2023.07.06

OpenPLC の学習 （Arduino Uno R3 使用）

1. OpenPLC開発環境構築手順

手持ちの PC に下記 3つのアプリを順にインストールしてください。

1. OpenPLC Editor

https://openplcproject.com/download-windows#

寄付される方は＄口を選択、されない方は「JUST DOWNLOAD」を選択し、ソフト exe をダウンロード。ファイルが大きいのでネット環境により時間がかかります。
インストールは初期設定のまま進める。

2. OpenPLC Runtime

https://openplcproject.com/docs/installing-openplc-runtime-on-windows/

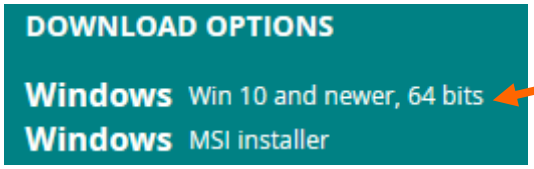
ここをクリックしてダウンロード

インストールは初期設定のまま。Windows コマンドプロッセサ窓が開いて実行されます。

3. Arduino IDE

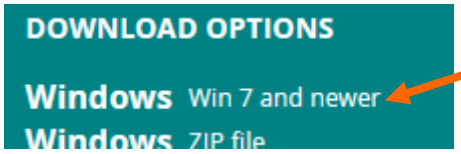
https://www.arduino.cc/en/Main/Software

より ダウンロード 今（230606）のバージョンは ArduinoIDE 2.1.0



これを選択

PC が Windows7 の場合は ArduinoIDE 1.8.19（このページの下のほうにある。）

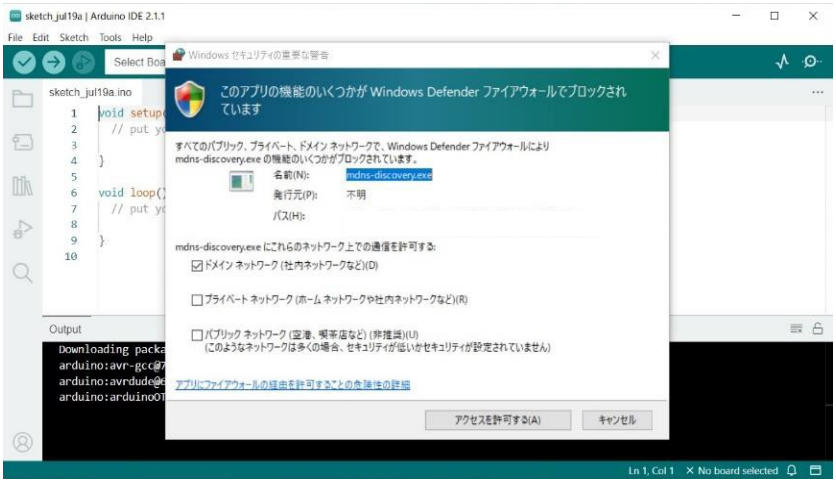


これを選択

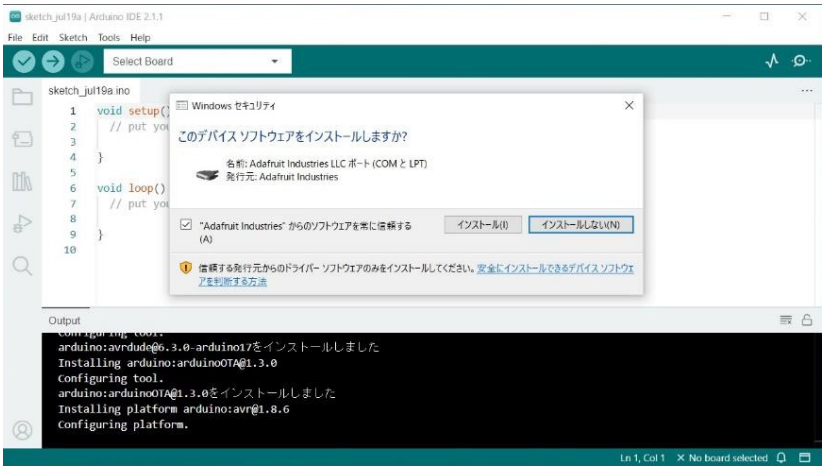
『実習では Arduino IDE は使用しませんが PC により Arduino USB 仮想 COM ポートのドライバがない場合があるのでインストールしておきます。』

Arduino IDE のインストールについて（補足）

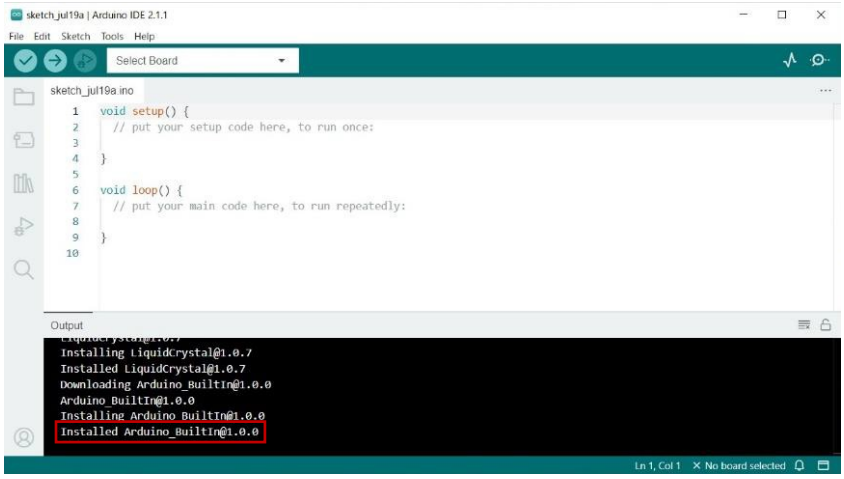
①ネットワークに接続した状態で「Arduino IDE」を開く
②セキュリティの警告が表示された場合は「アクセスを許可する」を選択



③ソフトウエアのインストールの確認が数回表示されるので、いずれも「インストール」を選択



④テキストウインドウに「Installed Arduino_BuiltIn@1.0.0」が表示されればインストール完了



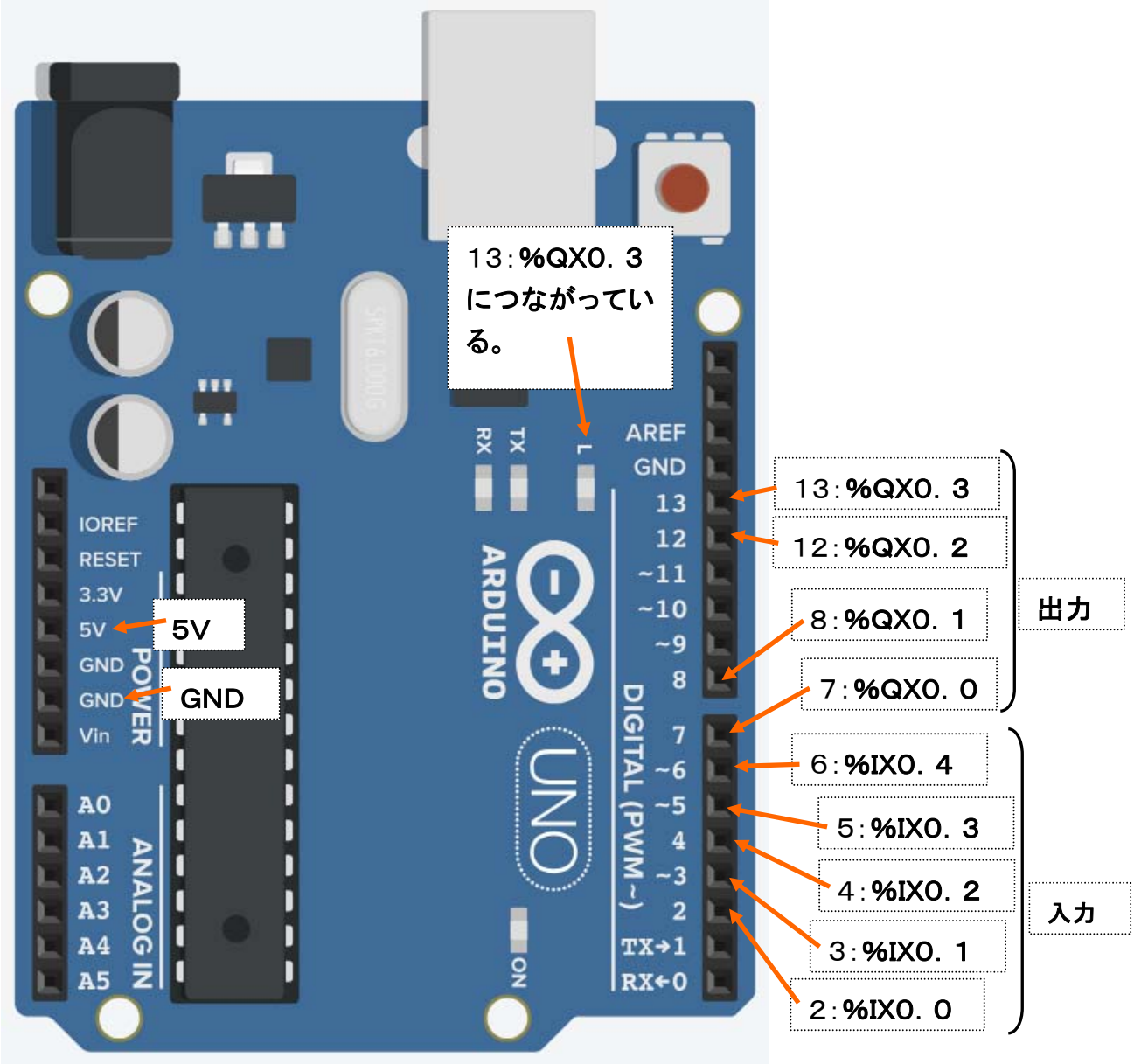
※Arduino が正しくインストールできていないと OpenPLC で作成したデータをアップロードできません。

2. Arduino　Uno R3 基板　配線

1. Arduino　Uno R3 基板　OpenPLC　 IO　割付け

https://openplcproject.com/docs/2-4-physical-addressing/

詳細はこのリンクで確認



Arduino　Uno R3　購入先リンク

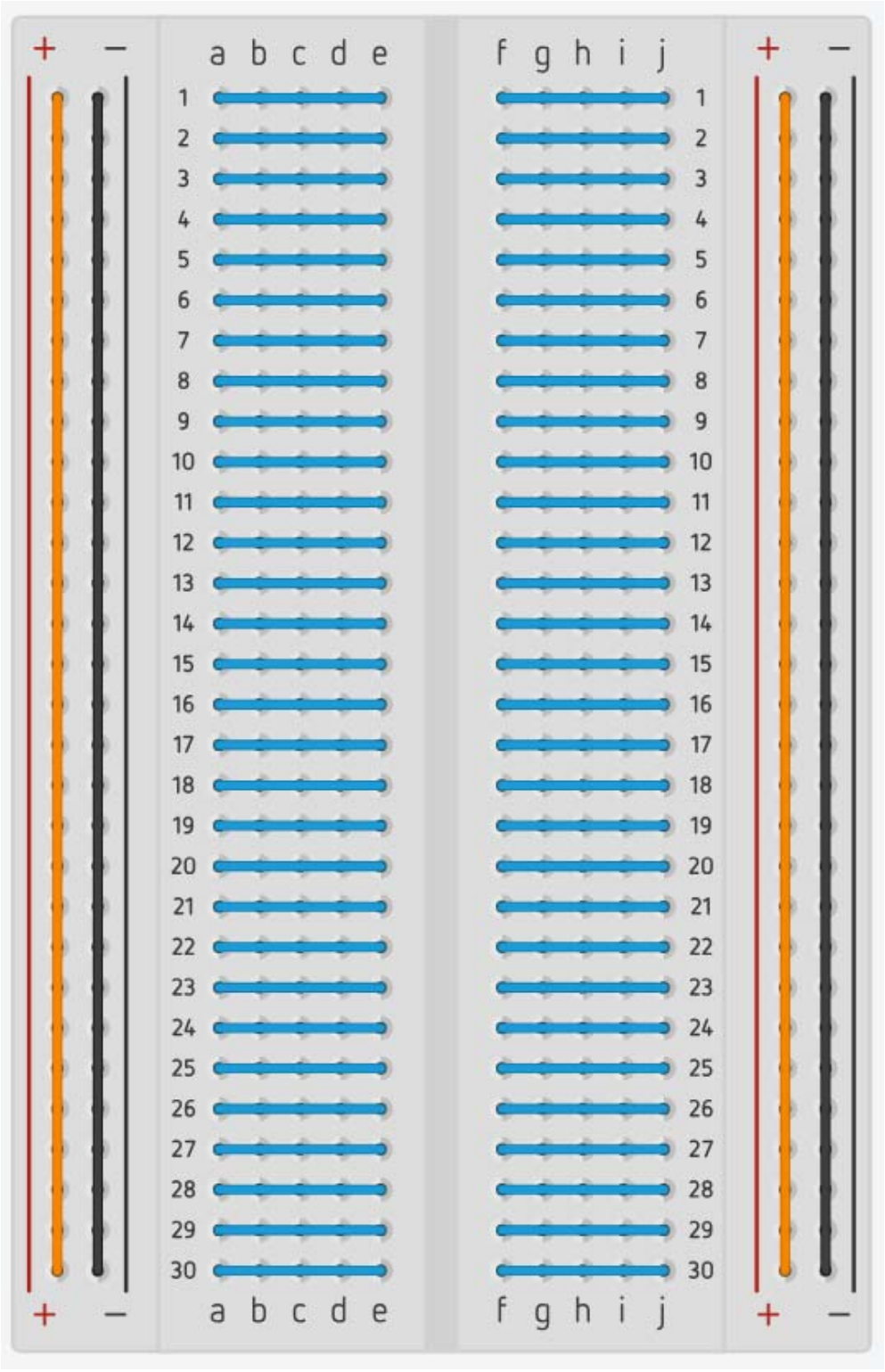
https://akizukidenshi.com/catalog/g/gM-07385/

