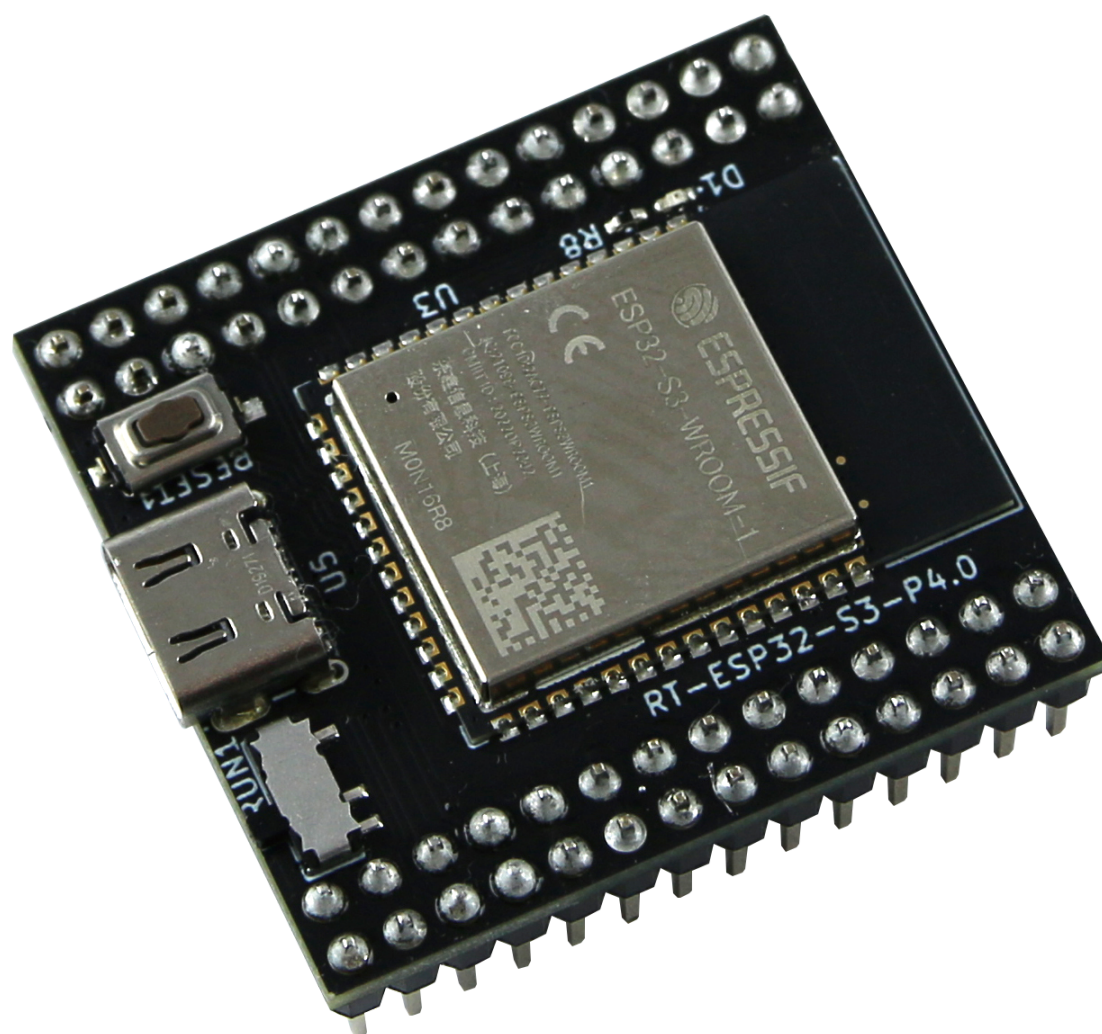


# ESP32-S3マイコンボード 入門テキスト サンプル



1.0版  
株式会社アールティ

# 目次

目次	1
ご使用になる前に	2
製品に対する注意事項	2
安全に関する注意事項	3
環境設定の補足	4
開発環境	4
使用しているツール、OSSのバージョン	5
サンプルプログラム	6
本書の構成について	6
コマンドとArduinoスケッチの表記について	7
STEP1 LEDを点滅させる	8
STEP2 スイッチを押してLEDを点灯させる	12
STEP3 ブザーを使って音を出す	17
STEP4 センサの値をシリアルモニタに表示する	25
STEP5 モータを使って直進する	35
STEP6 モータを使って旋回する	50
STEP7 P制御を使って壁に沿って走行する	54
STEP8 迷路を走破する	62
STEP9 micro-ROSを使ってPi:Co Classic3を動作させる(twist)	83
STEP10 micro-ROSを使ってRViz2上のPi:Co Classic3を同期させる(tf)	94
STEP11 micro-ROSを使ってセンサ値とバッテリーの電圧を確認する(pico_sensor)	104
STEP12 micro-ROSを使って走行した迷路を表示する	109
お問い合わせ	129
改訂履歴	129
Copyright・知的財産権について	129

## ご使用になる前に

この度は、弊社の「ESP32-S3マイコンボード（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**ESP32-S3マイコンボード インストールマニュアル**をお読みいただきますようお願いいたします。



## 製品に対する注意事項

---




- 初めてロボットを使用される方は、経験者と一緒に作業することをお勧めします。
- 本書は、Windows、macOSまたはLinuxの基本的な使用方法を理解している方を対象として構成されています。
- 本書の内容や本製品の仕様および外観ならびにweb上で公開しているデータおよび情報は、改良のため予告なく変更することがあります。改良版は、ご購入時点の製品、データおよび情報と異なる可能性があります。異なる点について交換、返金、返品、改変等はいたしかねますのでご了承くださいますようお願いいたします。
- 本書において記載されております画像、その他記載されている会社名、製品名等の固有名詞は各社の登録商標または商標です。なお、本文中では、TM、(R)マークは省略しております。

## 安全に関する注意事項

### 警告表示について

マーク	マークの定義
 <b>危険</b>	「 <b>危険</b> 」を表します。 取り扱いを誤った場合に、死亡または重傷を負う状態が生じることが想定され、かつ火災発生などの財物に重大な損害が生じる緊急性の高い事項を表します。
 <b>注意</b>	「 <b>注意</b> 」を表します。 取り扱いを誤った場合に、軽傷または物理的損害が生じることが想定される事項を表します。

### 危険内容および対応方法

	危険内容（行為、現象）	マーク	対応方法
