

Qumcum R321J Main Board Product Manual

バージョン	1.7
作 成 日	2022 年 2 月 4 日
最終更新日	2024 年 4 月 9 日

Qumcum サイト <https://qumcum.com/product-qx-001r321/>

目次

Qumcum R321J Main Board Product Manual	1
1 はじめに.....	4
1.1 本書で必要となる知識と想定する読者	4
1.2 本書の構成	4
2 注意事項.....	5
2.1 安全に関する注意事項	5
2.2 取扱い上の注意事項	6
2.3 無線モジュールの安全規制について	6
2.4 保証について	6
3 製品概要.....	7
3.1 製品の特長	7
3.2 仕様.....	8
3.3 各部の名称と説明	9
3.4 ブロック図	10
4 ご使用の前に	11
4.1 準備するもの	11
Qumcum 電源投入および USB ケーブル接続の準備	12
5 寸法図	15
インターフェース仕様	16
5.1 RGB-LED 出力用インターフェース	16
5.2 BEEP 音出力用インターフェース	17
5.3 VOICE 出力用インターフェース	18
5.4 Qumcum デジタルマイク入力用インターフェース	20
5.5 超音波センサ入力用インターフェース	21
5.6 サーボモータドライバ通信用インターフェース	22
6 付録.....	26
6.1 各デバイスとの接続について	26
サーボモータ用 5V 電源の接続	26
超音波センサの接続	27
スピーカの接続	27
サーボモータの接続	28
Qumcum デジタルマイクの接続	29
6.2 拡張用ポートの接続	30
I2C(3.3V/5V)ポートの接続	30

ICSP ポートの接続	31
6.3 サーボモータコマンド一覧.....	32
6.4 WiFi 接続サンプル	34
6.5 Bluetooth 接続サンプル.....	36
6.6 Qumcum R321J メインボード ESP32-WROOM-32D ピンアサイン	38
6.7 オプション品	39

1 はじめに

このたびは Qumcum R321J メインボード をご利用いただき、ありがとうございます。

Qumcum R321J メインボードは、クムクムロボットのメイン CPU およびデバイスインターフェース機能を有するボードでマイク、超音波センサなどのセンサおよび RGB-LED、BEEP、音声合成ライブラリ(別売り)による発話、サーボモータドライバ LSI を用いたモータ制御が可能です。

当ボードを使用することで、クムクムロボットによるプログラミング学習が可能となります。

本書では、Qumcum R321J メインボードの説明と基本的な仕様について記載します。

1.1 本書で必要となる知識と想定する読者

Espressif 社製 WiFi+Bluetooth/Bluetooth LE モジュール ESP32-WROOM-32 シリーズを用いた開発の基本的な知識

Arduino IDE による基本的なプログラミング知識

ESP32-WROOM-32 による IoT デバイス開発に興味をお持ちの開発者・技術者

1.2 本書の構成

はじめにお読みください

「はじめに」、「注意事項」

Qumcum R321J メインボードの仕様を紹介します

「製品概要」

Qumcum R321J メインボードのインターフェースの仕様を紹介します

「インターフェース仕様」

2 注意事項

2.1 安全に関する注意事項

本製品を安全にご使用いただくために、特に以下の点にご注意ください。



ご使用の前に必ず製品マニュアルおよび関連資料をお読みにになり、使用上の注意を守って正しく安全にお使いください。

マニュアルに記載されていない操作・拡張などを行う場合は、弊社 Web サイトに掲載されている資料やその他技術情報を十分に理解した上で、お客様自身の責任で安全にお使いください。

水・湿気・ほこり・油煙等の多い場所に設置しないでください。火災、故障、感電などの原因になる場合があります。

本製品に搭載されている部品の一部は、発熱により高温になる場合があります。周囲温度や取扱いによってはやけどの原因となる恐れがあります。本体の電源が入っている間、または電源切断後本体の温度が下がるまでの間は、基板上の電子部品、及びその周辺部分には触れないでください。

本製品を使用して、お客様の仕様による機器・システムを開発される場合は、製品マニュアルおよび関連資料、弊社 Web サイトで提供している技術情報のほか、関連するデバイスのデータシート等を熟読し、十分に理解した上で設計・開発を行ってください。また、信頼性および安全性を確保・維持するため、事前に十分な試験を実施してください。

本製品は、機能・精度において極めて高い信頼性・安全性が必要とされる用途(医療機器、交通関連機器、燃焼制御、安全装置等)での使用を意図しておりません。これらの設備や機器またはシステム等に使用された場合において、人身事故、火災、損害等が発生した場合、当社はいかなる責任も負いかねます。

無線 LAN 機能を搭載した製品は、心臓ペースメーカーや補聴器などの医療機器、火災報知器や自動ドアなどの自動制御器、電子レンジ、高度な電子機器やテレビ・ラジオに近接する場所、移動体識別用の構内無線局および特定小電力無線局の近くで使用しないでください。製品が発生する電波によりこれらの機器の誤作動を招く恐れがあります。

2.2 取扱い上の注意事項

本製品に改造を行った場合は保証対象外となりますので十分ご注意ください。また、改造やコネクタ等の増設を行う場合は、作業前に必ず動作確認を行ってください。

本製品や周辺回路に電源が入っている状態で、コネクタ着脱は、絶対に行わないでください。

本製品は、静電気により破壊されるおそれがあります。本製品を開封するときは、低湿度状態にならないよう注意し、静電防止用マットの使用、導電靴や人体アースなどによる作業者の帯電防止対策、備品の放電対策、静電気対策を施された環境下で行ってください。また、本製品を保管する際は、静電気を帯びやすいビニール袋やプラスチック容器などは避け、導電袋や導電性の容器・ラックなどに収納してください。

落下や衝撃などの強い振動を与えないでください。

無線機能を搭載した製品は、テレビ・ラジオに近接する場所で使用すると、受信障害を招く恐れがあります。この無線機は通信を行います。通信機能は、心臓ペースメーカーや除細動器等の植込み型医療機器の近く(15cm 程度以内)で使用しないでください。

2.3 無線モジュールの安全規制について

本製品に搭載されている 無線モジュール ESP32-WROOM-32 は、電気通信事業法に基づく設計認証と電波法に基づく工事設計認証を受けています。

これらの無線モジュールを国内で使用するときには無線局の免許は必要ありません。



以下の事項を行うと法律により罰せられることがあります。

- ・ 無線モジュールやアンテナを分解/改造すること。
- ・ 無線モジュールや筐体、基板等に直接印刷されている証明マーク・証明番号、または貼られている証明ラベルをはがす、消す、上からラベルを貼るなどし、見えない状態にすること。

2.4 保証について

本製品は、添付品およびソフトウェアは保証対象外となりますのでご注意ください。

WiFi+Bluetooth/Bluetooth LE モジュール ESP32-WROOM-32 適合証明情報

項目	内容
型式または名称	ESP32-WROOM-32D
電波法に基づく工事設計認証における認証番号	211-171102

3 製品概要

3.1 製品の特長

Qumcum R321J メインボードは、Espressif 社製 ESP32-WROOM-32 を搭載したロボット制御に特化した基板です。

コンパクト((W)69mm×(H)61mm)な基板に Wi-Fi + Bluetooth/Bluetooth LE モジュール、RGB-LED、アンプ回路(ボリューム付き)、サーボモータ(18 軸)制御用 LSI、I2C(3.3V/5V)・SPI インターフェース、シリアル・UART 変換回路を搭載しています。

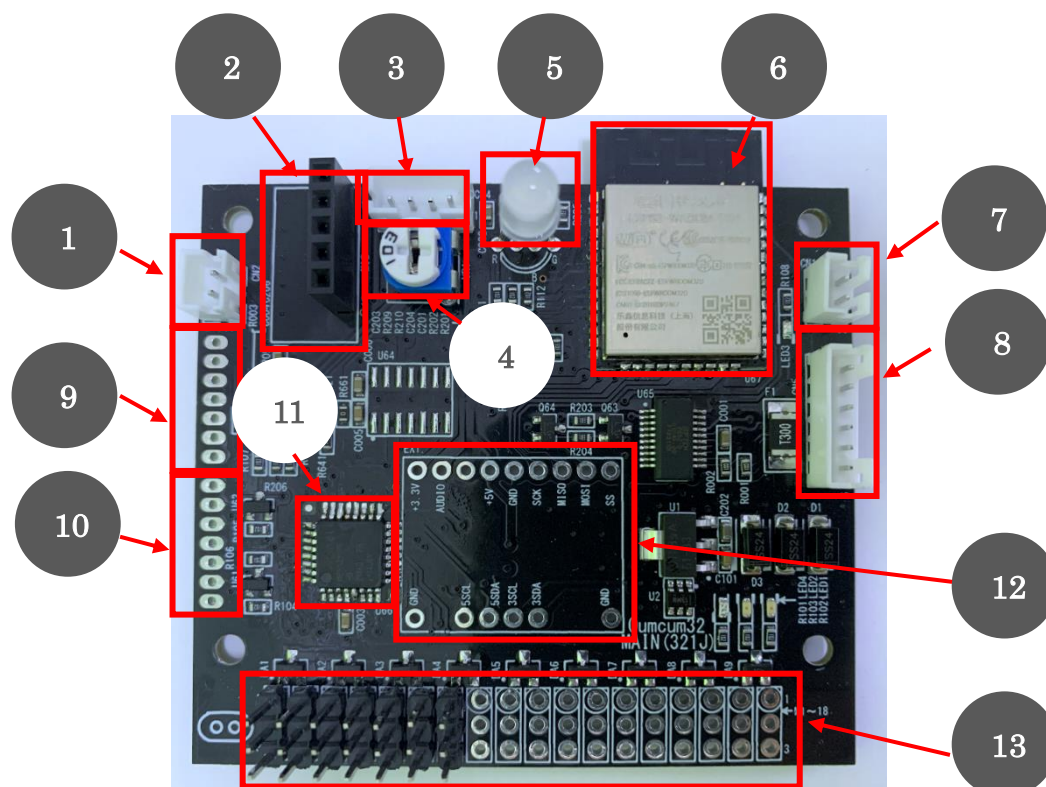
別売りの Qumcum スイッチボード(USB 2.0 Micro-B ポートを搭載)と接続するだけで PC と USB 接続し、簡単に Arduino IDE によるプログラムの書込み等が可能となります。

