

# VS-RC202 チュートリアル

## 目次

VS-RC202 チュートリアル.....	1
1. はじめに.....	3
2. ご注意.....	3
3. 各種ボタンと入出力ピン配置.....	3
4. ソフトウェアのセットアップ.....	4
(A)USB-UART 変換チップのドライバをインストールする.....	4
(B)Arduino IDE のダウンロード.....	6
(C)VS-RC202 を Arduino IDE でプログラムできるようにする.....	8
(D)VS-RC202 にプログラム以外のファイルを書き込めるようにする.....	10
(E)VS-RC202 のライブラリを使用できるようにする.....	12
5. 基本的な使い方.....	13
(A)電源について.....	13
(B)プログラムの書き方.....	14
(C)リセット.....	18
(D)ブートモード.....	18
6. サーボモータの使い方.....	19
(A)サーボモータピンの配置.....	19
(B)サーボモータを動かす.....	20
(C)複数のサーボモータを動かす.....	23
(D)サーボモータの動作の仕組み.....	25
(E)モーションを再生する.....	28
(F)サーボモータを脱力させる.....	33
(G)サーボモータのオフセットを設定する.....	35
(H)サーボモータの限界角度を設定する.....	37
(I)一通りの処理を入れる.....	39
7. LED モード.....	41
(A)LED モードとは？.....	41
(B)LED の接続.....	41
(C)LED をサーボモータのように制御する.....	42
(D)LED の輝度を直接制御する.....	44
8. ブザーモード.....	46

(A)ブザーの仕組み.....	46
(B)ブザーの使い方.....	46
9. センサの使い方.....	48
(A)アナログセンサ値の読み込み.....	49
(B)プルアップ抵抗を有効にする.....	50
(C)超音波センサの値の読み込み.....	51
10. スマートフォンから操作する.....	52
(A)ルータに接続して動かす.....	52
(B)ブラウザからのコマンドを受け付ける.....	55
(C)ブラウザにデータを送る.....	56
11. メモリマップを直接操作する.....	58
(A)メモリマップとは？.....	58
(B)メモリマップの書込.....	59
(C)メモリマップの読出.....	61

## 1. はじめに

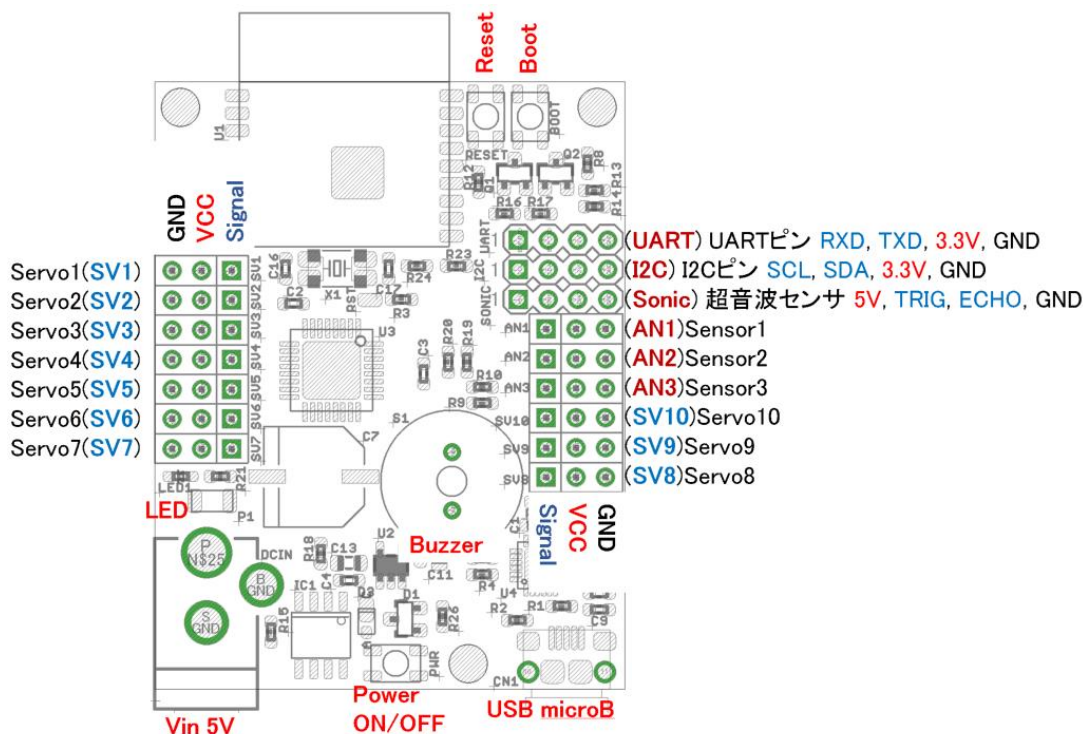
本書は、ESP-WROOM-02を搭載したワイヤレスロボット制御ボード「VS-RC202」のチュートリアル集です。VS-RC202 のプログラミングの学習にご使用ください。VS-RC202 の詳しい仕様について知りたい場合は VS-RC202 取扱説明書をご参照ください。

## 2. ご注意

本製品を取り扱う際には、注意事項に従い正しくお使いください。

- VS-RC202(以降 本ボード)に強い衝撃を与えないでください。
- 本ボードを水に濡らしたり、湿気やほこりの多い場所で使用したりしないでください。ショートなどによる故障が発生する恐れがあります。
- 本ボードから煙が発生した場合、すぐに電源をお切りください。
- 本ボードを幼児の近くで使用したり、幼児の手の届くところに保管したりしないでください。
- 動作中、基板上の素子が高温になることがありますので、絶対に触れないでください。
- 基板上の端子(金属部分)に触れると静電気により故障する恐れがあります。必ず基板の縁を触るようにしてください。
- 基板上の端子同士が金属などでショートすると、過電流により故障する可能性があります。

## 3. 各種ボタンと入出力ピン配置



[注 1]サーボモータと超音波センサの VCC と Vin は直結です。

[注 2]USB 給電がある場合は電源 OFF にできません。

## 4. ソフトウェアのセットアップ

Windows を例に PC で VS-RC202 を使用するためのセットアップ方法を説明します。2018 年 1 月 26 日時点の情報を基にしています。

### (A)USB-UART 変換チップのドライバをインストールする

VS-RC202 を USB で PC に接続して、認識できるようにするためにドライバをインストールします。下記の URL から、Windows 用ドライバをダウンロードします。

<https://jp.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

詳細 ▾ 製品 ▾ ソリューション ▾ コミュニティ&サポート ▾

Silicon Labs » 製品 » 開発ツール » ソフトウェア » USB - UARTブリッジVCPドライバ

## CP210x USB - UARTブリッジVCPドライバ

CP210x USB - UARTブリッジ仮想COMポート (VCP) ドライバは、CP210x 製品とのホスト通信を容易にするための仮想COMポートとしてのデバイス動作に必要です。これらのデバイスも、**ダイレクト・アクセス・ドライバ**を用いてホストと接続できます。このドライバは、アプリケーション・ノート 197 で詳述するスタティック例となります。以下は CP210x 用シリアル通信ガイドでダウンロードした例です。

AN197 : CP210x のシリアル通信ガイド

### ソフトウェアをダウンロード

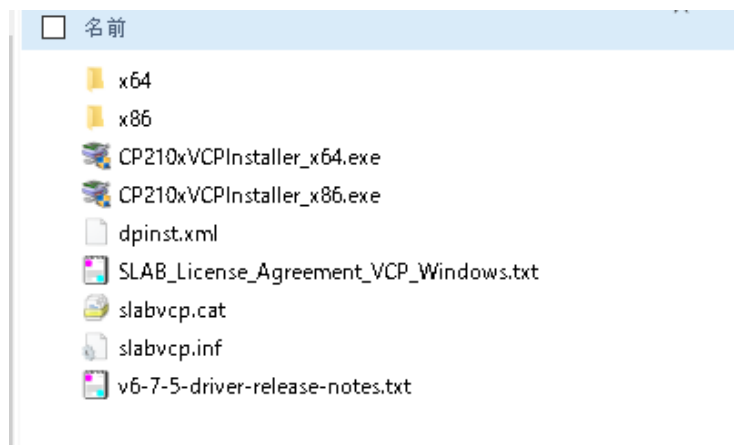
CP210x 製造 DLL およびランタイム DLL が更新されたため、CP210x Windows VCP ドライバ v6.0 以降ではこちらをご使用ください。影響を受けるアプリケーション・ノート・ソフトウェアのダウンロードは AN144SW.zip、AN205SW.zip および AN223SW.zip となっています。5x ドライバを使用していてサポートが必要な場合、アーカイブ版アプリケーション・ノート・ソフトウェアをダウンロードしてください。

[レガシー OS ソフトウェアおよびドライバ、パッケージのダウンロード、リンクおよびサポート情報 >](#)

### Download for Windows 7/8/8.1/10 (v6.7.5)

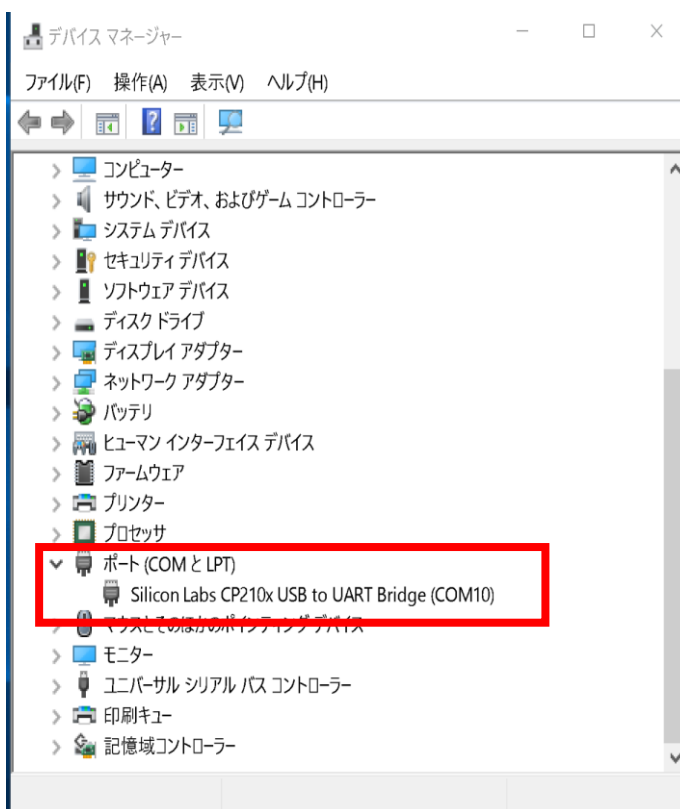
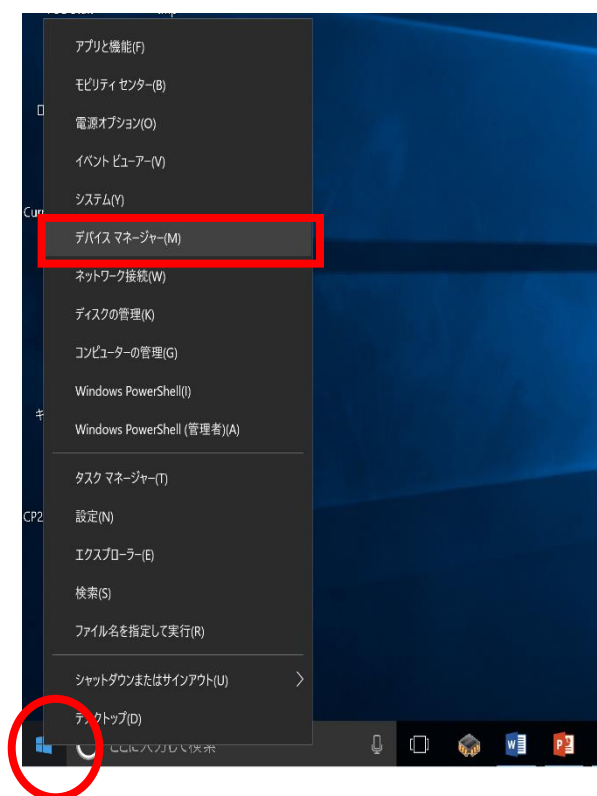
プラットフォーム	ソフトウェア	リリース・ノート
Windows 7/8/8.1/10	<b>VCPをダウンロード(5.3 MB)(デフォルト)</b>	VCP の改訂履歴をダウンロード
Windows 7/8/8.1/10	シリアル・エミュレーションによって VCP をダウンロード(5.3 MB) <a href="#">詳細はこちら。</a>	VCP の改訂履歴をダウンロード

CP210x\_Windows\_Drivers.zip を解凍すると、以下のファイルができるので、**64bitOS を使用している場合は CP210xVCPInstaller\_x64.exe** をダブルクリックしてインストールします。**32bitOS を使用している場合は CP210xVCPInstaller\_x86.exe** をダブルクリックしてインストールします。



ドライバをインストールしたら、VS-RC202 を USBmicroB ケーブルで PC に接続してデバイスマネージャーを確認します。Windows10 の場合は、スタートボタン上で右クリックして、デバイスマネージャーを選択します。

デバイスマネージャーウィンドウのポート(COMとLPT)に、**Silicon Labs CP210x USB to UART Bridge(COMxx)**と表示されていれば正常にデバイスを認識しています。



## (B)Arduino IDE のダウンロード

ピッコロボ IoT 付属の基板 VS-RC202 は Arduino IDE でソフトでプログラムを作成します。次は Arduino IDE をインストールします。下記の URL にアクセスして Windows Installer を選択します。

<https://www.arduino.cc/en/Main/Software>

HOME BUY SOFTWARE PRODUCTS LEARNING COMMUNITY SUPPORT

SOFTWARE ENGLISH

### Access the Online IDE

**ARDUINO WEB EDITOR**  
Start coding online with the [Arduino Web Editor](#), save your sketches in the cloud, and always have the most up-to-date version of the IDE, including all the contributed libraries and support for new Arduino boards. The Arduino Web Editor is one of the [Arduino Create platform's](#) tools.

[Try It Now](#)  
[Getting Started](#)

### Download the Arduino IDE

**ARDUINO 1.8.5**  
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

- Windows installer** [Admin Install](#)
- Windows app** [Get](#)
- Mac OS X 10.7 Lion or newer**
- Linux 32 bits**
- Linux 64 bits**
- Linux ARM**

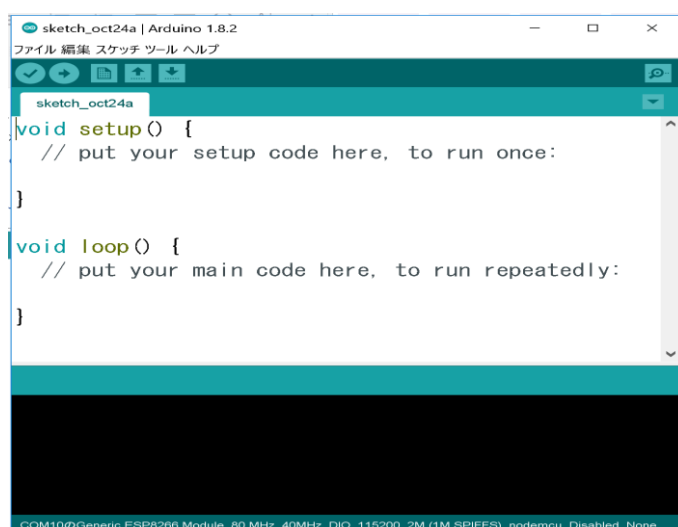
[Release Notes](#)  
[Source Code](#)  
[Checksums \(SHA512\)](#)

Windows Installer を選択すると、以下のような画面が現れるので、JUST DOWNLOAD をクリックして、ダウンロードを開始します。

(注)金額が表示されますが、これは Arduino IDE を開発しているプロジェクトに寄付できるというだけで、JUST DOWNLOAD を選択すれば、料金が発生することなどはありません。



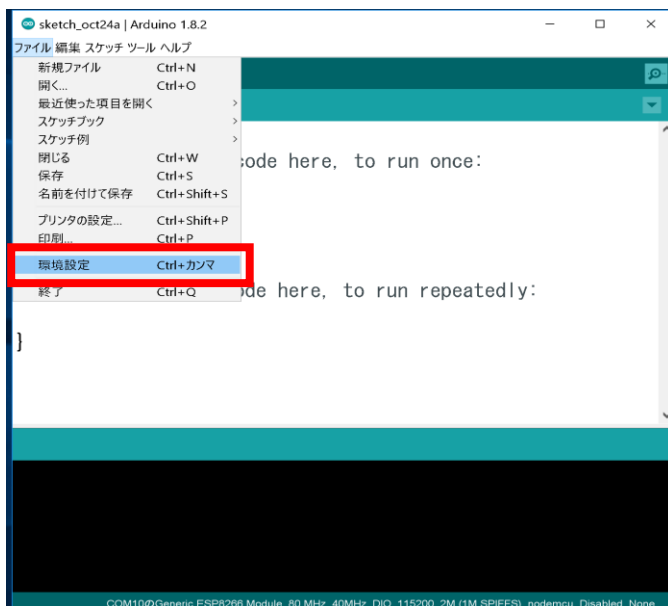
ダウンロードした arduino-xxxx-windows.exe をダブルクリックしてインストーラを起動します。後は画面の指示に従いインストールすれば、準備完了です。デスクトップの Arduino IDE のアイコンをダブルクリックして、以下のウィンドウが表示されれば正常にインストールされています。



