

---

# Arduino使い方ガイド    rev.8

# 本テキストの目的

---

- 学習テーマのプログラム作成のための基礎知識を身に着ける
  - Arduinoでマイコンプログラムを作れるようになる
  - LED, タクトスイッチが使用できる
  - ライブラリを使って時間制御ができるようになる

# 目次

---

## 1. 開発環境を準備しよう

Arduinoの概要、IDEのインストール

## 2. Arduinoを動かしてみよう

IDEの初期設定、Arduinoの接続、サンプルプログラム（LED点滅）を動かす

## 3. 初めてのArduinoプログラミング

スケッチを作る、タブを作る

サンプルプログラムを動かす

## 4. マイコンで、LED点灯制御してみよう

デジタル入出力ポートの制御、LED点滅制御プログラムを動かす  
シリアル通信を使う

## 5. マイコンで、タクトスイッチを使ってみよう

デジタル入力制御、タクトスイッチを読み込む  
スイッチ入力時のチャタリング制御

## 6. タイマを使って、時間をプログラミングしよう

Arduinoのライブラリを使用  
タイマ機能を使った時間制御プログラム

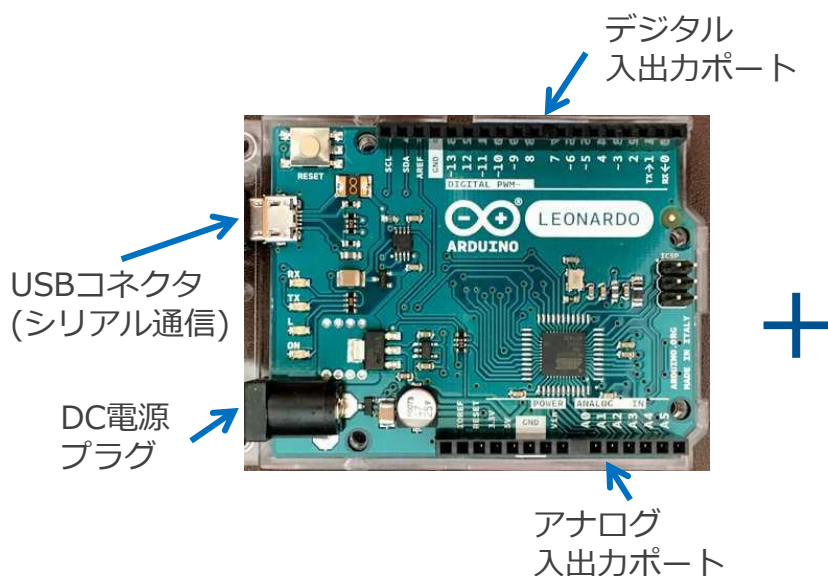
---

# 1. 開発環境を準備しよう

# Arduino(アルデューイーノ)とは

- Arduinoの構成

- マイコンボード
- 統合開発環境 (IDE : Integrated Development Environment)  
エディタ、コンパイラ、ライブラリ を統合したパッケージソフト



Arduino マイコンボード

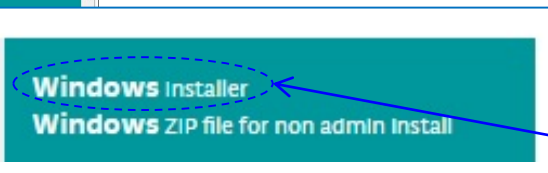
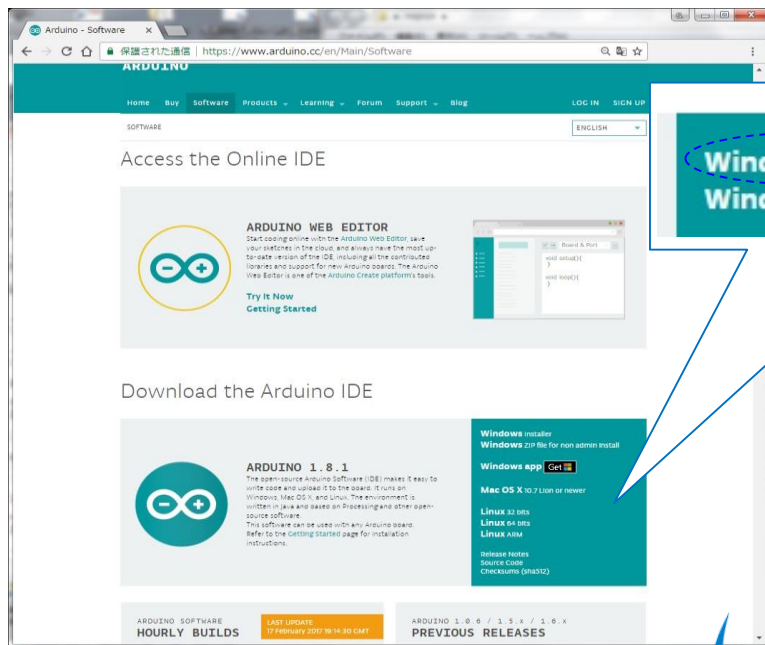


Arduino ソフトウェア  
(統合開発環境)

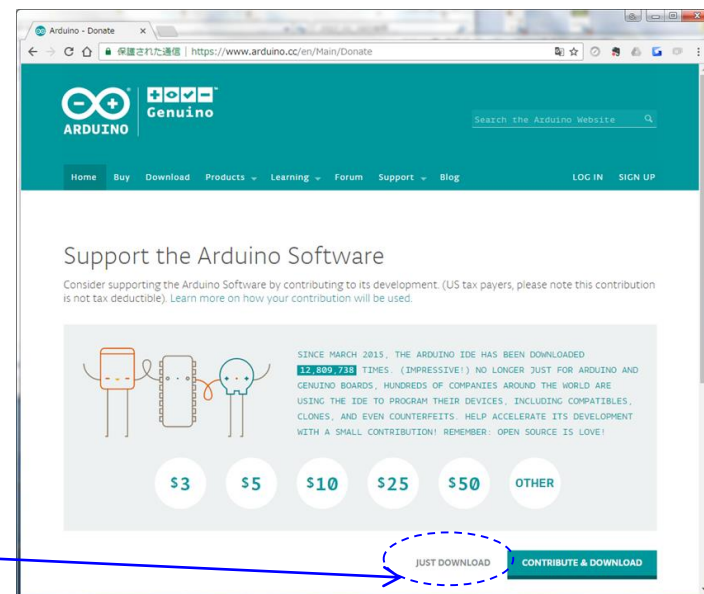
# Arduino ソフトウェアのインストール

- 公式ウェブサイトより、Arduinoソフトウェアをダウンロード

<https://www.arduino.cc/en/main/software>



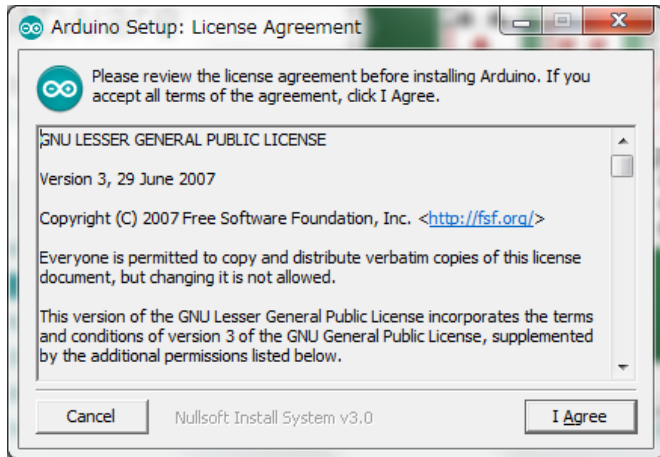
Download the Arduino IDE の  
Windows installer を Click



JUST DOWNLOAD を Click

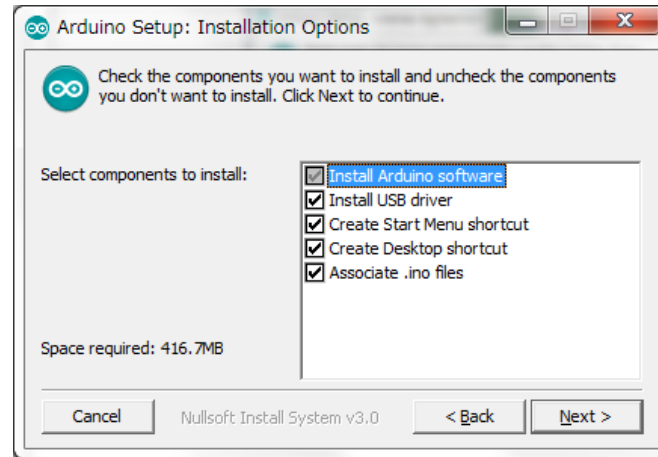
# インストール手順

1.



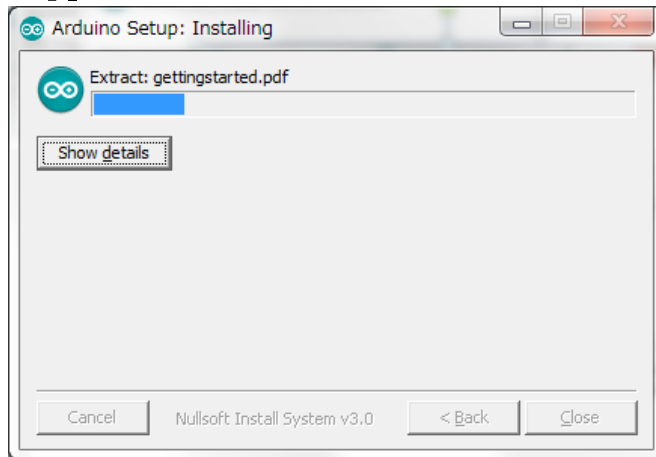
“I Agree” を Clickしてインストールを開始

2.



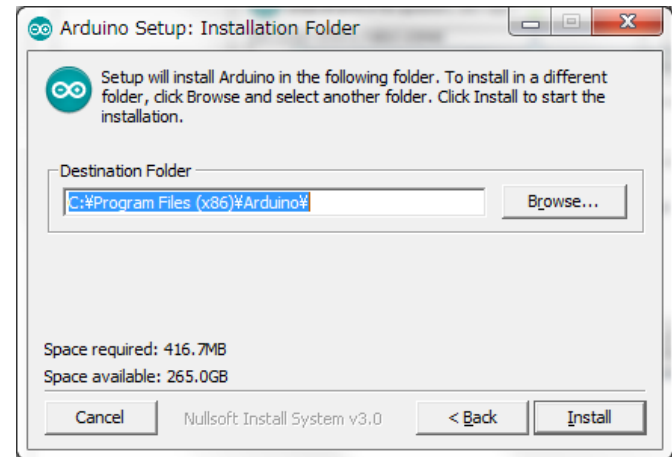
全てチェックを入れ、“Next” を Click

4.



