4 Player Pong Game

PIC16877A Microcontroller Implementation

Paolo Dela Peña, Jashandeep S. Bhuller, Danielle Espiritu, Takeshi Isobe, Julio Simeon
Computer Systems Organization Laboratory *(LBYCP1D EQ)*
Electronics and Communications Department
De la Salle University, Manila, Philippines

***Abstract*— Microcontrollers can be used for various applications. Not only are they used for clocks, and measurements, but they can also be used to integrate games. They offer a wide range features that can be utilized to turn a simple game into something more creative and interesting. By programming the microcontroller, and modifying the connections and components in a circuit, one can also come up with a gaming application. The group thought of integrating the concepts of the microcontroller in a game called "Pong". This is actually similar to tennis in terms of gameplay. There were also some modifications made to the game in order to have more players. It basically has a total of four players, where the first two players control their paddles by a switch, and the other players utilize an interface between the computer and the game for controlling their paddles. The group was successfully able to simulate, and make the hardware part of the project. The appropriate code, components, and connections were made to be able to run the game successfully.**

I. Introduction

Pong was one of the first computer games to be created in the gaming industry. This basically has "tennis like" features where users have a paddle and a ball. The main goal of the game is to defeat your opponent by being the first to reach 10 points. When the opponent misses a ball, a player receives a point, and the game continues until one of the player wins. A user has a choice of going against a computer controlled paddle, or another player. Originally developed by Allan Alcorn, this game was released in 1972 by Atari corporations. After years, Pong became a huge success, and became the first commercially successful game. Atari then released a home edition of Pong (the first version was played on Arcade machines) which sold 150,000 units. Nowadays, this game is considered to have started the video games industry,and as it proved, video games market can produce significant revenues.

As observed by the group, the usual projects made about microprocessors were related to clocks, or as measurements devices, but we wanted to come up with something different. The group then thought of making a project related to entertainment, and we choose the game "Pong" because of practicality, attainability, and benefits. With the knowledge we have about microprocessors, different simulation softwares, programming and various circuit components, we will create a game similar to "Pong" with some modifications to make it even better for other users.