## (C)VS-RC202 を Arduino IDE でプログラムできるようにする

VS-RC202 を Arduino IDE でプログラムできるようにするには、追加の設定ファイルをインストールする必要があります。

Arduino IDE を起動して、メニューのファイルから環境設定を選択します。



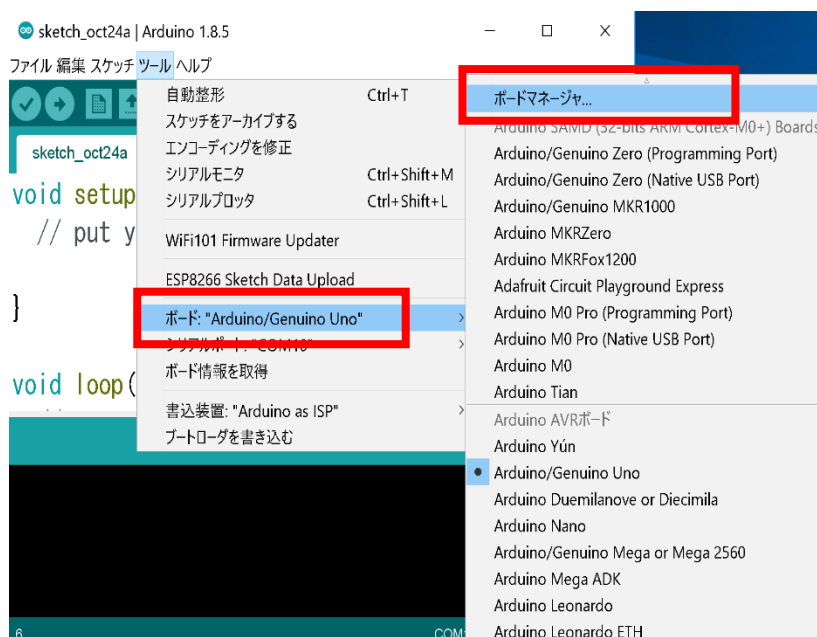
環境設定パネルの追加のボードマネージャ URL に以下の URL を張り付けて、OK ボタンを押します。

[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)





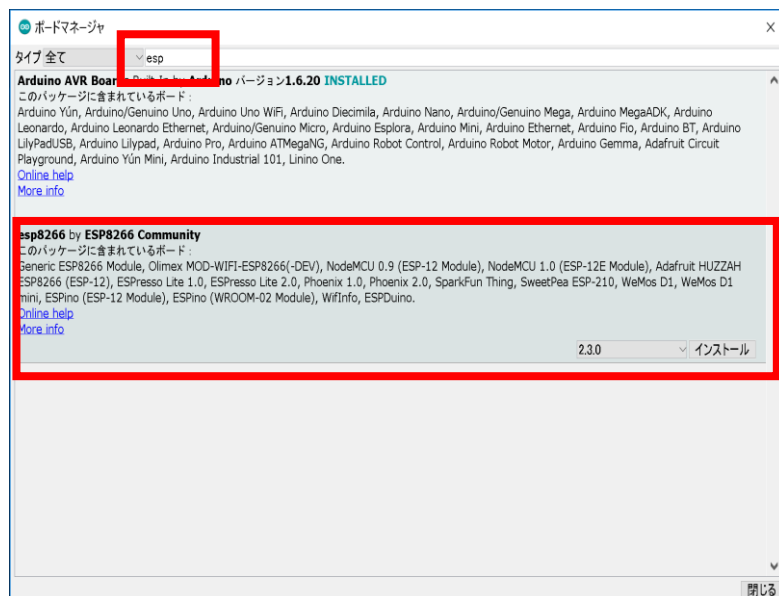
メニューのツールから、ボード>ボードマネージャを選択します。



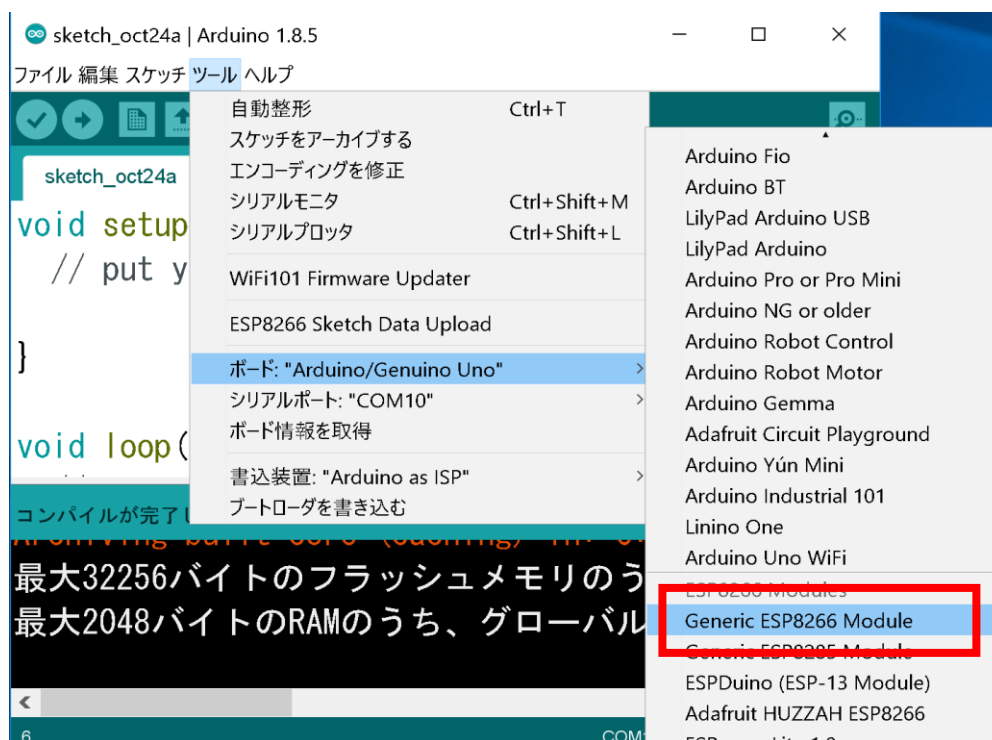
ボードマネージャが表示されたら上部にある検索ボックスで”esp”と入力します。

esp8266 by ESP8266 Community が見つかるので、バージョンを選択して、インストールします。

2019年8月1日現在、2.5.2で動作することを確認しています。



Esp8266 のインストールが終り、メニューのツール>ボードで **Generic ESP8266 Module** が選択できるようになっていれば、正常にインストールされています。(ボード選択画面で下のほうにスクロールすると見つかります)



#### (D)VS-RC202 にプログラム以外のファイルを書き込めるようにする

VS-RC202 は Wi-Fi と接続でき、簡易ウェブサーバーとして稼働し、HTML ファイルなどを配信することができます。ピッコロボ IoT をお使いの場合は、VS-RC202 にスマートフォンのブラウザで VS-RC202 の HTML のコントローラーを表示し、ピッコロボ本体を操作できます。

ここでは配信する HTML 等を事前に VS-RC202 にインストールする準備をします。以下の URL にアクセスし、**ESP8266FS- x x x .zip** をダウンロードします。

※ x x x にはバージョン名が入ります。

2019 年 6 月 13 日現在、0.4.0 で動作することを確認しています。

<https://github.com/esp8266/arduino-esp8266fs-plugin/releases/>

ESP8266FS-×××.zip を解凍すると、ESP8266FS というフォルダができます。Arduino のスケッチフォルダに tools というフォルダを作成して、ESP8266FS を tools の中にコピーします。

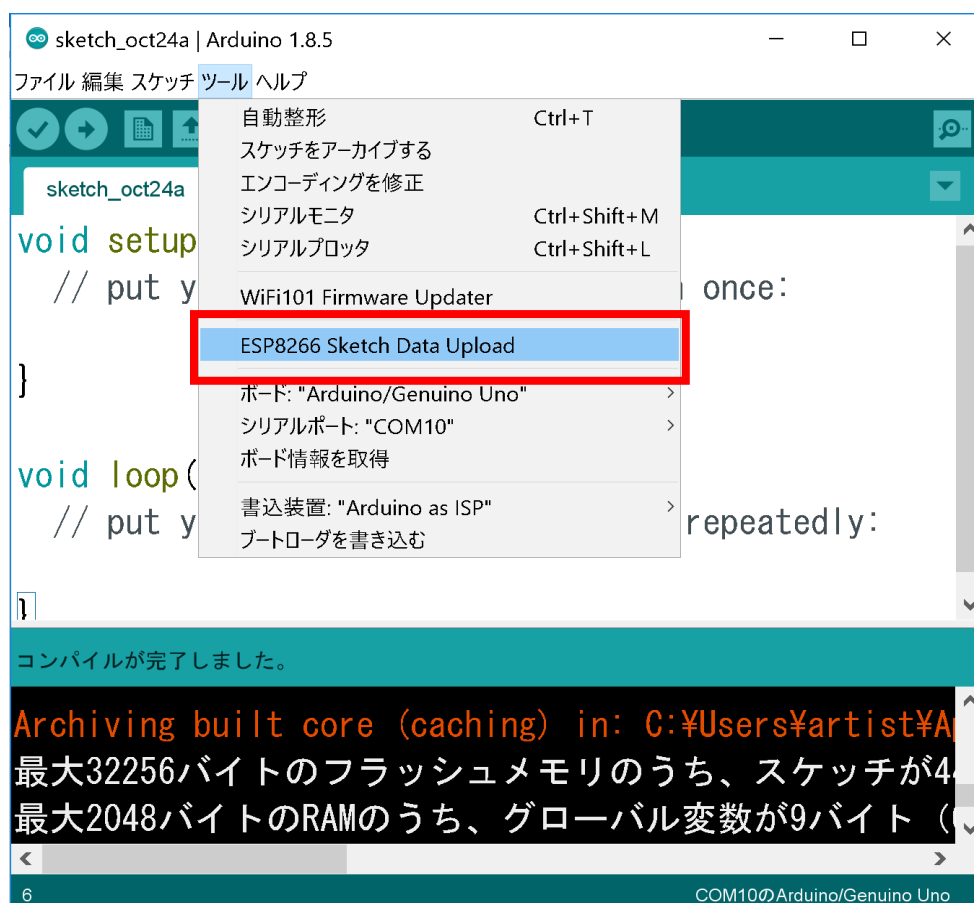
デフォルトでは Arduino のスケッチフォルダはドキュメントフォルダの直下にあります。

C:\Users\username\Documents\Arduino

以下のようにフォルダを作り、コピーします。

C:\Users\username\Documents\Arduino\tools\ESP8266FS\tool\esp8266fs.jar

コピーしたら一度 Arduino IDE を再起動して、メニューのツール>ESP8266 Sketch Data Upload が表示されていれば正常にインストールされています。



## (E)VS-RC202 のライブラリを使用できるようにする

以下の URL から `V-duino-ver.×××.zip` をダウンロードします。

※×××にはバージョン名が入ります。

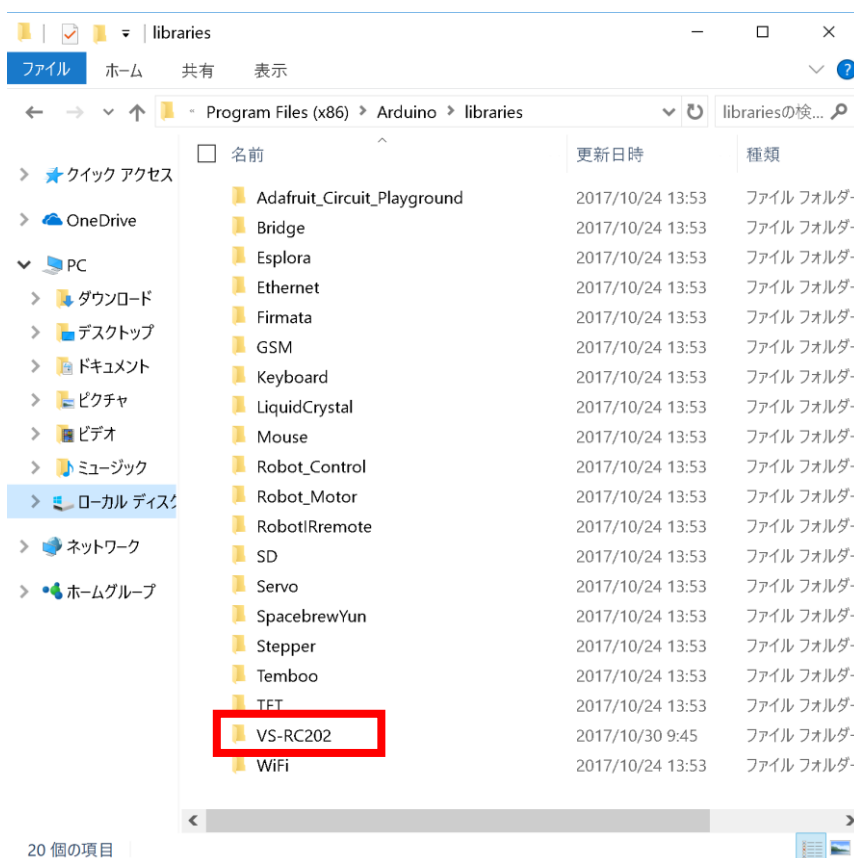
基本的に最新のバージョンを使用してください。

<https://github.com/vstoneofficial/V-duino/releases>

解凍すると、`V-duino-ver.×××` というフォルダができます。本フォルダ内に `VS-RC202` というフォルダがあるので、このフォルダを Arduino の `libraries` フォルダの中に移動させてください。

Windows10 であれば、`libraries` フォルダは以下の位置にあります。

**'C:\Program Files (x86)\Arduino\libraries' もしくは 'C:\Program Files\Arduino\libraries'**



以上で、ソフトウェアのセットアップは完了です。

## 5. 基本的な使い方

ここでは VS-RC202 を使用するにあたり、最低限知っておくべきことを説明します。

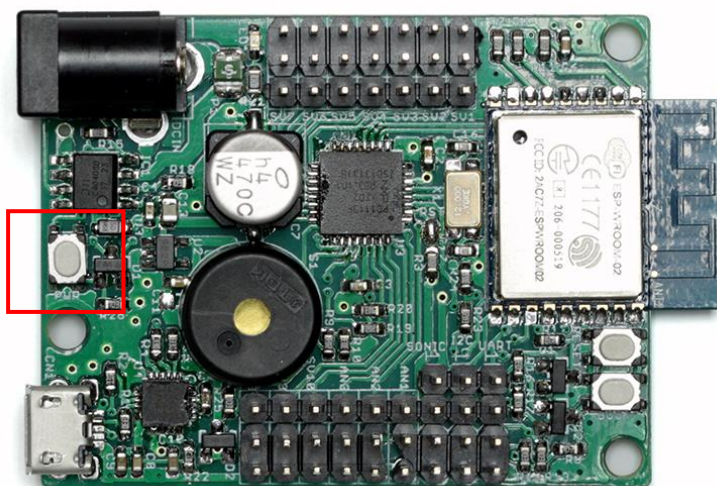
### (A)電源について

VS-RC202 は必ず **5V の AC アダプタ**、もしくは、**ニッケル水素充電電池4本(4.8V)**をご使用ください。5V より上の電源を使用するとボードやサーボモータが故障する原因になるため、絶対に使用しないでください。

VS-RC202 は、電源監視機能が備わっており、バッテリーの過放電を防止するようになっています。デフォルトでは**電源電圧が 4.6V を下回ると電源が落ちます**。

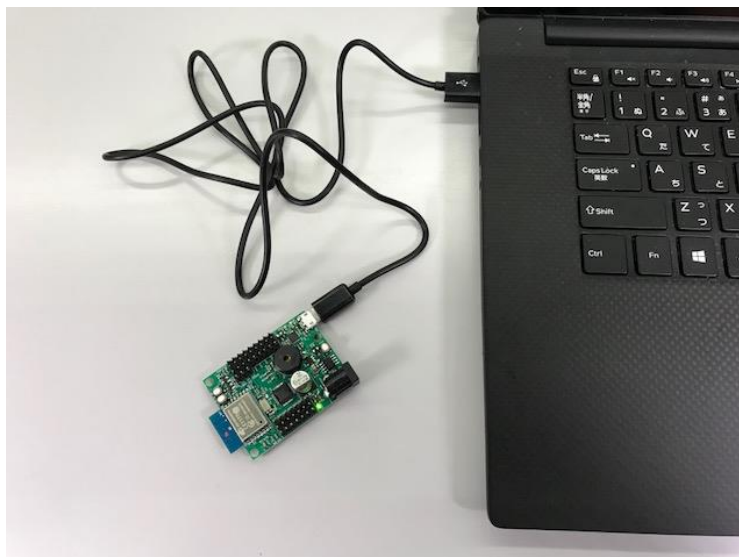
そのため、新品の電池や AC アダプタを使用しているにも関わらず電源が落ちる場合は、負荷に対して、電源出力不足による電圧降下が起こり、結果として電源 OFF 機能が働いている可能性があります。

USB を接続した場合は自動で電源が入りますが、AC アダプタやバッテリーだけで起動する場合は、写真の電源ボタンを押すと、電源が ON になります。電源が ON の状態で、電源ボタンを3秒以上長押ししてから指をはなすと、電源が OFF になります。

