

# mbedでプログラミング



# 目次

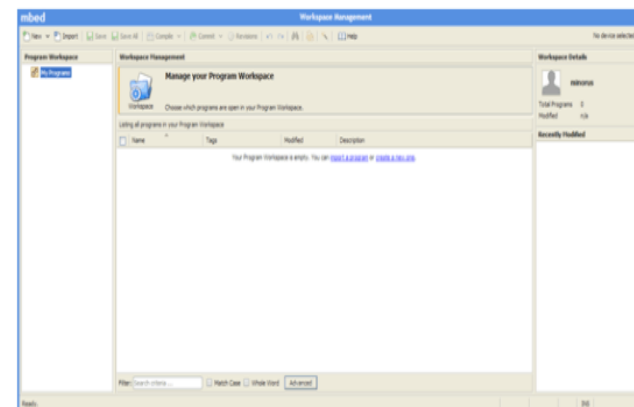
- mbedとは
- 使用するmbedマイコンボード
- サンプルプログラム書き込み
- PWM制御を使用したモータ速度制御
- PCとの通信

# 目次

- mbedとは
- 使用するmbedマイコンボード
- サンプルプログラム書き込み
- PWM制御を使用したモータ速度制御
- PCとの通信

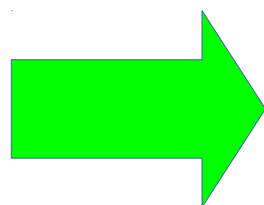
# mbed(エンベッド)とは・・・

ARM社のプロトタイピング用マイコンボード  
およびそのデバイスのプログラミング環境



# ①Web上でプログラム開発ができる

- インストール不要
- インターネットブラウザが動けば、OSを選ばない



すぐに動かせる！

# ①Web上でプログラム開発ができる

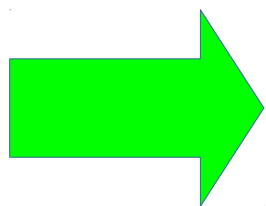
- インターネットブラウザが動けば、OSを選ばない

Q. ネットに繋がないと使えない？

A. 使えません。  
ただし、開発環境をインストールすればオフラインでも使えます。

## ②ライブラリが豊富

- 公式のライブラリから、全世界の開発者が作成した独自のライブラリまで様々



色々なことを簡単にできる！

## ③その他

- 開発言語 : C++
- マイコン : ARM(高性能で安価)
- 価格 : 安価(千円台)なボードが増えてきた



# Arduino と mbed

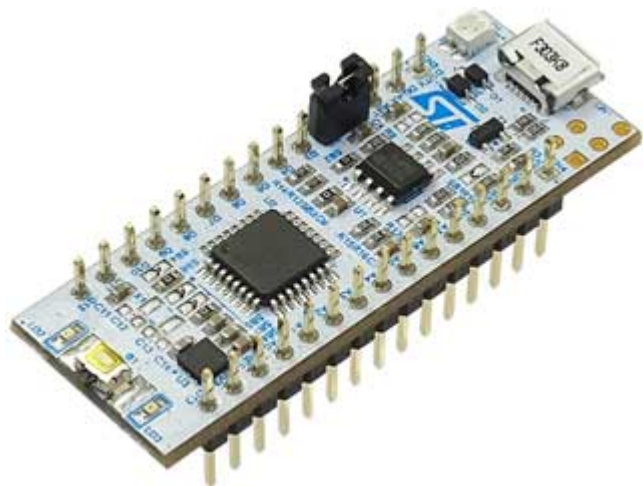
	Arduino	mbed
マイコン	AVR	ARM Cortex~
開発言語	C++ベース	C++
コンパイラ	RVDS	GCC
開発環境	JAVA(インスト要)	クラウド(インスト不要)
価格	2000円~8000円	1000円~10000円
性能	16MHz~	50MHz~1GHz

# 目次

- mbedとは
- 使用するmbedマイコンボード
- サンプルプログラム書き込み
- PWM制御を使用したモータ速度制御
- PCとの通信

# STM32 Nucleo Board

## STM32F303K8



※秋月電子で¥1600

小型、安価、高性能

CPU	STM32F303K8T6 (ARM Cortex-M4)
動作周波数	最大72MHz
CPU電源電圧	2.0V-3.6V
フラッシュ	64KB
SRAM	16KB
SPI	1
USART	2
CAN	1
ADC	9
DAC	3
GPIO	25
PWM	13~17

# 目次

- mbedとは
- 使用するmbedマイコンボード
- サンプルプログラム書き込み
- PWM制御を使用したモータ速度制御
- PCとの通信

# ①インターネットで”mbed”を検索



The screenshot shows a search engine interface. At the top, a search bar contains the text 'mbed' with a close button (X) on the right. Below the search bar, it indicates '約543,000件' (approximately 543,000 items). Underneath, there is a section for '検索ツール' (search tools) with a dropdown arrow. Below that, there are search suggestions: 'mbed or kl25z or ble', 'mbed 入門', and 'mbed オフライン コンパイル', followed by a 'で検索' (search) button. The main search results section starts with a link 'Home | mbed - このページを和訳' (Home | mbed - translate this page to Japanese) and a link 'www.mbed.com/ - キャッシュ' (www.mbed.com/ - cache). Below these is a paragraph: 'The ARM® mbed™ IoT Device Platform provides the operating system, cloud developer ecosystem to make the creation and deployment of commercial, stations possible at scale.' The next result is a link 'mbedを始めましょう! ("Let's get started!" in Japanese) | mbed' which is highlighted with a red rectangular box. Below this link is another link 'developer.mbed.org/users/nxplan/.../lets\_get\_started\_jp/ - キャッシュ' (developer.mbed.org/users/nxplan/.../lets\_get\_started\_jp/ - cache). At the bottom, there is a paragraph: 'PCにつなぐとmbed基板の真ん中にある青いLEDが光り, PCからは2MBのUSB×MBEDドライブがPC上に表示され ... このファイルには, そのmbedのシリアル番号へのリンク情報が入っています.'

mbed

約543,000件

検索ツール ▾

🔍 [mbed or kl25z or ble](#) [mbed 入門](#) [mbed オフライン コンパイル](#) で検索

[Home | mbed](#) - このページを和訳

[www.mbed.com/](#) - キャッシュ

The ARM® mbed™ IoT Device Platform provides the operating system, cloud developer ecosystem to make the creation and deployment of commercial, stations possible at scale.

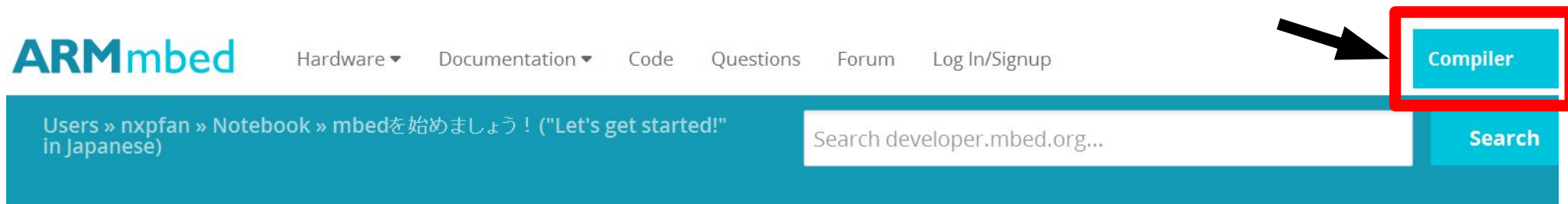
[mbedを始めましょう! \("Let's get started!" in Japanese\) | mbed](#)

[developer.mbed.org/users/nxplan/.../lets\\_get\\_started\\_jp/](#) - キャッシュ

PCにつなぐとmbed基板の真ん中にある青いLEDが光り, PCからは2MBのUSB×MBEDドライブがPC上に表示され ... このファイルには, そのmbedのシリアル番号へのリンク情報が入っています.

ここを  
クリック

## ②右上にある”Compiler”をクリック



### mbedを始めましょう! ("Let's get started!" in Japanese)

Page last updated 08 4月 2015, by NXP fan (in Japan). 18 replies

このノートブックページは日本でのmbed普及加速を目的にまとめたものです。

This notebook page has been written in Japanese only.

mbedを始めましょう!  
("Let's get started!" in Japanese)

Japanese getting start guide for new users

Page owner: NXP fan (in Japan)

### ③ログインする

**Login**

ユーザー名:

tory|

I've forgotten my username

Password:

●●●●●●●●

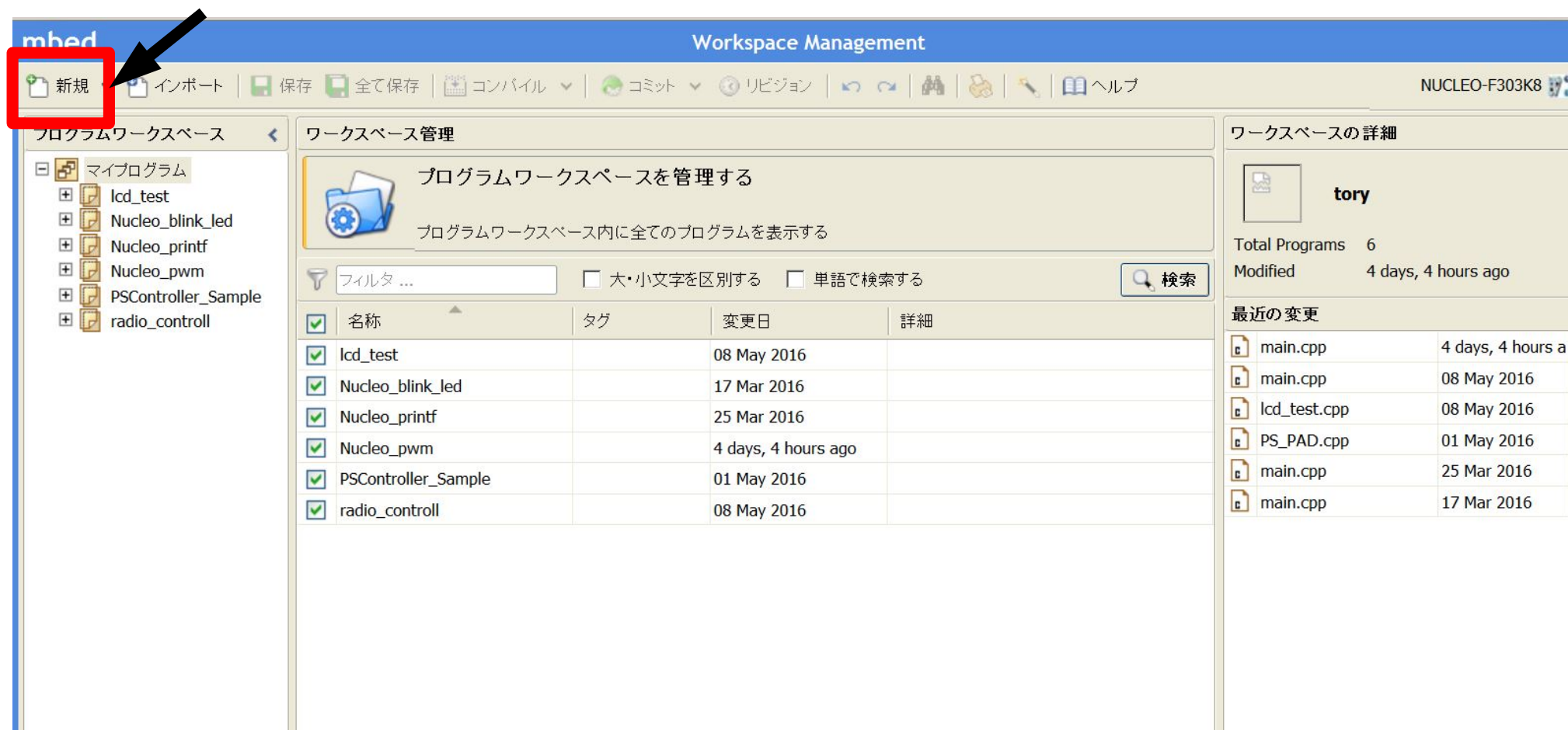
私は自分のパスワードを忘れてしまった

☐ 次回から入力を省略

Login

入力する  
(アカウント登録必要)

## ④左上の”新規”→”新しいプログラム”をクリック



The screenshot shows the mbed Workspace Management interface. The top toolbar contains buttons for '新規' (New), 'インポート' (Import), '保存' (Save), '全て保存' (Save All), 'コンパイル' (Compile), 'コミット' (Commit), 'リビジョン' (Revision), and 'ヘルプ' (Help). The '新規' button is highlighted with a red box and a black arrow.

The left sidebar shows a tree view of the workspace structure, including 'マイプログラム' (My Programs) and a list of programs: 'lcd\_test', 'Nucleo\_blink\_led', 'Nucleo\_printf', 'Nucleo\_pwm', 'PSController\_Sample', and 'radio\_controll'.

The main workspace management area displays a list of programs with the following columns: '名称' (Name), 'タグ' (Tag), '変更日' (Last Modified), and '詳細' (Details). The table contains the following data:

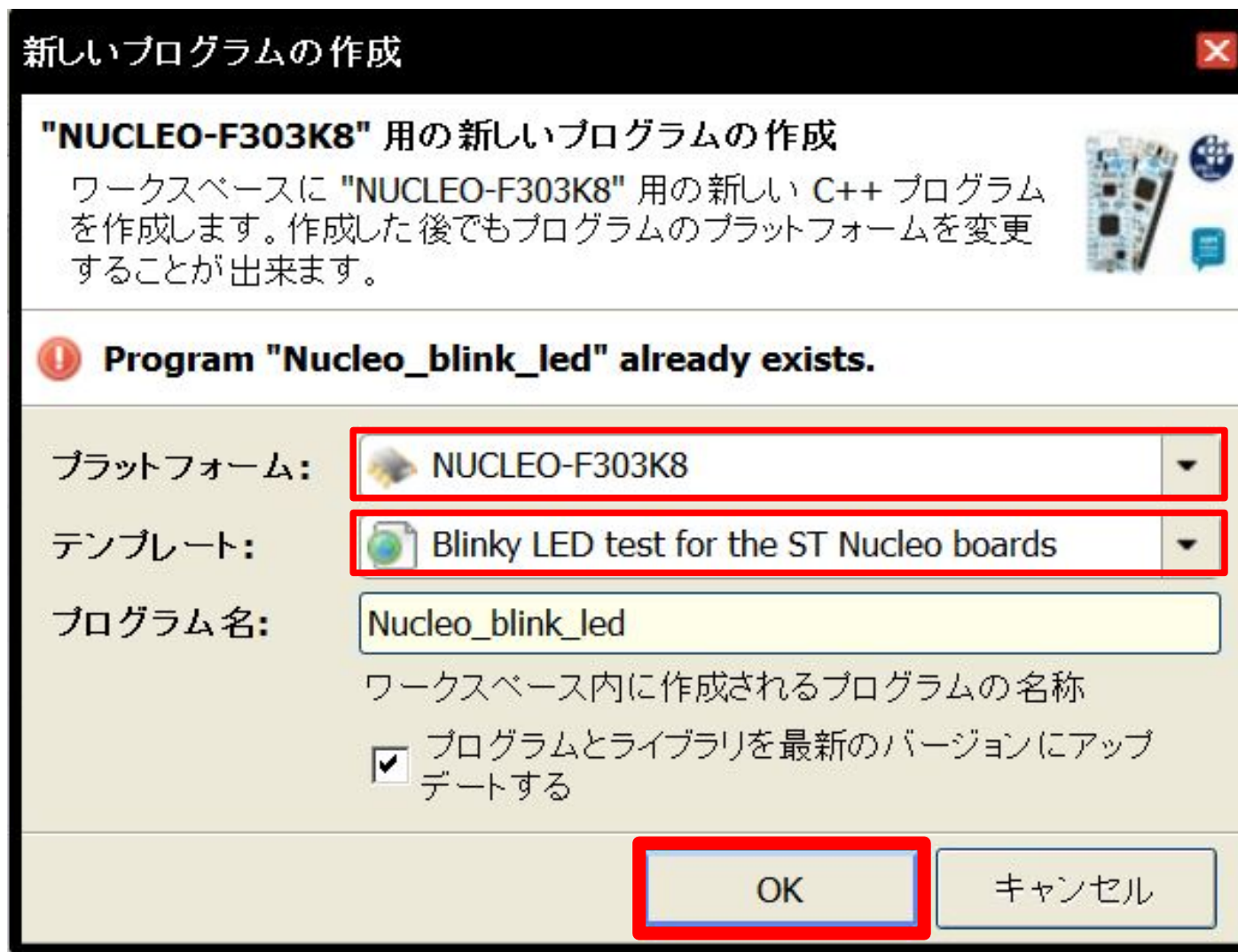
名称	タグ	変更日	詳細
lcd_test		08 May 2016	
Nucleo_blink_led		17 Mar 2016	
Nucleo_printf		25 Mar 2016	
Nucleo_pwm		4 days, 4 hours ago	
PSController_Sample		01 May 2016	
radio_controll		08 May 2016	

