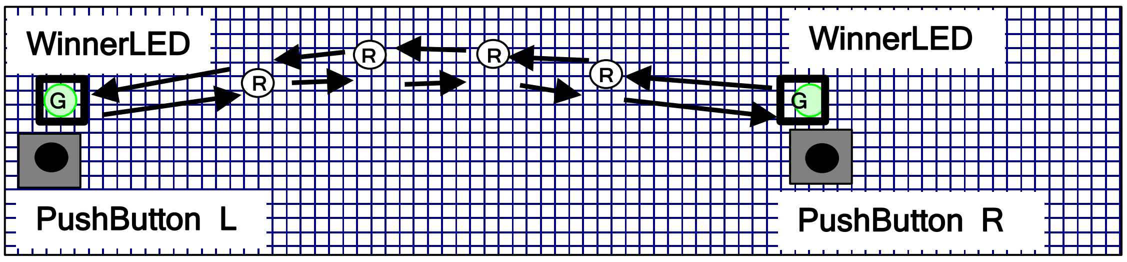
Name: _Vaishnav V. Rao____________ Roll Number: __190260045

Laboratory 3: Ping-Pong Game

Problem: LED ping-pong game

The objective of this exercise is to implement a ping-pong game as a finite-state-machine on your bread board. You will use four red, two green LEDs to indicate the position of the ball as it flies back and forth on the table. Two push-button switches are used as the 'bats' for hitting the ball.
The LEDs are arranged on the breadboard as shown in the diagram below. Each red LED (R) represents the position of a ping-pong ball, and the two green LEDs (G) represent the status of the players.
(Please see diagram on page 2 for how to connect the pushbutton switches for pull-up/pull-down)



Part A: Simple serve and volley..8

Game Specification:

A simple game proceeds as follows: there is a pushbutton at each end of the LED chain controlled by player L and R.

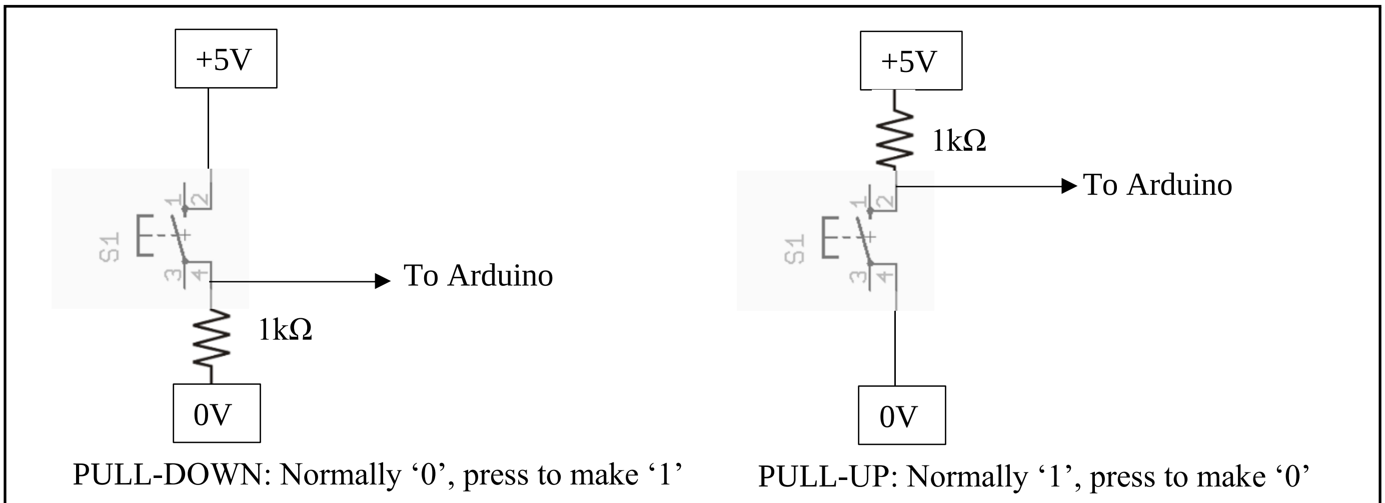
1. Initial state: player L has the ball.
2. When player L pushes the button to 'serve' the ball, the game starts and the LED's light up in sequence indicating the ping-pong ball is traveling across the 'table'.
3. When it reaches the opposite end, player R must push their return button at just the right time to bounce the ball back. Set a maximum allowed time for hitting the return push-button after the ball has landed at the last spot.
4. The Green status LED indicates successful return-of-serve. If a player presses their button to successfully return the ball within the allowed time, their green LED should light momentarily.
5. After successful return of serve, the ball's return path is indicated by the LED's lighting up in the reverse sequence (R to L).
6. If a player misses the return, the other player has won the game – the green LED corresponding to the winning player should blink four times.
7. The game continues until either player misses their return.

