VS-RC202instruction manual

Content

1. Introduction

This is the instruction manual to explains how to use and specifications of "VS-RC202" that is wireless robot controller board with ESP-WROOM-02. Before using the board, read this manual carefully and use safely.

2. Specifications

size	W40× D52 (mm) The size of the PCB part not including the antenna
weight	16.6g
CPU	ESP-WROOM-02
Power supply	DC 4.5 to 8.0 V NiMH 4
output	Servo motor (or LED) ×10 The piezoelectric buzzer × 1
input	Analog sensor input × 3 Ultrasonic sensor input × 1
interface	USB microB × 1 Serial port (3.3v level) × 1 I2CPort (3.3v level) × 1

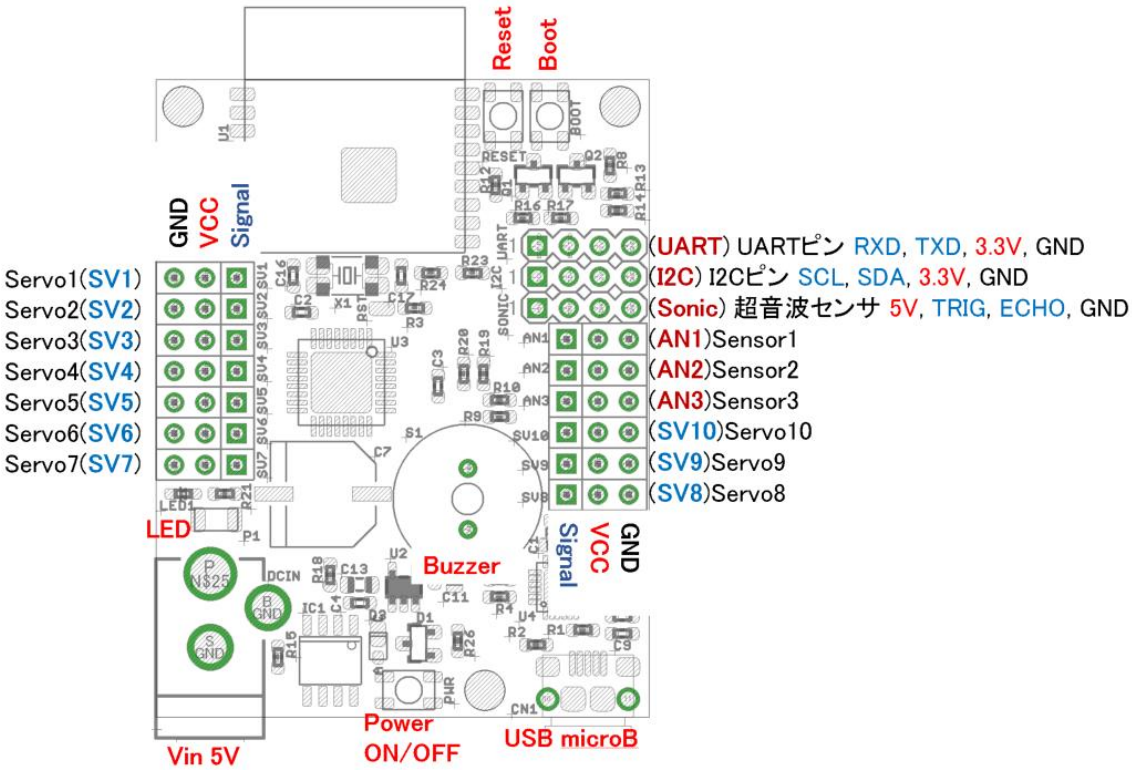
3. Notice

When using the board please note the following lists.

- Do NOT give a strong shock to the VS-RC202 (hereinafter referred to the board).
- Do NOT wet the board and use in a humid or dusty place.
  There is a risk of short circuit.
- If the smoke from the board has occurred, please turn off the power immediately.
- Please Do NOT use or store the board within reach of little children.
- When using the board, the components may become hot Do NOT touch them.
  Do NOT touch the metal part of the board there is a risk of short circuit with static electricity.
  Please hold the edge of the board.
- When the pin headers are short by metal parts, there is a risk of failure due to excessive current.

4. Circuit appearance

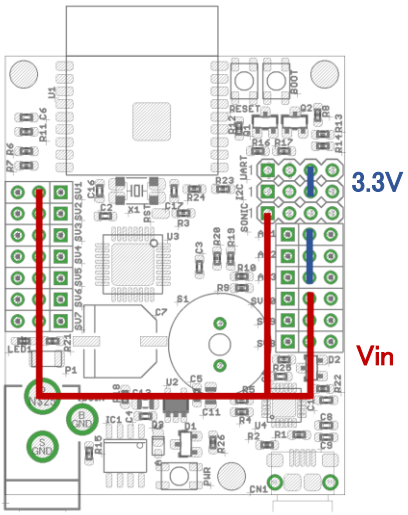
5. I/O



[Vin]

Power input range(Vin) is between 4.5V and 8V. Recommended voltage is between 4.8 and 5V. The power-supply voltage of the analog sensor, I2C and UART is 3.3V. Vin line is connected to the VCC of the servomotor and the ultrasonic sensor directly.

If you use a servo motor and ultrasonic sensor, a power supply voltage should be 5V.



[Power ON / OFF]

When you press the power button for 3 seconds and release you can turn ON/OFF the power of the baord. But if there is a USB power supply it will automatically power ON.

By default setting, if the power supply voltage is lower than 4.6V, the power will turn off automatically for battery protection. But there is the USB power supply it will not.

[LED]

LED lights up when power is ON.

[USB microB]

USBmicroB port is connected to the UART port of ESP-WROOM-02. It is for programming with Arduino IDE.

[Reset· Boot button]

When you press the Reset button, the ESP-WROOM-02 will restart. When you press the Reset button while pressing Boot button, ESP-WROOM-02 will start in boot mode.

[Servo1-10 (SV1-10)]

Only connect Servo motor or LED to SV1-10. If you drive plural of LED please use FET because there is a risk of the failure cause of output high current from LPC1113. OctopusLED light brick (blue) which you can buy vstone robot shop is easy and convenient led module. Please note that if you enable the buzzer function SV9,10 can not be used.

[Sensor1-3 (AN1-3)]

These are ADC input port. Connect analog sensors.

[SONIC]

This is the input port of the ultrasonic sensor. You can connect the HC-SR04.

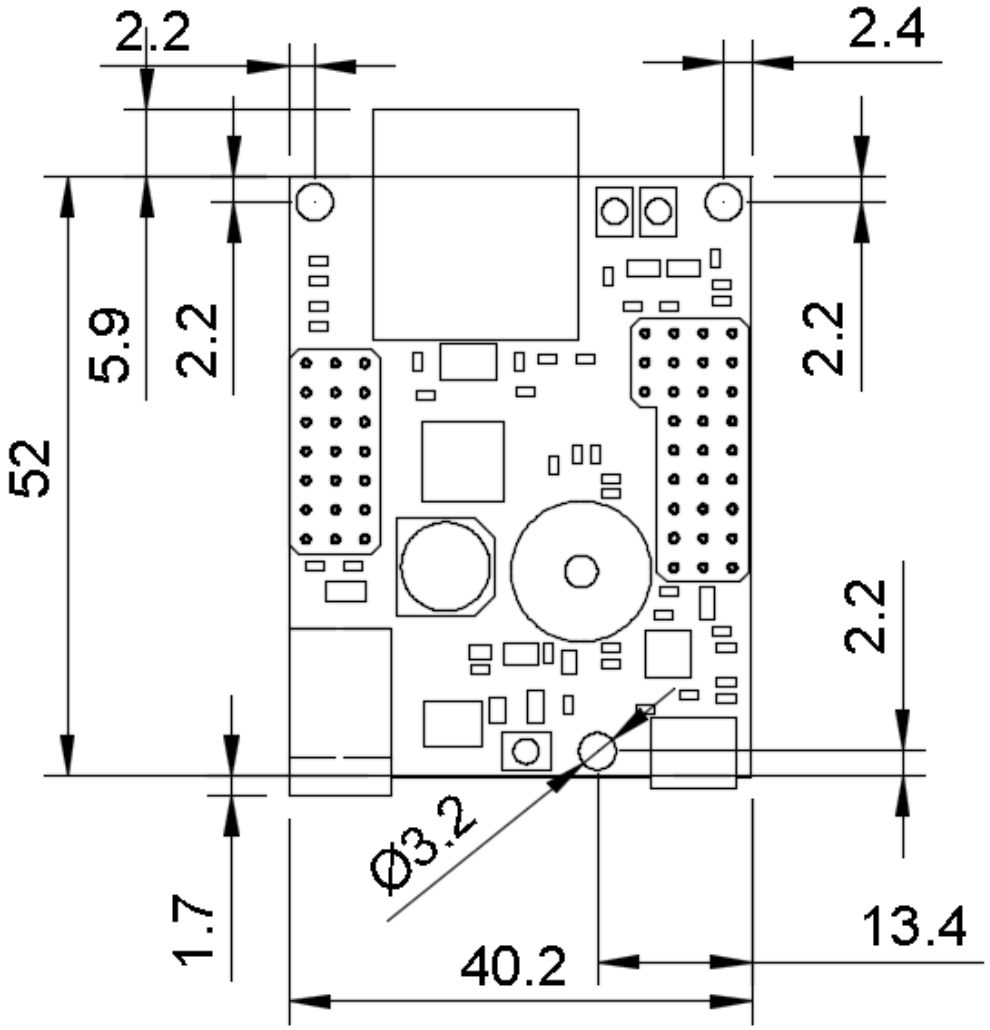
[I2C· UART]

I2C / UART bus port. Since the UART port also used to write the sketch of the ESP-WROOM-02, please do not connect anything when writing a sketch from the Arduino IDE.

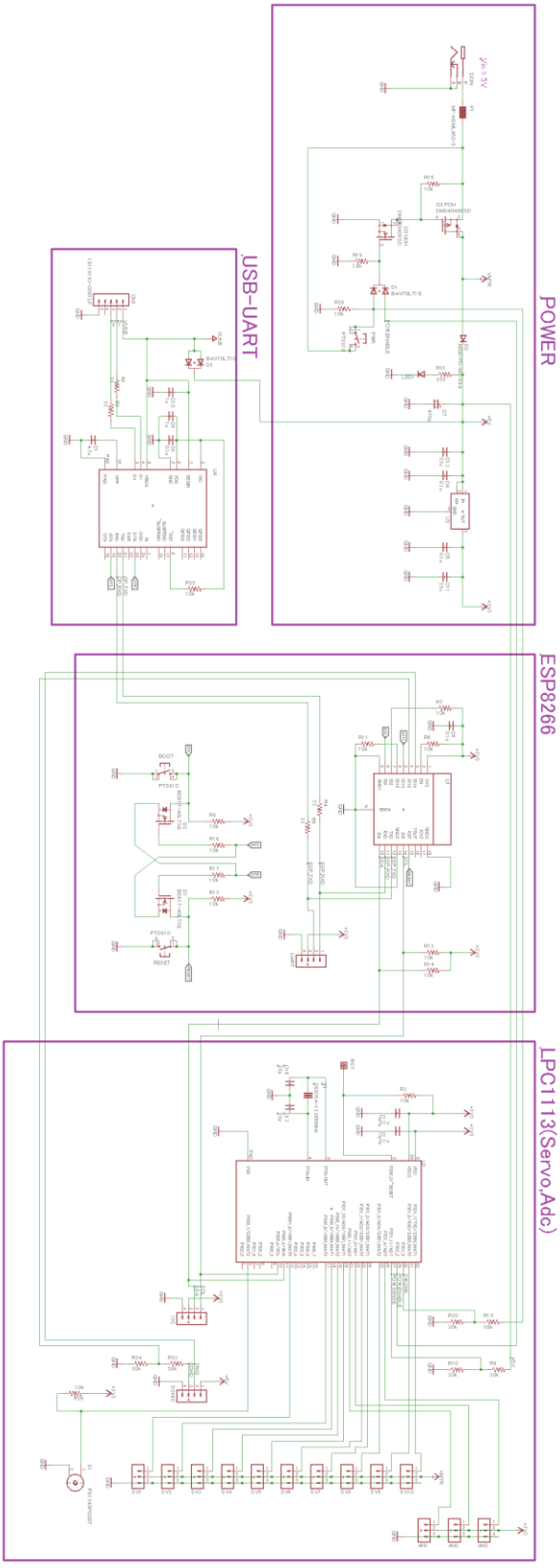
[Buzzer]

a piezoelectric buzzer.