メーカリンク

https://docs.arduino.cc/hardware/uno-rev3

ブレッドボード(BB-801) パターン図

購入先リンク https://akizukidenshi.com/catalog/g/gP-05294/



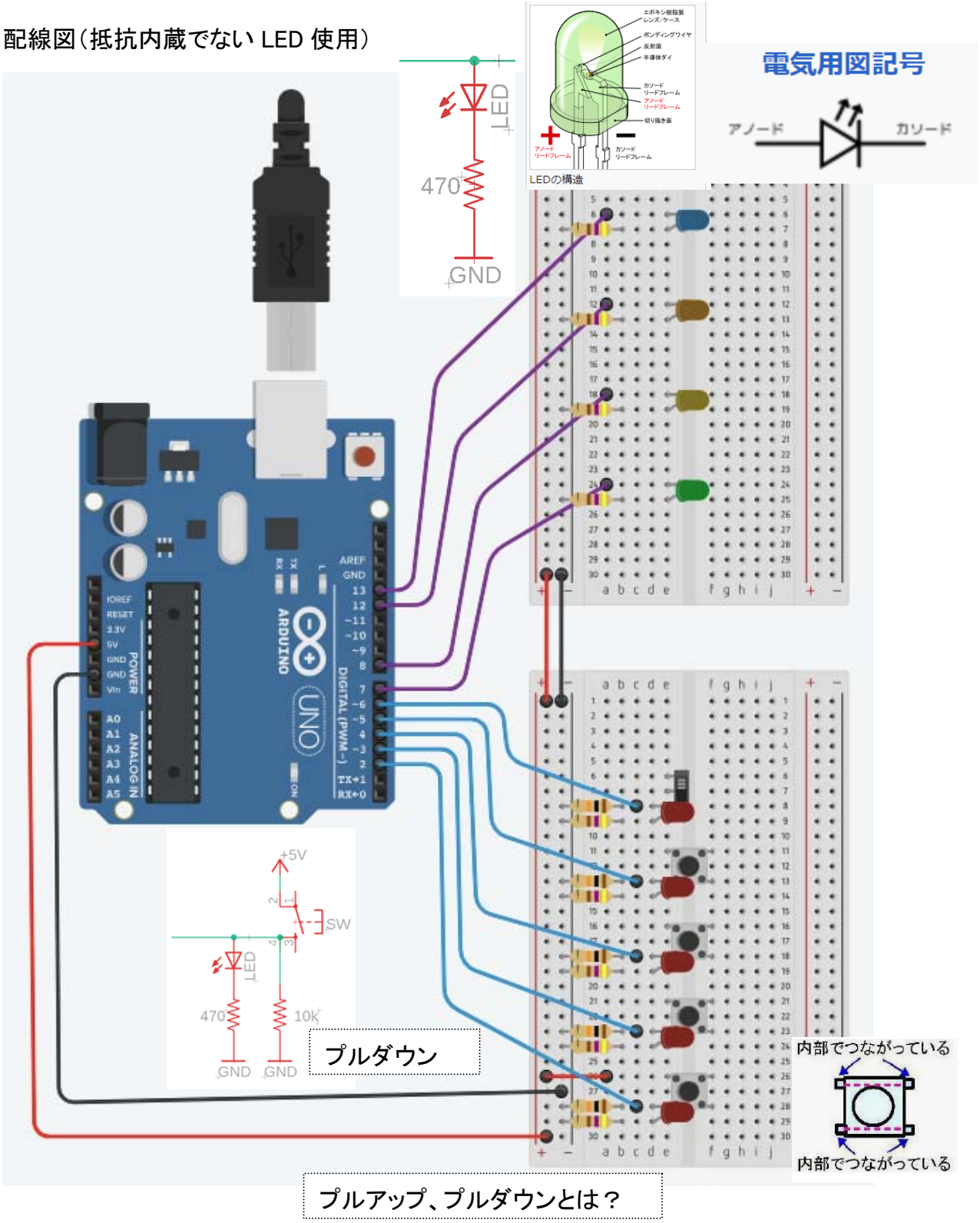
抵抗内蔵 LED　OSR5JA3134A

https://akizukidenshi.com/catalog/g/gI-16692/

タクトスイッチ　DTS-63-N-V-BLK

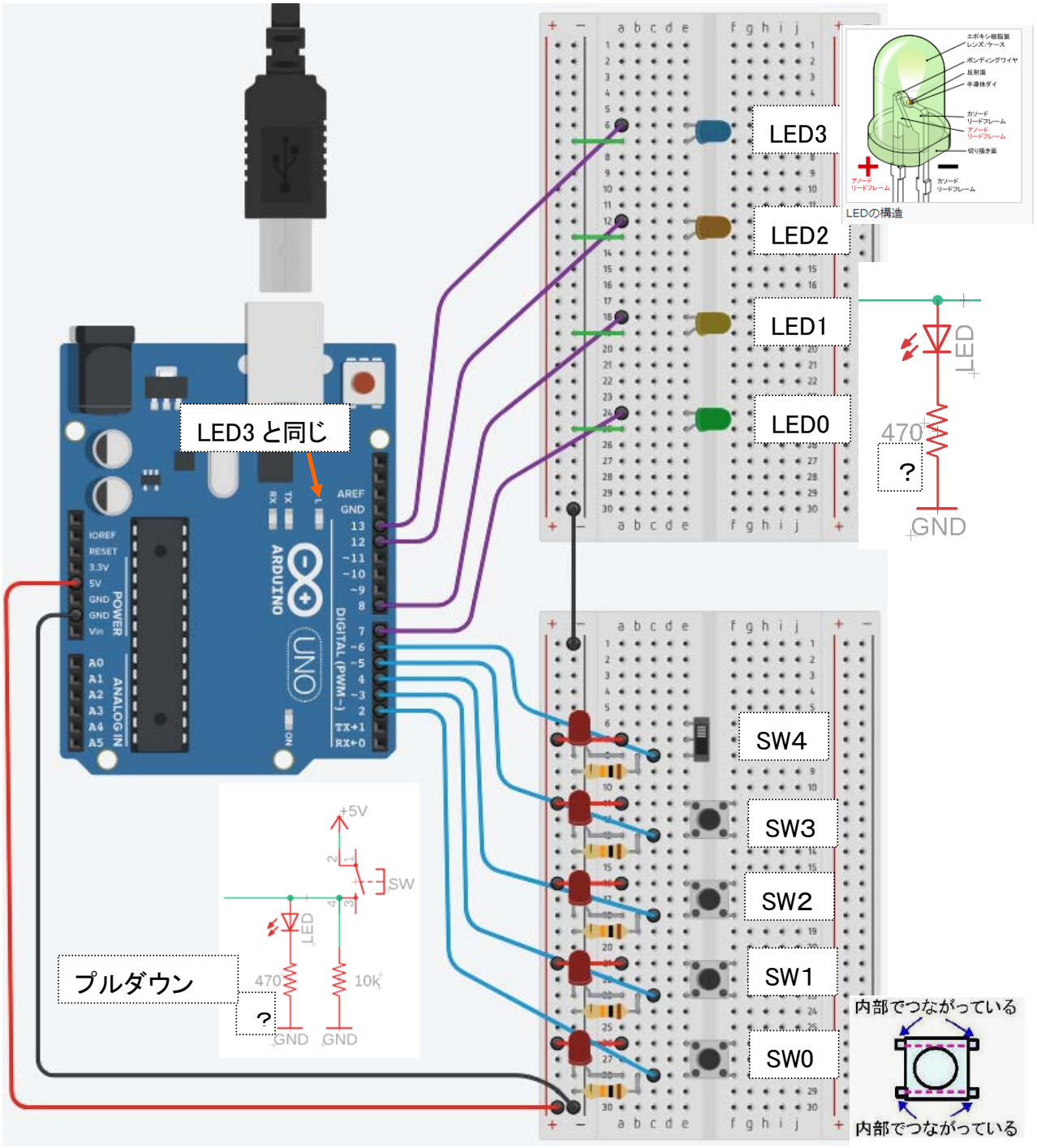
https://akizukidenshi.com/catalog/g/gP-03647/

配線図（抵抗内蔵でない LED 使用）



「AUTODESK　Tinkercad」で作成

配線図 （抵抗付き LED 使用）

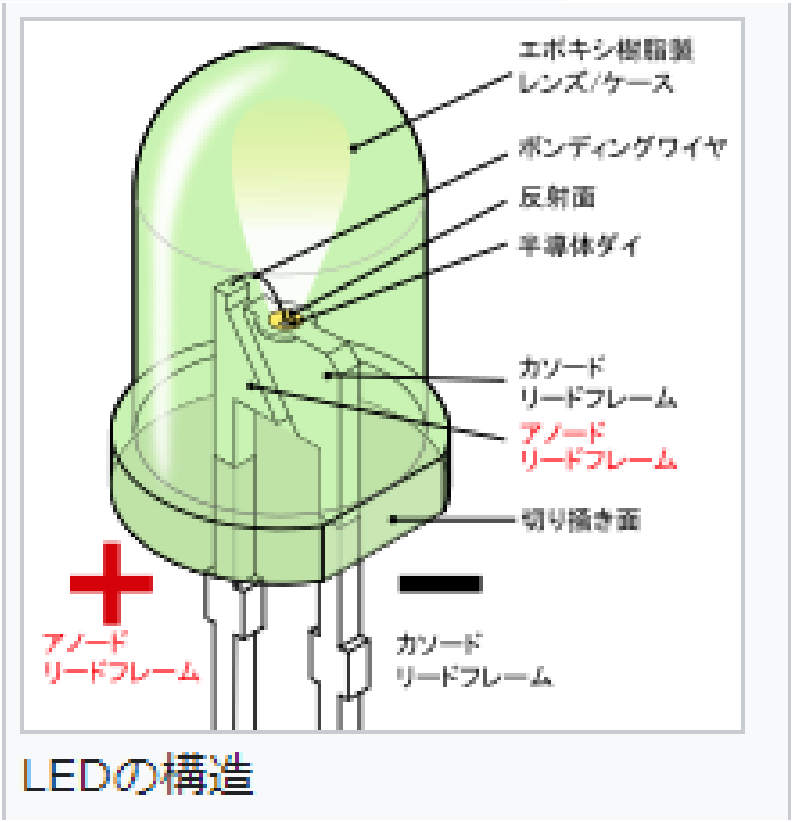


「AUTODESK　Tinkercad」で作成

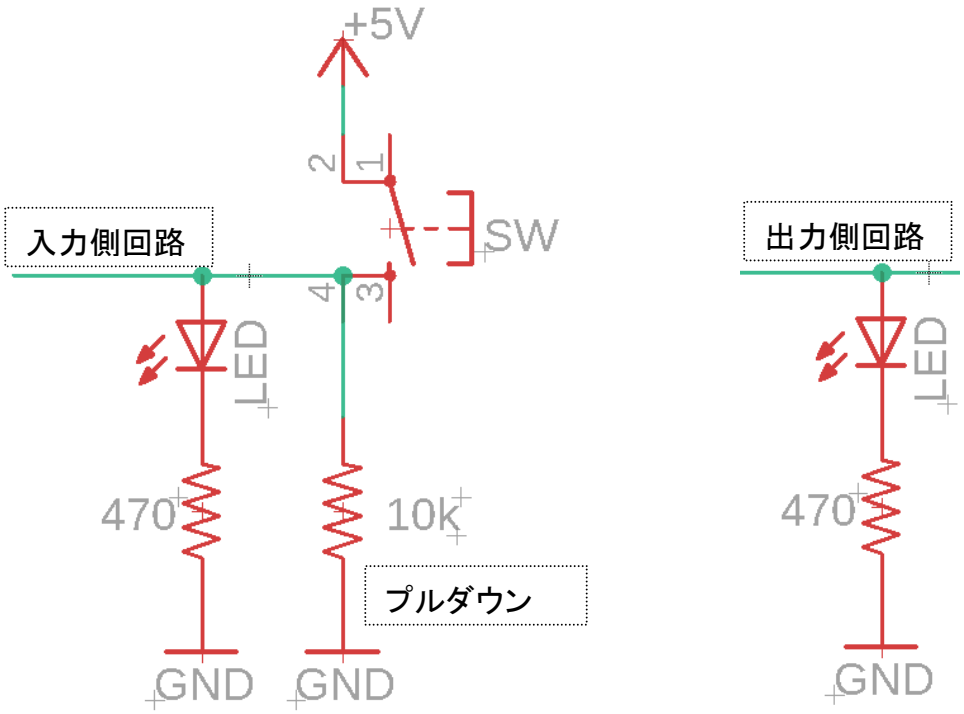
抵抗のカラーコード

https://www.jarl.org/Japanese/7_Technical/lib1/teikou.htm


電気用図記号
アノード
カソード


エポキシ樹脂製
レンズ/ケース
ボンディングワイヤ
反射面
半導体ダイ
カソード
リードフレーム
アノード
リードフレーム
切り欠き面
アノード
リードフレーム
カソード
リードフレーム

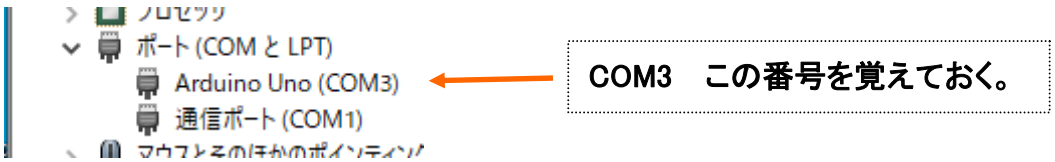
LEDの構造


+5V
2
1
SW
入力側回路
4
3
LED
470
10k
プルダウン
GND
GND
出力側回路
LED
470
GND

はじめの一歩

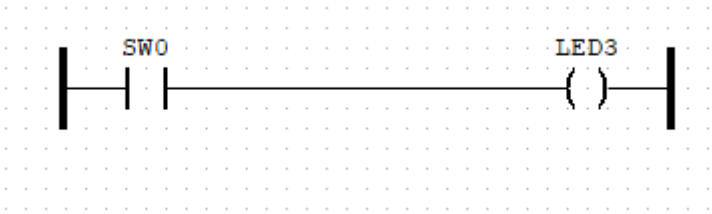
1．パソコンと Arduino を USB ケーブルで繋いでみる。

COM ポート番号の確認
デバイスマネージャーから

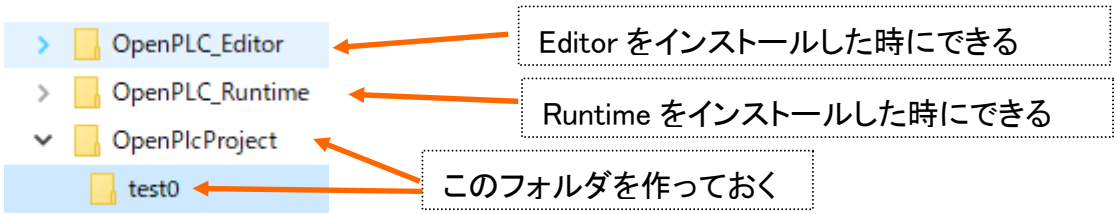


[デバイス マネージャー]を出すには
Windows10:
画面左下の（スタートボタン）を**右**クリックし、表示された「クイック リンク」メニューから[デバイス マネージャー]をクリックします。
Windows7:
画面左下の（スタートボタン）を**左**クリックし、コントロールパネル＞システムとセキュリティ＞システム　の左欄に[デバイス マネージャー]がある。

