# 令和　年度　スーパーサイエンス部学習会（Arduino）
# ～電子制御プログラミングを学ぶ～

月　日（　）
岡崎高校物理教室（北校舎 3F）

**【本学習会の目的・目標】**

目的：　電子制御プログラミングの基本的な考え方を理解する。
目標：　各種センサを利用した測定器を製作し，プログラミングによって実用化する。
　　製作した測定器を用いて簡単な実験を行い，データを分析する。

**【学習内容】**

□ 回路の作製
□ 基本の L チカ
□ 9 軸慣性計測ユニット（加速度・角速度・磁束密度センサ）によるデータ取得
□ 温度・湿度・気圧センサによるデータ取得
□ OLED（有機 EL ディスプレイ）への文字の表示
□ microSD カードへの保存
□ 簡易実験（実験計画の立案, 測定器のスケッチのカスタマイズ, 測定, データ分析, 発表）

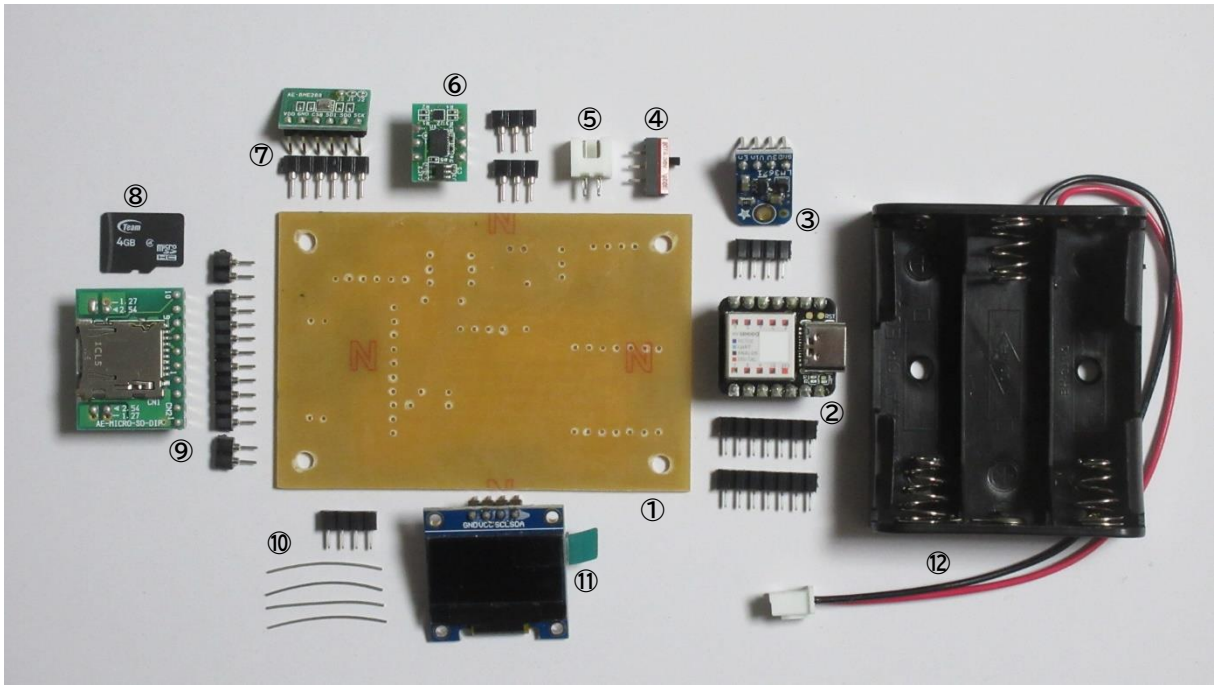
**【Arduino についての基礎知識】**

(1) Arduino（アルドゥイーノ／アルデュイーノ）はマイコン（マイクロコントローラー）の一種で，プログラミングした通りに動作する機能を持っている。
(2) プログラミング言語は C++に似ている。
(3) プログラミングのためのアプリ（IDE／統合開発環境）は無料である。
(4) プログラム（コード）のことを「スケッチ」と呼ぶ。
(5) ソフトが“オープン”である。他の人が開発・公開したプログラムを自由に使うことができる。これらを「ライブラリ」と呼ぶ。
(6) ハードも“オープン”で，互換機が多く存在する。
(7) 令和 4 年 8 月現在，純正品で最も基本的な「Arduino UNO R3」は 3,940 円である。今回使用する互換機の Seeeduino XIAO は 850 円である。
(8) 電子制御プログラミングのためのシステムとしては，Arduino の他に RaspberryPi（ラズベリーパイ／ラズパイ）が有名である。一般には RaspberryPi の方が高機能とされるが, Arduino は RaspberryPi と比べて安価であり, 起動が速く, また電力消費が少ない。

## 1. 回路の作製

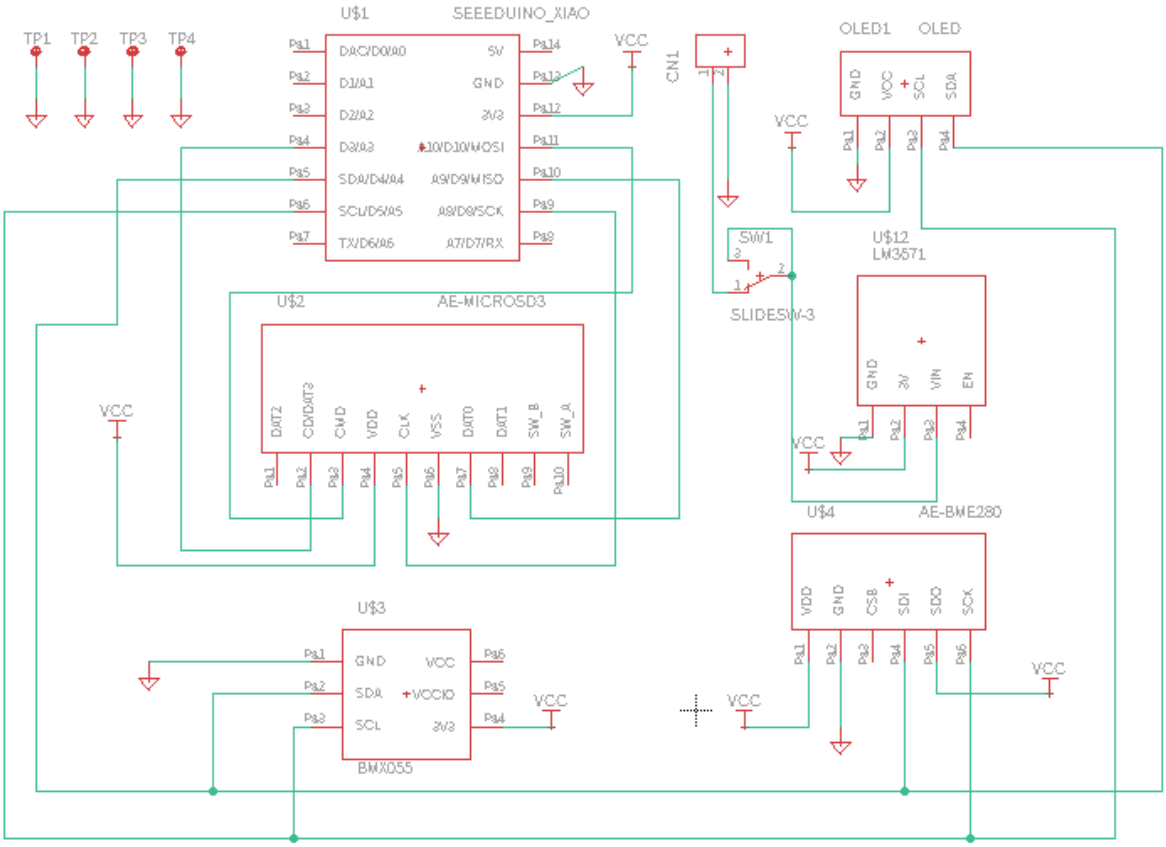
今回は純正の Arduino ではなく，互換機の Seeeduino XIAO（シードゥイーノシャオ）を用いて回路の作製・プログラミングを行う。Seeesuino XIAO は数ある Arduino 互換機の中でも安価かつコンパクトで使いやすいのが特徴である。

**【部品一覧】**




① プリント基板
② Seeeduino XIAO （+ シングルピンソケット 7P×2）
③ 3.3V DC-DC コンバータ LM3671 （+ シングルピンソケット 4P×1）
④ スライドスイッチ
⑤ XH コネクタ
⑥ 9 軸慣性計測ユニット AE-BMX055 （+ 丸ピンソケット 3P×2）
⑦ 温度･湿度･気圧センサモジュール AE-BME280 （+ 丸ピンソケット 6P×1）
⑧ microSD カード
⑨ microSD カードスロット （+ 丸ピンソケット 10P×1, 2P×2）
⑩ スズメッキ線（4 本）
⑪ OLED（有機 EL ディスプレイ） （+ シングルピンソケット 4P×1）
⑫ 電池ボックス　単 3×3 本用　（XH コネクタハウジング取付済）

