講習会

Arduino

12回目　最後

センサー入力

目的

- 3回目「ディジタル入力と出力」と4回目「アナログ入力と出力」を応用してセンサーの値を入力。Arduinoの処理として利用する。
- 入手したセンサー値から適切な処理を行う。

センサー

- 自然現象や人工物の機械的・電磁気的・熱的・音響的・化学的性質あるいはそれらで示される空間情報・時間情報を、何らかの科学的原理を応用して、人間や機械が扱い易い別媒体の信号に置き換える装置のことをいい、センサを利用した計測・判別を行うことを「センシング」という。
- 検知器とも呼ばれる。
- 世の中には数多くのセンサーがあり、人間に目や耳、鼻もセンサーとして考えることができる。

原理による分類　機械量

- 加速度
- 　加速度センサ

- 力
- 　ストレインゲージ(ひずみゲージ)
- 　ロードセル（荷重による変位量がわかっている物体とひずみゲージを組み合わせた荷重センサ）
- 　半導体圧力センサ

- 振動
- 　音波 - マイクロフォン
- 　超音波

熱

- 温度
- 　接触式
- 　　サーミスタ
- 　　抵抗測温体
- 　　熱電対
-
- 　非接触式
- 　　放射温度計

光・放射線

- 光
- 　光センサ
- 　光電素子
- 　フォトダイオード
- 赤外線
- 放射線

電気

- 電場
- 電流
- 電圧
- 電力

磁気

- 磁気センサ

化学

- におい
- イオン濃度
- ガス濃度

時空間による分類　時間

- 時計

位置

- 光位置センサ (PSD)
- リミットスイッチ

距離

- 超音波距離計
- 静電容量変位計
- 光学式測距
- 電磁波式測距

変位

- 差動トランス
- リニアエンコーダ

速度

- レーザードップラー振動速度計
- レーザドップラー流速計

回転角

- ポテンショメータ
- 回転角センサ

回転数

- タコジェネレータ
- ロータリエンコーダ

角速度

- ジャイロセンサ

一次元画像

- リニアイメージセンサ

二次元画像

- CCDイメージセンサ
- CMOSイメージセンサ

用途による分類

- 液
- 　漏液センサ（リークセンサ）　　（読み：ろうえき）
- 　液検知センサ（レベルセンサ）
- 硬度
- 湿度
- 流量
- 傾斜
- 地震センサ

- このように数多くのセンサーによって世の中の機械は動いている。
- センサーがないと機械を制御することはできない。
- 例えばボリュームとかでもアナログ値を読んで処理をしているという意味ではセンサーである。
- スイッチでも押した押さないを判別して処理をするためセンサーである。

マイコンカーにあるセンサー例

- ポテンショメータ
- 抵抗値から角度を測定

- ロータリーエンコーダ
- 回転量から速度や距離を測定

- 反射型デジタルセンサ
- 白線の有無を測定

使い方が簡単なセンサー

- スイッチ
- デジタル式の測距センサ

- これらはdigitalReadで入力処理が可能

- ポテンショメータ（ボリューム）

- analogReadで入力処理が可能

デジタル入力での簡単な方法

```
void setup() {
  pinMode(2,INPUT);
  pinMode(12,OUTPUT);
}

void loop() {
  if(digitalRead(2) == HIGH){
    digitalWrite(12,HIGH);
  }
  else{
    digitalWrite(12,LOW);
  }
}
```

