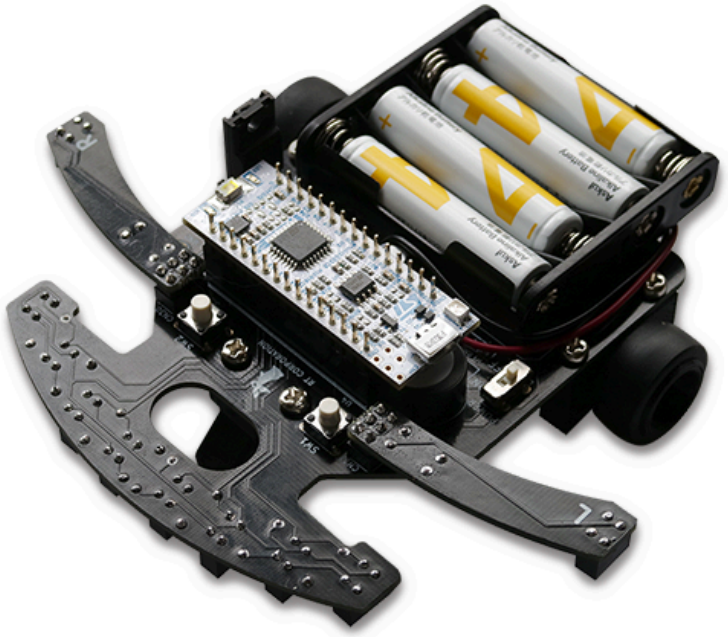
Training Tracer Ver.2
ソフトウェア解説マニュアル
(Arduino編)　サンプル



v1.0

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、株式会社アールティ（以下「弊社」といいます）の「Traning Tracer Ver.2（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Arduino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**Traning Tracer Ver.2 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等	バージョン	URL
Arduino IDE	2.3.2	https://www.arduino.cc/en/software
Arduino Core STM32	2.8.1	https://github.com/stm32duino/Arduino_Core_STM32
TrainingTracer_v2_Samples	1.0.0	https://github.com/rt-net/TrainingTracer_V2_Samples

STM32F303K8のピン接続先一覧

接続先には、プログラム中でdefineやintで宣言している名前を記載しています。

ピン番号	接続先		ピン番号	接続先		ピン番号	接続先
D0	DIR_L_PIN		D7	SW1_PIN		A0	ENC_A_R_PIN
D1	ENC_A_L_PIN		D8	SW2_PIN		A1	ENC_B_R_PIN
D2	BUZZER_PIN		D9	ENC_B_L_PIN		A2	LINE_R2_PIN
D3	Marker_L		D10	DIR_R_PIN		A3	LINE_R1_PIN
D4	BMX055_SDA		D11	PWM_R_PIN		A4	LINE_L1_PIN
D5	BMX055_SCL		D12	PWM_L_PIN		A5	LINE_L2_PIN
D6	Marker_R		D13	LED_PIN		A6	POWER_PIN

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TrainingTracer_v2_Samples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等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、モータ等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。最終的に本製品でロボトレー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STEP8までが、ロボトレーサとして必要な機能を確認するサンプルプログラムです。

STEP9がロボトレーサとして動作するサンプルプログラムです。

各STEPごとに演習問題を用意しています。現状の実力を見るには良い課題になっています。

サンプルプログラムの構成

サンプル	内容	開発環境
STEP1～STEP8	LEDやセンサ、モータ等、ロボトレーサとして必要な機能を動かす	Windows Linux(Ubuntu) macOS
STEP9	ロボトレー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」等で表記しています。

