Mini Pong

A mini report made by Mandar Joshi and Robin Dahlkvist

Our idea is a mini version of pong which can be played on the PICkit 2 Low Pin Count demo board. The aim of the game is to click the button when the LED is lighted up on the left side (if you are playing single player or player 1 in multiplayer) or if you are player 2 then clicking the button when the LED is lighted up on the right side. Pressing the button at the right time will send the light back to the other player, and the aim is to not press the button when it's not your LED which is lighting up, as well as time the button press such that it sends the LED in the other direction.

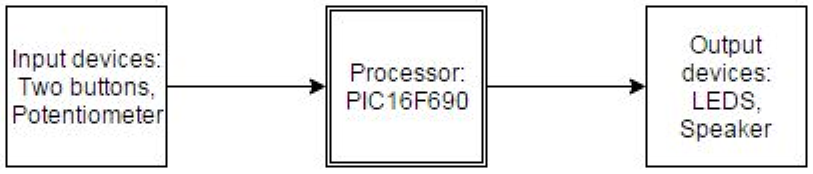
We have implemented a single player and a multiplayer mode where there are two buttons on the board, and there are 4 LEDS to play the game on. In both versions of the game you may also control the speed of the lights using the potentiometer.

We designed and implemented our product on the LPC board by using the components already built into it (1 button, 4 leds, 1 potentiometer), and we also added a button and a speaker to the breadboard.

We tested our product by experimenting and working our way forward with the code, every time we wanted to implement something new we always tested it out directly after compiling to see that everything worked and there were no new bugs.

We have also added a speaker for the game to make pong sounds, and these sounds play at both ends of the leds when the led is lighted up.

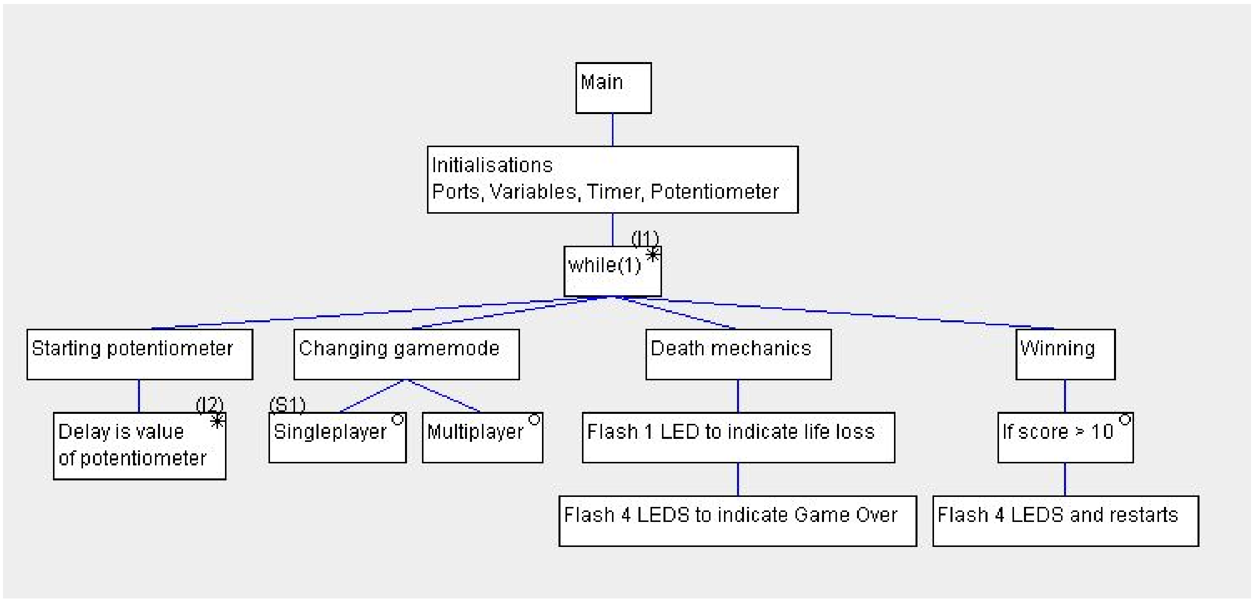
Block diagram of the product:


Input devices: Two buttons, Potentiometer
Processor: PIC16F690
Output devices: LEDS, Speaker

All our features implemented use this block diagram.