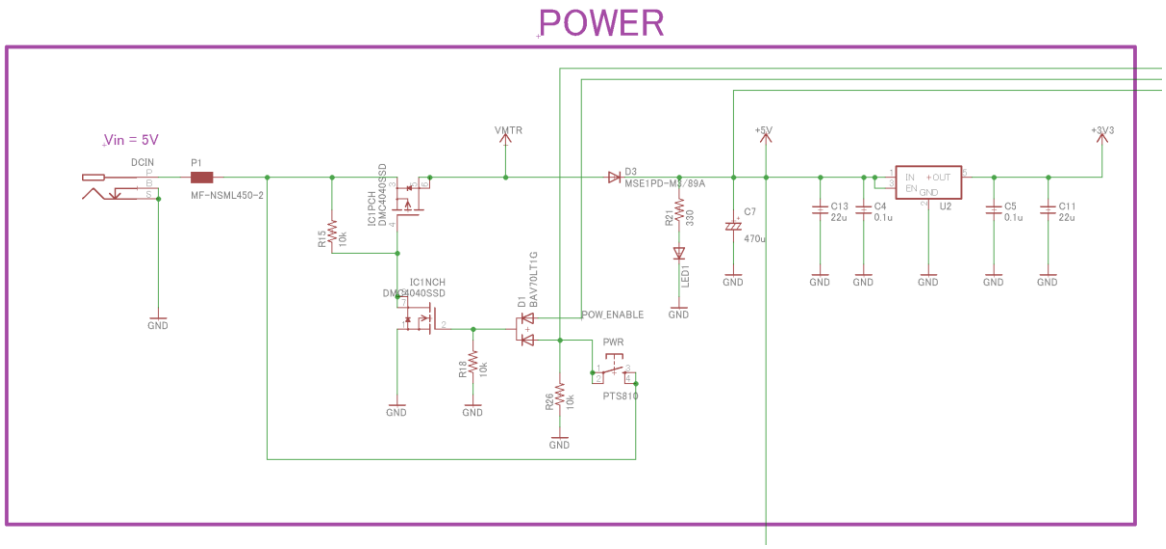
6. Board dimension

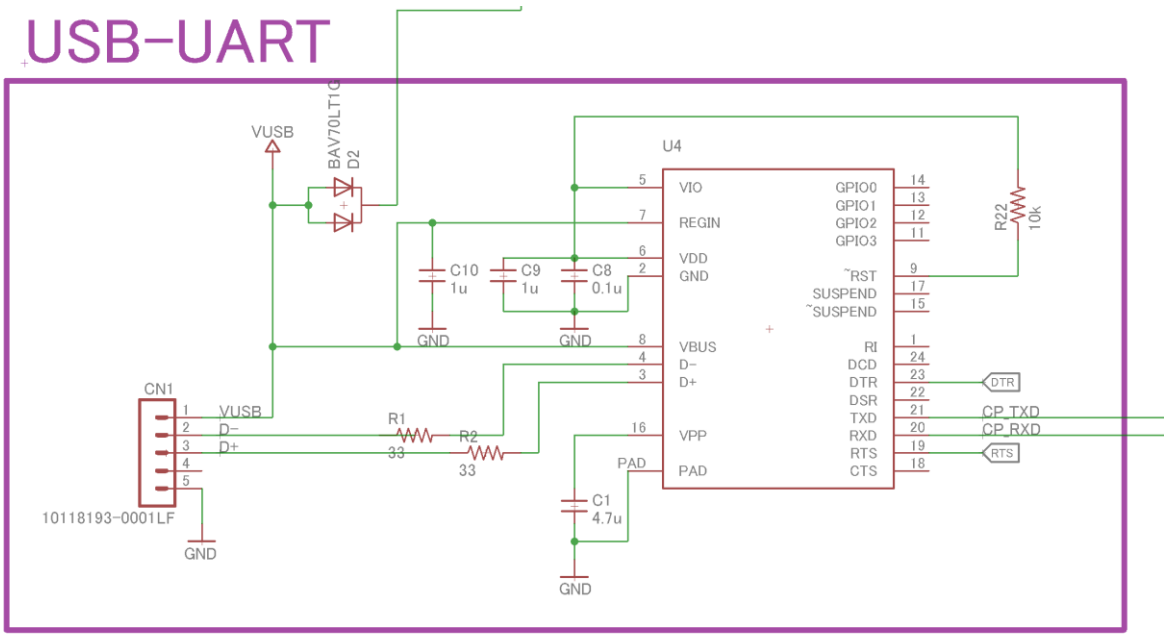


7. Circuit diagram

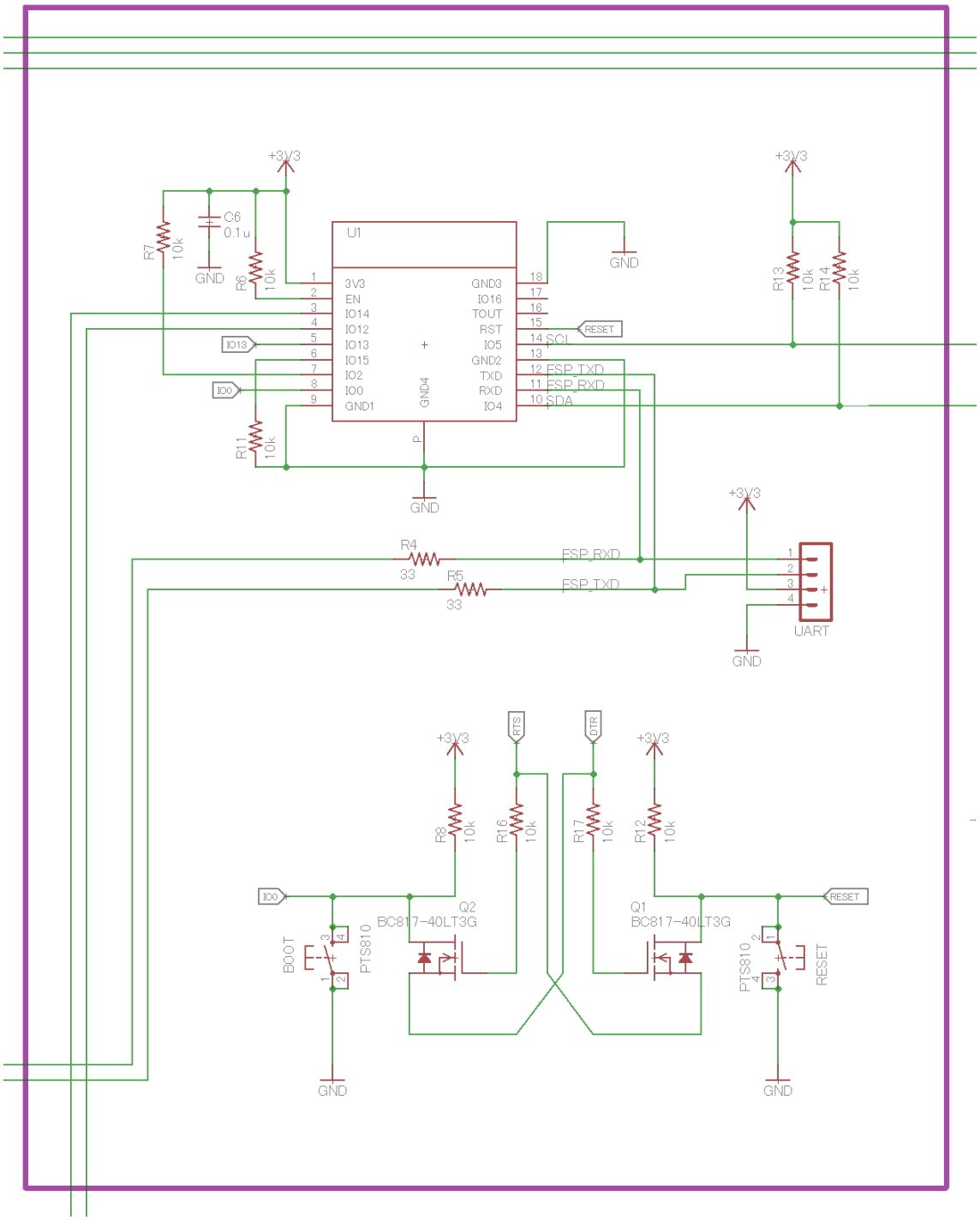


[Enlarged view]