The right sidebar shows the details for a workspace named 'tory'. It indicates 'Total Programs: 6' and 'Modified: 4 days, 4 hours ago'. Below this, a table lists the recent changes:

ファイル名	変更日
main.cpp	4 days, 4 hours ago
main.cpp	08 May 2016
lcd_test.cpp	08 May 2016
PS_PAD.cpp	01 May 2016
main.cpp	25 Mar 2016
main.cpp	17 Mar 2016



## ⑤プラットフォーム:NUCLEO-F303K8 テンプレート:Blinkly LED を選択→OK



# ⑥main.cppをダブルクリック

The screenshot shows the mbed IDE interface for a project named "/Nucleo\_blink\_led". The left sidebar displays the "マイプログラム" (My Programs) tree, where "main.cpp" is highlighted. The main workspace shows a table of files:

名称	サイズ	種類	変更日
main.cpp	0.2 kB	C/C++ ソース ファイル	moments ago
mbed		ライブラリビルド	moments ago

A red box highlights the "main.cpp" file, and a black arrow points to it. The right sidebar shows the "プログラムの詳細" (Program Details) panel, which includes fields for Name, Creation Date, Last Change, Last Build, URL, Revision, and Status. The status is "uncommitted changes".

プログラムのコンパイル出力: Nucleo\_blink\_led

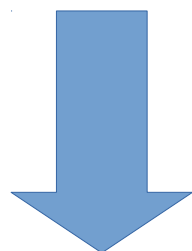
エラー番号   リソース   フォルダ   位置

## ⑦コンパイルをクリック もしくは Ctrl+D



## ⑧ ～.binファイルをダウンロード、mbedドライブへ保存

 Nucleo_blink_led_NUCLEO_F303K8.bin	2016/06/29 23:10	BIN ファイル
--	------------------	----------



保存(赤緑LED点滅が終了したら書き込み終了)



NODE\_F303K8 (D:)

空き領域 64.0 KB/72.0 KB

※保存先をmbedドライブに指定しておけば、コンパイルただけで自動で実行までできます

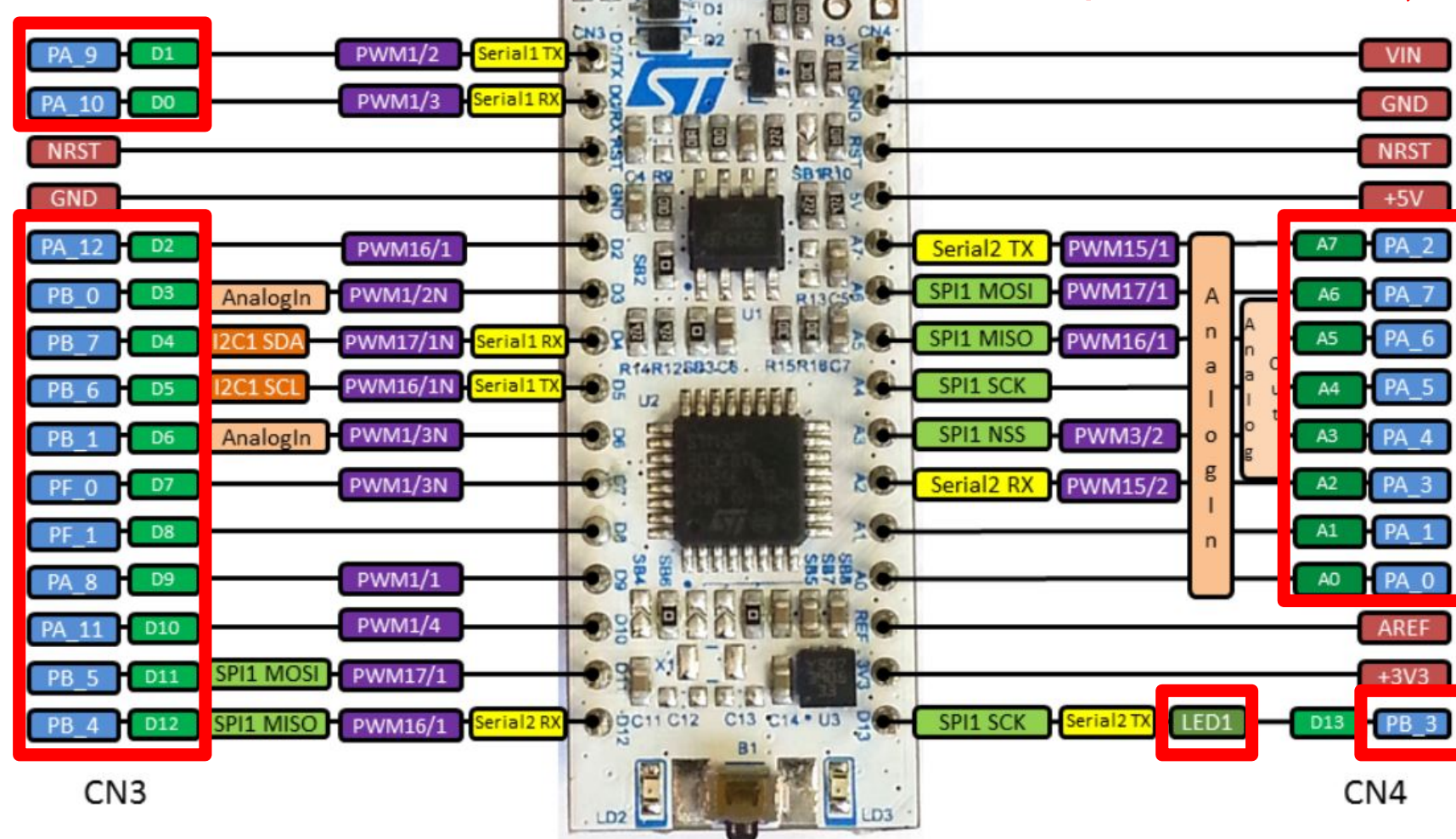
# LED点滅プログラム解説

```
1 #include "mbed.h" ← 必ず必要
2
3 DigitalOut myled(LED1); ← LED1ピンをmyledと名付け、Output機能で使うことを宣言
4
5 int main() {
6     while(1) { ← {}内を無限ループ
7         myled = 1; // LED is ON ← 1 = 3.3V
8         wait(0.2); // 200 ms
9         myled = 0; // LED is OFF ← 0 = 0V
10        wait(1.0); // 1 sec
11    }
12 }
```

# STM32F303K8ピンアサイン

life.augmented  
Nucleo F303K8

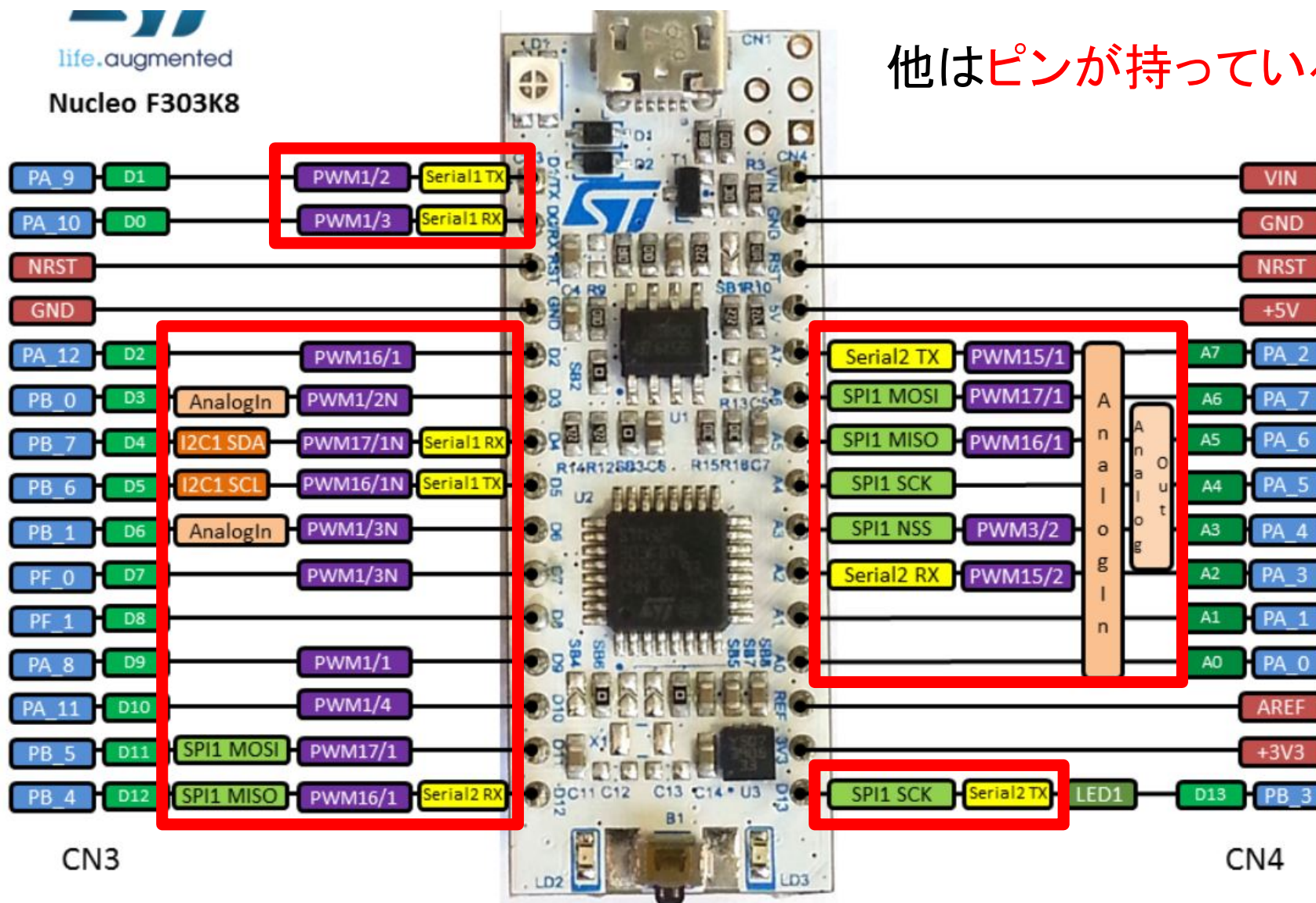
水色、緑色の番号は  
ピン名としてプログラムで使用可





# STM32F303K8ピンアサイン

他はピンが持っている機能の情報



# 目次

- mbedとは
- 使用するmbedマイコンボード
- サンプルプログラム書き込み
- PWM制御を使用したモータ速度制御
- PCとの通信



# モータにかかせないPWM制御とは①

- マイコンは”0”か”1”しか入力も出力もできない  
(今回使用するマイコンの場合は  $0=0[V]$ ,  $1=3.3[V]$ )
- モータの回転数を速くしたり遅くしたりするには  
電圧を細かく設定する必要がある  
→要するに0～1の間が必要になる



そこでPWM制御を使う

## モータにかかせないPWM制御とは②

- PWM = Pulse Width Modulation (パルス幅変調)
- パルス波のデューティ比を変化させて変調する変調方法です。そして、デューティ比とは周期的なパルス波を出したときの周期とパルス幅の比のことで、以下の式で表されます。