- Checklist of showing that it works:
  - Button, leds and potentiometer work with simple hello world programs
  - Implementing a rotating led, we had to implement our own shifter function instead of using the compiler instructions rr and rl
  - Configuring the potentiometer to change the frequency of how fast the leds rotate
  - Making the led “bounce” back from the other side and having to press the button (this was implemented in our shifter function)
  - Added a life and score feature for singleplayer
  - Added a multiplayer feature with another button on the breadboard, with support of switching between the two
  - Added number of lives remaining
  - Added sound to the game

Not too detailed JSP Chart of our main program:


Main
Initialisations
Ports, Variables, Timer, Potentiometer
(I1)
while(1) *
Starting potentiometer
Changing gamemode
Death mechanics
Winning
(I2)
Delay is value of potentiometer *
(S1)
Singleplayer o
Multiplayer o
Flash 1 LED to indicate life loss
If score > 10 o
Flash 4 LEDS to indicate Game Over
Flash 4 LEDS and restarts

Main Program Code:

```c
/*Mini-pong created by Mandar Joshi and Robin Dahlkvist*/

/* B Knudsen Cc5x C-compiler - not ANSI-C */
#include "16F690.h"
#pragma config |= 0x00D4
#include "lookup.c"
#define EIGHT_NOTE 250

//Variables
int LEDX; // visualisation for LEDS
char time;
int dirX; //Which direction the LEDS have
int score; //Score for reaching the other side in singleplayer
int lives; //lifes
int gamemode; //1 for single, 2 for multi
int ttd; //Time to death, a count down from when the leds reaches the player destination.
int alive; //If the player in single is alive or not
int blink; //Just a variable for blinking

//Functions
char LookUpNote(char); /* function prototype */
void notify(void); //Notfies when changed gamemode
void restart(void); //Resets the values
void delay(char); //regular delay
void delay10(char n); //Delay 10x
void sound (char); //Plays a note
int win(void); //Checking if win
void soundWin (void);
void flash(void); //Flashing LEDS when win
void init(void); //Initialization for some variables and registers
void shifter(void); //Shifting the leds

void main(void) {

  //Initialization
  init();
  while (1) {
    //POTENTIOMETER
    GO = 1;
    while (GO);
    time = ADRESH;
    time = ~time;
    //SOUND
    if (ttd == 0) {
      alive = 0;
      if (gamemode == 1) {
        lives--;
      }
      ttd = -1;
```

```
}
//Changing gamemode
if (PORTA.3 == 0 && PORTB.6 == 0) {

  if (gamemode == 1) {
    gamemode = 2;
    notify();
    restart();
  } else if (gamemode == 2) {
    gamemode = 1;
    notify();
    restart();
  }
}
if (alive) {

  if (win()) {
           soundWin();
    flash();
    restart();
  } else {
    shifter();
  }
} else if ((!alive && PORTA.3 == 1) || (!alive && PORTA.6 == 1)) {

  if (lives == 0) {
    //Indicates death
    if (blink == 1) {
      PORTC = 0b1111;
      blink = 0;
      delay(time);
    } else {
      PORTC = 0b0000;
      blink = 1;
      delay(time);
    }
  } else {
    //Indicates death
    if (blink == 1) {
      if (dirX == -1) {//Single player
                  if(gamemode == 1){
                         switch(lives){ //Displaying remaning lives.
                                case 1:
                                       PORTC = 0b0001;
                                       break;
                                case 2:
                                       PORTC = 0b0011;
                                       break;
                                case 3:
                                       PORTC = 0b0111;
```

```
                                    break;
                                case 4:
                                    PORTC = 0b1111;
