VS-RC202 tutorial

table of contents

1.First of all

This book is a collection of tutorials for the wireless robot control board "VS-RC202" equipped with ESP-WROOM-02. Please use it to learn programming for VS-RC202. If you would like to know the detailed specifications of VS-RC202, please refer to the VS-RC202 instruction manual.

2. Caution

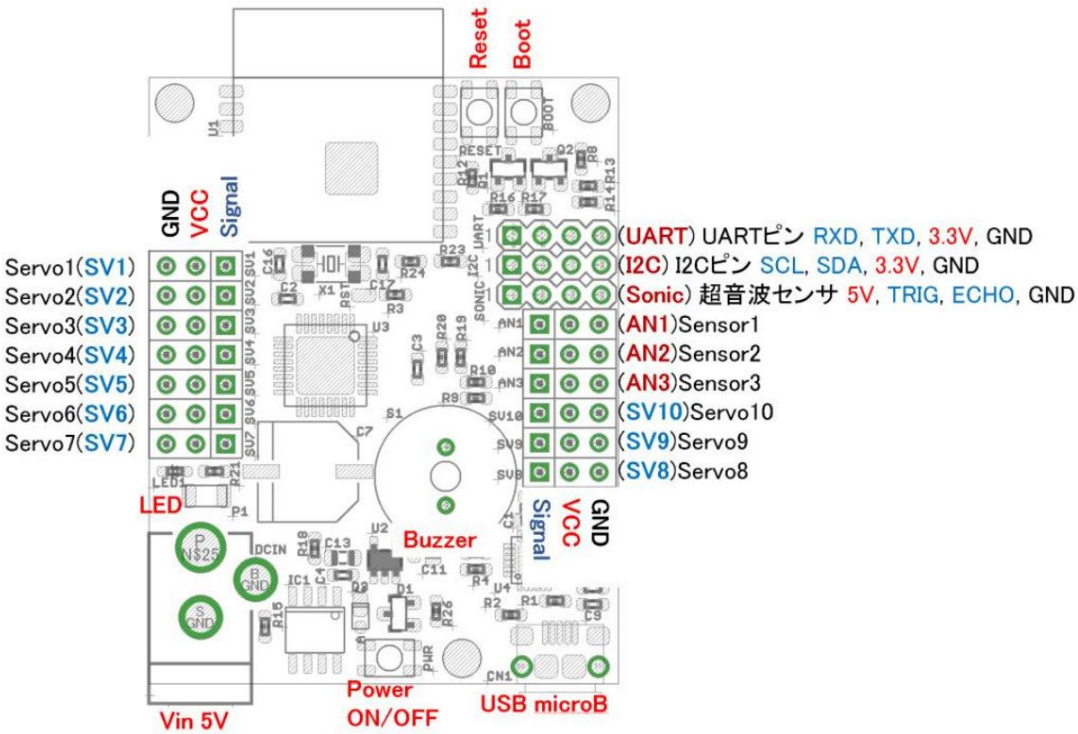
When handling this product, please follow the precautions and use it correctly.

- Do not subject the VS-RC202 (hereinafter referred to as this board) to strong impact.
- Do not get this board wet or use it in humid or dusty locations.

  Failure to do so may result in malfunctions due to short circuits, etc.
- If smoke is emitted from this board, immediately turn off the power. • Do not use this board near children or store it within the reach of children. • Never touch the elements on the board as they may become hot during operation. • Touching the terminals (metal parts) on the board may cause damage due to static electricity. Be sure to check the board

  Make sure to touch the edges.
- If the terminals on the board are shorted with metal, etc., there is a possibility of failure due to overcurrent.

3. Various buttons and input/output pin arrangement



[Note 1] VCC and Vin of the servo motor and ultrasonic sensor are directly connected.

[Note 2] The power cannot be turned off if USB power is being supplied.

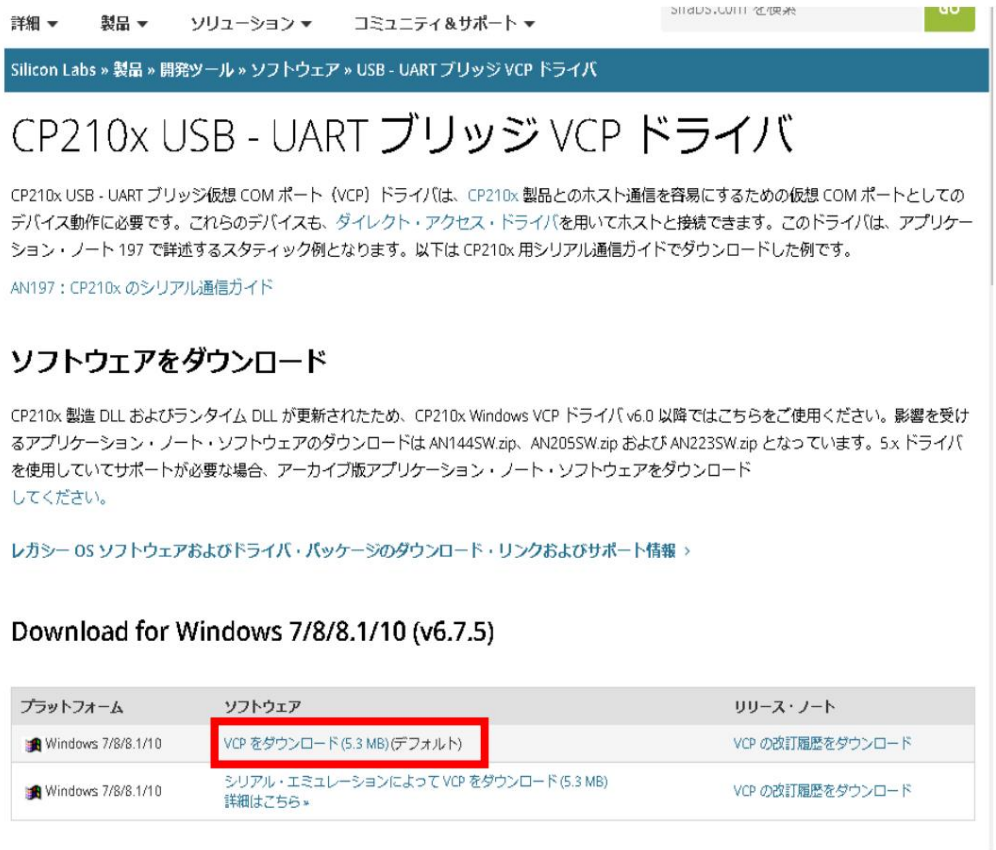
4. Software setup

This section explains how to set up VS-RC202 on a PC using Windows as an example. 2018 1 Based on information as of March 26th.

(A) Install the USB-UART conversion chip driver

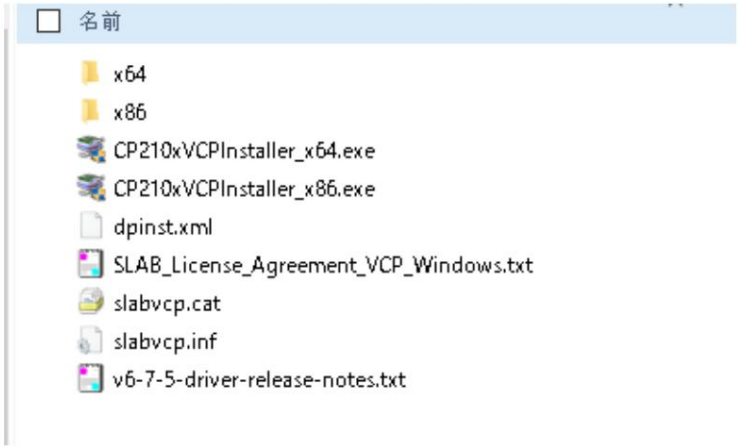
Connect the VS-RC202 to your PC via USB and install the driver to enable it to be recognized. vinegar. Download the Windows driver from the URL below.

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



When you unzip CP210x_Windows_Drivers.zip, the following files will be created. If you are using a 64bit OS , double-click CP210xVCPInstaller_x64.exe to install it. Uses 32bit OS

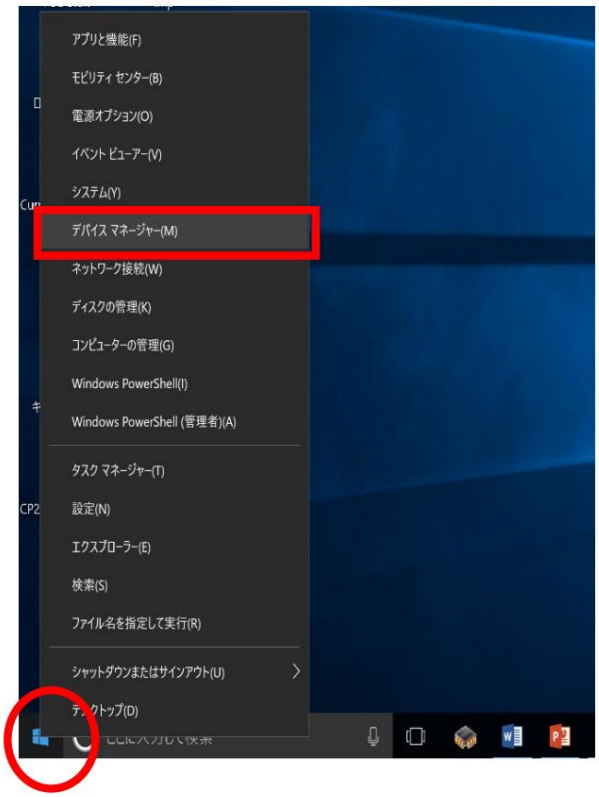
If so, double-click CP210xVCPInstaller_x86.exe to install it.

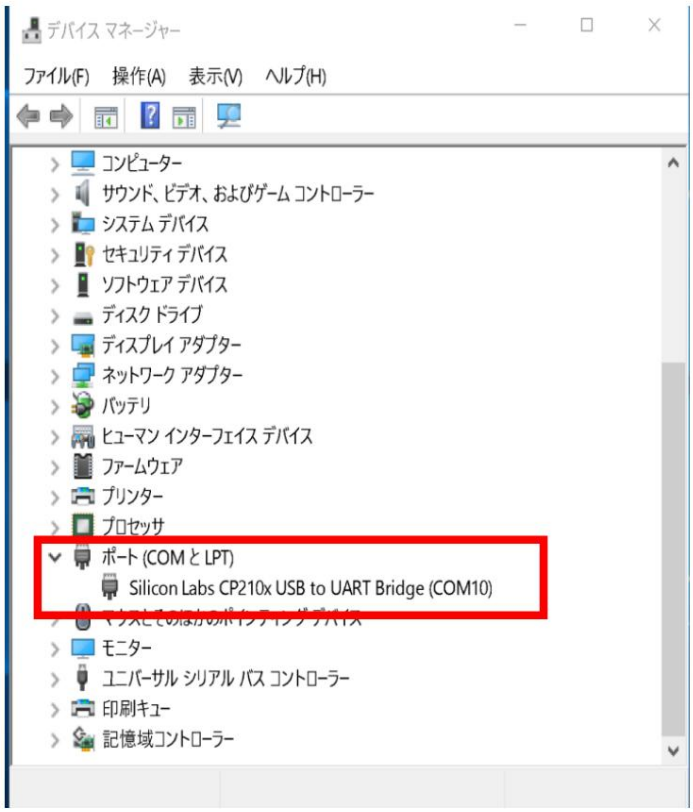


After installing the driver, connect the VS-RC202 to your PC using the USB microB cable and open the device manager. Check your manager. For Windows 10, right-click on the Start button and select Device Manager.

Silicon Labs CP210x USB to UART under Ports (COM and LPT) in the Device Manager window.

If Bridge (COMxx) is displayed, the device is recognized correctly.





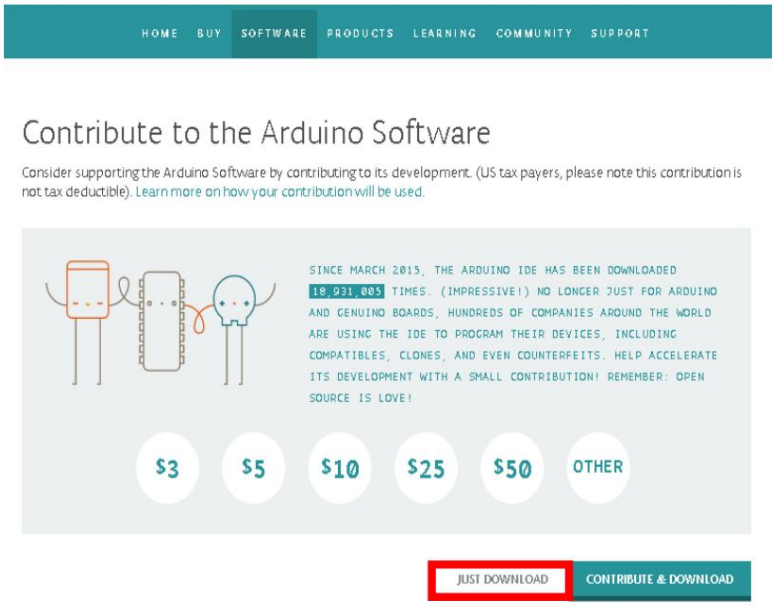
(B) Downloading Arduino IDE

The board VS-RC202 that comes with Piccorobo IoT is programmed using Arduino IDE. next Install Arduino IDE. Access the URL below and select Windows Installer.

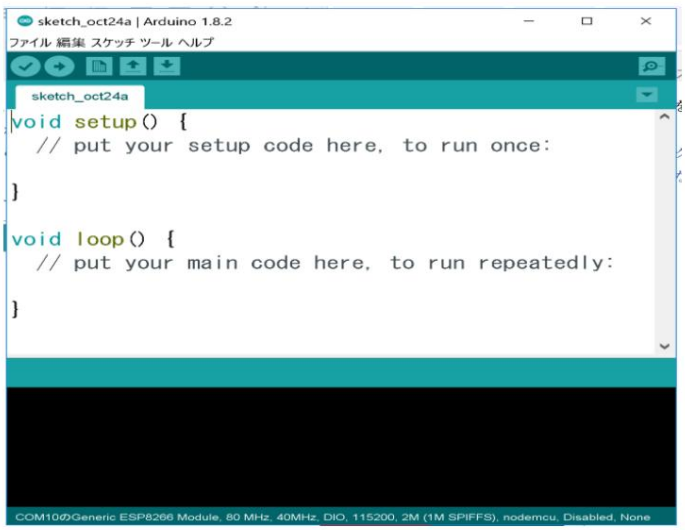
https://www.arduino.cc/en/Main/Software

When you select Windows Installer, a screen like the one below will appear, so click JUST DOWNLOAD. and start the download. (Note)

Although the amount is displayed, this is just a way to donate to the project developing the Arduino IDE; if you select JUST DOWNLOAD, there will be no charge.



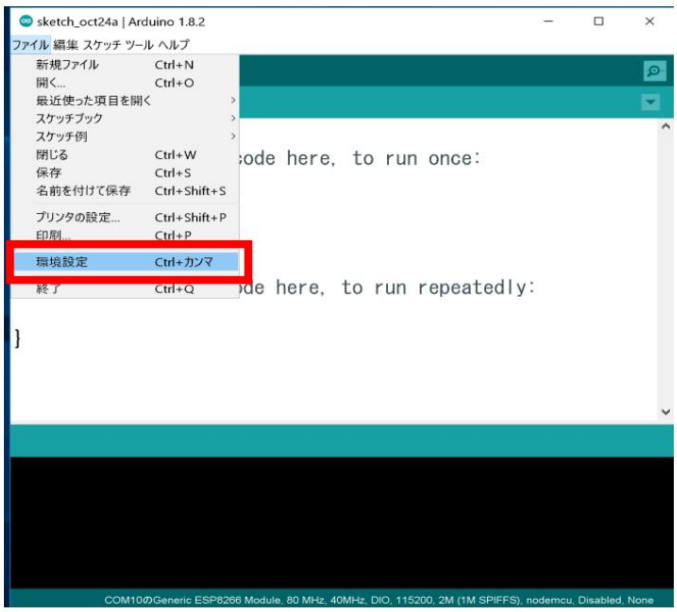
Double-click the downloaded arduino-xxxx-windows.exe to start the installer. After that, just follow the on-screen instructions and install it, and you're ready to go. Arduino IDE icon on the desktop Double-click , and if the following window is displayed, the installation has been successful.



