

PH435 Lab 4

Sankalp Gambhir
180260032

October 27, 2020

1. Simple Serve-and-Volley

1.1) The state machine is

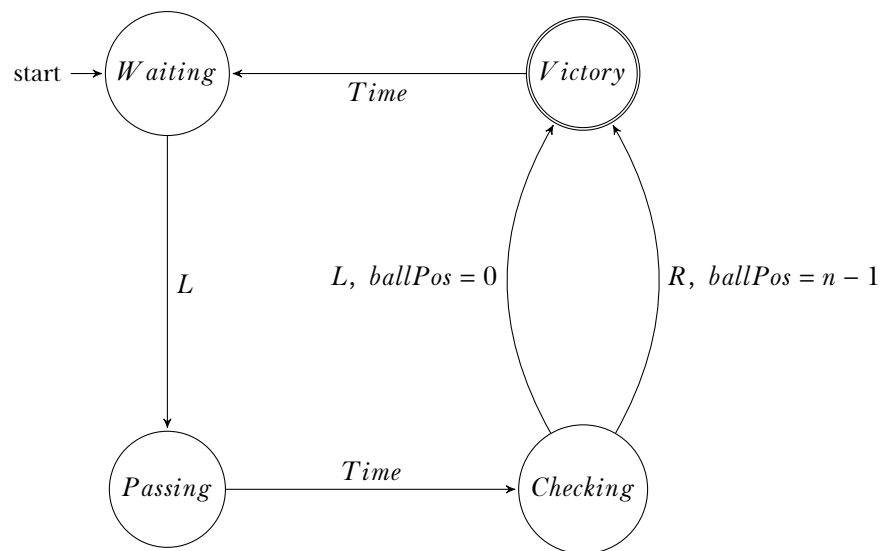


Figure 1: State Machine for the game.

The switches were connected, see [Figure 2](#). The following code was used for the interrupts:

```
1 // <-- Indicates a skip in code
2 volatile bool leftHit = false, rightHit = false;
3 //
4 volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
5 //
6
7 void setup(){
8     //
9     pinMode(leftPlayer, INPUT);
10    attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
11    pinMode(rightPlayer, INPUT);
12    attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
13 }
```

```

14
15 //
16
17 void leftPlayerHit(){
18     leftHit = true;
19     timeLeftHit = 0;
20     return;
21 }
22
23 void rightPlayerHit(){
24     rightHit = true;
25     timeRightHit = 0;
26     return;
27 }
28

