システム実験
実験12回レポート

6119019056 山口力也

2019/07/12 日提出

1　レポート 7.2.1

課題 7.2.1 で作成した装置について以下を報告せよ.

- プログラム (Arduino と Processing)
- 実行結果のスクリーンショット
- 物体の位置を正しく表示できたか？できなかった場合, その理由は何か? どんな改善が考えられるか?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 7.2.1(Arduino)

```
1  #include <Servo.h>
2  #include <MsTimer2.h>
3
4  Servo servo; //Servo クラスのインスタンス servo を生成
5  const int trig = 7; //Trig ピンに接続する Arduino のピン番号
6  const int echo = 8; //Echo ピンに接続する Arduino のピン番号
7  const int servoPin = 11; //サーボモータの信号線に接続する
       Arduino のピン番号
8  unsigned long interval; //Echo のパルス幅 (μs)
9  int distance; //距離 (cm)
10 int angle; //回す角度
11 float wait; //待機時間
12 boolean clockwise = false; //時計回りかどうか
13
14 void setup() {
15   Serial.begin(9600); //シリアル通信を 9600bps で初期化
16   pinMode(trig, OUTPUT); //trig を出力ポートに設定
17   pinMode(echo, INPUT); //echo を入力ポートに設定
```

```
18   servo.attach(servoPin); //制御の対象を servoPin に割り当て
19   angle = 0;
20   wait = 80;
21   //15秒で 180度回転したいので
22   //1度回転ごとに 15 / 180秒待てば良い
23   MsTimer2::set(wait,anglechange);
24   MsTimer2::start(); //タイマ割り込みスタート
25
26   //
27 }
28 void anglechange() {
29   servo.write(angle); //角度をサーボモータに出力
30   if (clockwise) angle --; //時計回りなら角度を小さくしていく
31   else {
32     angle ++; //半時計回りなら角度を大きくしていく
33   }
34   if (angle == 180) clockwise = true; //角度が 180に到達した
         ら時計回りに変更
35   if (angle == 0) clockwise = false; //角度が 0に到達したら半
         時計回りにする
36 }
37
38 void loop() {
39   digitalWrite(trig, HIGH); //10μs のパルスを超音波センサの
         Trig ピンに出力
40   delayMicroseconds(10);
41   digitalWrite(trig, LOW);
42   interval = pulseIn(echo, HIGH, 3452); //Echo 信号が HIGH
         である時間 (μs)を計測
43   //0.61*25 + 331.5 = 346.75
44   //346.75 * x * 100 / 2 = 60
45   //x = 3452
46   distance = 340 * interval / 10000 / 2; //距離 (cm)に変換
47
48   if (distance > 60 ) { //距離が 60cm を超えたら 60 とする
49     distance = 60;
50   }
51   Serial.write('H'); //ハンドシェイク用の記号
52   Serial.write(distance); //距離を送信,60までなので 1byte で ok
53   Serial.write(angle); //角度を送信,180までなので 1byte で ok
54   delay(60);
55    //Trig を再度 HIGH にするまでに 60ms 以上の間隔をあける (
         HC-SR04 の仕様)
56    //この処理をしない場合、動作が不安定になる
57 }
```

ソースコード 2: 課題 7.2.1(Processing)

```
58 import processing.serial.*;
59
60 Serial port;
61 int distance; //距離 (cm)
62 int angle; //角度 ( °)
63 int x; //対象のx 座標
64 int y; //対象のy 座標
65 float d_bef; //対象との距離 (前回)
66 float d_now; //対象との距離 (今)
67 void setup() {
68   frameRate(60); //draw()を 1秒間に 60回呼び出す
69   size(800,400); //600*200px のウィンドウを作成
70   background(255,255,255); //背景を白で描画
71   port = new Serial(this, "/dev/ttyACM0",9600); //
         Serial クラスのインスタンスを生成
72   d_bef = 0; //初期化
73   d_now = 0; //初期化
74 }
75
76 void draw() {
77   translate(400,400); //座標軸を (400,400)に移動
78
79   strokeWeight(5); //線の太さを 5に
80   stroke(0,0,0); //線の色を黒に
81   noFill(); //塗りつぶさない
82   arc(0,0,800,800,-PI,PI); //(0,0)中心,直径 800の半円を描画
83   arc(0,0,400,400,-PI,PI); //(0,0)中心,直径 400の半円を描画
84   line(0,0,0,-height);
85   strokeWeight(1);
86   stroke(0,0,255);
87   line(0,0,400 * cos(radians(angle)),-400 * sin(radians(
         angle)));
88   stroke(255,0,0);
89   d_now = map(distance,0,60,0,400); //距離を 0~60から 0~400
         に変える
90   println(d_now);
91   //if( d_now > 90 ) {
92     if ( d_bef != d_now){
93       ellipse(d_now*cos(radians(angle)),-d_now *sin(radians(
             angle)),8,8);
94     }
95   //}
96   d_bef = d_now; //現在の値を過去の値として格納
97   if ( angle == 180 || angle == 0 ) {
98     background(255,255,255); //180度か 0度になったら画面を初
```

```
            期化
 99   }
100 }
101
102 //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関数
103
104 void serialEvent(Serial p) {
105   if ( p.available() >= 3){
106     if (p.read() == 'H' ) {
107       distance = p.read();
108       angle = p.read();
109       p.clear();
110     }
111   }
112 }
```

