Infonique

iSEB Expansion Board 1200 0012 V1.1

|  |  |  |
| --- | --- | --- |
| Prepared by | Date | Version |
| Bing Ran | 15/7/2023 | 1.0 |

# Abstract

This document provides detailed of Infonique iSEB Expansion Board 1200 0012 V1.1 specification.

# Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Rev** | **Modifier** | **Changes** |
| **15-July-2023** | 1.0 | Bing Ran | First Draft |
| **10-Sept-2023** | 1.1 | Bing Ran | Further elaboratoin about WiFi and servo motor |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Contents

[Abstract 2](#__RefHeading___Toc1131_1720759766)

[Document History 2](#__RefHeading___Toc1133_1720759766)

[Contents 3](#__RefHeading___Toc1135_1720759766)

[1 Introduction 5](#__RefHeading___Toc1141_1720759766)

[2 Hardware 6](#__RefHeading___Toc450_3933630546)

[2.1 Schematic 6](#__RefHeading___Toc452_3933630546)

[2.2 Pinout 7](#__RefHeading___Toc1042_1019879922)

[2.2 PCB Layout 8](#__RefHeading___Toc456_3933630546)

[2.2.1 Label of legs 9](#__RefHeading___Toc458_3933630546)

[2.2.2 PWM control 10](#__RefHeading___Toc460_3933630546)

[2.2.3 Switch , Battery Connector & RGB Led 12](#__RefHeading___Toc1044_1019879922)

[2.3 Bom list 12](#__RefHeading___Toc1046_1019879922)

[3 Firmware 13](#__RefHeading___Toc464_3933630546)

[3.1 Specification of the ESP32 DevKit V1 13](#__RefHeading___Toc1048_1019879922)

[3.2 Environment set up 14](#__RefHeading___Toc1050_1019879922)

[3.3 WiFi 19](#__RefHeading___Toc996_1617206542)

[3.3.1 How the WiFi Code works 19](#__RefHeading___Toc1010_1617206542)

[3.3.2 WiFi server and control UI 20](#__RefHeading___Toc1052_1019879922)

[3.4 Servo Motor 25](#__RefHeading___Toc998_1617206542)

[3.4.1 How the Servo Motor Code works 26](#__RefHeading___Toc1000_1617206542)

[3.4.1.1 Setup 26](#__RefHeading___Toc1002_1617206542)

[3.4.1.2 Code of motorInit function 27](#__RefHeading___Toc1002_1617206542_Copy_1)

[3.4.1.3 Update duty cycle during runtime 28](#__RefHeading___Toc1004_1617206542)

[3.4.1.4 Set\_PWM\_to\_Servo 28](#__RefHeading___Toc1021_1617206542)

Table of Figures

[Figure 1: Robot that controlled by iSEB Expansion Board 1200 0012 V1.0. 5](#Figure!0|sequence)

[Figure 2: Schemaitc of iSEB Expansion Board 1200 0012 V1.0 6](#Figure!1|sequence)

[Figure 3: iSEB Expansion Board 1200 0012 V1.0 with ESP32 Module 8](#Figure!2|sequence)

[Figure 4: iSEB Expansion Board 1200 0012 V1.0 without ESP32 Module 8](#Figure!3|sequence)

[Figure 5: Classification of legs and arms 9](#Figure!4|sequence)

[Figure 6: PWM control port 10](#Figure!5|sequence)

[Figure 7: ports for robot’s claws 11](#Figure!6|sequence)

[Figure 8: Robot's femur 11](#Figure!7|sequence)

[Figure 9: Switch , Battery connector & RGB led 12](#Figure!8|sequence)

[Figure 10: Pinout of ESP32 DevKit V1 13](#Figure!9|sequence)

[Figure 11: How servo’s position controlled by PWM signal 25](#Figure!10|sequence)

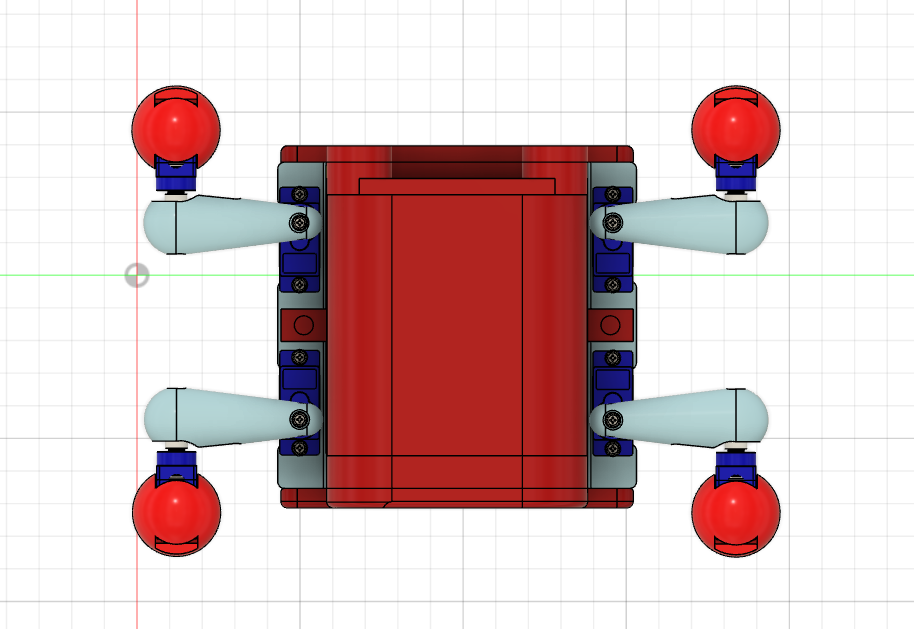
Index of Tables

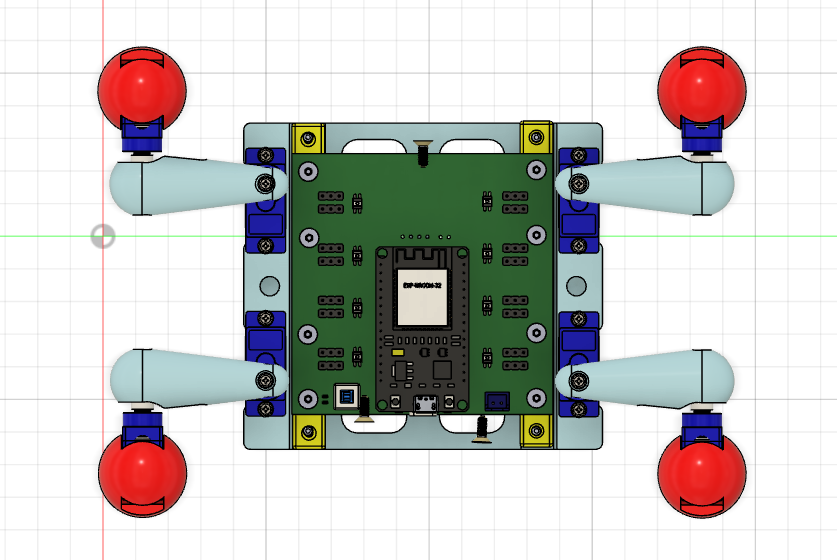
Table 1: Pinout 6

Table 2: Position vs GPIO vs Channel vs Connector matrix 25

# 1 Introduction

This document will discuss the details of the iSEB Expansion Board 1200 0012 V1.0. iSEB Expansion Board 1200 0012 V1.0 is design for a spider robot. It provide 8 rgb leds (red green blue leds ) and 16 pwm output to control the servo motors. The following figure is one of the robot that control with iSEB Expansion Board 1200 0012 V1.0.