3.2 仕様

Qumcum R321J メインボードの主な仕様は次のとおりです。

型番	R321J
プロセッサ	ESP32-WROOM-32D(Xtensa 32 bit LX6 MCU デュアルコア)
SRAM	520 KB
Flash	4 MB
外部接続用コネクタ	電源供給用 2P コネクタ スイッチボード接続用 6P コネクタ 超音波センサ用 4P コネクタ 外部スピーカ用 2P コネクタ
オプション接続用ポート	サーボモータ接続用ポート デジタルマイク接続用ポート I2C(3.3V)接続用ポート I2C(5.0V)接続用ポート ISP 接続用ポート
拡張ボードエリア	OPL3 FM シンセサイザーボード(予定)
無線通信機能	PCB アンテナ搭載 Wi-Fi:802.11 b/g/n(802.11 n は 150 Mbps まで) Bluetooth:Classic、BLE 4.2(デュアルモード)
外径サイズ	外径寸法：(W)69mm×(H)67mm 基板本体：(W)69mm×(H)61mm
本体 動作電圧	5.0 V
サーボモータ電源電圧	5.0 V
CPU 動作電圧	2.2～3.6 V
CPU 消費電流	平均 80 mA
CPU 動作温度	-40℃～+85℃

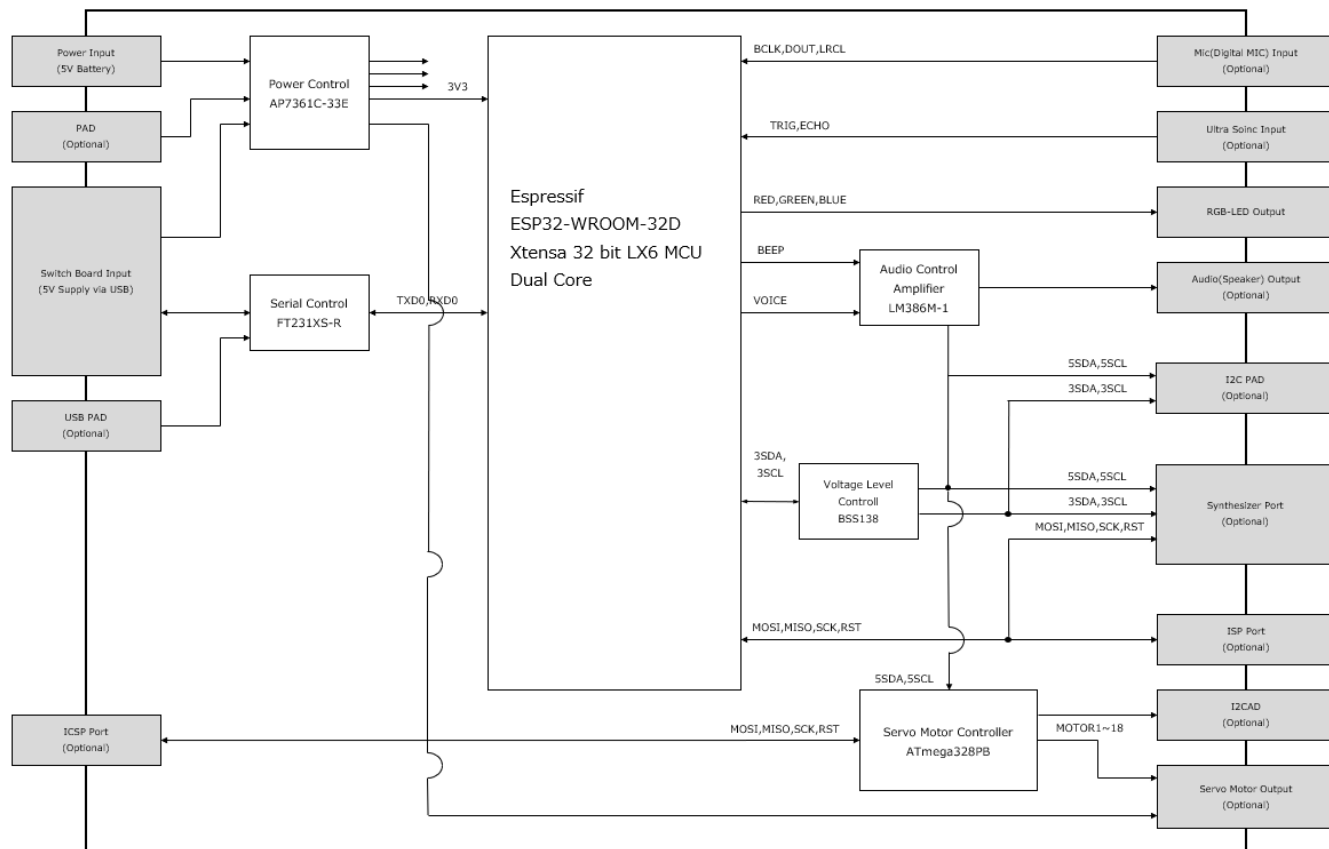
3.3 各部の名称と説明



番号	名称	説明
1	スピーカ用コネクタ	スピーカと接続を行うためのコネクタです
2	デジタルマイク接続用ポート	Qumcum デジタルマイクと接続を行うためのコネクタです
3	超音波センサ接続用コネクタ	超音波センサと接続を行うためのコネクタです
4	音量調整用ボリューム	スピーカーから音を出したり発話をする時の音量を調整するためのボリュームです
5	RGB-LED	RGB-LED です
6	CPU および無線モジュール	メイン CPU および無線通信モジュール(ESP32-WROOM-32D)です
7	電源コネクタ	電源（5.0V）と接続を行うためのコネクタです
8	スイッチボード接続用コネクタ	USB や電源スイッチ接続コネクタです。Qumcum スwitchボードなら簡単に接続できます
9	ISP 接続用ポート	外部デバイスと ISP で接続を行うためのポートです
10	外部 I2C デバイス接続用ポート	外部の I2C デバイス(3.3V/5.0V)と接続を行うためのポートです
11	サーボモータ制御用 LSI	サーボモータを制御するための専用 LSI です
12	拡張ボード取付用エリア	OPL3 FM シンセサイザーボード取り付けるためのポートです (予定)
13	サーボモータ接続用ポート	サーボモータを接続するためのポートです 18 軸まで接続することが可能です

3.4 ブロック図

Qumcum R321J メインボード のブロック図は次のとおりです。



4 ご使用の前に

Qumcum R321J メインボードを使用する前に、次のものを必要に応じて準備してください。

4.1 準備するもの

- ・ Qumcum R321J メインボード
- ・ 作業用 PC
- ・ Qumcum スイッチボード(別売り)
- ・ USB ケーブル(TypeA <--> micro TypeB)
- ・ 電源(サーボモータを使用する場合は必須。Qumcum 用本体電池ボックスを推奨)

【オプション】

- ・ Qumcum デジタルマイク(デジタルマイクによる音検知を使用する場合)
- ・ Qumcum 超音波センサ(超音波センサによる距離計測を使用する場合)
- ・ サーボモータ(サーボモータを使用する場合)
- ・ Qumcum スピーカ(BEEP、発話を使用する場合)

