講習会

Arduino

10回目

シリアル（パソコンに表示）

目的

- シリアル通信を利用してArduinoの処理によって生まれた数値をパソコンに表示する。
- Arduinoで作ったタイマーをProcessingを利用して表示する。

シリアル通信とは

- 電気通信において伝送路上を一度に1ビットずつ、逐次的にデータを送ることをいう。
- また、コンピュータにおいては、バス上を一度に1ビットずつ、逐次的にデータを送ることをいう。
- シリアル通信は長距離の通信やコンピュータネットワークで使われる。これは、電線の本数を減らすためであり、同時にパラレル通信を長距離で使うと同期が困難になるためである。バスは、一段上の速度を実現する際にシリアルバスとして実現され、技術が成熟してくるとその速度でパラレルバスが可能になるということを繰り返してきた。

- 少ない線（RXとTXの2本）で通信をするため一度に多くのデータ通信ができないけど、扱いは簡単である。
- Arduino Unoでは1つしかシリアル通信のポートがないけどMegaやESP32には3つある。それぞれシリアル通信の設定ができるので利用すると複数と通信することが可能。
- 例としてはProcessingでタイム表示しながらサーマルプリンターシールドで印刷をする等。

Processingとは

- キャセイ・レアスとベンジャミン・フライによるオープンソースプロジェクトであり、かつてはMITメディアラボで開発されていた。
- 電子アートとビジュアルデザインのためのプログラミング言語であり、統合開発環境である。
- アーティストによるコンテンツ制作作業のために、詳細な設定を行う関数を排除している。 視覚的なフィードバックが即座に得られるため、初心者がプログラミングを学習するのに適しており、電子スケッチブックの基盤としても利用できる。Java を単純化し、グラフィック機能に特化した言語といえる。

関数紹介

- begin()
- available()
- print()
- println()
- read()
- write()
- serialEvent()

begin()

- シリアルデータを送信する際の通信速度をbps(ビット/秒)で設定する。PCと通信するためには、300、1200、2400、4800、9600、14400、19200、28800、38400、57600、115200のどれかの値を利用する。特定の通信速度を要求する機器と通信するために、他の通信速度を設定することもできる。

- オプションの2番目の引数(config)は、データ長とパリティ、ストップビットを設定する。デフォルト(指定しない場合)は、8ビット・パリティなし・ストップビット1である。

- Serial.begin(baud);
- baud　　通信速度(単位はbps)
- void setupの中に書く

Arduino Unoの場合

- void setup() {
- Serial.begin(9600);　　//シリアルポートを開き、転送速度を9600bpsに設定する。
- }
-
- void loop() {}

Arduino Megaの場合

```
// Arduino Megaですべてのシリアルポートを使う
// (Serial, Serial1, Serial2, Serial3),
// それぞれ異なる通信速度を設定する
void setup(){
  Serial.begin(9600);
  Serial1.begin(38400);
  Serial2.begin(19200);
  Serial3.begin(4800);

  Serial.println("Hello Computer");
  Serial1.println("Hello Serial 1");
  Serial2.println("Hello Serial 2");
  Serial3.println("Hello Serial 3");
}

void loop() {}
```

available()

- シリアルポートから読み取り可能なバイト数(文字数)を取得する。これは、すでにArduinoボードに到着していて、シリアル通信用の受信バッファに格納されているデータである。シリアル通信用の受信バッファは64バイトまで格納することができる。available()は、Streamユーティリティクラスを継承している。

- Serial.available();

Arduino Unoの場合

```
int incomingByte = 0;   // 受信用

void setup() {
        Serial.begin(9600);     // /シリアルポートを開き、転送速度を9600bpsに設定する。
}

void loop() {
        // データを受信したときだけデータを送信する。
        if (Serial.available() > 0) {
                // 受信データを読む
                incomingByte = Serial.read();

                // 受信データを書く。
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
        }
}
```

Arduino Megaの場合

```
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
}

void loop() {
  // port 0 から読んで、port 1 に書く
  if (Serial.available()) {
    int inByte = Serial.read();
    Serial1.print(inByte, BYTE);
  }
  // port 1 から読んで、port 1 に書く
  if (Serial1.available()) {
    int inByte = Serial1.read();
    Serial.print(inByte, BYTE);
  }
}
```

print()

- シリアルポートにASCIIテキストとして可読文字を表示する。この関数には多くの呼び出し形式がある。数値はそれぞれの数字をASCII文字を使って表示される。浮動小数点は同様に数字をASCII文字を使って、デフォルトでは小数点以下第2位まで表示する。バイトは数値をASCII文字に変換して送信される。文字や文字列はそのまま送信される。

- Serial.print(78) は "78"
- Serial.print(1.23456) は "1.23"
- Serial.print('N') は "N"
- Serial.print("Hello world.") は "Hello world."