Figure 1: Robot that controlled by iSEB Expansion Board 1200 0012 V1.0.

# 2 Hardware

## 2.1 Schematic

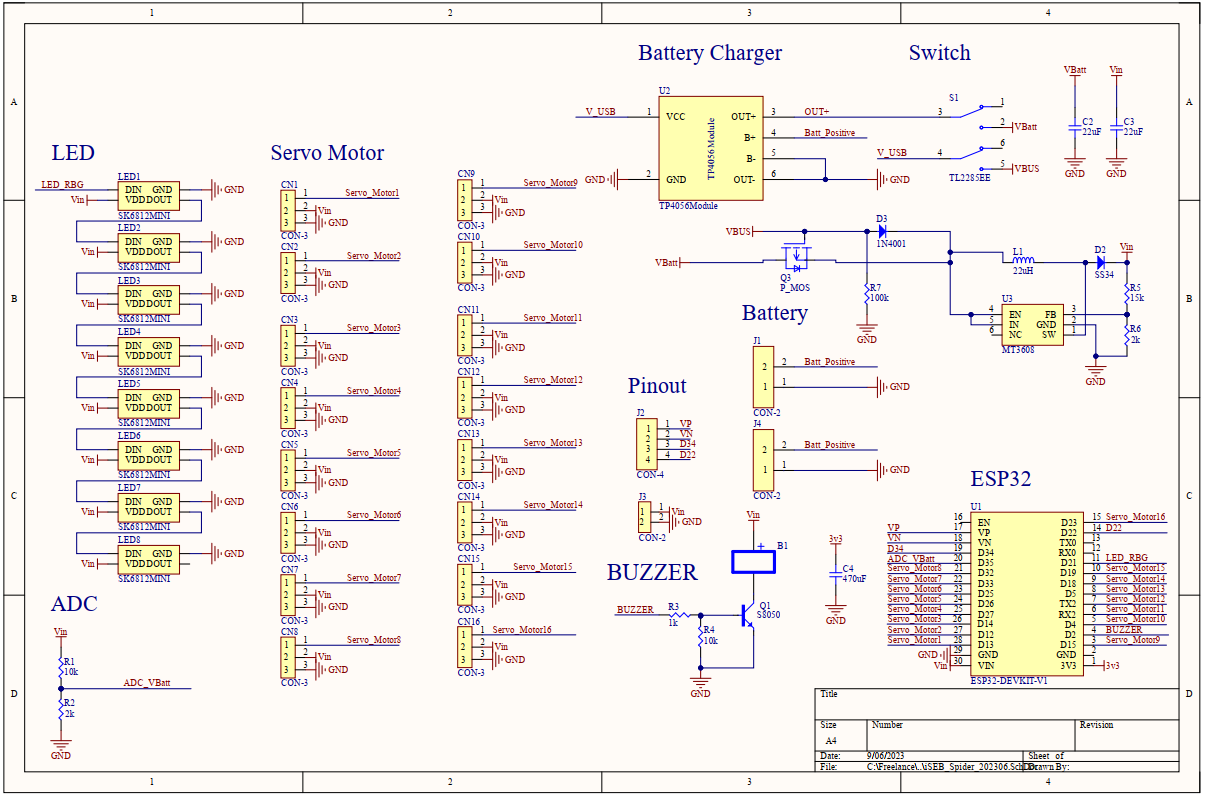


Figure 2: Schemaitc of iSEB Expansion Board 1200 0012 V1.0

## 2.2 Pinout

|  |  |  |  |
| --- | --- | --- | --- |
| **Pin** | **Function** | **Pin** | **Function** |
| EN | Enable Pin | D23 | PWM16 |
| VP | Unused | D22 | Unused |
| VN | Unused | TX0 | TX0 |
| D34 | Unused | RX0 | RX0 |
| D35 | ADC Vbatt | D21 | RGB Led |
| D32 | PWM8 | D19 | PWM15 |
| D33 | PWM7 | D18 | PWM14 |
| D25 | PWM6 | D05 | PWM13 |
| D26 | PWM5 | D17 | PWM12 |
| D27 | PWM4 | D16 | PWM11 |
| D14 | PWM3 | D04 | PWM10 |
| D12 | PWM2 | D02 | Buzzer |
| D13 | PWM1 | D15 | PWM9 |

Table 1: Pinout

## 2.2 PCB Layout

The following is the figure of the iSEB Expansion Board 1200 0012 V1.0

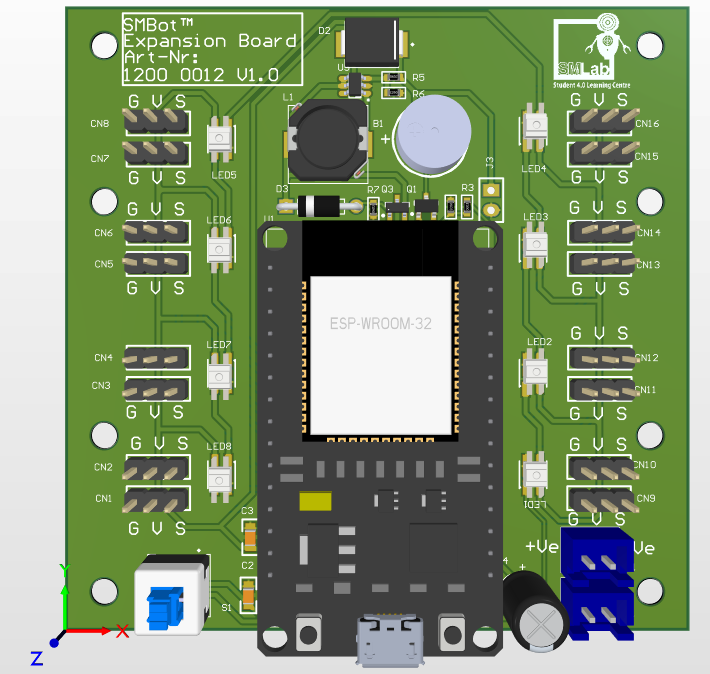


Figure 3: iSEB Expansion Board 1200 0012 V1.0 with ESP32 Module

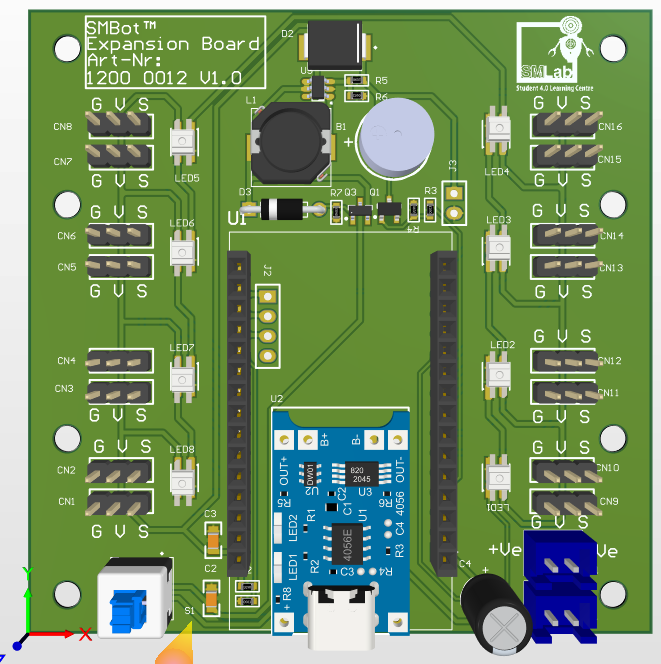


Figure 4: iSEB Expansion Board 1200 0012 V1.0 without ESP32 Module

## 2.2.1 Label of legs

The following figure is labeling the leg of the robot.