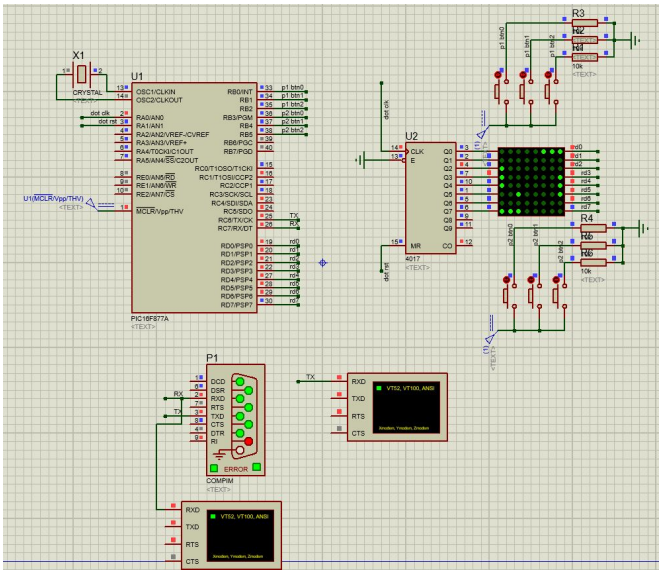
II. Gameplay

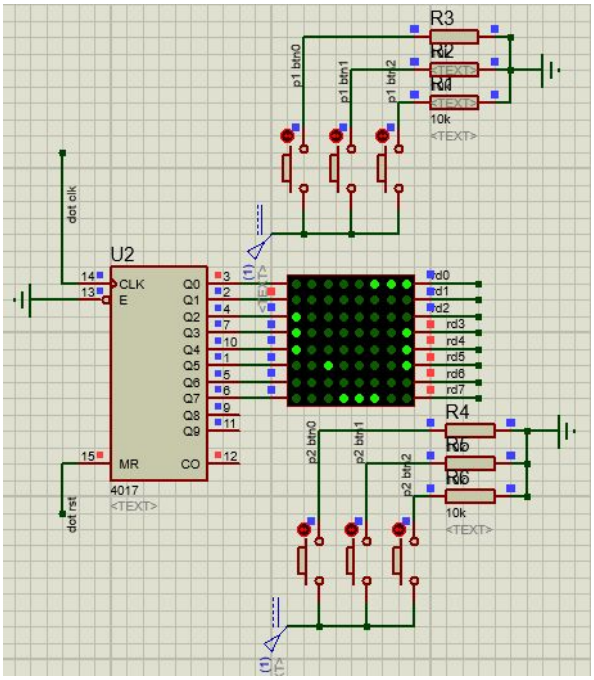
The main function of the project will be to serve as a gaming application of the modified "Pong" game for the users. There will basically be a total of 4 players, each of them with a paddle they can control. The first two players will have a manual button for controlling the movement of their paddles, while the other players will be using a computer software to move their paddles. The players will also have a button for starting the game. The group added the computer interface in order to enhance the features of the game. When the ball is missed, the game starts again, and continues.

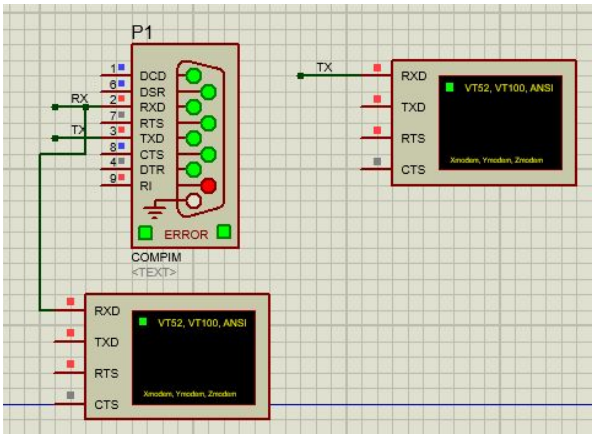
Some of the components used were the power source to turn on the game, switches for controlling the amount of players and gameplay, the resistors, a decade counter for display driving, the microprocessor for the program, and an LED matrix for the display.

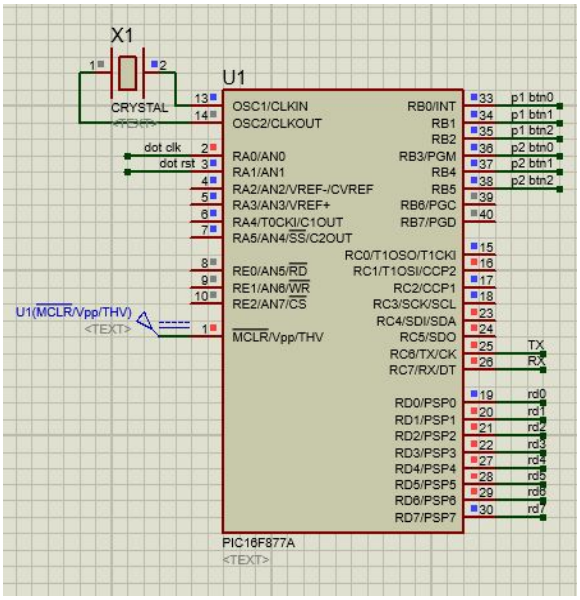
III. Simulations

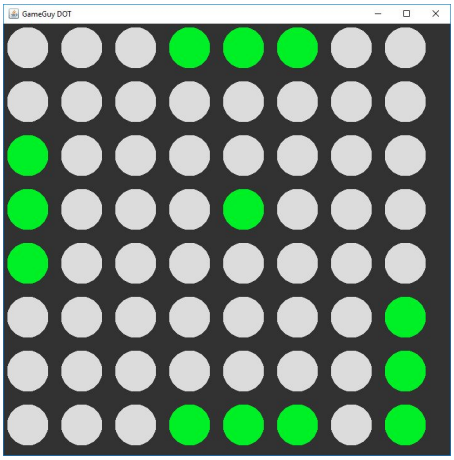
The group used Proteus to make the appropriate connections for the various components, and a simulation of the game. The group used a serial port to simulate the transmission of data between the microcontroller and the PC. The virtual terminal is also used for debugging purposes in order to determine that the correct data is being sent and received. The group also created a dedicated java program that would serve as display for the 3rd and 4th players.