このまま、インストール終了を待つ

3.



“Install” を Click  
(Destination Folderの変更はしない)

---

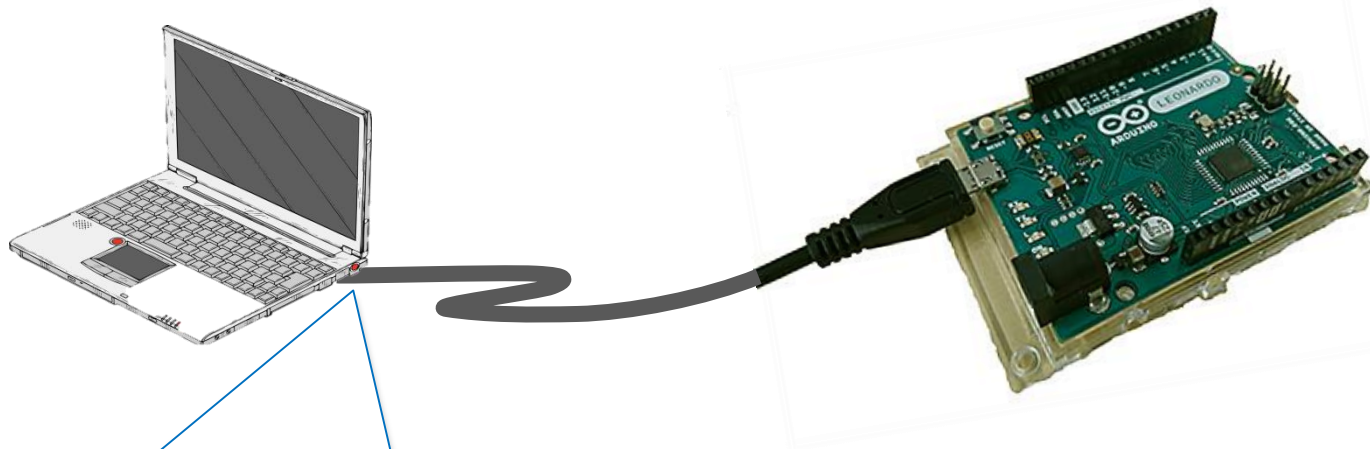
## 2. Arduinoを動かしてみよう



# ArduinoをPC/Macに接続する

- まず, ArduinoをPC (Mac) のUSB端子に接続します.

