

# システム実験

## 実験後期第2回レポート

6119019056 山口力也

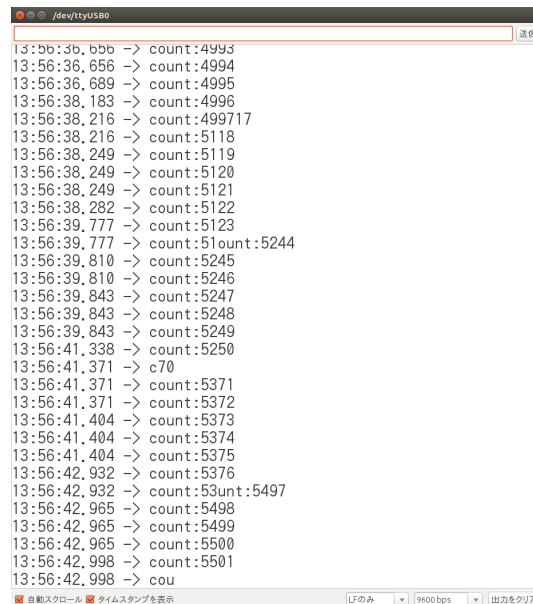
2019/10/17 日提出

### 1 課題 11.2.1

演習 11.2.2 でのシリアル通信のプログラム (プログラム 2) では, プログラム中で 50ms の delay を行っていた. この delay を行わない場合, シリアル通信の結果がどのように変化するか確認せよ.

表示内容がどのように変わった, 画面のスクリーンショットを示して解説せよ. また, 結果が変わった原因を考察せよ.

以下図 1 に課題 11.2.1 の画面のスクリーンショットを示す.



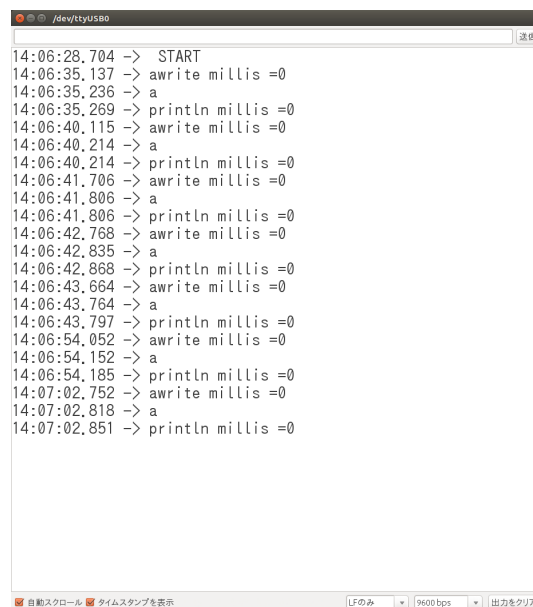
```
13:56:36,656 -> count:4993
13:56:36,656 -> count:4994
13:56:36,689 -> count:4995
13:56:38,183 -> count:4996
13:56:38,216 -> count:499717
13:56:38,216 -> count:5118
13:56:38,249 -> count:5119
13:56:38,249 -> count:5120
13:56:38,249 -> count:5121
13:56:38,282 -> count:5122
13:56:39,777 -> count:5123
13:56:39,777 -> count:51ount:5244
13:56:39,810 -> count:5245
13:56:39,810 -> count:5246
13:56:39,843 -> count:5247
13:56:39,843 -> count:5248
13:56:39,843 -> count:5249
13:56:41,338 -> count:5250
13:56:41,371 -> c70
13:56:41,371 -> count:5371
13:56:41,371 -> count:5372
13:56:41,404 -> count:5373
13:56:41,404 -> count:5374
13:56:41,404 -> count:5375
13:56:42,932 -> count:5376
13:56:42,932 -> count:53unt:5497
13:56:42,965 -> count:5498
13:56:42,965 -> count:5499
13:56:42,965 -> count:5500
13:56:42,998 -> count:5501
13:56:42,998 -> cou
```

図 1: 課題 11.2.1 の結果

delay を行わない場合, システムモニタの表示に不具合が起きた. これは, Arduino 側でシリアル通信により文字列を送信してからシリアルモニタに表示されるまでの間に次の文字列が送信されることが原因だと考えられる.