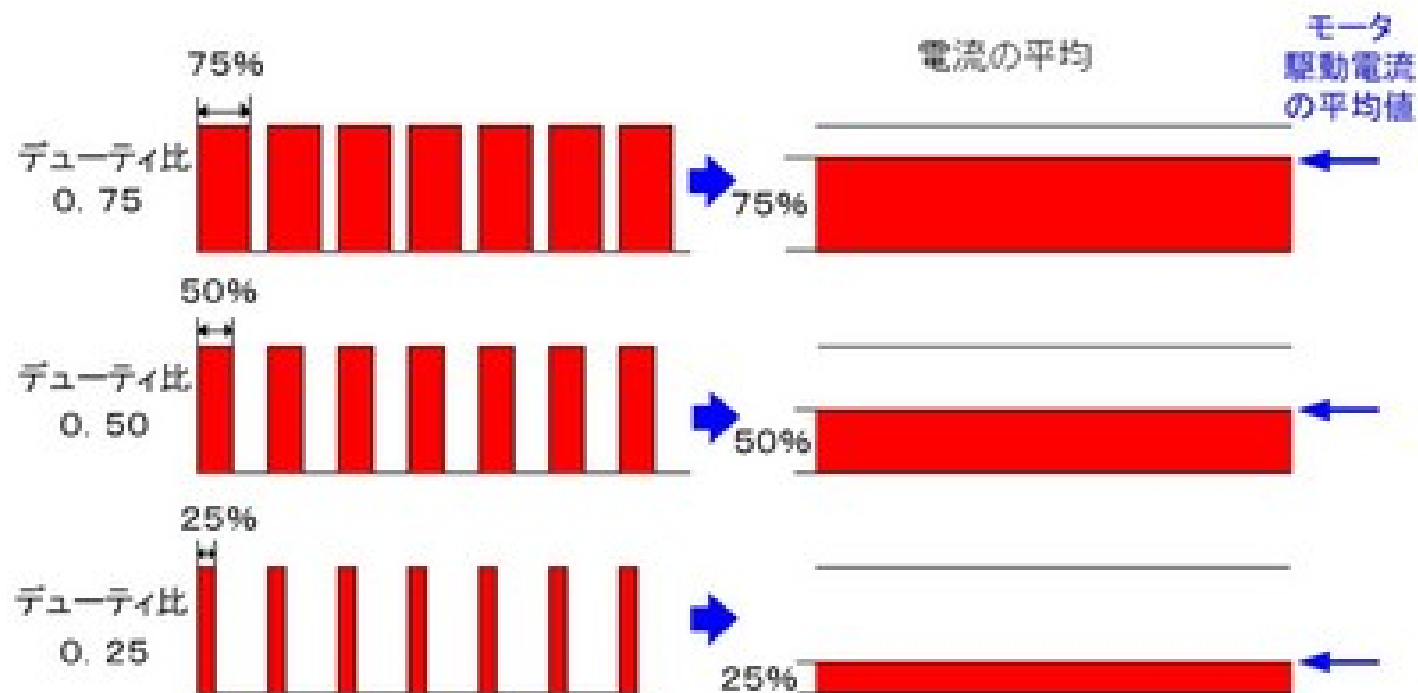
$$D = \tau / T$$

D : デューティ比

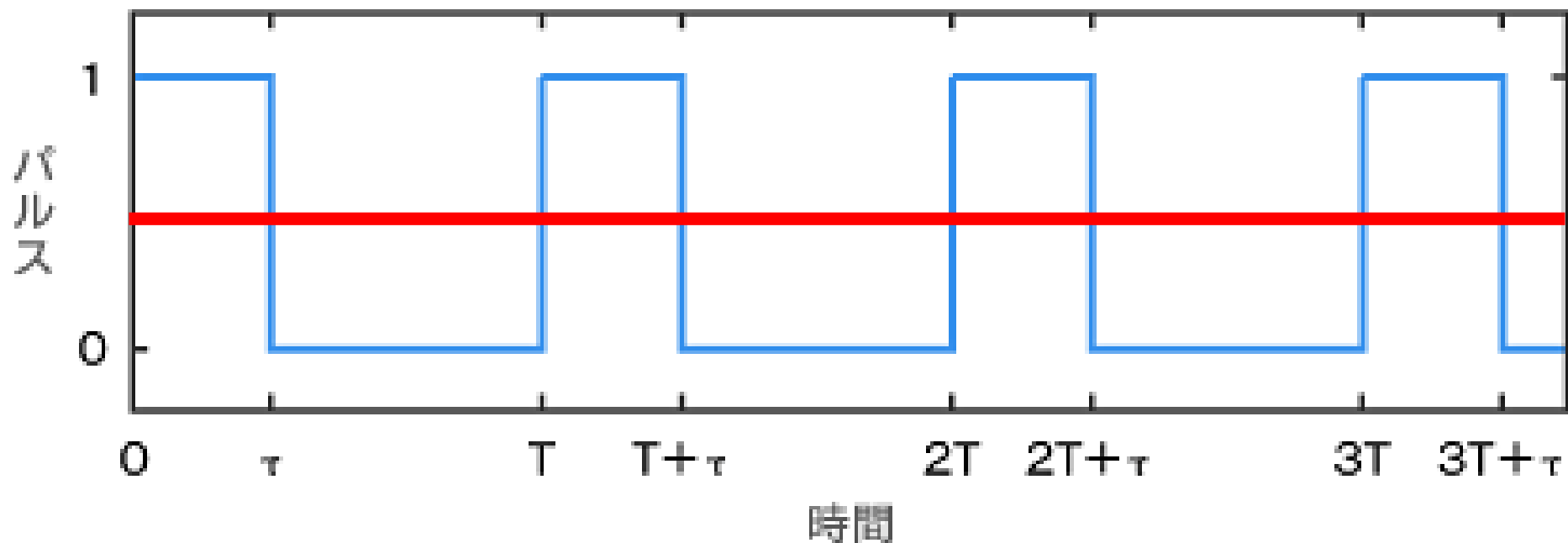
$\tau$  : パルス幅

T : 周期

1(High)と0(Low)をすばやく切り換えることで、  
中間量の電流が流れているときと同じ状態を作りだせる！



# PWMイメージ

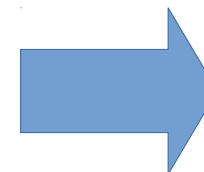


$$D = \tau / T$$

D : デューティー比  
 $\tau$  : パルス幅  
T : 周期

例えば・・・  
周期=10ms  
パルス幅=5ms  
デューティー比は0.5(50%)

つまり

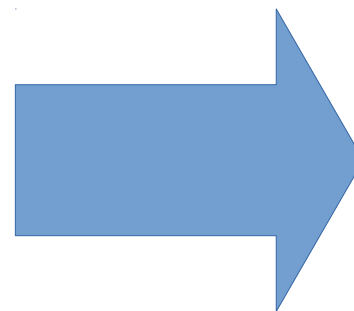


擬似的に  
 $3.3[\text{V}] \times 0.5 = 1.65[\text{V}]$   
を出している

# マイコンのPWM制御だけでモータは動かせるか

A. 動かせない。なぜなら...

- マイコンの出せる電圧、電流は...  
3.3V、数mA
- モータに加えたい電圧、電流は...  
~9V, 2A以上



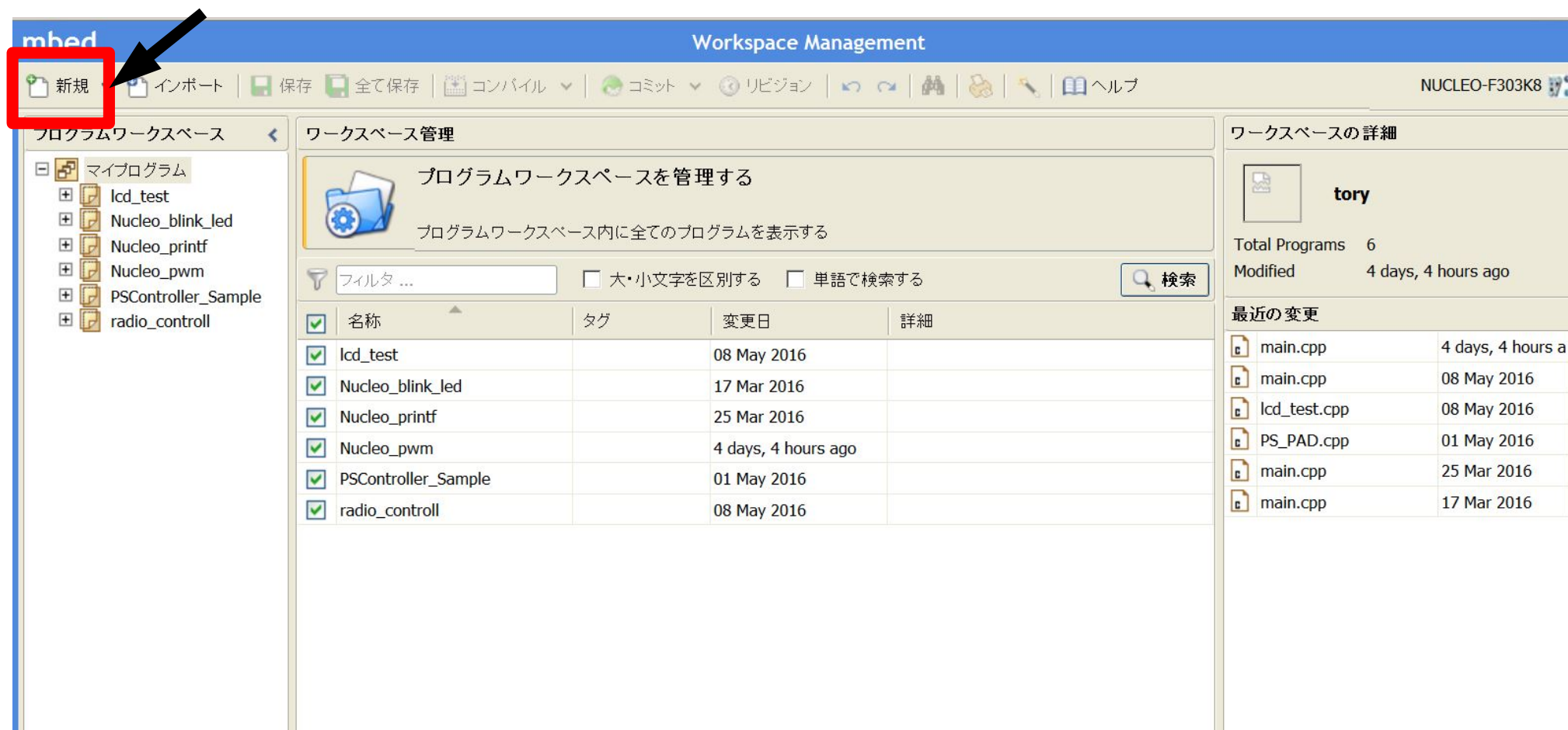
そこで  
モータドライバ  
を使う

## モータドライバの役割

- マイコンからの入力で、**電流電圧を増幅**させる
- 正転、逆転、ブレーキ、ストップ動作などができる



# ①左上の”新規”→”新しいプログラム”をクリック



The screenshot shows the mbed Workspace Management interface. The top toolbar contains buttons for '新規' (New), 'インポート' (Import), '保存' (Save), '全て保存' (Save All), 'コンパイル' (Compile), 'コミット' (Commit), 'リビジョン' (Revision), and 'ヘルプ' (Help). The '新規' button is highlighted with a red box and a black arrow.

The left sidebar shows a list of programs under 'マイプログラム' (My Programs):

- lcd\_test
- Nucleo\_blink\_led
- Nucleo\_printf
- Nucleo\_pwm
- PSController\_Sample
- radio\_controll

The central workspace management area shows a table of programs:

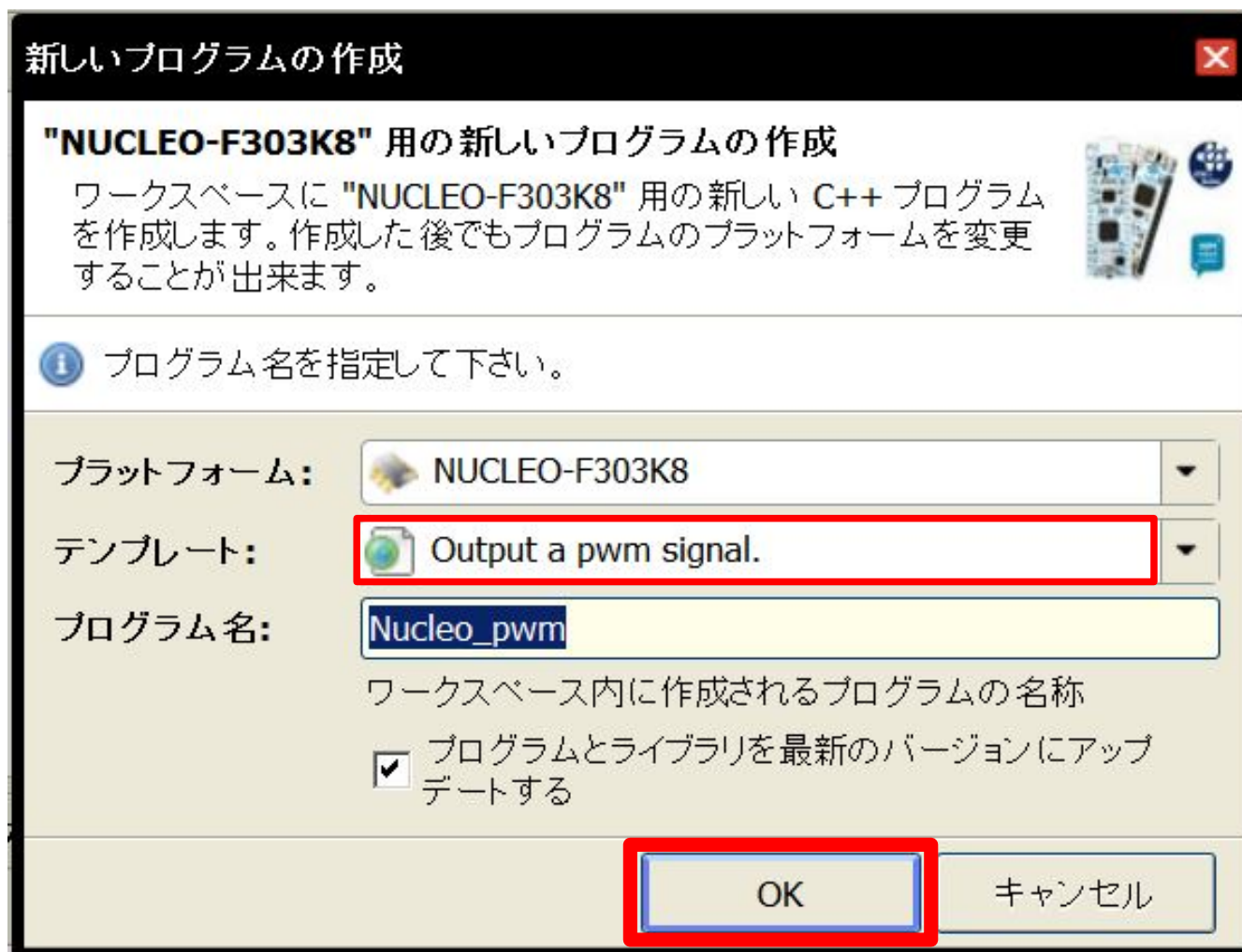
名称	タグ	変更日	詳細
lcd_test		08 May 2016	
Nucleo_blink_led		17 Mar 2016	
Nucleo_printf		25 Mar 2016	
Nucleo_pwm		4 days, 4 hours ago	
PSController_Sample		01 May 2016	
radio_controll		08 May 2016	

The right sidebar shows details for a workspace named 'tory':

- Total Programs: 6
- Modified: 4 days, 4 hours ago
- 最近の変更 (Recent Changes):

ファイル名	変更日
main.cpp	4 days, 4 hours ago
main.cpp	08 May 2016
lcd_test.cpp	08 May 2016
PS_PAD.cpp	01 May 2016
main.cpp	25 Mar 2016
main.cpp	17 Mar 2016

## ②テンプレート:Output a pwm signal を選択→OK





# ③main.cppをダブルクリック

mbed /Nucleo\_blink\_led

新規 インポート 保存 全て保存 コンパイル コミット リビジョン ヘルプ

プログラムワークスペース

マイプログラム

- lcd\_test
- Nucleo\_blink\_led
  - main.cpp
  - mbd
- Nucleo\_printf
- Nucleo\_pwm
- PSController\_Sample
- radio\_controll

プログラム: /Nucleo\_blink\_led

フィルタ ... ☐ 大・小文字を区別する ☐ 単語で検索する 検索

名称	サイズ	種類	変更日
main.cpp	0.2 kB	C/C++ ソース ファイル	moments ago
mbd		ライブラリビルド	moments ago

プログラムの詳細

概要 ビルド

名称 Nucleo\_blink\_led  
作成日 moments ago  
最終変更 moments ago  
最終ビルド Never  
URL n/a  
リビジョン no revisions  
ステータス uncommitted changes

ドキュメントは旧版です

アップ データ コミット リビジョン

エクスポート パブリッシュ ホームページ

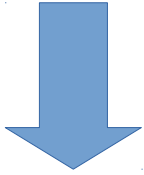
説明

プログラムのコンパイル出力: Nucleo\_blink\_led

詳細 エラー番号 リソース フォルダ 位置

# PWMプログラム解説

```
1 #include "mbed.h"
2 PwmOut mypwm(PWM_OUT); PWM_OUTピンをmypwmと名付け、pwm機能で使うことを宣言
3 DigitalOut myled(LED1); LED1ピンをmyledと名付け、Output機能で使うことを宣言
4
5 int main() {
6
7     mypwm.period_ms(10); PWMの周期を10msに設定
8     mypwm.pulsewidth_ms(1); PWMのON時間を1msに設定(デューティー比10%)
9
10    printf("pwm set to %.2f %%\n", mypwm.read() * 100);
11
12    while(1) {
13        myled = !myled;
14        wait(1);
15    }
16 }
```



デューティー比10%なので、  
 $3.3[V] \times 0.1 = 0.33[V]$  が出力される

# ピン名がどのピンを指定しているかわからないときは...

ここにカーソルを合わせると...

```
1 #include "mbed.h"
2
3 PwmOut mypwm(PWM_OUT);
4
5 DigitalOut myled(LED1);
6
7 int main() {
8
9     mypwm.period_ms(10);
10    mypwm.pulsewidth_ms(1);
11
12    printf("pwm set to %.2f %%\n", mypwm.read() * 100);
13
14    while(1) {
15        myled = !myled;
16        wait(1);
17    }
18 }
```

