Tecnológico de Monterrey

Reporte Solución del reto

Diseño con lógica programable

Guillermo Ramos Domínguez A01736484

Arick Morelos del Campo A01735692

IRS

16/03/23

Introducción

Dentro de la materia Lógica programable y durante las 5 semanas que duro el curso se trabajó en dar solución a un reto especifico, el cual consistía en el desarrollo de un sistema ciber-físico. Este consiste en la combinación de software y hardware que incluye dispositivos electrónicos configurables que se pueden conectar con el mundo real a través de sensores y actuadores. En este caso el sistema a desarrollar fue un mini videojuego para dos personas, implementando el uso de una tarjeta ARTY y un acelerómetro como parte de las maquinas que el juego iba a poseer. Esto con el fin de mostrar a los estudiantes el funcionamiento de este tipo de sistemas, así como de sus conexiones.

Desarrollo

En el caso de nuestro equipo nuestro juego consiste en una versión mejorada del clásico juego pong donde en esta ocasión hay 3 jugadores uno que controla cada barra y uno que ahora gracias a la tarjeta ARTY y el acelerómetro puede controlar la dirección que va a tomar la pelota dentro de la partida.

Si bien no era una tarea muy complicada, si era algo bastante complicado de hacer. Por los mismo en nuestro equipo tuvimos que dividir nuestro codigo en diferentes secciones. Estas secciones son el top level, el modelo uart, codigo ensamblador y el codidgo de processing.

Codigo del top level:

Esta parte es la que se encarga del funcionamiento directo con la tarjeta ARTY, encargándose de establecer la comunicación entre el programa del juego y la tarjeta. Esto con el fin de hacer que el acelerómetro por medio de este codigo y de vivado. Mande los datos que va recibiendo el acelerómetro a la terminal serial. Para que de este modo la información que reciba sirva como datos para establecer el movimiento que tendra la pelota.

```
library IEEE;
library work;
use IEEE.STD_LOGIC_1164.ALL;
entity Top_level is
--generic(
--                                          n : integer := 8
                        --ancho del registro
--            );
    Port    (
        clk : in std_logic;
        rst : in std_logic;
        leds: out std_logic_vector(3 downto 0);
        MISO : in STD_LOGIC;
        SCLK : buffer STD_LOGIC;
        SS_N    :    buffer        STD_LOGIC_VECTOR(0
DOWNTO 0);
        MOSI : out STD_LOGIC;
        TX : out STD_LOGIC;
        RX : in STD_LOGIC
        );
end Top_level;
architecture Behavioral of Top_level is
component modulo_uart is
    Port (
                CLK : in STD_LOGIC;
                RST : in STD_LOGIC;
                --pines de comunicaciÃ³n con PicoBlaze
              PORT_ID  :  in  STD_LOGIC_VECTOR  (7
downto 0);
             INPUT_PORT : in STD_LOGIC_VECTOR (7
downto 0);
            OUTPUT_PORT : out STD_LOGIC_VECTOR
(7 downto 0);
           WRITE_STROBE : in STD_LOGIC;
                --pines de comunicaciÃ³n serial
                TX : out STD_LOGIC;
                RX : in STD_LOGIC
                );
end Component;
            signal  registro    :    STD_LOGIC_VECTOR(7
downto 0);
            signal          registro_temp          :
STD_LOGIC_VECTOR(7 downto 0);
            signal in_port_q :  std_logic_vector (7 downto
0);
            signal  write :  std_logic;
            signal puerto : std_logic_vector (7 downto 0);
            signal led_select :  std_logic_vector (7 downto
0);
begin
        uart :  modulo_uart
              port map (
                        CLK => clk,
                        RST => rst,
                     PORT_ID => puerto,
                   INPUT_PORT => led_select,
                  OUTPUT_PORT => registro_temp,
                 WRITE_STROBE => write,
                         TX => TX,
```

```
                RX => RX
        );
    bloque_1 : entity work.modulo_spi port map(
                                clk=>clk,
                                rst =>rst,
                                port_id => puerto,
                                OUTPUT_PORT
=>registro,
                                MISO =>MISO,
                                SCLK => SCLK,
                                SS_N =>SS_N,
                                MOSI => MOSI   );
                    bloque_2    :    entity
work.embedded_kcpsm6 port map(
clk=>clk,
                                rst=>rst,
in_port =>in_port_q ,
                                out_port    =>
led_select,
                                port_id => puerto,
                                write_strobe
=>write);
                    bloque_3    :    entity
work.puerto_salida_leds port map(
clk=>clk,
                                rst=>rst ,
                                SALIDA =>leds,
                                write_strobe
=>write,
                                PORT_ID    =>
puerto ,
                                INPUT_PORT
=>led_select );
    bloque_4 : entity work.registro_puerto_entrada
port map(
clk=>clk,
                                rst=>rst ,
                                D =>registro,
                                Q => in_port_q  );
end Behavioral;
```

