システム実験
実験6回レポート

6119019056 山口力也

2019/05/31 日提出

1 Arduino から Processing へのデータ送信:通信方式 1

演習 3.1.7 の 5 で考察した内容を報告せよ.

フレームレートの値を大きくすると, 描画の頻度が増える. しかしながら, データ受信のタイミングより早くなると描画が間に合わなくなるので逆に遅延が発生した. フレームレートの値を小さくすると描画のタイミングが遅いので, 半固定抵抗を素早く上下させたときの値が読み取れていなかった.

2 Arduino から Processing へのデータ送信:通信方式 2

演習 3.1.8 の 5 で考察した内容を報告せよ. ハンドシェイクで通信を行うと, 描画とデータ受信が常に交互に行われていた. コネクションが確立されてから通信を行うためだと考えられる. ハンドシェイク (3-way handshaking) について調べると,TCP/IP でも使われているようだった.

3 繰り返し処理による描画 (2D バージョン)

課題 3.1.1 で作成したスケッチ (draw_latice_2D) を報告せよ.

以下ソースコード 1 にスケッチを示す.

ソースコード 1: 課題 3.1.1

```
1 int xn = 10; //x 軸の格子
2 int yn = 10; //y 軸の格子
3
4
```

```
int x,y; //x 座標と y 座標を定義

size(600,600); //ウィンドウサイズ 600*600
background(255); //背景白

stroke(0); //線の色黒
for(int i =0; i<=yn; i++){ //格子の横線
  y = i*height/yn; //線のy 座標
  line(0,y,width,y); //線を描画
}

for(int i = 0; i<= xn; i++){ //格子の縦線
  x = i*width/xn; //線のx 座標
  line(x,0,x,height); //線を描画
}

stroke(255,0,0);
for(int i_c =0; i_c<xn; i_c++){ //i_c はマス目のインデックス
  for(int i_r = 0; i_r < yn; i_r++){
    x = width/(2*xn) + i_c*width/xn; //30 + 60*i_c
    y = height/(2*yn) + i_r*height/yn; //30 + 60*i_r
    ellipse(x,y,width/xn,height/yn); //円を描画
  }
}
```

4 動画の作成:移動する楕円 (2D バージョン)

課題 3.1.2 で作成したスケッチ (draw_latice_2D) を報告せよ.
以下ソースコード 2 にスケッチを示す.

ソースコード 2: 課題 3.1.2

```
int xn = 10; //分割数
int yn = 10; //分割数
int i_c; //楕円を描くマス目番号
int i_r; //楕円を描くマス目番号
int flag_upper_right = 0; //右上端フラグ
int flag_upper_left = 0; //左上端フラグ
int flag_down_right =0; //右下端フラグ
int flag_down_left = 0; //左下端フラグ

void setup(){
  size(600,600); //ウィンドウサイズ 600*600
  frameRate(20); //フレームレート 20
  i_c = 0; //列
```

```
42   i_r = 0; //行
43 }
44 void draw()
45 {
46   int x,y; //x 座標と y 座標
47   background(255); //背景白
48   stroke(0); //線の色黒
49   for(int i = 0; i<= yn; i++){
50     y = i*height/yn; //線のy 座標
51     line(0,y,width,y);
52   }
53   for(int i = 0; i<= xn; i++){ //格子の縦線
54     x = i*width/xn; //線のx 座標
55     line(x,0,x,height);
56   }
57   stroke(255,0,0);
58   x = width/(2*xn) + i_c*width/xn; //30 + 60*i_c
59   y = height/(2*yn) + i_r*height/yn; //30 + 60*i_r
60   ellipse(x,y,width/xn,height/yn);
61   //楕円を描くマス目番号の更新
62   if ( i_c == 0 && i_r == 0){ //左上端の時
63     flag_upper_left = 1;
64     flag_upper_right = 0;
65     flag_down_left = 0 ;
66     flag_down_right = 0;
67     println("upperleftnow");
68   }
69   if ( i_c == xn-1 && i_r == 0){ //右上端の時
70     flag_upper_left = 0;
71     flag_upper_right = 1;
72     flag_down_left = 0 ;
73     flag_down_right = 0;
74     println("upperrightnow");
75   }
76   if ( i_c == 0 && i_r == yn-1){ //左下端の時
77
78     flag_upper_left = 0;
79     flag_upper_right = 0;
80     flag_down_left = 1 ;
81     flag_down_right = 0;
82     println("downleftnow");
83   }
84   if ( i_c == xn-1 && i_r == yn-1){ //右下端の時
85
86     flag_upper_left = 0;
87     flag_upper_right = 0;
```

```
88      flag_down_left = 0 ;
89      flag_down_right = 1;
90      println("downrightnow");
91    }
92    if(flag_upper_left == 1){
93      i_c++;
94    }
95    if(flag_upper_right == 1){
96      i_c--;
97      i_r++;
98    }
99    if(flag_down_left == 1){
100     i_c++;
101   }
102   if(flag_down_right == 1){
103     i_c--;
104     i_r--;
105   }
106
107 }
```