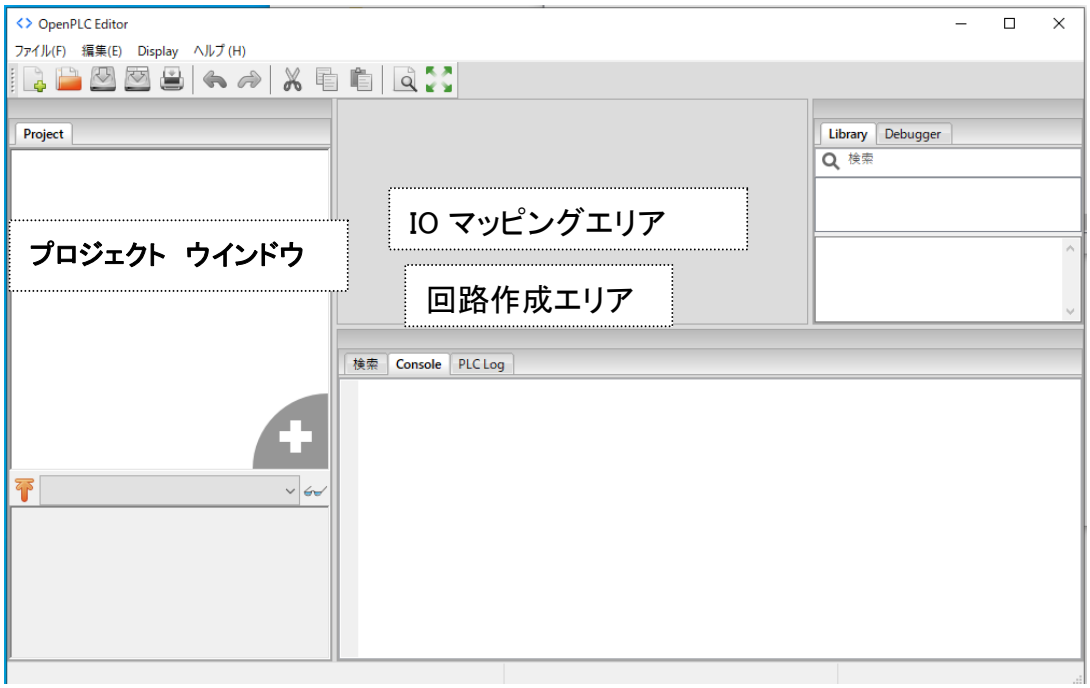
2．L チカ回路で確認



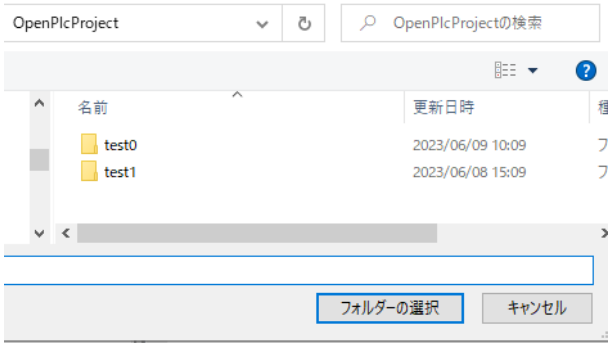
準備として、プロジェクトを入れるフォルダーを作っておく。（フォルダ名に日本語が入っていると立ち上がらないかも？）



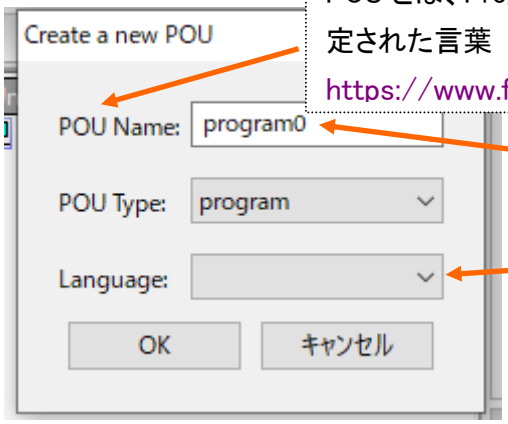
OpenPLC_Editor を起動する



ファイル New



test0 を選択 （あらかじめフォルダを作っておかないと「パスが存在しません」とメッセージがでる。） 名前を付けて保存の時もフォルダを作っておく。



POU とは、Program Organization Unit（プログラム構成ユニット） IEC61131-3 に規定された言葉 詳細は下記で
https://www.fa.omron.co.jp/product/special/sysmac/plcopen/feature1.html

test0 に変更

LD を選択
（ラダーダイアグラム：Ladder Diagram）

設定完了で OK をクリック

プロジェクト名

リソース

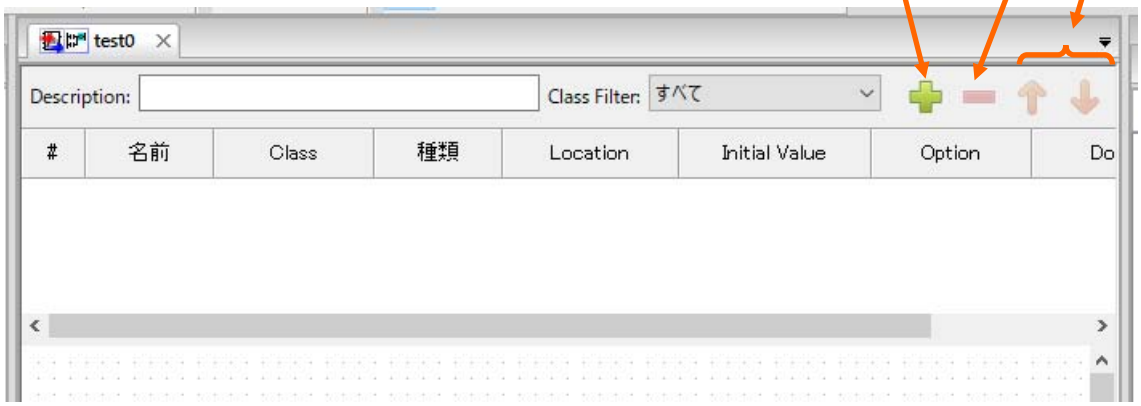
これでプロジェクトの設定は完了

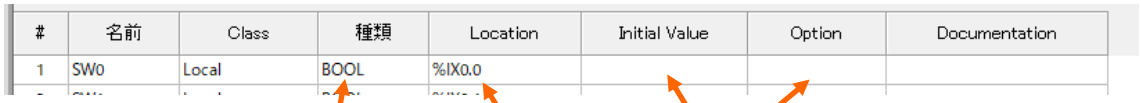
順番を変えるとき

追加

削除

IO マッピング





手入力

空白

コメント（日本語 OK）

「BASE Type」→「BOOL」を選択

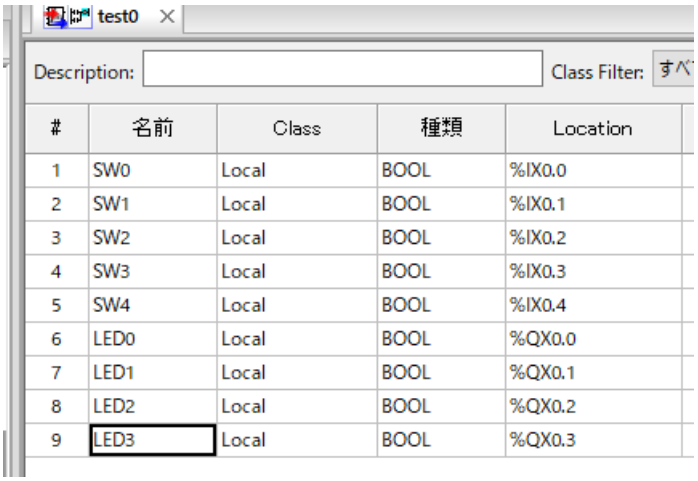
true 真

false 偽

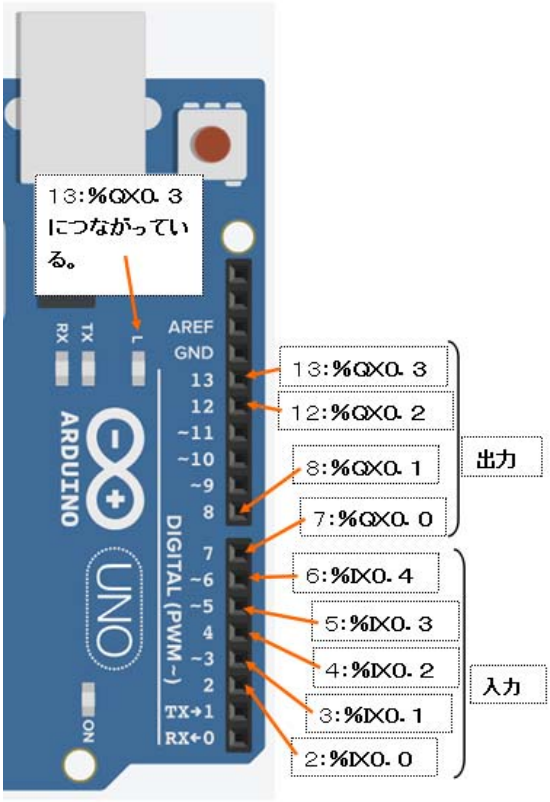
IO のマッピング

1. Arduino Uno R3 基板 OpenPLC IO 割付け
https://openplcproject.com/docs/2-4-physical-addressing/

を参考にして

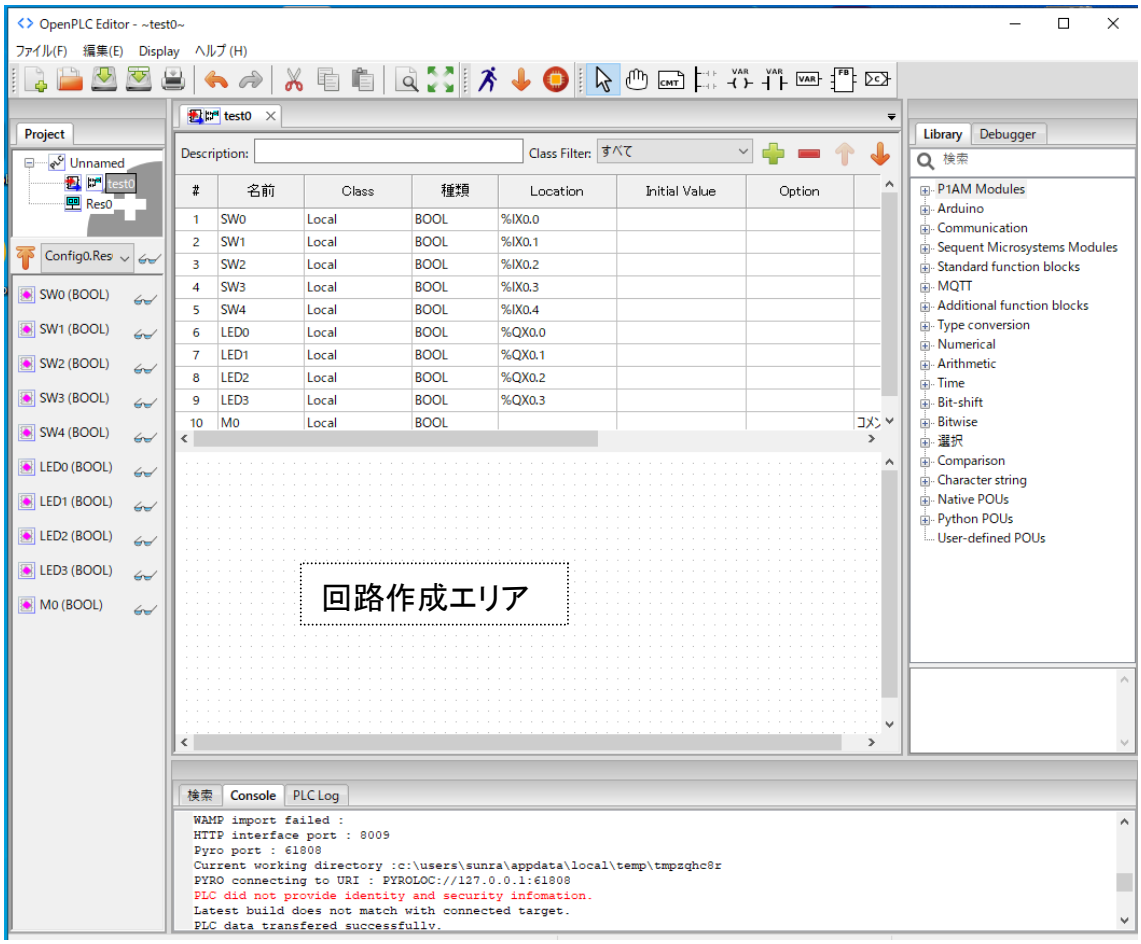


内部メモリ（補助リレー）では Location は設定しない。

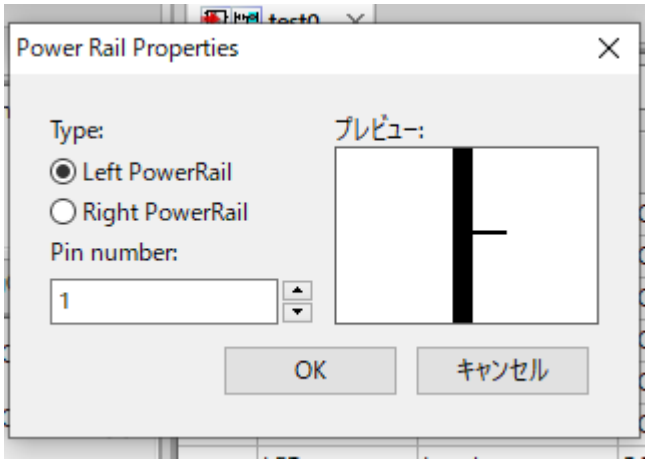


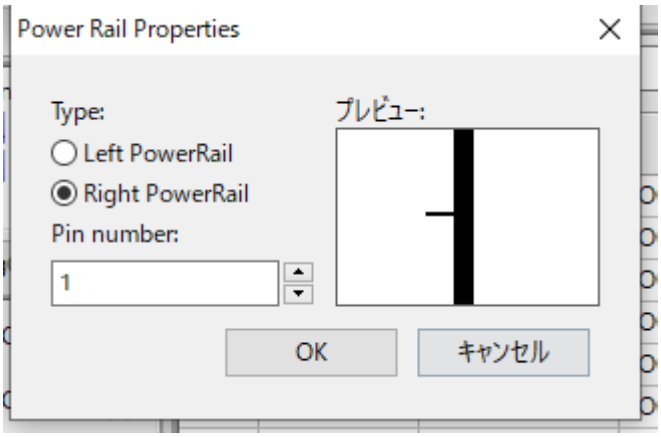
1	SW0	Local	BOOL	%IX0.0
2	SW1	Local	BOOL	%IX0.1
3	SW2	Local	BOOL	%IX0.2
4	SW3	Local	BOOL	%IX0.3
5	SW4	Local	BOOL	%IX0.4
6	LED0	Local	BOOL	%QX0.0
7	LED1	Local	BOOL	%QX0.1
8	LED2	Local	BOOL	%QX0.2
9	LED3	Local	BOOL	%QX0.3

