

# 第5回 Arduino入門

## I2C通信編

プレゼン by いっちー

# 目次

1. I2Cとは
2. センサの多くがI2Cに対応
3. マイコンでのI2C通信例
4. アドレスとは
5. タイミングパラメータ
6. データ書込み
7. データ読み込み
8. Arduinoの設定
9. 加速度センサの設定
10. シリアルモニタの表示
11. 回路図
12. 結線図
13. 結線写真
14. WHO\_AM\_I
15. I2C読み込みプログラム
16. I2C読み込みスクリプト概要①
17. I2C読み込みスクリプト概要②
18. センサデータ読み込みプログラム
19. センサデータ読み込み概要①
20. センサデータ読み込み概要②
21. センサデータ読み込み概要③
22. Unityへ送る。

# 1. I2Cとは

アイスクウェアドシーと読むよ

フィリップス社で開発されたシリアルバス(シリアル通信の一種)である。

低速な周辺機器をマザーボードへ接続したり、組み込みシステム、携帯電話などで使われている。

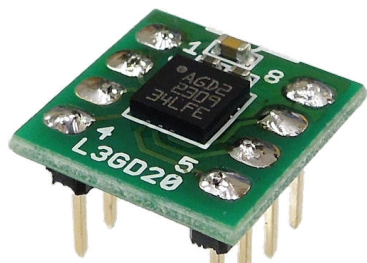
ウィキペディアより

いろんなセンサで使われているよ。  
他にもEEPROMや液晶、モータードライバなども制御できるよ。

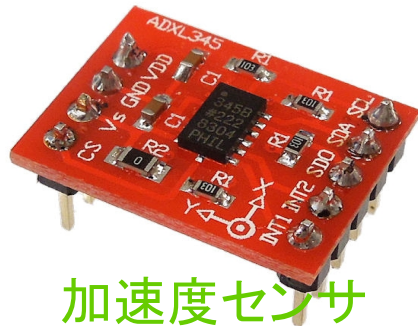
## <通信速度>

1. 標準モード 100kbps
2. ファーストモード 400kbps

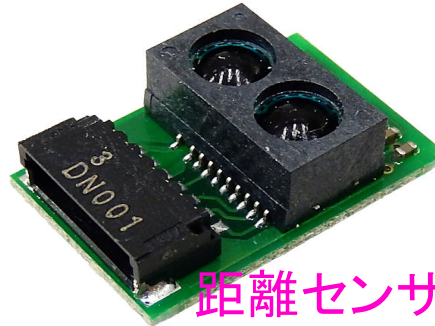
## 2. センサの多くがI2Cに対応



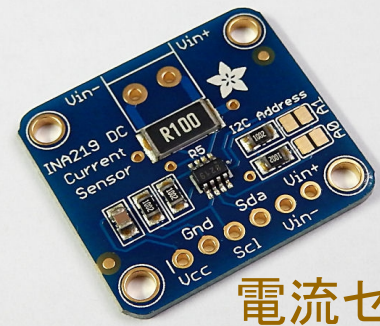
ジャイロセンサ



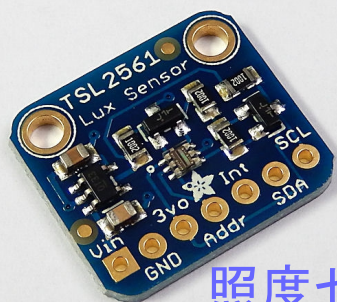
加速度センサ



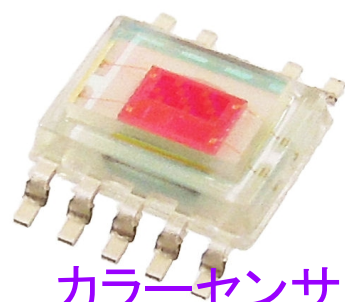
距離センサ



電流センサ