また, 以下図 1 に実行結果のスクリーンショットを示す.

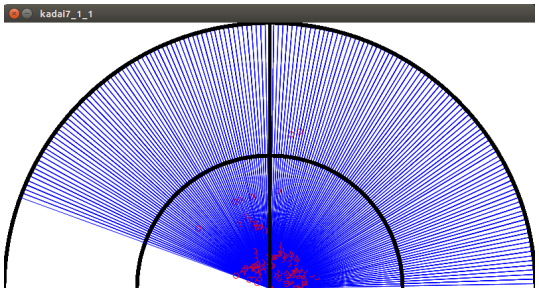

図 1: 課題 7.2.1 の実行結果

比較的近い距離でノイズが現れて, 物体の位置を正しく表示できなかった. これはエアコンの音や周りの音など, 高周波の音が入ると超音波センサと干渉した結果ノイズとして表れたと考えられる.
改善方法としては 2 つあると考えられる. 比較的近い距離でノイズが表れているためある一定の距離以下をプロットしないようにする方法が 1 つ. もう一つはローパスフィルタのようなものを通して高周波ノイズを除去することで他の環境音などと干渉した結果をカットしてしまう方法である.

2 レポート 7.2.2

課題 7.2.2 で作成した装置について以下を報告せよ.

- プログラム (Arduino と Processing)
- 実行結果のスクリーンショット

- どのようなアルゴリズムでノイズを除去したか詳細に解説せよ.
- 除去できなかったノイズがある場合, なぜ除去できなかったか考察せよ.

以下ソースコード 3, ソースコード 4 にそれぞれ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 7.2.2(Arduino)

```
113 #include <Servo.h>
114 #include <MsTimer2.h>
115
116 Servo servo; //Servo クラスのインスタンス servo を生成
117 const int trig = 7; //Trig ピンに接続する Arduino のピン番号
118 const int echo = 8; //Echo ピンに接続する Arduino のピン番号
119 const int servoPin = 11; //サーボモータの信号線に接続する
        Arduino のピン番号
120 unsigned long interval; //Echo のパルス幅 (μs)
121 int distance; //距離 (cm)
122 int angle; //回す角度
123 float wait; //待機時間
124 boolean clockwise = false; //時計回りかどうか
125
126 void setup() {
127   Serial.begin(9600); //シリアル通信を 9600bps で初期化
128   pinMode(trig, OUTPUT); //trig を出力ポートに設定
129   pinMode(echo, INPUT); //echo を入力ポートに設定
130   servo.attach(servoPin); //制御の対象を servoPin に割り当て
131   angle = 0;
132   wait = 80;
133   //15秒で 180度回転したいので
134   //1度回転ごとに 15 / 180秒待てば良い
135   MsTimer2::set(wait,anglechange);
136   MsTimer2::start(); //タイマ割り込みスタート
137
138   //
139 }
140 void anglechange() {
141   servo.write(angle); //角度をサーボモータに出力
142   if (clockwise) angle --; //時計回りなら角度を小さくしていく
143   else {
144     angle ++; //半時計回りなら角度を大きくしていく
145   }
146   if (angle == 180) clockwise = true; //角度が 180に到達した
        ら時計回りに変更
147   if (angle == 0) clockwise = false; //角度が 0に到達したら半
        時計回りにする
148 }
```

```
149
150 void loop() {
151   digitalWrite(trig, HIGH); //10μs のパルスを超音波センサの
          Trig ピンに出力
152   delayMicroseconds(10);
153   digitalWrite(trig, LOW);
154   interval = pulseIn(echo, HIGH, 3452); //Echo 信号が HIGH
          である時間 (μs)を計測
155   //0.61*25 + 331.5 = 346.75
156   //346.75 * x * 100 / 2 = 60
157   //x = 3452
158   distance = 340 * interval / 10000 / 2; //距離 (cm)に変換
159
160   if (distance > 60 ) { //距離が 60cm を超えたら 60 とする
161     distance = 60;
162   }
163   Serial.write('H'); //ハンドシェイク用の記号
164   Serial.write(distance); //距離を送信,60までなので 1byte で ok
165   Serial.write(angle); //角度を送信,180までなので 1byte で ok
166   delay(60);
167    //Trig を再度 HIGH にするまでに 60ms 以上の間隔をあける (
          HC-SR04 の仕様)
168    //この処理をしない場合､動作が不安定になる
169 }
```

