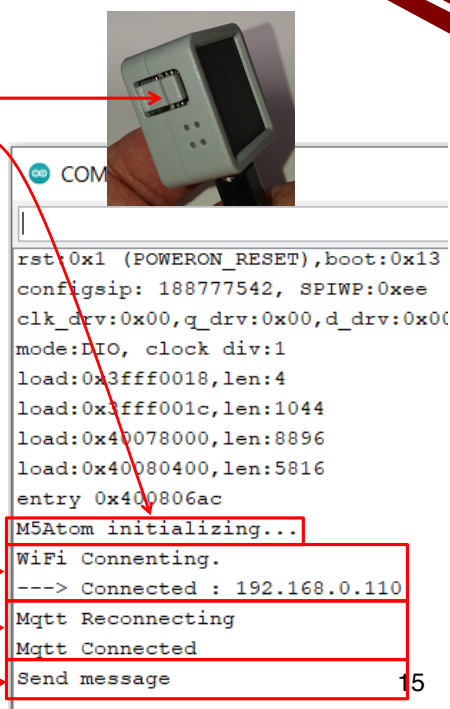
◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①



動作確認 2/5 WiFi と MQTT Broker 接続

- ① M5ATOMの左側にある **RESET SW** を押す
※シリアルターミナルに**リセット時メッセージ**が表示される
- ② 指定 **WiFi アクセスポイントに接続・IPアドレス表示**
- ③ **MQTT Broker に接続確認**
- ④ M5ATOMのボタンを押下
→ **メッセージ発行動作実行確認**

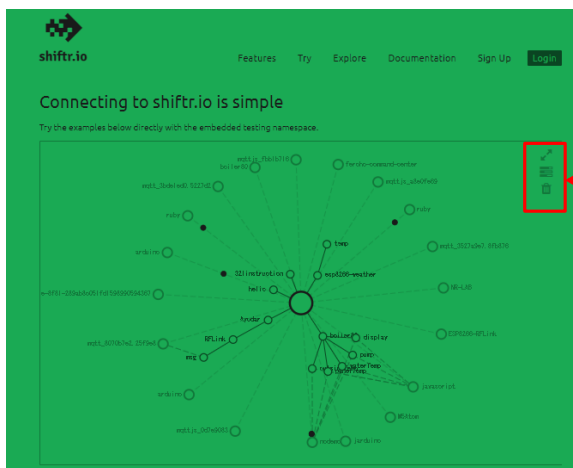
◇MQTTメッセージ発行動作を行なったことが確認できる



動作確認 3/5 WEBで確認準備

① ブラウザで次のURLにアクセスすると図のページが開き、インターネット上のメッセージが見える

<https://legacy.shiftr.io/try>



② 画面右上の3つのアイコンを適宜クリックすれば表示を見やすく調整できる

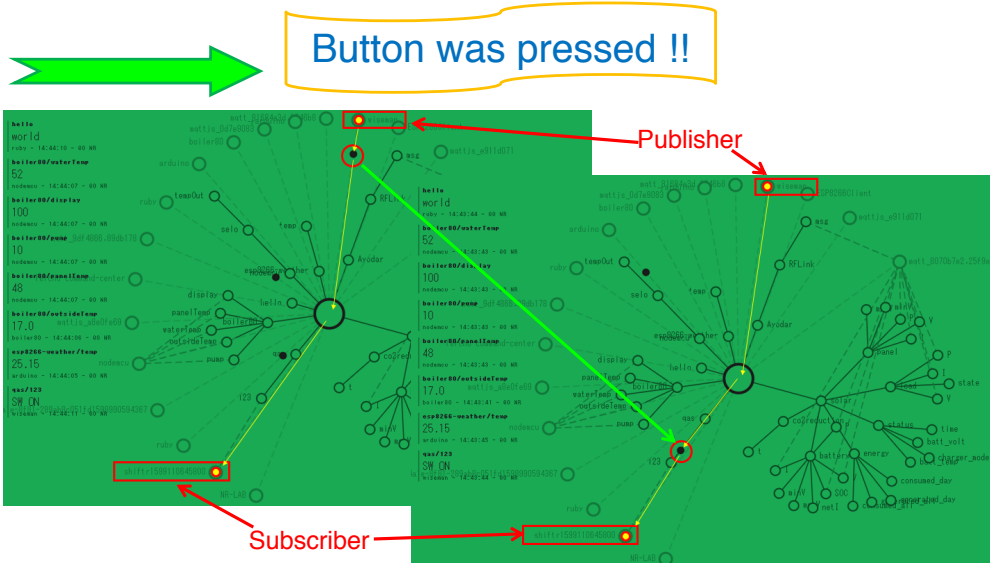


※ 動き回る黒丸● → メッセージの移動
 黒丸が出現する白丸 → メッセージ発行者
 中央の大きな○ → MQTT Broker
 メッセージが流れ着く先の白丸 → トピック名

一般社団法人全国専門学校情報教育協会

動作確認 4/5 WEBでメッセージの確認

③ ブラウザを注目しながらM5ATOMのボタンを押すとメッセージが流れる様子が見える



一般社団法人全国専門学校情報教育協会

17

システム構想

- ◇PCからメッセージを発行することができる
 - コマンドプロンプトから以下のコマンドを発行すれば、“Hello !!” が発行される
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "Hello !!"
※qas/123 と Hello !! はそれぞれ、トピック名とメッセージである
 - ◇シリアル通信でLEDを制御したことを思い出して、その際作成したプログラムを利用する
 - ◇メッセージを3文字として、その内容を以下のようにする
 - 0文字目:LED X座標 0~4
 - 1文字目:LED Y座標 0~4
 - 2文字目:色 1:青 2:赤 3:緑 4:消灯 0:全LED消灯※これは、シリアル通信の受信で実験したメッセージそのものだ！
 - ◇例えば、X=1, Y=3 のLEDを赤にしたかったら以下のコマンドを実行すればよい
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "132"
- ※ここでは curl コマンドの詳細は説明しない
このような便利なコマンドは、自身で調べて記録しておくべきだ(これはLinuxにもある)

ソースコード 1/7 (M5A_MQTT_2)

◇#include と #define 、通信関連オブジェクト部分

```
#include <M5Atom.h> // マイコンボードライブラリ
#include <WiFi.h> // WiFiライブラリ
#include <PubSubClient.h> // MQTTクライアントライブラリ

#define WiFi_SSID "Planex_24-E68A9A" // WiFiアクセスポイントSSID
#define WiFi_PASS "7D438B6945" // 同 パスワード
#define MQTT_BROKER "broker.shiftr.io" // MQTTブローカーURL
#define MQTT_PORT 1883 // MQTT BROKER PORT
#define MQTT_CLIENT_NAME "M5Atom" // MQTTブローカー接続時のクライアント名
#define MQTT_USER "try" // 公開ユーザー名(固定)
#define MQTT_PASS "try" // 公開パスワード(固定)
#define MQTT_TOPIC "qas/123" // TOPIC名
#define MQTT_QOS 0 // Quality of Service(サービスの品質)
WiFiClient espClient; // WiFiコントロールオブジェクト
PubSubClient client(espClient); // MQTTクライアントコントロールオブジェクト
```

ソースコード 2/7 (M5A_MQTT_2)

◇グローバル変数 と WiFiアクセスポイント接続関数

```
char flg=0; // メッセージ受信フラグ 0:未受信 1:メッセージ到着
char msg[10]; // 受信メッセージ格納用(少し大きめに確保した)

void wifi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時...表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n---> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}
```

ソースコード 3/7 (M5A_MQTT_2)

◇MQTTブローカー接続関数

```
void mqtt_connect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS)) { // MQTT接続実行！
      Serial.println("connected"); // つながったメッセージ
      client.subscribe(MQTT_TOPIC, MQTT_QOS); // メッセージ購読登録
      Serial.println("Subscribing!"); // 購読しているよメッセージ
    } else { // うまく行かなかった場合
      Serial.print("failed, rc="); // 失敗原因コードの表示
      Serial.print(client.state()); // 同上
      Serial.println(" try again in 5 seconds"); // 5秒待ちますメッセージ
      // Wait 5 seconds before retrying
      delay(5000); // しばし待つ
    }
  }
}
```

ソースコード 4/7 (M5A_MQTT_2)

◇メッセージが発行された際に呼び出される(コールバック)関数

```
void callback(char* topic, byte* payload, unsigned int length) {
  int i;

  Serial.print("Message arrived ["); // メッセージが到着したよメッセージ
  Serial.print(topic);              // 念のためトピック名表示
  Serial.print("] ");              // 括弧閉じ
  for (i = 0; i < length; i++) {   // (メッセージ文字長文)メッセージ取り出し
    msg[i] = (char)payload[i];
    Serial.print((char)payload[i]); // メッセージ表示
  }
  Serial.println();                // 改行を付けて！！
  flg = 1;                          // メッセージ到着フラグをONする loop()内でこのフラグを見て処理する
}
```

ソースコード 5/7 (M5A_MQTT_2)

◇LEDを全消灯する際に利用する関数 と 初期化

```
void fill_led(unsigned long col){ // LEDをすべて同じ色にする
  for(int i=0; i<25; i++){
    M5.dis.drawpix(i, col); //
  }
}

void setup() { // 初期化
  M5.begin(true, false, true); // SerialEnable , I2CEnable , DisplayEnable
  wifi_connect(); // WiFiアクセスポイント接続
  client.setCallback(callback); // メッセージ発行時に呼び出される関数を登録する
}
```

ソースコード 6/7 (M5A_MQTT_2)

◇通常処理全体 エラー処理とLED制御は次で説明)

```
void loop() { // 通常処理
  int i;
  int x; // LED X座標
  int y; // LED Y座標
  int col; // LED 色指定
  unsigned long color; // 色コード

  mqtt_connect(); // MQTT 接続が切れているといけないので、調べて必要なら再接続
  client.loop(); // MQTT接続状況更新 ... これがなかなか難しい

  if(flag==1){ // 受信メッセージあり?
    flag = 0; // フラグクリア
    x = msg[0]-'0'; // X座標計算
    y = msg[1]-'0'; // Y座標計算
    col = msg[2]-'0'; // 色番号計算

    // エラーチェックブロック
    // LED制御ブロック
  }
}
```

26

一般社団法人全国専門学校情報教育協会

ソースコード 7/7 (M5A_MQTT_2)

◇エラーチェックブロック と LED制御ブロック

//エラーチェックブロック

```
if(x<0 || x>4){ // X座標チェック
  Serial.println("Invalid Position X!!");
  return; // エラーメッセージを送信してOSに戻る
}
if(y<0 || y>4){ // Y座標チェック
  Serial.println("Invalid Position Y!!");
  return; // エラーメッセージを送信してOSに戻る
}
if(col<0 || col>4){ // 色番号チェック
  Serial.println("Invalid Color!!");
  return; // エラーメッセージを送信してOSに戻る
}
```

// LED制御ブロック

```
if(col == 0){ // 色番号指定が0なら全LED消灯
  fill_led(color = 0x000000); // 既に宣言した関数をCall
  return; // LEDをすべて消灯して正誤をOSに戻す
} else if(col == 4){ // 色番号4なら指定のLEDだけ消灯
  color = 0x000000; // 色コード 黒=消灯
} else {
  color = 0x0000ff << (col-1)*8; // 色コード計算 ※
}
M5.dis.drawpix(x, y, color); // 座標・色コード指定LED点灯
}
```

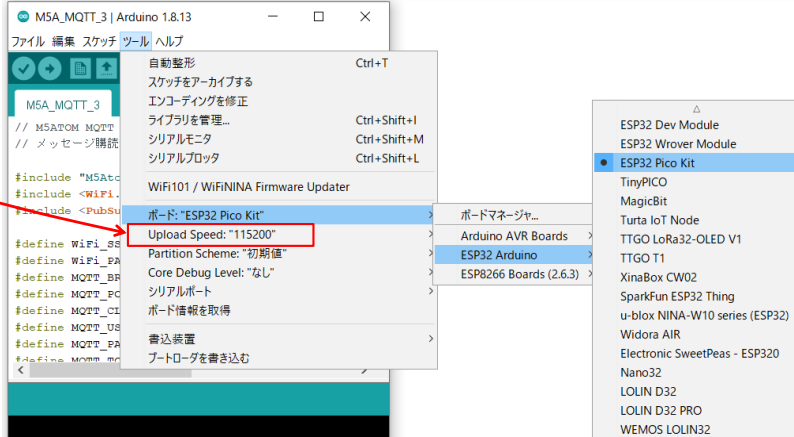
※色番号 1なら0x0000ff
2なら0x00ff00
3なら0xff0000
のように、ffの位置が8ビットずつずらされる

27

一般社団法人全国専門学校情報教育協会

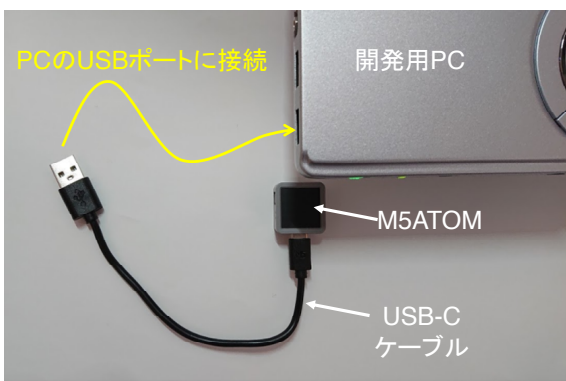
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う

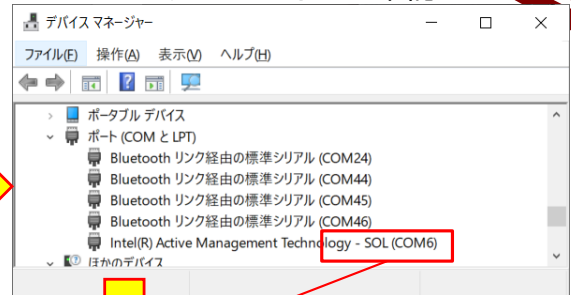


マイコンをPCと接続

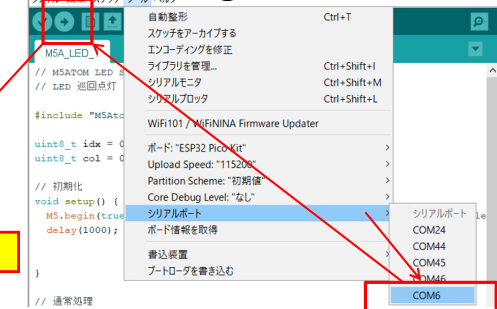
①. M5ATOMをPCと接続



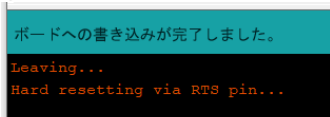
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

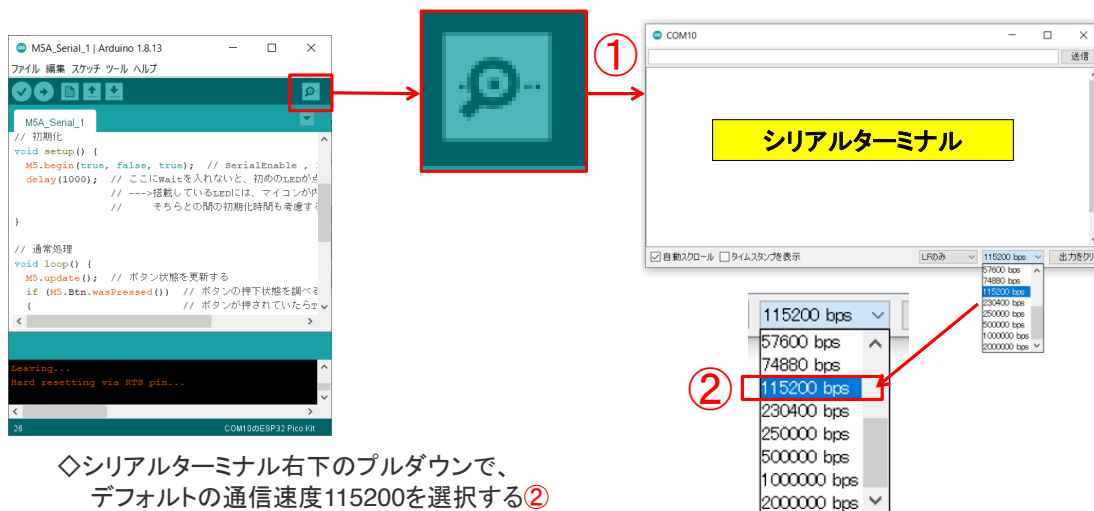


④. コンパイル



動作確認 1/3 シリアルターミナルの準備

◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①

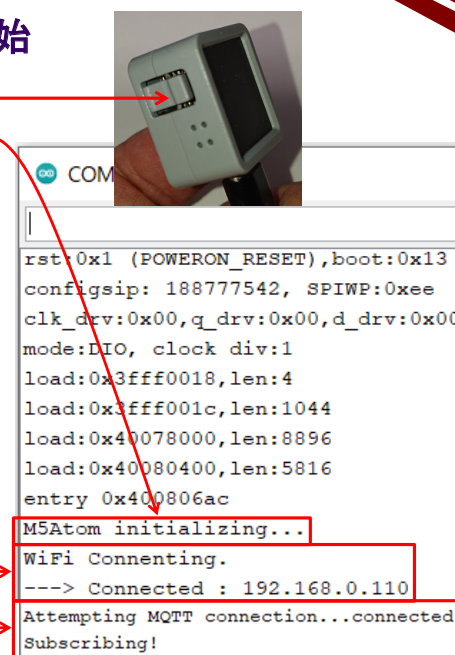


◇シリアルターミナル右下のプルダウンで、デフォルトの通信速度115200を選択する②

動作確認 2/3 MQTT Broker 接続・購読開始

- ① M5ATOMの左側にある **RESET SW** を押す
※シリアルターミナルに**リセット時メッセージ**が表示される
- ② 指定 **WiFi アクセスポイントに接続・IPアドレス表示**
- ③ **MQTT Broker に接続・購読開始**

◇MQTTメッセージ購読が開始されたことが確認できる



動作確認 3/3 メッセージ購読確認

◇PCからメッセージを発行する→ コマンドプロンプトからコマンドを発行する

- ① X=0, Y=0 のLEDを青にする → 以下のコマンドを発行する
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "001"
- ② X=2, Y=2 のLEDを赤にする → 以下のコマンドを発行する
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "222"
- ③ X=4, Y=4 のLEDを緑にする → 以下のコマンドを発行する
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "443"
- ⑤ X=2, Y=2 のLEDを消灯する → 以下のコマンドを発行する
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "224"
- ⑥ 全消灯メッセージ "000" とするとLEDはすべて消灯する

◇シリアルターミナルにも購読したメッセージが表示される

```
Message arrived [qas/123] 443  
Message arrived [qas/123] 224  
Message arrived [qas/123] 000
```

※この時、shiftr.io のページでもメッセージが移動する様子が観察できる！



2.6 サーボモータの回転角度制御

M5Atom

Servo Motor



サーボモータ

- ◇サーボモータは、主軸の回転角度が制御できる
- ◇右図のサーボモータは主軸が最大180度回転する
- ◇主軸にホーン(白いパーツ)を取り付けて使う
- ◇位置決めに使われる
- ◇データシートの一部を下記に示す

Specifications:

Weight: 9g

Dimension: 23×12.2×29mm

Stall torque: 1.8kg/cm(4.8v)

Gear type: POM gear set

Operating speed: 0.12 sec/60degree(4.8v)

Operating voltage: 4.8v

Temperature range: 0℃_ 55℃

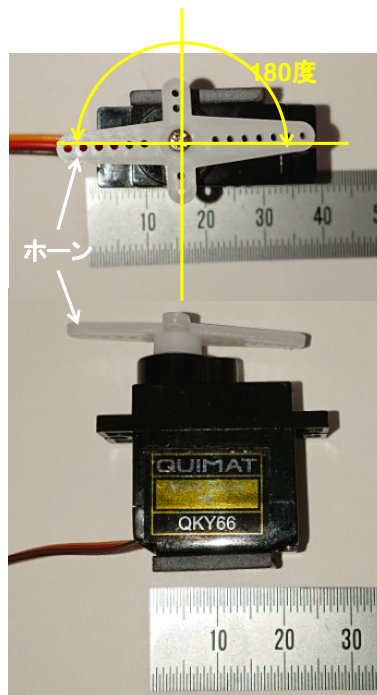
Dead band width: 1us

Power Supply: Through External Adapter

servo wire length: 25 cm

Servo Plug: JR (Fits JR and Futaba)

※主軸中心から1cmの所に糸を取り付けて、1.8kg未満の加重を引き上げる力がある



一般社団法人全国専門学校情報教育協会

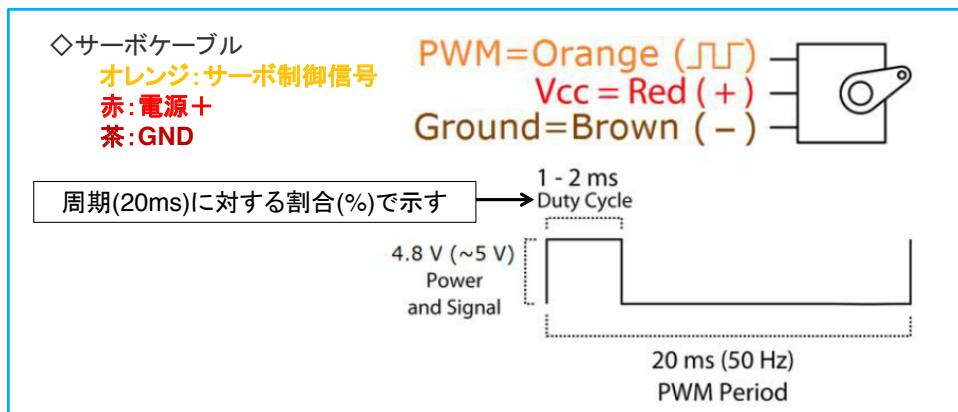
1

サーボモータ制御信号

◇PWM周期20ms(50Hz) → Pulse Width Modulation

◇Datasheetは【"0"(1msパルス)で中央、"90"(2msパルス)で中央、"-90"(~1msパルス)で一番左】としているが、これは誤りで、**実際は上から見て【0度で一番右に、180度で一番左に位置決めされる】**

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

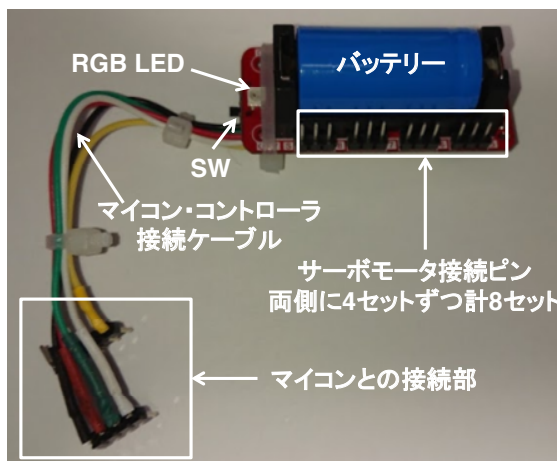


一般社団法人全国専門学校情報教育協会

2

サーボモータ電源とコントローラ

- ◇サーボモータ電源はマイコンとは別の電源を確保する → マイコンの安定動作のため
- ◇電源を搭載したコントローラの利用 → サーボモータ8個別々に制御可能 ※フルカラーLED付属
- ◇マイコンとの接続は、別途作成したケーブル利用



- ◇サーボコントローラに角度とサーボ CH をセットすれば、搭載している IC が PWM 制御パルスを出力する → マイコン側は、CHと角度を指定するだけ
- ◇RGB LEDは、RGB各々の明るさを8bitで指定する
- ◇サーボCHおよびRGB各々のレジスタはI2Cアドレスが決まっている
- ◇サーボモータ電源は、バッテリーから供給される
- ◇バッテリー充電は、マイコンとの接続ケーブルを外して M5Stick-Cを接続してUSBケーブルで充電するかまたは、専用充電器を使用する

サーボモータを制御する

SERVO MOTOR CONTROL

目論見 → モデル工場の設備振動シミュレーション

◇M5ATOMからサーボコントローラを介してPWMパルスを出力し、サーボモータを左右に回転運動させる

→ 加速度センサを載せて振動を測定する

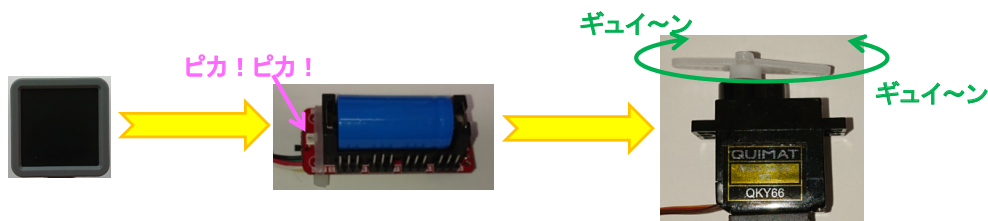
→ **モデル工場の設備振動シミュレーション**



システム構想

◇M5ATOMによるサーボモータ制御 → 0度から180度の往復回転運動を繰り返す

- ①. IIC_servo ライブラリ (servo.cpp + servo.h) が提供されている
※ソースコードと同一フォルダで同時にコンパイル → ライブラリインストールの必要なし
※別マイコン(M5Stick-C)用のライブラリをM5ATOM用に変更したもの
- ②. 次の関数が見える
 - ・ IIC_Servo_Init() → コントローラの初期化
 - ・ Servo_angle_set(HC, 角度) → サーボモータ角度制御 CHIは1~8、角度は0~180で指定
 - ・ RGB_set(R,G,B) → LED点灯...R,G,Bの明るさは、それぞれ0~255で指定



ソースコード 1/2 (M5A_Servo_1)

◇初期化処理部

```
#include "M5Atom.h" // “ ”は<>でも良い
#include "IIC_servo.h" // “ ”でなければならぬ→カレントディレクトリにヘッダファイルがあるから

// 内蔵加速度センサが利用するGPIOポートを開放する → コントローラICとの通信に使用する
bool IMU6886Flag = false; //sda:25 and scl:21 are free.

void setup(){ // 初期化
  M5.begin(true, true, false); //serial, I2C, LED
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init(); //sda:21 scl:25 Servo Controller用にIICを設定(初めに開放したポート)
                    // IICというのは、マイコンとデバイスを通信で結ぶI/F
}
```

ソースコード 2/2 (M5A_Servo_1)

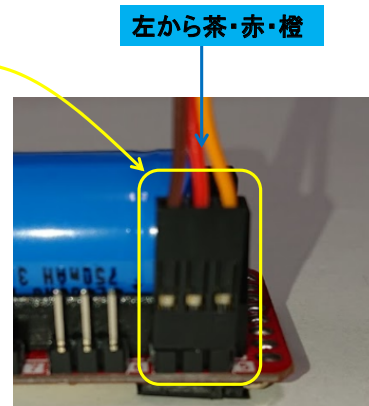
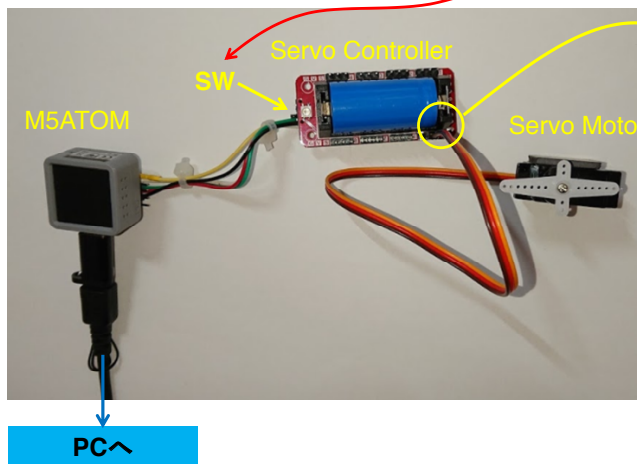
◇通常処理部

```
void loop() { // 通常処理
  RGB_set(128,0,0); // red ピカ！
  delay(200); // しばし待つ
  RGB_set(0,128,0); // green ピカ！
  delay(200); // しばし待つ
  RGB_set(0,0,128); // blue ピカ！
  delay(200); // しばし待つ
  RGB_set(0,0,0); //RGB 消灯

  for(int i=1;i<9;i++){ // 全CH(どのCHに接続してもサーボは動く)を制御
    Servo_angle_set(i,0); // 0度
  }
  delay(1000); // しばし待つ
  for(int i=1;i<9;i++){ // 全CH(どのCHに接続してもサーボは動く)を制御
    Servo_angle_set(i,180); // 180度
  }
  delay(1000); // しばし待つ
}
```

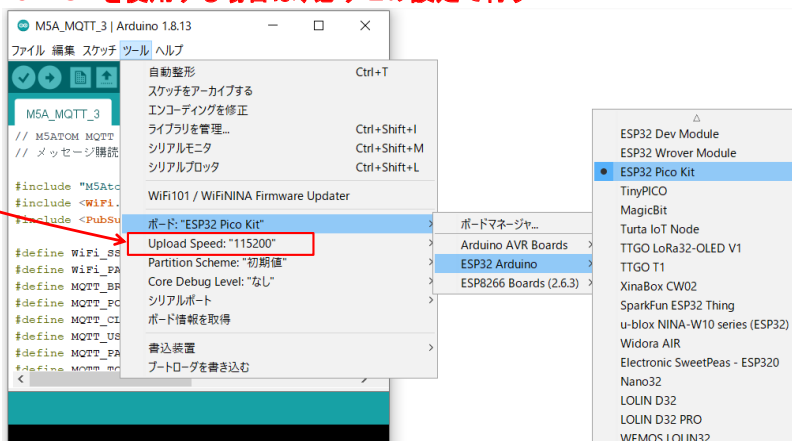
マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、外部基板などを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い



マイコンボードの選択

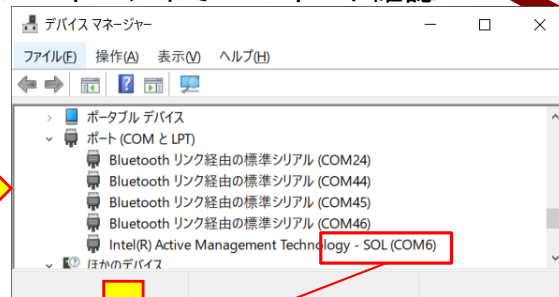
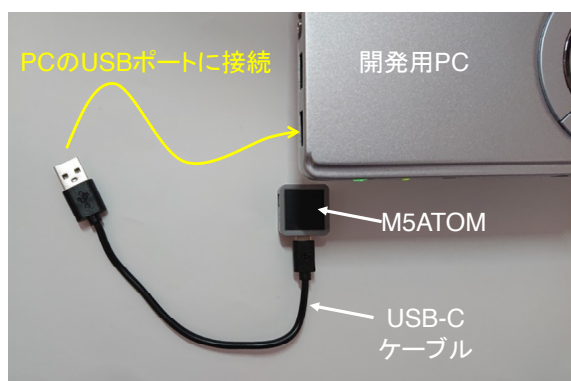
- ◇以下のように IDE で **ツール → ボード → ...とたどり、ESP32 Pico Kit** を選択する
- ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に **シリアルポートの Upload Speed は、115200 にセット**する(これより早いと書込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う



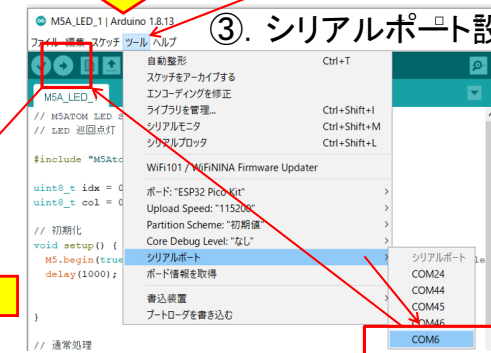
②. デバイスマネージャでCOMポート確認

マイコンをPCと接続

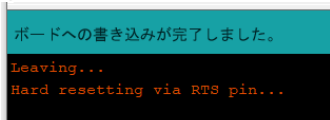
①. M5ATOMをPCと接続



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル

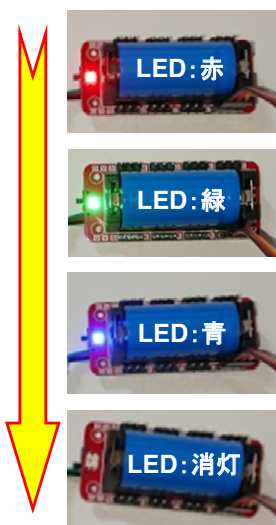


一般社団法人全国専門学校情報教育協会

11

動作確認

◇書き込み完了と同時に、新しいプログラムがスタートしている



◇サーボコントローラのLEDが、赤→緑→青と点灯した後、サーボモータが180度の往復回転運動を行う動作が繰り返される



一般社団法人全国専門学校情報教育協会

12



MQTTでサーボモータを制御する

SERVO MOTOR CONTROL BY MQTT MESSAGE

一般社団法人全国専門学校情報教育協会

13

システム構想

- ◇MQTTメッセージでLEDを制御したことを思い出そう
- ◇PCからメッセージを発行することができる
 - コマンドプロンプトから以下のコマンドを発行すれば、“Hello !!” が発行される
 - `curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "Hello !!"`
 - ※`qas/123`と`Hello !!`はそれぞれ、トピック名とメッセージである
- ◇MQTTでLEDを制御したことを思い出して、その際作成したプログラムを利用する
- ◇メッセージを3文字として、その内容でサーボモータの回転角度を指定する
 - 例 : 0度 → 000
 - 90度 → 090
 - 180度 → 180
- ◇例えば、サーボモータを45度の位置に回転する場合は以下のコマンドを実行すればよい
- `curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "045"`



一般社団法人全国専門学校情報教育協会

14

ソースコード 1/6 (M5A_Servo_2)

◇#include と #define 、通信関連オブジェクト部分

```
#include "M5Atom.h" // マイコンボードライブラリ “ ”は<>でもよい
#include <WiFi.h> // WiFiライブラリ
#include <PubSubClient.h> // MQTTクライアントライブラリ
#include "IIC_servo.h" // サーボ制御ライブラリ
#define WiFi_SSID "Planex_24-E68A9A" // WiFiアクセスポイントSSID
#define WiFi_PASS "7D438B6945" // 同 パスワード
#define MQTT_BROKER "broker.shiftr.io" // MQTTブローカーURL
#define MQTT_PORT 1883 // MQTT BROKER PORT
#define MQTT_CLIENT_NAME "M5Atom" // MQTTブローカ接続時のクライアント名
#define MQTT_USER "try" // 公開ユーザー名(固定)
#define MQTT_PASS "try" // 公開パスワード(固定)
#define MQTT_TOPIC "qas/123" // TOPIC名
#define MQTT_QOS 0 // Quality of Service(サービスの品質)
WiFiClient espClient; // WiFiコントロールオブジェクト
PubSubClient client(espClient); // MQTTクライアントコントロールオブジェクト
// 内蔵加速度センサが利用するGPIOポートを開放する
bool IMU6886Flag = false; //sda:25 and scl:21 are free.
```

15

ソースコード 2/6 (M5A_Servo_2)

◇グローバル変数 と WiFiアクセスポイント接続関数

```
char flg=0; // メッセージ受信フラグ 0:未受信 1:メッセージ到着
char msg[10]; // 受信メッセージ格納用(少し大きめに確保した)

void wifi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時...表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n---> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}
```

16

ソースコード 3/6 (M5A_Servo_2)

◇MQTTブローカー接続関数

```
void mqtt_connect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS)) { // MQTT接続実行！
      Serial.println("connected"); // つながったメッセージ
      client.subscribe(MQTT_TOPIC, MQTT_QOS); // メッセージ購読登録
      Serial.println("Subscribing!"); // 購読しているよメッセージ
    } else { // うまく行かなかった場合
      Serial.print("failed, rc="); // 失敗原因コードの表示
      Serial.print(client.state()); // 同上
      Serial.println(" try again in 5 seconds"); // 5秒待ちますメッセージ
      // Wait 5 seconds before retrying
      delay(5000); // しばし待つ
    }
  }
}
```

ソースコード 4/6 (M5A_Servo_2)

◇メッセージが発行された際に呼び出される(コールバック)関数

```
void callback(char* topic, byte* payload, unsigned int length) {
  int i;

  Serial.print("Message arrived ["); // メッセージが到着したよメッセージ
  Serial.print(topic); // 念のためトピック名表示
  Serial.print("] "); // 括弧閉じ
  for (i = 0; i < length; i++) { // (メッセージ文字長文)メッセージ取り出し
    msg[i] = (char)payload[i];
    Serial.print((char)payload[i]); // メッセージ表示
  }
  Serial.println(); // 改行を付けて！！
  flg = 1; // メッセージ到着フラグをONする loop()内でこのフラグを見て処理する
}
```

ソースコード 5/6 (M5A_Servo_2)

◇初期化関数

```
void setup() { // 初期化
  M5.begin(true, true, true); // SerialEnable , I2CEnable , DisplayEnable
  wifi_connect(); // WiFiアクセスポイント接続
  client.setCallback(callback); // メッセージ発行時に呼び出される関数を登録する
  mqtt_connect(); // MQTT 再接続・・・(接続が切れていたら)

  Serial.printf("%n MQTT_Servo Control-2%n"); // シリアルポートにメッセージ出力
  IIC_Servo_Init(); //sda:21 scl:25 Servo Controller用にIICを設定
}
```

ソースコード 6/6 (M5A_Servo_2)

◇通常処理全体 エラー処理とLED制御は次で説明)

```
void loop() { // 通常処理
  int i;
  int a; // サーボモータ角度

  mqtt_connect(); // MQTT 接続が切れているといけないので、調べて必要なら再接続
  client.loop(); // MQTT接続状況更新 ... これがなかなか難しい

  if(flag==1){ // 受信メッセージあり?
    flag = 0; // フラグクリア

    a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // サーボモータ回転角度計算

    // エラーチェックブロック
    // LED点滅ブロック
    // サーボモータコントローラ出力 ブロック
  }
}
```

ソースコード 6/6 (M5A_Servo_2)

◇エラーチェックブロック と LED制御ブロック

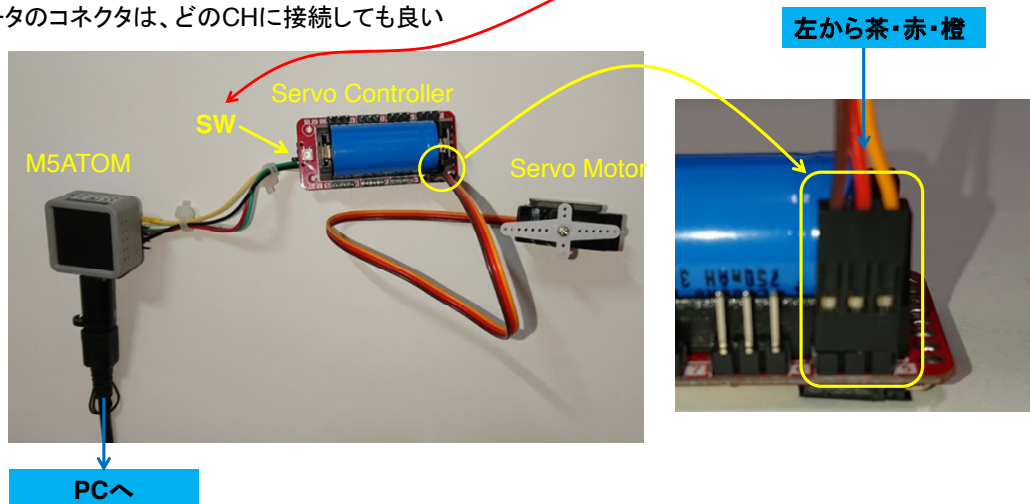
```
//エラーチェックブロック
if(a<0 || a>180){ // 角度チェック
  Serial.println("Over Rotation Angle!!");
  return; // エラーメッセージを送信してOSに戻る
}
```

```
// LED点滅ブロック
RGB_set(128,0,0); // red ピカ！
delay(200);
RGB_set(0,128,0); // green ピカ！
delay(200);
RGB_set(0,0,128); // blue ピカ！
delay(200);
RGB_set(0,0,0); //RGB 消灯
```

```
// サーボモータ制御ブロック
// 全CHIに角度設定(どのピンに接続してもサーボが動く)
for(int i=1;i<9;i++){
  Servo_angle_set(i,a); // a=指定角度
}
```