(C) Enabling VS-RC202 to be programmed with Arduino IDE

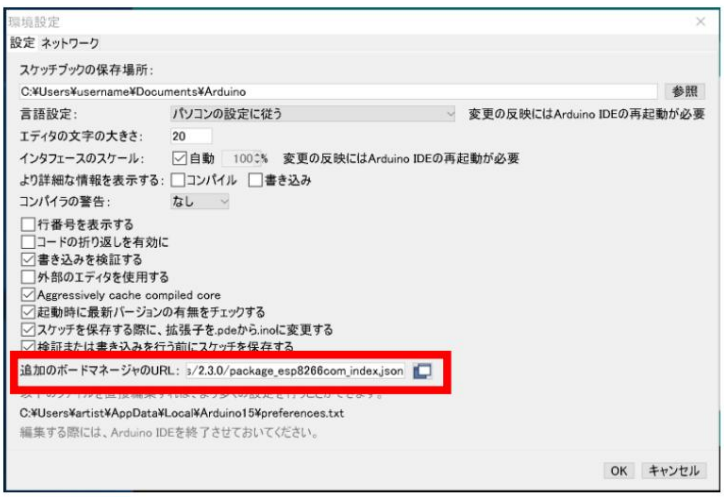
To be able to program the VS-RC202 with the Arduino IDE, install additional configuration files. must be

Start the Arduino IDE and select Preferences from the File menu.

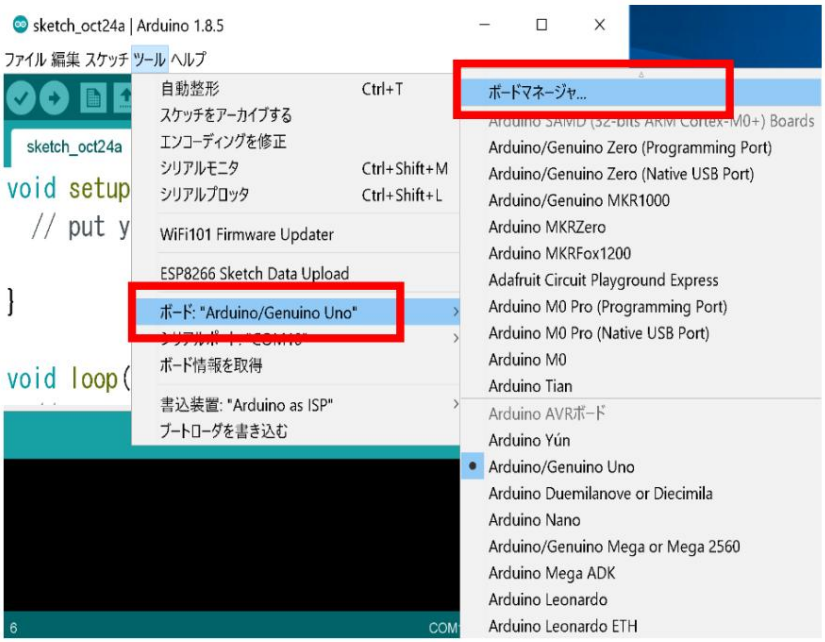


Paste the following URL into Additional Board Manager URL in the configuration panel and press the OK button.

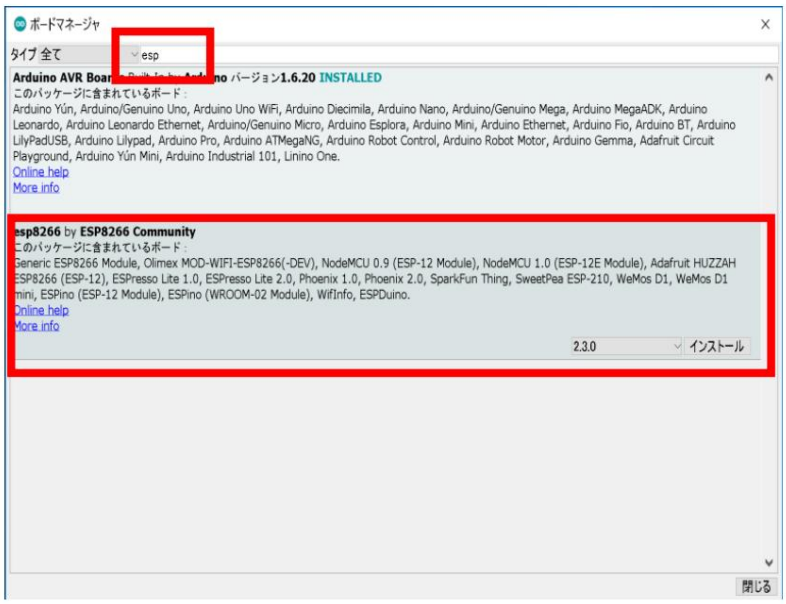
http://arduino.esp8266.com/stable/package_esp8266com_index.json



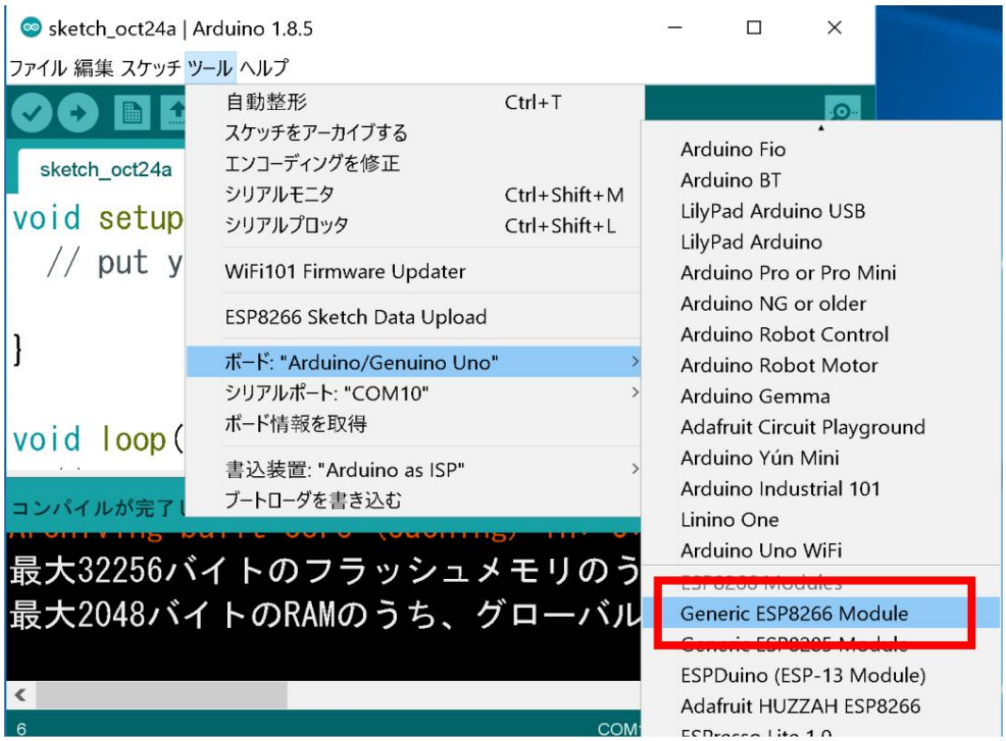
Select Board > Board Manager from the Tools menu.



When the board manager appears, enter "esp" in the search box at the top.
You will find esp8266 by ESP8266 Community, select the version and install it.
As of August 1, 2019, we have confirmed that it works with 2.5.2.



Once the Esp8266 has been installed, if the Generic ESP8266 Module can be selected from the Tools>Board menu, it has been successfully installed. (Scroll down on the board selection screen. (you can find it by rolling)



(D)Enable files other than programs to be written to VS-RC202

VS-RC202 can be connected to Wi-Fi and operates as a simple web server, distributing HTML files etc. You can If you are using Piccorobo IoT, you can display the VS-RC202's HTML controller on your smartphone's browser and operate Piccorobo itself.

Here, we will prepare to install the HTML etc. to be distributed on VS-RC202 in advance. URL below Access and download ESP8266FS-×××.zip. *The version name is entered in ×××.

As of June 13, 2019, we have confirmed that it works with 0.4.0.

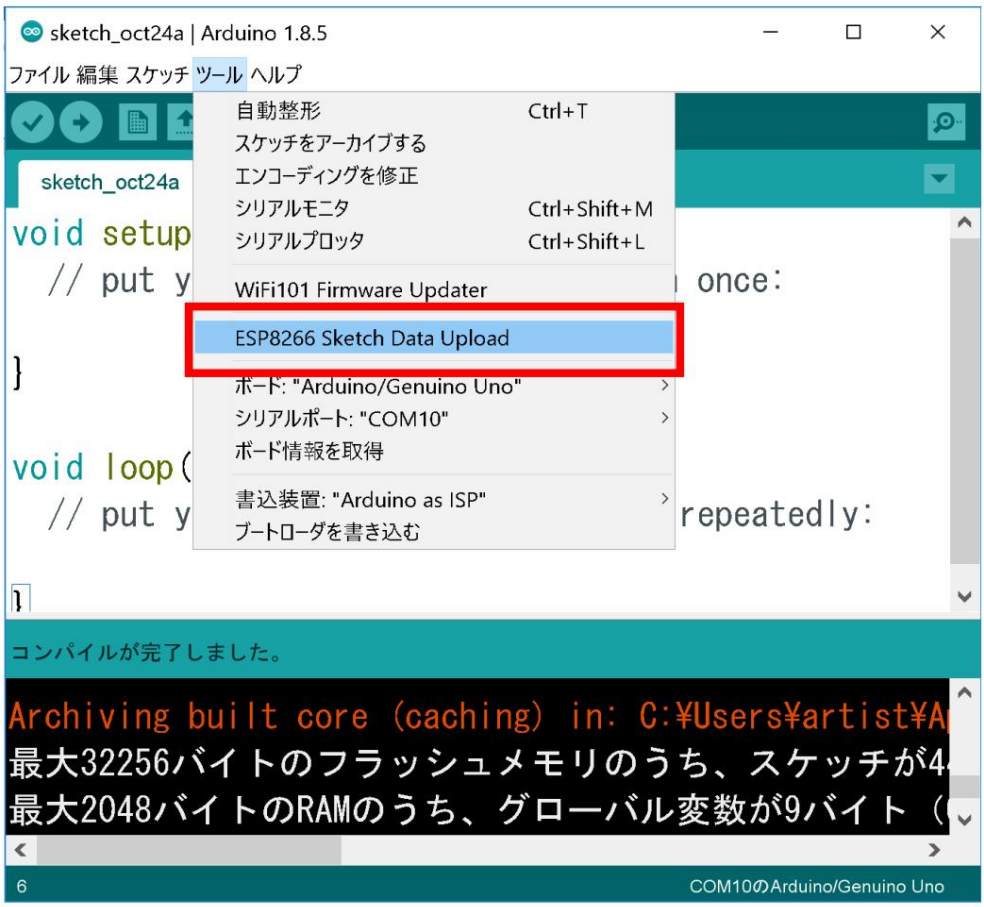
https://github.com/esp8266/arduino-esp8266fs-plugin/releases/

When you unzip ESP8266FS-×××.zip, a folder called ESP8266FS will be created. Arduino sketch Create a folder called tools in the folder and copy ESP8266FS into tools.

By default, the Arduino sketch folder is located directly under the document folder.
C:\Users\username\Documents\Arduino

Create a folder as shown below and copy it.
C:\Users\username\Documents\Arduino\tools\ESP8266FS\tool\esp8266fs.jar

Once copied, restart the Arduino IDE and if Tools > ESP8266 Sketch Data Upload is displayed in the menu, it has been successfully installed.



(E) Enabling the use of the VS-RC202 library

Download V-duino-ver.×××.zip from the URL below .

*The version name is entered in ×××.

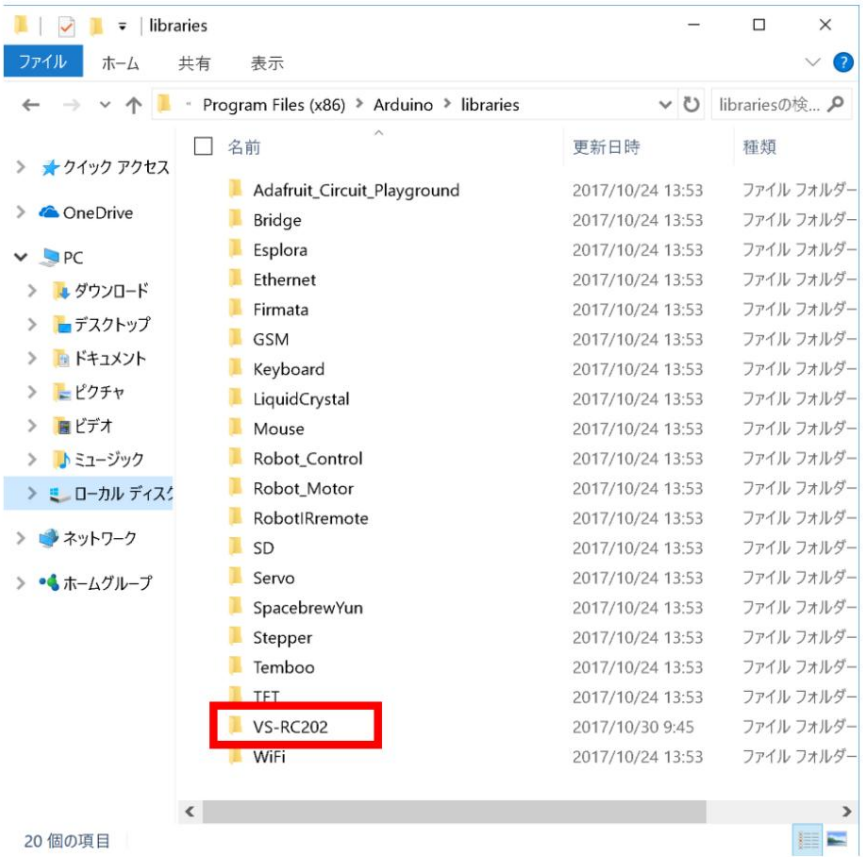
Please basically use the latest version.

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

When you unzip it, a folder called V-duino-ver.××× will be created. There is a file called VS-RC202 in this folder. There is a folder, so move this folder into the Arduino's libraries folder.

For Windows 10, the libraries folder is located at the following location.

'C:\Program Files (x86)\Arduino\libraries' or 'C:\Program Files\Arduino\libraries'



This completes the software setup.

5. Basic usage

Here we will explain the minimum things you need to know when using VS-RC202.

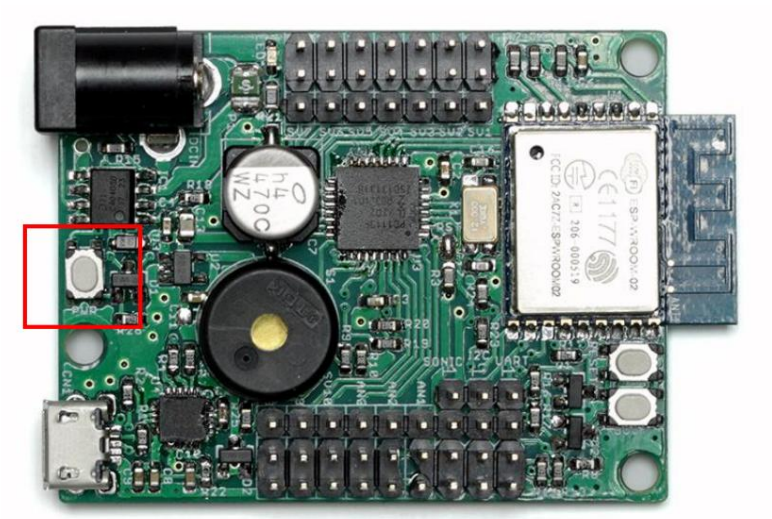
(A)About the power supply

Be sure to use a 5V AC adapter or 4 nickel metal hydride rechargeable batteries (4.8V) with the VS-RC202 . Never use a power supply higher than 5V, as it may cause damage to the board or servo motor.
please do not.