## 

Figure 5: Classification of legs and arms

## 2.2.2 PWM control

There are 16 PWM control port in iSEB Expansion Board 1200 0012 V1.0. The figure below is showing the locaiton of the 16 PWM control port.

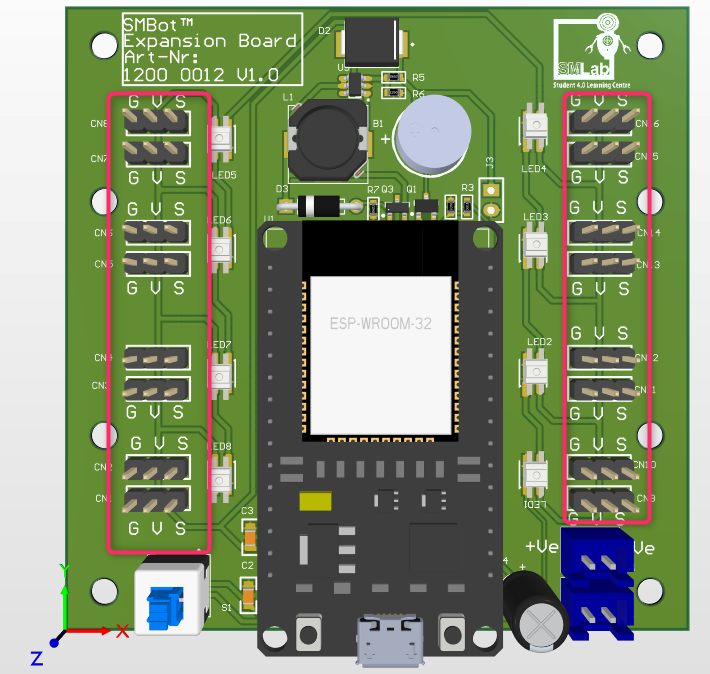
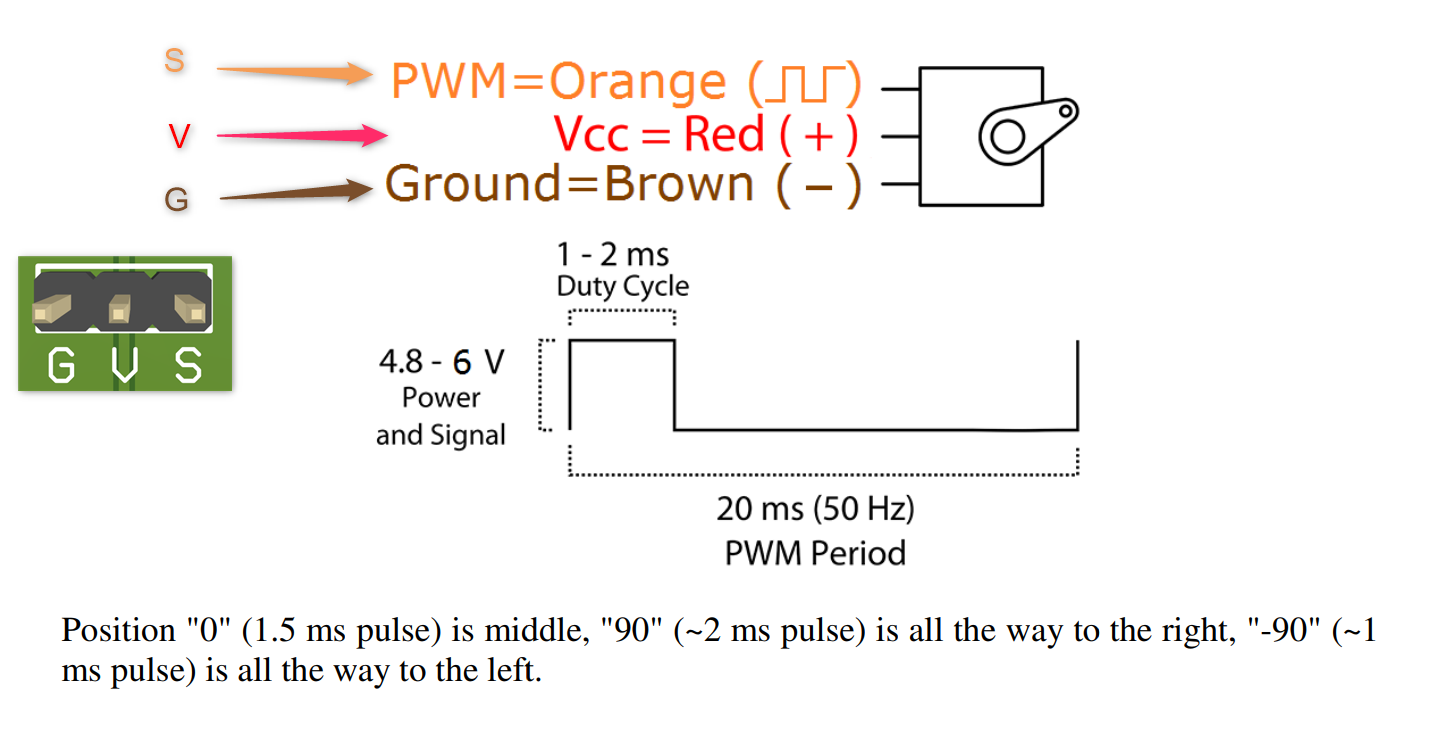
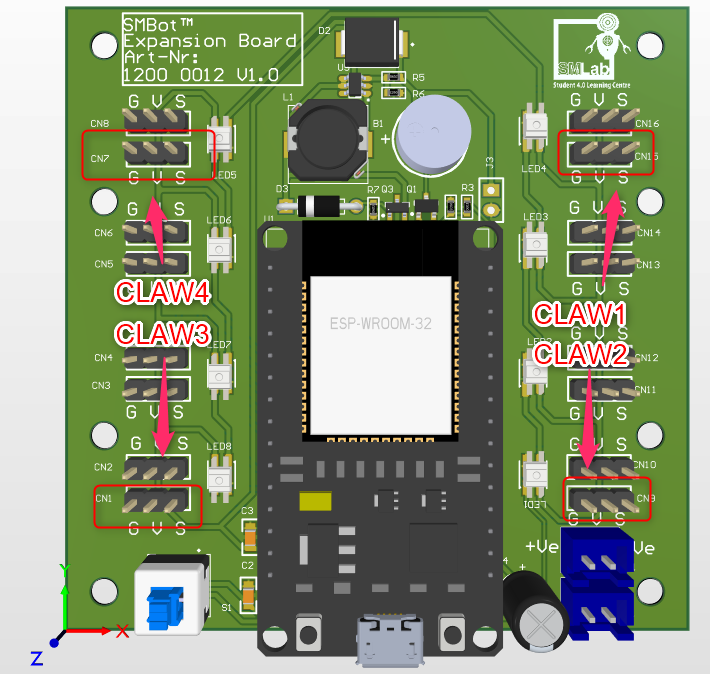