作業時	Pi:Co Classic3に本製品を搭載して走行した時、周辺の金属に触れてショートする。または、本製品の操作時に金属物(指輪やネックレス等)が触れてショートする。	 <b>危険</b>	運用時、周辺に金属物がないことを確認してください。また、操作時に金属製の指輪等を外した状態で操作してください。
組立時	Pi:Co Classic3にバッテリーを繋げた状態で本製品を交換する。	 <b>危険</b>	Pi:Co Classic3の電源スイッチをOFFにし、バッテリーを外した状態で本製品を交換してください。
組立時	本製品のJP1パッドをショートした状態で本製品をPi:Co Classic3に取り付け、Pi:Co Classic3の回路が破壊される。	 <b>注意</b>	JP1パッドをショートした場合は、本製品をPi:Co Classic3に取り付けしないでください。

## サンプルプログラム

ここでは、本製品のサンプルプログラム集であるmicro-ROS Arduino examples for Pi:Co Classic3を用いて、本製品の開発方法とPi:Co Classic3の動かし方を解説します。あらかじめ別紙のインストール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Pi:Co Classic3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の構成を工夫しています。本書の前半では、Arduino IDEの使い方とPi:Co Classic3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。本製品とPi:Co Classic3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Pi:Co Classic3の動かし方がより理解しやすくなると思います。

本書の後半からROS 2とmicro-ROSが登場します。前半でPi:Co Classic3の動かし方を習得しているため、micro-ROSの使い方を集中して学ぶことができます。

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Pi:Co Classic3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。

STEP9からSTEP11がmicro-ROSを使ってPCの画面にセンサの値を表示させたり、PCからPi:Co Classic3を移動させたりする基本的なサンプルプログラムです。

STEP12はmicro-ROSとマイクロマウスのプログラムを組み合わせる迷路の探索を可視化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 macOS Linux
STEP8	マイクロマウス競技用のプログラム	Windows macOS Linux
STEP9～STEP11	micro-ROSを用いてセンサやモータなどを動かす	Linux (Ubuntu)
STEP12	micro-ROSを用いたマイクロマウス競技用のプログラム	Linux (Ubuntu)