コードとしては 1 と同じものになる.

ソースコード 4: 課題 7.2.2(Processing)

```
170 import processing.serial.*;
171
172 Serial port;
173 int distance; //距離 (cm)
174 int angle; //角度 (°)
175 int x; //対象のx 座標
176 int y; //対象のy 座標
177 float d_bef_bef; //対象との距離 (前々回)
178 float d_bef; //対象との距離 (前回)
179 float d_now; //対象との距離 (今)
180 void setup() {
181   frameRate(60); //draw()を 1秒間に 60回呼び出す
182   size(800,400); //600*200px のウィンドウを作成
183   background(255,255,255); //背景を白で描画
184   port = new Serial(this, "/dev/ttyACM0",9600); //
          Serial クラスのインスタンスを生成
185   d_bef = 0; //初期化
186   d_now = 0; //初期化
```

```
187 }
188
189 void draw() {
190   translate(400,400); //座標軸を (400,400)に移動
191
192   strokeWeight(5); //線の太さを 5に
193   stroke(0,0,0); //線の色を黒に
194   noFill(); //塗りつぶさない
195   arc(0,0,800,800,-PI,PI); //(0,0)中心,直径 800の半円を描画
196   arc(0,0,400,400,-PI,PI); //(0,0)中心,直径 400の半円を描画
197   line(0,0,0,-height); //90度の線を描画
198   strokeWeight(1); //線の太さを 1に
199   stroke(0,0,255); //線の色を青に
200   //レーダーの線を描画
201   line(0,0,400 * cos(radians(angle)),-400 * sin(radians(
          angle)));
202   stroke(255,0,0); //線の色を赤に
203   d_now = map(distance,0,60,0,400); //距離を 0~60から 0~400
          に変える
204   println(d_now); //確認用
205   if( d_now > 90 ) { //もし範囲が 90以上なら
206     if ( d_bef_bef == d_bef && d_bef == d_now){ //過去 2回と
            現在の値を比較してすべて同じだったら
207       //対象物として円を描画
208       ellipse(d_now*cos(radians(angle)),-d_now *sin(radians(
              angle)),8,8);
209     }
210   }
211   d_bef_bef = d_bef; //前回の値を前々回の値として格納
212   d_bef = d_now; //現在の値を過去の値として格納
213   if ( angle == 180 || angle == 0 ) {
214     background(255,255,255); //180度か 0度になったら画面を初
            期化
215   }
216 }
217 //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関数
218 void serialEvent(Serial p) {
219   if ( p.available() >= 3){ //送られてきたものが 3つ以上なら
220     if (p.read() == 'H' ) { //ハンドシェイク用の記号'H'が送
            られてきたら
221       distance = p.read(); //距離を取得
222       angle = p.read(); //角度を取得
223       p.clear();
224     }
225   }
226 }
```

また, 以下図 2 に実行結果のスクリーンショットを示す.