Figure 6: PWM control port

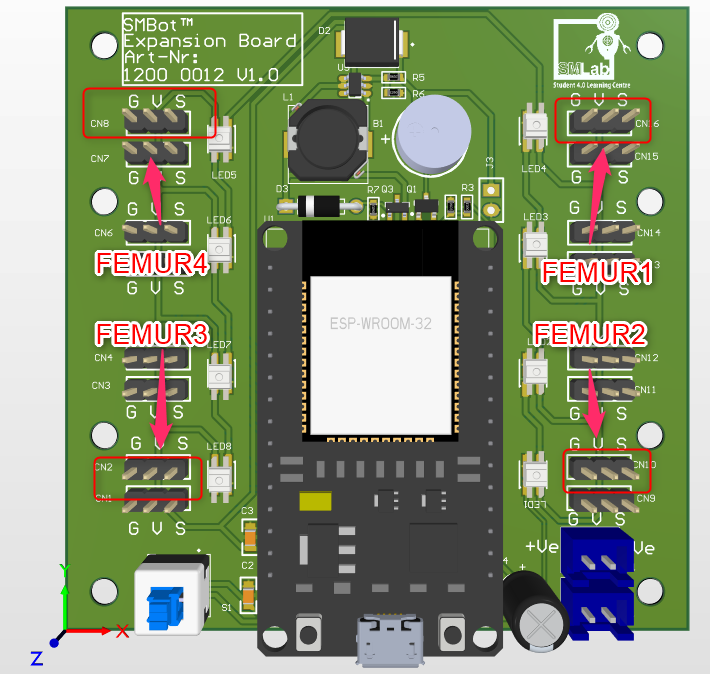
2.2.2.1 PWM Control Servo Motor Connection



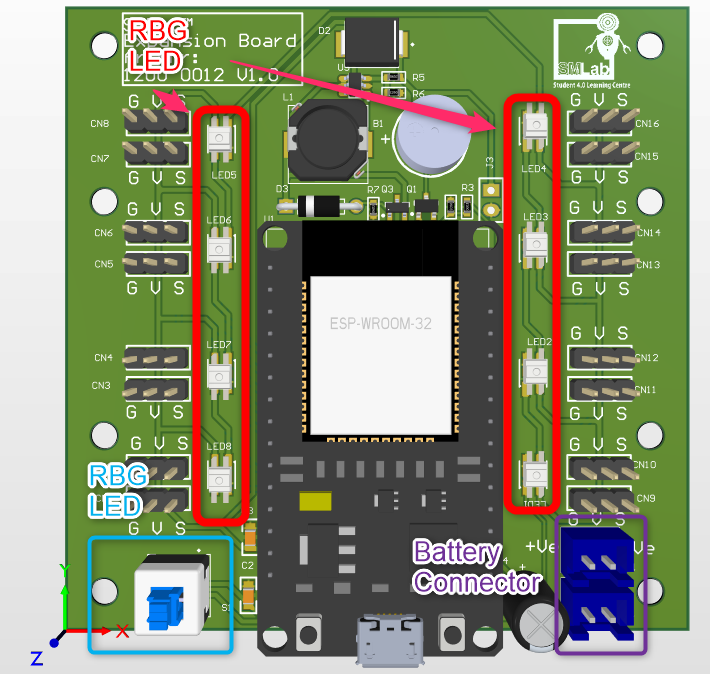
The figure below is specifying the port for each robot’s claw.

Figure 7: ports for robot’s claws

The figure below is specifying the port for each robot’s femur.

Figure 8: Robot's femur

## 2.2.3 Switch , Battery Connector & RGB Led

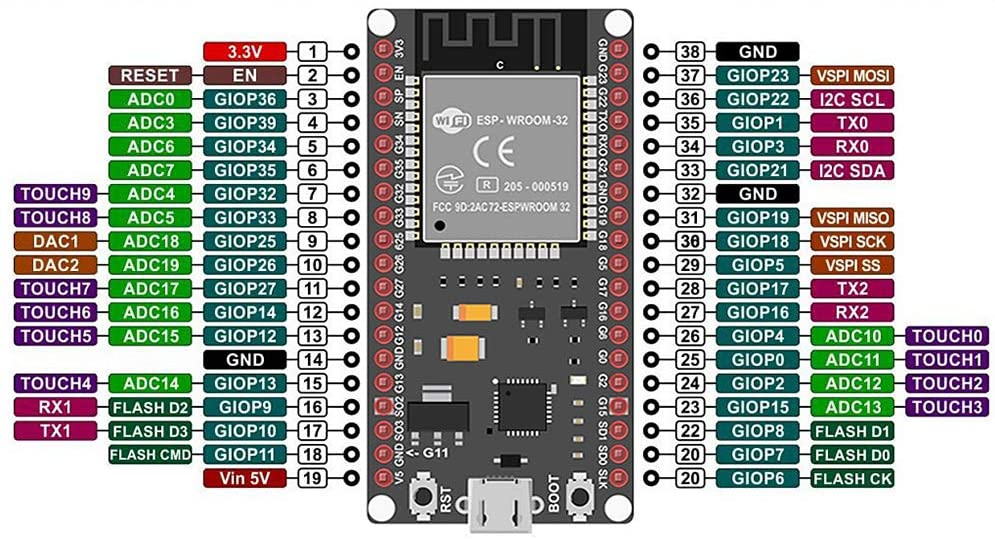
Figure 9: Switch , Battery connector & RGB led

## 2.3 Bom list

* iSEB Expansion Board 1200 0012 V1.0 with ESP32 Module x 1
* 18650 Battery x 2
* ESP32-DEVKIT-V1 x 1
* Spider chassis set x 1

# 3 Firmware

The iSEB Expansion Board 1200 0012 V1.0 is using ESP32 DevKit V1. The figure is showing the pinout of ESP32 DevKit V1.The microcontroller is esp-wroom-32 module.