## 2 課題 11.2.2

演習 11.2.4 において, 地磁気センサとシリアル通信に対して, プログラム中の経過時間の計測を行った. 本課題では, Serial.write() と Serial.println() にそれぞれかかる時間を millis() と micros() の両方で計測し, 比較をせよ. また作成したプログラムと行った時間計測の結果を示し, 結果について考察せよ. 以下図 2 に millis() 関数を用いた場合の画面のスクリーンショットを示す.

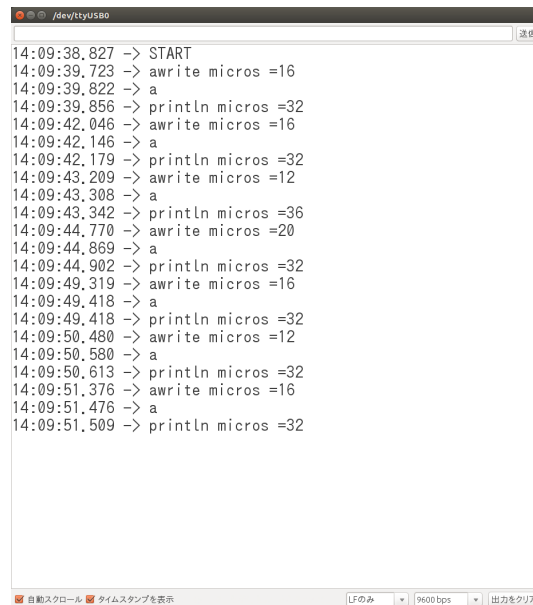


```
14:06:28.704 -> START
14:06:35.137 -> awrite millis =0
14:06:35.236 -> a
14:06:35.269 -> println millis =0
14:06:40.115 -> awrite millis =0
14:06:40.214 -> a
14:06:40.214 -> println millis =0
14:06:41.706 -> awrite millis =0
14:06:41.806 -> a
14:06:41.806 -> println millis =0
14:06:42.768 -> awrite millis =0
14:06:42.835 -> a
14:06:42.868 -> println millis =0
14:06:43.664 -> awrite millis =0
14:06:43.764 -> a
14:06:43.797 -> println millis =0
14:06:54.052 -> awrite millis =0
14:06:54.152 -> a
14:06:54.185 -> println millis =0
14:07:02.752 -> awrite millis =0
14:07:02.818 -> a
14:07:02.851 -> println millis =0
```

図 2: 課題 11.2.2 の結果 (millis)

millis() 関数を用いた場合は Serial.write() と Serial.println() を用いた場合に差は見られなかった.

以下図 3 に micros() 関数を用いた場合の画面のスクリーンショットを示す.



```
14:09:38.827 -> START
14:09:39.723 -> awrite micros =16
14:09:39.822 -> a
14:09:39.856 -> println micros =32
14:09:42.046 -> awrite micros =16
14:09:42.146 -> a
14:09:42.179 -> println micros =32
14:09:43.209 -> awrite micros =12
14:09:43.308 -> a
14:09:43.342 -> println micros =36
14:09:44.770 -> awrite micros =20
14:09:44.869 -> a
14:09:44.902 -> println micros =32
14:09:49.319 -> awrite micros =16
14:09:49.418 -> a
14:09:49.418 -> println micros =32
14:09:50.480 -> awrite micros =12
14:09:50.580 -> a
14:09:50.613 -> println micros =32
14:09:51.376 -> awrite micros =16
14:09:51.476 -> a
14:09:51.509 -> println micros =32
```

図 3: 課題 11.2.2 の結果 (micros)

micros() 関数を用いた場合は Serial.write() の方が Serial.println に比べて約半分程度の速さということがわかった. これらの結果から  $\mu s$  のオーダーで速さを求められる時や一回のループで何百回と Serial.print() してる場合は Serial.write() が有用であると考えられる.

以下ソースコード 1 に作成したプログラムのソースコードを示す.

