システム実験
基礎実験5レポート

6119019056 山口力也

2019/05/24 日提出

1　基礎実験第五回の概要

基礎実験第 5 回目の実験の目的, 実験した実験の概要, および理解した事柄を 100 200 字程度で説明せよ.

本実験では,Arduino の持つ通信機能のうちシリアル通信に関する実験を行う. シリアル通信は, 古くからコンピュータ間の通信に使用されてきた通信方式であり, 多くのマイコンんいしリアル通信機能が搭載されている. シリアル通信を用いることで, マイコンの外部に接続されたデバイスと決められたプロトコルに従って通信を行える. 以下に実験目標を示す.

- シリアル通信の原理を理解する.
- シリアル通信を使いこなす.
- マイコンとパソコン間でデータを送受信できる.
- センサのデータをマイコン・パソコンで処理できる.
- ディジタル I/O,AD 変換, 割り込みなどを統合的に扱える.

2　復習と確認

ブレッドボード上に回路を実装する際の注意点を説明せよ. また,C 言語によるプログラムとスケッチの違いについて説明せよ. さらに, シリアルモニタ使用に関する注意点を述べよ.

汎用的な回路を同じブレッドボード上に実装することが度々あるが, 関係のない回路と繋がらないように注意するべきである. また,LED などを実装する際は LED と電源の間に電流制限抵抗を用いることも忘れてはならない.C 言語によるプログラムとスケッチの違いについては, 実装する際にある.C 言語の場合実行ファイルを実行したときのみプログラムが実行されるが, スケッチ

の場合一度マイコンにスケッチを書き込むと電源を入れるたび書き込んだスケッチが動作する. そのため, 例えば単に電源を入れただけと思っていても前回書き込んだプログラムが実行されている可能性があるので注意する必要がある. シリアルモニタ使用に関しては, いくつか注意点がある. シリアル通信使用する際, コンピュータとのデータ転送レートを設定する必要がある.Arduinoではデフォルトで 9600bps が設定されているが, マイコン側の転送レートを変更した場合にはパソコン側のシリアルモニタの転送レート設定も同様に変更しないと正しくデータを受信できない.

3 回路実装

演習 2.5.3 において実装したブレッドボード配線図を報告せよ.
以下図 1 に実装したブレッドボード配線図を示す.

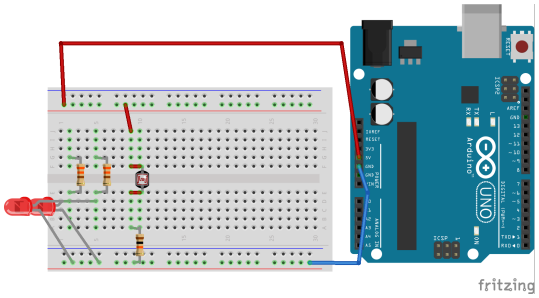

図 1: 演習 2.5.3 で実装した回路の配線図

4 マイコンの受信データに対応した LED の状態変化およびマイコンから LED の状態を送信

課題 2.5.1 において実装したスケッチを報告せよ. また, スケッチ作成において工夫した点を記せ. さらに, マイコンが受信した文字を数字に変換して処理する方法を提案せよ.
以下ソースコード 1 に実装したスケッチを示す.

ソースコード 1: 課題 2.5.1

```
1 const int LED_RED_PIN = 13; //LED(赤)を 13番ポートに定義
2 const int LED_YEL_PIN = 9; //LED(黄)を 9番ポートに定義
3 int output_red = LOW; //赤色LED への出力用
4 int output_yel = LOW;
5 void setup() {
6   pinMode(LED_RED_PIN,OUTPUT); //赤色LED を出力として設定
```

```
7   pinMode(LED_YEL_PIN,OUTPUT); //黄色LED を出力として設定
8   Serial.begin(9600); //シリアル通信開始:転送レート 9600
9 }
10
11 void loop() {
12   if(Serial.available()>0){ //パソコンからデータを受信
13     int recv = Serial.read();
14     recv -= 48; //数字に変換
15     Serial.println(recv);
16     if(recv == 0){ //'0'の場合
17       output_red = HIGH;
18       output_yel = LOW;
19     }
20     else if(recv == 1){ //'1'の場合
21       output_red = LOW;
22       output_yel = HIGH;
23     }
24     else if(recv == -38){ //改行コードが来た時
25       //何もしない
26     }
27     else{
28       output_red = HIGH;
29       output_yel = HIGH;
30     }
31     //Serial.println("I received!"); //パソコンへ応答を返す (
           送信)
32     Serial.print("LED1:");
33     Serial.println(output_red);
34     Serial.print("LED2:");
35     Serial.println(output_yel);
36     digitalWrite(LED_RED_PIN,output_red); //赤色
           LED の点灯・消灯
37     digitalWrite(LED_YEL_PIN,output_yel); //赤色
           LED の点灯・消灯
38   }
39 }
```

工夫した点は, 改行コードが来た時の対応をつくったところである. マイコンが受信した文字を数字に変換するには,ASCII テーブルから変換する必要がある. 例えばマイコンが 10 進数で”65” という文字を受信した場合,ASCII テーブルから変換されて,”A” という文字になる. 数字に変換したい場合は A という文字をもう一度”65” という文字に変換してやる必要がある.