Figure 10: Pinout of ESP32 DevKit V1

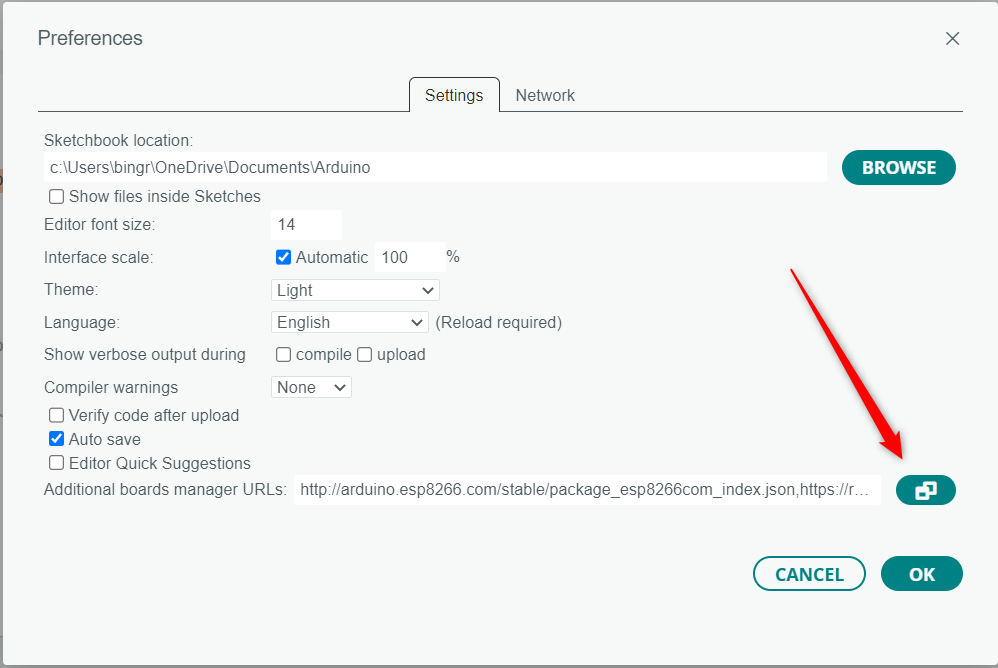
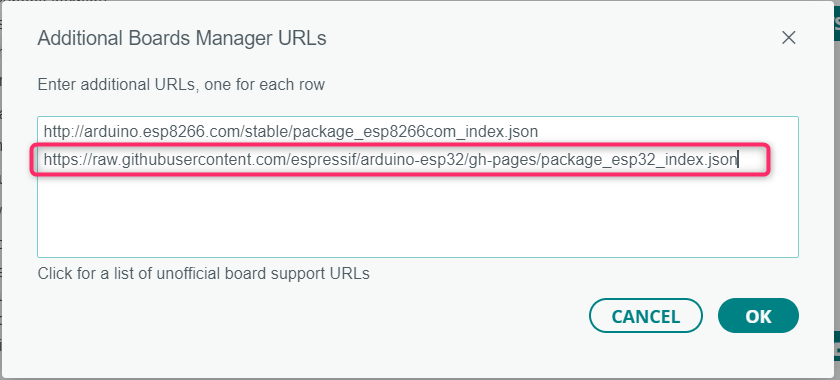
## 3.1 Specification of the ESP32 DevKit V1

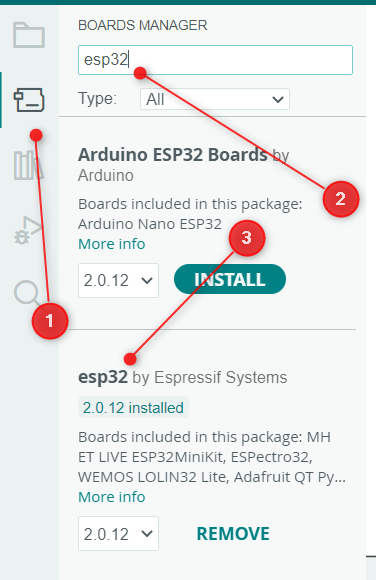
Microcontroller: Tensilica 32-bit Single-/Dual-core CPU Xtensa LX6

* Operating Voltage: 3.3V
* Input Voltage: 7-12V
* Digital I/O Pins (DIO): 25
* Analog Input Pins (ADC): 6
* Analog Outputs Pins (DAC): 2
* UARTs: 3
* SPIs: 2
* I2Cs: 3
* Flash Memory: 4 MB
* SRAM: 520 KB
* Clock Speed: 240 Mhz
* Wi-Fi: IEEE 802.11 b/g/n/e/i:
  + Integrated TR switch, balun, LNA, power amplifier and matching network
  + WEP or WPA/WPA2 authentication, or open networks
* Dimensions: 51.5x29x5mm

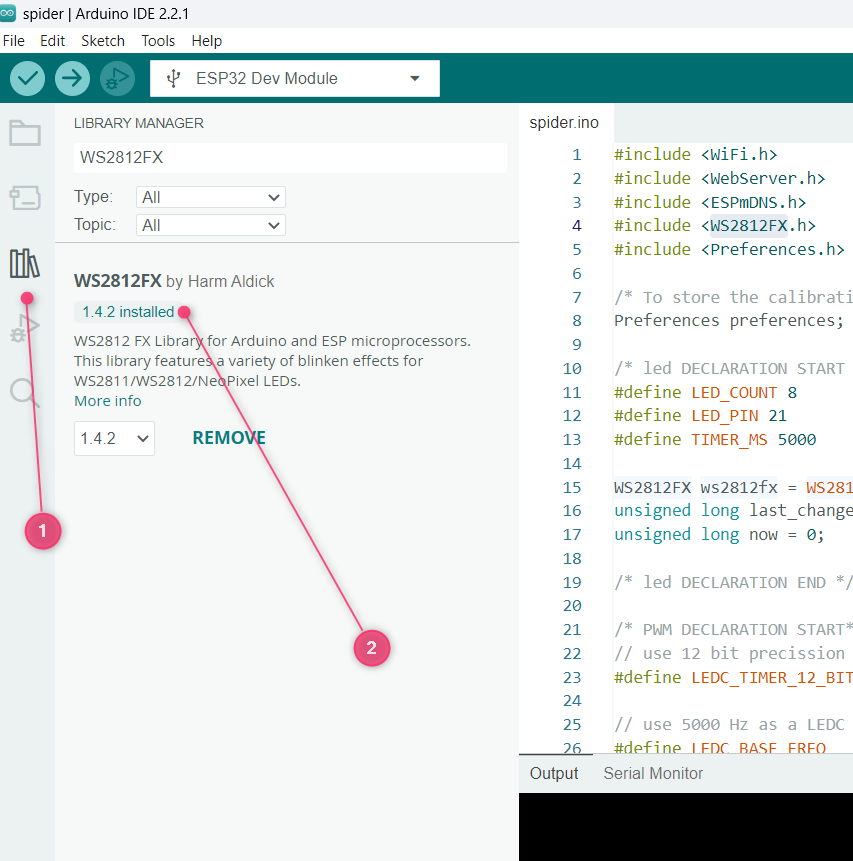
## 3.2 Environment set up

We need to set up the environment to flash the binary to ESP32 DevKit V1.

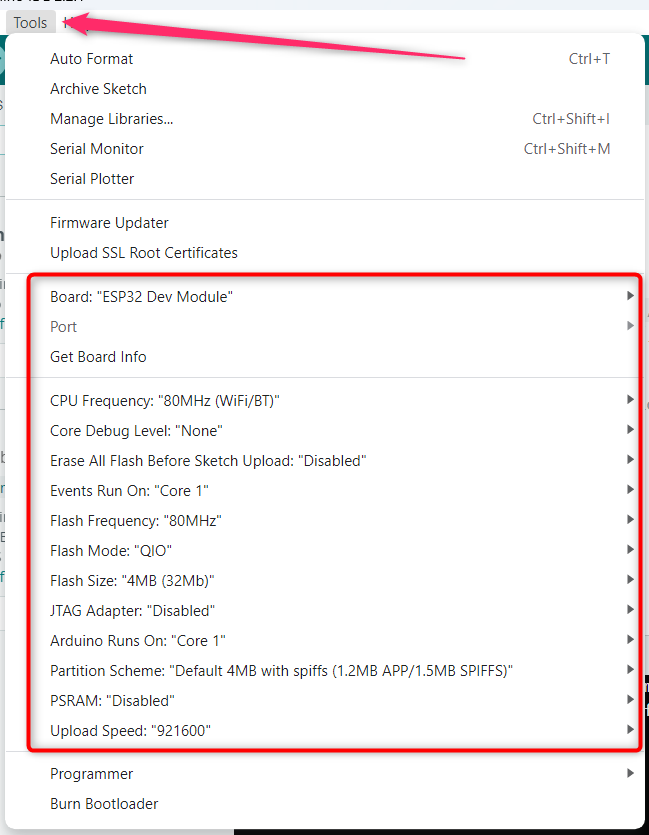
* Install Arduino IDE is requried to install. ( Snapshot is base on Arduino IDE 2.2.0 )
* Add <https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json> to Board Managers and install ESP32 libary.
  + Select Files-> Preferences and click on the icon
  + Add Boards Manager URLs
  + Install ESP32 by Espressif Systems at Board Manager.