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。

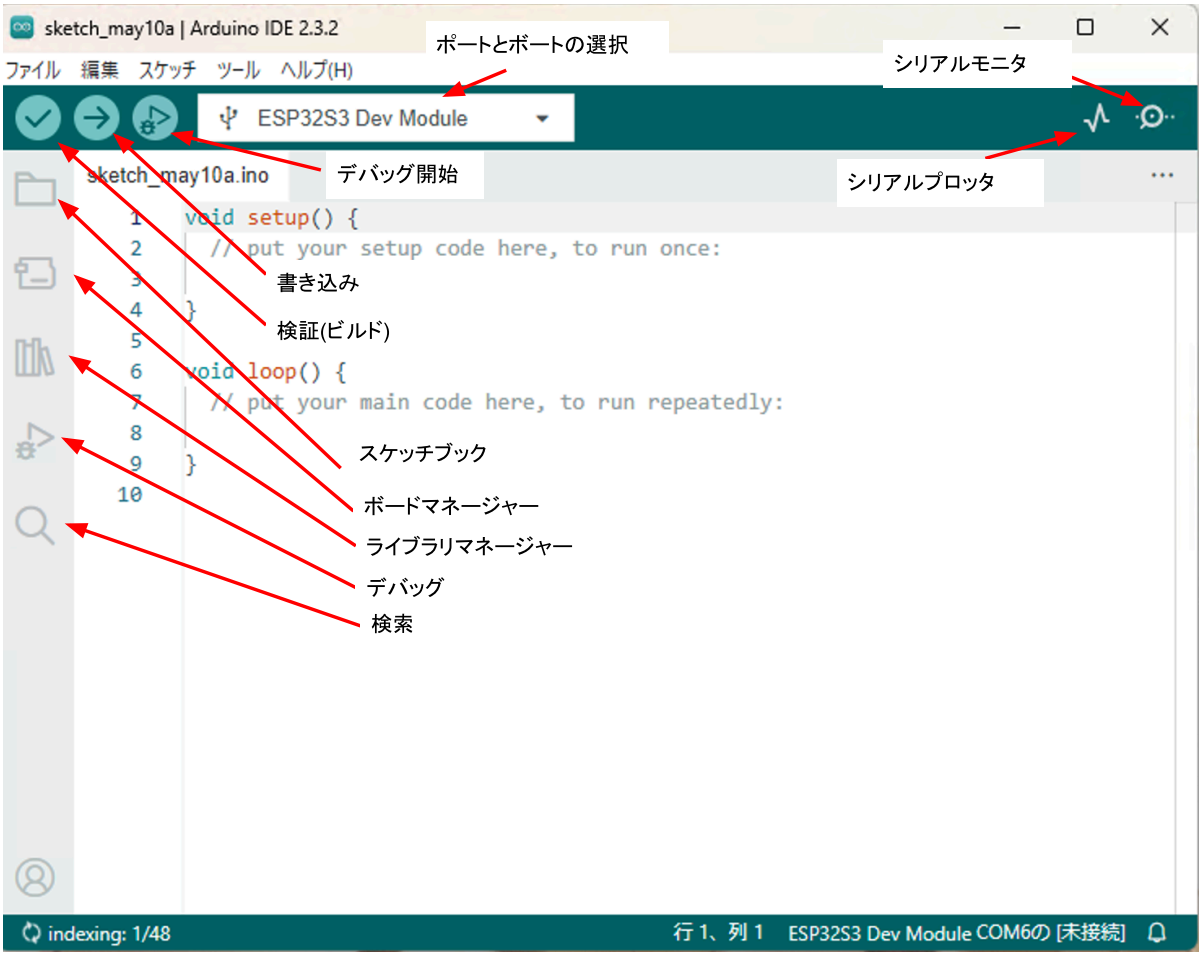
```C/C++
void setup(){
```

```
 // 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

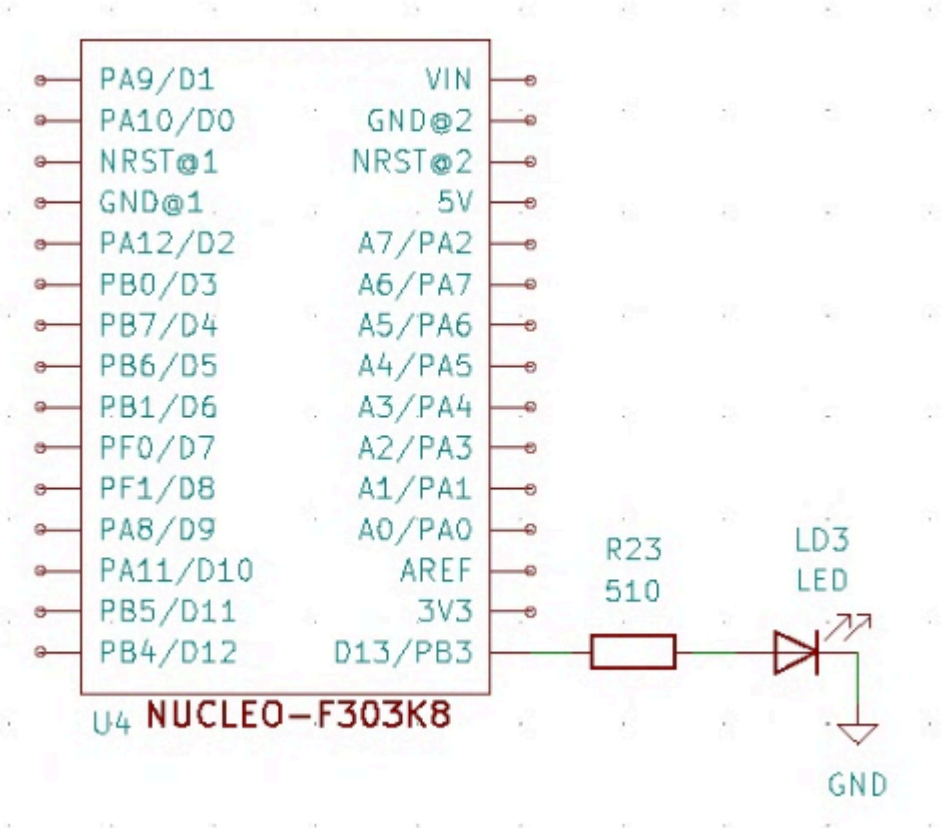
STEP1 LEDを点滅させる

概要

Nucleoボード上のLD3を1秒間点灯、1秒間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。
このLEDはNucleoボードに搭載されているLEDなので、TrainingTracerの回路図には載っていません。



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。

プログラムソース

STEP1のプログラムを図に示します。

図1-1