- オプションの第2引数は、底や形式を指定する。BIN(2進表記)、OCT(8進表記)、DEC(10進表記)、HEX(16進表記)が指定可能である。浮動小数点に対しては、この引数は、小数点以下第何位までを表示するかを指定する。

- Serial.print(78, BIN) は "1001110"
- Serial.print(78, OCT) は "116"
- Serial.print(78, DEC) は "78"
- Serial.print(78, HEX) は "4E"
- Serial.println(1.23456, 0) は "1"
- Serial.println(1.23456, 2) は "1.23"
- Serial.println(1.23456, 4) は "1.2346"

- F()を使って、フラッシュメモリ上の文字列をSerial.print()に送ることもできる。例えば、以下のように使う。

- Serial.print(F(“Hello World”))
- 1バイトの文字を送信する場合は、Serial.write()を使うこと。

- Serial.print(val);
- Serial.print(val, format);

- val　表示する値
- format　　表示する値が整数型の場合は底、浮動小数型の場合は小数点以下の表示桁数

- 表示したバイト数が返ってくる。
- 戻り値の利用はオプション。

```
//forループを使い、さまざまな形式で表示する。
    int x = 0;   // 変数
    void setup() {
      Serial.begin(9600);     // シリアルポートを開き、転送速度を9600bpsに設定する。
    }
    void loop() {
      // print labels
      Serial.print("NO FORMAT");      // ラベルを表示
      Serial.print("¥t");             // タブを表示
      Serial.print("DEC");
      Serial.print("¥t");
      Serial.print("HEX");
      Serial.print("¥t");
      Serial.print("OCT");
      Serial.print("¥t");
      Serial.print("BIN");
      Serial.print("¥t");
      for(x=0; x< 64; x++){    // ASCIIデータの一部
        Serial.print(x);       // 数字をASCIIテキストに変換して表示 -  "DEC"と同じ
        Serial.print("¥t");    // タブを表示
        Serial.print(x, DEC);  // 数字をASCIIテキストに変換して10進数表示
        Serial.print("¥t");    // タブを表示
        Serial.print(x, HEX);  // 数字をASCIIテキストに変換して16進数表示
        Serial.print("¥t");    // タブを表示
        Serial.print(x, OCT);  // 数字をASCIIテキストに変換して8進数表示
        Serial.print("¥t");    // タブを表示
        Serial.println(x, BIN);  // 数字をASCIIテキストに変換して2進数表示
        //                       "println"   を使っているので、改行も表示
        delay(200);           // 200ミリ秒待つ
      }
      Serial.println("");     // もう一つ改行を表示
    }
```

println()

- シリアルポートにASCIIテキストとして可読文字を表示する。指定した文字を表示した後、復帰（CR、ASCIIコード13, '¥r')と改行(LF、ASCIIコード10,'¥n')とを送信する。このコマンドは、指定した引数の最後にCR/LFを送信する以外は、Serial.print()と同様である。

- Serial.println(val);
- Serial.println(val, format);

- val　表示する値
- format　　表示する値が整数型の場合は底、浮動小数型の場合は小数点以下の表示桁数

- 表示したバイト数が返ってくる。
- 戻り値の利用はオプション。

```
/*
  アナログ入力
 0番ピンからアナログ値を読み、読み取った値を書き出す。
 created 24 March 2006
 by Tom Igoe
 */
int analogValue = 0;    // アナログ値を保持する変数
void setup() {
  // シリアルポートを開き、転送速度を9600bpsに設定する。
  Serial.begin(9600);
}

void loop() {
analogValue = analogRead(0); // read the analog input on pin 0:
  // print it out in many formats:
  Serial.println(analogValue);       // 10進数表示
  Serial.println(analogValue, DEC);  // 10進数表示
  Serial.println(analogValue, HEX);  // 16進数表示
  Serial.println(analogValue, OCT);  // 8進数表示
  Serial.println(analogValue, BIN);  // 2進数表示
  delay(10);  // 10ミリ秒待つ
}
```

read()

- 受信したシリアルデータを読む。read()は、Streamユーティリティクラスを継承している。

- Serial.read();
- 受信したシリアルデータの最初の1バイトが返ってくる。
- データがない場合は-1。

```
int incomingByte = 0;   // 受信用

void setup() {
        Serial.begin(9600);     // /シリアルポートを開き、転送速度を9600bpsに設定する。
}

void loop() {
if (Serial.available() > 0) {// データを受信したときだけデータを送信する。

    incomingByte = Serial.read(); // 受信データを読む

    Serial.print("I received: ");
    Serial.println(incomingByte, DEC); // 受信データを書く。

      }
}
```

write()