回路作成

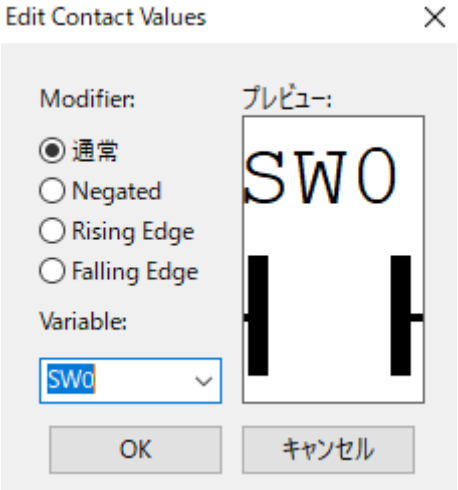


回路作成エリア内で右クリック　追加　Power Rail

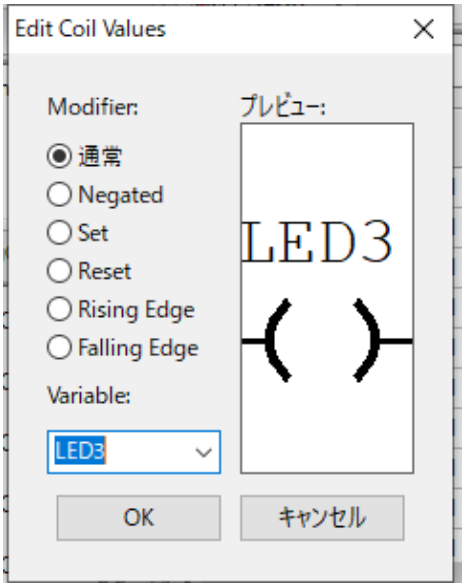




回路作成エリア内で右クリック　追加　Contact

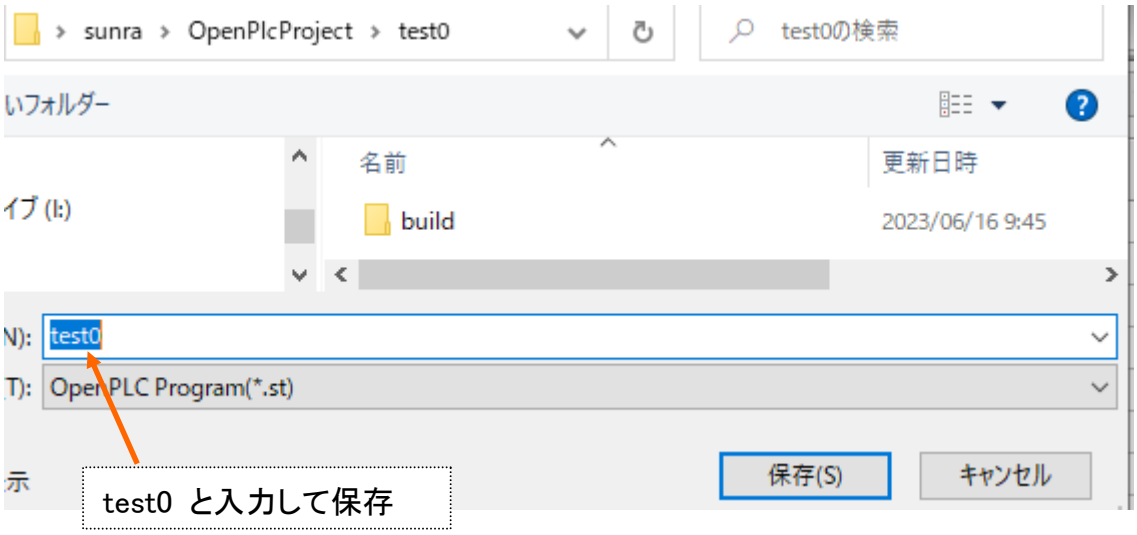


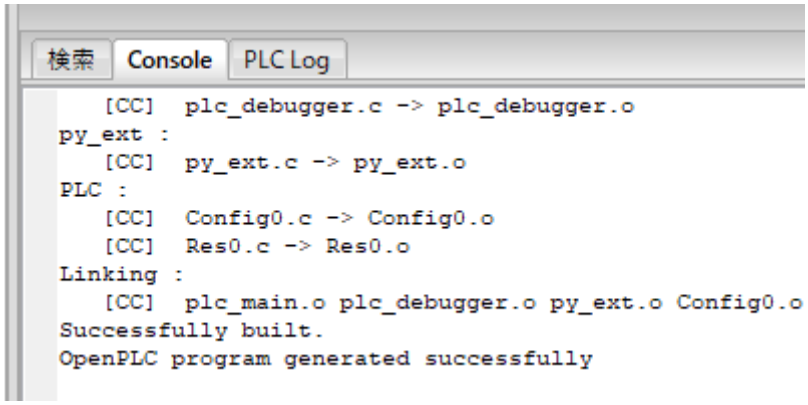
同様にして Coil　を追加

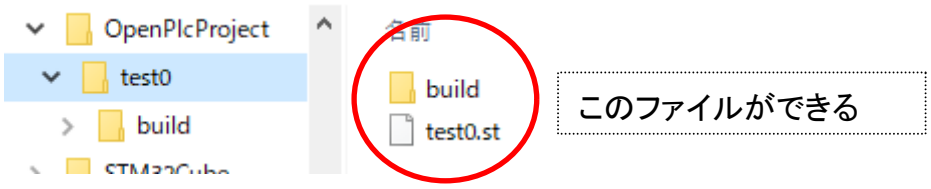


配線をする。

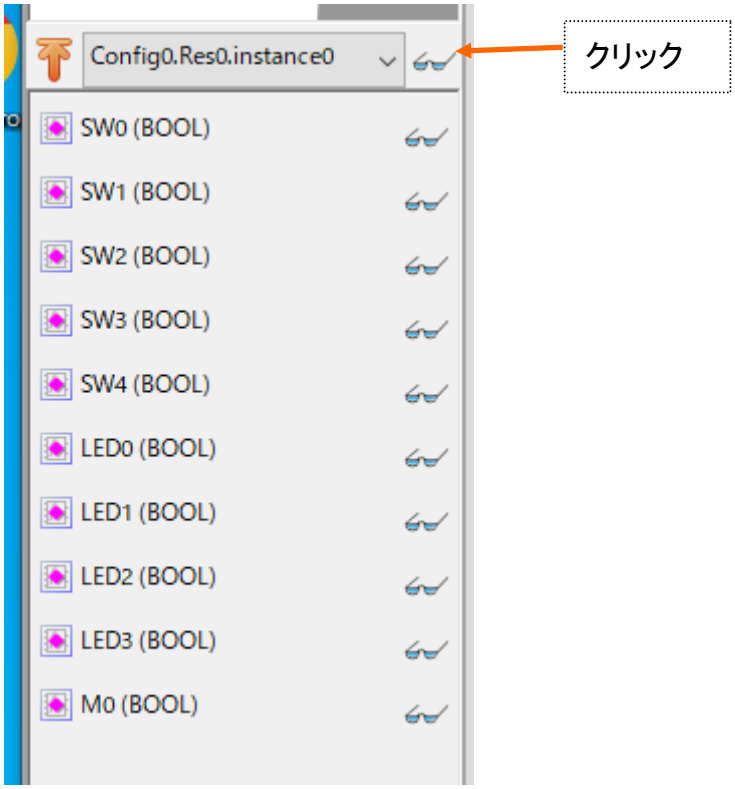
回路をコンパイルして確認

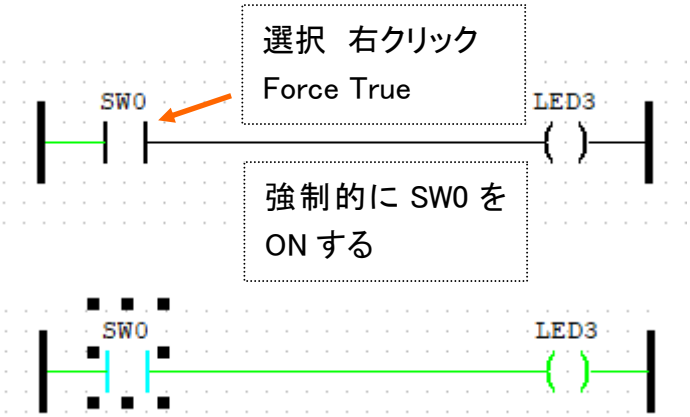

sunra › OpenPlcProject › test0
test0の検索
いフォルダー
名前
更新日時
build
2023/06/16 9:45
イブ (I:)
N):
test0
T):
Open PLC Program(*.st)
示
保存(S)
キャンセル
test0 と入力して保存

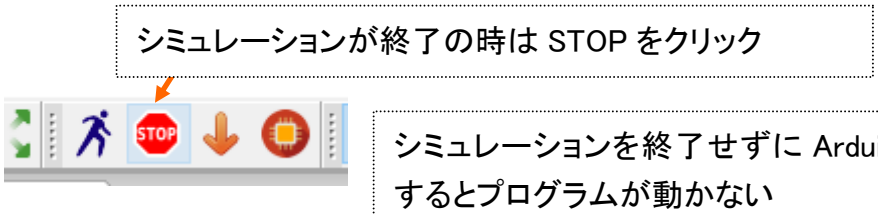

検索
Console
PLC Log
[CC] plc_debugger.c -> plc_debugger.o
py_ext :
[CC] py_ext.c -> py_ext.o
PLC :
[CC] Config0.c -> Config0.o
[CC] Res0.c -> Res0.o
Linking :
[CC] plc_main.o plc_debugger.o py_ext.o Config0.o
Successfully built.
OpenPLC program generated successfully


OpenPlcProject
test0
build
名前
build
test0.st
このファイルができる

シミュレーションで確認



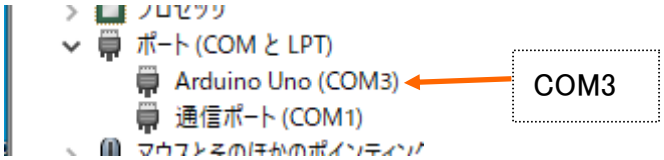


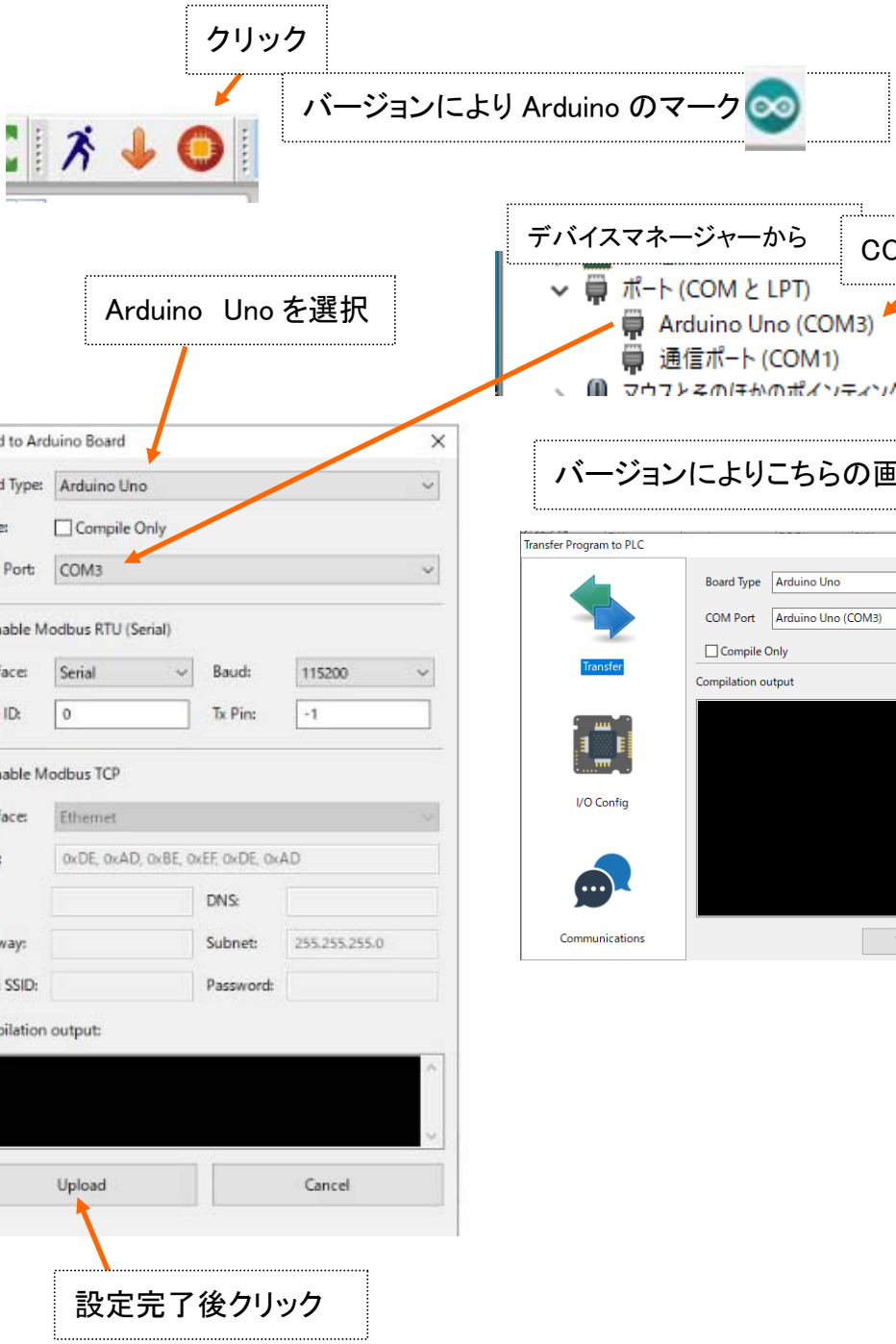


次に Arduino　Uno　で確認
PC と Arduino を接続（USB）

COM ポート番号の確認
デバイスマネージャーから
Win10：画面左下の （スタートボタン）を右クリックし、表示された「クイック　リンク」メニューから[デバイス　マネージャー]をクリックします。

Win7：画面左下の （スタートボタン）を左クリックし、コントロールパネル＞システムとセキュリティ＞システム　　の左欄に[デバイス　マネージャー]がある。






```
Uploading program to Arduino board at COM3...

Done!
```

この表示が出て転送完了

最初はRuntimeを転送するため時間がかかる。2回目からは短時間

次は実機（Arduino　Uno）にて

まだ配線していないがSW0を押すとLED3（基板内の LED)が点灯する。はず

一旦　配線の為に USB ケーブルを抜く

Arduino とブレッドボード IO　SW　LED　接続
（　ニッパーでリード線を切るときの注意：ビニール袋の中で、少し斜め切り　）

配線図の　GND,、5V、SW0　のみ接続する

USB ケーブルを接続する。

SW0 を押して Arduino 基板内の LED が点灯することを確認

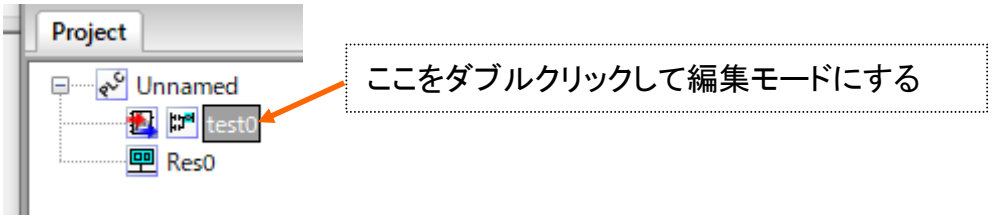
これで最初の一歩は終了です。＼(^o^)／

このプログラムを**保存**しておく。

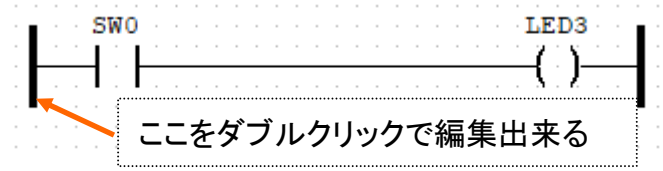
USB ケーブルを抜いて Arduino とブレッドボード IO の配線を行う。（全ての SW と LED）

