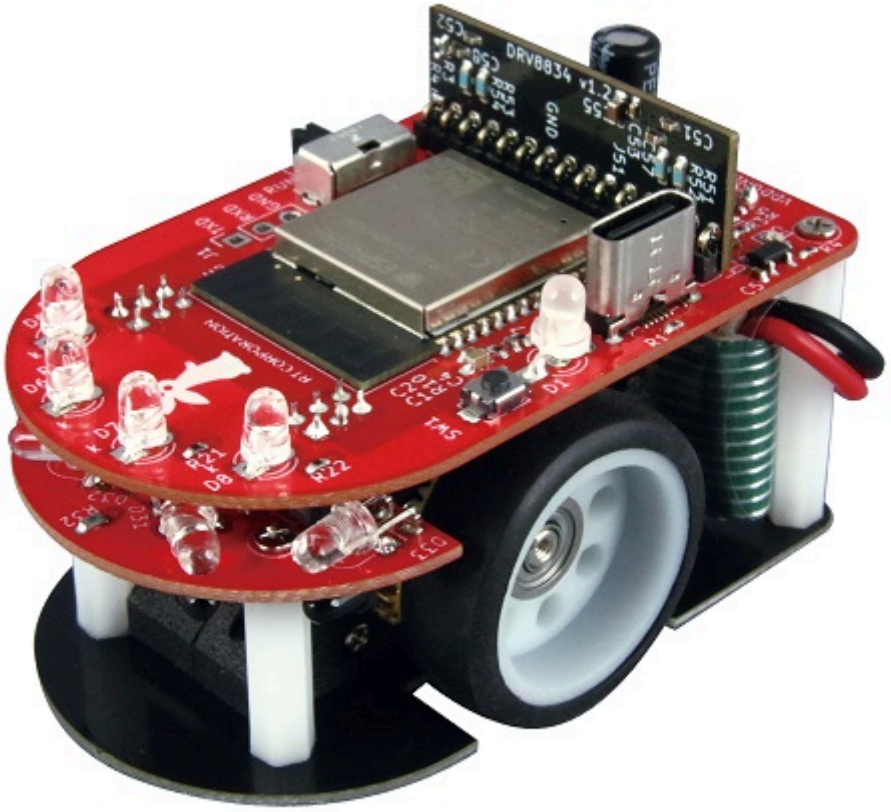
Pi:Co V2
ソフトウェア解説マニュアル
(Arduino編)　サンプル



1.0版

株式会社アールティ

RT CORPORATION

目次

ご使用になる前に

この度は、弊社の「Pi:Co V2（以下「本製品」といいます）」をお買い上げいただき、誠にありがとうございます。

本書は、本製品用のサンプルプログラムについて解説を記載しています。本製品をご使用になる前に、別紙の**Pi:Co V2 入門ガイド**をお読みいただきますようお願いします。

使用しているツール、OSSのバージョン

ここでは、**Arduino開発環境構築マニュアル**の環境設定について補足説明します。
このマニュアルを書いている時点で使用しているツールやOSSのバージョンは以下になります。

本書で使用する主なツールおよびOSS

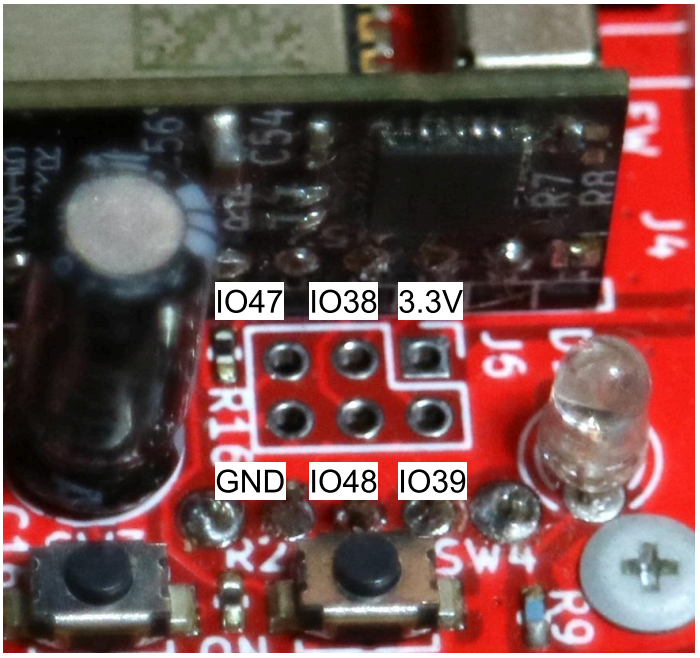
ツールやOSSなど	バージョン	URL
Arduino IDE	2.3.2	https://www.arduino.cc/en/software
Arduino core for the ESP32	3.0.1	https://github.com/espressif/arduino-esp32/releases
pico_v2_arduino_examples	1.0.0	https://github.com/rt-net/pico_v2_arduino_examples