**【回路図】**



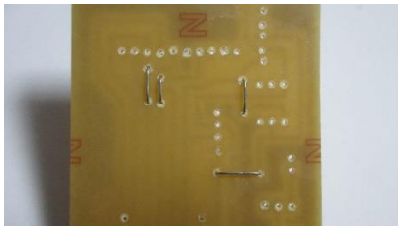
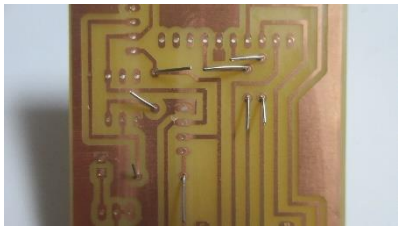
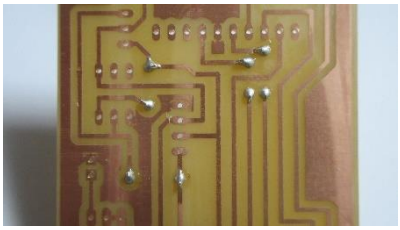

本学習会のメインはプログラミングなので，ハードウェアの詳細は省略する。回路図の意味を全て理解する必要はないが，一応，参考までに概略を示しておく。

- Seeeduino XIAO，センサモジュール，OLED，microSD スロットは 3.3V の電圧で動作する。（ちょうど 3.3V の電池がないため）専用の IC（LM3671）で乾電池 3 本の 4.5V を 3.3V に変換している。なお，PC と USB 接続しているときには，USB の 5V を Seeeduino XIAO が 3.3V に変換しているので LM3671 は必要ない。
- Seeeduino XIAO とセンサモジュール／OLED は「$I^2C$」という規格でデータの通信をする。それぞれの IC には，$I^2C$ 通信をするためのピンが 2 本ずつ決められており（SDA, SCL）,それらを全部並列接続してよい。IC 内で別々のアドレスを割り当てているため，混線せず識別することができる。このため，複雑な機能を持つ回路でありながら，配線はシンプルである。
- Seeeduino XIAO と microSD カード（スロット）は「SPI」という規格でデータ通信をする。これは上述の $I^2C$ とは別なので，配線の系統もセンサモジュールとは別になっている。

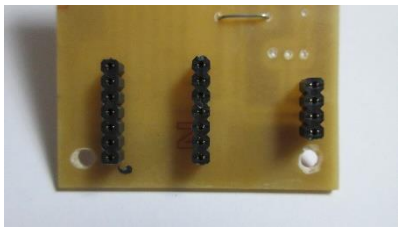
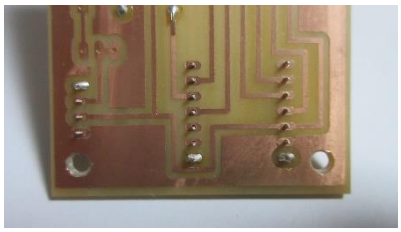
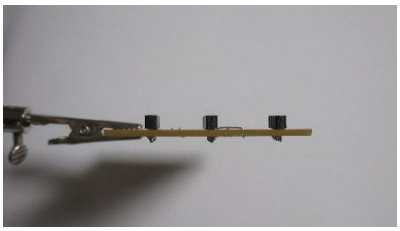
**【はんだ付けの実際】**

はんだ付けをするときに注意するべき点は以下の通りである。

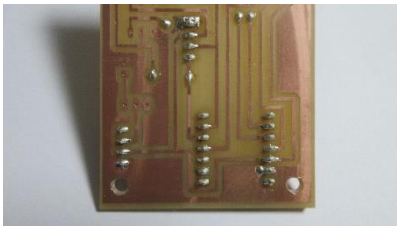
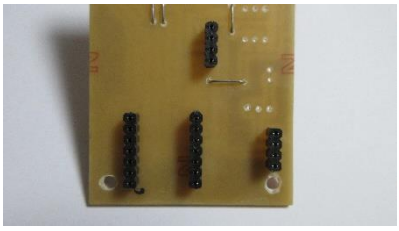
- **火傷しないこと。**
- はんだを付けすぎないこと。
- 低い部品から取り付けていくこと。
- 極性がある部品は，取り付ける向きに注意すること。
- こて先を基板の銅箔と部品のリード線の両方に当ててから，はんだを付けて融かす。はんだが流れるように融けて行き渡ったら，こて先を離す。

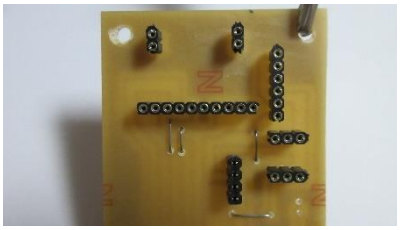
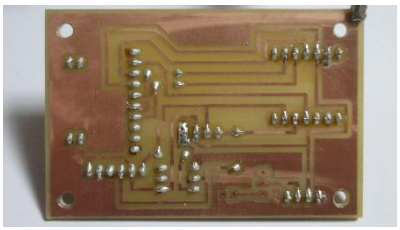
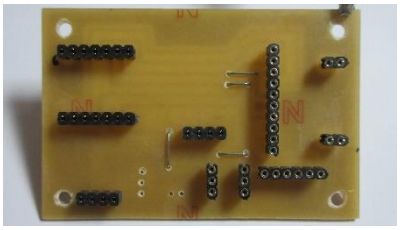
(1) スズメッキ線を写真のようにまげて取り付け，はんだ付けし，余分な線をニッパで切る。切るときに基板の銅線を切らないように注意する。

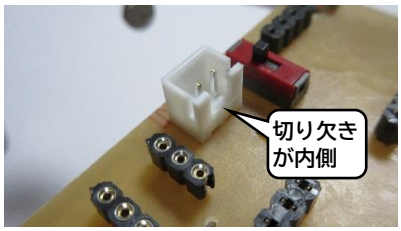
(2) シングルピンソケットを差し込み，**基板に垂直になるように**端の一箇所だけはんだ付けする。横から見て斜めになっていたら，はんだを融かして付け直す。

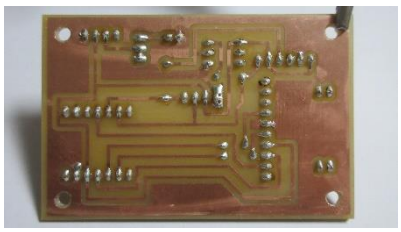
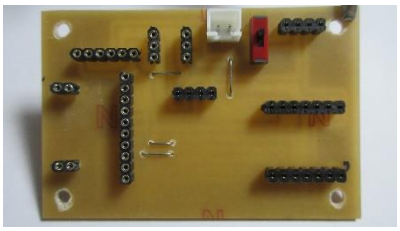



斜めでないことを確認したら残りもはんだ付けする。シングルピンソケットは４箇所である。

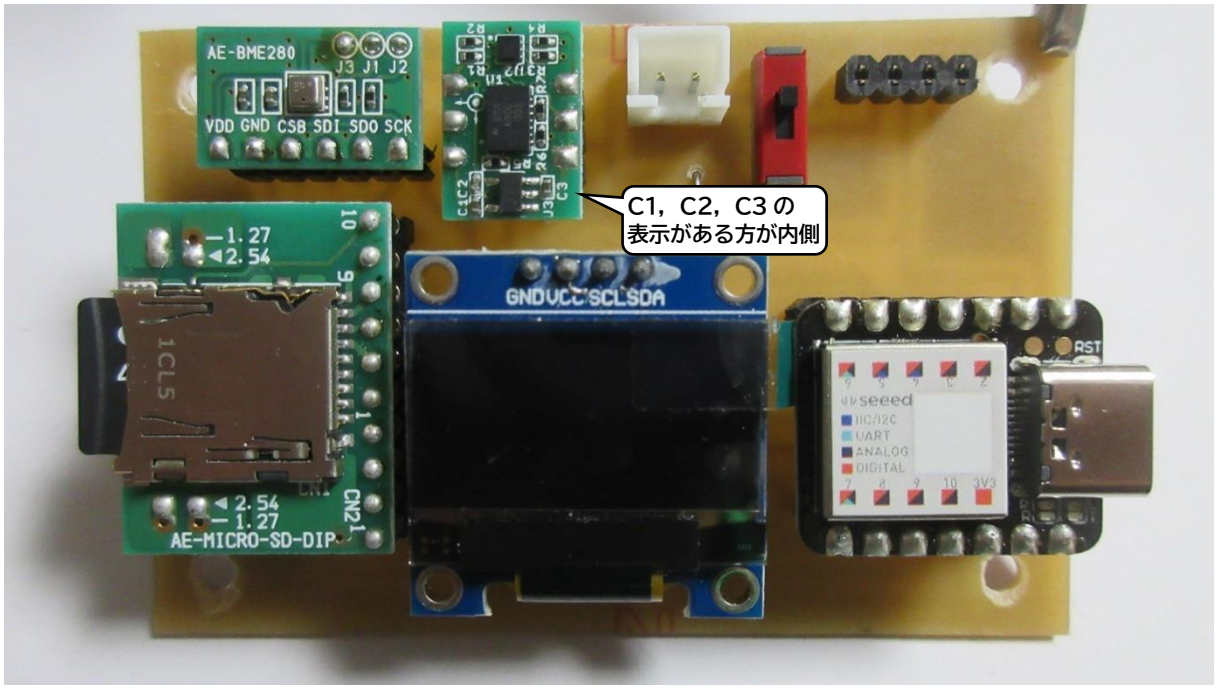




(3) 丸ピンソケットも，シングルピンソケットと同様に斜めになっていないことを確認しながらはんだ付けする。丸ピンソケットは６箇所である。



(4) XH コネクタとスライドスイッチをはんだ付けする。**XH コネクタは切り欠きのある方が内側になるようにする。**




(5) Seeeduino XIAO，OLED，microSD カードスロット，AE-BMX055（9 軸慣性計測ユニット），AE-BME280（温度・湿度・気圧センサモジュール）を，向きに注意しながらソケットに差し込む。また，microSD カードをスロットに差し込んでおく。**特に AE-BMX055 は向きがやや分かりにくいので，写真をよく見て正しく差し込むこと。この時点では，まだ LM3671（3.3V DC-DC コンバータ）をソケットに差してはならない。電池も接続してはならない。**