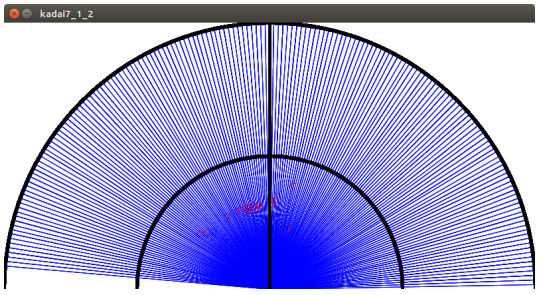

図 2: 課題 7.2.2 の実行結果

1 で, 比較的近い距離にノイズがあったことからある一定の距離より近い距離で対象を誤観測した場合はプロットしないようにした. 実際の対象物はある程度の距離をもたせるようにした. しかしながらそれだけではまだ不完全だったため, 前回と前々回の値を比較して全く同じ値であった場合プロットしないようにした. 対象との距離が一定であった場合うまくプロットされない問題は残ったがノイズ自体は比較的除去できた. 除去できなかったノイズのようなものはないが, 先程述べたとおり除去しすぎて実際に取りたいプロット (対象物の) が得られていないため改善する必要があると考えられる.

3 発展課題レポート 7.2.3

発展課題 7.2.3 で作成した装置について以下を報告せよ.

- プログラム (Arduino と Processing)
- 実行結果のスクリーンショット
- どのようなアルゴリズムでノイズを除去したか詳細に解説せよ.
- 正確にカウントできなかった場合, なぜカウントできなかったかを考察せよ.

以下ソースコード 5, ソースコード 6 にそれぞれ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 5: 発展課題 7.2.3(Arduino)

```
227 #include <Servo.h>
228 #include <MsTimer2.h>
229
230 Servo servo; //Servo クラスのインスタンス servo を生成
```

```
231 const int trig = 7; //Trig ピンに接続する Arduino のピン番号
232 const int echo = 8; //Echo ピンに接続する Arduino のピン番号
233 const int servoPin = 11; //サーボモータの信号線に接続する
        Arduino のピン番号
234 unsigned long interval; //Echo のパルス幅 (μs)
235 int distance; //距離 (cm)
236 int angle; //回す角度
237 float wait; //待機時間
238 boolean clockwise = false; //時計回りかどうか
239
240 void setup() {
241   Serial.begin(9600); //シリアル通信を 9600bps で初期化
242   pinMode(trig, OUTPUT); //trig を出力ポートに設定
243   pinMode(echo, INPUT); //echo を入力ポートに設定
244   servo.attach(servoPin); //制御の対象を servoPin に割り当て
245   angle = 0;
246   wait = 80;
247   //15秒で 180度回転したいので
248   //1度回転ごとに 15 / 180秒待てば良い
249   MsTimer2::set(wait,anglechange);
250   MsTimer2::start(); //タイマ割り込みスタート
251
252   //
253 }
254 void anglechange() {
255   servo.write(angle); //角度をサーボモータに出力
256   if (clockwise) angle --; //時計回りなら角度を小さくしていく
257   else {
258     angle ++; //半時計回りなら角度を大きくしていく
259   }
260   if (angle == 180) clockwise = true; //角度が 180に到達した
        ら時計回りに変更
261   if (angle == 0) clockwise = false; //角度が 0に到達したら半
        時計回りにする
262 }
263
264 void loop() {
265   digitalWrite(trig, HIGH); //10μs のパルスを超音波センサの
        Trig ピンに出力
266   delayMicroseconds(10);
267   digitalWrite(trig, LOW);
268   interval = pulseIn(echo, HIGH, 3452); //Echo 信号が HIGH
        である時間 (μs)を計測
269   //0.61*25 + 331.5 = 346.75
270   //346.75 * x * 100 / 2 = 60
271   //x = 3452
```

```
272   distance = 340 * interval / 10000 / 2; //距離 (cm)に変換
273
274   if (distance > 60 ) { //距離が 60cm を超えたら 60 とする
275     distance = 60;
276   }
277   Serial.write('H'); //ハンドシェイク用の記号
278   Serial.write(distance); //距離を送信,60までなので 1byte で ok
279   Serial.write(angle); //角度を送信,180までなので 1byte で ok
280   delay(60);
281    //Trig を再度 HIGH にするまでに 60ms 以上の間隔をあける (
          HC-SR04 の仕様)
282    //この処理をしない場合､動作が不安定になる
283 }
```

コードとしては 1 と同じものになる.