Note:

1. After L serves the ball, R can return it only when it reaches the right end of the table. Hitting the return button *before* the nearest LED is lit has no effect
2. In the Reset state, you can assume that Player L serves the ball.

Note: Your completed solution document must contain:

1. State diagram of your finite state machine.
2. Algorithm coded into Arduino program, please use fixed width font (`Courier new`) when putting code in your solution document. Code must be fully documented.
3. Working demo. Video uploaded to google drive, and a screenshot with SSO login showing original solution.


+5V
S1
To Arduino
1kΩ
0V
PULL-DOWN: Normally '0', press to make '1'
+5V
1kΩ
To Arduino
S1
0V
PULL-UP: Normally '1', press to make '0'

Scheme for push-button connections, use either Pull-Up or Pull-down.

A.1) Reading switches 2.5

Connect your two switches. Write code blocks using Interrupt Service Routines (ISR) to recognize the button press in your Arduino program.
Recall that ISR should contain minimum code execution – only shared variables (declared volatile the code should be modified. The main `loop()` function can check values asynchronously and display status.

Demo Link:
https://drive.google.com/file/d/1RI6deu7WvSOEepybXk-MY5ofn8OmChuD/view?usp=sharing

Code:

```
const byte ledPinL = 4;
const byte ledPinR = 13;
const byte switchL = 2;
const byte switchR = 3;

volatile byte flagL=LOW;
volatile byte flagR=LOW;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPinL, OUTPUT);
  pinMode(ledPinR, OUTPUT);

  pinMode(switchL, INPUT_PULLUP);
  pinMode(switchR, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(switchL), isr_L, FALLING);
//interrupt for left button press
  attachInterrupt(digitalPinToInterrupt(switchR), isr_R, FALLING);
//interrupt for right button press

}

void loop() {
  if(flagL) {
    digitalWrite(ledPinL, HIGH); //lights up left LED momentarily if
flagL=HIGH
    delay(300);
    digitalWrite(ledPinL, LOW);
    flagL= LOW; //resets flagL
  }
  if(flagR) {
    digitalWrite(ledPinR, HIGH); //lights up right LED momentarily if
flagR=HIGH
    delay(300);
    digitalWrite(ledPinR, LOW);
    flagR= LOW; //resets flagR
  }
}

void isr_L() {
  flagL= HIGH; //interrupt routine for left button press
}

void isr_R() {
  flagR= HIGH; //interrupt routine for right buton press
}
```

Name: _Vaishnav V. Rao____________ Roll Number: __190260045

A.2) Set up the circuits 2.5

Connect set of red and green LEDs on your breadboard. Write code to make the LED's light up sequentially.
The δt between sequence of LED's lighting up is fixed for this module (choose ~ 1 sec). i.e. the speed of the ball as it travels across your breadboard 'table' is fixed. But you should set this as variable, because it will change in Part B. Wire this up corresponding to breadboard layout shown on Page 1. Marks for neat connections!

Demo Link:
https://drive.google.com/file/d/1SCeTCqR2bf_4dO7v3WK8mxYzvnlVK0gW/view?usp=sharing

Code:

```
const byte red1= 6;
const byte red2= 8;
const byte red3= 10;
const byte red4= 12;
int t= 1000; // sets the speed of lighting up
void setup() {
  // put your setup code here, to run once:
  pinMode(red1, OUTPUT);
  pinMode(red2, OUTPUT);   //setting up output
  pinMode(red3, OUTPUT);
  pinMode(red4, OUTPUT);
}

void loop() {
  for(int i=6; i<= 12; i=i+2) {
    digitalWrite(i,HIGH);  //forward sequence of lighting
    delay(t);
    digitalWrite(i,LOW);
  }
  delay(t);

  for(int i=12; i>= 6; i=i-2) {
    digitalWrite(i,HIGH);  //backward sequence of lighting
    delay(t);
    digitalWrite(i,LOW);
  }
  delay(t);
}
```

