ＡＲＤＵＩＮＯワークショップ

センシングネット社　小林　靖

２０１９年７月１８日

小林プロフィール

- 旭化成エレクトロニクス株式会社にて、半導体回路設計に従事
- 主な設計分野
  アナログーオーディオ、通信、センサー、パワーマネジメント
  デジタルーインターフェース、信号処理
- 趣味での電気いじり
  オーディオアンプ、パソコン製作

ワークショップ概要

- 本日のワークショップは、チーム毎、個人毎に各自のパソコンで、Ａｒｄｕｉｎｏキットを使って、電子回路製作を実践する。
  - Ｃ言語でのプログラミング作成
  - センサーやＬＥＤを使ったＡｒｄｕｉｎｏハードウエア製作
  - 合わせて、センサーインターフェース（ＳＰＩ、Ｉ２Ｃ、ＵＡＲＴ）の理解

インターフェース毎に、上記内容を繰り返してワークショップを行う予定。
Ａｒｄｕｉｎｏを初めて使う方にも、分かり易く進める予定。

Ａｒｄｕｉｎｏワークショップの目的

- センサーからのセンシングデータをＰＣなどに送るシステムを開発する目的に、Ａｒｄｕｉｎｏボード（及びその互換ボード）を用いる事を想定。
- この目的の為のマイコンボードは、Ａｒｄｕｉｎｏ以外にも多くあるが、Ａｒｄｕｉｎｏは、比較的容易に、また、互換ボードを使えば、低コストで、システム開発可能。
- また、Ａｒｄｕｉｎｏを使った電子工作例は、ＷＥＢ上に大変多く、自分の開発したいものに、近い例が、容易に検索可能。

（ＰＣへのデータ転送は、ワークショップではＵＳＢによる有線
互換ボードには、ＷｉＦｉやＢＬＥなど無線でも可能。）

ＡＲＤＵＩＮＯとは

- ウィキペディアを参照

https://ja.wikipedia.org/wiki/Arduino

本日のファイルのダウンロード先やＵＲＬ

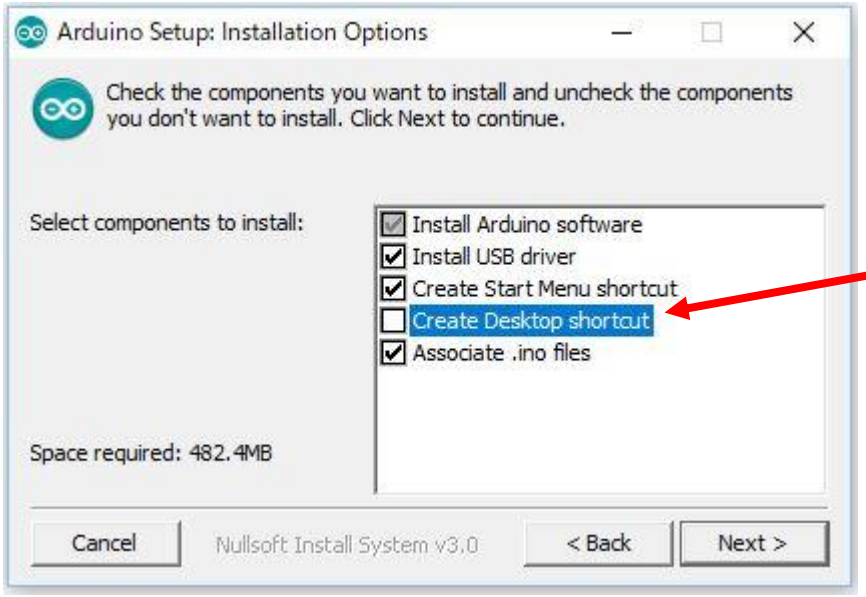
http://supersensingforum.com/meeting/190718.html

１．Ａｒｄｕｉｎｏインストール

（１）ネットワークに接続　→　SSID　P-Comp_Wifi

（２）Arduinoのインストール　→　https://www.arduino.cc/

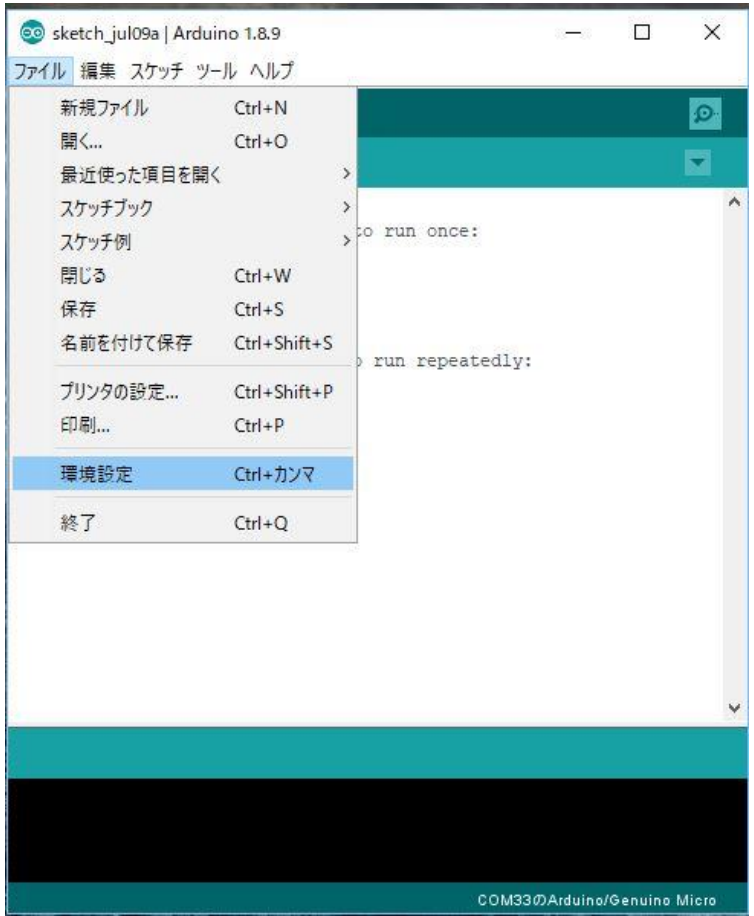
　　アクセスして、ダウンロード後、インストール

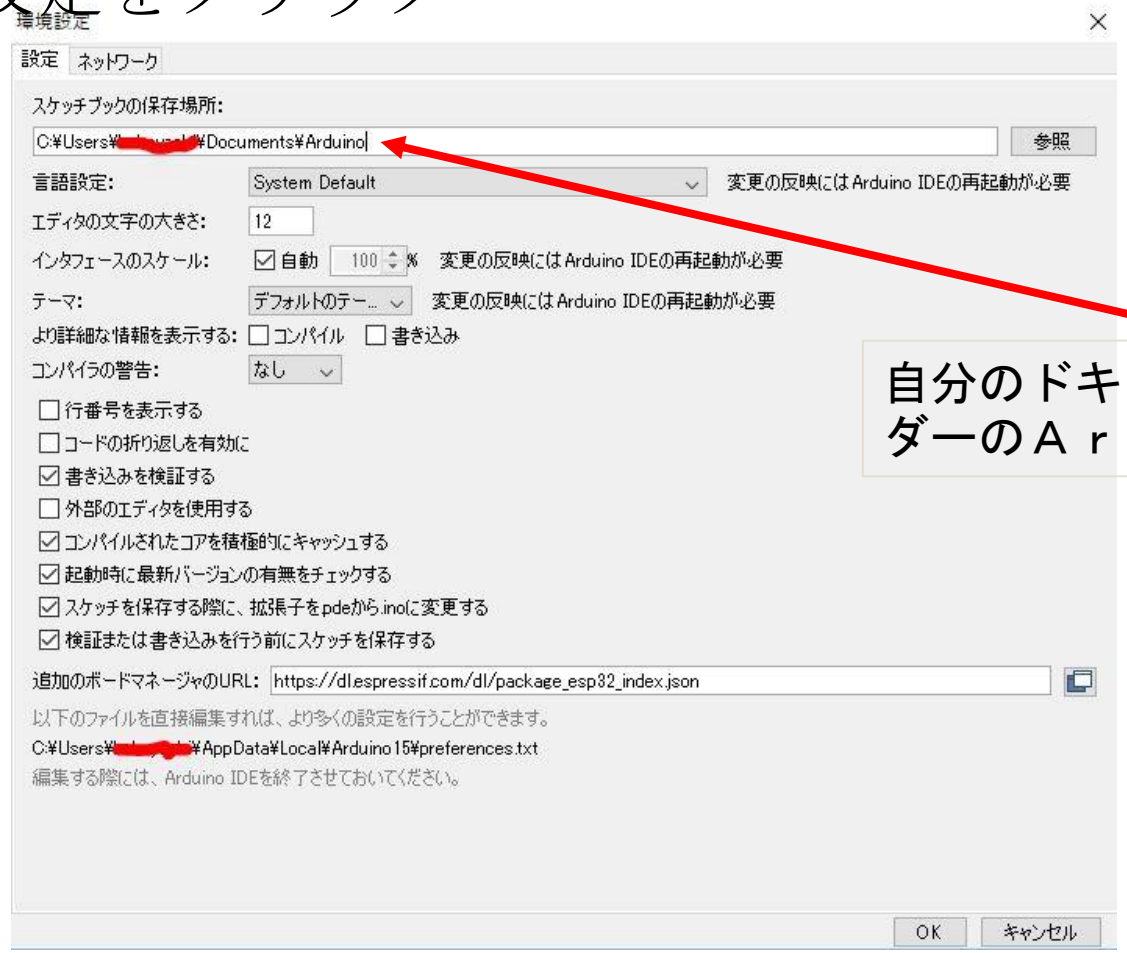


Desktopにアイコンを作りたくない場合

２．Ａｒｄｕｉｎｏの環境設定

・ファイルメニューの環境設定をクリック

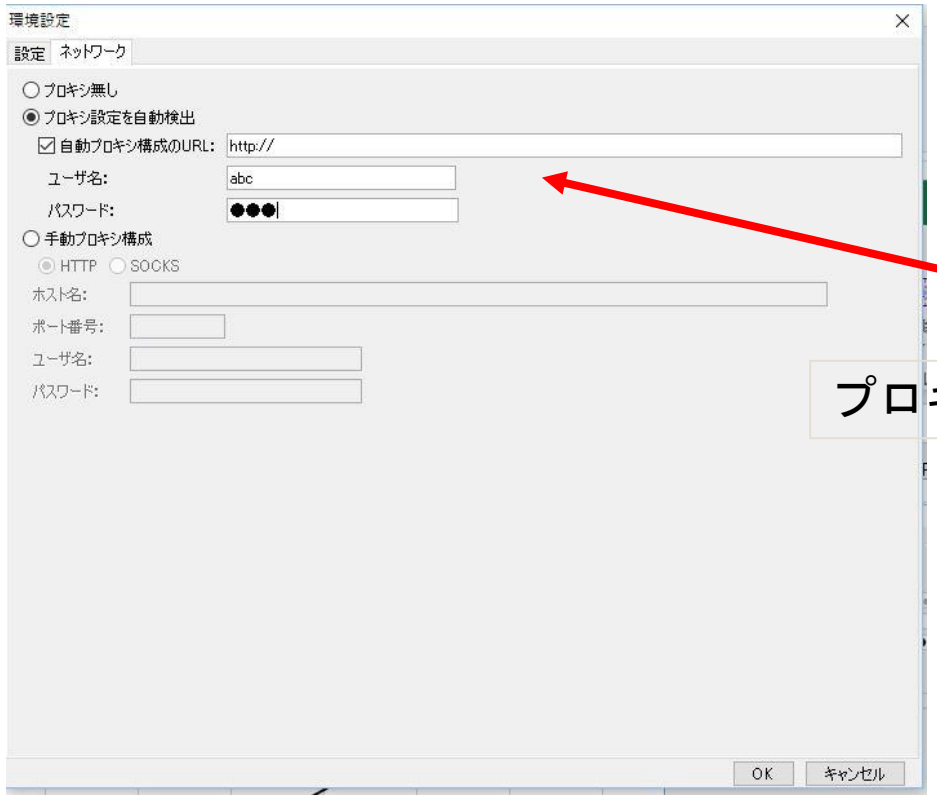




自分のドキュメントフォルダーのＡｒｄｕｉｎｏ

２．Ａｒｄｕｉｎｏの環境設定（参考）

- 会社でプロキシ経由でネットワークを使われている場合
  （但し、本日のワークショップでは、設定の必要なし）



プロキシのＵＲＬ、ユーザー名、パスワードを設定

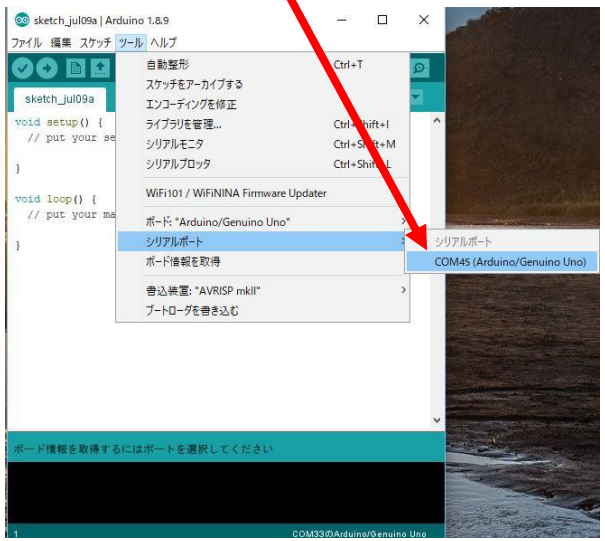
３．Ａｒｄｕｉｎｏボードの接続

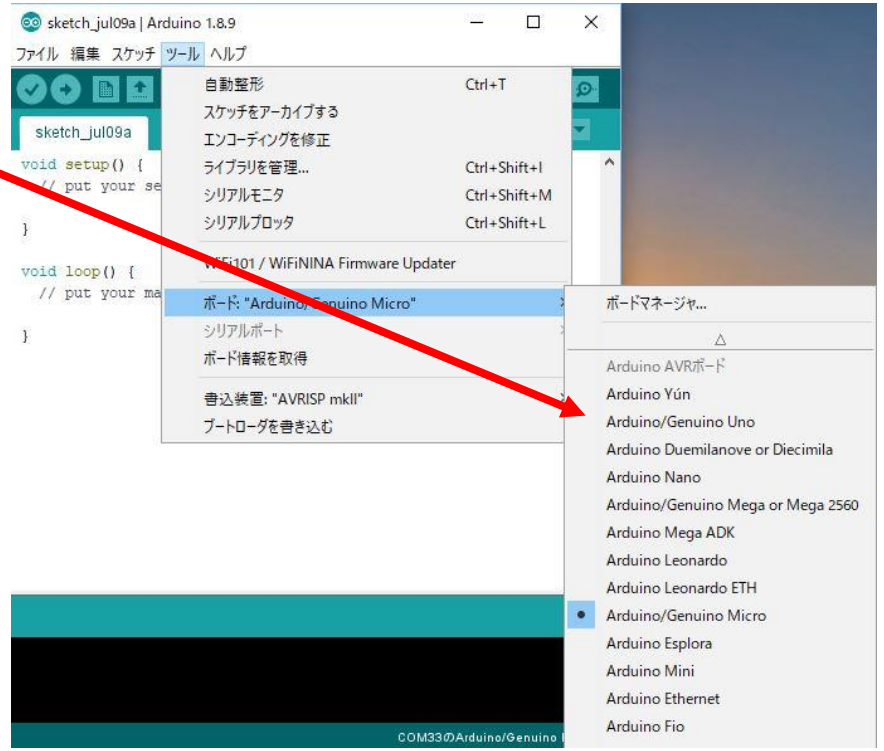
（１）ＵＳＢケーブルで、ＡｒｄｕｉｎｏボードとＰＣを接続

（２）ツールメニューでボードを選択

Arduino/Genuino Unoを選択

（３）シリアルポートを選択





４．プログラミングスタート

- 次のＵＲＬにアクセス　（Arduino　日本語リファレンス）
  http://www.musashinodenpa.com/arduino/ref/
  Arduino言語を参照

