PH435 Lab 4

Sankalp Gambhir
180260032

October 27, 2020

1. Simple Serve-and-Volley

1.1) The state machine is

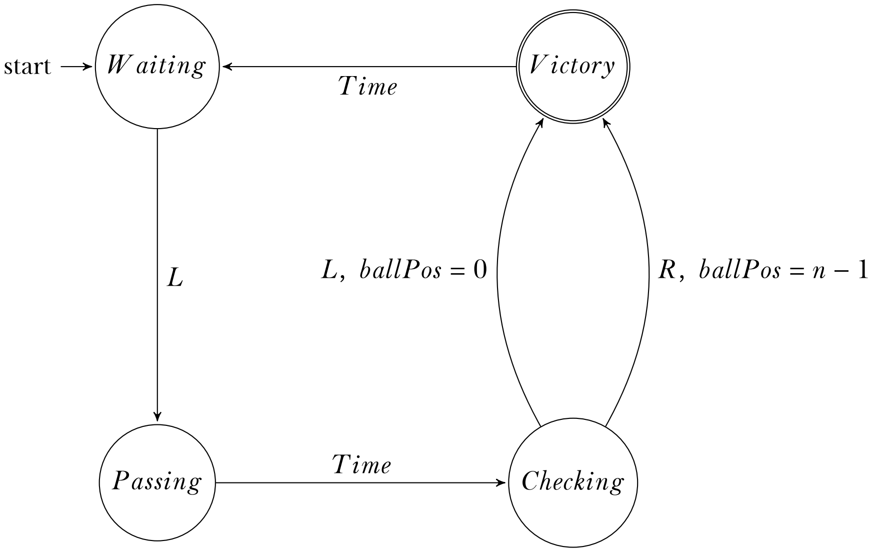


Figure 1: State Machine for the game.

The switches were connected, see Figure 2. The following code was used for the interrupts:

```
// <-- Indicates a skip in code
volatile bool leftHit = false, rightHit = false;
//
volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
//

void setup(){
    //
    pinMode(leftPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
    pinMode(rightPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
}
```

```

//

void leftPlayerHit(){
  leftHit = true;
  timeLeftHit = 0;
  return;
}

void rightPlayerHit(){
  rightHit = true;
  timeRightHit = 0;
  return;
}