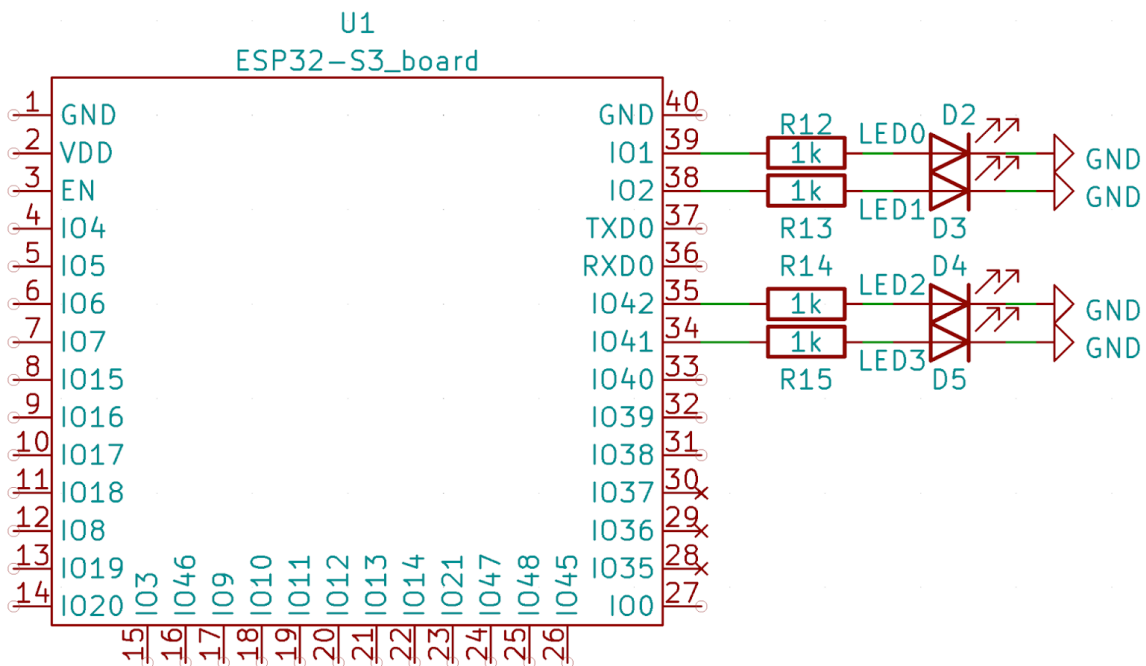
## STEP1 LEDを点滅させる

### 概要

前方にある4つの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 1
#define LED1 2
#define LED2 42
#define LED3 41
```

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
```

という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。

Pi:Co Classic3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IO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（General Purpose Input/Output：汎用I/Oポート）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、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"mode"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。pinMode関数はArduino IDEに組み込まれているため、#includeが不要です。

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。Pi:Co Classic3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digitalWrite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と、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、ミリ秒) 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。

## STEP12 micro-ROSを使って走行した迷路を表示する

### 概要

STEP8、STEP10、STEP11を統合した、マイクロマウスクラシック競技の迷路を走行するサンプルプログラムです。取得した迷路情報をmarkerトピック（visualization\_msgs/Marker）として配信し、RViz2で確認します。

### プログラム

STEP12で新たに追加されたプログラムは解説のところで紹介します。

### 実行方法

STEP10、STEP11と同じようにROS 2とmicro-ROSの環境を用意します。

#### Pi:Co Classic3側

STEP10、STEP11と同じようにset\_microros\_wifi\_transports()でWi-Fiの設定を行います。ただし、microROS.inoファイルに関数が記述されているためご注意ください。