VS-RC202 is equipped with a power monitoring function to prevent over-discharge of the battery.
vinegar. By default, the power will be turned off when the supply voltage drops below 4.6V.

Therefore, if the power goes out even though you are using new batteries or an AC adapter, there is a possibility that a voltage drop occurs due to insufficient power output to the load, and as a result, the power OFF function is activated. .

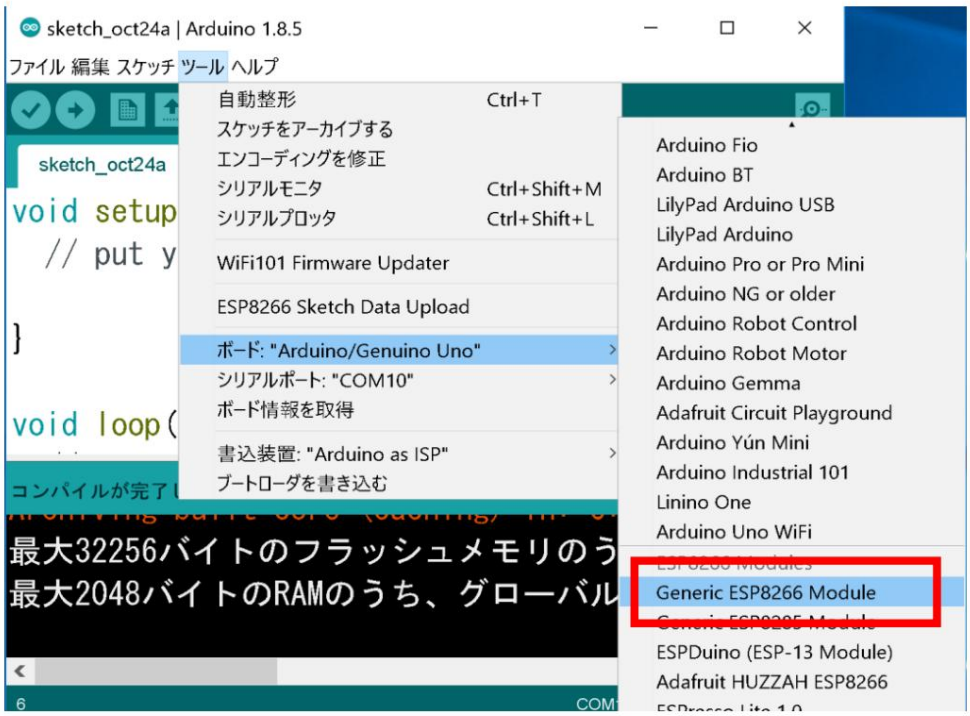
If you connect the USB, the power will turn on automatically, but if you want to start up using only the AC adapter or battery, press the power button shown in the photo to turn on the power. When the power is on, press and hold the power button for 3 seconds or more and then release your finger to turn the power off.



(B) How to write a program

Connect VS-RC202 to your PC using a USBmicroB cable.

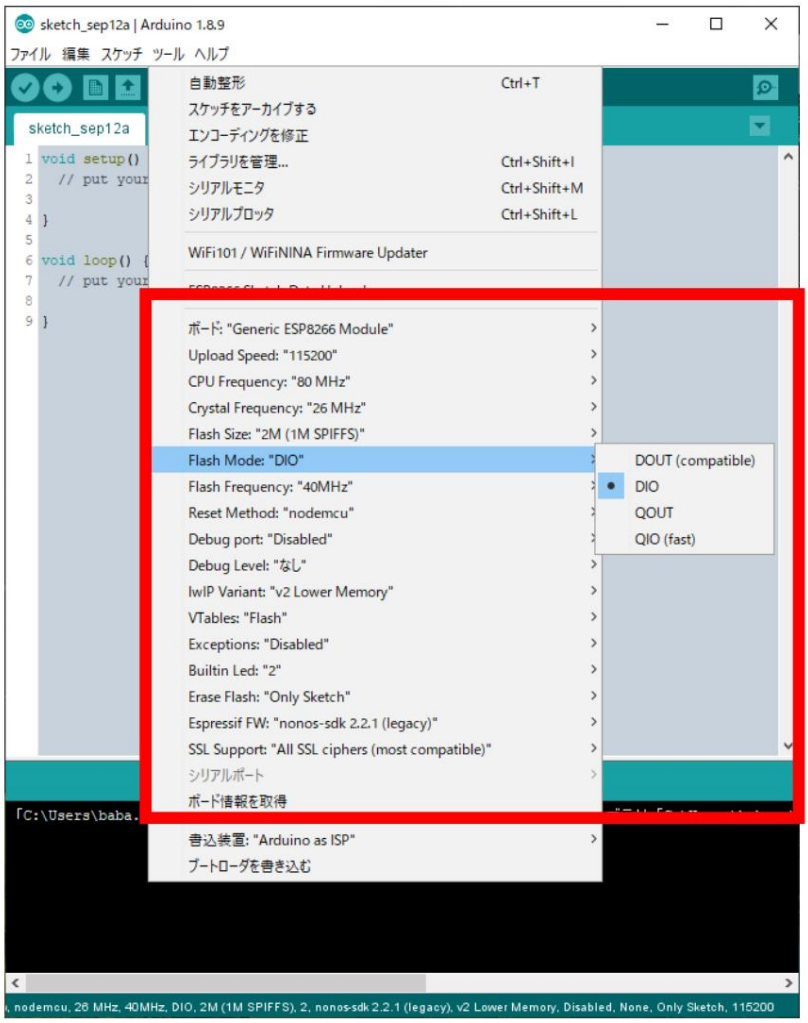
Select Generic ESP8266 Module from Tools > Board in the Arduino IDE menu.



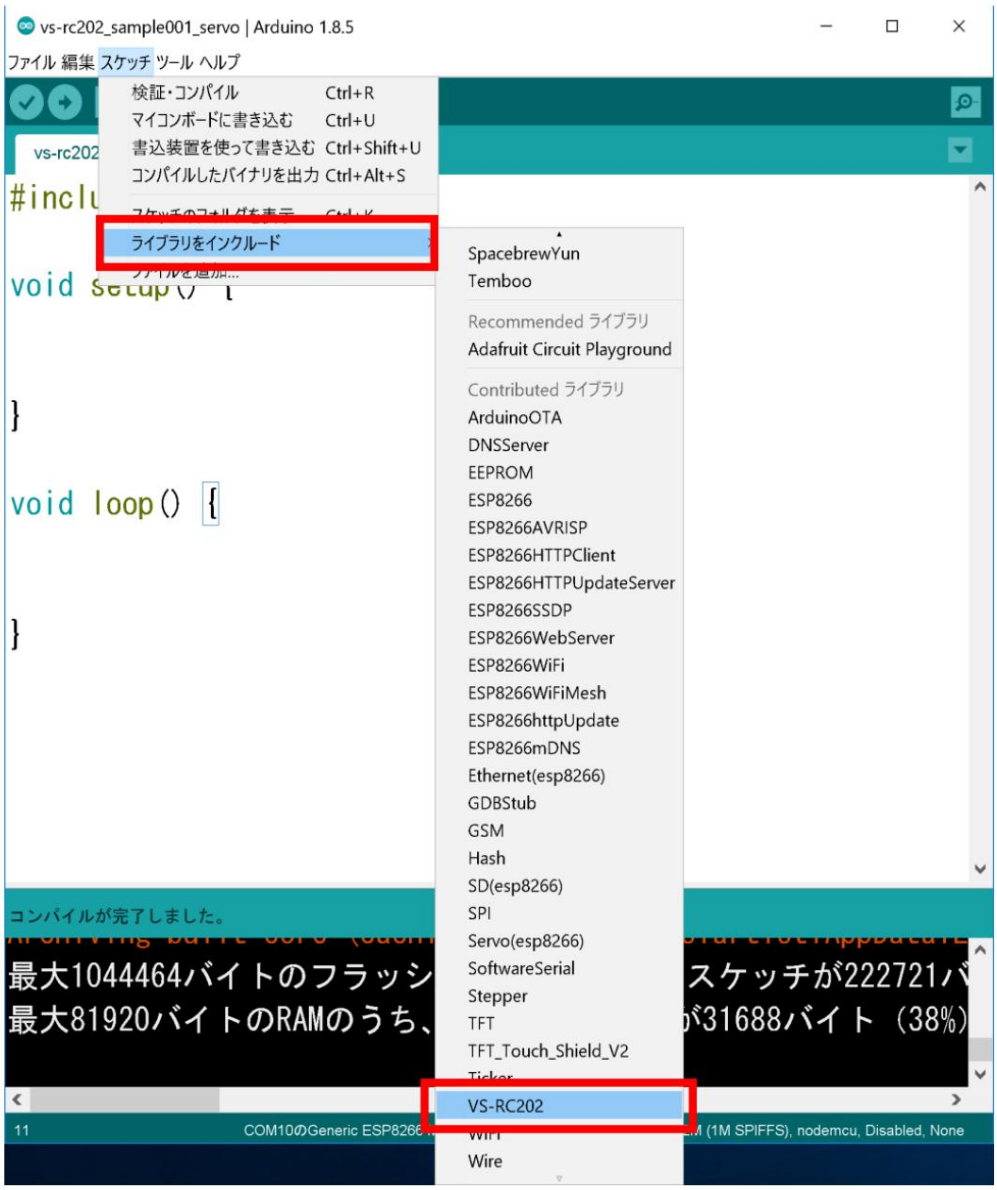
When you select Generic ESP8266 Module, multiple settings will appear in the tools menu, so use the following settings.

Set it like this: Once set, it will be automatically selected from next time onwards.

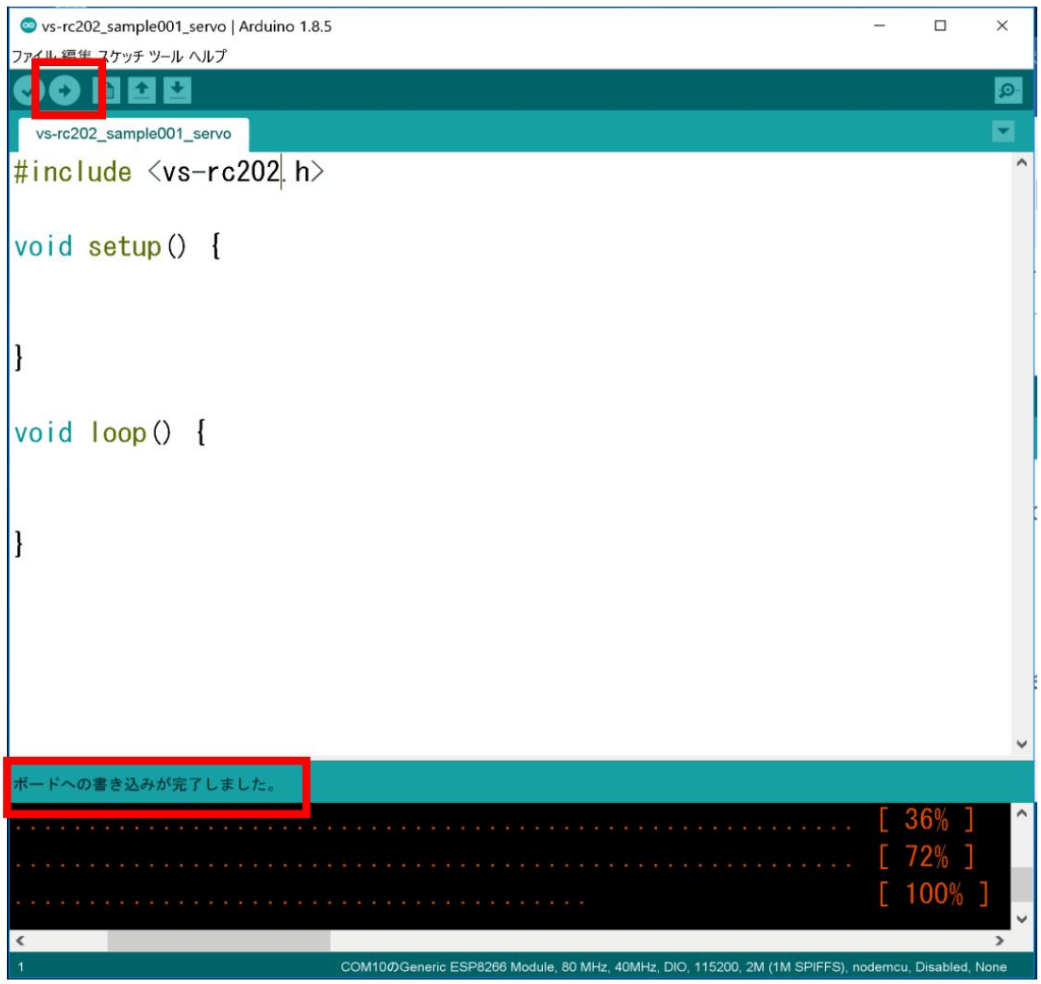
ÿ Flash Mode	:DIO
ÿ Flash Frequency	: 40Mhz
ÿ CPU Frequency	: 80Mhz
ÿCrystal Frequency	: 26Mhz
ÿ Flash Size	: 2M(1M SPIFFS)
ÿ Debug port	: disabled
ÿ Debug level	: none
ÿReset Method	: nodemcu
ÿ Upload Speed	: 115200
ÿ Serial port	: Port to which VS-RC202 is connected



Next, include the VS-RC202 library. Ink library from menu sketch
Select Route > VS-RC202 to include it. When using VS-RC202, include the library every time you create a project.

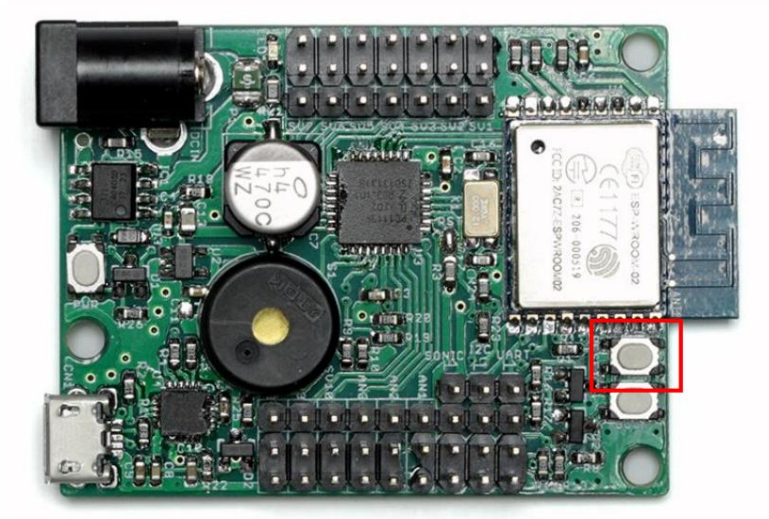


Connect the VS-RC202 to the PC, select the VS-RC202 port from the tools menu, and press the write sketch button (right arrow button). If the message “Writing to the board has been completed” is displayed, writing is successful. If an error occurs, the board settings may be incorrect or library inclusion may have failed. Please check the steps again.



(C)Reset

The button next to ESP-WROOM-02 is the reset button. When you press the reset button, ESP-WROOM-02 will restart. Use this when you want to run the program again.



(D) Boot mode

The button next to the reset button is the boot button. Reset Method of Arduino IDE Tool Settings If you set it to "ck", you can write the program by pressing the reset button while holding down the boot button. (enter boot mode manually).

If there are no circumstances such as some parts of the board being broken and the program not being able to be written with the "nodemcu" setting. There is no need to use it.

6. How to use servo motor

Now, let's move the servo motor.