5 電圧変化の時間軸グラフによる可視化 (点のプロット)

課題 3.1.3 で作成した Processing のスケッチを報告せよ. また, 出力したウィンドウのスナップショットを報告せよ. 以下ソースコード 3 にスケッチを示す.

ソースコード 3: 課題 3.1.3

```
108 import processing.serial.*; // Serial ライブラリを取り込む
109 Serial port; // Serial クラスのオブジェクトを宣言
110 int val,count,x,y; //変数宣言
111 void setup()
112 {
113   size(800,300); // サイズ 800 × 300 のウィンドウ生成
114   port = new Serial(this, "/dev/ttyACM0", 9600);//Serial ク
          ラスのインスタンス生成
115   val = 0;
116   count = 0;
117   x = 0;
118   y = 0;
119   background(255); //背景白
120 }
121 void draw()
```

```
122 {
123   stroke(255,0,0); //赤色
124   strokeWeight(5); //太さを 5
125   x = count; //count を代入
126   y = (255 - val)*300/255; //val=0で 300,val=255で 0
127   point(x,y); //点を描画
128   if( count == 800){ //右端についたら
129     count = 0; //初期化
130     background(255); //背景をクリア
131   }
132   println("R"); // 描画タイミング (確認用)
133 }
134 // シリアルポートにデータが到着するたびに呼び出される割り込み
      関数
135 void serialEvent(Serial p) { // p にはデータが到着したシリア
      ルポートに対応するインスタンス (ここでは port )が代入され
      る
136   val = p.read(); // 受信バッファから 1 バイト読み込み
137   println("<-");
138   count++;
139 // データ受信タイミング (確認用)
140 }
```

また, 以下図 1 に出力したウィンドウのスナップショットを示す.

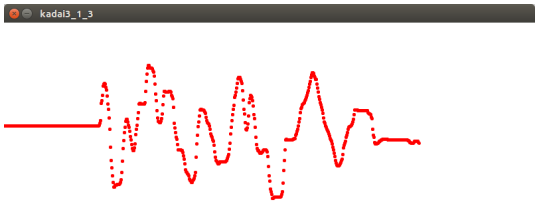

図 1: 課題 3.1.3 の出力画像

6 電圧変化の時間軸グラフによる可視化 (折れ線)

課題 3.1.4 で作成した Processing のスケッチを報告せよ. また, 出力したウィンドウのスナップショットを報告せよ. 以下ソースコード 4 にスケッチを示す.

ソースコード 4: 課題 3.1.4

```
141 import processing.serial.*; // Serial ライブラリを取り込む
142 Serial port; // Serial クラスのオブジェクトを宣言
143 int val,count,new_x,new_y,old_x,old_y; //それぞれ値を定義
144 void setup()
```

```
145 {
146   size(800,300); // サイズ 800 × 300 のウィンドウ生成
147   port = new Serial(this, "/dev/ttyACM0", 9600);//Serial クラスのインスタンス生成
148   val = 0; //Arduino から送られてきたデータを格納する
149   count = 0; //受信した回数 (x 座標)
150   old_x = 0; //折れ線グラフ用の前回のx の値
151   old_y = 0; //折れ線グラフ用の前回のy の値
152   new_x = 0; //折れ線グラフ用の今のx の値
153   new_y = 0; //折れ線グラフ用の今のy の値
154   background(255); //  背景を白に設定
155 }
156 void draw()
157 {
158   stroke(255,0,0); //線の色を赤に設定
159   strokeWeight(5); //線の太さを"5"に設定
160   new_x = count; //x 座標の値は count そのもの
161   new_y = (255 - val)*300/255; //y 座標の値は val=255のときy=0(一番上),val=0のときy=300(一番下)
162   point(new_x,new_y); //点を描画
163   strokeWeight(3); //線の太さを"3"に設定
164   line(old_x,old_y,new_x,new_y); //折れ線用に線を描画
165   old_x = new_x; //今のx の値を前回の値として格納
166   old_y = new_y; //今のy の値を前回の値として格納
167   if( count == 800){ //もし右端まできたら
168     count = 0; //count 初期化
169     background(255); //背景を白に (クリア)
170   }
171   println("R"); // 描画タイミング (確認用)
172 }
173 // シリアルポートにデータが到着するたびに呼び出される割り込み関数
174 void serialEvent(Serial p) { // p にはデータが到着したシリアルポートに対応するインスタンス (ここでは port )が代入される
175   val = p.read(); // 受信バッファから 1 バイト読み込み
176   println("<-"); // データ受信タイミング (確認用)
177   count++; //count を 1 増やす(x 座標を 1 つ右にずらす)
178 }
```