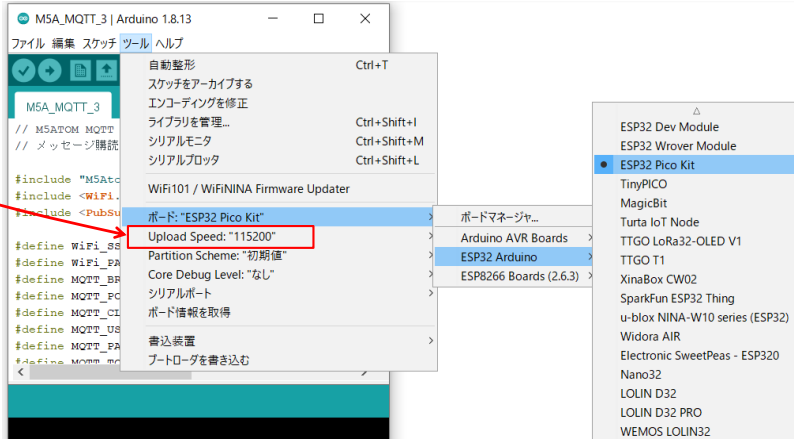
マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、外部基板などを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い



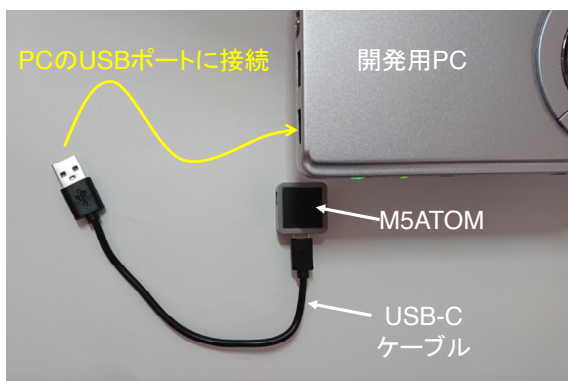
マイコンボードの選択

- ◇ 以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
 - ※ ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ← M5Atom ライブラリが無い場合
- ◇ 同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇ 以後、M5ATOMを使用する場合は、必ずこの設定で行う

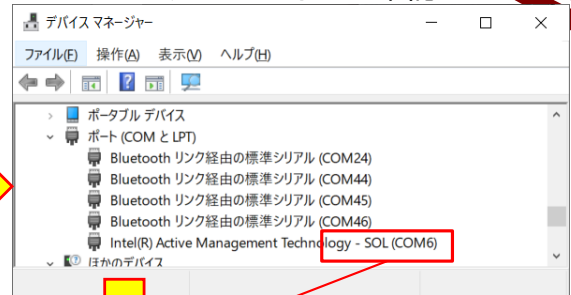


マイコンをPCと接続

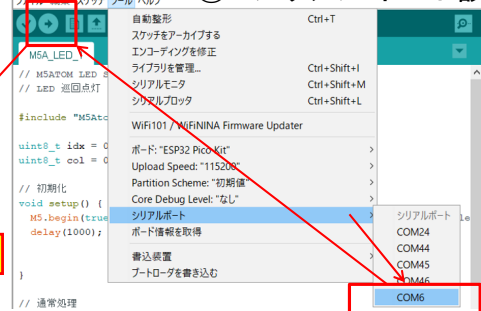
①. M5ATOMをPCと接続



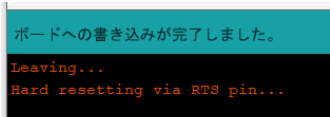
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



動作確認

◇下図のように、コマンドプロンプトで curl コマンドを用いてメッセージを発行する

```

C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "000"
OK
C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "090"
OK
C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "180"
OK
C:\Users\user>
  
```

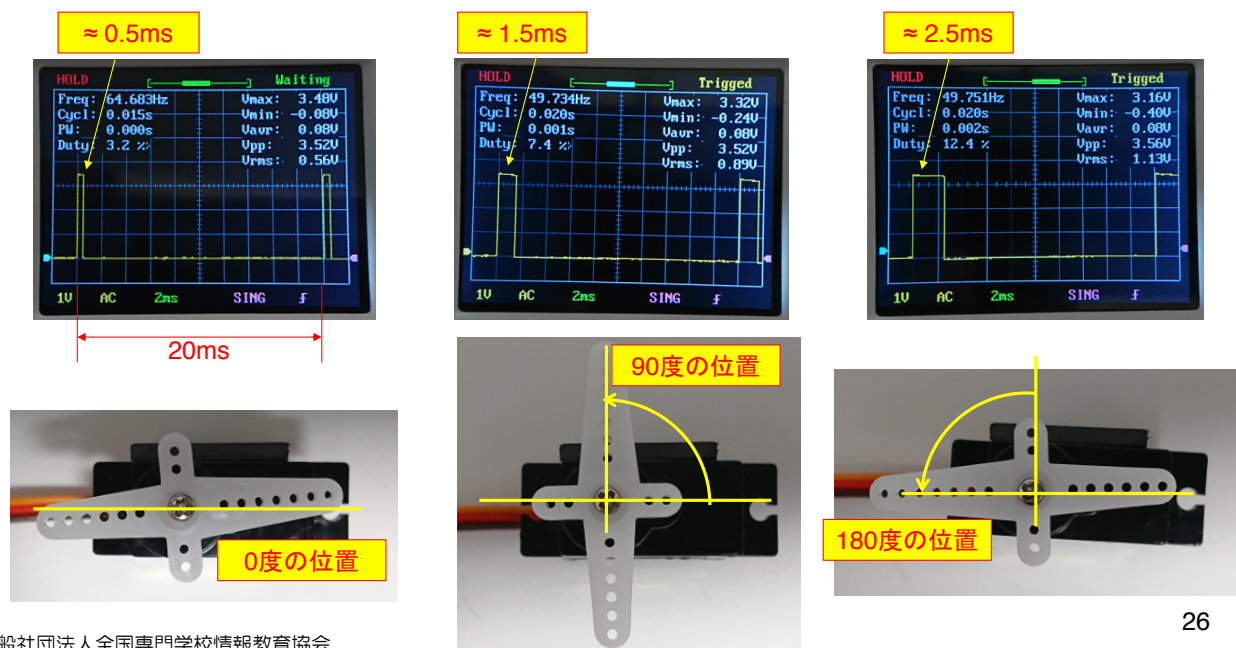
◇シリアルモニタを起動しておく、M5ATOMが受信したメッセージが表示され、その後LEDが赤→緑→青と点灯し、サーボモータが180度の往復回転運動をする

```

Message arrived [qas/123] 000
Message arrived [qas/123] 090
Message arrived [qas/123] 180
  
```

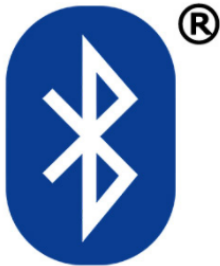


パルスと回転の様子（簡易オシロスコープ）



2.7 Bluetooth の活用

M5Atom



Bluetooth



一般社団法人全国専門学校情報教育協会

Bluetooth Class

Table 1: ESP32-PICO-D4 Specifications

Categories	Items	Specifications
Certification	Bluetooth certification	BQB
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps) A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth V4.2 BR/EDR and BLE specification NZIE receiver with -97 dBm sensitivity
	Radio	Class-1, class-2 and class-3 transmitter ← Class1~3
	Audio	AFH CVSD and SBC

	最大出力	通信距離
Class 1	100mW	約100m
Class 2	2.5mW	約10m
Class 3	1mW	約1m

注) 通信距離は目安

一般社団法人全国専門学校情報教育協会

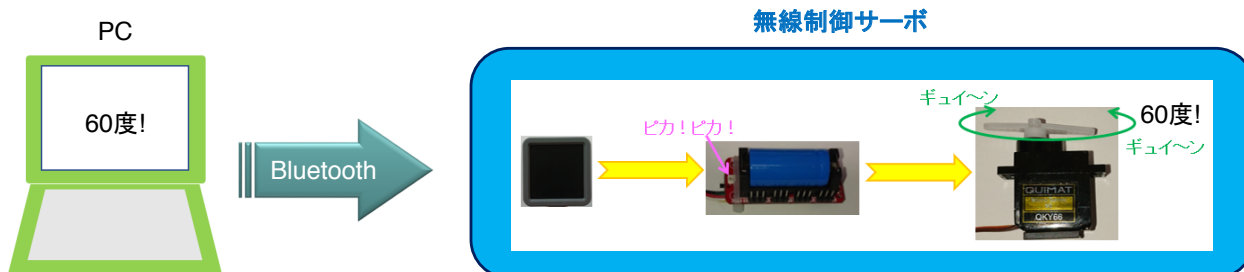
1

目論見 → 無線制御サーボシステム

◇PCからBluetooth経由でサーボモータの位置決め角度を制御する

→ 無線化サーボ

◇無線化 → IoTエンジニア必須スキル → 実験室的にも欠かせない

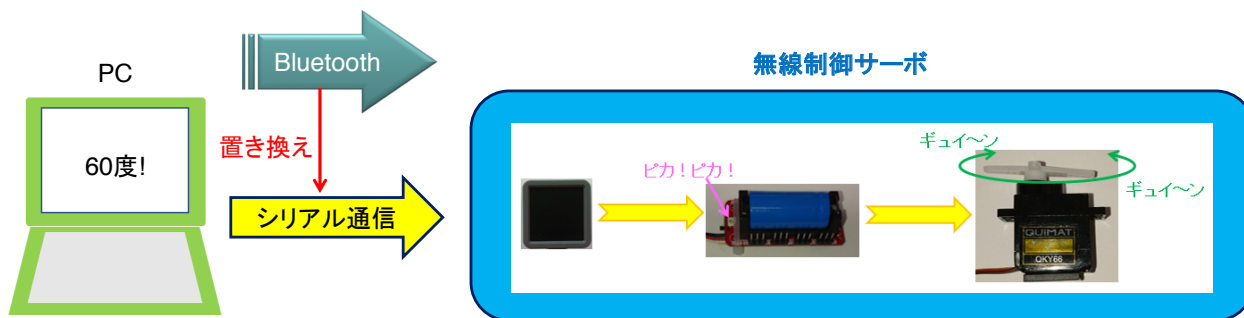


システム構想

◇2段階で開発する

①. MQTT経由サーボモータ制御システムをシリアル通信経由に改造する
→ PC から M5ATOM にシリアル通信でサーボモータ角度を与える

②. ①をBluetooth経由に改造する
→ BluetoothSerial ライブラリが使える
→ BluetoothSerialの使い方はシリアル通信と同じ





シリアル通信でサーボモータを制御する

SERVO MOTOR CONTROL BY SERIAL COMMUNICATION

ソースコード 1/3 (M5A_Servo_Serial_3)

◇初期化処理部

```
#include <M5Atom.h>
#include "IIC_servo.h"

// 内蔵加速度センサが利用するGPIOポートを開放する → コントローラICとの通信に使用する
bool IMU6886Flag = false; //sda:25 and scl:21 are free.

int i; // メッセージバッファインデックス
char msg[10]; // 受信メッセージ格納用バッファ

void setup(){ // 初期化
  M5.begin(true, true, true); //serial, I2C, LED
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init(); //sda:21 scl:25 Servo Controller用にIICを設定(初めに開放したポート)
  i=0; // メッセージバッファ初期化
}
```

ソースコード 2/3 (M5A_Servo_Serial_3)

◇通常処理部

```
void loop() { // 通常処理
  int a; // サーボモータ角度
  char c; // 受信データ 1文字分

  if(Serial.available()){ // 受信データあり?
    c = Serial.read(); // 1文字 Read
    msg[i++] = c; // 受信した文字を格納
    if(c == '\n'){ // 改行ならば電文の終端
      Serial.print("Received message ---> "); // 受信した角度表示
      Serial.print(msg); // 同上 角度部
      i=0; // メッセージバッファ初期化
      a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // 角度計算

      //エラーチェック部
      //LED点滅部
      //サーボ制御部
    }
  }
}
```

6

一般社団法人全国専門学校情報教育協会

ソースコード 3/3 (M5A_Servo_Serial_3)

◇エラーチェック部

```
if(a<0 || a>180){ // 角度チェック 0° から180° の範囲外はエラー
  Serial.println("Over Rotation Angle!!");
  return; // エラーメッセージを送信してOSに戻る
}
```

◇LED点滅部

```
RGB_set(128,0,0); // red ピカ!
delay(200); // しばし待つ
RGB_set(0,128,0); // green ピカ!
delay(200); // しばし待つ
RGB_set(0,0,128); // blue ピカ!
delay(200); // しばし待つ
RGB_set(0,0,0); //RGB 消灯
```

◇サーボ制御部

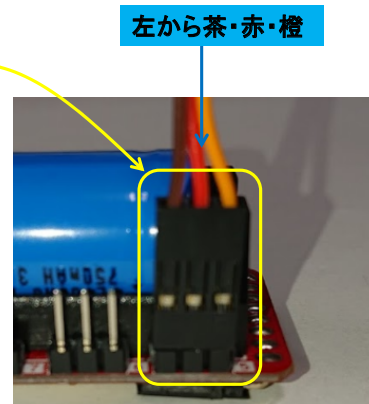
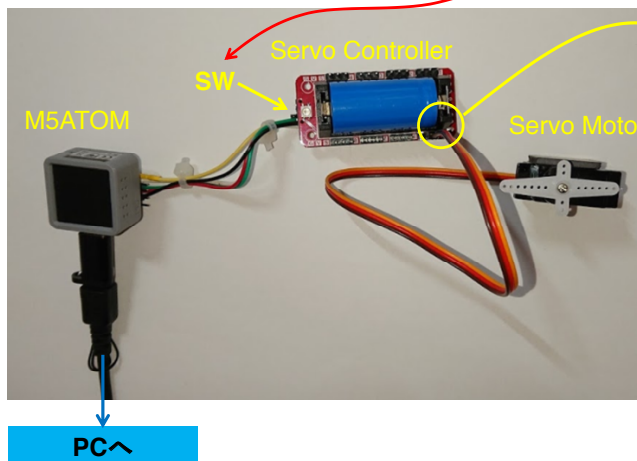
```
for(int j=1;j<9;j++){ // 全CH(どのピンに接続してもサーボが動く)
  Servo_angle_set( j, a ); // a=指定角度
}
```

7

一般社団法人全国専門学校情報教育協会

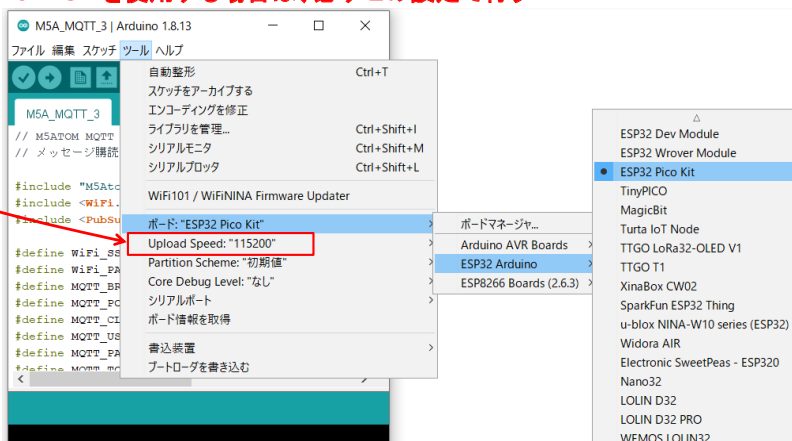
マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、外部基板などを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い



マイコンボードの選択

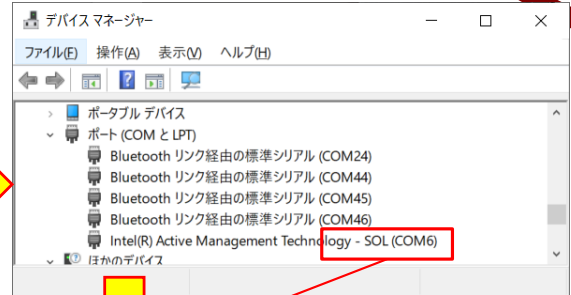
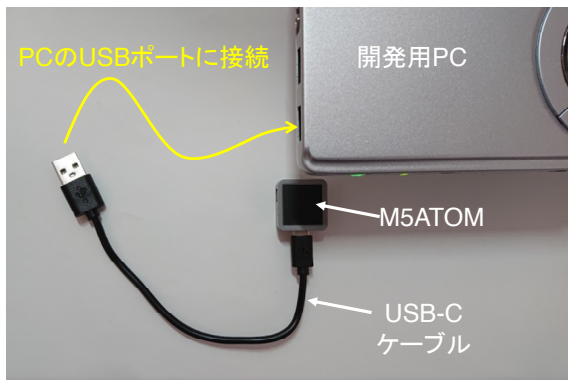
- ◇以下のように IDE で **ツール → ボード → ...とたどり、ESP32 Pico Kit** を選択する
- ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に **シリアルポートの Upload Speed は、115200 にセット**する(これより早いと書込みに失敗する)
- ◇以後、M5ATOMを使用する場合は、必ずこの設定で行う



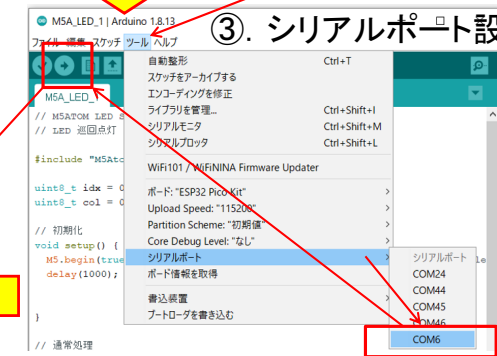
②. デバイスマネージャでCOMポート確認

マイコンをPCと接続

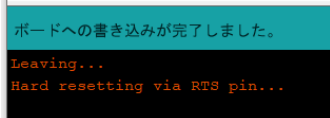
①. M5ATOMをPCと接続



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



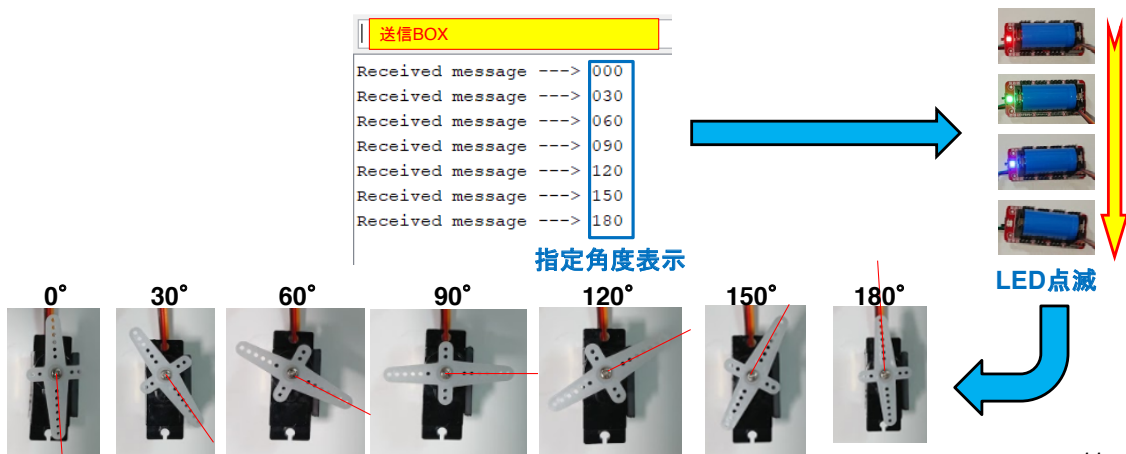
一般社団法人全国専門学校情報教育協会

10

動作確認

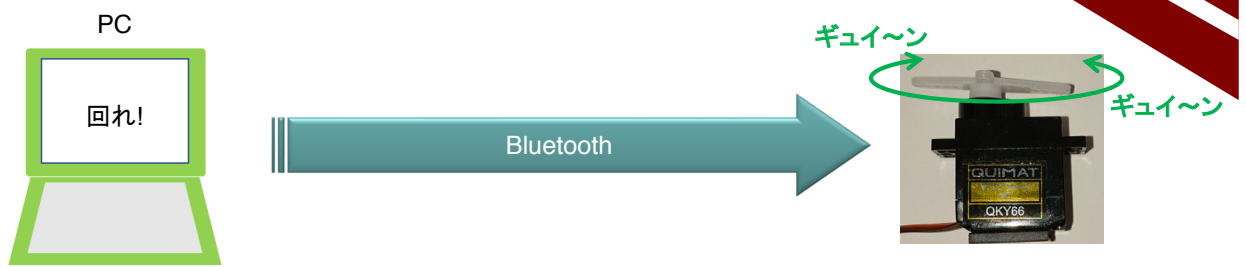
◇シリアルモニタを起動して(IDE右上の虫眼鏡マーク)0度から30度刻みで180度まで位置決めをしてみよう!

- ①. シリアルモニタの**送信BOX**に指定角度(0度なら000、90度なら090、120度なら120)を入力して**ENTER** または **送信ボタン**をクリックする
- ②. シリアルモニタに受信したメッセージ(指定角度)が表示された後、サーボコントローラのLEDが、**赤→緑→青**と点灯した後、ホーンが指定角度に位置決めされる様子が確認できる



一般社団法人全国専門学校情報教育協会

11



Bluetooth でサーボモータを制御する

SERVO MOTOR CONTROL BY BLUETOOTH

システム構想

- ◇シリアル通信を Bluetooth で行うプロトコル SPP (Serial Port Protocol) がある
- ◇いま動作確認を終えたシステムの通信を **Serial通信からBluetooth通信に置き換える**
- ◇M5ATOMの開発環境には SPP用 **BluetoothSerial ライブラリ**が含まれている

※IDEで スケッチ → ライブラリをインクルード とたどれば BluetoothSerial を確認できる

- ◇先に動作確認を終えたプログラムを流用すれば、容易に実現できる

→ ソースコードを少し変更するだけで実現できそう！！

ソースコード 1/3 (M5A_Servo_Bluetooth_Serial_4)

◇初期化処理部

```
#include <M5Atom.h>
#include "IIC_servo.h"
#include <BluetoothSerial.h>
BluetoothSerial bts; // Bluetooth Serial Object
// 内蔵加速度センサが利用するGPIOポートを開放する → コントローラICとの通信に使用する
bool IMU6886Flag = false; //sda:25 and scl:21 are free.
int i; // メッセージバッファインデックス
char msg[10]; // 受信メッセージ格納用バッファ

void setup(){ // 初期化 bts.setpin("1234");が必要な場合がある。
  M5.begin(true, true, true); //serial, I2C, LED
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init(); //sda:21 scl:25 Servo Controller用にIICを設定(初めに開放したポート)
  i=0; // メッセージバッファインデックス初期化
  bts.begin(" * * * * "); // PC側でペアリングするデバイス名称 ※各自ユニークな名称にする!!
  delay(1000); //しばし待つ
  Serial.print(" Servo Bluetooth Serial Control-4%n"); // Serial Terminal Message
  bts.print(" Servo Bluetooth Serial Control-4%n"); // Bluetooth Terminal Message
}
```

※青字部分を追加・変更する

14

一般社団法人全国専門学校情報教育協会

ソースコード 2/3 (M5A_Servo_Bluetooth_Serial_4)

◇通常処理部

```
void loop() { // 通常処理
  int a; // サーボモータ角度
  char c; // 受信データ 1文字分
  if(bts.available()){ // Bluetoothに受信データあり?
    c = bts.read(); // Bluetooth から1文字 Read
    msg[i++] = c; // 受信した文字を格納
    if(c == '\n'){ // 改行ならば電文の終端
      Serial.print("Received message ---> "); // 受信した角度表示
      Serial.print(msg); // 同上 角度部
      bts.print("Received message ---> "); // Bluetooth Terminal にも表示
      bts.print(msg);
      i=0; // メッセージバッファ初期化
      a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // 角度計算(文字コード→数値)
      //エラーチェック部
      //LED点滅部
      //サーボ制御部
    }
  }
}
```

15

一般社団法人全国専門学校情報教育協会

ソースコード 3/3 (M5A_Servo_Bluetooth_Serial_4)

◇エラーチェック部

```
if(a<0 || a>180){ // 角度チェック 0° から180° の範囲外はエラー
  bts.println("Over Rotation Angle!!"); // Bluetooth にエラーメッセージ出力
  return; // エラーメッセージを送信してOSIに戻る
}
```

◇LED点滅部

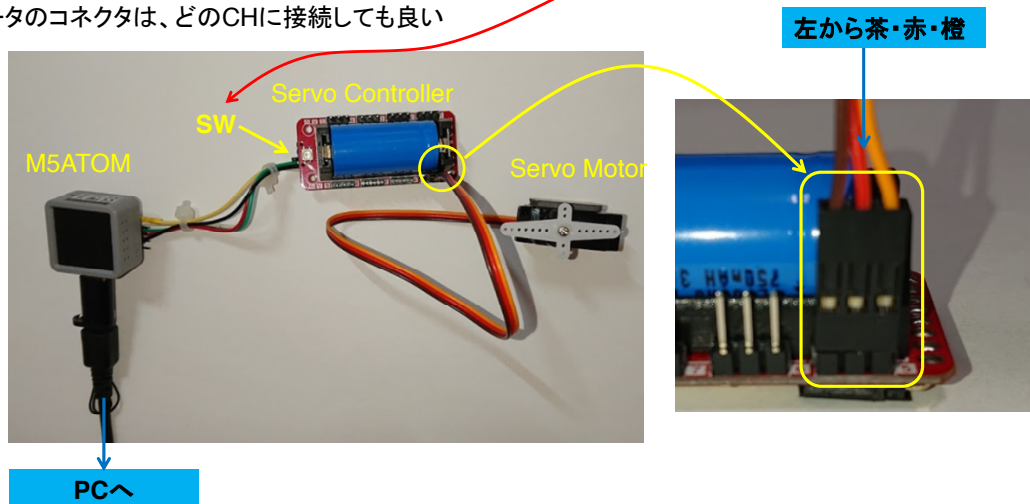
```
RGB_set(128,0,0); // red ピカ！
delay(200); // しばし待つ
RGB_set(0,128,0); // green ピカ！
delay(200); // しばし待つ
RGB_set(0,0,128); // blue ピカ！
delay(200); // しばし待つ
RGB_set(0,0,0); //RGB 消灯
```

◇サーボ制御部

```
for(int j=1;j<9;j++){ // 全CH(どのピンに接続してもサーボが動く)
  Servo_angle_set( j, a ); // a=指定角度
}
```

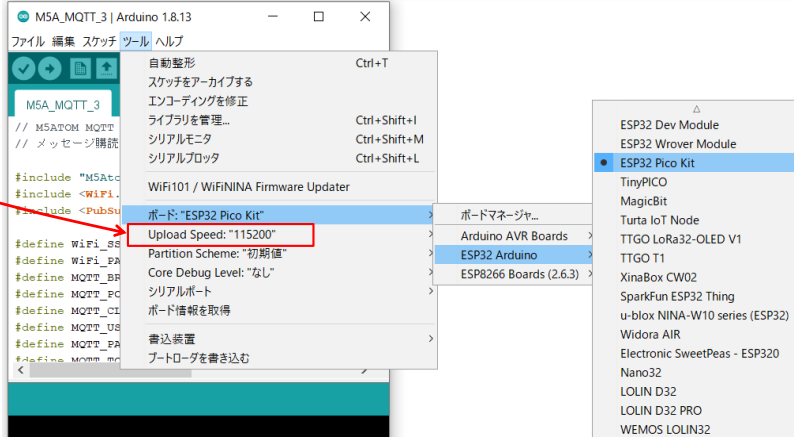
マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、外部基板などを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHに接続しても良い



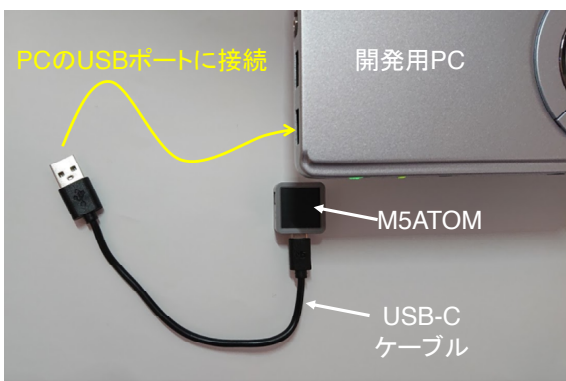
マイコンボードの選択

- ◇ 以下のように IDE で ツール → ボード → ...とたどり、ESP32 Pico Kit を選択する
- ※ ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ← M5Atom ライブラリが無い場合
- ◇ 同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇ 以後、M5ATOMを使用する場合は、必ずこの設定で行う

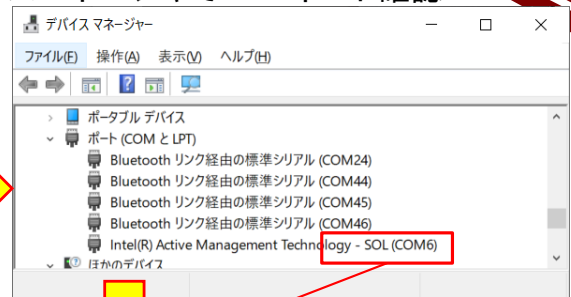


マイコンをPCと接続

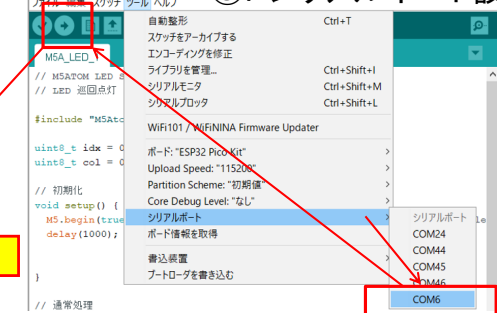
①. M5ATOMをPCと接続



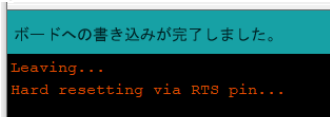
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



動作確認 1/3 Bluetooth Device の追加(ペアリング)

◇書き込みが完了したら、数秒待って、以下の手順でデバイスのペアリングを行う

①スタート→設定



②デバイス



③Bluetoothとその他のデバイス



④デバイスを追加する



⑤プログラム指定のデバイス名選択



⑤ペアリング完了



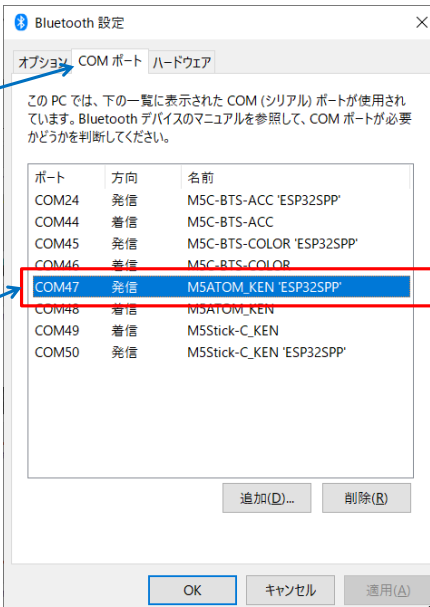
一般社団法人全国専門学校情報教育協会

20

動作確認 2/3 Bluetooth Serial の発信用ポート番号を確認

- ◇スタートボタンから
設定→デバイス→その他のBluetoothオプション
とたどり、【COMポート】タブを選択する
- ◇ペアリングしたBluetoothデバイス名の方が【発信】と
なっているCOMポート番号を確認しメモする
- ◇メモしたCOMポート番号をIDEのシリアルポートで
選択する

発信のCOMポート



ポート	方向	名前
COM24	発信	M5C-BTS-ACC 'ESP32SPP'
COM44	着信	M5C-BTS-ACC
COM45	発信	M5C-BTS-COLOR 'ESP32SPP'
COM46	着信	M5C-BTS-COLOR
COM47	発信	MSATOM_KEN 'ESP32SPP'
COM48	着信	MSATOM_KEN
COM49	着信	M5Stick-C_KEN
COM50	発信	M5Stick-C_KEN 'ESP32SPP'

一般社団法人全国専門学校情報教育協会

21



動作確認 3/3

- ◇シリアルターミナルを開く(IDE右上の虫眼鏡マーク)
- ◇送信BOXに 000 ~ 180 の角度を入力してENTER キー(または送信ボタン)を押下する

```

送信BOX
Received message ---> 000
Received message ---> 030
Received message ---> 060
Received message ---> 090
Received message ---> 120
Received message ---> 150
Received message ---> 180
Received message ---> 000

```

- ◇シリアルモニタに M5ATOMが受信したメッセージが表示され、その後LEDが赤→緑→青と点灯し、サーボモータが指定角度に位置決めされる



2.8 LEDの動作確認

M5Stick-C

センシング用マイコン



M5Stick-Cのパッケージ

◇M5Stick-Cには、図のようなパッケージがある

①本体とUSBケーブルのみ



◇この講座では、最もシンプルな
①のパッケージを使う
◇付属USBケーブルとは別の
長いものが実習キットに付属
している

②各種マウンター付属



③液晶が大きいサイズ



1

一般社団法人全国専門学校情報教育協会

内蔵LEDを点灯・消灯させてみよう！

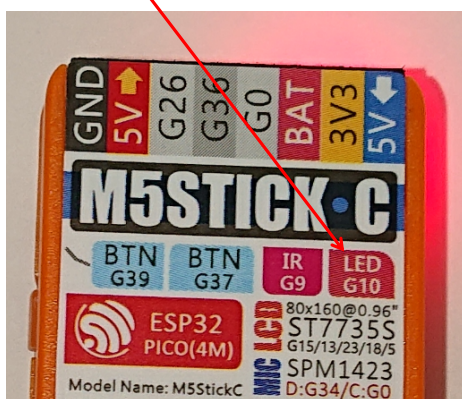
LED点滅

2

一般社団法人全国専門学校情報教育協会

実装しているLED

- ◇M5Stick-C は、1個の赤色LEDを内蔵している
- ◇1個でもLEDは強力な表示機能を提供する
- ◇裏面にあるピン配置でLEDがどのGPIOに接続されているかが分かる



センシング用



M5Stick-C

システム構想

- ◇内蔵LEDを単純に点滅させる → これはどのマイコンでも、初めにやってみること
- ◇M5Stick-Cのマイコンボードライブラリは、マイコン開発環境の準備でIDE内にインストールされている
→ ソースコードの冒頭で以下のようにライブラリを取り込む

```
#include <M5StickC.h>
```
- ◇マイコンボードを初期化するために、M5.begin() 関数を setup() 関数内で呼び出す
- ◇LEDを制御するピンは、ライブラリ中で以下のように定義されている
M5_LED または GPIO_NUM_10
※前の図で示したM5Stick-C裏面の LED が G10 の表記は GPIO_NUM_10 を意味する
なお、GPIOとは、汎用の入出力(General Purpose Input Output)を意味している
- ※GPIOは入出力どちらにでも使える信号
→ 入力か出力かは、プログラムで指定する → 通常これは setup() 関数の中で行う
※通常の処理中にI/Oを切り替えることもある
- ◇内蔵LEDは、制御信号が負論理なので、LEDの制御では注意する
→ 正論理 : High→点灯、Low→消灯
→ 負論理 : High→消灯、Low→点灯

ソースコード 1/1 (M5C_LED_1)

◇初期化部分までのソースコード(C++)

```
#include <M5StickC.h>           // 対象マイコンのライブラリ

void setup(){                   // 初期化部
  M5.begin();

  // LED → GPIO_NUM_10 または M5_LEDでアクセスできる
  pinMode(GPIO_NUM_10, OUTPUT);

}

void loop() {                  // 通常処理部
  digitalWrite(GPIO_NUM_10, LOW); // GPIO_NUM_10で点灯(負論理)
  delay(2000);                 // しばし待つ (2000ms = 2sec)
  digitalWrite(M5_LED, HIGH);  // M5_LEDで消灯(負論理)
  delay(2000); // しばし待つ
}
```

※両方を使ってみただけ

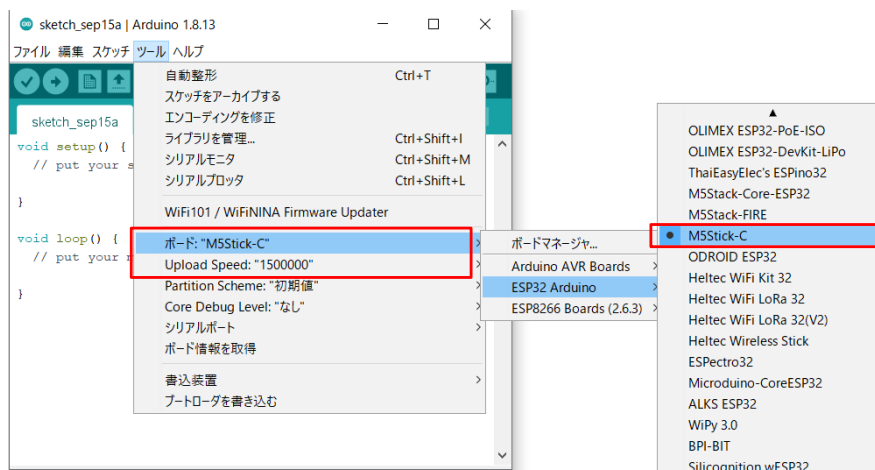
TABキーまたはスペースで行頭をそろえる

一般社団法人全国専門学校情報教育協会

5

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、M5Stick-C を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、M5Stick-Cを使用する場合は、必ずこの設定で行う

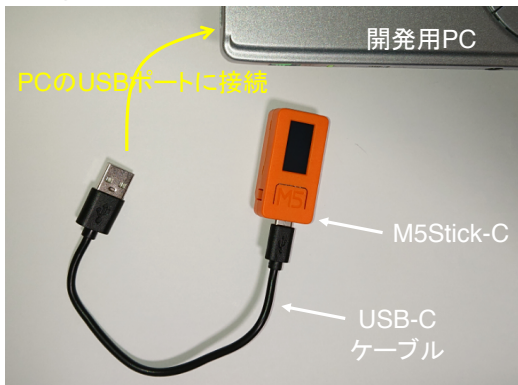


一般社団法人全国専門学校情報教育協会

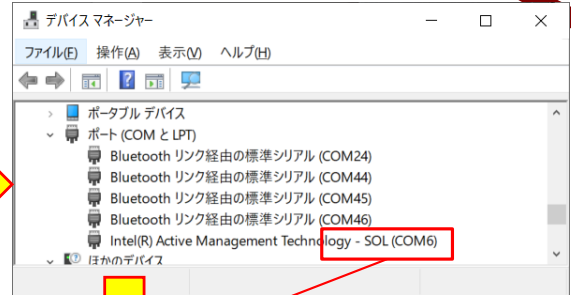
6

マイコンをPCと接続

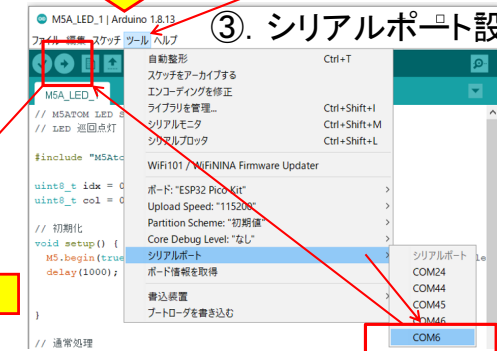
①. M5Stick-CをPCと接続



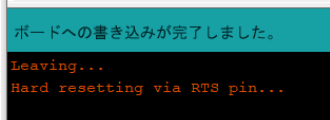
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル

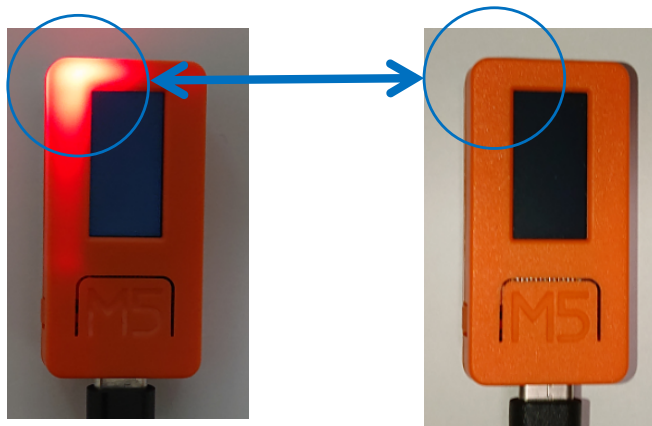


一般社団法人全国専門学校情報教育協会

7

動作確認

- ◇新しいプログラムは書き込みが終了すると、自動的にスタートする
- ◇LEDはプログラムで指定した delay() の間隔で点滅を繰り返す



- ◇LEDは、点灯するだけでは気づかれない場合もある
→ 短い間隔で点滅を繰り返したのち、点灯するなど、点灯パターンを変えることで十分な表示機能が得られる ※点灯パターンにより内部エラーを知らせる等

一般社団法人全国専門学校情報教育協会

8

2.9 Button クラスの動作確認

M5Stick-C

Button(押しボタンスイッチ)



一般社団法人全国専門学校情報教育協会

M5Stick-C Buttonクラス

- ◇ M5Stick-Cには、M5ATOMと同様にライブラリ中にButtonというSW(スイッチ)を取り扱うButtonクラスがある
- ◇ その中でボタンの状態を検出できる関数が定義されている
- ◇ IDEのデフォルトのスケッチ保存先の以下のパスにヘッダファイルがある
C:¥Users¥user¥Documents¥Arduino¥libraries¥ M5StickC ¥src¥utility¥Button.h
- ◇ ヘッダファイル内のButtonクラス定義部分を下に示す

```
class Button {
public:
    Button(uint8_t pin, uint8_t invert, uint32_t dbTime);
    uint8_t read();
    uint8_t isPressed();
    uint8_t isReleased();
    uint8_t wasPressed();
    uint8_t wasReleased();
    uint8_t pressedFor(uint32_t ms);
    uint8_t releasedFor(uint32_t ms);
    uint8_t wasReleasefor(uint32_t ms);
    uint32_t lastChange();
};
```

...