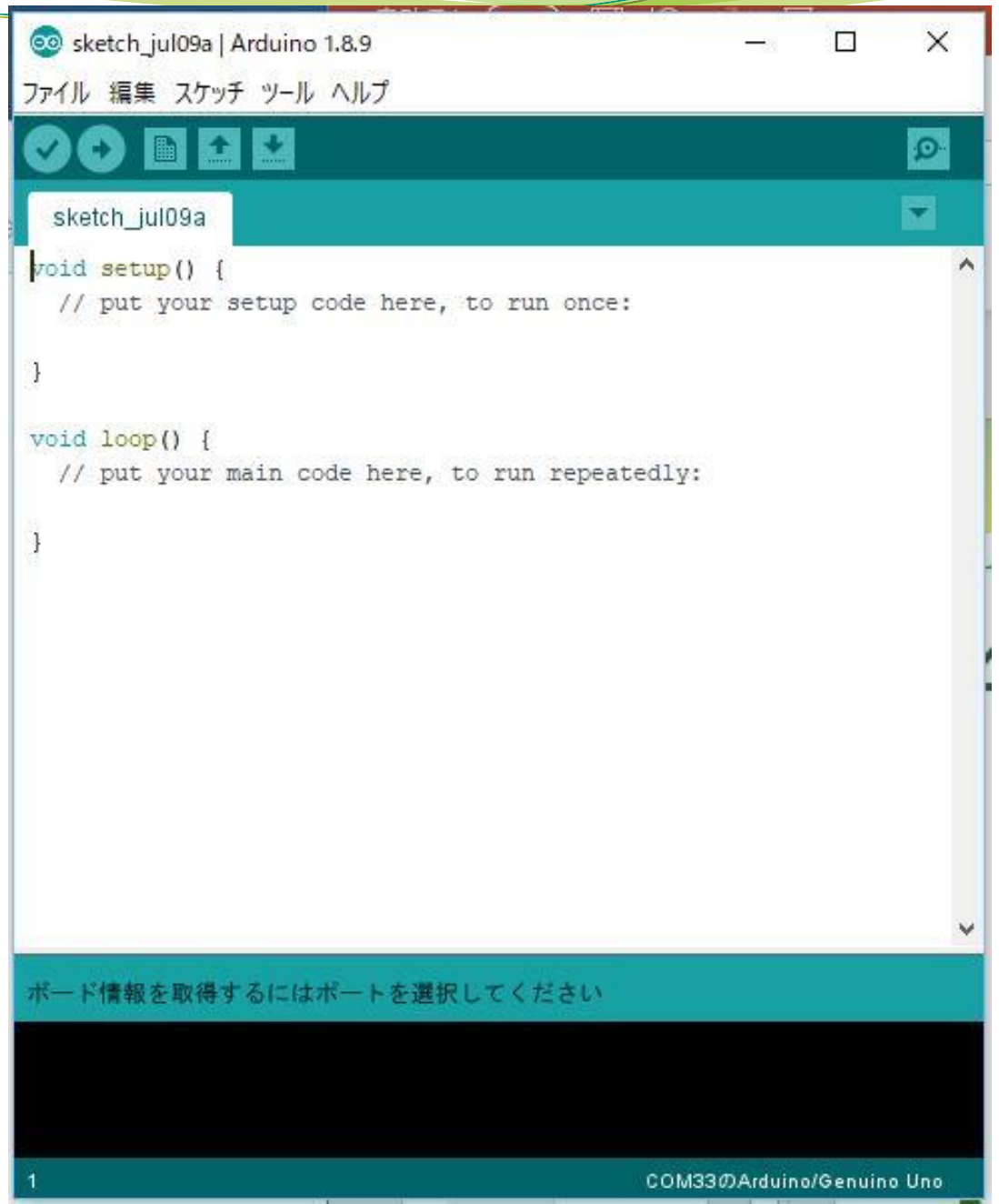
＜Ａｒｄｕｉｎｏ言語＞

ｓｅｔｕｐ（）

Ｌｏｏｐ（）

を参照

ＡＲＤＵＩＮＯの起動


sketch_jul09a | Arduino 1.8.9
ファイル 編集 スケッチ ツール ヘルプ
sketch_jul09a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
ボード情報を取得するにはボートを選択してください
1
COM33のArduino/Genuino Uno

基本的なプログラミング言語内容

- 基本的な文法を参照。→　；、｛｝、コメント
- データ型　→　byte、int、unsigned int、float
- 算術演算子　→　四則演算
- 制御文　→　if、if else、for、while

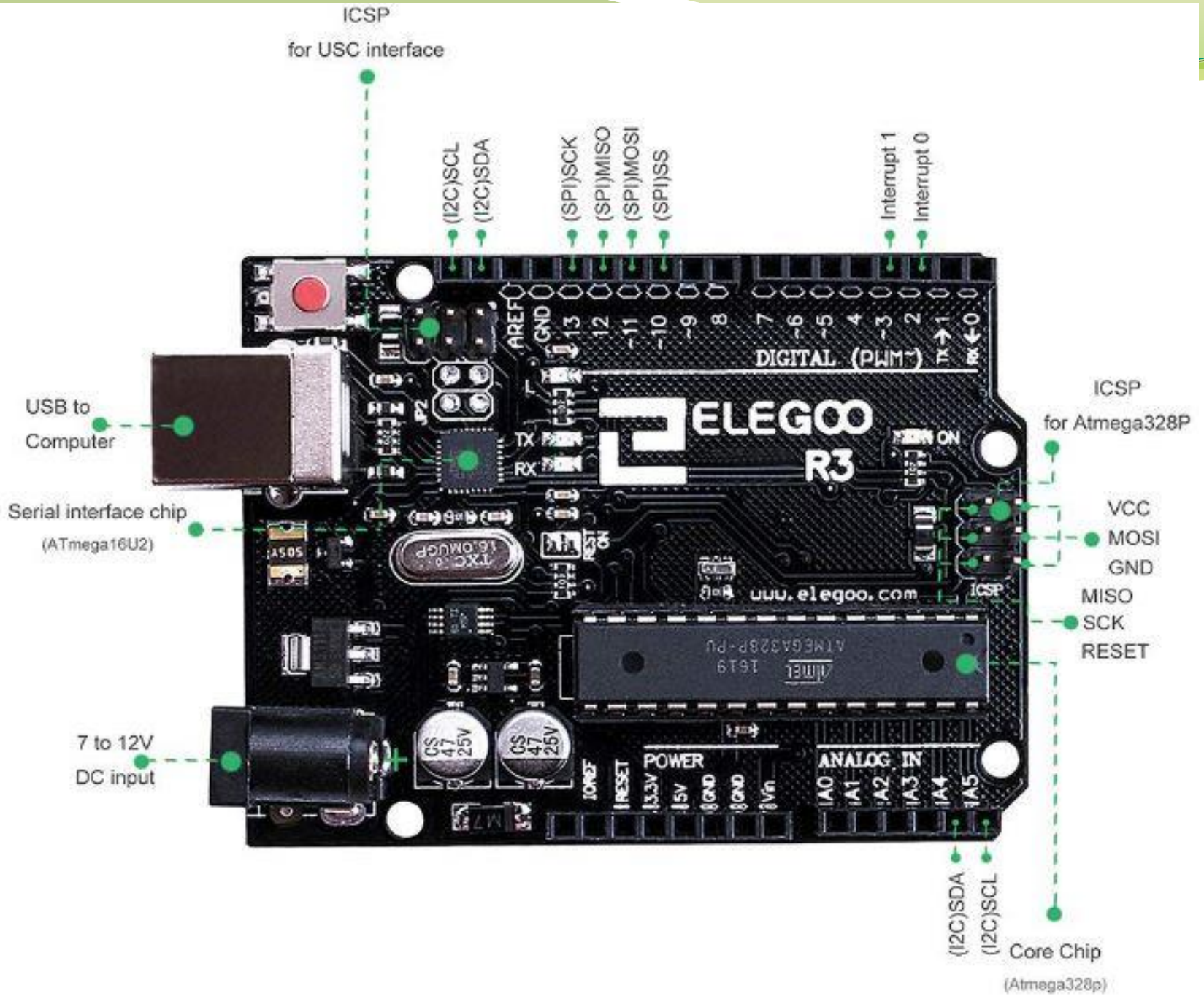
５．LEDを光らせよう１

- デジタル入出力関数　pinModeとdigitalWriteを参照
- 参考として、次のＵＲＬにアクセス（Arduinoチュートリアル基礎編）
  http://www.musashinodenpa.com/arduino/ref/index.php?f=2

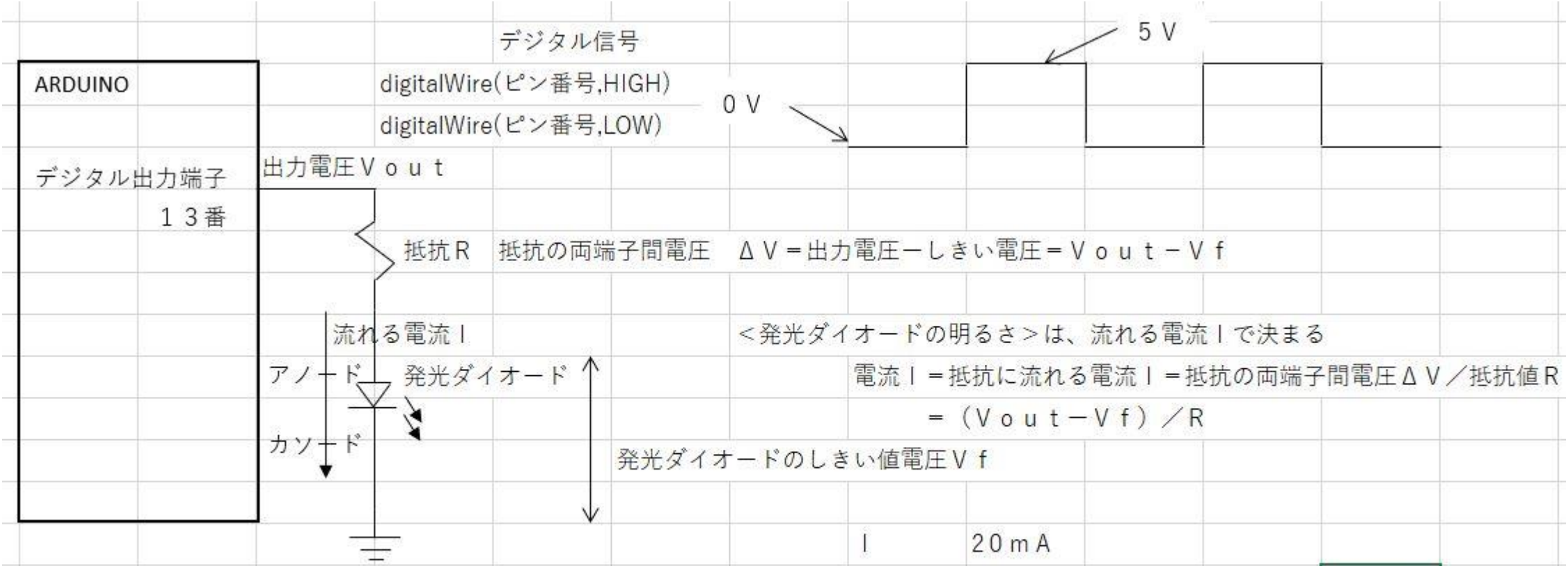
＜Ａｒｄｕｉｎｏボードの信号入出力について＞

Ａｒｄｕｉｎｏボードを手に取って見てみる

- デジタル信号の入出力　→　０番から１３番
- アナログ入力　→　Ａ０番からＡ５番（デジタル入出力としても可）
- アナログ出力（ＰＷＭ信号）　→　デジタル信号の～の付いたピン


ICSP
for USC interface
(I2C)SCL
(I2C)SDA
(SPI)SCK
(SPI)MISO
(SPI)MOSI
(SPI)SS
Interrupt 1
Interrupt 0
USB to
Computer
Serial interface chip
(ATmega16U2)
7 to 12V
DC input
ELEGOO
R3
DIGITAL (PWM~)
POWER
ANALOG IN
www.elegoo.com
ICSP
for Atmega328P
VCC
MOSI
GND
MISO
SCK
RESET
(I2C)SDA
(I2C)SCL
Core Chip
(Atmega328p)

発光ダイオード駆動回路



フォーラムのＷｏｒｋＳｈｏｐのページからダウンロード
発光ダイオードのデータシート
（内容確認）

回路の設計手順

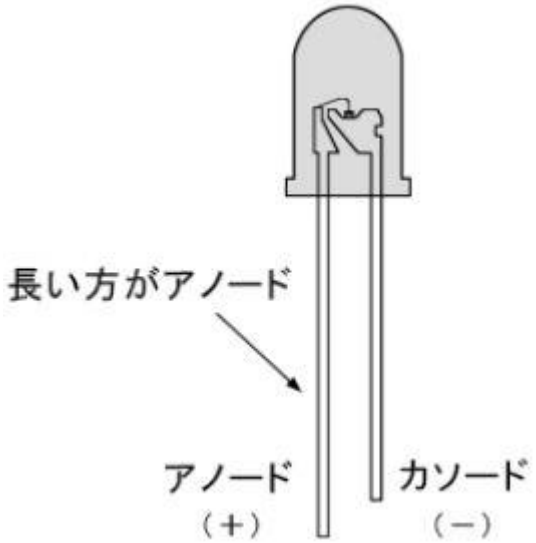
＜目的　ダイオードを光らせる＞
（１）仕様を決める　→　色、明るさ、方向、点滅間隔
（２）回路概要を決める　→　ダイオード、抵抗
（３）使用する部品をデータシートを見て決める　→　ダイオード
（４）部品の定数を算出する　→　抵抗値
（５）詳細な回路を作成する

場合によっては、（１）から（５）の間でループを繰り返す
データシートは、部品の選定時のネット注文会社（DIGIKEY、MOUSER、ＲＳコンポーネンツ、秋月電子など）の部品ページある