**注意:ここではまだ, enPiTシールドを挿さないでください**

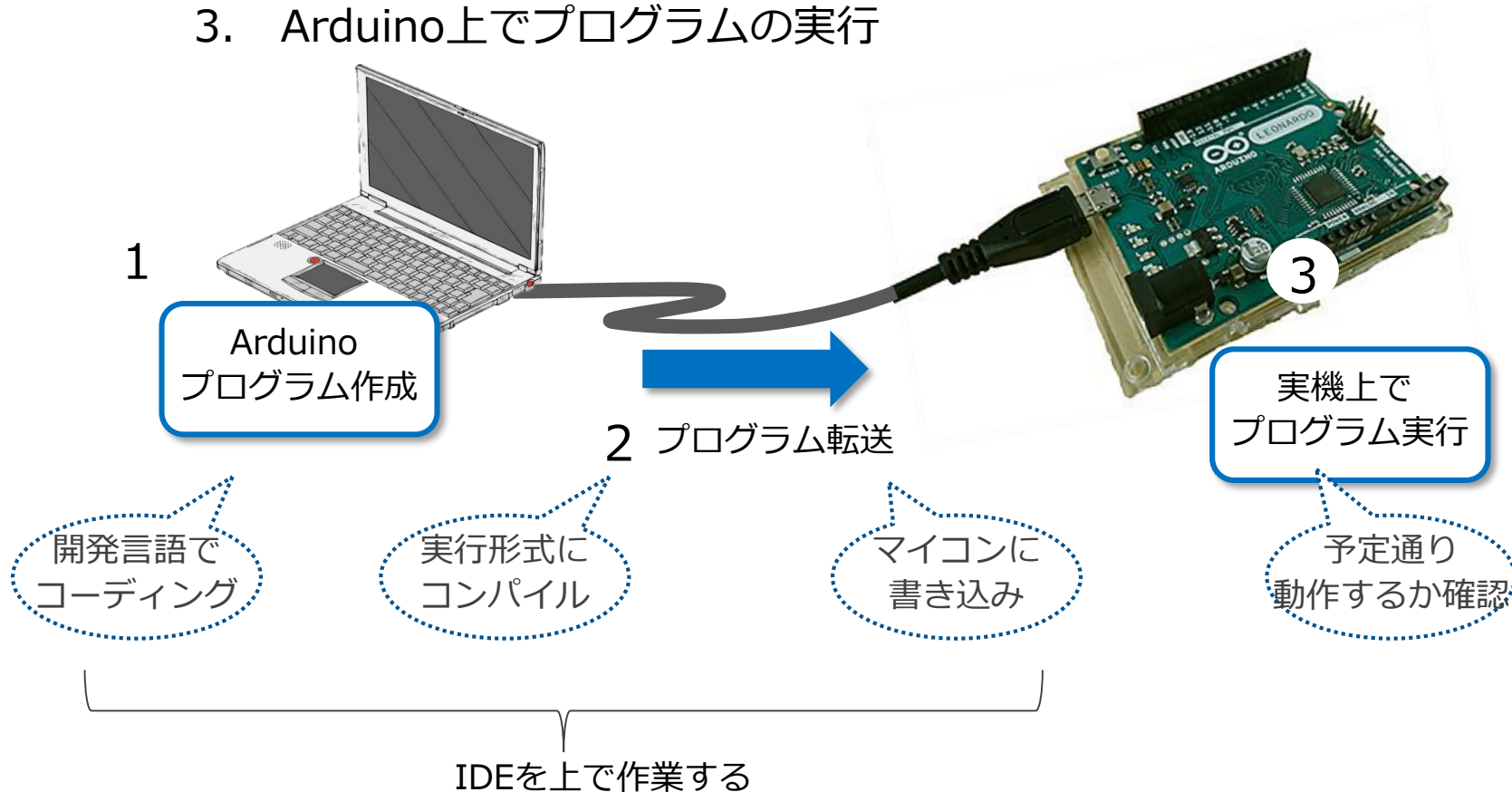


ボードをPCに接続すると、自動的にデバイスドライバのインストールが始まる  
『デバイスドライバ ソフトウェアが正しくインストールされました』が表示されたら完了

# Arduinoでプログラムを動かす

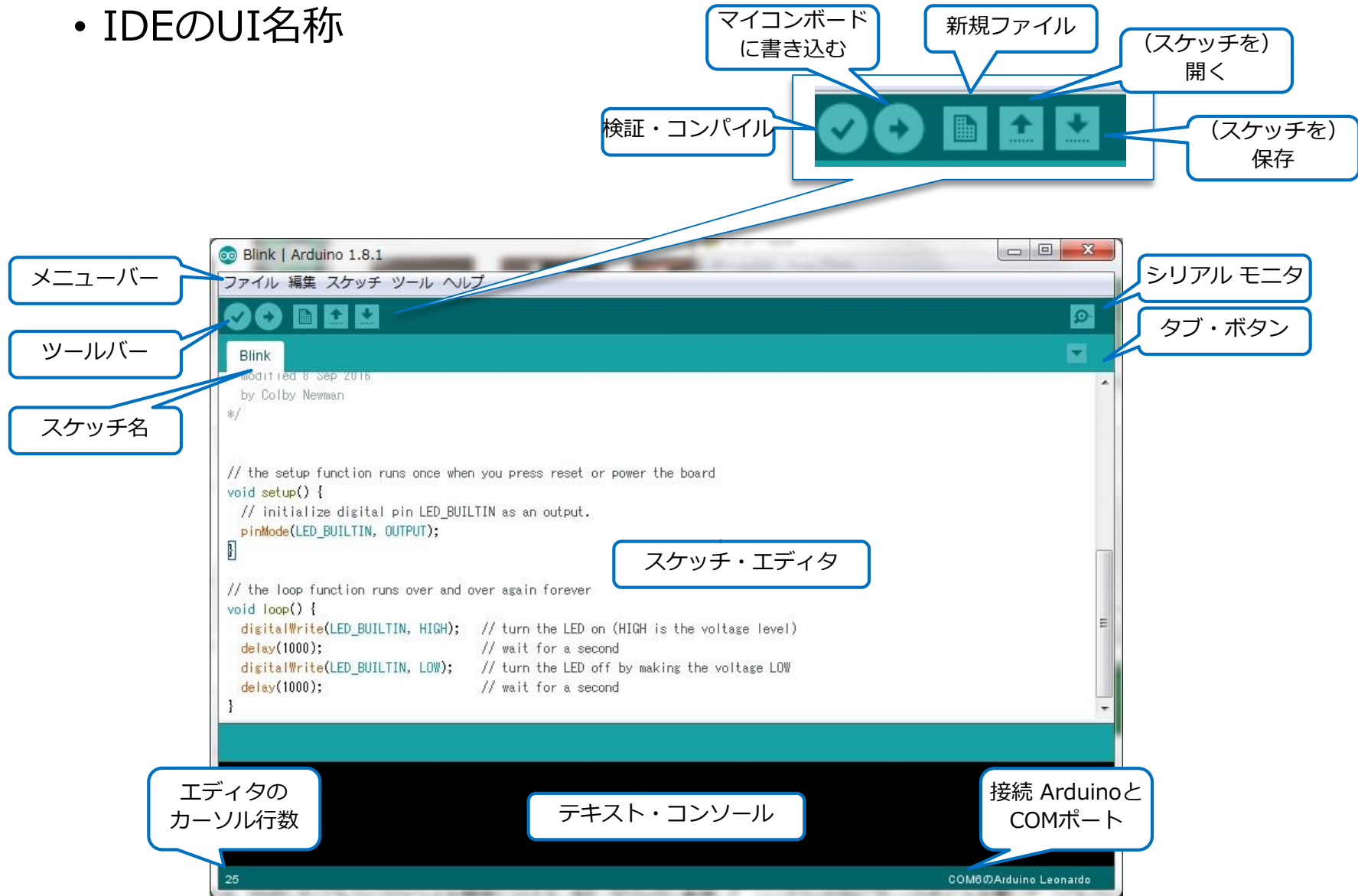
- Arduinoでプログラムを動かす手順は、以下の通り

1. PC上でのプログラムの作成
2. PCからArduinoへのプログラムの転送
3. Arduino上でプログラムの実行



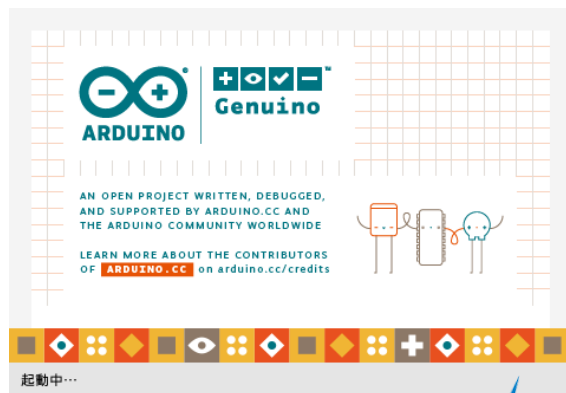
# IDEは何するもの？

- IDEのUI名称

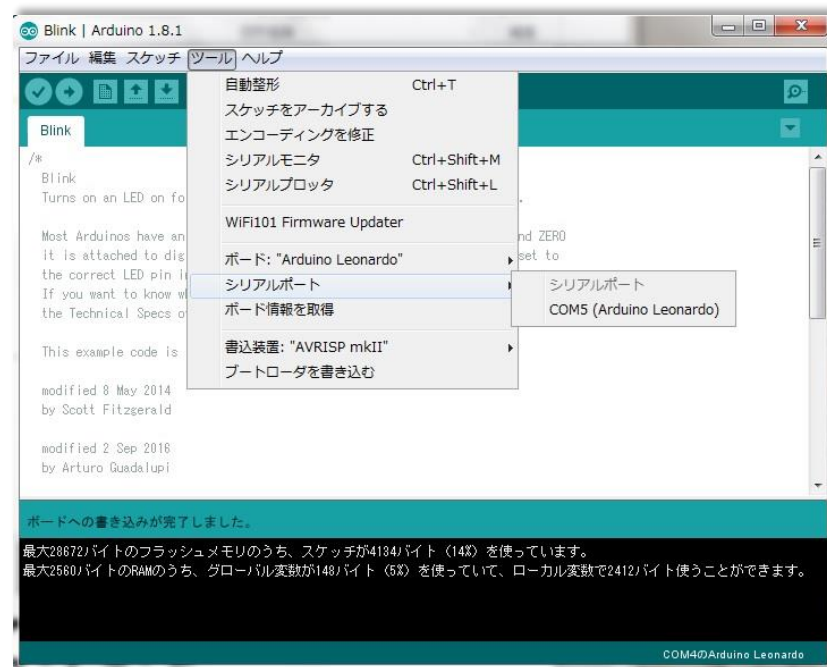


# Arduino IDE 初期設定

- Arduinoを接続して、IDE (arduino.exe) を立ち上げ初期設定する
  - ボード バージョン設定
  - シリアルポート設定



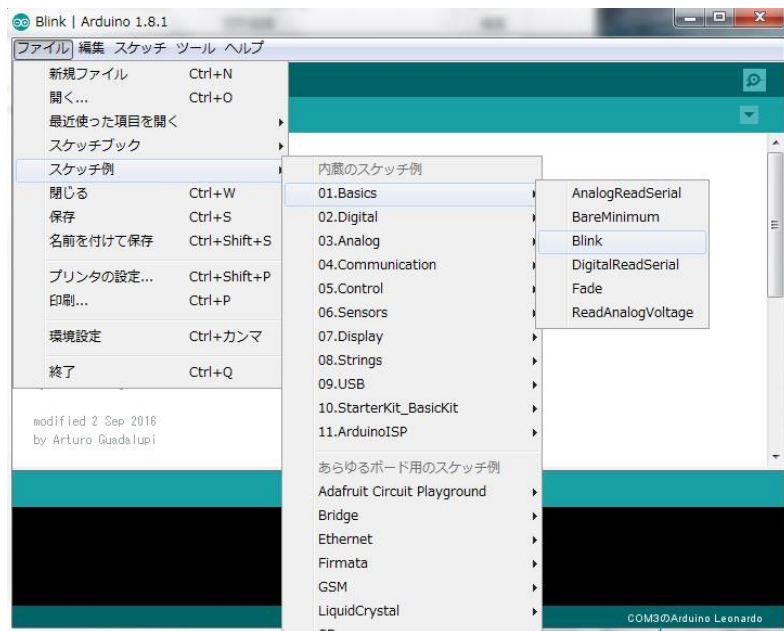
起動画面



エディタ画面

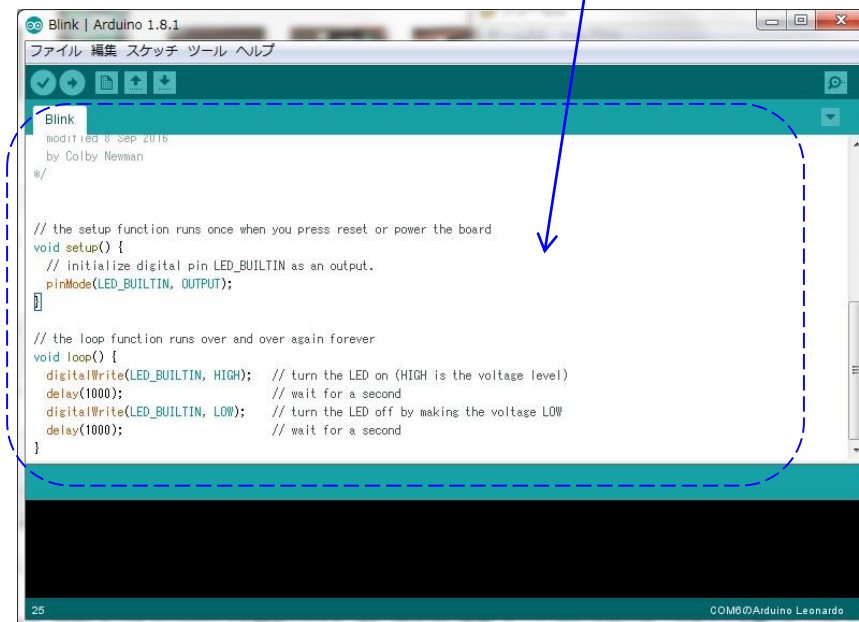
# Arduinoの動作確認-1

- サンプルを使って、Arduinoが動作することを確認
  - スケッチ例 : Blink を読み込む



サンプルコード 読み込み時の画面

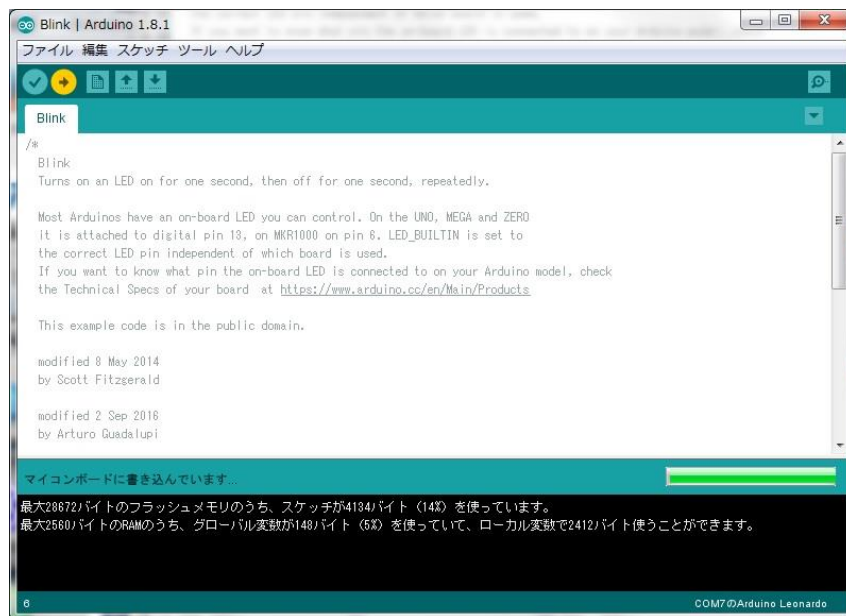
Blink のソースコードが  
エディタに表示される



サンプルコード : Blink を読み込んだ画面

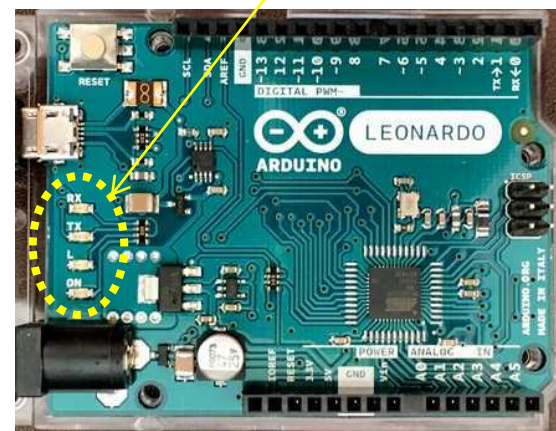
# Arduinoの動作確認-2

- サンプルをボードに書き込み、Arduinoボードが動作することを確認
  1. プログラムのコンパイル & 書き込み & 実行
  2. マイコンボードのLEDが、以下のように動作することを確認
    - マイコンボードに書き込み中 … L, TX, RXがチカチカ
    - Blinkプログラム動作中 … L だけが一定周期で点滅