A.3) Putting it together 3

Combine code from modules 1.1 and 1.2 to demonstrate a simple serve-and-volley ping pong game: Start with detecting L key press → LEDs light in sequence from left to right → R key press detection enabled → within some pre-defined timeout, if R key press is detected, reverse the LED lighting sequence from R to L. If timeout expires, player loses. Use green LED to indicate win/lose status.
You may find it useful to selectively use the functions `interrupts()` and `nointerrupts()` for your program to selectively become sensitive to L key press or R key press in the sequential operations above: when the ball is in 'flight' on the LEDs across the breadboard, keeping the respond pushbutton pressed should not affect the game's operation!

Demo Link:
https://drive.google.com/file/d/11cFP6znspacG4go48gaPbS6T2K7h5sVI/view?usp=sharing

Code:

```
const byte ledPinL = 4;
const byte ledPinR = 13;
const byte switchL = 2;
const byte switchR = 3;

volatile byte flagL=LOW; //indicates left button press
volatile byte flagR=LOW; //indicates right button press

const byte red1= 6;
const byte red2= 8;
const byte red3= 10;
const byte red4= 12;
int t= 500; // sets the speed of lighting up
int timeout=1000; //decides the timeout for a missed shot
int dt1r, dt2r, dt1l, dt2l; //dynamic time variables to decide hit or miss
volatile byte startflag=HIGH;   //HIGH if game is not yet started; LOW if
game is in progress

void setup() {
  // put your setup code here, to run once:
  pinMode(red1, OUTPUT);
  pinMode(red2, OUTPUT);   //setting up output
  pinMode(red3, OUTPUT);
  pinMode(red4, OUTPUT);

  pinMode(ledPinL, OUTPUT);  //green LED pins
  pinMode(ledPinR, OUTPUT);

  pinMode(switchL, INPUT_PULLUP); //interrupt pins
  pinMode(switchR, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(switchL),isr_L,           FALLING);
//interrupt for left button press
  attachInterrupt(digitalPinToInterrupt(switchR),isr_R,           FALLING);
//interrupt for right button press
}

void forward() {                  //forward sequence of lighting
  for(int i=6; i<= 12; i=i+2) {
       digitalWrite(i,HIGH);
       delay(t);
       digitalWrite(i,LOW);
     }
}

void backward() {                  //backward sequence of lighting
  for(int i=12; i>= 6; i=i-2) {
        digitalWrite(i,HIGH);
        delay(t);
        digitalWrite(i,LOW);
      }
}

void leftwin() {
  for(int j=0; j<4; j++) {        //Indicates left player has won
    digitalWrite(ledPinL, HIGH);
    delay(200);
    digitalWrite(ledPinL, LOW);
    delay(200);
   }
  startflag=HIGH;     //resetting game variables
  dt2r=dt1r;
  dt2l=dt1l;
  flagL=flagR=LOW;
}

void rightwin() {
  for(int j=0; j<4; j++) {        //Indicates right player has won
     digitalWrite(ledPinR, HIGH);
     delay(200);
     digitalWrite(ledPinR, LOW);
     delay(200);
   }
  startflag=HIGH;     //resetting game variables
  dt2r=dt1r;
  dt2l=dt1l;
  flagL=flagR=LOW;
}

void loop() {

  if( ((dt2l-dt1l)> 0 && (dt2l-dt1l)<timeout) || (startflag && flagL) ) {
    //enters the if either left player's shot is within timeout if left
button is pressed at the beginning of game

    flagL=LOW;  //resets left button flag
    startflag=LOW; //sets to LOW to indicate game is in progress
    digitalWrite(ledPinL, HIGH); //lights up left LED momentarily to indicate
successful hit
    delay(300);
    digitalWrite(ledPinL, LOW);

    forward();  // ball moves left to right
    if(flagR) {flagR=LOW;}  // if right button has been pressed during flight
time of ball, it gets reset
    dt1r=millis();
    delay(t); // buffer time for right player to press button

    if(flagR){dt2r=millis(); flagR=LOW;}  //if right player has pressed the
button, notes down time
    else{dt2r=dt1r;}   //else sets the times equal to ensure that return of
ball does not occur
```

```
  }

  else if(!startflag) {rightwin();} //right player wins if left player's
shot is unsuccessful

  if( ((dt2r-dt1r)> 0 && (dt2r-dt1r)<timeout) ) {
    // enters if right player's shot is within timeout

    flagR=LOW; //resets right button flag
    digitalWrite(ledPinR, HIGH); //lights up right LED momentarily to
indicate successful hit
    delay(300);
    digitalWrite(ledPinR, LOW);

    backward(); //ball moves to right to left
    if(flagL) {flagL=LOW;}  //if left button has been pressed during flight
time of ball, it gets reset
    dt1l=millis();
    delay(t);   //buffer time for left player to press the button

    if(flagL){dt2l=millis(); flagL=LOW;}  //if left player has pressed the
button, notes down time
    else {dt2l=dt1l;}    //else sets the times equal to ensure that return
of ball does not occur
  }

  else if(!startflag) {leftwin();}   //left player wins if right player's
shot is unsuccessful

}

void isr_L() {
  flagL= HIGH; //interrupt routine for left button press
}

void isr_R() {
  flagR= HIGH; //interrupt routine for right button press
}
```