一般社団法人全国専門学校情報教育協会

1

M5Stick-Cの2つのボタン

- ◇ M5Stick-Cには、プログラムで使用できる【Aボタン】と【Bボタン】がある
 - プログラムからはそれぞれ M5.BtnA と M5.BtnB でアクセスできる
- ◇ この BtnA と BtnB は、IDEのデフォルトのスケッチ保存先の以下のパスにあるヘッダファイルで定義されている

C:\Users\user\Documents\Arduino\libraries\M5StickC\src\M5StickC.h

→ 右に該当部分を示す



```

82 ↓
83 ↓
84 ↓
85 public:↓
86 M5StickC();↓
87 void begin(bool LCDEnable=true, bool PowerEnable=true);↓
88 void update();↓
89 ↓
90 //!LCD↓
91 M5Display Lcd = M5Display();↓
92 ↓
93 //!Power↓
94 AXP192 Axp = AXP192();↓
95 ↓
96 #define DEBOUNCE_MS 10↓
97 Button BtnA = Button(BUTTON_A_PIN, true, DEBOUNCE_MS);↓
98 Button BtnB = Button(BUTTON_B_PIN, true, DEBOUNCE_MS);↓
99 //!RTC↓
100 RTC Rtc;↓
101 ↓
  
```

Buttonクラスの機能

- ◇ Buttonの関数は、以下の機能がある
- ◇ ここでは、M5.BtnA.wasPressed() のようにしてボタンの押下状態を調べる
- ◇ ボタンの状態は M5.update() を用いて更新されるので、通常処理の中でこの関数をCallする

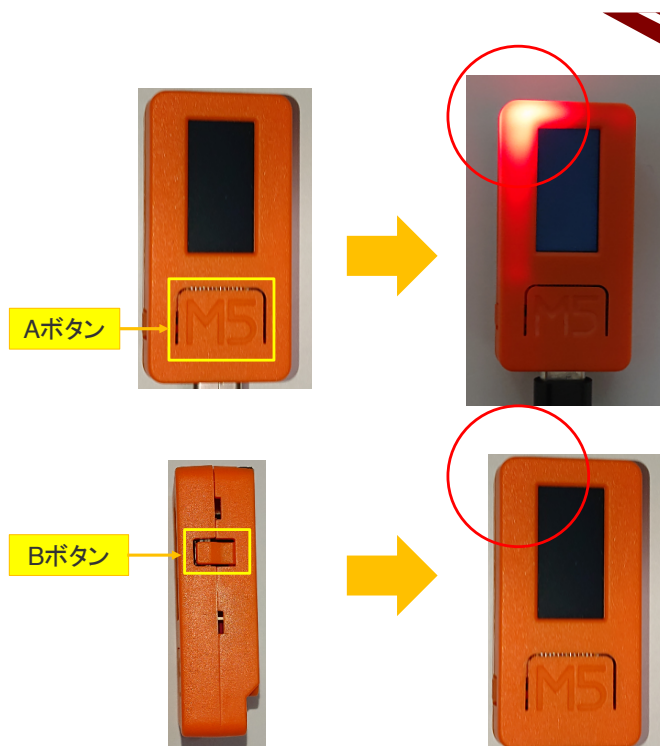
関数	機能
M5.update()	ボタンの状態を更新する関数 ※loop()関数内で必ず実行する
isPressed()	ボタンを押しているかどうかを返す ※ボタンを押している間は常にTRUEが戻る
isReleased()	ボタンを離しているかどうかを返す ※ボタンを押していない間は常にTRUEが戻る
wasPressed()	ボタンを押してから最初に呼び出した時だけ、TRUEを返す
wasReleased()	ボタンを押して、離してから最初に呼び出した時だけTRUEを返す
pressedFor(ms)	ボタンを指定時間以上押している場合にTRUEが返される
releasedFor(ms)	ボタンを離してから指定時間以上経過している場合にTRUEが返れる
wasReleasefor(ms)	指定時間以上ボタンを押し、離してから最初に呼び出した時だけTRUEを返す
lastChange()	最後にボタンの状態が変更された時の millis() の値が返却される 現在のmillis()からの差分が経過時間になる

システム構想

◇2つのボタンの押下で以下の処理を行う

- ①Aボタン押下 → LED点灯
- ②Bボタン押下 → LED消灯

◇SWの押下が検出できれば、IoTで様々な事象を記録できる



一般社団法人全国専門学校情報教育協会

4

ソースコード 1/2 (M5C_Button_1)

◇初期化部分までのソースコード
※M5A_LED_1 と同じ

```
#include <M5StickC.h> // M5Stick-Cライブラリ

void setup(){ // 初期化部
  M5.begin(); // M5Stick-Cリセット

  // LED ON(GPIO_NUM_10 or M5_LED) // どちらでアクセスしても良い
  pinMode(GPIO_NUM_10, OUTPUT); // GPIOピンを出力に設定する
  digitalWrite(M5_LED, HIGH); // M5_LEDで消灯
}
```

一般社団法人全国専門学校情報教育協会

5

ソースコード 2/2 (M5C_Button_1)

◇通常処理部分

※ボタンの押下状態を調べるために `M5.BtnA.wasPressed()` `M5.BtnB.wasPressed()` をCallする
また、ボタン状態の更新のために `M5.update()` をCallする

```
void loop() {  
  M5.update(); // M5Stick-C 状態を更新  
  
  if(M5.BtnA.wasPressed()== true) {  
    digitalWrite(GPIO_NUM_10, LOW); // GPIO_NUM_10で点灯  
  }  
  
  if(M5.BtnB.wasPressed()== true) {  
    digitalWrite(M5_LED, HIGH); // M5_LEDで消灯  
  }  
}
```

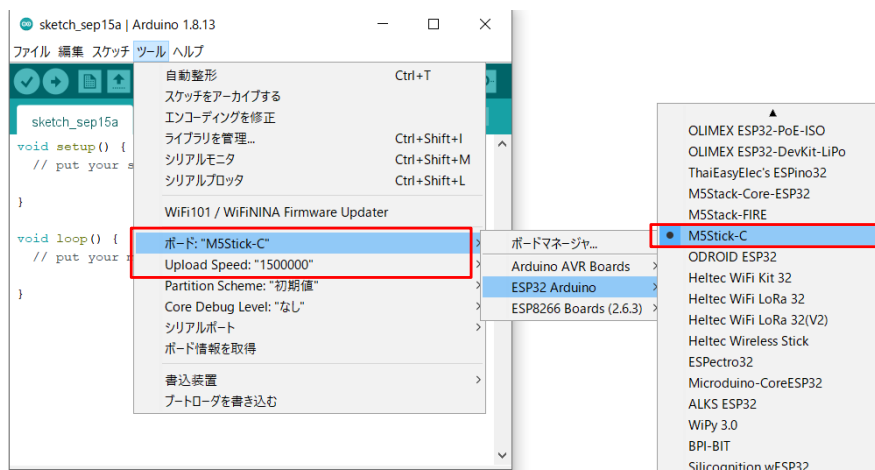
マイコンボードの選択

◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する

※ボード以後の表示は、使用しているIDEの状況に応じて変わる

◇同様に シリアルポートの Upload Speed は、1500000 にセットする

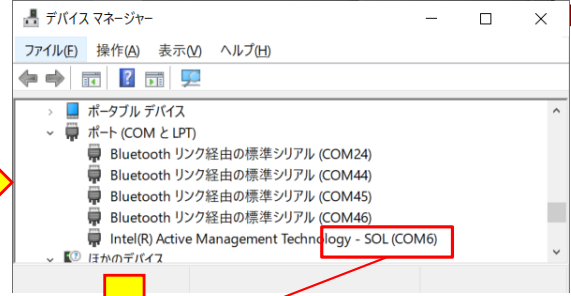
◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う



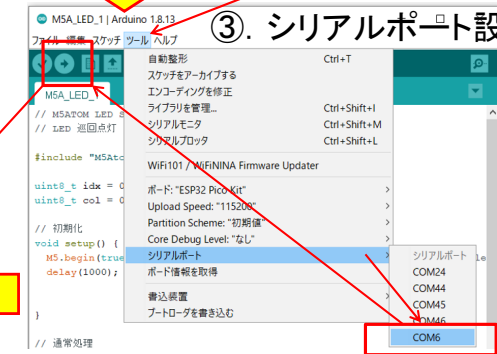
②. デバイスマネージャでCOMポート確認

マイコンをPCと接続

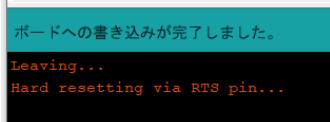
①. M5Stick-CをPCと接続



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



一般社団法人全国専門学校情報教育協会

8

動作確認

- ◇新しいプログラムは書き込みが終了すると、自動的にスタートしている
- ◇Aを押下するとLEDが点灯し、Bボタンを押下するとLEDが消灯することを確認しよう！



◇これで2つのボタンが使えるようになった

※左側面のボタンもプログラムで使用することができるが、電源制御が絡むので利用する際は、慎重に調査してからシステムを開発すること

一般社団法人全国専門学校情報教育協会

9

2.10 シリアル通信計測

M5Stick-C

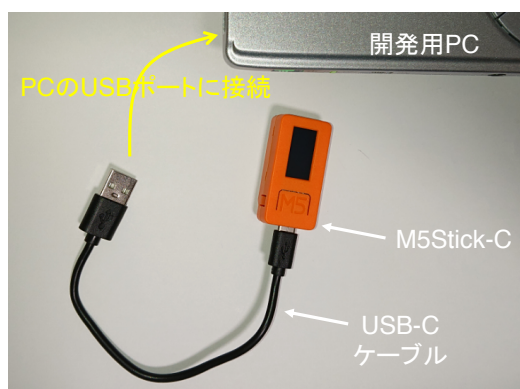
シリアル通信



一般社団法人全国専門学校情報教育協会

M5Stick-CもPCとUSB接続される

- ◇M5Stick-Cは、プログラム書き込みの際にはPCとUSB接続します
- ◇マイコンはUSB経由のシリアル通信によって受け取ったプログラムをフラッシュメモリに書き込みます
- ◇今回は、このシリアル通信を利用して、PCとの間で通信を行います



一般社団法人全国専門学校情報教育協会

1



固定のメッセージをPCに向けて送信する

送信

システム構想

- ◇単純なメッセージ送信を行う
 - ◇メッセージは固定
 - ◇既にLEDとボタンが使えるようになっている
 - ボタン押下に同期して【LED点灯制御+メッセージ送信】を行う
- M5Stick-Cには A、B 2つのボタンがある

◇以下のライブラリ関数が準備されている

- ①. ボタン押下 → M5.update() と M5.BtnA.wasPressed()
M5.BtnB.wasPressed()
- ②. LED点灯 → digitalWrite(M5_LED, LOW) ※負論理なのでLOWで点灯
- ③. LED消灯 → digitalWrite(M5_LED, HIGH)

- ③. メッセージ送信 → Serial.print() または Serial.println()
※押されたボタンをメッセージで伝える

◇過去の資産(ボタンのプログラム)の流用を考えて進める

シリアル通信のデフォルト速度

- ◇M5Stick-Cは、シリアルポートの初期化処理も、デフォルトでライブラリが処理をする
- ◇C:\Users\user\Documents\Arduino\libraries\M5Atom\src\M5StickC.cpp の該当部分を示す

```
10 void M5StickC::begin(bool LCDEnable, bool PowerEnable, bool SerialEnable){  
11     ↓  
12     ///  
13     if (isInited) return; ↓  
14     else isInited = true; ↓  
15     ↓  
16     ///  
17     if (SerialEnable) { ↓  
18         Serial.begin(115200); ↓  
19         Serial.flush(); ↓  
20         delay(50); ↓  
21         Serial.print("M5StickC initializing..."); ↓  
22     } ↓
```

※この部分でシリアルポートの初期化が行われている Serial.begin()のパラメータが通信速度
初期メッセージは改行コードを出力していないので Serial.println() に変更しても良い

ソースコード 1/2 (M5C_Serial_1)

- ◇初期化処理まで

```
#include <M5StickC.h>           // マイコンボードライブラリ  
  
void setup(){                   // 初期化部  
    M5.begin();  
  
    // LED ON(GPIO_NUM_10 or M5_LED)  
    pinMode(M5_LED, OUTPUT);    // LEDピンを出力に設定する  
    digitalWrite(M5_LED, HIGH); // LEDをあらかじめ消灯しておく  
}
```


ソースコード 2/2 (M5C_Serial_1)

◇通常処理では 押されたボタンを通知するメッセージを送信している

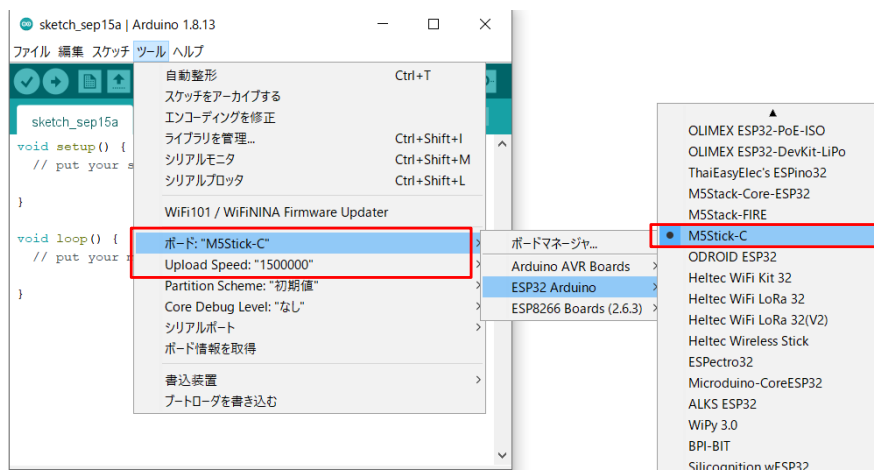
```
void loop() { // 通常処理部
  M5.update(); // M5Stick-C 状態を更新

  if(M5.BtnA.wasPressed()== true) { // Aボタンが押下されたか？
    digitalWrite(M5_LED, LOW); // LED点灯(負論理)
    Serial.println("Button A was Pressed!!"); // Aボタンが押された
  }

  if(M5.BtnB.wasPressed()== true) { // Bボタンが押下されたか？
    digitalWrite(M5_LED, HIGH); // LED消灯(負論理)
    Serial.println("Button B was Pressed!!"); // Bボタンが押された
  }
}
```

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

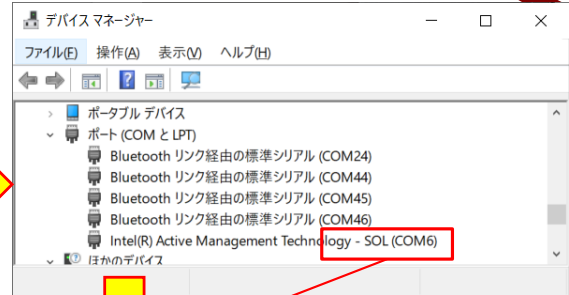


マイコンをPCと接続

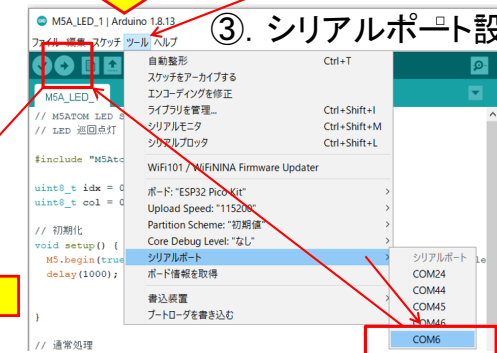
①. M5Stick-CをPCと接続



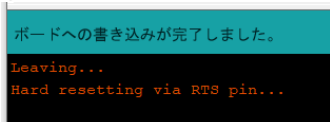
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



一般社団法人全国専門学校情報教育協会

8

動作確認 1/3

- ◇新しいプログラムは書き込みが終了すると、自動的にスタートしている
- ◇Aを押下するとLEDが点灯し、Bボタンを押下するとLEDが消灯することを確認しよう！



◇これで2つのボタンが使えるようになった

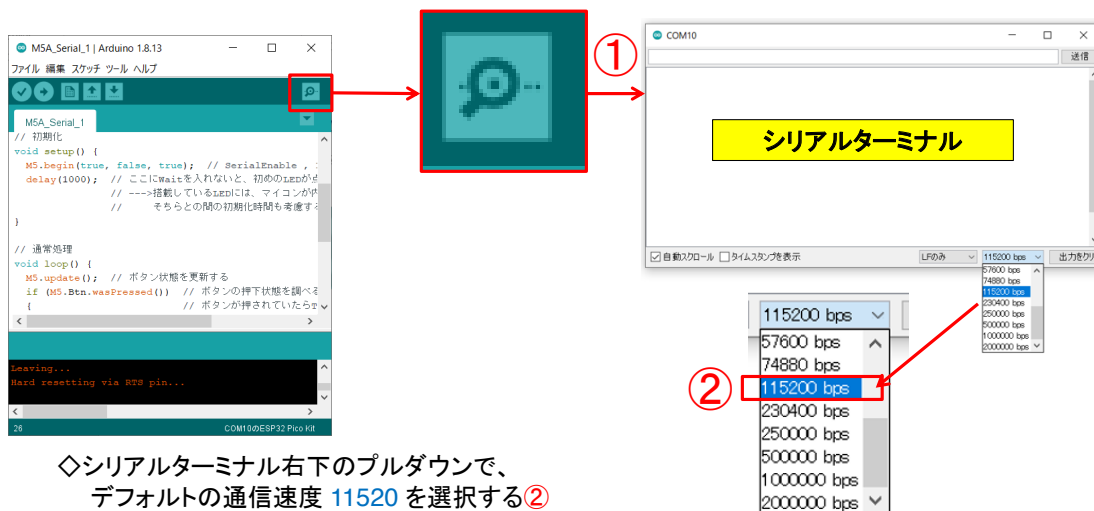
※左側面のボタンもプログラムで使用することができるが、電源制御が絡むので利用する際は、慎重に調査してからシステムを開発すること

一般社団法人全国専門学校情報教育協会

9

動作確認 2/3 メッセージ送信の確認準備

◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①

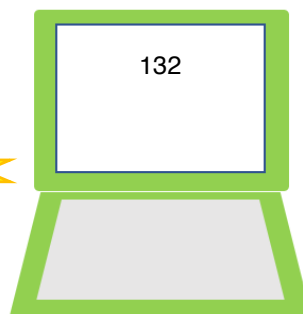
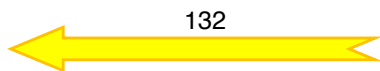


◇シリアルターミナル右下のプルダウンで、デフォルトの通信速度 11520 を選択する②

動作確認 3/3 メッセージ送信の確認

◇Aボタンを押すとLEDが点灯し、メッセージ(Button A was pressed!!)が表示される
 ◇Bボタンを押すとLEDが消灯し、メッセージ(Button B was pressed!!)が表示される





PCからメッセージを受信する

受信

システム構想

- ◇次は受信を行う
- ◇受信メッセージでLEDを制御しよう
- ◇メッセージ設計と処理設計
 - ①. メッセージ長 : 1バイト+改行コード
 - ②. 改行コードまで受信して、メッセージの解析と対応する処理を行う
 - ③. 受信バッファは、文字型配列で10バイト程度確保する(誤った長いメッセージに対応)
 - ④. メッセージの内容
 - 0文字目:LED ON/OFF 指示 0:消灯 1:点灯 それ以外:点滅

◇以下のライブラリ関数を使用する

- ①. Serial.available() → 受信バッファにデータがあるか調べる
- ②. Serial.read() → 1文字受信する
- ③. digitalWrite(M5_LED, LOW) → LED点灯
- ④. digitalWrite(M5_LED, HIGH) → LED消灯

ソースコード 1/3 (M5C_Serial_2)

◇初期化処理まで

```
#include <M5StickC.h>           // マイコンボードライブラリ

int i;                          // 1文字受信時のバッファインデックス
char buf[10];                   // メッセージバッファ

void setup(){                   // 初期化部
  M5.begin();

  pinMode(M5_LED, OUTPUT);      // LEDピンを出力に設定する
  digitalWrite(M5_LED, HIGH);  // LEDをあらかじめ消灯しておく
  i=0;                          // バッファインデックス初期化
}
```

ソースコード 2/3 (M5C_Serial_2)

◇通常処理 最も外側部分

```
void loop() {                  // 通常処理部
  char c;                      // 受信データ 1文字分

  if(Serial.available()){      // 受信データあり?
    c = Serial.read();         // 1文字 Read
    buf[i++] = c;              // 受信した文字を格納
    if(c == '\n'){             // 改行ならば電文の終端
      i=0;                     // バッファインデックス初期化

      // LED点灯制御部
    }
  }
}
```

ソースコード 3/3 (M5C_Serial_2)

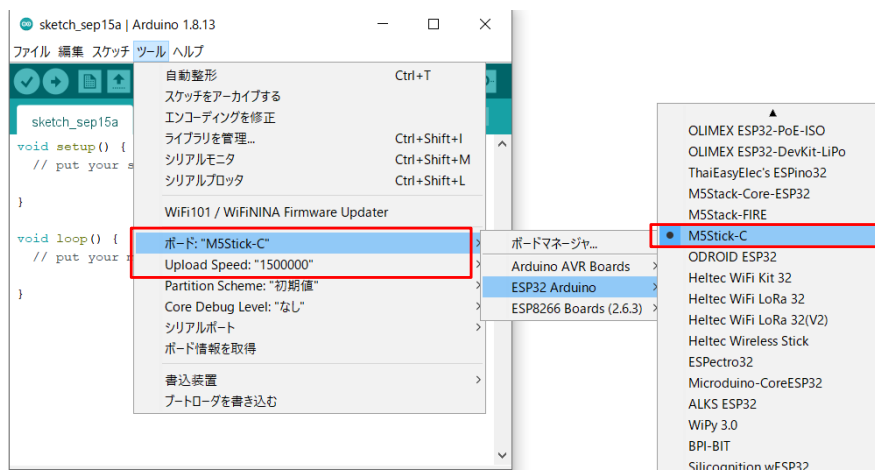
◇LED点灯制御部

```
switch (buf[0]){
  case '0': // 0→LED消灯
    digitalWrite(M5_LED, HIGH); // LED消灯(負論理)
    Serial.println("LED OFF!!"); // LED消灯メッセージ
    break;
  case '1': // 1→LED点灯
    digitalWrite(M5_LED, LOW); // LED点灯(負論理)
    Serial.println("LED ON!!"); // LED点灯メッセージ
    break;
  default: // 不明なメッセージ
    digitalWrite(M5_LED, LOW); // LED点灯(負論理)
    delay(100); // しばし待つ
    digitalWrite(M5_LED, HIGH); // LED消灯(負論理)
    delay(100); // しばし待つ
    ... ..
    Serial.println("Unknown command!!"); // 不明なコマンドメッセージ
    break;
}
```

LEDピカピカ

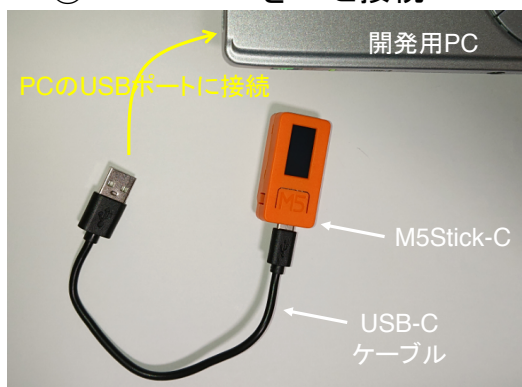
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

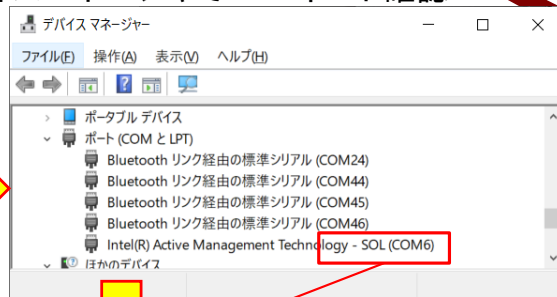


マイコンをPCと接続

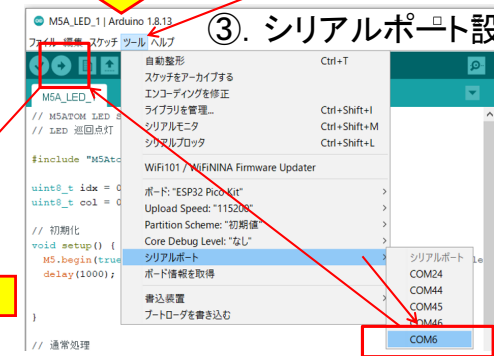
①. M5Stick-CをPCと接続



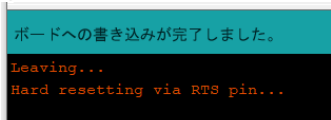
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル

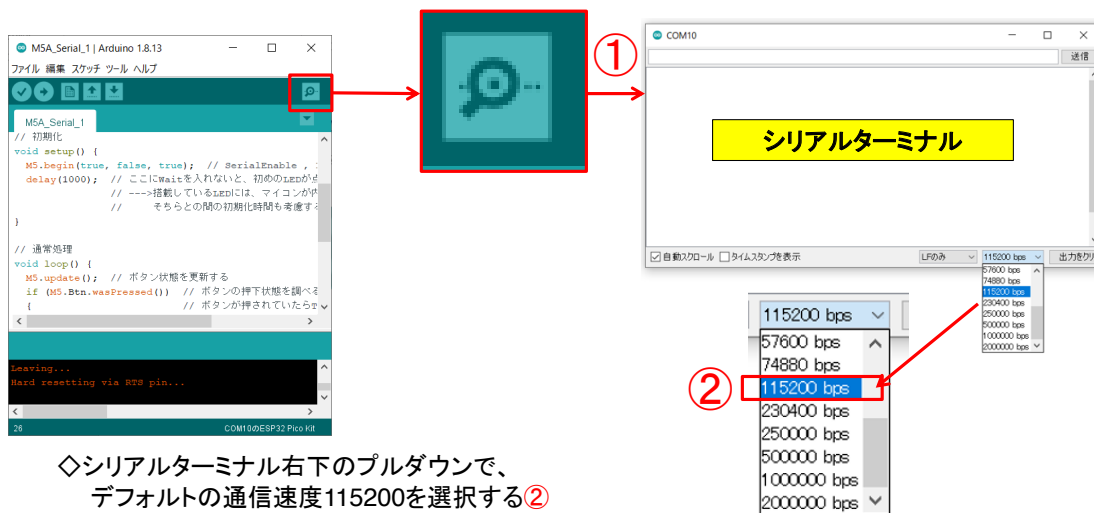


一般社団法人全国専門学校情報教育協会

18

動作確認 1/2 メッセージ受信の確認準備

◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①



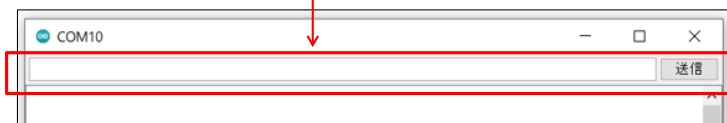
◇シリアルターミナル右下のプルダウンで、デフォルトの通信速度115200を選択する②

一般社団法人全国専門学校情報教育協会

19

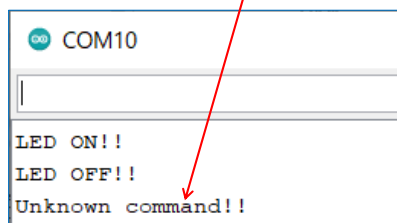
動作確認 2/2

◇シリアルターミナルの送信BOXに以下の電文を記述して ENTER Key を押下する
または 送信ボタンをクリックする



◇電文を 1桁の半角数字で入力する

- ①. 1 ENTER → LEDが点灯しLED ONメッセージ
- ②. 0 ENTER → LEDが消灯しLED OFFメッセージ
- ③. 0, 1以外 ENTER → LEDが点滅しUnknown command!!メッセージ



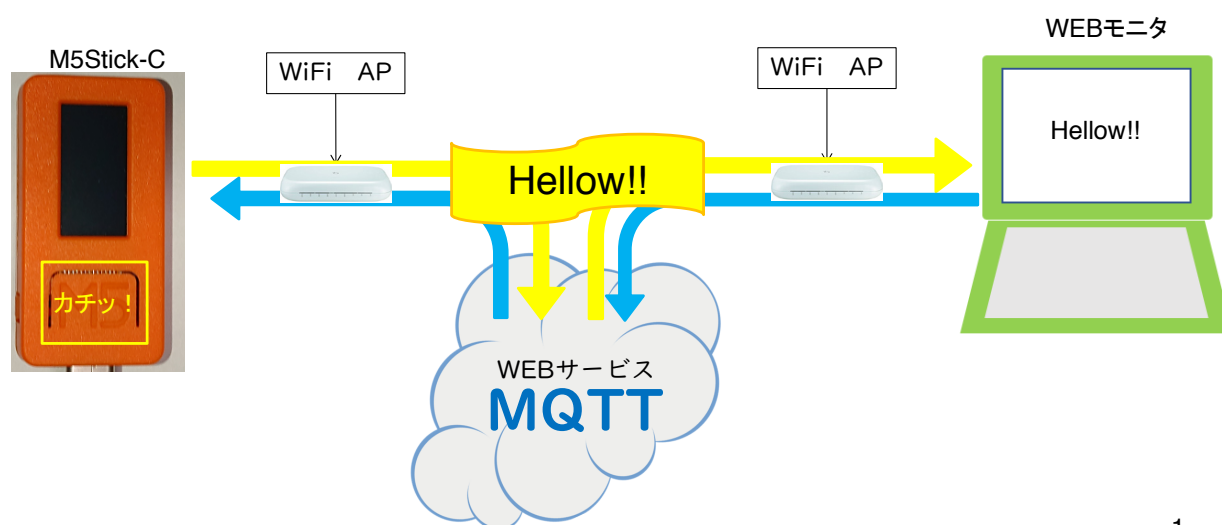
2.1.1 WEB 経由の通信計測

M5Stick-C

MQTT (WEB経由通信)



目論見 → WEB経由メッセージ交換



一般社団法人全国専門学校情報教育協会

1

MQTTサービス

- ◇MQTTは、Message Queue Telemetry Transport の頭文字
- ◇サービスプロバイダ (MQTT Broker) が世界中にある
- ◇短いメッセージ交換に特化したWEBサービス
- ◇登録不要で即利用できる Broker が多い

- ◇WEBを経由するメッセージ交換 → インターネットに接続できれば場所を選ばない
- ◇マイコン⇄マイコン間、PC⇄PC間、マイコン⇄PC間でのメッセージ交換が可能
- ◇言語依存しない (マイコン向けC++・MicroPython・PC用Python・同C++・・・etc)
ライブラリが必要 → 本講座では PubSubClient を利用するのでIDEにインストールする

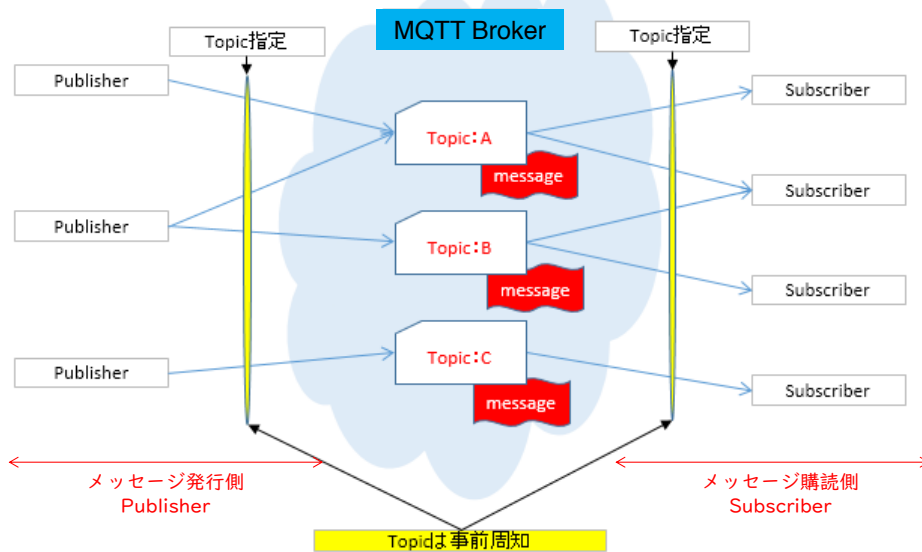
- ◇ROS (Robot OS) でnode間通信にも採用されている仕組み
- ◇ルールが簡単

一般社団法人全国専門学校情報教育協会

2

MQTTの仕組み

◇トピック名付きメッセージを発行すると、購読登録しているクライアントに通知される



一般社団法人全国専門学校情報教育協会

3

MQTT Broker

◇サービスを提供している MQTT Broker は世界中に沢山ある

- test.mosquito.org
- broker.hivemq.com
- broker.shiftr.io ← 今回利用する
- broker.mqttdashboard.com
- iot.eclipse.org

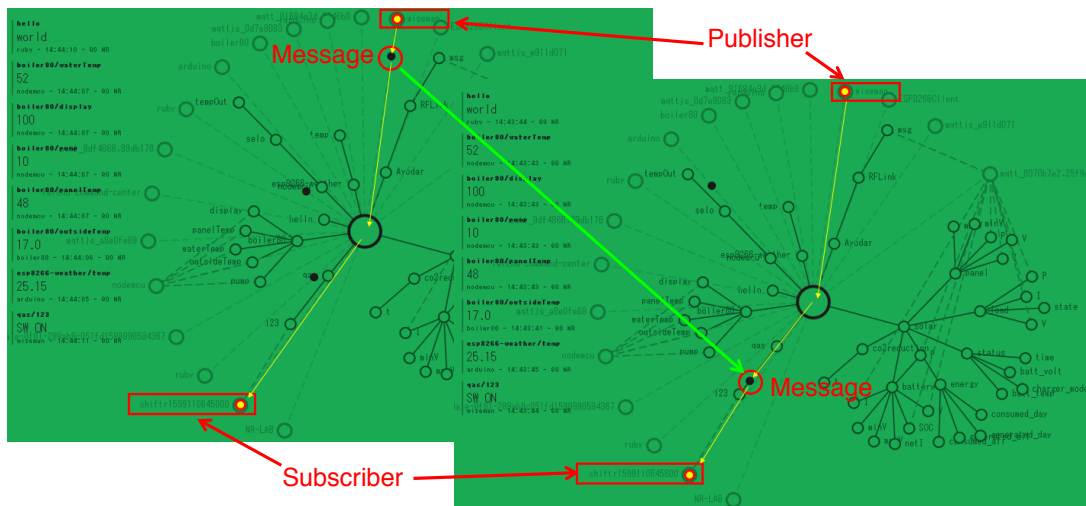
... etc

一般社団法人全国専門学校情報教育協会

4

broker.shiftr.io

- ◇shiftr.io は、オープンMQTTの利用が認められており、図のようにPC上でメッセージの動きが見える
- ◇赤丸○で示しているのが発行したメッセージの流れである 中央の大きな丸○はBrokerを意味する

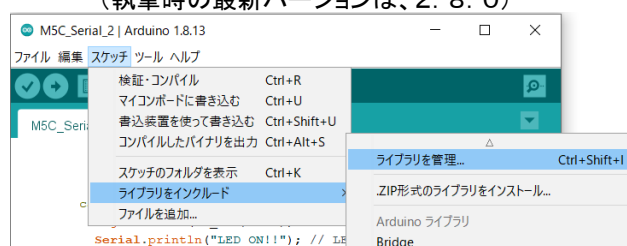


一般社団法人全国専門学校情報教育協会

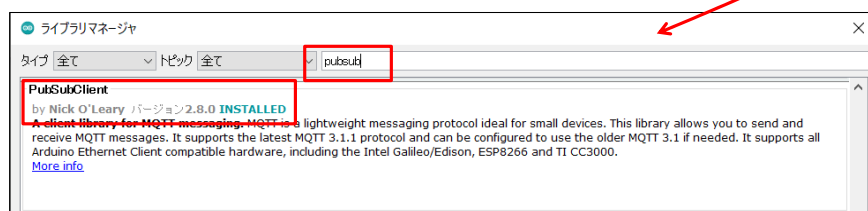
5

ライブラリの準備

- ◇IDEで **スケッチ** → **ライブラリをインクルード** → **ライブラリを管理** → **ライブラリマネージャ**を開く
- ◇検索窓に【**pubsub**】と入力して表示される一覧から **PubSubClient by Nick O'Leary** を選択し、インストールする → **Installed** と表示される
- ※同様のライブラリが数多く存在するので他のものと混同しないように！！
(執筆時の最新バージョンは、2. 8. 0)



注)すでにM5ATOMでこのライブラリをインストールしてあれば、改めてインストールする必要は無い



一般社団法人全国専門学校情報教育協会

6



Hellow!!