Código modelo UART:

Esta parte se encarga de establecer la conexión con la terminal serial para que así. Cuando el programa se active hacer que los datos se envíen a la terminal y esta los proyecte con el programa que contiene el juego.

```
----------------------------------------------------------------------------------
-- LEE_TX_LISTO          en X"11";
-- ESCRIBE_DATO_TX       en X"12";
-- LEE_DATO_RX           en X"13"
-- LEE_DATO_LISTO_RX     en X"14"
-- ESCRIBE_DATO_RX_LEIDO en X"15"
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity modulo_uart is
    Port (
                CLK : in STD_LOGIC;
                RST : in STD_LOGIC;
                --pines de comunicaciÃ³n con PicoBlaze
                PORT_ID : in STD_LOGIC_VECTOR (7 downto 0);
                INPUT_PORT : in STD_LOGIC_VECTOR (7 downto 0);
                OUTPUT_PORT : out STD_LOGIC_VECTOR (7 downto 0);
                WRITE_STROBE : in STD_LOGIC;
                --pines de comunicaciÃ³n serial
                TX : out STD_LOGIC;
                RX : in STD_LOGIC
                );
end modulo_uart;

architecture Behavioral of modulo_uart is
--se declaran los componentes internos
component baud_rate_clock is
    Port (
                CLK : in  STD_LOGIC;
            EN_16_X_BAUD : out  STD_LOGIC
            );
end component;

component uart_tx6 is
  Port (            data_in : in std_logic_vector(7 downto 0);
                en_16_x_baud : in std_logic;
                  serial_out : out std_logic;
                buffer_write : in std_logic;
          buffer_data_present : out std_logic;
```

```
        buffer_half_full : out std_logic;
            buffer_full : out std_logic;
            buffer_reset : in std_logic;
                clk : in std_logic);
 end component;

component uart_rx6 is
 Port (          serial_in : in std_logic;
            en_16_x_baud : in std_logic;
               data_out : out std_logic_vector(7 downto
0);
            buffer_read : in std_logic;
       buffer_data_present : out std_logic;
         buffer_half_full : out std_logic;
            buffer_full : out std_logic;
            buffer_reset : in std_logic;
                clk : in std_logic);
 end component;

--declaraciÃ³n de seÃ±ales para uart_tx
signal tx_buffer_write, tx_buffer_full : std_logic;
signal tx_data_in : std_logic_vector(7 downto 0);
type fsm_tx_edos is (S1, S2);
signal fsm_tx_sig, fsm_tx_pres : fsm_tx_edos;
signal tx_busy : std_logic;
signal dato_tx_sig, dato_tx_pres : std_logic_vector(7
downto 0);
constant LEE_TX_LISTO   : std_logic_vector(7 downto
0) := X"11";
constant ESCRIBE_DATO_TX : std_logic_vector(7
downto 0) := X"12";

--declaraciÃ³n de seÃ±ales para uart_rx
signal rx_data_out : std_logic_vector(7 downto 0);
signal rx_buffer_read, rx_buffer_data_present :
std_logic;
signal rx_buffer_read_pres, rx_buffer_read_sig :
std_logic;
constant LEE_DATO_RX          : std_logic_vector(7
downto 0) := X"13";
constant LEE_DATO_LISTO_RX          :
std_logic_vector(7 downto 0) := X"14";
constant ESCRIBE_DATO_RX_LEIDO          :
std_logic_vector(7 downto 0) := X"15";

--declaraciÃ³n de seÃ±ales del baud rate generator
signal s_en_16_x_baud : std_logic;

--declaraciÃ³n de seÃ±ales temporales
signal temporal_tx, temporal_rx : std_logic_vector(7
downto 0);

begin
      baud_rate_generator : baud_rate_clock
                  port map(
                        CLK => CLK,
                     EN_16_X_BAUD          =>
s_en_16_x_baud
```

```
                );

        uart_tx :   uart_tx6