#### ソースコード 1: 課題 11.2.2

```
1 #include <ZumoMotors.h> // Zumo 用モータ制御ライブラリの読み込み
2 #include <Pushbutton.h> //
  Zumo 用ユーザボタンライブラリの読み込み
3 #include <Wire.h> // I2C/TWI 通信デバイスの制御用ライブラリの
  読み込み
4
5 unsigned long int timeStart = 0; // 時間計測用変数
6 unsigned long int timeEnd = 0; // 時間計測用変数
7
8 int cnt = 0; // 送信回数の確認用カウンタ
9
10 Pushbutton button(ZUMO_BUTTON);
11
12 void setup()
13 {
14   Serial.begin(9600); // シリアル通信の初期化
15   Wire.begin(); // Wire ライブラリの初期化と, I2C バスとの接続
```

```

16
17   button.waitForButton(); // ユーザボタンが押されるまで待つ
18   Serial.println("START"); // 文字列をシリアル送信
19 }
20
21 void loop()
22 {
23   button.waitForButton(); // ユーザボタンが押されるまで待つ
24   //millis
25   /*
26   timeStart = millis(); // 現在の時間取得
27   Serial.write("a");
28   timeEnd = millis(); // 現在の時間取得
29
30   delay(50); // 50ms 待つ
31   Serial.print("write millis =");
32   Serial.println(timeEnd-timeStart); // 経過時間をシリアル送
      信
33
34   timeStart = millis(); // 現在の時間取得
35   Serial.println("a");
36   timeEnd = millis(); // 現在の時間取得
37
38   delay(50);
39   Serial.print("println millis =");// 150ms 待つ
40   Serial.println(timeEnd-timeStart); // 経過時間をシリアル送
      信
41   */
42   timeStart = micros(); // 現在の時間取得
43   Serial.write("a");
44   timeEnd = micros(); // 現在の時間取得
45
46   delay(50); // 50ms 待つ
47   Serial.print("write micros =");
48   Serial.println(timeEnd-timeStart); // 経過時間をシリアル送
      信
49
50   timeStart = micros(); // 現在の時間取得
51   Serial.println("a");
52   timeEnd = micros(); // 現在の時間取得
53
54   delay(50);
55   Serial.print("println micros =");// 150ms 待つ
56   Serial.println(timeEnd-timeStart);
57 }

```

---

### 3 課題 11.2.3

演習 11.2.7 において, 画面を 4 分割して 1 台の Zumo の情報のグラフ描画を行ったが, 本課題では, このプログラムを拡張させ, 2 台の Zumo 情報を同時に Processing 上に描画せよ. ここで, 2 台の Zumo を接続して動作させるとスムーズな描画が行われない原因について考察せよ. 以下図 4 にプログラムの実行結果のスクリーンショットを示す.

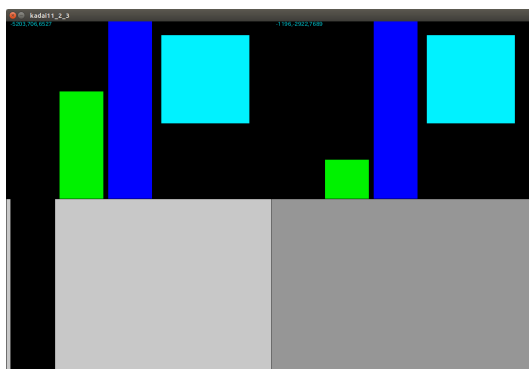


図 4: 課題 11.2.3 の結果

zumo2 台の情報は各 Arduino からそれぞれ送られており, 特に順番などが決まっていないため 1 台目の情報が送られている途中に 2 台目の情報が送られたものを描画する際に遅延が生じてスムーズな描画になっていないと考えられる. 以下ソースコード 2 に Arduino の作成したプログラムのソースコードを示す.