M5Stick-Cでメッセージを発行してみよう

MQTT PUBLISH MESSAGES

システム構想

◇ボタン押下に同期してWEB経由でメッセージを発行する（固定メッセージ）

- ①. ボタンはA,B 2つあるので、各々押下されたボタンを通知するメッセージとする
- ②. パターンを変えてLEDを点滅する Aボタン→ゆっくり1回点滅
Bボタン→素早く2回点滅

◇M5Stick-Cは、WiFi接続機能を持っている → WiFi 経由でWEBサービスを利用する

- ・ 接続先アクセスポイントのSSIDとPASSWORDを調査する

SSID = "*****"

PASSWORD = "*****"

◇MQTT Broker として、broker.shfitr.io の公開MQTTサービスを利用する 接続情報は下記

- ・ 接続先URL = broker.shfitr.io
- ・ 接続ポート番号 = 1883（固定）
- ・ MQTTクライアント名 = "M5Stick-C"
- ・ MQTTユーザーID = "try"
- ・ MQTTパスワード = "try"

◇トピック名 → "qas/123" とする



Hellow!!

ソースコード 1/4 (M5C_MQTT_1)

◇#include #define 部分

```
#include <M5StickC.h>
#include <WiFi.h>
#include <PubSubClient.h>

#define WiFi_SSID " * * * * * " // 使用するWiFiのSSID
#define WiFi_PASS " * * * * * " // 使用するWiFiのPassword
#define MQTT_BROKER "broker.shiftr.io" // MQTT Broker 利用するブローカーのURL
#define MQTT_PORT 1883 // MQTT BROKER PORT 固定
#define MQTT_CLIENT_NAME "M5Stick-C" // MQTTブローカ接続時のクライアント名
#define MQTT_USER "try" // 公開クライアントユーザー名 決まっている
#define MQTT_PASS "try" // 同、パスワード 決まっている
#define MQTT_TOPIC "qas/123" // 事前に取り決めた TOPIC名

WiFiClient espClient; // WiFi接続クライアントオブジェクト
PubSubClient client(espClient); // MQTTクライアント接続オブジェクト
```

ソースコード 2/4 (M5C_MQTT_1)

◇WiFiアクセスポイント接続関数 と 初期化処理部分

```
void wfi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時・・・表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n--> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}

void setup() { // 初期化処理
  M5.begin(true, false, true); // SerialEnable , I2CEnable , DisplayEnable
  wfi_connect(); // WiFiアクセスポイント接続
  pinMode(M5_LED, OUTPUT); // LED ピン出力に設定
  digitalWrite(M5_LED, HIGH); // LED消灯
}
```

ソースコード 3/4 (M5C_MQTT_1)

◇通常処理部

```
void loop() { // 通常処理
  client.loop(); // MQTT接続状況更新
  while(!client.connected()){ // MQTT接続
    Serial.println("Mqtt Reconnecting"); // MQTT接続 試みているメッセージ
    if( client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS) ){
      Serial.println("Mqtt Connected"); // MQTT接続成功メッセージ
      break; // 接続が成功したので、while()ループから抜ける
    }
  }

  // ボタン処理部 ← M5ATOMと異なるボタン処理
}
```

ソースコード 4/4 (M5C_MQTT_1)

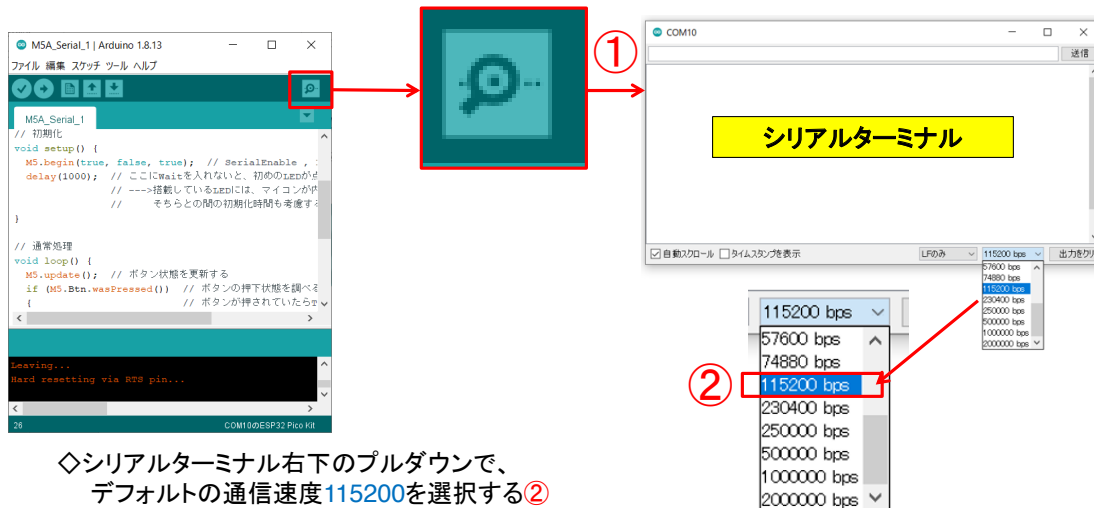
◇ボタン処理部

```
M5.update(); // M5Stick-Cボタン状況更新
if (M5.BtnA.wasPressed()){ // Aボタンが押されていたか?
  client.publish(MQTT_TOPIC, "Button A was pressed !!"); // ここでAボタン押下メッセージを送信
  Serial.println("Send message (Button A was pressed!!)"); // シリアルターミナルにも出力
  digitalWrite(M5_LED, LOW); // LED点灯
  delay(1000); // しばし待つ
  digitalWrite(M5_LED, HIGH); // LED消灯
} // 点滅は1回だけ

if (M5.BtnB.wasPressed()){ // Bボタンが押されていたか?
  client.publish(MQTT_TOPIC, "Button B was pressed !!"); // ここでBボタン押下メッセージを送信
  Serial.println("Send message (Button B was pressed!!)"); // シリアルターミナルにも出力
  digitalWrite(M5_LED, LOW); // LED点灯
  delay(200); // しばし待つ
  digitalWrite(M5_LED, HIGH); // LED消灯
  delay(200); // しばし待つ
  digitalWrite(M5_LED, LOW); // LED点灯
  delay(200); // しばし待つ
  digitalWrite(M5_LED, HIGH); // LED消灯
} // 点滅を2回繰り返す
```

シリアルターミナルの起動 (マイコンリセット時メッセージを見るためにシリアルモニタを起動する)

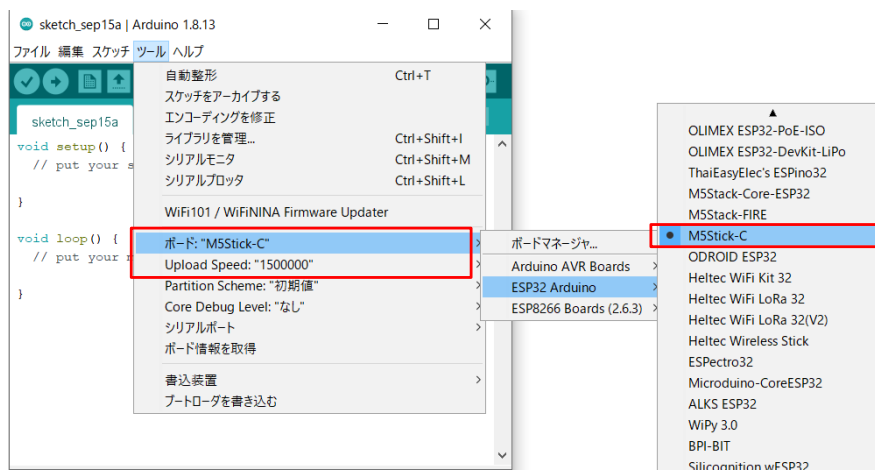
- ◇WindowsのデバイスマネージャでCOMポート番号を確認し、
ツール→シリアルポート で、COMポートを選択する
- ◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①



- ◇シリアルターミナル右下のプルダウンで、
デフォルトの通信速度115200を選択する②

マイコンボードの選択

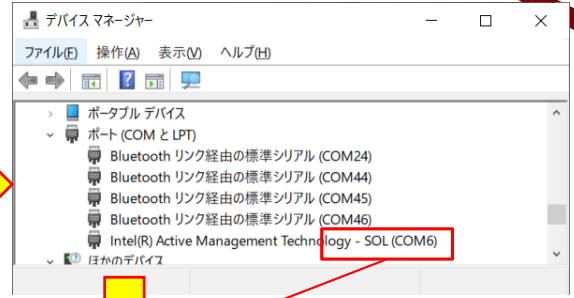
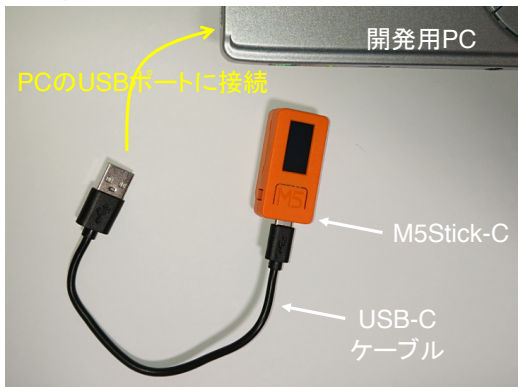
- ◇以下のように IDE で ツール → ボード → ...とたどり、M5Stick-C を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、M5Stick-Cを使用する場合は、必ずこの設定で行う



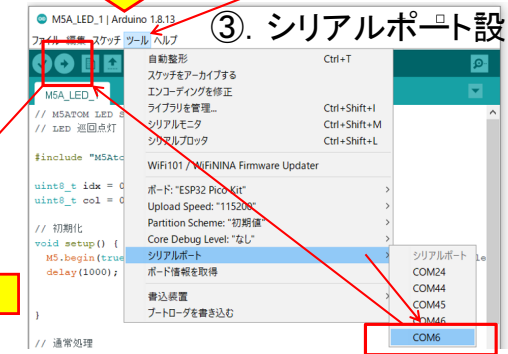
②. デバイスマネージャでCOMポート確認

マイコンをPCと接続

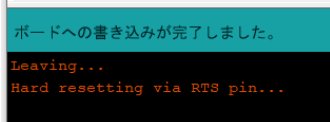
①. M5Stick-CをPCと接続



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



一般社団法人全国専門学校情報教育協会

15

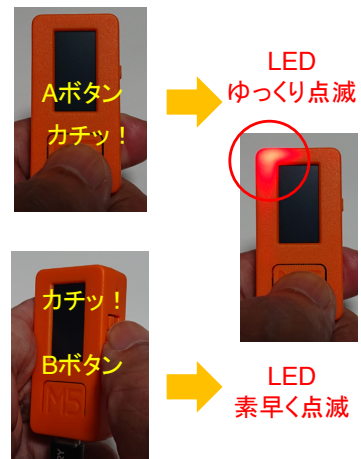
動作確認 1/4 メッセージ送信の確認

◇書き込みが完了すると M5Stick-CはResetされ、その際のメッセージがシリアルモニタに表示される

◇ボタンを操作する

- ①. Aボタンを押下する → LEDがゆっくり1度点滅する(右図)
- ②. 同時にシリアルモニタにメッセージが表示される(下図)
- ③. Bボタンを押下する → LEDが素早く2回点滅する(右図)
- ④. 同時にシリアルモニタにメッセージが表示される(下図)

```
M5stickC initializing...OK
WiFi Connecting..
---> Connected : 192.168.0.72
Mqtt Reconnecting
Mqtt Connected
Send message (Button A was pressed!!)
Send message (Button B was pressed!!)
```



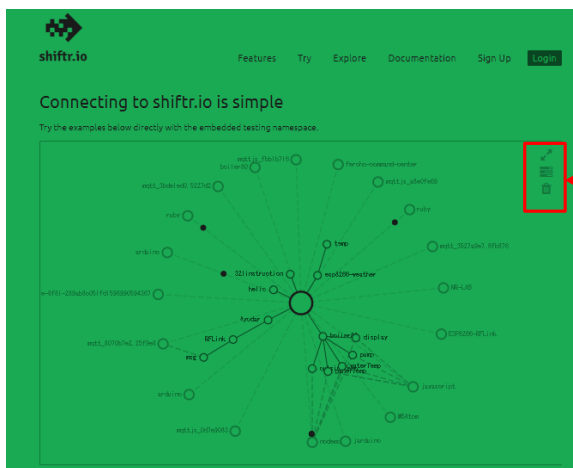
一般社団法人全国専門学校情報教育協会

16

動作確認 2/4 WEBで確認準備

① ブラウザで次のURLにアクセスすると図のページが開き、インターネット上のメッセージが見える

<https://legacy.shiftr.io/try>



② 画面右上の3つのアイコンを適宜クリックすれば表示を見やすく調整できる

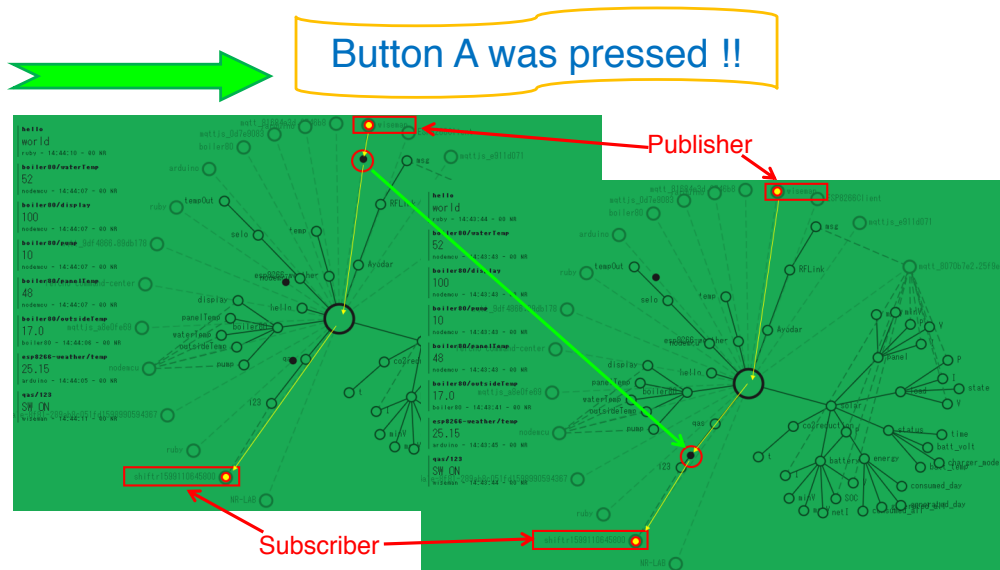


※ 動き回る黒丸● → メッセージの移動
 黒丸が出現する白丸 → メッセージ発行者
 中央の大きな○ → MQTT Broker
 メッセージが流れ着く先の白丸 → トピック名

一般社団法人全国専門学校情報教育協会

動作確認 3/4 WEBでメッセージの確認

③ ブラウザを注目しながらM5Stick-Cのボタンを押すとメッセージが流れる様子が見える



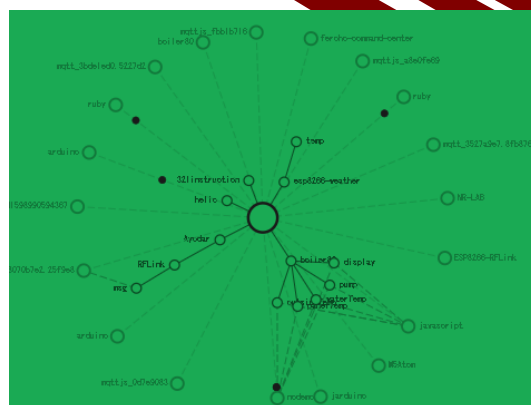
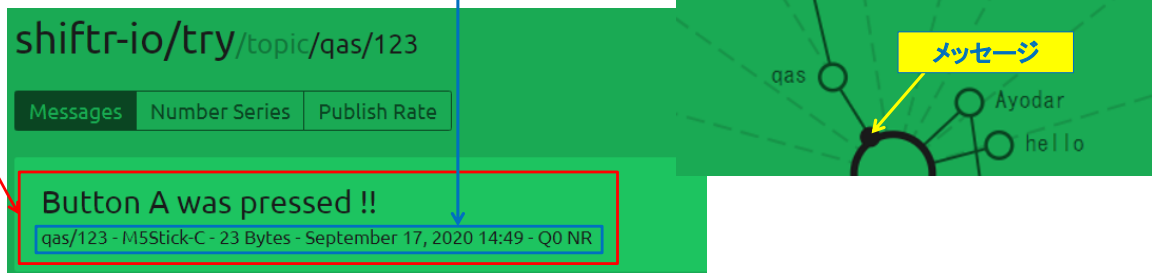
一般社団法人全国専門学校情報教育協会

動作確認 4/4 WEBでメッセージの確認

④ ブラウザを注目しながらM5Stick-Cのボタンを押すとメッセージが流れる様子が見える

- ⑤ トピック名の丸印をクリックすると下のページが開く
 ⑥ **ブラウザの更新ボタンを押下すると発行したメッセージを確認できる**

※発行者・トピック名・時刻 が記録されている



M5Stick-Cでメッセージを購読してみよう

MQTT SUBSCRIBE MESSAGES

システム構想

- ◇PCからメッセージを発行することができる
 - コマンドプロンプトから以下のコマンドを発行すれば、“Hello !!” が発行される
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "Hello !!"
※qas/123 と Hello !! はそれぞれ、トピック名とメッセージである
- ◇シリアル通信でLEDを制御したことを思い出して、その際作成したプログラムを利用する
- ◇メッセージを1文字として、その内容を以下のようにする
 - 0文字目:LED ON / OFF 指示 0 : 消灯 1 : 点灯 それ以外: 点滅
 - ※これは、シリアル通信の受信で実験したメッセージそのものだ！
- ◇例えば、LEDを点灯したければ以下のコマンドを実行すればよい
curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "1"

※ここでは curl コマンドの詳細は説明しない
このような便利なコマンドは、自身で調べて記録しておくべきだ(これはLinuxにもある)

ソースコード 1/6 (M5C_MQTT_2)

◇#include と #define 、通信関連オブジェクト部分

```
#include "M5StickC.h" // マイコンボードライブラリ
#include <WiFi.h> // WiFiライブラリ
#include <PubSubClient.h> // MQTTクライアントライブラリ

#define WiFi_SSID " * * * * * " // WiFiアクセスポイントSSID
#define WiFi_PASS " * * * * * " // 同 パスワード
#define MQTT_BROKER "broker.shiftr.io" // MQTTブローカーURL
#define MQTT_PORT 1883 // MQTT BROKER PORT
#define MQTT_CLIENT_NAME "M5Stick-C" // MQTTブローカー接続時のクライアント名
#define MQTT_USER "try" // 公開ユーザー名(固定)
#define MQTT_PASS "try" // 公開パスワード(固定)
#define MQTT_TOPIC "qas/123" // TOPIC名
#define MQTT_QOS 0 // Quality of Service(サービスの品質)

WiFiClient espClient; // WiFiクライアントオブジェクト
PubSubClient client(espClient); // MQTTクライアントコントロールオブジェクト
```

ソースコード 2/6 (M5C_MQTT_2)

◇グローバル変数 と WiFiアクセスポイント接続関数

```
char flg=0; // メッセージ受信フラグ 0:未受信 1:メッセージ到着
char msg[10]; // 受信メッセージ格納用

void wifi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時...表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n---> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}
```

ソースコード 3/6 (M5C_MQTT_2)

◇MQTTブローカー接続関数

```
void mqtt_connect() {
  // Loop until we're reconnected // 再接続するまでループするぞ！！
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection..."); // MQTT接続を行っているよ！メッセージ
    if (client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS)) { // MQTT 接続実行！
      Serial.println("connected"); // つながった！メッセージ
      client.subscribe(MQTT_TOPIC, MQTT_QOS); // メッセージ購読登録
      Serial.println("Subscribing!"); // 購読しているよ！メッセージ
    } else { // うまく行かなかった場合
      Serial.print("failed, rc="); // 失敗、その原因コードは！メッセージ
      Serial.print(client.state()); // クライアントの状態(原因コードが表示される)
      Serial.println(" try again in 5 seconds"); // 5秒待って試行する！メッセージ
      delay(5000); // 5秒待つ
    }
  }
}
```

ソースコード 4/6 (M5C_MQTT_2)

◇メッセージが発行された際に呼び出される(コールバック)関数

```
void callback(char* topic, byte* payload, unsigned int length) {
  int i;                                // メッセージバッファのインデックス

  Serial.print("Message arrived ["); // メッセージが到着したよ！
  Serial.print(topic);                // トピック名は！
  Serial.print("] ");
  for (i = 0; i < length; i++) {      // メッセージの内容は！
    msg[i] = (char)payload[i];
    Serial.print((char)payload[i]);    // 1文字ずつ表示する
  }
  Serial.println();                   // 改行しておく
  flg = 1;                             // メッセージ到着 loop()内でこのフラグを見て処理する
}
```

ソースコード 5/6 (M5C_MQTT_2)

◇LEDを全消灯する際に利用する関数 と 初期化

```
void setup() {                          // 初期化部
  M5.begin(true, false, true);          // M5Stick-C初期化
  wifi_connect();                       // WiFiアクセスポイント接続
  client.setCallback(callback);          // メッセージ購読時処理の登録
  pinMode(M5_LED, OUTPUT);              // LEDピン出力に設定
  digitalWrite(M5_LED, HIGH);           // LED消灯しておく
}
```

ソースコード 6/6 (M5C_MQTT_2)

◇通常処理全体

```
void loop() { // 通常処理
  int i;

  mqtt_connect(); // MQTT 接続が切れているといけないので、調べて必要なら再接続
  client.loop(); // MQTT接続状況更新 ... これがなかなか難しい
  if(flag==1){ // 受信メッセージあり?
    flag = 0; // メッセージ到着フラグクリア
    if(msg[0]=='0'){ // '0'か?
      digitalWrite(M5_LED, HIGH); // LED消灯
    }else if(msg[0]=='1'){ // '1'か?
      digitalWrite(M5_LED, LOW); // LED点灯
    }else{ // '0'でも'1'でもない! 不明コマンド
      Serial.println("Invalid Message!!"); // エラーメッセージ
      // ... .. 素早いLED点滅 3回
      return; // OSに戻る
    }
  }
}
```

一般社団法人全国専門学校情報教育協会

27

シリアルターミナルの起動 (マイコンリセット時メッセージを見るためにシリアルモニタを起動する)

◇WindowsのデバイスマネージャでCOMポート番号を確認し、
ツール→シリアルポート で、COMポートを選択する

◇IDEの右上にある 虫眼鏡マーク のボタンをクリックするとシリアルターミナルが起動する①

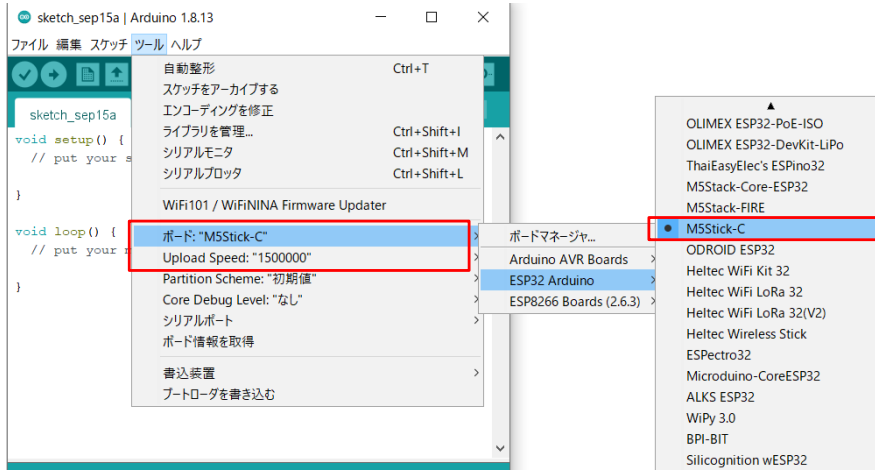
◇シリアルターミナル右下のプルダウンで、
デフォルトの通信速度115200を選択する②

一般社団法人全国専門学校情報教育協会

28

マイコンボードの選択

- ◇ 以下のように IDE で ツール → ボード → ...とたどり、M5Stick-C を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇ 同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇ 以後、M5Stick-Cを使用する場合は、必ずこの設定で行う

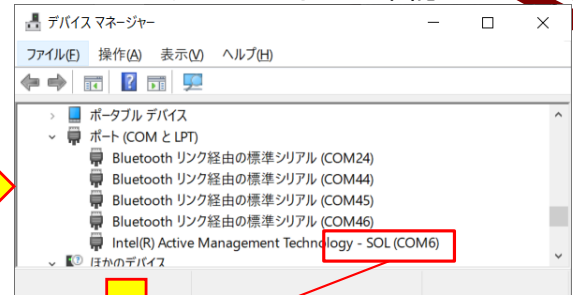


マイコンをPCと接続

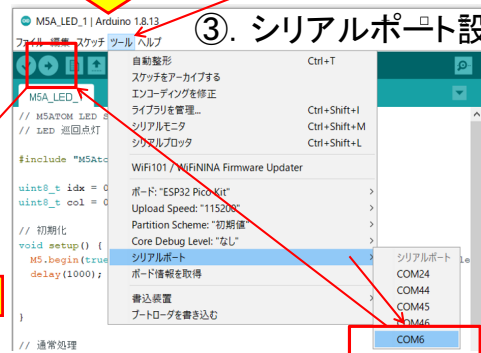
- ①. M5Stick-CをPCと接続



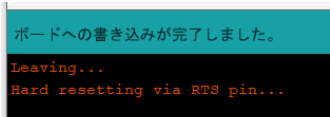
- ②. デバイスマネージャでCOMポート確認



- ③. シリアルポート設定



- ⑤. 書き込み完了



- ④. コンパイル



動作確認 1/2

◇書き込みが完了すると、M5Stick-Cはリセットされプログラムの実行が始まる

→ シリアルモニタに、その際のメッセージが表示される

◇Windowsコマンドプロンプトで以下のコマンドを実行する

- ①. C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "1"
→ M5Stick-CのLEDが点灯し、シリアルモニタにメッセージ到着が表示される
- ②. C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "0"
→ M5Stick-CのLEDが消灯し、シリアルモニタにメッセージ到着が表示される
- ③. C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "9"
→ M5Stick-CのLEDが素早く3回点滅し、シリアルモニタにエラーメッセージが表示される

```

コマンド プロンプト
C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "1"
OK
C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "0"
OK
C:¥Users¥user>curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "9"
OK
C:¥Users¥user>
    
```

```

WiFi Connecting.
---> Connected : 192.168.0.72
Attempting MQTT connection...connected
Subscribing!
Message arrived [qas/123] 1
Message arrived [qas/123] 0
Message arrived [qas/123] 9
Invalid Message!!
    
```

動作確認 2/2 WEBで確認

① ブラウザで次のURLにアクセスすると図のページが開き、インターネット上のメッセージが見える

<https://legacy.shiftr.io/try>



② 画面右上の3つのアイコンを適宜クリックすれば表示を見やすく調整できる



※ 動き回る黒丸● → メッセージの移動
 黒丸が出現する白丸 → メッセージ発行者
 中央の大きな○ → MQTT Broker
 メッセージが流れ着く先の白丸 → トピック名

2.12 フルカラー LCD の計測

M5Stick-C

LCD(液晶表示器)



一般社団法人全国専門学校情報教育協会

M5Stick-C LCDの簡単なテスト

◇M5Stick-Cには、80×160ドットのカラー液晶表示器が、強力な表示機能をシステムにもたらしめている

◇LCDは、SPI I/F(※)を持つST7735Sという液晶コントロールICで制御されている

※Serial Peripheral Interface: コンピュータ内部でデバイスを接続する目的のI/F

◇簡単な表示テストの例を図に示す

※左から、白・赤・緑・青・黒、文字列、矩形描画、塗りつぶし、円、塗りつぶし、三角形



◇以後、上記テストのソースコードを示し、詳細を解説する

問題 : 次頁以後の参考ソースコードを実行しなさい

一般社団法人全国専門学校情報教育協会

1

ソースコード 1/4 (M5C_Display_Test_1)

◇#include から初期化部

```
#include <M5StickC.h>

void setup() { // 初期化部
  M5.begin();

  // Lcd display 色
  M5.Lcd.fillScreen(WHITE); // 白
  delay(5000);
  M5.Lcd.fillScreen(RED); // 赤
  delay(5000);
  M5.Lcd.fillScreen(GREEN); // 緑
  delay(5000);
  M5.Lcd.fillScreen(BLUE); // 青
  delay(5000);
  M5.Lcd.fillScreen(BLACK); // 黒
  delay(5000);

  // 右に続く・・・
}

// text print 文字列
M5.Lcd.fillScreen(BLACK); //背景 黒
M5.Lcd.setCursor(0, 10); //カーソル位置
M5.Lcd.setTextColor(WHITE); //文字色
M5.Lcd.setTextSize(1); //文字サイズ
M5.Lcd.printf("Display Test!"); //文字列表示

delay(5000);

// draw graphic
M5.Lcd.drawRect(15, 55, 50, 50, BLUE); //青矩形
delay(5000);
M5.Lcd.fillRect(15, 55, 50, 50, BLUE); //塗りつぶし
delay(5000);
M5.Lcd.drawCircle(40, 80, 30, RED); //赤円
delay(5000);
M5.Lcd.fillCircle(40, 80, 30, RED); //塗つぶし
delay(5000);
}
```

一般社団法人全国専門学校情報教育協会

2

ソースコード 2/2 (M5C_Display_Test_1)

◇通常処理部

```
void loop(){ // 通常処理部

  //rand draw
  M5.Lcd.fillTriangle(
    random(M5.Lcd.width()-1), // 三角形の3頂点座標を指定
    random(M5.Lcd.height()-1), //
    random(M5.Lcd.width()-1), //
    random(M5.Lcd.height()-1), //
    random(M5.Lcd.width()-1), //
    random(M5.Lcd.height()-1), //
    random(0xfffe) ); // 塗の色指定
}
```

問題 : 上記の各頂点のパラメータで -1 しているのは何故か？

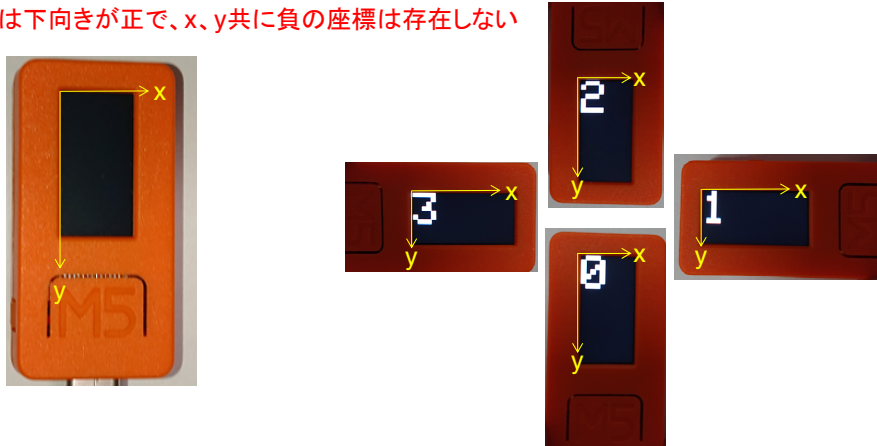
一般社団法人全国専門学校情報教育協会

3

LCDの座標系

- ◇下図左のように見れば、幅80ドット×高さ160ドットのLCDになっている
- ◇図形描画や点を打つ場合(グラフ描画など)は、座標系を考えた描画プログラムが必要となる
- ◇座標系の原点は、LCDの左上(x,y) = (0,0) となっているが、座標系の向きを90度ずつ回転できる
→ M5.Lcd.setRotation(R) R=0,1,2,3 LCD左上が原点、水平方向=X軸、垂直方向=Y軸 となる

※y方向は下向きが正で、x、y共に負の座標は存在しない



※上図は、いずれも原点に文字サイズ7で描画した (M5C_Display_Test_2)

色

- ◇ライブラリで色コードが定義され、図形輪郭線・塗・文字の各色は、**定義名**で指定できる
- ◇RGB各色を各々5bit, 6bit, 5bitで表現(RGB565モード)した色を16bitコードに変換する関数も使える
M5.Lcd.color565(r, g, b)
- ◇文字の色は M5.Lcd.setTextColor(色名) で指定できる ※図形の色は、描画の関数で指定できる

#define	BLACK	0x0000	/* 0, 0, 0 */
#define	NAVY	0x000F	/* 0, 0, 128 */
#define	DARKGREEN	0x03E0	/* 0, 128, 0 */
#define	DARKCYAN	0x03EF	/* 0, 128, 128 */
#define	MAROON	0x7800	/* 128, 0, 0 */
#define	PURPLE	0x780F	/* 128, 0, 128 */
#define	OLIVE	0x7BE0	/* 128, 128, 0 */
#define	LIGHTGREY	0xC618	/* 192, 192, 192 */
#define	DARKGREY	0x7BEF	/* 128, 128, 128 */
#define	BLUE	0x001F	/* 0, 0, 255 */
#define	GREEN	0x07E0	/* 0, 255, 0 */
#define	CYAN	0x07FF	/* 0, 255, 255 */
#define	RED	0xF800	/* 255, 0, 0 */
#define	MAGENTA	0xF81F	/* 255, 0, 255 */
#define	YELLOW	0xFFE0	/* 255, 255, 0 */
#define	WHITE	0xFFFF	/* 255, 255, 255 */
#define	ORANGE	0xFD20	/* 255, 165, 0 */
#define	GREENYELLOW	0xAFE5	/* 173, 255, 47 */
#define	PINK	0xF81F	//問題: ピンクの値は10進数でいくつか?

文字描画

◇文字描画は、次の関数が準備されている

・カーソル指定

- ①. M5.Lcd.print(string) → 文字列描画
- ②. M5.Lcd.println(string) → 改行付き
- ③. M5.Lcd.printf("%d", i) → 書式指定文字列描画

・座標指定

- ④. M5.Lcd.drawString(string, x, y) → 座標指定
- ⑤. M5.Lcd.drawString(string, x, y, font) → 座標・フォント指定

※文字列・文字描画関数は他にもある

◇文字サイズ、文字色を変えて M5.Lcd.println(string) を用いた描画例(下図) [M5C_Display_Test_3](#) がある
上から順に文字サイズ1,2,3,4 各々色の名称を描画したもの

◇グラフの描画では、ドット単位の位置指定が重要なので、
座標指定文字列描画が威力を発揮する



6

画面塗りつぶし

◇画面全体の塗りつぶしには M5.Lcd.fillScreen(color) 関数が使える

図は左から、WHITE、RED、GREEN、BLUE、BLACK で塗りつぶした様子

※参考ソースコード [M5C_Display_Test_1](#)



7

図形描画

◇図形描画関数は以下のものがある
※color は色定義名、または色コード16bit)

- ①. M5.Lcd.drawPixel(x, y, color) → 点描画 x、yは点の座標
- ②. M5.Lcd.drawLine(x0, y0, x1, y1, color) → 線描画 x0、y0は開始位置座標、x1、y1は終了位置座標
- ③. M5.Lcd.drawCircle(x, y, r, color) → 円描画 x、yは円の中心座標、rは半径
- ④. M5.Lcd.drawRect(x, y, w, h, color) → 矩形描画 x、yは左上の座標、w、hは矩形の幅、高さ
- ⑤. M5.Lcd.drawTriangle(x0, y0, x1, y1, x2, y2, color) → 三角形描画 xn,ynは3つの頂点の座標
- ⑥. M5.Lcd.fillCircle(x, y, r, color) → 円塗りつぶし描画 x、yは中心座標、rは半径
- ⑦. M5.Lcd.fillRect(x, y, w, h, color) → 矩形塗りつぶし x、yは左上の座標、w、hは矩形の幅、高さ
- ⑧. M5.Lcd.fillTriangle(x0, y0, x1, y1, x2, y2, color) → 三角形描画 xn,ynは3つの頂点の座標

※参考ソースコード [M5C_Display_Test_1](#)



文字フォント

◇文字フォントは、次の関数で指定できる

- ①. M5.Lcd.setTextFont(font) → あらかじめフォントを指定する
- ②. M5.Lcd.drawString(string, x, y, font) → 描画時に指定する

font = 1 → Adafruit 8ピクセルASCIIフォント (defaultらしい)
2 → 16ピクセルASCIIフォント → 1のフォントを2倍の大きさにしたもの
4 → 26ピクセルASCIIフォント
6 → 26ピクセル数字フォント } サイズが大きすぎて実用的ではない

※参考ソースコード [M5C_Display_Test_4](#)



サンプル

◇内蔵加速度センサによるGの計測値をプロットしている様子



一般社団法人全国専門学校情報教育協会

10

M5Stick-C

Servo Motor



一般社団法人全国専門学校情報教育協会

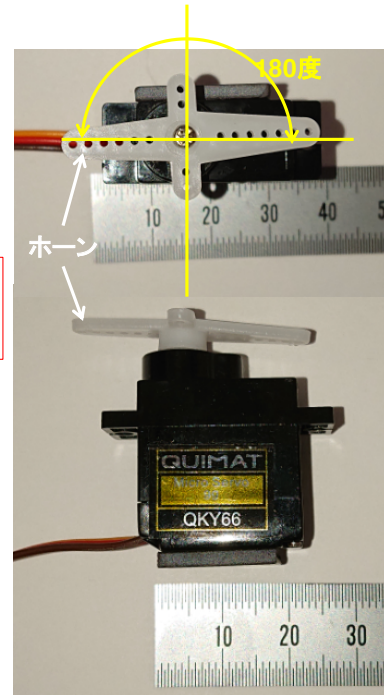
サーボモータ

- ◇サーボモータは、主軸の回転角度が制御できる
- ◇右図のサーボモータは主軸が最大180度回転する
- ◇主軸にホーン(白いパーツ)を取り付けて使う
- ◇位置決めに使われる
- ◇データシートの一部を下記に示す

Specifications:

Weight: 9g
 Dimension: 23×12.2×29mm
Stall torque: 1.8kg/cm(4.8v)
 Gear type: POM gear set
 Operating speed: 0.12 sec/60degree(4.8v)
 Operating voltage: 4.8v
 Temperature range: 0℃_ 55℃
 Dead band width: 1us
 Power Supply: Through External Adapter
 servo wire length: 25 cm
 Servo Plug: JR (Fits JR and Futaba)

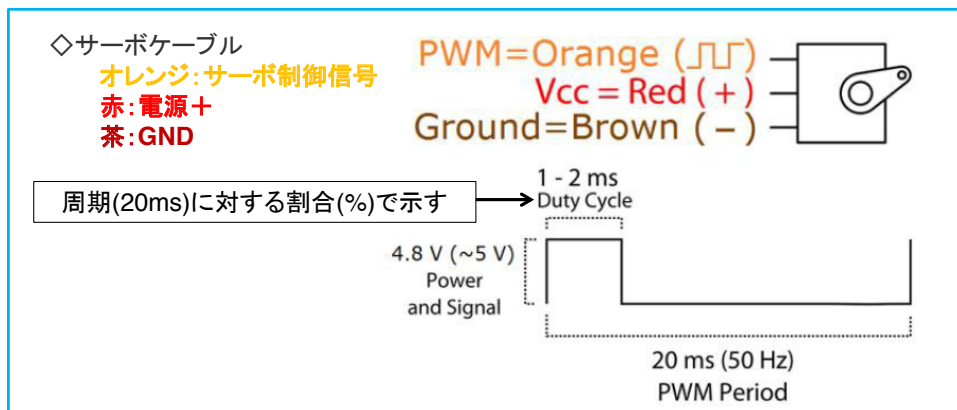
※主軸中心から1cmの所に糸を取り付けて、1.8kg未満の加重を引き上げる力がある



サーボモータ制御信号

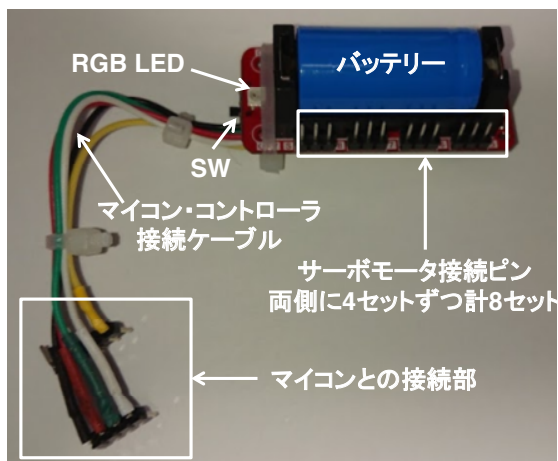
- ◇PWM周期20ms(50Hz) → Pulse Width Modulation
- ◇Datasheetは【"0"(1msパルス)で中央、"90"(2msパルス)で中央、"-90"(~1msパルス)で一番左】としているが、これは誤りで、**実際は上から見て【0度で一番右に、180度で一番左に位置決めされる】**

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.