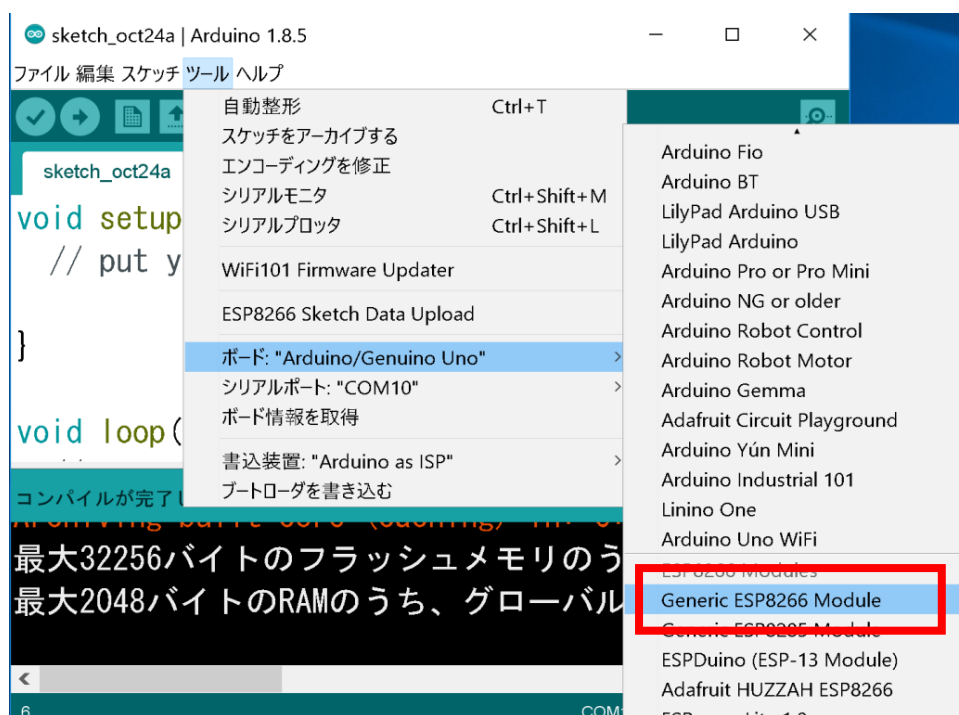


## (B)プログラムの書き方

VS-RC202 を USBmicroB ケーブルで PC と接続します。

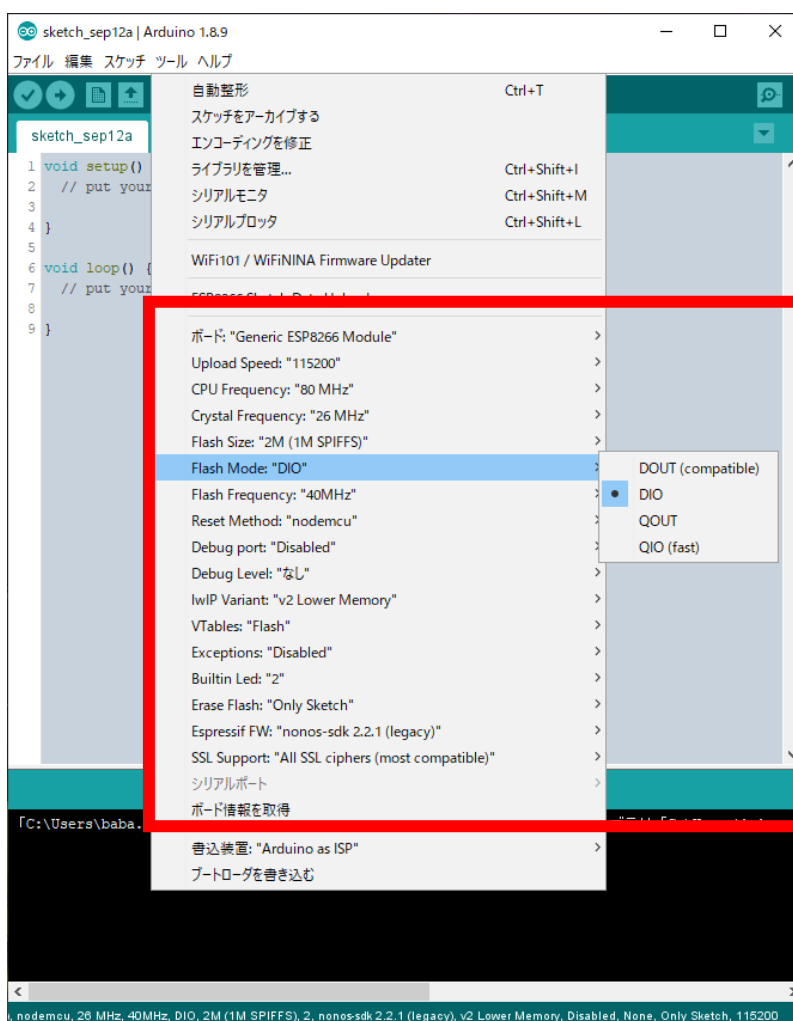


Arduino IDE のメニューのツール>ボードで Generic ESP8266 Module を選択します。

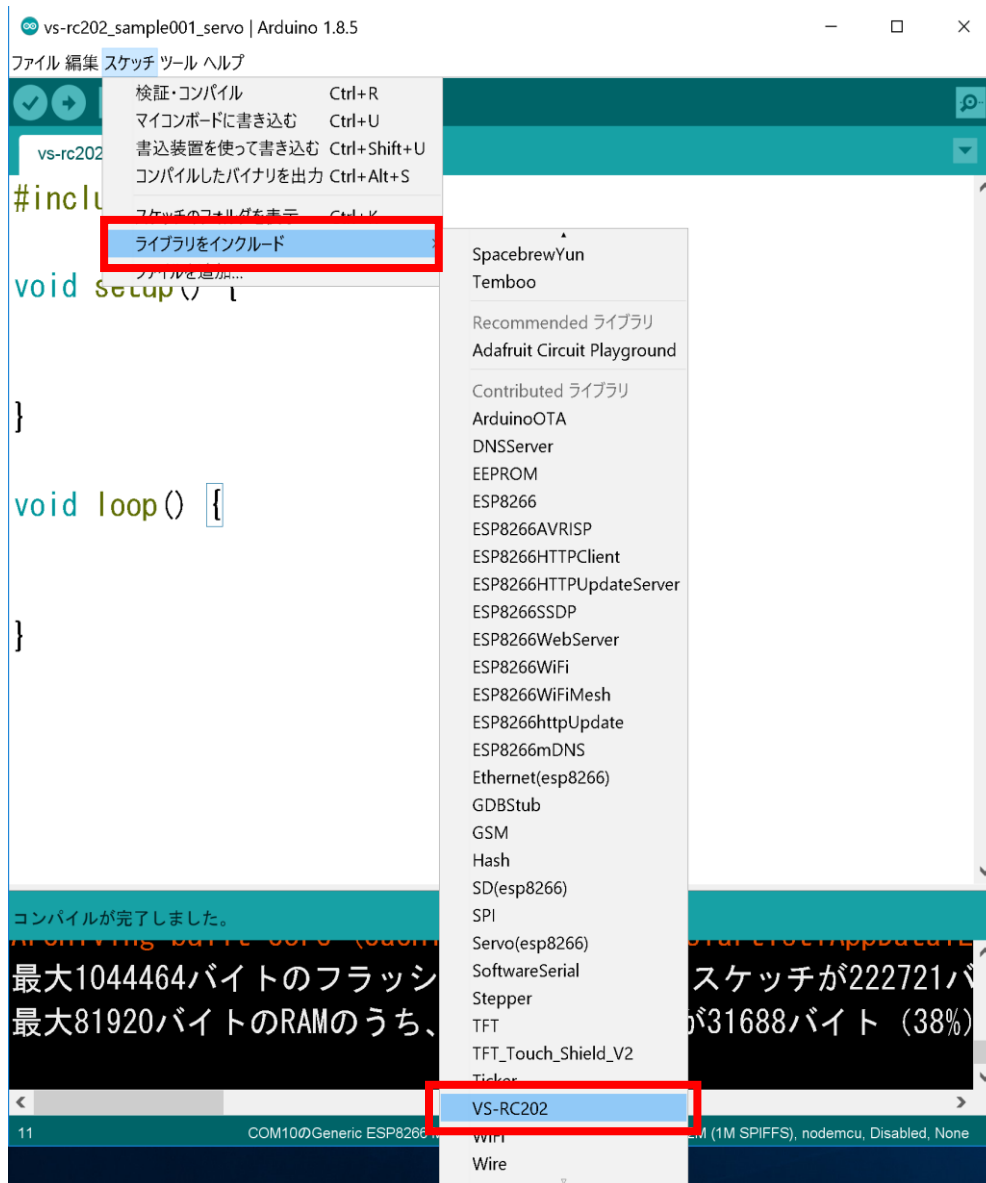


Generic ESP8266 Module を選択すると、メニューのツールに複数の設定がでてくるので、以下のように設定します。一度設定すれば、次回以降は自動的に選択されます。

- Flash Mode : DIO
- Flash Frequency : 40Mhz
- CPU Frequency : 80Mhz
- Crystal Frequency : 26Mhz
- Flash Size : 2M(1M SPIFFS)
- Debug port : disabled
- Debug level : なし
- Reset Method : nodemcu
- Upload Speed : 115200
- シリアルポート : VS-RC202 が接続されているポート

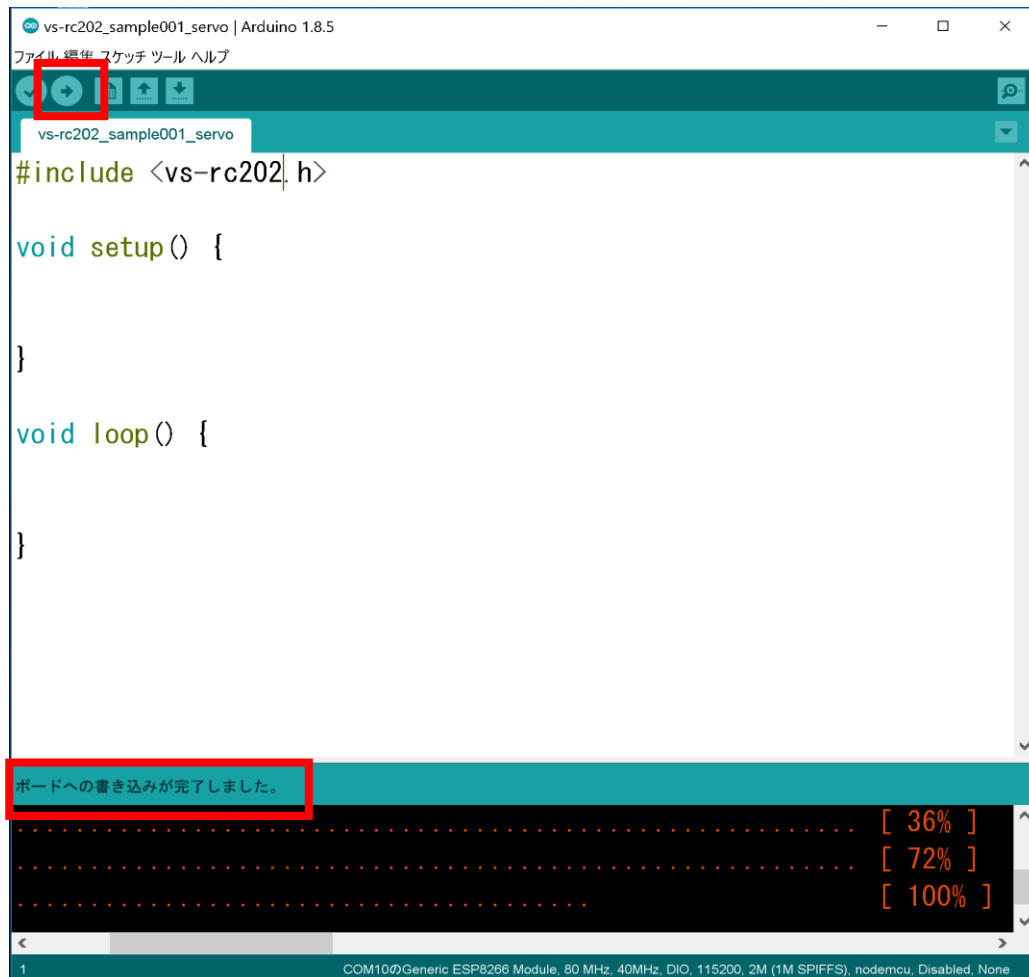


次に、VS-RC202 のライブラリをインクルードします。メニューのスケッチから、ライブラリをインクルード>VS-RC202 を選択するとインクルードされます。VS-RC202 を使う場合は、プロジェクトを作成する毎にライブラリをインクルードしてください。



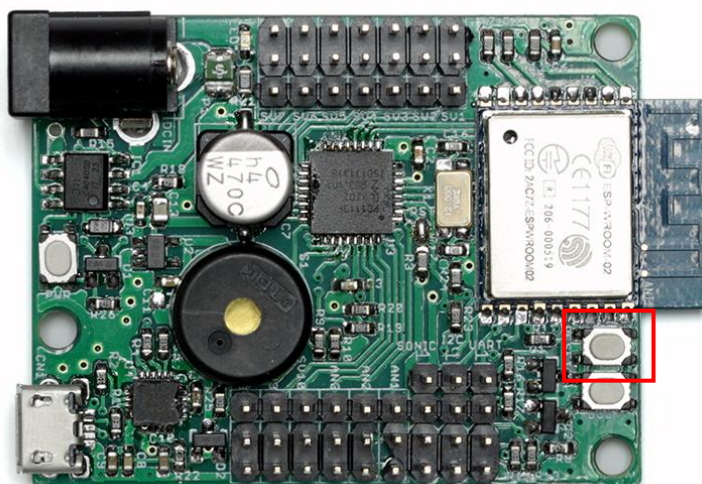


VS-RC202 を PC に接続し、メニューのツールから VS-RC202 のポートを選択した状態で、スケッチ書き込みボタン(右矢印のボタン)を押します。“**ボードへの書き込みが完了しました。**”とメッセージが表示されれば、書き込み成功です。エラーが出る場合は、ボードの設定が間違っているか、ライブラリのインクルードが失敗している可能性があります。再度、手順を確認してください。



### (C)リセット

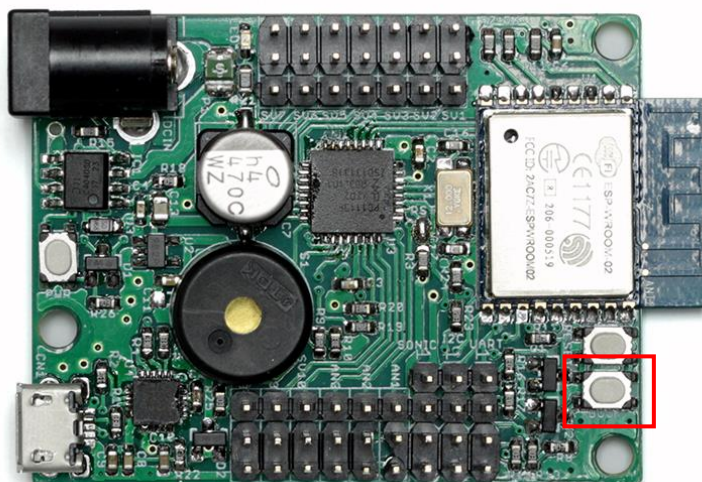
ESP-WROOM-02 の隣にあるボタンはリセットボタンです。リセットボタンを押すと ESP-WROOM-02 が再起動します。プログラムを再度実行したい場合に利用します。



### (D)ブートモード

リセットボタンの隣のボタンは、ブートボタンです。Arduino IDE のツールの設定の Reset Method を”ck”にした場合、ブートボタンを押しながら、リセットボタンを押すとプログラムの書き込みができるようになります(マニュアルでブートモードに入る)。