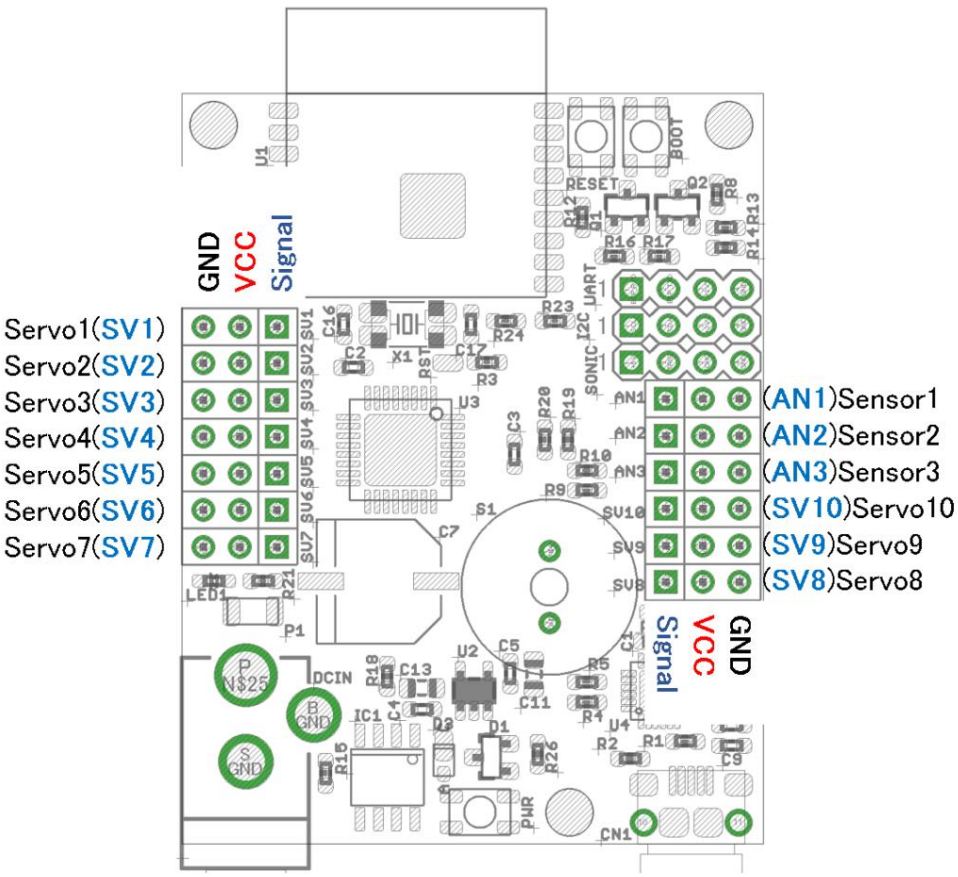
(A) Servo motor pin arrangement

The pin arrangement of VS-RC202 is as follows. When connecting a servo motor, use the following Please note the difference. It will be the cause of the failure.

ÿ Do not connect Signal (signal line) and GND in the opposite direction. ÿ Do not connect to anything other than servo motor pins (SV1-SV10) by mistake.

In the following explanation, each pin of VS-RC202 will be described by the name shown in the figure below.

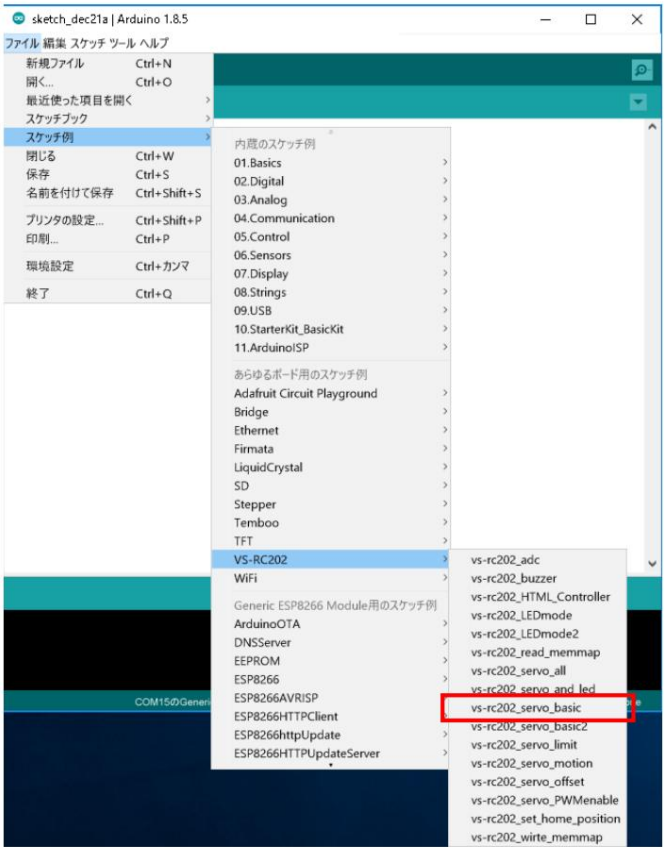
(Example) " Please connect the servo motor cable to the SV1 pin"



(B) Move the servo motor

Let's start moving the servo motor.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_basic from the Arduino IDE menu. please.



After opening the sketch, connect the VS-RC202 and PC with a USB microB cable, and write the sketch. please.

After writing the sketch, connect the servo motor to SV1 and connect the battery box to VS-Connect to the DC jack of RC202. Then press the power button. If the sketch has been written correctly, the servo will start moving left and right.

Now, let me explain a little about the contents of the sketch.

First, let's take a look inside setup(). The inside of setup() is executed only once at startup. Normally
Write settings reading and initialization processing. Be sure to initialize the servo motor settings etc. in setup().
Call initLib().

In the VS-RC202, the servo motor is in a weak state even when the power is turned on. PWM with servoEnable() function
Turn on the signal.

The setServoMovingTime() function specifies the servo motor transition time. The unit is milliseconds.
setServoMovingTime(1000) means move to the target position in 1000 milliseconds from now on.
Become.

```
#include <vs-rc202.h>                    //Include library

void setup() {

    initLib();                           //Library initialization, always run first
    servoEnable(1, 1);                   //Enable PWM signal sent to SV1
    setServoMovingTime(1000); //Set the servo motor transition time
}
```

[Function description]

Format: initLib();

Initialize the VS-RC202 library. Execute once in Setup().

Format: servoEnable(pin number, ONOFF flag);

Example 1: servoEnable(1, 0); Turn off the PWM signal of SV1

Example 2: servoEnable(2, 1); Turn on the PWM signal of SV2

Format: servoServoMovingTime (milliseconds);

Example 1: setServoMovingTime (500); After that, the target angle is reached in 500ms

Example 2: setServoMovingTime (1000); After that, the target angle is reached in 1000ms

Next, let's take a look inside loop() (some parts are omitted). Inside loop(), it is executed repeatedly.

Write the process to do so. This time, we want to keep the servo motor moving, so we write the servo motor movement inside loop().

I will write

First, use the setServoDeg() function to set the position you want SV1 to move to (range: -1800 to 1800). Next

Move the servo motor using the moveServo() function. In this code, ``Specify the target position and move

This command is repeated.This time, the next command is sent after the servo motor has rotated completely.

Therefore, we put a delay() function in between.

```
void loop() {


    setServoDeg(1, 0);              //Set the target position of SV1 to 0
    moveServo();                    //Move the servo motor
    delay(1200);                    //Stop for 1200 ms
                :
}
```

[Function description]

Format: setServoDeg (pin number, position);

Example 1: setServoDeg(1, 0); //Set the target position of SV1 to 0

Example 2: setServoDeg(2, 1500); //Set the target position of SV2 to 1500

Format: moveServo ();

When this function is executed, the servo motor will start moving.

ÿsummaryÿ

1. Inside setup() is initialization processing, inside loop() is repetition processing

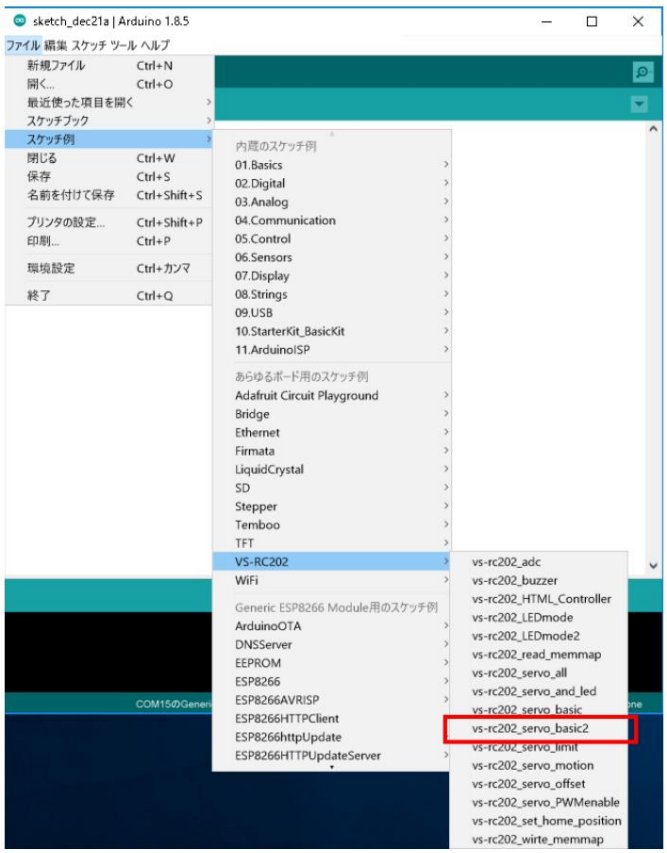
2. Initialize the library and enable the PWM signal in setup()

3. To move the servo motor, it is necessary to set the transition time, set the target angle, and start.

(C) Move multiple servo motors

Next, let's move multiple servo motors at the same time.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_basic2 from the Arduino IDE menu. please.



After opening the sketch, connect the VS-RC202 and PC with a USB microB cable, and write the sketch. please.

After writing the sketch, connect the servo motor to SV1, SV2, SV3, and SV4 and connect the battery. Connect the box to the DC jack of VS-RC202. Then press the power button. If writing is successful, SV1,3 and SV2,4 should move symmetrically.

Now, let me explain a little about the contents of the sketch.

Inside setup() is the same as last time, only initialization and enabling the PWM signal. The only difference is
The setServoMovingTime() function is moving inside loop().

Well then. Let's take a look at loop() (some parts are omitted).

You can see that setServoMovingTime(), setServoDeg(), and moveServo() are repeated in order.
Masu. Change the moving speed of the servo motor by setting setServoMovingTime() every time.
can do.

Also, input the target positions of multiple servo motors using the setServoDeg() function, and then use the moveServo() function.
By executing this, it is possible to move multiple servo motors at the same time.

```
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
            :
}
    delay(700);
```

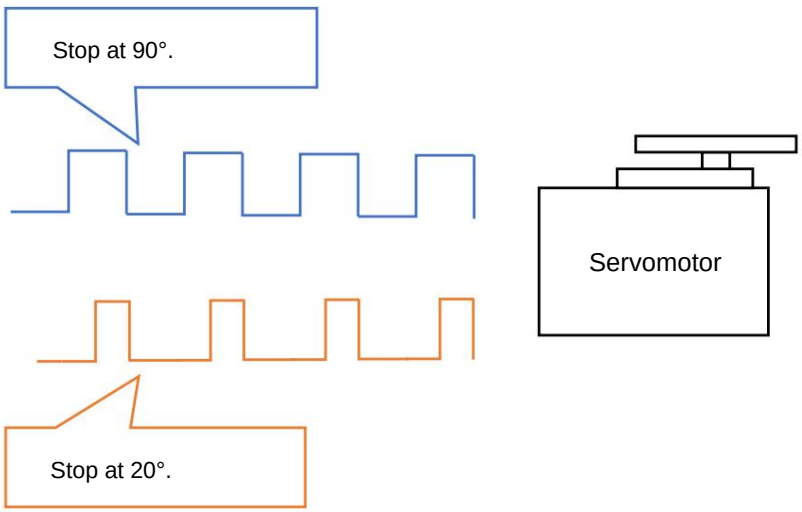
ÿsummaryÿ

1. You can change the servo speed by executing the setServoMovingTime() function every time.
2. If you set multiple servo motors using the setServoDeg() function and execute the moveServo() function, you can
   Can operate several servo motors at the same time

(D) Mechanism of servo motor operation

Here, in order to better understand the program, we will explain the mechanism of servo motor operation based on the sketches used so far.

A servo motor stops the motor shaft at a specific position by sending pulses called PWM signals.
You can. By changing the ratio of the HIGH portion of this pulse (called the duty ratio), the stopping position changes.



The PWM signal stops the servo motor shaft at a specific position and moves it at any speed.
It's not a thing. If you want to rotate at any speed, use the following method.

The period of the servo motor's PWM signal is approximately 20 milliseconds. Therefore, if you change the duty ratio slightly every 20 milliseconds, the stopping position of the motor axis will shift slightly, making it appear as if the servo motor is rotating slowly.

Using the sample sketch function as an example, the flow is as follows.

servoEnable(1, 1)

The PWM signal turns ON and the servo motor axis points to the center position as the initial position.

setServoMovingTime(1000)

setServoDeg(1, 1800)

moveServo();

Move to position 1800 with a transition time of 1000 ms. The position changes every 20 milliseconds, so the stop position changes 50 times.

(Transition time – PWM period) = number of changes

1000msec/20msec = 50

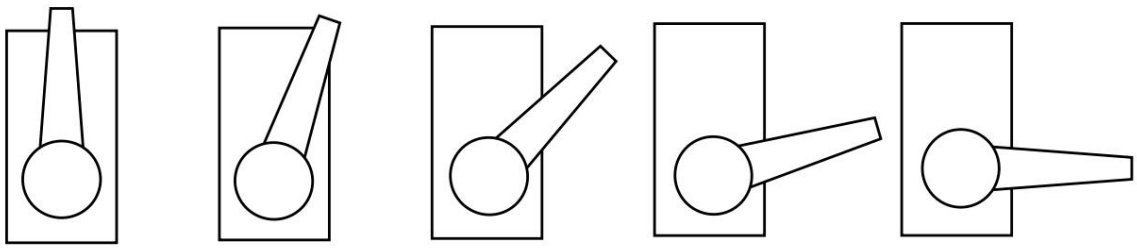
One position change is 36.

(Target position – Start position)/Number of changes = Amount of change per time

(1800 - 0)/50 = 36

(Note) In the VS-RC202 library, in order to precisely control the position, specify the position with a value between -1800 and 1800 instead of an angle between -90 and 90°.

Rather than rotating a motor, it feels more like determining the start and end positions like a flip book. It's like deciding the speed at which the pages turn. Did you get the image?



Position:0

Time: 0ms

Position:360

Time: 200ms

Position:900

Time: 500ms

Position: 1440

Time: 800ms

Position: 1800

Time: 1000ms

Note that changing the position little by little toward the target position in this way is called interpolation, but VS-

In RC202, the main processing is done not by the main CPU, ESP-WROOM2, but by the ARM chip. So

Therefore, ESP-WROOM2 only informs the ARM chip of the target position and transition time, and performs communication processing etc.

Now I can concentrate on it.

ÿsummaryÿ

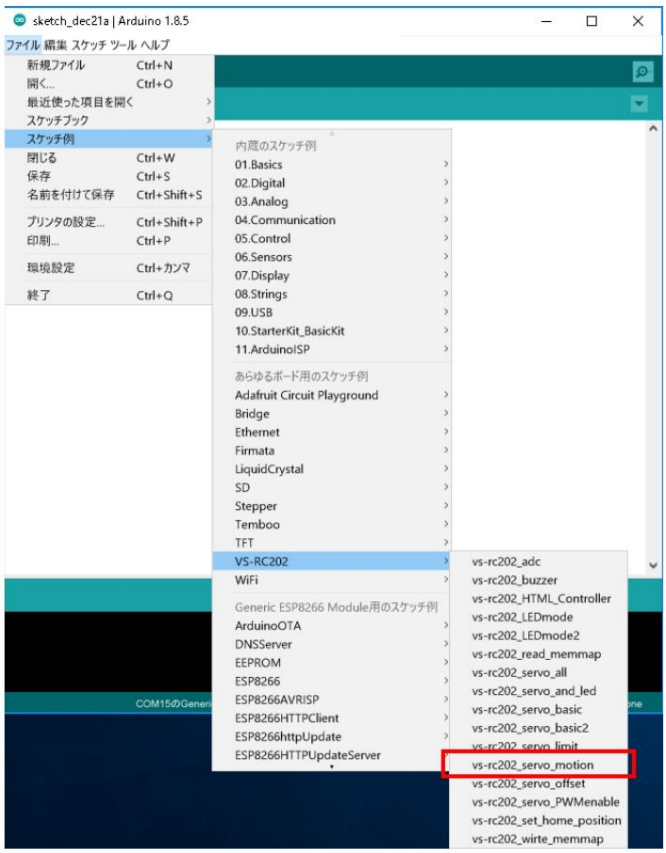
1. Servo motor is driven by PWM signal
2. The period of the servo motor PWM signal is 20 milliseconds.
3. The servo motor is controlled using a flipbook method that changes the stop position little by little.