サーボモータ電源とコントローラ

- ◇サーボモータ電源はマイコンとは別の電源を確保する → マイコンの安定動作のため
- ◇電源を搭載したコントローラの利用 → サーボモータ8個別々に制御可能 ※フルカラーLED付属
- ◇マイコンとの接続は、別途作成したケーブル利用



- ◇サーボコントローラに角度とサーボ CH をセットすれば、搭載している IC が PWM 制御パルスを出力する → マイコン側は、CHと角度を指定するだけ
- ◇RGB LEDは、RGB各々の明るさを8bitで指定する
- ◇サーボCHおよびRGB各々のレジスタはI2Cアドレスが決まっている
- ◇サーボモータ電源は、バッテリーから供給される
- ◇バッテリー充電は、マイコンとの接続ケーブルを外して M5Stick-Cを接続してUSBケーブルで充電するかまたは、専用充電器を使用する

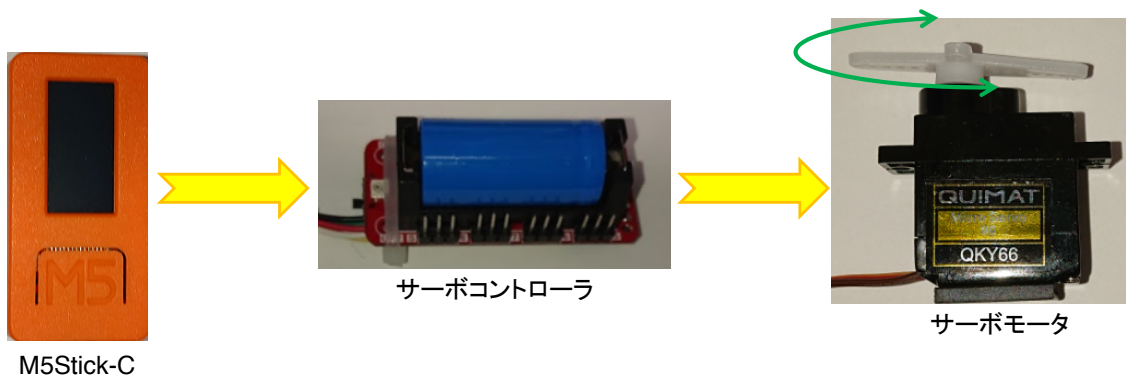
サーボモータを制御する

SERVO MOTOR CONTROL



目論見 → 細かい位置決め技術の習得

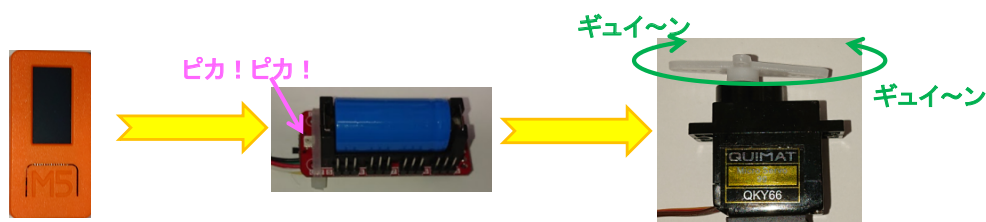
- ◇M5Stick-Cからコントローラを介してPWMパルスを出力し、サーボモータを左右に回転運動させる
 - 細かい位置決めに応用できる
 - ラジコン飛行機では、回転運動をラダー(方向舵)やエルロン(昇降舵)の制御に
 - 自動車では、ステアリングの制御に利用



システム構想

◇M5Stick-Cによるサーボモータ制御 → 0度から180度の往復回転運動を繰り返す

- ①. IIC_servo ライブラリ (servo.cpp + servo.h) が提供されている
 - ※ソースコードと同一フォルダで同時にコンパイル → ライブラリインストールの必要なし
- ②. 次の関数が使える
 - ・ IIC_Servo_Init() → コントローラの初期化
 - ・ Servo_angle_set(HC, 角度) → サーボモータ角度制御 CHIは1~8、角度は0~180で指定
 - ・ RGB_set(R,G,B) → LED点灯...R,G,Bの明るさは、それぞれ0~255で指定



ソースコード 1/2 (M5C_Servo_1)

◇初期化処理部

```
#include "M5StickC.h" // M5Stick-C ライブラリ
#include "IIC_servo.h" // サーボモータライブラリ

void setup(){ // 初期化
  M5.begin(true, true, false); //serial, I2C, LED
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init(); //sda:0 scl:26 Servo Controller用にIICを設定
                    // IICというのは、マイコンとデバイスを通信で結ぶI/F
}
```

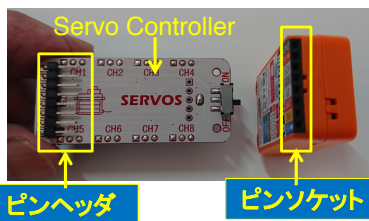
ソースコード 2/2 (M5C_Servo_1)

◇通常処理部

```
void loop() { // 通常処理
  RGB_set(128,0,0); // red ピカ！
  delay(200); // しばし待つ
  RGB_set(0,128,0); // green ピカ！
  delay(200); // しばし待つ
  RGB_set(0,0,128); // blue ピカ！
  delay(200); // しばし待つ
  RGB_set(0,0,0); //RGB 消灯

  for(int i=1;i<9;i++){ // 全CH(どのCHに接続してもサーボは動く)を制御
    Servo_angle_set(i,0); // 0度
  }
  delay(1000); // しばし待つ
  for(int i=1;i<9;i++){ // 全CH(どのCHに接続してもサーボは動く)を制御
    Servo_angle_set(i,180); // 180度
  }
  delay(1000); // しばし待つ
}
```

サーボコントローラとM5Stick-Cの接続



- ◇サーボコントローラとM5Stick-Cを以下の手順で接続する
- ①. コントローラの裏側にあるピンヘッダを確認
※別基板などが釣りつけられてる場合は、慎重に取り外す
 - ②. M5Stick-Cのピンソケットを確認

- ③. サーボコントローラのピンヘッダとM5Stick-Cのピンソケットが合うように、配置する



- ④. サーボコントローラのピンヘッダをM5Stick-Cのピンソケットに差し込む

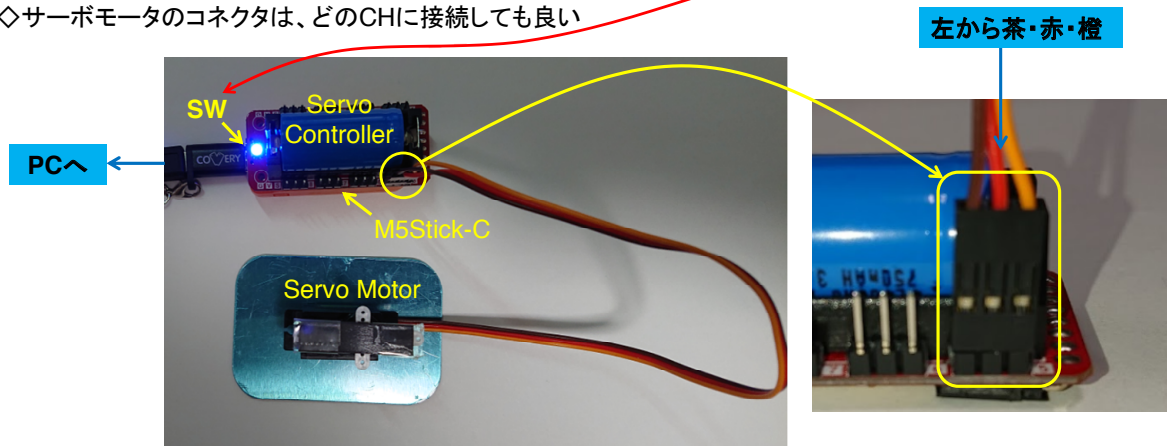


※この状態でM5Stick-Cに通電すると、サーボコントローラのバッテリーにも充電できる 9

一般社団法人全国専門学校情報教育協会

マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、サーボコントローラを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い

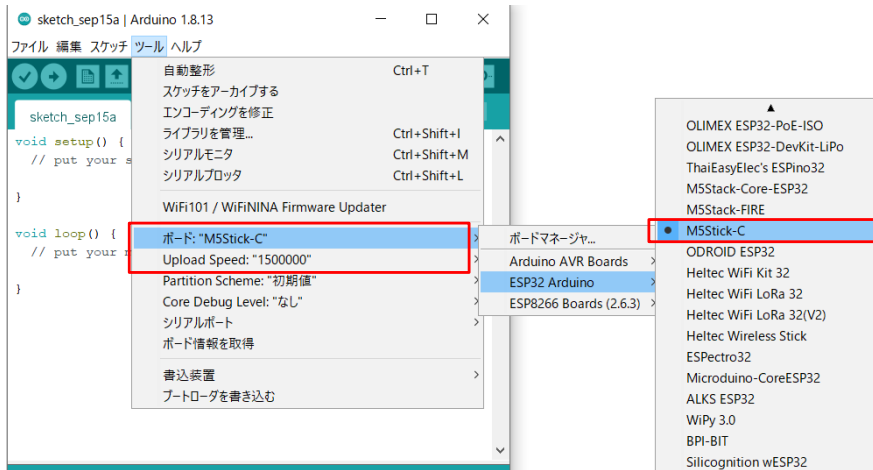


一般社団法人全国専門学校情報教育協会

10

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

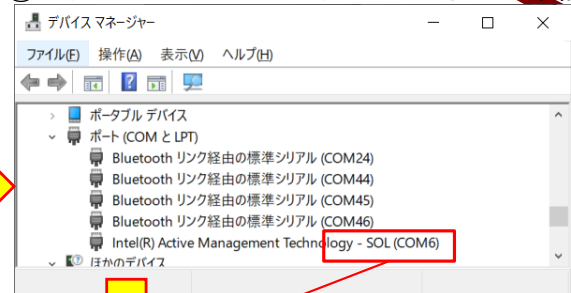


マイコンをPCと接続

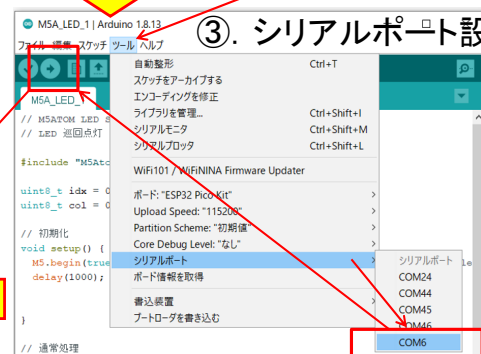
①. M5Stick-CをPCと接続



②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

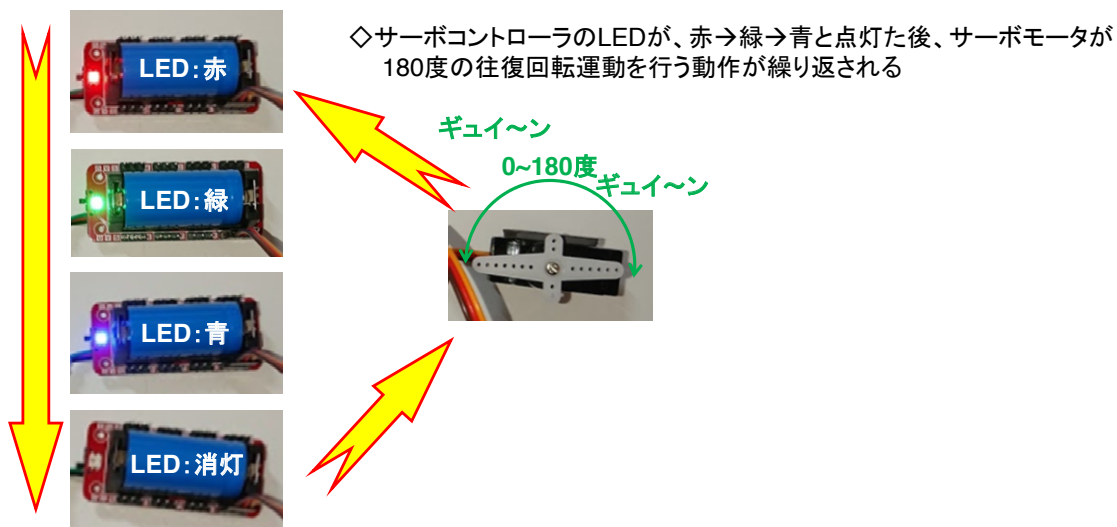


④. コンパイル



動作確認

◇書き込み完了と同時に、新しいプログラムがスタートしている



MQTTでサーボモータを制御する

SERVO MOTOR CONTROL BY MQTT MESSAGE



システム構想

- ◇MQTTメッセージでLEDを制御したことを思い出そう
- ◇PCからメッセージを発行することができる
 - コマンドプロンプトから以下のコマンドを発行すれば、“Hello !!” が発行される
 - `curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "Hello !!"`
 - ※`qas/123` と `Hello !!` はそれぞれ、トピック名とメッセージである
- ◇MQTTでLEDを制御したことを思い出して、その際作成したプログラムを利用する
- ◇メッセージを3文字として、その内容でサーボモータの回転角度を指定する
 - 例 : 0度 → 000
 - 90度 → 090
 - 180度 → 180
- ◇例えば、サーボモータを45度の位置に回転する場合は以下のコマンドを実行すればよい
- `curl -X POST "http://try.try@broker.shiftr.io/qas/123" -d "045"`

ソースコード 1/6 (M5C_Servo_2)

◇`#include` と `#define` 、通信関連オブジェクト部分

```
#include "M5StickC.h"           // マイコンボードライブラリ
#include <WiFi.h>                // WiFiライブラリ
#include <PubSubClient.h>        // MQTTクライアントライブラリ
#include "IIC_servo.h"          // サーボ制御ライブラリ
#define WiFi_SSID " * * * * * " // WiFiアクセスポイントSSID
#define WiFi_PASS " * * * * * " // 同 パスワード
#define MQTT_BROKER "broker.shiftr.io" // MQTTブローカーURL
#define MQTT_PORT 1883          // MQTT BROKER PORT
#define MQTT_CLIENT_NAME "M5Stick-C" // MQTTブローカー接続時のクライアント名
#define MQTT_USER "try"         // 公開ユーザー名(固定)
#define MQTT_PASS "try"         // 公開パスワード(固定)
#define MQTT_TOPIC "qas/123"   // TOPIC名
#define MQTT_QOS 0              // Quality of Service(サービスの品質)
WiFiClient espClient;          // WiFiコントロールオブジェクト
PubSubClient client(espClient); // MQTTクライアントコントロールオブジェクト
```

ソースコード 2/6 (M5C_Servo_2)

◇グローバル変数 と WiFiアクセスポイント接続関数

```
char flg=0; // メッセージ受信フラグ 0:未受信 1:メッセージ到着
char msg[10]; // 受信メッセージ格納用(少し大きめに確保した)

void wifi_connect(void){ // WiFiアクセスポイントへの接続
  Serial.print("WiFi Connecting"); // 接続を試みているメッセージ
  WiFi.begin(WiFi_SSID, WiFi_PASS); // 接続開始
  while (WiFi.status() != WL_CONNECTED) { // アクセスポイント接続待ち
    Serial.print("."); // Wait時...表示
    delay(1000); // しばし待つ
  }
  Serial.print("\n---> Connected : "); // 接続成功メッセージ
  Serial.println(WiFi.localIP()); // 自機のIPアドレス表示
  client.setServer(MQTT_BROKER, MQTT_PORT); // MQTTブローカへの接続設定
}
```

ソースコード 3/6 (M5C_Servo_2)

◇MQTTブローカー接続関数

```
void mqtt_connect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect(MQTT_CLIENT_NAME, MQTT_USER, MQTT_PASS)) { // MQTT接続実行！
      Serial.println("connected"); // つながったメッセージ
      client.subscribe(MQTT_TOPIC, MQTT_QOS); // メッセージ購読登録
      Serial.println("Subscribing!"); // 購読しているよメッセージ
    } else { // うまく行かなかった場合
      Serial.print("failed, rc="); // 失敗原因コードの表示
      Serial.print(client.state()); // 同上
      Serial.println(" try again in 5 seconds"); // 5秒待ちますメッセージ
      // Wait 5 seconds before retrying
      delay(5000); // しばし待つ
    }
  }
}
```

ソースコード 4/6 (M5C_Servo_2)

◇メッセージが発行された際に呼び出される(コールバック)関数

```
void callback(char* topic, byte* payload, unsigned int length) {
  int i;

  Serial.print("Message arrived ["); // メッセージが到着したよメッセージ
  Serial.print(topic);              // 念のためトピック名表示
  Serial.print("]");                // 括弧閉じ
  for (i = 0; i < length; i++) {    // (メッセージ文字長文)メッセージ取り出し
    msg[i] = (char)payload[i];
    Serial.print((char)payload[i]); // メッセージ表示
  }
  Serial.println();                 // 改行を付けて！！
  flg = 1;                          // メッセージ到着フラグをONする loop()内でこのフラグを見て処理する
}
```

ソースコード 5/6 (M5C_Servo_2)

◇初期化関数

```
void setup() { // 初期化部
  M5.begin(true, true, true); // SerialEnable , I2CEnable , DisplayEnable
  wifi_connect();           // WiFiアクセスポイント接続
  client.setCallback(callback); // メッセージ発行時に呼び出される関数を登録する
  mqtt_connect();          // MQTT 再接続・・・(接続が切れていたら)

  Serial.printf("%n MQTT_Servo Control-2%n"); // シリアルポートにメッセージ出力
  IIC_Servo_Init(); //sda:0 scl:26 Servo Controller用にIICを設定
}
```


ソースコード 6/6 (M5C_Servo_2)

◇通常処理全体 エラー処理とLED制御は次で説明)

```
void loop() { // 通常処理部
  int i; // メッセージバッファインデックス
  int a; // サーボモータ角度

  mqtt_connect(); // MQTT 接続が切れているといけないので、調べて必要なら再接続
  client.loop(); // MQTT接続状況更新 ... これがなかなか難しい

  if(flag==1){ // 受信メッセージあり?
    flag = 0; // フラグクリア

    a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // サーボモータ回転角度計算

    // エラーチェックブロック
    // LED点滅ブロック
    // サーボモータコントローラ出力 ブロック
  }
}
```

ソースコード 6/6 (M5C_Servo_2)

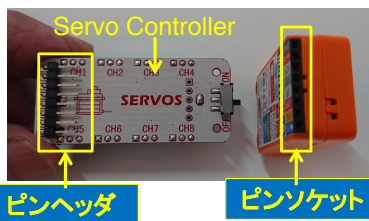
◇エラーチェックブロック と LED制御ブロック

```
//エラーチェックブロック
if(a<0 || a>180){ // 角度チェック
  Serial.println("Over Rotation Angle!!");
  return; // エラーメッセージを送信してOSに戻る
}

// LED点滅ブロック
RGB_set(128,0,0); // red ピカ!
delay(200);
RGB_set(0,128,0); // green ピカ!
delay(200);
RGB_set(0,0,128); // blue ピカ!
delay(200);
RGB_set(0,0,0); //RGB 消灯
```

```
// サーボモータ制御ブロック
// 全CHIに角度設定(どのピンに接続してもサーボが動く)
for(int i=1;i<9;i++){
  Servo_angle_set(i,a); // a=指定角度
}
```

サーボコントローラとM5Stick-Cの接続



- ◇サーボコントローラとM5Stick-Cを以下の手順で接続する
- ①. コントローラの裏側にあるピンヘッダを確認
※別基板などが釣りつけられてる場合は、慎重に取り外す
 - ②. M5Stick-Cのピンソケットを確認

- ③. サーボコントローラのピンヘッダとM5Stick-Cのピンソケットが合うように、配置する



- ④. サーボコントローラのピンヘッダをM5Stick-Cのピンソケットに差し込む

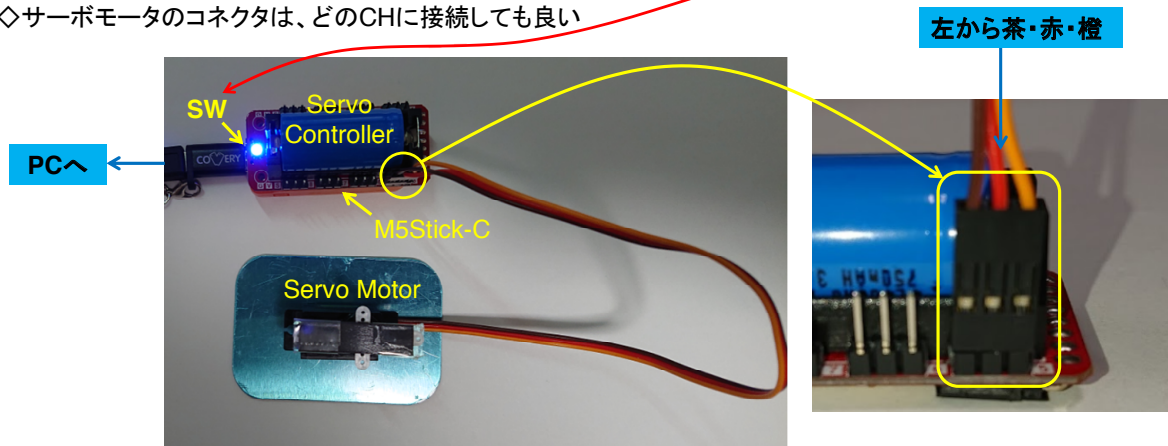


※この状態でM5Stick-Cに通電すると、サーボコントローラのバッテリーにも充電できる23

一般社団法人全国専門学校情報教育協会

マイコン書込み前のセッティング

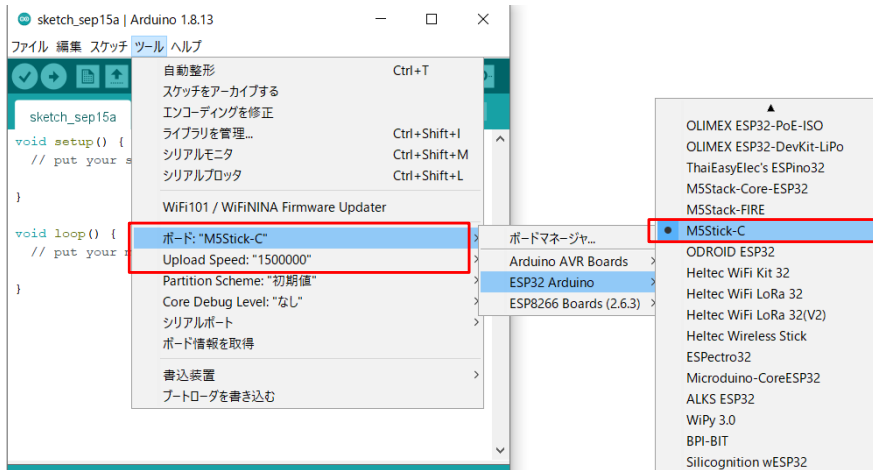
- ◇今回はマイコン単独ではなく、サーボコントローラを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い



一般社団法人全国専門学校情報教育協会

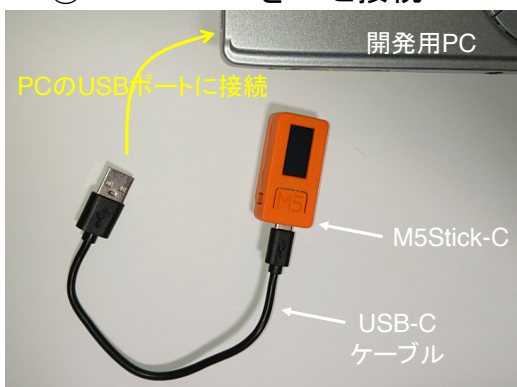
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、M5Stick-C を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、M5Stick-Cを使用する場合は、必ずこの設定で行う

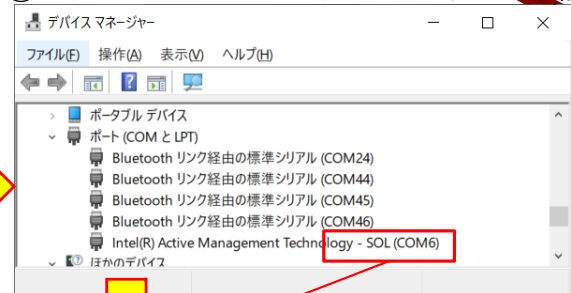


マイコンをPCと接続

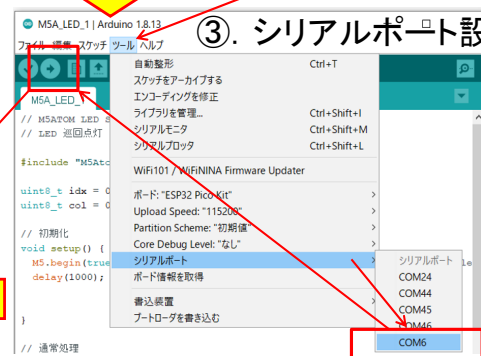
①. M5Stick-CをPCと接続



②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



動作確認

◇下図のように、コマンドプロンプトで curl コマンドを用いてメッセージを発行する

```

C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "000"
OK
C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "090"
OK
C:\Users\user> curl -X POST "http://try:try@broker.shiftr.io/qas/123" -d "180"
OK
C:\Users\user>
  
```

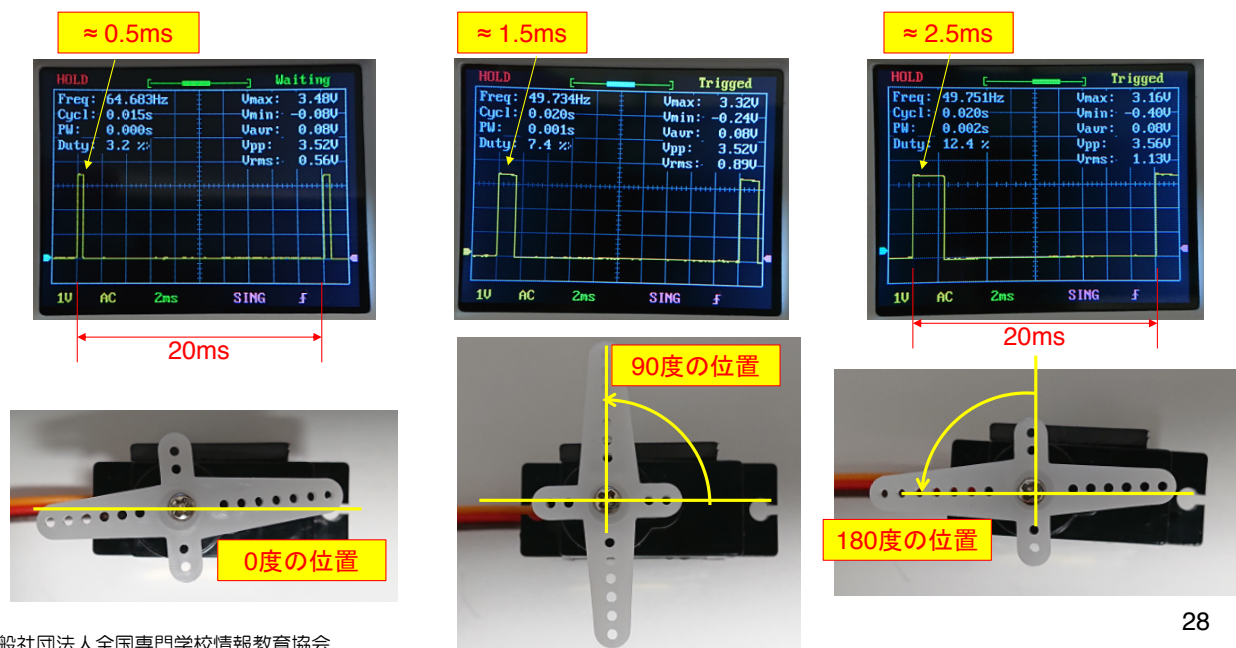
◇シリアルモニタを起動しておく、M5ATOMが受信したメッセージが表示され、その後LEDが赤→緑→青と点灯し、サーボモータが180度の往復回転運動をする

```

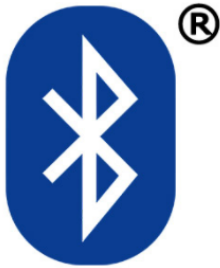
Message arrived [qas/123] 000
Message arrived [qas/123] 090
Message arrived [qas/123] 180
  
```



パルスと回転の様子（簡易オシロスコープ）



M5Stick-C



Bluetooth



一般社団法人全国専門学校情報教育協会

Bluetooth Class

Table 1: ESP32-PICO-D4 Specifications

Categories	Items	Specifications
Certification	Bluetooth certification	BQB
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps) A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth V4.2 BR/EDR and BLE specification NZIE receiver with -97 dBm sensitivity
	Radio	Class-1, class-2 and class-3 transmitter ← Class1~3 AFH
	Audio	CVSD and SBC

	最大出力	通信距離
Class 1	100mW	約100m
Class 2	2.5mW	約10m
Class 3	1mW	約1m

注) 通信距離は目安

一般社団法人全国専門学校情報教育協会

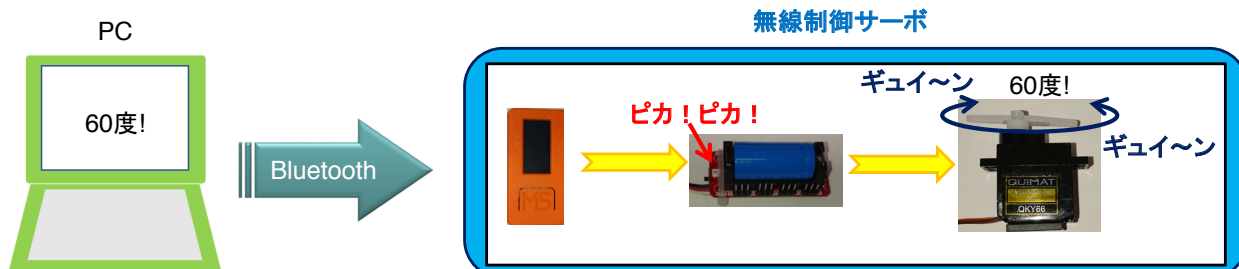
1

目論見 → 無線制御サーボシステム

◇PCからBluetooth経由でサーボモータの位置決め角度を制御する

→ 無線化サーボ

◇無線化 → IoTエンジニア必須スキル → 実験室的にも欠かせない

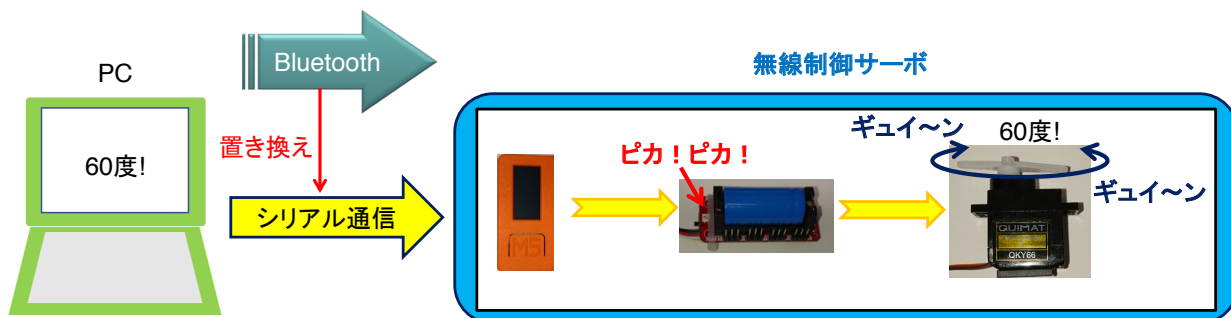


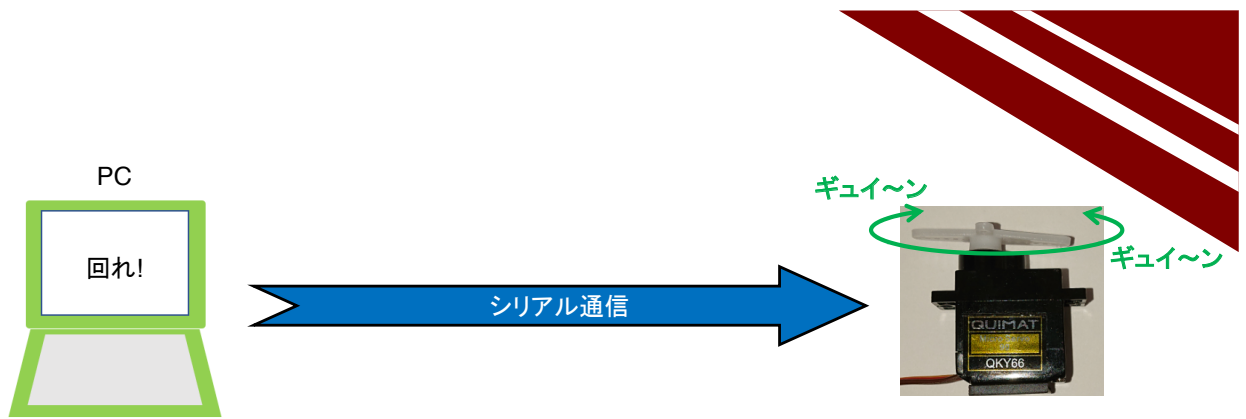
システム構想

◇2段階で開発する

①. MQTT経由サーボモータ制御システムをシリアル通信経由に改造する
→ PC から M5Stick-C にシリアル通信でサーボモータ角度を与える

②. ①をBluetooth経由に改造する
→ BluetoothSerial ライブラリが使える
→ BluetoothSerialの使い方はシリアル通信と同じ





シリアル通信でサーボモータを制御する

SERVO MOTOR CONTROL BY SERIAL COMMUNICATION

ソースコード 1/3 (M5C_Servo_Serial_3)

◇初期化処理部

```
#include "M5StickC.h" // M5Stick-C用ライブラリ
#include "IIC_servo.h" // サーボモータライブラリ

int i; // メッセージバッファインデックス
char msg[10]; // 受信メッセージ格納用バッファ

void setup(){ // 初期化部
  M5.begin(true, true, true); //serial, I2C, LED
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init(); //sda:0 scl:26 Servo Controller用にIICを設定(初めに開放したポート)
  i=0; // メッセージバッファインデックス初期化
}
```

ソースコード 2/3 (M5C_Servo_Serial_3)

◇通常処理部

```
void loop() { // 通常処理
  int a; // サーボモータ角度
  char c; // 受信データ 1文字分

  if(Serial.available()){ // 受信データあり?
    c = Serial.read(); // 1文字 Read
    msg[i++] = c; // 受信した文字を格納
    if(c == '\n'){ // 改行ならば電文の終端
      Serial.print("Received message ---> "); // 受信した角度表示
      Serial.print(msg); // 同上 角度部
      i=0; // メッセージバッファインデックス初期化
      a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // 角度計算

      //エラーチェック部
      //LED点滅部
      //サーボ制御部
    }
  }
}
```

6

一般社団法人全国専門学校情報教育協会

ソースコード 3/3 (M5C_Servo_Serial_3)

◇エラーチェック部

```
if(a<0 || a>180){ // 角度チェック 0° から180° の範囲外はエラー
  Serial.println("Over Rotation Angle!!");
  return; // エラーメッセージを送信してOSに戻る
}
```

◇LED点滅部

```
RGB_set(128,0,0); // red ピカ!
delay(200); // しばし待つ
RGB_set(0,128,0); // green ピカ!
delay(200); // しばし待つ
RGB_set(0,0,128); // blue ピカ!
delay(200); // しばし待つ
RGB_set(0,0,0); //RGB 消灯
```

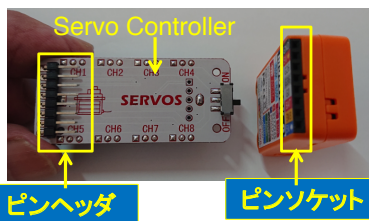
◇サーボ制御部

```
for(int j=1;j<9;j++){ // 全CH(どのピンに接続してもサーボが動く)
  Servo_angle_set( j, a ); // a=指定角度
}
```

7

一般社団法人全国専門学校情報教育協会

サーボコントローラとM5Stick-Cの接続



- ◇サーボコントローラとM5Stick-Cを以下の手順で接続する
- ①. コントローラの裏側にあるピンヘッダを確認
※別基板などが釣りつけられてる場合は、慎重に取り外す
 - ②. M5Stick-Cのピンソケットを確認