基板の部品が一部壊れて、”nodemcu”設定でプログラムが書き込めない等の事情がない場合は特に使用する必要はありません。



## 6. サーボモータの使い方

それでは、いよいよサーボモータを動かしてみましょう。

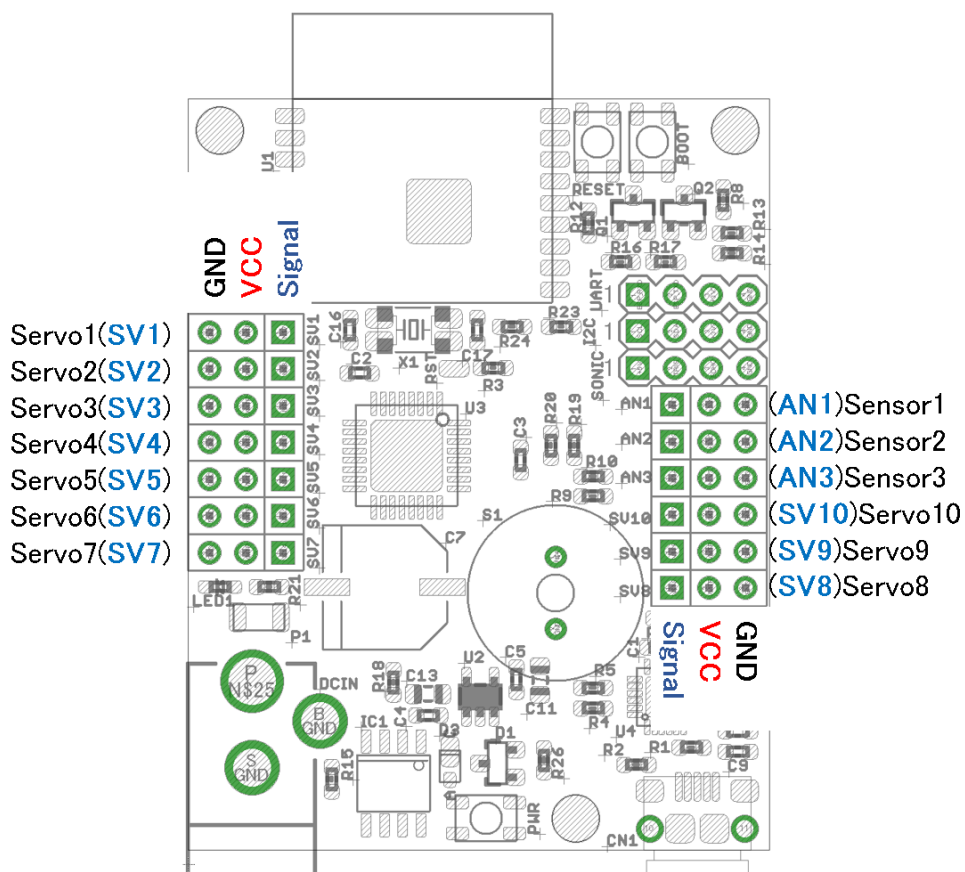
### (A)サーボモータピンの配置

VS-RC202 のピン配置は以下のようになっています。サーボモータを接続する場合は以下の間違いに気を付けてください。故障の原因となります。

- Signal(信号線)と GND を逆向きに接続しないで下さい
- サーボモータピン(SV1-SV10)以外に間違って接続しないで下さい

これ以降の説明では、VS-RC202 の各ピンを下図に記載の名称で記述します。

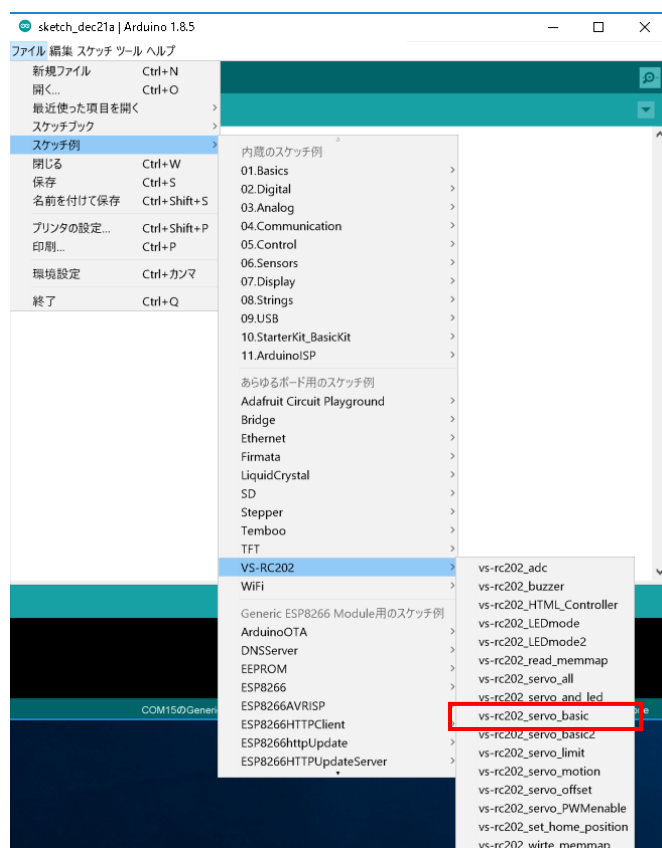
(例) ”SV1 のピンにサーボモータのケーブルを接続してください”



## (B)サーボモータを動かす

さっそくサーボモータを動かしてみましよう。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_basic を選択してください。



スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。

スケッチの書き込みが終わったら、**SV1** にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。そして、電源ボタンを押します。スケッチが正常に書きこめていれば、サーボが左右に動き出します。

それでは、少しスケッチの内容に関して説明します。

まずは、`setup()`の中を見ていきましょう。`setup()`の中は、起動時1回だけ実行されます。通常は設定の読出しや、初期化処理を書きます。必ず `setup()`内でサーボモータの設定等を初期化する `initLib()`を呼び出します。

VS-RC202 は、電源 ONしてもサーボモータは脱力状態にあります。`servoEnable()`関数で PWM 信号を ON にします。

`setServoMovingTime()`関数はサーボモータの遷移時間を指定します。単位はミリ秒です。`setServoMovingTime(1000)`だと、これ以降は1000ミリ秒で目標位置まで動きなさいという意味になります。

```
#include <vs-rc202.h>           //ライブラリのインクルード

void setup() {

  initLib();                    //ライブラリの初期化、必ず最初に実行する
  servoEnable(1, 1);           //SV1 に送る PWM 信号を有効にする
  setServoMovingTime(1000);    //サーボモータの遷移時間を設定する
}
```

#### 【関数の説明】

##### 書式: `initLib()`;

VS-RC202 ライブラリの初期化をする。`Setup()`で一度実行する。

##### 書式: `servoEnable(ピン番号, ONOFF フラグ)`;

例1: `servoEnable(1, 0)`; SV1 の PWM 信号を OFF

例2: `servoEnable(2, 1)`; SV2の PWM 信号を ON

##### 書式: `servoServoMovingTime(ミリ秒)`;

例1: `setServoMovingTime (500)`; 以降は 500ms で目標角度に到達

例2: `setServoMovingTime (1000)`; 以降は 1000ms で目標角度に到達

次に、loop()の中を見ていきます(一部省略しています)。loop()の中は、繰り返し何回も実行させる処理を書きます。今回は、サーボモータを動かしたいので、loop()内にサーボモータの動きを書きます。

まずは、setServoDeg()関数で SV1 に動いてほしい位置を設定します(範囲:-1800~1800)。次にmoveServo()関数でサーボモータを動かします。今回のコードでは、”目標位置を指定して、動きなさい”という命令を繰り返しています。今回はサーボモータが回転しきってから次の命令を送るために、間に delay()関数を挟んでいます。

```
void loop() {  
  
    setServoDeg(1, 0);    //SV1 の目標位置を 0 にする  
    moveServo();         //サーボモータを動かす  
    delay(1200);        //1200 ミリ秒停止する  
    :  
}
```

#### 【関数の説明】

**書式: setServoDeg (ピン番号, 位置);**

例1: setServoDeg(1, 0); //SV1 の目標位置を 0 にする

例2: setServoDeg(2, 1500); //SV2 の目標位置を 1500 にする

**書式: moveServo ();**

この関数を実行すると、サーボモータが動き出します。

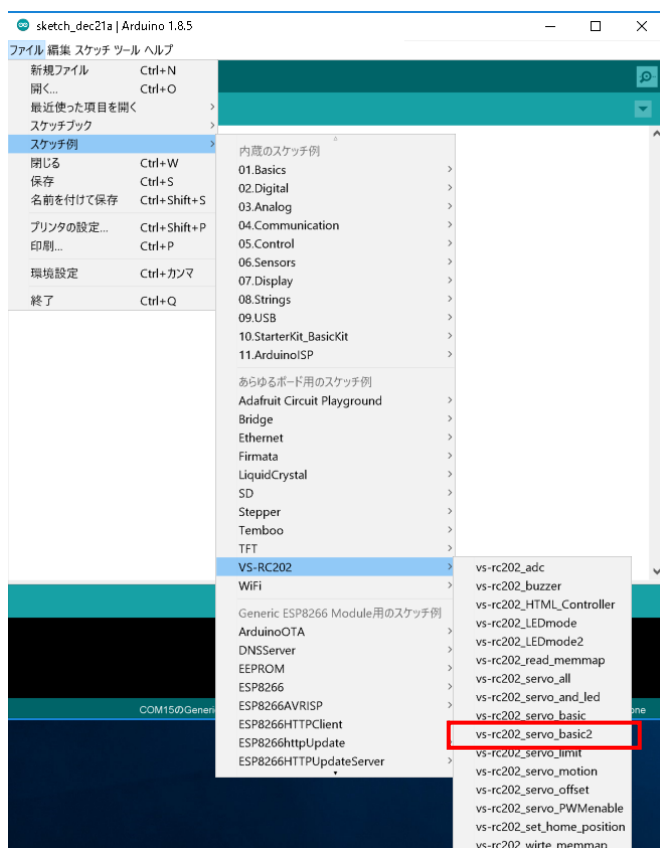
#### 【まとめ】

1. setup()内は初期化処理、loop()内は繰り返し処理
2. setup()内で、ライブラリの初期化、PWM 信号の有効化を行う
3. サーボモータを動かすには、遷移時間の設定、目標角度の設定、開始が必要

### (C)複数のサーボモータを動かす

次に複数のサーボモータを同時に動かしてみましょう。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_basic2 を選択してください。



スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。

スケッチの書き込みが終わったら、SV1, SV2, SV3, SV4 にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。そして、電源ボタンを押します。書き込みが成功していれば、SV1,3 と SV2,4 がそれぞれ、対称の動きをするはずですが。

それでは、少しスケッチの内容に関して説明します。

setup()の中は、前回と同様で、初期化とPWM信号の有効化だけです。唯一の違いは、setServoMovingTime()関数がloop()内に移動していることです。

それでは、loop()を見てみましょう(一部省略しています)。

setServoMovingTime(), setServoDeg(), moveServo()を順番に繰り返していることがわかると思います。毎回 setServoMovingTime()をセットすることにより、サーボモータの移動速度を変化させることができます。

また、setServoDeg()関数で複数のサーボモータの目標位置を入力してから、moveServo()関数を実行することにより、複数のサーボモータを同時に動かすことが可能になります。

```
void loop() {  
  
    setServoMovingTime(500); //Set moving time to the target position  
    setServoDeg(1, 0);      //Set SV1 servo target position  
    setServoDeg(2, 0);      //Set SV2 servo target position  
    setServoDeg(3, 0);      //Set SV3 servo target position  
    setServoDeg(4, 0);      //Set SV4 servo target position  
    moveServo();            //Start to move servo  
  
    :  
}
```

### 【まとめ】

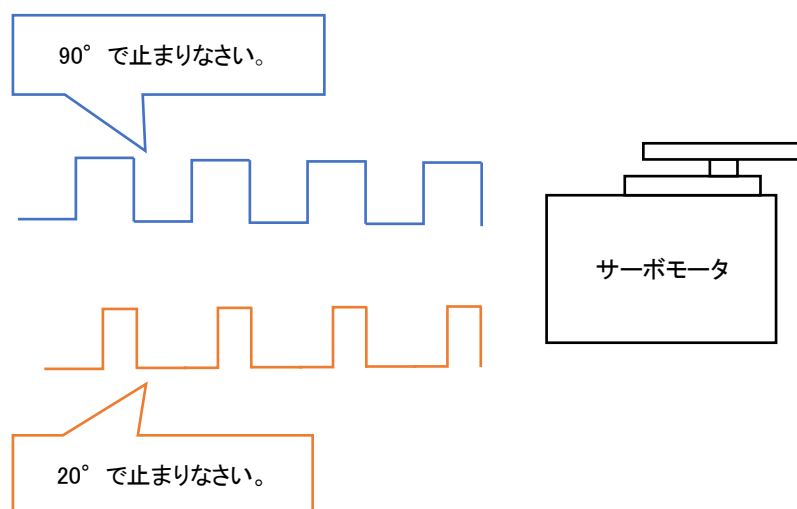
1. setServoMovingTime()関数を毎回実行すれば、サーボの速度を変更できる
2. setServoDeg()関数で複数のサーボモータの設定をして、moveServo()関数を実行すれば、複数のサーボモータを同時に動かせる



## (D)サーボモータの動作の仕組み

ここでは、プログラムをより理解するために、今まで使用したスケッチを基にサーボモータの動作の仕組みを説明します。

サーボモータは PWM 信号と言われるパルスを送ることにより、モータの軸を特定の位置で停止させることができます。このパルスの HIGH の部分の比率(duty 比という)を変えることにより、停止する位置が変化します。



PWM 信号はサーボモータの軸を特定の位置に停止させるものであって、任意の速度で動かすものではありません。もし任意の速度で回転させたい場合は、以下の方法をとります。

サーボモータの PWM 信号の周期は約 20 ミリ秒です。よって、20 ミリ秒ごとに、duty 比をほんの少しずつ変化させていけば、モータの軸の停止位置がわずかにずれていき、ゆっくりサーボモータが回転しているように見えます。

サンプルスケッチの関数を例に説明すると、以下のような流れになります。

```
servoEnable(1, 1)
```

PWM 信号が ON になり、サーボモータの軸は初期位置として、中心位置を向きます。

```
setServoMovingTime(1000)
```

```
setServoDeg(1, 1800)
```

```
moveServo();
```

遷移時間 1000 ミリ秒で、1800 の位置に移動します。

20 ミリ秒ごとに位置が変化するので、

50 回停止位置が変化します。

**(遷移時間 - PWM の周期) = 変化回数**

**1000msec/20msec = 50**

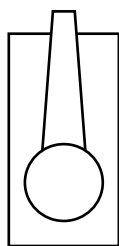
1 回の位置の変化は 36 となります。

**(目標位置 - 開始位置)/変化回数 = 1 回の変化量**

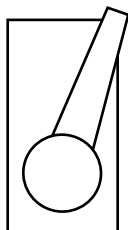
**(1800 - 0)/50 = 36**

(注)VS-RC202 のライブラリでは、細かく位置を制御するため、 $-90\sim 90^\circ$  の角度ではなく、 $-1800\sim 1800$  の数値で位置を指定します。

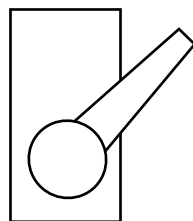
感覚としては、モータを回転させるというよりは、パラパラ漫画のように開始位置と終了位置を決めて、ページをめくるスピードを決めるような感じになります。イメージできたでしょうか？



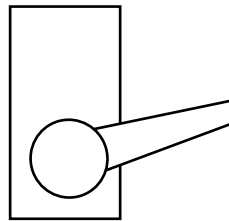
位置:0  
時間:0ms



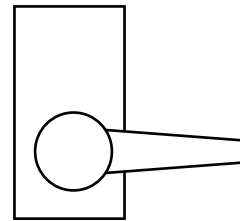
位置:360  
時間:200ms



位置:900  
時間:500ms



位置:1440  
時間:800ms



位置:1800  
時間:1000ms

なお、このように目標位置に向かって少しずつ位置を変化させることを補間といいます。VS-RC202ではメインCPUであるESP-WROOM2ではなく、ARMチップで主な処理を行っています。そのため、ESP-WROOM2はARMチップに目標位置と遷移時間を知らせるだけで、通信処理などに専念できるようになっています。

#### 【まとめ】

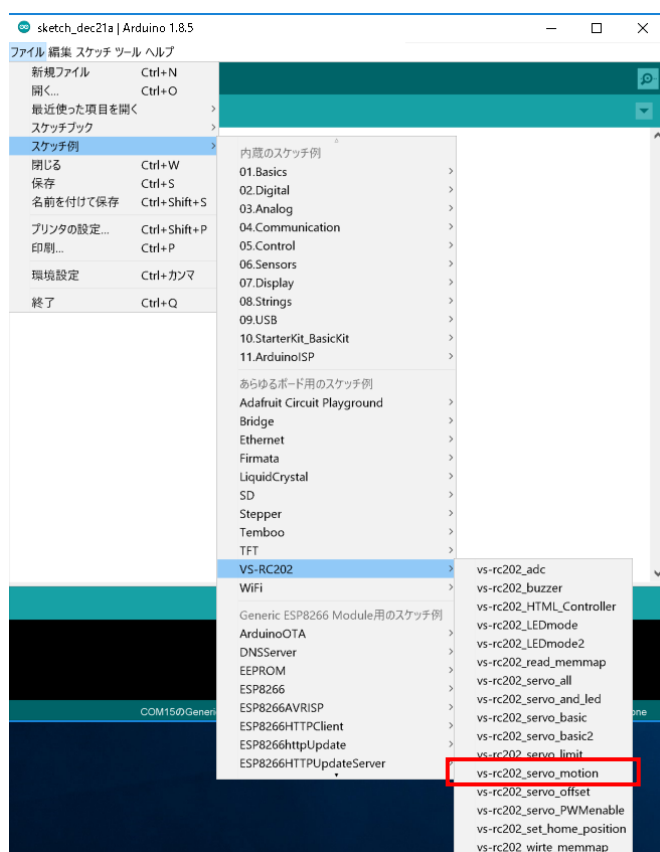
1. サーボモータはPWM信号で動かす
2. サーボモータのPWM信号の周期は20ミリ秒
3. サーボモータの制御は停止位置を少しずつ変化させるパラパラ漫画方式

## (E)モーションを再生する

もう少し踏みこんだサーボモータの制御方法について説明します。皆さんはロボットアームや二足歩行ロボットを見たことはありますか？それらのロボットも基本的には、複数のサーボモータをパラパラ漫画的に動かして、一つの動作を行っています。例えば、二足歩行ロボットなどが歩く場合、全身のサーボモータが次々と角度を変えて、ポーズをとりますよね。このサーボモータの一連の動きを**モーション**といいます。

ここでは、VS-RC202 を使って、モーションを再生する方法を説明します。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_motion を選択してください。



スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。

スケッチの書き込みが終わったら、**SV1, SV2, SV3, SV4** にサーボモータを接続して(もう接続しているかもしれませんが)、バッテリーボックスを VS-RC202 の DC ジャックに接続します。

Arduino IDE のシリアルモニタを開いてください。ボーレートは 115200 にします。そして、VS-RC202 のリセットボタンを押してください。最初によくわからない文字がでますが、これは ESP-WROOM-2 が起動時に出力する文字列なので気にしないでください。



w, a を入力して、Enter キーを押すと、サーボモータが数回動きます。

s, d を入力して、Enter キーを押すと、サーボモータが動き続けます。

x を入力して、Enter キーを押すと、何も起きないと思いますが、一応モーションは再生されています。

詳しくプログラムの内容を見ていきましょう。スケッチの最初のほうに配列がありますが、これがモーションです。配列の[]内の内容ですが、以下のようになります。

{遷移時間、SV1 の目標位置、SV2 の目標位置、…、SV10 の目標位置}

この配列が2次元配列になっており、順番に実行されていくイメージです。

(注)SV5-SV10 はサーボモータを繋げていないため、常に目標位置を0にしています。

```
int motion0[1][11] = {
    {600,0,0,0,0,0,0,0,0,0,0},
};

int motion1[3][11] = {
    {600,1000,1000,1000,1000,0,0,0,0,0,0},
    {1200,-1000,-1000,-1000,-1000,0,0,0,0,0,0},
    {600,0,0,0,0,0,0,0,0,0,0},
};

:
```