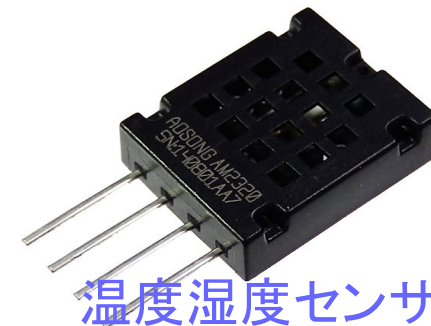
照度センサ



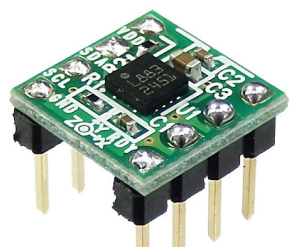
カラーセンサ



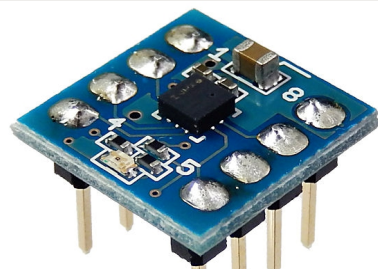
温度センサ



温度湿度センサ



地磁気センサ



気圧センサ



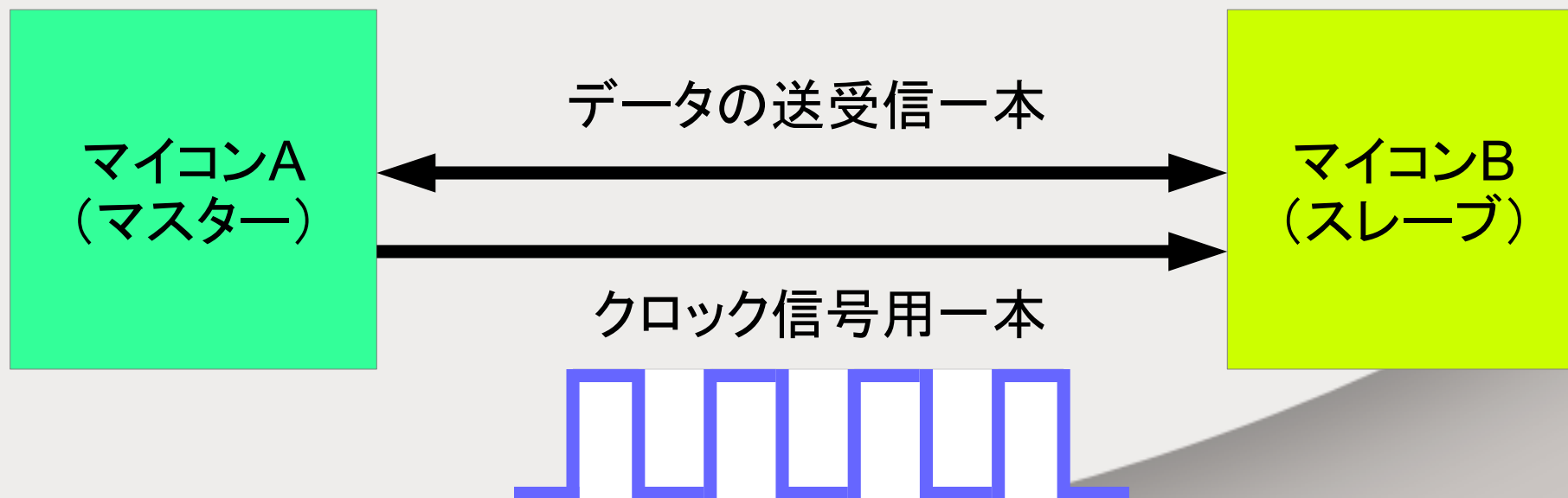
雷センサ

### 3. マイコンでのI2C通信例

- I2C通信をする場合はマスター（同期用クロック信号およびデータの書込み・読み込みの指示を行う側）とスレーブ（受け側。設定の必要なし）を決める。

同期用クロックを送信する。  
データの書込み・読み込みもこちらから指示を出す。

設定の必要なし



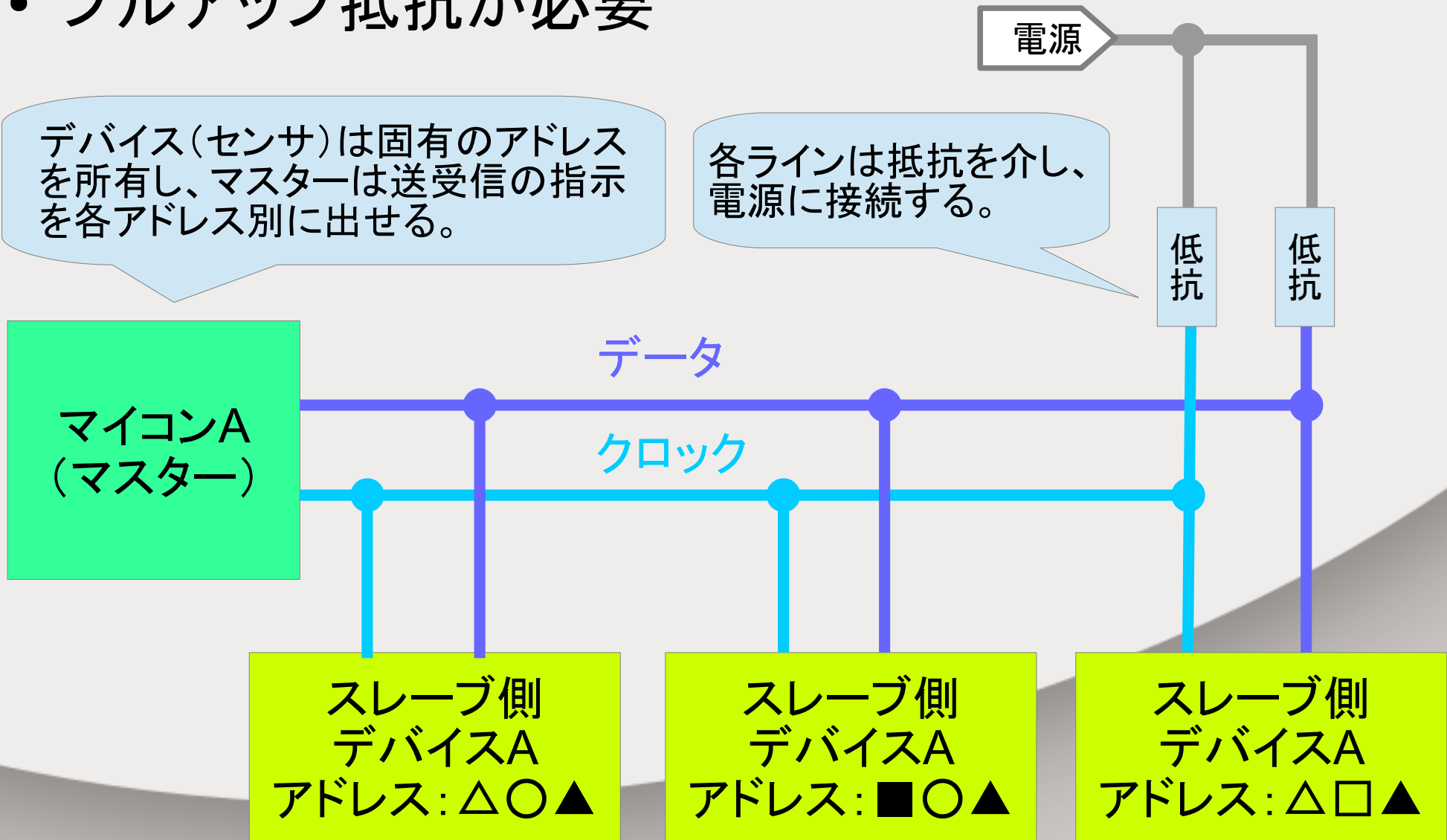
同期用のクロック信号をマスターから送信する。

# 4. 接続方法

- スレーブは複数つなぐことができる
- プルアップ抵抗が必要

デバイス(センサ)は固有のアドレスを所有し、マスターは送受信の指示を各アドレス別に出せる。

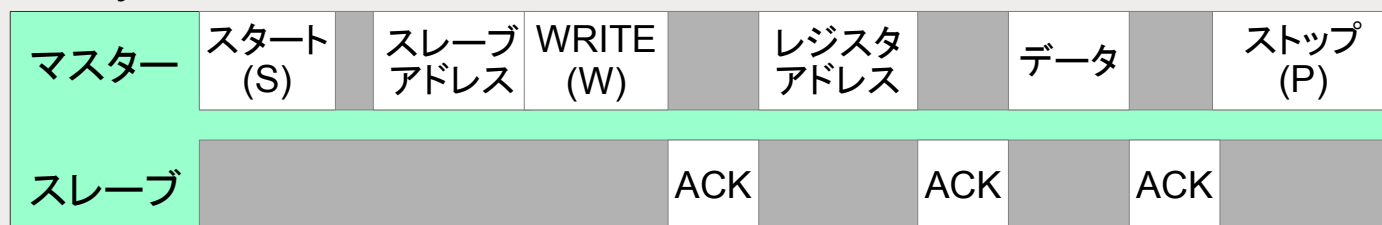
各ラインは抵抗を介し、電源に接続する。



# 5. タイミングパラメータ

- I2C通信の基本フォーマットは次の通り

< 1byteのデータ書込み >



< 複数のbyteのデータ書込み >



< 1byteのデータ読み込み >

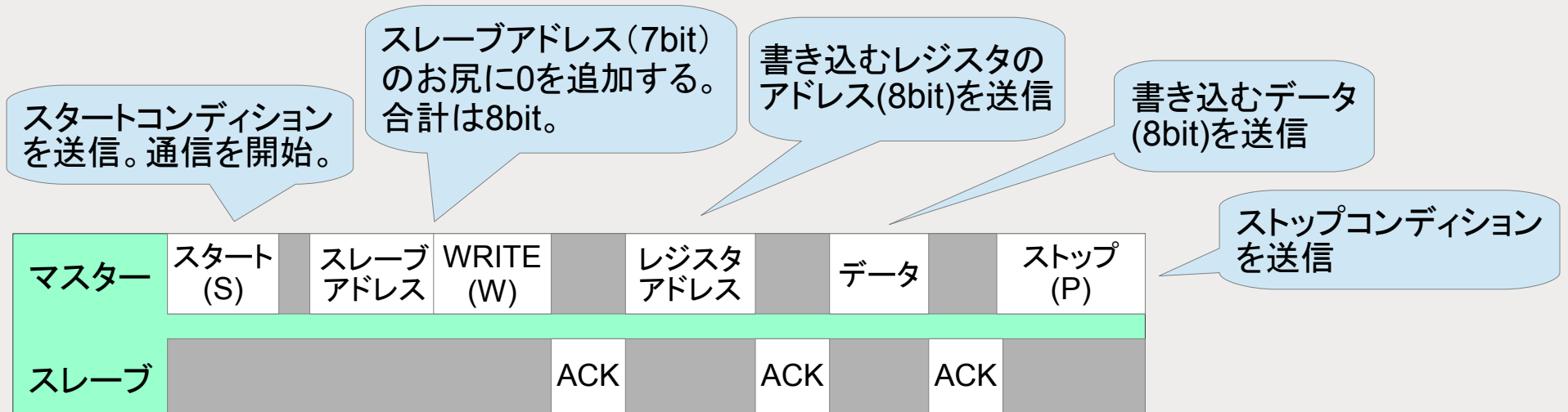


< 複数のbyteのデータ読み込み >



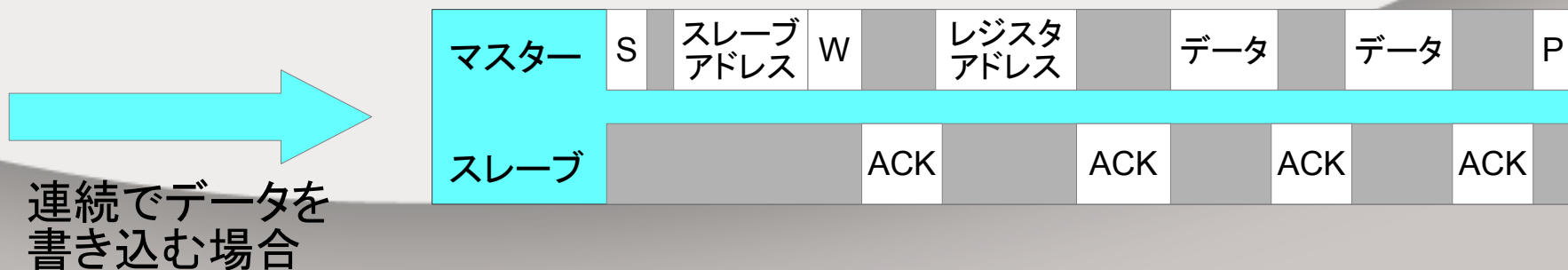
# 6. データ書込み

- スタートコンディション クロックがHighのときに、データがHigh→Lowに変化
- ストップコンディション クロックがHighのときに、データがLow→Highに変化
- WRITE 書込み要求。1bitの0。