#### ソースコード 2: 課題 11.2.3(Arduino)

```
1 #include <ZumoMotors.h> // Zumo 用モータ制御ライブラリの読み込み
2 #include <Pushbutton.h> //
  Zumo 用ユーザボタンライブラリの読み込み
3 #include <Wire.h> // I2C/TWI 通信デバイスの制御用ライブラリの
  読み込み
4 #include <LSM303.h> // 加速度,地磁気センサ用ライブラリの読み
  込み
5
6 int cnt = 0; // 送信回数の確認用カウンタ
7
8 #define CRB_REG_M_2_5GAUSS 0x60 // CRB_REG_M の値 : 地磁気セ
  ンサのスケールを +/-2.5 ガウスに設定
9 #define CRA_REG_M_220HZ 0x1C // CRA_REG_M の値 : 地磁気センサ
  のアップデートレートを 220 Hz に設定
10
```

```

11 Pushbutton button(ZUMO_BUTTON);
12 LSM303 compass;
13
14 void setup()
15 {
16   Serial.begin(9600); // シリアル通信の初期化
17   Wire.begin(); // Wire ライブラリの初期化と, I2C バスとの接続
18
19   compass.init(); // LSM303 の初期化
20   compass.enableDefault(); // 加速度センサ・地磁気センサ を利
      用可能にする
21   compass.writeReg(LSM303::CRB_REG_M, CRB_REG_M_2_5GAUSS); //
      地磁気センサーのスケールを +/-2.5 ガウスに設定
22   compass.writeReg(LSM303::CRA_REG_M, CRA_REG_M_220HZ); // 地
      磁気センサのアップデートレートを 220 Hz に設定
23
24   button.waitForButton(); // ユーザボタンが押されるまで待つ
25 }
26
27 void loop()
28 {
29   compass.read(); // 地磁気センサの値を読む
30   Serial.print(compass.m.x); // 地磁気センサから得られた値 (x
      )をシリアル送信
31   Serial.print(","); // 区切り文字(,)をシリアル送信
32   Serial.print(compass.m.y); // 地磁気センサから得られた値 (y
      )をシリアル送信
33   Serial.print(","); // 区切り文字(,)をシリアル送信
34
35   Serial.println(String(cnt)); // 送信回数カウンタをシリアル
      送信
36   cnt += 1; // 送信回数カウンタをインクリメント
37   delay(100); // 100ms 待つ
38 }

```

また, 以下ソースコード 3 に Arduino の作成したプログラムのソースコードを示す.

---

#### ソースコード 3: 課題 11.2.3(Processing)

---

```

1 import processing.serial.*;
2
3 Serial port1; // 1台目のZumo のシリアル通信用
4 Serial port2; // 2台目のZumo のシリアル通信用
5
6 String myString1 = null;
7 String myString2 = null;

```

```

8 float red=0, green=0, blue=0;
9 float red2=0, green2=0, blue2=0;
10 int LF = 10; // LF (Linefeed) のアスキーコード
11
12 int zumo_id = 0;
13
14 int graph_width =100; // グラフの幅を定義
15
16 void setup() {
17     size(1200, 800); // 幅 1200px, 高さ 800px のウインドウを
        生成
18     port1 = new Serial(this, "/dev/ttyUSB0", 9600); // Serial
        クラスのインスタンスを生成
19     port1.clear();
20     port1.bufferUntil(0x0d); // LF = 0x0d までバッファ
21
22     // *** ヒント:ここで
        port2の「ポート指定, クリア, LF までバッファ」を行う ***
        //
23
24     port2 = new Serial(this, "/dev/ttyUSB1", 9600); // Serial
        クラスのインスタンスを生成
25     port2.clear();
26     port2.bufferUntil(0x0d);
27
28     background(0); // 背景色を黒に
29
30     fill(100, 100, 100); rect(width/2, 0, width/2, height/2);
        // 右上の領域を塗りつぶす
31     fill(200, 200, 200); rect(0, height/2, width/2, height
        /2); // 左下の領域を塗りつぶす
32     fill(150, 150, 150); rect(width/2, height/2, width/2,
        height/2); // 右下の領域を塗りつぶす
33 }
34
35 void draw() {
36     if(zumo_id == 1){ // データを受信したときだけ書き換える(1
        番目のZumo)
37         fill(0, 0, 0); rect(0,0, width/2,height/2); // 対象画面
        の初期化(黒く塗りつぶす)
38
39         // グラフなどの描画
40         fill(red,0,0); rect(10, height/2, 100, -red);
41         fill(0,green,0); rect(120, height/2, 100,-green);
42         fill(0,0,blue); rect(230,height/2, 100,-blue);
43

