講習会

Arduino

7回目

数値と関数（数学的関数）

目的

- Arduinoで使える数学的な関数を覚える。
- Arduino以外ではあまり見ないような関数が多い。
- 直接必要かと言われればそうでもないと思われるが、使いこなせば処理が簡単になることもある。
- 中には関数同士を組み合わせるものも存在する。

数学関数

- min
- max
- abs
- constrain
- map
- pow
- sqrt

min()

- 2つの数値の最小値を計算する。

- min(x, y);
- x　一つ目の数値(任意の型)
- y　二つ目の数値(任意の型)

- 小さいほうの数値が返される。

使用例

- sensVal = min(sensVal, 100);
- 　// sensValにsensValと100のうち、小さいほうを設定する
- 　// これにより、100より大きい値にならないことを保証する

注意

- 直感的ではないかもしれないが、max()は、変数の範囲の下限値を制限するときによく使われ、min()は上限値を制限するときに使われる。

- min()はマクロとして実装されているため、引数に他の関数を使わないようにすること。そのような使い方をすると、意図しない結果になることがある。
- min(a++, 100);　// このような使い方は意図しない結果を引き起こすので避ける。
- min(a, 100);
- a++;　　// 他の計算や関数はmin()の外で実行する

max()

- 2つの数値の最大値を計算する。

- max(x, y);
- x　　一つ目の数値(任意の型)
- y　　二つ目の数値(任意の型)

- 大きいほうの数値が返される。

使用例

- sensVal = max(senVal, 20);
- // sensValにsensValと20のうち、大きいほうを設定する
- // これにより、20より小さい値にならないことを保証する

注意

- 直感的ではないかもしれないが、max()は、変数の範囲の下限値を制限するときによく使われ、min()は上限値を制限するときに使われる。
- max()はマクロとして実装されているため、引数に他の関数を使わないようにすること。そのような使い方をすると、誤った結果になることがある。
- max(a--, 0);　// このような使い方は意図しない結果を引き起こすので避ける。
- min(a, 0);
- a--;　　// 他の計算や関数はmin()の外で実行する

abs()

- 数値の絶対値を計算する。

- abs(x);
- x　　数値

- ｘが0以上のときｘ、 ｘが0より小さいとき-xを返す。

使用例

- x = abs(x);
- //xの絶対値を返す

```
int x;
int y;
void setup(){
  Serial.begin(9600);
}

void loop(){
  x = 0-10;
  y = abs(x);
  Serial.println(y);
  delay(500);
}
```

注意

- abs()はマクロとして実装されているため、引数に他の関数を使わないようにすること。そのような使い方をすると、意図しない結果になることがある。
- abs(a++);　// このような使い方は意図しない結果を引き起こすので避ける。
- a++;
- abs(a);　// 他の計算や関数はabs()の外で実行する

constrain()

- 数値をある範囲に制限する。

- constrain(x, a, b);
- x　範囲を制限したい数値(任意の型)
- a　範囲の下限値(任意の型)
- b　範囲の上限値(任意の型)

