システム実験
実験7回レポート

6119019056 山口力也

2019/06/07 日提出

1　温度センサによる温度計測:基本フレームワーク

演習 3.2.3 で表示した時間-電圧グラフのスナップショットを報告せよ. 以下 1 にスナップショットを示す.

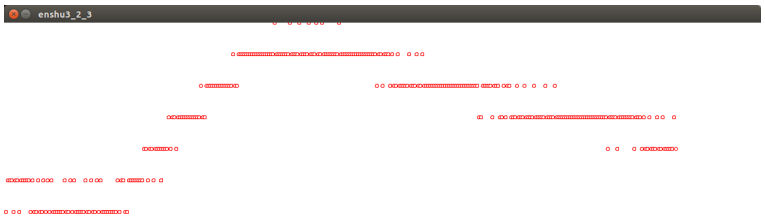

図 1: 演習 3.2.3 の出力画像

2　温度センサによる温度計測:複数サンプルの平均化

課題 3.2.1 で作成した Arduino スケッチと Processing スケッチを報告せよ. 表示した時間-電圧グラフのスナップショットを報告せよ. サンプルの平均化を行った場合と行わなかった場合の違いについて考察せよ.

以下ソースコード 1, ソースコード 2 にそれぞれ Arduino と Processing のスケッチを示す.

ソースコード 1: 課題 3.2.1(Arduino)

```
1 int sensorValue0; //センサの読み込んだ値 (0~1023)
2 unsigned long int timePrev, timeNow;
3 int count; //足しこんだ回数
```

```
4  long int sum; //足しこんだ合計
5  float average; //50ms 間の平均値
6  int intaverage;
7  void setup()
8  {
9          Serial.begin(9600); // シリアル通信を 9600bps で初期
               化
10         timePrev = millis(); // 経過時間の初期値
11 }
12 void loop()
13 {
14   count = 0; //初期化
15   sum = 0; //初期化
16   while(1){
17     timeNow = millis(); //現在時間を格納
18     if(timeNow - timePrev <= 50){ //50ms 経つまで
19        int sensorValue0 = analogRead(A0); //A0 の値を格納
20        //double vo = sensorValue*(5.0/1024.0);
21        //double Temp = (vo*1000.0 - 600.0)/10.0;
22        sum += sensorValue0;
23        count++;
24     }
25     else{
26       break;
27     }
28   }
29   average = (float)sum / (float)count;
30   intaverage = (int)(average * 100);
31   Serial.println(intaverage);
32         Serial.write('H');
33         Serial.write(timeNow >> 24);
34         // 1byte 目
35         Serial.write(timeNow >>16);
36         // 2byte 目
37         Serial.write(timeNow >> 8);
38         // 3byte 目
39         Serial.write(timeNow & 255);
40         // 4byte 目
41         Serial.write( intaverage >> 8 );
42         // 上位 1byte
43         Serial.write( intaverage & 255);
44         // 下位 1byte
45   timePrev = timeNow; //時間の更新
46 }
```

ソースコード 2: 課題 3.2.1(Processing)

```
47 import processing.serial.*;
48 Serial port;
49 int val;
50 int x, y;
51 int high, low;
52 int byte1, byte2, byte3, byte4;
53 int time, time_min, time_max;
54 int period;
55 float f;
56 void setup()
57 {
58         size(1200, 500);
59         port = new Serial(this, "/dev/ttyACM0", 9600);
60         period = 20000;
61         // 横軸の範囲は 20,000ms 間隔
62         time_min = 0;
63         // 横軸の範囲の初期値
64         time_max = period;// 横軸の範囲の初期値
65         x = 0; y = 0;
66         time = 0;
67         background(255);
68         frameRate(60);
69 }
70 void draw()
71 {
72         if ( time > time_max ) { // グラフの再描画
73                 time_min += period; // 横軸の範囲の更新
74                 time_max += period; // 横軸の範囲の更新
75                 background(255);
76         }
77         x = (int)map( time, time_min, time_max, 0, width );
               // x 座標値
78         y = (int)map( f, 174, 184, height*0.7, 0 );
79         // y 座標値
80         stroke(255, 0, 0);
81         ellipse(x, y, 5, 5);
82 }
83 void serialEvent(Serial p) {
84         if ( p.available() >= 7 ) {
85                 if ( p.read() == 'H' ) {
86                         byte1 = p.read();
87                         byte2 = p.read();
88                         byte3 = p.read();
89                         byte4 = p.read();
90                         time = (byte1 << 24 ) + (byte2 << 16
```

```
                                    ) + (byte3 << 8 ) + byte4; // 4
                                    バイトデータ
91                          high = p.read();
92                          low = p.read();
93                          val = (high << 8 ) + low; // 2 バイ
                                トデータ
94          f = (float)val /100.0;
95          println("val=",val);
96          println("f=",f);
97                          p.clear();
98                  }
99            }
100 }
```

また, 以下 2 にスナップショットを示す.