8bitのデータ受信ごとにACK(1bitの0)を返す。

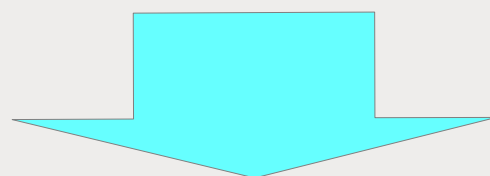
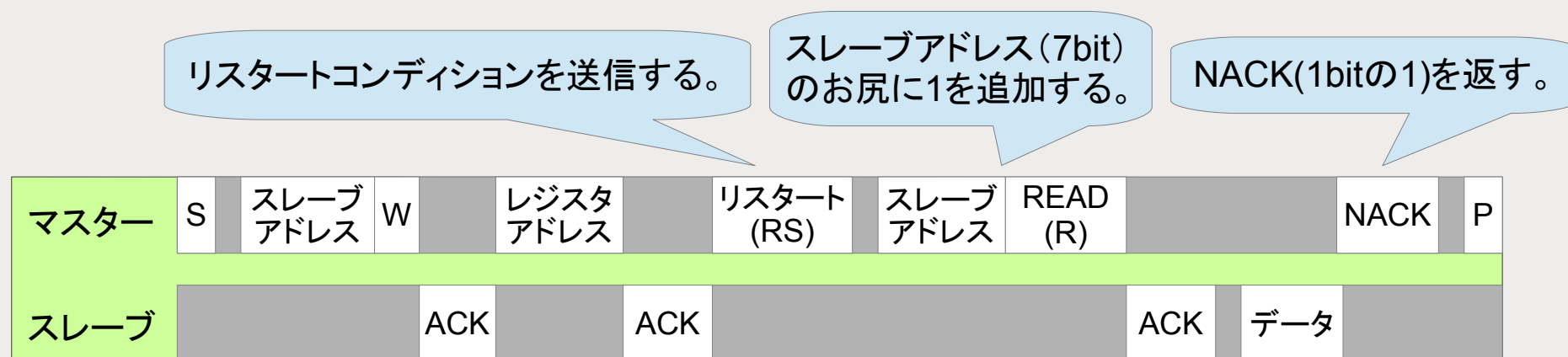
次のレジスタアドレスに書き込まれる





# 7. データ読み込み

- リスタートコンディション 一度、クロックをLowにしてから、データをHighにする。それからスタートコンディションを送る。
- READ 読み込み要求。1bitの1。