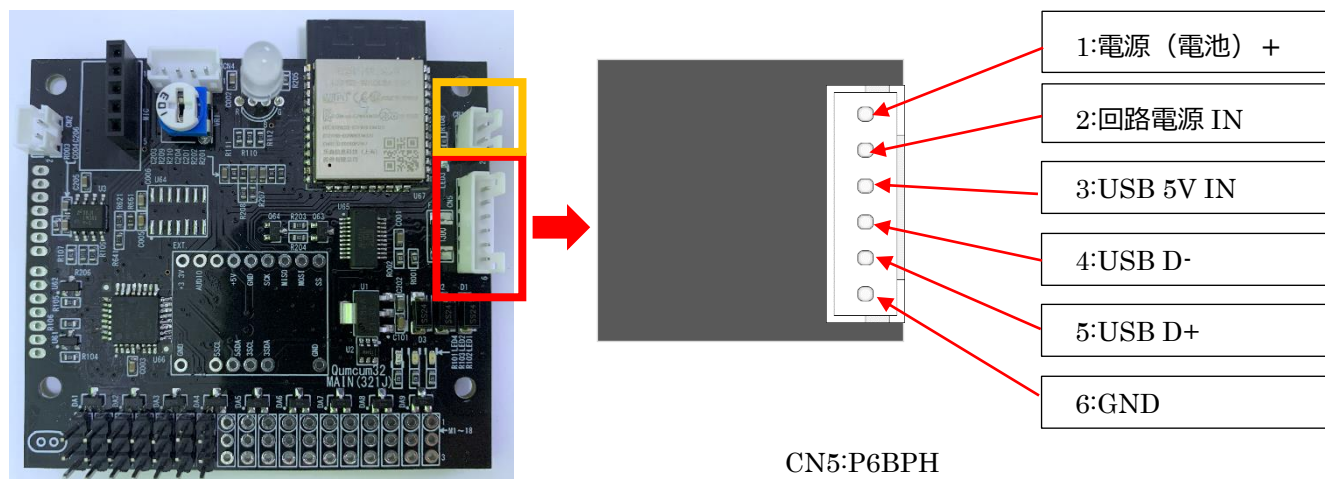


Qumcum はサーボモータを使っているため、電源（5V）が必要です。この電源でその他の電子回路全体にも電力が供給されます。

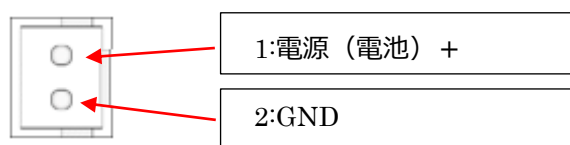
ただし USB ケーブルを接続している時には、USB からの電力が供給されるため、サーボモータ以外の電子回路に電力が供給されます。

Qumcum 電源投入および USB ケーブル接続の準備

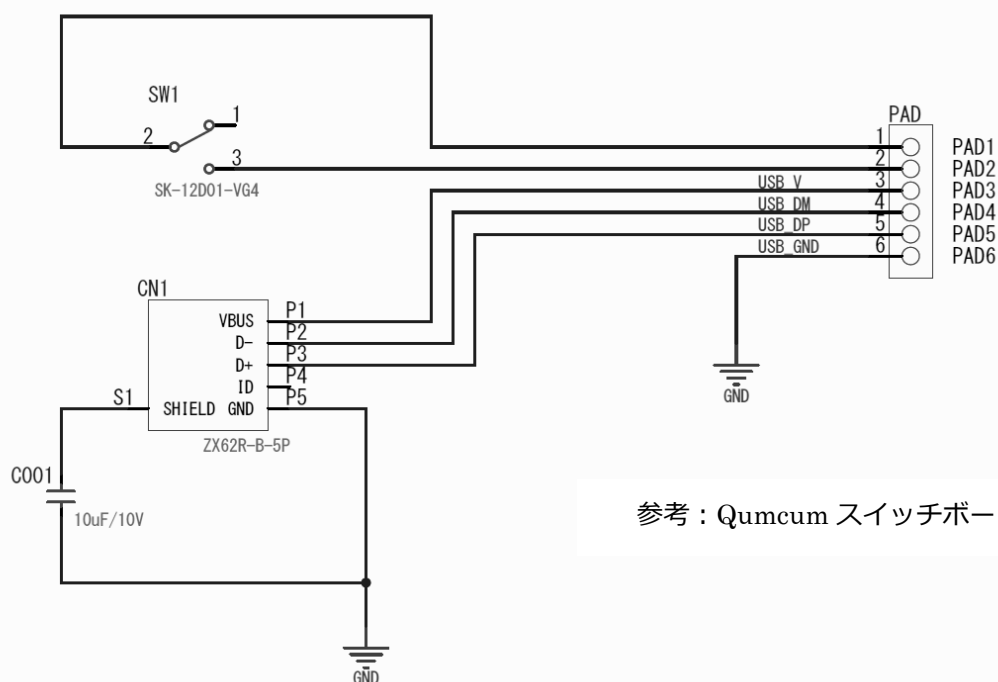
Qumcum R321J メインボードの電源を投入するには、Qumcum R321J メインボードの右側にある 6 ピンのコネクタに配線を行うことで USB から電源とモーター用外部 5V 電源を供給することが可能です。



電源コネクタ



Qumcum スイッチボードを使うことで以下の接続ができ、電源（5V）のスイッチと USB コネクタが使用できます。両コネクタ 1 番ピンは基板内部でつながっています。2 番ピン回路電源 IN から電子回路全体およびサーボモータへ電力が供給されます。3 番ピン USB5VIN からサーボモータへの電力は供給されません。

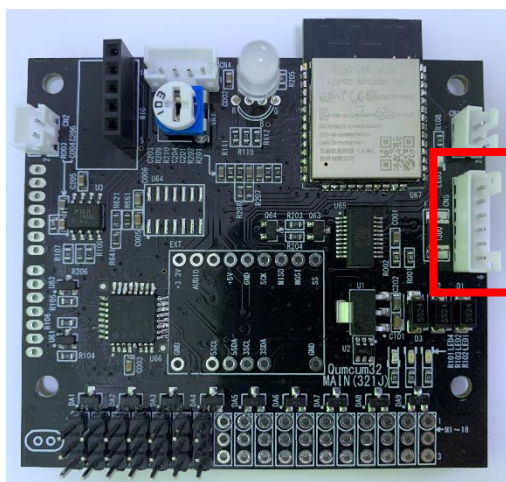


参考：Qumcum スイッチボード回路図

参考例：Qumcum スイッチボード(別売り)を使用する場合の接続方法

1. メインボードとスイッチボードの接続

Qumcum R321J メインボードに Qumcum スイッチボードを接続します



スイッチボード接続用
コネクタ(メス)



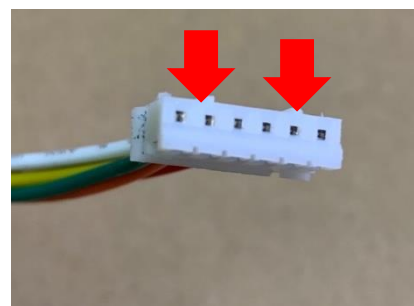
Qumcum
スイッチボード(別売り)

スイッチボードのコネクタにある 2 つの出っ張りが、メインボードのコネクタにある切り欠き部分に合うように差し込みます



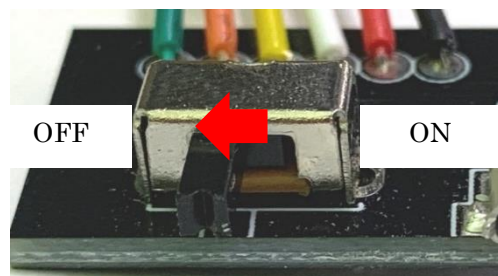
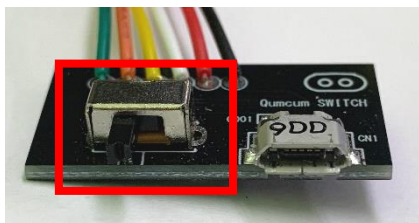
切り欠き部分

2 つの出っ張り



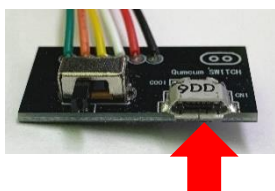
2. スイッチボードのスイッチの確認

スイッチボードのスイッチが OFF 側にあることを確認します



3. スイッチボードと USB ケーブルの接続

スイッチボードと USB ケーブルを接続します



USB コネクタ
micro TypeB(メス)



USB コネクタ
micro TypeB(オス)