Loop()内の getCommand()関数は前ページのシリアルモニタから入力を受け取る際に使っています。ここで受信した文字に応じて setMotionNumber()関数で再生するモーションを設定します。

そして、selectMotion()関数で再生したいモーションと、モーションの1次元目の要素数を引数として、playMotion() 関数もしくは playMotionOnce()関数で再生します。

playMotion()は次の命令がくるまで、モーションを再生し続けます。playMotionOnce()は一度だけモーションを再生します。

```
void loop() {
    getCommand();      //シリアル入力で再生するモーションを選択
    selectMotion();    //モーションを再生
}
```

サンプルスケッチの例でいうと、以下のような流れになります。実際にプログラムを動かして、コマンド通りにモーションが再生されていることを確認してみましょう。

1. シリアルモニタから 'w' を受信する(getCommand())
2. motion\_number = M\_NUM1 が設定される(getCommand())
3. motion\_number = M\_NUM1 なので、motion1 を再生する(selectMotion())

#### 【関数の説明】

**書式: setMotionNumber(モーション番号);**

再生したいモーション番号を設定する。

例: setMotionNumber(1); //モーション番号1をセット

**書式: getMotionNumber();**

現在設定されているモーション番号を取得する。

例: switch(getMotionNumber()){  
    case 1: //モーション番号が1の場合・・・  
        :  
        :

**書式: playMotion(モーション、モーションの要素数);**

**書式: playMotionOnce(モーション、モーションの要素数);**

例 1: playMotion(motion1, 3); //モーション1を再生し続ける

例 2: playMotionOnce(motion3, 5); //モーション3を1回再生

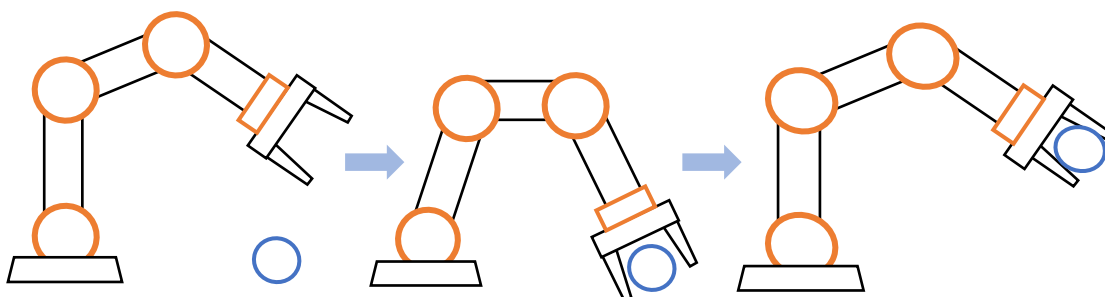
(注)より詳しい動作を知りたい人は VS-RC202 フォルダにある、VS-RC202.cpp のソースコードを見てください。

モーションの作り方ですが、基本は動かしたいロボットの途中のポーズをつなげていくイメージです。例えば、4軸のロボットアームで、ものを持ち上げたいとき、最低限以下の3つのポーズがあればよさそうですね。

- 最初のポーズ
- アームを下ろして物をつかんだポーズ
- アームを上げてものを持ち上げたポーズ

つまり、この3つのポーズのそれぞれの関節の位置と、各ポーズ間の遷移時間をきめれば、ものを持ち上げるモーションが作れそうです。

```
Arm_motion[3][11] = {  
    {600,0,0,0,0,0,0,0,0,0,0}, //最初のポーズ  
    {600,400,600,400,600,0,0,0,0,0}, //2番目のポーズ  
    {1000,0,0,0,600,0,0,0,0,0}, //最後のポーズ  
};
```



ぜひ自分だけのオリジナルロボットを作って、モーションを作ってみましょう！

#### 【まとめ】

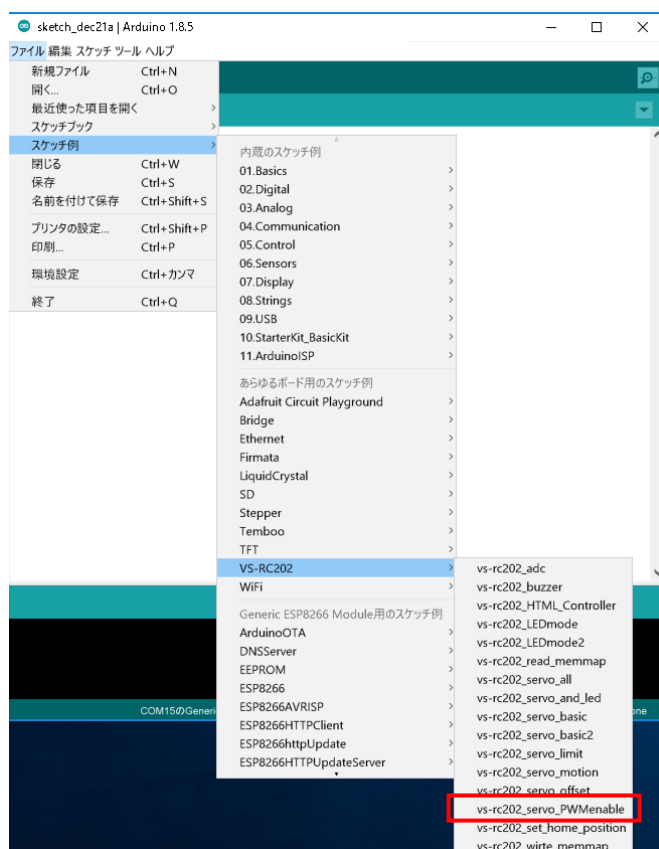
1. モーションはポーズの集合
2. 動かしたいロボットのポーズと、ポーズ間の遷移時間を決めればモーションは作れる



## (F)サーボモータを脱力させる

ロボットはサーボモータを動かし続けると、あっというまにバッテリー切れになります。できるだけ省エネがいいですね。ここでは、サーボモータの信号を ON/OFF する方法を説明します。

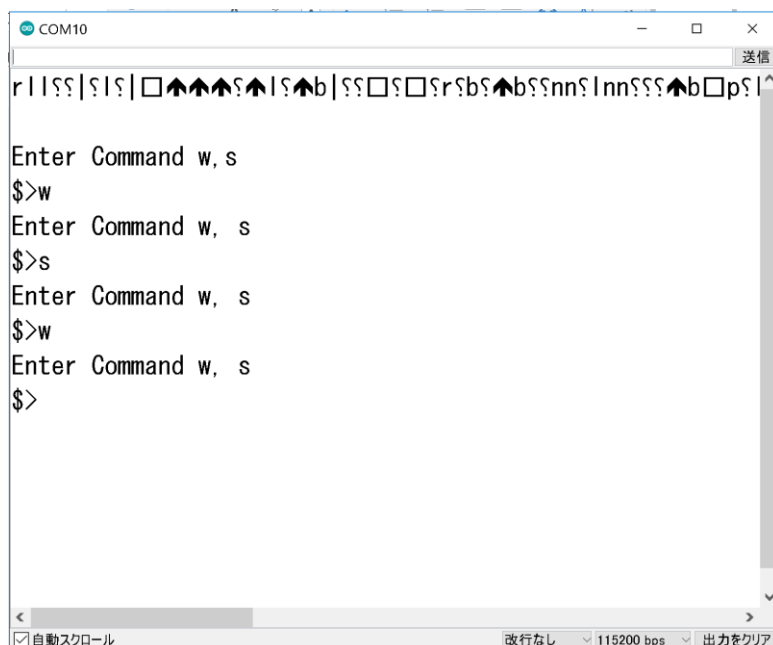
Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_PWMenable を選択してください。



スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。

スケッチの書き込みが終わったら、SV1, SV2, SV3, SV4 にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。

Arduino IDE のシリアルモニタを開いてください。ボーレートは 115200 にします。そして、VS-RC202 のリセットボタンを押してください。'w'を入力するとモータに力が入り、's'を入力すると脱力します。



servoEnable()関数はサーボモータの PWM 信号の ON/OFF を切り替えます。例えばロボットの起動時や、動作していないときなど必要ない状況ではできるだけサーボモータを脱力することでバッテリーを長持ちさせることができます。

ただし、脱力することで転倒や落下による破損といった事故が起きないように、必ずロボットを安全な姿勢にしてから脱力してください。

#### 【関数の説明】

**書式:** servoEnable(ピン番号, ON/OFF);

例 1: servoEnable(1, 0); //SV1 を OFF にする(脱力)

例 2: servoEnable(1, 1); //SV1 を ON にする(PWM 信号有効)

#### 【まとめ】

1. servoEnable()関数でサーボモータの ON/OFF を切り替える
2. 必要ないときは極力 OFF にすることでバッテリーが長持ちする

## (G)サーボモータのオフセットを設定する

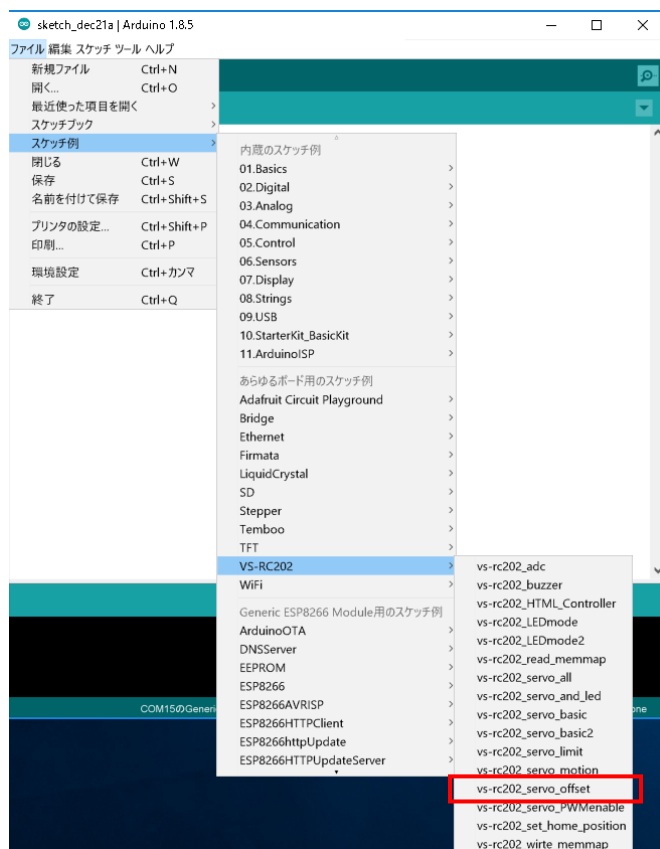
ここではサーボモータのオフセットについて説明します。

サーボモータの軸を中心(位置0)にした状態でサーボホーンを取り付けても、サーボホーンがまっすぐにならず、どうしてもずれた方向を向くことがあります。

これはサーボモータの個体差などにより生じる差ですが、ロボットを組み立てた状態で足や手が少し曲がっていたら嫌ですよ？

そこで、プログラム上で各サーボモータの中心角度となる PWM の信号をずらして、サーボホーンをまっすぐにします。このずらす値を**オフセット**と呼びます。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_offset を選択してください。



スケッチを開いたら VS-RC202 と PC を USBmicroB ケーブルで接続し、スケッチを書きこんでください。スケッチの書き込みが終わったら、**SV1, SV2, SV3, SV4** にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。サーボモータの初期位置が先程までと少しずれたのではないのでしょうか？

スケッチの `setup()` で、`setServoOffset()` という関数を呼んでいます。これは各サーボモータにオフセットを設定しています。

例えば、`setServoOffset(1, 100)` は SV1 のサーボモータの位置を 100 ずらすという意味です。これ以降は `setServoDeg(1, 500)` とした場合、実際には `setServoDeg(1, 600)` の信号が出ています。このように、オフセットを使えば、サーボモータの個体差をソフトウェアで修正することが可能です。

```
void setup() {  
    :  
    setServoOffset(1, 100); //Offset range:-500 to 500  
    setServoOffset(2, 500);  
    setServoOffset(3, -200);  
    setServoOffset(4, -500);  
}
```

#### 【関数の説明】

**書式:** `setServoOffset(ピン番号, オフセット);` //設定範囲:-500~500

例 1: `setServoOffset(1, 500);` //SV1 の中心位置を 500 ずらす

例 2: `setServoOffset(1, -500);` //SV1 の中心位置を-500 ずらす

#### 【まとめ】

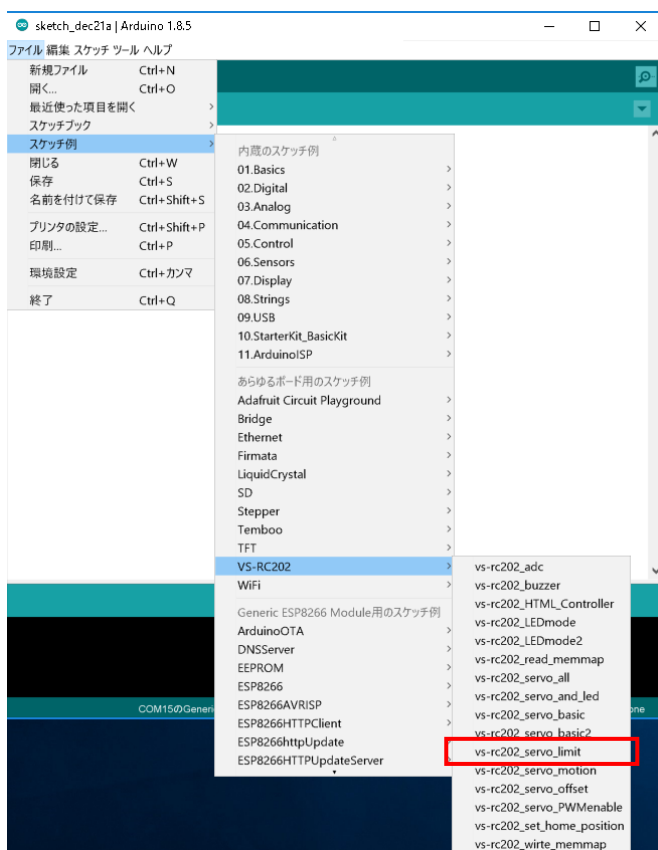
1. サーボモータは個体差のため、同じ PWM 信号でも停止位置がずれる
2. オフセットを設定するとソフト的に停止位置を合わせることができる

## (H)サーボモータの限界角度を設定する

サーボモータのオフセットと共に覚えておきたいのが限界角度です。大抵のサーボモータは中心から左右に約 90° 回転します。ただし、PWM 信号で本来の可動範囲外の信号を入力すると、ロックがかかってサーボモータが故障する原因になります。

VS-RC202 ではソフト的にサーボモータに入力できる PWM の範囲の限界値を設定することにより、サーボモータの故障を防ぐ仕組みがあります。では実際に設定方法を見てみましょう。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_limit を選択してください。



スケッチを開いたら VS-RC202 と PC を USBmicroB ケーブルで接続し、スケッチを書きこんでください。

スケッチの書き込みが終わったら、SV1 にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。

動きをみると、左右にサーボモータの軸が回転していますが、大きく回転するときと、小さく回転するときがあることに気づきます。これはサーボの限界可動範囲を変化させているためです。詳しく見ていきましょう。

スケッチの loop() で、setServoLimitL()/setServoLimitH() という関数を実行しています。これはサーボモータの負方向・正方向の限界可動範囲を設定する関数です。サーボモータは設定された限界可動範囲以上には回転できません。サンプルコードで目標位置-1800~1800 で回転しようと思いますが、限界可動範囲が-800~800 で設定されているときは可動範囲も-800~800 になります。

```
void loop() {  
  int sv_limit = 1800;           //Servo range limit 1800  
  int sv_time = sv_limit/2;      //Moving time 900  
  int sv_delay = sv_time + 100;  //Wait 1000  
  setServoLimitL(-sv_limit);     //負方向の限界位置=-1800  
  setServoLimitH(sv_limit);      //正方向の限界位置=1800  
  setServoMovingTime(sv_time);   //遷移時間=限界角度の半分  
  :  
}
```

#### 【関数の説明】

**書式: setServoLimitL(負方向の限界可動範囲); //設定範囲:-2500~2500**

**書式: setServoLimitH(正方向の限界可動範囲); //設定範囲:-2500~2500**

例 1: setServoLimitL(-1500); //負方向の限界可動範囲=-1500

例 2: setServoLimitH(800); //正方向の限界可動範囲=800

[注 1]サーボモータによって可動範囲は異なります。稼働範囲の狭いサーボモータで、限界角度を大きくするとロックの原因になるので慎重に設定してください。

[注 2]サーボモータの動く最大範囲はオフセット+限界可動範囲になります。

setServoOffset(SV1, 500); setServoLimitH(2500); の場合、SV1 は最大 3000 まで稼働します。

#### 【まとめ】

1. サーボモータは限界可動範囲内でしか回転しない
2. サーボモータによって可動範囲は異なる。限界可動範囲の設定は慎重にする
3. サーボの最大可動範囲はオフセット+限界可動範囲