```

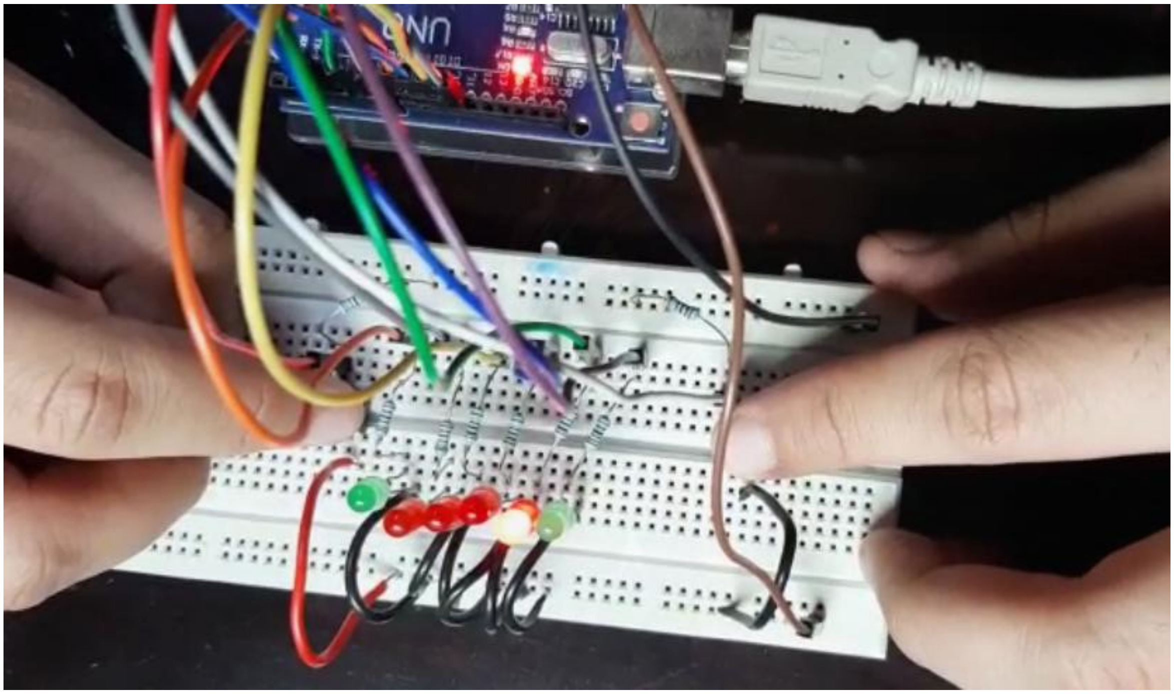


Figure 2: Connected setup

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:

```
// <-- Indicates a skip in code
// LED outputs
int redPin[4] = {5, 9, 10, 11}; // pins for red LEDs, from the left

//
uint8_t ball[2] = {0, 1}; // position 0, moving right
//
```

```
void loop(){
    //

    switch (state)
    {
    //

    case 1:
      // game in play
      delay(600); // delay to control clock speed
      if(ball[0] != 0 && ball[0] != 3){
        // somewhere in the middle
        ball[0] += ball[1]; // x = x + v.dt
        changeOutputs();
        break;
      }
      else{
        ballInRange = (ball[0] == 0) ? 1 : -1;
        state = (ball[0] == 0) ? 2 : 3;
        // start counter
        ballWaitStart = millis();
      }
      break;

   //
}

//

void changeOutputs(){
    for(int i = 0; i < 4; i++){
        if(ball[0] == i){
            analogWrite(redPin[i], 255);
        }
        else{
            analogWrite(redPin[i], 0);
        }
    }
}

```

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.

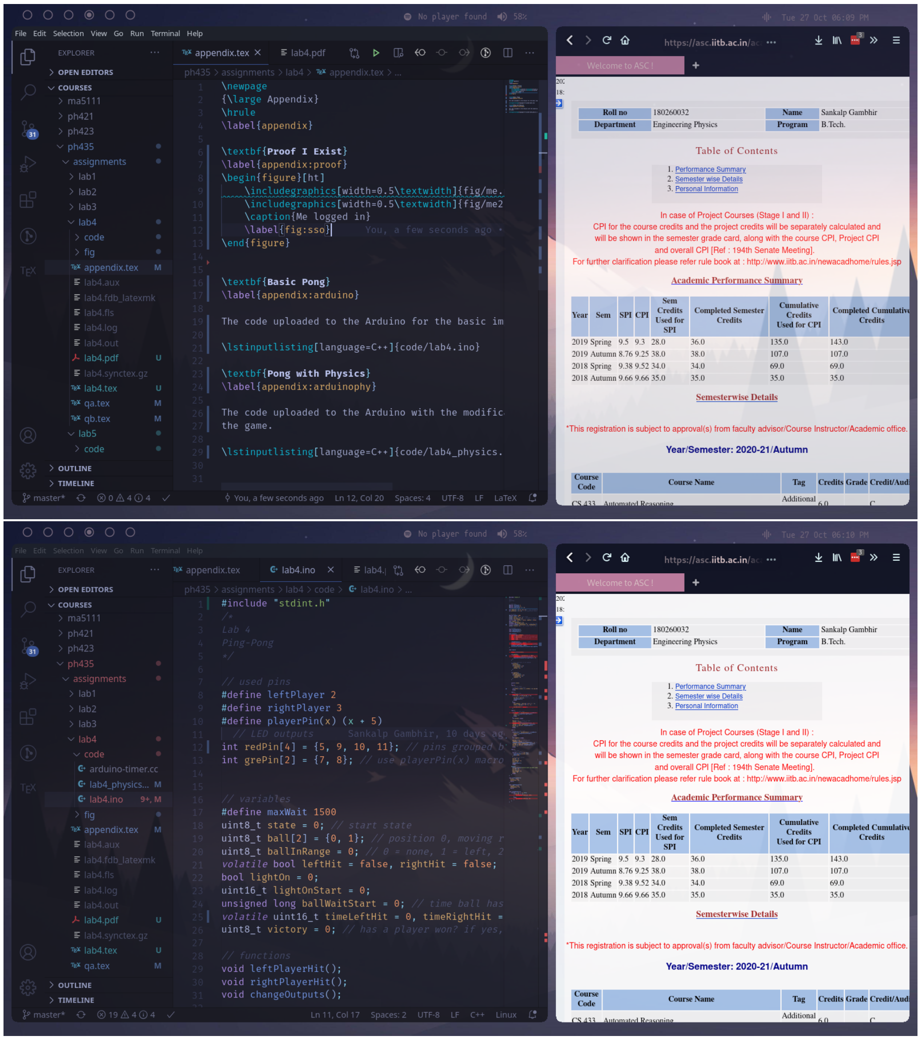
```
// <-- Indicates a skip
uint16_t delayBySpeed = 600;
//
void loop(){
    //

    switch (state)
    {
    //

    case 1:
      // game in play
      delay(600); // delay to control clock speed
      //
      break;
    //
    case 3:
      // ball in left range
      //
      if(!leftHit) break;
      //
      ball[1] = 1; // travel to right
      //
      delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
      break;
    case 4:
        // ball in right range
        //
        if(!rightHit) break;
        //
        ball[1] = -1; // travel to left
        //
        delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
        break;

   //
}

//