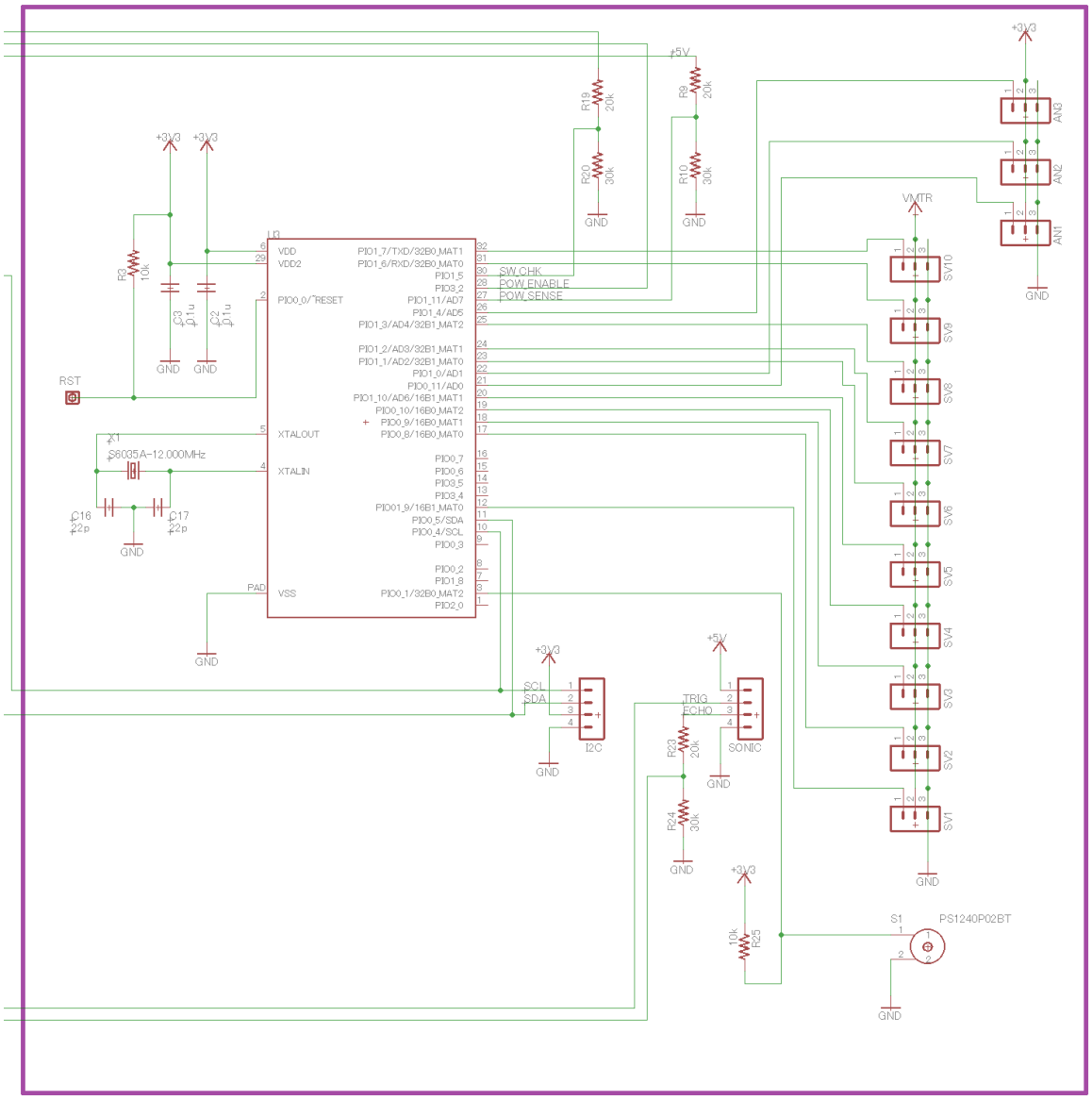




ESP8266


+3V3
C6
0.1 u
R7
10k
R6
10k
GND
U1
3V3
EN
IO14
IO12
IO13
IO15
IO2
IO0
GND1
GND4
GND3
IO16
TOUT
RST
IO5
GND2
TXD
RXD
IO4
RESET
SCL
ESP_TXD
ESP_RXD
SDA
IO13
IO0
R11
10k
P
R13
10k
R14
10k
R4
33
R5
33
ESP_RXD
ESP_TXD
UART
RTS
DTR
R8
10k
R16
10k
R17
10k
R12
10k
Q2
BC817-40LT3G
Q1
BC817-40LT3G
BOOT
PTS810
PTS810
RESET

LPC1113(Servo,Adc)



8. Software Overview of the VS-RC202

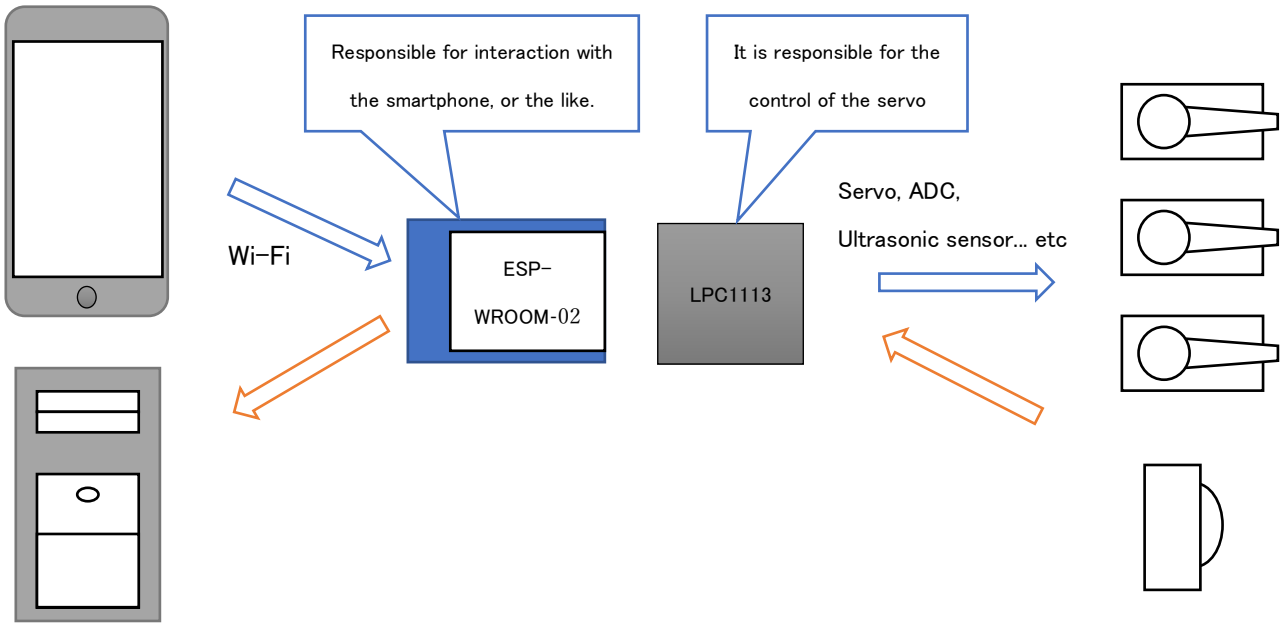
The concept of VS-RC202 is "Make it easy to develop a robot connected to the network ". which is Wi-Fi module “ESPWROOM-02” is mounted on the board and you can program it with the Arduino IDE.

ESP-WROOM-02 is not suitable for controlling motors and sensors directly. Therefore, the VSRC202 is equipped with ARM CPU(LPC1113), This CPU controls servo motors and sensors. In addition, it also has the function of interpolation processing of the servo motor.

You basically program either one of following method.

"Receiving the instruction from the smart phone via Wi-Fi, and transfer command to the LPC1113"

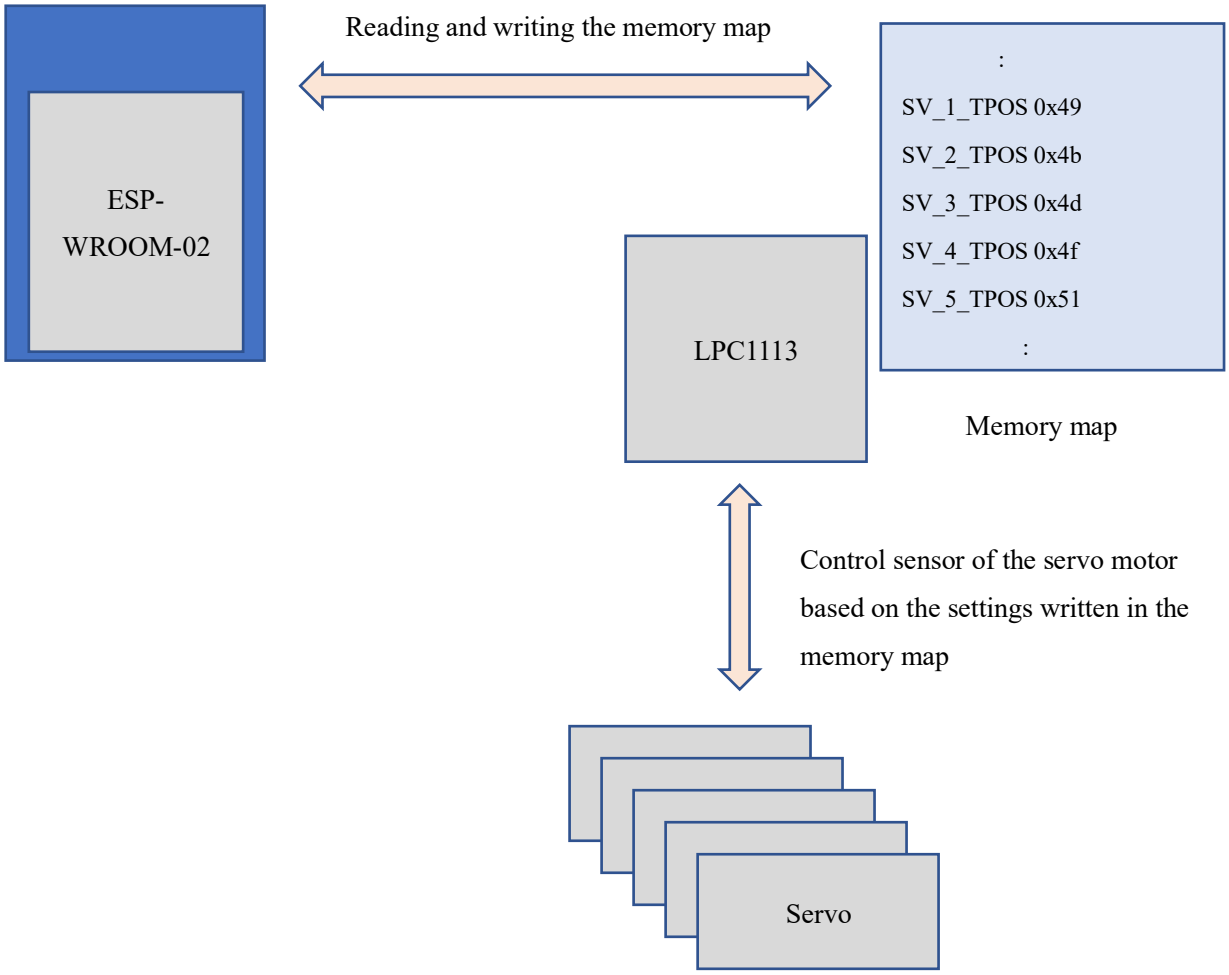
"Transfer some command to the LPC1113 and receive the result, and transfer the results to the cloud or web services "



9. How program works

LPC1113 that has been equipped on the VS-RC202 has a virtual register memory (memory map) in it. By reading and writing the value of the memory map, controls the servo motors and sensors.

You can make program which make ESP-WROOM-02 communicate with LPC1113 using memory map on the Arduino IDE.



