# 講習会

# Arduino

6回目

タイマー（時間の関数）

# 目的

- 時間に関する関数を覚え、プログラムの制御に使う。
- タイマー割込みを知る。

- 制御をするときにセンサー等の外部入力以外にも時間で制御したい時があるかもしれない。
- 例えば一定の間隔をあけてから制御したい場合や制御してからある時間以内ではこの制御、過ぎてからはこの制御をしたいなど。
- 時間に関する関数を紹介する。
- 時間による割り込みを入れることにより正確な計測を行うことができる。

# 時間に関する関数（電源を入れてから適用）

- millis()
- micros()

# millis()

- Arduinoボードが現在のプログラムを起動してから経過した時間をミリ秒単位で返却する。この値は、約50日後にオーバーフロー(ゼロに戻る)する。

- グローバル変数で　unsigned long millis()　と書いておくとよい。（書かなくても良い）
- 変数を宣言する際もunsigned long (time)とする。

(time)は時間として使いたい変数名

- プログラムが開始してからのミリ秒が返される。

# 注意

- millis()の返却値はunsigned longである。例えば int などの他のデータ型と計算しようとするとエラーになるかもしれない。符号付longも、unsigned longの半分の数値を格納できるだけなので、エラーになるかもしれない。
- unsigned long以外で宣言しないこと。

# 使用例

```
unsigned long time;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = millis();
  //プログラムを開始してからの時間を表示する
  Serial.println(time);
  // 大量のデータを送らないよう、1秒待つ
  delay(1000);
}
```

# micros()

- Arduinoボードが現在のプログラムを起動してから経過した時間をマイクロ秒単位で返却する。この値は、約70分後にオーバーフロー(ゼロに戻る)する。16MHzのArduinoボード(例えば、DuemilanoveやNano)では、この関数の精度は4 マイクロ秒である。すなわち、返却値はいつも4の倍数である)。8MHzのArduinoボード(例えば、LilyPad)では、この関数の精度は8マイクロ秒である。

- 注意:1ミリ秒は1000マイクロ秒で、1秒は1000000マイクロ秒である。

- unsigned long micros()で宣言される。（自分で宣言する必要はない）
- プログラムが開始してからのマイクロ秒が返される。

```
unsigned long time;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = micros();
  //プログラムを開始してからの時間を表示する
  Serial.println(time);
  // 大量のデータを送らないよう、1秒待つ
  delay(1000);
}
```

# 時間に関する関数（流れを停止させるもの）

- delay()
- delayMicroseconds()

# delay()

- 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。

- delay(ms);
- ms　一時停止する時間(ミリ秒単位)

# 使用例

```
int ledPin = 13; // デジタルピンの13番に接続されているLED

void setup() {
        pinMode(ledPin, OUTPUT); // デジタルピンを出力に設定する
}

void loop() {
        digitalWrite(ledPin, HIGH); // LEDを点ける
        delay(1000); // 1秒待つ（1000ms＝1s）
        digitalWrite(ledPin, LOW); // LEDを消す
        delay(1000); // 1秒待つ
}
```

# 注意

- delay()関数を使ってLEDを点滅させることは簡単だし、多くのスケッチではスイッチのチャタリングを防止するために少しの遅延を利用しているが、スケッチ でdelay()を利用することには重大な欠点がある。delay()関数を使っている間は、センサーから値を読み取ったり、計算をしたり、ピンの制御を行うことができない。その結果、ほとんどの動作を止めてしまう。タイミングを制御する別の方法を知りたいときには、millis()関数や参考のスケッチを参考にしてほしい。精通したプログラマは、スケッチが非常に単純であるとき以外は、10ミリ秒以上の時刻イベントを扱うときにはdelay()関数を利用しない。
- delay()関数は割り込みを禁止しないので、delay()の実行中も、Atmegaチップの制御を行うための処理は実行される。RXピンを使ったシリアル通信は記録さ れ、PWM(analogWrite())波とピンの状態は保持され、割り込みは意図したとおりに動作する。

# delayMicroseconds()

- 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。これは、将来のArduinoリリースで変更される可能性がある。数千マイクロ秒よりも長 い遅延が必要なときは、delay()を利用すること。

- delayMicroseconds(us);
- us　一時停止する時間(マイクロ秒単位)

# 使用例

- int outPin = 8;　　// デジタルピンの8番
- 
- void setup(){
- 　pinMode(outPin, OUTPUT);　　// 8番ピンを出力(OUTPUT)に設定
- }
- 
- void loop(){
- 　digitalWrite(outPin, HIGH);　// ピンをONにする
- 　delayMicroseconds(50);　　// 50マイクロ秒待つ
- 　digitalWrite(outPin, LOW);　// ピンをOFFにする
- 　delayMicroseconds(50);　　// 50マイクロ秒待つ
- }

- 8番ピンを出力に設定する。約100ミリ秒ごとにパルスを発生させる。概算なのは他の命令が実行されるためである。

# 注意

- この関数は、3マイクロ秒以上の範囲でとても正確に動作する。非常に小さい遅延時間に対しては、delayMicroseconds()の精度は保障できない。

- Arduino 0018以降から、delayMicroseconds()は割り込み禁止にしない。

# millis(),micros()を使うときの注意

- 電源をONにした時から処理が始まるということは電源をOFFにするまでずっと数値が増加するということになる。
- これらの関数に数値を代入することができないので手動で

  millis() = 0;

  にはできない。
- 自分で変数を作ってmillis()を代入すれば変数内でいじることは可能になるため、使い方次第ではタイマーを作ることができる。

# delay系の関数を使うときの注意

- delayを使っている間はほかの処理ができない。しかし、動作を止めるわけではないので入力はできないけど出力は実行したままである。
- 全ての動作を止めてからdelayを行いたいときはdelayを書く前にすべての動作を止める処理を書くこと。