```

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



```
\newpage
{\large Appendix}
\hrule
\label{appendix}

\textbf{Proof I Exist}
\label{appendix:proof}
\begin{figure}[ht]
    \includegraphics[width=0.5\textwidth]{fig/me.
    \includegraphics[width=0.5\textwidth]{fig/me2
    \caption{Me logged in}
    \label{fig:sso}       You, a few seconds ago •
\end{figure}


\textbf{Basic Pong}
\label{appendix:arduino}

The code uploaded to the Arduino for the basic im

\lstinputlisting[language=C++]{code/lab4.ino}

\textbf{Pong with Physics}
\label{appendix:arduinophy}

The code uploaded to the Arduino with the modific
the game.

\lstinputlisting[language=C++]{code/lab4_physics.
```

You, a few seconds ago Ln 12, Col 20 Spaces: 4 UTF-8 LF LaTeX

Roll no	180260032	Name	Sankalp Gambhir
Department	Engineering Physics	Program	B.Tech.

Table of Contents

1. Performance Summary
2. Semester wise Details
3. Personal Information

In case of Project Courses (Stage I and II) :
CPI for the course credits and the project credits will be separately calculated and will be shown in the semester grade card, along with the course CPI, Project CPI and overall CPI [Ref : 194th Senate Meeting].
For further clarification please refer rule book at : http://www.iitb.ac.in/newacadhome/rules.jsp

Academic Performance Summary

Year	Sem	SPI	CPI	Sem Credits Used for SPI	Completed Semester Credits	Cumulative Credits Used for CPI	Completed Cumulative Credits
2019	Spring	9.5	9.3	28.0	36.0	135.0	143.0
2019	Autumn	8.76	9.25	38.0	38.0	107.0	107.0
2018	Spring	9.38	9.52	34.0	34.0	69.0	69.0
2018	Autumn	9.66	9.66	35.0	35.0	35.0	35.0

Semesterwise Details

*This registration is subject to approval(s) from faculty advisor/Course Instructor/Academic office.