```

```

44     fill(red, green, blue); rect(350,30,200,200);
45
46     if(myString1 != null){
47         text(myString1, 10, 10); // シリアル通信で受信したテキ
           ストの表示
48     }
49 }
50
51     // *** ヒント:ここに 2番目のZumo 用の処理を書く ***//
52 if(zumo_id == 2){ // データを受信したときだけ書き換える(1
           番目のZumo)
53     fill(0, 0, 0); rect(width/2,0, width/2,height/2); // 対
           象画面の初期化(黒く塗りつぶす)
54
55     // グラフなどの描画
56     fill(red,0,0); rect(10+width/2, height/2, 100, -red2);
57     fill(0,green,0); rect(120+width/2, height/2, 100,-green2
           );
58     fill(0,0,blue); rect(230+width/2,height/2, 100,-blue2);
59
60     fill(red, green, blue); rect(350+width/2,30,200,200);
61
62     if(myString2 != null){
63         text(myString2, 10+width/2, 10); // シリアル通信で受信
           したテキストの表示
64     }
65 }
66
67
68 }
69
70 // 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
71 void serialEvent(Serial p) {
72     if ((p == port1 || p == port2) && (p.available() > 0)) {
           // 割り込みシリアル通信が,
           port1 か, port2 で, なおかつデータが入っている時
73     zumo_id = 1; // データをやり取りしたロボットのIDを記憶
74
75     if(p == port1) zumo_id = 1; // データをやり取りしたロボッ
           トのIDを記憶
76     else if(p == port2) zumo_id = 2;
77
78     myString1 = port1.readStringUntil(LF); //文字データの最後
           まで読み込み
79     if(myString1 != null){ //文字が入ってたら

```



```

80     myString1 = trim(myString1); //行末の改行 '¥n' を削除
81
82     float data[] = float(split(myString1, ',')); // カンマ
        で区切られた値を分割
83
84     // 受信した数値の数が3つで,NaN (Not a Number) ではない
        時
85     if (data.length == 3 && data[0] != Float.NaN && data
        [1] != Float.NaN && data[2] != Float.NaN){
86         // キャリブレーションあり
87         red = map(data[0], -5000, 1000, 0, 255); // 値を0
            ～255にマッピング
88         green = map(data[1], -5000, 1000, 0, 255); // 値
            を0～255にマッピング
89         blue = map(data[2], -5000, 1000, 0, 255); // 値を0
            ～255にマッピング
90     }
91 }
92 myString2 = port2.readStringUntil(LF); //文字データの最後
        まで読み込み
93 if(myString2 != null){ //文字が入ってたら
94     myString2 = trim(myString2); //行末の改行 '¥n' を削除
95
96     float data[] = float(split(myString2, ',')); // カンマ
        で区切られた値を分割
97
98     // 受信した数値の数が3つで,NaN (Not a Number) ではない
        時
99     if (data.length == 3 && data[0] != Float.NaN && data
        [1] != Float.NaN && data[2] != Float.NaN){
100         // キャリブレーションあり
101         red2 = map(data[0], -5000, 1000, 0, 255); // 値を0
            ～255にマッピング
102         green2 = map(data[1], -5000, 1000, 0, 255); // 値
            を0～255にマッピング
103         blue2 = map(data[2], -5000, 1000, 0, 255); // 値
            を0～255にマッピング
104     }
105 }
106 }
107 }