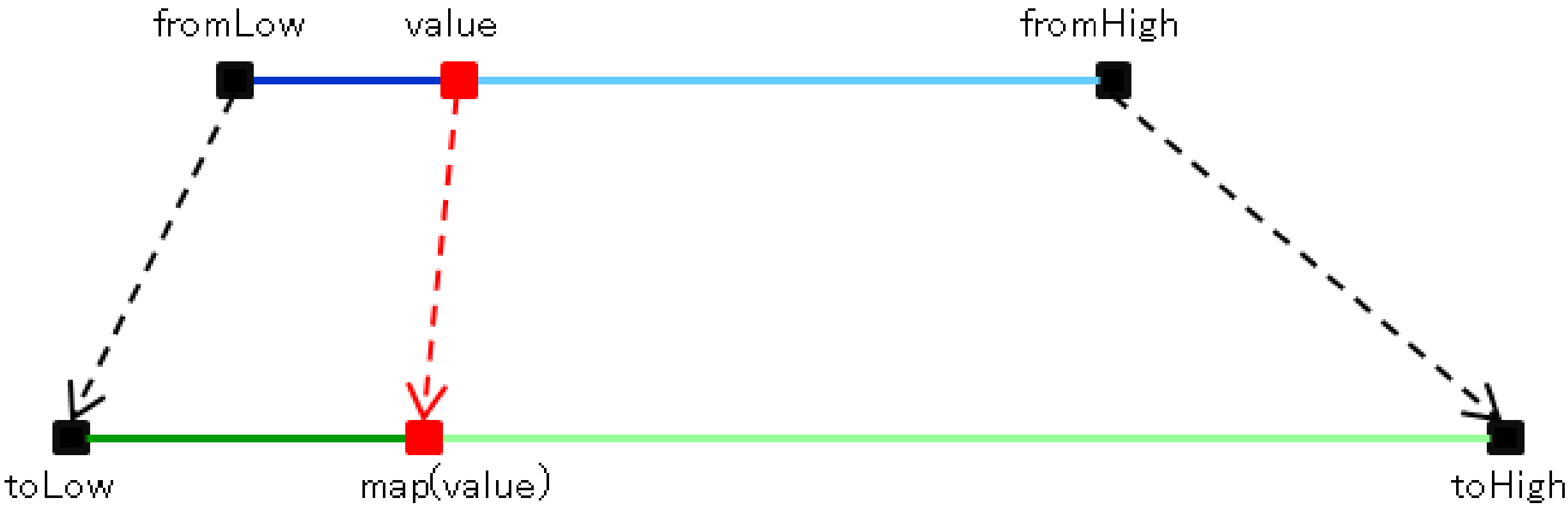
- xが、aとbとの間にあるときはｘ、ｘがaより小さいときはa、ｘがｂより大きいときはbを返す。

使用例

- sensVal = constrain(sensVal, 10, 150);
- // sensorの値の範囲を10から150に制限する。

map()

- ある範囲の数値を別の範囲の数値に対応づける。すなわち、fromLowはtoLowに、fromHighは、toHighに、fromLowとfromHighの間の数値はtoLowとtoHighの間の数値に対応づけられる。
- 範囲外の値を意識して使う場合もあるし有効でもあるので、変換する値は指定された範囲に限定しない。範囲を制限する必要がある場合は、constrain()関数を、この関数の前後で利用すればいい。
- 下限値は上限値よりも大きくてもいいし、上限値は下限値より小さくてもいい。これにより、例えば、値の範囲を逆転させることができる。



(value - fromLow) : (fromHigh − value) = (map − toLow) : (toHigh − map) となるように変換

- y = map(x, 1, 50, 50, 1);
- 負の値を使うこともでき,以下のような使い方も可能である。

- y = map(x, 1, 50, 50, -100);
- map()関数は整数で計算をするため、計算結果が小数になっても、小数は生成しない。小数点以下は切り捨てられる。四捨五入をしたり平均をとったりはしない。

- long map(value, fromLow, fromHigh, toLow, toHigh);
- value　　対応させる値
- fromLow 対応前の値の下限
- fromHigh 対応前の値の上限
- toLow　　対応後の値の下限
- toHigh　　対応後の値の上限

- 対応付けられた値が返される。

使用例

- /* analogRead()で読み取った10ビット(0から1023)の値を8ビット(0から255)に対応付ける */
- void setup() {}
-
- void loop()
- {
- int val = analogRead(0);
- val = map(val, 0, 1023, 0, 255);
- analogWrite(9, val);
- }

- 興味のある人向け。この関数のすべてのソースは以下の通り。

- long map(long x, long in_min, long in_max, long out_min, long out_max){
- 　return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
- }

組み合わせ

- constrainとmapを組み合わせるとこのようなことができる。

- 次の例は可変抵抗から入力した値を0から180に変換したものである。

```
#include <Servo.h>

#define ana A0
#define ser 9
int check;
Servo myservo;

void setup(){
  myservo.attach(ser);
  Serial.begin(9600);
}

void loop(){
  check = analogRead(ana)/4;
  check = map(check, 0, 255, 0, 180);
  check = constrain(check, 0, 180);
  myservo.write(check);
  Serial.println(check);
}
```

pow()

- 数値のべき乗を計算する。小数も扱うことができるので、値や曲線の指数写像を作る際に便利である。

- double pow(base, exponent);
- base　　数値
- exponebt 指数

- 数値のべき乗が返される。

sqrt()