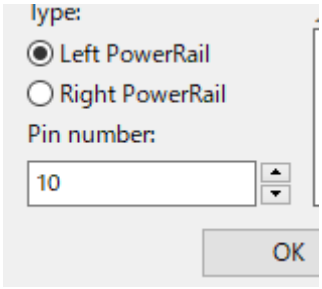
同様のプログラム作成し接続とプログラムの確認をする。

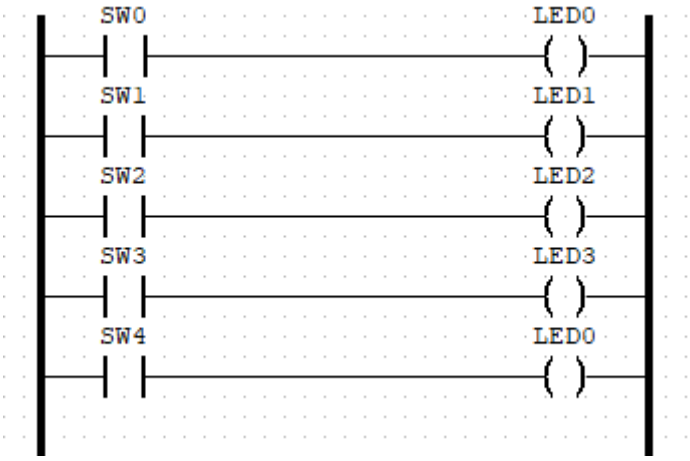
編集のやり方いろいろ



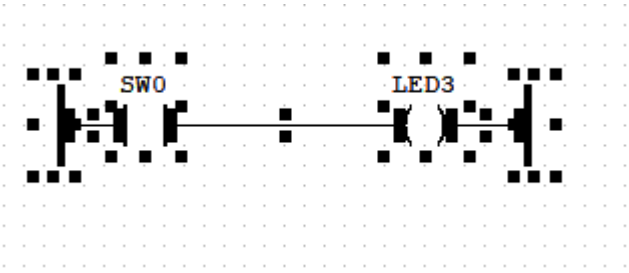
PowerRail を伸ばすには



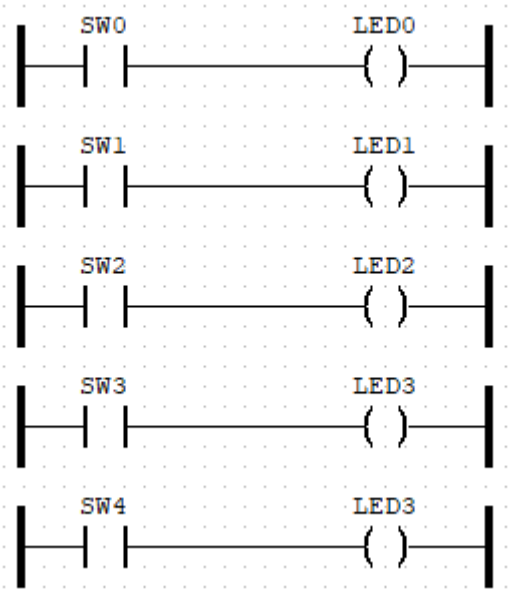




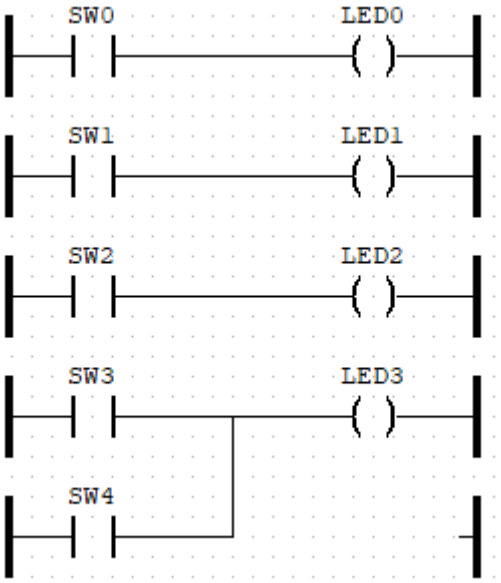
選択してコピー　AND　ペースト　もできる。
グループ選択もできる。



編集して

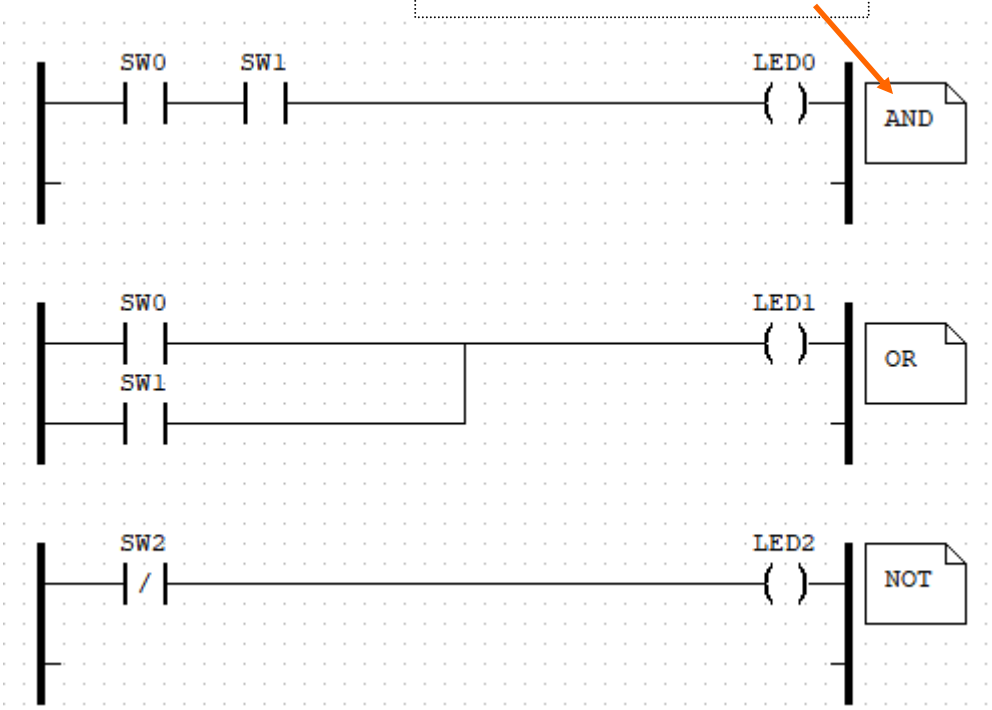


2 重コイルになっている



AND、OR、NOT 回路

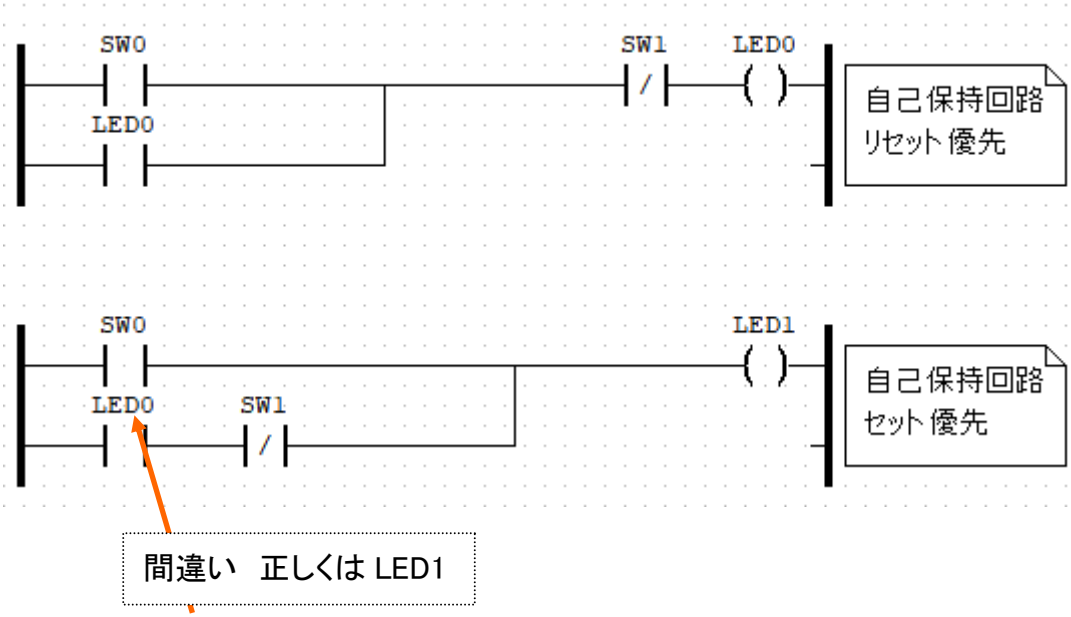
右クリック：追加＞Comment



自己保持回路（リセット優先）

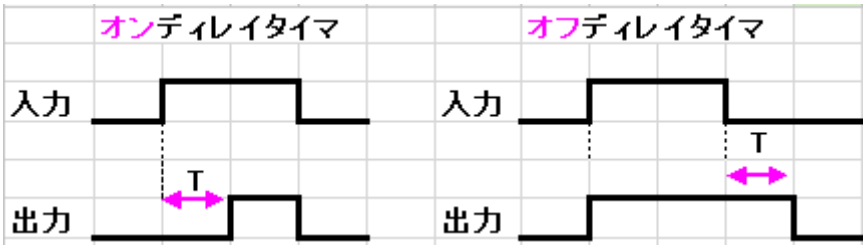
自己保持回路（セット優先）

セット：SW0　リセット：SW1 として

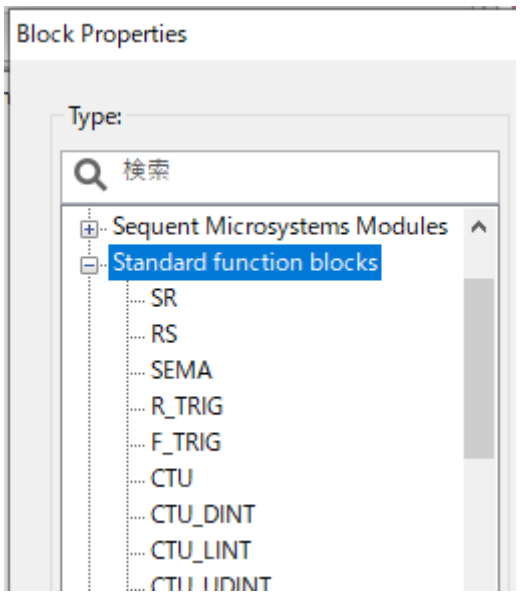


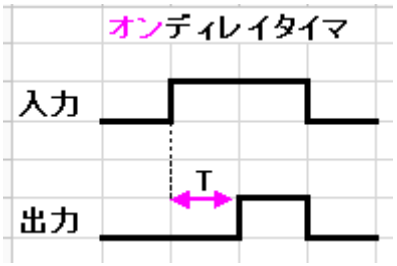
間違い　正しくは LED1

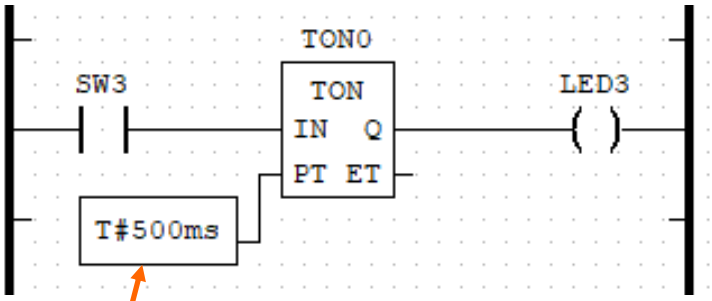
タイマー回路

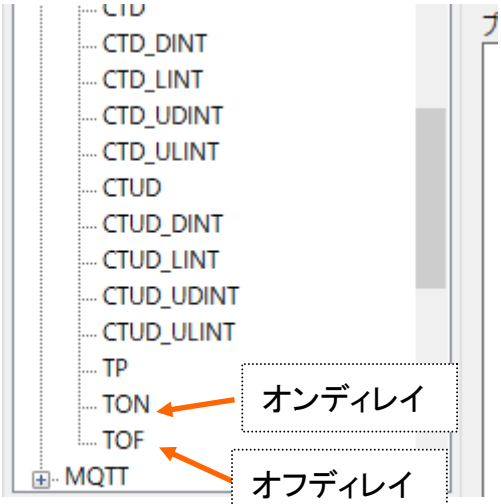


右クリック：追加＞Block

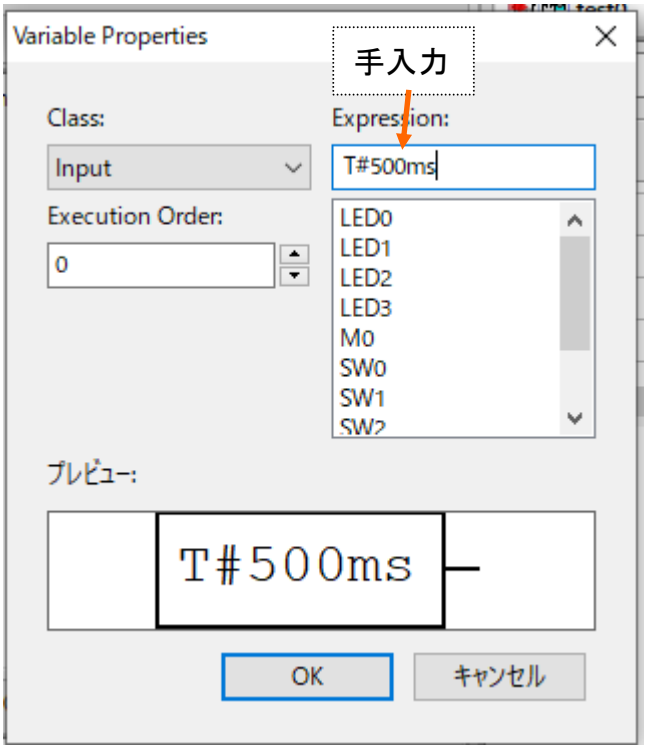




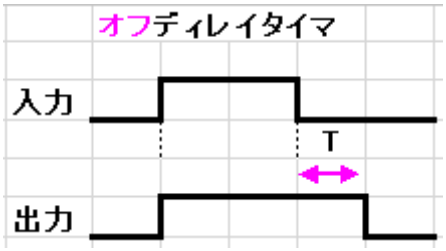




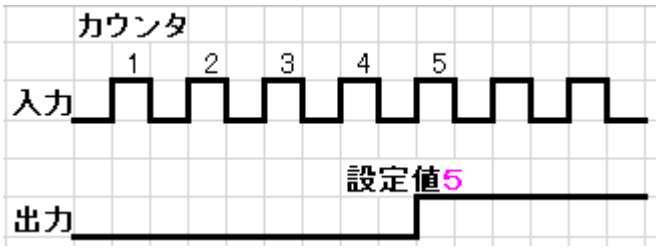
追加＞Variable

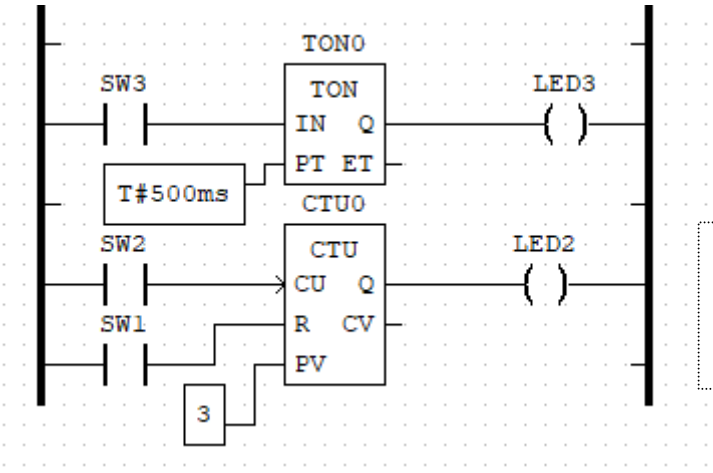


オフディレイタイマも試してみる。（センサーライトに使える。）



カウンタ回路





カウンタ
SW2：カウント入力
SW1:リセット

Open PLC function blocks　の説明　リンクは
https://hashi-rei-channel.hatenablog.com/entry/2022/06/18/102955