Part B: Physics and table tennis...12

After completing Part 1, add an element of physics into the game. Most of the code from Part 1 can be reused.
The 'physics' to be introduced into the game simulates the action of a real player hitting a real table tennis ball: the delay between the ball arriving in your court and the time at which you hit return determines the speed of the return.
[Think about it: if the ball has nearly stopped by the time you hit it with the same force as in Part A, you will give it a greater momentum travelling back compared to the case when your applied impact force is used to first stop the ball and then return it across the court]
Translate this effect into our breadboard game.

Set a maximum time that a player is allowed to wait before hitting return after the ball has reached their court. If the player waits till close to the limit – the ball returns at higher speed (i.e. the red ball position LED's light up faster in sequence with smaller δt between each one). If the serve is returned immediately upon the ball landing at a player's end, the return speed does not change much. In the first case, the ball has nearly come to a stop so all the force of the bat is used to give it (large) reverse momentum.

B.1) Modify ISR codes 3

Modify your ISR code and program written in Module 1.1 to measure the time lapse between a time *t=0* and the key press.

Demo Link:
https://drive.google.com/file/d/1dI4wGlmQ34dPeupUMzSWOGKYAcvHduxP/view?usp=sharing

Code:

```
const byte ledPinL = 4;
const byte ledPinR = 13;
const byte switchL = 2;
const byte switchR = 3;

volatile byte flagL=LOW; //variables go high when buttons are pressed
volatile byte flagR=LOW;

int t1=millis(); //this is t=0
int t2;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPinL, OUTPUT);
  pinMode(ledPinR, OUTPUT);

  pinMode(switchL, INPUT_PULLUP);
  pinMode(switchR, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(switchL), isr_L, FALLING);
//interrupt for left button press
  attachInterrupt(digitalPinToInterrupt(switchR), isr_R, FALLING);
//interrupt for right button press
```

```
  Serial.begin(9600); //initialises serial monitor

}

void loop() {
  if(flagL) {
    t2=millis(); //time when left button is pressed
    Serial.println(t2-t1); //outputs time between left button press and t=0

    digitalWrite(ledPinL, HIGH); //lights up left LED momentarily if 
flagL=HIGH
    delay(300);
    digitalWrite(ledPinL, LOW);
    flagL= LOW; //resets flagL
  }

  if(flagR) {
    t2=millis(); //time when right button is pressed
    Serial.println(t2-t1);  //outputs time between right button press and 
t=0

    digitalWrite(ledPinR, HIGH); //lights up right LED momentarily if 
flagR=HIGH
    delay(300);
    digitalWrite(ledPinR, LOW);
    flagR= LOW; //resets flagR
  }
}

void isr_L() {
  flagL= HIGH; //interrupt routine for left button press
}

void isr_R() {
  flagR= HIGH; //interrupt routine for right buton press
}
```

B.2) Calculate the time elapsed 3

t=0 occurs when the ball lands on the last LED nearest to the switch. Modify the code written for module 1.2, adding a variable to define *t=0* accordingly, at the end of sequence of LED's lighting up, and measure the time lapse between *t=0* and the time of the key press to return the ball.