また, 以下図 2 に出力したウィンドウのスナップショットを示す.

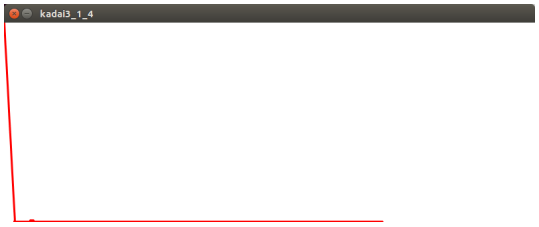

図 2: 課題 3.1.4 の出力画像

照度センサの場合, 図 2 のように変化が分かりづらかったため半固定抵抗のスナップショットを以下図 3 に示す.

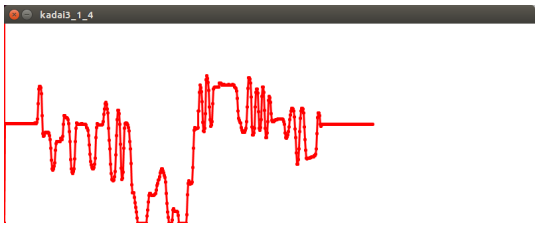

図 3: 課題 3.1.4 の出力画像 (半固定抵抗)

7　電圧変化のメーター表示

課題 3.1.5 で作成した Processing のスケッチを報告せよ. また, 出力したウィンドウのスナップショットを報告せよ. 以下ソースコード 5 にスケッチを示す.

ソースコード 5: 課題 3.1.5

```
179 import processing.serial.*; // Serial ライブラリを取り込む
180 Serial port; // Serial クラスのオブジェクトを宣言
181 float rad,x,y; //ラジアン,x,y をそれぞれ定義
182 int val; //Arduino から送られてくるデータ格納用
183
184 void setup()
185 {
186   size(500,500); // サイズ 500 × 500 のウィンドウ生成
187   port = new Serial(this, "/dev/ttyACM0", 9600);//Serial ク
        ラスのインスタンス生成
188
189 }
190 void draw()
191 {
192   background(255); //背景を白に設定
```

```
193   stroke(0,0,0); //線を黒に設定
194   ellipse(200,200,200,200); //円を描画
195   stroke(255,0,0); //線を赤に設定
196   strokeWeight(5); //線の太さを"5"に設定
197   rad = 2*PI*val/255; //受信したデータをラジアンに変換 (255
          で 2π ,0で 0)
198   x = 200 + 100*cos(rad - PI/2); //x 座標の値(中心 200から距
          離cos(rad - π/2)
199   y = 200 + 100*sin(rad - PI/2); //y 座標の値(中心 200から距
          離sin(rad - π/2)
200   line(200,200,x,y); //メータの線を描画
201   println("R"); // 描画タイミング (確認用)
202 }
203 // シリアルポートにデータが到着するたびに呼び出される割り込み
        関数
204 void serialEvent(Serial p) { // p にはデータが到着したシリア
        ルポートに対応するインスタンス (ここでは port )が代入され
        る
205   val = p.read(); // 受信バッファから 1 バイト読み込み
206   println("<-"); // データ受信タイミング (確認用)
207 }
```

また, 以下図 4 に出力したウィンドウのスナップショットを示す.