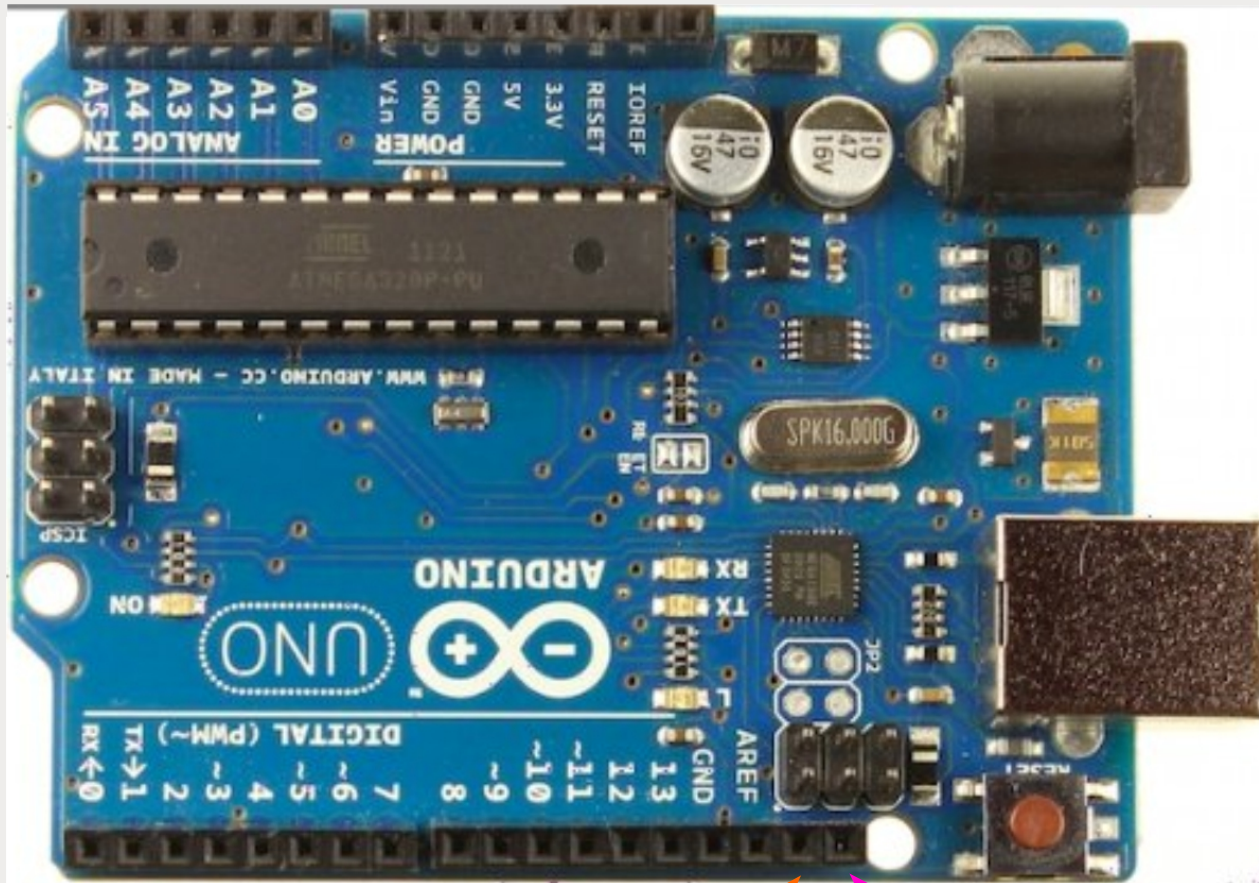
連続でデータを  
読み込む場合



次のレジスタアドレスのデータが読み込まれる

# 8. Arduinoの設定

- Arduino側の出力ピン

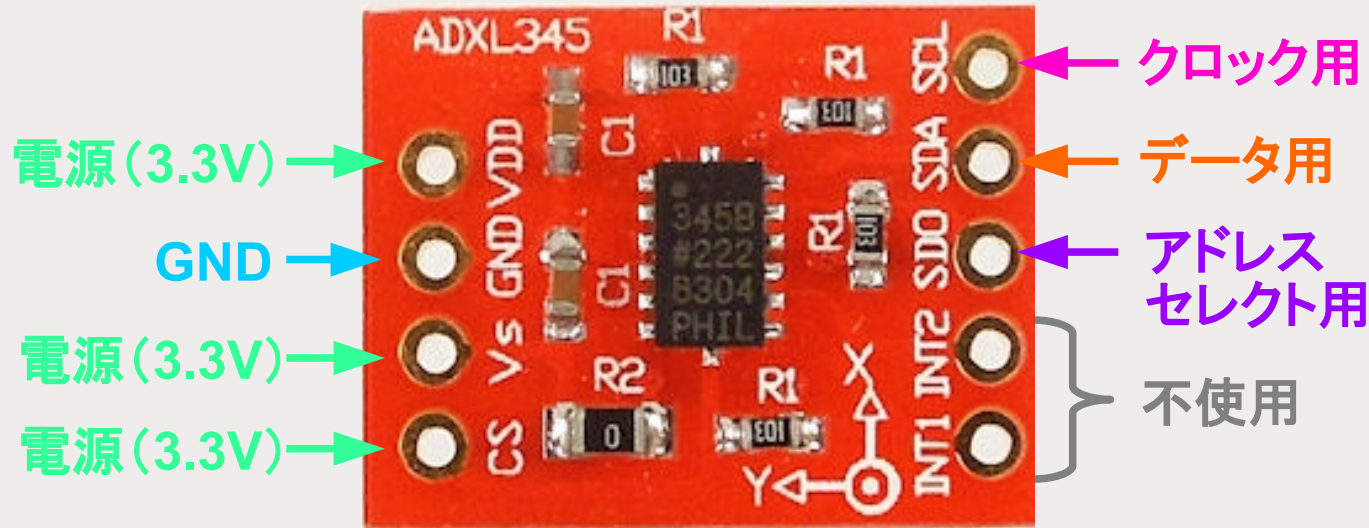


SDAピン  
データ出力用

SCLピン  
クロック用出力

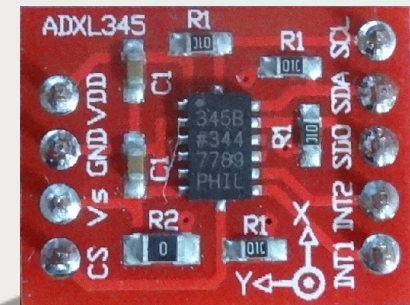
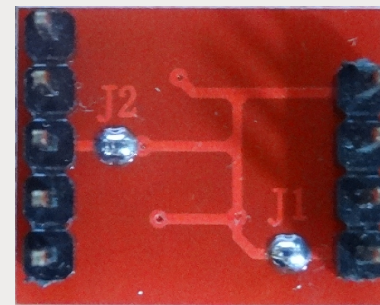
# 9. 加速度センサの設定

- ADXL345 (アナログデバイス) の出力ピン



SDOの接続先で  
センサのアドレス  
を変更可能  
・電源 → 0x1D  
・GND → 0x53

- ・プルアップ抵抗は基板に搭載。
- ・センサ仕様より  
電源電圧範囲 2.0~3.6V  
→ 3.3Vで使用する。

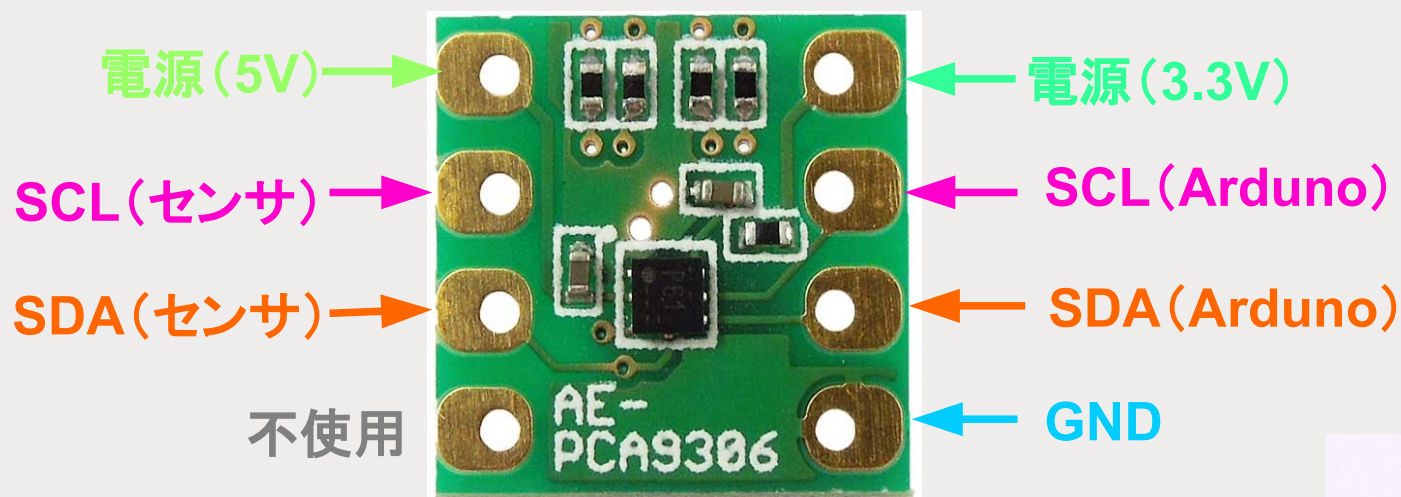


センサの使用する電圧値と  
Arduinoの信号の電圧値が合わないよ

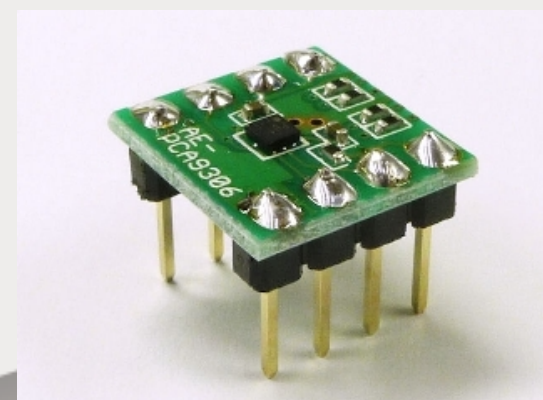
足部をはんだ付け  
はんだジャンパJ1・J2をはんだ付け  
→CSとSDOが電源に接続