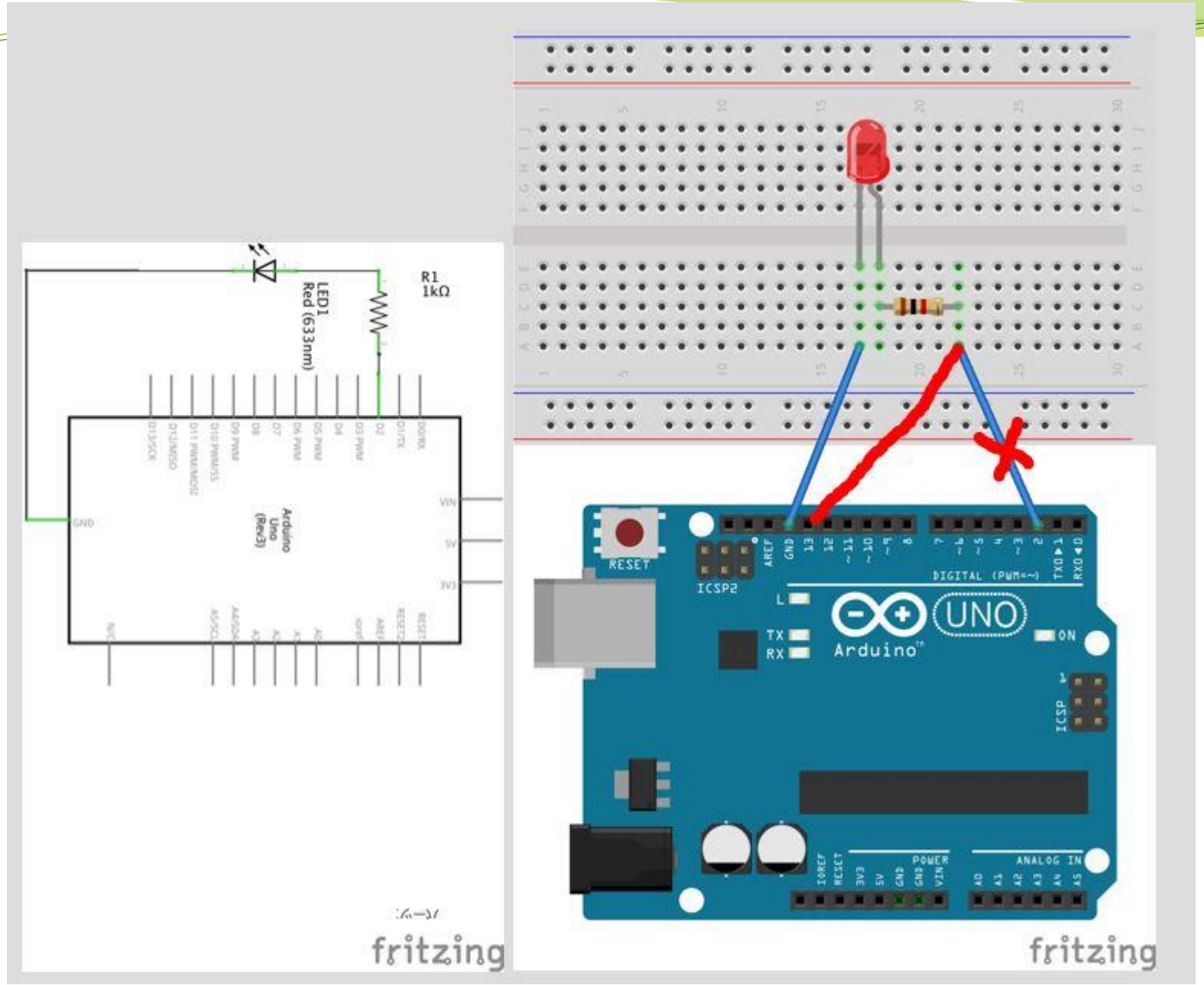
回路が決まったら、Ａｒｄｕｉｎｏでプログラミング

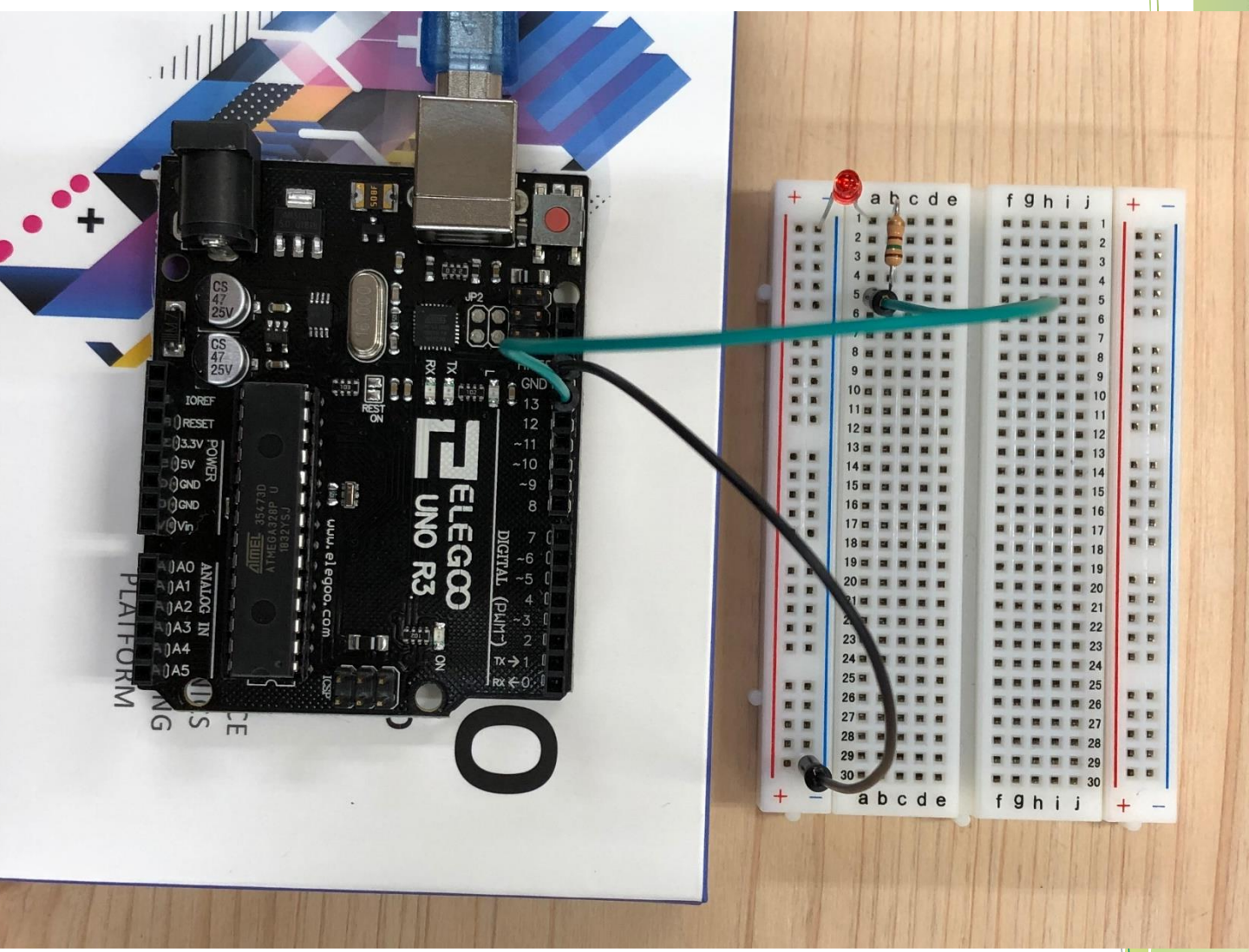
ブレットボードの配線

- 必ず、ＵＳＢ接続を外してから配線作業を行う事。



極性に注意
逆に接続すると定格の５Vになる。
（USBからの給電電圧に依存）
場合によっては、故障。


LED1
Red (633nm)
R1
1kΩ
Arduino
Uno
(Rev3)
GND
VIN
5V
3V3
fritzing
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
fritzing


ELEGOO
UNO R3
www.elegoo.com
POWER
ANALOG IN
DIGITAL (PWM~)
a b c d e
f g h i j

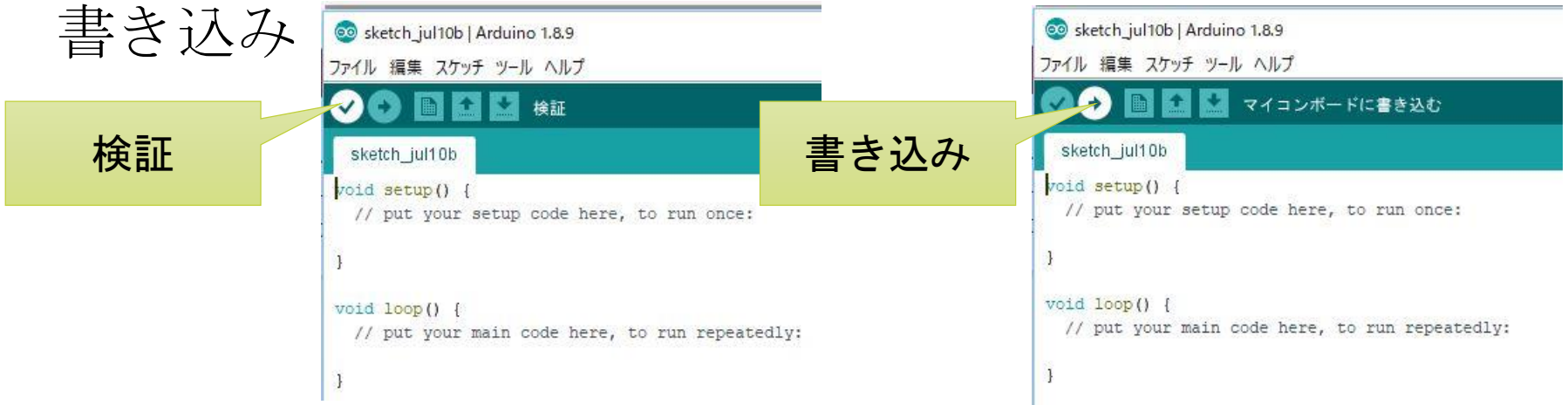
発光ダイオードのプログラム

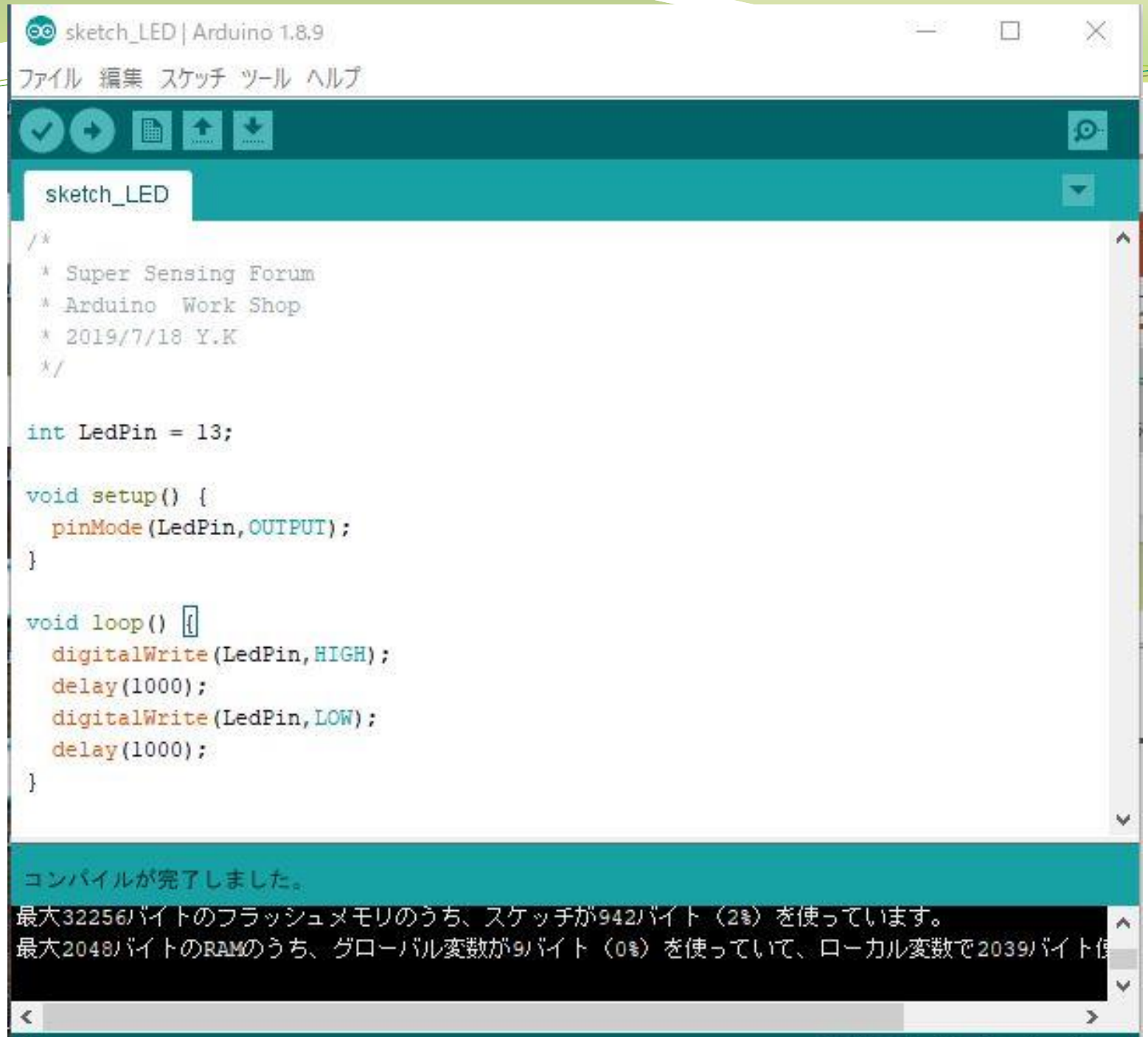
（１）pinMode()とdigitalWrite（）のピン番号は、後で変更を容易にする為、最初に変数定義で、int（整数型）で定義。

（２）Setup()で、pinModeで指定ピンを出力に設定。

（３）loop()で、digitalWriteでHIGH出力とLOW出力を１秒毎（delay）に設定した時間間隔で、トグルさせる。

（４）ファイルメニューの下の検証（コンパイル）とマイコンボードに書き込み




sketch_LED | Arduino 1.8.9
ファイル 編集 スケッチ ツール ヘルプ
sketch_LED
/*
* Super Sensing Forum
* Arduino Work Shop
* 2019/7/18 Y.K
*/
int LedPin = 13;
void setup() {
pinMode(LedPin,OUTPUT);
}
void loop() {
digitalWrite(LedPin,HIGH);
delay(1000);
digitalWrite(LedPin,LOW);
delay(1000);
}
コンパイルが完了しました。
最大32256バイトのフラッシュメモリのうち、スケッチが942バイト（2%）を使っています。
最大2048バイトのRAMのうち、グローバル変数が9バイト（0%）を使っていて、ローカル変数で2039バイト

ワーストケースの考え方

- 発光ダイオードのデーターシートの絶対最大定格から、ダイオードに流す事の出来る最大電流は、３０ｍA。
- 仮に、ＵＳＢ電源が＋５％が上限、発光ダイオードのＶｆのｍｉｎ値は、１．８Ｖ、抵抗値が、－２０％下限になると

  Ｉ＝（５．２５－１．８）／１２０＝２８．７５ｍA

  →　仕様に対して、最悪条件（ワーストケース）を考える事は重要

＜対策＞

- 抵抗値の誤差を５％以内とする。
- 抵抗値自体を少し大きい値にする。→　逆のケースで、暗くなる

目標となる仕様には、ある一定値ではなく、ある範囲という事になる

６．デジタル信号とは

- デジタル信号出力のLOWを‘０’、HIGHを‘１’として、２進数で考える。
- １０進数と２進数（１６進数）の対応表　→
- デジタル信号が、伝わるのは、１本の信号ライン（電線）に、０、１の信号のみ　→　１ビット
- ８ビットには、信号ライン８本
  １６ビットには、信号ライン１６本

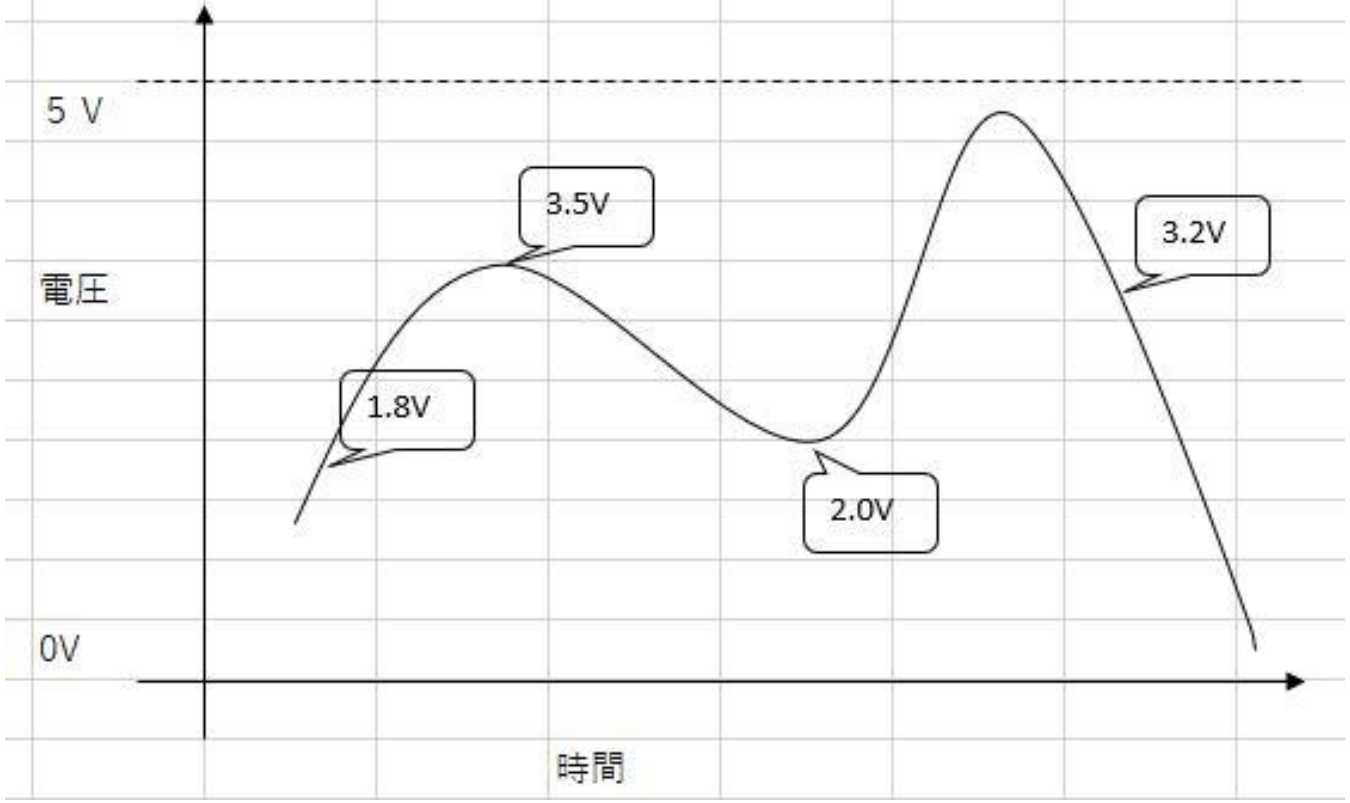
１０進数	２進数	べき乗	１６進数
0	0000	2^0	0
1	0001		1
2	0010	2^1	2
3	0011		3
4	0100	2^2	4
5	0101		5
6	0110		6
7	0111		7
8	1000	2^3	8
9	1001		9
10	1010		a
11	1011		b
12	1100		c
13	1101		d
14	1110		e
15	1111	2^4-1	f
16	10000	2^4	10
255	11111111	2^8-1	ff