10. Setup

[Set up software list]

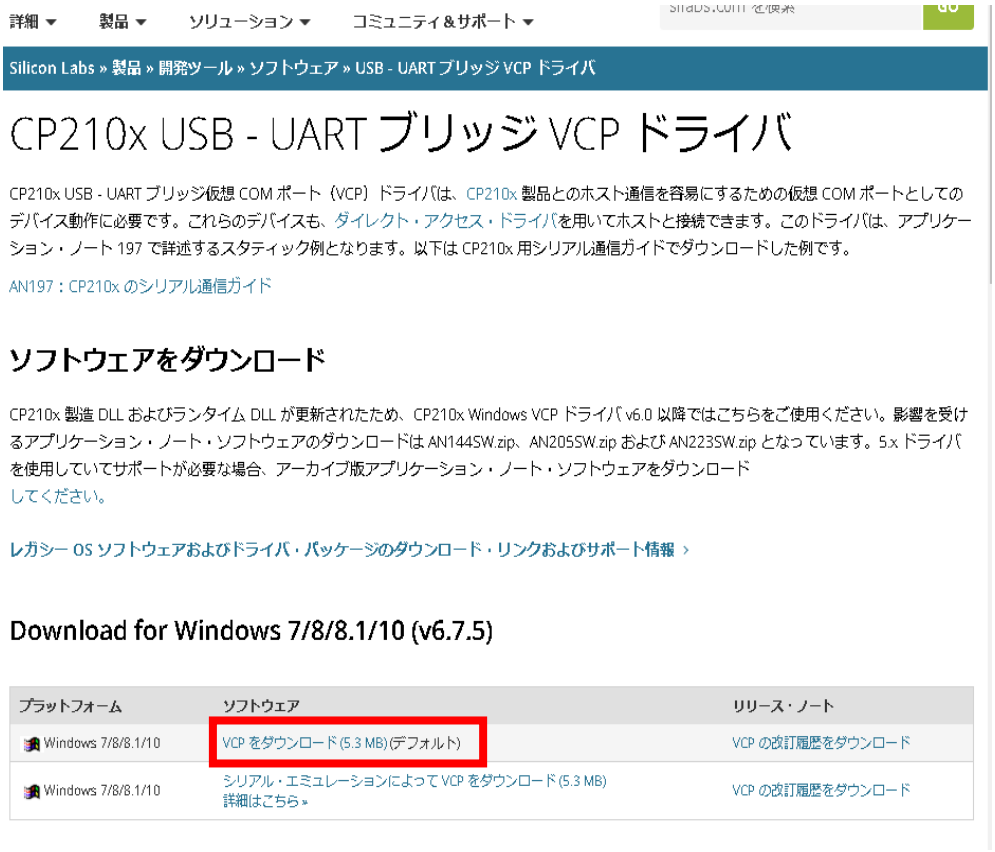
Describes how to set up the environment for VS-RC202 on the Windows10. This description is based on the information on 2018/1/26.

A) Install the driver of the USB-UART conversion chip

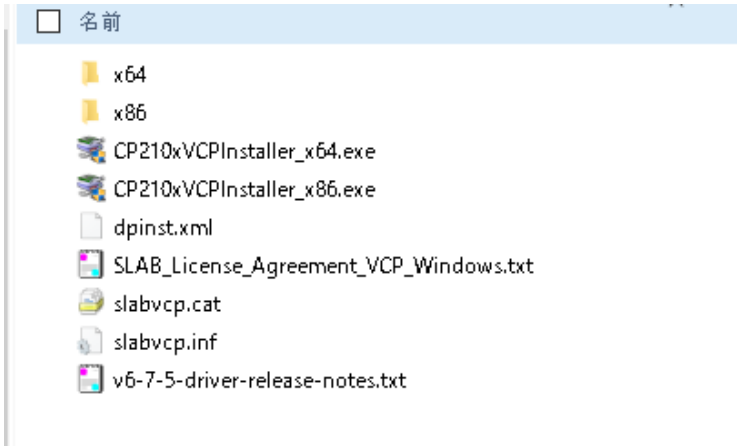
First download the USB-UART bridge driver for Windows in from the following URL, and install it.

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

[Note] Please do not connect the VS-RC202 to the PC before installing the driver.

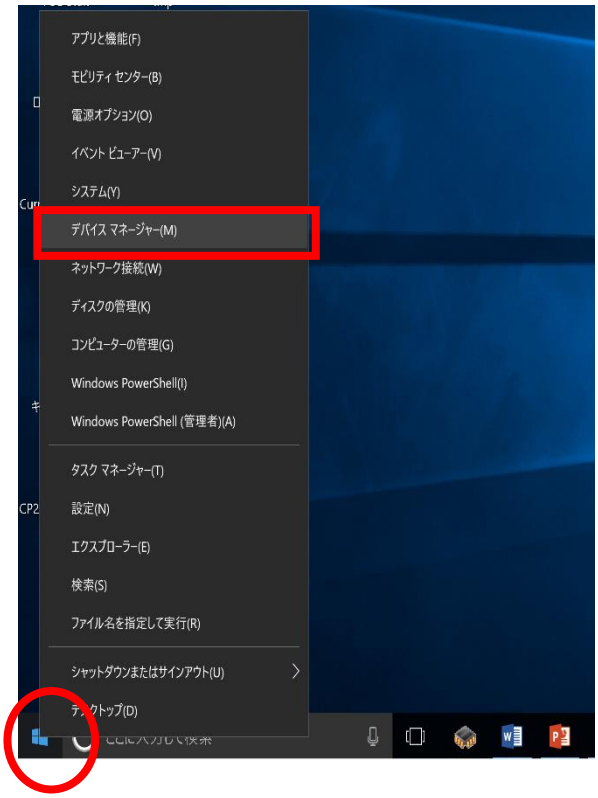


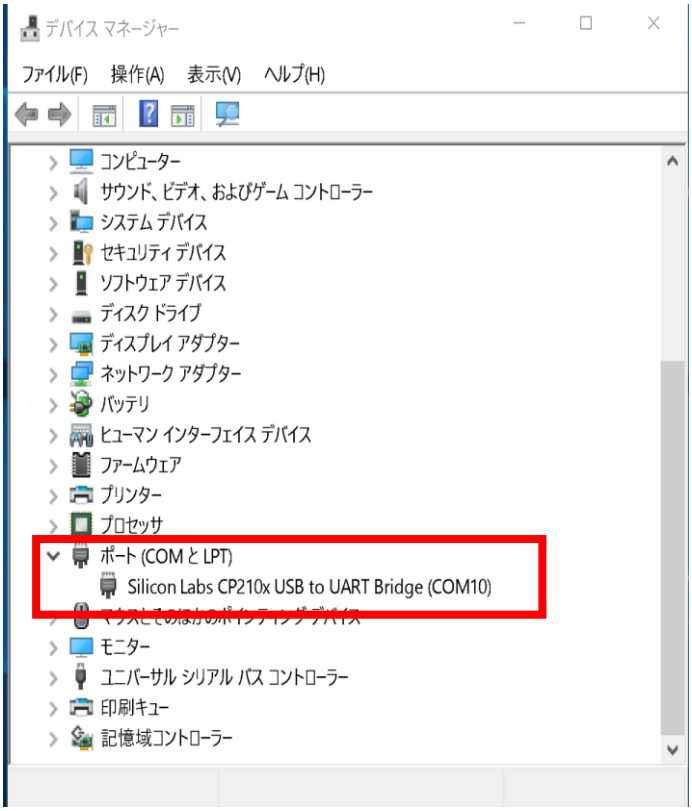
Unzip CP210x_Windows_Drivers.zipWhen and you can find the following files in the holder. If you use 64bit OS double-click **CP210xVCPInstaller_x64.exe and install it. If you use 32bit OS** double-click **CP210xVCPInstaller_x86.exe and install it**.



After installing the driver, connect the VS-RC202 to the PC with the USB cable and open device manager window. In the case of Windows10, right-click on the Start button and select the device manager.

If you can find Silicon Labs CP210x USB to UART Bridge (COMxx) in the (COM & LPT) on the device manager window, PC recognize the device correctly.





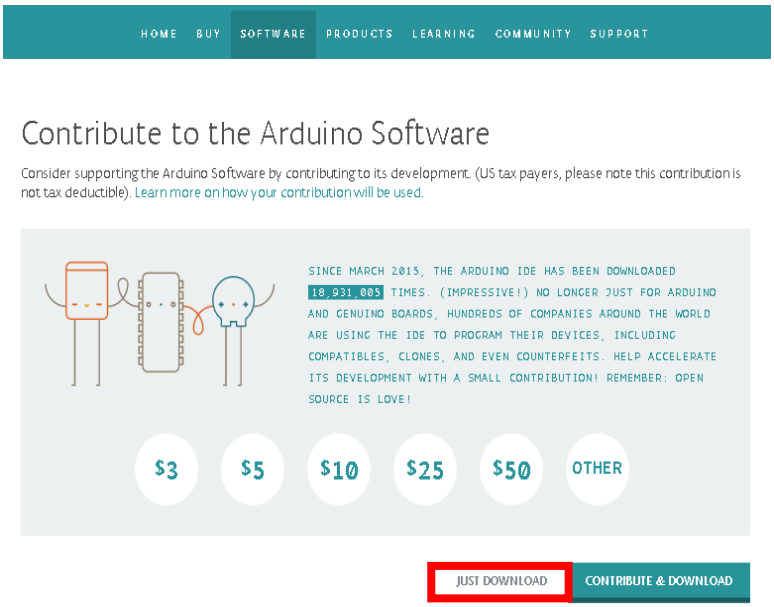
B) Download the Arduino IDE

You can program for VS-RC202 with the Arduino IDE. Open the following URL on your browser, and then select the Windows Installer of the latest Arduino IDE.

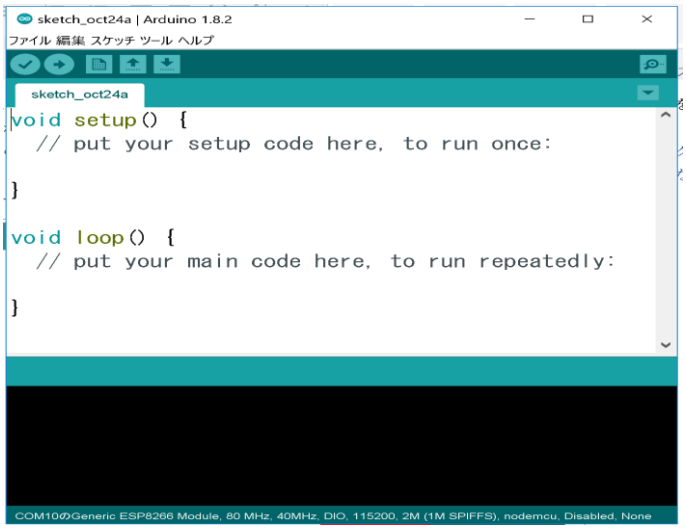
https://www.arduino.cc/en/Main/Software

After selecting the Windows Installer, the following page appears. Click the JUST DOWNLOAD, then download will begin.

(Notice) You can find something like prices, but these just mean that you can donate to the Arduino project . If you select just download you do not need to pay.



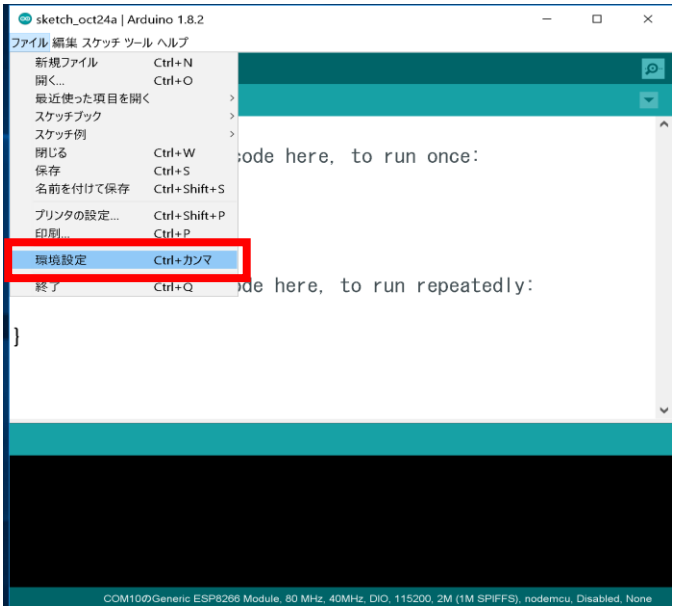
After downloading double-click arduino-xxxx-windows.exe and follow the installation instruction. Double-click the Arduino IDE icon on the desktop after installation, and then if you can see the following window, Arduino IDE has been installed successfully.