5　温度センサを用いたマイコンの温度情報の PC への送信およびそのグラフ化

演習 2.5.6 において作成したグラフを報告せよ.
以下図 2 に作成したグラフを示す.

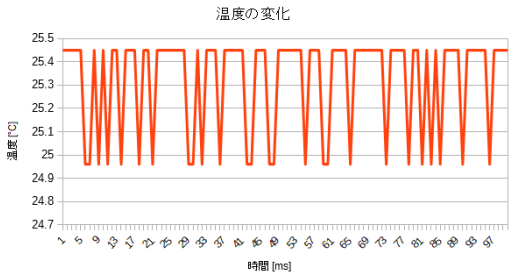

図 2: 演習 2.5.6 で作成したグラフ

6　millis 関数によるデータ解析区間の設定, 温度センサ情報の PC への送信および温度情報グラフ化

演習 2.5.7 において作成したグラフを報告せよ.
以下図 3 に作成したグラフを示す.

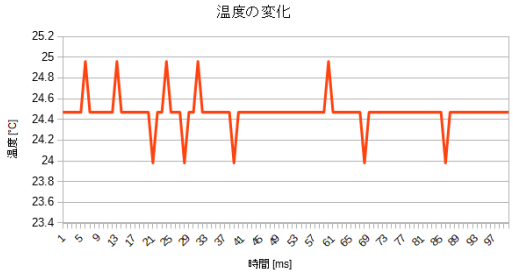

図 3: 演習 2.5.7 で作成したグラフ

7　温度センサを用いた解析済み温度情報のマイコンの PC への送信およびグラフ化

課題 2.5.2 で実装したスケッチおよび作成したグラフを報告せよ. また, 工夫した点を記せ. 演習 2.5.6 で作成されたグラフとの違いを説明せよ.

以下ソースコード 2 に実装したスケッチを示す.

ソースコード 2: 課題 2.5.2

```
40 unsigned long timePrev = 0;
41 int count = 0;
42 double average = 0;
43 double sum= 0;
44 void SendData(){
45   Serial.println(average);
46   count++;
47 }
48 void setup() {
49   Serial.begin(9600);
50   Serial.println("start!");
51   timePrev = millis();
52 }
53
54 void loop() {
55   if(count < 100){
56     double sum = 0; //合計格納用変数
57     long int i = 0; //合計測定回数格納用変数
58     while (1){//100ms 経つまで
59       unsigned long timeNow = millis(); //現在時間を格納
60       if(timeNow - timePrev <= 100){ //100ms 経つまで
61         int sensorValue = analogRead(A0); //A0 の値を格納
62         double vo = sensorValue*(5.0/1024.0);
63         double Temp = (vo*1000.0 - 600.0)/10.0;
64         sum += Temp;
65         i++;
66       }
67       else{
68         timePrev = timeNow;
69         break;
70       }
71    }
72    average = sum / i;
73    SendData();
74   }
75 }
```

工夫した点は,SendData という関数を作って, データを送る部分を分けて見やすくそして使いやすくしたところである.

また, 以下図 4 に作成したグラフを示す.

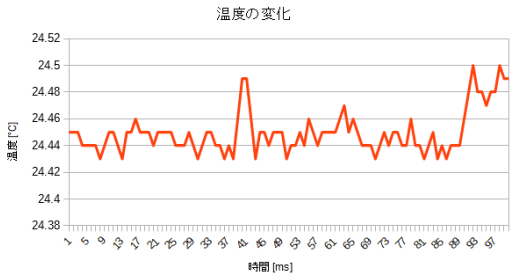

図 4: 課題 2.5.2 で作成したグラフ

演習 2.5.6 で作成された図 2 との違いは, データ解析区間を作ることでグラフが滑らかになっていることである. 平均値を取らずに, グラフを作ると何らかの影響で飛び出た値が出た時に, そのままその値を出力してしまうためがたがたのグラフになってしまう

8 照度センサによるマイコンから PC への解析済み温度情報の送信, グラフ化および解析結果に対応した LED の発光

課題 2.5.3 で実装したスケッチおよび作成したグラフを報告せよ. また, 工夫した点を記せ. さらに, データ解析 (区間平均の算出) の必要性について考察せよ.

以下ソースコード 3 に実装したスケッチを示す.