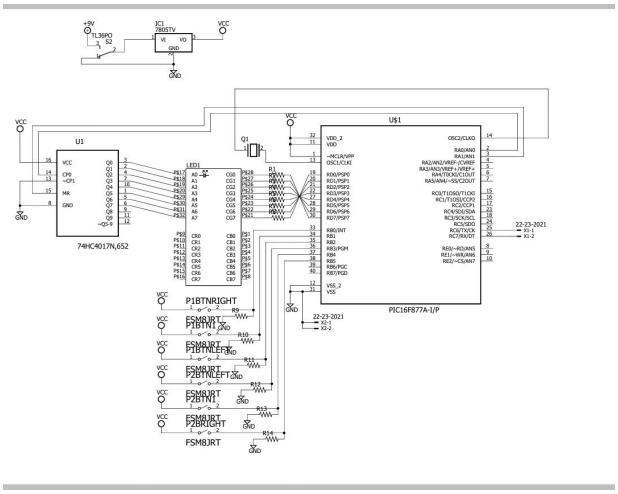




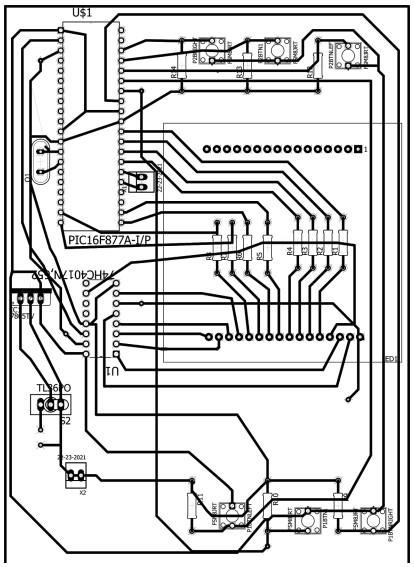
IV. Circuit Implementation

The circuit was implemented through the use of a breadboard and connecting wires. The group utilized the PIC16F877A microcontroller as well as an LED dot display An crystal oscillator was also used along with a USB adapter to power the circuit with a constant 5V source.