C) Enable to program VS-RC202 on the Arduino IDE

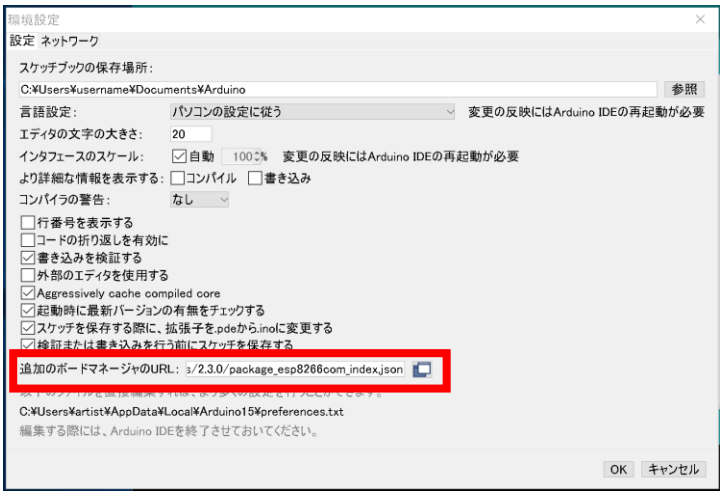
To make VS-RC202 enabled to be programed on the Arduino IDE, you need to install an additional configuration file.

Start the Arduino IDE, and choose Preferences from file on the menu bar.

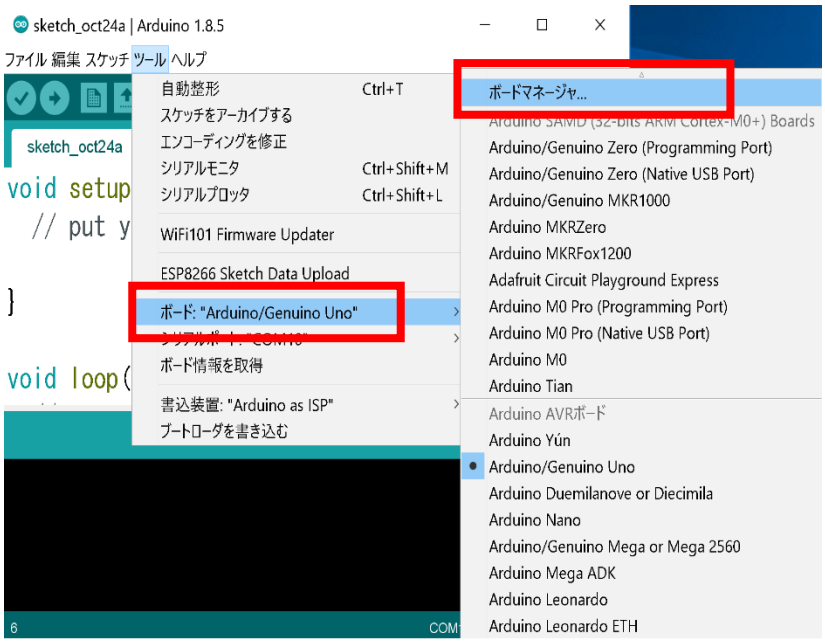


Copy and paste the following URL to the additional board manager URL on the Preferences panel, and press the OK button.

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

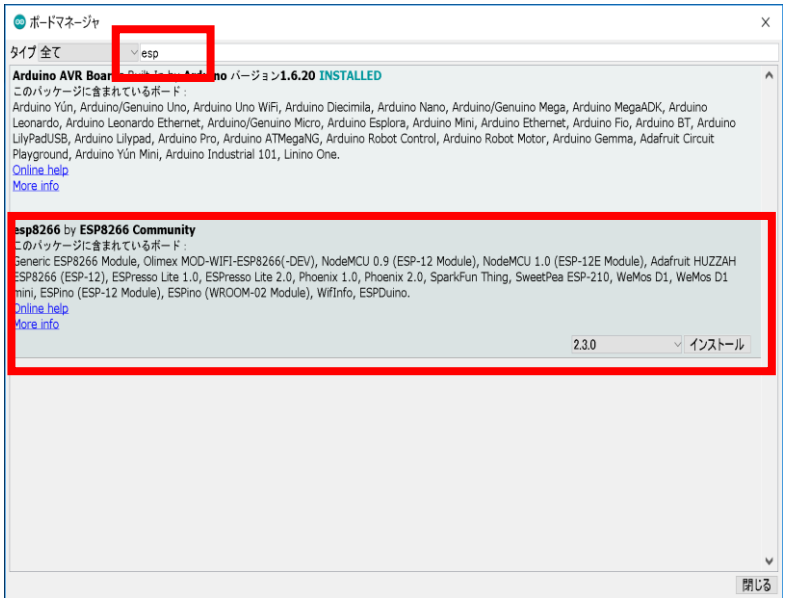


Select tools > board> board manager.

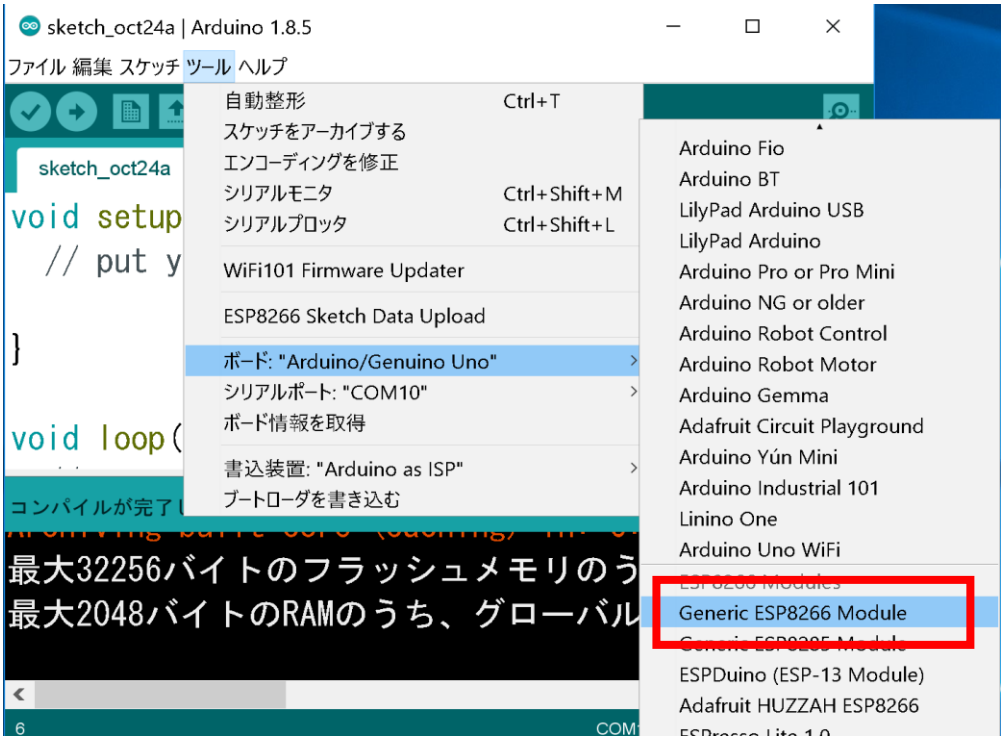


Enter "esp" in the search box on the board manager dialog. You may find esp8266 by ESP8266 Community, select the version and install it.

As of August 1, 2019, we have confirmed that it works with 2.5.2.



After installation of Esp8266, if you find tools>board> Generic ESP8266 Module, it has been installed successfully. (Scrolling board selection pulldown you can find it near the bottom)



D) Enable to use file system of the VS-RC202

VS-RC202 has a Wi-Fi function. It can operate as a simple web server and deliver the HTML file. In this section, upload HTML file to the VS-RC202.

You can upload files (like HTML) to flash memory of VSRC202 from the Arduino IDE by using the esp8266fs-plugin. Access the following URL and download ESP8266FS-XXX.zip.
*The version number is entered in XXX.

As of August 1, 2019, we have confirmed that it works with 0.4.0.

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

Unzip the ESP8266FS-***.zip, you can find folder named ESP8266FS. Move to sketch folder of Arduino and create tools folder and put the ESP8266FS folder in the tools folder.

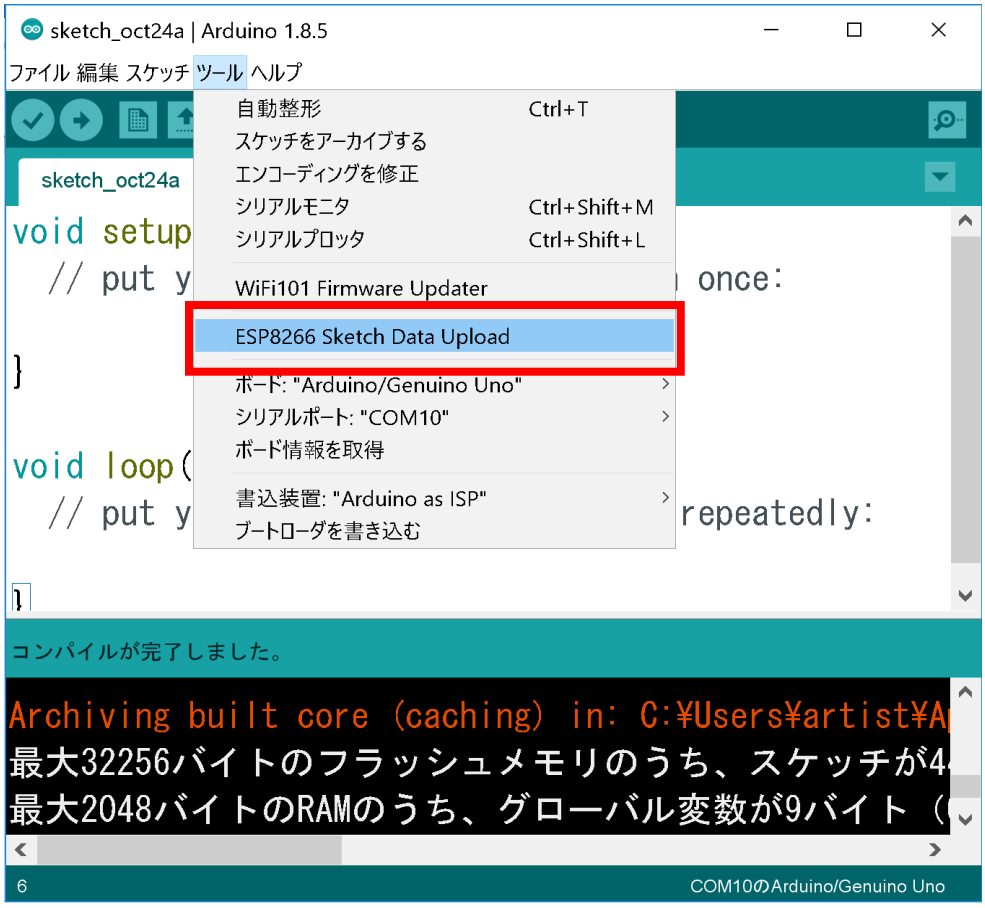
In the default setting, the Arduino Sketch folder is located under the Documents folder.