- バイナリデータをシリアルポートに書き込む。1バイトもしくはバイト列として送信される。数値を文字表現として送信したいときは、print()関数を利用すること。

- Serial.write(c);
- Serial.write(str);
- Serial.write(buffer, size);

- c　　送信する値(1バイト)
- str　バイト列として送信する文字列
- buf バイト列として送信する配列
- size 配列の大きさ

- 表示したバイト数が返ってくる。
- 戻り値の利用はオプション。

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

void loop(){
  Serial.write(45); // send a byte with the value 45

  int bytesSent = Serial.write("hello"); //send the string "hello" and return the length of the string.
}
```

serialEvent()

- データが利用可能な時に呼び出される関数を定義することができる。
- このデータを利用するときは、Serial.read()を利用すること。

- void serialEvent() { }

- この関数は、必要に応じて自分で定義する関数です。シリアル通信の受信バッファにデータがあるとき、loop()関数が終了して次のloop()を開始する前に呼び出されます。具体的には、以下のように実装されています。

- for (;;) {
- loop();
- if (serialEventRun) serialEventRun();
- }

シリアル通信を使って表示

- ここからは実際にパソコンやProcessingに表示する例を紹介する。

Arduinoからパソコン

- パソコンのシリアルモニタに表示をする。

- 例はセンサーの通過をシリアルモニタで確認
- 通過中は1、それ以外では0を返す。

```
#define sensor1 2
#define sensor2 17
#define sensor3 3
#define sensor4 18
int sensorin1;
int sensorin2;
int sensorin3;
int sensorin4;
//A0～A5→D14～D19 変換可能
volatile int sens1;
volatile int sens2;

void setup() {
  pinMode(sensor1, INPUT_PULLUP);
  pinMode(sensor2, INPUT_PULLUP);
  pinMode(sensor3, INPUT_PULLUP);
  pinMode(sensor4, INPUT_PULLUP);
  attachInterrupt(0, Sensor1, CHANGE);
  attachInterrupt(1, Sensor2, CHANGE);
  Serial.begin(9600);
}
```

```
void loop() {
  serial();
}

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

```
void serial() {//パソコンのシリアルモニタで表示する部分
  Serial.print("sensor1");
  Serial.print(" ");
  Serial.print(sens1);
  Serial.print(" ");
  Serial.print("sensor3");
  Serial.print(" ");
  Serial.println(sens2);
}
```

ArduinoからProcessingに送信

- ProcessingにArduinoで処理した数値を送信する

```
void proccesing() {
  timerIN();
  timerOUT();
  Serial.print("H");     // ヘッダ送信(先頭を示す文字)
  Serial.write(outTime_c);// OUTコース;センチ秒データ送信
  Serial.write(inTime_c);// INコースミリ秒データ送信
  Serial.write(outTime_m);// OUTコース;分データ送信
  Serial.write(outTime_s);// OUTコース;秒データ送信
  Serial.write(inTime_m); // INコース;分データ送信
  Serial.write(inTime_s); // INコース;秒データ送信
  Serial.print('¥n');
}
```

```
void timerIN() {
  pattern = button();
  switch (INTimepattern) {
    case 0:
      ltimemillis = millis();
      ltime2 = ltimemillis - ltime1;
      if (ltime2 > 999) {
        ltime1 = ltimemillis;
        lcnts += 1;
      }
      if (lcnts > 59) {
        lcnts = 0;
        lcntm += 1;
      }
      if (pattern == UP) {
        secondITIME = ltime2;
        secondlcntS = lcnts;
        secondlcntM = lcntm;
        inTime_c = secondITIME / 10;
        inTime_s = secondlcntS;
        inTime_m = secondlcntM;
      }
      lcntIN = 0;
      break;
```

```
    case 2:
      lcntIN++;
      if (pattern == LEFT) {
        lcntIN = 0;
        ltime2 = 0;
        INTimepattern = 4;
      }
      break;

    case 4:
      lcntIN++;
      ltimemillis = millis();
      ltime1 = ltimemillis;
      ltime2 = ltimemillis - ltime1;
      lcnts = 0;
      lcntm = 0;
      secondITIME = 0;
      secondIcntS = 0;
      secondIcntM = 0;
      inTime_c = secondITIME;
      inTime_s = secondIcntS;
      inTime_m = secondIcntM;
      if (lcntIN > 10 && pattern == DOWN) {
        ltime1 = ltimemillis;
        INTimepattern = 0;
      }
      break;
   }
  }
```

ProcessingでArduinoからのデータを受信

- // シリアルライブラリを取り入れる
- import processing.serial.*;

- // myPort（任意名）というインスタンスを用意
- Serial myPort;