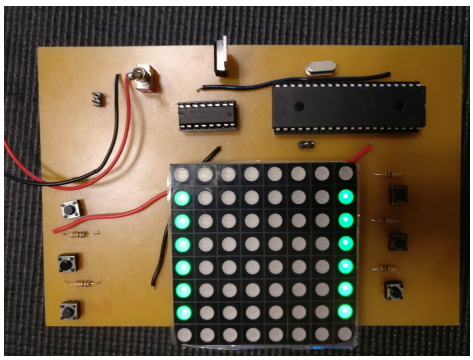
A. Schematic

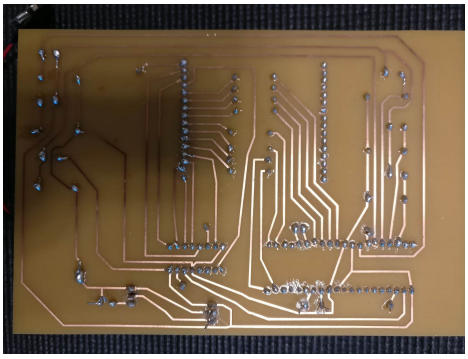


B. PCB Layout



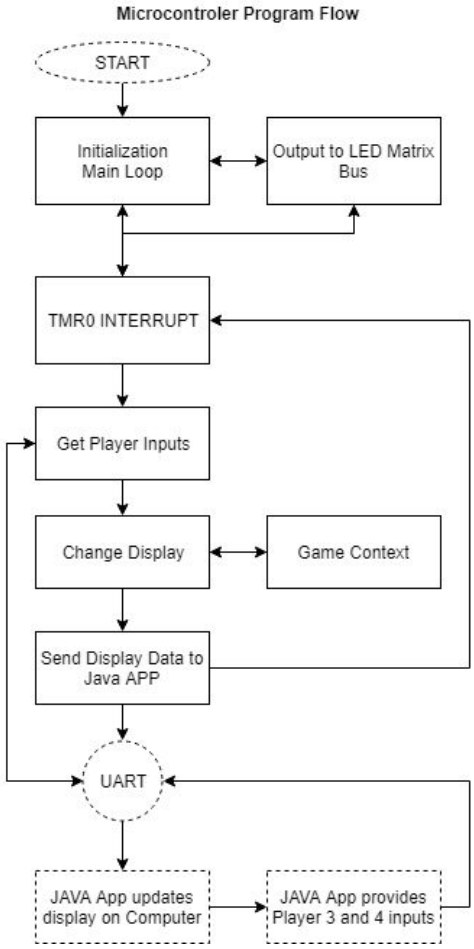
C. *Prototype*





V. Code

A. *Block Diagram*



B. *Program Flow*

a. *Main Loop*

The main loop's only purpose is to initialize the the PIC as well as drive the LED dot matrix display. In the circuit, the a row of leds is connected to PORTD of the PIC while their common anode is controlled by a decade counter. This implies that only one row is activated at a time and the LEDs that are lit up are determined by PORTD. An array contains the values that must be set to PORTD for every row of the display. Each bit would correspond to the state of a single LED in that row. The main loop will then cycle these rows indefinitely and output the desired image on the LED matrix.

b. *TMR0 Interrupt*

The TMR0 interrupt is responsible to update the displays when necessary. This function will change the values of the display array at certain intervals of time based on the current state of the game as well as the inputs of the players.

The TMR0 interrupt is also responsible of regularly sending the Java companion app the display data so the a representation of the LED display can be seen on the computer for the use of players 3 and 4.

c. *Get Player Inputs*

The Player inputs are retrieved from two locations: PORTB and the JAVA app for players 1-2 and 3-4 respectively. A Pin at PORTB corresponds to a button input of a player. The program will then test the states of these pins and save it to an array corresponding to that player. For players 3-4, the JAVA app would have sent a byte of data containing their inputs where each bit corresponds to a button. The program then takes all of this data to determine the next frame of the display

d. *Change Display*

Once all the inputs have been processed, the program will then assign new values to the display array which will be later sent to PORTD. When the program returns to the Main Loop, the display will now show the next frame.

e. *Send Display Data to JAva APP*

To send the display data, the array is converted to a HEX string and sent using the Universal asynchronous Receiver-Transmitter (UART) module of the microcontroller. In return, the Java app will send back the inputs of players 3-4.

The interrupt that triggered the sending of the display data is the same interrupt that triggers the java app to send the player inputs. This was done so that there is no mismatch or extra data being sent to the buffer of the UART module and ensuring there is no delay in player inputs.