```

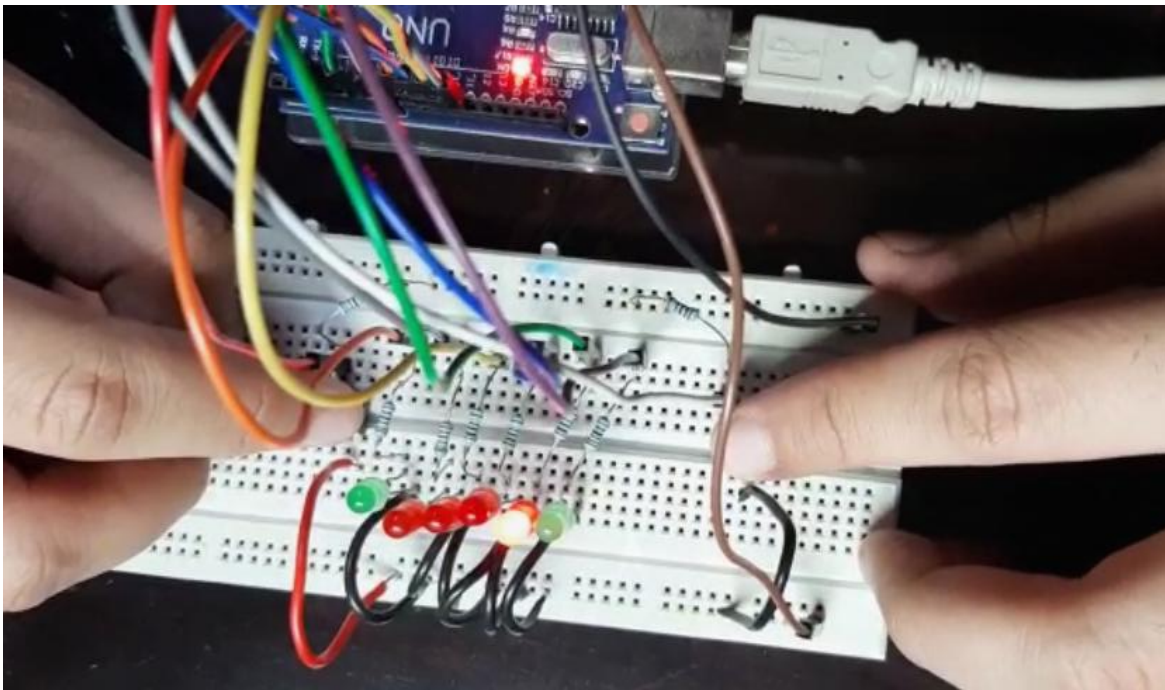


Figure 2: Connected setup

1.2) The LEDs were connected as in **Figure 2**. One of my red LEDs seems to not light up, so it looks a bit awkward in the demonstration. The following code is used for the light propagation:

```

1 // <-- Indicates a skip in code
2 // LED outputs
3 int redPin[4] = {5, 9, 10, 11}; // pins for red LEDs, from the left
4
5 //
6 uint8_t ball[2] = {0, 1}; // position 0, moving right
7 //

```

```

8 void loop(){
9     //
10
11     switch (state)
12     {
13         //
14
15         case 1:
16             // game in play
17             delay(600); // delay to control clock speed
18             if(ball[0] != 0 && ball[0] != 3){
19                 // somewhere in the middle
20                 ball[0] += ball[1]; // x = x + v.dt
21                 changeOutputs();
22                 break;
23             }
24             else{
25                 ballInRange = (ball[0] == 0) ? 1 : -1;
26                 state = (ball[0] == 0) ? 2 : 3;
27                 // start counter
28                 ballWaitStart = millis();
29             }
30             break;
31
32         //
33     }
34
35     //
36
37 void changeOutputs(){
38     for(int i = 0; i < 4; i++){
39         if(ball[0] == i){
40             analogWrite(redPin[i], 255);
41         }
42         else{
43             analogWrite(redPin[i], 0);
44         }
45     }
46 }
47

```

1.3) The setup was completed, and a video demonstration may be found in this [Google Drive Folder](#).

2. Physics and Table Tennis

2.1) The ISR code was unmodified. The previous implementation already included a press timer. The same was used to now also modify the delay between the ball skipping across the LEDs. The whole code may be found in the Appendix.

```

1 // <-- Indicates a skip
2 uint16_t delayBySpeed = 600;
3 //
4 void loop(){

```

```
5 //
6
7 switch (state)
8 {
9 //
10
11 case 1:
12 // game in play
13 delay(600); // delay to control clock speed
14 //
15 break;
16 //
17 case 3:
18 // ball in left range
19 //
20 if(!leftHit) break;
21 //
22 ball[1] = 1; // travel to right
23 //
24 delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
25 break;
26 case 4:
27 // ball in right range
28 //
29 if(!rightHit) break;
30 //
31 ball[1] = -1; // travel to left
32 //
33 delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
34 break;
35
36 //
37 }
38
39 //
40
```

- 2.2)** $t = 0$ was already implemented by way of ballWaitStart (measured against millis() instead of specifically counting).
- 2.3)** The final demonstration (though I found the speed changes hard to notice on video, they are noticeable while playing) is posted in the [same Google Drive Folder](#). The entire code is present in the Appendix for your viewing pleasure.