ビット演算について

- Ａｒｄｕｉｎｏ日本語リファレンスのビット演算子を参照のこと。
  ビットで演算した方が簡単な場合がある。
- Ｃ言語では、次の複合演算子もよく用いる。

７．アナログ信号とは

- 時間的に信号ラインの中で、電圧（アナログ量）が変動していくもの



- 但し、ビット演算などの処理はできない

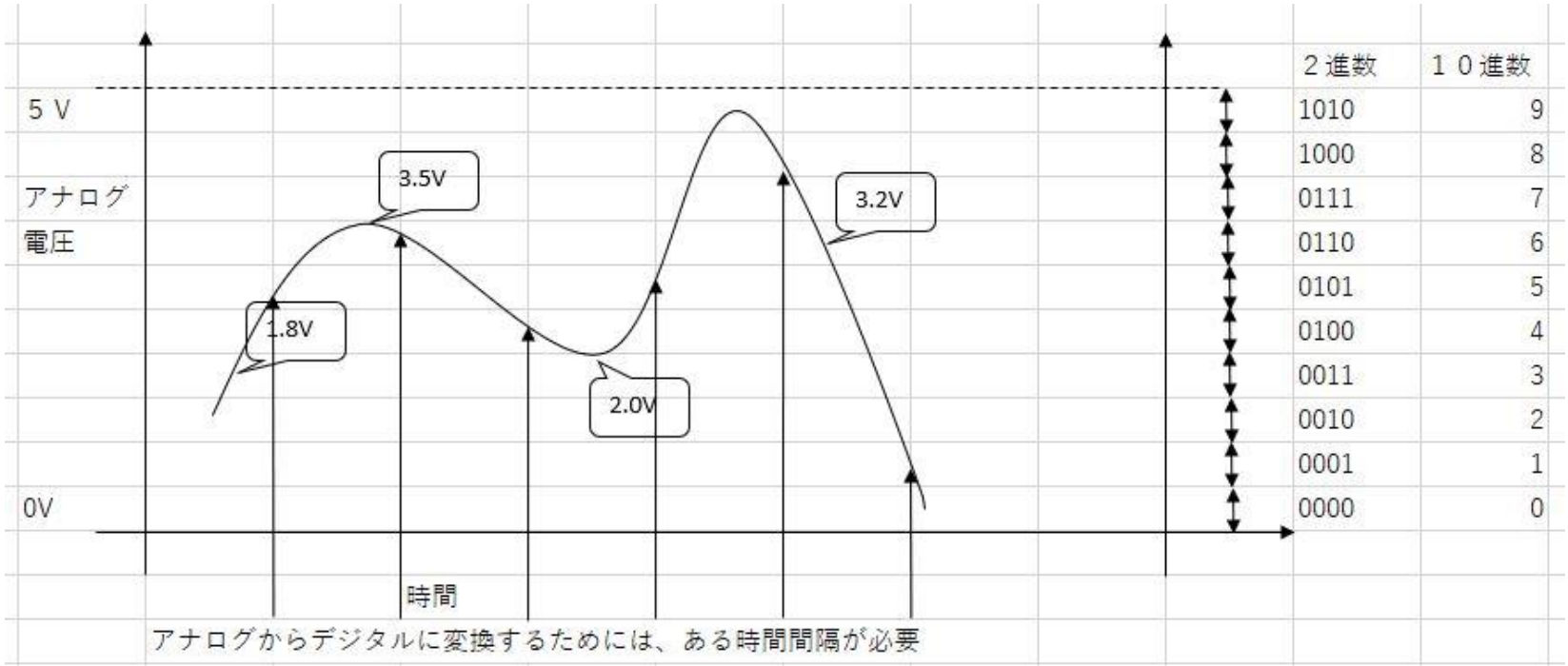
アナログ→デジタル変換について

＜アナログ信号からデジタル信号への変換器（ＡＤコンバータ）＞

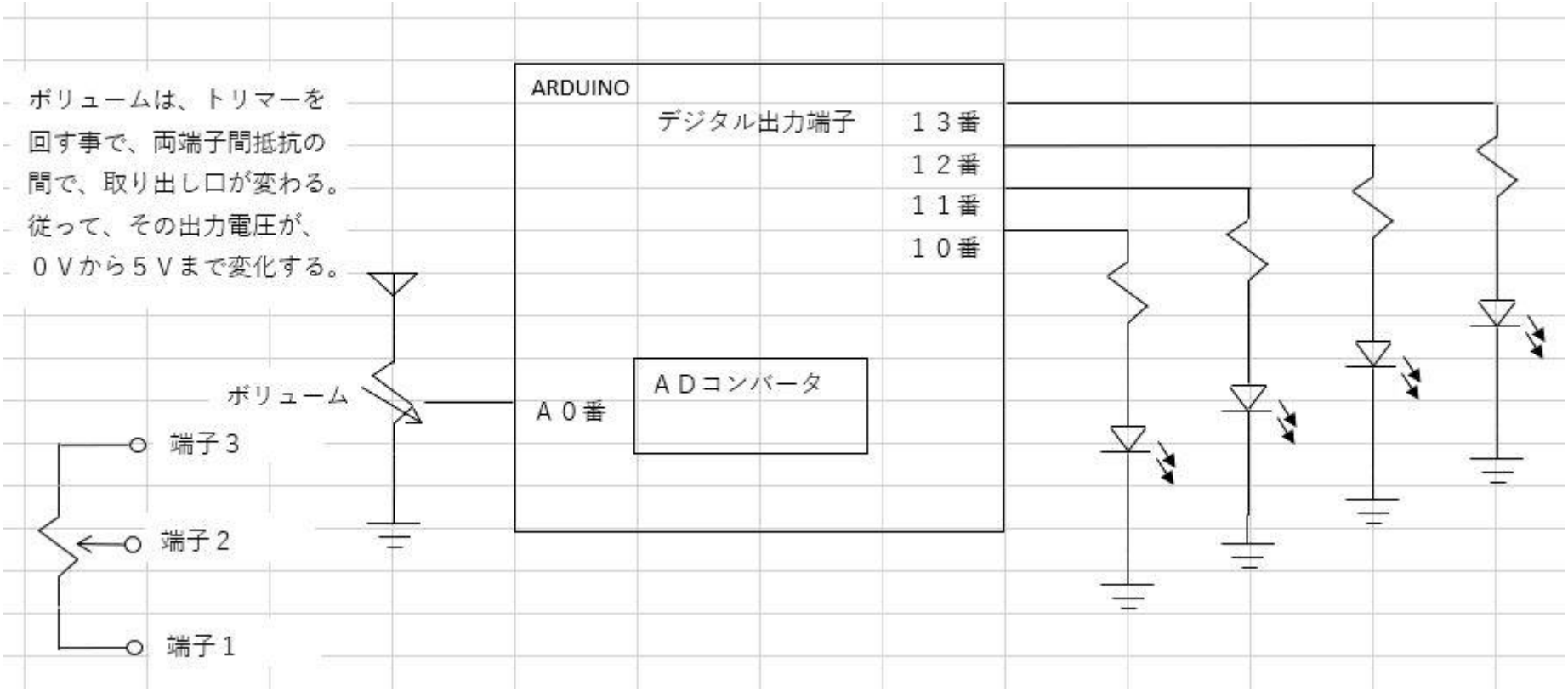
ある一定の時間間隔で、アナログ信号からデジタル信号へ変換する

その時間間隔をサンプリング周期（逆数をサンプリング周波数）

また、ＡＤ変換器が動作できる時間間隔以上の時間であれば、不定期な時間間隔で、デジタル信号に変換しても良し。

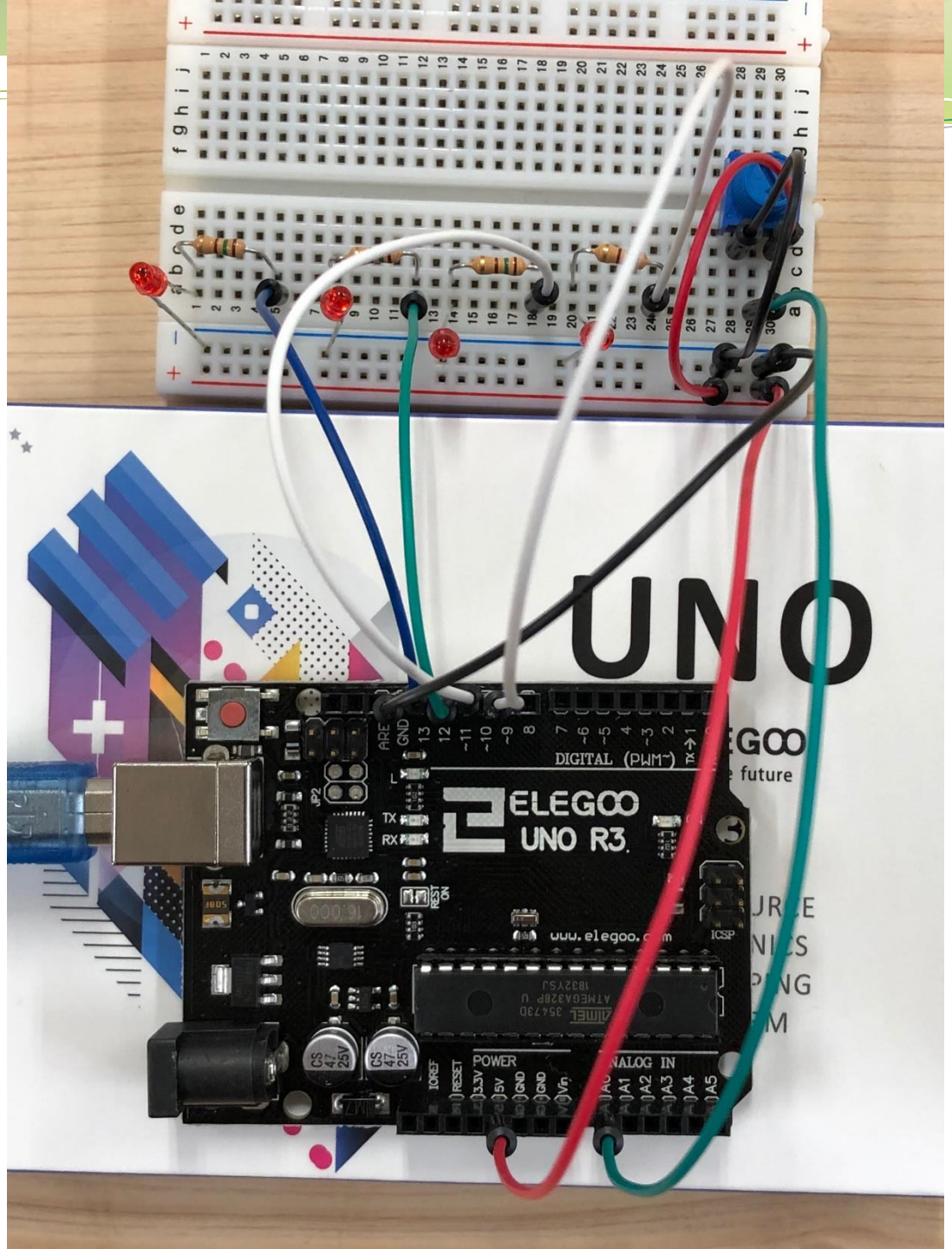


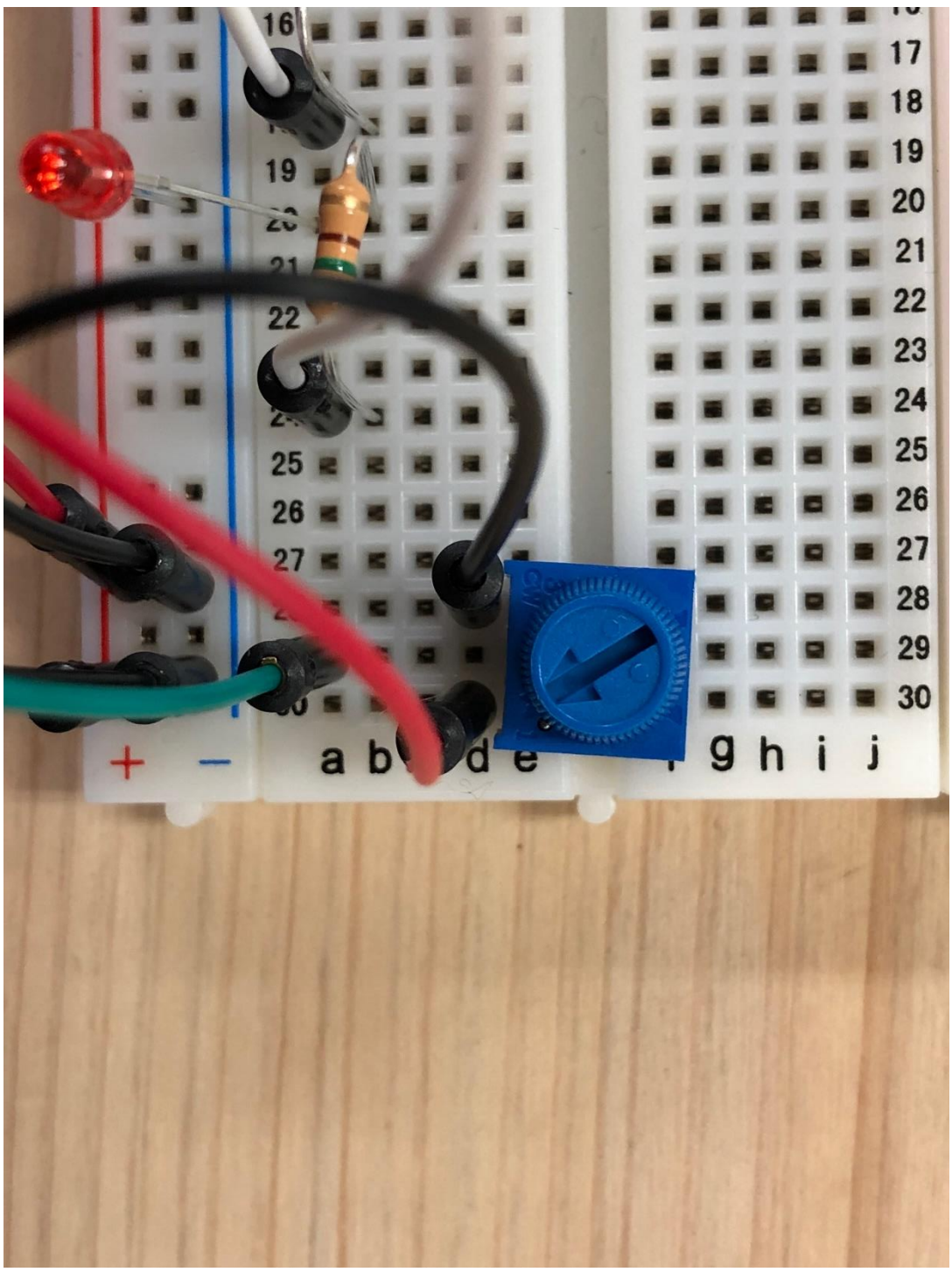
８．ＬＥＤを光らせよう　その２



ブレッドボードの配線　その２

- 各自で配線してみよう。


UNO
ELEGOO
UNO R3
DIGITAL (PWM~)
POWER
ANALOG IN
www.elegoo.com


16
17
18
19
19
20
21
21
22
22
23
24
25
25
26
26
27
27
28
29
30
+
−
a b d e
g h i j

アナログ入力のプログラム

- Ａｒｄｕｉｎｏ日本語リファレンスのアナログ入出力関数を参照。
- ＡＤコンバータは、１０ビット。従って、０から１０２３までの値を出力する。アナログ入力範囲は、０から電源（５Ｖ）まで。
- 抵抗ボリュームは、０から電源（５Ｖ）まで、出力電圧が、変化。

プログラムの仕様

- ボリュームの出力電圧をＡＤ変換した１０ビットのデータの内、
  上位４ビットのデータを、そのまま１０番から１３番までの
  発光ダイオードで表す。また、大体１秒毎にＡＤ変換を行う。

＜使うプログラム＞
　アナログ入力、
　ビット演算子、
　制御文（ｉｆなど）

プログラム作成してみよう

体験できた事

- ＡＤ変換のレートが早い（サンプリング周波数が高い）→　アナログの変動に対する追従性が良好。
- ボリュームがある一定以上変化しないと発光ダイオードが変わらない。
  →　ビット分解能が低い