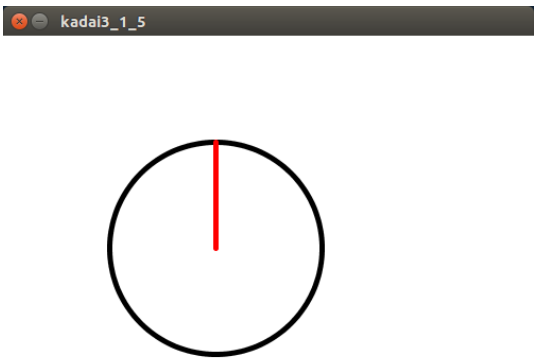

図 4: 課題 3.1.5 の出力画像

8　発展課題 3.1.1 複数データの送信

発展課題 3.1.1 で作成した Arduino と Processing のスケッチを報告せよ.
以下ソースコード 6,7 にそれぞれ Arduino と Processing のスケッチを示す.

ソースコード 6: 発展課題 3.1.1(Arduino)

```
208 int sensorValue1,sensorValue2,sendValue1,sendValue2;
209 unsigned long int timePrev, timeNow;
210 void setup()
211 {
212   Serial.begin(9600); // シリアル通信を 9600bps で初期化
213   timePrev = millis(); // 経過時間の初期値
214 }
215 void loop()
216 {
217   timeNow = millis(); // 現在の経過時間
218   sensorValue1 = analogRead(0); // A0 ピンの AD 変換結果を取
        得 (0-1023)
219   sensorValue2 = analogRead(1); // A1 ピンの AD 変換結果を取
        得 (0-1023)
220   if ( timeNow - timePrev >= 50 ) { // 50ms ごとに実行
221     sendValue1 = sensorValue1 /4; // 0-255 の値に変換
222     sendValue2 = sensorValue2 /4; // 0-255 の値に変換
223     Serial.write('A');
224     Serial.write(sendValue1); // 1 バイトのバイナリデータとし
          て値を送信
225     Serial.write(sendValue2); // 1 バイトのバイナリデータとし
          て値を送信
226     timePrev = timeNow; // 1 ステップ前の経過時間を更新
227   }
228 }
```

ソースコード 7: 発展課題 3.1.1(Processing)

```
229 import processing.serial.*; // Serial ライブラリを取り込む
230 Serial port; // Serial クラスのオブジェクトを宣言
231 int val1,val2; //縦幅と横幅
232 void setup()
233 {
234   size(256, 256); // サイズ 256 × 256 のウィンドウ生成
235   port = new Serial(this, "/dev/ttyACM0", 9600);//Serial ク
        ラスのインスタンス生成
236 }
237 void draw()
238 {
239   background(0); // 背景を黒
```

```
240   ellipse(125, 125, val1, val2); // 中心 (125,125) ,横幅
          val1,縦幅val2 の円 (白塗り)を描画
241   println("R"); // 描画タイミング (確認用)
242 }
243 // シリアルポートにデータが到着するたびに呼び出される割り込み
      関数
244 void serialEvent(Serial p) { // p にはデータが到着したシリア
      ルポートに対応するインスタンス (ここでは port )が代入され
      る
245   if(p.available() >= 3){
246     if(p.read() == 'A'){ //識別子A を定義
247       val1 = p.read(); //読み込み
248       val2 = p.read(); //読み込み
249       p.clear();
250       println("<-"); // データ受信タイミング (確認用)
251       println("val1=",val1); //確認用
252       println("val2=",val2); //確認用
253     }
254   }
255 }
```

また, 以下図 5 に出力したウィンドウのスナップショットを示す.

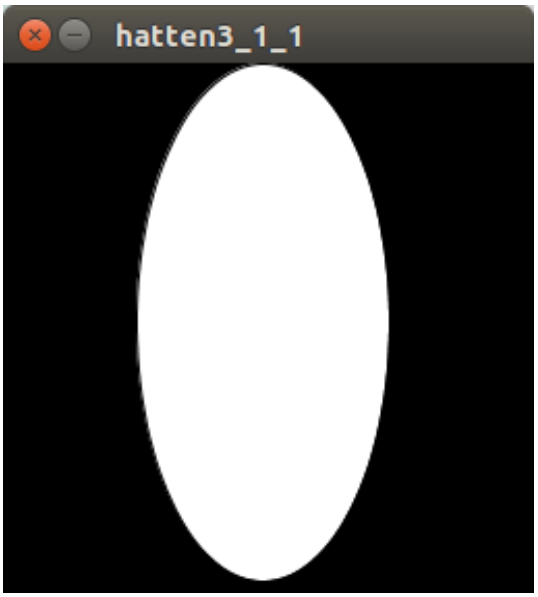

図 5: 発展課題 3.1.1 の出力画像

9　発展課題3.1.2 バーコレーションシミュレーション

こちらは時間が足りずにできなかった.