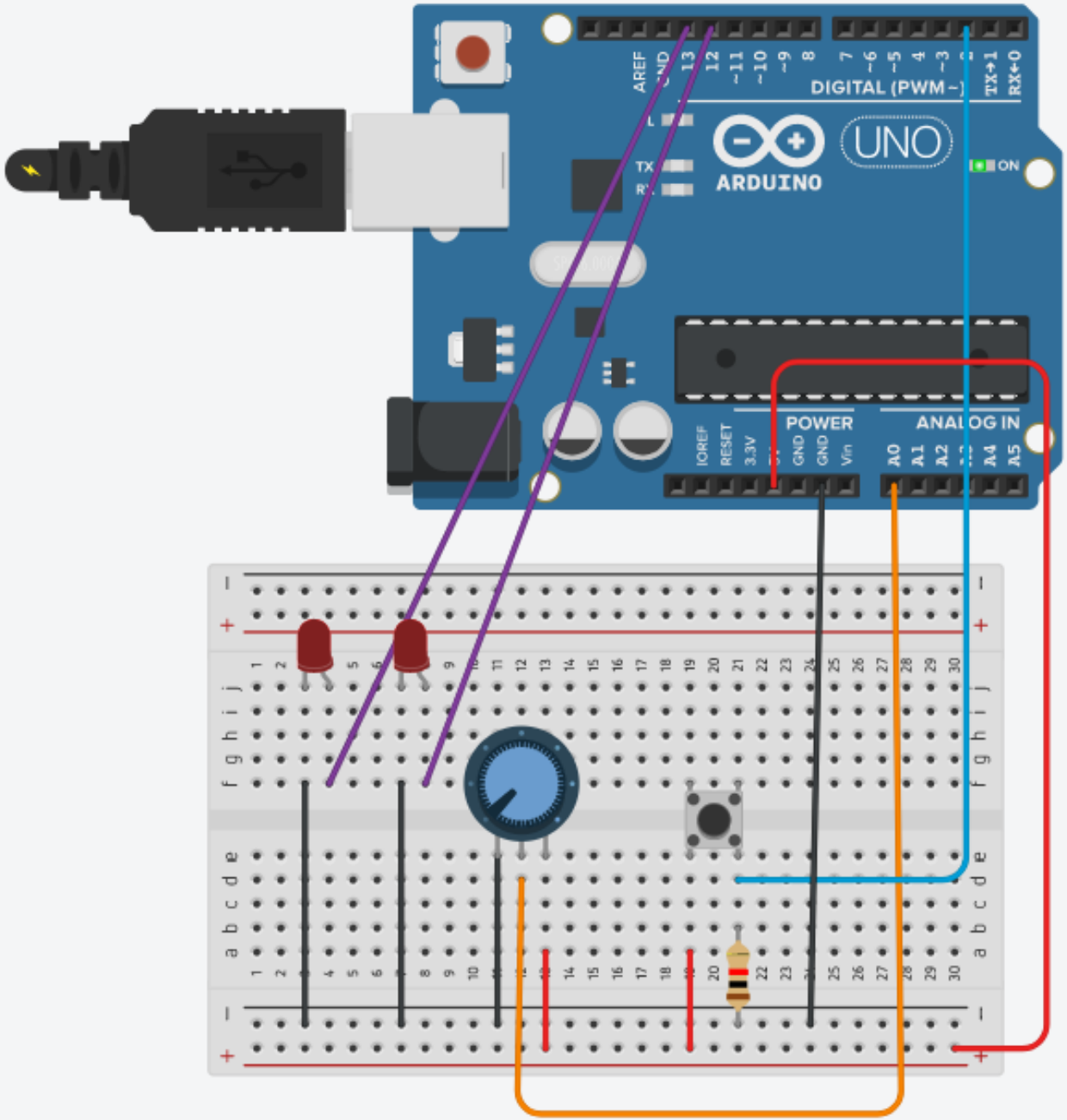
- スイッチを押すとLEDが点灯する
- スイッチが入力装置でありLEDが出力装置
- スイッチが押されたのかを判断しているためセンサーの役割を持つことが分かる。