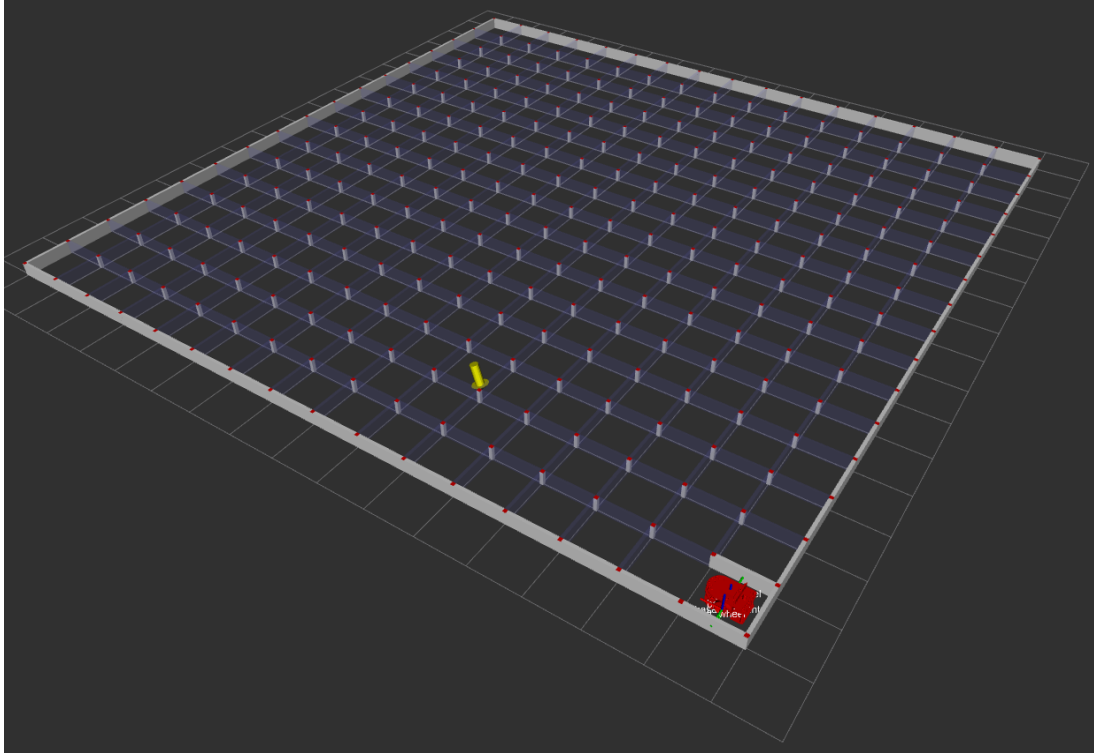
Pi:Co Classic3が正常に起動するとSTEP8と同様にスイッチ操作で走行モードを選択できます。起動中の動作はSTEP8と異なりますので、下記の表を確認してください。

起動中にmarkerトピックを配信するため、先にPC側でRViz2を起動しておくことを推奨します。

STEP12の起動動作

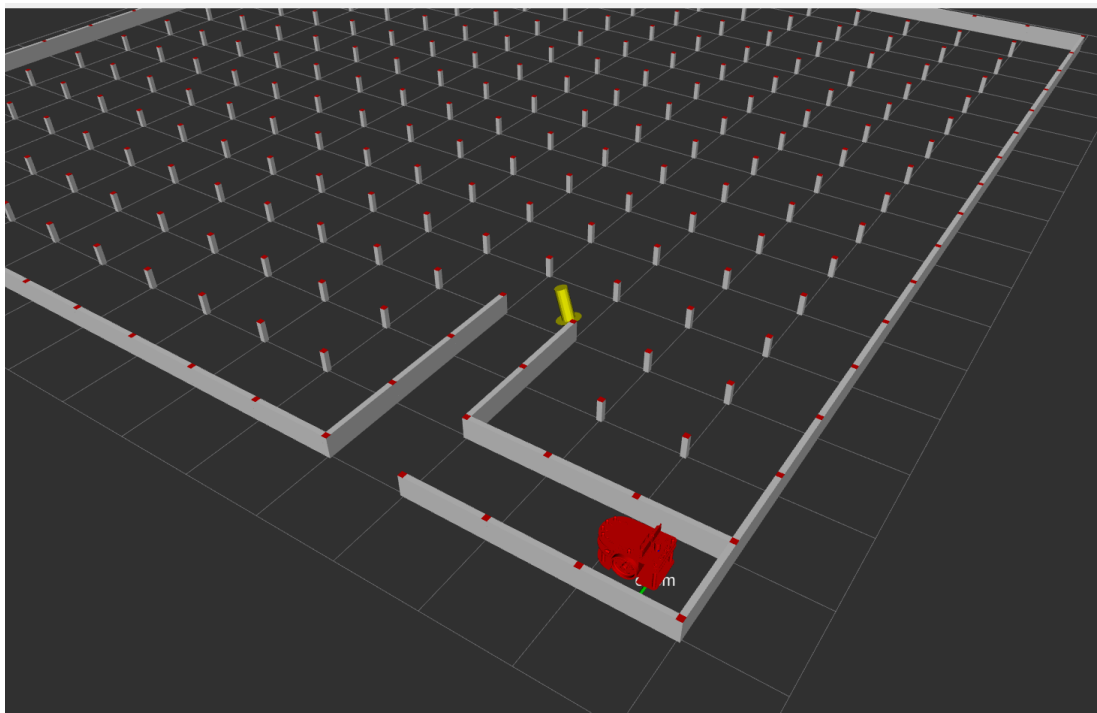
Pi:Co Classic3の状態	LEDとブザーの状態
①プログラム起動直後	ブザーが鳴る すべてのLEDが消灯する
②Wi-Fiの接続に成功	LED0～LED3が順番に点灯する
③micro-ROS-Agentとの通信に失敗した場合	LED0が点滅する
④markerトピックの配信中	配信進捗に合わせてLED0～LED3が点灯する
⑤走行モード選択中	走行モード番号に合わせてLED0～LED3が点灯する