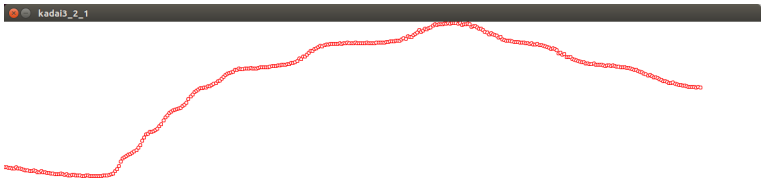

図 2: 課題 3.2.1 の出力画像

サンプルの平均化を行うとはずれ値を取ることがほとんどなくなる. 平均化を行わない場合センサを指でつまなくても, 値にある程度振れ幅があり, 階段状のグラフとなるが, 平均化を行うことで一定の値となる.

3 温度センサによる温度計測:時間-温度グラフの完成

課題 3.2.2 で作成した Processing スケッチを報告せよ. 表示した時間-電圧グラフのスナップショットを報告せよ.

以下ソースコード 3 に Processing のスケッチを示す.

ソースコード 3: 課題 3.2.2(Processing)

```
101 import processing.serial.*;
102 Serial port;
```

```
103 int val; //Arduino からの値格納用変数
104 int x, y; //円の中心値
105 int high, low; //Arduino からの値用(上位 8ビットと下位 8ビット)
106 int byte1, byte2, byte3, byte4; //
        time 用の 4 バイトの値格納用変数
107 int time, time_min, time_max; //現在時刻と最小値,最大値格納用
        変数
108 int t_min,t_max; //温度の最小値,最大値格納用変数
109 int period; //x 軸の範囲設定用
110 float f; //val を float 型に変換する用
111 int count; //y 軸表示用(段階的にy 軸の値を変える用)
112 int x_min,x_max,y_min,y_max;
113 void setup()
114 {
115         size(1200, 500); //1200*500のウィンドウを生成
116         port = new Serial(this, "/dev/ttyACM0", 9600);
117         period = 20000; // 横軸の範囲は 20,000ms 間隔
118         time_min = 0; // 横軸の範囲の初期値
119         time_max = period; // 横軸の範囲の初期値
120   t_min = 20; //温度の最低値
121   t_max = 31; //温度の最大値
122         x = 0; y = 0; //x,y の初期値
123         time = 0; //time の初期値
124   count = 0; //count の初期値
125   x_min = 100; //ウィンドウの一部に表示用
126   x_max = 600; //ウィンドウの一部に表示用
127   y_min = 100; //ウィンドウの一部に表示用
128   y_max = 250; //ウィンドウの一部に表示用
129         background(255); //背景を白
130         frameRate(60); //フレームレート 60に設定
131 }
132 void draw()
133 {
134         if ( time > time_max ) { // グラフの再描画
135                 background(255); //背景を白にクリア
136     /*
137     count = 0; //カウントを初期化
138     while ( count < (t_max - t_min +1)){ //y 軸を再表示
139       textSize(15); //テキストサイズを'15'に設定
140       fill(0); //文字の色を黒に設定
141       text("T=",30,500 - 480/(t_max - t_min)*count - 10);
              //"T="を表示
142       text(t_min+count,50,500 - 480/(t_max - t_min)*count -
              10); //20度から 31度まで表示
143       noFill(); //終了用
144       stroke(0, 0, 255); //文字の色を青に設定
```

```
145       line(0, 500 - 480/(t_max - t_min)*count-10, width,500
              - 500/(t_max - t_min)*count); //y 軸の線を描画
146       count ++; //カウントを増やす
147     }
148     */
149     stroke(0,0,0);
150     rect(x_min,y_min,(x_max - x_min) ,(y_max - y_min));
151     time_min += period; // 横軸の範囲の更新
152     time_max += period; // 横軸の範囲の更新
153         }
154         x = (int)map( time, time_min, time_max, x_min, x_max
                ); // x 座標値
155         y = (int)map( f, t_min, t_max, y_min, y_max ); // y
                座標値
156   /***************y 軸の表示部分*******************/
157   /*
158   textSize(15); //テキストサイズを'15'に設定
159   fill(0); //文字の色を黒に設定
160   text("T=",30,500 - 480/(t_max - t_min)*count - 10); //"T
        ="を表示
161   text(t_min+count,50,500 - 480/(t_max - t_min)*count -
        10); //20度から 31度まで表示
162   noFill(); //終了用
163   stroke(0, 0, 255); //文字の色を青に設定
164   line(0, 500 - 480/(t_max - t_min)*count-10, width,500 -
        500/(t_max - t_min)*count); //y 軸の線を描画
165   if ( count < (t_max - t_min +1)){ //31度まで表示していない
        なら
166     count ++; //カウントを 1増やす
167   }
168   */
169   /*******************円を表示*******************/
170   stroke(0,0,0);
171   noFill();
172   rect(x_min,y_min,(x_max - x_min),(y_max - y_min));
173   stroke(255, 0, 0);
174         ellipse(x, y, 5, 5);
175 }
176 void serialEvent(Serial p) {
177         if ( p.available() >= 7 ) {
178                 if ( p.read() == 'H' ) {
179                         byte1 = p.read(); //それぞれ 1
                            byte ずつ読み込んでいく
180                         byte2 = p.read(); //それぞれ 1
                            byte ずつ読み込んでいく
181                         byte3 = p.read(); //それぞれ 1
```

```
                        byte ずつ読み込んでいく
182                     byte4 = p.read(); //それぞれ 1
                        byte ずつ読み込んでいく
183                     time = (byte1 << 24 ) + (byte2 << 16
                        ) + (byte3 << 8 ) + byte4; // 4
                        バイトデータ
184                     high = p.read(); //上位 8ビットを格納
185                     low = p.read(); //下位 8ビットを格納
186                     val = (high << 8 ) + low; // 2 バイ
                        トデータ
187         f = (float)val /100.0; //float 型に直す
188         println("val=",val); //値確認用
189         println("f=",f); //値確認用
190                     p.clear();
191                 }
192           }
193 }
```