コンパイル & 書き込み 画面

LED : L, TX, RX  
点灯状態を確認する



マイコンボード



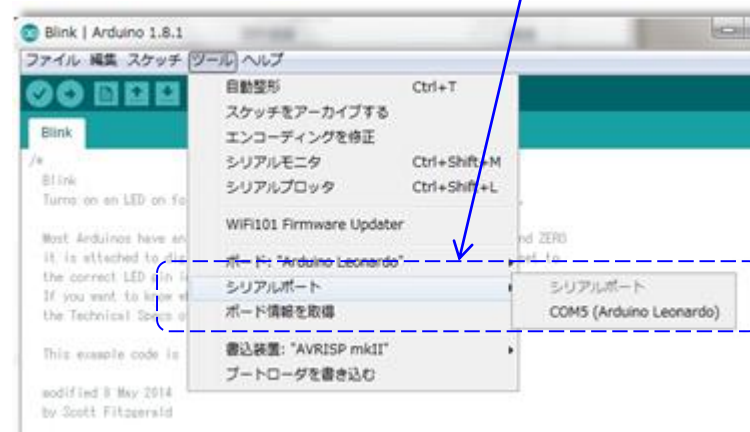
# 注意：転送エラー

- マイコンボードに書き込みでエラーが発生した場合



書き込みエラー発生 画面

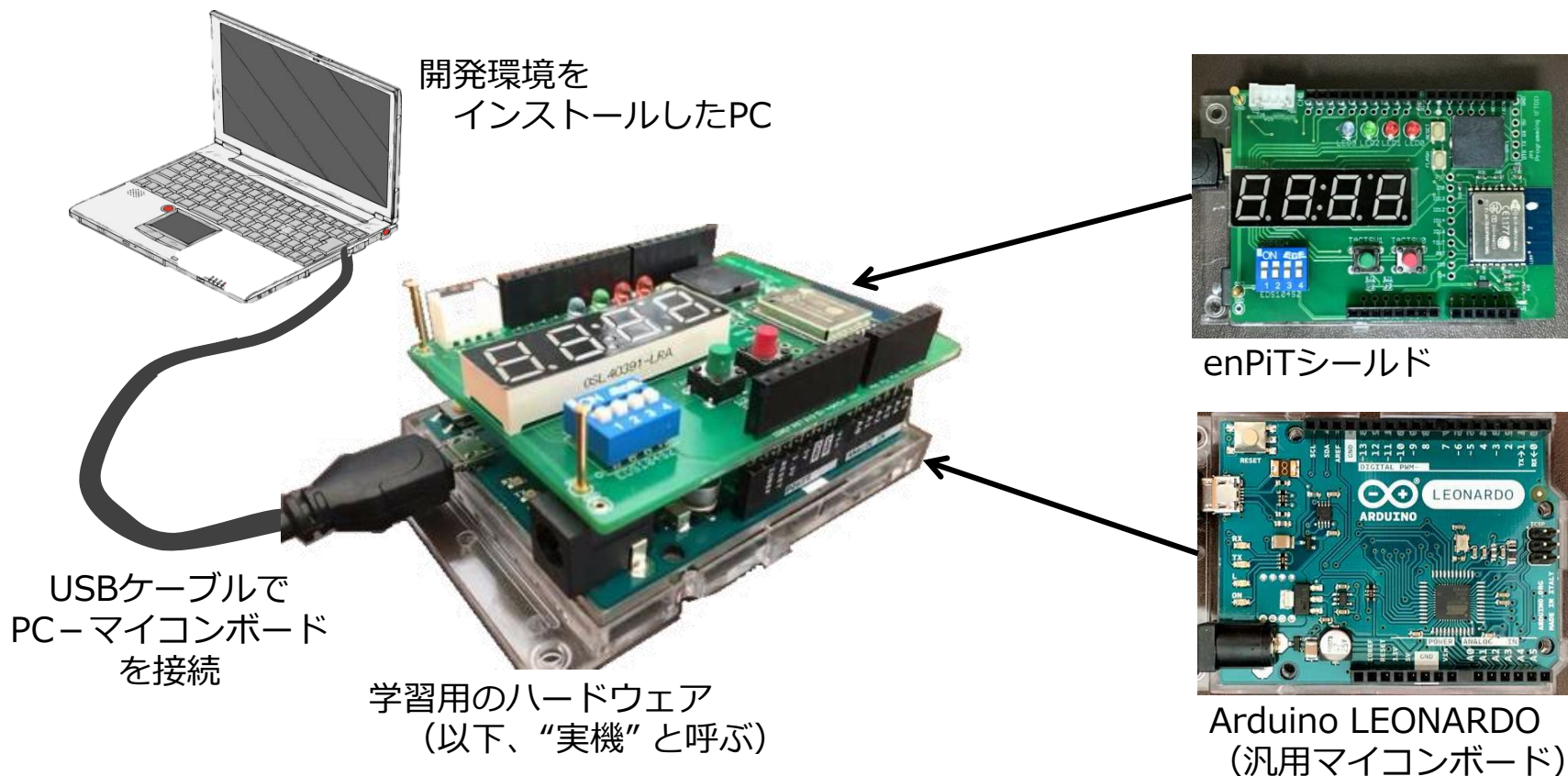
COM番号が変わっていたら、  
新しいCOMポートを選択する



# 学習用 ハードウェア

- 学習に必要なハードウェア構成とその接続方法

**注意：一旦PC/MacのUSB端子からArduinoを抜いてから  
enPiTシールドを挿してください**



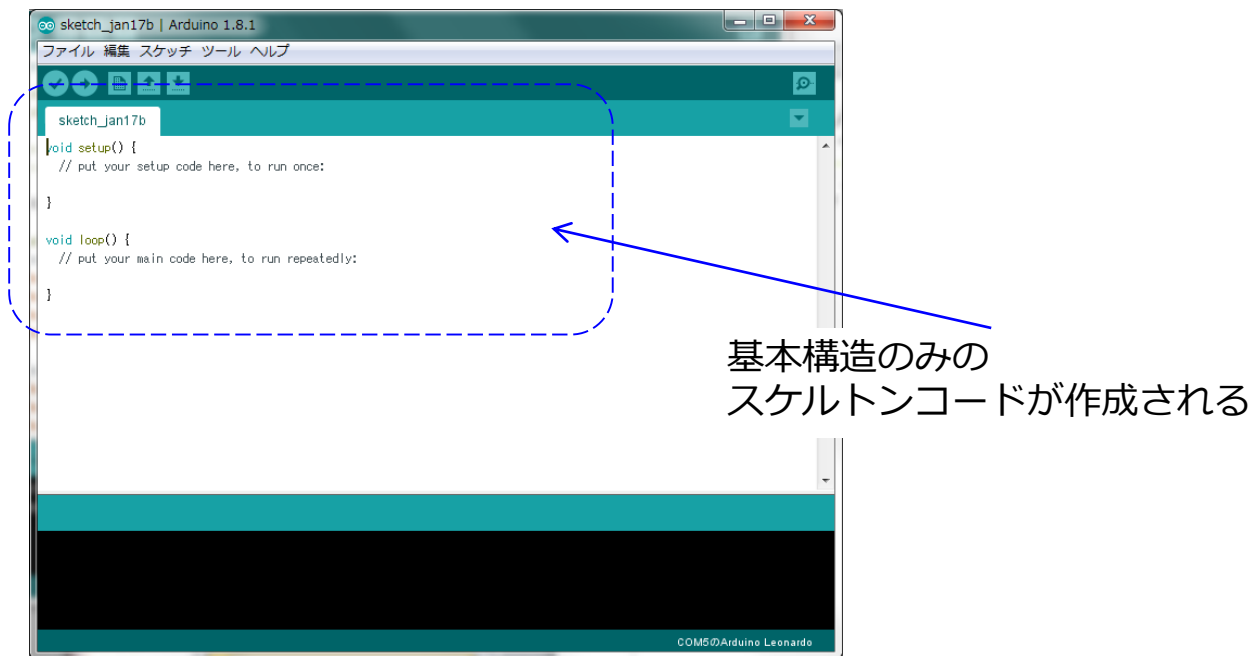


---

# 3.初めての Arduinoプログラミング

# スケッチを作ろう

- スケッチとは？
  - Arduinoにおけるプログラムのこと
  - マイコンボード書き込んで実行する、ひとまとまりのコード
- スケッチを作成する
  - 新しいスケッチを開いて、プログラムを入力する
  - スケッチを保存する (.icoファイル)



新しいスケッチを開いた 画面

# Arduinoプログラムの基本形構造

- スケッチには、必ず 関数 : setup()、loop() を書く

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  :  
*/
```

コメント

```
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}
```

初期設定関数

```
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
  delay(1000);                      // wait for a second  
  digitalWrite(LED_BUILTIN, LOW);  // turn the LED off by making the voltage LOW  
  delay(1000);                      // wait for a second  
}
```

繰り返し関数

サンプル例 Blink.ino のソースコード

# 補足：Blinkプログラムの解説

```
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);  
}
```

loop() 処理で制御する  
入出力ポートの初期設定

ここでは、LED\_BUILTIN という  
ポートを“出力”に設定している

```
void loop() {  
  digitalWrite(LED_BUILTIN, HIGH);  
  delay(1000);  
  digitalWrite(LED_BUILTIN, LOW);  
  delay(1000);  
}
```

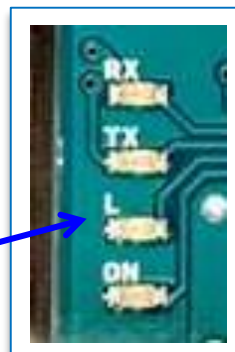
Arduinoに電源が入っている間  
繰り返し実行する処理

出力ポート LED\_BUILTIN へ  
HIGH または LOW を1秒間隔  
出力している

スケッチ例 Blink.ino のソースコード(抜粋)

**注意：**  
**enPiTシールドを挿した状態では、動作しません**

loop() に書かれた処理により  
LED：L が 1秒間隔で、点滅する



# 学習用プログラムを動かす

- 実機（Arduino Leonardo + enPiT シールド）上で動作する、プログラム [step1.ino](#) を作成してみよう
  - 新しいスケッチを開いて、下記のプログラムを入力する
  - プログラムをコンパイル&実行して、実機の動作を確認する

```
// sample program : Blink LED0, LED2
```

```
// LED lamps port number
```

```
#define LED0    5
```

```
#define LED2    10
```

```
#define LED_ON  HIGH
```

```
#define LED_OFF LOW
```

```
void setup() {
```

```
  // initialize the LED pin as outputs:
```

```
  pinMode( LED0, OUTPUT );
```

```
  pinMode( LED2, OUTPUT );
```

```
}
```

つづき

```
void loop() {
```

```
  digitalWrite( LED0, LED_ON );
```

```
  delay(500);
```

```
  digitalWrite( LED0, LED_OFF );
```

```
  delay(500);
```

```
  digitalWrite( LED2, LED_ON );
```

```
  delay(500);
```

```
  digitalWrite( LED2, LED_OFF );
```

```
  delay(500);
```

```
  return;
```

```
}
```

実機を使ったサンプルプログラム (step1.ino)



どのような動作をるすか？

# タブでソースコードを分割する

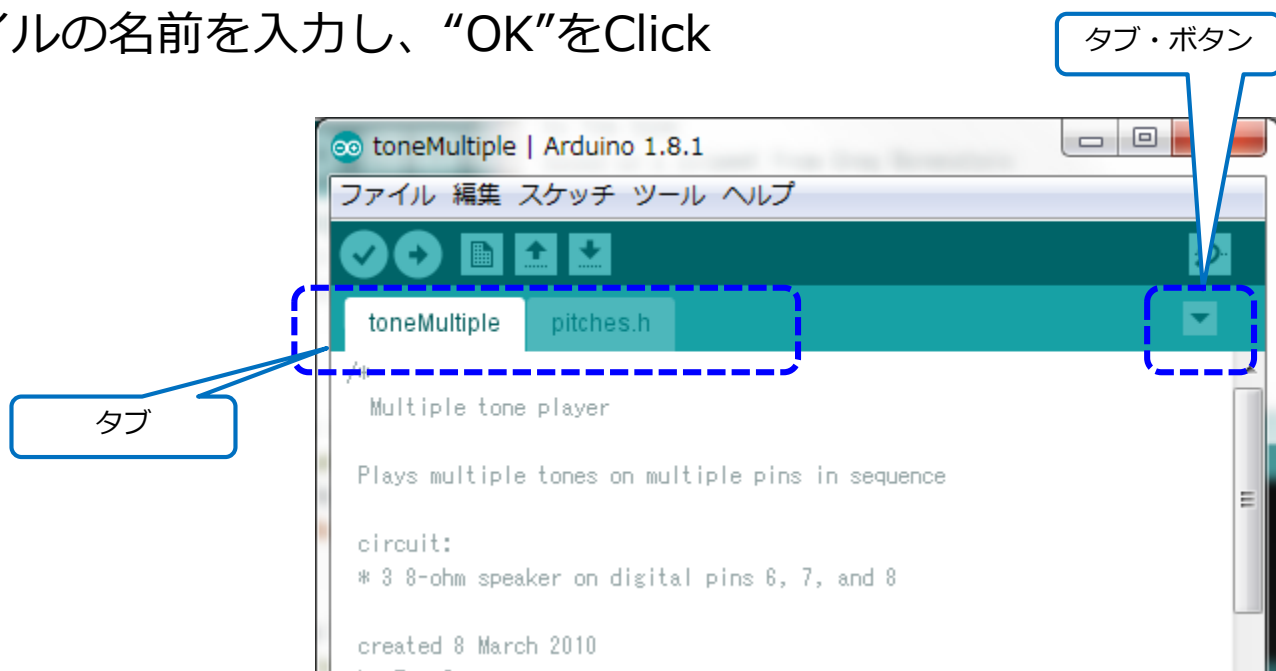
- タブとは？

ソースコードファイル、スケッチ内に複数タブを持つことができる

- タブ ... スケッチ内にあるタブを表す
- タブ・ボタン ... タブ操作に関するメニューボタン

- スケッチ内に別ファイルを作成

- タブ・ボタンをクリックし、“新規タブ” を選択
- 新規ファイルの名前を入力し、“OK”をクリック



# タブを追加してみる

- 入力した step1.ino を、タブを使って2つのファイルに分ける
  - 新規タブ enpitshield.h を追加する
  - 元のソースコードの、ピンアサイン定義は削除
  - #include で追加したファイルを参照する