ＡＤ変換に重要な項目　→　ビット分解能、変換レート

但し、ビット分解能が細かくても、

ビット間が等間隔　→　ＩＮＬ、ＤＮＬ、ひずみ率（Distortion）

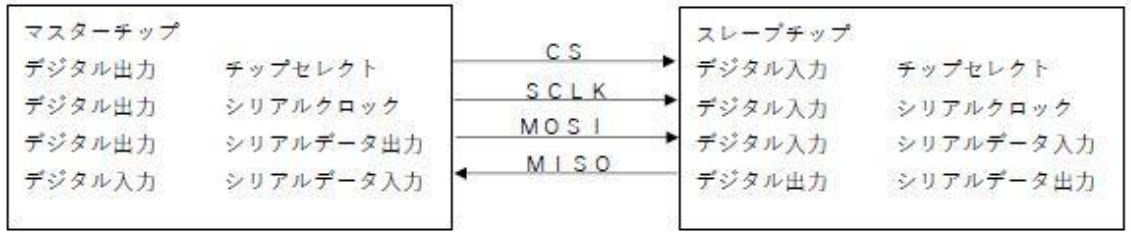
ノイズの影響　→　ＳＮ比

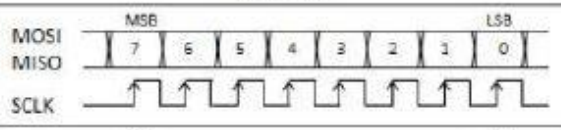
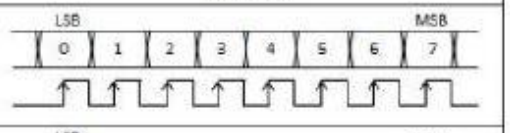
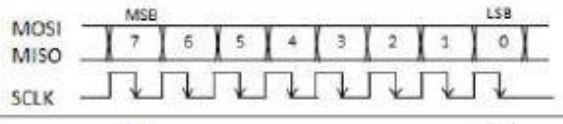
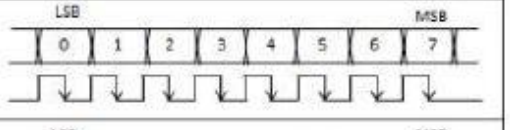
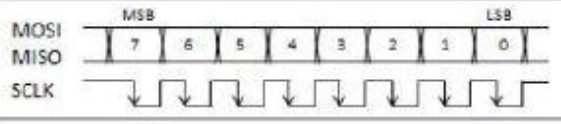
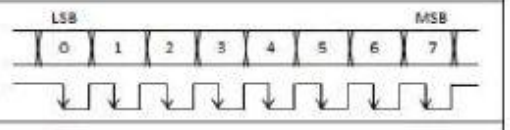
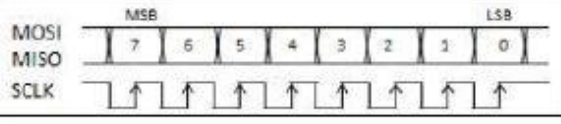
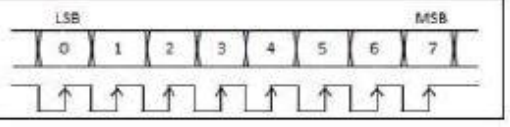
９．センサーのデジタルインターフェース

- 最近のセンサーデバイスは、ＡＤ変換を内臓して、出力がすでにデジタル信号になっているものが多い。
- 出力が、１０ビットだと、１０本のデジタル信号線が必要となり、効率的でない。
- 従って、デジタル出力信号を時間的に分けて送るシリアル伝送（インターフェース）が主流になっている。
- センサーなどの用途とマイコン通信用途には、次の３種類のインターフェースが使われている。

ＳＰＩ、ＵＡＲＴ、Ｉ２Ｃ

＜ＳＰＩ＞

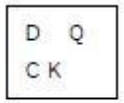


Setting	MSBFIRST	LSBFIRST
SPI_MODE_0		
SPI_MODE_1		
SPI_MODE_2		
SPI_MODE_3		

＜回路内部の原理＞

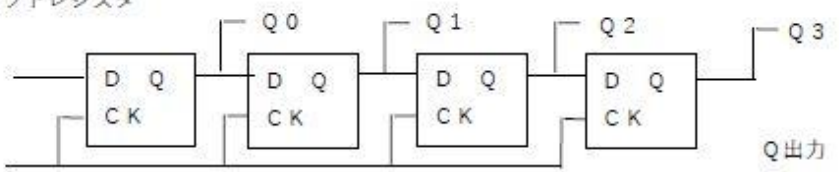
ロジックセル

Ｄフリップフロップ

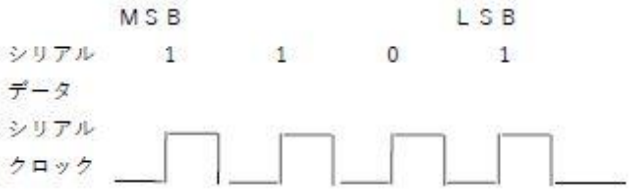


クロック入力の立ち上がりで
Ｄ入力の信号Ｈ、Ｌを取り入れる
取り入れた信号をＱ端子から出力

シフトレジスタ



Ｑ出力	出力データ
Ｑ３	1
Ｑ２	1
Ｑ１	0
Ｑ０	1

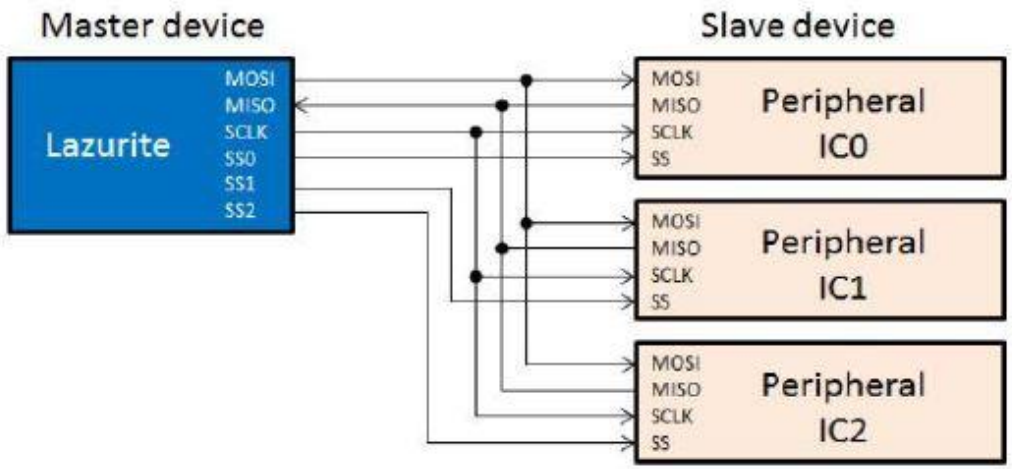


シリアルーパラエル変換

この逆

パラエルーシリアル変換

＜１個のマスターチップに複数のスレーブチップを接続する事が可能＞



チップセレクトが、例えば、'Ｌ'であるチップのみシリアルーパラエル変換する

SPIインターフェースの波形例

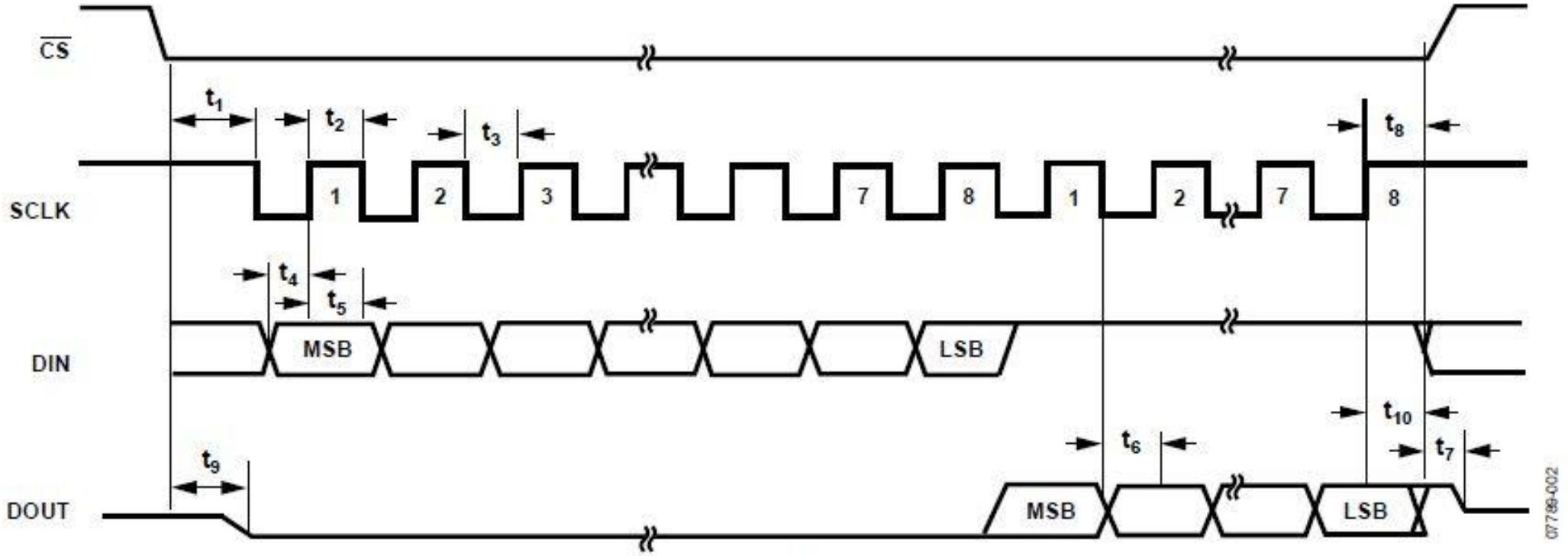


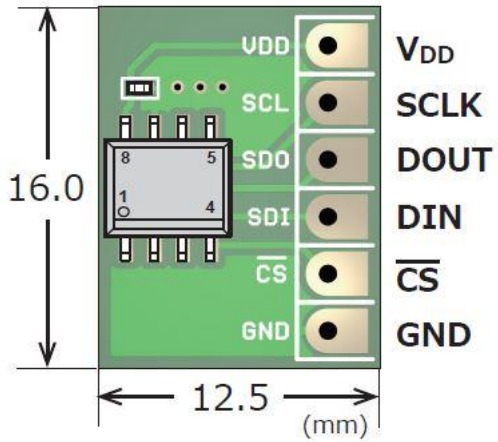
Figure 2. Detailed SPI Timing Diagram

モードは、３番　　MSBファースト

１１．温度センサーを動かそう

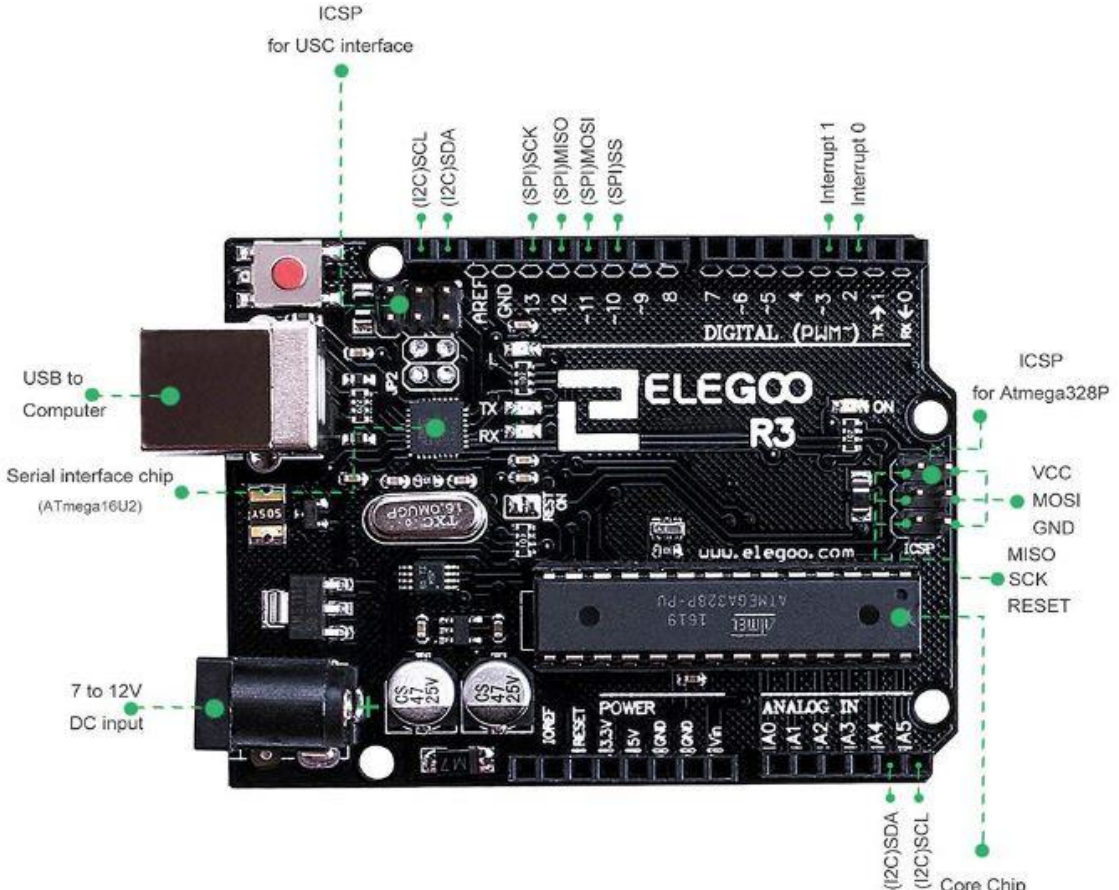
- ＳＰＩインターフェースを使って温度センサーモジュール（ＡＤＴ７３１０）を動かす。
- ボード配線を行う。

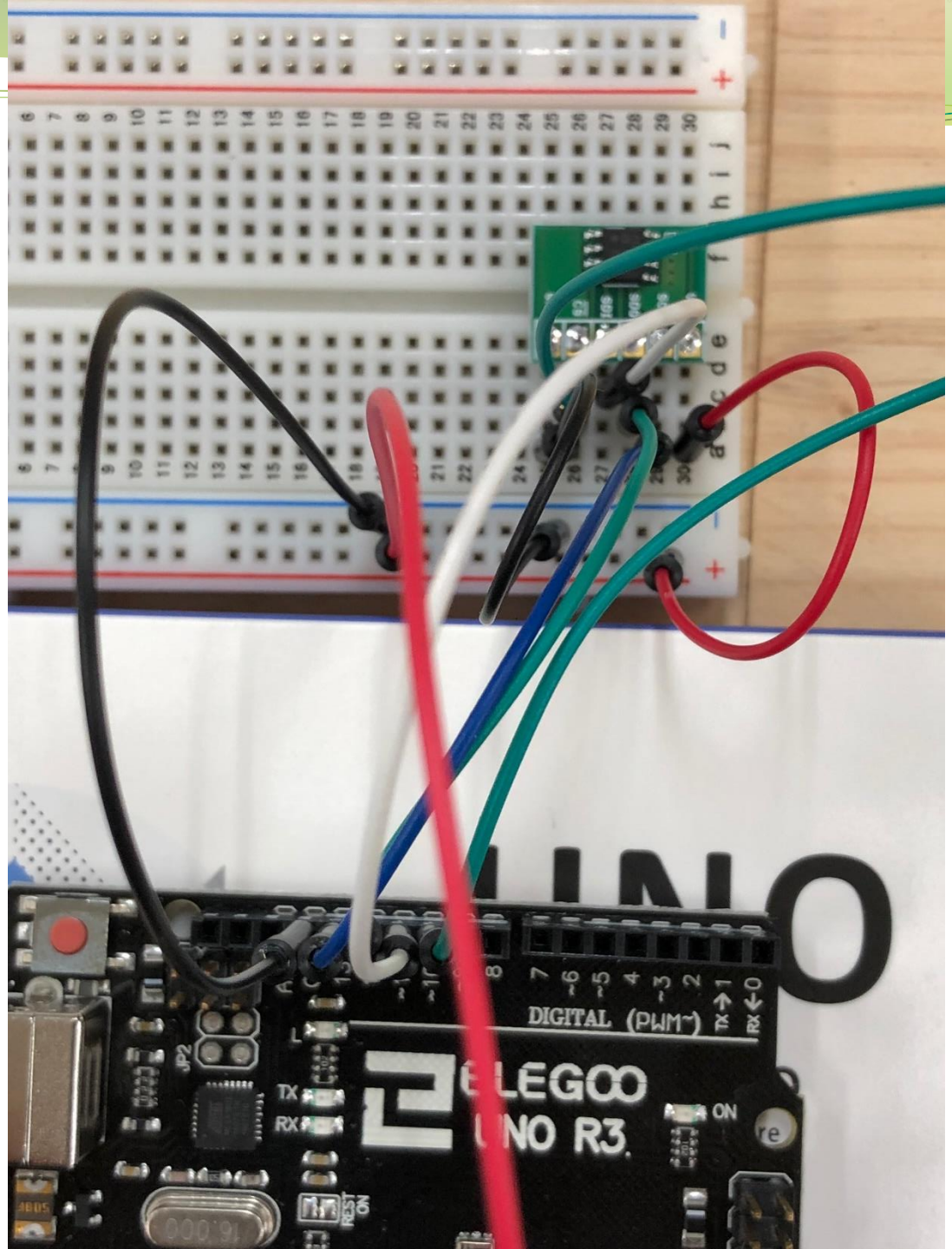
■基板外観と端子配置



（約２倍スケールです）

基板シルク印刷の都合で
SCL(SCLK)、SDO(DOUT)、
SDI(DIN) となっています。




DIGITAL (PWM~)
LEGOO
UNO R3
ON

データシートについて

- デバイスの特性をまとめたもので、世界共通語

＜構成＞

１．用途、概要、目次
２．ブロック図、回路構成
２．ピン仕様（ピン配置図、ピン内容、パッケージ図）
３．特性説明（絶対最大定格、動作条件、電気的特性、
　　デジタル特性（ＡＣ、ＤＣ、タイミングチャート）
４．機能詳細説明
５．レジスタ説明（レジスタマップ全体、各レジスタ項目）
６．パッケージ寸法、リフロー条件
７．推奨外部接続図例
８．注意事項（損害、軍事利用、用途外使用、法令）