Enumerator **PWM\_OUT**  
Enum: anon258  
PWM\_OUT = PA\_8,

定義を表示してくれる

## テスターで電圧を計測してみる

テスターの赤い方でマイコンのPA\_8番ピンを、  
黒い方でGNDピンを触ると、電圧が表示される

正しくプログラムが動作していれば0.3V程度が  
出力されているはず。

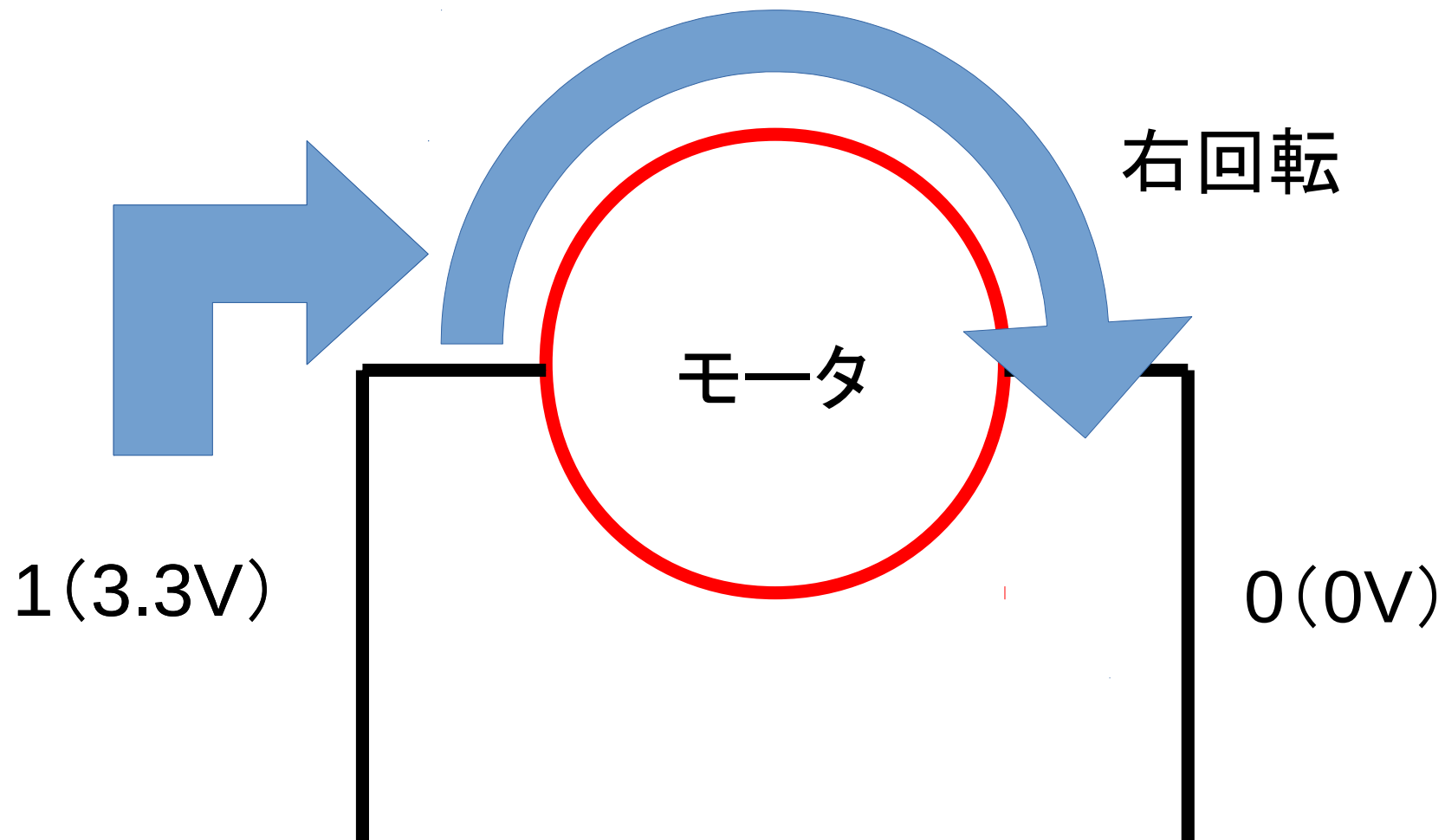
# 車輪ロボットのPWM制御プログラム

```
1 #include "mbed.h"
2 PwmOut m1_cw(PA_11);
3 PwmOut m1_ccw(PA_8);
4 PwmOut m2_cw(PB_4);
5 PwmOut m2_ccw(PB_5);
6 int main() {
7     while(1) {
8         //右車輪
9         m1_cw = 1; // 0~1 (プログラム数値) = 0V~3.3V (マイコン電圧) = 0V~6V (モータ電圧)
10        m1_ccw = 0;
11        //左車輪
12        m2_cw = 0.5; //
13        m2_ccw = 0;
14        //pc.printf("%f\n",psd_sensor);
15    }
16 }
```