```
// Pin assignment of enPit2 shield
```

```
// LED lamps
```

```
#define LED0    5
```

```
#define LED1    6
```

```
#define LED2   10
```

```
#define LED3   11
```

```
#define LED_ON  HIGH
```

```
#define LED_OFF LOW
```

**enpitshield.h**

```
// sample program : Blink LED0, LED2  
#include "enpitshield.h"
```

```
void setup() {
```

```
    // initialize the LED pin as an output:
```

```
    pinMode( LED0, OUTPUT );
```

```
    pinMode( LED2, OUTPUT );
```

```
}
```

```
    :  
    :
```

step1.ino (抜粋)

# 演習1：

サンプルコード step1.ino をベースに、以下の動作になるようにプログラムを変更する

- 1) LED0 ～ LED3を順番に点滅させるプログラムを作る  
LED0:点灯 → LED0:消灯 → LED1:点灯 → … → LED3:消灯 (繰り返し)  
点灯・消灯時間は、それぞれ500ms とする

## 【アドバンスド】

- 2) LED0 ～ LED3の点滅時間を自由に変えて、何かリズムを作りだそう  
例：三々七拍子をLEDで表現する

変更したプログラムは、  
別スケッチにして保存しましょう！





---

## 4. マイコンで、 LED点灯制御しよう

# 入出力デバイス制御

- Arduinoで使用する入出力デバイスの制御方法について

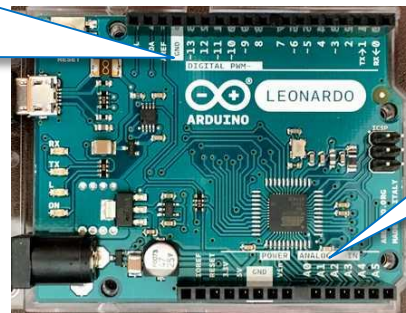
- **デジタル入出力ポート**   
HIGH/LOW の二値で扱う信号の入力 もしくは 出力  
入力 : スwitchの ON/OFF など、二値を読み込むのに使用  
出力 : LED点灯(ON)・消灯(OFF)など、二値を出力するのに使用
- **アナログ入力ポート**   
連続する入力信号を扱う



デジタル入出力ポート



一部をenPiTシールドの  
LEDに接続



Arduino マイコンボード



アナログ入力ポート

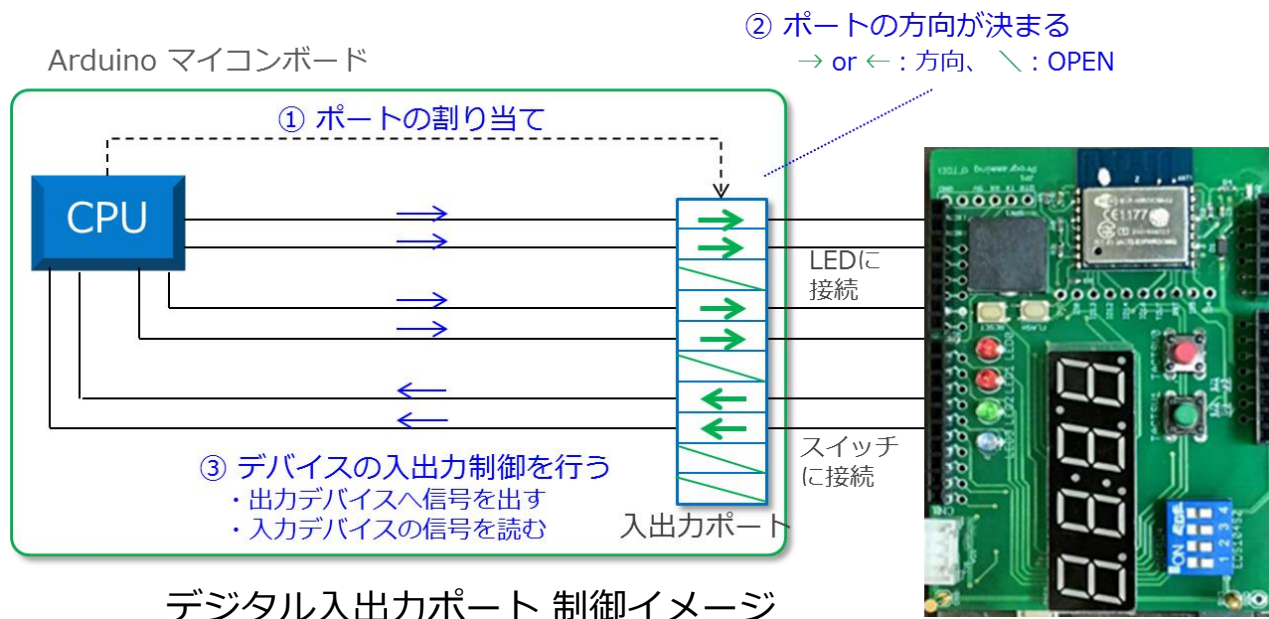
# デジタル入出力ポート

- デジタル入出力ポート

- デジタル入出力ポートは、デジタル信号の出入口
- ポート(pin)ごとに、入力/出力 の設定がえられる

- ポートの割り当て

- システム起動時に、ポートの方向(入口/出口)を宣言する
- その後、プログラムから入出力デバイスの制御が可能



# デジタル入出力ポートの制御方法

- ピンのI/O初期設定方法

- 指定したピンの入出力方向を設定する
- `pinMode( pin, mode )`
  - pin : 設定したいピンの番号
  - Mode : INPUT、OUTPUT、INPUT\_PULLUP (\*)

- デジタル出力制御

- 指定したピンに  
HIGH または LOW を出力する
- `digitalWrite( pin, value )`
  - pin : 出力するピンの番号
  - value : HIGH、 LOWWriteによる出力電圧は  
HIGH = 3.3V  
LOW = 0V (GND)

(\*) INPUT\_PULLUP :  
入力設定の一種ですが、本システムでは使用しません

```
// sample program : Blink LED0, LED2  
#include "enpitshield.h"
```

```
void setup() {  
    // initialize the LED pin as an output:  
    pinMode( LED0, OUTPUT );  
    pinMode( LED2, OUTPUT );  
}
```

```
void loop() {  
    digitalWrite( LED0, LED_ON );  
    delay(500);  
    digitalWrite( LED0, LED_OFF );  
    delay(500);  
    :  
}
```

step1.ino (抜粋)

# 演習2：

- 学習テーマ（enPiTシールド使用）のI/O定義を完成させる
  - 学習テーマのハードウェア仕様のピンアサインを参照し、デジタルI/Oポートの定義(enpitshield.h)を完成させる

ピン	機能	備考
D0,D1	ESP HW UART	
D2	N.C.	
D3	N.C	
D5	LED0	HIGHで点灯
D6	LED1	HIGHで点灯
D7	圧電スピーカ	
D8	スライドSW4	ONでHIGH OFFでLOW
D9	スライド SW3	ONでHIGH OFFでLOW
D10	LED2	HIGHで点灯
D11	LED3	HIGHで点灯
D12	スライド SW2	ONでHIGH OFFでLOW
D13	スライド SW1	ONでHIGH OFFでLOW

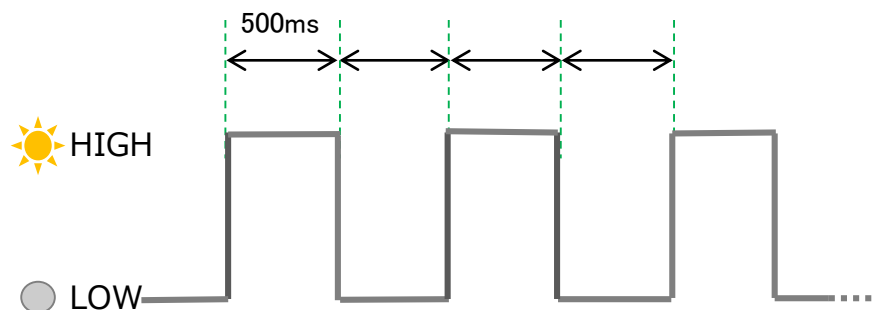
ピン	機能	備考
A0	Tact SW0	通常LOW 押したらHIGH
A1	Tact SW1	通常LOW 押したらHIGH
A2	N.C.	
A3	N.C.	
A4	N.C.	
A5	N.C.	
SCL,SDA	I2C	・ 7セグLED ・ GROVEコネクタ
POWER (7pins)		

ハードウェアのピンアサイン

コード変更後は、動作に変わりはないとも  
実機で動作確認を行いましょう

# LEDの点灯・消灯制御

- LEDを指定時間で点灯・消灯させる
  - LED 点灯・消灯 を500ms間隔で切り替える



- 点灯・消灯間隔を制御する
  - 指定時間プログラムを一時停止
  - `delay(ms)`
    - ms : 一時停止する時間  
単位 ミリ秒

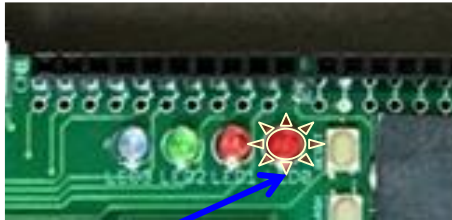
```
void loop() {  
    digitalWrite( LED0, LED_ON );  
    delay(500);  
    digitalWrite( LED0, LED_OFF );  
    delay(500);  
    :  
}
```

step1.ino (抜粋)

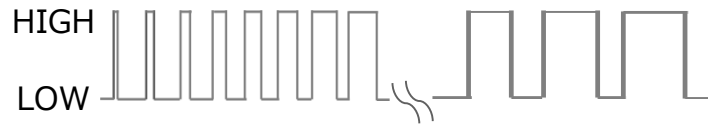
※ `delay()` のパラメータはunsigned long型です。  
32767より大きい整数を指定するときは、値の後ろにULを付け加えます。  
例 `delay(60000UL);`

# プログラムのデバッグ

- スケッチ step1.ino の loop() 内を、下記に書きかえる
  - 書き換えたら、新たなスケッチ step2.ino に保存
  - プログラムを実行して、動作を確認する



Step2.ino を実行すると  
LED0の輝度が徐々に変わって点灯する



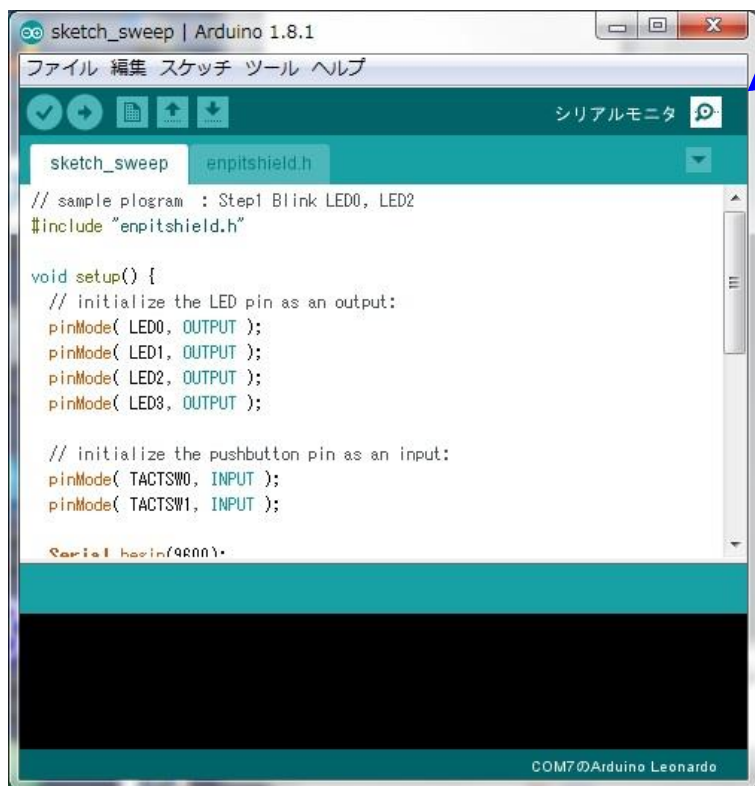
動作途中のLED点灯時間を  
知る方法は？