Demo Link:
https://drive.google.com/file/d/1uh9SxPsdzFm3c_JQMl4j2SMsMZOTFvaH/view?usp=sharing

Code:

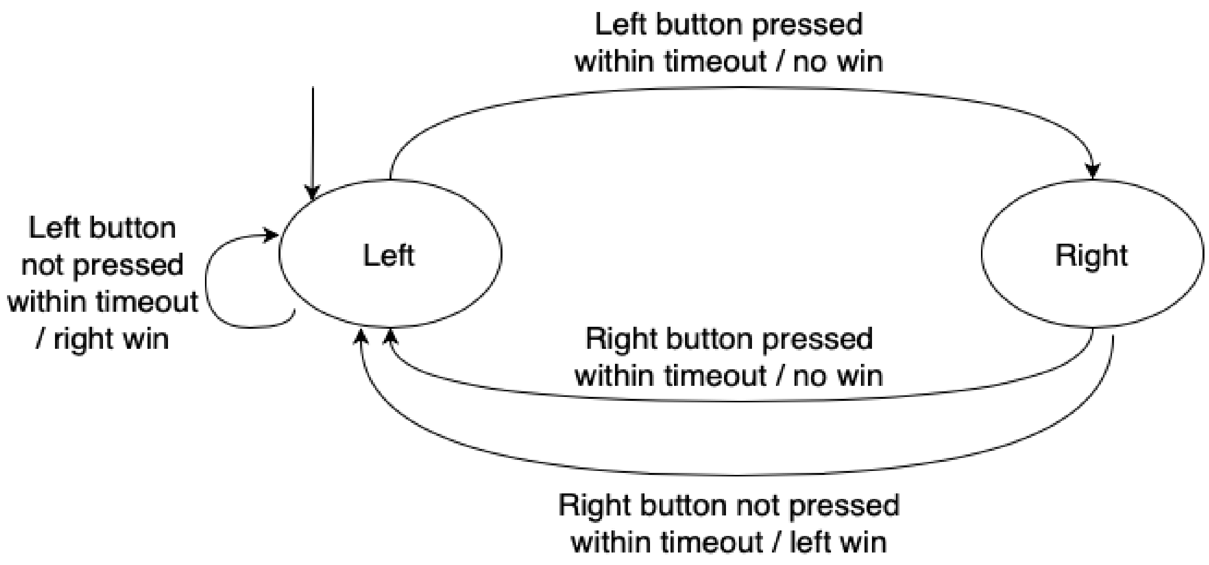
```
const byte ledPinL = 4;
const byte ledPinR = 13;
const byte switchL = 2;
const byte switchR = 3;

const byte red1= 6;
const byte red2= 8;
const byte red3= 10;
const byte red4= 12;
int tspeed= 1000; // sets the speed of lighting up
int timeout= 1000;

int t1, t2;  //to measure time differences

volatile byte flagL=LOW; //variables go high when buttons are pressed
volatile byte flagR=LOW;

void setup() {
  // put your setup code here, to run once:
  pinMode(red1, OUTPUT);
  pinMode(red2, OUTPUT);   //setting up output
  pinMode(red3, OUTPUT);
  pinMode(red4, OUTPUT);

  pinMode(ledPinL, OUTPUT);
  pinMode(ledPinR, OUTPUT);

  pinMode(switchL, INPUT_PULLUP);
  pinMode(switchR, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(switchL),      isr_L,      FALLING);
//interrupt for left button press
  attachInterrupt(digitalPinToInterrupt(switchR),      isr_R,      FALLING);
//interrupt for right button press
  Serial.begin(9600); //initialises serial monitor
}

void forward() {
  for(int i=6; i<= 12; i=i+2) {
    digitalWrite(i,HIGH);  //forward sequence of lighting
    delay(tspeed);
    digitalWrite(i,LOW);
  }
  t1= millis();
}
```

```
void backward() {
  for(int i=12; i>= 6; i=i-2) {
    digitalWrite(i,HIGH);  //backward sequence of lighting
    delay(tspeed);
    digitalWrite(i,LOW);
  }
  t1= millis();
}
void loop() {

  forward();  //ball moves left to right
  if(flagR) {flagR=LOW;}   //if right button has been pressed in between,
reset it to not get negative times
  delay(timeout);  // waits for this much time for user to press right button

  if(flagR) {  //if right button has been pressed in the duration of timeout
    Serial.println(t2-t1);   //prints time difference between right button
press and right LED going off
    flagR=LOW; //reset right button flag
  }

  backward();  //ball moves right to left
  if(flagL) {flagL=LOW;}  //if left button has been pressed in between, reset
it to not get negative times
  delay(timeout); // waits for this much time for user to press left button

  if(flagL) {  //if left button has been pressed in the duration of timeout
    Serial.println(t2-t1);   //prints time difference between left button
press and left LED going off
    flagL=LOW;  //reset left button flag
  }

}

void isr_L() {
  flagL= HIGH; //interrupt routine for left button press
  t2=millis();  // notes time of left button press
}

void isr_R() {
  flagR= HIGH; //interrupt routine for right buton press
  t2=millis();  //notes time of right button press
}
```

