# 講習会

# Arduino

3回目

ディジタル入力と出力

# ディジタルとは

- デジタル量とは、離散量（とびとびの値しかない量）のこと。
- 連続量を表すアナログと反対の概念である。
- 工業的には、状態を示す量を量子化・離散化して処理（取得、蓄積、加工、伝送など）を行う方式のことである。

- Ardinoの出力では0が出力なし（0 V)、1が出力あり（5 V）となる。

# ディジタルの概要

- データの数値化にあたっては量子化を行い、整数値で表現するのが一般的である。例えば、上昇中の位置では、階段の何段目かがデジタルで、坂道中の位置がアナログである。整数で表現するか、実数で表現するかの違いがある。デジタルでは、データ量を離散的な値（離散量）として表現することになり、それらの中間の量は誤差を含んだ隣の離散量で表現する。この誤差は適切な量子化を行うことで実用上影響のない範囲にすることができ、データ量に比例したアナログ量を用いるのとほぼ等価な処理が提供できる。
- 今日のコンピュータの主流であるデジタルコンピュータでは、0と1だけからなる2進数を物理的な表現形式（電圧の高・低など）として用いるので、デジタルは0と1から成るという説明がよくなされる。しかし、はっきりと区別できる2以上の状態で表現されているデータは、どれもデジタルと呼ぶことができる。

# ディジタルの特徴

- デジタルデータは、離散値として数値化しているので、アナログデータと比べて短期的には劣化しにくい特性をもつ。伝送・記録再生などを行う場合、デジタル量もアナログ量と同様に電圧・電流などの電気信号に置き換えて取り扱われる。外乱が生じて信号にノイズが混入した場合、アナログ処理では特別な処理を行わない限り信号に混じったノイズを取り除くことが困難である。これに対しデジタル処理では、数値は離散化してあって中間値をもたないので、ノイズによって生じた誤差が一定量以下ならばそれを無視でき、数値データを劣化する前の値に復元することができる（例えば、データが整数表現の場合、1がノイズによって0.8や1.2に変化しても1と認識させられる）。
- 実際の記録・伝送などではノイズなどの影響が無視できず、元のデータと異なるデータが再生されてしまうこともある（1が0.4または1.6に変化すると、異なる値0または2として再生される）。しかし、データをあらかじめ誤り訂正符号などを使って冗長化しておくと、途中で劣化しても自動的に修復したり、誤りの発生を検出して再送を要求したりすることができ、信頼性の高い処理を提供することが可能になる。

- 0か1の2つしかない。
- 小数点という概念はない。
- 連続的ではない。
- 小数点1桁目を四捨五入するという考えでよい。
  （0.4以下なら0、0.5以上なら1）

# デジタル処理

- デジタル化処理
- デジタル処理の適用

# デジタル化処理

- アナログデータをデジタルデータに変換することを「デジタル化する」、「デジタイズする」などという。

# デジタル処理の適用

- デジタルデータをそのまま扱う場合（単純なリニアサンプリング）について述べる。
- 実際のデジタル処理では、2進数1桁をビットとし、8ビットなどのまとまった単位をオクテットまたはバイトとして取り扱い、さらにそのまとまりをワードという単位として取り扱うことが多い。これは処理装置や記憶装置の語長に合わせて効率よく使えるようにするためである。
- デジタルデータにおいては、表現可能な数値範囲を超えたり、最小値に近い数値を扱ったりする際には注意が必要である。
- アナログ処理では、多少入力電圧が規定より超過しても影響がないか、わずかな影響で済む場合もある。しかしデジタル処理では、定義された最大値を超えた場合には桁あふれ（オーバーフロー）となり、値は量子化誤差が無視できず、S/N比の劣化として現れることが多い。また、最小値に近い数値以後の演算処理の結果は保証されない。さらに、数値計算の際に不用意な処理手順による桁落ちが生じ、著しい有効桁数の減少を招くこともあるので、注意を要する。