```
void loop() {  
  
  int light , repeat;  
  
  for ( light=0; light<=10; light++ ) {  
    for ( repeat=0; repeat<50; repeat++ ) {  
  
      digitalWrite( LED0, LED_ON );  
      delay( light );  
      digitalWrite( LED0, LED_OFF );  
      delay( 10-light );  
  
    }  
  }  
  
  return;  
}
```

step2.ino (抜粋)

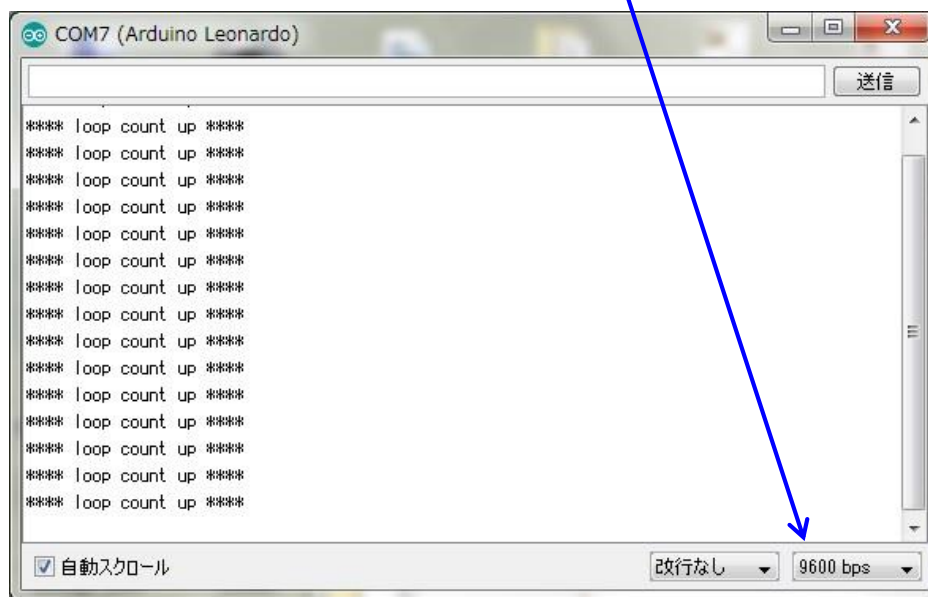
# シリアルモニタ

- PC – Arduinoボード間のUSBのシリアル通信を利用
  - Arduino → PC ヘデータ送信
  - シリアルモニタ画面に、受信データ表示



“シリアルモニタボタン” を Click

通信レートを選択



シリアルモニタ画面



# シリアルモニタのデバッグ活用

- シリアル通信転送レート設定

- シリアル通信の初期設定、データ転送レートをbps(baud)で指定
- `Serial.begin(speed)`
  - speed : データ転送レート 単位 : bps(ビット/秒)  
300, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200

- データ送信

- ASCIIテキストでデータをシリアルポートへ出力
- `Serial.print(data, format)`
  - data : 出力する値、すべての型に対応
  - format : 基数 BIN, OCT, DEC, HEX  
または 有効桁数  
(浮動小数点数を出力する場合)

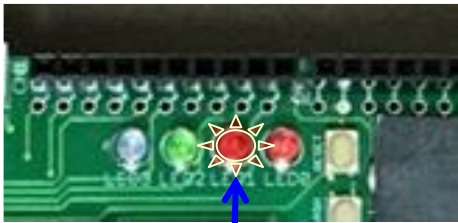
```
void setup(){
    :
    Serial.begin(9600);
}

void loop() {
    int light , repeat;
    for ( light=0; light<10; light++ ) {
        Serial.println( light, DEC );
        for ( repeat=0; repeat<50;
repeat++ ) {
            :
        }
```

step2.ino にデバッグプリント追加 (抜粋)

# アナログ出力(PWM出力)

- スケッチ step2.ino の loop() 内を、下記に書きかえる
  - 書き換えたら、新たなスケッチ step2a.inoに保存
  - プログラムを実行して、動作を確認する



Step2a.ino を実行すると  
(Step2.inoと類似の動作)  
LED1の輝度が徐々に変わって点灯する

```
void loop() {  
  int light;  
  
  for ( light=0; light<=10; light++ ) {  
    analogWrite( LED1, (255 * (light*10)) / 100 );  
    delay(250);  
  }  
  return;  
}
```

step2a.ino (抜粋)

- アナログ出力 (PWM出力) 制御
  - 指定したピンからアナログ値(PWM波)を出力する
  - `analogWrite( pin, value )`
    - pin : 出力するピンの番号
    - value : デューティ比 (0から255)  
Writeによる出力電圧は     0 = 0V    255 = 3.3V

# 補足：PWMによるアナログ出力

- PWM (Pulse Width Modulation)

- HIGHとLOWを交互に出力することにより，平均電圧を制御する

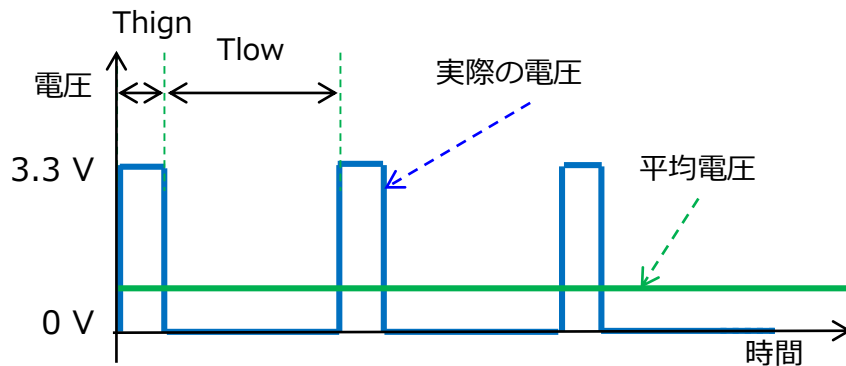
- パルス信号の周期は一定

- HIGHの時間：Thigh                      LOWの時間：Tlow

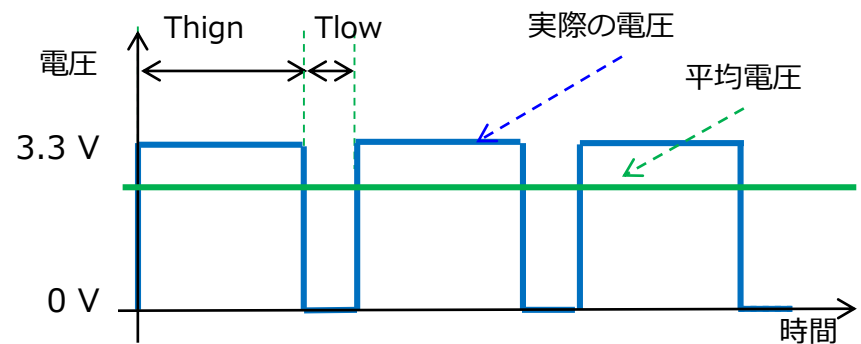
- Thighを短くする = 電圧を下げる， Thighを長くする = 電圧を上げる

- analogWrite() では，Thigh の割合（デューティ比）を指定し，アナログ出力レベルを制御する

※ PWM出力は，すべてのデジタル出力ポートで利用できる機能ではありません。  
どのポートで利用できるかは，Arduinoの仕様書で確認すること



デューティ比 20%の出力電圧



デューティ比 80%の出力電圧

# 演習3：

---

- 新たなスケッチを開いて、以下のプログラムを作成する
  - 1) LED0を1s周期の点滅(点灯・消灯時間 各500ms)に、  
LED3を250ms周期の点滅(点灯・消灯時間 各125ms)で同時に点滅させる

## 【アドバンスド】

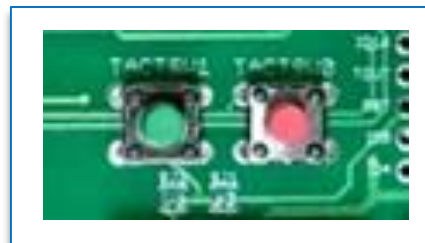
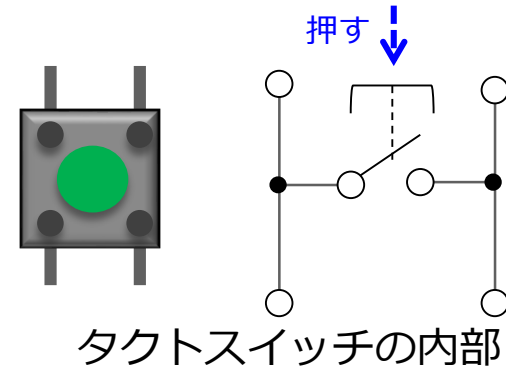
- 2) -1)のプログラムを使って、システム開始からの経過時間を10秒ごとに、  
シリアルモニタに出力する

---

## 5. タクトスイッチを制御しよう

# タクトスイッチによる入力

- タクトスイッチは、デジタル入力としてよく使われる電子部品
- タクトスイッチの仕組み
  - 上部がバネ式のボタン
  - 押すと接点が繋がり、通電する
- タクトスイッチの接続状態読み込み
  - 押した状態(ON) : HIGH
  - 通常 (OFF) : LOW



enPiTシールド のタクトスイッチ

# デジタル入力ポートの制御方法

- 入力制御

- 指定したピンの値を読み込む

- `digitalRead(pin)`

- pin : 読み込むピンの番号
- 戻り値 : HIGH または LOW

※ 入力に使用するピンは、初期設定でINPUTに設定してあること