- ③. サーボコントローラのピンヘッダとM5Stick-Cのピンソケットが合うように、配置する

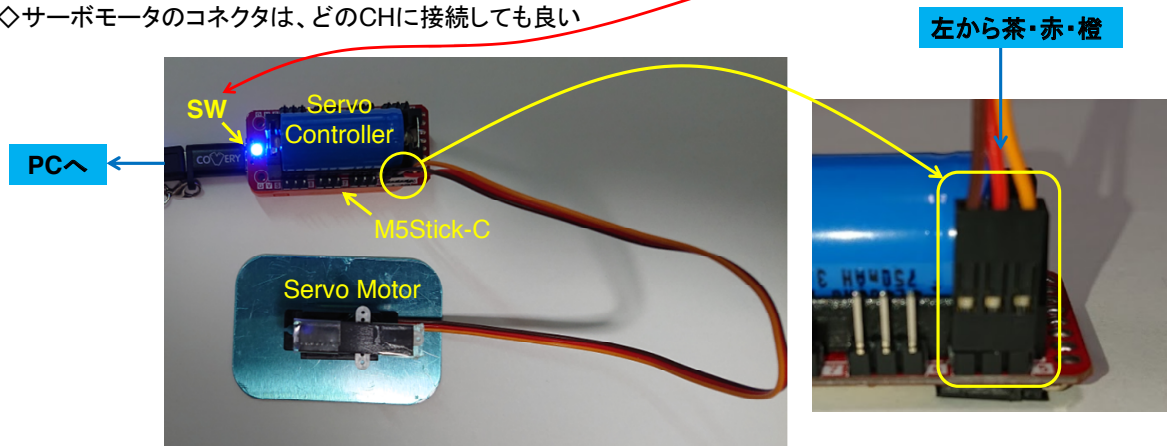


- ④. サーボコントローラのピンヘッダをM5Stick-Cのピンソケットに差し込む



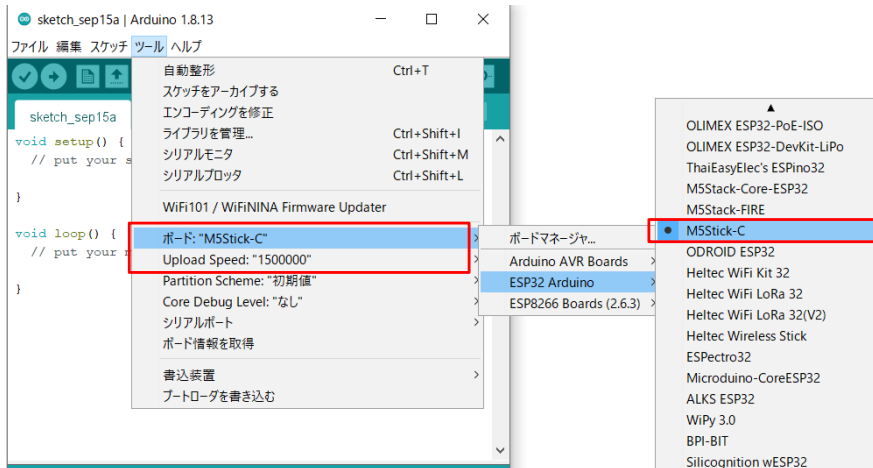
マイコン書込み前のセッティング

- ◇今回はマイコン単独ではなく、サーボコントローラを接続する必要があるので、プログラム書込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHIに接続しても良い



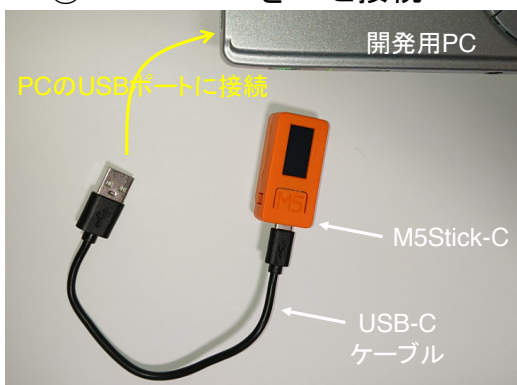
マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

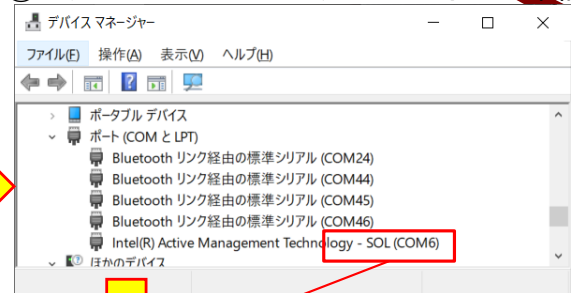


マイコンをPCと接続

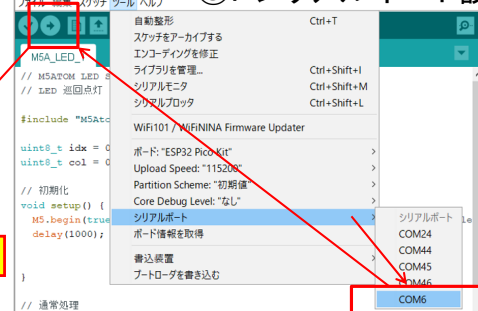
①. M5Stick-CをPCと接続



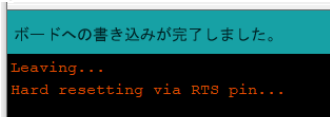
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



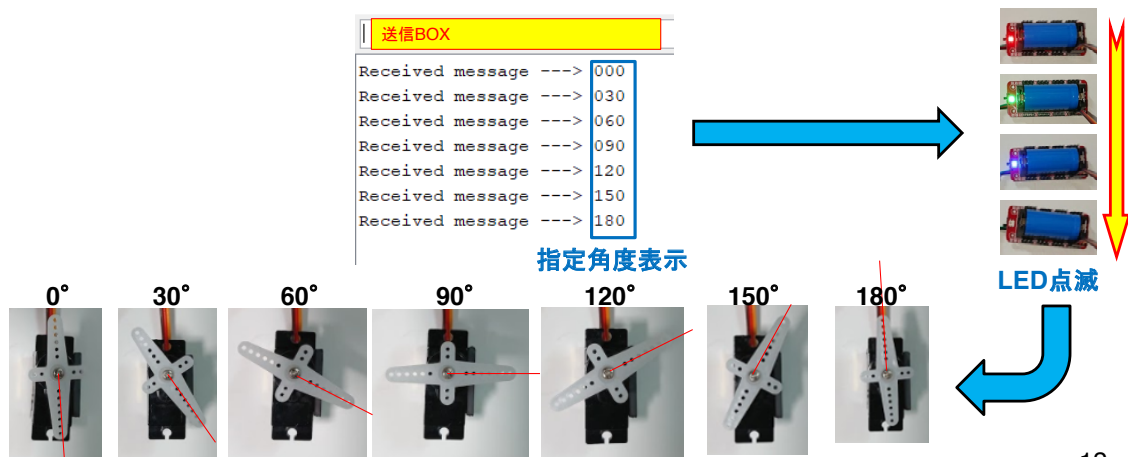
④. コンパイル



動作確認

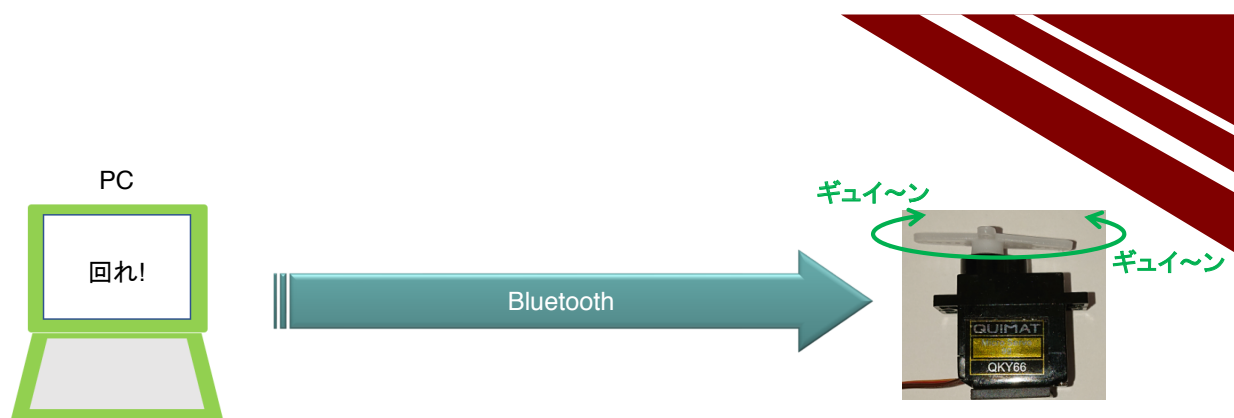
◇シリアルモニタを起動して(IDE右上の虫眼鏡マーク)0度から30度刻みで180度まで位置決めをしてみよう!

- ①. シリアルモニタの**送信BOX**に指定角度(0度なら000、90度なら090、120度なら120)を入力して**ENTER** または **送信ボタン**をクリックする
- ②. シリアルモニタに受信したメッセージ(指定角度)が表示された後、サーボコントローラのLEDが、**赤→緑→青**と点灯した後、ホーンが指定角度に位置決めされる様子が確認できる



一般社団法人全国専門学校情報教育協会

12



Bluetooth でサーボモータを制御する

SERVO MOTOR CONTROL BY BLUETOOTH

一般社団法人全国専門学校情報教育協会

13

システム構想

- ◇シリアル通信を Bluetooth で行うプロトコル SPP (Serial Port Protocol) がある
- ◇いま動作確認を終えたシステムの通信を **Serial通信からBluetooth通信に置き換える**
- ◇M5Stick-Cの開発環境には SPP用 **BluetoothSerial ライブラリ**が含まれている

※IDEで スケッチ → ライブラリをインクルード とたどれば BluetoothSerial を確認できる

- ◇先に動作確認を終えたプログラムを流用すれば、容易に実現できる

→ ソースコードを少し変更するだけで実現できそう！！

ソースコード 1/3 (M5C_Servo_Bluetooth_Serial_4)

◇初期化処理部

```
#include "M5StickC.h"
#include "IIC_servo.h"
#include <BluetoothSerial.h>
BluetoothSerial bts;
int i;
char msg[10];

void setup(){
  M5.begin(true, true, true);
  Serial.printf("%n IIC_servo%n");
  IIC_Servo_Init();
  i=0;
  bts.begin(" * * * * "); // PC側でペアリングするデバイス名称 ※各自ユニークな名称にすること！！
  delay(1000);
  Serial.print(" Servo Bluetooth Serial Control-4%n"); // Serial Terminal Message
  bts.print(" Servo Bluetooth Serial Control-4%n"); // Bluetooth Terminal Message
}
```

※青字部分を追加・変更する

ソースコード 2/3 (M5C_Servo_Bluetooth_Serial_4)

◇通常処理部

```
void loop() { // 通常処理部
  int a; // サーボモータ角度
  char c; // 受信データ 1文字分
  if(bts.available()){ // Bluetoothに受信データあり?
    c = bts.read(); // Bluetooth から1文字 Read
    msg[i++] = c; // 受信した文字を格納
    if(c == '\n'){ // 改行ならば電文の終端
      Serial.print("Received message ---> "); // 受信した角度表示
      Serial.print(msg); // 同上 角度部
      bts.print("Received message ---> "); // Bluetooth Terminal にも表示
      bts.print(msg);
      i=0; // メッセージバッファインデックス初期化
      a = (msg[0]-'0')*100 +(msg[1]-'0')*10 +(msg[2]-'0'); // 角度計算(文字コード→数値)
      //エラーチェック部
      //LED点滅部
      //サーボ制御部
    }
  }
}
```

16

一般社団法人全国専門学校情報教育協会

ソースコード 3/3 (M5C_Servo_Bluetooth_Serial_4)

◇エラーチェック部

```
if(a<0 || a>180){ // 角度チェック 0° から180° の範囲外はエラー
  bts.println("Over Rotation Angle!!"); // Bluetooth にエラーメッセージ出力
  return; // エラーメッセージを送信してOSIに戻る
}
```

◇LED点滅部

```
RGB_set(128,0,0); // red ピカ！
delay(200); // しばし待つ
RGB_set(0,128,0); // green ピカ！
delay(200); // しばし待つ
RGB_set(0,0,128); // blue ピカ！
delay(200); // しばし待つ
RGB_set(0,0,0); //RGB 消灯
```

◇サーボ制御部

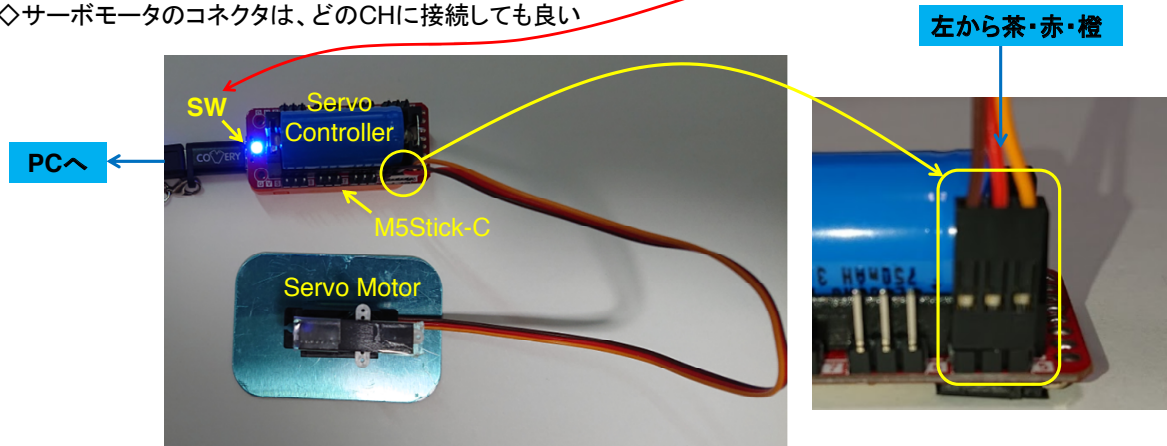
```
for(int j=1;j<9;j++){ // 全CH(どのピンに接続してもサーボが動く)
  Servo_angle_set( j, a ); // a=指定角度
}
```

17

一般社団法人全国専門学校情報教育協会

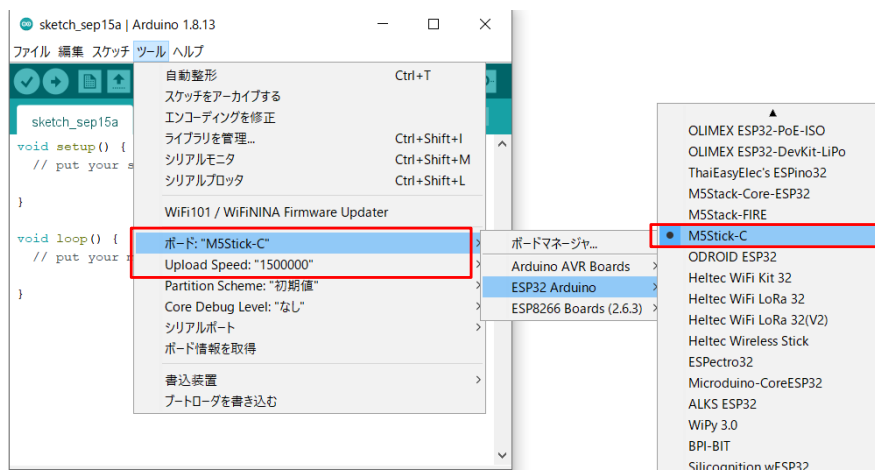
マイコン書き込み前のセッティング

- ◇今回はマイコン単独ではなく、サーボコントローラを接続する必要があるので、プログラム書き込み前に、全デバイスを図のように接続し、**コントローラのSWをON**にしておく
- ◇サーボモータのコネクタは、どのCHに接続しても良い



マイコンボードの選択

- ◇以下のように IDE で **ツール → ボード → ...とたどり、M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-Cを使用する場合は、必ずこの設定で行う**

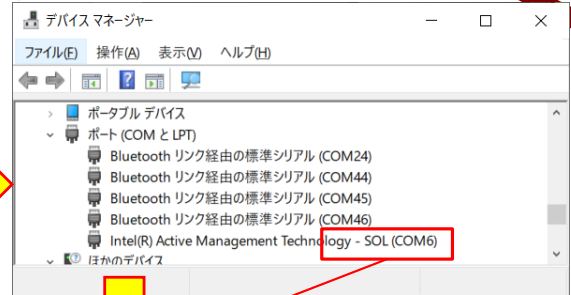


マイコンをPCと接続

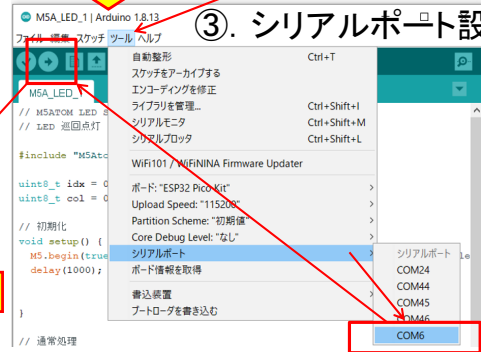
①. M5Stick-CをPCと接続



②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



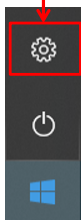
一般社団法人全国専門学校情報教育協会

20

動作確認 1/3 Bluetooth Device の追加(ペアリング)

◇書き込みが完了したら、数秒待って、以下の手順でデバイスのペアリングを行う

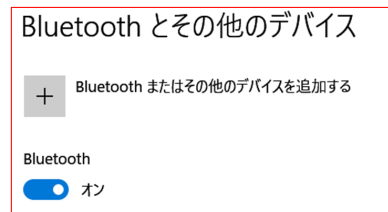
①スタート→設定



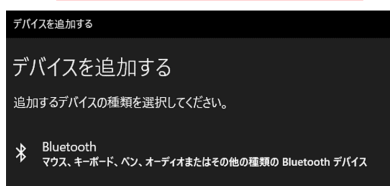
②デバイス



③Bluetoothとその他のデバイス



④デバイスを追加する



⑤プログラム指定のデバイス名選択



⑤ペアリング完了



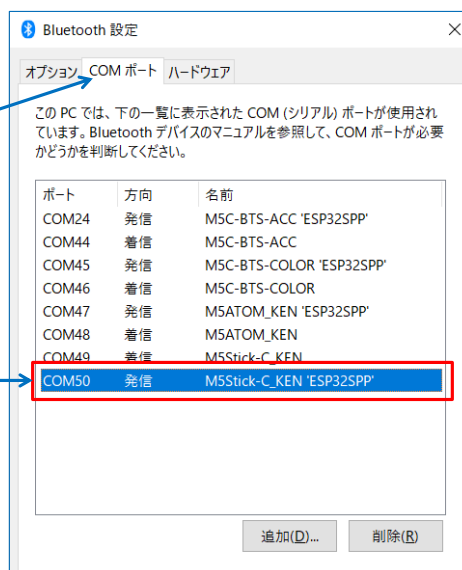
一般社団法人全国専門学校情報教育協会

21

動作確認 2/3 Bluetooth Serial の発信用ポート番号を確認

- ◇スタートボタンから
設定→デバイス→その他のBluetoothオプション
とたどり、【COMポート】タブを選択する
- ◇ペアリングしたBluetoothデバイス名の方向が【発信】と
なっているCOMポート番号を確認しメモする
- ◇メモしたCOMポート番号をIDEのシリアルポートで
選択する

発信のCOMポート

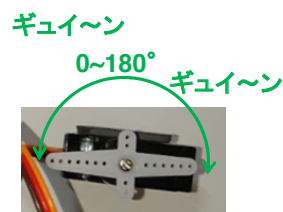


動作確認 3/3

- ◇シリアルターミナルを開く(IDE右上の虫眼鏡マーク)
- ◇送信BOXに 000 ~ 180 の角度を入力してENTER キー(または送信ボタン)を押下する



- ◇シリアルモニタに M5Sick-Cが受信したメッセージが表示され、その後LEDが赤→緑→青と点灯し、サーボモータが指定角度に位置決めされる



2.13 加速度センサの活用

M5Stick-C

加速度センサ (Accelerometer)



一般社団法人全国専門学校情報教育協会

加速度

- ◇地球上では、重力が我々の体や物体を地球中心に向かって引いている
- ◇物体が落下するとき、重力で落下速度が徐々に早くなる
- ◇重力による単位時間当たりの落下速度の変化を表すもの

→ 【重力加速度】

- ◇重力加速度 = 9.80665 m/s^2 → これを 1G(大文字)と表す ※G値などということもある

※地球上どこでも等しいわけではない 概ね 9.8 m/s^2 (1秒間落ちると 9.8 m/s 速度が大きくなる)

- ◇M5Stick-Cには、3軸加速度センサ(MPU6886)が内蔵されている

※3軸 → X,Y,Z

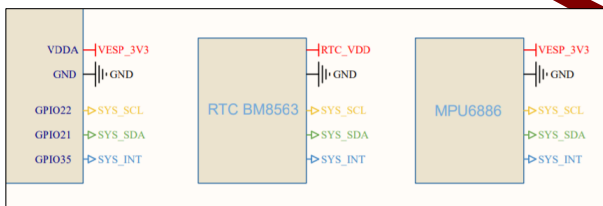
- ◇MPU6886は、ジャイロセンサも内蔵しているので、6軸センサとも言われる

一般社団法人全国専門学校情報教育協会

1

MPU6886 Data Sheet

- ◇M5Stick-Cの回路図(右図)では、MPU6886は I2C I/Fでマイコンと接続されている
- ◇データシートでは、SPI I/Fでも接続が可能である (下図)

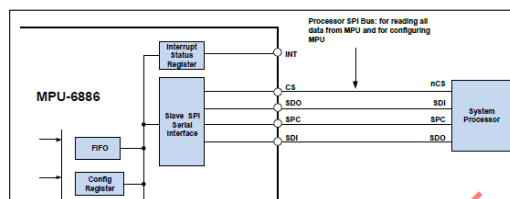
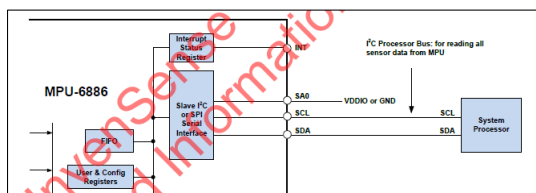


6.1 I²C AND SPI SERIAL INTERFACES

The internal registers and memory of the MPU-6886 can be accessed using either I²C at 400 kHz or SPI at 10 MHz. SPI operates in four-wire mode.

PIN NUMBER	PIN NAME	PIN DESCRIPTION
23	SCL / SPC	I ² C serial clock (SCL); SPI serial clock (SPC)
24	SDA / SDI	I ² C serial data (SDA); SPI serial data input (SDI)
9	SA0 / SDO	I ² C Slave Address LSB (SA0); SPI serial data output (SDO)
22	CS	Chip select (0 = SPI mode)

Table 13. Serial Interface



16bitADCを持つ3軸加速度計と信号調整

4.8 THREE-AXIS MEMS ACCELEROMETER WITH 16-BIT ADCS AND SIGNAL CONDITIONING

The MPU-6886's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-6886's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0g on the X- and Y-axes and +1g on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to $\pm 2g$, $\pm 4g$, $\pm 8g$, or $\pm 16g$.

4.8 16bit ADCを持つ3軸加速度計と信号調整

MPU-6886の3軸加速度計は、各軸について別々の試験質量を用いている。特定の軸に沿った加速度は、対応する試験質量での変化を含み、容量センサが変化の差を検出している。MPU-6886の構造は、加速度計の構造変化に対する感受性を、熱ドリフトと同様に減少させている。**デバイスが平面に置かれている場合、X軸とY軸では0Gを、そしてZ軸では+1Gを測定する。**この加速度計のスケール係数は、工場で校正されており、供給電圧とは表面上独立している。各センサはデジタル出力のための専用 Σ - Δ ADCを持っている。デジタル出力のフルスケール範囲は **$\pm 2g$ 、 $\pm 4g$ 、 $\pm 8g$ 、または $\pm 16g$ に調整できる。**

MPU6886のコマンド

◇データシートには、MPU6886の初期化やデータ読み出し用レジスタが列挙されており、ライブラリではこれらにアクセスして、加速度の値を取得している

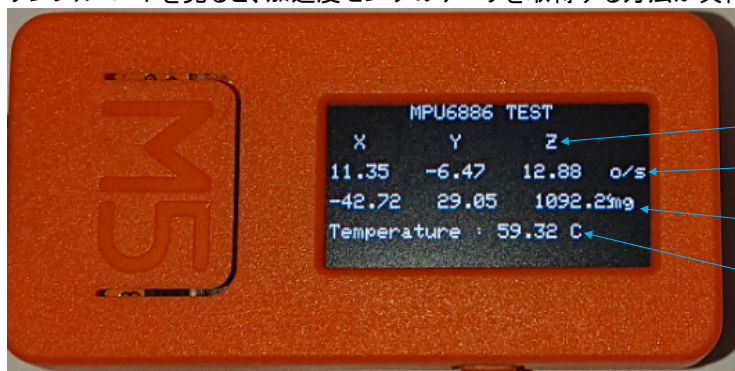
ADDR (HEX)	ADDR (DEC.)	REGISTER NAME	SERIAL I/F	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
37	55	INT_PIN_CFG	READ/WRITE	INT_LEVEL	INT_OPEN	LATCH_INT_EN	INT_RD_CLEAR	FSYNC_INT_LEVEL	FSYNC_INT_MODE_EN	-	-
38	56	INT_ENABLE	READ/WRITE	WOM_X_INT_EN	WOM_Y_INT_EN	WOM_Z_INT_EN	FIFO_OVERFLOW_EN	-	GDRIVE_INT_EN	-	DATA_RDY_INT_EN
39	57	FIFO_WM_INT_STATUS	READ to CLEAR	-	FIFO_WM_INT	-	-	-	-	-	-
3A	58	INT_STATUS	READ to CLEAR	WOM_X_INT	WOM_Y_INT	WOM_Z_INT	FIFO_OVERFLOW_INT	-	GDRIVE_INT	-	DATA_RDY_INT
3B	59	ACCEL_XOUT_H	READ	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	READ	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	READ	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	READ	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	READ	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	READ	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	READ	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	READ	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	READ	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	READ	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	READ	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	READ	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	READ	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	READ	GYRO_ZOUT[7:0]							

一般社団法人全国専門学校情報教育協会

4

サンプルプログラム

- ◇IDEには、M5Stick-C用の様々なサンプルコードが含まれている
- ◇IDEのメニューで ファイル → スケッチの例 → M5StickC → Basics → MPU6886 を選択すると6軸センサMPU6886のサンプルスケッチが開く
- ◇これをそのままコンパイルして実行してみよう(下図)
 - LCDにX,Y,Z軸ごとにジャイロセンサの検出値と加速度センサの検出値、センサチップ内温度が表示される
- ◇サンプルコードを見ると、加速度センサのデータを取得する方法が具体的に分かる



軸
 ジャイロのスコープ検出値※
 G値
 チップ内温度

※単位にo/sとあるのは、deg/sのdegをOで表現しているらしい

一般社団法人全国専門学校情報教育協会

5

サンプルソースコード 1/2 (MPU6886)

◇冒頭から初期化部

※6軸センサ用ライブラリはM5StickCのライブラリに含まれているので、
初期化はM5.MPU6886.Init()をCallするだけでよい

```
#include <M5StickC.h>

float accX = 0; // 加速度・ジャイロ各3軸用
float accY = 0;
float accZ = 0;
float gyroX = 0;
float gyroY = 0;
float gyroZ = 0;
float temp = 0; // チップ内部温度
```

```
void setup() {
  // put your setup code here, to run once:
  M5.begin();
  M5.Lcd.setRotation(3);
  M5.Lcd.fillScreen(BLACK);
  M5.Lcd.setTextSize(1);
  M5.Lcd.setCursor(40, 0);
  M5.Lcd.println("MPU6886 TEST");
  M5.Lcd.setCursor(0, 15);
  M5.Lcd.println(" X   Y   Z");
  M5.MPU6886.Init(); // 6軸センサの初期化
}
```

サンプルソースコード 2/2 (MPU6886)

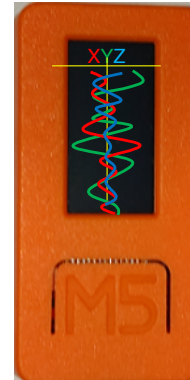
◇通常処理部

```
void loop() {
  M5.MPU6886.getGyroData(&gyroX,&gyroY,&gyroZ); // ジャイロセンサデータ取得
  M5.MPU6886.getAccelData(&accX,&accY,&accZ); // 加速度センサデータ取得
  M5.MPU6886.getTempData(&temp); // 温度センサデータ取得

  M5.Lcd.setCursor(0, 30);
  M5.Lcd.printf("%.2f %.2f %.2f ", gyroX, gyroY, gyroZ);
  M5.Lcd.setCursor(140, 30);
  M5.Lcd.print("o/s");
  M5.Lcd.setCursor(0, 45);
  M5.Lcd.printf("%.2f %.2f %.2f ", accX * 1000, accY * 1000, accZ * 1000);
  M5.Lcd.setCursor(140, 45);
  M5.Lcd.print("mg");
  M5.Lcd.setCursor(0, 60);
  M5.Lcd.printf("Temperature : %.2f C", temp);
  delay(100);
}
```

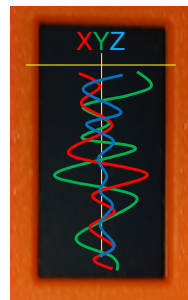
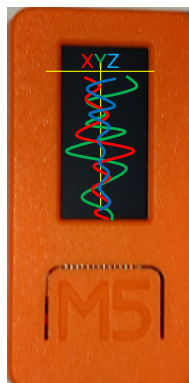
加速度センサ

ACCERELOMETER



目論見 3軸振動計

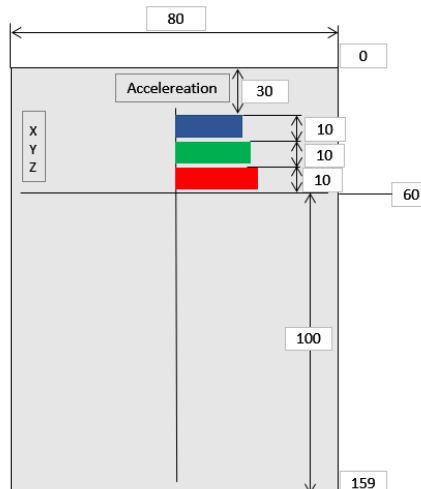
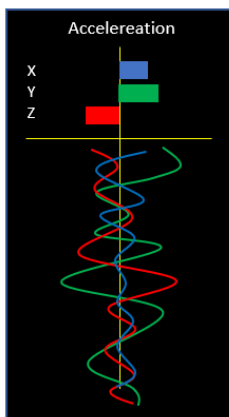
- ◇今回は、3軸のG値を計測してリアルタイムにグラフ化する
- ◇M5Stick-Cには、マグネットが内蔵されていて金属部分に取り付けられる
→ 3軸のGを測れば、どちら方向にどれだけ振動しているのかが見える！ → 【見える化】
- ◇LCDは小さいが、その表現力に期待して、図のようなグラフ表示を試みる



システム構想

◇画面設計：以下のような画面にする

- ①. グラフ部分は棒グラフ部(中央の基準線の右側: $G \geq 0$ 、左側 $C < 0$)と折れ線グラフ部にする
- ②. 折れ線グラフ部は、最新のデータを水平基準軸下にプロットする
- ③. 折れ線グラフ部が下に流れるようにするために描画データを3軸分保存する必要がある
3軸 × 100ドットの配列を確保する



一般社団法人全国専門学校情報教育協会

10

ソースコード 1/9 (M5C_ACC_GRAPH_1)

◇冒頭から変数宣言

```
#include <M5StickC.h>
```

```
#define maxG 30 // グラフにする際の1G当たりのドット数
```

```
float accX = 0.0F; // X方向加速度
```

```
float accY = 0.0F; // Y方向加速度
```

```
float accZ = 0.0F; // Z方向加速度
```

```
int AX; // グラフのためのX軸加速度
```

```
int AY; // グラフのためのY軸加速度
```

```
int AZ; // グラフのためのZ軸加速度
```

```
int i=0; // 描画データ配列用インデックス
```

```
int acv[3][100]; // 3軸の加速度データ100プロット分
```

```
float gyroX = 0.0F; // ジャイロX
```

```
float gyroY = 0.0F; // ジャイロY
```

```
float gyroZ = 0.0F; // ジャイロZ
```

```
float pitch = 0.0F; // ピッチ
```

```
float roll = 0.0F; // ロール
```

```
float yaw = 0.0F; // ヨー
```

```
int mode = 1; // 出力モード
```

```
//(0:ジャイロ 1:加速度 2:ピッチ・ロール・ヨー)
```

一般社団法人全国専門学校情報教育協会

11

ソースコード 2/9 (M5C_ACC_GRAPH_1)

◇初期化部 前半

```
void setup() { // 初期化部
  M5.begin(); // M5Stick-C初期化
  M5.IMU.Init(); // 慣性測定装置(IMU:Inertial Measurement Unit)の初期化

  M5.Lcd.setTextDatum(0); // LCDの向き設定(縦長)
  M5.Lcd.fillScreen(BLACK); // 黒で塗りつぶし
  M5.Lcd.setCursor(3, 10); // カーソル位置
  M5.Lcd.setTextColor(WHITE); // 文字色 白
  M5.Lcd.setTextSize(1); // 文字の大きさ
  M5.Lcd.printf("Acceleration"); // タイトル

  M5.Lcd.drawLine(10, 60, 70, 60, BLUE); //--- グラフの水平基準軸
  M5.Lcd.drawLine(40, 20, 40, 159, BLUE); //| グラフの垂直基準軸

  M5.Lcd.setCursor(29, 20); // +-表示のカーソル位置
  M5.Lcd.printf("- +"); // + - 表示
```

ソースコード 3/9 (M5C_ACC_GRAPH_1)

◇初期化部 後半

```
//BAR (x, y, xw, yw, c)
M5.Lcd.fillRect(41, 30, 10, 8, BLUE); // 棒グラフ
M5.Lcd.fillRect(41, 40, 10, 8, GREEN);
M5.Lcd.fillRect(41, 50, 10, 8, RED);

//加速度データ配列初期化
for(i=0;i<100;i++){
  acv[0][i]=0;
  acv[1][i]=0;
  acv[2][i]=0;
}
}
```

ソースコード 4/9 (M5C_ACC_GRAPH_1)

◇通常処理部 序盤

```
void loop() { // 通常処理部
  M5.update(); // M5Stick-C内部更新(ボタン押下状況)
  M5.IMU.getGyroData(&gyroX, &gyroY, &gyroZ); // IMUから各データを読み出す
  M5.IMU.getAccelData(&accX, &accY, &accZ);
  M5.IMU.getAhrsData(&pitch, &roll, &yaw);

  if ( mode == -1 || M5.BtnA.wasReleased() ) { // ボタンが押されていたら出力モード変更
    mode++;
    mode = mode % 3;

    // プロッタ用のタイトル出力
    if ( mode == 0 ) {
      Serial.printf("gyroX,gyroY,gyroZ\n");
    } else if ( mode == 1 ) {
      Serial.printf("accX,accY,accZ\n");
    } else if ( mode == 2 ) {
      Serial.printf("pitch,roll,yaw\n");
    }
  }
}
```

14

一般社団法人全国専門学校情報教育協会

ソースコード 5/9 (M5C_ACC_GRAPH_1)

◇通常処理部 中盤

```
if ( mode == 0 ) { // データ出力
  Serial.printf("%6.2f,%6.2f,%6.2f\n", gyroX, gyroY, gyroZ); // 角加速度 :
} else if ( mode == 1 ) {
  Serial.printf("%5.2f,%5.2f,%5.2f\n", accX, accY, accZ); // 加速度
} else if ( mode == 2 ) {
  Serial.printf("%5.2f,%5.2f,%5.2f\n", pitch, roll, yaw); // ピッチ・ロール・ヨー
}

//BAR (x, y, xw, yw, c)
M5.Lcd.fillRect(0, 30, 80, 30, BLACK); //棒グラフ領域のクリア

AX = (int)(accX * maxG); // グラフ用加速度値の計算
AY = (int)(accY * maxG);
AZ = (int)(accZ * maxG);
acv[0][i]=AX; // 現在の表示値格納
acv[1][i]=AY;
acv[2][i]=AZ;
if(++i >= 100){i = 0;} // 配列はリングバッファ--->インデックス更新
```

15

一般社団法人全国専門学校情報教育協会

ソースコード 6/9 (M5C_ACC_GRAPH_1)

◇通常処理部 棒グラフ部

```
if(AX > 0){ // 正のX棒グラフ
  M5.Lcd.fillRect(41, 30, AX, 8, BLUE);
}else{ // 負のX棒グラフ
  M5.Lcd.fillRect(40+AX, 30, -(AX), 8, BLUE);
}

if(AY > 0){ // 正のY棒グラフ
  M5.Lcd.fillRect(41, 40, AY, 8, GREEN);
}else{ // 負のY棒グラフ
  M5.Lcd.fillRect(40+AY, 40, -(AY), 8, GREEN);
}

if(AZ > 0){ // 正のZ棒グラフ
  M5.Lcd.fillRect(41, 50, AZ, 8, RED);
}else{ // 負のZ棒グラフ
  M5.Lcd.fillRect(40+AZ, 50, -(AZ), 8, RED);
}
```

ソースコード 7/9 (M5C_ACC_GRAPH_1)

◇通常処理部 基準軸と凡例

```
M5.Lcd.drawLine(40, 20, 40, 60, BLUE); // | 垂直基準軸 上書き
M5.Lcd.setCursor(2, 30); // 凡例 X
M5.Lcd.printf("X");

M5.Lcd.setCursor(2, 40); // 凡例 Y
M5.Lcd.printf("Y");

M5.Lcd.setCursor(2, 50); // 凡例 Z
M5.Lcd.printf("Z");

accPlot(i); // 折れ線グラフ部(と言ってもドット)描画
delay(50);
}
```

ソースコード 8/9 (M5C_ACC_GRAPH_1)

◇折れ線グラフ部 前半

```
void accPlot(int i){ // 折れ線グラフ部描画関数
  int n;
  int y;

  //グラフ領域のクリア
  //BAR (x, y, xw, yw, c)
  M5.Lcd.fillRect(0, 61, 80, 99, BLACK);
  //グラフのXY軸
  M5.Lcd.drawLine(10, 60, 70, 60, BLUE); //---
  M5.Lcd.drawLine(40, 20, 40, 159, BLUE); // |

  if(i==0){
    n=99;
  }else{
    n=i-1;
  }
}
```

ソースコード 9/9 (M5C_ACC_GRAPH_1)

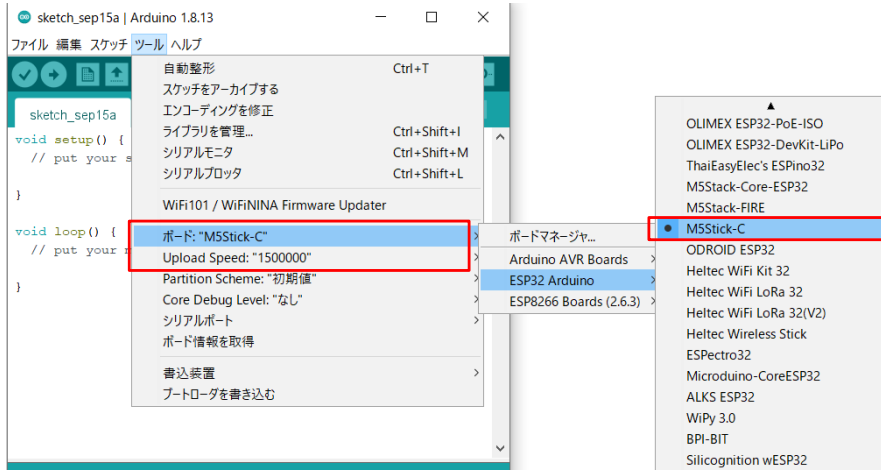
◇折れ線グラフ部 後半

```
for(y=0;;y++){
  if(acv[0][n] >= 0){
    M5.Lcd.drawPixel(41 + acv[0][n], 61 + y, BLUE); // 正のX
  }else{
    M5.Lcd.drawPixel(40 + acv[0][n], 61 + y, BLUE); // 負のX
  }
  if(acv[1][n] >= 0){
    M5.Lcd.drawPixel(41 + acv[1][n], 61 + y, GREEN); // 正のY
  }else{
    M5.Lcd.drawPixel(40 + acv[1][n], 61 + y, GREEN); // 負のY
  }
  if(acv[2][n] >= 0){
    M5.Lcd.drawPixel(41 + acv[2][n], 61 + y, RED); // 正のZ
  }else{
    M5.Lcd.drawPixel(40 + acv[2][n], 61 + y, RED); // 負のZ
  }
}
```

```
n--;
  if(n<0){n=99;}
  if(n==i){break;}
}
```