# 10. 電圧変換モジュールの設定

- I2Cバス用双方向電圧レベル変換モジュール (PCA9306) の出力ピン



5Vの信号を3.3Vに  
3.3Vの信号を5Vに変換するよ

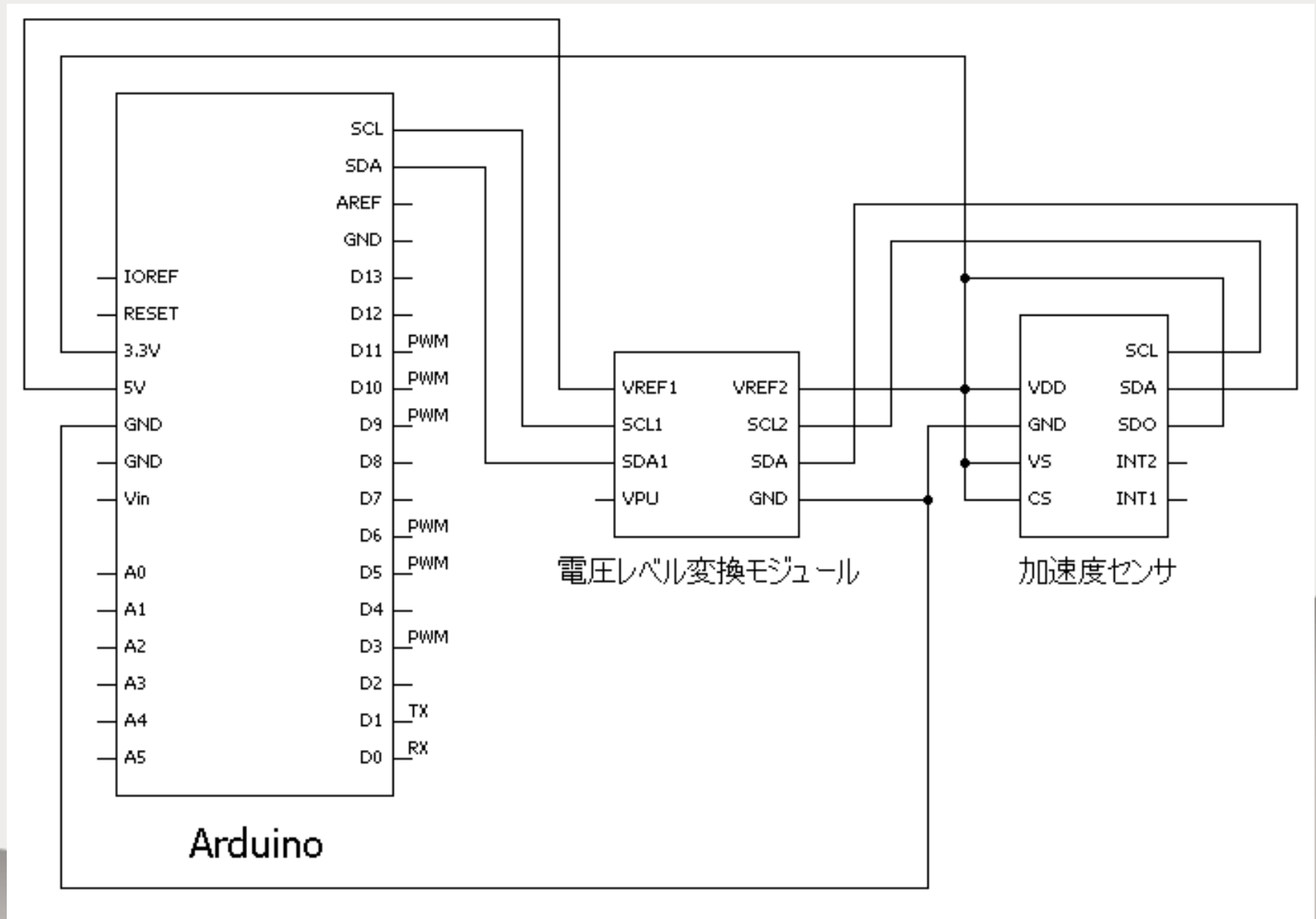


足部をはんだ付け



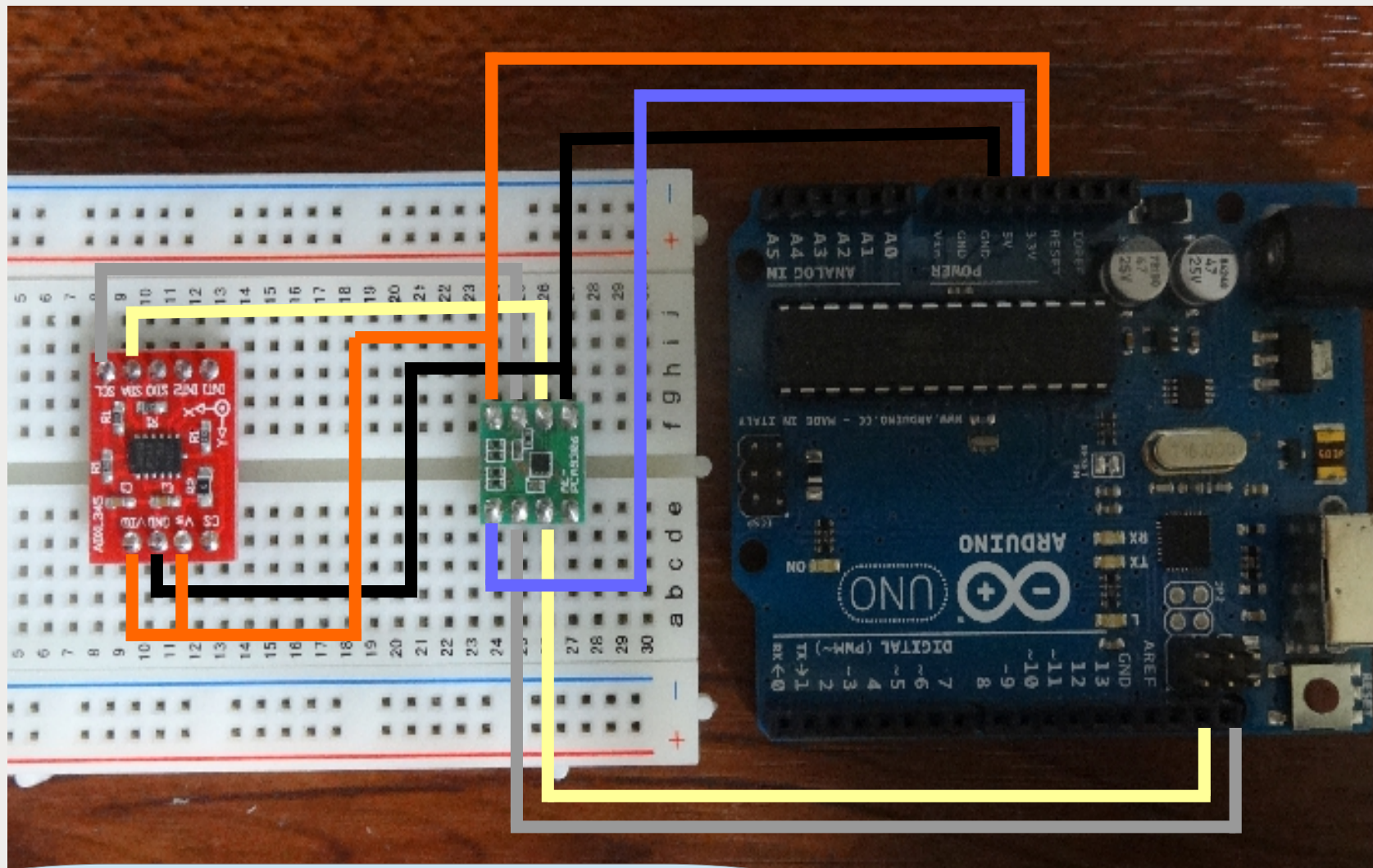
# 11. 回路図

- Arduino・加速度センサ・電圧レベル変換モジュールの接続



# 12. 結線図

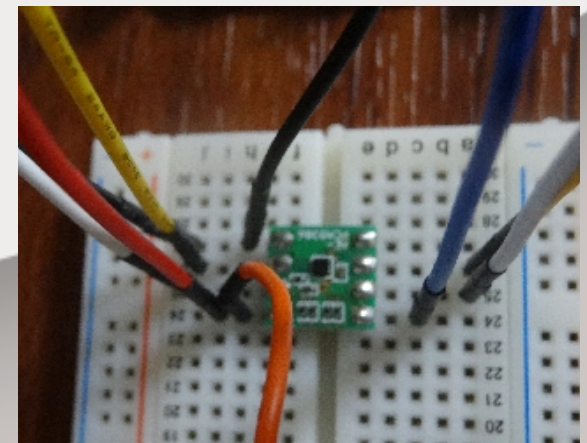
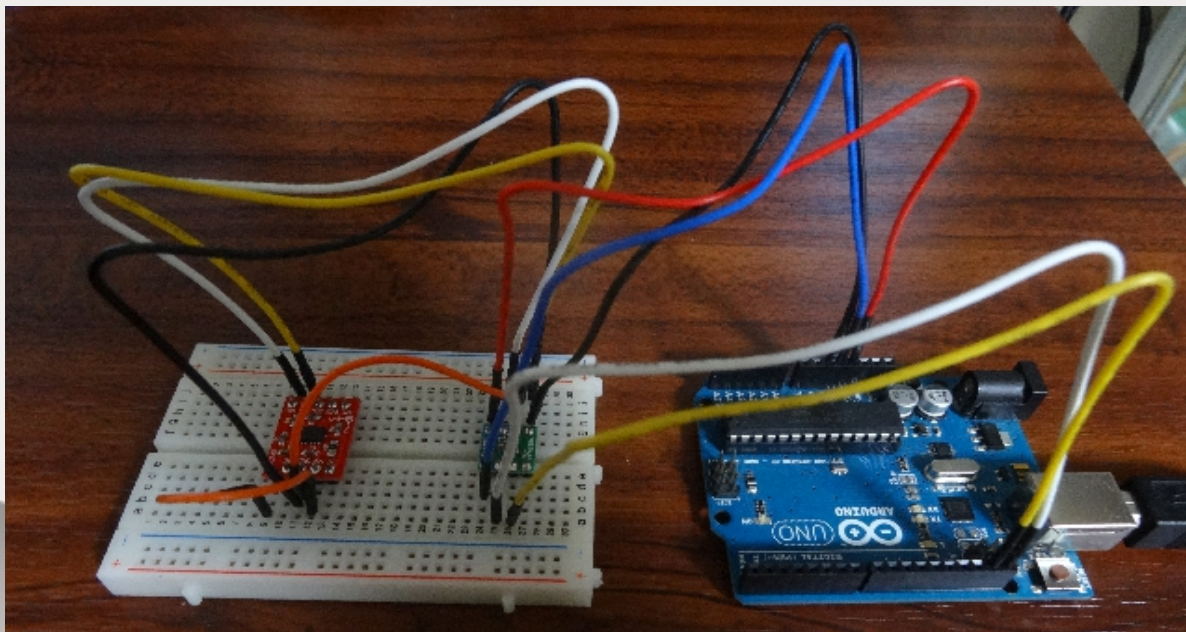
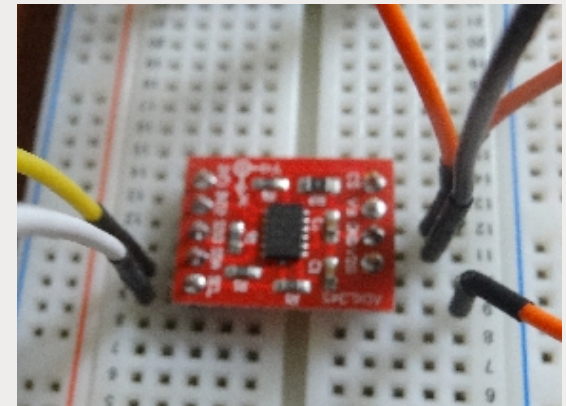
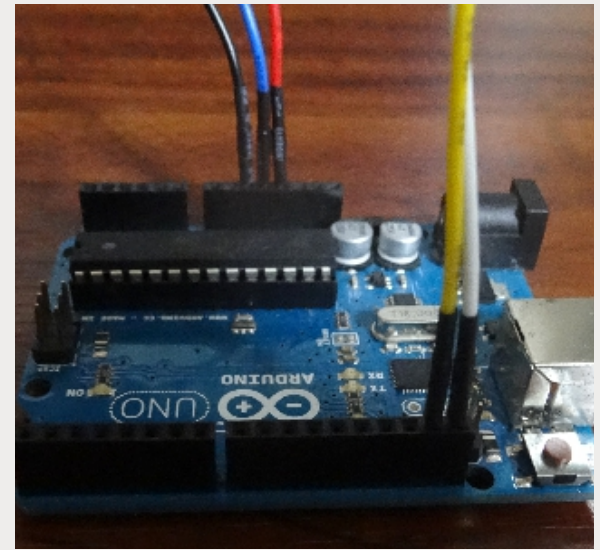
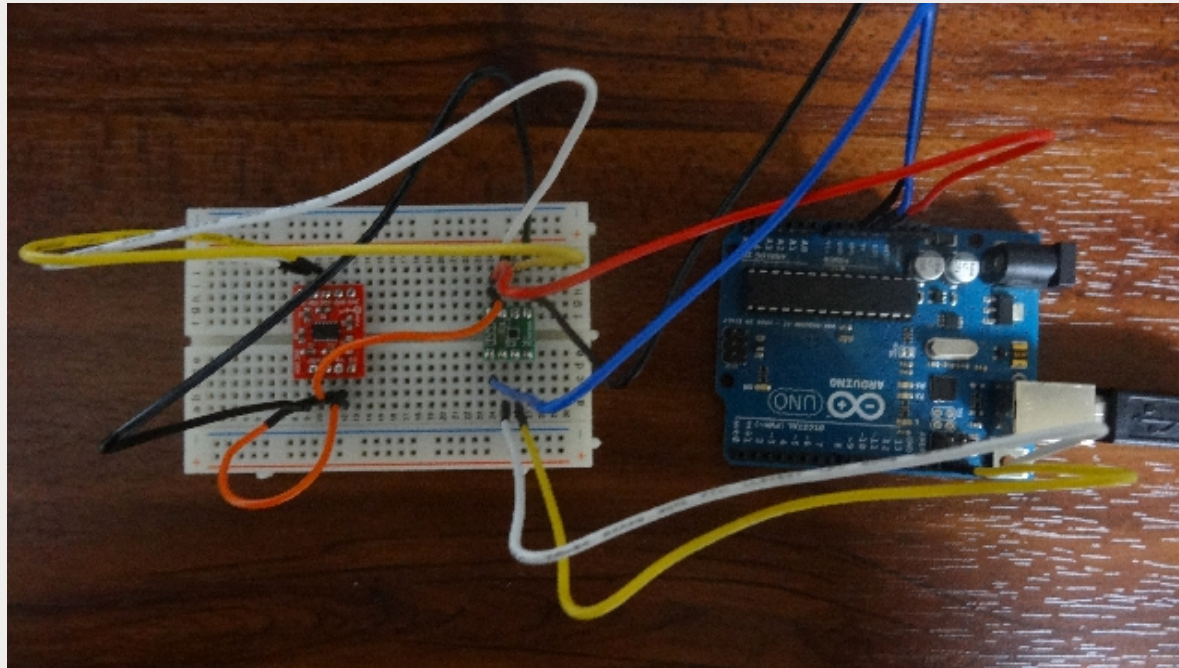
- 下記のように結線する。



センサのCS・SDOは基板裏面で結線済み



# 13. 結線写真



# 14. WHO\_AM\_I

- デバイスは固有のIDを持っている。

## レジスタ・マップ ※ADXL345データシートより抜粋

表 19. レジスタ・マップ

Address		Name	Type	Reset Value	Description
Hex	Dec				
<u>0x00</u>	0	DEVID	R	<u>11100101</u>	Device ID
0x01 to 0x1C	1 to 28	Reserved			Reserved; do not access
0x1D	29	THRESH_TAP	R/W	00000000	Tap threshold
0x1E	30	OFSX	R/W	00000000	X-axis offset
0x1F	31	OFSY	R/W	00000000	Y-axis offset