ESP32-S3のIO接続先一覧

下表のIO番号はESP32-S3のポート番号です。接続先には、プログラム中のdefine文で定義されている名前を記載しています。

IO番号	接続先		IO番号	接続先		IO番号	接続先
0	モード 切替スイッチ		12	MOTOR_EN		37	未接続
1	SLED_S		13	SW_L		38	外部端子
2	SLED_F		14	SW_R		39	外部端子
3	未接続		15	LED2		40	BUZZER
4	AD3		16	LED3		41	LED1
5	AD4		17	BLED1		42	LED0
6	AD1		18	BLED0		45	PWM_R
7	AD2		19	USB_D-		46	PWM_L
8	AD0		20	USB_D+		47	外部端子
9	未接続		21	CW_R		48	外部端子
10	未接続		35	未接続		TXD0	TXD
11	CW_L		36	未接続		RXD0	RXD

外部端子はメイン基板に用意されています。サンプルプログラムでは使用しません。



サンプルプログラムの解説

ここでは、本製品のサンプルプログラム集であるpico_v2_arduino_examplesを用いて、本製品の開発方法と動かし方を解説します。あらかじめ別紙の**Arduino開発環境構築マニュアル**を参考にしてサンプルプログラムをインストールしてください。

本製品のサンプルプログラムはArduino IDEで開発できるため、ロボットプログラミング初心者にとって優しい環境で本製品の動かし方を学べます。例えば、Arduino IDEでは、よく使われる初期設定が関数として用意されており、Serial.beginやtimerBeginのような関数を実行するだけで初期化ができるようになっています。またdigitalWriteやanalogReadなどの関数名を見るだけで何をするのか判別できることも初心者を助けてくれます。

本書の構成について

さらに、ロボットプログラミング初心者のために、本書ではArduino IDEの使い方と本製品の動かし方を解説しています。Arduino IDEでのプログラミング経験がない方や、LEDやブザー、モータなどを動かしたことがない方でも理解できる内容です。本製品の回路図も載せているので、プログラムと合わせて読むと本製品の動かし方がより理解しやすくなると思います。

本書で扱うサンプルプログラムの構成は次のとおりです。最終的に本製品でマイクロマウス競技に出場できるようにサンプルプログラムが作られています。

STEP1からSTEP7までが、マイクロマウスとして必要な機能を確認するサンプルプログラムです。

STEP8がマイクロマウスとして動作するサンプルプログラムです。

サンプルプログラムの構成

サンプル	内容	開発環境
STEP1～STEP7	LEDやセンサ、モータなど、マイクロマウスとして必要な機能を動かす	Windows Linux（Ubuntu） macOS
STEP8	マイクロマウス競技用のプログラム	Windows Linux（Ubuntu） macOS