ＡＤＴ７３１０のデーターシート

ADT7310のデーターシートの内容確認

レジスタマップ

Table 6. ADT7310 Registers

Register Address	Description	Power-On Default
0x00	Status	0x80
0x01	Configuration	0x00
0x02	Temperature value	0x0000
0x03	ID	0xCX
0x04	T_{CRIT} setpoint	0x4980 (147°C)
0x05	T_{HYST} setpoint	0x05 (5°C)
0x06	T_{HIGH} setpoint	0x2000 (64°C)
0x07	T_{LOW} setpoint	0x0500 (10°C)

温度と出力コードの関係

Table 5. 13-Bit Temperature Data Format

Temperature	Digital Output (Binary) Bits[15:3]	Digital Output (Hex)
−55°C	1 1100 1001 0000	0x1C90
−50°C	1 1100 1110 0000	0x1CE0
−25°C	1 1110 0111 0000	0x1E70
−0.0625°C	1 1111 1111 1111	0x1FFF
0°C	0 0000 0000 0000	0x000
+0.0625°C	0 0000 0000 0001	0x001
+25°C	0 0001 1001 0000	0x190
+50°C	0 0011 0010 0000	0x320
+125°C	0 0111 1101 0000	0x7D0
+150°C	0 1001 0110 0000	0x960

温度の連続読みだしモードを使うので、その部分のデーターシートを参照

ADT7310のSPIレジスタ

Table 6. ADT7310 Registers

Register Address	Description	Power-On Default
0x00	Status	0x80
0x01	Configuration	0x00
0x02	Temperature value	0x0000
0x03	ID	0xCX
0x04	T_{CRIT} setpoint	0x4980 (147°C)
0x05	T_{HYST} setpoint	0x05 (5°C)
0x06	T_{HIGH} setpoint	0x2000 (64°C)
0x07	T_{LOW} setpoint	0x0500 (10°C)

Table 15. Command Byte

C7	C6	C5	C4	C3	C2	C1	C0
0	R/$\overline{W}$	Register address			Continuous read	0	0

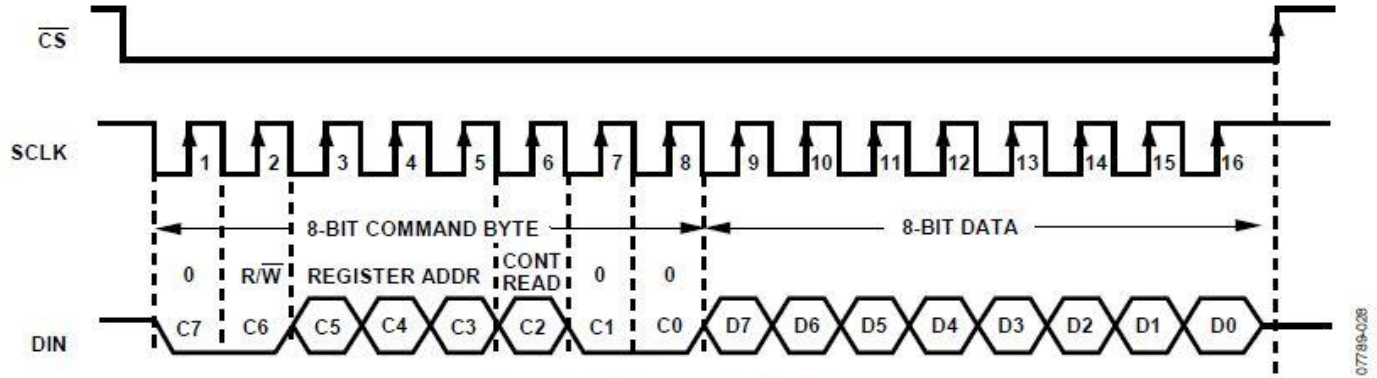


Figure 18. Writing to an 8-Bit Register

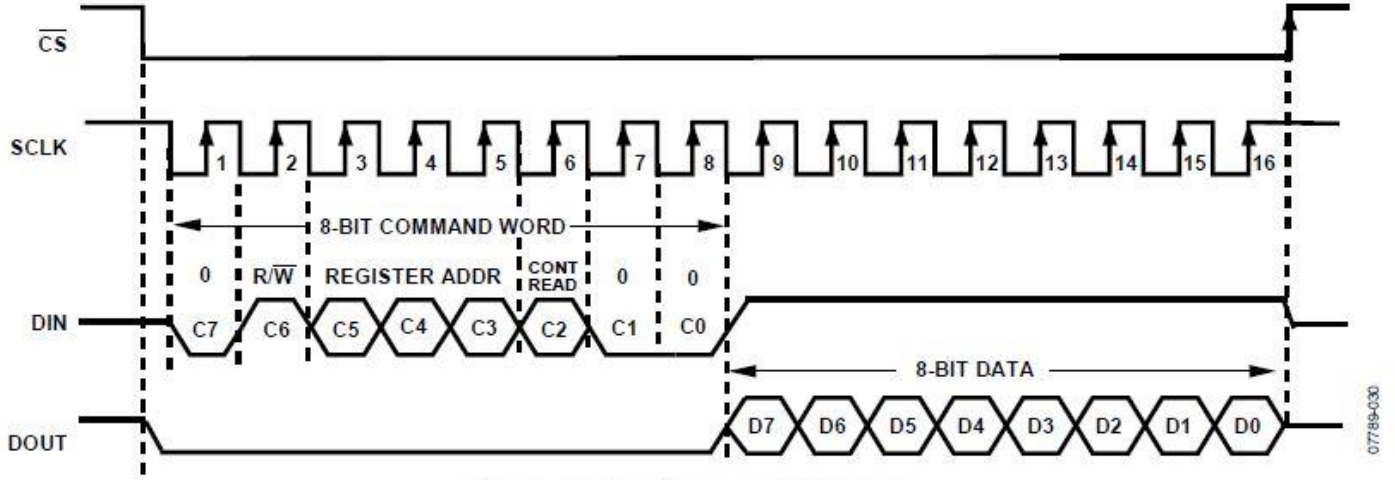


Figure 20. Read from an 8-Bit Register

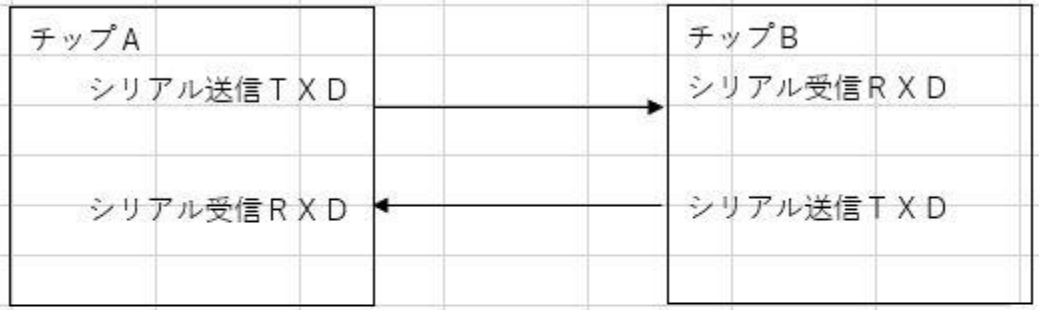
温度センサーのプログラミング

今回は、連続で温度データを読み取れるようにする。
（１）#includeで<SPI.h>を定義
＜setup＞
（２）１０番ピンを出力に定義して、まず、HIGHを出力
（３）SPI.beginでスタート
（４）SPIモード、MSBファースト、クロック分周を定義
（５）１０番ピンをLOW出力（チップセレクト状態）
（６）SPI.transfer(0x54)で、連続読みだしの命令をレジスタに設定
（７）２４０mS待つ
＜loop＞
（８）int　val；　float　temp；を定義
（９）val=SPI.transfer(0)で8ビットを読み出して、valの上位1バイトとする（8ビットシフト）
（１０）val=SPI.transfer(0)で同様に読み出して、valの下位1バイトとする（ORと取る）
（１１）浮動小数点変数tempにvalを１６．０で割った値を入れる
（１２）１秒待つ

１２．ＵＡＲＴインターフェース

ＳＰＩと異なり、対等な１対１のシリアル通信

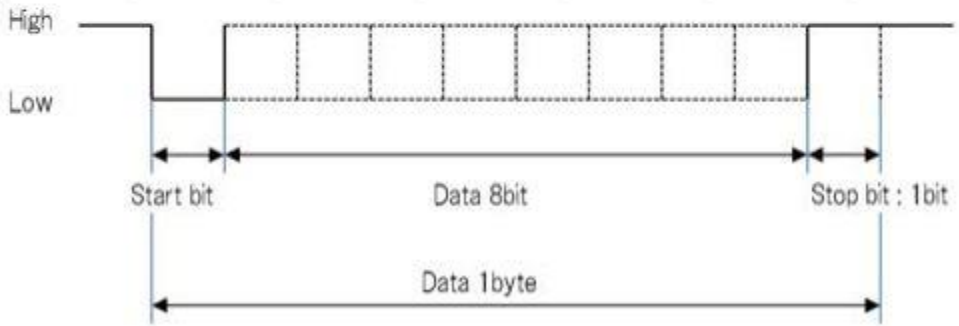
＜ＵＡＲＴ通信＞　Universal Asynchronous Receiver and Transmitter

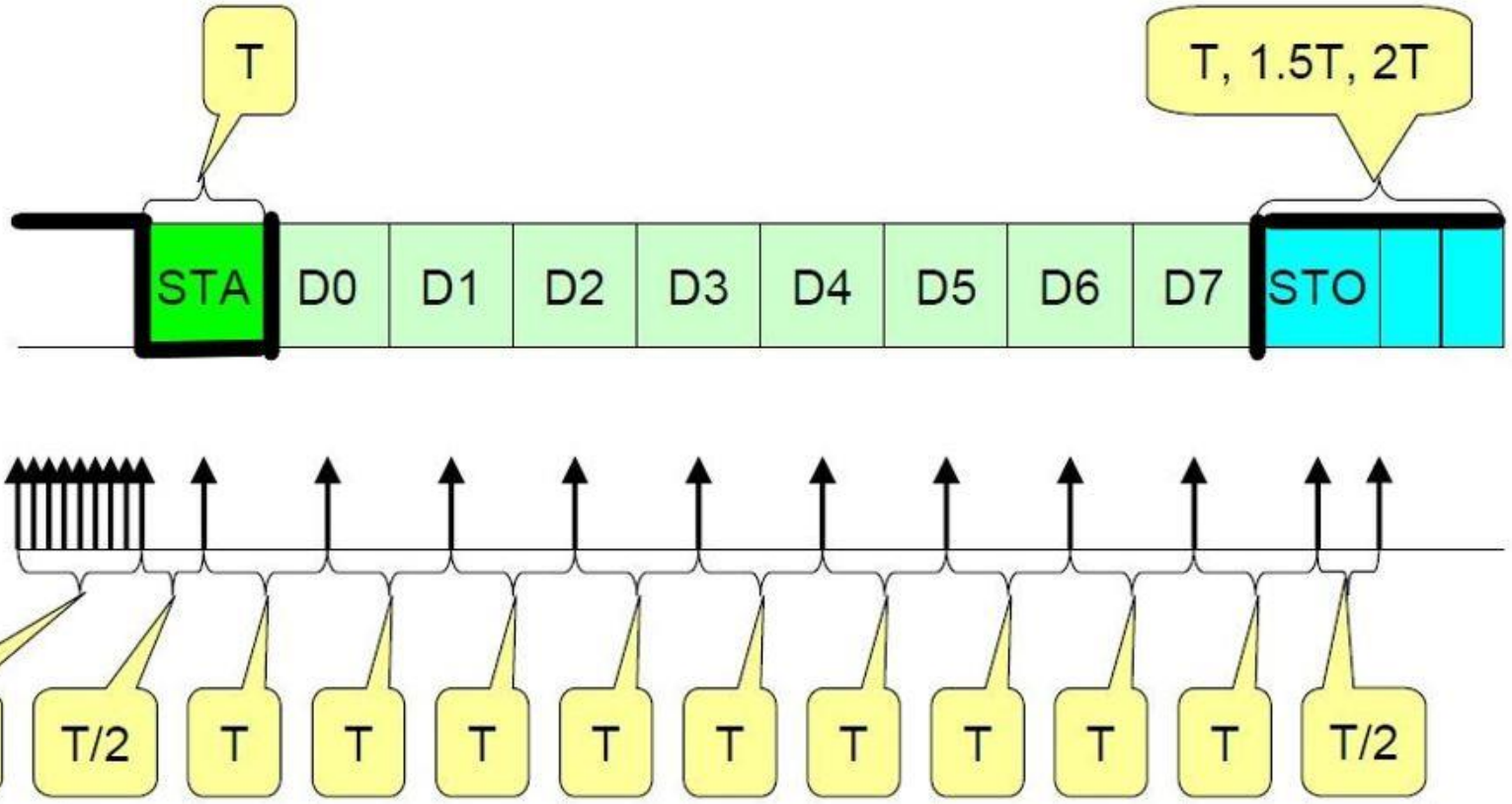


（１）　送信の始めは、必ず１ビット分のスタートビットあり
（２）　ＬＳＢから８ビットのデータ送信
（３）　終了に、ストップビット　１ビット

ＴＸＤもＲＸＤも同じ送信ー受信フォーマット

データ：８ビット
パリティ：なし
ストップビット：１ビット





（1）データレート（ボーレート）よりも早い周期で、受信データの立下りを検出し、そこからボーレートの半分の周期後に、データが'L'であれば、スタートビットと判断

（2）スタートビットを検出したら、その後は、ボーレート毎に、データを取り込み（シリアルクロックで取り込むのと同じ）

（3）８ビットのデータ取得後、ストップビット（'H'）を検出出来たら、正常に受信できたと判断する

つまり、あらかじめ、送信側と受信側で、ボーレートは、いくつであるか決めておく必要がある

規格で決まっているボーレート（ｂｐｓ）

2400	4800	9600	19200	38400	57600	76800	115200

１３．ＵＡＲＴを使ってみよう

- ＢＬＥなどの通信モジュールのインターフェースに使われている
- また、ＵＳＢ接続を介したＰＣとＡＲＤＵＩＮＯのデータのやり取りにも使われている
- Ａｒｄｕｉｎｏ日本語リファレンスのシリアル通信を参照のこと。
- 今回は、Serial.printを使う。

温度センサーのプログラミング　追加

今回は、連続で温度データを読み取れるようにする。

（１）#includeで<SPI.h>を定義

＜setup＞

（２）１０番ピンを出力に定義して、まず、HIGHを出力

（３）SPI.beginでスタート

（４）SPIモード、MSBファースト、クロック分周を定義

（５）１０番ピンをLOW出力（チップセレクト状態）

（６）SPI.transfer(0x54)で、連続読みだしの命令をレジスタに設定

（７）２４０mS待つ

＜loop＞

（８）int　val；　float　temp；を定義

（９）val=SPI.transfer(0)で8ビットを読み出して、valの上位1バイトとする（8ビットシフト）

（１０）val=SPI.transfer(0)で同様に読み出して、valの下位1バイトとする（ORと取る）

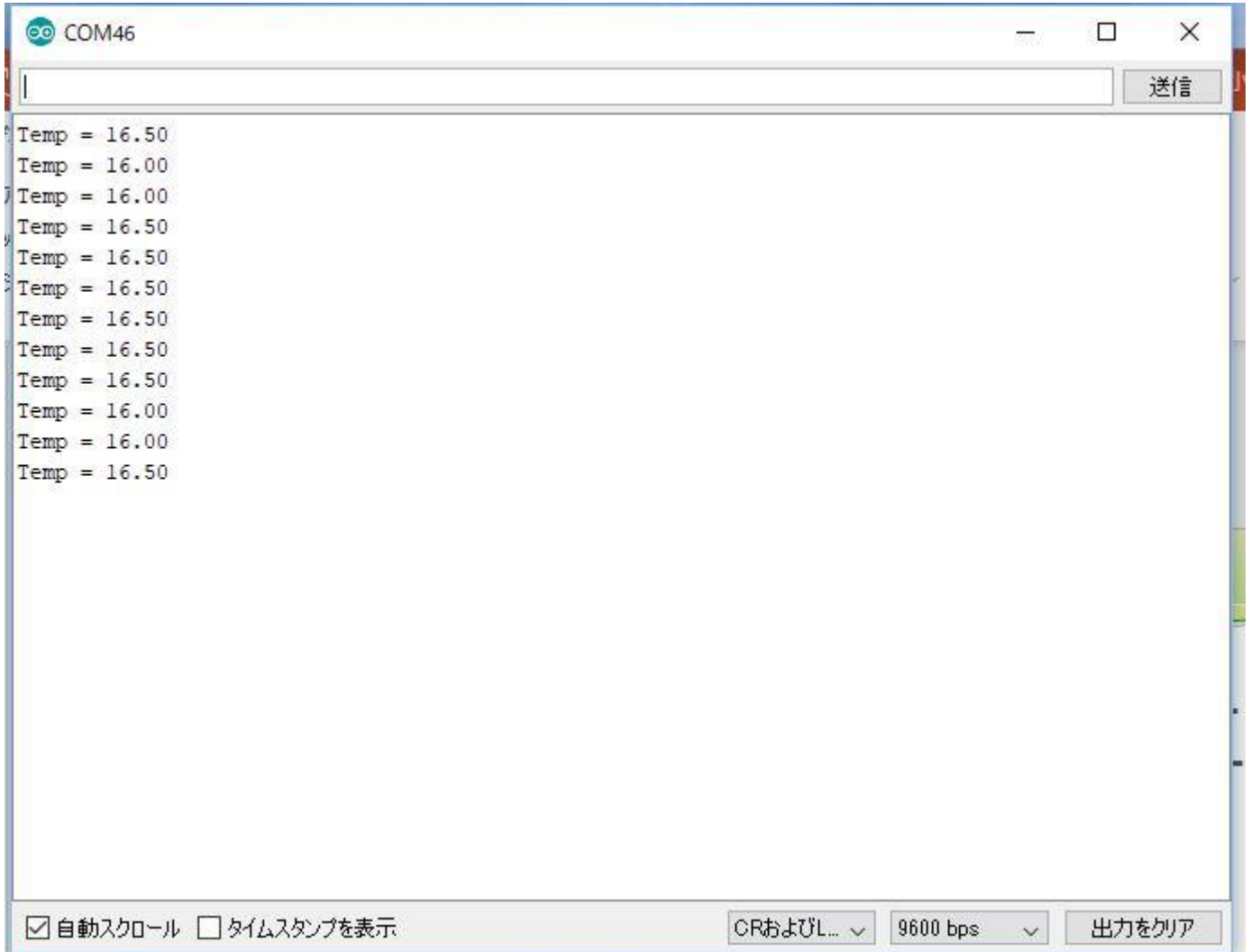
（１１）浮動小数点変数tempにvalを１６．０で割った値を入れる

（１２）１秒待つ

Serial.begin(115200)