スイッチボードと
USB ケーブルを接続した状態

4. PC との接続

USB ケーブルと PC を接続します

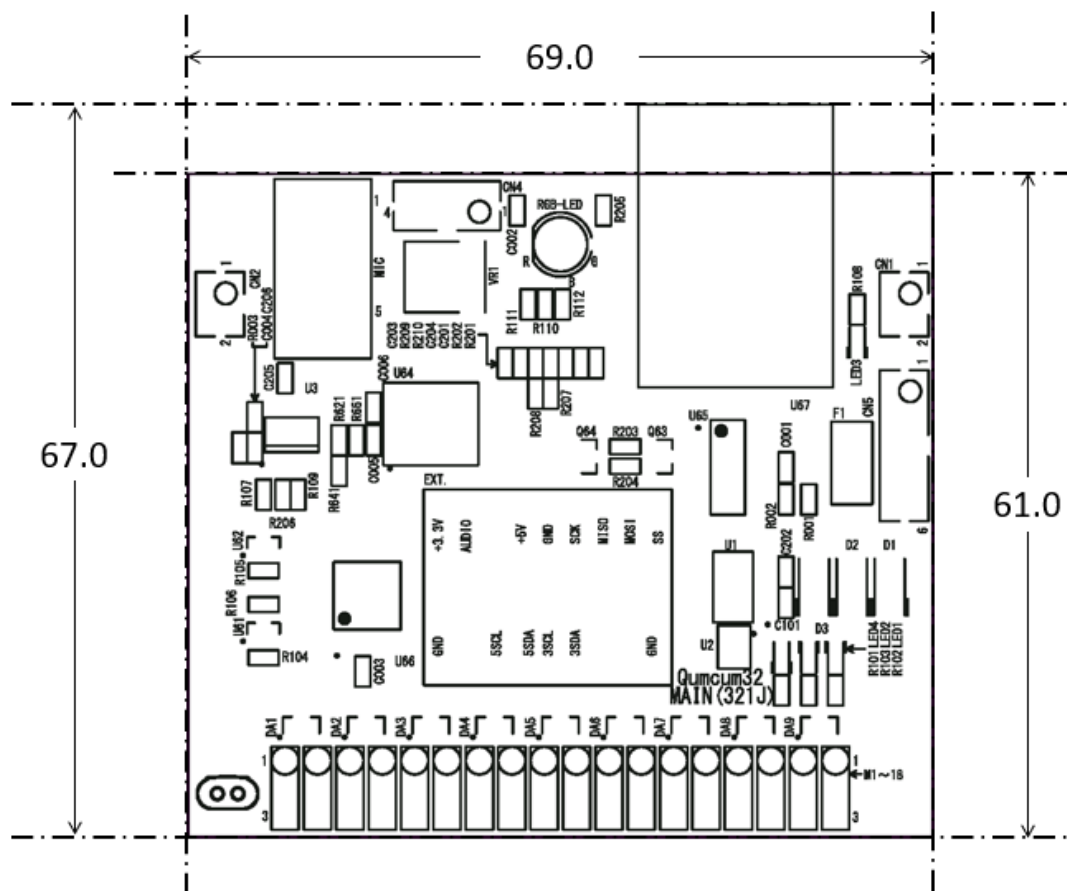


モーターが突然動作する可能性がありますので、必ずスイッチボードのスイッチが OFF になっていることを確認してください。USB ケーブルを PC に接続すると Qumcum R321J メインボードに電源が投入され、Qumcum R321J メインボードにプログラムが書かれている場合はプログラムが起動します。



USB コネクタ TypeA(オス)を PC の
USB ポート TypeA(メス)に接続します

5 寸法図



インターフェース仕様

各デバイスへの Arduino IDE によるアクセス例を記載します。

Arduino IDE で ESP32-WROOM-32 用の開発環境構築・プログラムのビルド方法については、下記サイトなどを参考にして別途ご確認ください。

参考 URL : [Arduino core for the ESP32 のインストール](#)

5.1 RGB-LED 出力用インターフェース

RGB-LED は下記のようにアサインされています

対象	IO
RGB-LED(赤)	IO32
RGB-LED(緑)	IO27
RGB-LED(青)	IO33

サンプルソース

```
void setup() {
  pinMode(32, OUTPUT); // RGB-LED (Red)
  pinMode(27, OUTPUT); // RGB-LED (Green)
  pinMode(33, OUTPUT); // RGB-LED (Blue)
}

void loop() {
  // RGB-LED (Red) ON
  digitalWrite(32, HIGH);
  // Wait 200msec
  delay(200);
  // RGB-LED (Red) OFF
  digitalWrite(32, LOW);
  // Wait 200msec
  delay(200);

  // RGB-LED (Green) ON
  digitalWrite(27, HIGH);
  // Wait 200msec
  delay(200);
  // RGB-LED (Green) OFF
  digitalWrite(27, LOW);
  // Wait 200msec
  delay(200);

  // RGB-LED (Blue) ON
  digitalWrite(33, HIGH);
  // Wait 200msec
  delay(200);
  // RGB-LED (Blue) OFF
```



```
digitalWrite(33, LOW);  
// Wait 200msec  
delay(200);  
}
```

5.2 BEEP 音出力用インターフェース

BEEP は下記のようにアサインされています

BEEP 音を確認するにはスピーカを接続しておく必要があります。

スピーカの接続方法は「[スピーカの接続](#)」を参照してください

対象	IO
BEEP	IO4

サンプルソース

```
void setup() {  
  // チャンネル 0、キャリア周波数 12kHz、8 ビットレンジ  
  ledcSetup(0, 12000, 8);  
  // BEEP を出力するピンを設定する  
  ledcAttachPin(4, 0);  
}  
  
void loop() {  
  // 440Hz の音を 100msec 出力する  
  ledcWriteTone(0, 440);  
  delay(62);  
  // 880Hz の音を 100msec 出力する  
  ledcWriteTone(0, 880);  
  delay(62);  
  // 音を止める(消音)  
  ledcWriteTone(0, 0);  
  delay(1000);  
}
```

5.3 VOICE 出力用インターフェース

VOICE は下記のようにアサインされています

VOICE 音を確認するにはスピーカを接続しておく必要があります。

スピーカの接続方法は「[スピーカの接続](#)」を参照してください

対象	IO
VOICE	IO25

VOICE の音声合成には Aquestalk ライブラリを使用しています

Aquestalk ライブラリの入手およびセットアップ方法については下記をご参照ください

参考 URL: [ESP32 で音声合成\(AquesTalk pico for ESP32\)](#)

サンプルソース

```
#include "driver/i2s.h"
#include "aquestalk.h"

uint32_t workbuf[AQ_SIZE_WORKBUF];

const int i2s_num = 0; // i2s port number

i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_TX | I2S_MODE_DAC_BUILT_IN),
    .sample_rate = 24000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = (i2s_comm_format_t) I2S_COMM_FORMAT_STAND_MSB,
    .intr_alloc_flags = 0,
    .dma_buf_count = 4,
    .dma_buf_len = 384,
    .use_apll = 0,
    .tx_desc_auto_clear = false,
    .fixed_mclk = 0
};

void setup() {
    CAqTkPicoF_Init(workbuf, 32, "XXXX-XXXX-XXXX-XXXX");
    AqResample_Reset();

    // DAC Setup
    i2s_driver_install((i2s_port_t)i2s_num, &i2s_config, 0, NULL);
    i2s_set_clk((i2s_port_t)i2s_num, i2s_config.sample_rate, I2S_BITS_PER_SAMPLE_16BIT, I2S_CHANNEL_MONO);
    i2s_set_dac_mode(I2S_DAC_CHANNEL_RIGHT_EN); // R だけの出力にする
}

void loop() {
    int iret = CAqTkPicoF_SetKoe((const uint8_t*)"ohayo--", 120, 0xffffU); //Set speaking speed 120
```

```
for(;;) {
    int16_t wav[32];
    uint16_t len;
    iret = CAqTkPicoF_SyntheFrame(wav, &len);
    if(iret) {
        // I2S の最終バッファをクリアさせる
        vTaskDelay(100);
        //i2s_zero_dma_buffer((i2s_port_t) i2s_num);
        break;
    }

    int i;
    size_t transBytes = 0;
    uint16_t sample[2];
    for(i = 0; i < len; i++) {
        esp_err_t ret = ESP_OK;
        int16_t wav3[3];
        AqResample_Conv(wav[i], wav3);
        // I2S の DMA バッファへ書き込み
        for(int k = 0; k < 3; k++) {
            sample[0] = sample[1] = ((uint16_t)wav3[k])^0x8000U; // モノラルをステレオデータに変換(単純多重化)

            ret = i2s_write((i2s_port_t) i2s_num, (const char*)sample, sizeof(uint16_t)*1, &transBytes, portMAX_DELAY);
            if(ret != ESP_OK) {
                // エラー
                break;
            }
            if(transBytes < sizeof(uint16_t) * 1) {
                // タイムアウト
                ret = ESP_FAIL;
                break;
            }
        }
        if(ret != ESP_OK) {
            break;
        }
    }
}

delay(500);
}
```

5.4 Qumcum デジタルマイク入力用インターフェース

Qumcum デジタルマイク(別売り)用インターフェースは下記のようにアサインされています

Qumcum デジタルマイクを使用するには Qumcum デジタルマイクを接続しておく必要があります。

Qumcum デジタルマイクの接続方法は「[Qumcum デジタルマイクの接続](#)」を参照してください

対象	IO
マイク入力(マイクの DOUT)	IO26
ベースクロック(マイクの BCLK)	IO16
LR クロック(マイクの LRCL)	IO17

Qumcum デジタルマイクは I2S を使ってデータを取得します

サンプルソース

```
#include "driver/i2s.h"

const i2s_port_t I2S_PORT_MIC = I2S_NUM_1;

const i2s_config_t i2s_config_mic = {
    .mode = i2s_mode_t(I2S_MODE_MASTER|I2S_MODE_RX),
    .sample_rate = 16000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = 0,
    .dma_buf_count = 4,
    .dma_buf_len = 8
};

const i2s_pin_config_t pin_config_mic = {
    .bck_io_num = 16, // BCLK
    .ws_io_num = 17, // LRCL
    .data_out_num = I2S_PIN_NO_CHANGE,
    .data_in_num = 26 // DOUT
};

void setup() {
    Serial.begin(115200);

    esp_err_t err;
    err = i2s_driver_install(I2S_PORT_MIC, &i2s_config_mic, 0, NULL);
    if (err != ESP_OK) {
        return;
    }
    err = i2s_set_pin(I2S_PORT_MIC, &pin_config_mic);
    if (err != ESP_OK) {
        return;
    }
}
```

```

void loop() {
  int32_t samples[2];    // サンプリングデータ 32bit x 2 (R&L)
  size_t bytes_read;    // 受信データ長

  esp_err_t err = i2s_read(I2S_PORT_MIC, (char *)samples, 8, &bytes_read, portMAX_DELAY);
  if ((err == ESP_OK) && (bytes_read == 8)) {
    // 24 ビットマイクの I2S 32 ビット長 PCM データを int16_t 変数に収まるようビットシフト
    int16_t mic_value = samples[0]>>14;
    Serial.println(mic_value);
  }
  delay(20);
}