## 2. 基本のＬチカ

このセクションは，プログラミングに慣れるためのトレーニングである。Ｌチカとは「LED の点滅」のことである。Seeeduino XIAO には内蔵 LED が搭載されているので，それを利用してみよう。

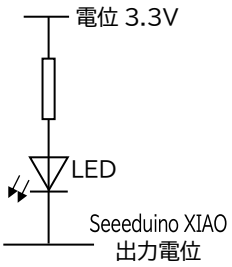
(1) Seeeduino XIAO と PC を USB ケーブルで接続し，Arduino IDE を起動する。
メニュー［ツール］→ ボード：”Seeeduino XIAO”
シリアルポートを選択


(2) スケッチを作成する。
ファイル名を設定して次のように記述する。コメント文（各行の//以降の文）は入力しなくてよい。ファイル名は自動で「sketch_aug10a」のように設定されるが，分かりやすいファイル名（「test_blink_01」「Lchika_01」など）に変更しておくとよい。

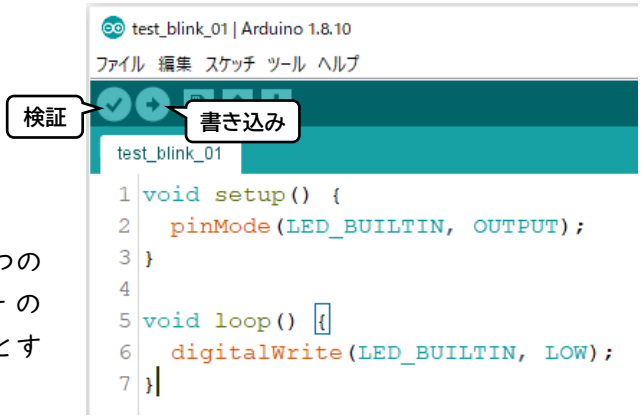
```
void setup() {                          // 以下の処理を１回だけ行う（初期設定）
    pinMode(LED_BUILTIN, OUTPUT);       // 内蔵 LED をデジタル出力に設定
}

void loop() {                           // 以下の処理を繰り返す
    digitalWrite(LED_BUILTIN, LOW);     // 内蔵 LED を点灯
}
```

アプリを起動，または新規ファイルを作成した段階で，void setup() { }と void loop() { }は既に入力されているので，入力は楽であろう。それぞれ初期設定を行う部分と繰り返し処理を行う部分である。Arduino ではこの２つは必須であり，決まり文句のようなものだと思えばよい。

Arduino などのマイコンは基本的にデジタルで動作する。Seeeduino XIAO は「0」が電位 0V，「1」が電位 3.3V に対応している。上記スケッチの「LOW」は出力電位を 0V にするという意味である。内蔵 LED は予め他方が 3.3V 側に接続されているため，「LOW」を出力することで LED に電位差（電圧）が加えられ点灯する，というわけである。ちなみに「LOW」ではなく「HIGH」とすると 3.3V が出力され，LED の電位差は 0V となり消灯する。




(3) スケッチを書き込む。
検証（コンパイル）→書き込み
橙色の LED が点灯すれば成功である。




(4) 他の内蔵 LED を点灯させてみる。
Seeeduino XIAO にはこの橙色の LED の他に２つの青色 LED が搭載されている。上記スケッチの「LED_BUILTIN」を「PIN_LED2」「PIN_LED3」とするとこれらを点灯させることができる。
橙色の LED はスケッチを書き込むときにも点灯するので，制御できていることが明確に分かるようにするために青色に変更しておこう。

(5) 内蔵 LED を点滅させてみる。

このセクションでの目標は「L チカ」，すなわち LED の点滅であった。そこでスケッチを次のように変えてみよう。0 から作成し直す必要はなく，先に作成したスケッチを変更すればよい。（「保存」ではなく）「**名前を付けて保存**」を選び，ファイル名を「test_blink_02」「Lchika_02」などに付け直せば元のスケッチも残しておける。**今後，作成したスケッチの一部を再利用することが多いので，元のスケッチを残しておくとよい。**変更した箇所は**太字**で示してある。

なお，今後スケッチが徐々に長くなっていくが，行頭の位置合わせには Space キーではなく Tab キーを使うと便利である。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
}

void loop() {
    digitalWrite(PIN_LED2, LOW);          // 内蔵 LED を点灯
    digitalWrite(PIN_LED2, HIGH);         // 内蔵 LED を消灯
}
```

これを書き込んでみよう。

点滅ではなく点灯し続けてしまったのではないだろうか。実はきちんと点滅しているが，Arduino は命令を処理するのが非常に速いので，人間の目には点灯し続けているように見えてしまうのである。

そこで，「一定時間何もしない」命令（遅延関数）を加えてみよう。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
}

void loop() {
    digitalWrite(PIN_LED2, LOW);          // 内蔵 LED を点灯
    delay(1000);                          // 1s 待つ
    digitalWrite(PIN_LED2, HIGH);         // 内蔵 LED を消灯
    delay(1000);                          // 1s 待つ
}
```

明らかに点滅するようになったはずである。delay(1000)は 1000ms 待つ，という意味である。この数字を変えると点滅の周期が変わることを確認してみよう。

(6) 繰り返し処理をする。

前述のように，Arduino では setup()関数と loop()関数が必須である。setup()は{ }の中の処理をはじめに 1 回だけ行い，loop()は{ }の中の処理を（電源が OFF になるまで）無限に繰り返すものである。したがって，上のスケッチは LED の点滅が 1 回だけではなくずっと繰り返されるのである。

それでは，決まった回数だけ同じ処理を繰り返すにはどのようにすればよいだろうか。Arduino ではこのようなとき for 文を用いる。スケッチを次のように変更してみよう。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
    for (int n=0; n<=4; n++) {                    // 以下の処理を 5 回繰り返す
        digitalWrite(PIN_LED2, LOW);
        delay(1000);
        digitalWrite(PIN_LED2, HIGH);
        delay(1000);
    }
}

void loop() {}
```

for 文は，繰り返し回数を数えながら同じ処理を繰り返すものである。

for (int n=0; n<=4; n++) { }は，n=0 からスタートし n を 1 ずつ増やしながら 4 になるまで{ }中の処理を行う，という意味である。上の例では n=0,1,2,3,4 の 5 回分処理をする。ここの条件を変更して，繰り返し回数を変えてみよう。

n++ は,「n に 1 を足して，改めて n に代入する」という意味である。また，n の前の int は，このスケッチでは変数 n を整数（integer）として使う，という意味である。このように Arduino では，スケッチに使用する変数の型を定義しておく必要がある。

この LED 点滅の命令が setup()の方に書かれていることに注意しよう。これを loop()の方に書くと，5 回点滅の処理を繰り返すとこになるので，結局はずっと点滅し続けてしまう。

繰り返し処理には while 文を用いてもよい。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
    int n=0;                                      // n を整数として定義し 0 を代入する
    while (n<=4) {                                // 以下の処理を n が 4 になるまで繰り返す
        digitalWrite(PIN_LED2, LOW);
        delay(1000);
        digitalWrite(PIN_LED2, HIGH);
        delay(1000);
        n++;
    }
}

void loop() {}
```