                                    break;
                            }
                        }
                        else{
                        PORTC = 0b0001;
                        }
                } else if (dirX == 1) {//Multiplayer
                  PORTC = 0b1000;
                }

                blink = 0;
                delay(time);
              } else {
                PORTC = 0b0000;
                blink = 1;
                delay(time);
              }
            }
          } else if ((!alive && PORTA.3 == 0 && dirX == -1) || (!alive && PORTB.6 == 0 && dirX ==
1)) {
                    PORTC = 0b0001;       //Starting position;
                    alive = 1;     //Puts back to life
                    ttd = 255;     //Timer so the user dont die
                    delay(200); //Just a delay
          }
  }
}
int win(void) {
  if (score == 15) {
    return 1;
  } else {
    return 0;
  }
}
void init(void) {
  //LEDS
  TRISC = 0;

  //POTM
  TRISA.0 = 1;
  ANSEL.0 = 1;
  ADCON1 = 0b0.001.0000;
  ADCON0 = 0b0.0.0000.0.1;

  //2nd button
  TRISB.6 = 1;
  WPUB.6 = 1;
  OPTION_REG.7 = 0;


  lives = 4; //Lives for single player
  LEDX = 0b0010;      //Start postition
  dirX = -1;
  score = 0; //Score 15 to win
  gamemode = 1; //Singleplayer
  blink = 0; //Random var
  alive = 1;   //Boolean for alive or not
  ttd = 255;   //Timer till death

}
void shifter(void) {

    switch (gamemode) {
      //SINGLEPLAYER
    case 1:

      //Left side click
      if (LEDX == 0b0001 && dirX == -1 && PORTA.3 == 0) {
        dirX = 1;
        score++;
        ttd = 2;
               sound(50);
      }
      //Waiting for player to press the button
      else if (LEDX == 0b0001 && dirX == -1 && PORTA.3 == 1) {
        ttd--; //Timer till death
        delay(time);

      }
      if (LEDX == 0b0001 && dirX == 1) {
        LEDX = 0b0010;
        PORTC = LEDX;
        delay(time);

      }

      if (LEDX == 0b1000 && dirX == 1) {
        LEDX = 0b0100;
        PORTC = LEDX;
        dirX = -1;
               sound(250);
        delay(time);


      }

      //LEDS moving
      if (LEDX == 0b0010 && dirX == 1) {
        LEDX = 0b0100;
        PORTC = LEDX;
        delay(time);
      }
      if (LEDX == 0b0010 && dirX == -1) {
        LEDX = 0b0001;
        PORTC = LEDX;
        delay(time);
      }

      if (LEDX == 0b0100 && dirX == 1) {
        LEDX = 0b1000;
        PORTC = LEDX;
        delay(time);
      }
      if (LEDX == 0b0100 && dirX == -1) {
        LEDX = 0b0010;
        PORTC = LEDX;
        delay(time);
      }

      break;
      /*********************************************/

      //MULTIPLAYER
    case 2:

      //Left side click
      if (LEDX == 0b0001 && dirX == -1 && PORTA.3 == 0) {
        dirX = 1;
        ttd = 2;
        //score++;
              sound(50);
      }
      //Waiting for player to press the button
      else if (LEDX == 0b0001 && dirX == -1 && PORTA.3 == 1) {
        ttd--; //Timer till death
        delay(time);
      }
      if (LEDX == 0b0001 && dirX == 1) {
        LEDX = 0b0010;
        PORTC = LEDX;
        delay(time);
      }

      //Right side click
      if (LEDX == 0b1000 && dirX == 1 && PORTB.6 == 0) {
        dirX = -1;
        ttd = 2;
```

```
                sound(250);
        }
        //Waiting for player to press the button
        else if (LEDX == 0b1000 && dirX == 1 && PORTB.6 == 1) {
          ttd--; //Timer till death
          delay(time);
        }
        if (LEDX == 0b1000 && dirX == -1) {
          LEDX = 0b0100;
          PORTC = LEDX;
          delay(time);
        }

        if (LEDX == 0b0010 && dirX == 1) {
          LEDX = 0b0100;
          PORTC = LEDX;