Appendix

I log in, therefore I am

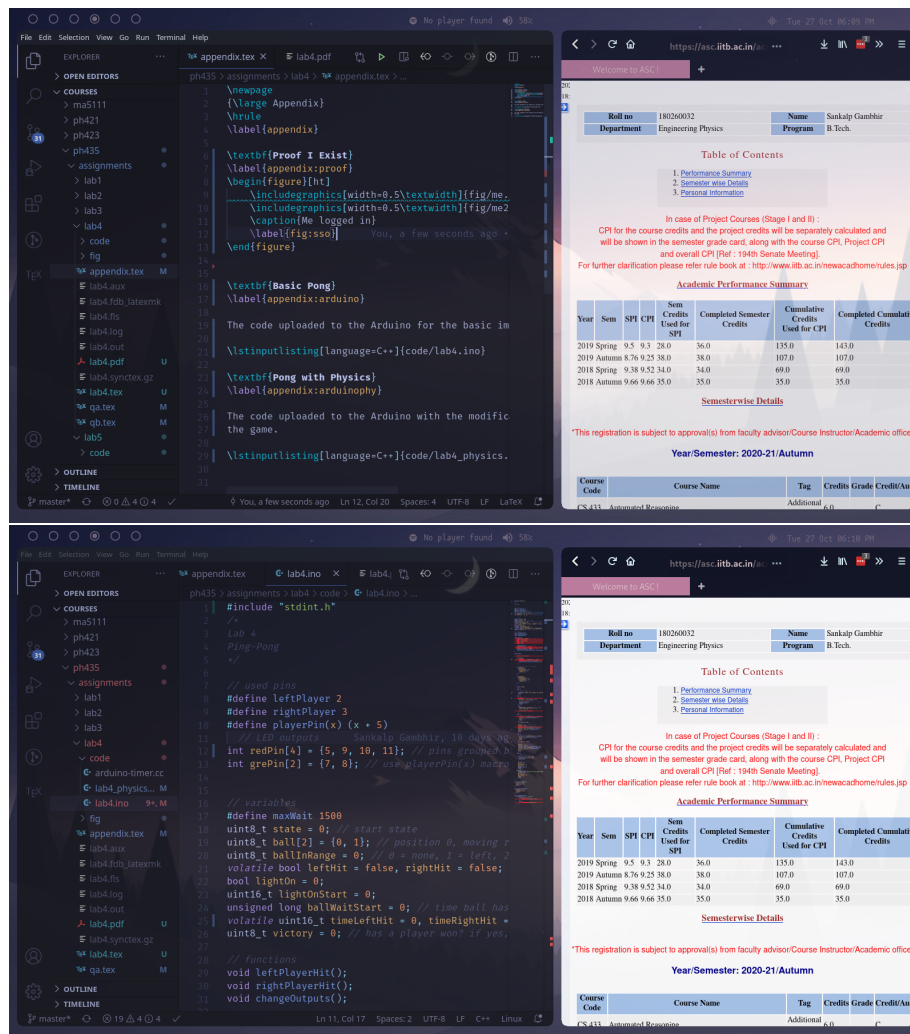


Figure 3: Me logged in

Basic Pong

The code uploaded to the Arduino for the basic implementation of the pong game.

```
1 /*
2 Lab 4
3 Ping-Pong
4 */
5
6 // used pins
```

```
7 #define leftPlayer 2
8 #define rightPlayer 3
9 #define playerPin(x) (x + 5)
10 // LED outputs
11 int redPin[4] = {5, 9, 10, 11}; // pins grouped by the same frequency
12 int grePin[2] = {7, 8}; // use playerPin(x) macro instead for compile time resolution
13
14
15 // variables
16 #define maxWait 1500
17 uint8_t state = 0; // start state
18 uint8_t ball[2] = {0, 1}; // position 0, moving right
19 uint8_t ballInRange = 0; // 0 = none, 1 = left, 2 = right
20 volatile bool leftHit = false, rightHit = false;
21 bool lightOn = 0;
22 uint16_t lightOnStart = 0;
23 unsigned long ballWaitStart = 0; // time ball has been near player
24 volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
25 uint8_t victory = 0; // has a player won? if yes, which?
26
27 // functions
28 void leftPlayerHit();
29 void rightPlayerHit();
30 void changeOutputs();
31
32 void setup(){
33     Serial.begin(1000000); // why not
34     DDRD |= 0b11111110; // pins 1-7 as output. 0 is Tx
35     DDRB |= 0b11111111; // set all pins 8-13 (and dummy bits) to output
36     PORTD = 0x0; // initialise as low
37     PORTB = 0x0;
38
39     changeOutputs(); // initial lights
40     pinMode(leftPlayer, INPUT);
41     attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
42     pinMode(rightPlayer, INPUT);
43     attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
44 }
45
46 void loop(){
47
48     Serial.println(state);
49
50     if(lightOn){
51         if((millis() - lightOnStart) >= 300){
52             digitalWrite(playerPin(leftPlayer), 0);
53             digitalWrite(playerPin(rightPlayer), 0);
54         }
55     }
56
57     // reset interrupts if not handled
58     if(leftHit){
59         timeLeftHit++;
60         if(timeLeftHit > 3){
61             timeLeftHit = 0;
62             leftHit = 0;
63         }
64     }
```

```
65
66     if(rightHit){
67         timeRightHit++;
68         if(timeRightHit > 3){
69             timeRightHit = 0;
70             rightHit = 0;
71         }
72     }
73
74
75     switch (state)
76     {
77     case 0:
78         // initial state
79         // wait for first input
80         if(leftHit) {state = 1; leftHit = 0; ball[0] += ball[1]; changeOutputs();}
81         break;
82
83     case 1:
84         // game in play
85         delay(600); // delay to control clock speed
86         if(ball[0] != 0 && ball[0] != 3){
87             // somewhere in the middle
88             ball[0] += ball[1]; // x = x + v.dt
89             changeOutputs();
90             break;
91         }
92         else{
93             ballInRange = (ball[0] == 0) ? 1 : -1;
94             state = (ball[0] == 0) ? 2 : 3;
95             // start counter
96             ballWaitStart = millis();
97         }
98
99     case 2:
100         // ball in left range
101         if((millis() - ballWaitStart) >= maxWait){
102             victory = playerPin(rightPlayer);
103             state = 4;
104             break;
105         }
106         if(!leftHit) break;
107
108         ball[1] = 1; // travel to right
109         leftHit = false; // reset
110         digitalWrite(playerPin(leftPlayer), 1); // light up
111         lightOn = true; lightOnStart = millis();
112         ball[0] += ball[1]; changeOutputs();