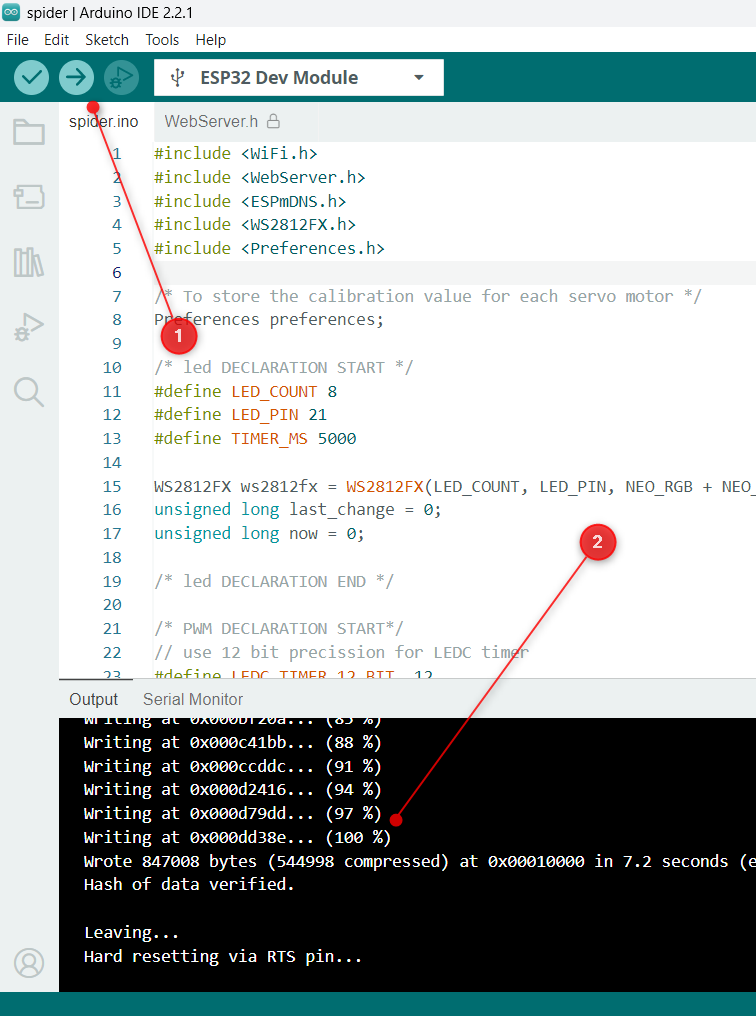
* Install WS2812FX by Harm Aldick ( version 1.4.2 ) library.
  + Snapshot of install libary



* Update the upload setting



* Click upload button and the firmware will be flashed successfully if the snapshot below is seen.



* The environment set up is done if the binary able to flash to ESP32 DevKit V1..

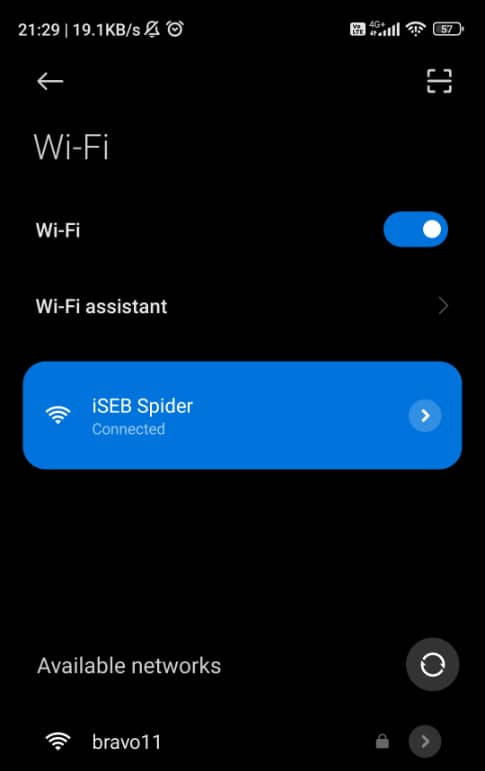
## 3.3 WiFi

### 3.3.1 How the WiFi Code works

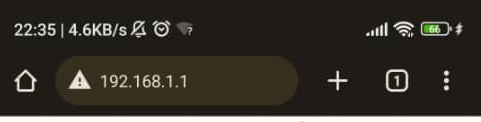
* Firstly we need to include WiFi and WebServer library
  + #include <WiFi.h>
    - WiFi.h - esp32 Wifi support.
  + #include <WebServer.h>
    - WebServer.h - Dead simple web-server. Supports only one simultaneous client, knows how to handle GET and POST.
* Secondly we need to insert our ssid and password
  + const char\* ssid = "iSEB Crab";  // Enter SSID here
  + const char\* password = "12345678";  //Enter Password here
* Then we set our web server to port 80
  + WebServer server(80);
* We have to setup the WiFi in setup function
  + To start the Wi-Fi as an Access Point.
    - WiFi.softAP(ssid);/\* without password \*/
    - WiFi.softAP(ssid,password);/\* with password \*/
  + Function used to configure the IP as static (fixed) as well as the gateway and subnet.
    - WiFi.softAPConfig(local\_ip, gateway, subnet); /\* to add exception to server \*/
  + Set up handling of web page
    - server.on("/",handleIndex);
    - server.on("/editor", handleEditor);
    - server.on("/controller", handleController);
    - server.on("/zero", handleZero);
    - server.on("/setting",handleSetting);
    - server.on("/save", handleSave);
  + Enable the server
    - server.begin();
* We have to handle the user request in loop funtion
  + - server.handleClient();

### 3.3.2 WiFi server and control UI

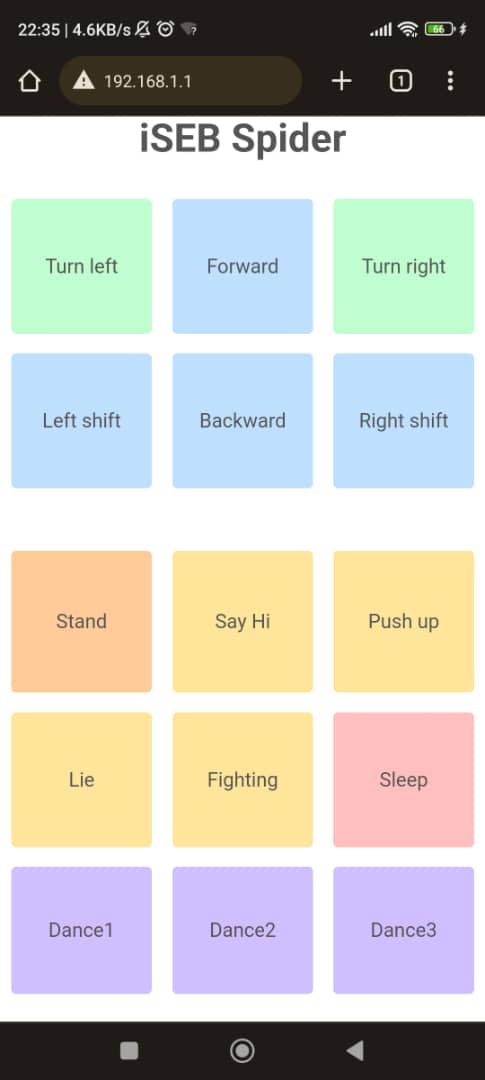
* After flash succesfully, the iSEB Spider should be appear in the WiFi list. The figure below is showing the iSEB spider is appeared in the WiFi list.