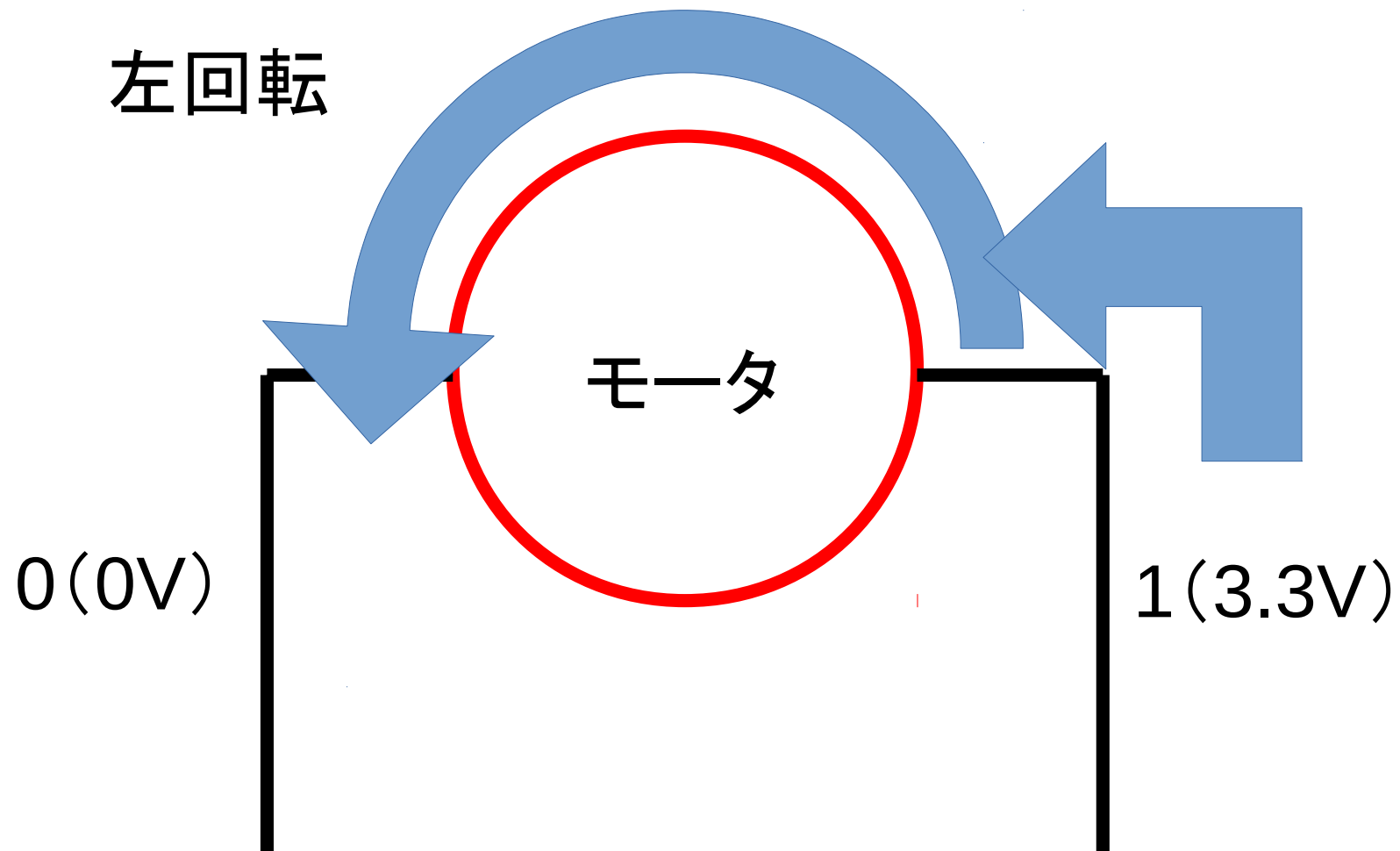
} 直接デューティ比を設定可能  
0~1で速度が変化する

※周期を指定しない場合  
自動で20msに設定される。

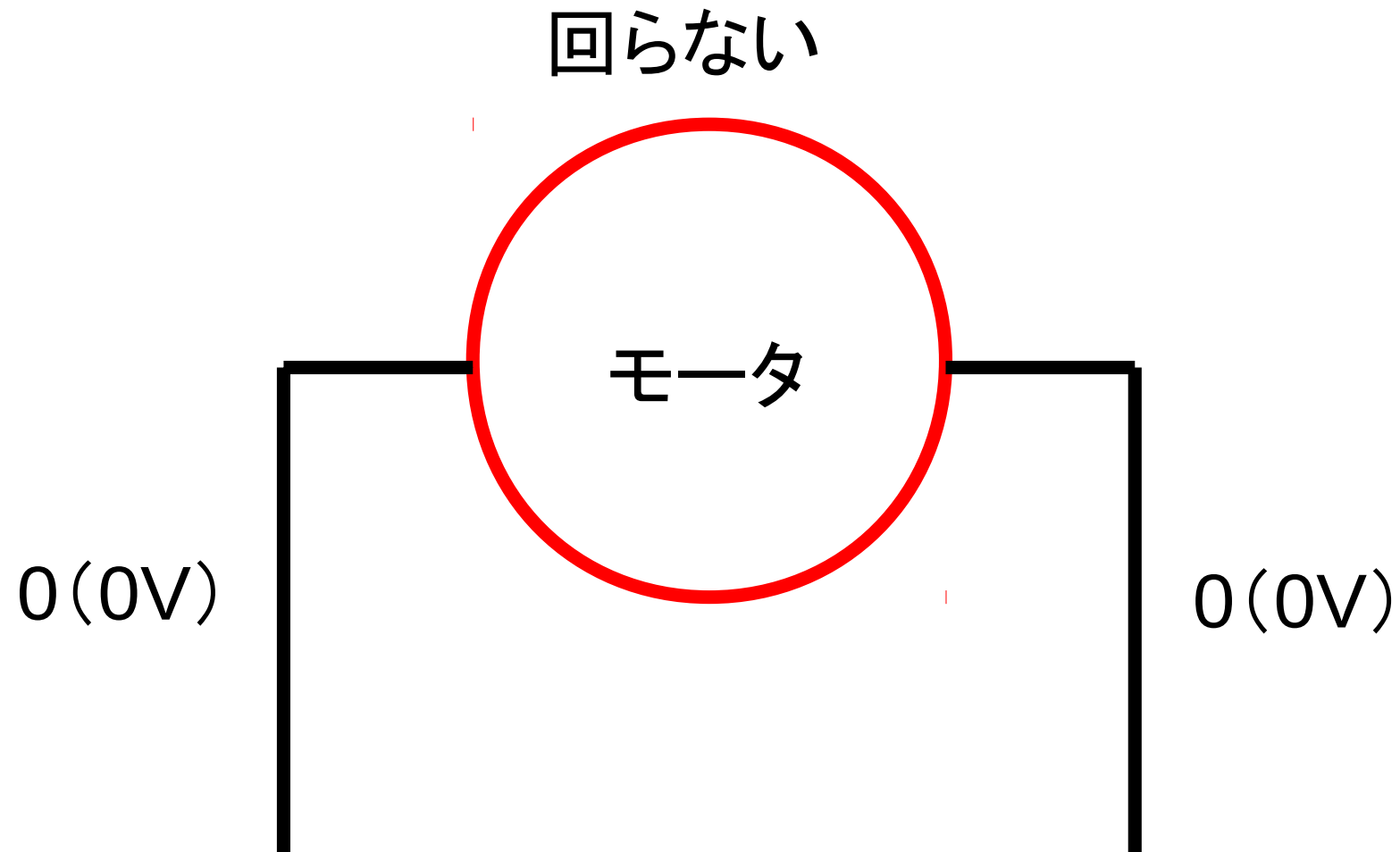
# モータの回る仕組み



# モータの回る仕組み



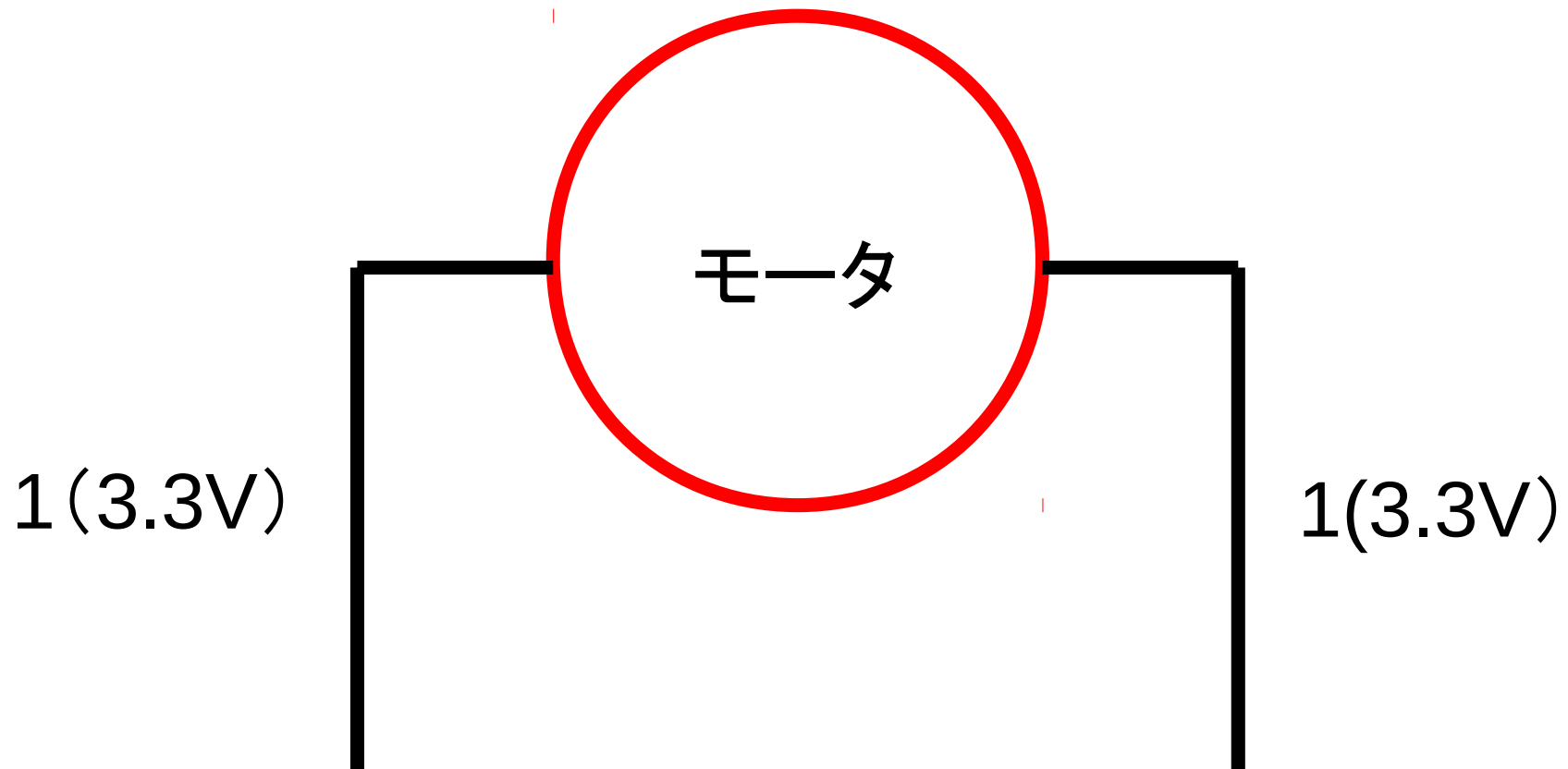
# モータの回る仕組み





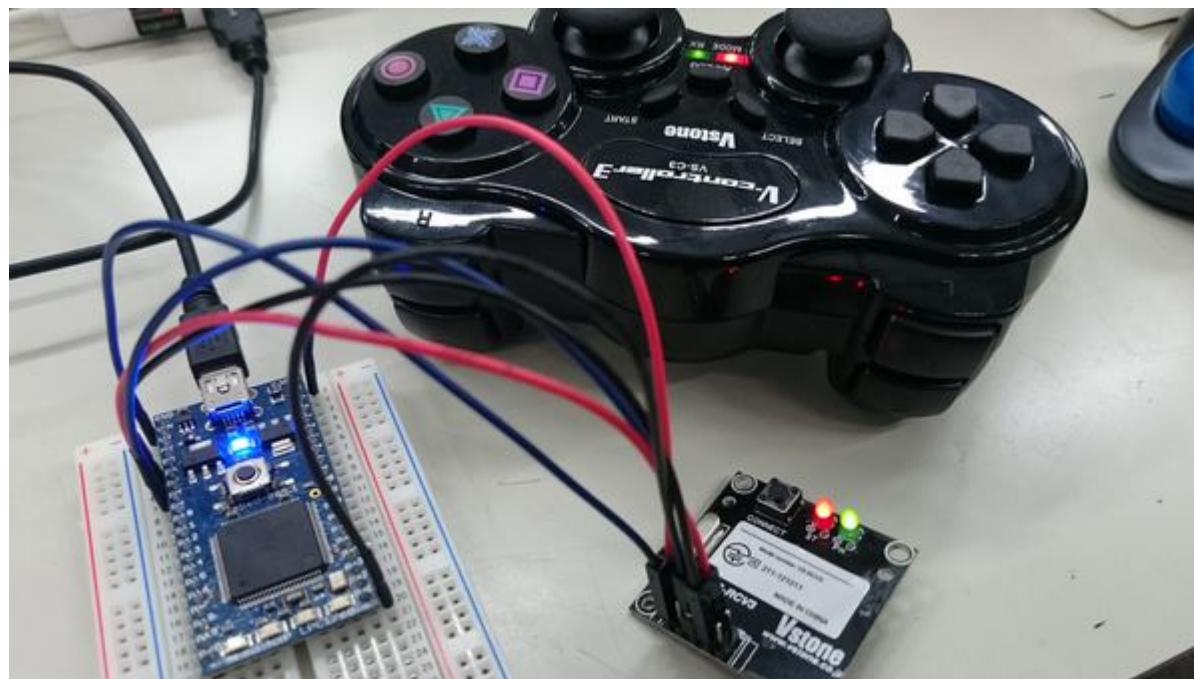
# モータの回る仕組み

回らない



デューティー比を変更して、左右の  
車輪の速度が変わることを確認する

# コントローラを使う



# “mbed VS-C3” で検索

mbed CS-C3 × 🔍 検索 + 条件指定

約25,400件

検索ツール ▼

[mbedでVS-C3を使ってみた | Memoteki - About tiryoh.com](#)

[tiryoh.com](#) > ホーム > マイコン > LPC - キャッシュ

2015年9月24日 - mbedでvstoneのコントローラ**VS-C3**を使ってみました。その使い方をメモ。このコントローラはPS2のコントローラ(DUALSHOCK2)と(信号の)互換性があるので DUALSHOCK2でも同様にmbedで扱えると思います。利用するライブラリ ...

← ここをクリック

# 以下のリンク先をクリック

## Memoteki

自分のメモは自分にとって最もわかりやすい教科書であると信じてたい。

About tiryoh.com

About this blog

ホーム > マイコン > LPC >

## mbedでVS-C3を使ってみた

📅 2015/09/24 🔄 2016/04/23

B! 0

👍 いいね! 0

🐦 ツイート

📬 LINEで送る

👛 Pocket

mbedでvstoneのコントローラVS-C3を使ってみました。その使い方をメモ。

このコントローラはPS2のコントローラ（DUALSHOCK2）と（信号の）互換性があるので  
DUALSHOCK2でも同様にmbedで扱えると思います。

利用するライブラリはSugaKoubouさんのPS\_PADです。

[https://developer.mbed.org/users/okini3939/code/PS\\_PAD/](https://developer.mbed.org/users/okini3939/code/PS_PAD/)

サンプルプログラムはこちら。

[https://developer.mbed.org/users/ds074704261/code/PSController\\_Sample/](https://developer.mbed.org/users/ds074704261/code/PSController_Sample/)

接続方法はサンプルプログラムを見ればわかると思います。

← ここをクリック

# “Import into Compiler”をクリック

ここをクリック



Suga koubou / PS\_PAD

PlayStation Controller library

**Dependents:** PSController\_Sample cobra\_stick\_PS2 basehybrid\_PSPAD Base\_Hybrid\_V3 ... more

Home

History

Graph

API Documentation

Wiki

Pull Requests

## PlayStation Controller

SONY プレイステーション2用コントローラーを mbed に接続して使えるライブラリです。

SPI 機能を使います。

サードパーティ製や、VSTONE製のコントローラー (VS-C1、VS-C3) も使用できます。

Repository toolbox

Import into Compiler

Export to desktop IDE

+ Follow

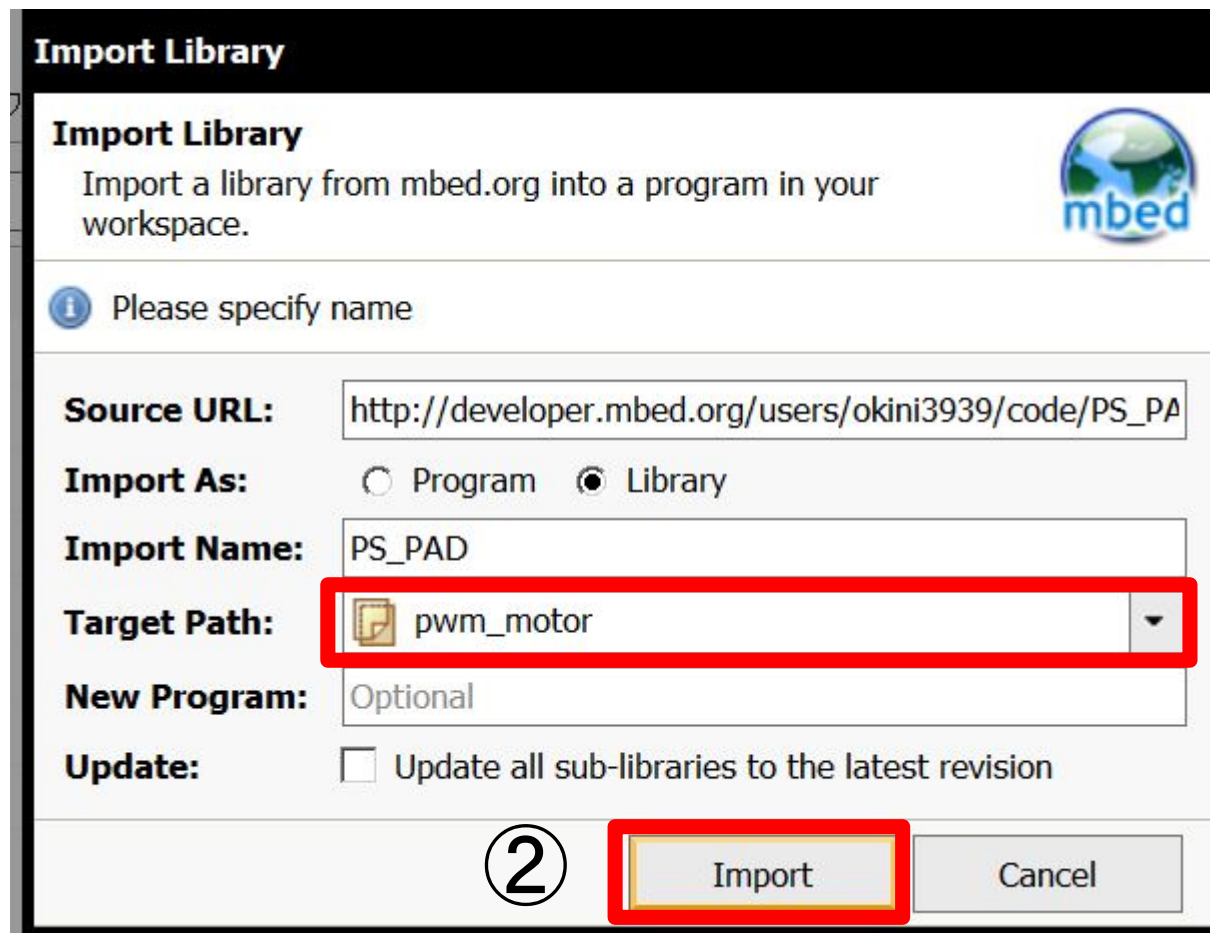
Embed url:

<<library /users/okini3939/c

Clone repository to  
desktop:

hg clone https://tory@develc


# Target pathを指定し、Importをクリック



**Import Library**

Import a library from mbed.org into a program in your workspace.




 Please specify name

**Source URL:**


**Import As:** ☐ Program ☒ Library

**Import Name:**

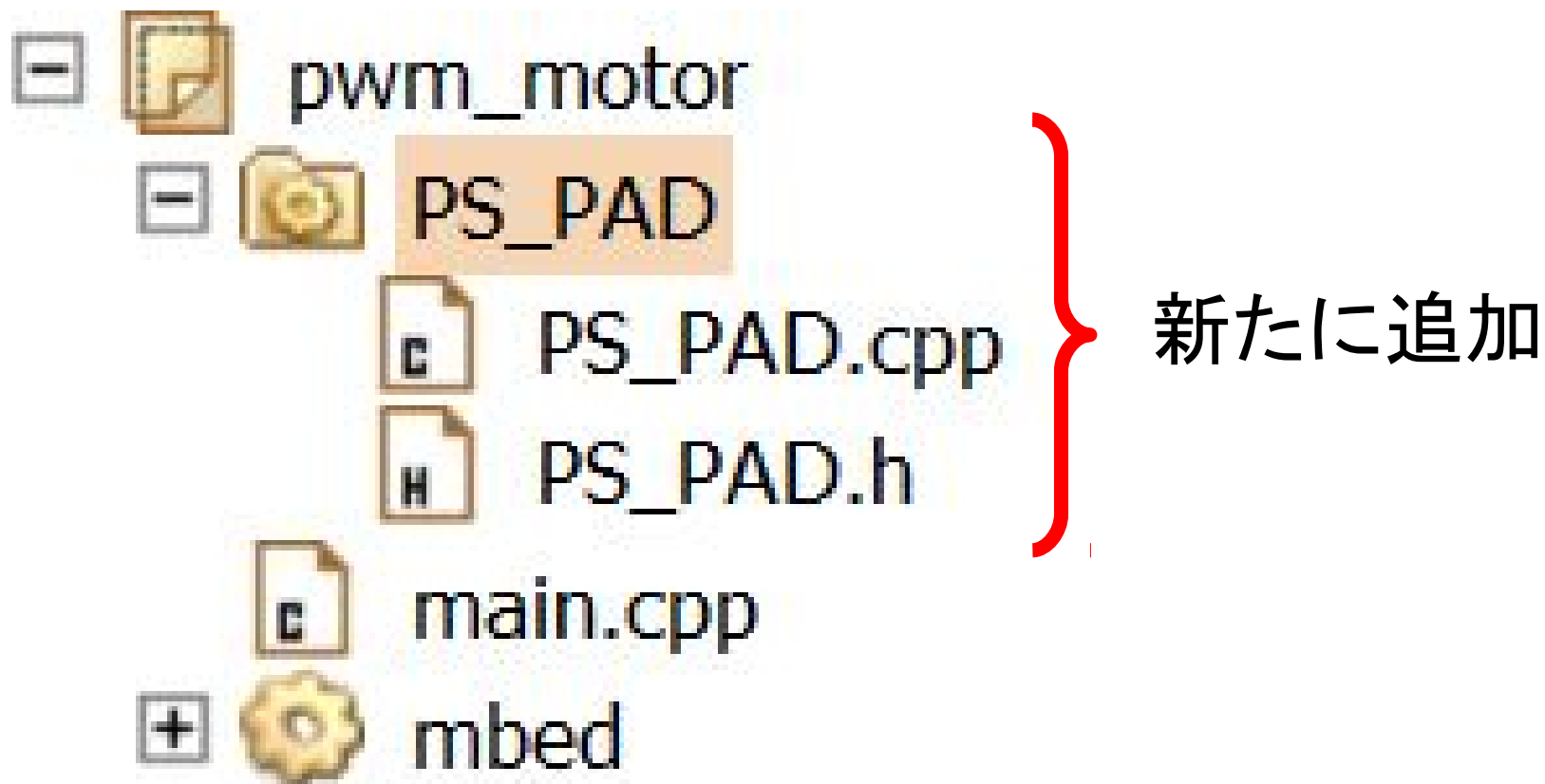
**Target Path:**   

**New Program:**

**Update:** ☐ Update all sub-libraries to the latest revision

**Buttons:**  

# ライブラリがインポートされていればOK





```
1 #include "mbed.h"
2 #include "PS_PAD.h" ← ライブラリ読み込み
3
4 PS_PAD vsc3(PA_7, PA_6, PA_5, PA_4); ← 通信ピン番号指定
5
6 PwmOut m1_cw(PA_11);
7 PwmOut m1_ccw(PA_8); ← PWM出力ピン指定(モータ)
8 PwmOut m2_cw(PB_4);
9 PwmOut m2_ccw(PB_5);
10
11 float r_y = 0, l_y = 0; ← アナログスティック値用変数
12 int start_signal = 0; ← STARTボタン認識用変数
13
14 int main() {
15     vsc3.init(); ← コントローラ初期化関数
16     while(1) {
17         vsc3.poll(); ← コントローラ初期化関数
18         if(vsc3.read(PS_PAD::PAD_START) == 1) start_signal = 1; ← STARTボタンで開始
19         if(start_signal == 1) {
20             r_y = vsc3.read(PS_PAD::ANALOG_RY); ← STARTボタン押されたら、読み込み開始
21             l_y = vsc3.read(PS_PAD::ANALOG_LY); ← (r_y, l_yにコントローラからの値が入力される)
22         }
23         wait(0.05); ← 0.05秒待つ(正常に通信するために必ず必要)
24     }
25 }
```

```
1 #include "mbed.h"
2 #include "PS_PAD.h"
3
4 PS_PAD vsc3(PA_7, PA_6, PA_5, PA_4);
5
6 PwmOut m1_cw(PA_11);
7 PwmOut m1_ccw(PA_8);
8 PwmOut m2_cw(PB_4);
9 PwmOut m2_ccw(PB_5);
10
11 float r_y = 0, l_y = 0;
12 int start_signal = 0;
13
14 int main() {
15     vsc3.init();
16     while(1) {
17         vsc3.poll();
18         if(vsc3.read(PS_PAD::PAD_START) == 1) start_signal = 1;
19         if(start_signal == 1) {
20             r_y = vsc3.read(PS_PAD::ANALOG_RY);
21             l_y = vsc3.read(PS_PAD::ANALOG_LY);
22         }
23         wait(0.05);
24     }
25 }
```