Arduinoスケッチの表記について

Arduino IDEで扱うプログラムはスケッチと呼ばれます。本書では、スケッチ特有の説明でない限り、スケッチを「プログラム」や「コード」などで表記しています。

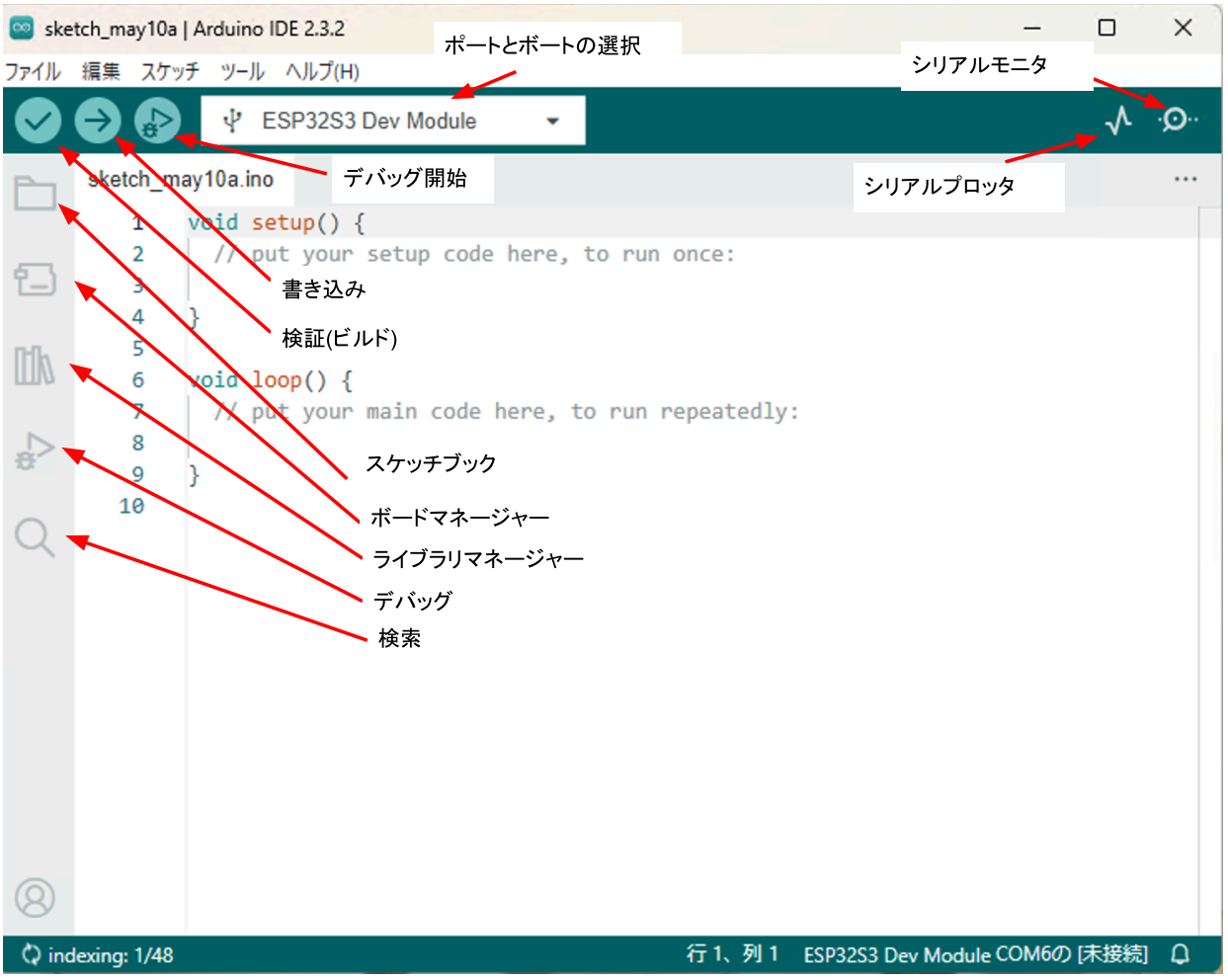
Arduinoスケッチは下記のように表記します。”C/C++”より下の行がスケッチの内容です。

```C/C++
void setup(){
 // ピンモードを出力に設定します
 pinMode(LED0, OUTPUT);
}

void loop(){
 // LEDを点滅させます
 digitalWrite(LED0, HIGH);
 delay(500);
 digitalWrite(LED0, LOW);
 delay(500);
}
```