(7) 明るさを変える。

p.7 で作成したスケッチを次のように変更してみよう。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
}

void loop() {
    digitalWrite(PIN_LED2, LOW);                  // 内蔵 LED を点灯
    delay(5);                                     // 5ms 待つ
    digitalWrite(PIN_LED2, HIGH);                 // 内蔵 LED を消灯
    delay(5);                                     // 5ms 待つ
}
```

遅延を 1s から 5ms に変えただけだが，その周期が速すぎて点灯し続けているように見えてしまう（delay 文がなかったときとほとんど同じである）。試しに，基板を軽く左右に振ってみよう。実際には LED が点滅していることが確認できるはずである。

このとき，常時点灯しているときと比べて少し暗く光っているのに気づいただろうか。

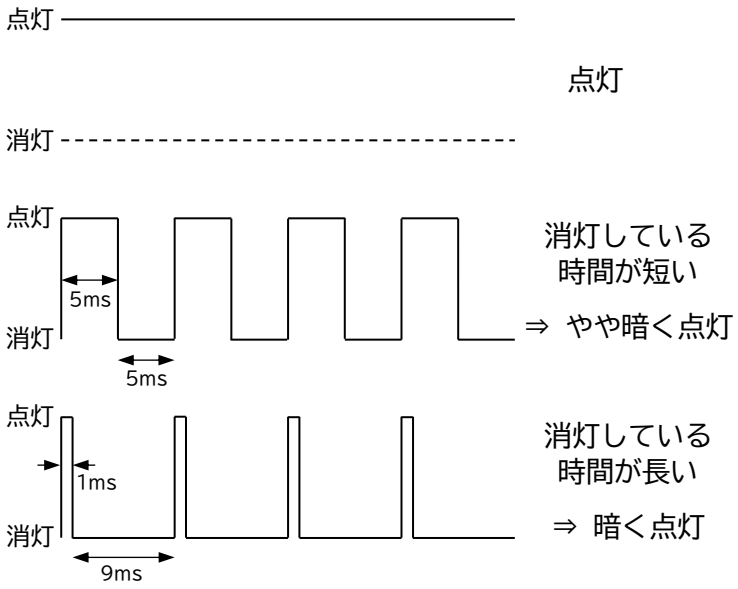
さらに次のように変更してみよう。さらに暗くなっていることが分かるはずである。

```
void setup() {
    pinMode(PIN_LED2, OUTPUT);
}

void loop() {
    digitalWrite(PIN_LED2, LOW);        // 内蔵 LED を点灯
    delay(1);                           // 1ms 待つ
    digitalWrite(PIN_LED2, HIGH);       // 内蔵 LED を消灯
    delay(9);                           // 9ms 待つ
}
```

前述のように，Arduino は基本的にデジタルで動作するため，LED はは光るか光らないかのいずれかである。しかし，高速で処理できることを利用すると，明るさを変えることができるのである。

ただ，明るさを変えたいだけなのにそれなりに長いスケッチを書く必要がある，とも言える。

なお，デジタルをアナログ的に使うために，パルスの幅を変調する方法のことを PWM（Pulse Width Modulation）と呼ぶ。




次のスケッチにしてみよう。

かなりシンプルになったが，これでも LED が暗く光っていることが分かる。

```
void setup() {}

void loop() {
    analogWrite(PIN_LED2,223);              // 内蔵 LED を点灯
}
```

Arduino には PWM を簡単に実現するための関数 analogWrite が予め用意されているのである。これまで setup で記述してきた pinMode(PIN_LED2, OUTPUT）は，内蔵 LED をデジタル出力に設定するためのものだったので，今回は不要である。

スケッチ中の 223 は明るさを決める数値である。ここには 0～255 までの整数を入力する。数値を大きくすると暗くなる。この数値を変えてみて，明るさが変わることを確認してみよう。

**問 1** for と analogWrite を利用して，「徐々に明るくなって徐々に暗くなる」点滅を繰り返すスケッチを作成せよ。

## 3. 9 軸慣性計測ユニット（加速度・角速度・磁束密度センサ）によるデータ取得

いよいよ，測定器にチャレンジしよう。

まずは 9 軸慣性計測ユニット（9 軸 IMU）を使ってみよう。加速度，角速度（ジャイロ），磁束密度（磁気）について，それぞれ 3 方向のデータを同時に取得できる便利なセンサである。

今回使用するのは BOSCH 社の BMX055 というセンサである。BMX055 が取得したデータは，「$I^2C$」という通信規格で Arduino に送ることになっている。そのデータを処理するにはハードウェアに関する専門的な知識が必要であり，本来なら非常に長いスケッチを記述しなくてはならない。

そこで，ライブラリを使ってみよう。Arduino は“オープン”であり，他の人が開発・公開したスケッチ（ライブラリ）を自由に使うことができるのであった。BMX055 を利用するためのライブラリでは，$I^2C$ による複雑なデータ処理をいくつかの関数としてまとめてあり，分かりやすい命令で実現できる。（なお，今回使用するのは，秋月電子通商が公開しているサンプルスケッチの関数群を，新海がライブラリの形に整えたものであり，ライブラリとして一般に公開されているものではない。）

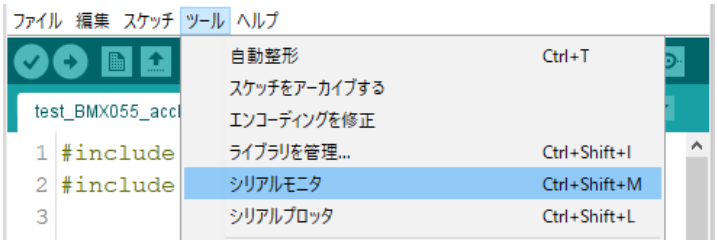
(1) センサ制御の基本スケッチ

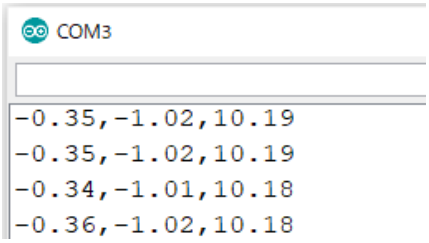
まず，加速度を測定するスケッチを作成してみよう。

```
#include <Wire.h>              // Arduino とセンサ間で I2C 通信をするライブラリの読み込み
#include <BMX055.h>            // 加速度・角速度・磁束密度センサを利用するライブラリの読み込み

BMX055 myBMX055;               // このスケッチでライブラリを使うためのインスタンスを生成

void setup() {
    Wire.begin();                                  // I2C 通信を開始する宣言
    Serial.begin(9600);                            // 通信速度（bps）の設定
    myBMX055.BMX055_Init();                        // センサによるデータを初期化
    delay(300);
}

void loop() {
    myBMX055.BMX055_Accl();                        // 加速度[m/s2]の読み取り

    Serial.print(myBMX055.xAccl);                  // 加速度の表示
    Serial.print(",");
    Serial.print(myBMX055.yAccl);
    Serial.print(",");
    Serial.println(myBMX055.zAccl);                // 改行

    delay(100);                                    // 100ms（0.1s）待つ
}
```

スケッチを書き込んでも何も起こっていないように見えるが，きちんと測定できているはずである。PC のディスプレイにデータを表示してみよう。

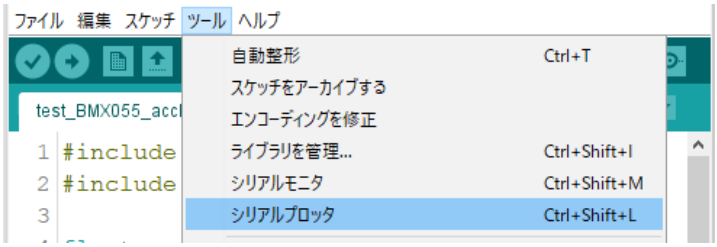
メニュー［ツール］→［シリアルモニタ］

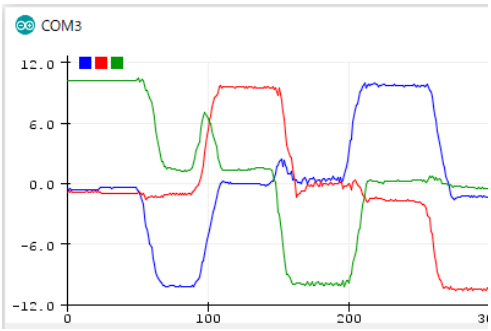



次々と数値が表示されていくのが確認できるはずである。グラフ化することもできる。

メニュー［ツール］→［シリアルプロッタ］




基板を傾けたり動かしたりしてみよう。ディスプレイに示された加速度の値が変化するはずである。静止していても 0m/s$^2$ となっていないが，これは重力加速度を測定しているからである。例えばセンサの$x$軸，$y$軸を水平にすると，$z$軸方向の加速度が 9.8m/s$^2$ または-9.8m/s$^2$ となる。

スケッチには見慣れない命令がいくつか出てきたが，それぞれのコメント文を参照してほしい。

(2) クラスとインスタンス

コメント文にある「インスタンス」という用語については説明が必要であろう。今回使用したライブラリは以下の関数と変数がまとまったもの（クラス）で，その名称が BMX055 である。

| | |
|---|---|
| BMX055_Init() | センサを初期化する関数 |
| BMX055_Accl() | 加速度[m/s$^2$]を測定し，下記の変数に一時的に保存する関数 |
| BMX055_Gyro() | 角速度[°/s]を測定し，下記の変数に一時的に保存する関数 |
| BMX055_Mag() | 磁束密度[μT]を測定し，下記の変数に一時的に保存する関数 |
| xAccl，yAccl，zAccl | 3 方向の加速度の値を一時保存するための変数，float（実数）型 |
| xGyro，yGyro，zGyro | 3 方向の角速度の値を一時保存するための変数，float（実数）型 |
| xMag，yMag，zMag | 3 方向の磁束密度の値を一時保存するための変数，int（整数）型 |

これらを使うにはクラスをもとにインスタンスを生成する（一般的な型から具体的な実体を作る）必要がある。