※ r\_y, l\_y にコントローラからの値が入ってくるので、PCに表示させたい

```
1 #include "mbed.h"
2 #include "PS_PAD.h"
3
4 Serial pc(PA_2, PA_3); ← 追加(ピン指定, PA_2,PA_3はUSBに接続されている)
5 PS_PAD vsc3(PA_7, PA_6, PA_5, PA_4);
6
7 PwmOut m1_cw(PA_11);
8 PwmOut m1_ccw(PA_8);
9 PwmOut m2_cw(PB_4);
10 PwmOut m2_ccw(PB_5);
11
12 float r_y = 0, l_y = 0;
13 int start_signal = 0;
14
15 int main() {
16     vsc3.init();
17     while(1) {
18         vsc3.poll();
19         if(vsc3.read(PS_PAD::PAD_START) == 1) start_signal = 1;
20         if(start_signal == 1) {
21             r_y = vsc3.read(PS_PAD::ANALOG_RY);
22             l_y = vsc3.read(PS_PAD::ANALOG_LY);
23         }
24         wait(0.05);
25         pc.printf("%f %f\n", r_y, l_y); ← 追加(PCにr_y,l_yを表示させる)
26     }
27 }
```

## PCとの通信

- センサやモータなどの値を  
PC画面に表示させて確認したい

# ダウンロードするもの

- Windowsの場合  
STLINK Driver  
Teraterm(ネットから検索してダウンロードしてください)
- Macの場合  
特になし(標準で入っている”ターミナル”を使用します)

# ST LINK Driverダウンロード～インストール (windowsのみ)

①以下サイトにアクセス

<http://www.st.com/en/embedded-software/stsw-link009.html>

② get softwareをクリック

## GET SOFTWARE

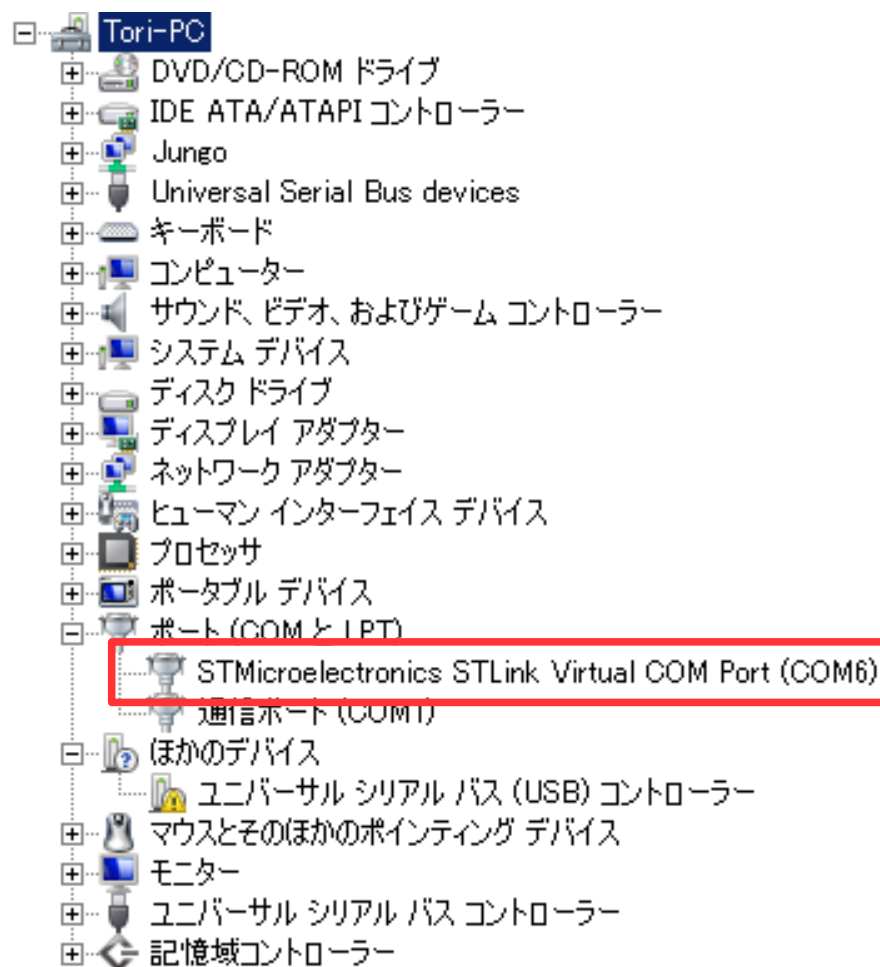
Part Number ▲	Software Version ◆	Marketing Status ◆	Supplier ◆	Order from ST ◆
STSW-LINK009	1.02	Active	ST	<a href="#">Get Software</a>

③アカウントを登録してダウンロード、インストール

x86が32ビットPC用

x64が64ビットPC用

# コントロールパネル→デバイスマネージャで 以下のようになっていればOK

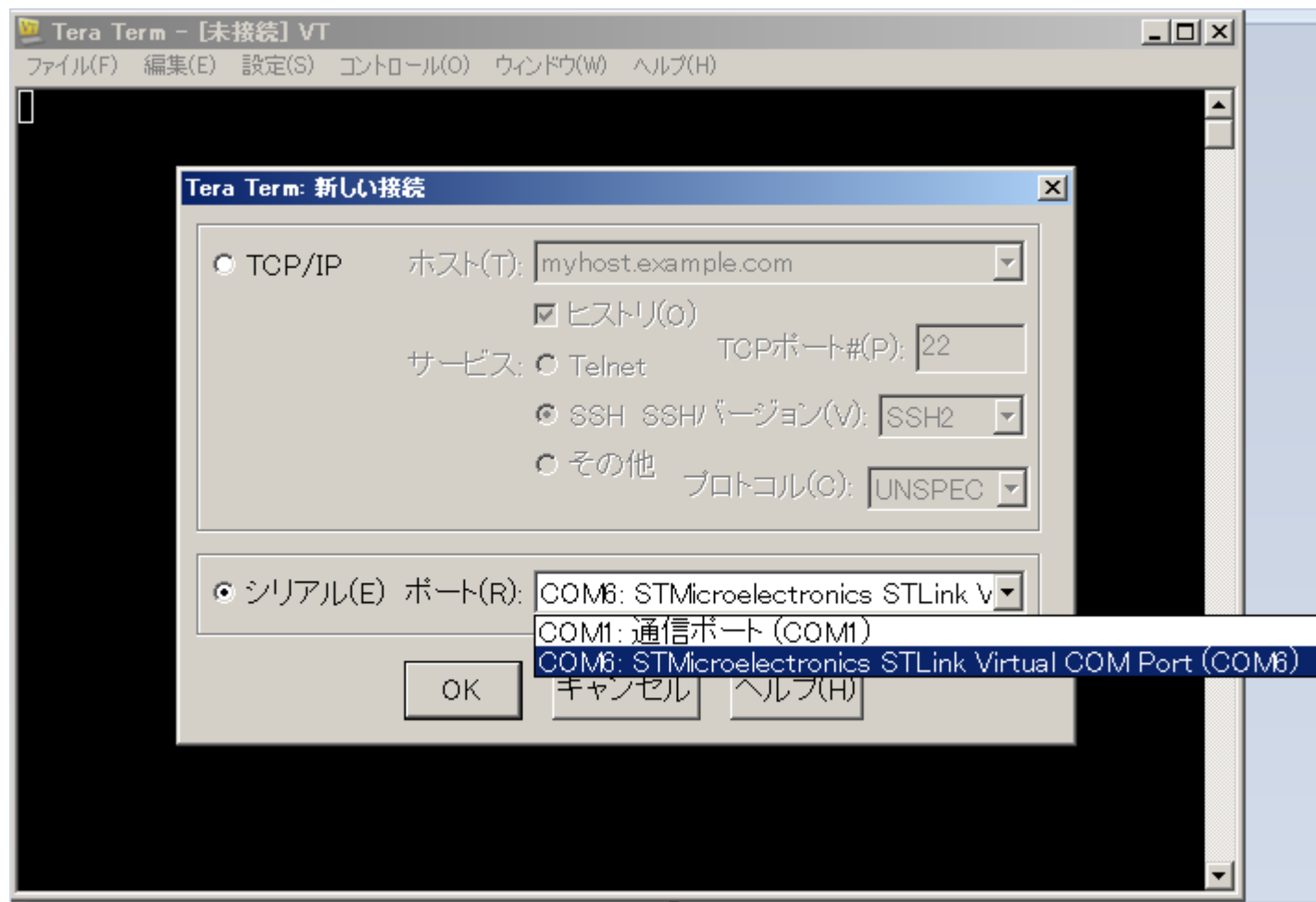


マイコンをPCに接続した時  
こうなっていればOK

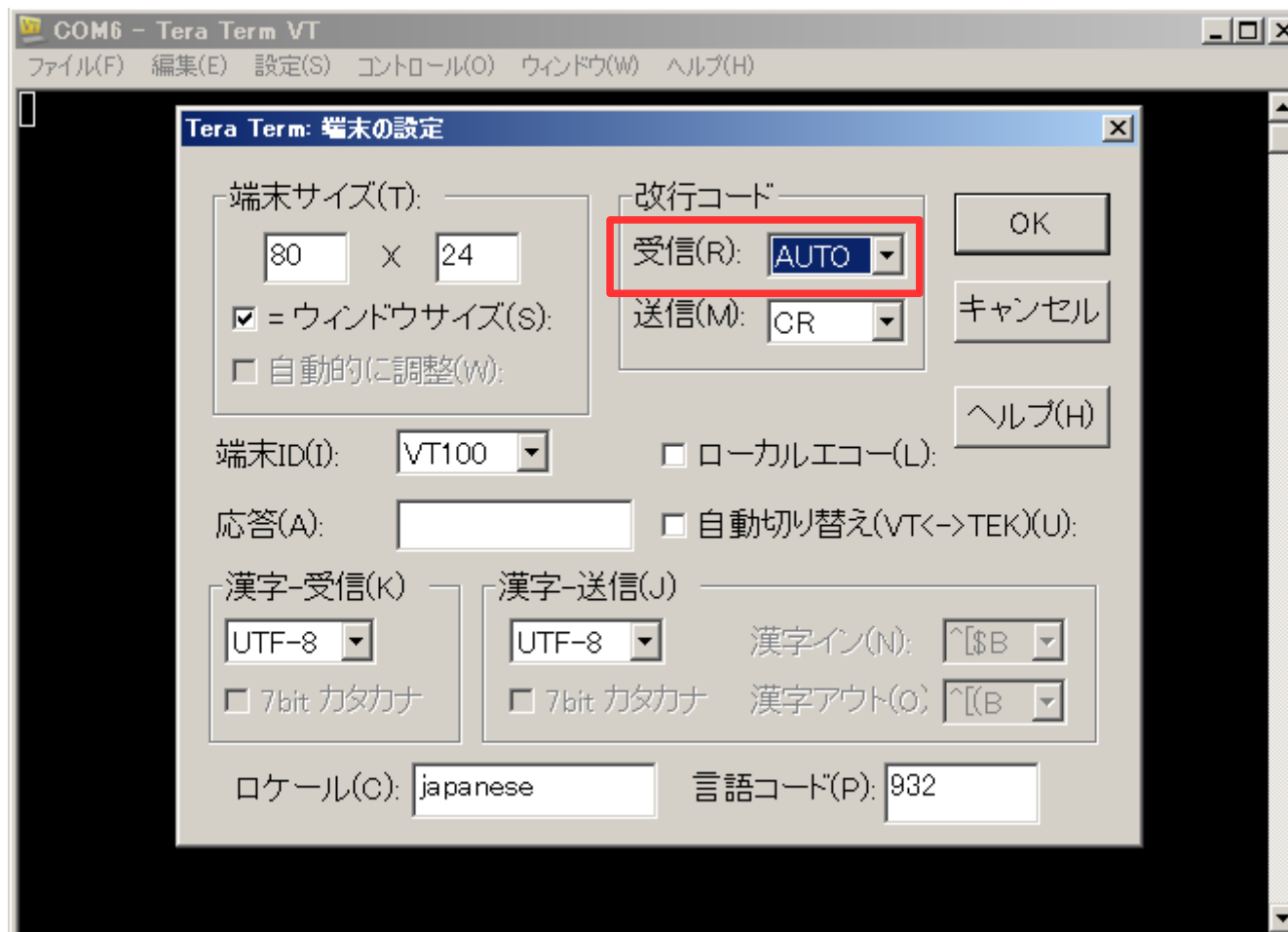
# TeraTerm使い方 (windowsのみ)