## ①一通りの処理を入れる

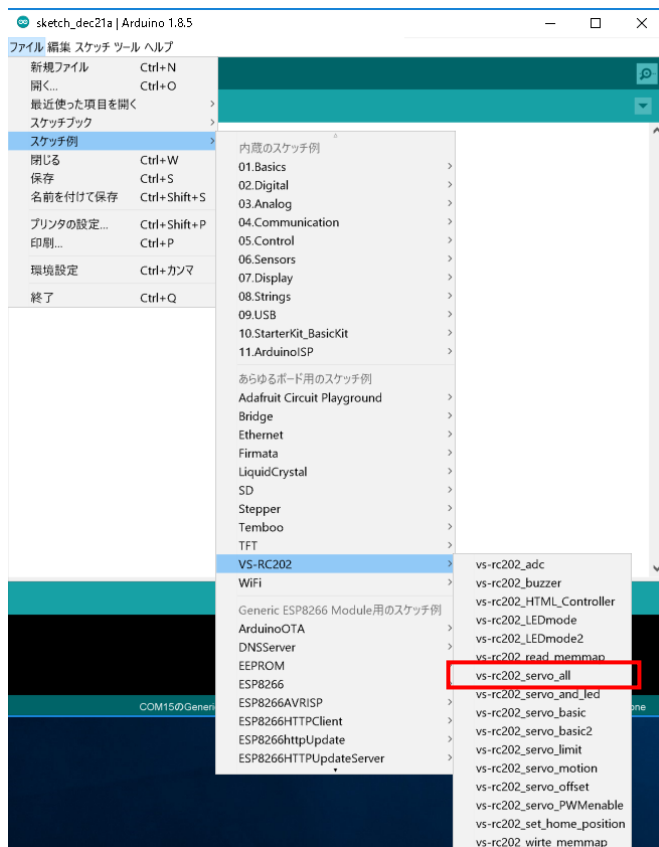
ここまでで、一通りのサーボモータの制御方法は学びましたが、実際にコードを書くとすれば、以下のような流れになります。

```
//モーションと関数の定義
int motion1[] = ...;
getCommand();
selectMotion();

//初期化
void setup() {
  //vs-rc202 ライブラリを初期化
  //サーボモータの限界可動範囲設定
  //サーボモータのオフセットを設定
  //サーボモータの PWM を有効にする(任意のタイミング)
}

//ループ
void loop(){
  //コマンド読み取り
  //モーションの実行
}
```

前ページの処理内容で実際に動かしてみましよう。Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_servo\_all を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。



スケッチの書き込みが終わったら、**SV1, SV2, SV3, SV4** にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。Arduino IDE のシリアルモニタを開いてください。ボーレートは 115200 にします。

w, a を入力して、Enter キーを押すと、サーボモータが数回動きます。

s, d を入力して、Enter キーを押すと、サーボモータが動き続けます。

x を入力して、Enter キーを押すと、何も起きないと思いますが、一応モーションは再生されています。

以上で、サーボモータの制御は一通り完了です。



## 7. LED モード

### (A)LED モードとは？

LED モードはサーボモータピンに LED を接続して使用するモードのことです。サーボモータは PWM の duty 比が非常に低いため、サーボモータと同じ信号では LED を明るく光らせることができません。LED モードでは duty 比を高めて、LED を明るく光らせることが可能です。

LED モードのピンにサーボモータを接続すると故障する可能性があるため、絶対に行わないでください。

サーボモータの PWM の duty 比は低い。

LED は duty 比が高くても使える。



### (B)LED の接続

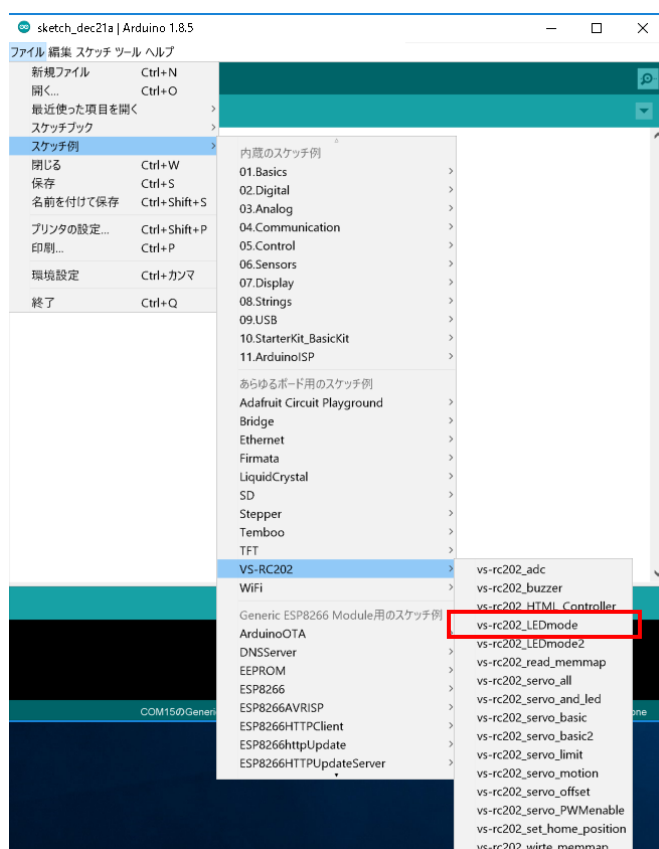
LED を光らせるにはある程度の電流が必要となるため、通常はマイコンから直接ドライブせず、間に FET や専用の LED ドライバ IC を使用します。VS-RC202 は小型の LED1~2 個であれば、サーボモータピンから直接 LED を制御できますが、それ以上を接続する場合は LED をドライブする FET を使用するなどして下さい。

FET と LED を搭載する OctopusLED ライトブリック(青)を使用すると簡単です。ロボットショップで購入いただけます。

[https://www.vstone.co.jp/robotshop/index.php?main\\_page=product\\_info&manufacturers\\_id=97&products\\_id=4356](https://www.vstone.co.jp/robotshop/index.php?main_page=product_info&manufacturers_id=97&products_id=4356)

## (C)LED をサーボモータのように制御する

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_LEDmode を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。



スケッチを開いたら VS-RC202 と PC を USBmicroB ケーブルで接続し、スケッチを書きこんでください。

スケッチの書き込みが終わったら、SV1 に LED を接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。

LED がゆっくり点滅しているのが確認できたでしょうか？LED モードでは、遷移時間に応じて LED の輝度が変わっていきます。目標輝度と遷移時間を設定することにより、少しずつ LED の輝度を変化させることができます。

```
void setup() {
  initLib();           //Initialize vs-rc202 library
  servoEnable(1, 1);  //Enable SV1 PWM
  setLedMode(1,1);    //Set SV1 LEDmode
  setServoMovingTime(1000); //LED brightness changing time
}

void loop() {
  setLedBrightness(1, 0); //Set LED1 brightness 0
  moveServo();
  delay(INTERVAL);

  setLedBrightness(1, 1000); //SetLED1 brightness 1000
  moveServo();
  delay(INTERVAL);
}
```

#### 【関数の説明】

**書式: setLedMode(ピン番号, 有効無効フラグ);**

**書式: setLedBrightness(ピン番号, 輝度); //輝度の設定範囲:0~1000**

例 1 setLedMode(10, 1); //SV10 を LED モードにする

setLedMode(10, 0); // SV10 を通常モードにする

例 2: setLedBrightness(10, 1000); //LED10 の輝度を 1000 にする

#### 【まとめ】

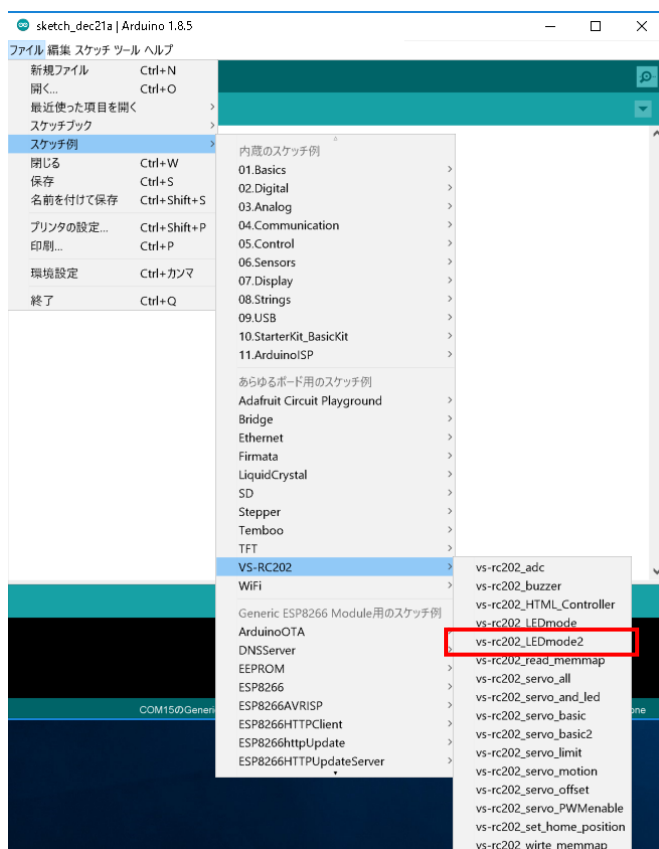
1. setLedMode()でピンを LED モードに変更できる
2. 輝度の設定には setLedBrightness()を使う
3. LED の輝度は遷移時間に合わせて変化する

## (D)LED の輝度を直接制御する

setLedBrightness()はサーボモータと同じように LED の輝度を変化させることができますが、moveServo()のタイミングで輝度が変わるため、サーボモータと独立して使用するのが難しい場合があります。

そこで、サーボモータと LED を併用する場合は、setLedBrightnessDirect()という関数を使うと便利です。

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_LEDmode2 を選択します。スケッチを開いたら、VS-RC202とPCをUSBmicroBケーブルで接続して、スケッチを書きこんでください。



スケッチの書き込みが終わったら、SV1, 2, 3, 4 にサーボモータを接続し、SV10 に LED を接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続します。

LED が点灯した状態でサーボモータが動いているのが確認できたでしょうか？  
setLedBrightnessDirect()関数を使うと、LED の輝度を即座に反映させることができ、モーションの再生の影響を受けません。

```
void setup() {
  initLib(); //Initilize vs-rc202 library
  servoEnable(1, 1); //Enable SV1 PWM
  servoEnable(2, 1);
  servoEnable(3, 1);
  servoEnable(4, 1);
  servoEnable(10, 1);
  setLedMode(10, 1); //Set SV10 LEDmode
  setLedBrightnessDirect(10, 1000); //Set LED10 brightness 1000 directly
}

void loop() {
  playMotionOnce(motion, 2);
  delay(20);
}
```

#### 【関数の説明】

**書式**: setLedBrightnessDirect(ピン番号, 輝度); //輝度の設定範囲:0~1000

例 1: setLedBrightnessDirect(10, 1000); //LED10 の輝度を即座に 1000 にする

輝度を 1 以上設定していれば、moveServo()をしても影響を受けない。輝度が 0 の場合は moveServo()で輝度に変化する

setLedBrightnessDirect(10, 100); //モーションを再生しても影響を受けない

setLedBrightnessDircet(10, 0); //モーションを再生すると、輝度に変化する

#### 【まとめ】

1. LED の輝度を即座に変化させる場合は setLedBrightnessDirect()を使用する
2. setLedBrightnessDirect()を使用するとモーションの再生の影響を受けない

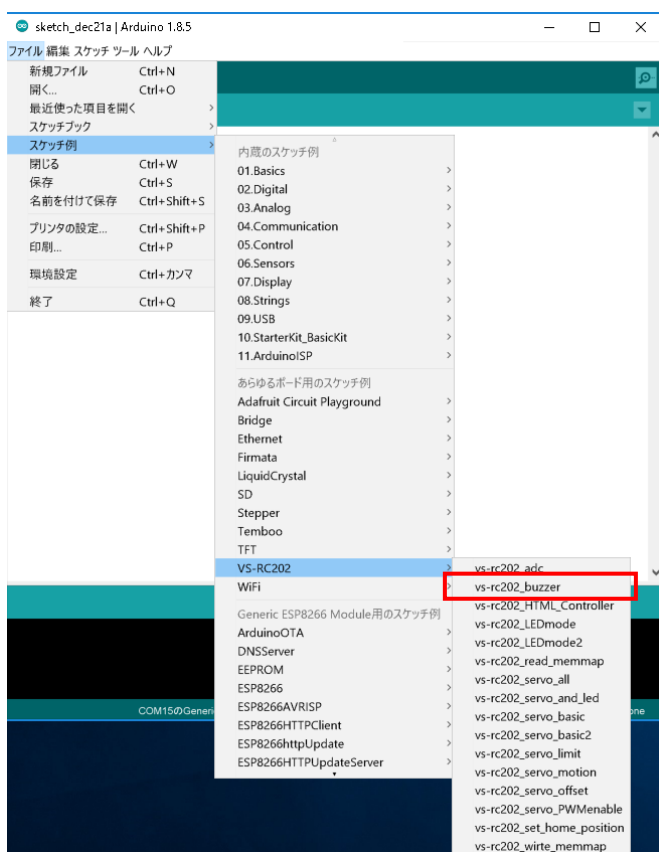
## 8. ブザーモード

### (A) ブザーの仕組み

VS-RC202 には圧電ブザーが搭載されています。圧電ブザーには金属板が入っており、一定の周波数で PWM 信号を与えると、その周波数で金属板が揺れて音が出ます。よって、PWM 信号の周波数そのまま音の周波数となります。

### (B) ブザーの使い方

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_buzzer を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。



スケッチの書き込みが終わったら、きらきら星が鳴り始めるのが確認できます。

スケッチの内容を見てみます。基本的には buzzerEnable() を 1 度だけ実行して、setBuzzer() で鳴らしたい音を設定するだけです。注意点として、ブザーを有効にした場合は、SV9、10 は使えなくなります。

```
void setup() {  
  
  initLib();           //Initilize vs-rc202 library  
  buzzerEnable(1);     //Enable buzzer  
                      //When buzzer is enabled SV9,10 cannot be used  
}  
  
void loop() {  
  setBuzzer(PC5, BEAT4, TANG);  
  setBuzzer(PC5, BEAT4, TANG);  
  setBuzzer(PG5, BEAT4, TANG);  
  :  
}
```

#### 【関数の説明】

**書式: buzzerEnable(Enable(1)/Disable(0)); //ブザーを有効、無効切り換え**

**書式: setBuzzer(音程、音の長さ、息継ぎ); //音を鳴らす**

例 1: buzzerEnable(1); //ブザー有効(SV9、10 使用不可)  
buzzerEnable(0); //ブザー無効(SV9、10 使用可能)

例 2: setBuzzer(PC5, BEAT4, TANG); //C5 の四分音符をタンギングして鳴らす  
setBuzzer(PA5, BEAT2, SLUR); //A5 の二分音符をスラーで鳴らす

setBuzzer() は音符を表現しており、単音のメロディーを鳴らすことができます。  
音の一覧は別紙 VS-RC202 取扱説明書の P32-35 をご参照ください。

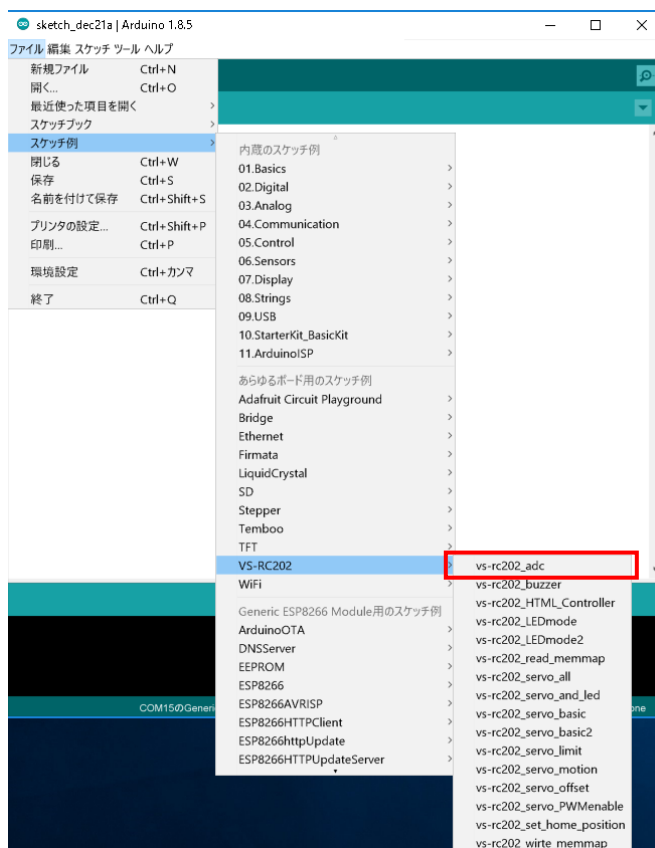
#### 【まとめ】

1. ブザーを使用する場合は buzzerEnable() を使用する
2. buzzerEnable(1) の場合は、SV9、10 は使用できない
3. setBuzzer() で単音のメロディーを鳴らすことができる