```
BMX055 myBMX055;
```

は，BMX055 というクラスをもとにして myBMX055 というインスタンスを生成する，という意味である。このスケッチでは，これ以後 myBMX055 という名称でライブラリが使えるようになる。インスタンスはそのスケッチでしか通用しないため my～と名付けることが多いが，名称は他の関数などと重複しなければ何でもよい。

実際にインスタンスを使ったのが次の文である。

```
myBMX055.BMX055_Init();
myBMX055.BMX055_Accl();
```

上の文は（自分が使う）myBMX055 を初期化する，下の文は（自分が使う）myBMX055 で加速度を測定するという意味である。インスタンス名とクラスで定義した関数をピリオド「.」でつなげればよい。

インスタンス名.クラスで定義される関数や変数

(3) 表示方法のバリエーション

加速度を表示する部分

```
Serial.print(myBMX055.xAccl);
Serial.print(",");
Serial.print(myBMX055.yAccl);
Serial.print(",");
Serial.println(myBMX055.zAccl);
```

を，次のように変更してみよう。

```
Serial.print(myBMX055.xAccl1,1);
Serial.print(",");
Serial.print(myBMX055.yAccl,2);
Serial.print(",");
Serial.println(myBMX055.zAccl,3);
```

シリアルモニタに表示される小数点以下の桁数が変わるはずである。

また，これらの 5 行の代わりに，

```
Serial.println(String(myBMX055.xAccl)+","+String(myBMX055.yAccl)
                                        +","+String(myBMX055.zAccl));
```

または，

```
Serial.println(String(myBMX055.xAccl,1)+","
        +String(myBMX055.yAccl,2)+","+String(myBMX055.zAccl,3));
```

としてみよう。これでも同じ結果になるはずである。1 行でも書けるわけだ。行数を減らしたいと考えるか，1 行を短くしてスケッチを見やすくしたいと考えるかは，単に好みの問題なのでどちらでもよい。1 行にする場合，複数のデータを + で連結すればよいが，それを実現するにはデータが「文字列」でなければならない。xAccl，yAccl，zAccl は実数（float 型）として定義されているので，String()という命令を加えて文字列に変換しておかなければならない。

**問 2** 角速度，磁束密度を測定するスケッチを書き，シリアルモニタとシリアルプロッタで確認せよ。加速度測定のスケッチを一部修正すればよい。

## 4. 温度・湿度・気圧センサによるデータ取得

続いて，温度・湿度・気圧センサを使ってみよう。使用するのは BOSCH 社の BME280 というセンサである。BME280 も $I^2C$ 規格に対応しており，ライブラリが公開されている。今回は SparkFun 社のライブラリを利用する。このライブラリ（BME280 クラス）では非常に多くの関数・変数が定義されているが，今回は次の 4 つのみを用いる。

| | |
|---|---|
| beginI2C() | センサを初期化する関数 |
| readTempC() | 温度[℃]を測定して実数（float 型）を返す |
| readFloatHumidity() | 湿度を[%]を測定して実数（float 型）を返す |
| readFloatPressure() | 気圧[100hPa]を測定して実数（float 型）を返す |

```
#include <Wire.h>
#include <SparkFunBME280.h>

BME280 myBME280;                        // クラスからインスタンスを生成

void setup() {
    Wire.begin();
    Serial.begin(9600);
    myBME280.beginI2C();                // センサを初期化
}

void loop() {

}
```

**問 3** 上記スケッチは loop 関数の部分が未完成である。1s おきに温度，湿度，気圧を測定するスケッチを完成させよ。インスタンス名は変更してもよい。また，センサに指をあてて温度が上がっていく様子をシリアルモニタで確認せよ。

## 5. OLED（有機 EL ディスプレイ）への文字の表示

一応，測定器の原型はできた。しかし，いちいち PC に接続していては使いにくい。作製した回路には OLED（有機 EL ディスプレイ）があるので，それに測定値を表示してみよう。

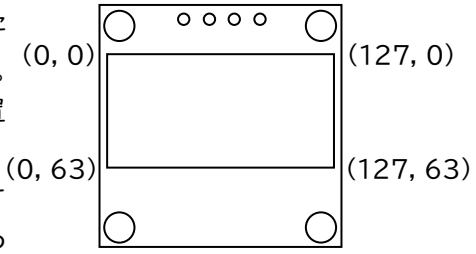
今回の OLED には SSD1306 という IC が使われている。SSD1306 も $I^2C$ 規格で Arduino とデータ通信をするためのライブラリが公開されている。今回は Adafruit 社のライブラリ（Adafruit_SSD1306 クラス）を利用する。

まず，「Hello World !」と表示するスケッチを書いてみよう。使用する関数の意味などはコメント文を参照してほしい。

```
#include <SPI.h>                                    // 各種ライブラリの読み込み
#include <Wire.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH      128                       // 各種定数を分かりやすい文字で定義しておく
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS    0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
                                  // Adafruit_SSD1306 クラスからインスタンス myDisplay を生成

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

    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);                                   // エラーが出たらプログラムを実行しない
    }

    myDisplay.clearDisplay();                       // 画面を消去
    myDisplay.display();                            // 上記命令を画面に出力
    delay(1000);                                    // 1s 待つ

    myDisplay.setTextSize(1);                       // 文字サイズ（括弧内の数値）を指定
    myDisplay.setTextColor(SSD1306_WHITE);          // 文字の色を指定
    myDisplay.setCursor(0, 0);                      // カーソルを(0, 0)に移動
    myDisplay.cp437(true);                          // フォントの種類を指定

    myDisplay.println(F("Hello World !"));          // "Hello World ! "と表示して改行
    myDisplay.display();                            // 以上の命令を画面に出力
}

void loop() {}
```

コメント文の説明を参考にして，「Hello World !」以外の文字を表示してみよう。ただし表示できるのは半角英数のみである。

また，文字サイズ，表示位置も変えてみよう。カーソルの位置は座標で指定するが，原点は左上隅なので注意しよう。




このスケッチをベースに，p.10 で作成したセンサ制御のスケッチを組み合わせれば，測定値を OLED に表示できるはずである。例として，加速度を表示するスケッチを書いてみよう。

上の Hello World ！スケッチを開いておき，加速度測定のスケッチから適宜コピー&ペーストすればよい。センサによる測定値の表示を確認できればよいので$x$方向のみでもよい。下記の例で

は 3 方向とも表示するスケッチになっている。

```
#include <SPI.h>                              // 各種ライブラリの読み込み
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <BMX055.h>

#define SCREEN_WIDTH      128          // 各種定数を分かりやすい文字で定義
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS    0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;                              // クラスからインスタンスを生成

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

    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);                             // エラーが出たらプログラムを実行しない
    }

    myDisplay.clearDisplay();                 // 画面を消去
    myDisplay.display();                      // 上記命令を画面に出力
    myBMX055.BMX055_Init();                   // センサを初期化
    delay(500);                               // 0.5s 待つ

    myDisplay.setTextSize(1);                 // 文字サイズを指定
    myDisplay.setTextColor(SSD1306_WHITE);    // 文字の色を指定
    myDisplay.cp437(true);                    // フォントの種類を指定
}

void loop() {
    myBMX055.BMX055_Accl();                   // 加速度[m/s²]の読み取り

    myDisplay.clearDisplay();                 // 画面を消去
    myDisplay.setCursor(0, 0);                // カーソルを(0, 0)に移動
    myDisplay.println(F("Accl"));             // "Accl"と表示して改行
    myDisplay.println();                      // 空白行を入れて改行

    myDisplay.print(F(" xAccl = "));          // "xAccl = "と表示
    myDisplay.print(String(myBMX055.xAccl));  // x 方向の加速度を文字列に変換して表示
    myDisplay.println(F(" m/s2"));            // " m/s2"と表示して改行
    myDisplay.println();                      // 空白行を入れて改行

    myDisplay.print(F(" yAccl = "));          // 同上（y 方向の加速度を表示）
    myDisplay.print(String(myBMX055.yAccl));
    myDisplay.println(F(" m/s2"));
    myDisplay.println();

    myDisplay.print(F(" zAccl = "));          // 同上（z 方向の加速度を表示）
    myDisplay.print(String(myBMX055.zAccl));
    myDisplay.println(F(" m/s2"));

    myDisplay.display();                      // 以上の命令を画面に出力

    delay(100);                               // 0.1s 待つ
}
```

OLED に加速度が表示され，0.1s ごとに値が更新されていくのが確認できるであろう。

Arduino IDE を一端閉じてみよう。測定が継続できていることが OLED で確認できるはずである。すなわち，USB による給電以外は PC は必要ないということである。この時点で，外部（電池など）から給電するようにすれば，測定器として単独で使用できることになる。

なお，このスケッチでは Serial.print() を使っていないので，シリアルモニタやシリアルプロッタを開いても PC のディスプレイではデータを確認できない。

**問 4** 上記スケッチを利用し，温度，湿度，気圧を 1s おきに OLED に表示するスケッチを書け。

**問 5** 上記スケッチと問 4 を利用し，加速度，角速度，磁束密度，温度，湿度，気圧を 1s おきに OLED に表示するスケッチを書け。

## 6. microSD カードへの保存

これで一応，測定器として使える形になった。しかし，このまま実験で使うと，OLED に表示されたデータを見ながら記録用紙に書き写していかなければならない。この方法は現代ではスマートとは言えない。

そこで最後のステップとして，センサで測定したデータを microSD カードに保存してみよう。

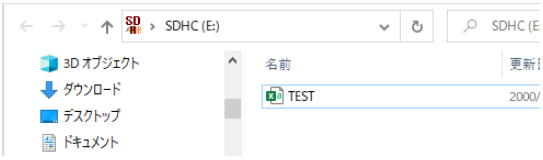
Arduino では microSD カードを制御するためのライブラリが標準で用意されているので，それを利用する。まず，microSD カードに簡単な数値を書き込むスケッチを作成してみよう。