                port map(
                                data_in => tx_data_in,
                            en_16_x_baud                =>
s_en_16_x_baud,
                             serial_out => TX,
                            buffer_write => tx_buffer_write,
                        buffer_data_present => open,
                         buffer_half_full => open,
                             buffer_full => tx_buffer_full,
                            buffer_reset => '0',
                                    clk => CLK
                );

        uart_rx :   uart_rx6
                port map(
                                serial_in => RX,
                            en_16_x_baud                =>
s_en_16_x_baud,
                                data_out => rx_data_out,
                             buffer_read                =>
rx_buffer_read,
                            buffer_data_present         =>
rx_buffer_data_present,
                            buffer_half_full => open,
                                buffer_full => open,
                            buffer_reset => '0',
                                    clk => CLK
                );

        --lectura del estado del buffer de TX y del estado de la mÃ¡quina para TX
        --lectura si hay dato RX nuevo, lectura del dato RX e informe que nuevo dato RX ha sido leÃ-do
        temporal_tx <= "0000000" & (tx_buffer_full or tx_busy);
        temporal_rx <= "0000000" & (rx_buffer_data_present);
        OUTPUT_PORT <= temporal_tx when (PORT_ID = LEE_TX_LISTO) else
                        temporal_rx when (PORT_ID = LEE_DATO_LISTO_RX) else
                        rx_data_out when (PORT_ID = LEE_DATO_RX) else
                        (others => '0');

        --escritura del dato al buffer de tx
        dato_tx_sig <= INPUT_PORT when (PORT_ID = ESCRIBE_DATO_TX and WRITE_STROBE = '1') else
                        dato_tx_pres;

        --fsm para escribir un dato a tx, tx estÃ¡ siempre habilitado
        tx_data_in <= dato_tx_pres;
        maquina_tx : process(PORT_ID, WRITE_STROBE, fsm_tx_pres)
                begin
                        --valores por omisiÃ³n
```

```
                tx_busy <= '0';
                fsm_tx_sig <= S1;
                tx_buffer_write <= '0';

                case (fsm_tx_pres) is
                    when S1 => --en espera de que se escriba el dato TX
                        if(PORT_ID = ESCRIBE_DATO_TX and WRITE_STROBE = '1') then
                            fsm_tx_sig <= S2;
                            tx_busy <= '1';
                        end if;
                    when S2 => --dato listo, aplica escritura, tx aÃºn ocupado
                        tx_busy <= '1';
                        tx_buffer_write <= '1';

                end case;
            end process maquina_tx;

    --informa que dato RX actual ya ha sido leÃ-do
    rx_buffer_read_sig <= '1' when (PORT_ID = ESCRIBE_DATO_RX_LEIDO and WRITE_STROBE = '1') else
                          '0';
    rx_buffer_read <= rx_buffer_read_pres;


    --proceso secuencial
    secuencial : process(CLK, RST)
        begin
            if(RST = '1') then
                fsm_tx_pres <= S1;
                dato_tx_pres <= (others => '0');
                rx_buffer_read_pres <= '0';
            elsif(CLK'event and CLK = '1') then
                fsm_tx_pres <= fsm_tx_sig;
                dato_tx_pres <= dato_tx_sig;
                rx_buffer_read_pres <= rx_buffer_read_sig;
            end if;
        end process secuencial;


end Behavioral;
```