アドレス0x00のレジスタに固有アドレス「11100101」が格納されている。  
これを読み込んでみる。

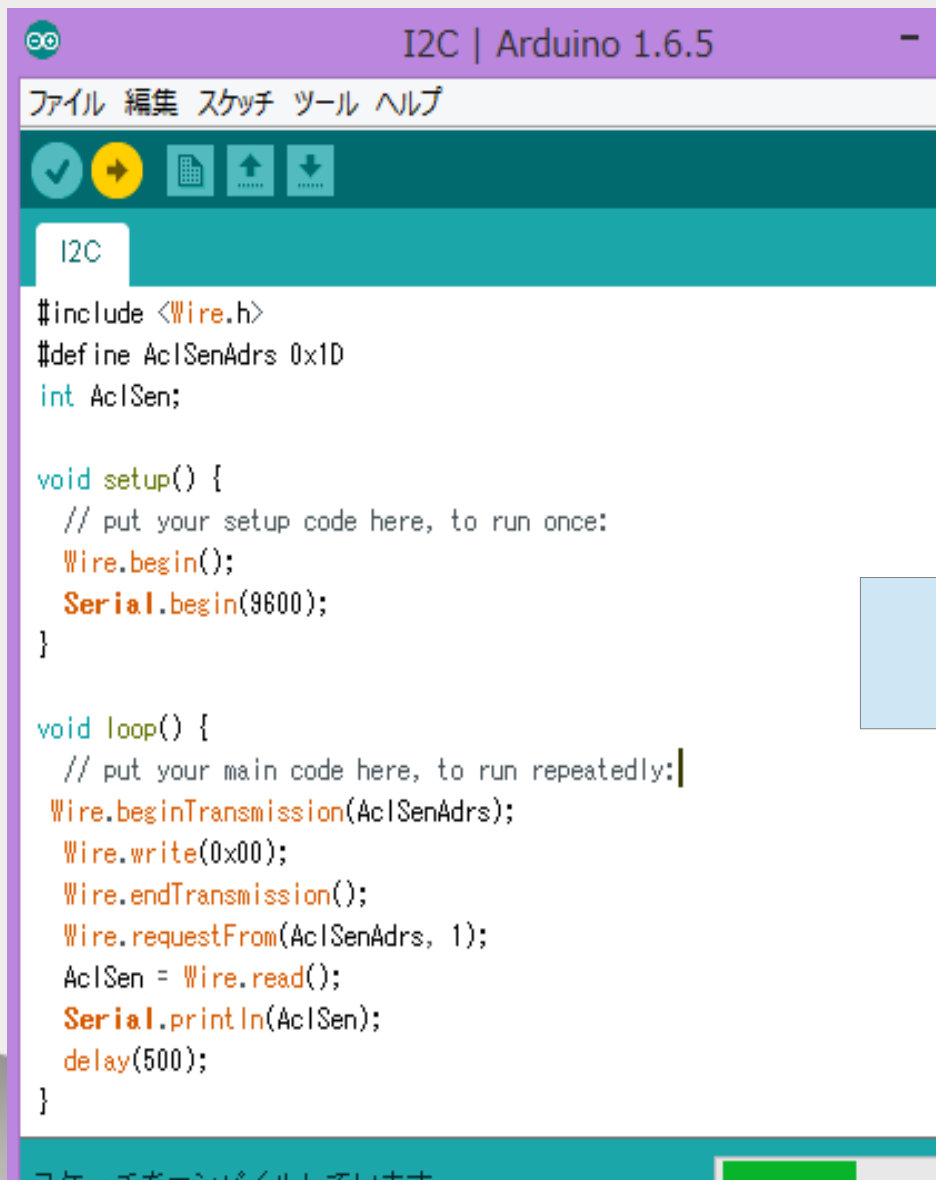
$$11100101(2進数) = 229(10進数)$$

まずはI2C通信ができているか確認するため、  
固有IDを確認する。



# 15. I2C読み込みプログラム

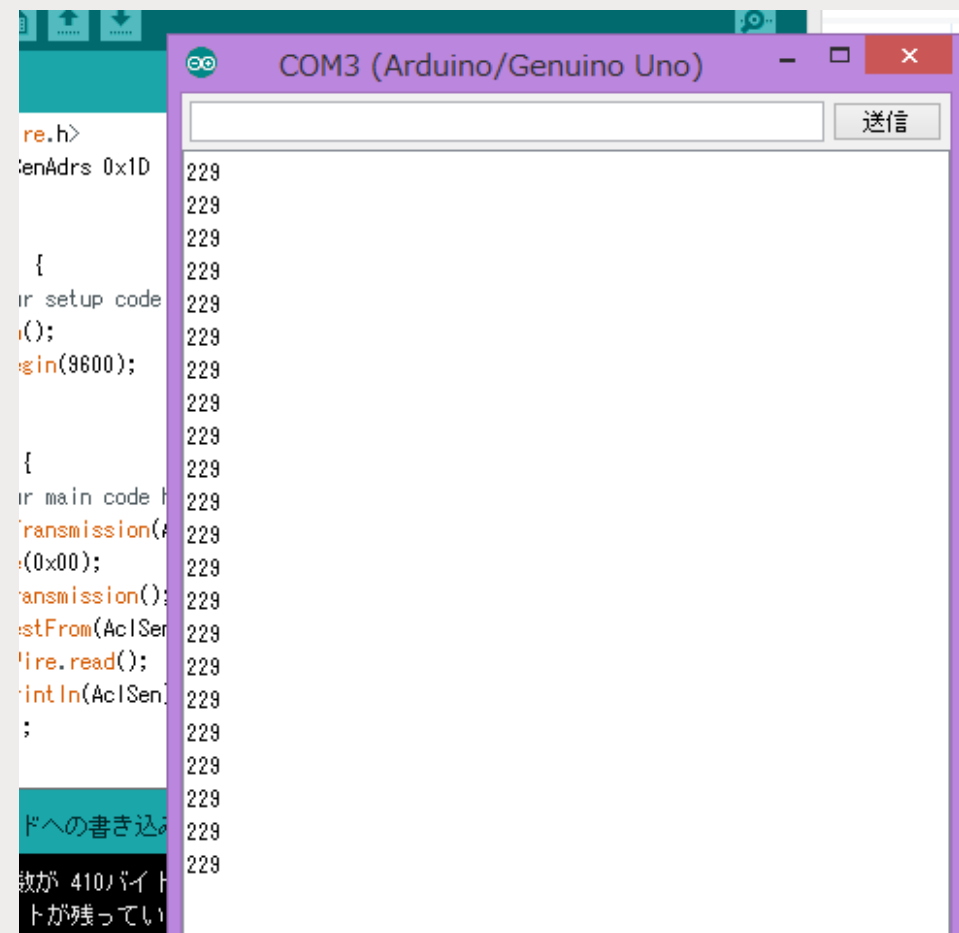
- プログラムをArduinoに書込む



```
I2C | Arduino 1.6.5
ファイル 編集 スケッチ ツール ヘルプ
I2C
#include <Wire.h>
#define AcISenAdrs 0x1D
int AcISen;

void setup() {
  // put your setup code here, to run once:
  Wire.begin();
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Wire.beginTransmission(AcISenAdrs);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(AcISenAdrs, 1);
  AcISen = Wire.read();
  Serial.println(AcISen);
  delay(500);
}
```



```
COM3 (Arduino/Genuino Uno)
送信
re.h>
SenAdrs 0x1D 229
229
229
{ 229
r setup code 229
(); 229
gin(9600); 229
229
229
{ 229
r main code h 229
ransmission(a 229
(0x00); 229
ransmission(); 229
stFrom(AcISen 229
ire.read(); 229
int ln(AcISen 229
; 229
229
229
ドへの書き込 229
229
効果が 410 バイト 229
トが残ってい
```

ツールからシリアルモニタを開く。  
固有ID「229」が表示される。

# 16. I2C読み込みスクリプト概要①

```
#include <Wire.h>
```

→I2C用ライブラリ「Wire.h」を追加する。

```
#define AclSenAdrs 0x1D
```

→AclSenAdrsを「0x1D」と定義する。(センサのアドレス)  
これ以降AclSenAdrsは0x1Dと書いたことと同じになる。

```
int AclSen
```

→int(整数型)の変数AclSenを宣言する。

```
Wire.begin()
```

→Wireライブラリの初期化。Arduino側をマスタと定義する。

```
Serial.begin(9600)
```

→マイコン側の通信速度を設定9600bps(ビット/秒)にする。

```
Serial.println(AclSen)
```

→パソコンに文字列(文字+改行)「AclSen」を送信する。

```
delay(500)
```

→500ms待つ。

# 17. I2C読み込みスクリプト概要②

Wire.beginTransmission(AclSenAdrs)

→I2C通信の開始。スレーブ側のアドレスの定義する。  
endTransmission()で送信を実行する。

Wire.write(0x00)

→読み込みを開始するレジスタアドレスの定義。

Wire.endTransmission()

→スレーブデバイスに対する送信を完了。

Wire.requestFrom(AclSenAdrs, 1)

→スレーブアドレスをもう一度定義。  
レジスタアドレス「0x00」から1アドレスのデータを読み込む。

AclSen = Wire.read()

→読込んだデータをAclSenに格納。

<参考: 1byteのデータ読み込みタイミングパラメーター>



# 18. センサデータ読み込みプログラム

- プログラムをArduinoに書込む

```
I2C_AclSen | Arduino 1.6.5
ファイル 編集 スケッチ ツール ヘルプ
I2C_AclSen
#include <Wire.h>
int AcISenL, AcISenH, AcISen;
#define AcISenAdrs 0x1D

void setup() {
  Wire.begin();
  Serial.begin(9600);

  Wire.beginTransmission(AcISenAdrs);
  Wire.write(0x2D);
  Wire.write(0x08);
  Wire.endTransmission();
}

void loop() {
  Wire.beginTransmission(AcISenAdrs);
  Wire.write(0x32);
  Wire.endTransmission();
  Wire.requestFrom(AcISenAdrs, 2);
  while(Wire.available()){
    AcISenL = Wire.read();
    AcISenH = Wire.read();
  }
  AcISen = 0;
  AcISen = AcISenL + AcISenH * 0x100;
  Serial.println(AcISen);
  delay(500);
}
```



```
COM3 (Arduino/Genuino Uno)
送信
0
-1
0
-1
-1
-1
121
177
190
216
183
81
-29
-64
-89
-77
-76
-61
-27
47
78
79
87
書き込みが完了
```

加速度センサを動かすと、値が変化する。  
±2Gを±256として表示する。

# 19. センサデータ読み込み概要①

Wire.beginTransmission(AclSenAdrs)