Year/Semester: 2020-21/Autumn

Course Code	Course Name	Tag	Credits	Grade	Credit/Audi
CS 433	Automated Reasoning	Additional	6.0		C

```
#include "stdint.h"
/*
Lab 4
Ping-Pong
*/

// used pins
#define leftPlayer 2
#define rightPlayer 3
#define playerPin(x) (x + 5)
  // LED outputs      Sankalp Gambhir, 10 days ag
int redPin[4] = {5, 9, 10, 11}; // pins grouped b
int grePin[2] = {7, 8}; // use playerPin(x) macro


// variables
#define maxWait 1500
uint8_t state = 0; // start state
uint8_t ball[2] = {0, 1}; // position 0, moving r
uint8_t ballInRange = 0; // 0 = none, 1 = left, 2
volatile bool leftHit = false, rightHit = false;
bool lightOn = 0;
uint16_t lightOnStart = 0;
unsigned long ballWaitStart = 0; // time ball has
volatile uint16_t timeLeftHit = 0, timeRightHit =
uint8_t victory = 0; // has a player won? if yes,

// functions
void leftPlayerHit();
void rightPlayerHit();
void changeOutputs();
```

Roll no	180260032	Name	Sankalp Gambhir
Department	Engineering Physics	Program	B.Tech.

Table of Contents

1. Performance Summary
2. Semester wise Details
3. Personal Information

In case of Project Courses (Stage I and II) :
CPI for the course credits and the project credits will be separately calculated and will be shown in the semester grade card, along with the course CPI, Project CPI and overall CPI [Ref : 194th Senate Meeting].
For further clarification please refer rule book at : http://www.iitb.ac.in/newacadhome/rules.jsp

Academic Performance Summary

Year	Sem	SPI	CPI	Sem Credits Used for SPI	Completed Semester Credits	Cumulative Credits Used for CPI	Completed Cumulative Credits
2019	Spring	9.5	9.3	28.0	36.0	135.0	143.0
2019	Autumn	8.76	9.25	38.0	38.0	107.0	107.0
2018	Spring	9.38	9.52	34.0	34.0	69.0	69.0
2018	Autumn	9.66	9.66	35.0	35.0	35.0	35.0

Semesterwise Details

*This registration is subject to approval(s) from faculty advisor/Course Instructor/Academic office.

Year/Semester: 2020-21/Autumn

Course Code	Course Name	Tag	Credits	Grade	Credit/Audi
CS 433	Automated Reasoning	Additional	6.0		C