Codigo ensamblador:

Esta parte se hace cargo del funcionamiento del acelerómetro estableciendo los pines de salida y de entrada de los datos. Tanto los que este envía como que va recibiendo o detectando conforme vamos moviendo la placa una vez lo conectamos a la tarjeta.

```
;aplicaci¨®n que usa el USART en modo loopback
                ;memoria de programa de 1k
                ;
                ;puertos del spi
                CONSTANT PuertoLeeListoTX,    11
                CONSTANT  PuertoEscribeDatoTX,
12
                CONSTANT       PuertoLeeDatoRX,
13
                CONSTANT      PuertoDatoListoRX,
14
                CONSTANT     PuertoDatoRXLeido,
15
                CONSTANT PuertoLeeXMSB, 42
                CONSTANT PuertoLeeYMSB, 44
                CONSTANT PuertoLeeZMSB, 46
                CONSTANT PuertoSalida,  02
                ;
                NAMEREG s5, DatoSerial
                NAMEREG s6, EstadoTX
                NAMEREG s7, EstadoRX
                NAMEREG         s8, LeidoRX
                ;
                NAMEREG sC, DatoAccelX
                NAMEREG sB, DatoPruebaX
                ;
                NAMEREG sA, DatoAccelY
                NAMEREG s3, DatoPruebaY
                ;
                NAMEREG s4, leds
                ;inicio de programa
start:
                ;env¨ªa mensaje por tx
                LOAD            DatoSerial, "L"
                CALL            tx_uart
                LOAD            DatoSerial, "I"
                CALL            tx_uart
                LOAD            DatoSerial, "S"
                CALL            tx_uart
                LOAD            DatoSerial, "T"
                CALL            tx_uart
                LOAD            DatoSerial, "O"
                CALL            tx_uart
                ;

loop:
                ;lee cada uno de los componentes
del aceler¨®metro
```

```
                INPUT           DatoAccelX,
PuertoLeeXMSB
                LOAD            DatoPruebaX,
DatoAccelX
                ;
                ;
                INPUT           DatoAccelY,
PuertoLeeYMSB
                LOAD            DatoPruebaY,
DatoAccelY
                ;
                ;
                COMPARE
        DatoPruebaX, 00000000'b
                JUMP            Z, COMPX0
                COMPARE
        DatoPruebaX, 11111111'b
                JUMP            Z, COMPX1
COMPX0:
                COMPARE
        DatoPruebaY, 00000000'b
                JUMP            z, salida00
                COMPARE
        DatoPruebaY, 11111111'b
                JUMP            z, salida10
COMPX1:
                COMPARE
        DatoPruebaY, 00000000'b
                JUMP            z, salida01
                COMPARE
        DatoPruebaY, 11111111'b
                JUMP            z, salida11
                ;
                ;
salida00:
                LOAD            DatoSerial, "1"
                LOAD            leds, 00000001'b
                OUTPUT                  leds,
PuertoSalida
                CALL            tx_uart
                JUMP            loop
salida01:
                LOAD            DatoSerial, "2"
                LOAD            leds, 00000010'b
                OUTPUT                  leds,
PuertoSalida
                CALL            tx_uart
                JUMP            loop
salida10:
                LOAD            DatoSerial, "3"
                LOAD            leds, 00000100'b
                OUTPUT                  leds,
PuertoSalida
                CALL            tx_uart
                JUMP            loop
salida11:
                LOAD            DatoSerial, "4"
                LOAD            leds, 00001000'b
```

```
                OUTPUT          leds,
PuertoSalida
                CALL            tx_uart
                JUMP            loop
                ;
                ;

tx_uart:
                INPUT           EstadoTX,
PuertoLeeListoTX
                COMPARE
        EstadoTX, 01
                JUMP            Z, tx_uart
                OUTPUT
        DatoSerial, PuertoEscribeDatoTX
                RETURN
                ;
                ;
delay_1s:
                LOAD s2, BE
                LOAD s1, BC
                LOAD s0, 20
                RETURN
delay_loop:
                SUB             s0, 1'd
                SUBCY           s1, 0'd
                SUBCY           s2, 0'd
                JUMP            NZ, delay_loop
                RETURN
                ;
```

Código en processing:

Esta es la parte central ya que contiene todo el funcionamiento del juego. Encargándose de establecer cada parte de este desde las barras y su movimiento,

```
import processing.serial.*;
//seteamos variables
Serial port;  // Create object from Serial class
float x, y, speedX, speedY;
int pos1 = 500;
int pos2 = 500;
int puntos1 = 0;
int puntos2 = 0;
float diam =15;
float rectSize = 200;
boolean up = false, down=false,w= false, s=false;
PFont font;
int serial = 1;

// seteamos el ambiente
void setup() {
  fullScreen();
  fill(0, 255, 0);
  reset();
  port = new Serial(this, "COM15", 115200);
}

// resete de la bola y puntos
void reset() {
  x = width/2;
  y = height/2;
  speedX = random(3, 5);
  speedY = random(3, 5);
}

// deteccion de las teclas para jugador 1 y 2
void keyPressed() {
  switch(keyCode) {
    case UP : up = true; break;
    case DOWN : down = true; break;
  }
  switch(key){
    case 'w' : w = true; break;
    case 's' : s = true; break;
  }
}
void keyReleased() {
  switch(keyCode) {
    case DOWN : down = false; break;
    case UP : up = false; break;
  }
```

```
    switch(key){
    case 'w' : w = false; break;
    case 's' : s = false; break;
  }
}
void serialEvent(Serial p){
  while(port.available()==1){
    serial = port.readChar();
  }
}
//draw
void draw() {


  background(0);
  // bola
  ellipse(x, y, diam, diam);


  //puntos
  fill(color(255,255,255));
  font = loadFont("Arial-Black-48.vlw");
  textFont(font,50);
  text(puntos2, 300, 250);
  text(puntos1, 1600, 250);


  //controles de movimiento j1
  if (up == true){
    if (pos1 <= 100){
      pos1=100;
    }
    else{
    pos1 = pos1 - 20;
    }
    //text(pos1, 1850, 50);
  }

if (down == true){
  if (pos1 >= 1000){
    pos1=1000;
  }
  else{
  pos1 = pos1 + 20;
  }
  //text(pos1, 1850, 50);
}


// controloes de movimiento j2
  if (w == true){
  if (pos2 <= 100){
    pos2=100;
  }
  else{
  pos2 = pos2 - 20;
  }
  //text(pos2, 30, 50);
}
```

```
if (s == true){
  if (pos2 >= 1000){
    pos2=1000;
  }
  else{
  pos2 = pos2 + 20;
  }
  //text(pos2, 30, 50);
}

 //velocidades con serial
if(serial == '1' || serial == '2'){
  speedX = speedX + .1;
 }
 if(serial == '3' || serial == '4'){
  speedX = speedX - .1;
 }
 if(serial == '2' || serial == '4'){
  speedY = speedY - .1;
 }
 if(serial == '1' || serial == '3'){
  speedY = speedY + .1;;
 }

 //figura de los coliders
 fill(color(5,0,0));
 rect( 0, 0, 10, 1200);
 rect( 1910, 0, 10, 1200);

 //figura de los jugadores
 fill(color(255,255,255));
 rect(width-10, pos1-rectSize/2, 10, rectSize);
 rect(width-1890, pos2 - rectSize/2, 10, rectSize);

 x += speedX;
 y += speedY;

 // puntos coliders
 if ( x > 0 && x < 8 && y > 0 && y < 1200 ) {
  puntos1 = puntos1 + 1;
 }

 if ( x > 1900 && x < 1908 && y > 0 && y < 1200 ) {
  puntos2 = puntos2 + 1;
 }

 // coliders de rebote j1 y j2
 if ( x > width-30 && x < width -20 && y > pos1-
rectSize/2 && y < pos1+rectSize/2 ) {
  speedX = speedX * -1;
 }

 if ( x > width-1900 && x < width -1890 && y > pos2-
rectSize/2 && y < pos2+rectSize/2 ) {
  speedX = speedX * -1;
```

```
}
```

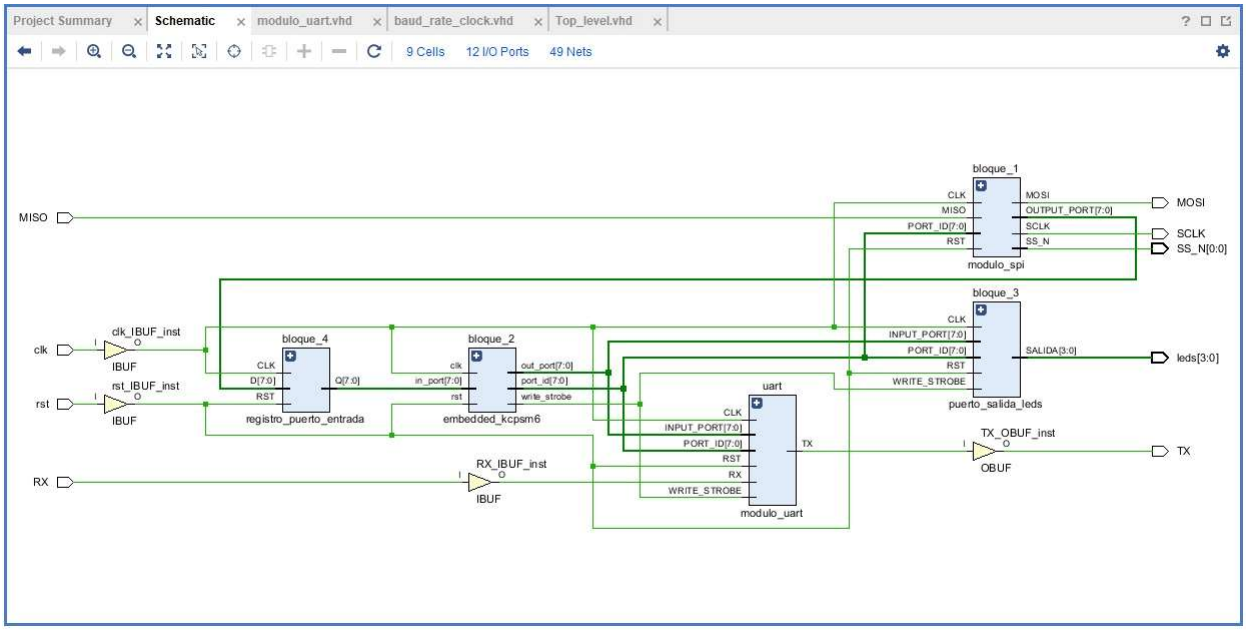
```
if ( y > height || y < 0 ) {
  speedY *= -1;
}
```

```
}

void mousePressed() {
  reset();
}
```