(E) Play motion

Let's take a little more in-depth explanation of how to control a servo motor. Have you ever seen a robot arm or a bipedal robot? These robots basically perform a single action by moving multiple servo motors in a flip-floppy manner. For example, when a bipedal robot walks, the servo motors all over its body change the angle one after another to strike a pose. This series of movements of the servo motor is called a motion.

Here we will explain how to play motion using VS-RC202.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_motion from the Arduino IDE menu. please.



After opening the sketch, connect the VS-RC202 and PC with a USB microB cable, and write the sketch. please.

Once you have finished writing the sketch, connect the servo motors to SV1, SV2, SV3, and SV4.), connect the battery box to the VS-RC202's DC jack.

Open the Arduino IDE's serial monitor. Set the baud rate to 115200. Then press the reset button on VS-RC202. You will see some characters that you don't understand at first, but don't worry about them because these are the characters that ESP-WROOM-2 outputs when it starts up.

Type w, a and press Enter to move the servo motor several times. If you type s, d and press the Enter key, the servo motor will continue to move. If you type x and press the Enter key, nothing will happen, but the motion will still play.

Let's take a closer look at the program contents. There is an array near the beginning of the sketch, which is
It's motion. The contents inside {} of the array are as follows. {Transition time,
SV1 target position, SV2 target position, ..., SV10 target position}

This array is a two-dimensional array, and it is an image that is executed in order.
(Note) Since SV5-SV10 is not connected to a servo motor, the target position is always set to 0.

```
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},
                    };
                            :
```

The getCommand() function inside Loop() is used to receive input from the serial monitor on the previous page.
Masu. Set the motion to be played using the setMotionNumber() function according to the characters received here.

Then, use the selectMotion() function with the motion you want to play and the number of elements in the first dimension of the motion as arguments.
and play it using the playMotion() or playMotionOnce() function.

playMotion() continues playing the motion until the next instruction. playMotionOnce() only once
Play motion.

```
void loop() {
    getCommand(); //Select the motion to play with serial input
    selectMotion(); //play motion
}
```

In the example of the sample sketch, the flow would be as follows. Actually run the program and copy

Let's check that the motion is playing according to the command.

1. Receive 'w' from serial monitor (getCommand())
2. motion_number = M_NUM1 is set (getCommand())
3. Since motion_number = M_NUM1, play motion1 (selectMotion())

[Function description]

Format: setMotionNumber(motion number);

Set the motion number you want to play.

Example: setMotionNumber(1); //Set motion number 1

Format: getMotionNumber();

Get the currently set motion number.

```
Example: switch(getMotionNumber()){
        case 1:                                    //If the motion number is 1...
                        :
```

Format: playMotion(motion, number of motion elements);

Format: playMotionOnce(motion, number of motion elements);

Example 1: playMotion(motion1, //Continue playing motion 1

3); Example 2: playMotionOnce(motion3, 5); //Play motion 3 once

(Note) If you want to know more detailed operation, please refer to the source code of VS-RC202.cpp located in the VS-RC202 folder.

Let's take a look.

The basic idea of how to create a motion is to connect intermediate poses of the robot you want to move. For example, if you want to lift something with a 4-axis robot arm, it would be good to have at least the following three poses.

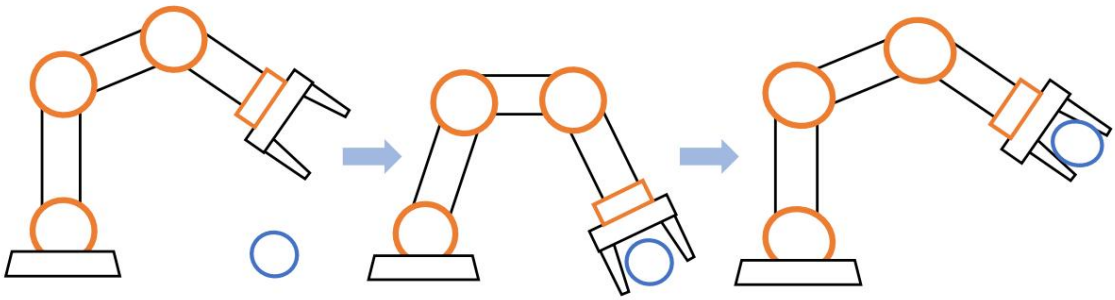
ÿ First pose ÿ Pose

where the arm is lowered and grabbed

something ÿ Pose where the arm is raised and picked up something

In other words, if you determine the position of each joint in these three poses and the transition time between each pose, you can I think I can make a motion to lift it up.

```
Arm_motion[3][11] = {
                {600,0,0,0,0,0,0,0,0,0,0},                  //first pose
                {600,400,600,400,600,0,0,0,0,0,0}, //second pose
                {1000,0,0,0,600,0,0,0,0,0,0},               //last pose
};
```



Let's create your own original robot and create its own motions!

[Summary]

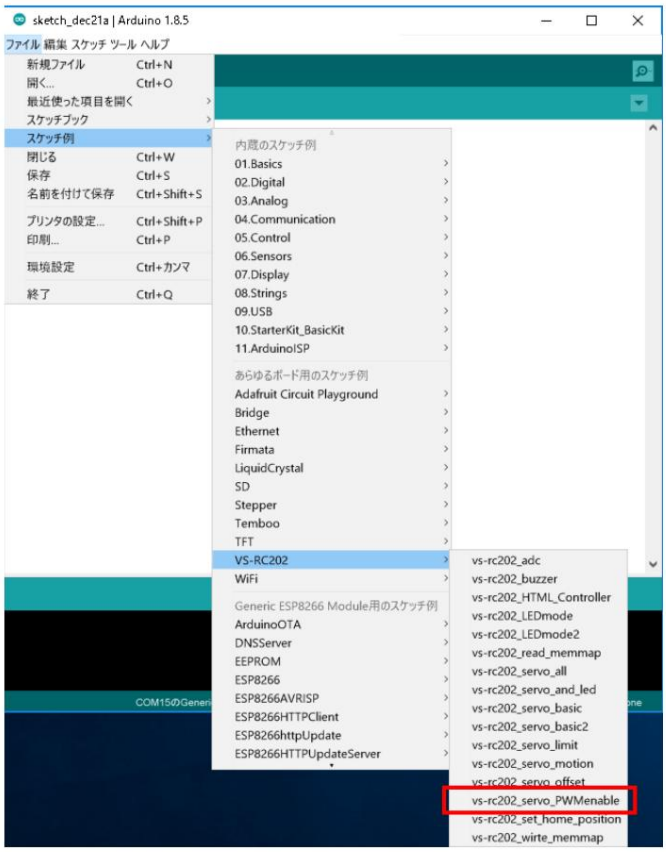
1. Motion is a collection of poses

2. Motion can be created by deciding the pose of the robot you want to move and the transition time between poses.

(F) Relax the servo motor

If a robot keeps moving its servo motors, its battery will quickly run out. as much as possible
It's good to save energy. This section explains how to turn the servo motor signal ON/OFF.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_PWMenable from the Arduino IDE menu.
Please choose.



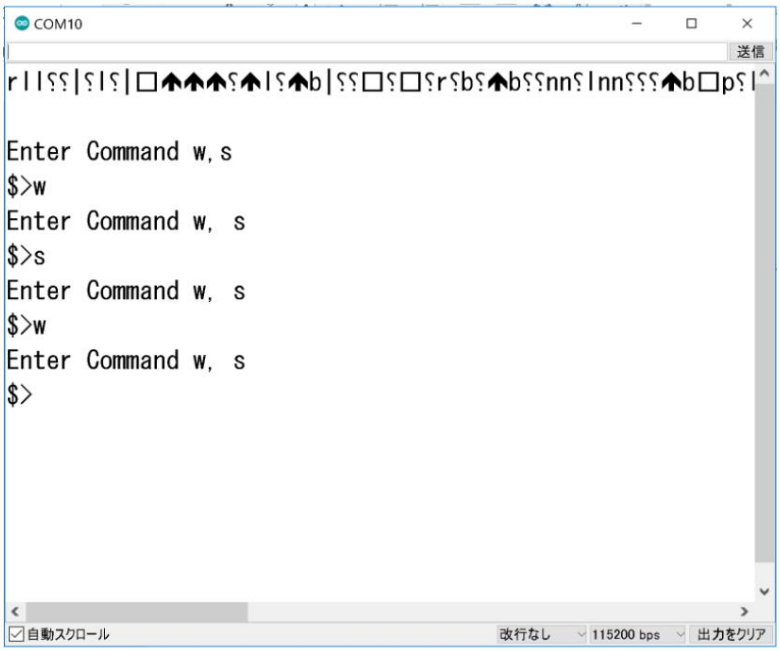
After opening the sketch, connect the VS-RC202 and the PC with a USB microB cable and start writing the sketch.
Please.

After writing the sketch, connect the servo motors to SV1, SV2, SV3, and SV4 , and connect the battery box to the DC jack of VS-RC202.

Open the Arduino IDE's serial monitor. Set the baud rate to 115200. And VS-

Press the reset button on RC202. Enter 'w' to apply force to the motor, and enter 's' to release it.

I will do my best.



The servoEnable() function switches the servo motor's PWM signal ON/OFF. For example, starting a robot

In situations where it is not needed, such as when the servo motor is running or not operating, reduce the power of the servo motor as much as possible to save battery power.

You can make your terry last longer.

However, be sure to keep the robot safe so that it does not lose its strength and cause accidents such as falling or falling damage.

Get into a full posture and then relax.

[Function description]

Format: servoEnable(pin number, ON/OFF); Example

1: servoEnable(1, 0); Example //Turn SV1 OFF (weakness)

2: servoEnable(1, 1); //Turn on SV1 (PWM signal enabled)

[Summary]

1. Switch the servo motor ON/OFF using the servoEnable() function

2. Extend the battery life by turning it off as much as possible when it is not needed.

(G) Setting the servo motor offset

Here we will explain the offset of the servo motor. Even if you install

the servo horn with the servo motor axis centered (position 0), the servo horn will not stay in place.

Sometimes it doesn't straighten and ends up facing in the wrong direction.

This is a difference caused by individual differences in servo motors, etc., but when the robot is assembled, the legs and hands are

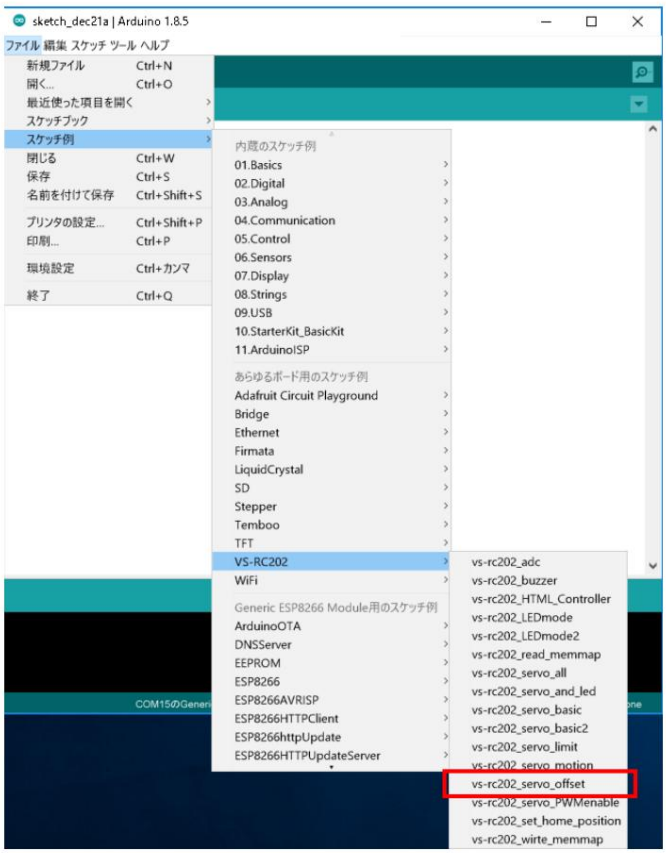
You don't like it if it's a little crooked, right?

Therefore, by shifting the PWM signal that is the center angle of each servo motor in the program, the servo horn

straighten. This shifting value is called an offset.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_offset from the Arduino IDE menu.

please.



After opening the sketch, connect the VS-RC202 and the PC with a USB microB cable, and write the sketch.

Please. After writing the sketch, connect the servo motor to SV1, SV2, SV3, and SV4 , and connect the battery box to

the DC jack of VS-RC202. Isn't the initial position of the servo motor slightly different from before?

In the sketch's setup(), a function called setServoOffset() is called. This sets an offset for each servo motor. For example, setServoOffset(1,

100) means to shift the servo motor position of SV1 by 100. From now on, if you call setServoDeg(1, 500), the signal setServoDeg(1, 600) will actually be output. In this way, by using offset, it is possible to correct individual differences in servo motors using software.

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

```
[Function
description] Format: setServoOffset(pin number, offset); //Setting range: -500 to 500
Example 1: setServoOffset(1, 500);          //Shift the center position of SV1 by 500
Example 2: setServoOffset(1, -500);         //Shift the center position of SV1 by -500
```

[Summary]

1. Due to individual differences in servo motors, the stop position may vary even with the same PWM signal.

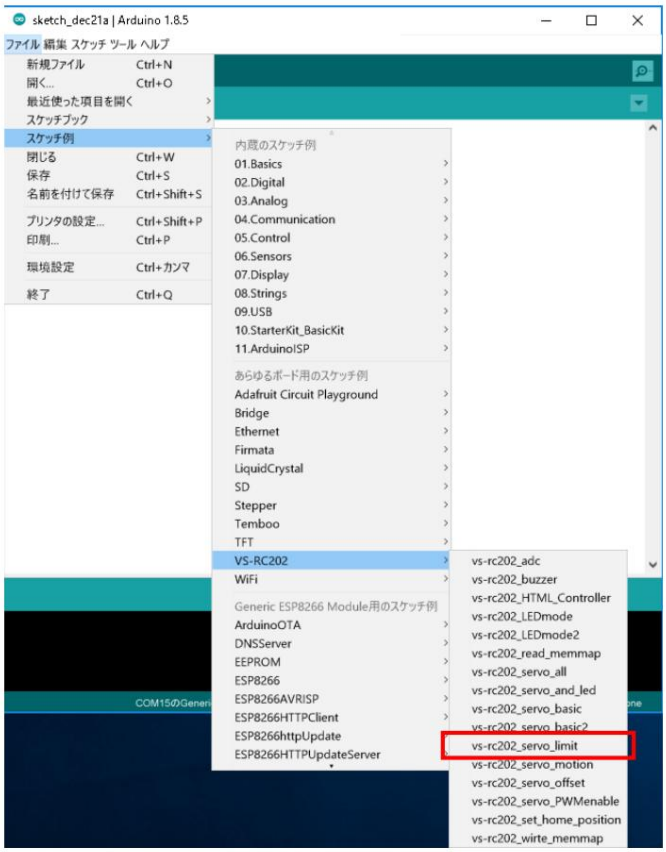
2. By setting an offset, you can adjust the stopping position using software.

(H) Setting the limit angle of the servo motor

Along with the offset of the servo motor, the other thing to remember is the limit angle. Most servo motors are Rotate approximately 90° left and right from the center. However, if you input a PWM signal outside the original movable range, Doing so may cause the servo motor to become locked and malfunction.

VS-RC202 allows you to set the limit value of the PWM range that can be input to the servo motor using software. There is a mechanism to prevent servo motor failure. Now let's take a look at how to actually set it up.

Select File > Sketch Example > VS-RC202 > vs-rc202_servo_limit from the Arduino IDE menu. please.