markerトピックの配信が終わると次のように表示されます。黄色い矢印はゴール座標を示しています。



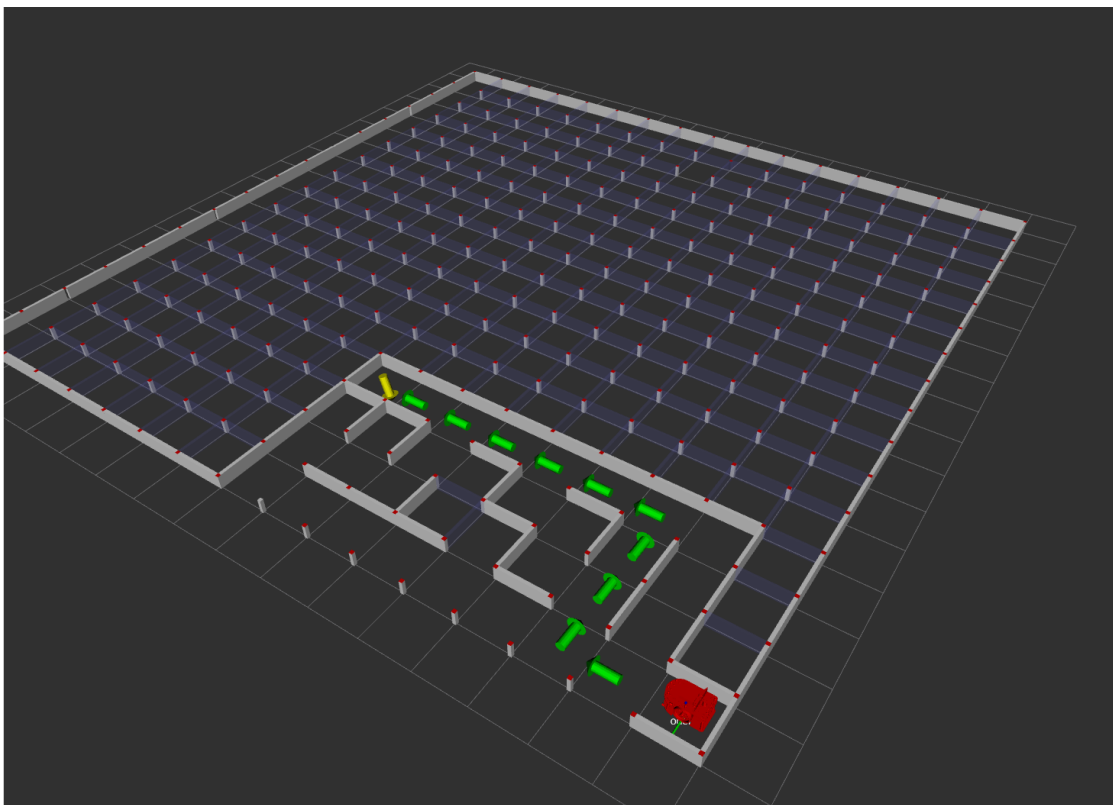
起動後のRViz2画面（ゴール座標 $x=2$ 、 $y=6$ ）

起動後はSTEP8と同様にスイッチ操作で走行モードを選択します。探索走行（モード番号2）を実行すると、走行中に更新された迷路情報がmarkerトピックとして配信されます。走行途中に更新された壁情報は、次のようにRViz2へ表示されます。