Serial.print(“temp = “);
Serial.println(temp,2);

ARDUINOでのシリアル通信モニター



１４．Ｉ２Ｃインターフェース

＜Ｉ２Ｃ＞

スレーブチップは、複数接続可能

マスターチップ
デジタル入出力シリアルクロックＳＣＬ
デジタル入出力　シリアルデータＳＤＡ

ＳＣＬ
ＳＤＡ

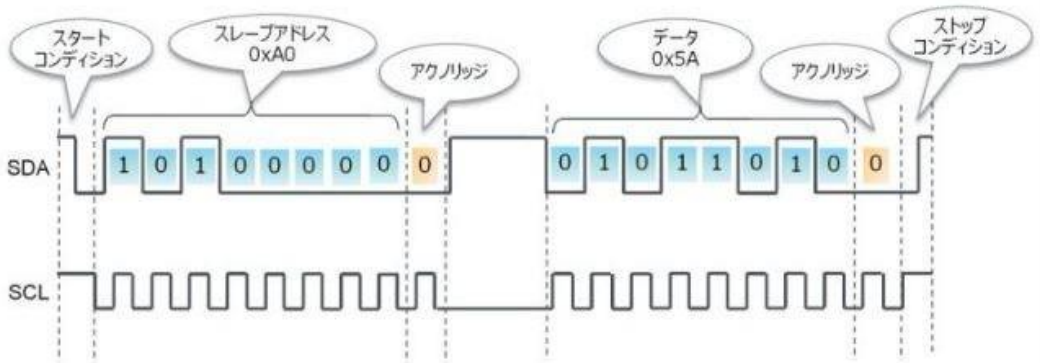
スレーブチップ
デジタル入出力シリアルクロックＳＣＬ
デジタル入出力　シリアルデータＳＤＡ

電源
電源
抵抗
抵抗

スレーブチップ
デジタル入出力シリアルクロックＳＣＬ
デジタル入出力　シリアルデータＳＤＡ

スレーブチップ
デジタル入出力シリアルクロックＳＣＬ
デジタル入出力　シリアルデータＳＤＡ

ＳＣＬ、ＳＤＡともに、プルアップ抵抗が必要



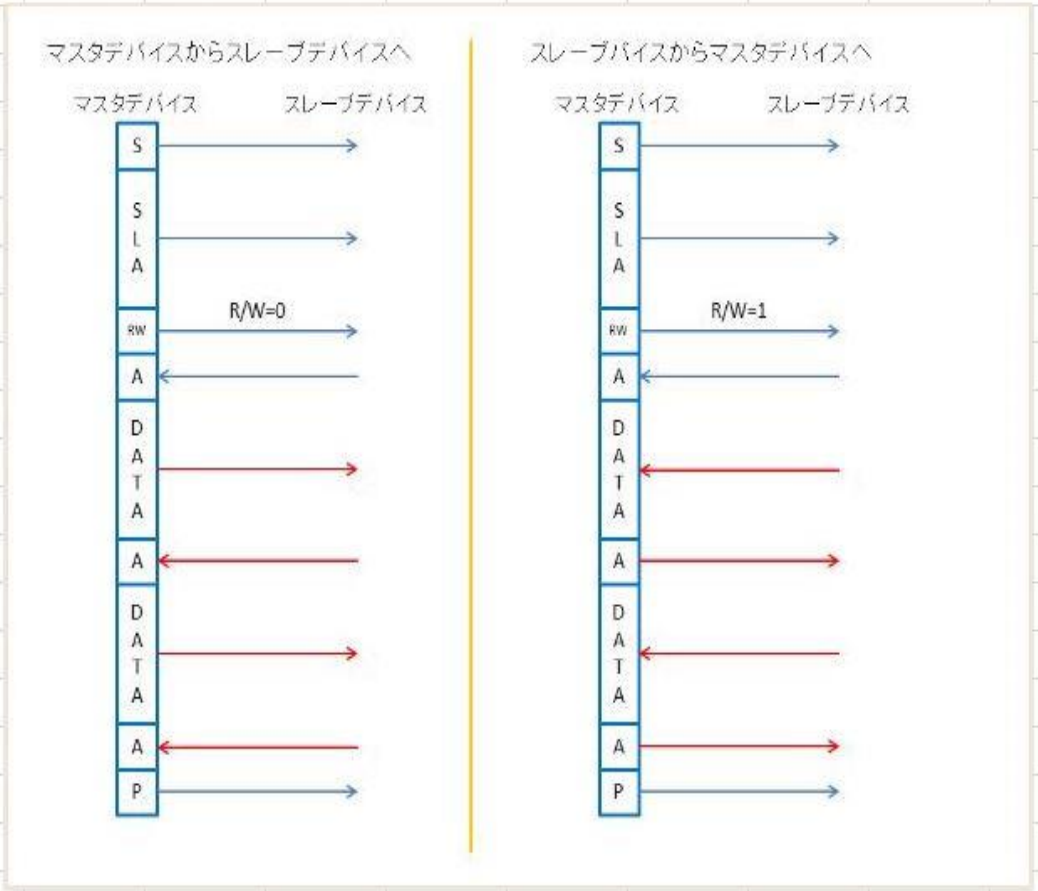
ＳＰＩインターフェースのように、チップセレクト信号がないが、スレーブチップを区別する必要がある

マスターチップが送信するシリアルデータに、スレーブチップ毎の区別データ（スレーブアドレス）を最初に入れる

Ｉ２Ｃインターフェース

I2C通信の基本フォーマットを次に示す。

項目	内容	発行主体	説明
S	スタート	M	通信開始
ＳＬＡ	スレーブアドレス	M	このアドレス（７ビット）に一致するスレーブチップ（S）が、今後、応答する
R／W	リードライト	M	０の時は、MからSへ書き込み　１の時は、SからMへ書き込み
A	アクノリッジ	S	応答するSが、'L'を返答　'L'でなければ、プルアップされているので'H'となる
			対応Sがないか、異常事態が発生したと認識し、通信ストップ
ＤＡＴＡ	データ	M／S	M→S、S→Mへのデータ転送が８ビット行われる
A	アクノリッジ	M／S	受け手側が、受信できた場合に返答する
P	ストップ	M	通信終了



I2CのArduino

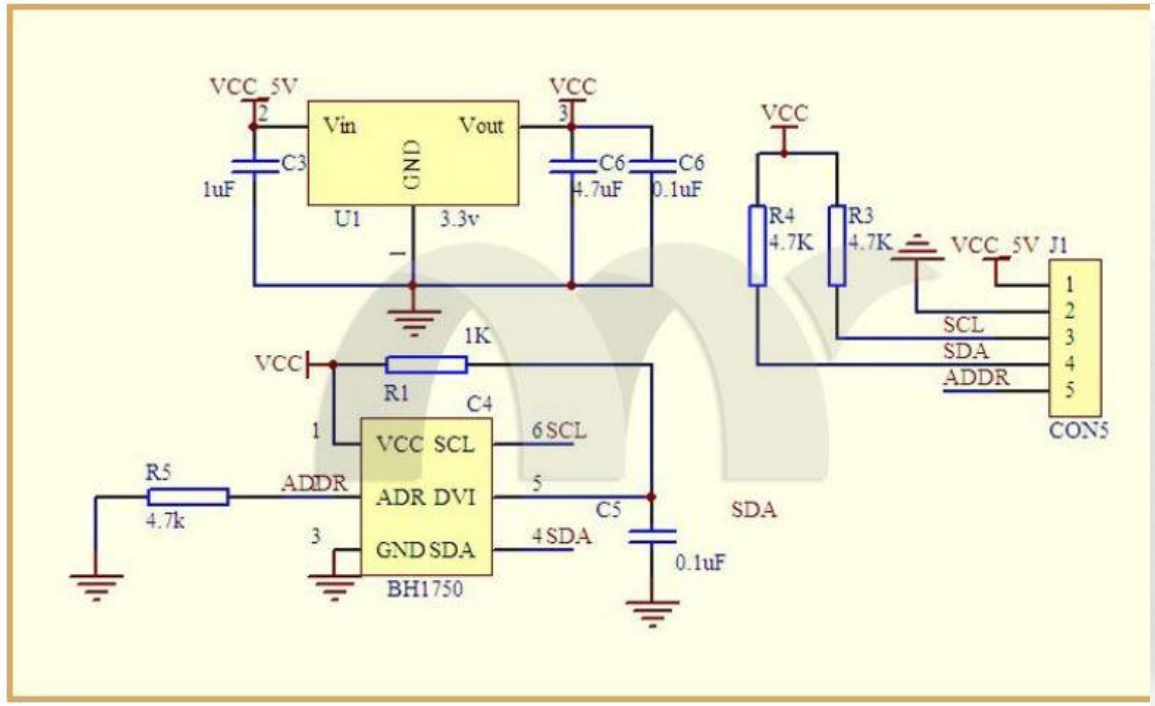
```
#include <Wire.h>

#define i2c_add 0x23

void i2c_write(int adr,int wdata) {
  int st;
  Wire.beginTransmission(i2c_add);
  Wire.write(adr);
  Wire.write(wdata);
  st = Wire.endTransmission();
  while (st != 0) {
    st = Wire.endTransmission();
    delay(10);
  }
}
```

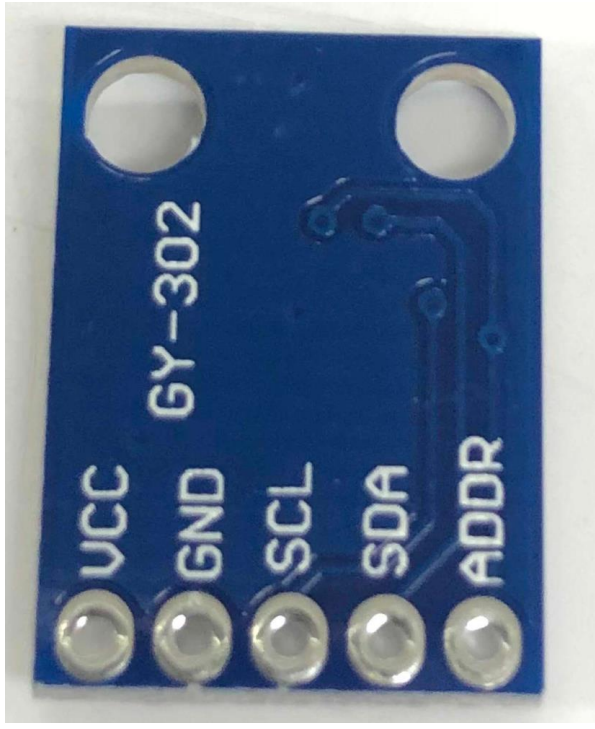
```
int i2c_read(int adr) {
  int dread;
  int st;
  Wire.beginTransmission(i2c_add);
  Wire.write(adr);
  st = Wire.endTransmission();
  while (st != 0) {
    st = Wire.endTransmission();
    delay(10);
  }
  Wire.requestFrom(i2c_add,1);
  dread = Wire.read();
  st = Wire.available();
  while (st !=0) {
    st = Wire.available();
    delay(10);
  }
  return dread;
}
```

```
void setup() {
  int rdata;

  i2c_write(0x00,0x11);

  rdata = i2c_read(0x22);

}
```

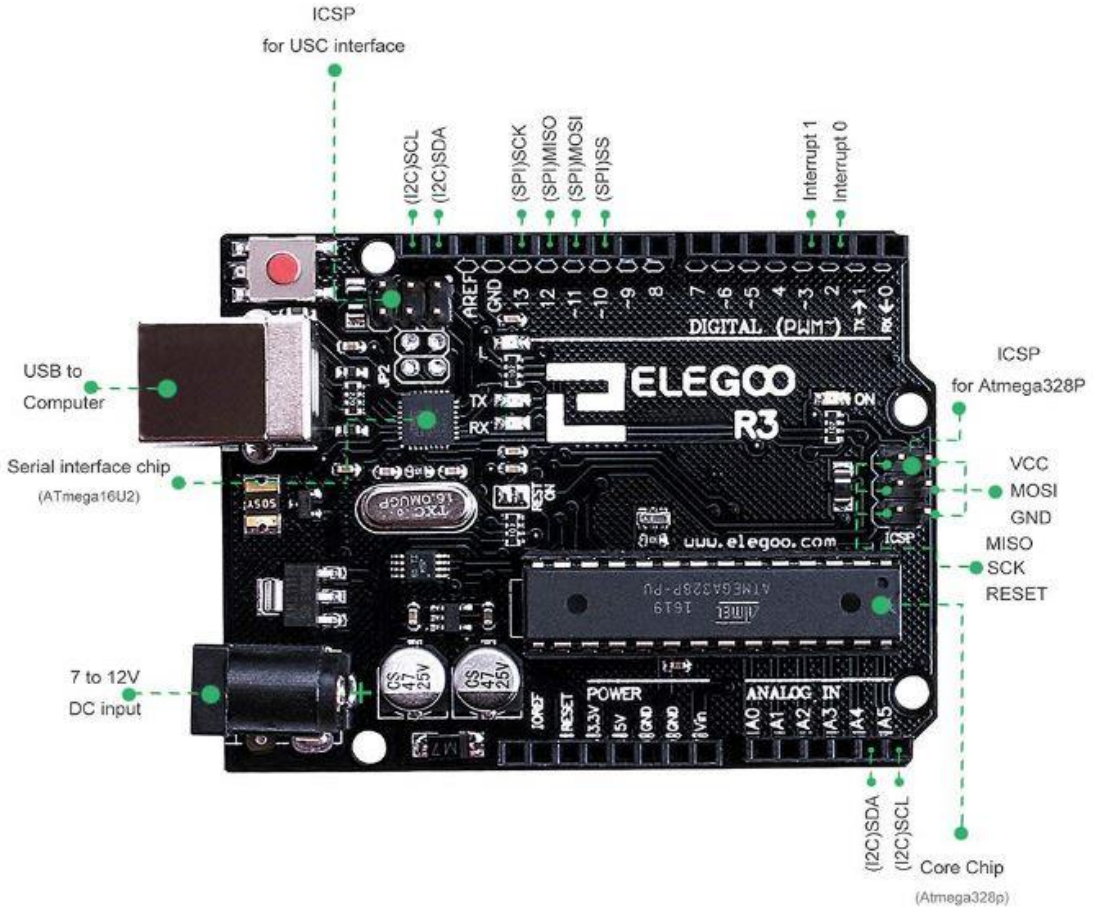
１５．光センサーを動かしてみよう

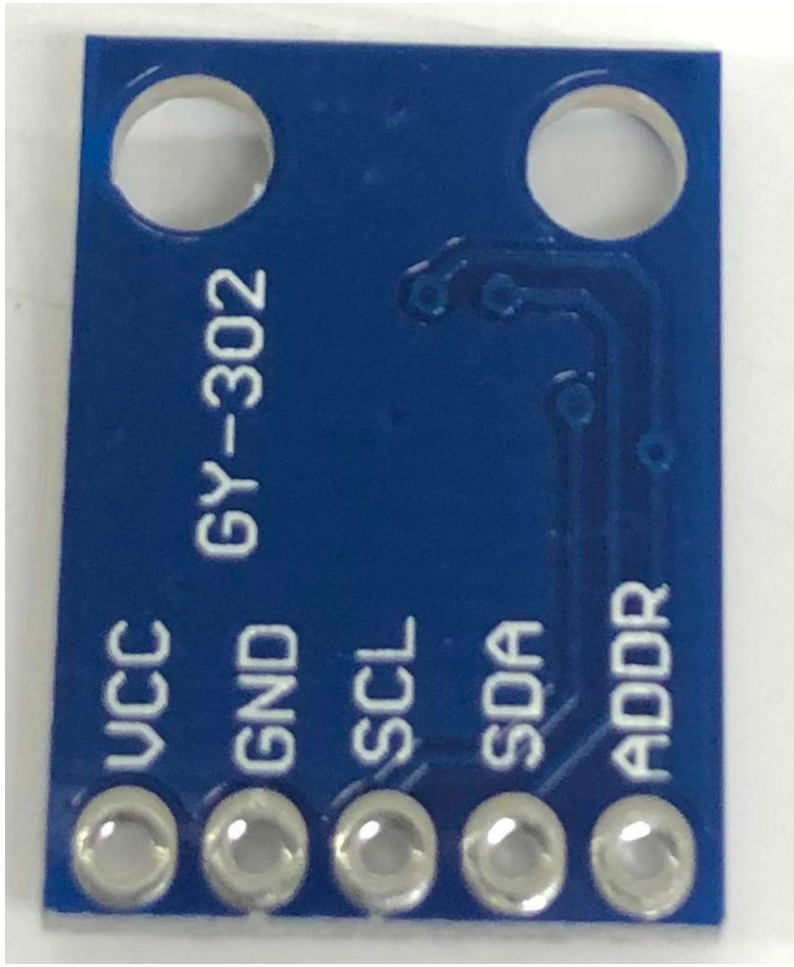
- 光センサーモジュール　Ｉ２Ｃインターフェースで動かす
- センサーは、BH1750（ローム社製）　データーシートを参照の事
- モジュールの回路図は以下の通り