After opening the sketch, connect the VS-RC202 and the PC with a USB microB cable, and write the sketch. Please.

After writing the sketch, connect the servo motor to SV1 and connect the battery box to VS-RC202. Connect to the DC jack of RC202.

Looking at the movement, the servo motor shaft rotates left and right, but sometimes it rotates a lot and sometimes it rotates a little.
I realize that there are times when I do. This is because the limit movement range of the servo is changing. Details
Let's take a closer look.

The sketch's loop() executes the functions setServoLimitL()/setServoLimitH(). This is sa
This is a function that sets the limit movement range of the motor in the negative and positive directions. The servo motor is
It cannot be rotated beyond the range of motion. The sample code tries to rotate at the target position -1800 to 1800.
However, if the limit movement range is set to -800 to 800, the movement range will also be -800 to 800.

```
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);                    //Negative direction limit position=-1800
    setServoLimitH(sv_limit);                     //Positive limit position=1800
    setServoMovingTime(sv_time); //Transition time = half of the limit angle
                :
```

[Function description]

Format: setServoLimitL(negative direction limit movement range); //Setting range: -2500 to 2500 Format:
setServoLimitH(positive direction limit movement range); //Setting range: -2500 to 2500 Example 1:
setServoLimitL(-1500) ; Example 2: //Negative direction limit movement range=-1500
setServoLimitH(800); //Maximum movement range in the positive direction = 800

[Note 1] The movable range varies depending on the servo motor. Servo motor with narrow operating range
If the limit angle is too large, it may cause locking, so please set it carefully.
[Note 2] The maximum movement range of the servo motor is the offset + limit movement range.
setServoOffset(SV1, 500); setServoLimitH(2500);, SV1 will operate up to 3000.
I will work.

[Summary]

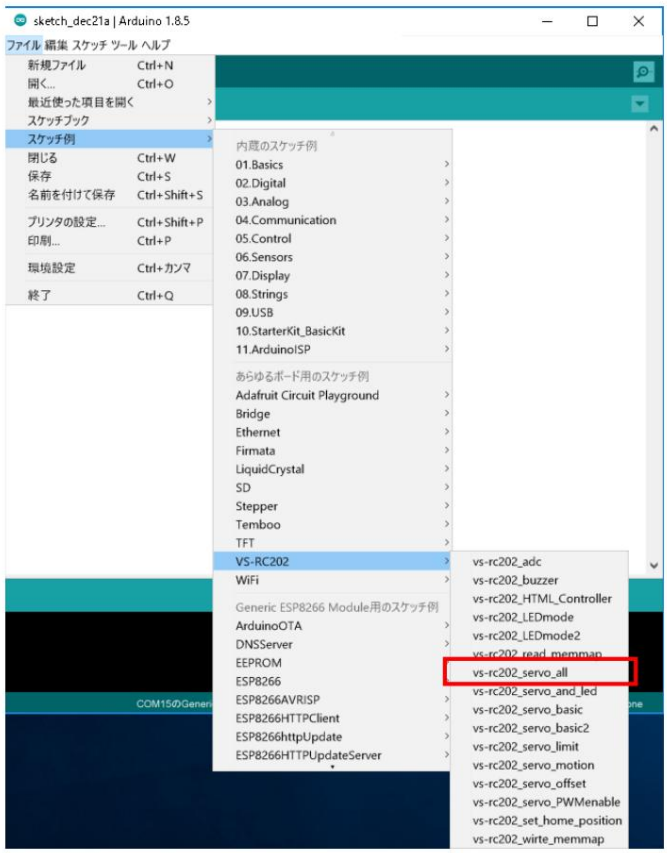
1. Servo motors only rotate within their limit range of motion.
2. The movable range varies depending on the servo motor. Be careful when setting the limit range of motion
3. The maximum movable range of the servo is the offset + limit movable range

(I) Insert a set of processing

Up to this point, you have learned how to control a servo motor, but if you want to actually write the code, The flow will be as follows.

```
// Motion and function definition
int motion1[][] =       …;
getCommand(){};
selectMotion(){};

//Initialization
void setup() {
    //Initialize the vs-rc202 library //Set
    the limit movement range of the servo
    motor //Set the offset of the servo motor //
    Enable PWM of the servo motor (arbitrary timing)
}

//loop
void loop(){
    //Read command //
    Execute motion
}
```

Let's actually run the process on the previous page. File > Sketch in the Arduino IDE menu. Select Example > VS-RC202 > vs-rc202_servo_all. Once you open the sketch, connect the VS-RC202 to your PC using the USBmicroB cable and write the sketch.



After writing the sketch, connect the servo motor to SV1, SV2, SV3, and SV4 and connect the battery. Connect the box to the DC jack of VS-RC202. Open the Arduino IDE's serial monitor. Set the baud rate to 115200.

Type w, a and press Enter to move the servo motor several times. Type s, d and press Enter to keep the servo motor moving. If you type x and press the Enter key, nothing will happen, but the motion will still play.

This completes the control of the servo motor.

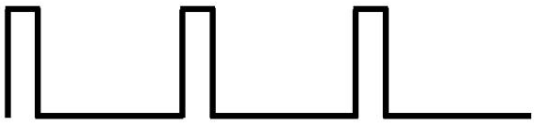
7.LED mode

(A)What is LED mode?

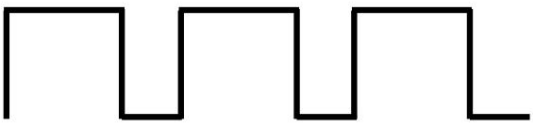
LED mode is a mode in which an LED is connected to the servo motor pin. Servo motor is
Since the duty ratio of PWM is very low, the same signal as the servo motor cannot make the LED shine brightly.
not. In LED mode, it is possible to increase the duty ratio and make the LED shine brightly.

Never connect a servo motor to the LED mode pins as this may cause damage.
Please.

The PWM duty ratio of the servo motor is low.



LED can be used even if the duty ratio is high.



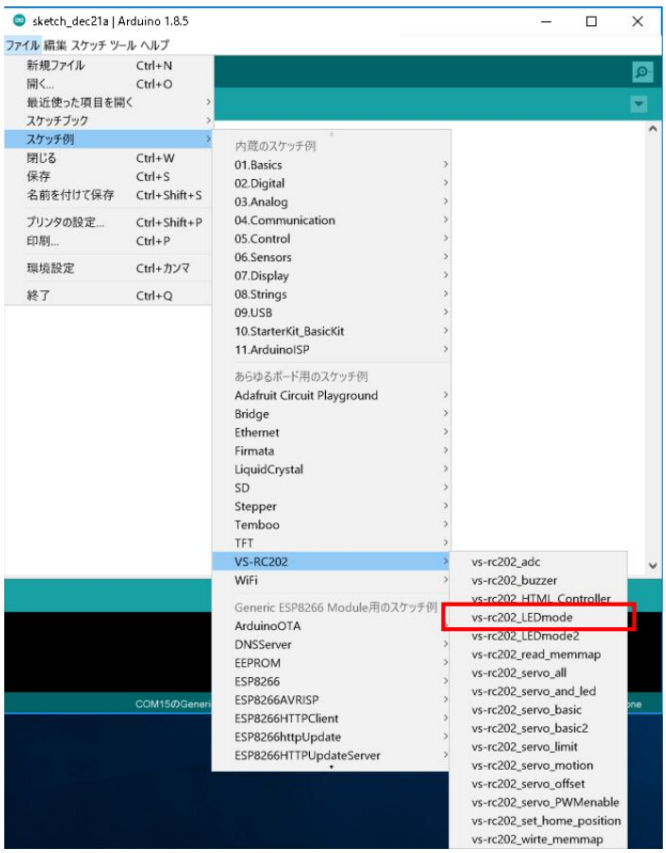
(B) LED connection

Since a certain amount of current is required to make an LED shine, it is usually not directly driven by a microcontroller.
A FET or a dedicated LED driver IC is used in between. VS-RC202 is suitable for one or two small LEDs.
You can control the LED directly from the servo motor pin, but if you connect more than that, you need to drive the LED.
Please use a FET that is compatible with the above.

It's easy with the OctopusLED light brick (blue), which has a FET and an LED. at the robot shop
Available for purchase.
https://www.vstone.co.jp/robotshop/index.php?main_page=product_info&manufacturers_id=97&pr
oducts_id=4356

(C) Control LED like a servo motor

Select File > Sketch Example > VS-RC202 > vs-rc202_LEDmode from the Arduino IDE menu.
vinegar. Once you open the sketch, connect the VS-RC202 and your PC using the USB microB cable, and
write the sketch.



After opening the sketch, connect the VS-RC202 and the PC with a USB microB cable, and write the sketch.
Please.

After writing the sketch, connect the LED to SV1 and connect the battery box to VS-RC202.
Connect to DC jack.

Did you see the LED flashing slowly? In LED mode, depending on the transition time

The brightness of the LED will change. By setting the target brightness and transition time, you can gradually increase the brightness of the LED.

You can change the degree.

```
void setup() {
    initLib();                                  //Initilize 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);
}
```

[Function description]

Format: setLedMode(pin number, enable/disable flag);

Format: setLedBrightness(pin number, brightness); //Brightness setting range: 0 to 1000

Example 1 setLedMode(10, 1); //Set SV10 to LED mode

setLedMode(10, 0); // Set SV10 to normal mode

Example 2: setLedBrightness(10, 1000); //Set the brightness of LED10 to 1000

[Summary]

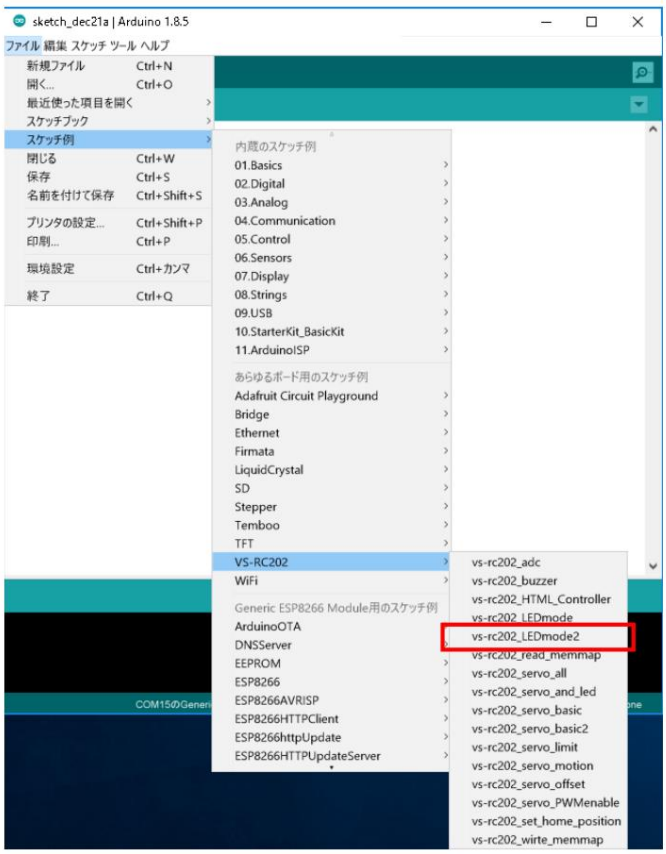
1. Pins can be changed to LED mode with setLedMode()
2. Use setLedBrightness() to set brightness
3. LED brightness changes according to transition time

(D)Direct control of LED brightness

setLedBrightness() can change the brightness of the LED in the same way as a servo motor, but since the brightness changes at the timing of moveServo(), it may be difficult to use it independently from the servo motor.

Therefore, when using a servo motor and LED together, it is convenient to use the function setLedBrightnessDirect(). It's profit.

Select File > Sketch Example > VS-RC202 > vs-rc202_LEDmode2 from the Arduino IDE menu. vinegar. Once you open the sketch, connect the VS-RC202 and your PC using the USB microB cable, and write the sketch.



After writing the sketch, connect the servo motors to SV1, 2, 3, and 4, and connect the LED to SV10. and connect the battery box to the DC jack of VS-RC202.

Were you able to confirm that the servo motor is running while the LED is lit?

Using the setLedBrightnessDirect() function, you can instantly change the LED brightness and it will not be affected by motion playback.

```
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);
}
```

[Function description]

Format: setLedBrightnessDirect(pin number, brightness); //Brightness setting range: 0 to 1000

Example 1: setLedBrightnessDirect(10, 1000); //Set the brightness of LED10 to 1000 immediately

If the brightness is set to 1 or higher, it will not be affected by moveServo(). Brightness is 0

In this case, the brightness changes with moveServo()

setLedBrightnessDirect(10, 100); //Playing the motion will not affect it

setLedBrightnessDircet(10, 0); //Playing the motion will change the brightness

[Summary]

1. Use setLedBrightnessDirect() if you want to change the LED brightness immediately.

2. Using setLedBrightnessDirect() is not affected by motion playback

8. Buzzer mode

(A) Buzzer mechanism

VS-RC202 is equipped with a piezoelectric buzzer. A piezoelectric buzzer contains a metal plate and has a certain

If you give a PWM signal at a frequency, the metal plate will shake at that frequency and produce sound. Therefore, the PWM signal

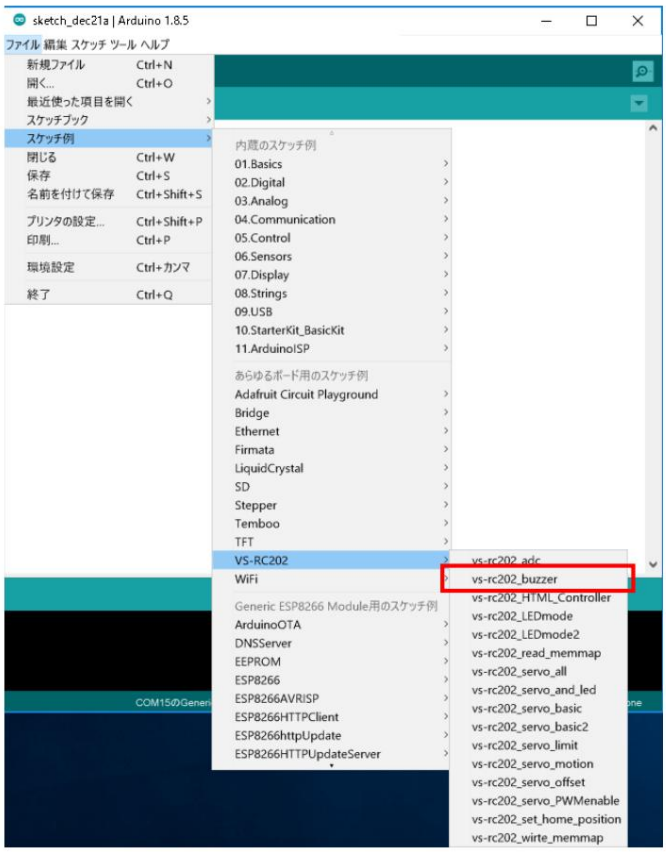
The frequency becomes the frequency of the sound.

(B) How to use the buzzer

Select File > Sketch Examples > VS-RC202 > vs-rc202_buzzer from the Arduino IDE menu.

After opening the sketch, connect the VS-RC202 and the PC with a USB microB cable and write the sketch.

Please.



Once you've finished writing your sketch, you'll see the twinkle, twinkle, star begin to chime.

Let's take a look at the contents of the sketch. Basically, run buzzerEnable() only once,
Just set the sound you want to play with setBuzzer(). Please note that if you enable the buzzer, SV9,
10 will no longer be available.

```
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);
                        :
```

[Function description]

Format: buzzerEnable(Enable(1)/Disable(0)); //Enable/disable the buzzer Format:
setBuzzer(pitch, length, breath); //Example of making a sound 1:

```
buzzerEnable(1) ;                       //Buzzer enabled (SV9, 10 not available)
        buzzerEnable(0);                //Disable buzzer (SV9, 10 usable)

Example 2: setBuzzer(PC5, BEAT4, TANG); //Sounds a C5 quarter note with tonguing
           setBuzzer(PA5, BEAT2, SLUR); //Slur the A5 half note
```

setBuzzer() represents a musical note and can play a single note melody.
For a list of sounds, please refer to pages 32-35 of the attached VS-RC202 instruction manual.