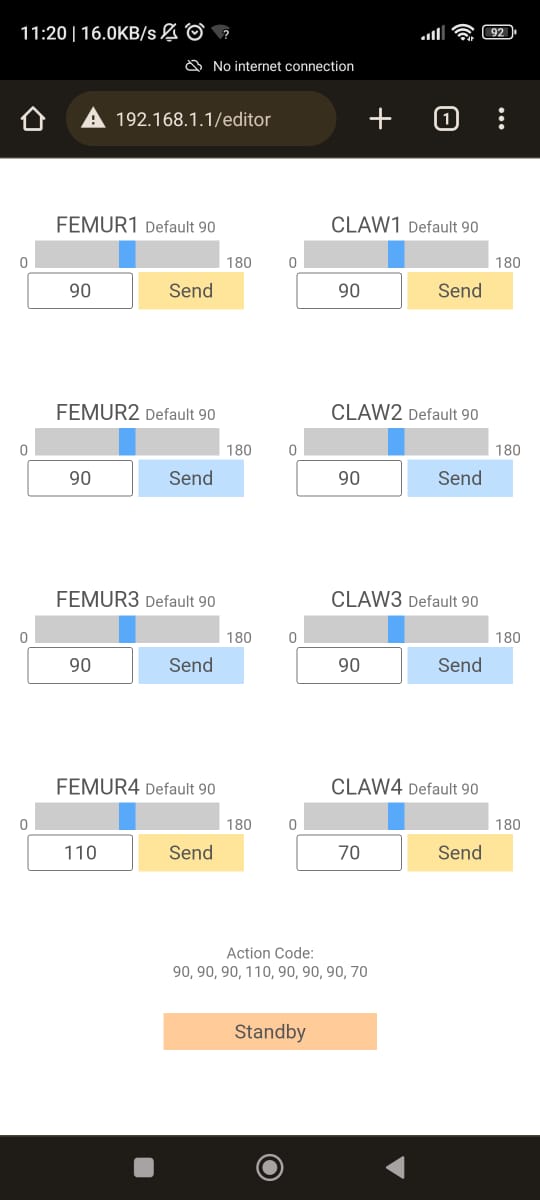
* Connect to the ISEB Spider and access 192.168.1.1



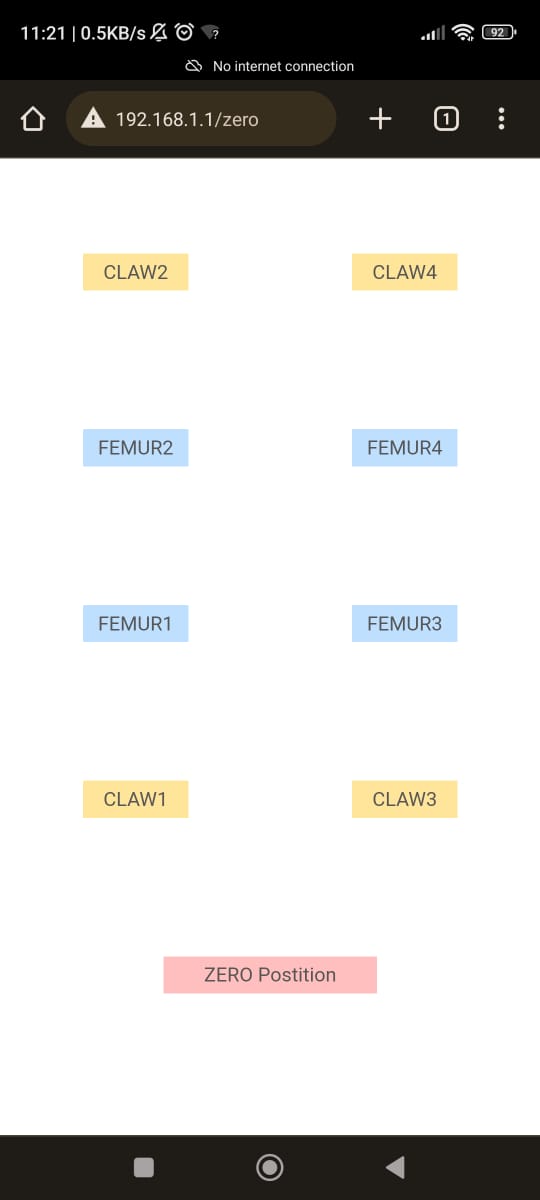
* The control page will be show as below



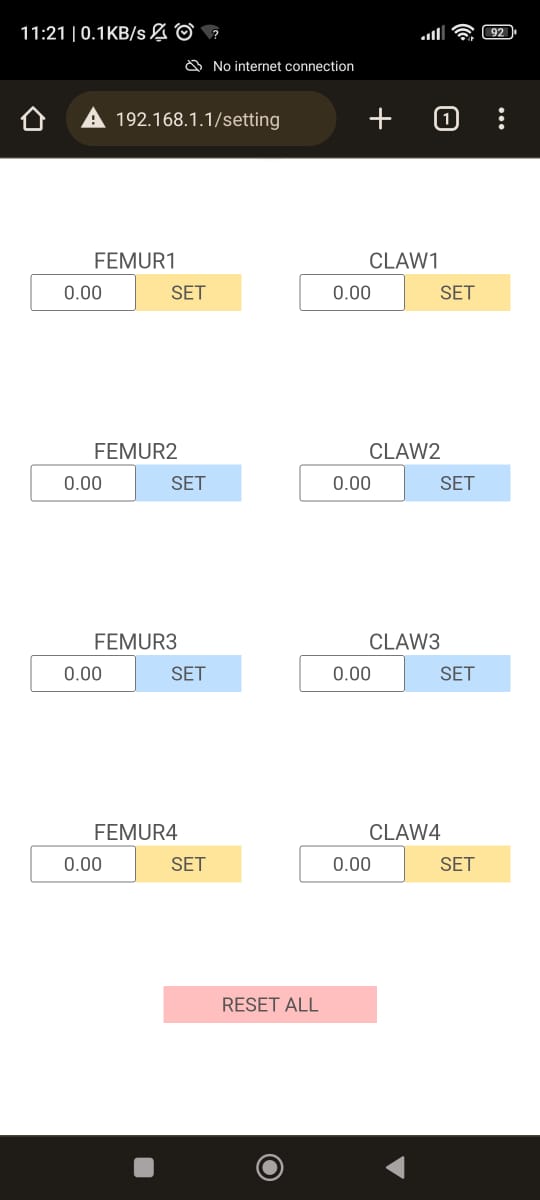
* The Motion Editor Page



* The Zero Page

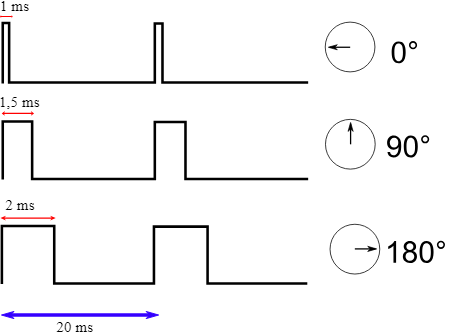


* The Setting page



## 3.4 Servo Motor

* The servo motor used in the iSEB Crab is TowerPro SG90 servo .
* The wire colors are Red = Battery(+) Brown = Battery(-) Orange = Signal
* The figure below show how the servo motor angle control by pwm
* Servo motor control with 50 Hz pulse width modulated (PWM) signal, which produces a pulse every 20ms.