C: ¥ Users ¥username¥ Documents ¥ Arduino

Locate ESP8266 file as below

C: ¥ Users ¥ username ¥ Documents ¥ Arduino¥ Tools ¥ ESP8266FS ¥ tool ¥ esp8266fs.jar

After locating file restart Arduino IDE and make sure that you can find tools> ESP8266 Sketch Data Upload.



E) Enable to use the VS-RC202 libraries

Download the V-duino-ver.XXX.zip from the following URL.

*The version number is entered in XXX.

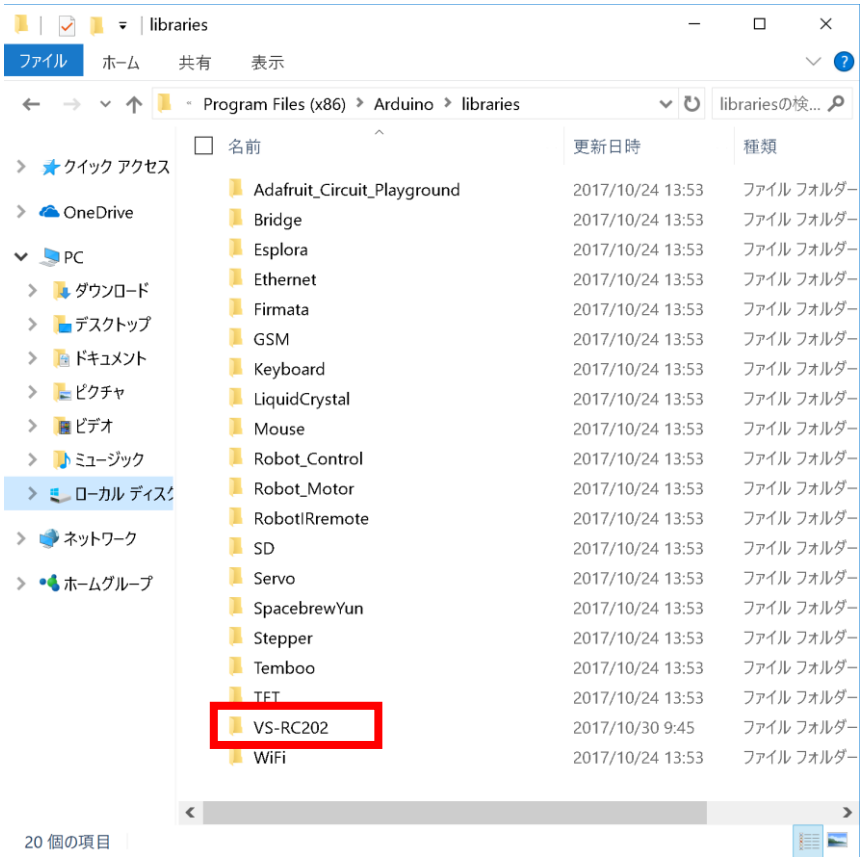
Please use the latest version.

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

Unzip it and you can find V-duino-ver.XXX folder. VS-RC202 folder is in it. Move the VS-RC202 folder into the Arduino libraries folder.

If you use Windows10 (64bit version), libraries folder is located in following path.

C: ¥ Program Files (x86) ¥ Arduino ¥ libraries



The setup of the software is completed.

11. vs-rc202.cpp List of Functions

A) Initialization

function	void initLib ()
Overview	To initialize libs. Must be executed at first.
Arguments and Return value	[Argument] None [Return value] None

Example of use	setup () {
	initLib ();
	}

B) Sensor / Power Management

function	void setSensEnable (int flag)
Overview	To set sensor reading enable/disable. default Enable
Arguments and Return value	[argument]Enable (1)/Disable (0) [Return value] None

Example of use	setup() {
	setSensEnable (1); //Sensor reading Enable
	setSensEnable (0); //Sensor reading Disble
	}

function	void setPullupEnable (int id, int flag)
Overview	To set pull-up resistor of the sensor pins enable/disable. default Disable
Arguments and Return value	[argument] Enable (1) / Disable (0) [Return value] None

Example of use	setup() {
	setPullupEnable (1); //Pull up Enable
	setPullupEnable (0); //Pull-up disable
	}

function	float getDist ()
Overview	To get the value of the ultrasonic sensor
Arguments and Return value	[Arguments None Return values] distance of the ultrasonic sensor (cm)

Example of use	loop() {
	float dist = getDist (); //Get the distance
	Serial.println (dist); //Display the distance
	}

function	int readSens (int id)
Overview	Get the value of the sensor(ADC)
Arguments and Return value	Arguments sensor number (1-3) [Return value] sensor value

Example of use	loop() {
	int sens [3];
	sens [0] = readSens (1);
	sens [1] = readSens (2);
	sens [2] = readSens (3);
	
	Serial.println (sens [0]);
	Serial.println (sens [1]);
	Serial.println (sens [2]);
	delay (500);
	}

function	int readPow ()
Overview	To get the power supply voltage
Arguments and Return value	[Argument] None [Return value] power supply voltage (mV)

Example of use	loop() {
	int power_voltage = readPow (); //Get the power supply voltage
	Serial.print ("Power voltage:");
	Serial.print (power_voltage); //Display the power supply voltage
	Serial.println ();
	delay (500);
	}

function	void powerOff ()
Overview	To turn off the power. (You can turn off the power in the remote control with this method)
Arguments and Return value	[Argument] None [Return value] None

Example of use	eventHandler {
	powerOff (); //You can program that in the case receive some command via Wi-Fi,
	// this method turns off the power.
	}

C) Servo motor control

function	void servoEnable (int id, int flag)
Overview	To set PWM enable/disable of each servo pin.
Arguments and Return value	[Argument] servo motor ID, Enable (1) / Disable (0) [Return value] None

Example of use	setup () {
	servoEnable (1, 1); // SV1 PWM Enable
	servoEnable (2, 0); // SV2 PWM Disable
	}

function	void setServoOffset (int id, int deg)
Overview	To set offset to each servo motor for adjust individual differences of the servo motor. Range -500 ~ 500
Arguments and Return value	Arguments servomotor ID, offset [Return value None

Example of use	setup () {
	setServoOffset (1, 500); // SV1 offset 500
	setServoOffset (2, -500); // SV2 offset -500
	}

function	void setServoLimitL (int deg)
Overview	To set the counterclockwise limit position of the servo motor. Range -2500 ~ 2500
Arguments and Return value	[Argument] limit position [Return value] None

Example of use	Setup () {
	setServoLimitL (-1800); //Counterclockwise limit position of the servomotor -1800
	serServoLimitL (500); //Limit position counterclockwise of the servo motor 500
	}

function	void setServoLimitH (int deg)
Overview	To set the clockwise limit position of the servo motor. Range -2500 ~ 2500
Arguments and Return value	[Argument] limit position [Return value] None

Example of use	Setup () {
	setServoLimitH (1800); //Limit position of clockwise servo motor1800
	serServoLimitH (-500); //Limit position of clockwise servomotor -500
	}

function	void setServoMode (int sv_mode)
Overview	To set the operation mode of the servo motor. Default, the sequential execution mode (1) Override mode (0): When you update the target position, ccurrent moving will be canceled and does next moving immidialtely. Sequential execution mode (1): When you update the target position, next moving will be done after current moving.
Arguments and Return value	[Argument] override mode (0) /Sequential execution mode (1) [Return value] None

Example of use	Setup () {
	setServoMode (0); //Override mode
	serServoMode (1); //Sequential execution mode
	}

function	void setServoMovingTime (int moving_time)
Overview	To set moving time of the servo motor to the target position. After executing setServoMovingTime() all servo motors move to the target position in setting time. Executing setServoMovingTime() again, moving time will be updated. Range 20 ~ 100000 (millisecond)
Arguments and Return value	[Argument] moving time (milliseconds) [Return value] None

Example of use	Setup () {
	setServoMovingTime (1000); //Moving time 1000ms
	setServoMovingTime (600); //Moving time 600ms
	}

function	void setServoDeg (int id, int deg)
Overview	To set the target position to the servo motor. This method is used with setServoMovingTime ().
Arguments and Return value	Arguments servomotor ID, target position [Return value None

Example of use	Loop () {
	setServoMovingTime (1000); //Moving time 1000msec
	setServoDeg (1, 1500); //Target position of SV1 = 1500
	setServoDeg (2, -1200); //Target position of SV2 = -1200
	moveServo (); // SV1,2 start to move to target position in 1000msec
	delay (1200); //Wait moving time + 200msec
	setServoMovingTime (1500);
	setServoDeg (1, -1200);
	setServoDeg (2,1500);
	moveServo ();
	:

function	void moveServo ()
Overview	To move servo motors. Moving speed is based on movint time set by setServoMovingTime().
Arguments and Return value	[Argument] None [Return value] None

Example of use	loop() {
	setServoMovingTime (1500); //Set moving time
	setServoDeg (1, 1500); //Set SV target position
	moveServo (); //Start moving
	}

function	void stopServo ()
Overview	To stop servo motors.
Arguments and Return value	[Argument] None [Return value] None

Example of use	loop() {
	setServoMovingTime (1500);
	setServoDeg (1, 1500);
	moveServo (); //Start moving
	moveServo ();
	delay (500);
	stopServo (); //Stop moving immidiately
	}

function	int readServoLimitL ()
Overview	To get the the counter-clockwise limit position of the servo motor
Arguments and Return value	[Arguments] None [Return values] limit position counterclockwise

Example of use	loop() {
	int limit_l = reasServoLimitL (); //Get counterclockwise limit position
	Serial.println (limit_l);
	delay (500);
	}

function	int readServoLimitH ()
Overview	Get clockwise limit position of the servo motor
Arguments and Return value	[Arguments None Return values] clockwise limit position

Example of use	loop() {
	int limit_h = reasServoLimitH (); //Get clockwise limit position
	Serial.println (limit_h);
	delay (500);
	}

function	int readServoOffset (int id)
Overview	To get the offset of the servo motor
Arguments and Return value	[Arguments] servomotor ID [Return value] Offset

Example of use	loop() {
	int offset = readServoOffset (1); //Get SV1 offset
	Serial.println (offset);
	delay (500);
	}

function	int readServoMovingTime ()
Overview	To get the movings time of the servo motor
Arguments and Return value	[Arguments] None [Return values] moving time of the servo motor

Example of use	loop() {
	int mv_time = readServoMovingTime (); //Get the moving time
	Serial.println (mv_time);
	delay (500);
	}

function	int readServoPos (int id)
Overview	To get the current position of the servomotor
Arguments and Return value	[Argument] servo motor ID [Return value] current position of the servo motor specified in