B.3) Complete the game!..6

Combine code from modules B.1 and B.2. Adjust *dt* between sequential lighting of LED's based on the time lapse between *t=0* when ball lands at the last LED, and the corresponding return key is pressed. If time lapse is small, *dt* does not change much, if time lapse is large, *dt* is reduced i.e. the ball returns at higher speed. Include some reasonable timeout – if the ball is not returned within the timeout period, the player loses, like in a real game of ping pong!

Final FSM diagram: (This is for the simple serve and volley game)


Left button pressed within timeout / no win
Left button not pressed within timeout / right win
Left
Right
Right button pressed within timeout / no win
Right button not pressed within timeout / left win

Here, the states signify which side of the table, the ball is at. Game starts off with the ball on the left and returns there every time a game is finished.

The statements on the arrows indicate the conditions of state transitions and the corresponding outputs. The outputs are after '/'.

The intermediate states of ball travelling are not shown as they aren't interrupted and are just redundant in the FSM diagram.

We can perform encoding as follows:
States
Left =0; Right= 1;

Output
No win = 00; Left win = 10; Right win = 01;

Inputs:
Each of the four cases can be encoded with 00,01,10,11

Function used to determine return speed:

`tspeed` is set by default to 1000ms. This speed of the ball is used in the initial service.

Let time lapsed between LED going off and button being pressed be `dt`.

If `dt=0`, LEDs light up with delay 1000ms (slow speed of return)
If `dt= timeout` (maximum lapsed time between LED going off and button being pressed), LEDs light up with delay 100ms (fast speed of return)

We take a linear function between the `newspeed` and `dt` to get
`newspeed = tspeed - (dt*(tspeed-100)/timeout)`

Demo Link:
https://drive.google.com/file/d/173C5RC71EUfJAOAAwPjEWcxvKaahEt9G/view?usp=sharing