```
// Tact switches
#define TACTSW0 A0
#define TACTSW1 A1

#define TACTSW_ON HIGH
#define TACTSW_OFF LOW

// LED lamps
#define LED0 5
#define LED1 6
#define LED2 10
#define LED3 11

#define LED_ON HIGH
#define LED_OFF LOW
```

enpitshield.h

```
void setup() {
  pinMode( TACTSW0, INPUT );
  pinMode( LED3, OUTPUT );
}

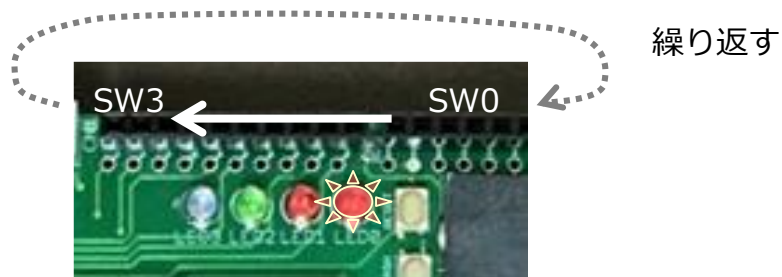
void loop() {
  if ( digitalRead( TACTSW0 ) == TACTSW_ON ) {
    digitalWrite( LED3, LED_ON );
  }
  else {
    digitalWrite( LED3, LED_OFF );
  }
  return;
}
```

TACTSW0=ONの間 LED3点灯(step3.ino)

# 演習4：

- プログラム step3.ino をベースに、スイッチ入力制御プログラムを作成する

-1) TACTSW0 を押すごとに、LED0～3 が順番に点灯する



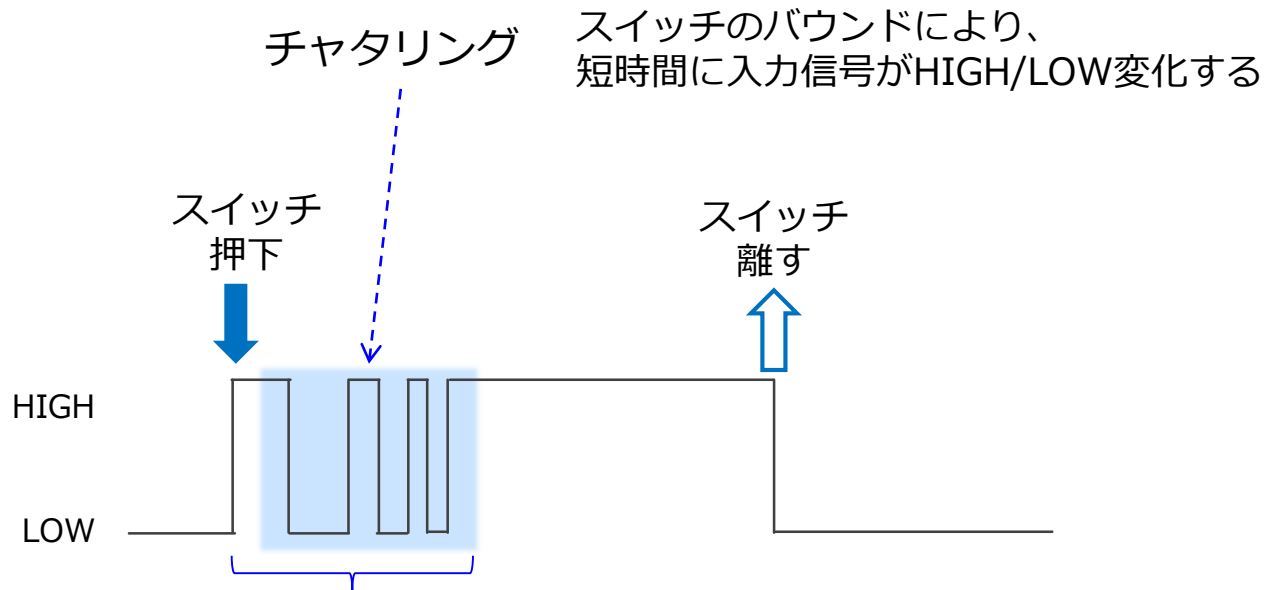
補足

- 点灯するLEDは、常時1つだけ
- SW0を一回押したら、LEDは一つだけ進む
- SWを押し続けても、次には進まない



# タクトスイッチのチャタリング

- スイッチを押したとき、「接点同士の衝突」により物理的なバウンドが発生する



## チャタリング除去

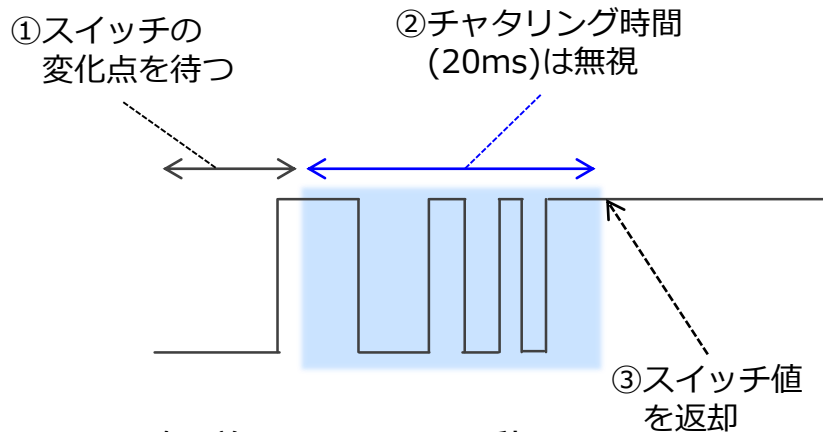
ソフトウェアでスイッチ読み取り時、  
押下検出から、チャタリングを想定時間は無視する

# チャタリング除去方法

## • チャタリング除去の例

### ➤ 仕様：

- TACTSW0 を押すごとに、LED3 を点灯/消灯する
- チャタリング時間は、最大20msと想定



解説：chtsw()の動き

```
void loop() {  
    static boolean sw = false;  
    if ( chtsw( TACTSW0 ) == TACTSW_ON ) {  
        if ( sw ) {  
            digitalWrite( LED3, LED_ON );  
        } else {  
            digitalWrite( LED3, LED_OFF );  
        }  
        sw = !sw;  
    }  
    return;  
}
```

```
boolean chtsw( byte dx ) {  
    boolean tsw = digitalRead( dx );  
    while ( tsw == digitalRead( dx ) ); } ①  
    delay(20); ②  
    return !tsw; ③  
}
```

チャタリング除去サンプルコード  
(step4.ino)

# 演習5：

---

- チャタリングを考慮した、スイッチ入力制御プログラムを作る

-1) 演習4 で作成したプログラムに、チャタリング対策を追加する

## 【アドバンスド】

- 2) step4.ino で、スイッチ押下待ちの間に、他のLEDを点滅させるにはどうしたらよいかを検討する

---

## 6. タイマを使って、 時間をプログラミングしよう

# マイコンプログラムの時間制御

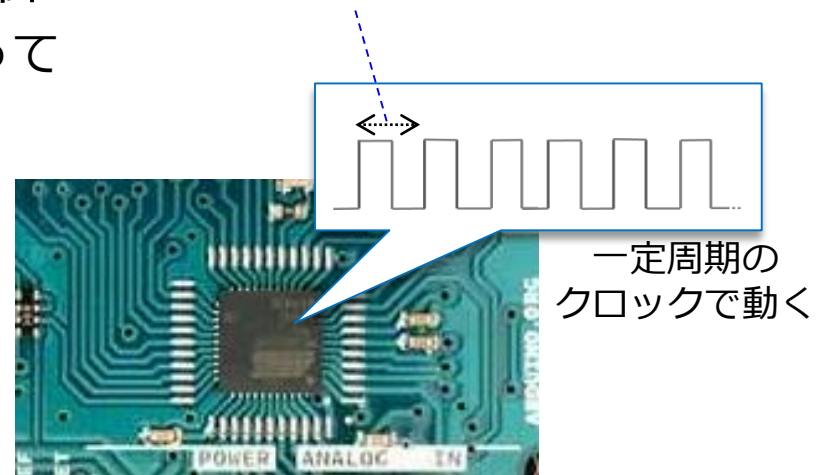
- ソフトウェアのみの時間制御
  - ループによる時間の調整
    - CPUのクロック周期とコードから時間を割り出す
    - 待ち時間の間他の処理は出来ない