ソースコード 6: 発展課題 7.2.3(Processing)

```
284 import processing.serial.*;
285
286 Serial port;
287 int distance; //距離 (cm)
288 int angle; //角度 ( °)
289 int x; //対象のx 座標
290 int y; //対象のy 座標
291 float d_bef_bef; //対象との距離 (前々回)
292 float d_bef; //対象との距離 (前回)
293 float d_now; //対象との距離 (今)
294 int count; //対象の個数をカウント
295 float count_d_bef;
296 float count_d_now;
297
298 void setup() {
299   frameRate(60); //draw()を 1秒間に 60回呼び出す
300   size(800,400); //600*200px のウィンドウを作成
301   background(255,255,255); //背景を白で描画
302   port = new Serial(this, "/dev/ttyACM0",9600); //
          Serial クラスのインスタンスを生成
303   d_bef = 0; //初期化
304   d_now = 0; //初期化
305   count = 0;
306 }
307
308 void draw() {
309   translate(400,400); //座標軸を (400,400)に移動
310
311   strokeWeight(5); //線の太さを 5に
```

```
312   stroke(0,0,0); //線の色を黒に
313   noFill(); //塗りつぶさない
314   arc(0,0,800,800,-PI,PI); //(0,0)中心,直径 800の半円を描画
315   arc(0,0,400,400,-PI,PI); //(0,0)中心,直径 400の半円を描画
316   line(0,0,0,-height); //90度の線を描画
317   strokeWeight(1); //線の太さを 1に
318   stroke(0,0,255); //線の色を青に
319   //レーダーの線を描画
320   line(0,0,400 * cos(radians(angle)),-400 * sin(radians(
          angle)));
321   stroke(255,0,0); //線の色を赤に
322   d_now = map(distance,0,60,0,400); //距離を 0~60から 0~400
          に変える
323   println(d_now); //確認用
324
325   if( d_now > 90 ) { //もし範囲が 90以上なら
326     if ( d_bef_bef == d_bef && d_bef == d_now){ //過去 2回と
            現在の値を比較してすべて同じだったら
327       //対象物として円を描画
328       ellipse(d_now*cos(radians(angle)),-d_now *sin(radians(
              angle)),8,8);
329
330      }
331   }
332
333   if (abs(d_now - d_bef) > 10 && abs(d_bef - d_bef_bef) >
          10){
334     count++;
335   }
336   d_bef_bef = d_bef; //前回の値を前々回の値として格納
337   d_bef = d_now; //現在の値を過去の値として格納
338
339    if ( angle == 180 || angle == 0 ) {
340     background(255,255,255); //180度か 0度になったら画面を初
            期化
341          fill(0);
342          textSize(40);
343          text(count,-350,-350);
344     count = 0; //カウントを初期化
345   }
346 }
347 //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関数
348 void serialEvent(Serial p) {
349   if ( p.available() >= 3){ //送られてきたものが 3つ以上なら
350     if (p.read() == 'H' ) { //ハンドシェイク用の記号'H'が送
            られてきたら
```

```
351         distance = p.read(); //距離を取得
352         angle = p.read(); //角度を取得
353         p.clear();
354       }
355     }
356 }
```

また, 以下図 3 に実行結果のスクリーンショットを示す.

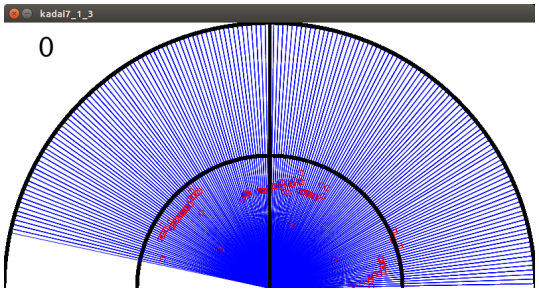

図 3: 発展課題 7.2.3 の実行結果

アルゴリズムとしては距離が一定以上離れていた場合, 過去 2 回の値と比較して現在の値と前回の値, 前回の値と前々回の値を比較してそれぞれの距離 (絶対値) が一定距離を超えていた場合カウントを増やすようにした.
また, 正確にカウントできない場合が多くあったが, これは条件設定が難しいためだと考えられる. ある一定の距離というのが明確に示されておらず, 実験的にこのくらいの距離だろうというのを確かめて設定したためその時々で曖昧な設定になりうまくカウントできなかったと考えられる.