```

5.5 超音波センサ入力用インターフェース

超音波センサ(別売り)用インターフェースは下記のようにアサインされています

超音波センサを使用するには超音波センサを接続しておく必要があります。

超音波センサの接続方法は「[超音波センサの接続](#)」を参照してください

対象	IO
トリガ	IO5
エコー	IO35

超音波センサはトリガとエコーを使ってデータを取得します

サンプルソース

```

double Duration = 0; // 受信した間隔
double Distance = 0; // 距離

void setup() {
  Serial.begin(115200);
  // ピンモードを設定する
  pinMode(35, INPUT); // エコー
  pinMode(5, OUTPUT); // トリガ
}

void loop() {
  // 超音波センサの計測準備
  digitalWrite(5, LOW);
  delayMicroseconds(2);
  // 超音波センサの計測開始
  digitalWrite(5, HIGH);
  delayMicroseconds(10);
  // 超音波センサの計測終了

```

```
digitalWrite(5, LOW);

// エコーの HIGH だった時間を計測する
Duration = pulseIn(35, HIGH);
if(Duration > 0) {
    Duration = Duration / 2; // 往復距離を半分にする
    Distance = Duration * 340 * 100 / 1000000; // 音速を 340m/s に設定
    Serial.println(Distance);
}
delay(500);
}
```

5.6 サーボモータドライバ通信インターフェース

サーボモータドライバとは I2C による通信を行っています。

サーボモータを使用するにはサーボモータ(別売り)およびサーボモータ用電源(別売り)が必要です。

Qumcum R321J メインボードにサーボモータを接続しておく必要があります。

サーボモータ用電源の接続方法は「[サーボモータ用 5V 電源の接続](#)」を参照してください

サーボモータの接続方法は「[サーボモータの接続](#)」を参照してください

サーボモータ駆動コマンド詳細は「[6.3 コマンド一覧](#)」を参照してください。

サーボモータドライバ I2C アドレス	8
---------------------	---

サンプルソース

```
#include <Wire.h>

#define SERVO_IC_I2C_ADDR 8

void send_servo_driver_cmd(const uint8_t* tx, int size)
{
    Wire.beginTransmission(SERVO_IC_I2C_ADDR);
    while(size-- > 0){
        Wire.write(*tx++);
    }
    Wire.endTransmission();
}

uint16_t recv_servo_driver_reply(uint8_t* rx, int size)
{
    uint16_t n=0;
    Wire.requestFrom(SERVO_IC_I2C_ADDR, size); // request len bytes from Slave ID #addr
    while (Wire.available()){
        *rx++ = Wire.read();
        n++;
    }
    return n;
}
```

```

}

void setup() {
    Serial.begin(115200);

    // I2C のクロックを 100,000Hz に設定
    Wire.setClock(100000);
    Wire.begin();

    uint8_t cmd[10];    // 送信電文格納バッファ
    uint8_t reply[10];  // 応答電文格納バッファ

    // サーボモータ動作モード設定
    cmd[0] = 0;    // cmdno
    cmd[1] = 1;    // sw
    // コマンドを送信する
    send_servo_driver_cmd(cmd, 2);

    // コマンドバッファはクリアしておく
    memset(cmd, 0x00, sizeof(cmd));

    // PWM パルス出力許可
    cmd[0] = 2;    // cmdno
    cmd[1] = 3;    // bOut
    *(uint16_t*)&(cmd[2]) = 1;
    // コマンドを送信する
    send_servo_driver_cmd(cmd, 4);

    delay(100);
}

void loop() {
    uint8_t cmd[10];    // 送信電文格納バッファ
    uint8_t reply[10];  // 応答電文格納バッファ
    uint16_t AngleHD;   // 取得した角度(1=1/10 度)

    // コマンドバッファはクリアしておく
    memset(cmd, 0x00, sizeof(cmd));

    // モータの角度を 0 度にセット
    cmd[0] = 4;    // cmdno
    cmd[1] = 3;    // ch
    *(uint16_t*)&(cmd[2]) = 0;    // AngleHD 角度(1=1/10 度)
    *(uint16_t*)&(cmd[4]) = 1000; // mTime 動作時間(msec)
    // コマンドを送信する
    send_servo_driver_cmd(cmd, 6);

    // コマンドバッファはクリアしておく
    memset(cmd, 0x00, sizeof(cmd));

    // モーター斉スタート
    cmd[0] = 1;    // cmdno
    // コマンドを送信する
    send_servo_driver_cmd(cmd, 1);

    delay(1000);

    // コマンドバッファはクリアしておく
    memset(cmd, 0x00, sizeof(cmd));
}

```

```
// モータの現在座標を取得する
cmd[0] = 6;    // cmdno
cmd[1] = 3;    // ch
// コマンドを送信する
send_servo_driver_cmd(cmd, 2);
// 応答を受信する
recv_servo_driver_reply(reply, 2);
AngleHD = *(uint16_t*)reply;
Serial.print("Angle: ");
Serial.println(AngleHD);

// コマンドバッファはクリアしておく
memset(cmd, 0x00, sizeof(cmd));

// モータの角度を 180 度にセット
cmd[0] = 4;    // cmdno
cmd[1] = 3;    // ch
*(uint16_t*)(&(cmd[2])) = 1800;    // AngleHD 角度(1=1/10 度)
*(uint16_t*)(&(cmd[4])) = 1000;    // mTime 動作時間(msec)
// コマンドを送信する
send_servo_driver_cmd(cmd, 6);

// コマンドバッファはクリアしておく
memset(cmd, 0x00, sizeof(cmd));

// モーター斉スタート
cmd[0] = 1;    // cmdno
// コマンドを送信する
send_servo_driver_cmd(cmd, 1);

for(int j = 0; j < 10; j++) {
    delay(100);

    // コマンドバッファはクリアしておく
    memset(cmd, 0x00, sizeof(cmd));
    // モータの現在座標を取得する
    cmd[0] = 6;    // cmdno
    cmd[1] = 3;    // ch
    // コマンドを送信する
    send_servo_driver_cmd(cmd, 2);
    // 応答を受信する
    recv_servo_driver_reply(reply, 2);
    AngleHD = *(uint16_t*)reply;
    Serial.print("Angle: ");
    Serial.println(AngleHD);
}
delay(400);

// コマンドバッファはクリアしておく
memset(cmd, 0x00, sizeof(cmd));

// モータの角度を 90 度にセット
cmd[0] = 4;    // cmdno
cmd[1] = 3;    // ch
*(uint16_t*)(&(cmd[2])) = 900;    // AngleHD 角度(1=1/10 度)
*(uint16_t*)(&(cmd[4])) = 1000;    // mTime 動作時間(msec)
// コマンドを送信する
send_servo_driver_cmd(cmd, 6);
```



```
// コマンドバッファはクリアしておく
memset(cmd, 0x00, sizeof(cmd));

// モーター斉スタート
cmd[0] = 1; // cmdno
// コマンドを送信する
send_servo_driver_cmd(cmd, 1);

delay(1000);
delay(100);
}
```



サーボモータを動作させると突入電流が流れます。複数のサーボモータを同時に動作させると電源容量を超えてしまう可能性があります。電源容量を超えると CPU の電源電圧もさがるためリセットがかかり動作が中断されてしまいます。できるだけ動作は同時に行わないようにしてください。最初に「PWM パルス出力許可」したときにも突入電流が流れます。