```

---

## 4 課題 11.2.4

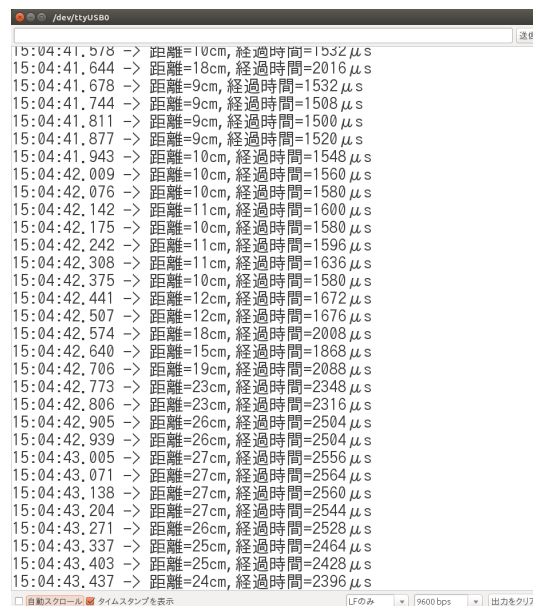
3 台の Zumo ロボットと PC を接続し, 各 Zumo ロボットと Processing をシリアル通信でつなぎ 1 台の zumo マシンが何か作業をしてそれが終わると PC に終わったことを知らせ, 2 台目の zumo マシンが続いて作業しそれが終わると 3 台目のマシンが作業するプログラムを構築せよ. 例えば 3 台の Zumo を並べモータを制御しそれぞれ順に回転させていくなどさせると見た目にもわかりやすい. また, その制御状況 (各 Zumo ロボットの動いている動いていないなど) を Processing に描画せよ.

課題 11.2.4 については実験時間中に終わらなかった.

## 5 課題 11.2.5

前回のシステム実験の課題 2 のプログラムを参考に, 超音波センサを利用するプログラムを改変し, 動作 1 回のループにかかる時間を計測せよ. 計測時間はシリアル通信で確認し, また Processing でグラフィカルに表示する.

以下図 5 にプログラムの実行結果のスクリーンショットを示す.



```
15:04:41.578 -> 距離=10cm, 経過時間=1532 μs
15:04:41.644 -> 距離=18cm, 経過時間=2016 μs
15:04:41.678 -> 距離=9cm, 経過時間=1532 μs
15:04:41.744 -> 距離=9cm, 経過時間=1508 μs
15:04:41.811 -> 距離=9cm, 経過時間=1500 μs
15:04:41.877 -> 距離=9cm, 経過時間=1520 μs
15:04:41.943 -> 距離=10cm, 経過時間=1548 μs
15:04:42.009 -> 距離=10cm, 経過時間=1560 μs
15:04:42.076 -> 距離=10cm, 経過時間=1580 μs
15:04:42.142 -> 距離=11cm, 経過時間=1600 μs
15:04:42.175 -> 距離=10cm, 経過時間=1580 μs
15:04:42.242 -> 距離=11cm, 経過時間=1596 μs
15:04:42.308 -> 距離=11cm, 経過時間=1636 μs
15:04:42.375 -> 距離=10cm, 経過時間=1580 μs
15:04:42.441 -> 距離=12cm, 経過時間=1672 μs
15:04:42.507 -> 距離=12cm, 経過時間=1676 μs
15:04:42.574 -> 距離=18cm, 経過時間=2008 μs
15:04:42.640 -> 距離=15cm, 経過時間=1868 μs
15:04:42.706 -> 距離=19cm, 経過時間=2088 μs
15:04:42.773 -> 距離=23cm, 経過時間=2348 μs
15:04:42.806 -> 距離=23cm, 経過時間=2316 μs
15:04:42.905 -> 距離=26cm, 経過時間=2504 μs
15:04:42.939 -> 距離=26cm, 経過時間=2504 μs
15:04:43.005 -> 距離=27cm, 経過時間=2556 μs
15:04:43.071 -> 距離=27cm, 経過時間=2564 μs
15:04:43.138 -> 距離=27cm, 経過時間=2560 μs
15:04:43.204 -> 距離=27cm, 経過時間=2544 μs
15:04:43.271 -> 距離=26cm, 経過時間=2528 μs
15:04:43.337 -> 距離=25cm, 経過時間=2464 μs
15:04:43.403 -> 距離=25cm, 経過時間=2428 μs
15:04:43.437 -> 距離=24cm, 経過時間=2396 μs
```

図 5: 課題 11.2.5 の結果

Processing 側のグラフィカルな表示に関しては実験中にスクリーンショットを撮るのを忘れてしまっていない.

作成した Arduino のプログラムは以下ソースコード 4 に Processing はソースコード 5 に示す.