```C/C++
#define LED_PIN D13
```

図1-2

```C/C++
void setup() {
  // put your setup code here, to run once:
  pinMode(LED_Pin,OUTPUT);
  pinMode(D12,OUTPUT);
  digitalWrite(D12,LOW);
}
```

図1-3

```C/C++
void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(LED_PIN,HIGH); //LEDを点灯
  delay(1000);                //1秒待つ
  digitalWrite(LED_PIN,LOW);  //LED消灯
  delay(1000);                //1秒待つ
}
```

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

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 LED_PIN D13
```

　上記のdefine文を書くと、プログラム中の”LED_PIN ”と書かれた所が” D13 “に置き換わってコンパイルされます。
　上記のdefine文で定義している”D13"は、Arduino Nanoのデジタル入出力ピンの番号を示しています。NUCLEO-F303K8は、Arduino Nanoとスケッチ上でのプログラム的な互換性を持っており、GPIOのD0から D13とアナログピンのA0からA7を有しており、これらのピンは、一部を除きArduino IDE上で同じ記述で動作することができます。
　#defineを使わない場合、図1-2のpinMode(LED_PIN ,OUTPUT)はpinMode(D13,OUTPUT)となります。開発しているときは、”D13”と書かれていてもLED_Pinと覚えていられるのですが、少し時間が経った後だと”D13”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(LED_PIN,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等は、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/

D12に関してはモータを制御するために使用しているピンなのですが、マイコンの起動時デフォルト設定がプルアップになっているため、ピンの出力がHIGHになります。そのためモータが回転してしまいます。、モータが回転しないようにD12のピンの出力をLOWに設定して、モータが回転し続けないようにしています。

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(LED_PIN,HIGH)でLEDを点灯させた後、delay(1000)の記述があります。この記述がなくても、digitalWrite(LED_PIN,HIGH)の点灯とdigitalWrite(LED_PIN,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(1000)と書かれているので点灯時間が1000[ms]、消灯時間が1000[ms]となります。

演習問題1

サンプルプログラムでは、LEDが1秒ごとに点滅していますが、これを0.5秒の点滅に変更してみましょう。

解答例は、Arduino_Exercises¥Exercise1¥Exercise1.inoにあります。