** complete codes and HEX files of the micro-controller and the Java companion app can be found in Section VIII.*

```
char dot_display[10] = {0x0F,0xfd,0xfb,0xf7,0xef,0xdf,0xbf,0x7f,'\n','\0'};
int i=0, pucktest=0;
char temp[3];
char tempin[20];
char extra=0;
char sendHEXdata[21];

//player input vairables
char player1_btns[3] = {0,0,0}, player2_btns[3] = {0,0,0};
char player34_inputs=0;
char p34_delimiter[5] = {':', 'E', 'N', 'D', '\0'};
char puck1 = 0x1c, puck2 = 0x38, puck3=1, puck4=6;

//game variables
char ball_xspeed=0, ball_yspeed=0, ball_xposition = 0x20, ball_yposition=1;
int paddlespeed = 1220/10;
int puckspeed = 1220/6;
char serve=0;

char p1=0, p2=0, p3=0, p4=0;  //player enables
char mode=0;
/*
0 = player select
1 = playmode
*/
char auto_mode=0;

//control variables
int tmr0_i=0, count=0, tmr0_mark=1221;

void output_to_dot_display();
void updategame();
void drawPucks();
void puckupdate();
void movePlayerPucks();
void getPlayerInputs();

void player_select();
void playerselecttimer();

void game();

void interrupt(){

     if(intcon.tmr0if){
         if(mode==0){
             player_select();
         }
         else if(mode==1){
             game();
         }
         if(tmr0_i%40==0){

strcpy(sendHEXdata,"DOT:");
             for(i=0; i<8; i++){

ByteToHex(dot_display[i], temp);
                 strcat(sendHEXdata, temp);
             }

uart1_write_text(sendHEXdata);

              if(uart1_data_ready()){
                   player34_inputs = uart1_read();
              }


         }
         tmr0_i++;
         intcon.tmr0if = 0;
     }
}

void game(){
     if(tmr0_i%paddlespeed==0){
          movePlayerPucks();
     }
     if(tmr0_i%puckspeed==0){
         puckupdate();
     }
     if(tmr0_i==1220){

          tmr0_i=0;
     }
     else if(tmr0_i%20==0){ //60Hz refresh
         updategame();
         getPlayerInputs();

         if((portb.f1 && portb.f0 && portb.f2)
          || (portb.f3 && portb.f4 && portb.f5)
          || ((player34_inputs&0x07)==0x07)
          || ((player34_inputs&0x38)==0x38)){
             mode=0;
             p1=0;
             p2=0;
             p3=0;
             p4=0;

          }
     }
}

  void player_select(){
       if(tmr0_i%80==0){

           //drawPucks();
           if(p1)
               dot_display[0]=0x81;
           else
               dot_display[0]=0xff;
           if(p2)
               dot_display[7]=0x81;
           else
               dot_display[7]=0xff;
```

```
112.
113.            extra = 0xff;
114.            if(p3==1)
115.                extra-=0x80;
116.            if(p4==1)
117.                extra-=0x01;
118.            for(i=1;i<7;i++){
119.                dot_display[i]=extra;
120.            }
121.            //if(tmr0_i!=1220)
122.                playerselecttimer();
123.        }
124.        if(tmr0_i%304==0){
125.
126.        }
127.        if(tmr0_i%20==0){
128.            if(p1)
129.                p1=(!portb.f0);
130.            else
131.                p1=portb.f2;
132.            if(p2)
133.                p2=(!portb.f5);
134.            else
135.                p2=portb.f3;
136.            if(p3)
137.
   p3=((player34_inputs&0x08)==0);
138.            else
139.
   p3=((player34_inputs&0x20)!=0);
140.
141.            if(p4)
142.
   p4=((player34_inputs&0x01)==0);
143.            else
144.
   p4=((player34_inputs&0x04)!=0);
145.
146.            if(!((portb.f1) ||
   (portb.f4) ||
   ((player34_inputs&0x12)!=0)))
147.                count=0;
148.
149.
150.        }
151.        if(tmr0_i==1220){
152.            /*for(i=0;i<8;i++){
153.                dot_display[i]=0xff;
154.            }*/
155.            tmr0_i=0;
156.            //playerselecttimer();
157.            if(count==4){
158.                ball_xspeed=0;
159.                ball_yspeed=0;
160.                mode=1;
161.                count=0;
162.                tmr0_mark=1221;
163.                 if(p1){
164.                    serve=1;
165.
   ball_yposition=1;
166.                 }
167.                 else if(p2){
168.                    serve=2;
169.
   ball_yposition=6;
170.                 }
171.                 else if(p3){
172.                    serve=3;
173.
   ball_xposition=0x40;
174.                 }
175.                 else if(p4){
176.                    serve=4;
177.
   ball_xposition=0x02;
178.                 }
179.            }
180.            else
181.                count++;
182.
183.        }
184.    }
185.
186.    void playerselecttimer(){
187.        switch(count){
188.            case 4:
189.                dot_display[3]-=0x10;
190.            case 3:
191.                dot_display[4]-=0x10;
192.            case 2:
193.                dot_display[4]-=0x08;
194.            case 1:
195.                dot_display[3]-=0x08;
196.                break;
197.        }
198.
199.    }
200.
201.    void main() {
202.        trisa=0;
203.        trisd=0;
204.        trisc=0x80;
205.        trisb=0xff;
206.
207.        intcon = 0xA0;
208.        option_reg = 0xC3;
209.        porta.f1=0;
210.        porta.f1=1;
211.        uart1_init(9600);
212.        for(i=0; i<8; i++){
213.            dot_display[i]=0xff;
214.        }
215.        while(1){
216.            output_to_dot_display();
217.        }
218.    }
219.
220.    void updategame(){
221.        for(i=0; i<8; i++){
222.            dot_display[i] = 0xff;
223.        }
224.
   dot_display[ball_yposition]-=ball_xpos
   ition;
225.        drawPucks();
226.    }
227.
228.    void drawPucks(){
229.        dot_display[0]-=puck1*p1;
230.        dot_display[7]-=puck2*p2;
231.
232.        if(p3){
233.            dot_display[puck3-1] -=
```

```
    (dot_display[puck3-1]>0x80)*0x80;
234.            dot_display[puck3] -=
    0x80;
235.            dot_display[puck3+1] -=
    (dot_display[puck3+1]>0x80)*0x80;
236.        }
237.        if(p4){
238.            dot_display[puck4-1] -=
    ((dot_display[puck4-1]&0x01)!=0)*0x01;
239.            dot_display[puck4] -=
    0x01;
240.            dot_display[puck4+1] -=
    ((dot_display[puck4+1]&0x01)!=0)*0x01;
241.        }
242.   }
243.
244.   void getPlayerInputs(){
245.        player1_btns[0] = portb.f0;
246.        player1_btns[1] = portb.f1;
247.        player1_btns[2] = portb.f2;
248.        player2_btns[0] = portb.f3;
249.        player2_btns[1] = portb.f4;
250.        player2_btns[2] = portb.f5;
251.   }
252.
253.   void movePlayerPucks(){
254.        if(auto_mode){
255.            puck1 = ball_xposition +
    (2*ball_xposition) +
    (ball_xposition/2);
256.            puck2=puck1;
257.            puck3=ball_yposition;
258.            puck4=puck3;
259.        }
260.        else{
261.            if(p1){
262.                if(puck1>0x07)
263.
    puck1/=1+player1_btns[2];
264.                if(puck1<0xe0)
265.
    puck1*=1+player1_btns[0];
266.            }
267.            if(p2){
268.                if(puck2>0x07)
269.
    puck2/=1+player2_btns[2];
270.                if(puck2<0xe0)
271.
    puck2*=1+player2_btns[0];
272.            }
273.            if(p3){
274.                if(puck3<6)
275.
    puck3+=(player34_inputs&0x08)!=0;
276.                if(puck3>1)
277.
    puck3-=(player34_inputs&0x20)!=0;
278.            }
279.            if(p4){
280.                if(puck4<6)
281.
    puck4+=(player34_inputs&0x04)!=0;
282.                if(puck4>1)
283.
    puck4-=(player34_inputs&0x01)!=0;
284.            }
285.            if(serve!=0)
286.                puckupdate();
287.        }
288.   }
289.
290.   void puckupdate(){
291.        if(serve!=0){
292.             switch(serve){
293.                 case 1:
294.                     ball_xposition =
    puck1 & (puck1*2) & (puck1/2);
295.
    if(player1_btns[1])
296.
    ball_yspeed=1;
297.                     break;
298.                 case 2:
299.                     ball_xposition =
    puck2 & (puck2*2) & (puck2/2);
300.
    if(player2_btns[1])
301.
    ball_yspeed=-1;
302.                     break;
303.                 case 3:
304.                     ball_yposition =
    puck3;
305.
    if(player34_inputs & 0x10)
306.
    ball_xspeed=1;
307.                     break;
308.                 case 4:
309.                     ball_yposition =
    puck4;
310.
    if(player34_inputs & 0x02)
311.
    ball_xspeed=-1;
312.                     break;
313.             }
314.             if(ball_xspeed!=0 ||
    ball_yspeed!=0)
315.                 serve=0;
316.        }
317.        else{
318.              switch(ball_yposition){
319.                  case 0:
320.                      if(!p1){
321.
    ball_yspeed=1;
322.                      }
323.                      else{
324.                          serve=1;
325.
    ball_xspeed=0;
326.
    ball_yspeed=0;
327.
    ball_yposition=1;
328.                      }
329.                      break;
330.                  case 1:
331.                      if(p1){
332.
    if((puck1/ball_xposition==1)){
333.
```

```
   ball_yspeed=1;
334.
   if(ball_xspeed!=1)
335.
   ball_xspeed++;
336.                            }
337.                            else
   if((puck1/ball_xposition)==3){