また, 以下 3 にスナップショットを示す.

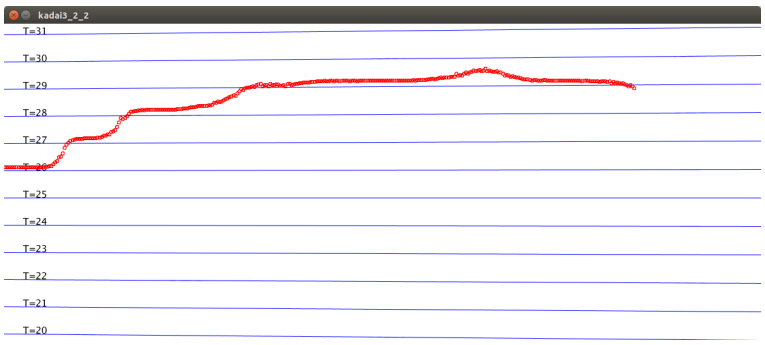

図 3: 課題 3.2.2 の出力画像

4 温度センサによる温度計測:グラフ配置

課題 3.2.3 で作成した Processing スケッチを報告せよ. 表示した時間-電圧グラフのスナップショットを報告せよ.

以下ソースコード 4 に Processing のスケッチを示す.

ソースコード 4: 課題 3.2.3(Processing)

```
194 import processing.serial.*;
195 Serial port;
196 int val; //Arduino からの値格納用変数
197 int x, y; //円の中心値
```

```
198 int high, low; //Arduino からの値用(上位 8ビットと下位 8ビット)
199 int byte1, byte2, byte3, byte4; //
    time 用の 4 バイトの値格納用変数
200 int time, time_min, time_max; //現在時刻と最小値,最大値格納用
    変数
201 int t_min,t_max; //温度の最小値,最大値格納用変数
202 int period; //x 軸の範囲設定用
203 float f; //val を float 型に変換する用
204 int count; //y 軸表示用(段階的にy 軸の値を変える用)
205 int x_min,x_max,y_min,y_max;
206 void setup()
207 {
208         size(1200, 500); //1200*500のウィンドウを生成
209         port = new Serial(this, "/dev/ttyACM0", 9600);
210         period = 20000; // 横軸の範囲は 20,000ms 間隔
211         time_min = 0; // 横軸の範囲の初期値
212         time_max = period; // 横軸の範囲の初期値
213   t_min = 20; //温度の最低値
214   t_max = 31; //温度の最大値
215         x = 0; y = 0; //x,y の初期値
216         time = 0; //time の初期値
217   count = 0; //count の初期値
218   x_min = 100; //ウィンドウの一部に表示用
219   x_max = 600; //ウィンドウの一部に表示用
220   y_min = 100; //ウィンドウの一部に表示用
221   y_max = 250; //ウィンドウの一部に表示用
222         background(255); //背景を白
223         frameRate(60); //フレームレート 60に設定
224 }
225 void draw()
226 {
227         if ( time > time_max ) { // グラフの再描画
228                 background(255); //背景を白にクリア
229
230     count = 0;
231     while ( count < (t_max - t_min + 1)){
232       textSize(13); //テキストサイズを'15'に設定
233       fill(0); //文字の色を黒に設定
234       text("T=",110,150 - 150/(t_max - t_min)*count + 100);
            //"T="を表示
235       text(t_min+count,130,150 - 150/(t_max - t_min)*count
          +100); //20度から 31度まで表示
236       noFill(); //終了用
237       stroke(0, 0, 255); //文字の色を青に設定
238       line(x_min, 150 - 150/(t_max - t_min)*count + 100,
          x_max,150 - 150/(t_max - t_min)*count+ 100); //
```

```
            y 軸の線を描画
239       count++;
240     }
241     stroke(0,0,0);
242     rect(x_min,y_min,(x_max - x_min) ,(y_max - y_min));
243     time_min += period; // 横軸の範囲の更新
244     time_max += period; // 横軸の範囲の更新