Example of use	loop() {
	int sv1_pos = readServoPos (1); //Get SV1 current position
	Serial.println (sv1_pos);
	delay (500);
	}

function	int servoAvailable ()
Overview	To get servo motor's statsu(moving or not). This method is used in Sequential execution mode (setServoMode (1)). In sequential execution mode, if SV_n_TPOS is updated, updated motion will be executed after completed current motion. ServoAvailable() returns 0 updated motion is not executed. ServoAvailable() returns 1 updated motion is executed (You can write next motion).
Arguments and Return value	[Argument] None [Return value] 0/ 1

Example of use	if (servoAvailable ()) { //Check you can write next motion or not.
	setMotion (motion [SV_NUM + 1]); //Writing the next motion
	}

function	void setMotion (int motion [SV_NUM + 1])
Overview	To set moving time and the target positions of all the servo motors. This method is user with servoAvailable(). If execute setMotion() when servoAvailable()=0, buffer (next motion) is overwritten.
Arguments and Return value	[Argument] motion array [Return value] None

Example of use	//From the elements 1, moving time, target position of SV1 ~ SV10
	int nextMotion [SV_NUM + 1] = {600,1000,1000,1000,1000,0,0,0,0,0,0};
	if (servoAvailable ()) { // Check you can write next motion or not.
	setMotion (nextMotion); //Writing the next motion
	}
	moveServo (); //Start moving

function	void setMotion (int motion [SV_NUM + 1], int sv_num)
Overview	To set moving time and the target positions of sv_num of servo motors from SV1. This method is user with servoAvailable(). If execute setMotion() when servoAvailable()=0, buffer (next motion) is overwritten.
Arguments and Return value	[Argument] motion array, the number of the servo motor [Return value] None

Example of use	#define sv_num = 4;
	//From the elements 1, moving time, target position of SV1 ~ SV4
	int nextPose [sv_num + 1] = {600,1000,1000,1000,1000};
	
	if (servoAvailable ()) { // Check you can write next motion or not.
	setMotion (nextPose, sv_num); //Writing the next motion
	}
	moveServo (); //Start moving

D) Play motion

Following 4 methods are used together.

function	void setMotionNumber (int motion_number)
Overview	To set the value to global variable motion_number To set the number of the motion you want to play in case received command from outside. This method is used with getMotionNumber () User can decide that which motion number corresponds to which motion.
Arguments and Return value	[Argument] motion number [Return value] None

function	int getMotionNumber ()
Overview	To get the value of the global variable motion_number
Arguments and Return value	[Argument] None [Return value] motion number

function	void playMotion (int motion [] [SV_NUM + 1], int array_length)
Overview	To repeat to play the motion. Motion is 2-dimensional array {transition time, SV1 target position ~, SV10 target value} In the case of setting array_length = 10. motion [0] -> motion [1] -> ... ->motion [9] -> motion [0] ... repeatedly executed
Arguments and Return value	[Argument] a two-dimensional array of motion, the number of motion [Return value] None

function	void playMotionOnce (int motion [] [SV_NUM + 1], int array_length)
Overview	To play the motion only once. Motion is 2-dimensional array {transition time, SV1 target position ~, SV10 target value} In the case of setting array_length = 10. motion [0] -> motion [1] -> ... ->motion [9] -> motion [0] ... executed just one time.
Arguments and Return value	[Argument] a two-dimensional array of motion, the number of motion [Return value] None

Motion playback sample code	

Example of use	//Prepare a motion in a two-dimensional array
	int motion1 [3] [11] = {{300,0,0, -600,0,0,0,0,0,0,0},
	{300,300,0, -600,0,0,0,0,0,0,0},
	{500,300,0,600,0,0,0,0,0,0,0}};
	int motion2 [4] [11] = {...};
	int getCommandFromUart () {
	//Function that receives a command via UART
	}
	//Sensor value as trigger, the motion number with MotionNumber().
	getCommand () {
	int cmd = getCommandFromUart ();
	swtich (cmd) {
	case 1:
	setMotionNumber (1);
	break;
	case 2:
	:
	}
	}
	//Run the motion which is set in motion number now.
	void selectMotion () {
	switch (getMotionNumber ()) {
	case 1:
	playMotion (motion1, 5);
	break;
	:
	}
	}
	//Repeat the command reading and running motion in the main loop
	void loop () {
	getCommand ();
	selectMotion ();
	}

E) LED Control

function	void setLedMode (int id, int flag)
Overview	To set LED mode to servo motor pin
Arguments and Return value	[Argument] servo motor ID, Enable (1) / Disable (0) [Return value] None

Example of use	servoEnable (1, 1); //Enable PWM of SV1
	servoEnable (2, 1); //Enable PWM of SV12
	setLedMode (1, 1); //Set LED mode to SV1
	setLedMode (2, 1); //Set LED mode to SV2
	setLedMode (2, 0); //Set normal mode to SV2

function	void setLedBrightness (int id, int brightness)
Overview	To set LED brightness to the pin. Range 0-1000. When moveServo() is executed, brightness of LED will change.
Arguments and Return value	[Argument] ID of LED, Brightness [Return value] None

Example of use	servoEnable (1, 1);
	setLedMode (1, 1); //Set LED mode to SV1
	setServoMovingTime (1000);
	setLedBrightness (1, 500); //Set brightness 500 to SV1
	moveServo (); //Change brightness of SV1 in 1000msec.

function	void setLedBrightnessDirect (int id, int brightness)
Overview	To set the LED brightness to the pin. Range 0-1000. After setting, LED brightness will change immediately. This method has higher priority than setLedBrightness(). If you want to use settLedBrightness (int id, int brightness), brightness 0f setLedBrightnessDirect() must be 0.
Arguments and Return value	[Argument] ID of LED, Brightness [Return value] None

Example of use	servoEnable (1, 1);
	setLedMode (1, 1); //Set LED mode to SV1
	setLedBrightnessDirect (1, 500); //Set brightness 500 to SV1

F) Buzzer Control

function	void buzzerEnable (int flag)
Overview	To switch Enable/Disable of the buzzer. (notice) When you enable the buzzer SV9, 10 will be disabled.
Arguments and Return value	[argument]Enable (1) / Disable (0) [Return value] None

Example of use	buzzerEnable (1); //To enable the buzzer. SV9, 10 are disabled.
	buzzerEnable (0); //To disable the buzzer. SV9, 10 are enabled.

function	void setBuzzer (int scale, int beat, int tang)
Overview	Sound the buzzer. The value of each argument is the macro (refer to next page). scale =Pitch (C D E F G A B) beat = The length of the sound (whole note, half notes etc) tang = tonguing or slur
Arguments and Return value	[Argument] pitch, the length of the sound, tonguing or slur [Return value] None

Example of use	buzzerEnable (1);
	setBuzzer (PC4, BEAT4, TANG); //Sound the quarter note C4 with tonguing
	setBuzzer (PC4, BEAT4, TANG);
	setBuzzer (PC4, BEAT4, TANG);
	
	setBuzzer (PC4, BEAT4, SLUR); / Sound the quarter note C4 with slur
	setBuzzer (PC4, BEAT4, SLUR);
	setBuzzer (PC4, BEAT4, SLUR);
	
	/*
	Tempo is based on whole note (BEAT1 = 1000millisecond)
	If you want to change the tempo, you need to change BEAT1 in the library.
	#define BEAT1 1000 //Whole note
	#define BEAT2 BEAT1/2 //Note 2 minutes
	#define BEAT4 BEAT1/4 //4 minutes note
	#define BEAT8 BEAT1/8 //Eighth note
	
	#define TANG_LENGTH 20 //Breathing time of tonguing = 20millisecond
	*/

Buzzer for macro List	
Interval	0 = Soundless 1 to88 = 1 of the keyboard -88 SH = # #define PN 0 #define PA0 1 #define PA0_SH 2 #define PB0 3 #define PC1 4 #define PC1_SH 5 #define PD1 6 #define PD1_SH 7 #define PE1 8 #define PF1 9 #define PF1_SH 10 #define PG1 11 #define PG1_SH 12 #define PA1 13 #define PA1_SH 14 #define PB1 15 #define PC2 16 #define PC2_SH 17 #define PD2 18 #define PD2_SH 19 #define PE2 20 #define PF2 21 #define PF2_SH 22 #define PG2 23 #define PG2_SH 24 #define PA2 25 #define PA2_SH 26 #define PB2 27 #define PC3 28 #define PC3_SH 29 #define PD3 30 #define PD3_SH 31

	#define PE3 32 #define PF3 33 #define PF3_SH 34 #define PG3 35 #define PG3_SH 36 #define PA3 37 #define PA3_SH 38 #define PB3 39 #define PC4 40 #define PC4_SH 41 #define PD4 42 #define PD4_SH 43 #define PE4 44 #define PF4 45 #define PF4_SH 46 #define PG4 47 #define PG4_SH 48 #define PA4 49 #define PA4_SH 50 #define PB4 51 #define PC5 52 #define PC5_SH 53 #define PD5 54 #define PD5_SH 55 #define PE5 56 #define PF5 57 #define PF5_SH 58 #define PG5 59 #define PG5_SH 60 #define PA5 61 #define PA5_SH 62 #define PB5 63 #define PC6 64 #define PC6_SH 65 #define PD6 66 #define PD6_SH 67

	#define PE6 68 #define PF6 69 #define PF6_SH 70 #define PG6 71 #define PG6_SH 72 #define PA6 73 #define PA6_SH 74 #define PB6 75 #define PC7 76 #define PC7_SH 77 #define PD7 78 #define PD7_SH 79 #define PE7 80 #define PF7 81 #define PF7_SH 82 #define PG7 83 #define PG7_SH 84 #define PA7 85 #define PA7_SH 86 #define PB7 87 #define PC8 88
The length of the sound [ms]	#define BEAT1 1000 //Whole note #define BEAT2 BEAT1/2 //Half note #define BEAT4 BEAT1/4 //Quarter note #define BEAT8 BEAT1/8 //Eighth note
Tonguing and Slur	#define TAN 20 //The length of the tonguing [ms] #define SLUR 0 //Slur #define TANG 1 //Tonguing

G) Reading and writing of the memory map

Refer to the appendix about the address of the memory map.

function	int read2byte (unsigned char addr)
Overview	To read 2byte data from the memory map
Arguments and Return value	[Argument] the address of memory map [Return value]2byte data

Example of use	int sv_1_pos = read2byte (SV_1_POS); // Read the current position of SV1
	int sv_2_pos = read2byte (SV_2_POS); // Read the current position of SV2

function	int read1byte (unsigned char addr)
Overview	To read 1byte data from the memory map
Arguments and Return value	[Argument]the address of memory map [Return value]1byte data

Example of use	int sens_enable = read1byte (SENS_ENABLE); //Read sensor enable flag
	int pwm_enable = read1byte (PWM_ENABLE1); //Read SV1 PWM enable flag.

function	int write2byte (unsigned char addr, short data)
Overview	To write 2byte data in the memory map
Arguments and Return value	[Argument]the address of memory map [Return value] 2byte data

Example of use	write2byte (SV_1_TPOS, 1000); // Write 1000 to SV1 target position address.
	write2byte (SV_2_TPOS, 500); // Write 1000 to SV2 target position address.

function	int write1byte (unsigned char addr, short data)
Overview	To write 1byte data in the memory map
Arguments and Return value	[Argument]the address of memory map [Return value] 1byte data

Example of use	write1byte (PWM_ENABLE1, 1); //Write SV1 PWM enable flag.
	write1byte (LED_MODE1, 1); // Write SV1 LED mode flag.