- // シリアルポートの設定
- // Arduinoが接続されているシリアルポートにあわせて書き換える
- myPort = new Serial(this, "COM3", 250000);
- }

- thisにしておく
- Arduinoで使用したポートと合わせる
- Arduinoでセットした速度と合わせる

```
void serialEvent(Serial p){
  if ( myPort.available() >= 7 ) {  // ヘッダ + 時間データ で合計 7 バイト
    if ( myPort.read() == 'H' ) {   // ヘッダ文字を見つけたところから読み取る
      receivedData[0] = myPort.read(); // OUT:分読み込み
      receivedData[1] = myPort.read(); // OUT:秒読み込み
      receivedData[2] = myPort.read(); // OUT:センチ秒読み込み
      receivedData[3] = myPort.read(); // IN:分読み込み
      receivedData[4] = myPort.read(); // IN:秒読み込み
      receivedData[5] = myPort.read(); // IN:センチ秒読み込み
    }
  }
}
```

- 変数宣言
- int NUM = 6; //センサーの数
- int[] receivedData = new int[NUM];　//センサーの値を格納する配列

- // 受信バッファから描画用変数に格納
- secondIcntM = receivedData[2];
- secondIcntS = receivedData[3];
- secondITIME = receivedData[0];
- inTime_m = receivedData[4];
- inTime_s = receivedData[5];
- inTime_c = receivedData[1];

Arduinoとprocessingの比較

```
void proccesing() {
  timerIN();
  timerOUT();
  Serial.print("H");     // ヘッダ送信(先頭を示す文字)
  Serial.write(outTime_c);// OUTコース;センチ秒データ送信
  Serial.write(inTime_c);// INコースミリ秒データ送信
  Serial.write(outTime_m);// OUTコース;分データ送信
  Serial.write(outTime_s);// OUTコース;秒データ送信
  Serial.write(inTime_m); // INコース;分データ送信
  Serial.write(inTime_s); // INコース;秒データ送信
  Serial.print('¥n');
}
```

```
void serialEvent(Serial p){
  if ( myPort.available() >= 7 ) {  // ヘッダ + 時間データ で合計７バイト
    if ( myPort.read() == 'H' ) {   // ヘッダ文字を見つけたところから読み取る
      receivedData[0] = myPort.read(); // OUT:分読み込み
      receivedData[1] = myPort.read(); // OUT:秒読み込み
      receivedData[2] = myPort.read(); // OUT:センチ秒読み込み
      receivedData[3] = myPort.read(); // IN:分読み込み
      receivedData[4] = myPort.read(); // IN:秒読み込み
      receivedData[5] = myPort.read(); // IN:センチ秒読み込み
    }
  }
}
```

- Arduino側　Processingにデータ送信
- Serial.write(outTime_c);// OUTコース;センチ秒データ送信
- Processing側　Arduinoからデータ受信
- receivedData[0] = myPort.read(); // OUT:分読み込み
- secondITIME = receivedData[0];//受信したデータを変数に格納

- 上から順番に送信
- Serial.write(outTime_c);// OUTコース;センチ秒データ送信
- Serial.write(inTime_c);// INコースミリ秒データ送信
- Serial.write(outTime_m);// OUTコース;分データ送信
- Serial.write(outTime_s);// OUTコース;秒データ送信
- Serial.write(inTime_m); // INコース;分データ送信
- Serial.write(inTime_s); // INコース;秒データ送信

- 上から順番に受信　コメントに意味はない
- receivedData[0] = myPort.read(); // OUT:分読み込み
- receivedData[1] = myPort.read(); // OUT:秒読み込み
- receivedData[2] = myPort.read(); // OUT:センチ秒読み込み
- receivedData[3] = myPort.read(); // IN:分読み込み
- receivedData[4] = myPort.read(); // IN:秒読み込み
- receivedData[5] = myPort.read(); // IN:センチ秒読み込み

- //対応した変数に数値を格納
- secondIcntM = receivedData[2];
- secondIcntS = receivedData[3];
- secondITIME = receivedData[0];
- inTime_m = receivedData[4];
- inTime_s = receivedData[5];
- inTime_c = receivedData[1];

- サンプルプログラムから確認することが可能

終わりに

- ArduinoからProcessingに数値を送ることでスタートゲートのタイマーをパソコンで表示するだけではなく本格的な表示をすることが可能となる。
- マイコンカーとは直接関係はないけど、スタートゲートの開発には必要不可欠なので使えるようにすること。
- 次回はシールド（ライブラリと基板）を取り扱う。

課題

- タイマーの値をProcessingを使用して表示する。
- タイマーに関しては今までの資料を見ること。
- 今回の資料を見ればProcessingに数値を送ることができる。

- Proccesing側のプログラムは課題ファイル1として提供する。