245         }
246         x = (int)map( time, time_min, time_max, x_min, x_max
                ); // x 座標値
247         y = (int)map( f, t_min, t_max, y_max, y_min ); // y
                座標値
248   /***************y 軸の表示部分*******************/
249
250   textSize(13); //テキストサイズを'15'に設定
251   fill(0); //文字の色を黒に設定
252   text("T=",110,150 - (150/(t_max - t_min)*count )+ 100);
          //"T="を表示
253   text(t_min+count,130,150 - 150/(t_max - t_min)*count
          +100); //20度から 31度まで表示
254   noFill(); //終了用
255   stroke(0, 0, 255); //文字の色を青に設定
256   line(x_min, 150 - 150/(t_max - t_min)*count + 100, x_max
          ,150 - 150/(t_max - t_min)*count+ 100); //
          y 軸の線を描画
257   if ( count < (t_max - t_min)){ //31度まで表示していないな
          ら
258     count ++; //カウントを 1増やす
259   }
260
261   /*******************円を表示*******************/
262   stroke(0,0,0);
263   noFill();
264   rect(x_min,y_min,(x_max - x_min),(y_max - y_min));
265   stroke(255, 0, 0);
266         ellipse(x, y, 5, 5);
267 }
268 void serialEvent(Serial p) {
269         if ( p.available() >= 7 ) {
270                 if ( p.read() == 'H' ) {
271                         byte1 = p.read(); //それぞれ 1
                                byte ずつ読み込んでいく
272                         byte2 = p.read(); //それぞれ 1
                                byte ずつ読み込んでいく
273                         byte3 = p.read(); //それぞれ 1
                                byte ずつ読み込んでいく
```

```
274                         byte4 = p.read(); //それぞれ 1
                                byte ずつ読み込んでいく
275                         time = (byte1 << 24 ) + (byte2 << 16
                                ) + (byte3 << 8 ) + byte4; // 4
                                バイトデータ
276                         high = p.read(); //上位 8ビットを格納
277                         low = p.read(); //下位 8ビットを格納
278                         val = (high << 8 ) + low; // 2 バイ
                                トデータ
279       f = (float)val /100.0; //float 型に直す
280       println("val=",val); //値確認用
281       println("f=",f); //値確認用
282                         p.clear();
283                 }
284         }
285 }
```

また, 以下 4 にスナップショットを示す.

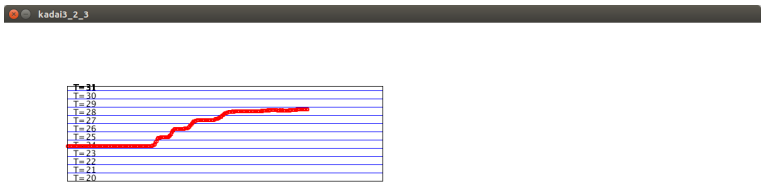

図 4: 課題 3.2.3 の出力画像