ソースコード 4: 課題 11.2.5(Arduino)

---

```
1 #include <ZumoMotors.h> //モータライブラリの読み込み
2 #include <Pushbutton.h> //ボタンライブラリの読み込み
3
4 const int trig = 2; //Trig ピン 2 番
5 const int echo = 4; //Echo ピン 4 番
6 const int buttonPin = 12; //ボタンピンは 12番
7
8 unsigned long int timeStart = 0;
9 unsigned long int timeEnd = 0;
10
11 unsigned long interval; //Echo のパルス幅( $\mu$ s)
12 int distance; //距離 (cm)
13
14 ZumoMotors motors; //ZumoMotors クラスのインスタンス生成
15 Pushbutton button(ZUMO_BUTTON); //
    Pushbutton クラスのインスタンスを生成
16
17 void setup() {
18     Serial.begin(9600);
19     pinMode(trig,OUTPUT); //trig を出力
20     pinMode(echo,INPUT); //echo を入力
21     button.waitForButton();
22 }
23
24 void loop() {
25     //10 $\mu$ s のパルスを超音波センサの Trig ピンに出力
26     timeStart = micros(); //開始時間
27
28     digitalWrite(trig,HIGH);
29     delayMicroseconds(10);
30     digitalWrite(trig,LOW);
31     interval = pulseIn(echo,HIGH,23068); //
        echo が high である時間を計測
32     distance = 340* interval / 10000/ 2; //距離 (cm)に変換
33
34
35     //距離が 10cm 以下なら後進
36     if( distance < 10 ){
37         motors.setLeftSpeed(-100);
38         motors.setRightSpeed(-100);
39     }
40     //距離が 10cm 以上なら前進
```

```

41  if(distance > 10) {
42      motors.setLeftSpeed(100);
43      motors.setRightSpeed(100);
44  }
45  //距離が 10cm なら停止
46  if(distance == 10) {
47      motors.setLeftSpeed(0);
48      motors.setRightSpeed(0);
49  }
50  timeEnd = micros(); //終了時間
51  Serial.print("距離=");
52  Serial.print(distance);
53  Serial.print("cm");
54  Serial.print(",経過時間=");
55  Serial.print(timeEnd - timeStart); //経過時間をシリアル通信
56  Serial.println("μs");
57  delay(60);
58 }

```

---

ソースコード 5: 課題 11.2.5(Processing)

---

```

1  import processing.serial.*;
2
3  Serial port1; // 1台目のZumo のシリアル通信用
4  Serial port2; // 2台目のZumo のシリアル通信用
5  Serial port3;
6
7  String myString1 = null;
8  String myString2 = null;
9
10 int LF = 10; // LF (Linefeed) のアスキーコード
11
12 int zumo_id = 0;
13
14 int graph_width =100; // グラフの幅を定義
15
16 void setup() {
17     size(1200, 800); // 幅 1200px, 高さ 800px のウインドウを
        生成
18     port1 = new Serial(this, "/dev/ttyUSB0", 9600); // Serial
        クラスのインスタンスを生成
19     port1.clear();
20     port1.bufferUntil(0x0d); // LF = 0x0d までバッファ
21
22     // *** ヒント:ここで
        port2の「ポート指定, クリア, LF までバッファ」を行う ***