# デジタル処理の符号化

様々な分野でそれぞれ適切な表現形式を用いてデータを符号化している。

- 数値は、整数や浮動小数点型、固定小数点型などとして扱える。
- 文字は、文字コードで文字とコードを対応させることができる。
- 音声は、PCMなどでデジタル化できる。楽譜情報を電子化したものはMIDI、MMLなど。
- 絵、映像は、光をRGBなど色の成分に分解し、各色の明るさなどを数値化する。
- 図形は、ベクタ形式による。この形式は、狭義には線分の始点と終点の座標を数値で記録する。広義には、各種の図形に対して、例えば円なら、「図形コード＝円、中心座標、半径」を記録する。これらのデータからの例えば円を描くことは図形表示ソフトウェアに任せる。

# ディジタルとして使えるピン

- D0~D13（ディジタルピン）
- A0~A5（D14~D19）

（アナログ入力ピンだがpinModeで設定すればディジタルピンとして使用可能）

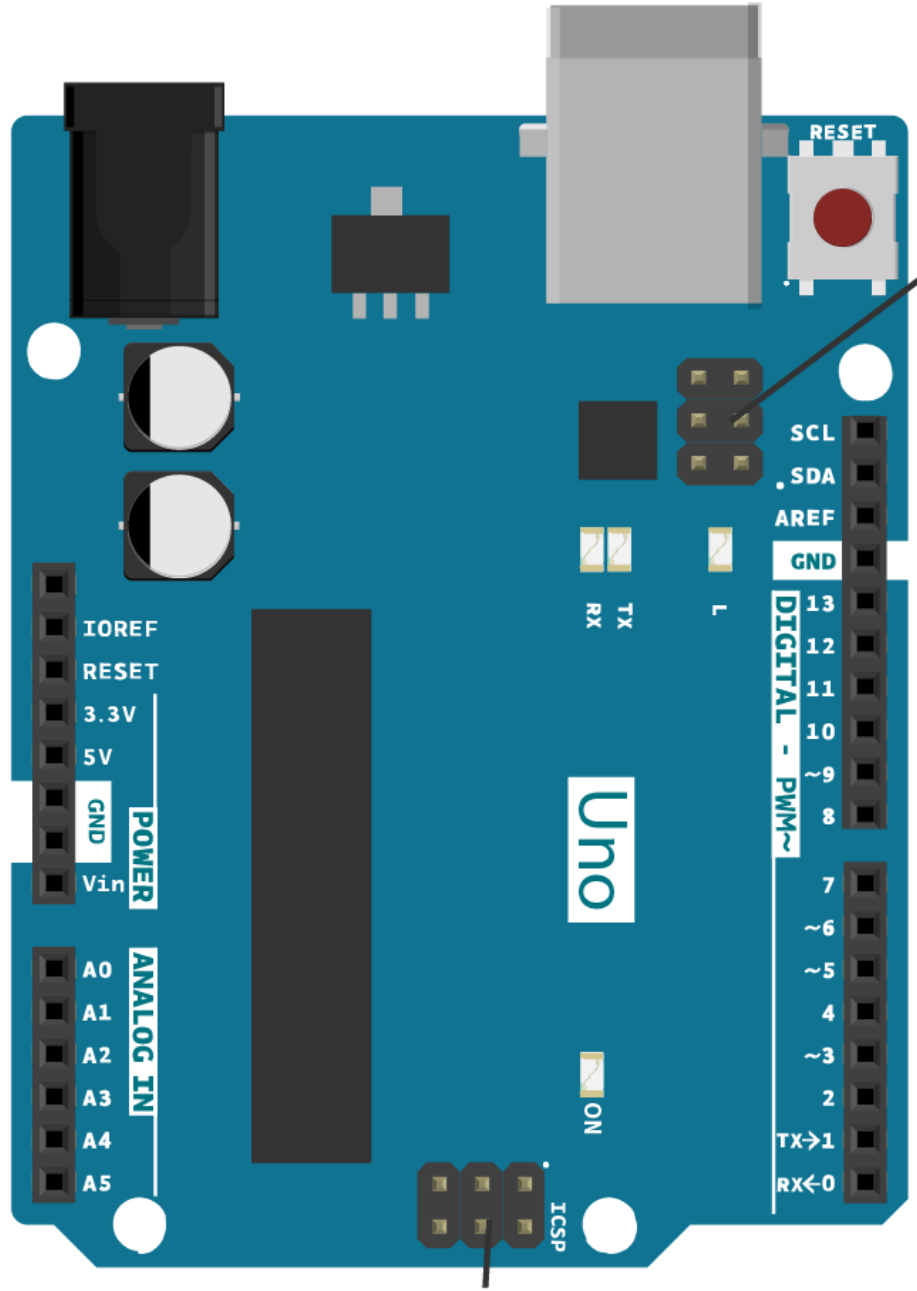
上記は出力でも入力でも可能

IORef: 5V

Vin: 7-12V DC max.

Serial: Serial is attached to pins 0 and 1, and to the USB-Serial micrcontroller on board.

The Uno has a second microcontroller on board to handle USB-to-serial communications.
This is the ICSP header for that microcontroller.




IORef
Reset
+3.3V
+5V
Gnd
Gnd
Vin

ADC0 GPIO14
ADC1 GPIO15
ADC2 GPIO16
ADC3 GPIO17
SDA ADC4 GPIO18
SCL ADC5 GPIO19

Comm. ADC GPIO

ICSP:

| | | |
|---|---|---|