アナログ入力での簡単な方法

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

void loop() {
if(analogRead(A0) > 500){
    digitalWrite(13,HIGH);
  }
  else{
    digitalWrite(13,LOW);
  }
}
```

- ボリュームを回して半分を超えたらLEDが点灯
- ボリュームが入力装置でありLEDが出力装置
- ボリュームでの電圧降下から残った電位を判断しているためセンサーの役割を持つことが分かる。


DIGITAL (PWM~)
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
GND
GND
Vin
A0
A1
A2
A4
A5
TX→1
RX←0

動画

- https://www.youtube.com/watch?v=8hHZWZOJAo0

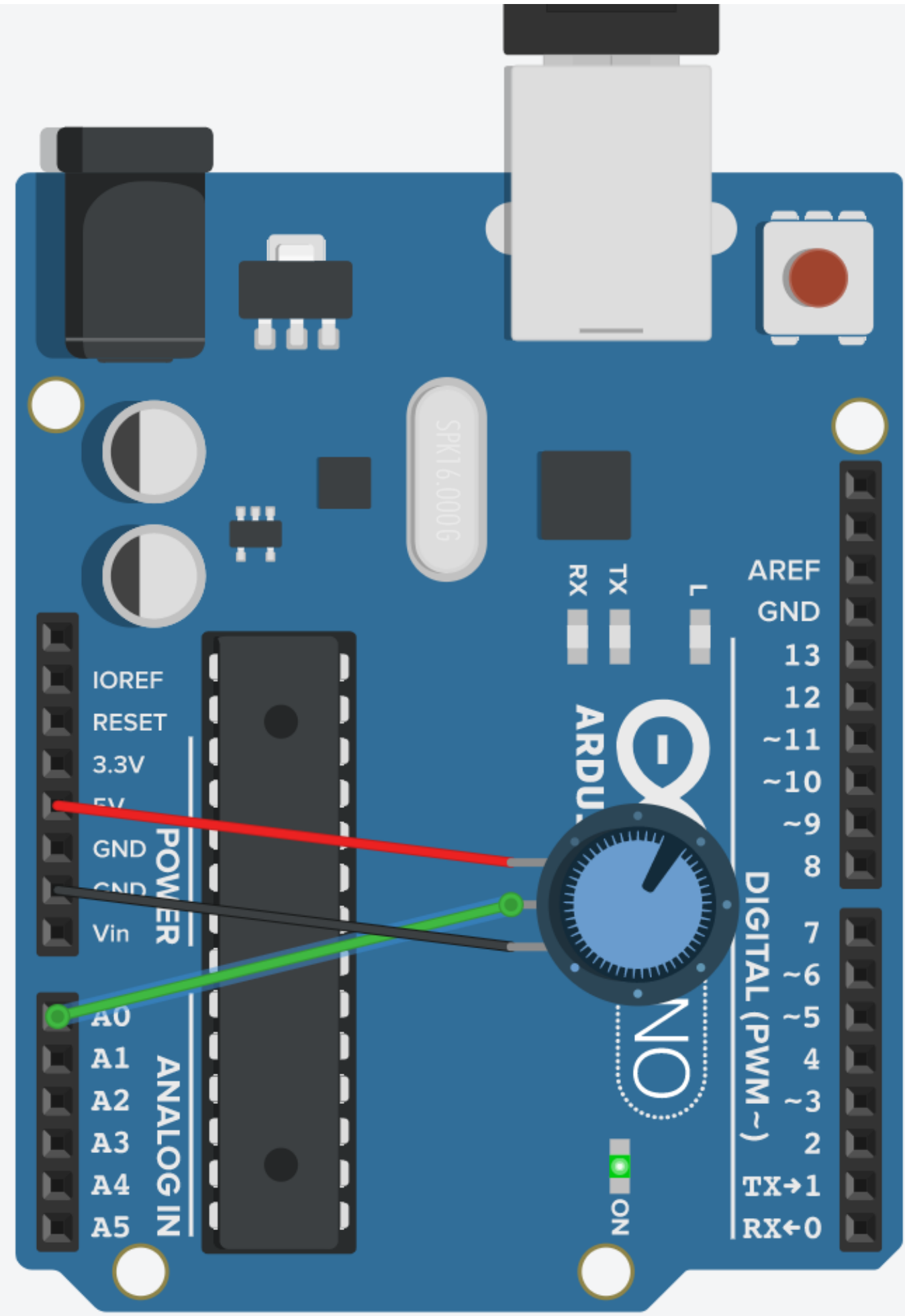
ポテンショメータの例2

```
int check;
int check2;
int test;