113         state = 1;
114         break;
115
116     case 3:
117         // ball in right range
118         if((millis() - ballWaitStart) >= maxWait){
119             victory = playerPin(leftPlayer);
120             state = 4;
121             break;
122         }
123     }
```

```
123     if(!rightHit) break;
124
125     ball[1] = -1; // travel to left
126     rightHit = false; // reset
127     digitalWrite(playerPin(rightPlayer), 1); // light up
128     lightOn = true; lightOnStart = millis();
129     ball[0] += ball[1]; changeOutputs();
130     state = 1;
131     break;
132
133     case 4:
134         // ball miss
135         // flash 4 times;
136         for(int i = 1; i <= 8; i++){
137             digitalWrite(victory, i%2);
138             delay(250);
139         }
140         // go back to initial
141         ball[0] = 0;
142         ball[1] = 1;
143         ballInRange = 0;
144         state = 0;
145         changeOutputs();
146         break;
147
148     default:
149         Serial.println("Invalid state variable! Defaulting to initial.");
150         ball[0] = 0;
151         ball[1] = 1;
152         ballInRange = 0;
153         state = 0;
154         break;
155     }
156 }
157
158 void leftPlayerHit(){
159     leftHit = true;
160     timeLeftHit = 0;
161     return;
162 }
163
164 void rightPlayerHit(){
165     rightHit = true;
166     timeRightHit = 0;
167     return;
168 }
169
170 void changeOutputs(){
171     for(int i = 0; i < 4; i++){
172         if(ball[0] == i){
173             analogWrite(redPin[i], 255);
174         }
175         else{
176             analogWrite(redPin[i], 0);
177         }
178     }
179 }
```


Pong with Physics

The code uploaded to the Arduino with the modifications implementing speed for the game.

```

1  /*
2  Lab 4.20
3  Ping-Pong
4  With Physics
5  */
6
7  // used pins
8  #define leftPlayer 2
9  #define rightPlayer 3
10 #define playerPin(x) (x + 5)
11 // LED outputs
12 int redPin[4] = {5, 9, 10, 11}; // pins grouped by the same frequency
13 int grePin[2] = {7, 8}; // use playerPin(x) macro instead for compile time resolution
14
15
16 // variables
17 #define maxWait 1500
18 uint8_t state = 0; // start state
19 uint8_t ball[2] = {0, 1}; // position 0, moving right
20 uint8_t ballInRange = 0; // 0 = none, 1 = left, 2 = right
21 volatile bool leftHit = false, rightHit = false;
22 bool lightOn = 0;
23 uint16_t lightOnStart = 0;
24 unsigned long ballWaitStart = 0; // time ball has been near player
25 volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
26 uint16_t delayBySpeed = 600;
27 uint8_t victory = 0; // has a player won? if yes, which?
28
29 // functions
30 void leftPlayerHit();
31 void rightPlayerHit();
32 void changeOutputs();
33
34 void setup(){
35     Serial.begin(1000000); // why not
36     DDRD |= 0b11111110; // pins 1-7 as output. 0 is Tx
37     DDRB |= 0b11111111; // set all pins 8-13 (and dummy bits) to output
38     PORTD = 0x0; // initialise as low
39     PORTB = 0x0;
40
41     changeOutputs(); // initial lights
42     pinMode(leftPlayer, INPUT);
43     attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
44     pinMode(rightPlayer, INPUT);
45     attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
46 }
47
48 void loop(){
49
50     Serial.println(state);
51
52     if(lightOn){
53         if((millis() - lightOnStart) >= 300){
54             digitalWrite(playerPin(leftPlayer), 0);
55             digitalWrite(playerPin(rightPlayer), 0);