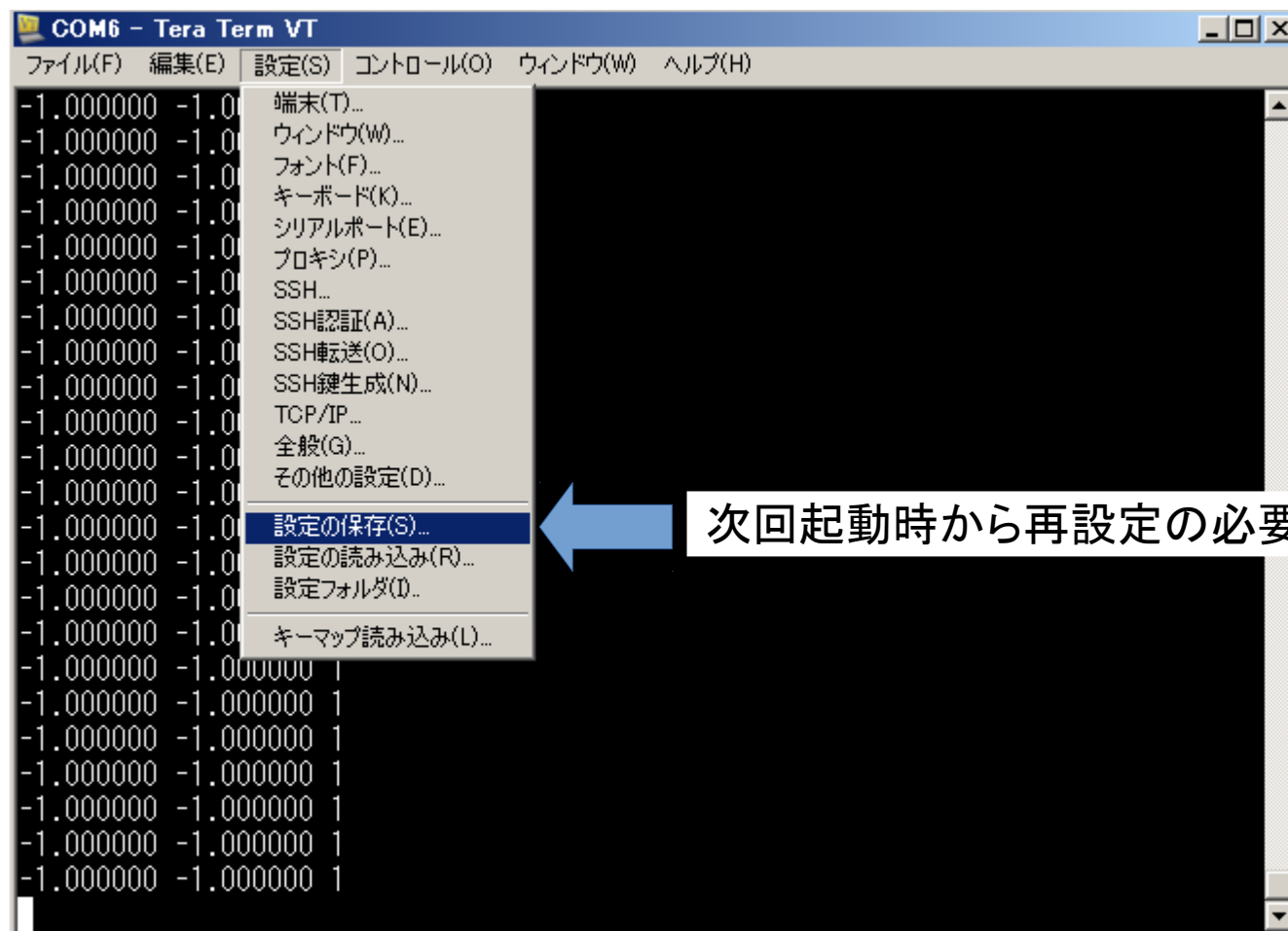
# STMicroelectronicsを選択



# 改行が正常に行えないため、 設定タブ→端末→受信を”AUTO”に設定



# 設定タブ→設定の保存

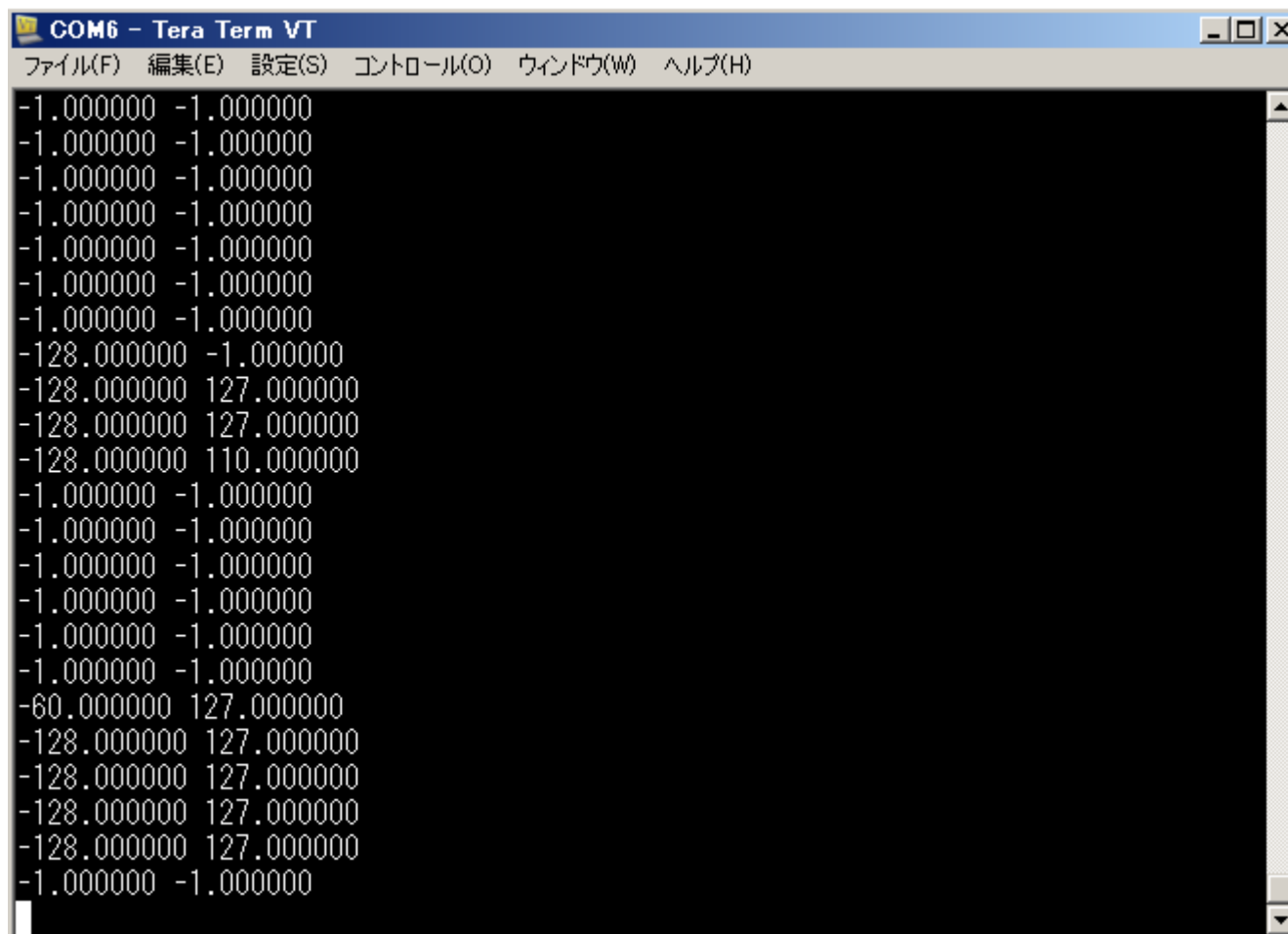


次回起動時から再設定の必要なくなる

# プログラム

- `Serial pc(PA_2, PA_3);` ←メイン文の前に追加
- `pc.printf("%~",表示させたい変数);`

# ターミナルソフトを使って、 値の変化を確認する



The screenshot shows a terminal window titled "COM6 - Tera Term VT". The menu bar includes "ファイル(F)", "編集(E)", "設定(S)", "コントロール(O)", "ウインドウ(W)", and "ヘルプ(H)". The terminal displays a list of numerical values, likely representing sensor data or system parameters, arranged in two columns. The values are as follows:

-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-128.000000	-1.000000
-128.000000	127.000000
-128.000000	127.000000
-128.000000	110.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-1.000000	-1.000000
-60.000000	127.000000
-128.000000	127.000000
-128.000000	127.000000
-128.000000	127.000000
-128.000000	127.000000
-1.000000	-1.000000

# コントローラからの値の変化に応じて 車輪速度を制御してみよう

```

1 #include "mbed.h"
2 #include "PS_PAD.h"
3
4 Serial pc(PA_2, PA_3);
5 PS_PAD vsc3(PA_7, PA_6, PA_5, PA_4);
6
7 PwmOut m1_cw(PA_11);
8 PwmOut m1_ccw(PA_8);
9 PwmOut m2_cw(PB_4);
10 PwmOut m2_ccw(PB_5);
11
12 float r_y = 0, l_y = 0;
13 int start_signal = 0;
14
15 int main() {
16     vsc3.init();
17     while(1) {
18         vsc3.poll();
19         if(vsc3.read(PS_PAD::PAD_START) == 1) start_signal = 1;
20         if(start_signal == 1) {
21             r_y = vsc3.read(PS_PAD::ANALOG_RY);
22             l_y = vsc3.read(PS_PAD::ANALOG_LY);
23         }
24         wait(0.05);
25         pc.printf("%f %f\n", r_y, l_y);
26     }
27 }

```

## ヒント

- ・このへんに追加
- ・ $r\_y, l\_y$ の変化によって、車輪速度を変える
- ・車輪の速度を変えるにはPWM制御を使う

# 車輪ロボットのPWM制御プログラム

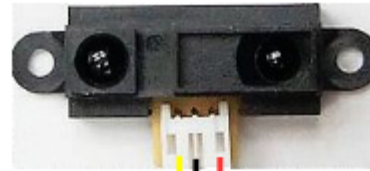
```
1 #include "mbed.h"
2 PwmOut m1_cw(PA_11);
3 PwmOut m1_ccw(PA_8);
4 PwmOut m2_cw(PB_4);
5 PwmOut m2_ccw(PB_5);
6 int main() {
7     while(1) {
8         //右車輪
9         m1_cw = 1; // 0~1 (プログラム数値) = 0V~3.3V (マイコン電圧) = 0V~6V (モータ電圧)
10        m1_ccw = 0;
11        //左車輪
12        m2_cw = 0.5; //
13        m2_ccw = 0;
14        //pc.printf("%f\n",psd_sensor);
15    }
16 }
```

} 直接デューティ比を設定可能  
0~1で速度が変化する

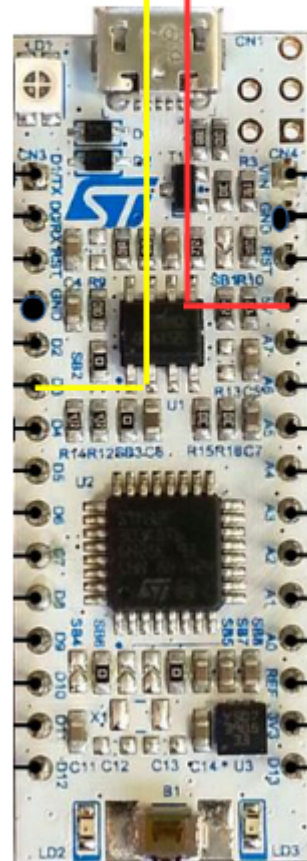
※周期を指定しない場合  
自動で20msに設定される。

# PSDセンサを使った衝突回避





\* 赤と黒の線に注意しながら配線



```
1 #include "mbed.h"
2 #include "PS_PAD.h"
3
4 Serial pc(PA_2, PA_3);
5 PS_PAD vsc3(PA_7, PA_6, PA_5, PA_4);
6
7 PwmOut m1_cw(PA_11);
8 PwmOut m1_ccw(PA_8);
9 PwmOut m2_cw(PB_4);
10 PwmOut m2_ccw(PB_5);
11
12 AnalogIn left(PA_0);
13 AnalogIn center(PA_1);
14 AnalogIn right(PB_1);
15 AnalogIn psd(PB_0); ←変更点①センサからのアナログ電圧入力ピン番号指定
16
17 float r_v = 0, l_v = 0;
18 double psd_sensor; ←変更点②センサ値用の変数宣言
19 int start_signal = 0;
```

```
38 int main() {
39     vsc3.init();
40     wait(0.001);
41     while(1) {
42
43         psd_sensor = psd; ←変更点③代入(psd_sensorにセンサからの値が入力される)
```

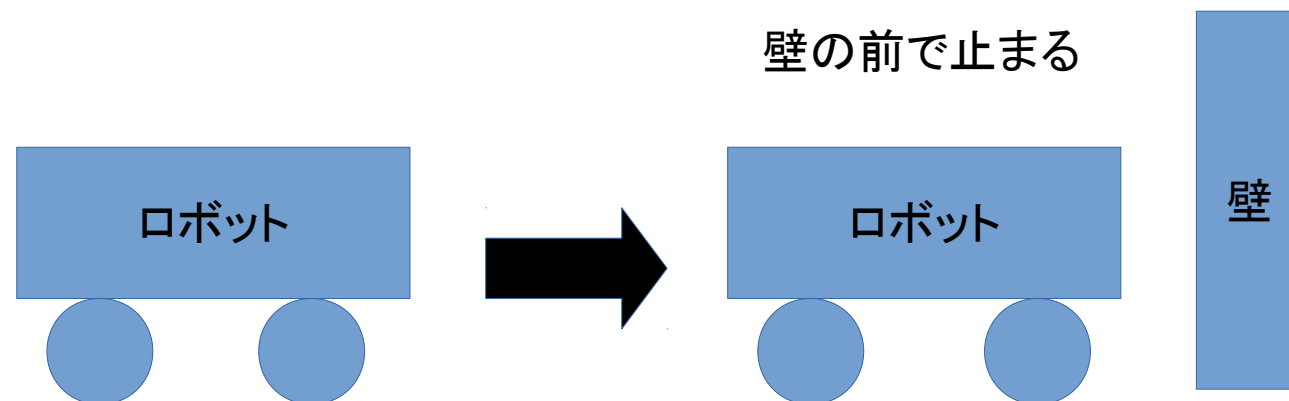
# PSDセンサからの値を使って 衝突回避させてみる

ヒント①:

PSDセンサの値をPCに表示し,壁に近づいた時の数値を確認する

ヒント②:

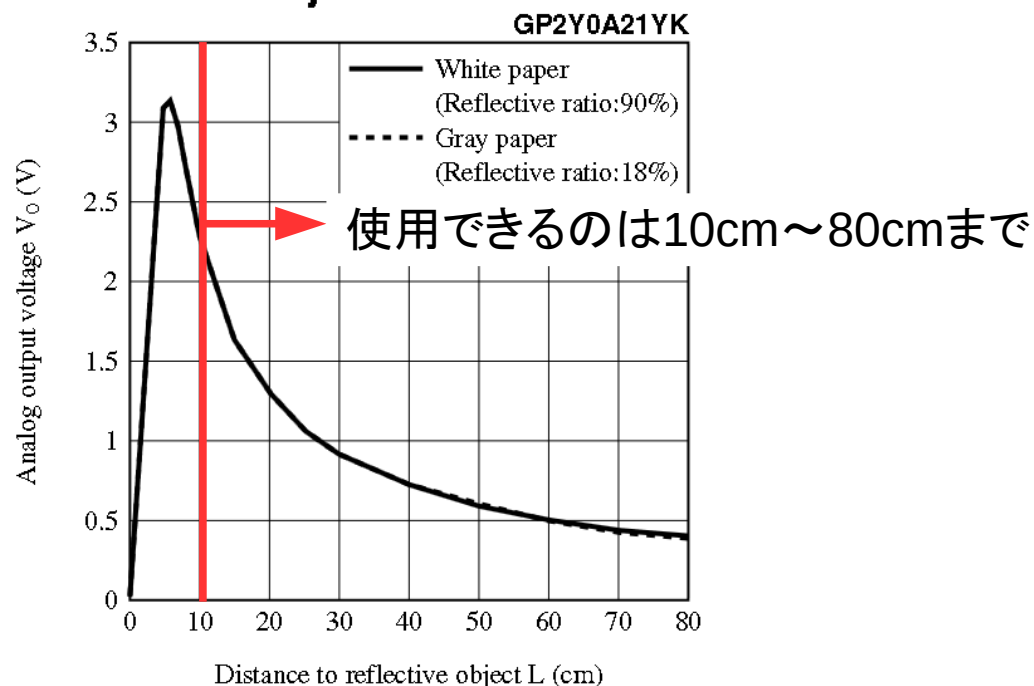
壁に近づいたら車輪速度を0にする



ヒント③:

- ・10cm以下は使用できない  
(同じ距離を示す数値が2つ存在してしまうため)