Figure 3: Me logged in

Basic Pong

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

```
/*
Lab 4
Ping-Pong
*/

// used pins
```

```
#define leftPlayer 2
#define rightPlayer 3
#define playerPin(x) (x + 5)
  // LED outputs
int redPin[4] = {5, 9, 10, 11}; // pins grouped by the same frequency`
int grePin[2] = {7, 8}; // use playerPin(x) macro instead for compile time resolution


// variables
#define maxWait 1500
uint8_t state = 0; // start state
uint8_t ball[2] = {0, 1}; // position 0, moving right
uint8_t ballInRange = 0; // 0 = none, 1 = left, 2 = right
volatile bool leftHit = false, rightHit = false;
bool lightOn = 0;
uint16_t lightOnStart = 0;
unsigned long ballWaitStart = 0; // time ball has been near player
volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
uint8_t victory = 0; // has a player won? if yes, which?

// functions
void leftPlayerHit();
void rightPlayerHit();
void changeOutputs();

void setup(){
    Serial.begin(1000000); // why not
    DDRD |= 0b11111110; // pins 1-7 as output. 0 is Tx
    DDRB |= 0b11111111; // set all pins 8-13 (and dummy bits) to output
    PORTD = 0x0;  // initialise as low
    PORTB = 0x0;

    changeOutputs(); // initial lights
    pinMode(leftPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
    pinMode(rightPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
}

void loop(){

    Serial.println(state);

    if(lightOn){
      if((millis() - lightOnStart) >= 300){
        digitalWrite(playerPin(leftPlayer), 0);
        digitalWrite(playerPin(rightPlayer), 0);
      }
    }

    // reset interrupts if not handled
    if(leftHit){
      timeLeftHit++;
      if(timeLeftHit > 3){
        timeLeftHit = 0;
        leftHit = 0;
      }
    }

    if(rightHit){
      timeRightHit++;
      if(timeRightHit > 3){
        timeRightHit = 0;
        rightHit = 0;
      }
    }


    switch (state)
    {
    case 0:
      // initial state
      // wait for first input
      if(leftHit) {state = 1; leftHit = 0; ball[0] += ball[1]; changeOutputs();}
      break;

    case 1:
      // game in play
      delay(600); // delay to control clock speed
      if(ball[0] != 0 && ball[0] != 3){
        // somewhere in the middle
        ball[0] += ball[1]; // x = x + v.dt
        changeOutputs();
        break;
      }
      else{
        ballInRange = (ball[0] == 0) ? 1 : -1;
        state = (ball[0] == 0) ? 2 : 3;
        // start counter
        ballWaitStart = millis();
      }

    case 2:
      // ball in left range
      if((millis() - ballWaitStart) >= maxWait){
        victory = playerPin(rightPlayer);
        state = 4;
        break;
      }
      if(!leftHit) break;

      ball[1] = 1; // travel to right
      leftHit = false; // reset
      digitalWrite(playerPin(leftPlayer), 1); // light up
      lightOn = true; lightOnStart = millis();
      ball[0] += ball[1]; changeOutputs();
      state = 1;
      break;

    case 3:
      // ball in right range
      if((millis() - ballWaitStart) >= maxWait){
        victory = playerPin(leftPlayer);
        state = 4;
        break;
      }
      if(!rightHit) break;

      ball[1] = -1; // travel to left
      rightHit = false; // reset
      digitalWrite(playerPin(rightPlayer), 1); // light up
      lightOn = true; lightOnStart = millis();
      ball[0] += ball[1]; changeOutputs();
      state = 1;
      break;

    case 4:
      // ball miss
      // flash 4 times;
      for(int i = 1; i <= 8; i++){
        digitalWrite(victory, i%2);
        delay(250);
      }
      // go back to initial
      ball[0] = 0;
      ball[1] = 1;
      ballInRange = 0;
      state = 0;
      changeOutputs();
      break;

    default:
      Serial.println("Invalid state variable! Defaulting to initial.");
      ball[0] = 0;
      ball[1] = 1;
      ballInRange = 0;
      state = 0;
      break;
    }
}

void leftPlayerHit(){
  leftHit = true;
  timeLeftHit = 0;
  return;
}

void rightPlayerHit(){
  rightHit = true;
  timeRightHit = 0;
  return;
}

void changeOutputs(){
    for(int i = 0; i < 4; i++){
        if(ball[0] == i){
            analogWrite(redPin[i], 255);
        }
        else{
            analogWrite(redPin[i], 0);
        }
    }
}
```