| Reset | SCK | MISO |
| Gnd | MOSI | +5V |

| GPIO | ADC | Comm. | PWM | Interrupts | |
|---|---|---|---|---|---|
| GPIO18 | ADC4 | SDA | | | |
| GPIO19 | ADC5 | SCL | | | |
| | AREF | | | | |
| Gnd | | | | | |
| GPIO13 | | SCK | | | LED |
| GPIO12 | | MISO | | | |
| GPIO11 | | MOSI | PWM11 | | |
| GPIO10 | | CS | PWM10 | | |
| GPIO9 | | | PWM9 | | |
| GPIO8 | | | | | |
| GPIO7 | | | | | |
| GPIO6 | | | PWM6 | | |
| GPIO5 | | | PWM5 | | |
| GPIO4 | | | | | |
| GPIO3 | | | PWM3 | INT1 | |
| GPIO2 | | | | INT0 | |
| GPIO1 | | TX | | | |
| GPIO0 | | RX | | | |

GPIO ADC Comm. PWM Interrupts

# ディジタル処理の関数

- pinMode()

# pinMode()

- 指定したピンを、入力に利用するのか出力に利用するのかを設定する。ピンの機能に関する詳細は、digital pins(このリンクはオリジナルの英語ページです)を参照。
- Arduino 1.0.1からは、モードにINPUT_PULLUPを指定することで内部プルアップ抵抗を有効にすることができる。さらに、モードにINPUTを指定した場合は、内部プルアップを明示的に無効にする。
- pinMode()を使ってピンのモードを変えると、ピンの電気的な振る舞いが変わる。

- pinMode(pin, mode);