          delay(time);
        }
        if (LEDX == 0b0010 && dirX == -1) {
          LEDX = 0b0001;
          PORTC = LEDX;
          delay(time);
        }

        if (LEDX == 0b0100 && dirX == 1) {
          LEDX = 0b1000;
          PORTC = LEDX;
          delay(time);
        }
        if (LEDX == 0b0100 && dirX == -1) {
          LEDX = 0b0010;
          PORTC = LEDX;
          delay(time);
        }

        break;

      }
    }
  //SOUND FUNCTIONS
  void sound (char note){
         int x;
         for(x = 0; x < 10; x++){
         bit out, button = 1;
                TRISA.4 =  0;
          char j;
          for(j = note; j > 0; j--)
              {
              nop(); nop(); nop(); nop();
              }
          out = !out;
          PORTA.4 = out;
    }
  }

  void soundWin (void){
         int x;
         int y;
         for(y = 0; y < 10; y++){
                for(x = 0; x < 10; x++){
                bit out, button = 1;
                        TRISA.4 =  0;
                        char j;
                        for(j = 152; j > 0; j--)
                               {
                               nop(); nop(); nop(); nop();
                               }
                        out = !out;
                        PORTA.4 = out;
    }
    delay(50);
    }
  }
  //Resetting the values.
  void restart(void) {
    score = 0;
    lives = 4;
    alive = 1;
    ttd = 255;

  }

  void flash(void) {
    int count = 0;
    while (count < 10) {
      PORTC = 0b0011;
      delay(time);
      PORTC = 0b1100;
      delay(time);
      count++;
    }
    PORTC = 0b0000;

  }
  //BORROWED DELAY FROM LABS
  void delay10(char n)
    /*
      Delays a multiple of 10 milliseconds using the TMR0 timer
      Clock : 4 MHz   => period T = 0.25 microseconds
      1 IS = 1 Instruction Cycle = 1 microsecond
      error: 0.16 percent
    */
    {
      char i;

      OPTION = 7;
      do {
        i = TMR0 + 39; /* 256 microsec * 39 = 10 ms */
        while (i != TMR0)
        ;
      } while (--n > 0);
    }

  void notify(void) {
    int count = 0;
    while (count < 10) {
      PORTC = 0b1001;
      delay(time);
      PORTC = 0b0000;
      delay(time);
      count++;
    }
    PORTC = 0b0001;

  }
  //BORROWED DELAY FROM LABS
  void delay(char millisec) {
    OPTION = 2; /* prescaler divide by 8 */
    do {
      TMR0 = 0;
      while (TMR0 < 125);
    } while (--millisec > 0);
  }

  /*    Low pin count demo board                J1
           ___________  ___________            1 RA5
          |           \/           |           2 RA4
    +5V---|Vdd      16F690      Vss|---GND     3 RA3
       ---|RA5        RA0/AN0/(PGD)|-<-RP1     4 RC5
    SP1---|RA4            RA1/(PGC)|---        5 RC4
    SW1---|RA3/!MCLR/(Vpp)  RA2/INT|---        6 RC3
       ---|RC5/CCP              RC0|->-DS1     7 RA0
       ---|RC4                  RC1|->-DS2     8 RA1
    DS4-<-|RC3                  RC2|->-DS3     9 RA2
          |RC6                  RB4|          10 RC0
          |RC7               RB5/Rx|          11 RC1
          |RB7/Tx               RB6|->-SW2    12 RC2
          |________________________|          13 +5V
                                              14 GND

  */
```

Fritzing Test Code:

```
/* B Knudsen Cc5x C-compiler - not ANSI-C */
#include "16F690.h"
#pragma config |= 0x00D4
void main( void) {
        TRISC = 0;
        TRISA.3 = 1; //Don't actually need this for PORTA.3, it is always an input
        while(1) {

                        if (PORTA.3 == 0) {
                                PORTC = 0b1111;
                        }
                        else {
                                PORTC = 0;
                        }

                }
}
```