```
#include <SPI.h>                          // 各種ライブラリの読み込み
#include <SD.h>

File myFile;                              // File クラスからインスタンス myFile を生成

void setup() {
      Serial.begin(9600);
      while (!Serial);

      if (!SD.begin(3)) {                 // エラーが出たらプログラムを実行しない
            while(1);
      }

      myFile = SD.open("test.csv", FILE_WRITE);
                                          // ファイル"test.csv"を開いて書き込みモードにする

      if (myFile) {
            myFile.println("data1,data2,data3,data4");     // "data1,data2,･･･"と書いて改行
            myFile.println("1,2,3,4");                     // "1,2,･･･"と書いて改行
            myFile.println("12,3.4,0.56,7890");            // "12,3.4,･･･"と書いて改行
            myFile.close();                               // ファイルを閉じる
      }
}

void loop(){}
```

詳細はコメント文の通りである。ファイル形式は csv で，一般的な表計算アプリ（Excel など）で読めるものである。

Seeeduino XIAO への書き込み・実行が終わったら，一旦，PC から USB ケーブルを抜いて，microSD カードを PC に差し込んで確認してみよう。

microSD カードに TEST(.CSV)というファイルが作成されている。これを開くと，1～3 行目のセルに，スケッチの命令通りの文字･数値が書き込まれているのが確認できる。例えば，




```
myFile.println("1,2,3,4");
```

のように，セルを分けたいときはカンマで区切ればよい。

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | data1 | data2 | data3 | data4 | | | |
| 2 | 1 | 2 | 3 | 4 | | | |
| 3 | 12 | 3.4 | 0.56 | 7890 | | | |
| 4 | | | | | | | |

センサで取得したデータを一定時間ごとに microSD カードに保存するには，「ファイルを開いてデータを取得し，それを書き込んで閉じる」命令を loop()に書けばよい。

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <BMX055.h>

BMX055 myBMX055;
File myFile;

int i = 1;                                          // データ識別のための ID 番号を定義

void setup() {
    Wire.begin();
    Serial1.begin(9600);                            // Serial を Serial1 に書き換えておく
    while (!Serial1);                               // 同上

    if (!SD.begin(3)) {
        while(1);
    }

    myFile = SD.open("data.csv", FILE_WRITE);       // ファイルを開いて書き込みモードにする
    if (myFile) {
        myFile.println("ID,xAccl,yAccl,zAccl");     // 1 行目にデータのキャプションを書く
        myFile.close();                             // 一旦，ファイルを閉じる
    }

    myBMX055.BMX055_Init();                         // センサを初期化
    delay(500);
}

void loop() {
    myFile = SD.open("data.csv", FILE_WRITE);       // ファイルを開いて書き込みモードにする

    if (myFile) {
        myBMX055.BMX055_Accl();                         // 加速度を測定
        myFile.print(String(i)+",");                    // ID 番号を書いて次のセルへ
        myFile.print(String(myBMX055.xAccl)+",");       // x 方向の加速度を書いて次のセルへ
        myFile.print(String(myBMX055.yAccl)+",");       // y 方向の加速度を書いて次のセルへ
        myFile.println(String(myBMX055.zAccl));         // z 方向の加速度を書いて改行
        myFile.close();                                 // ファイルを閉じる
    }

    i++;                                                // ID 番号を 1 つ増やす
    delay(100);
}
```

加速度のデータが microSD カードに書き込まれているのが PC で確認できただろうか。実験結果を分析するのに役立つであろう。グラフ化なども楽勝である。これと p.15 のスケッチを組み合わせれば，センサの測定値を OLED に表示しつつ，microSD カードに保存できるはずである。

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <BMX055.h>

#define SCREEN_WIDTH    128
#define SCREEN_HEIGHT   64
#define OLED_RESET      3
#define SCREEN_ADDRESS  0x3C                                    // 次頁へ続く
```

```
Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;
File myFile;

int i = 1;

void setup() {
    Wire.begin();
    Serial1.begin(9600);

    while (!Serial1);
    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);
    }
    if (!SD.begin(3)) {
        while(1);
    }

    myFile = SD.open("data.csv", FILE_WRITE);
    if (myFile) {
        myFile.println("ID,xAccl,yAccl,zAccl");
        myFile.close();
    }

    myDisplay.clearDisplay();
    myDisplay.display();
    myBMX055.BMX055_Init();
    delay(500);

    myDisplay.setTextSize(1);
    myDisplay.setTextColor(SSD1306_WHITE);
    myDisplay.cp437(true);
}

void loop() {
    myFile = SD.open("data.csv", FILE_WRITE);

    if (myFile) {
        myBMX055.BMX055_Accl();
        myFile.print(String(i)+",");
        myFile.print(String(myBMX055.xAccl)+",");
        myFile.print(String(myBMX055.yAccl)+",");
        myFile.println(String(myBMX055.zAccl));
        myFile.close();
    }

    myDisplay.clearDisplay();
    myDisplay.setCursor(0, 0);
    myDisplay.println(F("Accl"));
    myDisplay.println();

    myDisplay.print(F(" xAccl = "));
    myDisplay.print(String(myBMX055.xAccl));
    myDisplay.println(F(" m/s2"));
    myDisplay.println();

    myDisplay.print(F(" yAccl = "));
    myDisplay.print(String(myBMX055.yAccl));
    myDisplay.println(F(" m/s2"));
    myDisplay.println();                                                    // 次頁へ続く
```

```
    myDisplay.print(F(" zAccl = "));
    myDisplay.print(String(myBMX055.zAccl));
    myDisplay.println(F(" m/s2"));

    myDisplay.display();

    i++;
    delay(100);
}
```

**問 6** 上記スケッチと問 5 を利用し，加速度，角速度，磁束密度，温度，湿度，気圧を 0.1s おきに OLED に表示し microSD カードに保存するスケッチを書け。

## 7. 簡易実験（実験計画の立案，測定器のスケッチのカスタマイズ，測定，データ分析，発表）

作製した測定器を利用して簡単な実験をしてみよう。Seeeduino XIAO から USB ケーブルを抜き，LM3671 をソケットに差し，電池を接続し，スイッチを ON にすれば測定できる。

実験のテーマは何でもよい。測定したデータを Excel でグラフ化して分析せよ。そのグラフの概形を予想してから測定すること。測定器のスケッチは問 6 をベースとし，テーマに応じて適宜アレンジせよ。例えば，最後の delay()で指定する測定周期を変更したり，OLED に表示する，または microSD カードに保存するデータを精選したりせよ。

測定中に**スケッチを変更したい場合は，必ずスイッチを OFF にし，電池と LM3671 を外してから PC と接続すること。**

(1) テーマ

(2) 目的

(3) 方法

(4) 仮説（グラフの概形の予想）

(5) 結果

(6) 考察

## 8. 参考文献

高本孝頼（2014）「みんなの Arduino 入門」リックテレコム

## 9. OLED（有機 EL ディスプレイ）のグラフィック表示

このセクションは，時間が余った場合の「おまけ」である。OLED は文字だけでなくグラフィックにも対応している。グラフィック表示のためのライブラリは Adafruit_GFX.h である。

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>                    // グラフィック表示のためのライブラリの読み込み
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH      128
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS    0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup() {
    Wire.begin();
    Serial.begin(9600);
    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);
    }
    myDisplay.clearDisplay();
    myDisplay.display();
    delay(500);
}

void loop() {
    myDisplay.drawPixel(10,5,SSD1306_WHITE);                    // 点の描画
    myDisplay.drawLine(20,5,120,5,SSD1306_WHITE);               // 線の描画
    myDisplay.drawRect(5,15,20,20,SSD1306_WHITE);               // 長方形の描画
    myDisplay.fillRect(35,15,20,20,SSD1306_WHITE);              // 長方形の塗りつぶし
    myDisplay.drawCircle(80,25,10,SSD1306_WHITE);               // 円の描画
    myDisplay.fillCircle(110,25,10,SSD1306_WHITE);              // 円の塗りつぶし
    myDisplay.drawRoundRect(5,40,20,20,7,SSD1306_WHITE);        // 角の丸い長方形
    myDisplay.fillRoundRect(35,40,20,20,7,SSD1306_WHITE);
                                                                // 角の丸い長方形の塗りつぶし
    myDisplay.drawTriangle(80,40,70,60,90,60,SSD1306_WHITE);    // 三角形の描画
    myDisplay.fillTriangle(110,40,100,60,120,60,SSD1306_WHITE);
                                                                // 三角形の塗りつぶし
    myDisplay.display();
}
```