12. How to edit memory map directly

ESP-WROOM-02 and LPC1113 are connected on I2C bus, you can control VS-RC202 by editing memory map directly besides using methods in libraries. I2C address of device and memory map is registered as macro in VS-RC202 libraries.

Device address : DEV_ADDR
Address of each register : Refer to memory map at end of this manual
Data format : Little endian

Example 1 Write moving time of servo motors with Arduino Wire method

```
int moving_time = 1000;                    //Transition time 1000msec
char upper_sv_movint_time = moving_time << 8;
char lower_sv_movint_time = moving_time;

Wire.beginTransmission (DEV_ADDR);         //Send device address
Wire.write (SV_MV_TIME);                   //Send index register address
Write.write(lower_sv_movint_time);         //Lower byte of SV_MV_TIME
Write.write (upper_sv_movint_time);        //Upper byte of SV_MV_TIME
Wire.endTransmission ();                   //Send Stop condition
```

You can write all data into register which address is continuous in a single communication.
Example 2 Write moving time and target position of servo motor in a single communication.

```
Wire.beginTransmission (DEV_ADDR);         //Send device address
Wire.write (SV_MV_TIME);                   //Send index register address
Write.write (upper_sv_movint_time);        //Lower byte of SV_MV_TIM
Write.write (lower_sv_movint_time);        //Upper byte of SV_MV_TIME
Write.write (lower_sv_1_tpos);             //Lower byte of SV_1_TPOS
Write.write (upper_sv_1_tpos);             //Upper byte ofSV_1_TPOS
            :
Wire.endTransmission ();                   //Send Stop condition
```

If you want to read data from memory map, send the index address and the length of data you want to read from index address. Once write index address to the memory map and finish connection and then reconnect with read mode. You can read data from the index address.

Example 3 Read the current position of servo motor by using Arduino Wire method.

```
Wire.beginTransmission (DEV_ADDR);      // Establish the connection with write mode
Wire.write (SV_1_POS);                  //Write index address
Wire.endTransmission ();                //Finish the communication

unsigned char tmp [20];
int index = 0;
Wire.requestFrom (DEV_ADDR, 20);         //Re-connect with read mode
while (Wire.available ()) {              //Read 20 bytes from index address(SV_1_POS)
    tmp [index ++] = Wire.read ();
}
```

13. Memory map

[Addr] : Register address

[R/ W] : R [Readable], W [Writable]

[Sign] : U (unsigned), S(Signed)

[Byte] : Register size

Addr	R / W	Register Name	Sign	Byte	initial value	Remarks
0x00	R / W	SENS_ENABLE	U	1	0x01	Setting of ADC 0x00 = Disable 0x01=Enable
0x01	R / W	POW_OFF	U	1	0x00	Remote power OFF flag When 0x01 is written VS-RC202 will shutdown (With USB power supply, does NOT)
0x02	R	P_SW	U	1	0xff	Power button flag Pressed :0x01 Released :0x00
0x03	R / W	POW_ENABLE	U	1	0x01	Automatic shutdown setting If power supply voltage is less than P_TH, VS-RC202 will shut down. 0x00 =Disable 0x01 = Enable
0x0Four	R	SENS_1	U	2	0x0000	Sensor value (AN1 ~AN3) Range 0-1023
0x06	R	SENS_2	U	2	0x0000	
0x08	R	SENS_3	U	2	0x0000	
0x0a	R / W	PULLUP_1	U	1	0x00	Setting of pull-up resistor for AN1~ AN3 0x00 = Disable 0x01 = Enable
0x0b	R / W	PULLUP_2	U	1	0x00	
0x0c	R / W	PULLUP_3	U	1	0x00	
0x0e	R	POWER	U	1	0xff	Power-supply voltage (mA)
0xTen	R / W	P_TH	U	2	0xf811	Shutdown threshold voltage (mA) (Little-endian)0xf811 = 0x11f8 = 4600 mA
0x12	R / W	SV_L_LIMIT	S	2	0xf8f8	Counterclockwise limit position (range -2500 ~ 2500) **[Careful] Some servos have narrow movable range.** **DO NOT configure too large limit for the servos.**
0x1Four	R / W	SV_H_LIMIT	S	2	0x0807	Clockwise limit position (range -2500~ 2500) **[Careful] Some servos have narrow movable range.** **DO NOT configure too large limit for the servos.**
0x16	R / W	SV_1_OFFSET	S	2	0x0000	Offset of the servo motor 1 ~ 10 range -500 ~ 500
0x18	R / W	SV_2_OFFSET	S	2	0x0000	

0x1a	R / W	SV_3_OFFSET	S	2	0x0000	(Example)
						Set 200 offset to SV_1
						SV_1_OFFSET = 0x00c8 //Offset 200
						SV_1_TPOS = 0x05dc //Target position 1500
						//Offset + target position = 1700
						Actual target position = 0x06a4

0x1c	R / W	SV_4_OFFSET	S	2	0x0000	
0x1e	R / W	SV_5_OFFSET	S	2	0x0000	
0x20	R / W	SV_6_OFFSET	S	2	0x0000	
0xtwenty two	R / W	SV_7_OFFSET	S	2	0x0000	
0x2Four	R / W	SV_8_OFFSET	S	2	0x0000	
0x26	R / W	SV_9_OFFSET	S	2	0x0000	
0x28	R / W	SV_10_OFFSET	S	2	0x0000	
0x2a	R	SV_1_POS	S	2	0x0000	The current position of the servo motor SV1 to SV10
0x2c	R	SV_2_POS	S	2	0x0000	
0x2e	R	SV_3_POS	S	2	0x0000	
0x30	R	SV_4_POS	S	2	0x0000	
0x32	R	SV_5_POS	S	2	0x0000	
0x3Four	R	SV_6_POS	S	2	0x0000	
0x36	R	SV_7_POS	S	2	0x0000	
0x38	R	SV_8_POS	S	2	0x0000	
0x3a	R	SV_9_POS	S	2	0x0000	
0x3c	R	SV_10_POS	S	2	0x0000	
0x3e	R / W	PWM_ENABLE1	U	1	0x00	ON/OFF off the PWM signal 0x00 = PWM_OFF 0x01 = PWM_ON Immediately after power on, PWM is OFF (servo motors are free) **If you want to move servo motors, writes 0x01 to PWM_ENABLEn first.**
0x3f	R / W	PWM_ENABLE2	U	1	0x00	
0x40	R / W	PWM_ENABLE3	U	1	0x00	
0x41	R / W	PWM_ENABLEFour	U	1	0x00	
0x42	R / W	PWM_ENABLEFive	U	1	0x00	
0x43	R / W	PWM_ENABLE6	U	1	0x00	
0x44	R / W	PWM_ENABLE7	U	1	0x00	
0x45	R / W	PWM_ENABLE8	U	1	0x00	
0x46	R / W	PWM_ENABLE9	U	1	0x00	
0x47	R / W	PWM_ENABLETen	U	1	0x00	
0x48	R / W	SV_MV_TIME	U	2	0x0000	Moving time from the current position to the target position [ms] (Example) 0x03e8 = 1000 msec
0x4a	R / W	SV_1_TPOS	S	2	0x0000	Target position of the servomotor 1-10 Setting range = within limit position **Actual movable range = limit position + offset**
0x4c	R / W	SV_2_TPOS	S	2	0x0000	
0x4e	R / W	SV_3_TPOS	S	2	0x0000	

0x50	R / W	SV_4_TPOS	S	2	0x0000	(Example) SV_L_LIMIT = -1800 SV_H_LIMIT = 1800 SV_1_OFFSET = 500 Setting range -1800 to 1800 Actual movable range -1300 ~ 2300
0x52	R / W	SV_5_TPOS	S	2	0x0000	
0x54	R / W	SV_6_TPOS	S	2	0x0000	
0x56	R / W	SV_7_TPOS	S	2	0x0000	
0x58	R / W	SV_8_TPOS	S	2	0x0000	
0x5a	R / W	SV_9_TPOS	S	2	0x0000	
0x5c	R / W	SV_10_TPOS	S	2	0x0000	
0x5e	R / W	SV_START	U	1	0x00	The servos start to move when 0x01 is written in SV_START after setting the SV_MV_TIME and SV_n_TPOS.
0x5f	R / W	SV_CANCEL	U	1	0x00	To cancel the current movement by setting 0x01
0x60	R	SV_STATUS	U	1	0x00	0x00 servo motor is stopped 0x01 servo motor is moving
0x61	R	BUF_STATUS	U	1	0x00	0x00 buffer is empty 0x01 contains the following values in the buffer
0x62	R / W	SV_MODE	U	1	0x00	0x00 override mod 0x01 sequential execution mode

0x64	R / W	LED_MODE1	U	1	0x00	Switching the servo motor pin to LED mode
						0x00 =Servo motor mode 0x01 =LED mode
						In LED mode, SV_n_TPOS becomes LED brightness.
						range 0 ~ 1000
						
						(Ex) Switch SV10 to LED mode and turn on it.
						LED_MODE10 = 0x01
						SV_10_TPOS = 1000 //Max brightness
						SV_START = 0x01

0x65	R / W	LED_MODE2	U	1	0x00	
0x66	R / W	LED_MODE3	U	1	0x00	
0x67	R / W	LED_MODE4	U	1	0x00	
0x68	R / W	LED_MODE5	U	1	0x00	
0x69	R / W	LED_MODE6	U	1	0x00	
0x6a	R / W	LED_MODE7	U	1	0x00	
0x6b	R / W	LED_MODE8	U	1	0x00	
0x6c	R / W	LED_MODE9	U	1	0x00	
0x6d	R / W	LED_MODE10	U	1	0x00	
0x6e	R/ W	LED_BRIGHT1	U	2	0x0000	In LED mode, value (LED brightness) of LED_BRIGHTn be reflect immediately. LED_BRIGHTn has higher priority than V_n_TPOS range 0 ~ 1000 LED_BRIGHTnTo is used to light LED immediately. SV_n_TPOSTo is used to light LED slowly.
0x70	R/ W	LED_BRIGHT2	U	2	0x0000	
0x72	R/ W	LED_BRIGHT3	U	2	0x0000	
0x74	R/ W	LED_BRIGHT4	U	2	0x0000	
0x76	R/ W	LED_BRIGHT5	U	2	0x0000	
0x78	R/ W	LED_BRIGHT6	U	2	0x0000	
0x7a	R/ W	LED_BRIGHT7	U	2	0x0000	

0x7c	R/ W	LED_BRIGHT8	U	2	0x0000	In LED mode, if you want to use V_n_TPOS, LED_BRIGHTn must be 0x0000.
0x7e	R/ W	LED_BRIGHT9	U	2	0x0000	
0x80	R/ W	LED_BRIGHT10	U	2	0x0000	
0x82	R / W	BUZZER_ENABLE	U	1	0x00	Enable /Disable Buzzer 0x00 = buzzer Disable 0x01=Buzzer Enable **When buzzer is Enable SV9, 10 is not available**
0x83	R / W	BUZZER_SCALE	U	1	0x00	Tone of the buzzer 0x00 = silence 0x01 ~0x58 = Corresponding to the 88 board of the piano (Example) BUZZER_SCALE = 0x34 (= 52) C5 BUZZER_SCALE = 0x49 (= 73) A6