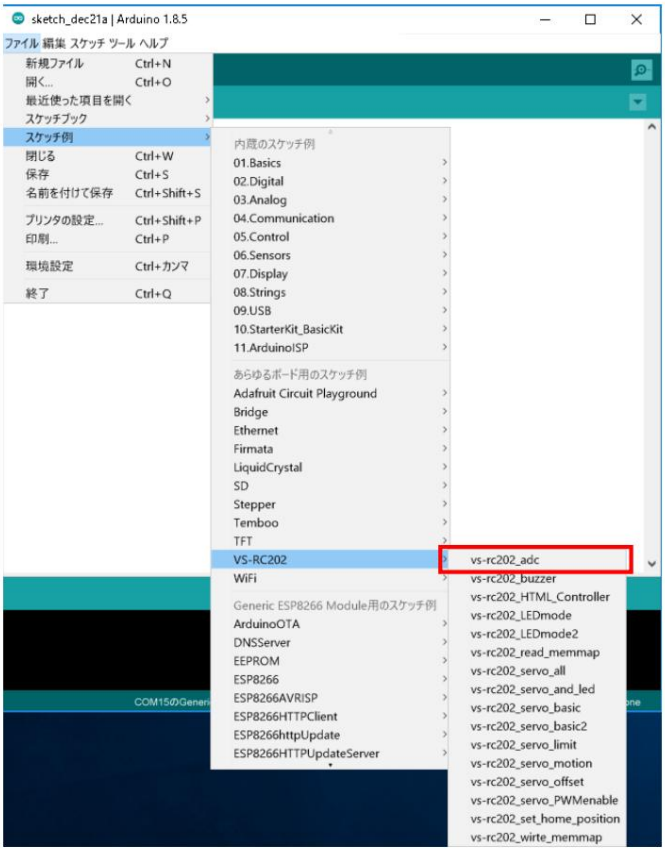
[Summary]

1. Use buzzerEnable() when using a buzzer
2. If buzzerEnable(1), SV9 and 10 cannot be used.
3. You can play a single note melody with setBuzzer()

9. How to use the sensor

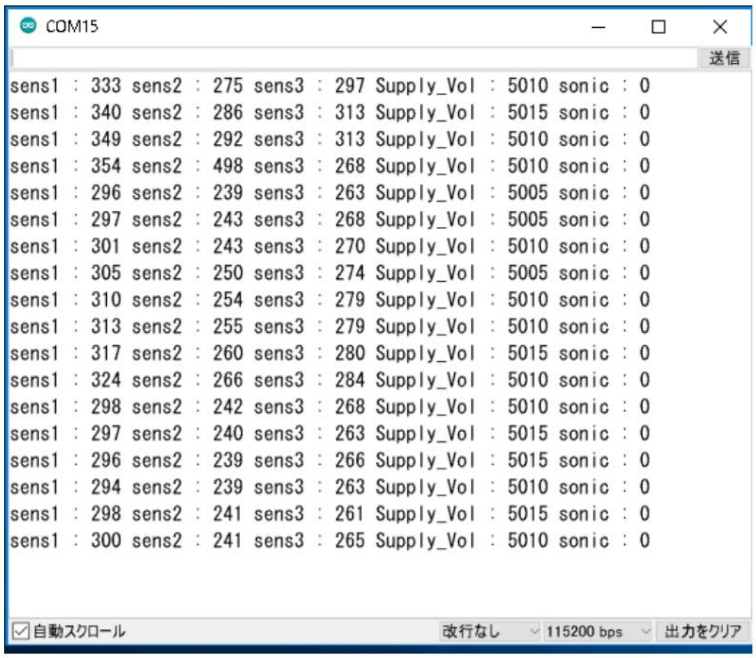
This section explains how to use the sensor. VS-RC202 has 3 analog sensors and 1 ultrasonic sensor
A wave sensor can be connected. Additionally, it is also possible to read the power supply voltage.

First, let's run the sample. Arduino IDE menu File>Sketch example>VS-
Select RC202>vs-rc202_adc. After opening the sketch, connect the VS-RC202 and PC to the USB microB cable.
Please connect with the cable and write the sketch.



After writing the sketch, display the Arduino serial monitor. The baud rate is
Select 115200.

If you see a message like the one below, it's working properly. Values can be obtained without connecting the sensor. (In this case, the value is not connected to anything, so the voltage is unstable and a halfway value is returned.)



Let's take a look at each function and the content it reads.

(A) Reading analog sensor values

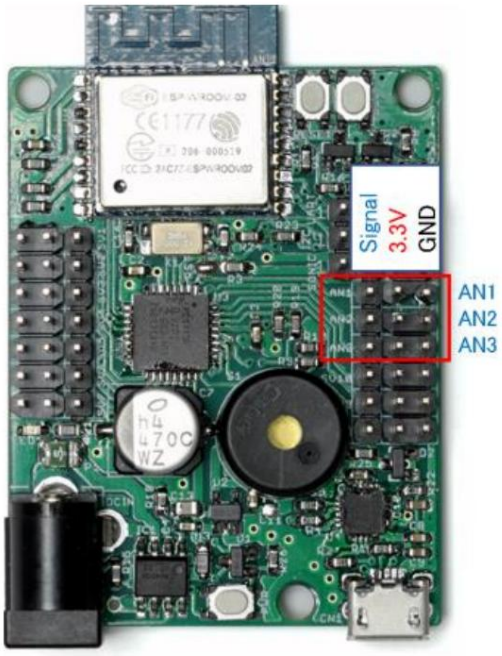
Enable sensor reading with setSensEnable() function is enabled by default.

vinegar. Read the voltage of the sensor connected to AN1-3.

can be included. Also readPow on battery

- You can read the voltage.

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

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



[Function description]

Format: readSensEnable(Enable(1)/Disable(0)); //Default is Enable Format:

readSens(pin number); Format: //Read sensor value

readPow(); //Read power supply voltage

Example 1: readSensEnable(0); //Disable sensor reading

Example 2: readSens(1); //load sensor 1

readSens(2); //load sensor 2

Example 3: readPow(); //Return value is mV

(B) Enable pull-up resistor

You can enable internal pull-up resistors on pins AN1-3. Connect the pull-down switch

Used when

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

[Function description]

Format: setPullupEnable(pin number, Enable(1)/Disable(0));

Example 1: setPullupEnable(1,1); //Enable internal pull-up of sensor 1

setPullupEnable(1,0); //Disable internal pull-up of sensor 1

(C) Reading the value of the ultrasonic sensor

VS-RC202 can connect ultrasonic sensor HC-SR04. The pin arrangement is as follows.


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

[Function description]

Format: getDist(); //Return value is float type distance (mm)

[Summary]

1. Obtain analog sensor values using readSens()
2. Use setPullupEnable() to enable the internal pull-up resistor
3. Obtain the ultrasonic sensor value using getDist()

10. Operate from your smartphone

Now that you have learned how to use the library, you can connect it to Wi-Fi and use your smartphone's browser.

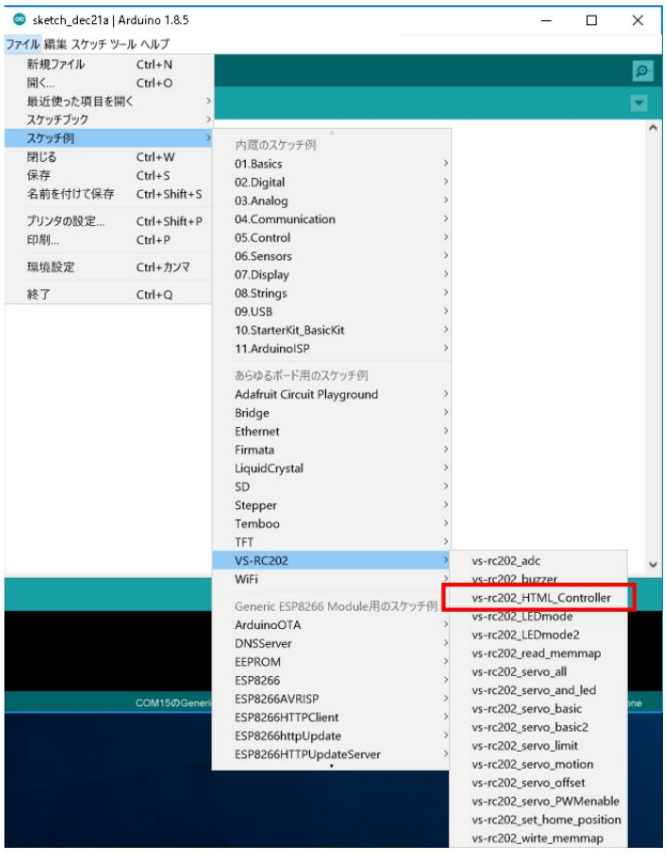
I will try to communicate with you.

(A) Connect to the router and run it

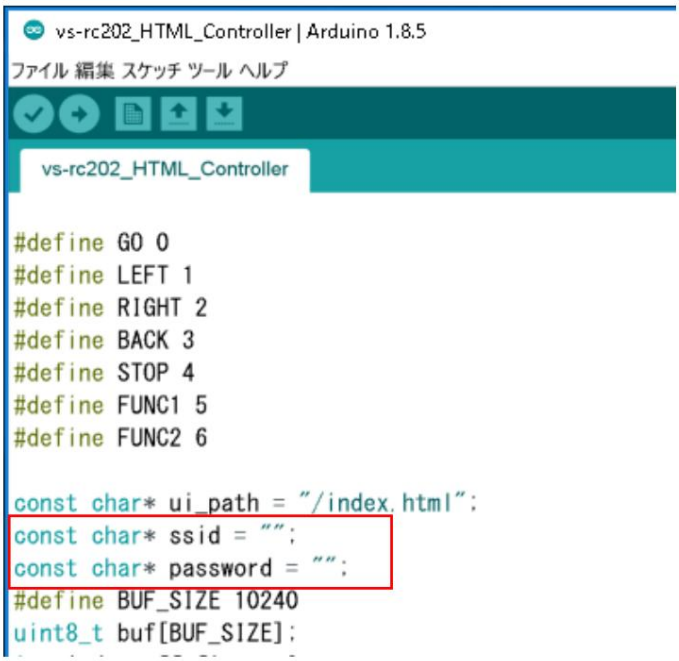
Select File > Sketch Examples > VS-RC202 > vs-rc202_HTML_Controller from the Arduino IDE menu.

Choose. vs-rc202_HTML_Controller is a learning platform for VS-RC202, Piccorobo IoT.

Although this is a sketch, it is possible to move the board alone.

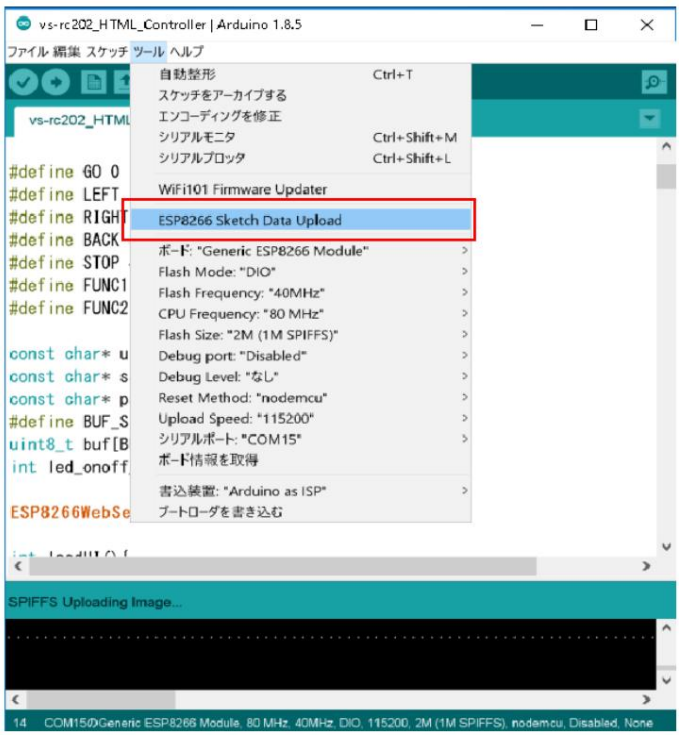


After opening the sketch, set the SSID and password of the Wi-Fi router you want to connect to on lines 17 and 18.



After setting the SSID and password, connect the VS-RC202 and PC with a USB microB cable and sketch. Please write it down.

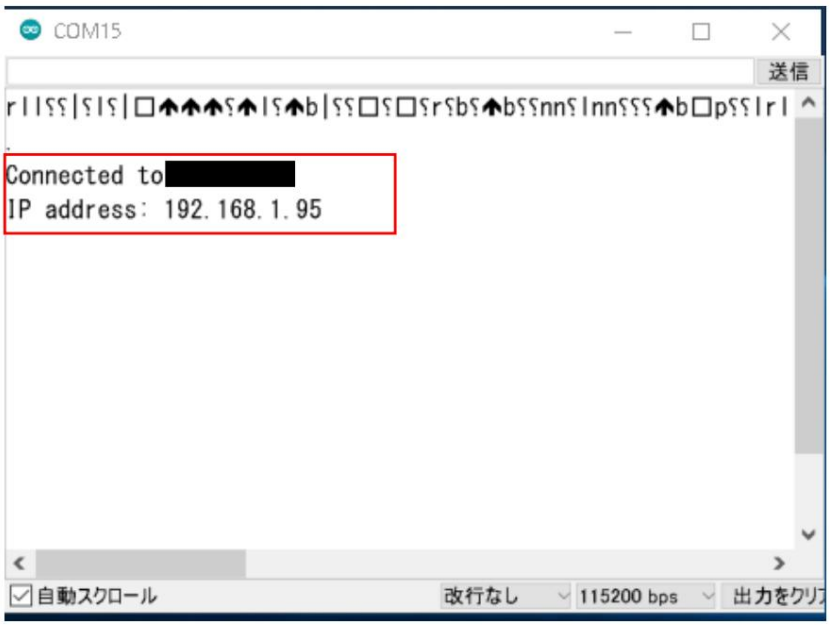
Next, select ESP8266 Sketch Data Upload from the Tools menu and upload the HTML. If you want to see the HTML content, select Sketch > Show Sketch Folder from the menu. You will find index.html inside the data folder. The maximum upload size is 1MB.



Once you have finished sketching and writing the HTML, display the Arduino serial monitor. Select 115200 for Baud rate. Then press the reset button on VS-RC202. Enter the SSID and password correctly.

If the password is set, the following display will appear.

If an error occurs, make sure you have not forgotten to upload the HTML file, or have entered the SSID and password correctly. Check to see if there is a difference.

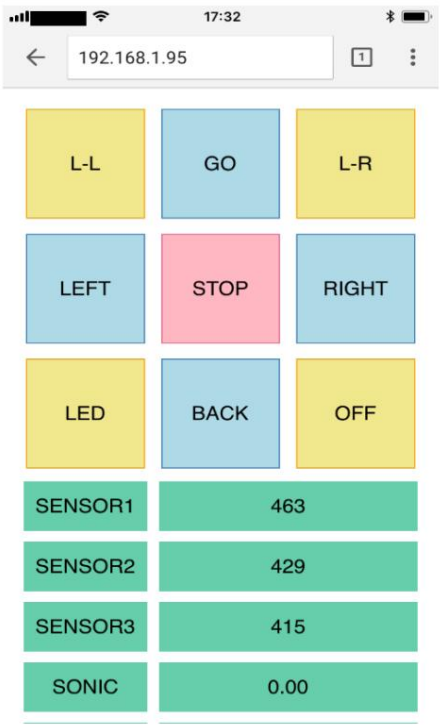


Connect servo motors to SV1, 2, 3, and 4, and connect SV9, Connect the LED to 10 and connect the battery box to the DC jack of VS-RC202.