- pin　設定するピンの番号
- mode　INPUTもしくはOUTPUT、INPUT_PULLUP

# デジタルピンのモードを定義する

- INPUT
- INPUT_PULLUP
- OUTPUT

# INPUTと設定されたピン

- pinMode()によってINPUTと設定されたArduinoのピンは、高インピーダンス状態となる。この状態は、例えば、100MΩの抵抗が直列接続されていてほとんど電流が流れないような状態と同様である。センサの値を読むには都合がいいが、LEDに電流を供給するのには向いていない。
- スイッチを接続したピンをINPUTに設定し、そのスイッチが開いた状態のときにスイッチの値を読むと、そのピンは浮いている状態なので、読み取った結果は不定である。切断状態のスイッチの値を正しく読むためには、プルアップ抵抗もしくはプルダウン抵抗を用いる。この抵抗の目的は、切断状態のスイッチを確定した状態にすることである。10kΩの抵抗がよく用いられる。これは、浮いた状態を確実に防ぐために十分小さく、かつ、スイッチが閉じた状態に多くの電流を流さないために十分大きいためである。
- プルダウン抵抗を利用すると、スイッチが開いている場合LOWとなり、スイッチを閉じるとHIGHになる。
- プルアップ抵抗を利用すると、スイッチが開いている場合HIGHとなり、スイッチを閉じるとLOWになる。

# INPUT_PULLUPと設定されたピン

- Arduinoで使われているAtmegaチップはプルアップ抵抗(内部で電源に接続されている抵抗)を持っており、利用可能である。外部のプルアップレジスタの代わりに内蔵のプルアップレジスタを使いたい場合は、pinMode()でINPUT_PULLUPを設定すればいい。
- INPUTもしくはINPUT_PULLUPで入力に設定されているピンにマイナスの電圧や入力電圧(5Vもしくは3.3V)以上の電圧をかけると、ピンが壊れるかもしれない。

# OUTPUTと設定されたピン

- pinMode()によってOUTPUTと設定されたピンは、低インピーダンス状態となる。これは、他の回路に十分な量の電流を供給できることを意味する。Atmegaのピンは40mAまでの電流を他のデバイスや回路に供給したり吸い込むことができる。これは、LEDに電流を供給するには便利だがセンサの値を読むには不都合だ。 OUTPUTと設定されたピンは、接地や5V電源に直接接続されると壊れる可能性がある。Atmegaのピンから供給される電流はほとんどのリレーやモーターを駆動するには不十分なので、接続のための回路が必要となる。
- 出力に設定されているピンに電圧をかけると、ピンが壊れるかもしれない。

```
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
        digitalWrite(ledPin, LOW); // LEDを消す
        delay(1000); // 1秒待つ
}
```

# 注意

- アナログピン(A0、A1…として表記される)もデジタルピンとして利用できる。

# LED_BUILTIN

- ほとんどのArduinoボードには抵抗と直列に接続されたLEDが接続されたピンがある。定数LED_BUILTINは、ボード上のLEDが接続されているピンの番号である。ほとんどのボードではLEDは13番ピンに接続されている。

- pinMode(LED_BUILTIN,OUTPUT);
- digitalWrite(LED_BUILTIN,HIGH);

# ディジタル入力の関数

- digitalRead()

# digitalRead()

- 指定したデジタルピンから、HIGHもしくはLOWの値を読み取る。

- digitalRead(pin);
- pin 値を読み取るデジタルピンの番号

- HIGHもしくはLOWが返ってくる

- int ledPin = 13; // デジタルピンの13番に接続されているLED
- int inPin = 7;  // デジタルピンの7番に接続されている押しボタン
- int val = 0;    // 読み取った値を格納する変数
-
- void setup(){
- pinMode(ledPin, OUTPUT);     // デジタルピンの13番を出力(OUTPUT)に設定する
- pinMode(inPin, INPUT);       // デジタルピンの7番を入力(INPUT)に設定する
- }
-
- void loop(){
-  val = digitalRead(inPin);     // 入力ピンから値を読み込む
-  digitalWrite(ledPin, val);   // 読み取った値をLEDを接続したピンに出力する
- }

# 注意

- ピンに何も接続していないときは、digitalRead()は、HIGHかLOWを不規則に返却する。
- アナログピン(A0、A1…として表記される)もデジタルピンとして利用できる。

# ディジタル出力の関数

- digitalWrite()