## 9. センサの使い方

ここでは、センサの使用方を説明します。VS-RC202 は3つのアナログセンサと、1つの超音波センサを接続できます。さらに、電源電圧を読み取ることも可能です。

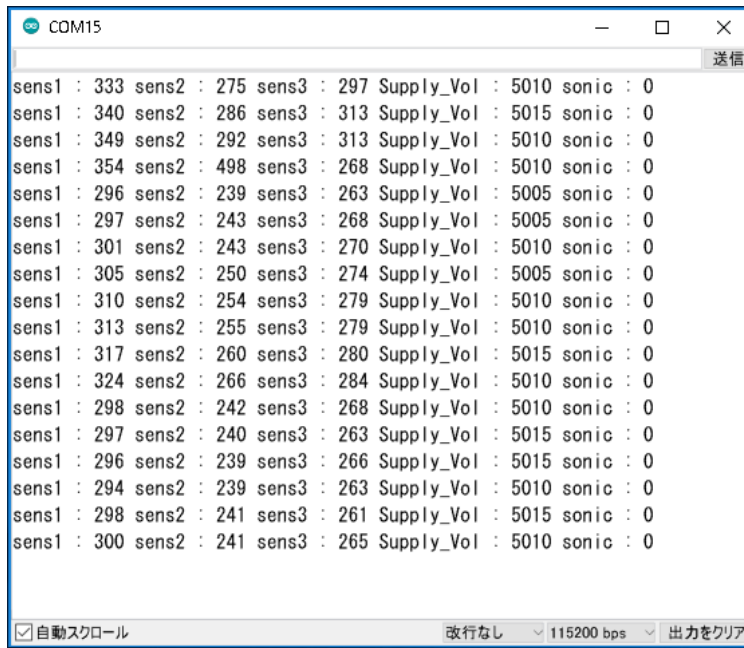
まずはサンプルを動かしてみましょ。Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_adc を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書きこんでください。



スケッチの書き込みが終わったら、Arduino のシリアルモニタを表示します。ボーレートは 115200 を選択します。



以下のようなメッセージが表示されれば、正常に動作しています。センサを繋がなくても値は取得できます。(この場合の値はなにも繋がっていないので電圧不定で中途半端な値が返ってきます)



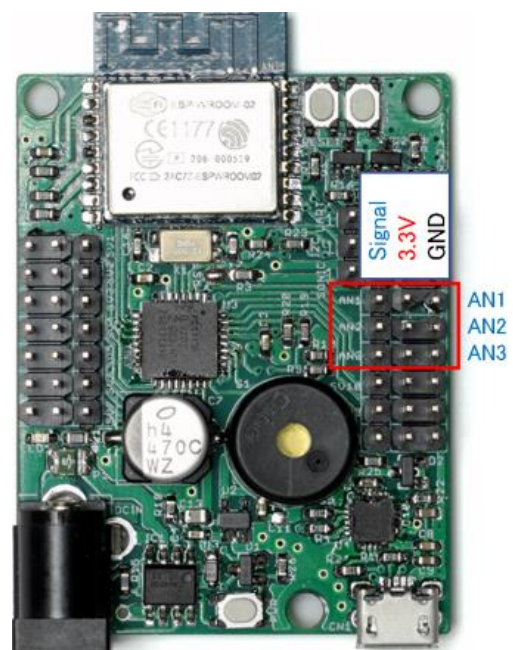
各関数と読み出している内容について見ていきます。

### (A)アナログセンサ値の読み込み

setSensEnable()関数でセンサ読込の有効化をしますが、デフォルトで有効になっています。AN1-3 に接続されたセンサの電圧を読み込むことができます。また readPow でバッテリー電圧を読み込むことができます。

```
void setup(){
  setSensEnable(1);
}

void loop(){
  sens[0] = readSens(1);
  sens[1] = readSens(2);
  sens[2] = readSens(3);
  sens[3] = readPow();
}
```



#### 【関数の説明】

**書式** : readSensEnable(Enable(1)/Disable(0)); //デフォルトは Enable

**書式** : readSens(ピン番号); //センサ値読み込み

**書式** : readPow(); //電源電圧読み込み

例 1 : readSensEnable(0); //センサ読込を Disable にする

例 2 : readSens(1); //センサ 1 を読み込む

readSens(2); //センサ 2 を読み込む

例 3 : readPow(); //戻り値は mV

#### (B)プルアップ抵抗を有効にする

AN1-3 のピンの内部プルアップ抵抗を有効にすることができます。プルダウンスイッチを接続する場合等に使用します。

```
void setup() {  
  initLib();  
  :  
  setPullupEnable(1,1); //Enable pullup of AN1  
  setPullupEnable(2,1); //Enable pullup of AN2  
  setPullupEnable(3,0); //Disable pullup of AN3  
  :  
}
```

#### 【関数の説明】

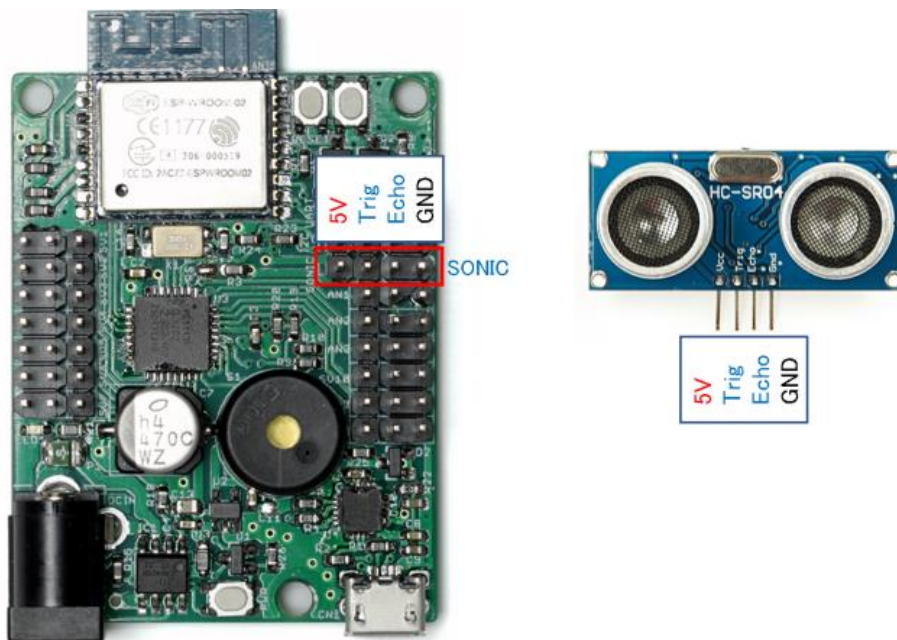
**書式** : setPullupEnable(ピン番号, Enable(1)/Disable(0));

例 1 : setPullupEnable(1,1); //センサ 1 の内部プルアップを Enable

setPullupEnable(1,0); //センサ 1 の内部プルアップを Disable

### (C)超音波センサの値の読み込み

VS-RC202 は超音波センサ HC-SR04 を接続することができます。ピン配列は以下の通りです。



```
void loop() {  
  int sens[5];  
  :  
  sens[4] = (int)getDist(); //Get distance from sonic sensor  
  :  
}
```

#### 【関数の説明】

**書式:** `getDist()`; //戻り値は float 型の距離(mm)

#### 【まとめ】

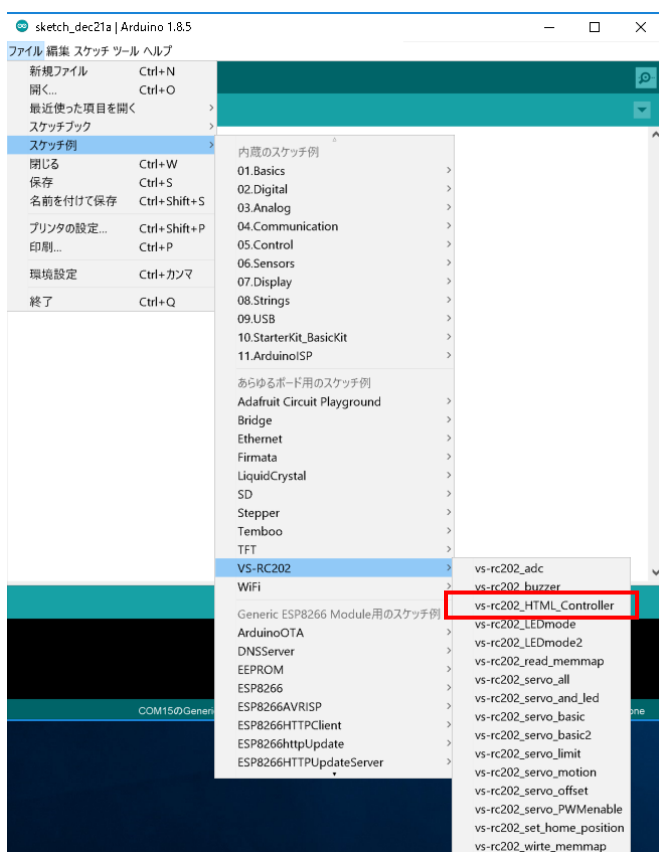
1. アナログセンサの値は `readSens()` で取得する
2. 内部プルアップ抵抗を有効にする場合は `setPullupEnable()` を使う
3. 超音波センサの値は `getDist()` で取得する

## 10. スマートフォンから操作する

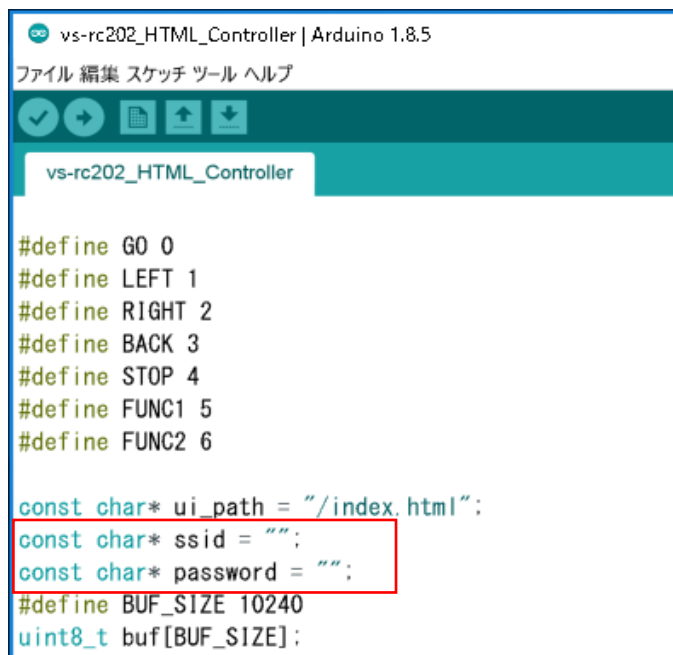
一通りライブラリの使い方を覚えたところで、いよいよ Wi-Fi に繋げて、スマートフォンのブラウザと通信をしてみます。

### (A) ルータに接続して動かす

Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_HTML\_Controller を選択します。vs-rc202\_HTML\_Controller は VS-RC202 の学習プラットフォームであるピッコロボ IoT 用のスケッチですが、基板単体でも動かすことは可能です。



スケッチを開いたら、17, 18 行目に接続したい Wi-Fi ルータの SSID とパスワードを設定します。



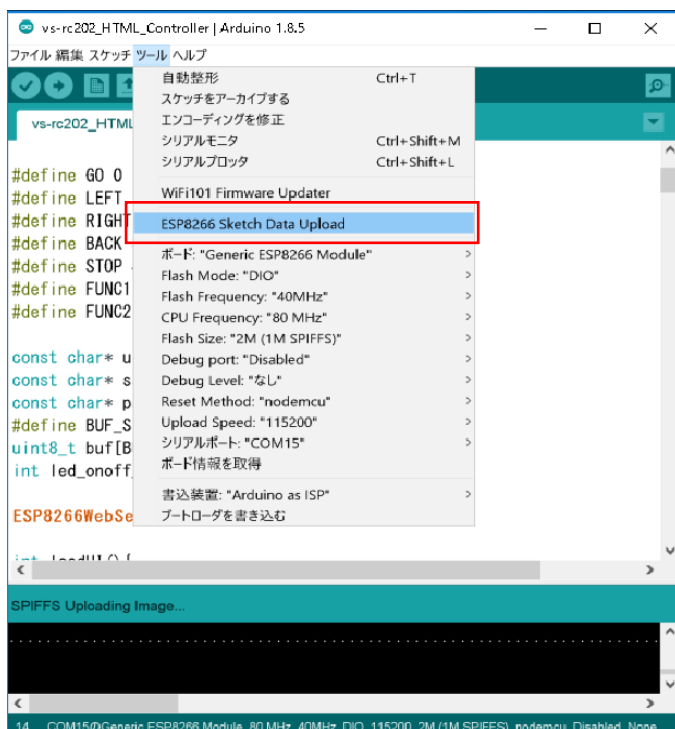
```
vs-rc202_HTML_Controller | Arduino 1.8.5
ファイル 編集 スケッチ ツール ヘルプ
vs-rc202_HTML_Controller

#define GO 0
#define LEFT 1
#define RIGHT 2
#define BACK 3
#define STOP 4
#define FUNC1 5
#define FUNC2 6

const char* ui_path = "/index.html";
const char* ssid = "";
const char* password = "";
#define BUF_SIZE 10240
uint8_t buf[BUF_SIZE];
```

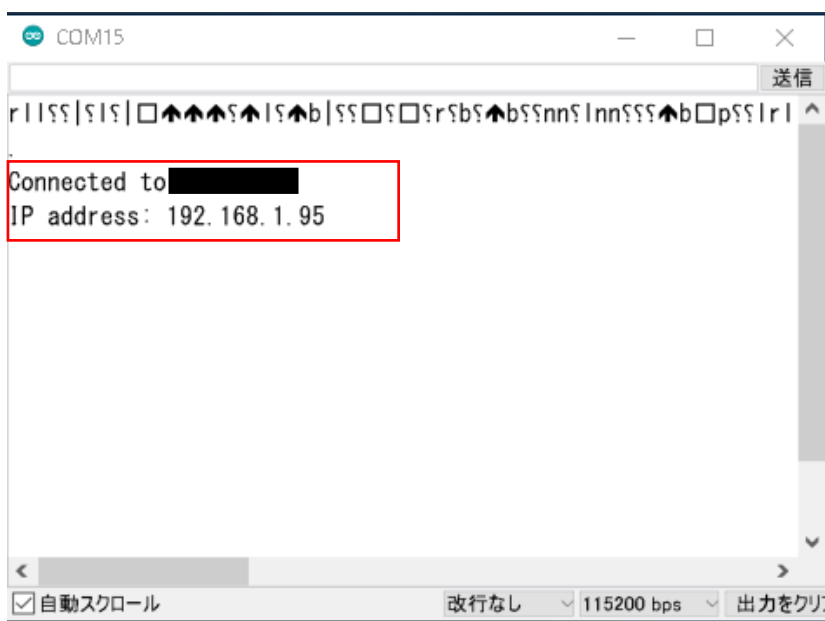
SSID とパスワードを設定したら、VS-RC202 と PC を USBmicroB ケーブルで接続し、スケッチを書きこんでください。

次に、メニューのツールから ESP8266 Sketch Data Upload を選択し、HTML をアップロードします。HTML の内容を見たい場合は、メニューのスケッチ > スケッチのフォルダを表示を選択してください。data フォルダ内に index.html が見つかります。アップロードの最大容量は 1M バイトです。



スケッチと HTML の書き込みが終わったら、Arduino のシリアルモニタを表示します。ボーレートは 115200 を選択します。そして、VS-RC202 のリセットボタンを押してください。正しく、SSID とパスワードが設定されていれば、以下のような表示がでます。

エラーが出る場合は、HTML ファイルのアップロードを忘れていないか、SSID・パスワードに間違いがないかを確認してください。

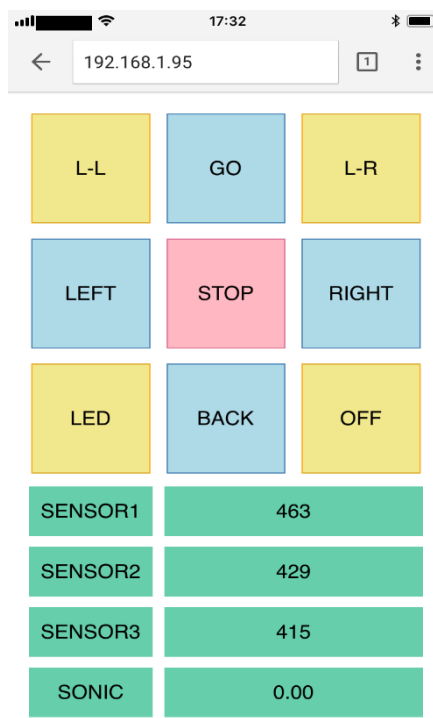


SV1, 2, 3, 4 にサーボモータを接続し、SV9, 10にLEDを接続して、バッテリーボックスをVS-RC202 の DC ジャックに接続します。

PCかスマートフォンを VS-RC202 と同一ネットワーク内の Wi-Fi に接続します。

この状態で VS-RC202 の IP アドレス(上記の例だと <http://192.168.1.95>)にブラウザで接続してみましょう。

右のような画面が表示されれば、正常に通信できています。

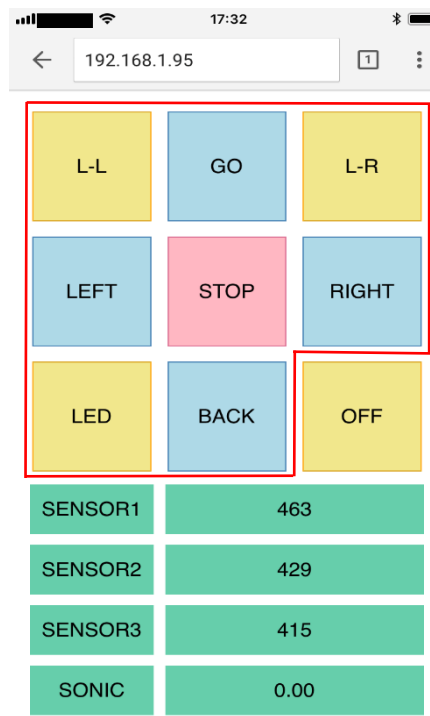


## (B)ブラウザからのコマンドを受け付ける

まずは動作を見てみましょう。ブラウザに表示された青のボタン又はL-L, L-Rを押すとサーボモータが動きます。LED と書かれたボタンを押すとLED が点灯または消灯します。