6 付録

6.1 各デバイスとの接続について

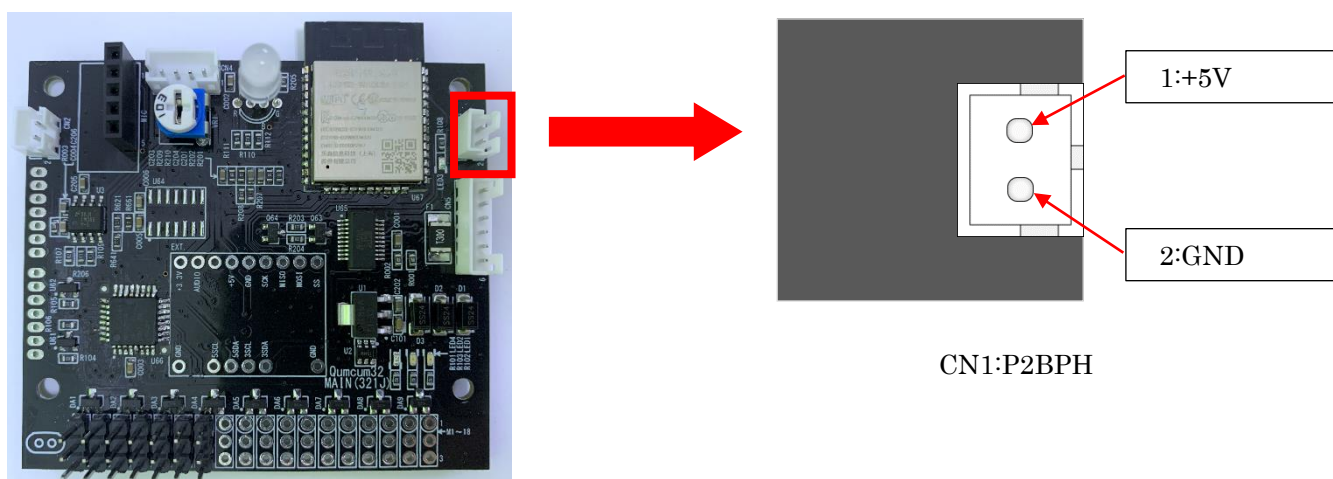
各デバイスと接続を行う場合は、Qumcum R321J メインボードの電源を OFF にした状態で行ってください。
Qumcum R321J メインボードの電源を ON のままで行うと故障の原因となることがあります。

サーボモータ用 5V 電源の接続

サーボモータ用 5V 電源は、サーボモータを接続して動かすときには必ず必要となります。

注意：電池で電源を供給する場合は充電電池(ニッケル水素充電電池 4 本直列)をお使いください。

サーボモータ用 5V 電源接続用のコネクタのピンアサインは下図のようになっています。

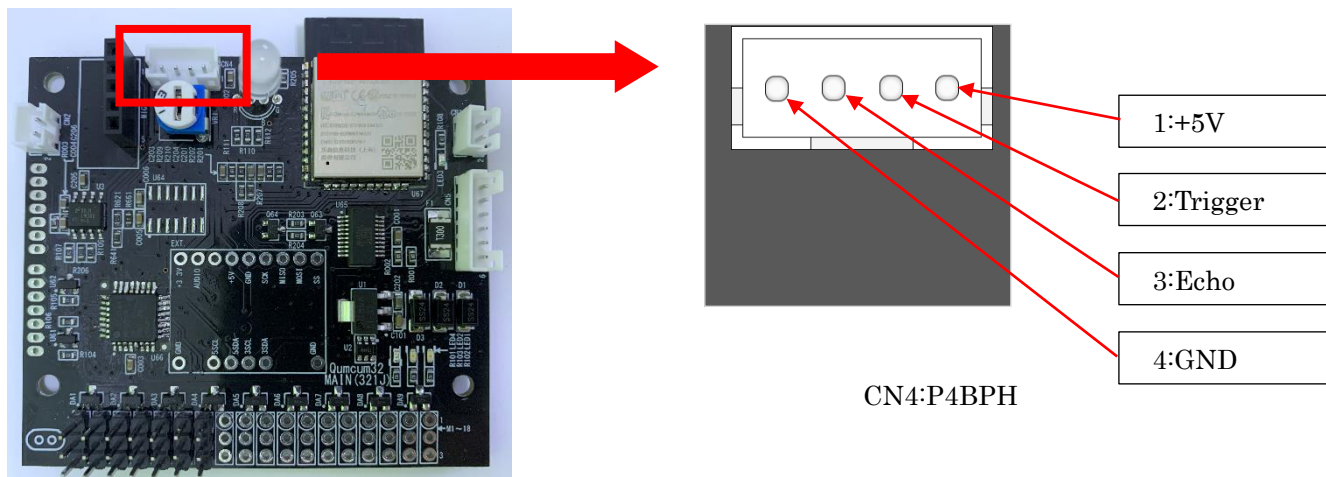


電源には外部スイッチ回路を接続することを想定していますので、このコネクタに電源を接続するだけでは電力が供給されません。

スイッチ回路の詳細は、「[Qumcum 電源投入および USB ケーブル接続の準備](#)」その他ページを参照してください。

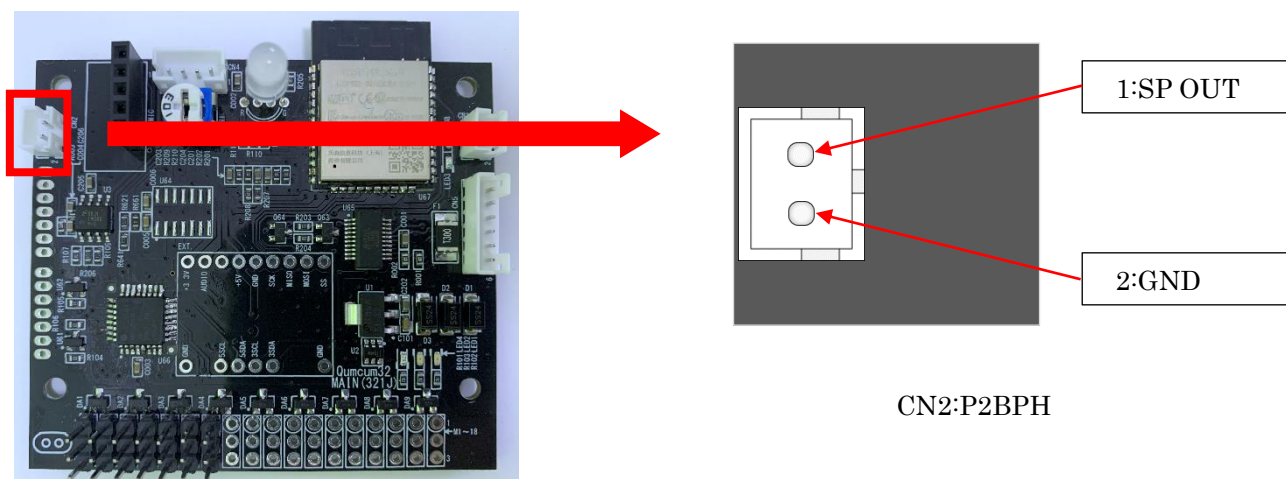
超音波センサの接続

超音波センサを Qumcum R321J メインボードと接続するためのコネクタのピンアサインは下図のようになっています。Qumcum では超音波センサに「HC-SR04」を使用しています。



スピーカの接続

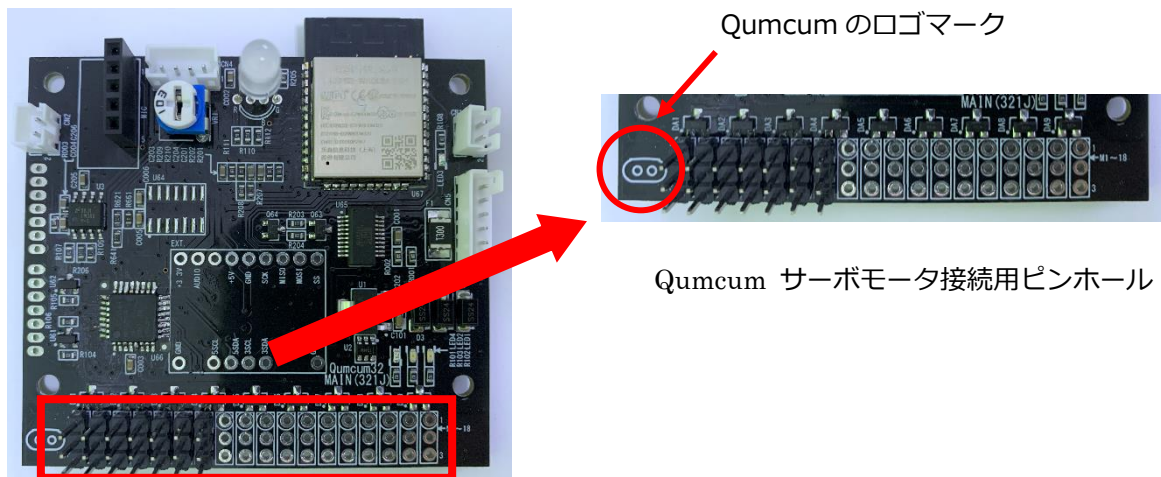
Qumcum R321J メインボードから音や音声合成の出力を行うにはスピーカと接続する必要があります。スピーカ用コネクタのピンアサインは下図のようになっています。



8Ωのスピーカを推奨します。

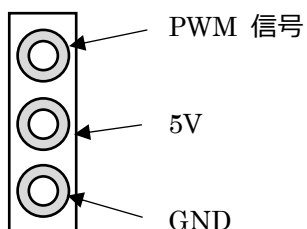
サーボモータの接続

サーボモータを Qumcum R321J メインボードと接続するには、サーボモータのコネクタにある切り欠きと超音波センサの接続用ケーブルにある白いコネクタの出っ張りの向きを合わせて差し込みます