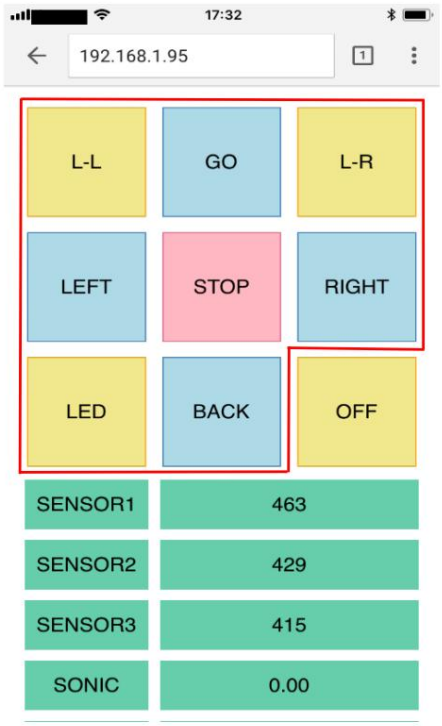
Connect your PC or smartphone to Wi-Fi in the same network as VS-RC202.

In this state, connect to the IP address of VS-RC202 (http://192.168.1.95 in the above example) using a browser. Let's. If a screen

like the one on the right is displayed, communication is successful. it's finished.



(B) Accept commands from the browser



First, let's see how it works. Press the blue button displayed on the browser or LL, LR to move the servo motor. Press the button labeled LED to turn the LED on or off.

Pressing OFF turns off the power to the board, so now Please do not press.

Next, let's take a look at how it is implemented within the sketch. It's a little long, but I understand everything is not necesary to. From the minimum necessary part Let's get started.

server.on() is written from line 262 of the sketch. This associates which function should be executed when a GET request comes from the browser to which URL.

For example, if there is a request to http://(VS-RC202 IP address)/go/, the Go() function will be executed.

Wow vinegar. In index.html, when each button is pressed, a GET request is made to the corresponding URL using javascript. am sending.

[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()>LL</button> <button
class="ten" onClick=sendGo()>GO</button> <button class="func"
onClick=sendF2()>LR </button>
                                        :
```

There is a Go() function on line 125 of the sketch. As a process, setMotionNumber() is executed. vinegar. This sets the number of the next motion to be executed in the global variable motion_number.

vinegar.

Based on this motion_number, the mode to be executed is in selectMotion() on line 91. is.

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

Serial.println(“Go”); is displaying debug messages on the serial monitor, and server.send(200, "text/html","Go") is returning "200 OK + Go" to the browser.

From the above, the basic flow of accepting commands from the browser is as follows.
ÿ Create the function you want to execute ÿ Associate the URL and function with server.on() ÿ If you want to execute a motion, set motion_number in the function and use selectMotion()

(C) Send data to browser

Next, if you want to send sensor data etc. to the browser, the method is basically the same as receiving commands from the browser. Sens() is related to http://(IP address of VS-RC202)/sens/ in server.on() It is attached.

The Sens() function on line 187 of the sketch reads the sensor data, converts it to a string, and It is sent back as the third argument of server.send().

```
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);
}
```

On the browser side, on line 126 of index.html, http://(VS-RC202's IP address is sent every 1 second.
I am sending a request to /sens/. Also, the sensor data received in the processing on lines 101 to 119 is
Inserting it into HTML.

From the above, the basic flow of sending data to the browser is as follows.
ÿ Create a function to read sensor data, etc. ÿ
Associate the URL and function with server.on()
ÿ Send string data as the third argument of serve.send() ÿ
Send and receive requests periodically on the browser side Parse and display string

[Summary] 1. Set SSID and password and connect to Wi-Fi router
2. Use server.on() to associate the URL with the function
you want to execute. 3. Play motion easily using
selectMotion(). 4. Send data to the browser using serve.send().

11. Manipulate memory map directly

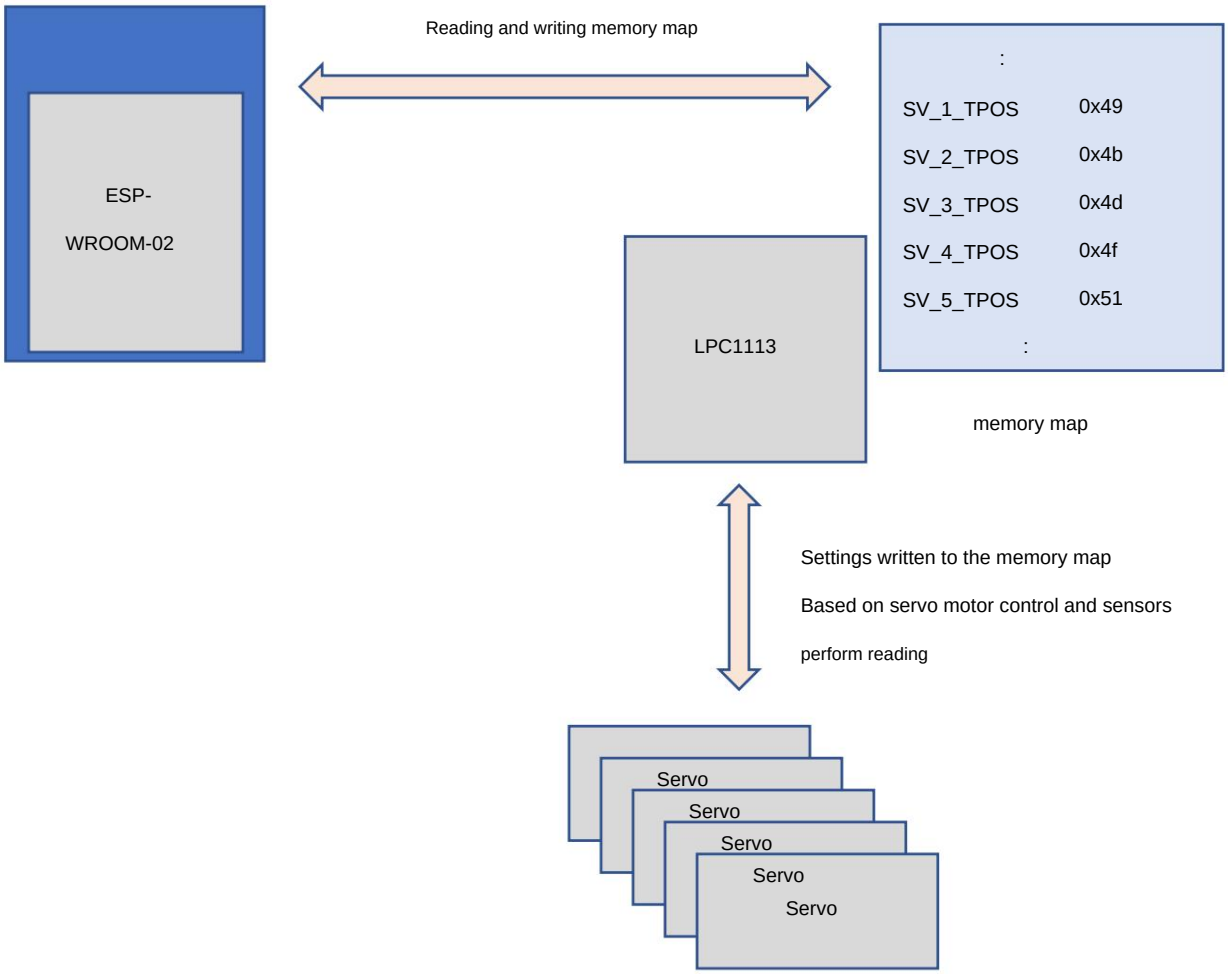
(A) What is a memory map?

The LPC1113 installed in the VS-RC202 has a virtual register memory called a memory map.
Masu. Control servo motors and sensors by reading and writing values in this memory map.

The sample code introduced so far actually writes values to the memory map of the function you want to call.
I'm here. If you learn how to handle the memory map directly, you will be able to create your own VS-RC202 functions.
Masu.

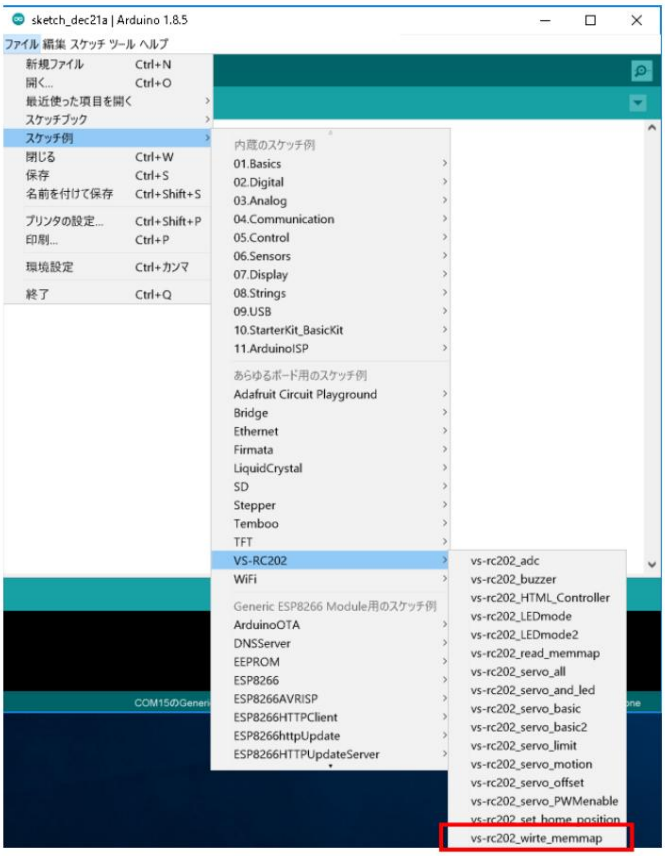
Details of the memory map are listed at the end of the separate VS-RC202 instruction manual.

https://www.vstone.co.jp/products/vs_rc202/download.html



(B) Writing memory map

If you want to operate a servo motor, write the command to the memory map. sample code Let's take a look. Select File > Sketch Examples > VS-RC202 > vs-rc202_write_memmap from the Arduino IDE menu. After opening the sketch, connect the VS-RC202 and PC to the USB microB cable. Please connect with the file and write the sketch.



After writing the sketch, connect the servo motor to SV1, SV2, SV3, and SV4 and connect the battery. Connect the box to the DC jack of VS-RC202 and press the power switch. Sketch written successfully If it is, the servo motor will continue to move.

Let's take a closer look at the contents of the sketch. moveServo() is called inside loop(). this is a function that sends a write command to LPC1113 via I2C. Arduino's standard Wire function is used for I2C communication. I am.

There is a memory map list on page 39 of the instruction manual for VS-RC202, but the transition from addresses 0x47 to 0x5b is This is the moving time and target position address.

In I2C, write communication is normally performed using the following steps.

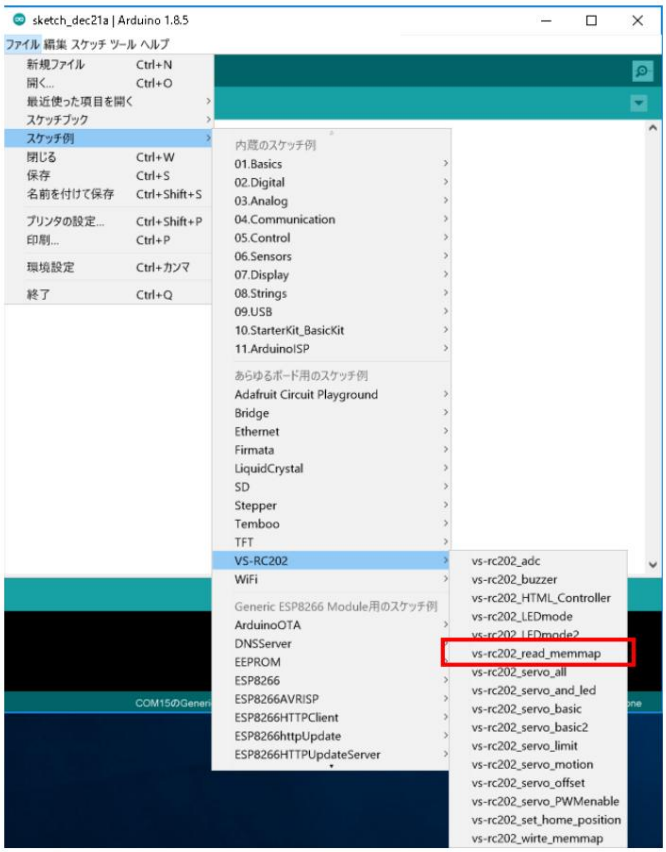
ÿ Specify the device you want to communicate with using the device address ÿ Specify the address of the device's memory you want to write to ÿ When you send the data to be written, it will be written in order starting from the address specified in ÿ ÿ Send a completion notification

In the sample code example, specify the address of LPC1113 and select SV_MV_TIME(0x47) in the memory map. The transition time and target position are written in order from start to finish, and 0x01 is written to SV_START at the end.

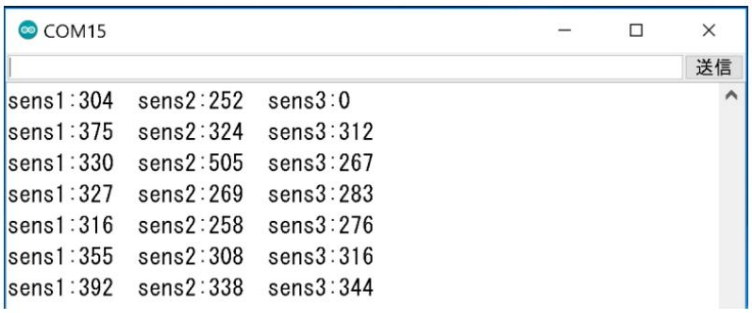
```
void moveServo(int motion[SV_NUM+1]){
    unsigned char byte_motion[2*(SV_NUM+1)];
    int i;

    //Since I2C can only send one byte of data at a time, //the motion array
    data is divided into upper 8 bits and lower 8 bits //stored in the array in little endian fashion.

    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); //Device address of LPC1113
    Wire.write(SV_MV_TIME);                    //Specify the starting address you want to write to
    for(i=0;i<2*(SV_NUM+1);i++){               //Write transition time and target position
        Wire.write(byte_motion[i]);
    }
    Wire.write(1);                             //Write 1 to SV_START
    Wire.endTransmission();                    //Communication ends
}
```

(C) Reading memory map

Next, we will explain the procedure for reading memory map values. This is the place to check sensor data and settings. Used when appropriate. Let's take a look at the sample code. Select File > Sketch Example > VS-RC202 > vs-rc202_read_memmap from the Arduino IDE menu. After opening the sketch, connect the VS-RC202 and PC with a USBmicroB cable and write the sketch.



After writing the sketch, display the Arduino serial monitor. Baud rate is 115200 Choose. If the sketch has been written successfully, the following message will be displayed.



Let's take a closer look at the contents of the sketch. readSens() is called inside loop(). This is a function that sends a read command to LPC1113 via I2C. Arduino's standard Wire function is used for I2C communication.

Addresses 0x01 to 0x06 (AN1 to AN3 sensor data) of the memory map on page 39 of the VS-RC202 instruction manual are specified and read.

In I2C, read communication is normally performed using the following steps.

ÿ Specify the device you want to communicate with using the device address ÿ Specify the address of the device's memory you want to read ÿ End communication once

ÿ Send a read request for the number of bytes you want to read from the address specified in ÿ

In the sample code example, specify the address of LPC1113 and write SENS_1 (0x04) in the memory map. 6 bytes of data are read from it.

```
void readSens(){

    Wire.beginTransmission(DEV_ADDR); //Specify the device address
    Wire.write(SENS_1);                 d//Specify the first address to read
    Wire.endTransmission();             //End communication once

    //Read 2byte
    unsigned char tmp[6];
    int index = 0;
    Wire.requestFrom(DEV_ADDR, 6); //Send a request to read the required number of bytes
    while (Wire.available()) {
        tmp[index++]=Wire.read();
    }

    //Convert byte to short short          //Convert the received data to a format that can be displayed
    sens1 = tmp[1] << 8 | tmp[0]; short sens2 =
    tmp[3] << 8 | tmp[2]; short sens3 = tmp[5] <<
    8 | tmp [Four];
}
```

[Summary]

1. All functions are realized by reading and writing memory maps.

2. Use I2C to access the LPC1113 memory map. 3. For writing,

specify the start address of the memory map and then write data. 4. For writing, use to

move the servo motor or write settings. 5. For reading, specify the start address of the

memory map, finish communication once, and then issue a read request. 6. Read is used to read sensor data and

settings.