マイコンボードの選択

- ◇以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

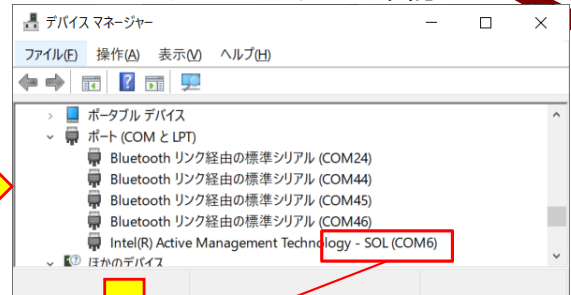


マイコンをPCと接続

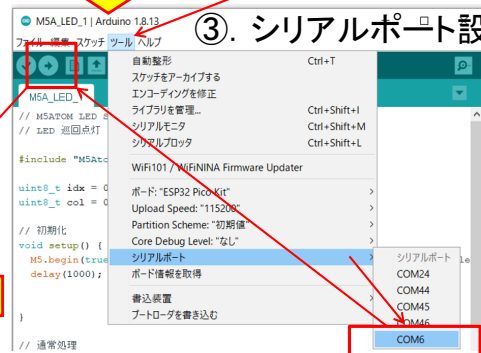
①. M5Stick-CをPCと接続



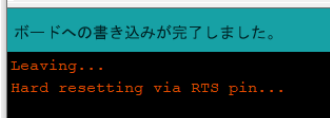
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

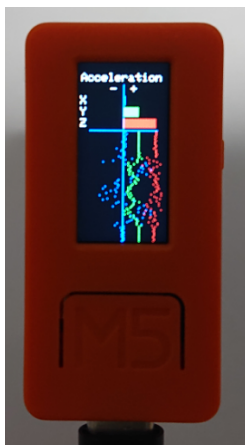


④. コンパイル



動作確認 1/2

- ◇プログラムが書き込まれると、マイコンが重力加速度Gを検出してグラフが左図のように描画される
- ◇シリアルモニタを起動すれば右図のように、計測した値が表示される
- ◇Aボタンを押下すると、シリアルモニタの表示が切り替わる(G値→角加速度→ピッチ・ロール・ヨー)



COM10		
-0.22,-0.00, 1.06		
-0.21,-0.00, 1.06		
-0.22,-0.00, 1.05		
-0.22, 0.00, 1.06		
-0.22,-0.00, 1.05		
-0.22, 0.00, 1.06		
-0.22,-0.00, 1.06		
-0.22,-0.00, 1.06		
-0.22,-0.00, 1.06		
-0.22, 0.00, 1.06		
-0.21,-0.00, 1.06		
-0.22, 0.00, 1.06		
-0.21, 0.00, 1.06		
-0.21, 0.00, 1.05		

G値

6.06,13.74,-70.18
6.00,13.76,-69.80
6.05,13.75,-69.43
6.05,13.77,-69.05
6.09,13.77,-68.70
6.05,13.77,-68.35
6.03,13.76,-68.00
6.02,13.71,-67.66
6.03,13.66,-67.31
6.03,13.66,-66.96
6.06,13.58,-66.65
6.05,13.58,-66.29
6.03,13.59,-65.91

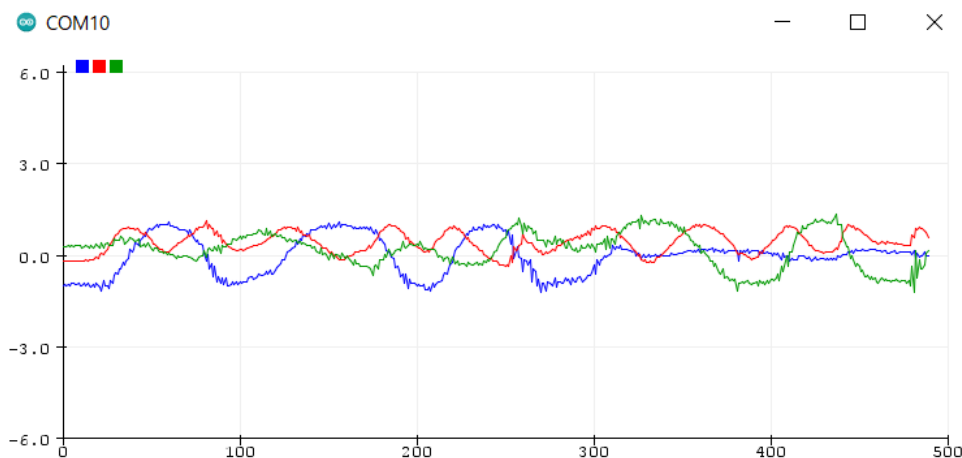
角加速度

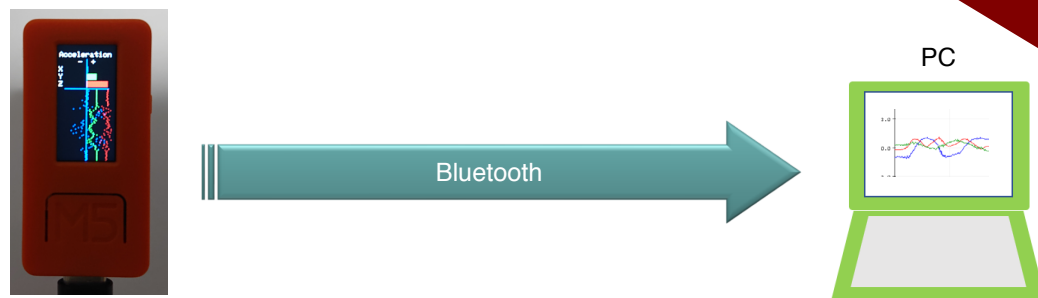
11.66, -8.00, 13.06
11.54, -7.02, 12.70
10.62, -6.04, 12.15
10.38, -6.90, 12.08
11.35, -6.90, 12.70
11.35, -7.14, 12.45
10.99, -6.84, 12.45
11.23, -7.20, 12.33
11.66, -7.81, 13.55
11.54, -6.84, 11.96
11.78, -7.39, 12.15
11.66, -7.14, 13.18
10.99, -7.32, 12.39

ピッチ・ロール・ヨー

動作確認 2/2

- ◇シリアルモニタウインドウを閉じ、IDEメニューで ツール → シリアルプロッタ を選択する
- ◇プログラムで各軸のG値を【,】カンマで区切って出力すればPCでも簡易グラフを描画できる
- ※ただし、データの記録はできない





Bluetooth 経由 加速度計

ACCERELOMETER VIA BLUETOOTH

システム構想

- ◇いま動作確認を終えたシステムの通信を**Serial通信**から**Bluetooth通信**に置き換える
- ◇M5Stick-Cの開発環境には SPP用 **BluetoothSerial** ライブラリが含まれている

※IDEで スケッチ → ライブラリをインクルード とたどれば BluetoothSerial を確認できる

- ◇先に動作確認を終えたプログラムを流用すれば、容易に実現できる

→ ソースコードを少し変更するだけで実現できる！！

- ◇M5Stick-Cは、小さいがバッテリーを内蔵しているので、Bluetooth通信を行えば、USBケーブルのない**無線センサユニット**が出来上がる

ソースコード 1/2 (M5C_ACC_GRAPH_Bluetooth_2)

- ◇初期化処理部 追加・変更する部分のみ下記する
- ◇冒頭部分

```
#include <M5StickC.h>
#include <BluetoothSerial.h> // Bluetooth Serial ライブラリ ※青字部分を追加・変更する

#define maxG 30 // G値をグラフにする際の1G当たりのドット数

BluetoothSerial bts; // Bluetooth Object

float accX = 0.0F; // X方向加速度
float accY = 0.0F; // Y方向加速度
float accZ = 0.0F; // Z方向加速度
... ..

void setup() { // 初期化部
  M5.begin(); // M5Stick-C初期化
  bts.begin("*****"); // PC側でペアリングするデバイス名称 (ユニークな名称)
  M5.IMU.Init(); // 慣性測定装置 (IMU: Inertial Measurement Unit) の初期化
  ... ..
```

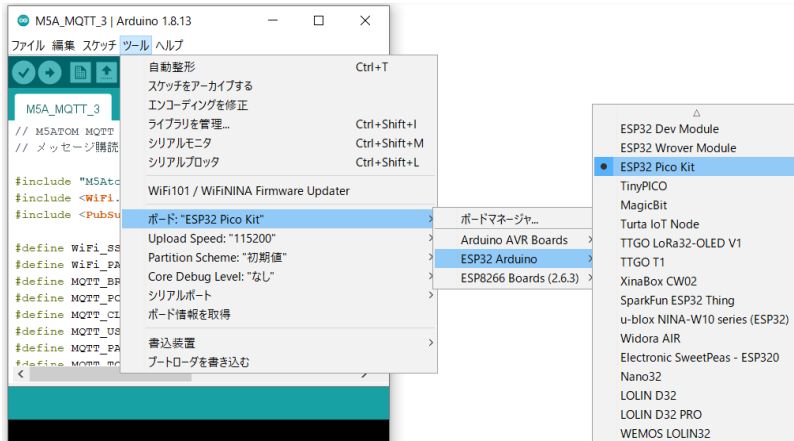
ソースコード 2/2 (M5C_ACC_GRAPH_Bluetooth_2)

- ◇通常処理部 追加・変更する部分のみ下記する

```
// プロッタ用のタイトル出力
if ( mode == 0 ) {
  bts.printf("gyroX,gyroY,gyroZ\n"); // Serial→btsに変更
} else if ( mode == 1 ) {
  bts.printf("accX,accY,accZ\n"); // Serial→btsに変更
} else if ( mode == 2 ) {
  bts.printf("pitch,roll,yaw\n"); // Serial→btsに変更
}
}
// データ出力
if ( mode == 0 ) {
  bts.printf("%.2f,%.2f,%.2f\n", gyroX, gyroY, gyroZ); // Serial→btsに変更
} else if ( mode == 1 ) {
  bts.printf("%.2f,%.2f,%.2f\n", accX, accY, accZ); // Serial→btsに変更
} else if ( mode == 2 ) {
  bts.printf("%.2f,%.2f,%.2f\n", pitch, roll, yaw); // Serial→btsに変更
}
... ..
```

マイコンボードの選択

- ◇以下のように IDE で **ツール** → **ボード** → **…とたどり、ESP32 Pico Kit** を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
 - ボード名 ESP32 Pico Kit が無い場合、NodeMCU-32S を探す ←M5Atom ライブラリが無い場合
- ◇同様に シリアルポートの Upload Speed は、115200 にセットする(これより早いと書き込みに失敗する)
- ◇以後、**M5ATOMを使用する場合は、必ずこの設定で行う**

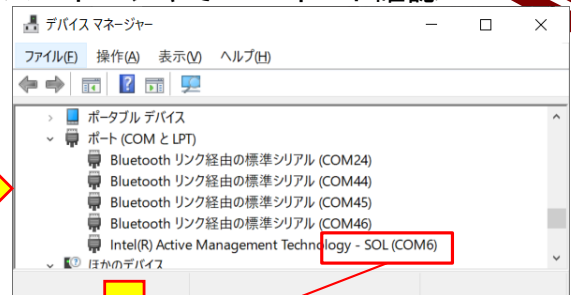


マイコンをPCと接続

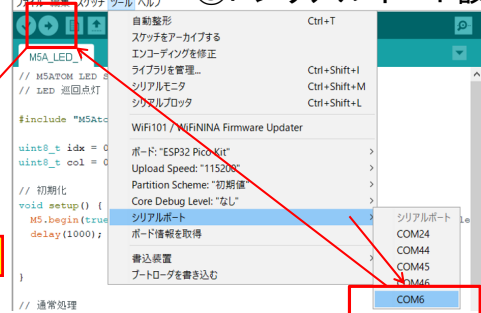
①. M5Stick-CをPCと接続



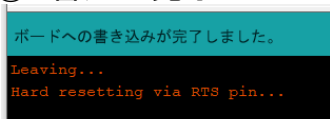
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



動作確認 1/4 Bluetooth Device の追加(ペアリング)

◇書き込みが完了したら、数秒待って、以下の手順でデバイスのペアリングを行う

①スタート→設定



②デバイス



③Bluetoothとその他のデバイス



④デバイスを追加する



⑤プログラム指定のデバイス名選択



⑥ペアリング完了




30

動作確認 2/4 Bluetooth Serial の発信用ポート番号を確認

- ◇スタートボタンから
設定→デバイス→その他のBluetoothオプション
とたどり、【COMポート】タブを選択する
- ◇ペアリングしたBluetoothデバイス名の方向が【発信】と
なっているCOMポート番号を確認しメモする
- ◇メモしたCOMポート番号をIDEのシリアルポートで
選択する

発信のCOMポート



Bluetooth 設定

オプション COM ポート ハードウェア

この PC では、下の一覧に表示された COM (シリアル) ポートが使用されています。Bluetooth デバイスのマニュアルを参照して、COM ポートが必要かどうかを判断してください。

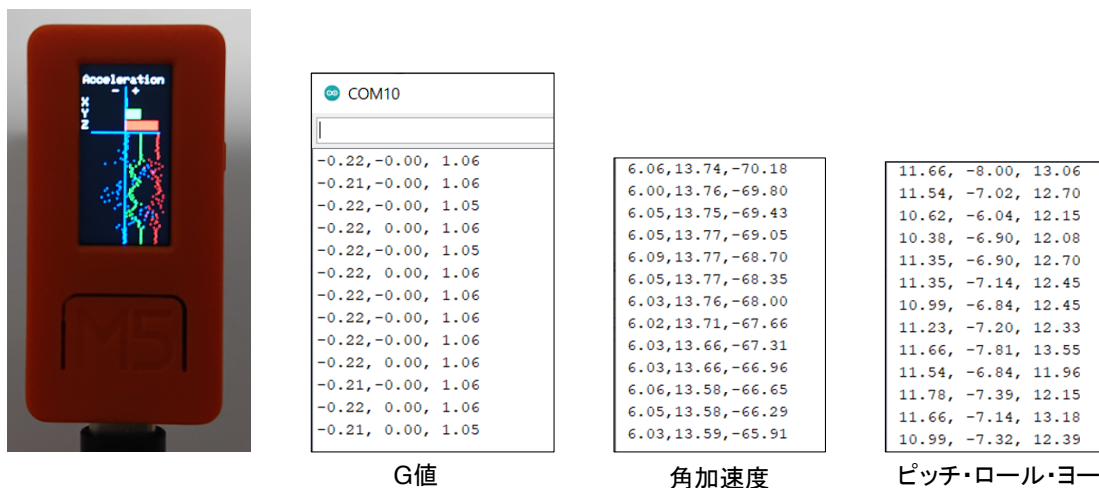
ポート	方向	名前
COM24	発信	M5C-BTS-ACC 'ESP32SPP'
COM44	着信	M5C-BTS-ACC
COM45	発信	M5C-BTS-COLOR 'ESP32SPP'
COM46	着信	M5C-BTS-COLOR
COM47	発信	M5ATOM_KEN 'ESP32SPP'
COM48	着信	M5ATOM_KEN
COM49	着信	M5Stick-C_KEN
COM50	発信	M5Stick-C_KEN 'ESP32SPP'

追加(D)... 削除(R)

31

動作確認 3/4

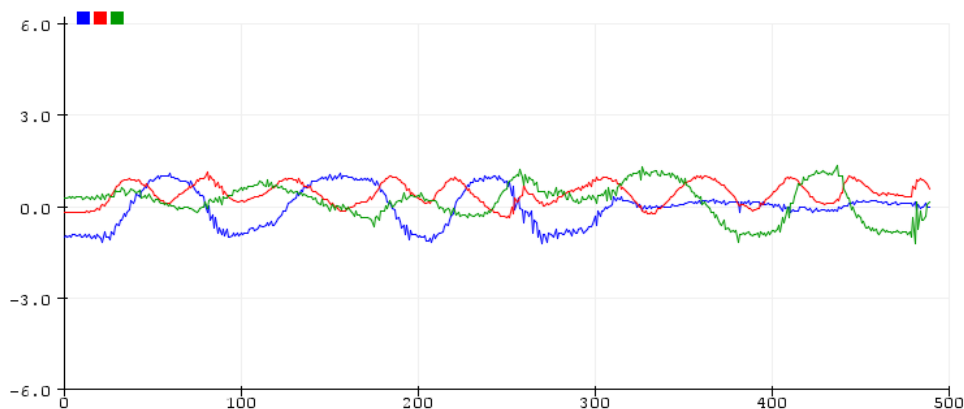
- ◇プログラムが書き込まれると、マイコンが重力加速度Gを検出してグラフが左図のように描画される
- ◇シリアルモニタを起動すれば右図のように、計測した値が表示される
- ◇Aボタンを押下すると、シリアルモニタの表示が切り替わる(G値→角加速度→ピッチ・ロール・ヨー)



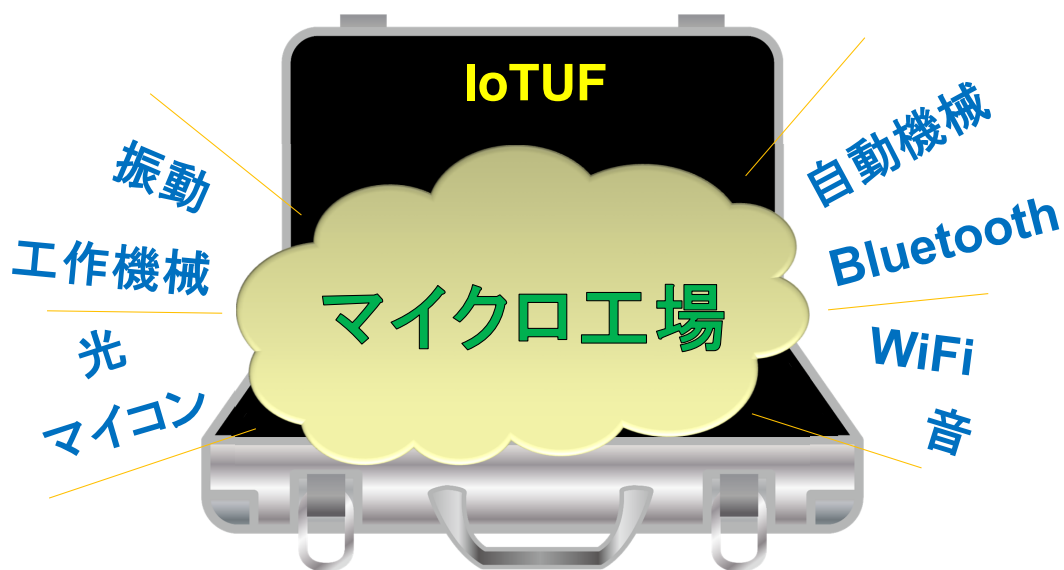
動作確認 4/4

- ◇シリアルモニタウインドウを閉じ、IDEメニューで ツール → シリアルプロッタ を選択する
- ◇プログラムで各軸のG値を【,】カンマで区切って出力すればPCでも簡易グラフを描画できる
- ※ただし、データの記録はできない

※M5Stick-Cはバッテリーを内蔵しているので、USBケーブルを外しても、このプロットは続けることができる



IoT μ Factory の実現



スチールプレート & マグネット

◇実習キットには、次のものが含まれている

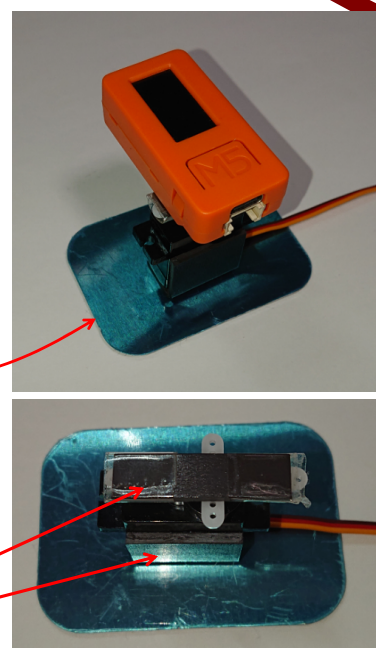
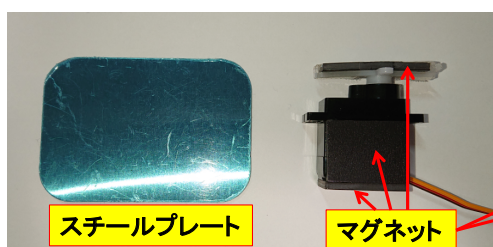
- ①. スチールプレート
- ②. マグネット付きサーボモータ

※すでにサーボモータ制御で利用したかもしれないが、マグネットでサーボモータをスチールプレートに固定すると、動作確認しやすい

◇M5Stick-Cの背面にもマグネットが内蔵されているので、これを利用すると、右上図のように、M5Stick-Cをサーボホーンに固定することができる

◇この状態でサーボモータを動作させれば、加速度センサで往復回転時の振動計測ができる

→ 工場設備の振動を測定する**極小モデル = μ 工場** となる



2.14 カラーセンサの活用

M5Stick-C

カラーセンサ



一般社団法人全国専門学校情報教育協会

光色検出

◇IoTでは、光を検出するセンサが多く用いられる

例： 製品や部材の通過確認・カウント → 生産実績計数
高さ・位置検出 → ワーク確認、位置決め
表面状態の検査 → 品質管理：反射光・カメラ画像

◇光色検出

光の3原色 → 赤・緑・青 の光波長に感度を持つセンサを用いる
3色別々のセンサを同時に用いる方法は、調整が面倒

1つのパッケージで3色の光を検出する センサ光色デジタル変換器で色を感知できる
光色デジタル変換器 : TCS34732 を使う



一般社団法人全国専門学校情報教育協会

TSC3472 Data Sheet Description

The TSC3472 device provides a digital return of red, green, blue (RGB), and clear light sensing values. An IR blocking filter, integrated on-chip and localized to the color sensing photodiodes, minimizes the IR spectral component of the incoming light and allows color measurements to be made accurately. The high sensitivity, wide dynamic range, and IR blocking filter make the TSC3472 an ideal color sensor solution for use under varying lighting conditions and through attenuating materials.

The TSC3472 color sensor has a wide range of applications including RGB LED backlight control, solid-state lighting, health/fitness products, industrial process controls and medical diagnostic equipment. In addition, the IR blocking filter enables the TSC3472 to perform ambient light sensing (ALS). Ambient light sensing is widely used in display-based products such as cell phones, notebooks, and TVs to sense the lighting environment and enable automatic display brightness for optimal viewing and power savings. The TSC3472, itself, can enter a lower-power wait state between light sensing measurements to further reduce the average power consumption.

説明

TSC3472デバイスは、赤・緑・青(RGB)そしてクリアの光感知値を提供する。色検知光ダイオードのためにチップ上で集積されて限定された**赤外光遮蔽フィルタ**が入射光の赤外光成分を最小化して、色測定を高精度化できる。高感度で広いダイナミックレンジ、そして赤外光遮蔽フィルタがTSC3472を、変化する照明条件や光を減衰させる物質を通した利用に対する理想的な色センサの解決策にする。

RGB LEDのバックライト制御、固体照明、健康/フィットネス製品、工業プロセス制御、医療診断装置を含む応用の広い幅を持っている。さらに、**赤外光遮蔽フィルタ**は、**環境光感知(ALS)性能**をTSC3472にもたらしめている。環境光感知は、光学的な鑑賞(見た目)や電力制限のために、携帯電話、ノートPC、そしてTVのような表示に主な機能をもつ製品に幅広く用いられており、**自動表示輝度調整**を可能にする。TSC3472 は平均電力消費の削減を進めるために、光感知測定の間、低電力待ち状態に入ることができる。

2

TSC3472 Detailed Description 1/2

The TSC3472 light-to-digital converter contains a 3×4 photodiode array, four analog-to-digital converters (ADC) that integrate the photodiode current, data registers, a state machine, and an I²C interface. The 3×4 photodiode array is composed of red-filtered, green-filtered, blue-filtered, and clear (unfiltered) photodiodes. In addition, the photodiodes are coated with an IR-blocking filter. The four integrating ADCs simultaneously convert the amplified photodiode currents to a 16-bit digital value. Upon completion of a conversion cycle, the results are transferred to the data registers, which are double-buffered to ensure the integrity of the data. All of the internal timing, as well as the low-power wait state, is controlled by the state machine.

詳細説明

TSC3472 光デジタル変換器は 3×4 光ダイオードアレイと光ダイオード電流を積分する4個のアナログ-デジタル変換器(ADC)、光データレジスタ、ステートマシン、および**I²C I/F**を持っている。 3×4 光ダイオードアレイは、赤、緑、青フィルタの掛けられたものと、クリア(フィルタなし)の光ダイオードから構成されている。さらに、光ダイオードは赤外光遮断フィルタでコートされている。**4つの集積ADCは、増幅された光電流を16bitデジタル値に同時変換する**。変換サイクルの完了で、その結果は、データレジスタに送られる。それは、データの蓄積を保証するためダブルバッファになっている。低電力待ち状態と同じように、すべての内部タイミングはステートマシンで制御されている。

3

TSC3472 Detailed Description 2/2

Communication of the TCS3472 data is accomplished over a fast, up to 400 kHz, two-wire I²C serial bus. The industry standard I²C bus facilitates easy, direct connection to microcontrollers and embedded processors.

In addition to the I²C bus, the TCS3472 provides a separate interrupt signal output. When interrupts are enabled, and user-defined thresholds are exceeded, the active-low interrupt is asserted and remains asserted until it is cleared by the controller. This interrupt feature simplifies and improves the efficiency of the system software by eliminating the need to poll the TCS3472. The user can define the upper and lower interrupt thresholds and apply an interrupt persistence filter. The interrupt persistence filter allows the user to define the number of consecutive out-of-threshold events necessary before generating an interrupt. The interrupt output is open-drain, so it can be wire-ORed with other devices.

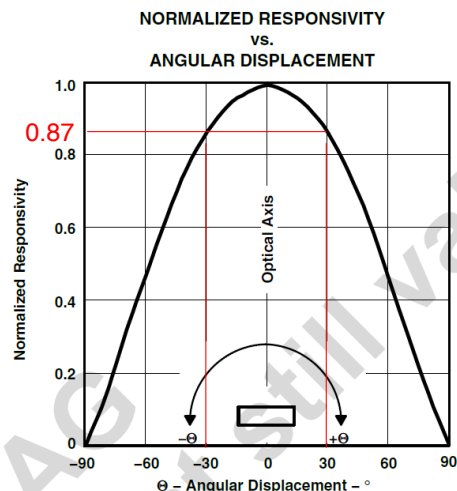
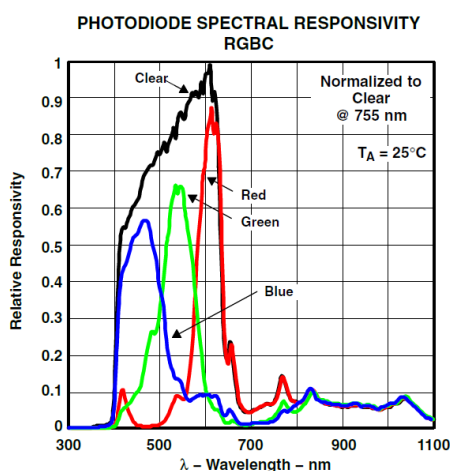
TC3472データの通信は、400kHzに上る高速の2線式I2Cシリアルバスで達成される。工業規格I2Cバスは、マイクロコントローラと組み込まれている計算機の直接接続を簡単に実現する。

さらに、I2Cバスに対してTCS3472は、別の割り込み信号出力を提供する。割り込みが可能な場合、そしてユーザー定義閾値を超える場合に、アクティブロー割り込みが宣言されて、それがプロセッサによってクリアされるまでは、そのまま残っている。この割り込みの特徴は、TCS3472問い合わせの必要を無くすことで、システムソフトウェアの効率を向上させている。ユーザーは上位、低位の割り込み閾値を決めて、割り込み持続フィルタに適用することができる。その割り込み持続フィルタは、ユーザーが割り込み生成前に必要な、連続する閾値外イベントの数を決められるようにしている。割り込み出力は、オープンドレインなので、他のデバイスとワイヤードオアすることができる。

TSC3472 Data Sheet

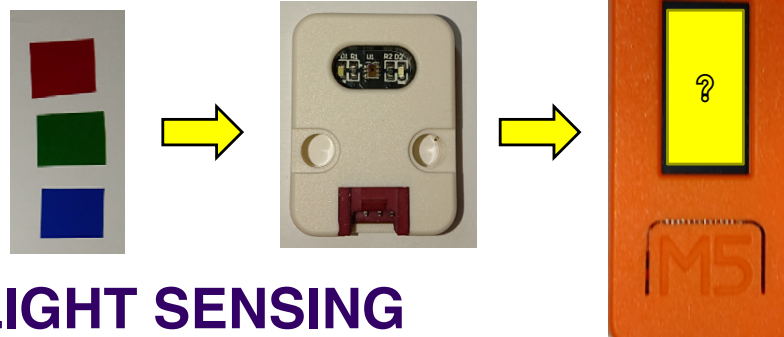
◇光ダイオードスペクトル感度および規格化感度対角度変異

- ①. およそ650nm波長まで感知する性能がある
- ②. ±30度の角度範囲で80%以上の応答性能



光色感知

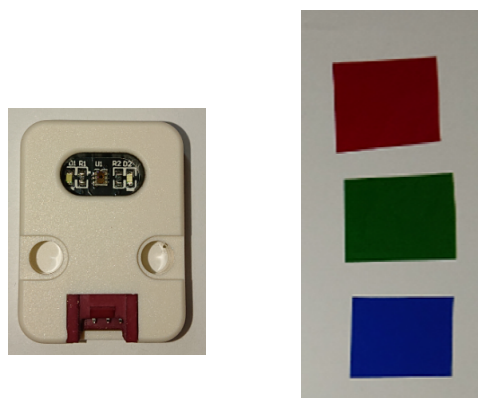
COLOR LIGHT SENSING



システム構想

◇カラーセンサが感知した光色を、M5Stick-Cのカラー液晶表示で再現してみよう！

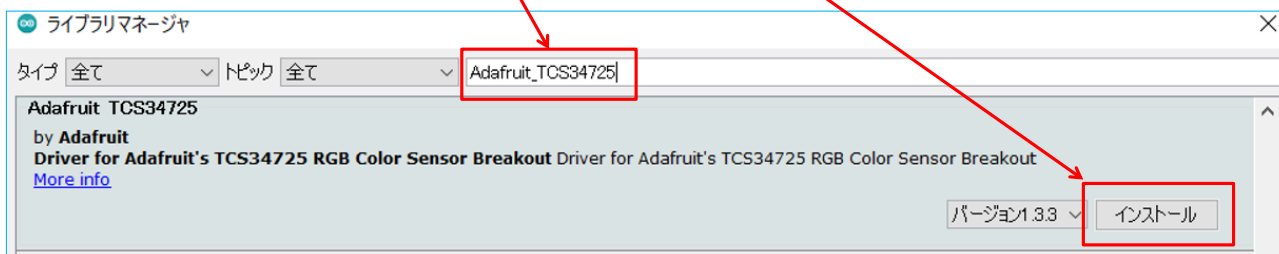
- ①. 色付きセロファンを通過・反射した光をカラーセンサで検出する
 - ②. 検出した光色データは、16bitデジタル値になる
 - ③. 光色データを液晶のカラーコードに変換して、M5Stick-Cで色再現する
- ※公開されているTCS3472 用ライブラリを利用する



ライブラリの追加インストール

◇光色センサを制御するために、I2C I/Fを用いてセンサにコマンドを送る
→ TCS3472 用ライブラリが公開されている

◇IDEのメニューで スケッチ → ライブラリをインクルード → ライブラリを管理 とたどる
◇ライブラリマネージャで、下図の**ライブラリ**を検索して**インストールする**（執筆時バージョン 1.3.3）



ソースコード 1/4 (M5C_Color_TCS3472_1)

◇冒頭から 初期化部 前半

```
#include <Wire.h> // センサ用ライブラリで使用する通信ライブラリ(I2C)
#include <M5StickC.h>
#include "Adafruit_TCS34725.h" // センサTCS3472用ライブラリ

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X); // センサオブジェクト (仮)積分時間、ゲイン×4倍

void setup() { // 初期化部
  delay(100); // センサステートマシン起動時間を十分確保

  M5.begin(true, true, true); // M5Stick-C初期化
  Serial.begin(115200); // シリアル通信速度
  Serial.println("Color View Test!");

  ... ..
}

以後に続く
```

ソースコード 2/4 (M5C_Color_TCS3472_1)

◇初期化部 後半

```
続き
    ...    ...

while(!tcs.begin()){ // 初期化に失敗した場合、センサーが接続されていないと判断する
    Serial.println("No TCS34725 found ... check your connections");
    M5.Lcd.setTextFont(1);
    M5.Lcd.setTextColor(WHITE);
    M5.Lcd.drawString("No Found sensor.",0,0);
    delay(1000); // さらに1秒待つ
}
tcs.setIntegrationTime(TCS34725_INTEGRATIONTIME_154MS); // 積分時間154MS
tcs.setGain(TCS34725_GAIN_4X); // ゲイン×4倍
}
```

ソースコード 3/4 (M5C_Color_TCS3472_1)

◇通常処理部 前半

```
void loop() {
    uint16_t clear, red, green, blue;

    delay(60); // takes 50ms to read

    tcs.getRawData(&red, &green, &blue, &clear); // 生データ取り込み

    // Figure out some basic hex code for visualization
    uint32_t sum = clear;
    float r, g, b;
    r = red; r /= sum; // クリア光の値で各色を規格化する(最大=1.0)
    g = green; g /= sum;
    b = blue; b /= sum;
    r *= 256; g *= 256; b *= 256; // 各色を8bit(最大256)にする
    uint16_t c = M5.Lcd.color565((int)r, (int)g, (int)b); // 各色256を16bit値(565)に変換
    M5.Lcd.fillScreen(c); // 変換したRGB565コードでLCDを塗りつぶす

    ...    ...
}
続く
```


ソースコード 4/4 (M5C_Color_TCS3472_1)

◇通常処理部 後半

```
続き
    ...
Serial.print("RGB=");
Serial.print(","); // なぜここに(,)カンマを入れているかは、後で分かる
Serial.print(r); // 赤の値(Max256)
Serial.print(",");
Serial.print(g); // 緑の値(Max256)
Serial.print(",");
Serial.print(b); // 青の値(Max256)
Serial.println("");

delay(100); // しばし待つ
}
```

センサの接続

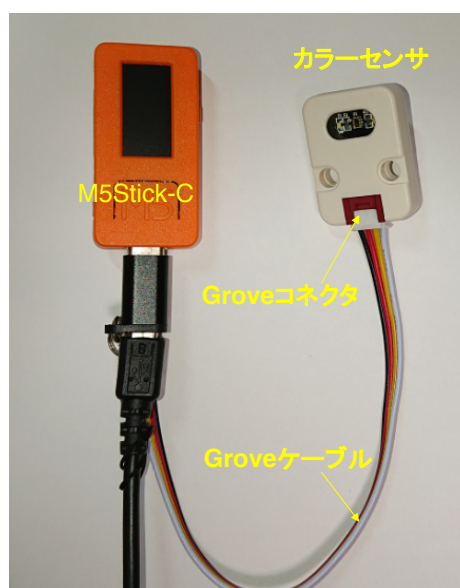
◇プログラム書込み前に、あらかじめセンサを
M5Stick-Cと接続しておく

◇右図のように、センサ付属の専用ケーブル両端を
センサとM5Stick-CのGroveコネクタに差し込む

※コネクタには向きがあり、反対向きには
差し込めないようになっている

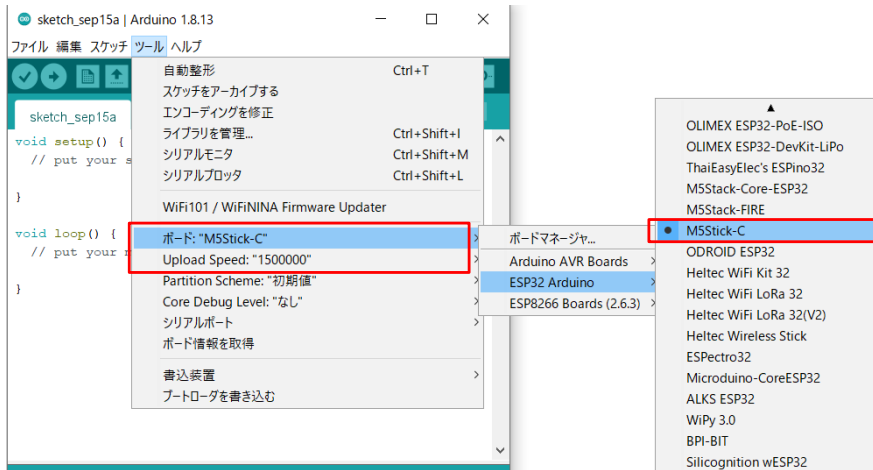
**注意) 実験が終わって、ケーブルを取り外す際は、
ケーブルにテンションを掛けず、コネクタを引いて
外すこと！**

◇センサを接続後、USBケーブルでM5Stick-Cと
PCを接続する



マイコンボードの選択

- ◇ 以下のように IDE で ツール → ボード → ...とたどり、**M5Stick-C** を選択する
※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇ 同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇ 以後、**M5Stick-C**を使用する場合は、必ずこの設定で行う

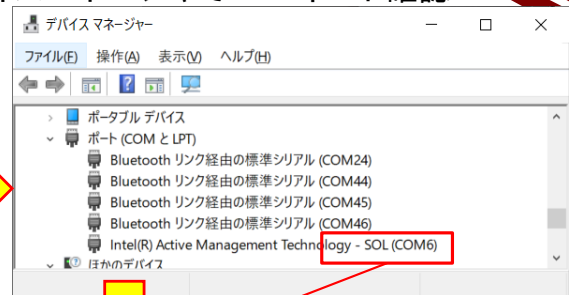


マイコンをPCと接続

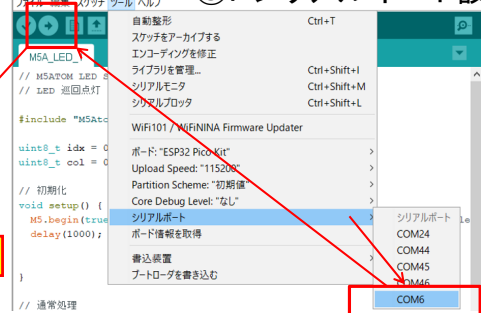
①. M5Stick-CをPCと接続



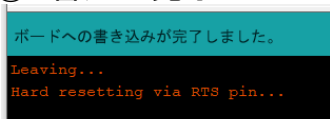
②. デバイスマネージャでCOMポート確認



③. シリアルポート設定



⑤. 書き込み完了

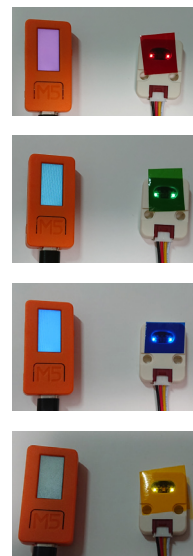


④. コンパイル



動作確認 1/2

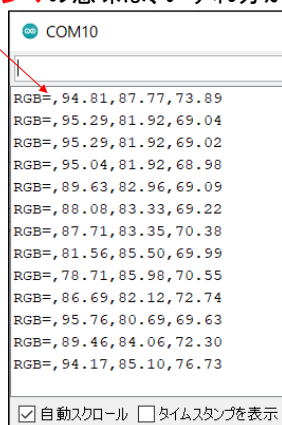
- ◇プログラムが書き込まれると、プログラムがスタートし、光センサが感知し色味をLCDに表示している
- ◇色セロファン紙を感知窓に載せると、感知した色がLCDに表示される
- ※印刷物で色の表現が難しいが、実験時の照明環境にも影響されるので、採光を工夫して、どのような照明環境が良いか調べてほしい