Qumcum サーボモータ接続用ピンホールはボードを上から見て Qumcum のロゴマークがあるほうが SV1 となり順に SV2、SV3…SV18 までとなります。

ピンアサインは下図のようになります。



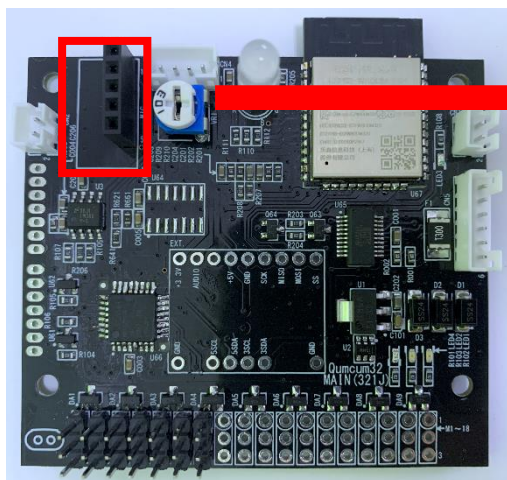
サーボモータ PS-1109MG の場合のモータのピンアサイン

橙 : PWM 信号

サーボモータは SG90 相当品の使用を想定しています。

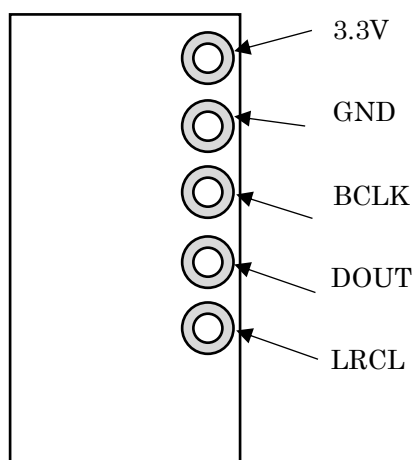
Qumcum デジタルマイクの接続

デジタルマイクを Qumcum R321J メインボードと接続するには、はんだづけを行う必要があります。



デジタルマイク取付部分

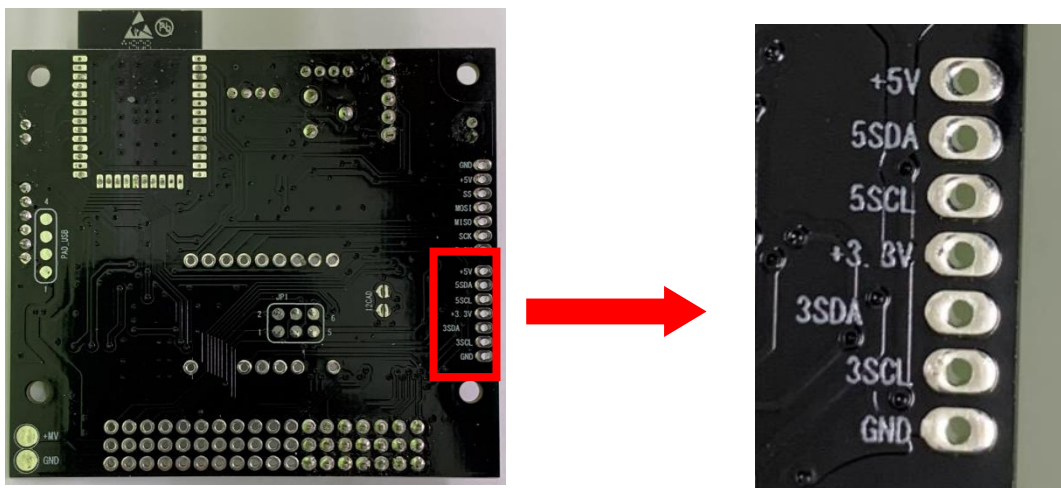
ピンアサインは下図のようになります。



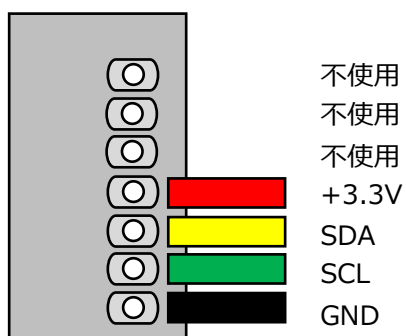
MEMS PCM マイクが接続できます。

6.2 拡張用ポートの接続

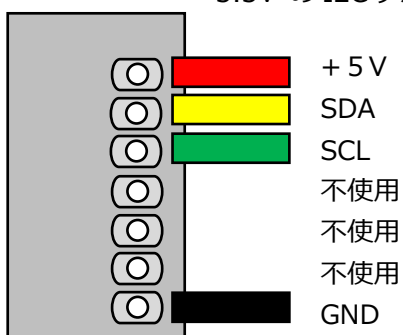
I2C(3.3V/5V)ポートの接続



I2C(3.3V/5V)ポート



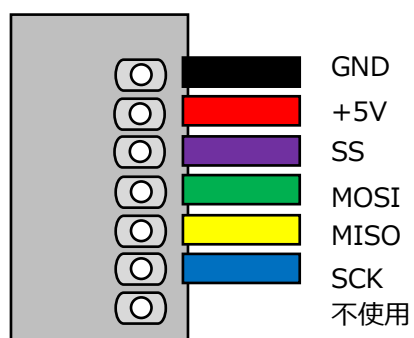
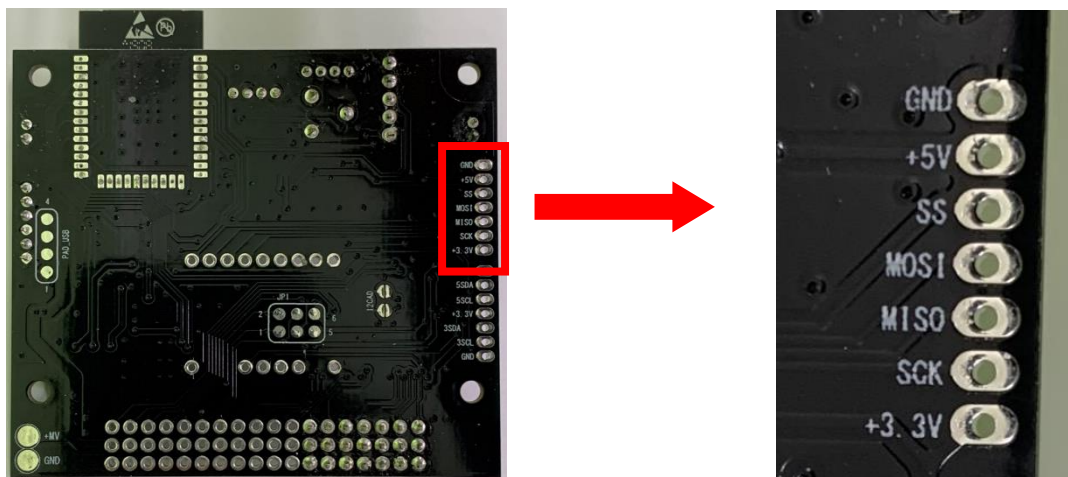
3.3V の I2C デバイスと通信する場合



5V の I2C デバイスと通信する場合

ICSP ポートの接続

当 ICSP ポートは、拡張ボードエリアの ICSP 通信のためのポートで、メイン CPU(ESP32-WROOM-32)では通常の I/O として接続されています。



ICSP デバイスと通信する場合

6.3 サーボモータコマンド一覧

サーボモータドライバ IC は I2C にて接続されています。サーボモータを駆動させるための送受信コマンド一覧です。使い方は、「[5.6 サーボモータドライバ通信用インターフェース](#)」のサンプルプログラムを参照してください。

※パルス幅単位 1 = 1/16 マイクロ秒（実パルス分解能は 1/8 マイクロ秒）

※角度単位 1 = 0.1 度

コマンド番号	動作	説明	送信バイト数	受信バイト数
0	サーボ動作モード設定	サーボ非同期動作、同期動作切り替え（デフォルトは非同期）	2	0
1	サーボ一斉スタート	サーボ同期動作時、一斉動作開始する。非同期モードでは何も変わらない。	1	0
2	PWM パルス出力許可	指定したサーボにパルス供給許可するか、Low 固定かの設定。初期はLow固定	3	0
3	サーボ目標設定 パルス幅指定	目標位置のパルス幅値、移動時間を指定して動作させる。	6	0
4	サーボ目標設定 角度指定	目標位置の角度、移動時間を指定して動作させる。	6	0
5	サーボポジションを得る (パルス幅単位)	サーボの内部演算現在位置をパルス幅値で返す。8320 から 39680 の値（8320=520us : 論理 0 度から 39680=2480us : 論理 180 度）。	2	2
6	サーボポジションを得る (角度単位)	サーボの内部演算現在位置を角度で返す。0 から 1800 の値。	2	2

コマンドはすべて、[コマンド番号-可変長パラメータ]の順にバイナリで送受信する。

サーボドライバへI2C送信する数値列は、常に

cmd, 引数1, 引数2, のフォーマットになる。

cmd はコマンド番号1バイト。引数1,2, ... はコマンド種類による。（ ）の中の数値はその引数のバイト数2バイトの引数はリトルエンディアン（例：256なら、0x00,0x01）