```

```

//
23
24 port2 = new Serial(this, "/dev/ttyUSB1", 9600); // Serial
    クラスのインスタンスを生成
25 port2.clear();
26 port2.bufferUntil(0x0d);
27
28 background(0); // 背景色を黒に
29
30 fill(100, 100, 100); rect(width/2, 0, width/2, height/2);
    // 右上の領域を塗りつぶす
31 fill(200, 200, 200); rect(0, height/2, width/2, height
    /2); // 左下の領域を塗りつぶす
32 fill(150, 150, 150); rect(width/2, height/2, width/2,
    height/2); // 右下の領域を塗りつぶす
33 }
34
35 void draw() {
36     if(zumo_id == 1){ // データを受信したときだけ書き換える(1
        番目のZumo)
37         fill(0, 0, 0); rect(0,0, width/2,height/2); // 対象画面
            の初期化(黒く塗りつぶす)
38
39         // グラフなどの描画
40         fill(red,0,0); rect(10, height/2, 100, -red);
41         fill(0,green,0); rect(120, height/2, 100,-green);
42         fill(0,0,blue); rect(230,height/2, 100,-blue);
43
44         fill(red, green, blue); rect(350,30,200,200);
45
46         if(myString1 != null){
47             text(myString1, 10, 10); // シリアル通信で受信したテキ
                ストの表示
48         }
49         textSize(100);
50         textAlign(CENTER, CENTER);
51         if (state == 0) {
52             text("GO", 200, 100);
53         } else {
54             text("BACK", 200, 100);
55         }
56     }
57
58     // *** ヒント:ここに2番目のZumo用の処理を書く ***//
59     if(zumo_id == 2){ // データを受信したときだけ書き換える(1
        番目のZumo)

```

```

60     fill(0, 0, 0); rect(width/2,0, width/2,height/2); // 対
        象画面の初期化(黒く塗りつぶす)
61
62     // グラフなどの描画
63     fill(red,0,0); rect(10+width/2, height/2, 100, -red2);
64     fill(0,green,0); rect(120+width/2, height/2, 100,-green2
        );
65     fill(0,0,blue); rect(230+width/2,height/2, 100,-blue2);
66
67     fill(red, green, blue); rect(350+width/2,30,200,200);
68
69     if(myString2 != null){
70         text(myString2, 10+width/2, 10); // シリアル通信で受信
            したテキストの表示
71     }
72 }
73
74
75 }
76
77 // 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
78 void serialEvent(Serial p) {
79     if ((p == port1 || p == port2) && (p.available() > 0)) {
        // 割り込みシリアル通信が,
        port1 か, port2 で, なおかつデータが入っている時
80     zumo_id = 1; // データをやり取りしたロボットのIDを記憶
81
82     if(p == port1) zumo_id = 1; // データをやり取りしたロボッ
        トのIDを記憶
83     else if(p == port2) zumo_id = 2;
84     else if(p == port3) zumo_id = 3;
85
86     myString1 = port1.readStringUntil(LF); //文字データの最後
        まで読み込み
87     if(myString1 != null){ //文字が入ってたら
88         myString1 = trim(myString1); //行末の改行 '¥n' を削除
89
90         float data[] = float(split(myString1, ',')); // カンマ
            で区切られた値を分割
91
92         // 受信した数値の数が3つで,NaN (Not a Number) ではない
            時
93         if (data.length == 3 && data[0] != Float.NaN && data
            [1] != Float.NaN && data[2] != Float.NaN){
94             // キャリブレーションあり

```

```

95         red = map(data[0], -5000, 1000, 0, 255); // 値を 0
           ~255にマッピング
96         green = map(data[1], -5000, 1000, 0, 255); // 値
           を 0~255にマッピング
97         blue = map(data[2], -5000, 1000, 0, 255); // 値を 0
           ~255にマッピング
98     }
99 }
100 myString2 = port2.readStringUntil(LF); //文字データの最後
    まで読み込み
101 if(myString2 != null){ //文字が入ってたら
102     myString2 = trim(myString2); //行末の改行 '¥n' を削除
103
104     float data[] = float(split(myString2, ',')); // カンマ
           で区切られた値を分割
105
106     // 受信した数値の数が3つで,NaN (Nota Number) ではない
           時
107     if (data.length == 3 && data[0] != Float.NaN && data
        [1] != Float.NaN && data[2] != Float.NaN){
108         // キャリブレーションあり
109         red2 = map(data[0], -5000, 1000, 0, 255); // 値を 0
           ~255にマッピング
110         green2 = map(data[1], -5000, 1000, 0, 255); // 値
           を 0~255にマッピング
111         blue2 = map(data[2], -5000, 1000, 0, 255); // 値
           を 0~255にマッピング
112     }
113 }
114 }
115 }

```

---

距離が変わると当然 1 回のループにかかる時間も変わる. 距離が遠ければ 1 回のループにかかる時間は伸びる. 距離が近ければ 1 回のループにかかる時間は短くなる.