上で挙げた関数の詳細は以下の通りである。

drawPixel(*x*座標，*y*座標，色)
drawLine(始点の*x*座標，始点の*y*座標，終点の*x*座標，終点の*y*座標，色)
drawRect(始点の*x*座標，始点の*y*座標，*x*方向の辺の長さ，*y*方向の辺の長さ，色)
fillRect(始点の*x*座標，始点の*y*座標，*x*方向の辺の長さ，*y*方向の辺の長さ，色)
drawCircle(中心の*x*座標，中心の*y*座標，半径，色)
fillCircle(中心の*x*座標，中心の*y*座標，半径，色)
drawRoundRect(始点の*x*座標，始点の*y*座標，*x*方向の辺の長さ，*y*方向の辺の長さ，角の半径，色)
fillRoundRect(始点の*x*座標，始点の*y*座標，*x*方向の辺の長さ，*y*方向の辺の長さ，角の半径，色)
drawTriangle(頂点 1 の*x*座標，頂点 1 の*y*座標，頂点 2 の*x*座標，頂点 2 の*y*座標，
頂点 3 の*x*座標，頂点 3 の*y*座標,色)
fillTriangle(頂点 1 の*x*座標，頂点 1 の*y*座標，頂点 2 の*x*座標，頂点 2 の*y*座標，
頂点 3 の*x*座標，頂点 3 の*y*座標,色)

これと，p.15 のスケッチを利用して次のように書いてみよう。“水準器”である。

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>                    // グラフィック表示のためのライブラリの読み込み
#include <Adafruit_SSD1306.h>
#include <BMX055.h>

#define SCREEN_WIDTH      128
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS    0x3C
#define radius            3                  // 水準器の“泡”の半径を定義

int x,y;                                     // 水準器の“泡”の座標を整数として定義
Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;

void setup() {
    Wire.begin();
    Serial.begin(9600);
    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);
    }
    myDisplay.clearDisplay();
    myDisplay.display();
    myBMX055.BMX055_Init();
    delay(500);
    myDisplay.setTextSize(1);
    myDisplay.setTextColor(SSD1306_WHITE);
    myDisplay.cp437(true);
}

void loop() {
    myBMX055.BMX055_Accl();                                  // 加速度を測定
    myDisplay.clearDisplay();
    myDisplay.drawLine(45,30,95,30,SSD1306_WHITE);   // x軸を表示
    myDisplay.drawLine(115,5,115,55,SSD1306_WHITE);  // y軸を表示

    myDisplay.setCursor(5, 7);                               // 加速度の数値を[m/s²]で表示
    myDisplay.print(String(myBMX055.xAccl,1)+" ");
    myDisplay.setCursor(5, 27);
    myDisplay.print(String(myBMX055.yAccl,1)+" ");
    myDisplay.setCursor(5, 47);
    myDisplay.println(String(myBMX055.zAccl,1));

    x = int(myBMX055.xAccl*2.5);         // 加速度のx成分を軸のサイズに合わせて整数に変換
    y = int(myBMX055.yAccl*2.5);         // 加速度のy成分を軸のサイズに合わせて整数に変換
    myDisplay.fillCircle(70 - x,30,radius,SSD1306_WHITE);
                                         // 加速度に応じて“泡”のx座標を移動
    myDisplay.fillCircle(115,30 + y,radius,SSD1306_WHITE);
                                         // 加速度に応じて“泡”のy座標を移動

    myDisplay.display();
    delay(100);
}
```

回路を傾けると，“泡”が移動するのが分かるであろう。

さらに余力があれば，$a-t$図や$T-t$図などをリアルタイムで OLED に表示するスケッチにも挑戦してみよう。

## 10. 解答例

問１

```
void setup() {}

void loop() {
	for (int n=255; n>=0; n--) {
		analogWrite(PIN_LED2,n);
		delay(10);
	}
	for (int n=0; n<256; n++) {
		analogWrite(PIN_LED2,n);
		delay(10);
	}
}
```

問２　角速度

```
#include <Wire.h>
#include <BMX055.h>

BMX055 myBMX055;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBMX055.BMX055_Init();
	delay(300);
}

void loop() {
	myBMX055.BMX055_Gyro();

	Serial.print(myBMX055.xGyro);
	Serial.print(",");
	Serial.print(myBMX055.yGyro);
	Serial.print(",");
	Serial.println(myBMX055.zGyro);

	delay(100);
}
```

問２　磁束密度

```
#include <Wire.h>
#include <BMX055.h>

BMX055 myBMX055;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBMX055.BMX055_Init();
	delay(300);
}

void loop() {
	myBMX055.BMX055_Mag();

	Serial.print(myBMX055.xMag);
	Serial.print(",");
	Serial.print(myBMX055.yMag);
	Serial.print(",");
	Serial.println(myBMX055.zMag);

	delay(100);
}
```

問２　加速度・角速度・磁束密度を同時に計測

```
#include <Wire.h>
#include <BMX055.h>

BMX055 myBMX055;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBMX055.BMX055_Init();
	delay(300);
}

void loop() {
	myBMX055.BMX055_Accl();
	myBMX055.BMX055_Gyro();
	myBMX055.BMX055_Mag();

	Serial.print(”Accel x:”);
	Serial.print(myBMX055.xAccl);
	Serial.print(”m/s2, y:”);
	Serial.print(myBMX055.yAccl);
	Serial.print(”m/s2, z:”);
	Serial.print(myBMX055.zAccl);
	Serial.println(”m/s2”);

	Serial.print(”Gyro x:”);
	Serial.print(myBMX055.xGyro);
	Serial.print(”deg/s, y:”);
	Serial.print(myBMX055.yGyro);
	Serial.print(”deg/s, z:”);
	Serial.print(myBMX055.zGyro);
	Serial.println(”deg/s”);

	Serial.print(”Mag x:”);
	Serial.print(myBMX055.xMag);
	Serial.print(”uT, y:”);
	Serial.print(myBMX055.yMag);
	Serial.print(”uT, z:”);
	Serial.print(myBMX055.zMag);
	Serial.println(”uT”);

	Serial.println();

	delay(100);
}
```

問３　温度

```
#include <Wire.h>
#include <SparkFunBME280.h>

BME280 myBME280;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBME280.beginI2C();
}

void loop() {
	Serial.println(myBME280.readTempC(),1);
	delay(1000);
}
```

問３　湿度

```
#include <Wire.h>
#include <SparkFunBME280.h>

BME280 myBME280;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBME280.beginI2C();
}

void loop() {
	Serial.println(readFloatHumidity(),1);
	delay(1000);
}
```

問３　気圧

```
#include <Wire.h>
#include <SparkFunBME280.h>

BME280 myBME280;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBME280.beginI2C();
}

void loop() {
	Serial.println(readFloatPressure() / 100.0,1);
	delay(1000);
}
```

問３　温度・湿度・気圧を同時に計測

```
#include <Wire.h>
#include <SparkFunBME280.h>

BME280 myBME280;

void setup() {
	Wire.begin();
	Serial.begin(9600);
	myBME280.beginI2C();
}

void loop() {
	Serial.print("Temp: ");
	Serial.print(myBME280.readTempC(), 1);

	Serial.print(" °C, Humidity: ");
	Serial.print(myBME280.readFloatHumidity(), 1);

	Serial.print(" %, Pressure: ");
	Serial.print(myBME280.readFloatPressure() / 100.0, 1);
	Serial.println(" hPa");

	delay(1000);
}
```

問４

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <SparkFunBME280.h>

#define SCREEN_WIDTH    128
#define SCREEN_HEIGHT   64
#define OLED_RESET      3
#define SCREEN_ADDRESS  0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BME280 myBME280;

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

    if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
        while(1);
    }

    myDisplay.clearDisplay();
    myDisplay.display();
    myBME280.beginI2C();
    delay(500);

    myDisplay.setTextSize(1);
    myDisplay.setTextColor(SSD1306_WHITE);
    myDisplay.cp437(true);
}

void loop() {
    myDisplay.clearDisplay();
    myDisplay.setCursor(0, 0);

    myDisplay.print(F("Temp = "));
    myDisplay.print(String(myBME280.readTempC(),1));
    myDisplay.println(F(" C"));
    myDisplay.println();

    myDisplay.print(F("Humidity = "));
    myDisplay.print(String(myBME280.readFloatHumidity(),1));
    myDisplay.println(F(" %"));
    myDisplay.println();

    myDisplay.print(F("Pressure = "));
    myDisplay.print(String(myBME280.readFloatPressure()/100.0,1));
    myDisplay.println(F(" hPa"));

    myDisplay.display();

    delay(1000);
}
```