void setup(){
 Serial.begin(9600);
  check = analogRead(A0);
}

void loop(){
 check2 = analogRead(A0);
  test = check2 - check;
  Serial.println(test);
}
```

- 始めの抵抗値を読んで、そこを基準とする。
- それ以降ボリュームを左右に傾けると数値が増減してどのくらいズレているのか分かる。
- ボリュームが入力装置である。
- 基準を0として正負の値が出力される。


IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
RX
TX
L
ON
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)

動画

- https://www.youtube.com/watch?v=qNwIKKZtwog

- Arduinoではデジタル、アナログの入力処理が非常に簡単であるためすぐに使えるようになると思われる。

- しかしこのままでは順番に読んでいるので優先順位とかがない。
- 読み飛ばしが起こる可能性があるため、センサーの入力に外部からの割り込み処理を入れることができる。
- 一応紹介するが確信を得ていない。

関数紹介（外部割り込み）

- attachInterrupt()
- detachInterrupt()

attachInterrupt()

- attachInterruptの1番目の引数は割り込み番号である。通常、デジタルピンを割り込み番号に変換するには、digitalPinToInterrupt(pin)を利用する必要がある。例えば、3番ピンを利用するときには、digitalPinToInterrupt(3)を、attachInterrupt()の1番目の引数に指定する。

ボード	割り込みに利用可能なデジタルピン
Uno、Nano、Mini、他の328ベースのボード	2、3
Mega、Mega2560、MegaADK	2、3、18、19、20、21
Micro、Leonardo、他の32u4ベースのボード	0、1、2、3、7
Zero	4番ピン以外のすべてのデジタルピン
Due	すべてのデジタルピン

注意

- 指定した関数の中では、delay()は動作せず、また、millis()によって返却される値も増加しない。
- その関数の実行中に受信したシリアルデータはなくなる可能性がある。
- 指定した関数の中で変更する変数は、volatile変数として宣言するべきである。下記のISRを参照のこと。

割り込みの利用

- マイクロコントローラのプログラム中で自動的に何かを行おうとしたときに、割り込みは有用であり、タイミング問題を容易に解決する。割り込みを使うよい例には、ロータリーエンコーダの読み込みや、ユーザ入力の監視などがある。

- プログラムが、ロータリーエンコーダからの信号を欠落することなく常に補足することを保証したいと思ったら、発生した信号を捕捉するためにプログラムは常にエンコーダがつながっているセンサーを監視しなければならないので、(割り込みを使わない場合は)何をするにもとても巧妙なプログラムを書く必要がある。
- クリックを捕捉するための音センサや硬貨が落ちるのを捕捉する赤外線センサといった、他のセンサーもインタフェースに関する同様の問題をもっている。
- これらの全ての状況において、割り込みを利用することで、呼び鈴が鳴ったことの検出を失敗することなく、他の作業をマイクロコントローラに自由に実行させることができる。

ISR(Interrupt Service Routine: 割り込みサービスルーチン)について

- ISRは、他の多くの関数が持っていない特徴を持つ特別な関数である。
- ISRは、引数をとることができず、戻り値もない。
- 一般に、ISRはできる限り短くかつ速くなければならない。スケッチ内に複数のISRがあっても、同時には一つだけが実行される。
- 他のISRは現在のISRの終了後、優先度に応じた順序で実行される。
- millis()の計測は割り込みに依存しているので、他のISRの実行中には増加しない。
- delay()の動作には割り込みが必要なので、ISRの中から呼ばれても動作しない。
- micros()は、最初は動作するが、1〜2秒後には乱れてしまう。delayMicroseconds()は(割り込みに依存する)カウンタを利用しないので、通常通り動作する。
- 一般的に、ISRとメインプログラムとのデータのやり取りにはグローバル変数を利用する。
- ISRとメインプログラムで共有する変数が確実に更新されるためには、その変数をvolatileとして宣言する必要がある。

- attachInterrupt(interrupt, function, mode);

- interrupt 割り込みの番号。digitalPinToInterrupt()を利用する。
- function 割り込みが発生したときに呼び出す関数へのポインタ。この関数は引数も返り値も持たない。割り込みサービスルーチンと呼ばれることもある。

- mode いつ割り込みサービスルーチンを呼び出すかを指定する。以下に示す4つの定数が定義されている。
- LOW ピンがLOWのときに割り込みサービスルーチンを呼び出す
- CHANGE ピンの値が変わった時に割り込みサービスルーチンを呼び出す
- RISING ピンがLOWからHIGHになった時に割り込みサービスルーチンを呼び出す
- FALLING ピンがHIGHからLOWになった時に割り込みサービスルーチンを呼び出す

例

```
int pin = 13;
volatile int state = LOW;

void setup() {
    pinMode(pin, OUTPUT);
    attachInterrupt(digitalPinToInterrupt(pin), blink, CHANGE);
}

void loop() {
    digitalWrite(pin, state);
}

void blink() {
    state = !state;
}
```

```
const byte ledPin = 13;
const byte interruptPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(interruptPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
}

void loop() {
  digitalWrite(ledPin, state);
}

void blink() {
  state = !state;
}
```

割り込み番号

- 通常、割り込み番号を直接スケッチに記述するのではなく、digitalPinToInterrupt(pin)を利用すべきである。ピン番号と割り込み番号のマッピングはボードにより異なる。割り込み番号を直接書くのは簡単に見えるが、他のボードで利用するときの互換性に問題がある。

- しかし、古いスケッチではしばしば割り込み番号を直接記述している。0(デジタルピン2番)と1(デジタルピン3番)がよく使われている。次の表に各ボードでのピンと割り込み番号の関係を示す。

ボード	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
32u4ベース(Leonard、Microなど)	3	2	0	1	7	
Due、Zero	(下記参照)					

Arduino Dueは強力な割り込み能力を持っている。すべてのピンに割り込みサービスルーチンを割り当てることができる。attaachInterrupt()でピン番号を直接指定可能である。

Arduino Zeroでは、4番ピン以外のピンに割り込みサービスルーチンを割り当てることができる。attaachInterrupt()でピン番号を直接指定可能である。

detachInterrupt()

- 指定した割り込みを抑制する。

- detachInterrupt(interruptNum);
- interrupt 割り込みを抑制する割り込みの番号。
  digitalPinToInterrupt(pin)を利用する。

外部割込みの例

- 次の例はスタートゲートタイマーを作った際に書いたプログラムの外部割込み部分である。
- 合っているかは分からない。