Pong with Physics

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

```
/*
Lab 4.20
Ping-Pong
With Physics
*/

// used pins
#define leftPlayer 2
#define rightPlayer 3
#define playerPin(x) (x + 5)
  // LED outputs
int redPin[4] = {5, 9, 10, 11}; // pins grouped by the same frequency
int grePin[2] = {7, 8}; // use playerPin(x) macro instead for compile time resolution


// variables
#define maxWait 1500
uint8_t state = 0; // start state
uint8_t ball[2] = {0, 1}; // position 0, moving right
uint8_t ballInRange = 0; // 0 = none, 1 = left, 2 = right
volatile bool leftHit = false, rightHit = false;
bool lightOn = 0;
uint16_t lightOnStart = 0;
unsigned long ballWaitStart = 0; // time ball has been near player
volatile uint16_t timeLeftHit = 0, timeRightHit = 0;
uint16_t delayBySpeed = 600;
uint8_t victory = 0; // has a player won? if yes, which?

// functions
void leftPlayerHit();
void rightPlayerHit();
void changeOutputs();

void setup(){
    Serial.begin(1000000); // why not
    DDRD |= 0b11111110; // pins 1-7 as output. 0 is Tx
    DDRB |= 0b11111111; // set all pins 8-13 (and dummy bits) to output
    PORTD = 0x0;  // initialise as low
    PORTB = 0x0;

    changeOutputs(); // initial lights
    pinMode(leftPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(leftPlayer), leftPlayerHit, FALLING);
    pinMode(rightPlayer, INPUT);
    attachInterrupt(digitalPinToInterrupt(rightPlayer), rightPlayerHit, FALLING);
}

void loop(){

    Serial.println(state);

    if(lightOn){
      if((millis() - lightOnStart) >= 300){
        digitalWrite(playerPin(leftPlayer), 0);
        digitalWrite(playerPin(rightPlayer), 0);
      }
    }

    // reset interrupts if not handled
    if(leftHit){
      timeLeftHit++;
      if(timeLeftHit > 3){
        timeLeftHit = 0;
        leftHit = 0;
      }
    }

    if(rightHit){
      timeRightHit++;
      if(timeRightHit > 3){
        timeRightHit = 0;
        rightHit = 0;
      }
    }


    switch (state)
    {
    case 0:
      // initial state
      // wait for first input
      if(leftHit) {state = 1; leftHit = 0; ball[0] += ball[1]; changeOutputs();}
      break;

    case 1:
      // game in play
      delay(delayBySpeed); // delay to control clock speed
      if(ball[0] != 0 && ball[0] != 3){
        // somewhere in the middle
        ball[0] += ball[1]; // x = x + v.dt
        changeOutputs();
        break;
      }
      else{
        ballInRange = (ball[0] == 0) ? 1 : -1;
        state = (ball[0] == 0) ? 2 : 3;
        // start counter
        ballWaitStart = millis();
      }

    case 2:
      // ball in left range
      if((millis() - ballWaitStart) >= maxWait){
        victory = playerPin(rightPlayer);
        state = 4;
        break;
      }
      if(!leftHit) break;

      ball[1] = 1; // travel to right
      leftHit = false; // reset
      digitalWrite(playerPin(leftPlayer), 1); // light up
      lightOn = true; lightOnStart = millis();
      ball[0] += ball[1]; changeOutputs();
      delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
      state = 1;
      break;

    case 3:
      // ball in right range
      if((millis() - ballWaitStart) >= maxWait){
        victory = playerPin(leftPlayer);
        state = 4;
        break;
      }
      if(!rightHit) break;

      ball[1] = -1; // travel to left
      rightHit = false; // reset
      digitalWrite(playerPin(rightPlayer), 1); // light up
      lightOn = true; lightOnStart = millis();
      ball[0] += ball[1]; changeOutputs();
      delayBySpeed = 300 + (millis() - ballWaitStart > 700 ? 0 : 300);
      state = 1;
      break;

    case 4:
      // ball miss
      // flash 4 times;
      for(int i = 1; i <= 8; i++){
        digitalWrite(victory, i%2);
        delay(250);
      }
      // go back to initial
      ball[0] = 0;
      ball[1] = 1;
      ballInRange = 0;
      state = 0;
      changeOutputs();
      break;

    default:
      Serial.println("Invalid state variable! Defaulting to initial.");
      ball[0] = 0;
      ball[1] = 1;
      ballInRange = 0;
      state = 0;
      break;
    }
}

void leftPlayerHit(){
  leftHit = true;
  timeLeftHit = 0;
  return;
}

void rightPlayerHit(){
  rightHit = true;
  timeRightHit = 0;
  return;
```

```
}

void changeOutputs(){
    int dir = (ball[1] > 0);
    for(int i = 0; i < 4; i++){
        if((ball[0] == i) || (ball[0] == (i - dir))){
            // leave a trail
            analogWrite(redPin[i], 255 * ((int) (ball[0] == i)));
        }
        else{
            analogWrite(redPin[i], 0);
        }
    }
}
```