# 使い分け

- millis()
- ロボットなど終始物理的に動き続けて、何かしらの処理を続けていないとダメなもの

- delay()
- イルミネーションなど物理的に動いてはいないもの。
- LEDを光らす、PWMを出力。

# 時間みたく使えるもの

- (例）cnt++

# cnt++

- 変数と演算で紹介したこの演算は上から順番に読むプログラムが、この式を書いた行を通過したときにcntを1増加させる。
- つまり、通過するたびに増加するのだから時間としても使えることになる。
- しかし、プログラムの行が多くなるたびに増加までの間隔は長くなる。
- あくまで参考として使うのでプログラムの行が追加されたら時間も変えないといけない。
- 例　if(cnt > 100)　if(cnt > 110)など
- cntは変数であるため、名前は自由。

# 例1　間隔が短い

- {cnt++;
- 1
- 2
- 3
- 4
- 5
- }

# 例2　間隔が長い

- {cnt++;　例1に比べて2倍の時間が掛かる。
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- }

# タイマーを作る

- タイマーで気を付ける部分は時間関数を使うのはミリ秒だけで良くて秒や分、時はただの変数で良いということである。

# 時間を増加

- time1 = millis();
- time3 = time1 – time2;
- if(time3 > 999){
- time2 = time1;
- s = s+1;
- }

# 時間をリセット

- time1 = millis();
- time3 = time2 – time1;
- if(s1 == HIGH){
- time2 = time1;  //time1とtime2は同じになる
- }

# タイマー割込み

- MsTimer2のライブラリを用いてタイマー割込みを行う。
- MsTimer2はライブラリであり、Arduinoのソフトには同梱されていない。
- 自分でライブラリをインストールする必要がある。

# MsTimer2とは？

- MsTimer2はTimer2を人間とインターフェースするための、小さくて使いやすいライブラリ。
- Timer2で1ミリ秒の分解能を「ハードコード」するため、MsTimer2と呼ばれている。

# Arduinoのタイマー

- Arduinoはハードウェアの機能としてタイマーを持っている。
- これはArduinoで使用しているAVRというマイコンの機能で、各種ライブラリ上から使われている。
- Arduinoで使用している主要なAVRマイコンでは、Timer0、Timer1、Timer2の３つのタイマーがある。
- これらのハードウェアのタイマーを利用し、一定時間ごとにある関数を「割り込み（interrupt）」で呼び出すことをArduinoのタイマー系ライブラリは実現している。

- Timer0 8bitのタイマーでArduinoの時間を管理する用途で利用されている。delay(), millis(), micros()などである。UNOでは5,6番ピンのPWMで利用されている。
- Timer1 16bitのタイマーでUNOではServoライブラリと9,10番ピンのPWMで利用されている。
- Timer2 8bitのタイマーでUNOではtone()と3,11番ピンのPWMで利用されている。

- Timerの説明にある通り、AVRのタイマー機能はArduinoの各種ライブラリによって利用されていて、タイマーライブラリを使う際は、利用されていないタイマーを使うなど競合を避けないければならない。PWMもまた、タイマーを利用しているため、同時に使用することは出来ない。

# ライブラリのインストール方法

- 「MsTimer2.zip」をダウンロードし、ArduinoIDEメニューから「スケッチ」→「ライブラリをインクルード」→「.ZIP形式のライブラリをインストール」をクリックし、ダウンロードしたファイルを選択しインストールする。
- GitHubからライブラリをダウンロードできる。

# MsTimer2::set(unsigned long ms, void ())

- unsigned long ms　時間の間隔
- Void()　呼び出す関数名

- オーバーフローする時間をmsで指定する。オーバーフローするたびに、関数が呼ばれる。fは引数なしのvoid型として宣言すること。

# MsTimer2::start()

- 割り込みを有効にする。

# MsTimer2::stop()

- 割り込みを無効にする。

# 1秒周期でピン13のLEDをオンオフする

```
#include <MsTimer2.h>

void flash() {
  static boolean output = HIGH;

  digitalWrite(13, output);
  output = !output;
}

void setup() {
  pinMode(13, OUTPUT);

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

void loop() {
  // loopは空
}
```

- Timer1や3も割り込みとしては存在するが、今回は2だけ知っていればよい。

- 以下のサイトに詳しく載っている

https://miso-engine.hatenablog.com/entry/2015/07/20/221014

# 終わりに

- 時間の関数を使えるようになるとセンサが反応して処理した後すぐにセンサが反応しても処理をしなくなるなど、休憩時間を設けることができるのでミスが少なくなる。
- チャタリング防止にもつながるので状況に応じてmillis()とdelay()を使い分けよう。
- 次回は数値と関数（数学的関数）を取り扱う。

# 課題

- タイマーを作ってみよう

- 言い忘れていたことがあれば指示する。

- TinkerCADを用いてタイマーを作ってみよう。
- 秒だけではなく分も表示してみよう。
- タクトスイッチを用いてリセットもやってみよう。

# シリアルでパソコンに表示させる

- setupにSerial.begin(9600);

- loopには
- Serial.print(分を表す文字);
- Serial.print(“ “);
- Serial.print(分を表す変数);
- Serial.print(“ “);
- Serial.print(“秒を表す文字“);
- Serial.print(“ “);
- Serial.print(秒を表す変数);
- Serial.print(“ “);
- Serial.print(“ミリ秒を表す文字“);
- Serial.print(“ “);
- Serial.println(ミリ秒を表す変数);