探索走行時のRViz2画面（ゴール座標 $x=3$ 、 $y=3$ ）

最短走行（モード番号3）を実行すると、走行経路が緑色の矢印で表示されます。



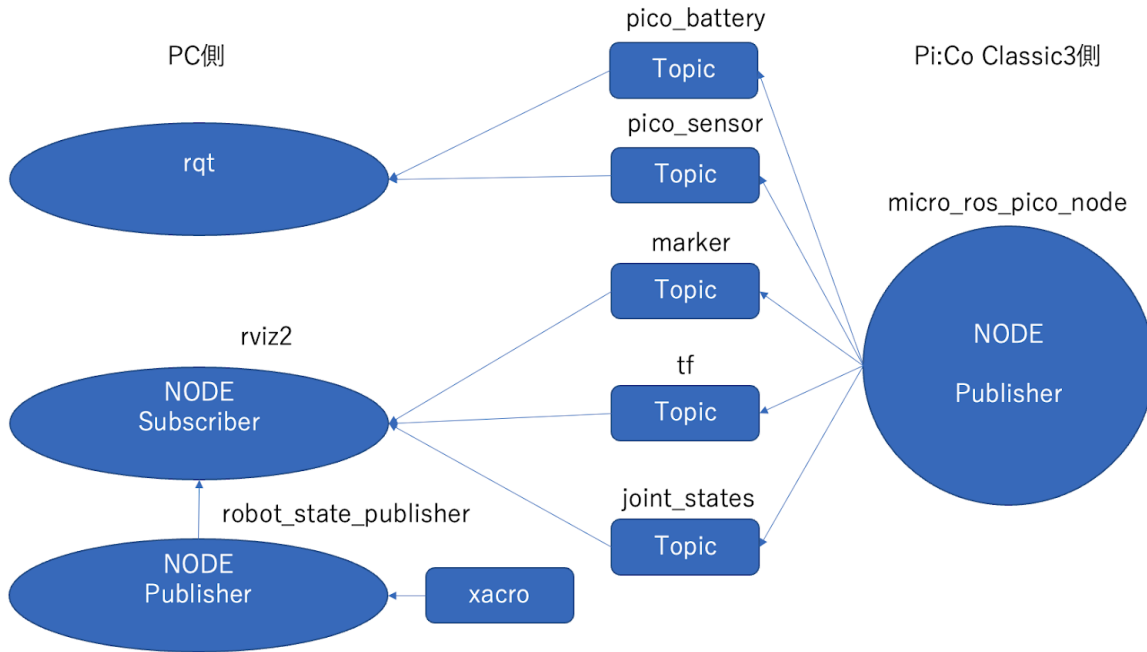
最短走行時のRViz2画面（ゴール座標 $x=3$ 、 $y=8$ ）

## 解説

STEP9からSTEP11まではmicro-ROSの設定をsetup関数があるファイルに記述しましたが、STEP12ではmicroROS.inoファイルに記述しています。そのほか、STEP8に対して変更があったところを解説します。

## ノード関係図

STEP12のノード関係図とトピックの一覧表を次に示します。マイクロマウスの競技中はマウスに対して外部から指示を与えてはいけないため、サンプルプログラムにはパブリッシャのみを実装しています。



STEP12のノード関係図

STEP12で登場するトピックの一覧表

トピック名	メッセージ型	メッセージ内容
marker	visualization_msgs/Marker	std_msgs/Header header string ns int32 id int32 type int32 action geometry_msgs/Pose pose geometry_msgs/Vector3 scale std_msgs/ColorRGBA color ※一部省略
tf	tf2_msgs/TFMessage	※STEP10に掲載
joint_states	sensor_msgs/JointState	※STEP10に掲載
pico_sensor	pico_msgs/LightSensor	※STEP11に掲載
pico_battery	std_msgs/Int16	※STEP11に掲載