Resultados

Esquemático

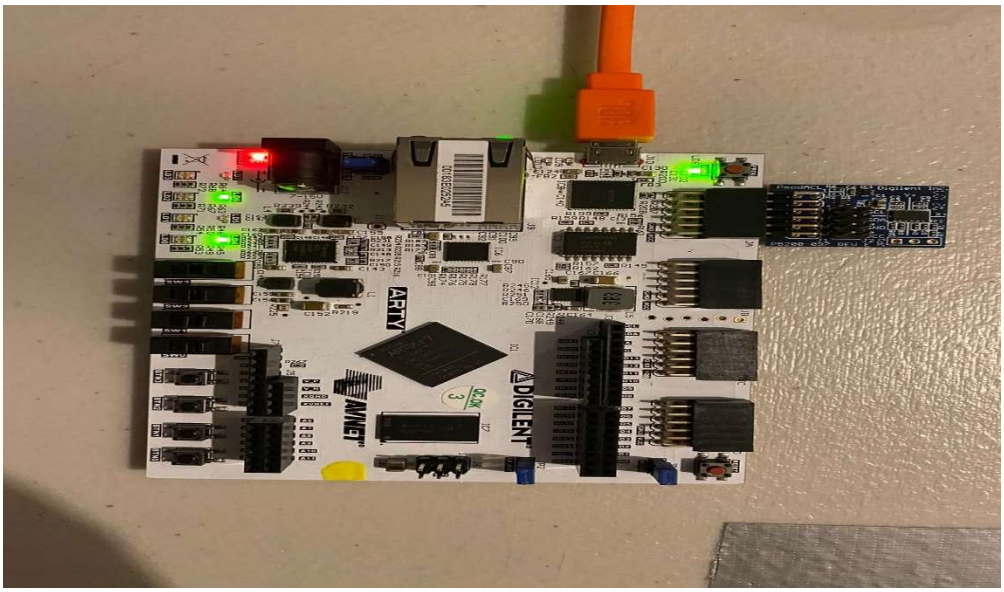


Este es una representación de las conexiones hechas en vivado con la tarjeta ARTY, de esta forma podemos ver que todas las conexiones, pines y el acelerómetro estan conectados y establecidos entre sí de manera correcta.

Juego

Esta es una vista de la pantalla del juego, con esto podemos ver que el programa funciona de manera correcta en todos sus elementos.

ARTY



Con esta vista podemos ver como se ve desde la tarjeta cuando se activa el programa, mostrando que la conexión es exitosa y el acelerómetro funciona de manera correcta.

Video del funcionamiento en completo del juego

https://drive.google.com/file/d/1tjRql3wFiacEz8Sa3-FhdICKC1SmTV2X/view?usp=sharing

Conclusiones

Realizar este proyecto fue una experiencia muy variada. Ya que habia cosas que no eran tan complejas como esperábamos en un inicio y otras que, aunque no eran complejas si eran tardadas de hacer. El mayor problema que tuvimos era que al ejecutar el programa en su totalidad funcionaba correctamente. Sin embargo, el puerto serial no recibía ningun dato del acelerómetro a pesar de que no habia errores de funcionamiento ni de conexión. A pesar de esto a base de prueba y error se pudo resolver y hacer que el juego funcionara como se tenia planeado. Volviéndose una experiencia interesante pero a la vez retadora, ya que traía una propuesta nueva e interesante que en un inicio puedo parecer no muy compleja pero conforme vamos avanzando se va volviendo un tanto más compleja de lo que parecía.