問 5

```
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <BMX055.h>
#include <SparkFunBME280.h>

#define SCREEN_WIDTH      128
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS  0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;
BME280 myBME280;

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

      if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
            while(1);
      }

      myDisplay.clearDisplay();
      myDisplay.display();
      myBMX055.BMX055_Init();
      myBME280.beginI2C();
      delay(500);

      myDisplay.setTextSize(1);
      myDisplay.setTextColor(SSD1306_WHITE);
      myDisplay.cp437(true);
}

void loop() {
      myBMX055.BMX055_Accl();
      myBMX055.BMX055_Gyro();
      myBMX055.BMX055_Mag();

      myDisplay.clearDisplay();
      myDisplay.setCursor(0, 0);

      myDisplay.print(F("Accl "));
      myDisplay.print(String(myBMX055.xAccl,1)+" ");
      myDisplay.print(String(myBMX055.yAccl,1)+" ");
      myDisplay.println(String(myBMX055.zAccl,1));
      myDisplay.print(F("Gyro "));
      myDisplay.print(String(myBMX055.xGyro,1)+" ");
      myDisplay.print(String(myBMX055.yGyro,1)+" ");
      myDisplay.println(String(myBMX055.zGyro,1));
      myDisplay.print(F("Mag   "));
      myDisplay.print(String(myBMX055.xMag)+" ");
      myDisplay.print(String(myBMX055.yMag)+" ");
      myDisplay.println(String(myBMX055.zMag));
      myDisplay.println();

      myDisplay.print(F("Temp "));
      myDisplay.println(String(myBME280.readTempC(),1));
      myDisplay.print(F("Humidity "));
      myDisplay.println(String(myBME280.readFloatHumidity(),1));
      myDisplay.print(F("Pressure "));
      myDisplay.print(String(myBME280.readFloatPressure()/100.0,1));

      myDisplay.display();
      delay(1000);
}
```

問６　解答例１　スケッチの行数を少なくする。

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <BMX055.h>
#include <SparkFunBME280.h>

#define SCREEN_WIDTH      128
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS  0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;
BME280 myBME280;
File myFile;

int i = 1;

void setup() {
      Wire.begin();
      Serial1.begin(9600);

      while (!Serial1);
      if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
            while(1);
      }
      if (!SD.begin(3)) {
            while(1);
      }

      myFile = SD.open("data.csv", FILE_WRITE);
      if (myFile) {
            myFile.println("ID,xAccl[m/s2],yAccl[m/s2],zAccl[m/s2],xGyro[deg/s],yGyro[deg/s],
                  zGyro[deg/s],xMag[uT],yMag[uT],zMag[uT],Temp[C],Humidity[%],Pressure[hPa]");
            myFile.close();
      }

      myDisplay.clearDisplay();
      myDisplay.display();
      myBMX055.BMX055_Init();
      myBME280.beginI2C();
      delay(500);

      myDisplay.setTextSize(1);
      myDisplay.setTextColor(SSD1306_WHITE);
      myDisplay.cp437(true);
}

void loop() {
      myFile = SD.open("data.csv", FILE_WRITE);

      if (myFile) {
            myBMX055.BMX055_Accl();
            myBMX055.BMX055_Gyro();
            myBMX055.BMX055_Mag();

            myFile.print(String(i)+",");

            myFile.print(String(myBMX055.xAccl)+","+String(myBMX055.yAccl)+","
                                                      +String(myBMX055.zAccl)+",");
            myFile.print(String(myBMX055.xGyro)+","+String(myBMX055.yGyro)+","
                                                      +String(myBMX055.zGyro)+",");
            myFile.print(String(myBMX055.xMag)+","+String(myBMX055.yMag)+","
                                                      +String(myBMX055.zMag)+",");

                                                                  // 次頁へ続く
```

```
            myFile.println(String(myBME280.readTempC())+","
                                    +String(myBME280.readFloatHumidity())+","
                                                +String(myBME280.readFloatPressure()/100.0,1));
            myFile.close();
        }

        myDisplay.clearDisplay();
        myDisplay.setCursor(0, 0);

        myDisplay.print(F("Accl "));
        myDisplay.println(String(myBMX055.xAccl,1)+" "+String(myBMX055.yAccl,1)
                                                                    +" "+String(myBMX055.zAccl,1));
        myDisplay.print(F("Gyro "));
        myDisplay.println(String(myBMX055.xGyro,1)+" "+String(myBMX055.yGyro,1)
                                                                    +" "+String(myBMX055.zGyro,1));
        myDisplay.print(F("Mag   "));
        myDisplay.println(String(myBMX055.xMag)+" "+String(myBMX055.yMag)
                                                                      +" "+String(myBMX055.zMag));

        myDisplay.println();

        myDisplay.print(F("Temp "));
        myDisplay.println(String(myBME280.readTempC(),1));
        myDisplay.print(F("Humidity "));
        myDisplay.println(String(myBME280.readFloatHumidity(),1));
        myDisplay.print(F("Pressure "));
        myDisplay.print(String(myBME280.readFloatPressure()/100.0,1));

        myDisplay.display();

        i++;
        delay(1000);
}
```

問６　解答例２　OLED の文字位置をなるべく揃える。スケッチ１行を短くする。

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Adafruit_SSD1306.h>
#include <BMX055.h>
#include <SparkFunBME280.h>

#define SCREEN_WIDTH      128
#define SCREEN_HEIGHT     64
#define OLED_RESET        3
#define SCREEN_ADDRESS  0x3C

Adafruit_SSD1306 myDisplay(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
BMX055 myBMX055;
BME280 myBME280;
File myFile;

int i = 1;

void setup() {
      Wire.begin();
      Serial1.begin(9600);

      while (!Serial1);
      if(!myDisplay.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
            while(1);
      }
      if (!SD.begin(3)) {
            while(1);
      }

      myFile = SD.open("data.csv", FILE_WRITE);
      if (myFile) {
            myFile.print("ID,xAccl[m/s2],yAccl[m/s2],zAccl[m/s2],");
            myFile.print("xGyro[deg/s],yGyro[deg/s],zGyro[deg/s],");
            myFile.print("xMag[uT],yMag[uT],zMag[uT]",);
            myFile.println("Temp[C],Humidity[%],Pressure[hPa]");
            myFile.close();
      }

      myDisplay.clearDisplay();
      myDisplay.display();
      myBMX055.BMX055_Init();
      myBME280.beginI2C();
      delay(500);

      myDisplay.setTextSize(1);
      myDisplay.setTextColor(SSD1306_WHITE);
      myDisplay.cp437(true);
}

void loop() {
      myFile = SD.open("data.csv", FILE_WRITE);

      if (myFile) {
            myBMX055.BMX055_Accl();
            myBMX055.BMX055_Gyro();
            myBMX055.BMX055_Mag();

            myFile.print(String(i)+",");
            myFile.print(String(myBMX055.xAccl)+",");
            myFile.print(String(myBMX055.yAccl)+",");
            myFile.print(String(myBMX055.zAccl)+",");
            myFile.print(String(myBMX055.xGyro)+",");
            myFile.print(String(myBMX055.yGyro)+",");
            myFile.print(String(myBMX055.zGyro)+",");                    // 次頁へ続く
```

```
            myFile.print(String(myBMX055.xMag)+",");
            myFile.print(String(myBMX055.yMag)+",");
            myFile.print(String(myBMX055.zMag)+",");
            myFile.print(String(myBME280.readTempC())+",");
            myFile.print(String(myBME280.readFloatHumidity())+",");
            myFile.println(String(myBME280.readFloatPressure()/100.0,1));
            myFile.close();
        }

        myDisplay.clearDisplay();

        myDisplay.setCursor(0, 0);
        myDisplay.print(F("a"));
        myDisplay.setCursor(15, 0);
        myDisplay.print(String(myBMX055.xAccl,1)+" ");
        myDisplay.setCursor(55, 0);
        myDisplay.print(String(myBMX055.yAccl,1)+" ");
        myDisplay.setCursor(95, 0);
        myDisplay.println(String(myBMX055.zAccl,1));

        myDisplay.setCursor(0, 10);
        myDisplay.print(F("w"));
        myDisplay.setCursor(15, 10);
        myDisplay.print(String(myBMX055.xGyro,1)+" ");
        myDisplay.setCursor(55, 10);
        myDisplay.print(String(myBMX055.yGyro,1)+" ");
        myDisplay.setCursor(95, 10);
        myDisplay.println(String(myBMX055.zGyro,1));

        myDisplay.setCursor(0, 20);
        myDisplay.print(F("B"));
        myDisplay.setCursor(15, 20);
        myDisplay.print(String(myBMX055.xMag)+" ");
        myDisplay.setCursor(55, 20);
        myDisplay.print(String(myBMX055.yMag)+" ");
        myDisplay.setCursor(95, 20);
        myDisplay.println(String(myBMX055.zMag));

        myDisplay.setCursor(0, 35);
        myDisplay.print(F("Temp"));
        myDisplay.setCursor(72, 35);
        myDisplay.println(String(myBME280.readTempC(),1)+" C");
        myDisplay.setCursor(0, 45);
        myDisplay.print(F("Humidity"));
        myDisplay.setCursor(72, 45);
        myDisplay.println(String(myBME280.readFloatHumidity(),1)+" %");
        myDisplay.setCursor(0, 55);
        myDisplay.print(F("Pressure"));
        myDisplay.setCursor(60, 55);
        myDisplay.print(String(myBME280.readFloatPressure()/100.0,1)+" hPa");

        myDisplay.display();

        i++;
        delay(1000);
}
```