Arduino IDEのアイコンについて

アイコンの名称は以下のようになっています。



Arduiono IDE　アイコンの名称

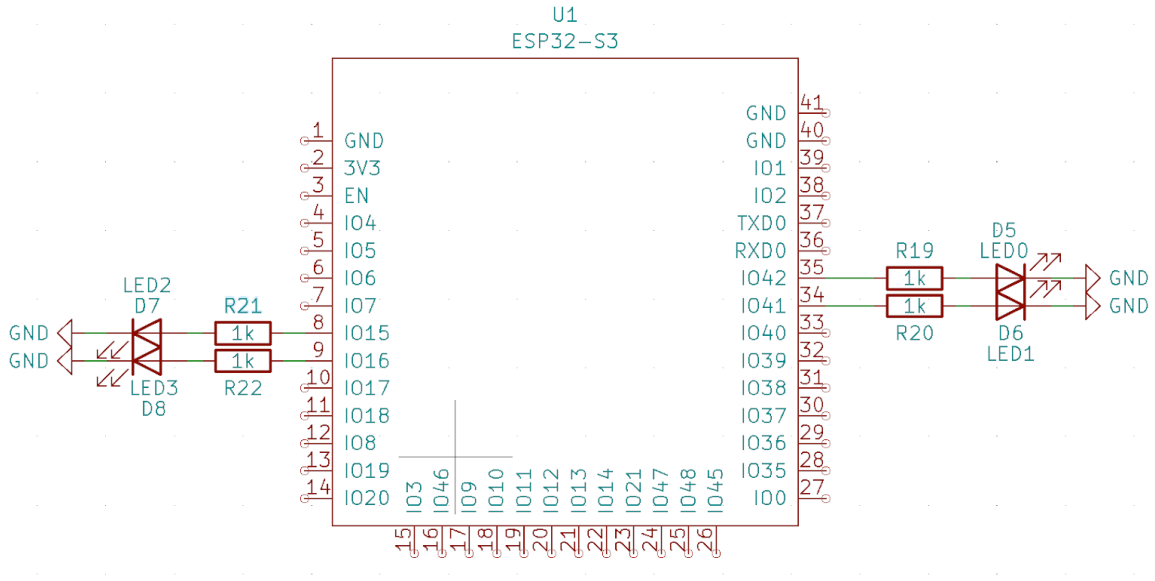
STEP1 LEDを点滅させる

概要

　前方にある4つのLED（モード表示用LED）で0.5秒間全点灯、0.5秒間全消灯を繰り返すサンプルプログラムです。

回路図

　LEDの点滅に関する回路要素を抜粋した回路図を下記に示します



LED駆動回路

回路の動作説明

　1を出力するとGPIO（General Purpose Input/Output：汎用I/Oポート）に接続されているLEDが点灯し、0を出力するとLEDが消灯する回路です。プログラムで”1”を出力するとGPIOでは3.3Vが出力、”0”を出力するとGPIOでは0Vが出力します。

プログラム

STEP1のプログラムを図で示します。

図1-1

```C/C++
#define LED0 42
#define LED1 41
#define LED2 15
#define LED3 16
```

図1-2

```C/C++
void setup(){
 // put your setup code here, to run once:
 pinMode(LED0, OUTPUT);
 pinMode(LED1, OUTPUT);
 pinMode(LED2, OUTPUT);
 pinMode(LED3, OUTPUT);
}
```

図1-3

```C/C++
void loop(){
 // put your main code here, to run repeatedly:
 digitalWrite(LED0, HIGH);
 digitalWrite(LED1, HIGH);
 digitalWrite(LED2, HIGH);
 digitalWrite(LED3, HIGH);
 delay(500);
 digitalWrite(LED0, LOW);
 digitalWrite(LED1, LOW);
 digitalWrite(LED2, LOW);
 digitalWrite(LED3, LOW);
 delay(500);
}
```

解説

マイコンのサンプルプログラムとして一番よく使われているのがLEDを点滅させるプログラムです。LEDが点滅することをLチカと呼びます。Lチカができることは、マイコンの開発環境が整ったということだけではなく、簡単なデバッグ環境が整ったことを意味します。STEP1以降では、プログラムの動作を確認するために、Lチカが活躍します。

main関数について

ArduinoスケッチはC言語ライクなプログラミング言語のため、C言語もしくはC++言語とほぼ同じ記述ができます。このサンプルプログラムではC言語ライクで記述しています。

C言語では、main関数というプログラム開始時に実行される関数を必ず記述します。しかし、Arduinoスケッチには、main関数がありません。その代わりにsetup関数とloop関数があります。setup関数を実行した後に、loop関数が実行される仕組みです。loop関数は、最後まで処理が終了するとloop関数の先頭に戻って再び処理を実行する特性を持っています。