Fig.5 Analog Output Voltage vs. Distance to Reflective Object



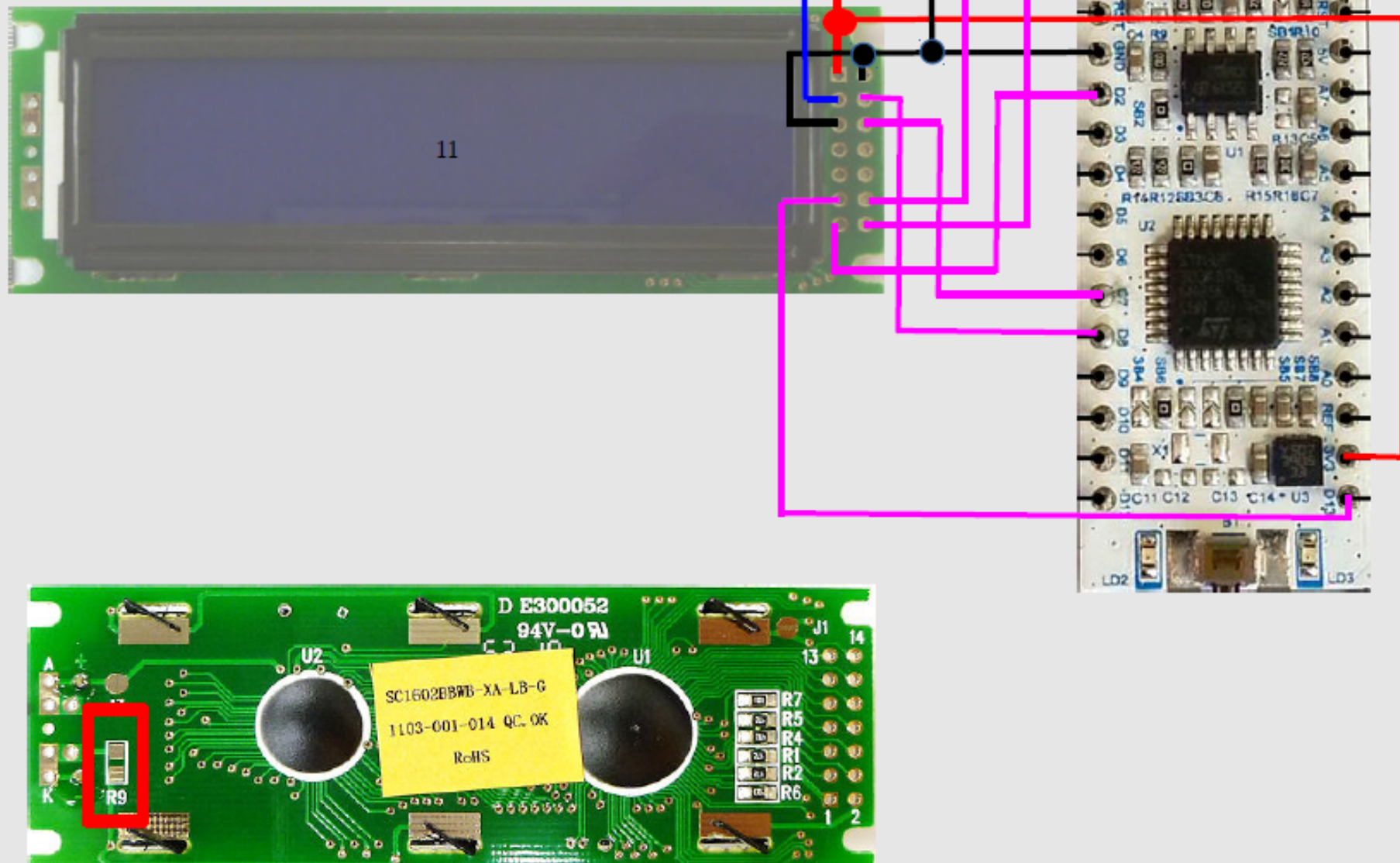
# lcdディスプレイを使う



## 準備するもの

- lcd(P-04794)
- スペース(P-01861)
- 2×7ヘッダピン×2
- 2×7リボンケーブル(C-02489)
- 半固定ボリューム10K $\Omega$  10個入り(P-02470)

# 回路図

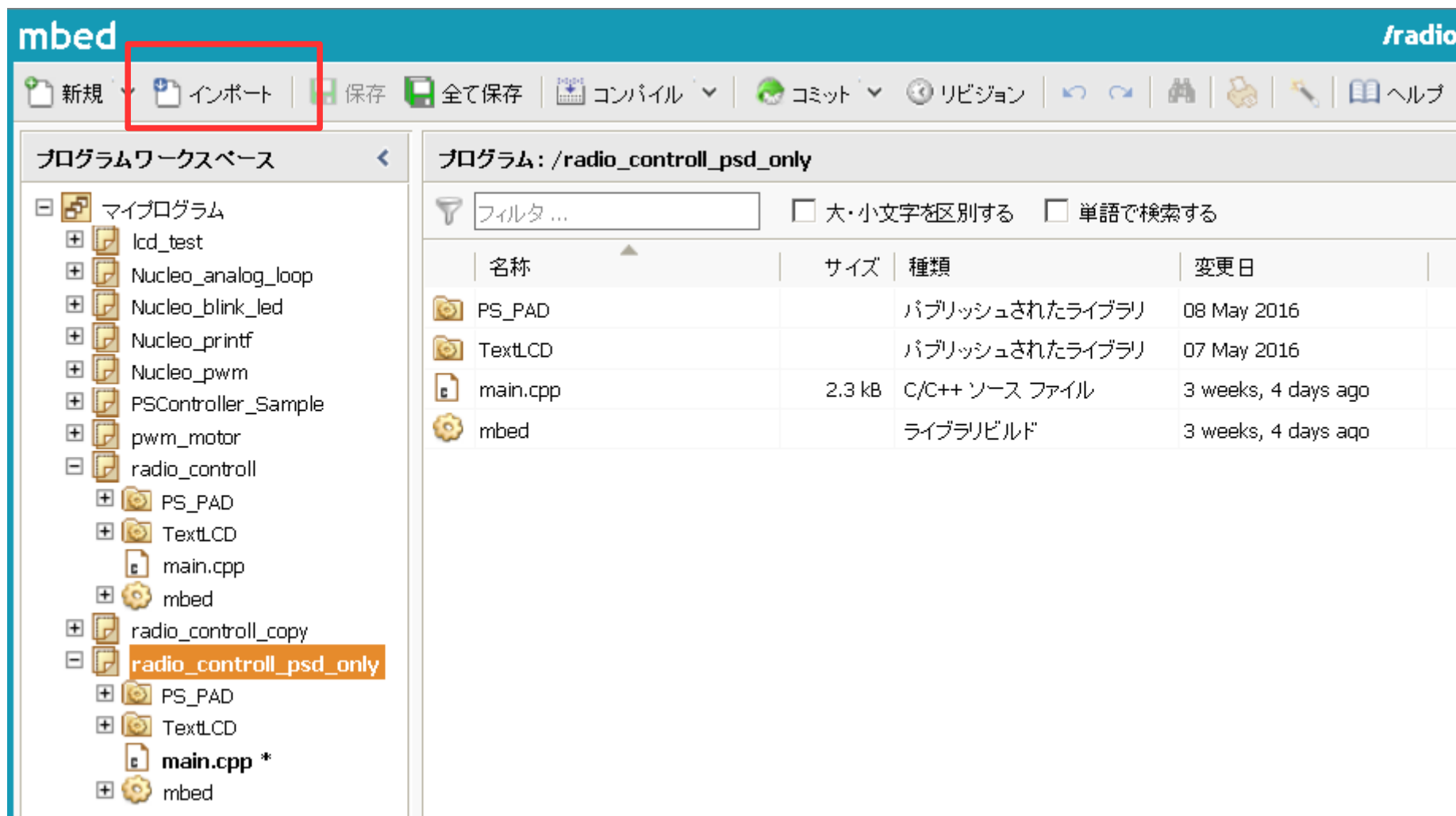


付属の抵抗でR9をつなぐ

# LCDライブラリのインポート



# インポートをクリック



The screenshot shows the mbed IDE interface. The top menu bar has a red box around the 'インポート' (Import) button. The left sidebar shows a tree view of the project structure, with 'radio\_controll\_psd\_only' selected. The right pane shows the details of the selected project, including a table of files and their properties.

プログラムワークスペース

マイプログラム

- lcd\_test
- Nucleo\_analog\_loop
- Nucleo\_blink\_led
- Nucleo\_printf
- Nucleo\_pwm
- PSController\_Sample
- pwm\_motor
- radio\_controll
  - PS\_PAD
  - TextLCD
  - main.cpp
  - mbed
- radio\_controll\_copy
- radio\_controll\_psd\_only**
  - PS\_PAD
  - TextLCD
  - main.cpp \*
  - mbed

プログラム: /radio\_controll\_psd\_only

フィルタ ... ☐ 大・小文字を区別する ☐ 単語で検索する

名称	サイズ	種類	変更日
PS_PAD		パブリッシュされたライブラリ	08 May 2016
TextLCD		パブリッシュされたライブラリ	07 May 2016
main.cpp	2.3 kB	C/C++ ソース ファイル	3 weeks, 4 days ago
mbed		ライブラリビルド	3 weeks, 4 days ago

# ライブラリタブを選択

インポートウィザード


**mbed.org から ライブラリ をインポートする**

リストから ライブラリ を選択してください。項目をワークスペースにドラッグ&ドロップする事も可能です。  
[ここをクリック](#)してURLからインポートもできます。


インポート!

プログラム   **ライブラリ**   ブックマーク   アップロード

TextLCD   検索

ワークスペースにインポートする "TextLCD" にマッチした mbed.org 内のパブリッシュされたライブラリのリスト

	名称	タグ	作成者	インポート	最終変更	説明
☆	TextLCD		<a href="#">Carlos Alberto Nas</a>	318	<a href="#">21 Nov 2012</a>	
☆	TextLCD		<a href="#">masa haru</a>	298	<a href="#">21 Dec 2010</a>	
☆	TextLCD		<a href="#">ben yun</a>	844	<a href="#">03 Aug 2011</a>	
☆	TextLCD		<a href="#">Michael Allan</a>	103	<a href="#">21 Mar 2012</a>	
☆	TextLCD	<a href="#">16x1 TextLCD</a>	<a href="#">Thomas Lunzer</a>	1285	<a href="#">24 May 2011</a>	
☆	TextLCD		<a href="#">masa haru</a>	215	<a href="#">12 Nov 2012</a>	

# "TextLCD"で検索

インポートウィザード

 **mbed.org から ライブラリ をインポートする**

リストから ライブラリ を選択してください。項目をワークスペースにドラッグ&ドロップする事も可能です。  
[ここをクリック](#)してURLからインポートもできます。

プログラム ライブラリ ブックマーク アップロード

TextLCD 検索

ワークスペースにインポートする "TextLCD" にマッチした mbed.org 内のパブリッシュされたライブラリのリスト

名称	タグ	作成者	インポート	最終変更	説明
☆ TextLCD		<a href="#">Carlos Alberto Nas</a>	318	<a href="#">21 Nov 2012</a>	
☆ TextLCD		<a href="#">masa haru</a>	298	<a href="#">21 Dec 2010</a>	
☆ TextLCD		<a href="#">ben yun</a>	844	<a href="#">03 Aug 2011</a>	
☆ TextLCD		<a href="#">Michael Allan</a>	103	<a href="#">21 Mar 2012</a>	
☆ TextLCD	<a href="#">16x1 TextLCD</a>	<a href="#">Thomas Lunzer</a>	1285	<a href="#">24 May 2011</a>	
☆ TextLCD		<a href="#">masa haru</a>	215	<a href="#">12 Nov 2012</a>	

# 名称:TextLCD,作成者:SimonFord のライブラリを選択

## インポートウィザード



### mbed.org から ライブラリ をインポートする

リストから ライブラリ を選択してください。項目をワークスペースにドラッグ&ドロップする事も可能です。  
[ここをクリック](#)してURLからインポートもできます。

プログラム

ライブラリ

ブックマーク


アップロード

ワークスペースにインポートする "TextLCD" にマッチした mbed.org 内のパブリッシュされたライブラリのリスト


名称	タグ	作成者	インポート	最終変更
★ TextLCD	HD44780 TextLCD	Simon Ford	44378	02 Jan 2014
★ Terminal	ANSI Serial Terminal	Simon Ford	1520	23 Nov 2010
★ New TextLCD	HD44780 PCF2116 TextLCD	Erik Kerger	1496	28 Mar 2011

# 右上のインポートをクリック

インポートウィザード

**mbed.org から ライブラリ をインポートする**

リストから ライブラリ を選択してください。項目をワークスペースにドラッグ & ドロップする事も可能です。  
[ここをクリック](#)してURLからインポートもできます。

**インポート!**

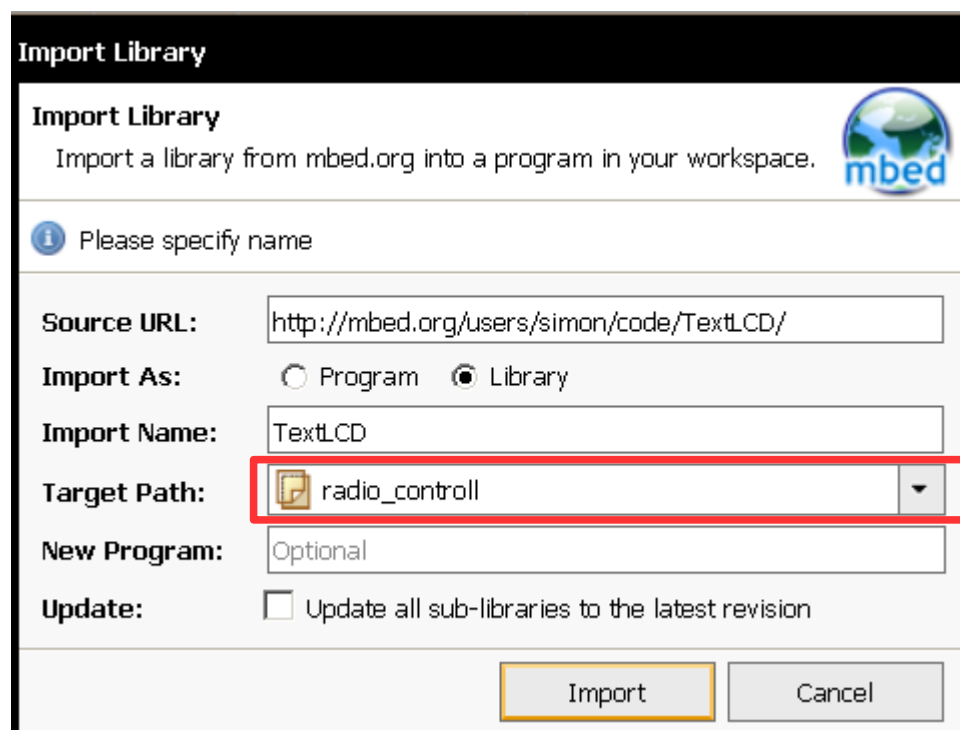
プログラム   ライブラリ   ブックマーク   アップロード

TextLCD   検索

ワークスペースにインポートする "TextLCD" にマッチした mbed.org 内のパブリッシュされたライブラリのリスト

名称	タグ	作成者	インポート	最終変更	説明
★ TextLCD	HD44780 TextLCD	Simon Ford	44378	02 Jan 2014	TextLCD library for controlling various LCD panels based on the HD44780 4-bit interface

# インポート先のプログラムを選択し、Importをクリック



**Import Library**

Import a library from mbed.org into a program in your workspace.

Please specify name

**Source URL:**

**Import As:** ☐ Program ☒ Library

**Import Name:**

**Target Path:**  ▼

**New Program:**

**Update:** ☐ Update all sub-libraries to the latest revision

**Import** **Cancel**

# LCD表示サンプルプログラム

```
1 #include "mbed.h"
2 #include "TextLCD.h"
3
4 TextLCD g_lcd(PF_1, PF_0, PB_3, PA_10, PA_12, PA_9); // RS, E, DB4, DB5, DB6, DB7
5
6 int main() {
7
8     while(1) {
9
10         g_lcd.locate(0, 1); ← 1段目に表示させたいときは(0,0)
11         g_lcd.printf( "hello");
12     }
13 }
```

# PSDセンサ値をLCDに表示させてみよう