Figure 11: How servo’s position controlled by PWM signal

### 3.4.1 How the Servo Motor Code works

#### 3.4.1.1 Setup

* We are using the LED Control library from ESP32 hal library to control servo motor.
* The LED control (LEDC) peripheral is primarly designed to control the intensity of LEDs, although it can also be used to generate PWM signals for other purposes. .
* For more details of the LEDC library can refer to the link
  + https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/api/ledc.html
* We able to generate PWM signals to control the servo motor.
* We have a motorInit funciton in the setup function to call the setup.
* We are calling funcition ledcSetupledc and ledcAttachPin in function motorInit.
* Function ledcSetupledc is used to setup the LEDC channel frequency and resolution.
  + uint32\_t ledcSetup(uint8\_t channel, uint32\_t freq, uint8\_t resolution\_bits);
    - channel select LEDC channel to config.
      * ESP32 have 16 channels
    - freq select frequency of pwm.
    - resolution\_bits select resolution for ledc channel.
      * range is 1-14 bits (1-20 bits for ESP32)
* Function ledcAttachPin is used to attach the pin to the LEDC channel.
  + void ledcAttachPin(uint8\_t pin, uint8\_t chan);
    - pin select GPIO pin.
    - chan select LEDC channel.
* The follow table is showing the GPIO vs Channel vs Connector in the example code

|  |  |  |  |
| --- | --- | --- | --- |
| **FEMUR/CLAW** | **GPIO** | **Channel** | **Connector** |
| FEMUR1 | 19 | 1 | CN15 |
| FEMUR1 | 15 | 2 | CN9 |
| FEMUR1 | 33 | 3 | CN7 |
| FEMUR1 | 13 | 4 | CN1 |
| CLAW1 | 23 | 5 | CN16 |
| CLAW2 | 4 | 6 | CN10 |
| CLAW3 | 32 | 7 | CN8 |
| CLAW4 | 12 | 8 | CN2 |

Table 2: Position vs GPIO vs Channel vs Connector matrix

#### 3.4.1.2 Code of motorInit function

// use 12 bit precission for LEDC timer

#define LEDC\_TIMER\_12\_BIT  12

// use 50 Hz as a LEDC base frequency

#define LEDC\_BASE\_FREQ     50

#define FEMUR\_1 0 /\* Chanel 0 \*/

#define FEMUR\_2 1 /\* Chanel 1 \*/

#define FEMUR\_3 2 /\* Chanel 2 \*/

#define FEMUR\_4 3 /\* Chanel 3 \*/

#define CLAW\_1 4 /\* Chanel 4 \*/

#define CLAW\_2 5 /\* Chanel 5 \*/

#define CLAW\_3 6 /\* Chanel 6 \*/

#define CLAW\_4 7 /\* Chanel 7 \*/

void motorInit()

{

  // Setup timer

  ledcSetup(FEMUR\_1, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(FEMUR\_2, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(FEMUR\_3, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(FEMUR\_4, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(CLAW\_1, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(CLAW\_2, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(CLAW\_3, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  ledcSetup(CLAW\_4, LEDC\_BASE\_FREQ, LEDC\_TIMER\_12\_BIT);

  // Attach timer to a led pin

  ledcAttachPin(19, FEMUR\_1);  /\* FEMUR\_1 \*//\* CN15 \*//\* PIN 19\*/

  ledcAttachPin(15, FEMUR\_2);  /\* FEMUR\_2 \*//\* CN9  \*//\* PIN 15\*/

  ledcAttachPin(33, FEMUR\_3);  /\* FEMUR\_3 \*//\* CN7  \*//\* PIN 33\*/

  ledcAttachPin(13, FEMUR\_4);  /\* FEMUR\_4\*//\* CN1  \*//\* PIN 13\*/

  ledcAttachPin(23, CLAW\_1);  /\* CLAW\_1 \*//\* CN16 \*//\* PIN 23\*/

  ledcAttachPin( 4, CLAW\_2);  /\* CLAW\_2 \*//\* CN10 \*//\* PIN  4\*/

  ledcAttachPin(32, CLAW\_3);  /\* CLAW\_3 \*//\* CN8  \*//\* PIN 32\*/

  ledcAttachPin(12, CLAW\_4);  /\* CLAW\_4 \*//\* CN2  \*//\* PIN 12\*/

  delay(50);

}

* From the code above we have set up pwm channel 0 to 7 to 50hz frequeny with resolution 12 bit with function ledcSetup
* We have assign GPIO pin to the pwm channel accordingly with function ledcAttachPin.

#### 3.4.1.3 Update duty cycle during runtime

* ESP32 will output pwm signal after we configure the frequenc , resolutoin to the pwm channel and assign the GPIO pin to each pwm channel.
* We can call LEDCWrite to update the duty cycle of the particular pwm channel.
* By updating duty cycle we can control the positoin of servo motor mention chalter 3.4
* Functoin ledcWrite is used to set duty for the LEDC channel.
  + void ledcWrite(uint8\_t chan, uint32\_t duty);
    - chan select the LEDC channel for writing duty.
    - duty select duty to be set for selected channel.
* In the example code, we have set the resolution bit to 12 bit hence there are 4095 steps for the reoslution.
* By calculation we set 409 to acheive 1ms duty cycle and 819 to achieve 2ms duty cycle.
* However the example we set min to 50 min and maximum to 550 due to base on testing the servo motor only react between 50 and 550 ( will further investigate on this issue suspect is due to servo motor but yet to confirm with scope ).
* For the servo postion array such as Servo\_Prg\_X, the position is store as position therefore a positoin convert to duty cycle is needed.
* Function Set\_PWM\_to\_Servo is to convert the position to duty cycle and update to the pwm channel
  + void Set\_PWM\_to\_Servo(int iServo, int iValue)
    - iServo select the LEDC channel for writing duty.
    - Ivalue select the position to convert to duty tobe set for selected channel.

#### 3.4.1.4 Set\_PWM\_to\_Servo

/\* MOTOR CODE START \*/

void Set\_PWM\_to\_Servo(int iServo, int iValue)

{

  Serial.print(F("iServo: "));

  Serial.print(iServo);

  Serial.print(F(" iValue: "));

  Serial.println(iValue);

  // 讀取 EEPROM 修正誤差

  iValue = (iValue\*MAX/180.0)+MIN; /\* convertion to pwm value \*/

  double NewPWM = iValue + preferences.getDouble((String(iServo)).c\_str(),0); /\* zero error calibration \*/

  /\* 0 = zero degree 550 = 180 degree\*/

  ledcWrite(iServo,NewPWM);

}

* We have printed the input parameter iServo and iValue for debug purpose.
* We have do conversion for iValue from position to duty cycle
* We have done the zero error calibration but currently not in use the value will always be zero.
* We will udpate the pwm channel value with ledcWrite.