```

```
56     }
57   }
58
59   // reset interrupts if not handled
60   if(leftHit){
61     timeLeftHit++;
62     if(timeLeftHit > 3){
63       timeLeftHit = 0;
64       leftHit = 0;
65     }
66   }
67
68   if(rightHit){
69     timeRightHit++;
70     if(timeRightHit > 3){
71       timeRightHit = 0;
72       rightHit = 0;
73     }
74   }
75
76
77   switch (state)
78   {
79   case 0:
80     // initial state
81     // wait for first input
82     if(leftHit) {state = 1; leftHit = 0; ball[0] += ball[1]; changeOutputs();}
83     break;
84
85   case 1:
86     // game in play
87     delay(delayBySpeed); // delay to control clock speed
88     if(ball[0] != 0 && ball[0] != 3){
89       // somewhere in the middle
90       ball[0] += ball[1]; // x = x + v.dt
91       changeOutputs();
92       break;
93     }
94     else{
95       ballInRange = (ball[0] == 0) ? 1 : -1;
96       state = (ball[0] == 0) ? 2 : 3;
97       // start counter
98       ballWaitStart = millis();
99     }
100
101   case 2:
102     // ball in left range
103     if((millis() - ballWaitStart) >= maxWait){
104       victory = playerPin(rightPlayer);
105       state = 4;
106       break;
107     }
108     if(!leftHit) break;
109
110     ball[1] = 1; // travel to right
111     leftHit = false; // reset
112     digitalWrite(playerPin(leftPlayer), 1); // light up
113     lightOn = true; lightOnStart = millis();
```

```
114     ball[0] += ball[1]; changeOutputs();
115     delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
116     state = 1;
117     break;
118
119 case 3:
120     // ball in right range
121     if((millis() - ballWaitStart) >= maxWait){
122         victory = playerPin(leftPlayer);
123         state = 4;
124         break;
125     }
126     if(!rightHit) break;
127
128     ball[1] = -1; // travel to left
129     rightHit = false; // reset
130     digitalWrite(playerPin(rightPlayer), 1); // light up
131     lightOn = true; lightOnStart = millis();
132     ball[0] += ball[1]; changeOutputs();
133     delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
134     state = 1;
135     break;
136
137 case 4:
138     // ball miss
139     // flash 4 times;
140     for(int i = 1; i <= 8; i++){
141         digitalWrite(victory, i%2);
142         delay(250);
143     }
144     // go back to initial
145     ball[0] = 0;
146     ball[1] = 1;
147     ballInRange = 0;
148     state = 0;
149     changeOutputs();
150     break;
151
152 default:
153     Serial.println("Invalid state variable! Defaulting to initial.");
154     ball[0] = 0;
155     ball[1] = 1;
156     ballInRange = 0;
157     state = 0;
158     break;
159 }
160 }
161
162 void leftPlayerHit(){
163     leftHit = true;
164     timeLeftHit = 0;
165     return;
166 }
167
168 void rightPlayerHit(){
169     rightHit = true;
170     timeRightHit = 0;
171     return;
```

```
172 }  
173  
174 void changeOutputs(){  
175     int dir = (ball[1] > 0);  
176     for(int i = 0; i < 4; i++){  
177         if((ball[0] == i) || (ball[0] == (i - dir))){  
178             // leave a trail  
179             analogWrite(redPin[i], 255 * ((int) (ball[0] == i)));  
180         }  
181         else{  
182             analogWrite(redPin[i], 0);  
183         }  
184     }  
185 }
```