例題　1．自己保持回路の応用

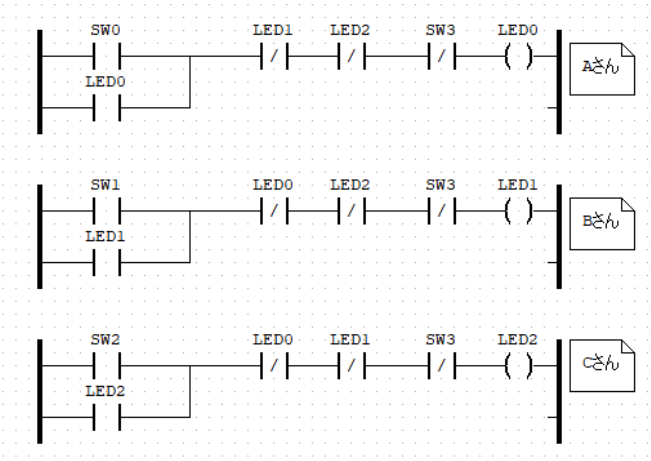
・クイズ３人用早押し回路

SW0　A さん　LED0

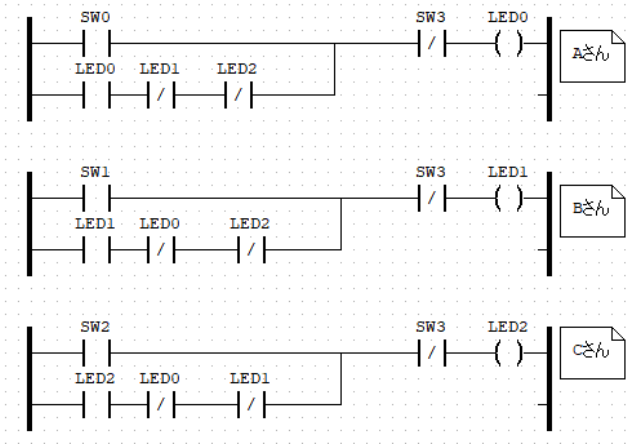
SW1　B さん　LED1

SW2　C さん　LED2

SW3　リセット

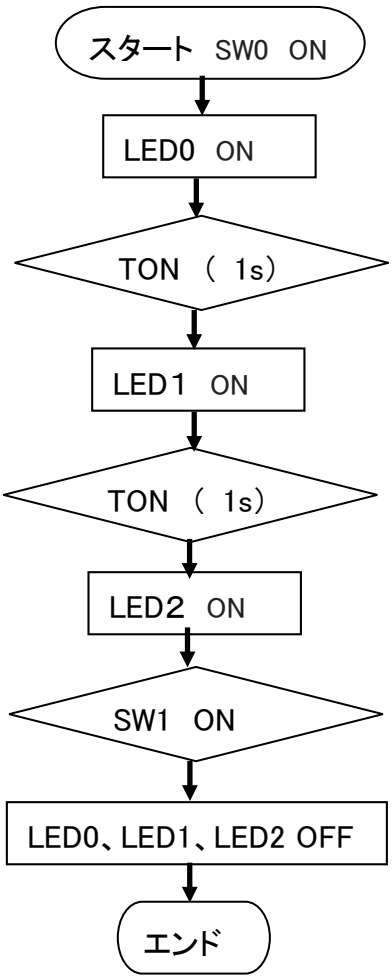


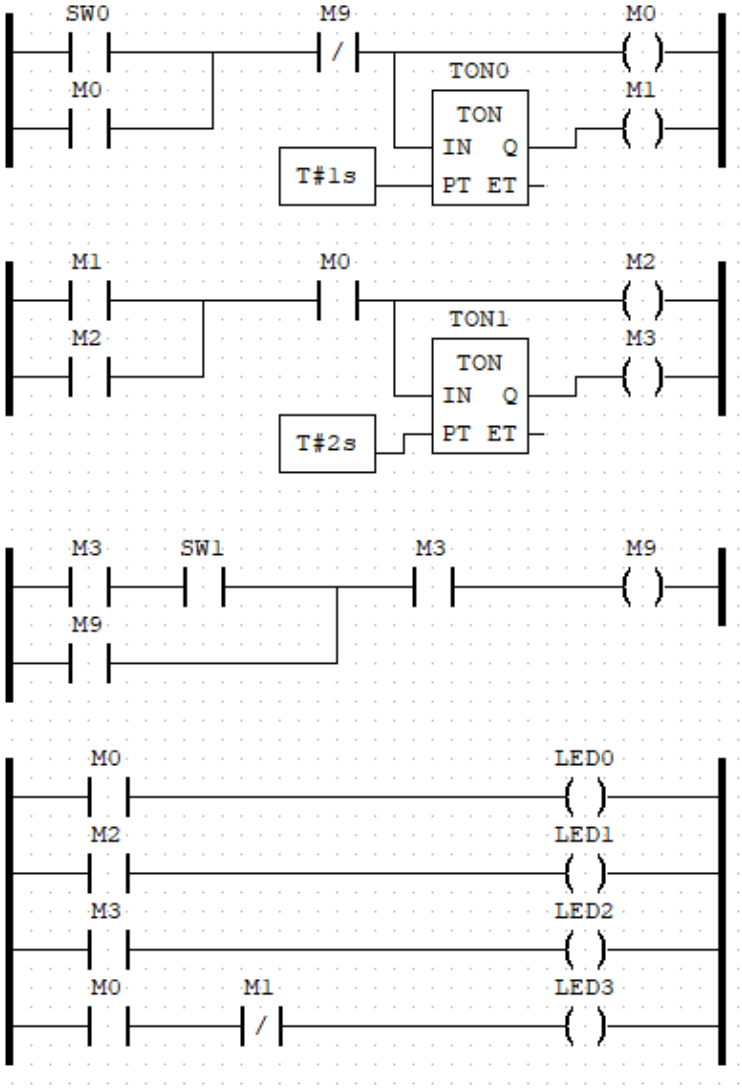
これはどうなるかな？

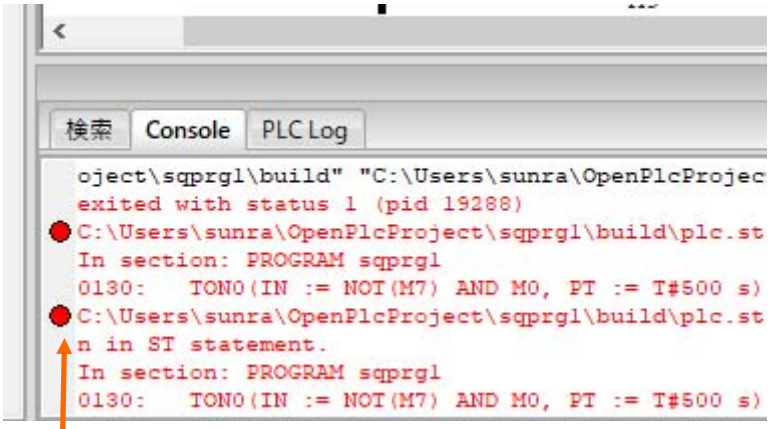


例題　2. 順序回路

・SW0 押したとき　LED0　→　LED1　→　LED2　を順に点灯させる。
（1s）　（1s）　SW1 を押してリセットする。

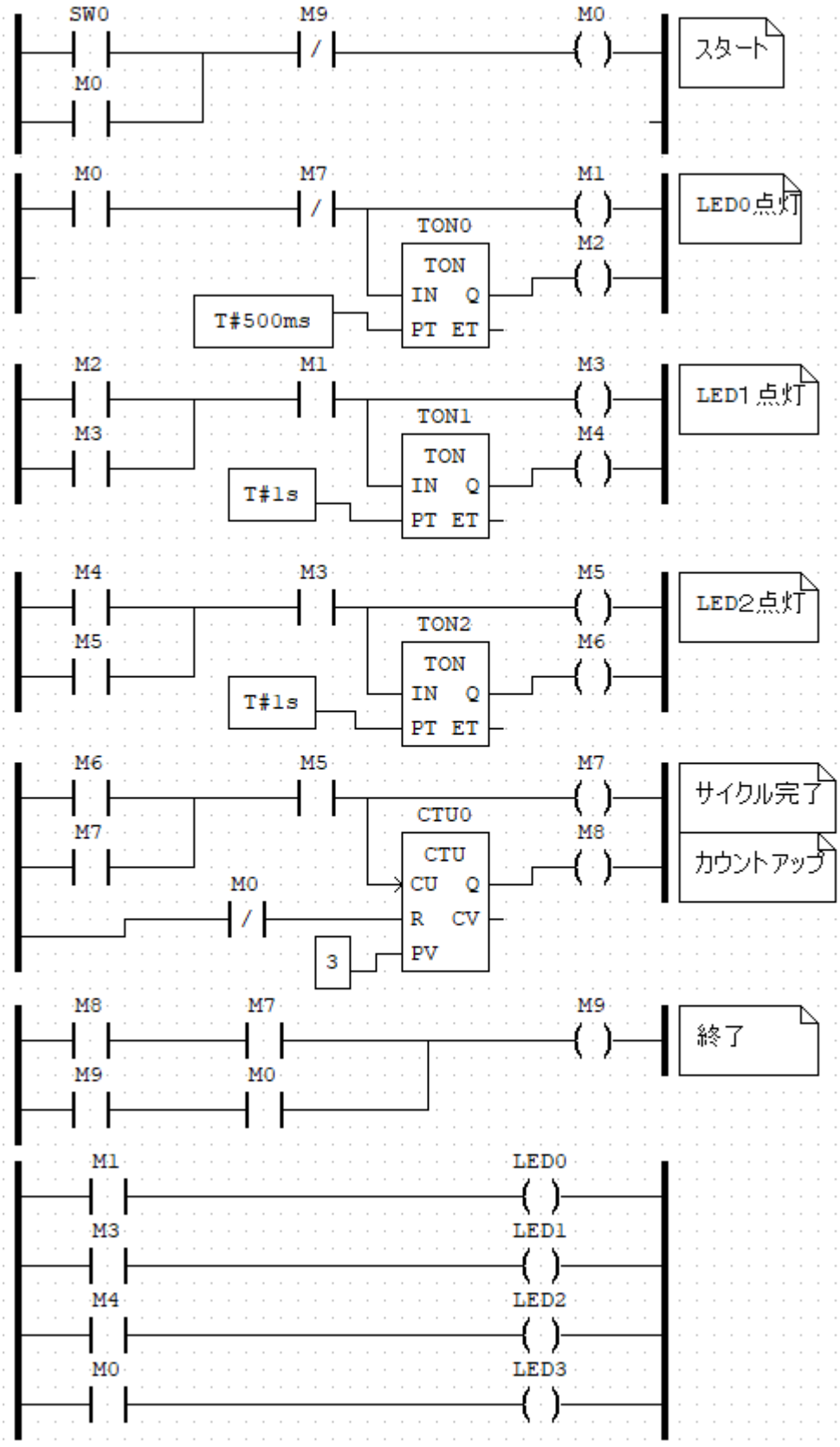






コンパイルエラーが出たとき　ここをダブルクリックすると　その場所がわかる。

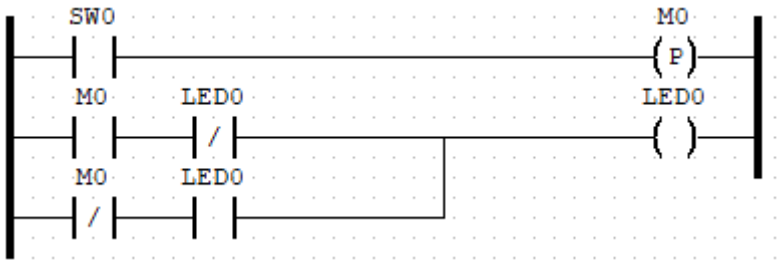
カウンタを使用した順序回路（シーケンス回路）



スタートの SW0は立ち上がりパルスにしたほうが良い

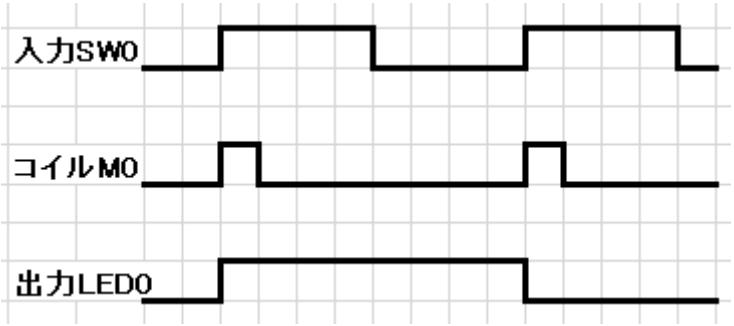
パルス回路でできること

立ち上がりパルス



SW0 を押すたびに LED0 は？？

スキャンの説明



シーケンス「あらかじめ決められた順序で処理を行うこと」 順序回路
PLC（Programmable Logic Controller）、シーケンサー（三菱電機の登録商標）
洗濯機のシーケンス sentakuki