```
volatile int sens1;
volatile int sens2;

void Sensor1() {//1
  if (digitalRead(sensor1) == HIGH) {
    sens1 = 1;
  }
  if (digitalRead(sensor1) == LOW) {
    sens1 = 0;
  }
}
void Sensor2() {//2
  if (digitalRead(sensor3) == HIGH) {
    sens2 = 1;
  }
  if (digitalRead(sensor3) == LOW) {
    sens2 = 0;
  }
}
```

```
• void setup(){
• pinMode(sensor1, INPUT_PULLUP);
• pinMode(sensor3, INPUT_PULLUP);
• attachInterrupt(0, Sensor1, CHANGE);//D2
• attachInterrupt(1, Sensor2, CHANGE);//D3
• }
```

関数紹介（割り込み）

- interrupts()
- noInterrupts()

interrupts()

- (noInterrupts()によって禁止された後)割り込みを許可する。割り込みによって、バックグラウンドである種の重要なタスクが起動する。割り込みはデフォルトでは許可されている。割り込みが禁止されているときには動作しない関数がある。また、外部からの受信したデータは無視される。割り込みは、プログラムが実行されるタイミングに少し悪影響を与える。特に、プログラムのクリティカルセクションでは割り込みを禁止することがある。

- void interrupts(void)

例

- void setup() {}
-
- void loop(){
- noInterrupts();
- // critical, time-sensitive code here
- interrupts();
- // other code here
- }

noInterrupts()

- 割り込みを禁止する。禁止した割り込みは、interrupts()で再度許可することができる。割り込みによって、バックグラウンドである種の重要なタスクが起動する。 割り込みはデフォルトでは許可されている。割り込みが禁止されているときには動作しない関数がある。 また、外部からの受信したデータは無視される。 割り込みは、プログラムが実行されるタイミングに少し悪影響を与える。 特に、プログラムのクリティカルセクションでは割り込みを禁止することがある。

- void noInterrupts(void)

例

- void setup() {}
-
- void loop(){
- noInterrupts();
- // クリティカルで、リアルタイム性の高いコード
- interrupts();
- // ほかのコード
- }

クリティカルセクション

- クリティカルセクション(Critical section)とは、計算機上において、単一のリソースに対して、複数の処理が同時期に実行されると、破綻をきたす部分を指す。クリティカルセクションにおいては、排他制御を行うなどしてアトミック性を確保する必要がある。
- リソースの同一性が保証されなくなる可能性がある場合は、クリティカルセクションでは常に排他制御を行う必要がある。
- クリティカルセクションの排他制御ではデッドロックに注意する必要がある。

- 今回は触れていないが細かい例が以下のサイトに書いている。
- https://www.ei.tohoku.ac.jp/xkozima/lab/electro-miyagino3.html

最後に

- 今回のセンサー入力に関する内容を持ってArduino講習会は終わりとなる。
- 今まで話した内容は自分の経験談を含めているので実際に利用されているものとは異なるものが多いと思われる。
- 複数のサイトを参考に作っている内容もあるので、ネットや本を参考に新しく調べてみると面白いと思われる。

試験について

- 3月23日金曜日の午後から今までの確認試験を行う。
- 範囲はArduinoの1回目から12回目まで全般を出す予定だが、出さない部分もあるかもしれない。偏りがある。
- 筆記（穴埋め）50点、説明2問20点、実技（TinkerCADによるシミュレーション）3問30点の予定。100点満点。60点以上が合格。
- 筆記は口頭で文章を読むから何の説明か、もしくは開いている部分を答える。（ホワイトオードに書いてその場で採点する）
- 説明はお題をPPで表示するから1人10分くらいで説明。
  （ホワイトボードを使ってよい）
- 実技はお題をPPで表示するからTinkerCADを使って回路とプログラムを組んでシミュレーションする。実際に動作すれば点が入る。

課題

- アナログ入力とデジタル入力でLEDの点灯をしてみる。
- アナログ入力で得た数値をシリアルモニタで表示してみる。
- 割り込み処理をしてみる（応用）。