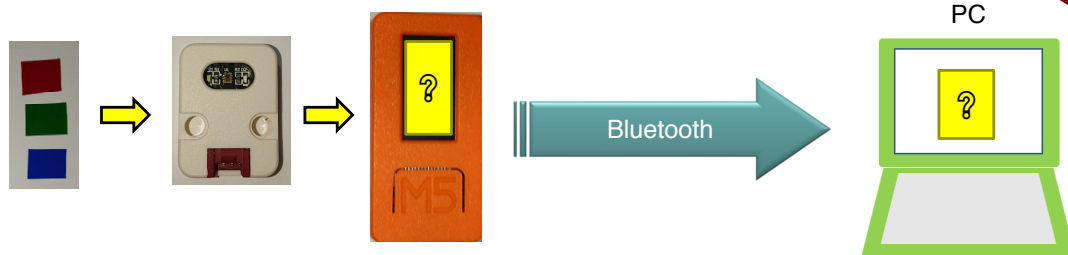


動作確認 2/2

- ◇この時シリアルモニタには、図のようなデータが表示されている
- このデータは、**R・G・B**各色の光の強さのデータである
- ◇次の実験では、このデータをPCで受信して、PC画面で色を再現する

→ RGB=の次の(,)カンマの意味は、いずれ分かる





感知データをBluetooth で送信する

COLOR LIGHT SENSING WITH BLUETOOTH

システム構想

- ◇いま動作確認を終えたシステムの通信を**Serial通信からBluetooth通信に置き換える**
- ◇M5Stick-Cの開発環境には SPP用 **BluetoothSerial ライブラリ**が含まれている
 - ※IDEで スケッチ → ライブラリをインクルード とたどれば BluetoothSerial を確認できる
- ◇先に動作確認を終えたプログラムを流用すれば、容易に実現できる
 - ソースコードを少し変更するだけで実現できる！！
- ◇M5Stick-Cは、小さいがバッテリーを内蔵しているので、Bluetooth通信を行えば、USBケーブルのない**無線センサユニット**が出来上がる

【追加】

- ◇PC側で受信した**色データを再現するシステムが必要**である
- ◇Arduino IDE とよく似た、PCグラフィックス向き開発言語 Processing を使おう

- ①. 母体言語であるJavaを単純化したもの
- ②. グラフィックス処理に特化した言語
- ③. 通信も得意である

ソースコード 1/2 (M5C_COLOR_TCS3472_BLT_2)

- ◇初期化処理部 追加・変更する部分のみ下記する
- ◇冒頭部分

```
#include <Wire.h>
#include <M5StickC.h>
#include "Adafruit_TCS34725.h" // センサTCS3472用ライブラリ “ ”は<>でも良い
#include <BluetoothSerial.h> //

Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);
BluetoothSerial bts; //

void setup() { // 初期化部
  bts.begin(" * * * * "); // ペアリングに必要なデバイス名。各自ユニークな名称にすること！！
  delay(100); // ステートマシン起動時間を十分確保

  ...
}
```

※青字部分を追加・変更する

ソースコード 2/2 (M5C_COLOR_TCS3472_BLT_2)

- ◇通常処理部の最後 追加・変更する部分のみ下記する

```
void loop() {
  ...
  ...

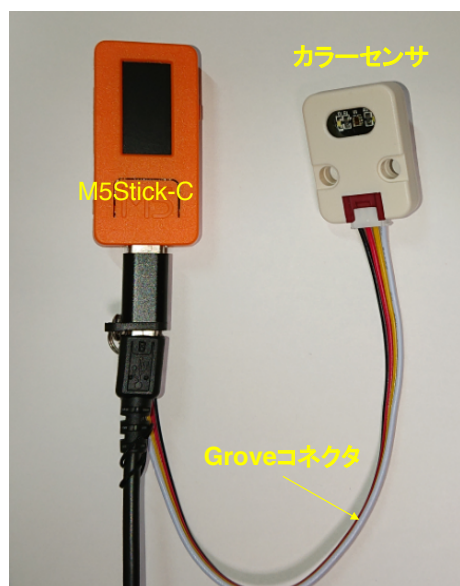
  bts.print("RGB=");
  bts.print(","); // なぜここに(,)カンマを入れているかは、後で分かる
  bts.print(r);
  bts.print(",");
  bts.print(g);
  bts.print(",");
  bts.print(b);
  bts.println("");

  delay(100); // しばし待つ
}
```

※青字部分を追加・変更する

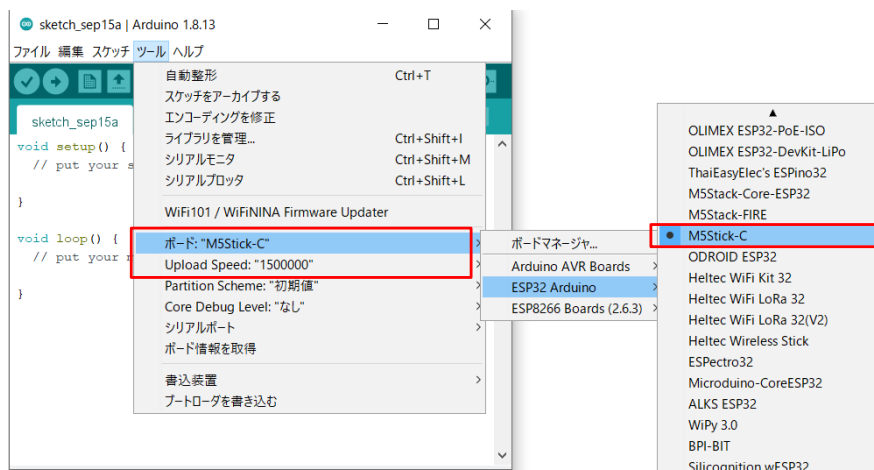
センサの接続

- ◇プログラム書込み前に、あらかじめセンサを M5Stick-Cと接続しておく
- ◇右図のように、センサ付属の専用ケーブル両端をセンサとM5Stick-CのGroveコネクタに差し込む
 - ※コネクタには向きがあり、反対向きには差し込めないようになっている
- 注意) 実験が終わって、ケーブルを取り外す際は、ケーブルにテンションを掛けず、コネクタを引いて外すこと!**
- ◇センサを接続後、USBケーブルでM5Stick-CとPCを接続する



マイコンボードの選択

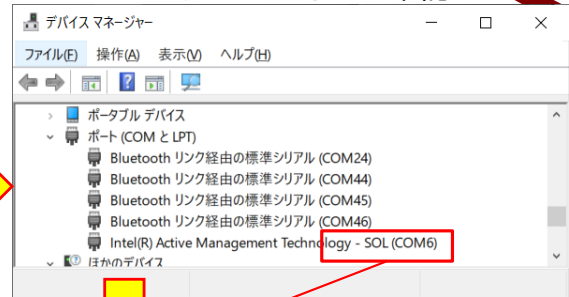
- ◇以下のように IDE で **ツール → ボード → ...とたどり、M5Stick-C** を選択する
 - ※ボード以後の表示は、使用しているIDEの状況に応じて変わる
- ◇同様に シリアルポートの Upload Speed は、1500000 にセットする
- ◇以後、M5Stick-Cを使用する場合は、必ずこの設定で行う



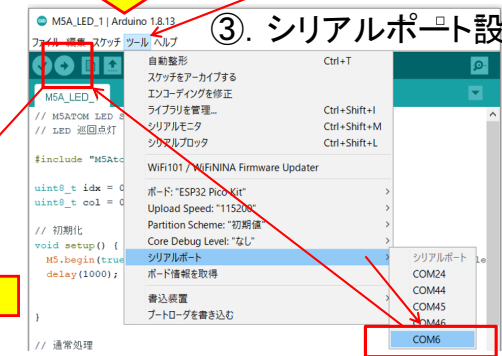
②. デバイスマネージャでCOMポート確認

マイコンをPCと接続

①. M5Stick-CをPCと接続



③. シリアルポート設定



⑤. 書き込み完了



④. コンパイル



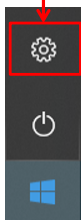
一般社団法人全国専門学校情報教育協会

24

Bluetooth Device の追加(ペアリング)

◇書き込みが完了したら、数秒待って、以下の手順でデバイスのペアリングを行う

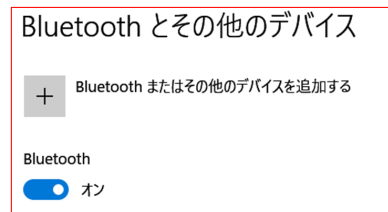
①スタート→設定



②デバイス



③Bluetoothとその他のデバイス



④デバイスを追加する



⑤プログラム指定のデバイス名選択



⑤ペアリング完了



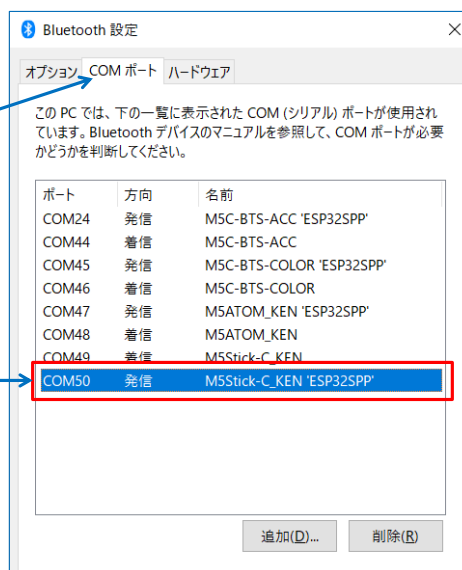
一般社団法人全国専門学校情報教育協会

25

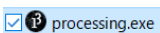
Bluetooth Serial の発信用ポート番号を確認

- ◇スタートボタンから
設定→デバイス→その他のBluetoothオプション
とたどり、【COMポート】タブを選択する
- ◇ペアリングしたBluetoothデバイス名の方向が【発信】と
なっているCOMポート番号を確認しメモする
- ◇メモしたCOMポート番号は、この後のPC側プログラム
(Processing)で使用する

発信のCOMポート



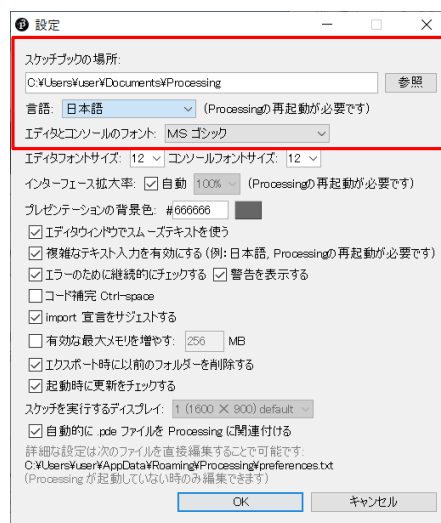
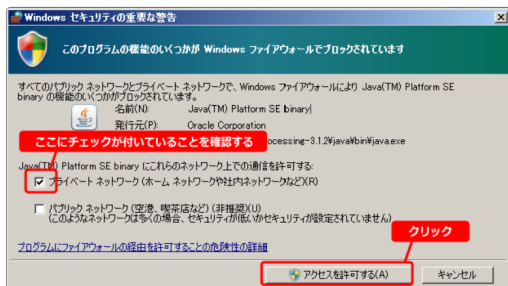
Processing環境構築

- ◇ここで、PC側環境を作る
- ◇<https://processing.org/> にアクセスして使用するPCのOSに応じてファイルをダウンロードする
- ◇ダウンロードしたファイルを解凍してできるフォルダ中の  を実行して、IDEが起動すれば、
環境構築は完了
- ※もしIDEが起動しない場合は、Javaをダウンロードしてインストールする



Processing起動時の警告と日本語環境設定

- ◇左図のような警告ウインドウが表示された場合は、チェックボックスを確認して、【アクセスを許可する】をクリックする
- ◇うまく起動したら、IDEメニューの **ファイル** → **設定** とたどり、右図の言語を日本語に設定し、スケッチブックの場所、フォントを確認する
※言語を変更した場合は再起動する



ソースコード 1/2 (P_M5C_Color_TSC3472_2)

- ◇冒頭から初期化部 (Processing)

```
import processing.serial.*; // Serial通信を利用するためのライブラリ

Serial myPort; // シリアルポートオブジェクト
String str_get_data = null; // 色データ文字列受信変数
String buff[]; // 色データを保管する文字列変数

void setup() // 初期化部
{
    size(500, 500); // 画面に矩形ウインドウを描く
    myPort = new Serial(this, "COM50", 115200); // Bluetooth接続で使用されるCOMポート番号を指定
                                                    // COMポートを速度115200で開く
}

```

ソースコード 2/2 (P_M5C_Color_TSC3472_2)

◇通常処理部 (Processing)

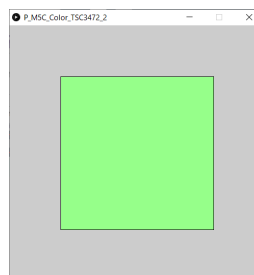
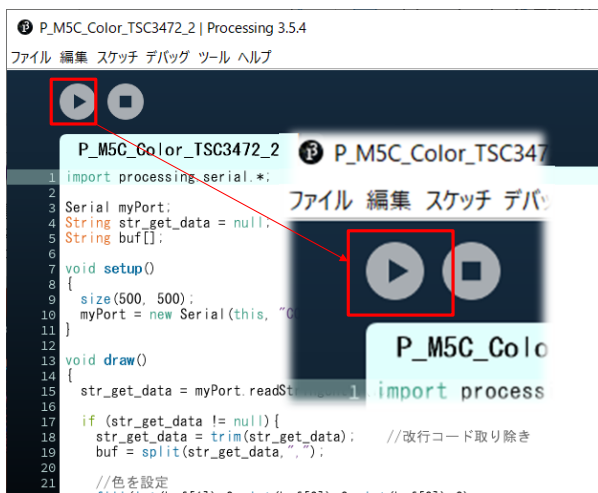
```
void draw()      // 通常処理部
{
  str_get_data = myPort.readStringUntil(10);           // シリアルポートで文字コード10が来るまで受信
                                                       // 10 = 0x0A = ラインフィードコード
  if (str_get_data != null){                          // 順データがNullでなければ処理する
    str_get_data = trim(str_get_data);                // 改行コード取り除き
    buf = split(str_get_data, ",");                  // (,)カンマで分離してbuf[]に保管
                                                       // カンマで分離して0番目はアクセスしない

    //色を設定
    fill(int(buf[1])*2, int(buf[2])*3, int(buf[3])*2); // 保管したRGBデータを整数に変換して、
                                                       // 矩形塗りつぶし色の指定
    rect(100, 100, 300, 300);                        // ウィンドウ内に指定色の矩形を描画する
    println(int(buf[1]),int(buf[2]),int(buf[3]));     // 念のため受け取ったデータを表示する
  }
}
```

念のため受け取ったデータを表示する

動作確認 1/2

- ◇M5Stick-Cはプログラムが書き込まれて、動作スタートしている → ペアリングできれば、動作している
- ◇Processingのソースコードが準備できたら、IDEの左上にある(▶)をクリックして実行する(左図)
- ◇しばらくすると右図のウィンドウが表示される



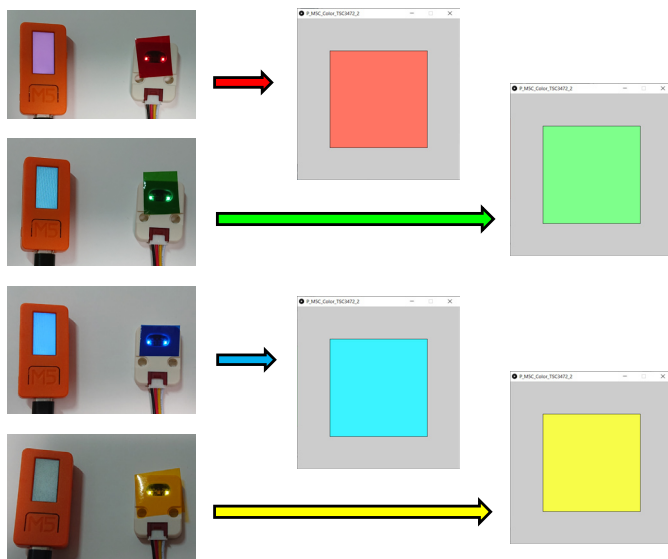


動作確認 2/2

◇センサの感知部窓に色セロファンを載せると、PCに表示された矩形内の色が変わる

◇この時M5Stick-CのLCDの色も変化しているが、色合いの違う様子を観察しておく

これで、光学系のセンシングも可能になってきた！！



2.15 IoTクラウドモデルの開発

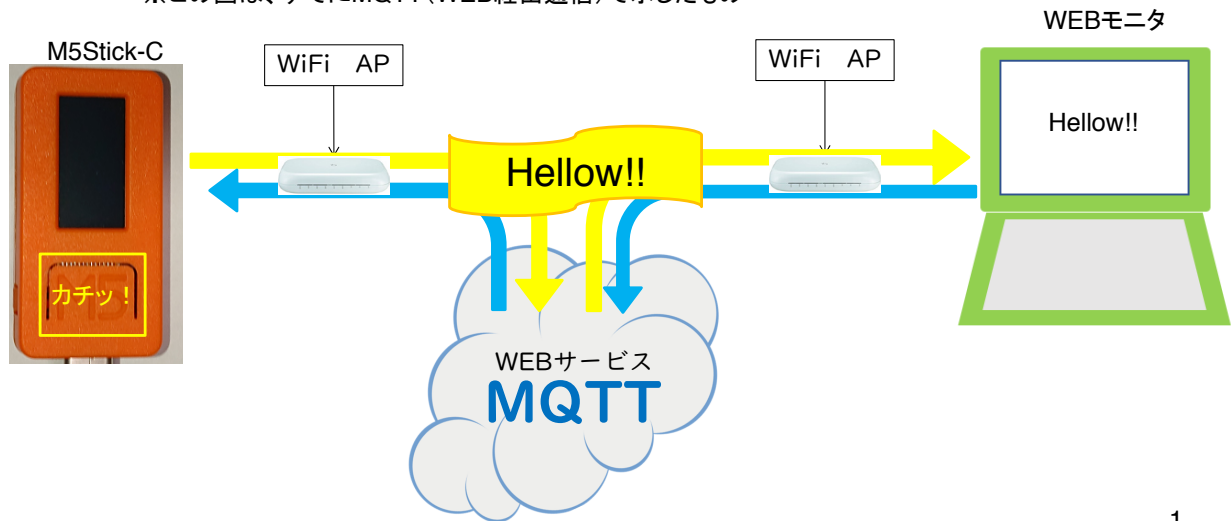
IoTクラウド

IoTクラウドモデル



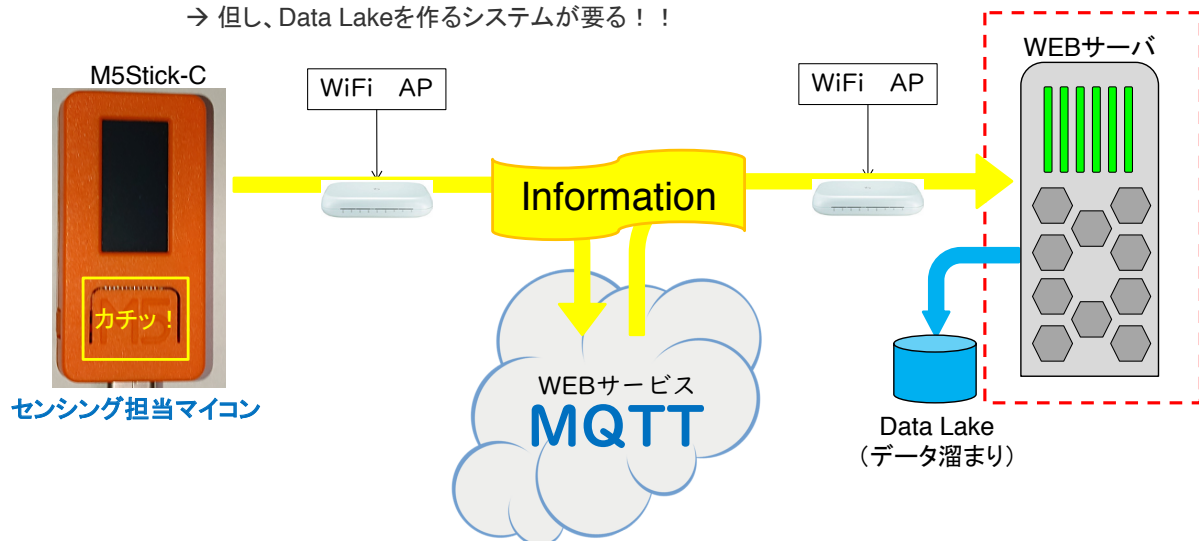
基礎技術 → WEB経由メッセージ交換

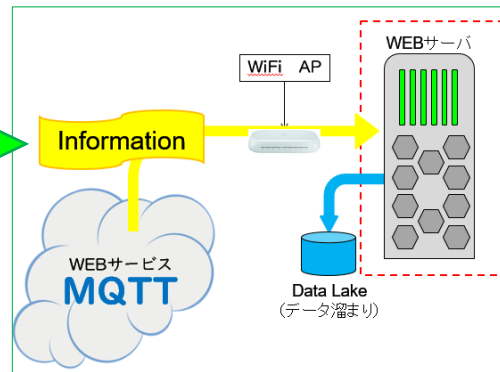
◇図は、M5Stick-Cのボタン押下メッセージWEB経由でモニタしている様子を示している
※この図は、すでにMQTT (WEB経由通信) で示したもの



目論見 → WEBモニタをサーバに置き換える

◇WEBモニタをサーバに置き換えれば、クラウドシステムの構成になる
→ 但し、Data Lakeを作るシステムが要る！！





WEB経由の情報を蓄積するData Lake開発

IOT CLOUD SYSTEM

システム構想

◇IoTクラウドシステムには、WEB上でクライアントから送られた情報を蓄積する役割が必要

◇そこで、WEBから取得するメッセージを蓄積するシステムを開発する

→ 蓄積するメッセージは、MQTTで受信した生のメッセージを蓄積する

→ Data Lakeと言われる

→ 釣った魚を活かしておく【生簀】のようなもの

(魚=Data、後の調理=データ加工 と考える)

◇ソフトウェアの開発には、Node-REDを用いる ← IoTのための【ビジュアルツール】とも呼ばれる

→ 複雑なプログラミングが不要、エッジマイコンでも稼働する

→ 習得に時間がかからない などメリットあり

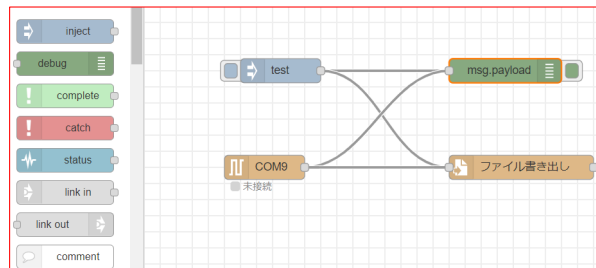
◇開発手順は

①. フローエディタ(右図)に、カプセルを配置

②. カプセルのプロパティを設定

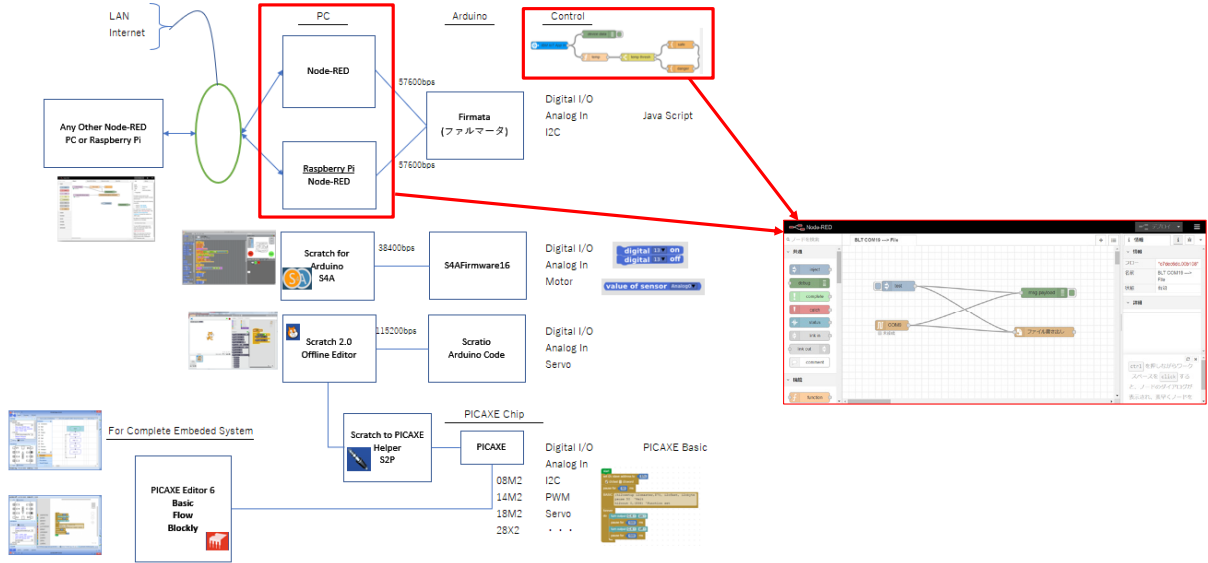
③. カプセル間を接続する

※このカプセルのことを【ノード】と呼ぶ



Node-REDと他のIDEとの関連

出典：Visual Programming Environment for Embedded System 2016.5.21 kenichi.harada



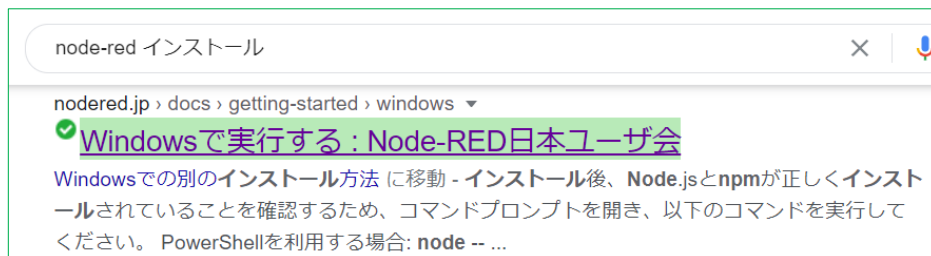
一般社団法人全国専門学校情報教育協会

5

Node-RED環境構築

◇Node-REDを稼働させるために、**2つのシステムをインストール**する

- ①. Node.js ← WEBからダウンロードしてインストールする → <https://nodejs.org/ja/>
- ②. Node-RED ← Node.jsの後に、コマンドプロンプトからインストールする



一般社団法人全国専門学校情報教育協会

6

Node.js インストール

◇Node.js は、次のURLからダウンロードできる

→ <https://nodejs.org/ja/> で**推奨版をダウンロード**して、そのファイルを実行する



◇コマンドプロンプトで下図のように入力して、バージョンが確認できればインストールは成功している

```
C:\Users\ken>node --version && npm --version
v12.18.4
6.14.6
```

Node-REDインストール

◇Node.js のバージョンを確認したコマンドプロンプトで、そのまま下図のように実行すれば Node-REDがインストールされる

※インターネット環境によっては、完了までに時間がかかる場合がある

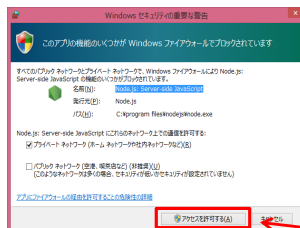
```
C:\Users\ken>npm install -g --unsafe-perm node-red
```

◇インストールが完了したら、Node-REDを起動してみよう

下図のようにしてnode-redを起動する → **終了する際は、Ctrl+C を押下する**

```
C:\Users\ken>node-red
```

◇Node-REDを起動すると、図のような警告メッセージが表示されるかもしれないが、**【アクセスを許可する】**を選択する

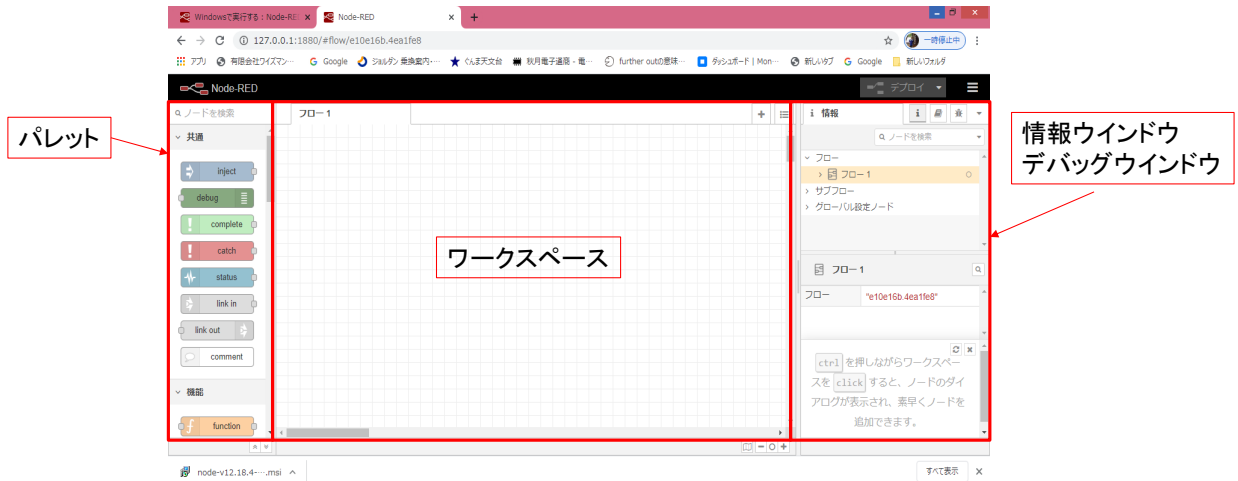


◇Node-REDが起動すると、コマンドプロンプトに下図のようにメッセージが表示される **ブラウザで該当のURLを開く**

```
29 Sep 12:22:44 - [info] Server now running at http://127.0.0.1:1880/
29 Sep 12:22:44 - [info] Starting flows
29 Sep 12:22:44 - [info] Started flows
```

Node-RED 概観

◇Node-REDのウィンドウは、左側にカプセルのような【ノード】があるパレットがあり、中央はワークスペース、右側が情報ウィンドウ・デバッグウィンドウなどとなっている



ノード

◇ノードは以下のものが標準で含まれているが、他にも多くのノードが開発され公開されている
 ◇ノードは、容易に追加・削除できる ※**赤枠**は今回使用するノード



メッセージ購読からData Lakeまで

◇MQTTブローカからメッセージを購読し、Data Lakeに蓄積保存するシステム

- ①. 準備されたフロー (Node-REDのプログラム)を読み込む
ウィンドウ右上の**3本線のアイコン**をクリックする
- ②. プルダウンから**【読み込み】**を選択し、実習キットCD内のSource Codeフォルダにある **flows (indent).json** を読み込む
→ 下図のようなフローが読み込まれる



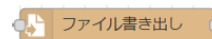
→ これが Node-RED のプログラム【フロー】である



file ノードの設定

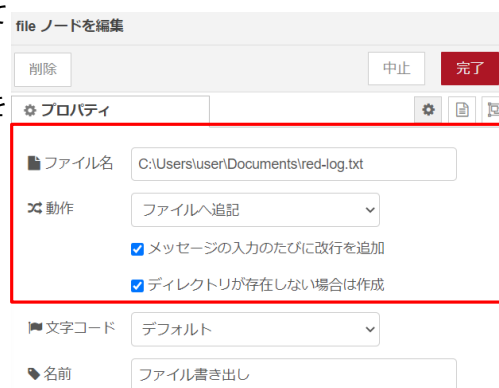
◇既に動作確認済みのフローでも、環境が異なると動作が保証されないので、環境に影響するノードの設定を行う

◇フロー右下のファイル書き出しノードをダブルクリックする →




- ①. fileノードのプロパティが開くので、Data Lakeとして働く**【ファイル名】をフルパスで記述**する
※この際、フォルダが存在しない場合は、下にある**【ディレクトリが存在しない場合は作成】にチェック**を入れる
- ②. 動作は**【ファイルへ追記】**を選択する

◇**【完了】**をクリックして保存する



mqtt in ノードの設定

◇フロー左下のMQTT Messageノードをダブルクリックする → 

- ①. mqtt inノードのプロパティが開くので、サーバとトピックを右図のように設定し、サーバの右にある鉛筆マークをクリックする
- ②. 接続タブの【サーバ】に【broker.shiftr.io】、【ポート】は【1883】
- ③. セキュリティタブの【ユーザ名】を【try】、【パスワード】を【try】にそれぞれ設定する



◇右上の【更新】をクリック 

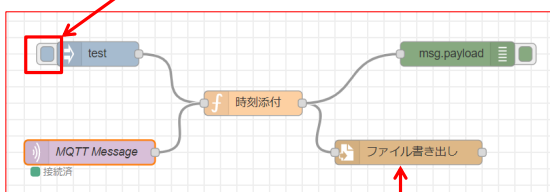
◇【完了】をクリックして保存する

◇ウインドウ右上の【デプロイ】ボタンをクリックすると変更が反映される → 

フローの動作テスト

◇右側デバッグウインドウ【虫マーク → ごみ箱ボタン】とクリックし、デバッグメッセージを空にする

◇test ノードの左端のボタンをクリックする



◇testボタンが押下された日付・時刻付きメッセージがローカルにデバッグノード(フロー右上)に送られて、その出力がデバッグウインドウに表示されると同時に、【ファイル書き出し】ノードでData Lakeが作成されている



MQTTメッセージ蓄積保存の動作テスト

- ◇IoTクラウド側は、Node-REDをそのまま稼働させておく
- ◇M5Stick-CのSW押下状態をWEB経由で送信した際のプログラムで、SW押下状態を記録してみる
 - **既に開発済みの【M5C_MQTT_1】プログラムをM5Stick-Cに書き込む**
 - 書き込み完了後、数十秒で、WiFi~MQTT Brokerへの接続が完了している

- ◇ここで、Node-REDのデバッグウィンドウと、IDEのシリアルモニタ両方を視野に入れて M5Stick-C の Aボタン・Bボタンをそれぞれ押す!!
 - 図のようなメッセージがシリアルモニタ・デバッグウィンドウに表示される

- ◇同時にData Lakeとなるファイルに下図のように**ボタン押下履歴**が**日付・時刻付き**で保存される

```
2020/09/29 15:52:32,test↓
2020/09/29 16:18:36,Button A was pressed !!
2020/09/29 16:18:42,Button B was pressed !!
[EOF]
```

```
COM10
OK
WiFi Connecting.
---> Connected : 192.168.0.72
Mqtt Reconnecting
Mqtt Reconnecting
Mqtt Connected
Send message (Button A was pressed!!)
Send message (Button B was pressed!!)
```

```
MQTT デバッグ
2020/09/29 15:52:33 node: f77f7bc.a55a788
msg.payload: string[24]
"2020/09/29 15:52:32,test"
2020/09/29 16:18:36 node: f77f7bc.a55a788
qas/123 : msg.payload : string[43]
"2020/09/29 16:18:36,Button A was pressed !!"
2020/09/29 16:18:42 node: f77f7bc.a55a788
qas/123 : msg.payload : string[43]
"2020/09/29 16:18:42,Button B was pressed !!"
```

このファイルをExcelで開き内容を確認せよ!! → Data Lake 分析は容易である

MQTTメッセージに日付時刻を追加している

- ◇M5Stick-Cが発行しているメッセージは、“Button A was pressed !!” のように、日付・時刻は無い
- ◇Node-REDのフローの中で購読した MQTT メッセージに日付・時刻を付加している
- ◇その処理を行っているのは、**【時刻添付】ノード**である
 - ※時刻添付ノードをダブルクリックすると、その処理内容が見える(下図)

```
var now = new Date();
var nowStr =
  "" + now.getFullYear() +
  "/" + ('0' + (now.getMonth() + 1)).slice(-2) +
  "/" + ('0' + now.getDate()).slice(-2) +
  " " + ('0' + now.getHours()).slice(-2) +
  ":" + ('0' + now.getMinutes()).slice(-2) +
  ":" + ('0' + now.getSeconds()).slice(-2) + ",";

msg.payload =
  // 日付
  nowStr + msg.payload;

return msg;
```

これは【JavaScript】である



2.16 自由なデータ分析 Data Lake の利用

- ◇このシステムでは、指定フォルダにMQTTから購読したメッセージを、日付時刻付きで記録している
- ◇システムがネットワークに接続されていれば、共有フォルダにこれを記録することもできる
→ フローを改良して、特定メッセージを受信したときに、該当ファイルを共有フォルダにコピーするなど・・・

◇ファイルの拡張子、txt を csv に変更して、Excelで開けば、図のように記録されたData Lakeの内容が見える

	A	B	C
1	2020/9/26 19:11	test	
2	2020/9/26 19:12	Button A was pressed !!	
3	2020/9/26 19:12	Button B was pressed !!	
4	2020/9/26 19:13	test	
5	2020/9/29 13:12	Hello world	
6	2020/9/29 13:17	Hello world	
7	2020/9/29 15:52	test	
8	2020/9/29 16:18	Button A was pressed !!	
9	2020/9/29 16:18	Button B was pressed !!	
10			

◇以後の解析方法は、このまま分析を行っても良いし、さらにDBへExportして別システムで解析してもよい
※利用の方法は、数多くある

◇メッセージをNode-REDのフローで、DBに追加する方法もある
しかし、DBにレコードを追加する際Indexなども追加されるので、BigDataとなりやすいIoTセンシングデータを逐次DBに追加するのは、無駄がある

◇その点、Data Lake はメッセージをそのままテキストで保存するので、データが増えても処理を行うホストの負担が軽いというメリットがある

まとめ

- ◇マイコン発生のメッセージをWEB経由(MQTT)でクラウドホストに伝達することができた
- ◇クラウドホストまでの間では、マイコンの無線通信の一手法として Bluetooth が利用できる
- ◇クラウドホストは、フロープログラミングで構築できる
- ◇Data Lake が単純な構造であることが分かった

※ Big Data となり得る IoTセンサ情報を、常時 Data Base に追加保存するのは無駄である
→ 実験したように、Data Lake に泳がせておいて、バッチなどで、必要なデータを抽出して Data Base に格納し、その後の作業を軽く行うのが理想的

- ◇MQTT メッセージは、マイコン側で購読も可能(M5ATOMで実験した)なので、M5Stick-Cでメッセージの簡易モニタも開発できる
※ 搭載のカラーLCDは小さいが、小さくてもそこに表示ができれば、後で外付けの大型表示器を用いればよい

◇Big DataとなりやすいIoTのデータは、直接DBに追加せず、Data Lakeで活かしておき、必要な時に必要なデータだけをDBに取り出して処理するのが良い

令和2年度「専修学校による地域産業中核的人材養成事業」
情報通信技術に対応した組込みシステム開発技術者育成のモデルカリキュラム開発と実証事業

教員用講義資料

令和3年2月

一般社団法人全国専門学校情報教育協会
〒164-0003 東京都中野区東中野 1-57-8 辻沢ビル 3F
電話：03-5332-5081 FAX 03-5332-5083

●本書の内容を無断で転記、掲載することは禁じます。