スタート 給水 洗い 排水 給水 すすぎ 排水 脱水 ストップ

スタート SW0

給水 LED0　　給水弁 LED0 ON 水槽上限 SW1 給水弁 LED0 OFF

洗い LED1　　洗い LED1 ON タイマー3分 洗い LED1 OFF

排水 LED2　　排水弁 LED2 ON タイマー1分 排水弁 LED2 OFF

給水 LED0　　給水弁 LED0 ON 水槽上限 SW1 給水弁 LED0 OFF

すすぎ LED1　　すすぎ LED1 ON タイマー3分 すすぎ LED1 OFF

排水 LED2　　排水弁 LED2 ON タイマー1分 排水弁 LED2 OFF

脱水 LED3　　脱水 LED3 ON タイマー1分 脱水 LED3

ストップ

クレーンゲーム

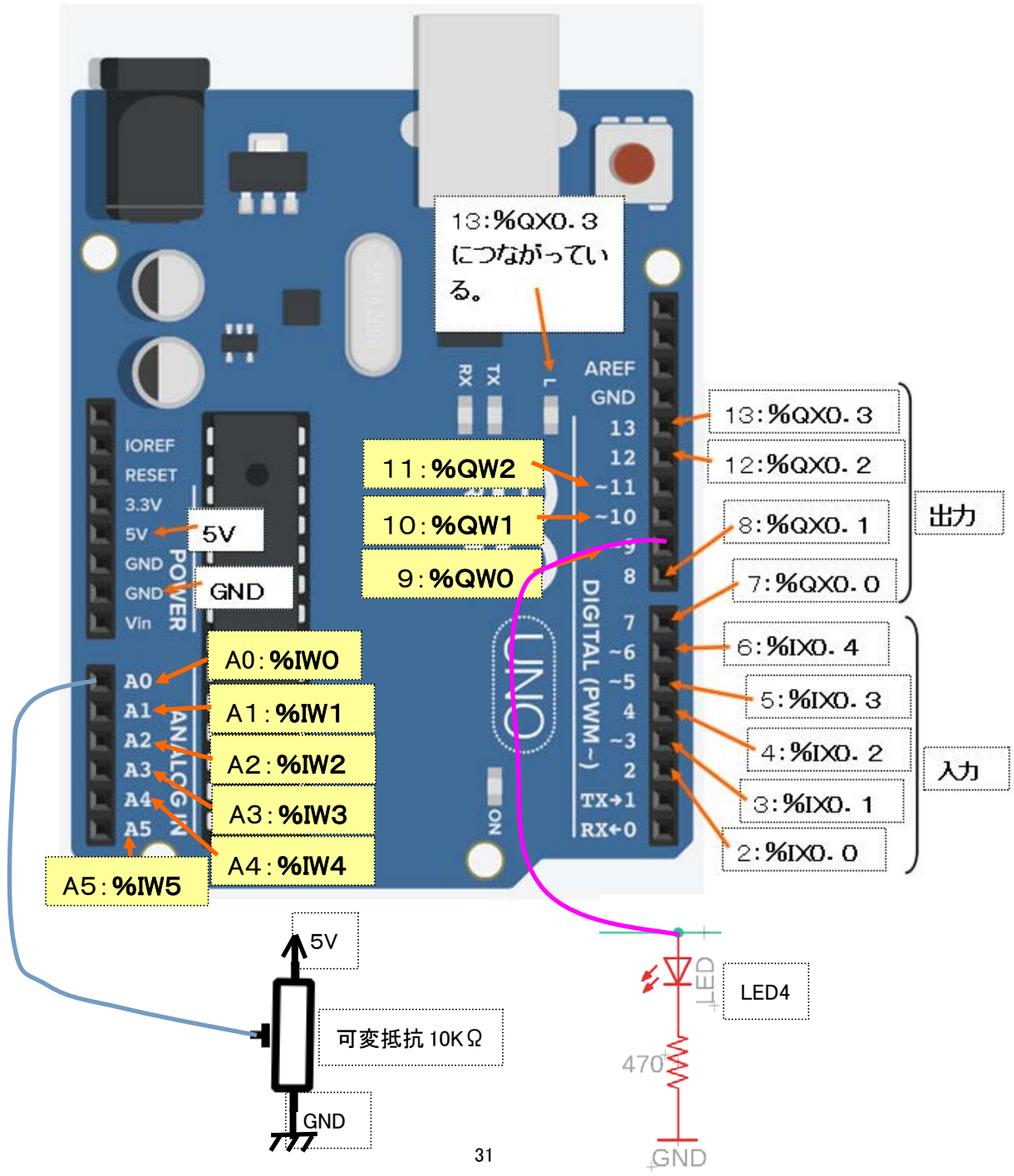
① SW0 を押している間　右に移動 （LED0 が右移動モータとして）
② SW1 を押している間　前に移動 （LED1が前移動モータとして）
③ SW2を押して一連の動作。
　下降（LED2　ON） ⇒ タイマー2秒待つ ⇒ 下降停止（LED2　OFF）
　⇒ つかむ（LED3　ON） ⇒ SW3 を押して終了。

アナログ入出力

IO 割付けは下記リンク（2.4PHYSICAL ADDRESSING）から

https://openplcproject.com/docs/2-4-physical-addressing/

Analog In	ADC 6 本 10bit	A0, A1, A2, A3, A4, A5	%IW0 - %IW5
Analog Out	DAC なし PWM 出力6本の内3本 使用可（OpenPLC では）	9, 10, 11	%QW0 - %QW2



アナログ出力

データ型 （詳細は http://www.musashinodenpa.com/arduino/ref/ ）

INT： −32768～32767

UINT(unsigned int)：0～65535

23	A0	Local	UINT	%IW0
24	LED4	Local	UINT	%QW0

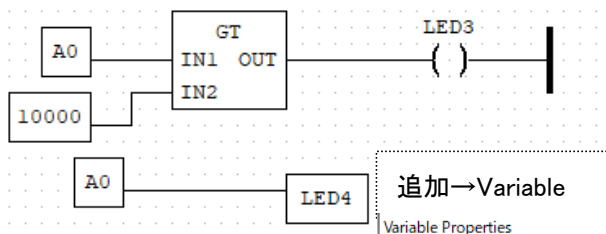
比較演算　GT：IN1>IN2 のとき OUT が ON

（ 追加→Block→Comparison→GT ）

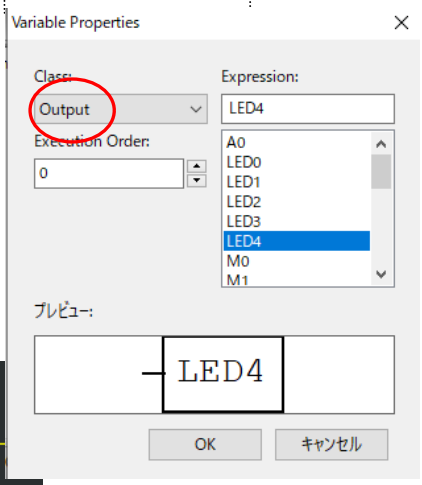
Aruduino uno の ADC は１０bit だが OpenPLC では１６bit に変換してるのかなー？

IN2 の比較値 10000 は

5v/65535 * 10000 = 0.76v



追加→Variable



分解能

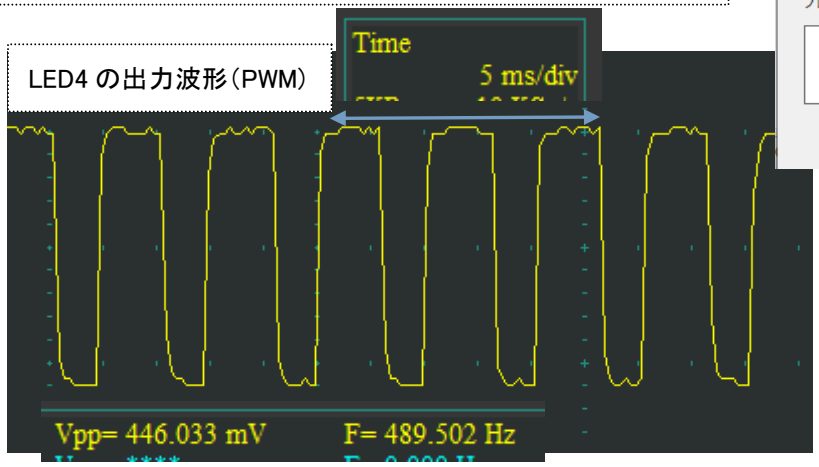
8bit：0～255　5v/256 = 0.0195v (19.5mv)

１0bit：0～１023　5v/1024 = 0.00488v (4.88mv)

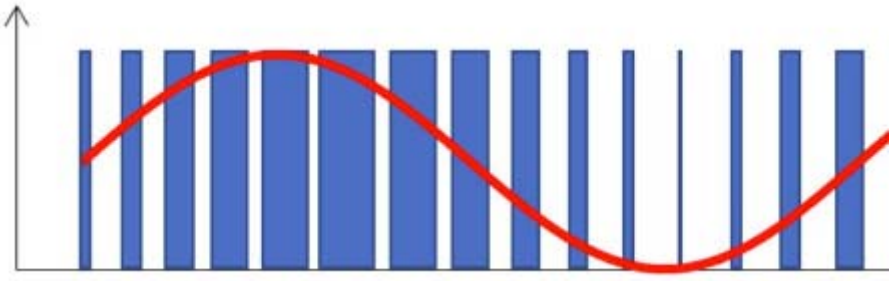
１2bit：0～4095　5v/4096 = 0.00122v (1.22mv)

１6bit：0～65535　5v/65536 = 0.0000763v (76.3uv)

LED4 の出力波形（PWM）



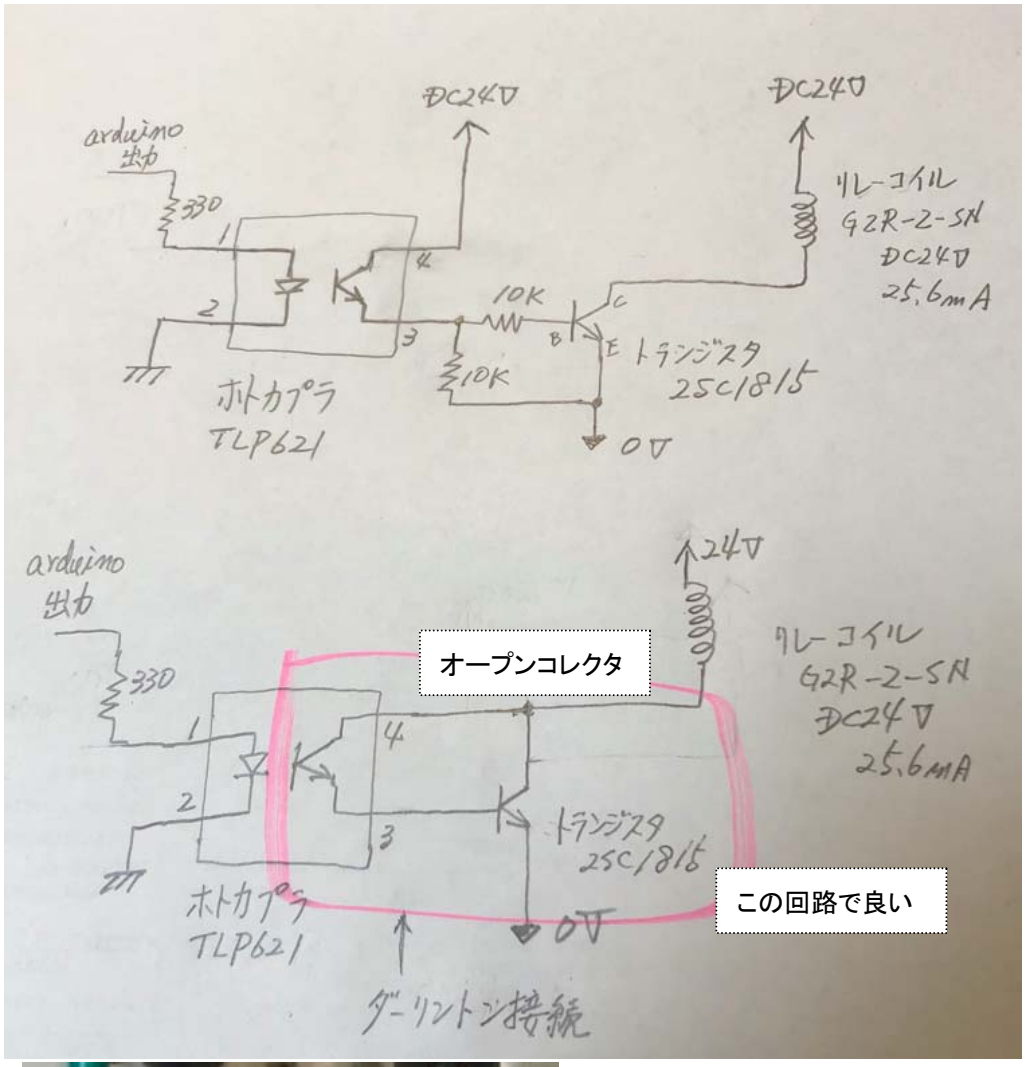
PWM の意味は(Pulse Width Modulation) パルス幅変調

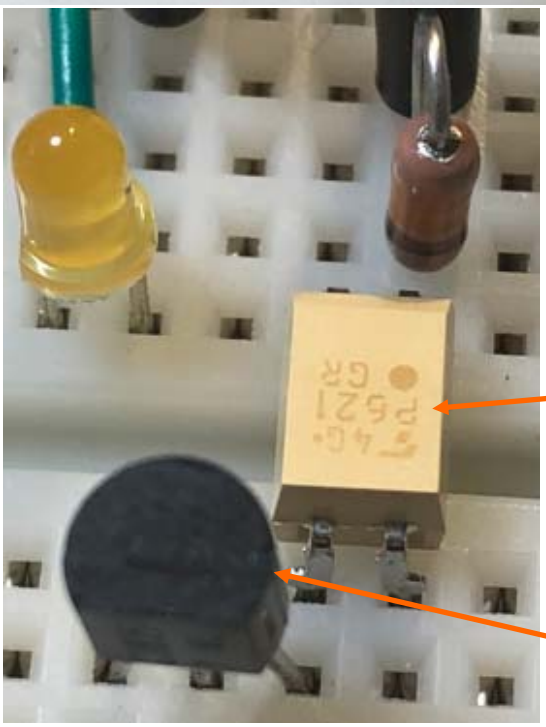


リンク https://jp.mathworks.com/discovery/pulse-width-modulation.html

・アイソレート IO　（isolate　分離）

Arduino　の入出力は DC5V 回路です。DC24V（12V）で動作させるにはアイソレート IO 回路が必要です。





ホトカプラ　TLP621

トランジスタ　2SC1815

参考　秋月電子

ラズパイＰＬＣ用ＤＣ２４Ｖアイソレート入／出基板パーツセット

https://akizukidenshi.com/catalog/g/gK-15645/

アナログ 4-20mA　（**ヨンニジュウ**）
例
温度　0℃　4mA
　　50℃　12mA
　　100℃　20mA

メリット
・長距離伝送でも減衰しにくい。
・断線検出が出来る。（0mA になるため）
・受け側の電流から電圧変換回路は抵抗250Ωで簡単にできる。（1v～5v）
デメリット
・2つ以上の入力機器を接続するときはループで接続。ループが断線すると、どの入力機器側でも0mA となる。
・テスターでの測定が困難。ループを切り離す必要がある。

インバータの高調波対策について
https://www.inverter.co.jp/member/teclib/inv/data/man/e6580982/e6580982.pdf

＊＊日目
分かりやすい回路を心掛ける。コメントをかく
自分で作った回路も忘れる。変更にそなえて
テクニック
大きなシーケンスの回路構成
客先により仕様がある。
現場での対応　シーケンス変更、デバッグ　トイレに行って頭を冷やす。

ハード

入力回路
出力回路　ホトカプラ、リレー、トランジスタ、電磁開閉器、電磁接触器、
接続する IO の仕様を
定格、抵抗負荷、コイル負荷、ダイオード、寿命は
コイル、コンデンサ、アイソレート
ノイズ対策、アースの取り方、シールド、
サージ対策
制御盤内の写真
有接点リレーシーケンスとの違い
PLC の変遷:
PLC メーカ
三菱:シーケンサ
オムロン、キーエンス、シャープ、安川
開発環境

火を入れるときの心得
確認しながら順番に
アイソレート IO 基板　実際に電磁弁、モータを動かすためには

- マイコンチップ:ATmega328P
- 動作電圧:5V
- 入力電圧（推奨）:7～12V （DCジャックもしくはVIN端子から入力）
- デジタルI/Oピン:14本（うち6本はPWM出力可能）
- アナログ入力ピン:6本（デジタルI/Oピンとしても利用可能）
- DC出力電流:1つのI/Oピン当り20mA程度、I/Oピン全部の合計100mAまで（1ピンあたり40mA以上流すと壊れます）
- DC出力電流:3.3V出力ピン 50mA
- Flashメモリ:32KB （うち0.5KBをブートローダーで使用）
- SRAM:2KB
- EEPROM:1KB
- クロックスピード:16MHz

入力回路

プルアップ（p117）、プルダウン(p117)、チャタリング（p125）、フィルター（ハード、ソフト）

閾値電圧（しきいちでんあつ、V_{th}、英語：Threshold voltage、スレッショルド電圧、スレッシュホールド電圧）とは、デジタル信号を H（もしくは 1）/L（もしくは 0）信号として検知するのに必要となるしきい値となる電位のことである。

スレショルドレベル

しきい値（閾値）． ディジタル信号の入力回路が，H（High）レベル，L（Low）レベルを確定入力として検出する限界の電圧のこと． TTL では 0.8V 以下を L レベル，2.0V 以上を H レベルと解釈するので，駆動側でこの条件を満たす必要がある．