コマンド番号 0

【送信コマンド】 cmd(1), sw(1)

【引数】 sw 0:非同期動作（デフォルト）、 1:同期動作

コマンド番号 1

【送信コマンド】 cmd(1)

コマンド番号 2

【送信コマンド】 cmd(1), ch(1), bOut(1)

【引数】

ch チャンネル番号 0から17

bOut 出力許可なら1、禁止なら0

コマンド番号 3

【送信コマンド】 cmd(1), ch(1), PulseHD(2), mTime(2)

【引数】

ch, チャンネル番号 0 ～ 17

PulseHD, パルス幅 8320 ～ 39680 を指定。※1=1/16uSec なので 8320=520us=約0度、
39680=2480us=約180度

mTime, 移動時間 ミリ秒で指定。 0は最速 ただし物理的移動速度以上には速くならない。

コマンド番号 4

【送信コマンド】 cmd(1), ch(1), AngleHD(2), mTime(2)

【引数】 ch, チャンネル番号 0 ～ 17

AngleHD, 高分解能角度 10倍の数値指定。※1=1/10度 なので 1800=180度

mTime, 移動時間 ミリ秒で指定。 0は最速 ただし物理的移動速度以上には速くならない。

コマンド番号 5

【送信コマンド】 cmd(1), ch(1)

【引数】 ch, チャンネル番号 0 ～ 17

【受信】 2バイト。ポジションを PulseHD単位で受信

コマンド番号 6

【送信コマンド】 cmd(1), ch(1)

【引数】 ch チャンネル番号 0 ～ 17

【受信】 2バイト。ポジションを Angle単位で受信

6.4 WiFi 接続サンプル

ESP32 には WiFi 接続機能がありますので、WiFi アクセスポイントに接続してインターネット上のサーバーと通信することが簡単にできます。また WiFi アクセスポイントとして動作させるなど多様な使い方が可能です。ここでは一番簡単なサンプルとして、WiFi アクセスポイントからインターネット上のサーバーに http 接続するサンプルを示します。

サンプルソース

```
#include <WiFi.h>

const char* ssid = "xxxxxxx"; //WiFi アクセスポイント SSID
const char* password = "xxxxxxx"; //WiFi アクセスポイント接続パスワード

const char* host = "example.com"; //接続先サーバー
const char* url = "/index.html"; //表示ページ URL

WiFiClient client;

void setup()
{
    Serial.begin(115200);
    delay(100);

    //WiFi 接続開始
    WiFi.begin(ssid, password);
    Serial.print("Waiting for WiFi... ");

    while(WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }

    Serial.println("");
    Serial.println("WiFi connected");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
}

void loop()
{
    //WiFi コネクションを確認する。
    if(WiFi.status() != WL_CONNECTED) {
        Serial.println("Lost WiFi connection!");
        delay(5000);
        return;
    }

    //HTTP サーバーへ接続
    Serial.print("connecting to "); Serial.println(host);
```

```
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
    Serial.println("Can't connect to the server!");
    delay(5000);
    return;
}

//HTTP サーバーへリクエスト送信
Serial.print("Requesting URL: "); Serial.println(url);
client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" + "Connection: close\r\n\r\n");

unsigned long timeout = millis();
while (client.available() == 0) {
    if (millis() - timeout > 5000) {
        Serial.println(">>> Client Timeout !");
        client.stop();
        return;
    }
}

// Read all the lines of the reply from server and print them to Serial
while (client.available()) {
    String line = client.readStringUntil('\r');
    line.trim();    //先頭の\rn を除去
    Serial.print(line);
}

Serial.println();
Serial.println("closing connection");
Serial.println();
Serial.println();
delay(10000);
}
```

6.5 Bluetooth 接続サンプル

ESP32 には Bluetooth 接続機能がありますので、スマホやタブレットなどに接続してスマホアプリから制御するような使い方ができます。ただスマホ側のアプリ開発も同時に行わないと目的の制御を行うことはできません。iPhone などでは開発環境も限定されますので少しハードルが高くなりますので、ここでは既存の実験用スマホアプリを使用して BLE（Bluetooth Low Energy）で ESP32 と接続できるサンプルを示します。

これを使えば Qumcum の ESP32 からスマートフォンへデータ送信、スマートフォンから Qumcum へデータを送信できます。Qumcum のインターフェースと組み合わせれば Bluetooth で Qumcum を操作することも可能です。

「BLE Scanner」と「nRF Connect for Mobile」があります。前者は Bluepixel Technologies、後者は Nordic Semiconductor 社の提供です。スマホアプリとして検索、インストールしてください。どちらもほぼ同じような操作性です。詳細はそれぞれのアプリの説明を参照してください。

サンプルソース

```
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>

#define SERVICE_UUID          "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID   "beb5483e-36e1-4688-b7f5-ea07361b26a8"

class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
        std::string value = pCharacteristic->getValue();
        if (value.length() > 0) {
            Serial.print("New value: ");
            for (int i = 0; i < value.length(); i++)
                Serial.print(value[i]);
            Serial.println();
        }
    }
};

void setup() {
    Serial.begin(115200);
    BLEDevice::init("MyESP32");
    BLEServer *pServer = BLEDevice::createServer();
    BLEService *pService = pServer->createService(SERVICE_UUID);
    BLECharacteristic *pCharacteristic = pService->createCharacteristic(
        CHARACTERISTIC_UUID,
        BLECharacteristic::PROPERTY_READ |
        BLECharacteristic::PROPERTY_WRITE
    );
    pCharacteristic->setCallbacks(new MyCallbacks());
    pCharacteristic->setValue("Hello World");
    pService->start();
}
```

```
BLEAdvertising *pAdvertising = pServer->getAdvertising();
pAdvertising->start();
}

void loop() {
    delay(1000);
}
```

6.6 Qumcum R321J メインボード ESP32-WROOM-32D ピンアサイン

Pin No	Name	IO	R321J Description
1	GND		GND
2	3V3		3V3
3	EN		EN
4	SENSOR_VP	36	Not connected
5	SENSOR_VN	39	SS
6	IO34	34	Battery Power
7	IO35	35	Ultrasonic Sensor(Echo)
8	IO32	32	RGB-LED(Red)
9	IO33	33	RGB-LED(Blue)
10	IO25	25	Output(Voice)
11	IO26	26	Digital MIC(DOUT)
12	IO27	27	RGB-LED(Green)
13	IO14	14	Not connected
14	IO12	12	Not connected
15	GND		GND
16	IO13	13	Not connected
17	SD2	9	Not connected
18	SD3	10	PAD SPI(SCK)
19	CMD	11	PAD SPI(MISO)
20	CLK	6	Not connected
21	SD0	7	Not connected
22	SD1	8	Not connected
23	IO15	15	PAD SPI(MOSI)
24	IO2	2	Not connected
25	IO0	0	Serial(RTS)
26	IO4	4	Output(Beep)
27	IO16	16	Digital MIC(BCLK)
28	IO17	17	Digital MIC(LRCL)
29	IO5	5	Ultrasonic Sensor(Trigger)
30	IO18	18	Output(LED3 for Bluetooth Link Indicator)
31	IO19	19	Not connected
32	NC		Not connected
33	IO21	21	I2C(SDA)
34	RXD0	3	USB Serial(TXD)
35	TXD0	1	USB Serial(RXD)
36	IO22	22	I2C(SCL)
37	IO23	23	Not connected
38	GND		GND
39	GND		GND

6.7 オプション品

本章では、Qumcum R321J メインボードのオプション品について説明します。

名称	型番
Qumcum スイッチボード	QX-P003
Qumcum 本体電池ボックス	QX-P019
Qumcum スピーカ	QX-P006
Qumcum 距離センサー	QX-P007
Qumcum デジタルマイク	QX-P004
Qumcum サーボモータ	QX-P020

改訂履歴

1.0	初版
1.1	・誤記修正
1.2	・誤記修正
1.3	・誤記修正
1.4	・誤記修正
1.5	・ 5.2 BEEP 音出力用インターフェース サンプル修正
1.6	・ 5.3 VOICE 出力用インターフェース サンプル修正
1.7	<ul style="list-style-type: none"> ・ 3.7 各部の名称と説明 電源関連コネクタ説明修正 ・ 4.1 準備するもの 電源関連、スイッチボード、回路説明修正、注意書き追加 ・ 5.3VOICE 出力用インターフェース I2S IDF2.x 対応 IO ピン番号間違い訂正 ・ 5.4Qumcum デジタルマイク入力用インターフェース I2S IDF2.x 対応 ・ 5.6 サーボモータドライバ通信用インターフェース 説明追加、注意書き追加 ・ 6.1 サーボモータ用 5V 電源の接続 説明追加修正、スピーカーの接続 説明追加、 サーボモータの接続 説明追加、Qumcum デジタルマイクの接続 説明追加 ・ 6.2 拡張用ポートの接続 ICSP(JP1)の接続 削除 ・ 6.3 サーボモータコマンド一覧 追加 ・ 6.4WiFi サンプル 追加 ・ 6.5Bluetooth 接続サンプル 追加