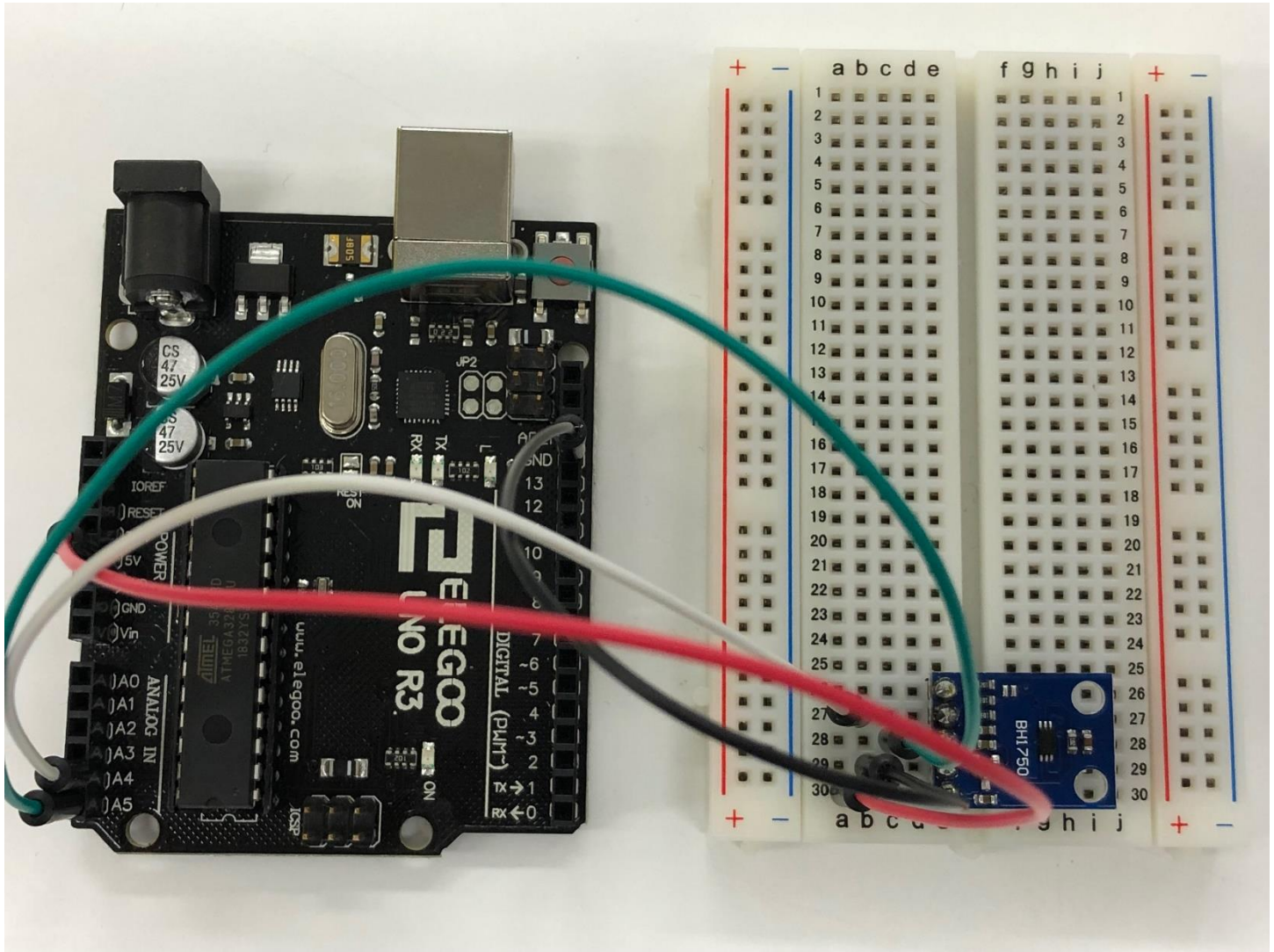




配線してみよう


ICSP
for USC interface
(I2C)SCL
(I2C)SDA
(SPI)SCK
(SPI)MISO
(SPI)MOSI
(SPI)SS
Interrupt 1
Interrupt 0
USB to
Computer
Serial interface chip
(ATmega16U2)
7 to 12V
DC input
ICSP
for Atmega328P
VCC
MOSI
GND
MISO
SCK
RESET
(I2C)SDA
(I2C)SCL
Core Chip
(Atmega328p)
DIGITAL (PWM~)
POWER
ANALOG IN
ELEGOO
R3
www.elegoo.com


GY-302
VCC
GND
SCL
SDA
ADDR


a b c d e
f g h i j
BH1750
UNO R3
www.elegoo.com
ANALOG IN
DIGITAL (PWM~)
POWER

GitHubライブラリーを使用する方法

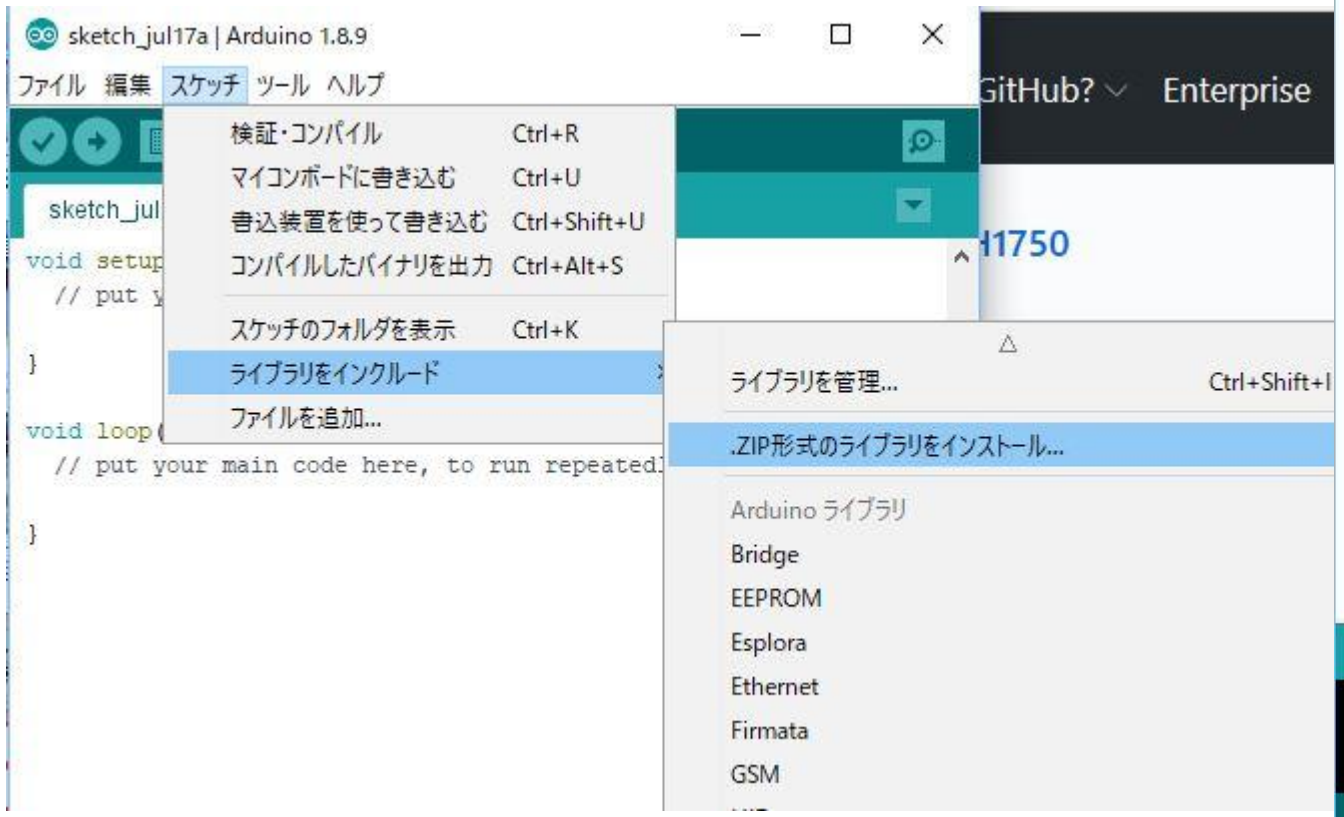
- 世の中にあるプログラムのソースファイルは、GitHubというサイトに公開されている。活用すると簡単にプログラミングする事ができる。
- 検索すると、大抵のセンサーに対するプログラム例を見つけられる。
- 基本的に、GitHubからライブラリーとしてインクルードする。

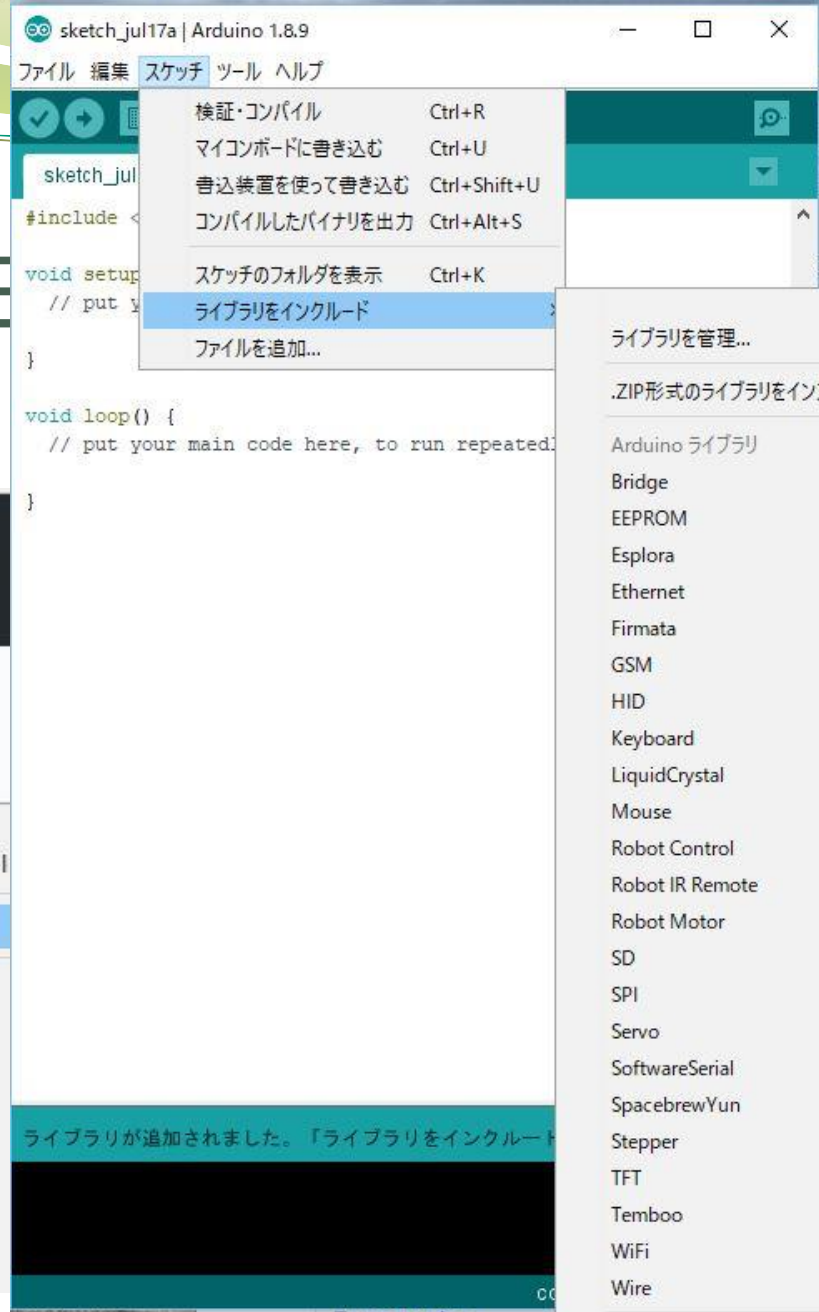
- 以下のURLからライブラリーをダウンロード

https://github.com/claws/BH1750

- 今回は、フォーラムからZIPファイルをダウンロード

GitHubライブラリーを使用





プログラミング

sketch_Light

```
/*
 * Super Sensing Forum
 * Arduino  Work Shop
 * 2019/7/18 Y.K
 */
#include <Wire.h>
#include <BH1750.h>

BH1750 lightMeter;

void setup() {

  Serial.begin(9600);
  Wire.begin();
  lightMeter.begin();
}

void loop() {
  float lux;

  lux = lightMeter.readLightLevel();
  Serial.print("Light = ");
  Serial.print(lux);
  Serial.println(" lx");
  delay(1000);
}
```

明るさLUXについて

- 照度の単位として、

1㎡当たり1ルーメンであれば、1ルクス

- 明るさの目安としては、

照度(ルクス)	明　る　さ　の　目　安	(ルクス)
１００，０００	・雪山・真夏の海岸	＞１００，０００
	・晴天昼太陽光	１００，０００
	・晴天午前10時太陽光	６５，０００
	・晴天午後３時太陽光	３５，０００
	・曇天昼太陽光	３２，０００
	・曇天午前10時太陽光	２５，０００
１０，０００	・曇天日出１時間後太陽光	２，０００
１，０００	・晴天日入１時間前太陽光	１，０００
	・パチンコ店内	１，０００
	・百貨店売場	５００～７００
	・蛍光灯照明事務所	４００～５００
	・日出入時	３００
	・30W蛍光灯２灯使用八畳間	３００
	・夜のアーケード	１５０～２００
１００	・街灯下	５０～１００
	・ライター@30cm	１５
１０	・ロウソク@20cm	１０～１５
	・市民薄明(太陽天頂距離９６度)	５
１	・月明り	０．５～１
	・航海薄明(太陽天頂距離１０２度)	０．０１
	・天文薄明(太陽天頂距離１０８度)	０．００１

まとめ

- ＬＥＤを用いてライティング（アナログ的にコントロール）
- 部品には、データーシートがあり、それを用いて設計する
- アナログ信号について　サンプリングと分解能
- 大抵の場合は、相反する仕様項目がある
- センサーのデジタルインターフェース　ＳＰＩ、ＵＡＲＴ、Ｉ２Ｃ
- それらを使ってプログラミング　ＧｉｔＨｕｂの活用

今後の活用

- 温度センサーで、いろいろなものを測定
  部屋の場所による違い、ペット等の居心地
- 光センサーで、いろいろなものを測定
  ベランダ（洗濯）の日光照射量把握
- 有線でなく、無線を使う
  無線モジュールとしては、ＵＡＲＴインターフェースが多い