アイソレート（絶縁分離）

ホトカプラ、

二日目

メーカ製 PLC の接続は

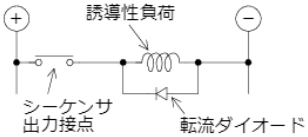
6. 出力仕様と外部配線

《出力回路の構成》

本製品のリレー出力回路には、リレー用内部保護回路を設けていません。誘導性負荷を使用するばあいは、保護回路内蔵のものを使用することをお勧めします。

保護回路を内蔵していない負荷を使用するばあいは、ノイズ軽減, 寿命のため外部に接点保護回路などを挿入してください。

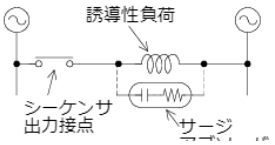
直流負荷



負荷と並列にダイオードを接続してください。
ダイオード(転流用)は、下記の仕様のものを使用してください。

項目	目安
逆電圧	負荷電圧の5~10倍
順電流	負荷電流以上

交流負荷



負荷と並列にサージアブソーバ(サージキラーやスパークキラーなどのCR複合部品)を接続してください。サージアブソーバの定格電圧については、使用している出力にあったものを選定してください。その他の仕様は下表を参照してください。

はじめてのシーケンサ　YouTube（約 41 分）

1/19(1:54)　https://www.youtube.com/watch?v=qTaGbKbzZZA

2/19(2:05)　https://www.youtube.com/watch?v=hi99gjbDDPM

3/19(4:18) 1.シーケンス制御とは　-　シーケンス制御を構成するもの　〈はじめてのシーケンサ(3/19)〉　- YouTube

4/19(2:25) 1.シーケンス制御とは　-　配線をしてみましょう　〈はじめてのシーケンサ(4/19)〉　- YouTube

5/19(2:38) 1.シーケンス制御とは　-　配線図やシーケンス図　〈はじめてのシーケンサ(5/19)〉　- YouTube

6/19(0:32)　2.シーケンサとは　-　シーケンサとは　〈はじめてのシーケンサ(6/19)〉　- YouTube

7/19(0:43)　2.シーケンサとは　-　シーケンサのしくみ　〈はじめてのシーケンサ(7/19)〉　- YouTube

8/19(2:40)　2.シーケンサとは　-　シーケンサの配線　〈はじめてのシーケンサ(8/19)〉　- YouTube

9/19(1:10) 2.シーケンサとは　-　シーケンサのプログラム　〈はじめてのシーケンサ(9/19)〉　- YouTube

10/19(0:37)　3.GX Works2　-　GX Works2 の操作　〈はじめてのシーケンサ(10/19)〉　- YouTube

11/19(1:12) 3.GX Works2　-　ソフトウェアの起動と画面構成　〈はじめてのシーケンサ(11/19)〉　- YouTube

12/19(2:08)　3.GX Works2　-　回路の作成　〈はじめてのシーケンサ(12/19)〉　- YouTube

13/19(2:33) 3.GX Works2　-　シーケンサへのプログラムの書込み　〈はじめてのシーケンサ(13/19)〉　- YouTube

14/19(2:09)　3.GX Works2　-　動作の確認　〈はじめてのシーケンサ(14/19)〉　- YouTube

15/19(2:40) 4.シーケンス命令について　-　プログラムのしくみ　〈はじめてのシーケンサ(15/19)〉　- YouTube

16/19(3;25) 4.シーケンス命令について　-　命令の使い方を覚えよう　〈はじめてのシーケンサ(16/19)〉　- YouTube

17/19(1:29)　5.プログラム演習　-　プログラム演習　〈はじめてのシーケンサ(17/19)〉　- YouTube

18/19(2:29)　5.プログラム演習　-　エスカレータの制御　〈はじめてのシーケンサ(18/19)〉　- YouTube

19/19(3:12)　おわりに　〈はじめてのシーケンサ(19/19)〉　- YouTube

はじめてのシーケンサ pdf
https://www.mitsubishielectric.com/fa/assist/satellite/data/jy997d41501e.pdf

GX developer 体験版（７日間）
GX Works2 体験版（３０日間）　で検索して　体験版をインストール
（ログインが必要）

キーエンス　KV STUDIO 体験版（５０回起動）

私の事例
自動供給取出し機
バームクーヘンを焼く機械

ハード、ソフト
回路図をいれて

エアー回路の電磁弁は

リレーコイルの保護回路は

アナログ 4-20mA https://memo-labo.com/keiso.php

機械製造の流れ

お客様と打ち合わせ
仕様書の確認　耐油、耐熱、防塵、配線仕様（フレキ配線等）、マークチューブの方向、塗装色、制御盤防塵防滴仕様
見積りの為の制御設計　購入部品リスト作成　見積書提出
交渉
制御設計（ハード）展開接続図、制御盤、操作盤、部品表
制御設計（ソフト）ラダーソフト作成（紙に書いて）→　パソコンでラダー打ち込み
機内配線（配線仕様があるので注意）
電源投入
I0 チェック
デバッグ　順番に　機械が思わぬ動きをするので
異常回路の確認
タクトタイム短縮
立ち合い検査納品
現地設置工事
機械稼働デバッグ　オンライン書き込み
検収

機械の改造の場合
図面の確認

Arduino IDE

L チカ

https://www.youtube.com/watch?v=4Th6xpR-DrE

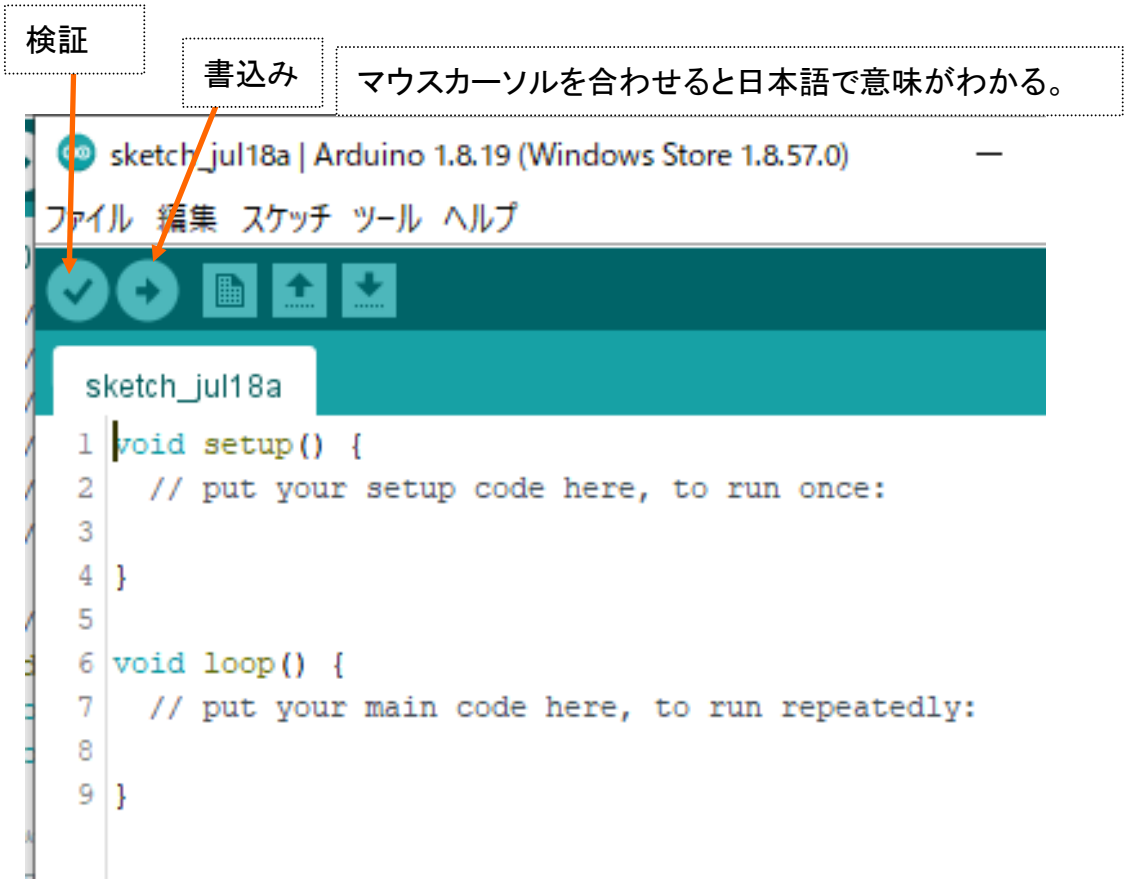
Arduino 日本語リファレンス

http://www.musashinodenpa.com/arduino/ref/

言語・関数

https://spiceman.jp/arduino-function-reference/

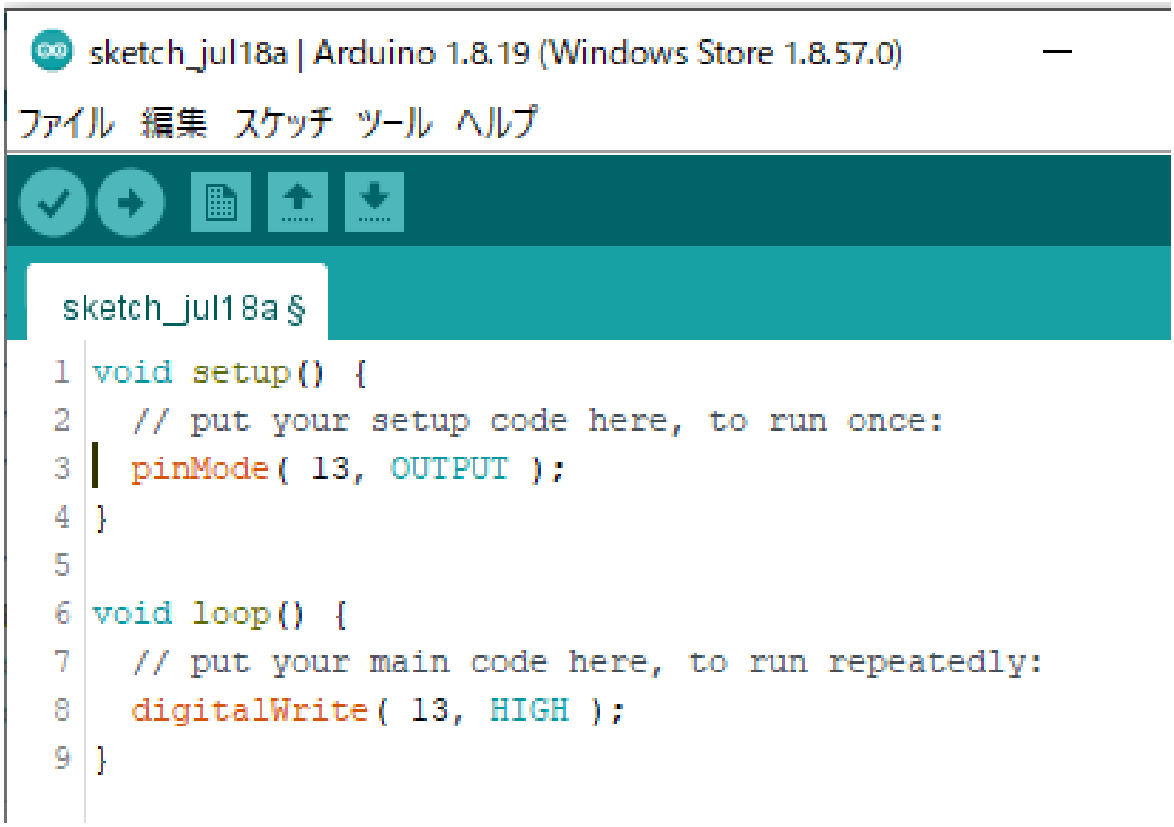
プログラムを作ってみる。(p58)



プログラムのことをスケッチと言う。

```
void setup() {
   //  一回だけ通る
}

Void loop() {
   //  繰り返し実行
}
```



ボードの設定

ツール → ボード → Arduino Uno

COM ポート番号の確認

ツール → シリアルポート → COM3 チェック

 書込み

点滅

プログラム（スケッチ）の基本構成

宣言文 変数や関数を定義する

Void setup() {
　　//初めに 1 回だけ実行
}

Void loop() {
　　//繰り返し実行
}

シリアルモニタで情報を表示する。

実行して シリアルモニタでモニタウインドウを開く

または ツール ＞ シリアルモニタ

ボーレートを合わせること

変数を使う

sketch_jul18a

```
//Arduino IDE ではプログラムのことをスケッチと言う。
//宣言文
//変数や関数を定義する
int SW0 = 2;
int SW1 = 3;
int SW2 = 4;
int SW3 = 5;
int SW4 = 6;
int LED0 = 7;
int LED1 = 8;
int LED2 = 12;
int LED3 = 13;

//初めに1回だけ実行
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode( SW0, INPUT );
  pinMode( SW1, INPUT );
  pinMode( SW2, INPUT );
  pinMode( SW3, INPUT );
  pinMode( SW4, INPUT );
  pinMode( LED0, OUTPUT );
  pinMode( LED1, OUTPUT );
  pinMode( LED2, OUTPUT );
  pinMode( LED3, OUTPUT );
}

//繰り返し実行
void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("From Arduino Message.");
  digitalWrite( LED0, HIGH );
  digitalWrite( LED1, LOW );
  delay(500);
  digitalWrite( LED0, LOW );
  digitalWrite( LED1, HIGH );
  delay(500);
}
```

if 文 を使う

```
//繰り返し実行
void loop() {
  // put your main code here, to run repeatedly:
  SW0_val = digitalRead(SW0);  // 入力ピンを読む
  //digitalWrite( SW0_val );//error になる
  //if( SW0_val ){
  if( SW0_val==1 ){
    digitalWrite( LED0, HIGH );
  }
  else{
    digitalWrite( LED0, LOW );
  }
  Serial.print("SW0_val=");
  Serial.println(SW0_val);
// digitalWrite( LED0, HIGH );
  digitalWrite( LED1, LOW );
  delay(500);
//digitalWrite( LED0, LOW );
  digitalWrite( LED1, HIGH );
  delay(500);
}
```

シリアル入力を使う

sketch_jul18a

```
//Arduino IDE ではプログラムのことをスケッチと言う。
//宣言文
//変数や関数を定義する
int SW0 = 2, SW0_val = 0;
int SW1 = 3, SW1_val = 0;
int SW2 = 4, SW2_val = 0;
int SW3 = 5, SW3_val = 0;
int SW4 = 6, SW4_val = 0;
int LED0 = 7;
int LED1 = 8;
int LED2 = 12;
int LED3 = 13;

int incomingByte = 0;  // 受信データ用

//初めに1回だけ実行
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode( SW0, INPUT );
  pinMode( SW1, INPUT );
  pinMode( SW2, INPUT );
  pinMode( SW3, INPUT );
  pinMode( SW4, INPUT );
  pinMode( LED0, OUTPUT );
  pinMode( LED1, OUTPUT );
  pinMode( LED2, OUTPUT );
  pinMode( LED3, OUTPUT );
}

//繰り返し実行
void loop() {
  if (Serial.available() > 0) { // 受信したデータが存在する
    incomingByte = Serial.read(); // 受信データを読み込む

    Serial.print("I received:(HEX) "); // 受信データを送りかえす
    Serial.print(incomingByte, HEX);
    Serial.print(" (DEC)");
    Serial.println(incomingByte, DEC);
  }
}
```

説明:

for　next 文

switch case 文

Arduino 日本語リファレンス

http://www.musashinodenpa.com/arduino/ref/

言語・関数

https://spiceman.jp/arduino-function-reference/

Arduino で検索したらプログラムのやり方がいっぱい出てくる。
いろいろ試してください。