# digitalWrite()

- デジタルピンにHIGHもしくはLOWの値を出力する。
- ピンが、pinMode()によって、出力(OUTPUT)に設定されているとき、ピンの電圧が引数に応じた値に設定される。引数にHIGHを設定した場合は5V(3.3Vのボードの場合は、3.3V)、引数にLOWを指定した場合は、0V（接地）となる。
- ピンが入力(INPUT)に設定されているときに、digitalWrite()によってHIGHを出力すると、内部のプルアップ抵抗が有効になる。また、LOWを出力すると、プルアップ抵抗が無効になる。ただし、内部のプルアップ抵抗を有効にするには、pinMode()でINPUT_PULLUPを設定することが推奨されている。
- 注意：pinMode()でピンをOUTPUTにせずにLEDをピンに接続し、digitalWrite(HIGH)を呼ぶと、LEDは暗く光る。pinMode()を明示的に呼ばなければ、digitalWrite()は内部のプルアップ抵抗が有効になり、電流制限抵抗のようにふるまう。

- digitalWrite(pin, value);

- pin　設定するピンの番号
- value　HIGH　もしくは　LOW

```
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
         digitalWrite(ledPin, LOW); // LEDを消す
         delay(1000); // 1秒待つ
}
```

# ピンのレベルを定義する

デジタルピンから値を読んだり、値を書いたりするとき、ピンが出力する値もしくはピンに入力する値は、

- HIGH
- LOW

の2種類だけである。

# HIGH

- (ピンに関連した)HIGHの意味は、ピンがINPUTに設定されているのかOUTPUTに設定されているのかによって少し異なる。
- ピンがpinMode()によってINPUTに設定 されている場合に、digitalRead()で値を読むと、以下の場合、Arduino（Atmega）は、HIGHを出力する。

- そのピンに3V以上の電圧がかかっている場合(5Vのボード)
- そのピンに2V以上の電圧がかかっている場合(3.3Vのボード)

- ピンがpinMode()によってINPUTに設定された後、digitalWrite()でHIGHに設定すると、内部の20キロオームのプルアップ抵抗を設定する。これにより、外部 の回路によりLOWにされるまでは、その入力ピンはHIGHとなる。これは、INPUT_PULLUPの動作原理と同じである。

- ピンがpinMode()によってOUTPUTに設定されていて、digitalWrite()によって HIGHを出力すると、ピンには以下の電圧が出力される。
- 5V(5Vのボード)
- 3.3V(3.3Vのボード)
- この状態では電流を供給する ことができる。例えば、抵抗を通して接地もしくはLOWに設定されているOUTPUT状態のピンに接続されているLEDを光らせることができる。

# LOW

- LOWの意味も、ピンがINPUTに設定されているのかOUTPUTに設定されているのかによって異なる。ピンがpinMode()によってINPUTに設定されている場合 に、digitalRead()で値を読むと、以下の場合、Arduinoは、LOWを出力する。

- そのピンに3V未満の電圧がかかっている場合(5Vのボード)
- そのピンに2V以上の電圧がかかっている場合(3.3Vのボード)

- ピンがpinMode()によってOUTPUTに設定されていて、digitalWrite()によってLOWを出力すると、ピンは0V(5Vのボードも3.3Vのボードも)となる。この状態では電流を吸い込むことができる。例えば、抵抗を通して+5V（もしくは+3.3V)に接続されているLEDを光らせることができる。

# 課題

- 次のプログラムを訳してみる

```
int ledPin = 13;
int inPin = 7;
int val = 0;
void setup(){
pinMode(ledPin, OUTPUT); pinMode(inPin, INPUT);
}

void loop(){
  val = digitalRead(inPin);
  digitalWrite(ledPin, val);
}
```

```
int ledPin = 13;
int inPin = 7;
int val = 0;

void setup(){
pinMode(ledPin, OUTPUT);
pinMode(inPin, INPUT);
}

void loop(){
  val = digitalRead(inPin);
  digitalWrite(ledPin, val);
}
```

# 終わりに

- 次回はアナログ入力と出力を取り扱う