```
long count;  
digitalWrite( LED3, LED_ON );  
for ( count=0; count <65535; count++ ) {  
    ;  
}  
digitalWrite( LED3, LED_OFF );
```

ループによる時間制御コードの例

- マイコンの内蔵機能を使った時間制御
  - マイコンが、クロック信号を使って時間をカウントする方法を提供

時間制御の基準となる



マイコン

# delay()関数による時間制御

- 指定時間、プログラムを待機させる

Arduino標準関数 : delay()、delayMicroseconds()

- 関数実行中は、CPU制御を握っている
  - ソフトウェアタイマと同様
- ただし、関数実行中も割り込みは有効
  - 割り込み発生時には、指定時間より長く待機することもある

関数名	説明	戻り値
void delay(unsigned long ms)	パラメータで指定した時間(ミリ秒単位)だけプログラムを一時停止する。1秒は1000ミリ秒である。	なし
void delayMicroseconds(unsigned int us)	パラメータで指定した時間(マイクロ秒単位)だけプログラムを一時停止する。1秒は1,000,000マイクロ秒である。 現状, 正確に発生できる遅延の最大値は16383である。	なし

Arduino 日本語リファレンスより

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

# millis()関数による時間制御

- Arduinoのプログラムをスタートした時からの時間取得  
Arduinoライブラリ関数 : millis()、 micros()
  - プログラムスタートから、関数実行時の時間(整数値)を返す
  - 時間待機するためには、アプリケーションが時間経過を計る

関数名	説明	戻り値
unsigned long millis()	Arduinoボードがプログラムの実行を開始した時から現在までの時間をミリ秒単位で返します。 約50日間でオーバーフローし、ゼロに戻ります。	プログラムが開始してからのミリ秒。
unsigned long micros()	Arduinoボードがプログラムの実行を開始した時から現在までの時間をマイクロ秒単位で返します。 約70分間でオーバーフローし、ゼロに戻ります。	プログラムが開始してからのマイクロ秒。

Arduino 日本語リファレンスより  
<http://www.musashinodenpa.com/arduino/ref/>

# タイマ割込みによる時間制御

---

- 割込み処理とは
  - 通常のプログラムの実行を強制的に中断し、別のプログラム処理を割り込ませる
  - 割り込み処理の終了後は、また元のプログラムを続行する
- Arduinoのタイマ割込み処理
  - 内蔵タイマー(TIMER2)を使用
  - loop関数を処理中にTIMER2割込みが入ると、loopの処理を一時中断し、指定した割込み処理を実行
  - 割込み処理終了すると、再びloop( )を再開

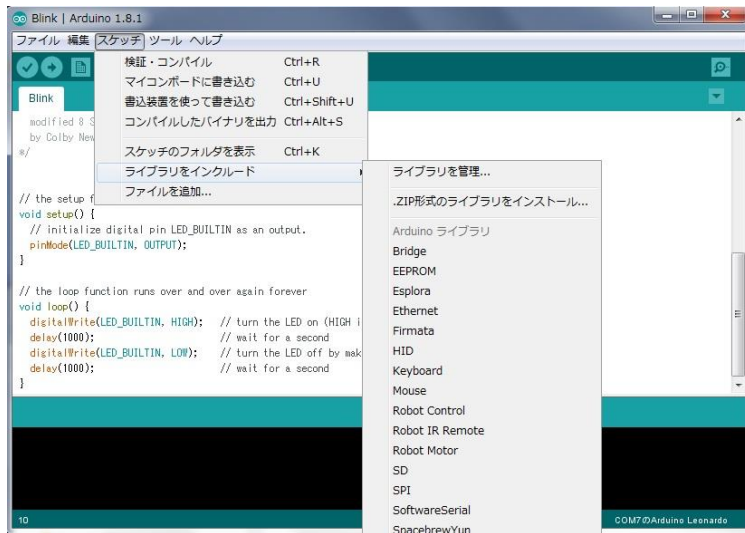
→ TIMER2 を使ったArduinoライブラリ  
MsTimer2 をインストール



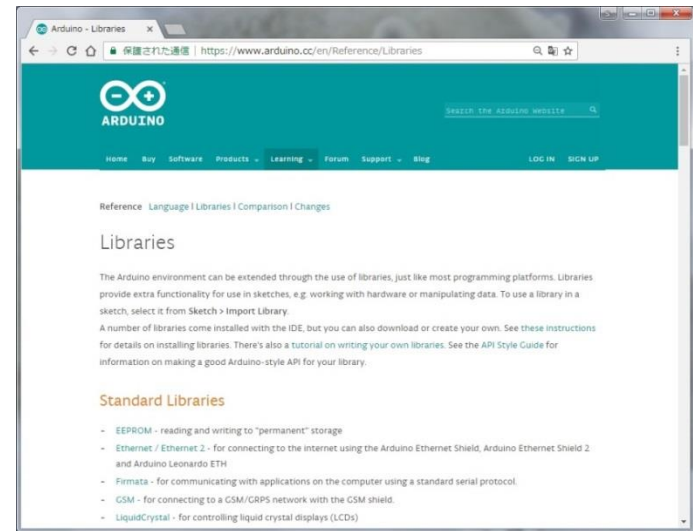
# 補足：Arduinoライブラリ

- Arduinoのライブラリ2種類

- あらかじめIDEに付属する標準のライブラリ
  - Arduino ソフトウェアのスケッチメニューより使用できる  
EEPROMへの書き込み, Ethernet通信 等
- ユーザ提供ライブラリ(Contributed Libraries)
  - Arduino公式サイトより、ダウンロードして使用する



標準のライブラリの使用画面

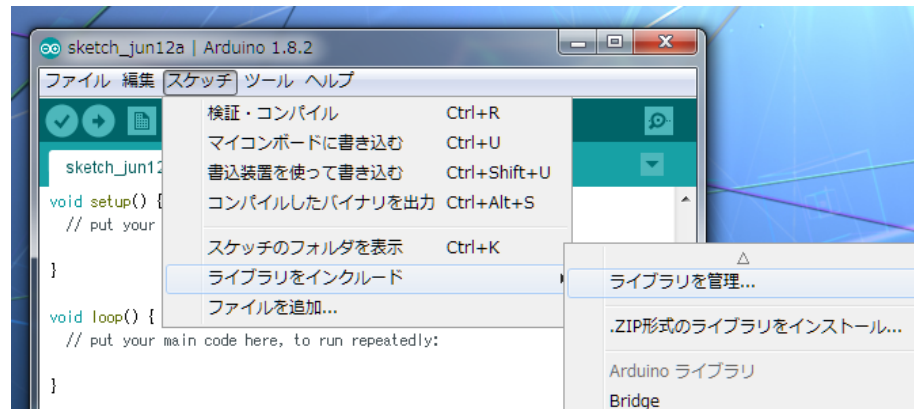


Arduino公式サイト ライブラリー一覧

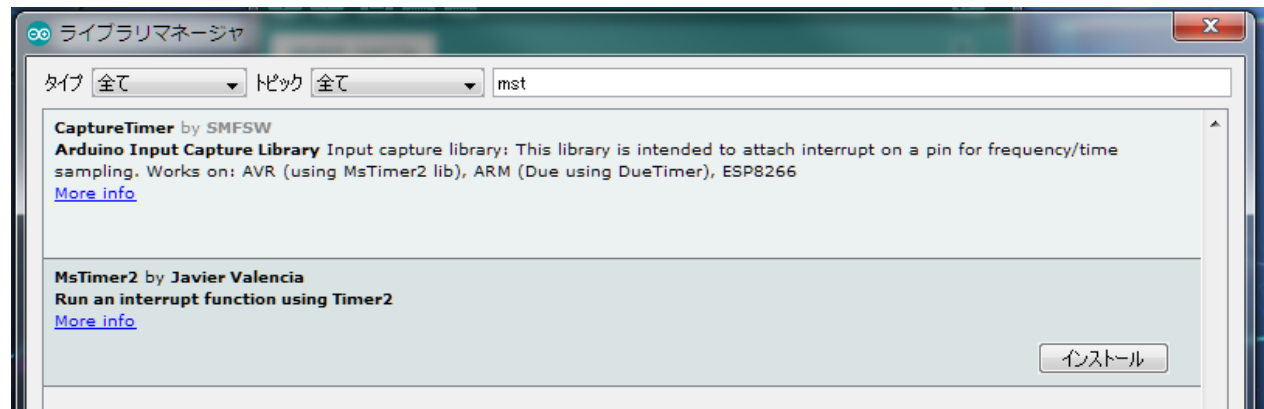
<https://www.arduino.cc/en/Reference/Libraries>

# タイマーライブラリ MsTimer2

- Arduino ライブラリをインストール
  - スケッチ→ライブラリをインクルード→ライブラリを管理



- Mstで検索するとMsTimer2が見つかる. クリックするとインストールボタンが現れるので, それをクリック



# MsTimer2の使い方

---

- MsTimer2 : ミリ秒単位でタイマ割込み処理を実行する
  - タイマ割込みの設定
    - `MsTimer2::set(unsigned long ms, void (*f)())`
      - ms : タイマ割込みを発生させる時間間隔を指定 (ミリ秒単位)
      - f : タイマ割込みで実行する関数  
関数 f() は、引数なしのvoid型
  - タイマ割込みを有効にする
    - `MsTimer2::start()`
      - 本メソッド実行後から、指定時間経過ごとに割込み処理 t() が呼び出される
  - タイマ割込みを無効にする
    - `MsTimer2::stop()`
    - 本メソッド実行後は、割込みは発生しなくなる

# タイマ割込みサンプル

- 例：500msごとに、LED0を 点灯/消灯 切り替える

MsTimer2ライブラリの  
ヘッダファイルを #include する

```
#include <MsTimer2.h>
#include "enpitshield.h"
```

割込み時実行する処理

```
void flash() {
    static boolean output = LED_OFF;

    digitalWrite(LED0, output);
    output = (output == LED_OFF ? LED_ON : LED_OFF);
}
```

set() メソッドで、  
500msごとに、flash() を  
実行するよう登録

```
void setup() {
    pinMode(LED0, OUTPUT);
```

start() メソッドで、  
割込みを有効にする

```
    MsTimer2::set(500, flash); // 500msごとにオンオフ
    MsTimer2::start();
}
```

```
void loop() {
    // loopは空
}
```

タイマ割込みサンプルプログラム  
step6.ino

# 演習6：

---

- ・ 演習3で作成した以下のプログラムを、タイマ割込みで実現する
  - 1) MsTimer2を使って, LED0を1s周期で点滅(点灯・消灯時間 各500ms) されましょう
  - 2)LED3を500ms周期の点滅(点灯・消灯時間 各250ms)で点滅させましょう
  - 3)上記 1 と 2 を同時に動作させてください

## 【アドバンスド】

- 4) -1)のプログラムの点滅表示を、スイッチ SW0 を押している間だけ 点滅するように処理を追加しましょう

# 参考文献

---

- みんなのArduino入門
  - 高本 孝頼 著 リックテレコム社
- Arduino公式サイト <https://www.arduino.cc/>
  - Getting Started <https://www.arduino.cc/en/Guide/HomePage>
  - Language Reference <https://www.arduino.cc/en/Reference/HomePage>
  - Leonardo Reference  
<https://www.arduino.cc/en/Main/ArduinoBoardLeonardo>
- スイッチサイエンス社
  - LEONARDOへのガイド  
<http://trac.switch-science.com/wiki/Guide/ArduinoLeonardo>
- Garretlab ホームページ
  - Arduinoガイド [http://garretlab.web.fc2.com/arduino\\_guide/index.html](http://garretlab.web.fc2.com/arduino_guide/index.html)
  - Arduinoリファレンス  
[http://garretlab.web.fc2.com/arduino\\_reference/index.html](http://garretlab.web.fc2.com/arduino_reference/index.html)