- 数値の平方根を計算する。

- double sqrt(x);
- x　　数値

- 数値の平方根が返される。

三角関数

- sin
- cos
- tan

sin()

- 角度(単位はラジアン)の正弦を計算する。結果は、-1から1の間である。

- sin(rad);
- rad 角度(単位はラジアン)

- 角度の正弦を返す。
- 変数に格納する場合はdoubleで宣言する。

cos()

- 角度(単位はラジアン)の余弦を計算する。結果は、-1から1の間である。

- cos(rad);
- rad 角度(単位はラジアン)

- 角度の余弦を返す。
- 変数に格納する場合はdoubleで宣言する。

tan()

- 角度(単位はラジアン)の正接を計算する。結果は、-∞から∞の間である。

- tan(rad);
- rad 角度(単位はラジアン)

- 角度の正接を返す。
- 変数に格納する場合はdoubleで宣言する。

ランダム関数

- randomSeed
- random

randomSeed()

- randomSeed()は疑似乱数生成器を初期化し、疑似乱数系列の任意の地点から開始させる。この系列はとても長く、乱雑で、いつも同じである。
- 以降のスケッチの実行において、random()によって生成される値列が異なることが重要なのであれば、何も接続していないピンをanalogRead()で読んだ値のような、とても乱雑な値をrandomSeed()に与えて、乱数生成器を初期化すること。
- 逆に、同じ疑似乱数系列を利用することが有効な場合もある。これは、乱数系列を利用する前に、randomSeed()を同一の値を使って呼び出すことで実現できる。

- randomSeed(seed);
- seed　　疑似乱数系列を初期化するための種
  (使わないアナログピン)

- 戻り値はない

使用例

```
long randNumber;

void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop(){
  randNumber = random(300);
  Serial.println(randNumber);

  delay(50);
}
```

- 引数のseedを0にして、randomSeed()を呼び出しても、何も起こらない。

random()

- 疑似乱数を生成する。

- random(max);
- random(min, max);
- min 生成させる乱数の下限値(その値を含む)
- max生成させる乱数の上限値(その値を含まない)

- minからmax-1までの乱数値が返される。

- random(30);なら0から29の範囲
- random(0, 101);なら0から100の範囲

注意

- 以降のスケッチの実行において、random()によって生成される値列が異なることが重要なのであれば、何も接続していないピンをanalogRead()で読んだ値のような、とても乱雑な値をrandomSeed()に与えて、乱数生成器を初期化する。

- 逆に、同じ疑似乱数系列を利用することが有効な場合もある。これは、乱数系列を利用する前に、randomSeed()を同一の値を使って呼び出すことで実現できる。

使用例

- long randNumber;
-
- void setup(){
- Serial.begin(9600);
-
- // アナログピンの0番に何もつながってなければ、スケッチを起動するたびに
- // でたらめな値が randomSeed()にわたされて、異なる疑似乱数列を生成する。
- randomSeed(analogRead(0));
- }

```
void loop() {
  // 0から299までの乱数を表示する
  randNumber = random(300);
  Serial.println(randNumber);

  // 10から19までの乱数を表示する
  randNumber = random(10, 20);
  Serial.println(randNumber);

  delay(50);
}
```

- minを省略した場合は、min=0とみなされる。
- minがmax以上の場合は、常にminが返される。

終わりに

- 今回紹介したものはほかのプログラム言語ではあまり見ないものだと思われる。
- 積極的に利用する必要はないと思われるが知っておくと処理が簡単になったりするので、使いたい人は使ってみよう。
- 次回は関数を作る（void）を取り扱う。

課題

- 次の2つをTinkerCADを使ってシミュレーションしてみよう。

課題1

- constrainとmapを組み合わせて可変抵抗からの入力値を自由に変化させてシリアルで表示させてみよう。
- 可変抵抗器の入力値はデフォルトでは0から1023。

- 0から100だけではなく、0から-100も表示させてみよう。

課題2

- ランダム関数を使用して乱数をシリアルに表示させてみよう。

シリアルで画面に映すsetup

- void setup(){
- Serial.begin(9600);
- }

シリアルで画面に映すloop

- void loop(){
- //プログラム
- Serial.println(出力する変数名);
- }