338.
   ball_yspeed=1;
339.                            }
340.                            else
   if((puck1/ball_xposition)==7){
341.
   ball_yspeed=1;
342.
   if(ball_xspeed!=0xff)
343.
   ball_xspeed--;
344.                            }
345.                       }
346.                      break;
347.                 case 6:
348.                      if(p2){
349.
   if((puck2/ball_xposition==1)){
350.
   ball_yspeed=-1;
351.
   if(ball_xspeed!=1)
352.
   ball_xspeed++;
353.                            }
354.                            else
   if((puck2/ball_xposition)==3){
355.
   ball_yspeed=-1;
356.                            }
357.                            else
   if((puck2/ball_xposition)==7){
358.
   ball_yspeed=-1;
359.
   if(ball_xspeed!=0xff)
360.
   ball_xspeed--;
361.                            }
362.                       }
363.                       break;
364.                 case 7:
365.                       if(!p2){
366.
   ball_yspeed=-1;
367.                       }
368.                       else{
369.                            serve=2;
370.
   ball_xspeed=0;
371.
   ball_yspeed=0;
372.
   ball_yposition=6;
373.                       }
374.                       break;
375.
376.             }
377.
```

```
378.             switch(ball_xposition){
379.                  case 1:
380.                       if(!p4){
381.
   ball_xspeed=1;
382.                       }
383.                       else{
384.                            serve=4;
385.
   ball_xspeed=0;
386.
   ball_yspeed=0;
387.
   ball_xposition=0x02;
388.                       }
389.                       break;
390.                 case 2:
391.                     if(p4){
392.                       pucktest =
   puck4-ball_yposition;
393.
   if(pucktest==0){
394.
   ball_xspeed=1;
395.                            }
396.
   if(pucktest==1){
397.
   ball_xspeed=1;
398.
   if(ball_yspeed!=0xff)
399.
   ball_yspeed--;
400.                            }
401.
   if(pucktest==-1){
402.
   ball_xspeed=1;
403.
   if(ball_yspeed!=1)
404.
   ball_yspeed++;;
405.                            }
406.                       }
407.                      break;
408.                 case 0x40:
409.                       if(p3){
410.                        pucktest =
   puck3-ball_yposition;
411.
   if(pucktest==0){
412.
   ball_xspeed=-1;
413.                            }
414.
   if(pucktest==1){
415.
   ball_xspeed=-1;
416.
   if(ball_yspeed!=0xff)
417.
   ball_yspeed--;
418.                            }
419.
   if(pucktest==-1){
420.
   ball_xspeed=-1;
```

```
421.
   if(ball_yspeed!=1)
422.
   ball_yspeed++;;
423.                              }
424.                            }
425.                         break;
426.
427.                    case 0x80:
428.                         if(!p3){
429.
   ball_xspeed=-1;
430.                         }
431.                         else{
432.                             serve=3;
433.
   ball_xspeed=0;
434.
   ball_yspeed=0;
435.
   ball_xposition=0x40;
436.                         }
437.                         break;
438.
439.               }
440.
441.               ball_yposition +=
   ball_yspeed ;
442.
443.               if(ball_xspeed==1)
444.                   ball_xposition*=2;
445.               else
   if(ball_xspeed==0xff)
446.                   ball_xposition/=2;
447.         }
448.    }
449.
450.
451.
452.    void output_to_dot_display(){
453.         //portd=0xff;
454.         porta.f1=0;
```

```
455.         porta.f1=1;
456.         portd=dot_display[0];
457.         //delay_ms(100);
458.         for(i=1; i<8; i++){
459.             portd=0xff;
460.             porta.f0=0;
461.             porta.f0=1;
462.             portd=dot_display[i];
463.             //delay_ms(100);
464.         }
465.
466.    }
```

VI. Conclusion

By understanding the functions, program, and connections in the microprocessor we used, we were able to successfully run the simulation of the game through a software, and by making the hardware part of the project, we were able to play the actual game. The various components we used such as the IC, resistors, buttons and microprocessor, and their connections were important in order to have a successful output. There are also some things that we could add to our project to improve its features such as having a bigger display, and a pointing system.

VIII. Resources

A. Prototype Demonstration Video

LED Matrix Pong Game - https://youtu.be/203vcCY0gcY

B. C Code and Java Companion App

https://github.com/Paolodpie/4-Player-Pong---LED-Matrix