OFF を押すと基板の電源が切れるので、今は押さないでください。

次に、スケッチ内でどのように実装されているかを見ていきます。少し長いですが、全て理解する必要はありません。必要最小限の部分から始めていきます。



スケッチの 262 行目から `server.on()` が書かれています。これはブラウザからどの URL に GET リクエストが来たら、どの関数を実行しなさいという関連付けを行っています。

例えば、`http://(VS-RC202のIPアドレス)/go/` にリクエストがあれば、`Go()` 関数が実行されます。`index.html` では各ボタンが押された場合に `javascript` で対応する URL に GET リクエストを送っています。

### [Arduino Sketch]

```
server.on("/", handleRoot);
server.on("/go/", Go);
server.on("/left/", Left);
server.on("/right/", Right);
server.on("/back/", Back);
server.on("/stop/", Stop);
:
```

### [index.html]

```
<button class="func" onClick=sendF1()>L-L</button>
<button class="ten" onClick=sendGo()>GO</button>
<button class="func" onClick=sendF2()>L-R</button>
:
```

スケッチの 125 行目に Go()関数があります。処理として、setMotionNumber()を実行していません。これは次に実行するモーションの番号をグローバル変数 motion\_number にセットしていません。

この motion\_number を基に、実行すべきモーションを 91 行目の selectMotion()内で実行しています。

Serial.println("Go");はシリアルモニタにデバッグメッセージを表示しており、server.send(200, "text/html", "Go")はブラウザに「200 OK + Go」を返しています。

```
void Go(){
  setMotionNumber(GO);
  Serial.println("Go");
  server.send(200, "text/html", "Go");
}
```

以上より、ブラウザからコマンドを受け付ける基本的な流れは、以下のようになります。

- ① 実行したい関数を作る
- ② server.on()で URL と関数を関連付ける
- ③ モーションを実行したい場合は関数内で motion\_number をセットして selectMotion()を使う

### (C)ブラウザにデータを送る

次に、ブラウザにセンサデータなどを送りたい場合ですが、基本的にはブラウザからコマンドを受け取る方法と同じです。server.on()で http://(VS-RG202 の IP アドレス)/sens/に Sens()が関連付けられています。

スケッチの 187 行目の Sens()関数では、センサデータを読み出して、文字列に変換して、server.send()の 3 つ目の引数として送り返しています。

```
void Sens(){
  :
  //Sensor data from lpc
  int data1 = readSens(1);
  :
  //Convert numeric to string
  String st_data1 = String(data1);
  :
  String res = String(st_data1+", "+st_data2+", "+st_data3+", "+st_data4+", "+st_data5);
  server.send(200, "text/html", res);
}
```



ブラウザ側の処理としては、index.html の 126 行目で、1 秒ごとに `http://(VS-RC202 の IP アドレス)/sens/` にリクエストを送っています。また、101~119 行目の処理で受信したセンサデータを HTML に挿入しています。

以上より、ブラウザにデータを送る基本的な流れは、以下のようになります。

- ① センサデータなどを読み取る関数を作る
- ② `server.on()` で URL と関数を関連付ける
- ③ `serve.send()` の第 3 引数として文字列データを送る
- ④ ブラウザ側で定期的なリクエストを送り、受信した文字列をパースして表示する

#### 【まとめ】

1. SSID とパスワードを設定して Wi-Fi ルータに繋げる
2. `server.on()` で URL と実行したい関数を関連付ける
3. モーションの再生は `selectMotion()` を使うと簡単
4. `serve.send()` でブラウザにデータを送信できる

## 11. メモリマップを直接操作する

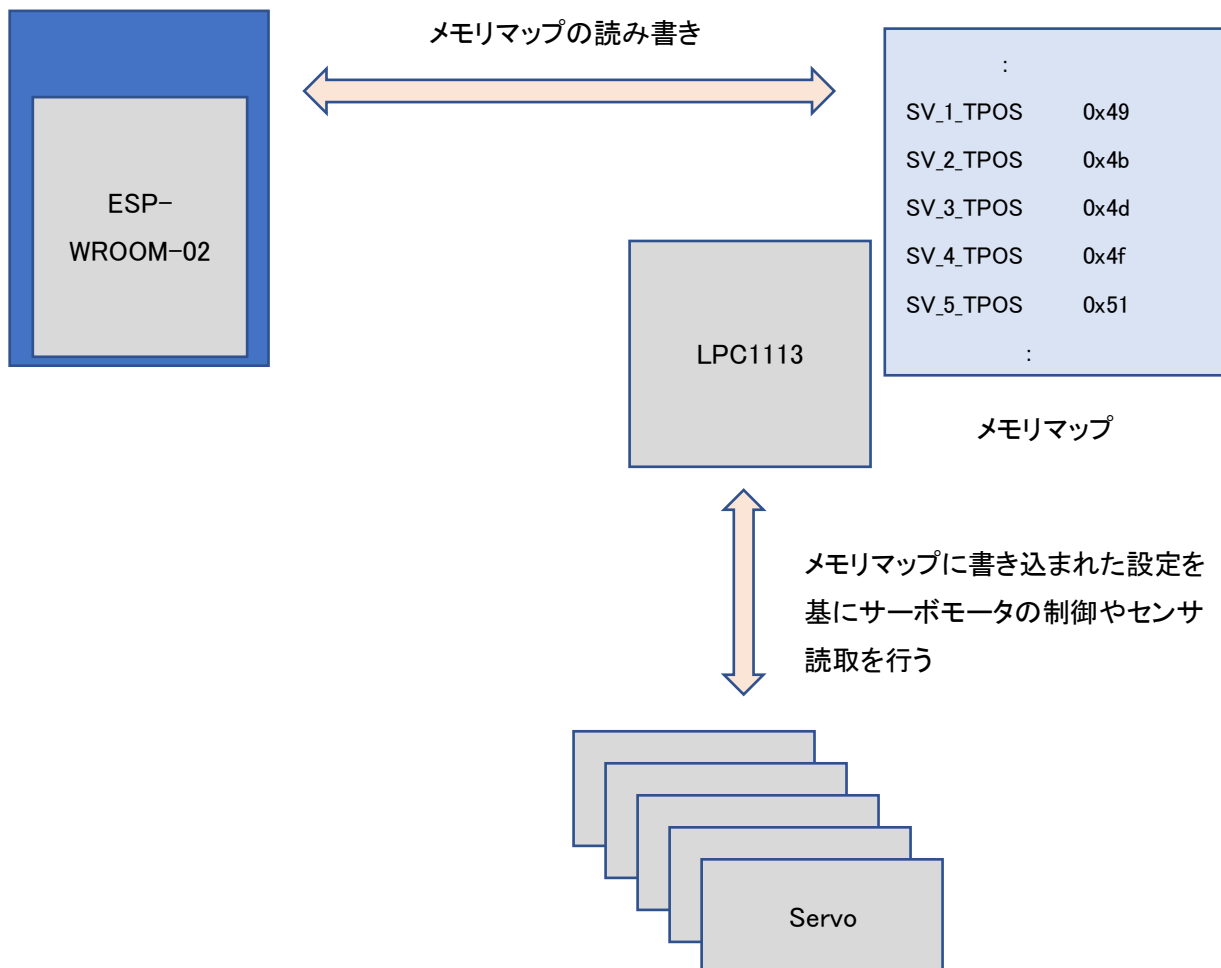
### (A)メモリマップとは？

VS-RC202 に搭載されている LPC1113 はメモリマップという仮想的なレジスタメモリを持っています。このメモリマップの値を読み書きすることにより、サーボモータやセンサを制御します。

これまでに紹介したサンプルコードも実際には呼び出したい機能のメモリマップに値を書き込んでいます。メモリマップを直接扱う方法を覚えれば、VS-RC202 の関数を自分で作れるようになります。

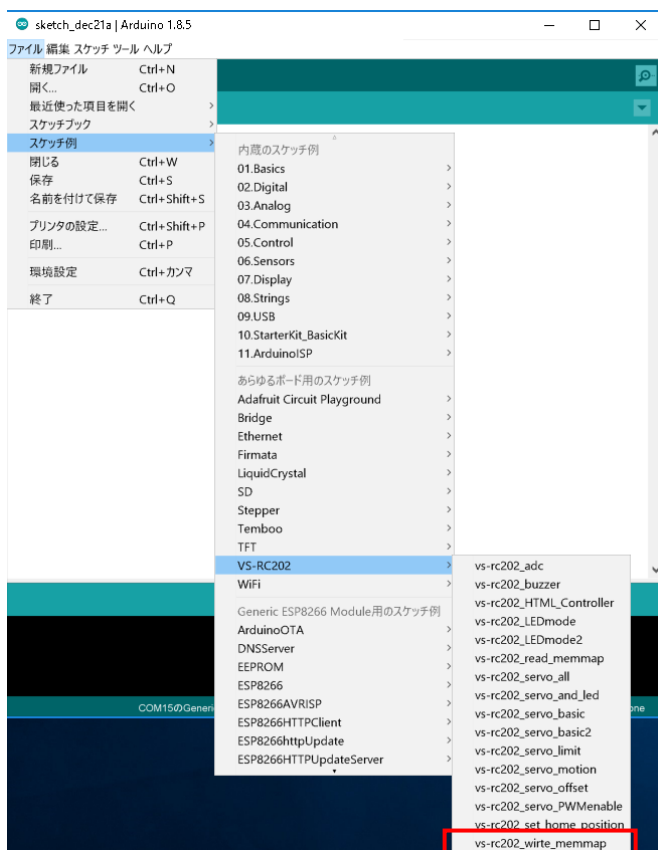
メモリマップの詳細は別紙 VS-RC202 取扱説明書の巻末に記載されています。

[https://www.vstone.co.jp/products/vs\\_rc202/download.html](https://www.vstone.co.jp/products/vs_rc202/download.html)



## (B)メモリアップの書込

サーボモータを動作させたい場合等は、メモリアップに命令を書き込みます。サンプルコードを見てみましょう。Arduino IDE のメニューのファイル > スケッチ例 > VS-RC202 > vs-rc202\_write\_memmap を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書き込んでください。



スケッチの書き込みが終わったら、SV1, SV2, SV3, SV4 にサーボモータを接続して、バッテリーボックスを VS-RC202 の DC ジャックに接続して電源スイッチを押します。正常にスケッチが書き込まれていれば、サーボモータが動き続けます。

スケッチの内容を詳しく見ていきましょう。loop()の中では moveServo()が呼ばれています。これは I2C で LPC1113 に書込命令を送る関数です。I2C の通信には Arduino の標準の Wire 関数を使っています。

VS-RG202 の取扱説明書 P39 にメモリマップ一覧がありますが、アドレス 0x47~0x5b ままでが遷移時間と目標位置のアドレスとなっています。

I2C では通常以下の手順で書込通信が行われます。

- ① 通信したいデバイスをデバイスアドレスで指定する
- ② デバイスの書き込みたいメモリのアドレスを指定する
- ③ 書き込むデータを送信すると、②で指定したアドレスから順番に書き込まれていく
- ④ 終了通知を送る

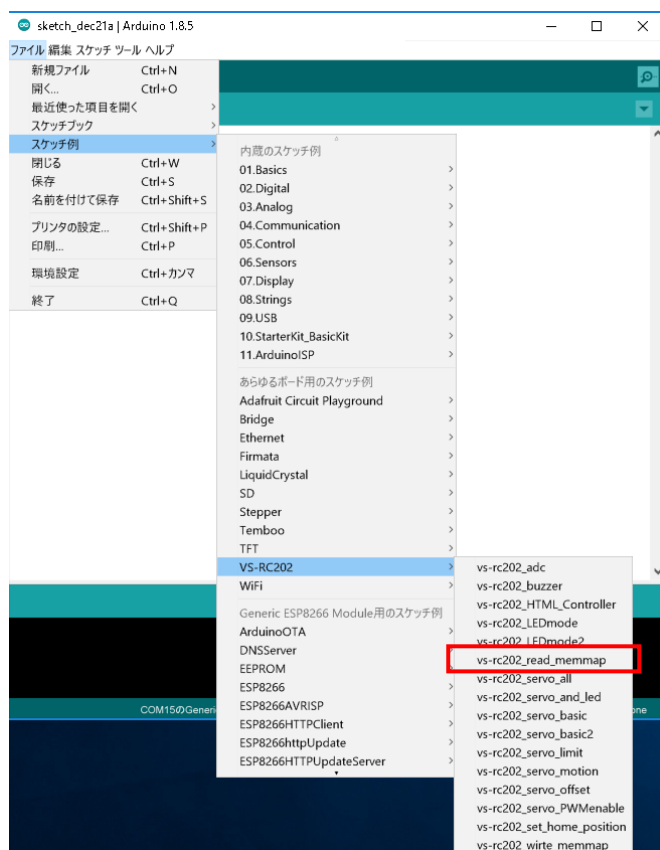
サンプルコードの例だと、LPC1113 のアドレスを指定して、メモリマップの SV\_MV\_TIME(0x47)から順番に遷移時間と目標位置を書き込み、最後に SV\_START に 0x01 を書き込んでいます。

```
void moveServo(int motion[SV_NUM+1]){
    unsigned char byte_motion[2*(SV_NUM+1)];
    int i;

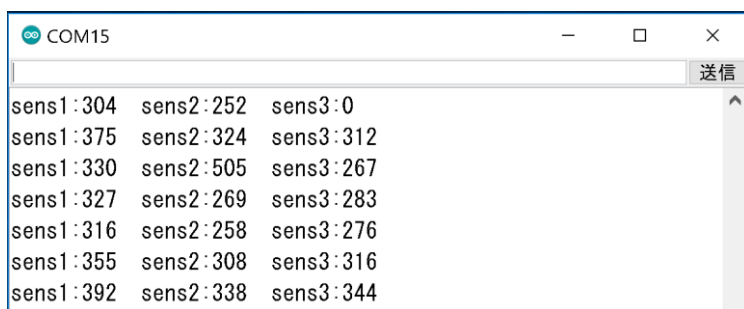
    //I2C では一度に1バイトしかデータが送信できなので、
    //モーション配列のデータを上位8ビットと下位8ビットに分けている
    //リトルエンディアン方式で配列に格納
    for(i=0;i<SV_NUM+1;i++){
        byte_motion[i*2] = motion[i];
        byte_motion[i*2+1] = motion[i] >> 8;
    }
    Wire.beginTransmission(DEV_ADDR); //LPC1113 のデバイスアドレス
    Wire.write(SV_MV_TIME);           //書き込みたい先頭アドレスを指定
    for(i=0;i<2*(SV_NUM+1);i++){     //遷移時間と目標位置を書き込み
        Wire.write(byte_motion[i]);
    }
    Wire.write(1);                    //SV_START に1を書き込み
    Wire.endTransmission();           //通信終了
}
```

## (C)メモリマップの読出

次にメモリマップの値を読み出す手順を説明します。こちらはセンサデータや設定を確認する場合等に使用します。サンプルコードを見てみましょう。Arduino IDE のメニューのファイル>スケッチ例>VS-RC202>vs-rc202\_read\_memmap を選択します。スケッチを開いたら、VS-RC202 と PC を USBmicroB ケーブルで接続して、スケッチを書き込んでください。



スケッチを書き込みが終わったら、Arduino のシリアルモニタを表示します。ボーレートは 115200 を選択します。正常にスケッチが書き込めていれば、以下のようなメッセージが表示されます。



スケッチの内容を詳しく見ていきましょう。loop()の中では readSens()が呼ばれています。これは I2C で LPC1113 に読込命令を送る関数です。I2C の通信には Arduino の標準の Wire 関数を使っています。

VS-RC202 の取扱説明書 P39 のメモリマップの、アドレス 0x01~0x06(AN1~AN3 のセンサデータ)を指定して読み出しています。

I2C では通常以下の手順で読込通信が行われます。

- ① 通信したいデバイスをデバイスアドレスで指定する
- ② デバイスの読み込みたいメモリのアドレスを指定する
- ③ 一旦通信終了
- ④ ②で指定したアドレスから読みたいバイト数だけ読込要求を送る

サンプルコードの例だと、LPC1113 のアドレスを指定して、メモリマップの SENS\_1(0x04)を書き込み、そこから 6 バイトのデータを読み出しています。

```
void readSens(){

    Wire.beginTransmission(DEV_ADDR); //デバイスアドレスを指定
    Wire.write(SENS_1);                d//読込する先頭アドレスを指定
    Wire.endTransmission();           //通信を一度終了

    //Read 2byte
    unsigned char tmp[6];
    int index = 0;
    Wire.requestFrom(DEV_ADDR, 6); //必要バイト数読込要求を送る
    while (Wire.available()) {
        tmp[index++] = Wire.read();
    }

    //Convert byte to short //受信データを表示できる形式に変換
    short sens1 = tmp[1] << 8 | tmp[0];
    short sens2 = tmp[3] << 8 | tmp[2];
    short sens3 = tmp[5] << 8 | tmp[4];
}
```

#### 【まとめ】

1. 全ての機能はメモリマップの読み書きで実現される
2. LPC1113 のメモリマップへのアクセスは I2C を使う
3. 書込ではメモリマップの先頭アドレスを指定し、続けてデータを書き込む
4. 書込はサーボモータを動かしたり、設定を書き込む場合に使用する
5. 読込ではメモリマップの先頭アドレスを指定し、通信を一度終了してから、読込要求を出す
6. 読込はセンサデータや設定を読み出す場合に使用する