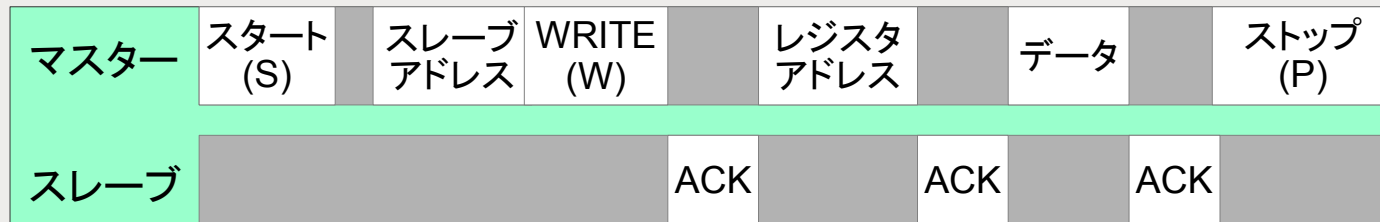
Wire.write(0x2D)

Wire.write(0x08)

Wire.endTransmission()

→書込みレジスタアドレス「0x2D」を定義し、  
データ「0x08(00001000)」の書き込みを実行する。  
Measureモードになり、測定が開始される。

<参考: 1byteのデータ書込みパラメーター>



レジスタ 0x2D—POWER\_CTL (読出し/書込み) ※ADXL345データシートより抜粋

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Measureビットを1にすると測定モードになる。0だとスタンバイモード。

初期値はすべて0となっている。

またレジスタ0x31を変更すると、測定レンジを±2g、±4g、±8g、±16g、変更できる。

初期値では±2gとなっている。

# 20. センサデータ読み込み概要②

Wire.requestFrom(AclSenAdrs,2)

→レジスタアドレス「0x32」から2アドレスのデータを読み込む。

Wire.available

→read()で読み取ることができるバイト数を返す。

このプログラムの場合、最初は2で、2回データを読み込むと0になる。

AclSenL = Wire.read()

AclSenH = Wire.read()

→レジスタアドレス「0x32」のデータをAclSenLに格納。

レジスタアドレス「0x33」のデータをAclSenHに格納。

X方向の加速度データは全10ビットある。「0x32」には下位の8ビット、「0x33」には上位2ビットが分けられて入っている。

<参考:複数のbyteのデータ読み込みパラメーター>

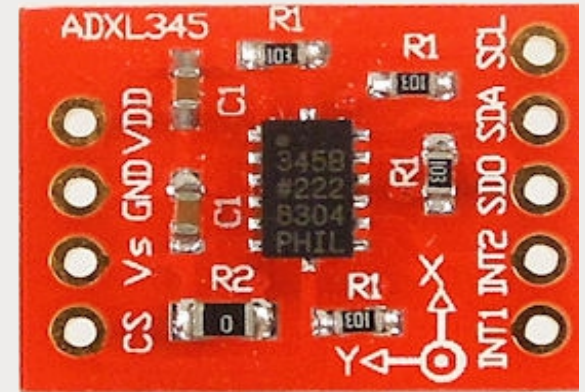


# 21. センサデータ読み取り概要③

AclSen = 0;

AclSen = AclSenL + AclSenH \* 0x100

→ 上位2ビットと下位8ビットをつなぎ合わせ、  
10ビットのデータを作成する。



↑ がX方向の正となる。

※ADXL345データシートより抜粋

OUTPUT RESOLUTION	Each axis		
All g Ranges	10-bit resolution	10	Bits
±2 g Range	Full resolution	10	Bits
±4 g Range	Full resolution	11	Bits
±8 g Range	Full resolution	12	Bits
±16 g Range	Full resolution	13	Bits

±2g設定では全データは10ビットになる。

※ADXL345データシートより抜粋

0x31	49	DATA_FORMAT	R/W	00000000	Data format control
0x32	50	DATA_X0	R	00000000	X-Axis Data 0
0x33	51	DATA_X1	R	00000000	X-Axis Data 1
0x34	52	DATA_Y0	R	00000000	Y-Axis Data 0

「0x32」にはX方向下位の8ビット、「0x33」には上位2ビットが分けられて入っている。

# 22. Unityへ送る。

## • Unityへ送る。

```
I2C_AclSen_PC
#include <Wire.h>
int AclSenL, AclSenH, AclSen;
#define AclSenAdrs 0x1D

void setup() {
  Wire.begin();
  Serial.begin(9600);

  Wire.beginTransmission(AclSenAdrs);
  Wire.write(0x2D);
  Wire.write(0x08);
  Wire.endTransmission();
}

void loop() {
  Wire.beginTransmission(AclSenAdrs);
  Wire.write(0x32);
  Wire.endTransmission();
  Wire.requestFrom(AclSenAdrs, 2);
  while(Wire.available()){
    AclSenL = Wire.read();
    AclSenH = Wire.read();
  }
  AclSen = 0;
  AclSen = AclSenL + AclSenH * 0x100;
  AclSen = AclSen/4;
  AclSen = AclSen + 127; //64-191
  // Serial.println(AclSen);
  Serial.write(AclSen);
  delay(500);
}
```

### <概要>

AclSen = AclSen/4;

→Unityにデータを送るためにデータを10ビットから8ビットに減らす。

AclSen = AclSen + 127;

→データが±になっているので正の整数データに変換する。  
-64~+64が0~+127になる。

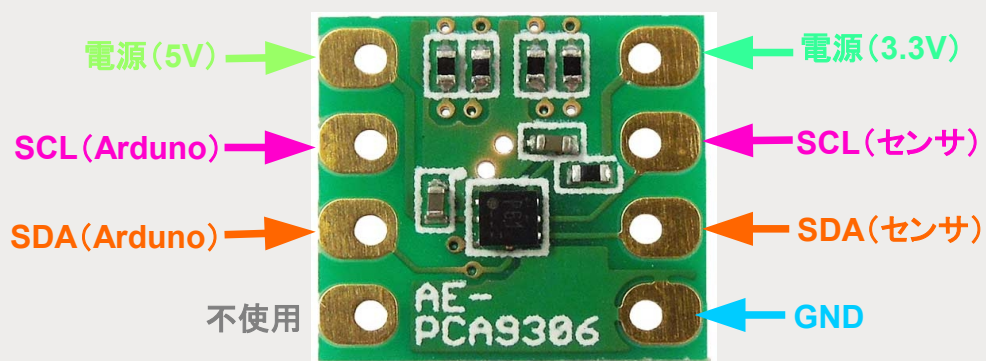
第三回と同じようにすれば、Unityとデータを送受信できるよ。

※第三回のUnityプログラムで受け取れるのは8ビットの正の整数データ

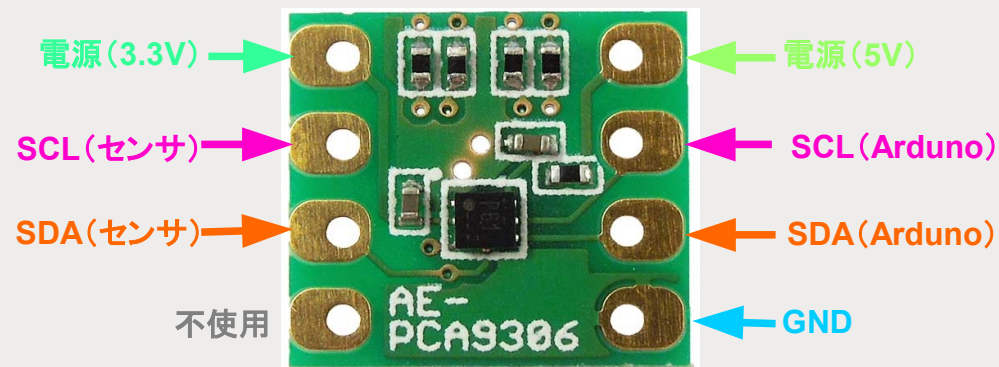


# 訂正

誤



正



つなぎ方が逆でした・・・。  
資料内のつなぎ方でも動くみたいだけど、  
右側は高いほうの電圧をつなぐのが正しいらしい。



みなさん、長い間お疲れさまでした。