ソースコード 3: 課題 2.5.3

```
76 unsigned long timePrev = 0;
77 int count = 0;
78 double average = 0;
79 double sum = 0;
80 int output = 0;
81 const int LED_PIN = 9; //9番ポートをLED に設定
82 void SendData(){
83   Serial.println(average);
84   count++;
85 }
86 void setup() {
87   analogWrite(LED_PIN,OUTPUT);
88   Serial.begin(9600);
89   Serial.println("start!");
90   timePrev = millis();
```

```
91 }
92
93 void loop() {
94   if(count < 30){ //15秒経つまで(500ms × 30)
95     double sum = 0; //合計格納用変数
96     long int i = 0; //合計測定回数格納用変数
97     while (1){//500ms経つまで
98       unsigned long timeNow = millis(); //現在時間を格納
99       if(timeNow - timePrev <= 500){ //500ms経つまで
100        int sensorValue = analogRead(A0); //A0の値を格納
101        double vo = sensorValue*(5.0/1024.0);
102        double L = 222*vo;
103        sum += L;
104        i++;
105      }
106      else{
107        timePrev = timeNow;
108        break;
109      }
110    }
111    average = sum / i;
112    SendData();
113    output = int(average*255);
114    if(output > 255){
115     output = 255;
116    }
117    analogWrite(LED_PIN,output); //照度1で255(MAX)出力
118  }
119 }
```

工夫した点は, 出力の部分で事前に照度の最大が 1 付近であったことを確認したのでそのあたりを最大値として, また照度が 1 を超えた際の処理も入れておいところである. また, 以下図 5 に作成したグラフを示す.

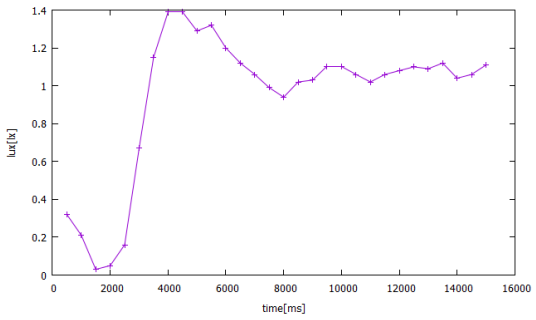

図 5: 課題 2.5.3 で作成したグラフ

データ解析の必要性について. データ解析区間があることで比較的にまとまった値がでるので, 突然変な値が何らかの理由で入ったとしても平均化されるためそこまでデータに支障がでない. そういう意味ではデータ解析区間を設けることは必要であると考えられる.

9 発展課題 2.5.1:マイコンのデータ送受信の確認

以下ソースコード 4 に実装したスケッチを示す.

ソースコード 4: 発展課題 2.5.1

```
120 void setup() {
121   Serial.begin(9600); //転送レート 9600に設定
122 }
123
124 void loop() {
125   if(Serial.available() > 0){ //パソコンからデータを受信
126     int receive = 0; //受信データ
127     int birthday[4] = {0}; //表示用配列
128     int kekka = 0;
129     int i = 0; //配列の要素カウント用
130     while(1){
131       if(Serial.available() > 0){ //受信したら
132         receive = Serial.read();
133         if( 47 < receive && receive <58){ //受信データが数字
              のASCII コードの時
134           receive -= 48; //数字に変換
135           birthday[i] = receive; //配列に入れる
136           i++; //配列の要素をずらす
137         }
138         if( receive == 10){ //改行が来たら
139           break; //ループを抜ける
140         }
141       }
142     }
143     birthday[2] += birthday[0]; //誕生日の 2桁目に誕生月の 2
          桁目を足す
144     if (birthday[3] + birthday[1] >= 10){//もし誕生日の 1桁
          目と誕生月の 1桁目の和が 10以上なら
145       birthday[2]++; //繰り上げで誕生日の 2桁目に 1足す
146       birthday[3] += birthday[1] -10; //誕生日の 1桁目は足し
          た数-10
147     }
148     else{ //誕生日の 1桁目と誕生月の 1桁目の和が 10未満なら
```

```
149       birthday[3] += birthday[1]; //誕生日の 1桁目に誕生月
              の 1桁目を足す
150     }
151     for ( i = 0; i < 4 ; i++){
152       Serial.print(birthday[i]); //シリアルモニタに配列の要素
              を表示
153     }
154     Serial.println(); //改行
155   }
156 }
```

工夫した点は, シリアルモニタから送信された文字を一度配列に格納したところである. 普通に Serial.read() した場合, 誕生月を誕生日に足す処理をする際煩雑なコードになってしまう. その対策として一度配列に格納し処理しやすい環境を作ってから処理をし, また繰り上げにも if 文で対応できるようにした.

10 発展課題 2.5.2:創生課題

こちらは時間が足りずに出来なかった.