Code:
```
const byte ledPinL = 4;
const byte ledPinR = 13;
const byte switchL = 2;
const byte switchR = 3;

volatile byte flagL=LOW; //indicates left button press
volatile byte flagR=LOW; //indicates right button press

const byte red1= 6;
const byte red2= 8;
const byte red3= 10;
const byte red4= 12;
int tspeed= 1000; // sets the default speed of lighting up
int timeout=1000; //decides the timeout for a missed shot
int dt1r, dt2r, dt1l, dt2l, dtr, dtl=0; //dynamic time variables to decide
hit or miss
volatile byte startflag=HIGH;   //HIGH if game is not yet started; LOW if
game is in progress

void setup() {
  // put your setup code here, to run once:
  pinMode(red1, OUTPUT);
  pinMode(red2, OUTPUT);   //setting up output
  pinMode(red3, OUTPUT);
  pinMode(red4, OUTPUT);

  pinMode(ledPinL, OUTPUT);  //green LED pins
  pinMode(ledPinR, OUTPUT);

  pinMode(switchL, INPUT_PULLUP); //interrupt pins
  pinMode(switchR, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(switchL), isr_L, FALLING);
//interrupt for left button press
  attachInterrupt(digitalPinToInterrupt(switchR), isr_R, FALLING);
//interrupt for right button press
}

int speed_det(int dt) {
  //function determines return speed
  Float newspeed = float(tspeed) -(float(dt)*((float(tspeed)-
100.0)/float(timeout)));
  //pressing immediately sets dt between LEDs to tspeed
  //pressing after timeout sets dt between LEDs to 100ms (this is max speed
of return)
  return(newspeed);
}

void forward(int dt) {
  //forward sequence of lighting
  int newspeed= speed_det(dt);    //calls the function to determine return
speed
  // here float to int implicit type conversion has occured
  for(int i=6; i<= 12; i=i+2) {
       digitalWrite(i,HIGH);
       delay(newspeed);
       digitalWrite(i,LOW);
      }
   dt1r=millis();
}

void backward(int dt) {
  //backward sequence of lighting
  int newspeed= speed_det(dt);     //calls the function to determine return
speed
  // here float to int implicit type conversion has occured
  for(int i=12; i>= 6; i=i-2) {
        digitalWrite(i,HIGH);
        delay(newspeed);
        digitalWrite(i,LOW);
       }
   dt1l=millis();
}

void leftwin() {
  //Indicates left player has won
  for(int j=0; j<4; j++) {
    digitalWrite(ledPinL, HIGH);
    delay(200);
    digitalWrite(ledPinL, LOW);
    delay(200);
   }
  //resetting game variables
  startflag=HIGH;
  dt2r=dt1r;
  dt2l=dt1l;
  dtr=0;
  dtl=0;
  flagL=flagR=LOW;
}

void rightwin() {
  //Indicates right player has won
  for(int j=0; j<4; j++) {
     digitalWrite(ledPinR, HIGH);
     delay(200);
     digitalWrite(ledPinR, LOW);
     delay(200);
   }
  //resetting game variables
  startflag=HIGH;
  dt2r=dt1r;
  dt2l=dt1l;
  dtr=0;
  dtl=0;
  flagL=flagR=LOW;
}

void loop() {

  if( ((dtl)> 0 && (dtl)<timeout) || (startflag && flagL) ) {
    //enters if either left player's shot is within timeout if left button
is pressed at the beginning of game

    flagL=LOW;  //resets left button flag
    startflag=LOW; //sets to LOW to indicate game is in progress
    digitalWrite(ledPinL, HIGH); //lights up left LED momentarily to
indicate successful hit
    delay(300);
    digitalWrite(ledPinL, LOW);

    forward(dtl);  // ball moves left to right
    if(flagR) {flagR=LOW;}  // if right button has been pressed during
flight time of ball, it gets reset
    delay(timeout); // buffer time for right player to press button

    if(flagR){dtr=dt2r-dt1r; flagR=LOW;}  //if right player has pressed the
button within timeout, updates dtr
    else{dtr=0;}   //else sets dtr=0 to ensure that return of ball does not
occur
  }

  else if(!startflag) {rightwin();} //right player wins if left player's
shot is unsuccessful

  if( ((dtr)> 0 && (dtr)<timeout) ) {
    // enters if right player's shot is within timeout

    flagR=LOW; //resets right button flag
    digitalWrite(ledPinR, HIGH); //lights up right LED momentarily to
indicate successful hit
    delay(300);
    digitalWrite(ledPinR, LOW);

    backward(dtr); //ball moves to right to left
    if(flagL) {flagL=LOW;}  //if left button has been pressed during flight
time of ball, it gets reset
    delay(timeout);   //buffer time for left player to press the button

    if(flagL){dtl=dt2l-dt1l; flagL=LOW;}  //if left player has pressed the
button within timeout, updates dtl
    else {dtl=0;}    //else sets the dtl=0 to ensure that return of ball
does not occur
  }

  else if(!startflag) {leftwin();}  //left player wins if right player's
shot is unsuccessful

}

void isr_L() {
  //interrupt routine for left button press
  flagL= HIGH;
  dt2l=millis(); //notes time of left button press
}
```

```
void isr_R() {
  //interrupt routine for right buton press
  flagR= HIGH;
  dt2r=millis();  //notes time of right button press
}
```

Note: In each part, include photos / screenshots / videos as appropriate