#defineについて

プログラムの初めにある#defineについて簡単に説明します。

define文はdefineの右にある文字列を、その右にある数字や文字列に置き換えてコンパイルするプリプロセッサのマクロ定義です。

```C/C++
#define LED0 1
```

上記のdefine文を書くと、プログラム中の”LED0”と書かれた所が” 1 “に置き換わってコンパイルされます。

本製品には表示用のLEDが4つあります。LEDを制御する文を分かりやすくするため、LED0、LED1、LED2、LED3という文字列を定義し、ESP32-S3のGPIOの番号と紐づけています。

上記のdefine文で定義している数字は、ESP32におけるGPIOの番号であり、IO1を意味します。ArduinoスケッチではESP32のIO番号を指定すればGPIOを制御できます。**ESP32のICのPin番号ではないので注意してください**。

#defineを使わない場合、図1-2のpinMode(LED0,OUTPUT)はpinMode(1,OUTPUT)となります。開発しているときは、”1”と書かれていてもLED0と覚えていられるのですが、少し時間が経った後だと”1”がどの機能に割り当てていたかを忘れます。なるべく数字ではなく文字列でプログラミングすることをお勧めします。

また、#defineで定義しておくと、プログラムの流用性が良くなります。例えば、仕様変更でLEDを駆動するポートが変わるとき、この#defineの番号を変更するだけで済みます。プログラムの奥深くまで修正する必要はありません。

setup関数について

次にsetup関数に記述されている内容について解説します。

setup関数は、loop関数を実行する前に一度だけ実行する関数です。ここにマイコンの機能の初期化や変数の初期化を記述することが多いです。このサンプルプログラムでは、LEDを点滅させることを目的としており、LEDの点滅は、マイコンから見ると出力になります。

GPIOを出力設定にするには、Arduino IDEのpinModeという組み込み関数を使います。Arduinoスケッチでは関数名の大文字小文字を認識するため、pinMode関数の”M”は大文字で記述してください。pinMode関数の第一引数にピン番号、第二引数に入出力の方向を指定します。一般的なC言語環境では、使用する関数のファイルをincludeする必要がありますが、pinMode関数はArduino IDE環境に組み込まれているため、includeをする必要がありません。

また、pinMode(LED0,OUTPUT) のようにGPIOの入出力を”INPUT”、”OUTPUT”という文字で設定できます。INPUTとOUTPUTもArduino IDEの機能として組み込まれています。INPUTとOUTPUTの文字で入出力の設定ができると、後から見てこのピンはOUTPUTに設定しているということが簡単にわかって便利です。

便利な機能なのですが、注意点があります。あらかじめArduino IDEに組み込まれている関数や変数などは、本来の使用用途とは違う目的で使わないようにしましょう。ユーザ側で別の機能として使うと、ビルドエラーになったり、意図せず変数の設定値が変更されたりします。定義されている関数や変数はArduinoのwebページで確認できます。
https://www.arduino.cc/reference/en/

loop関数について

最後にloop関数に記述されている内容について解説します。

loop関数には、繰り返しプログラムを実行する内容を記述します。C言語における while(1){ } と読み替えると理解しやすいでしょう。本製品の表示用LEDは1(=HIGH)にすると点灯、0(=LOW)にすると消灯します。

GPIOに”1”または”0”を出力するときはArduino IDEの組み込み関数digitilWriteを使います。第一引数にピン番号、第二引数に出力する値を入れます。ここでも出力する値を”HIGH”、”LOW”という文字列で設定することができます。HIGHは”1”、LOWは”0”と定義されているのでdigitalWrite関数以外でも使うことができます。

digitalWrite(LED0,HIGH)でLEDを点灯させた後、delay(500)の記述があります。この記述がなくても、digitalWrite(LED0,HIGH)の点灯とdigitalWrite(LED0,LOW)の消灯は繰り返し行われます。しかし、マイコンの処理速度が早いため、点滅していることを人の目では認識することができません。そこで、人の目でも認識できるように点灯と消灯の時間を設けています。

その遅延を操作する関数はArduino IDEの組み込み関数delayです。このdelay関数があると、引数の値 x ms（milli seconds、ミリビョウ）だけプログラムが停止します。サンプルではdelay(500)と書かれているので点灯時間が500[ms]、消灯時間が500[ms]となります。