<!--Diseño con Lógica Programable-->

Reto VideoJuego{

<Por="Oscar_Ortiz_Torres"/>

<Por="Yonathan_Romero_Amador"/>

}

Reto(Equipo1);

// PicoBlaze

// VHDL

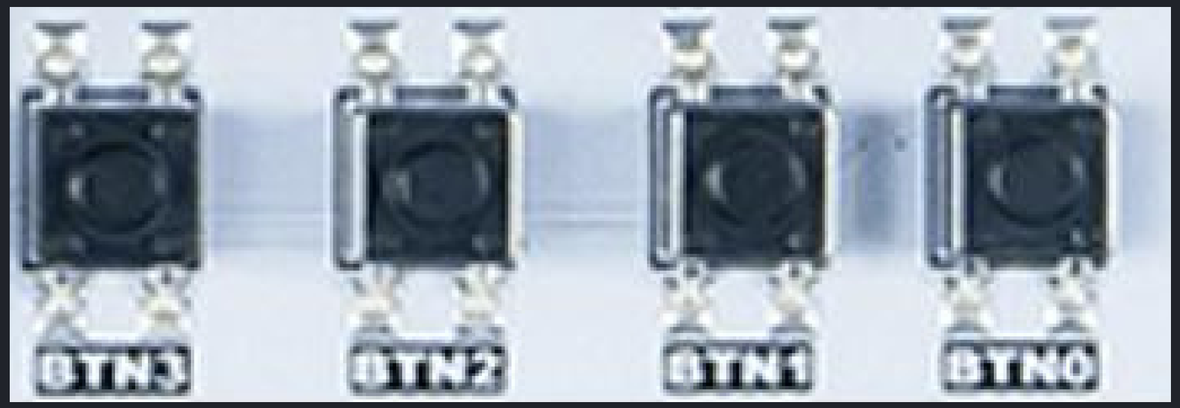
// Puerto Serial

// Processing

PicoBlaze

;Definicion de ;Perifericos

; 3 ; 2 ; 1 ; 0



;Se definieron 4 botones. Para ;controlar el juego

;El botón 3 se uso para moverse a la ;izquierda (“A”)

;El botón 2 se uso para saltar (“W”)

;El botón 1 se uso para disparar (“S”)

;El botón 0 se uso para moverse a la ;derecha (“D”)

;Lectura de ;Periferico

;Se crea un Puerto de Entrada (PuertoBT), el
;cual lee los botones. Debido a que se usan 8
;bits se usa una mascara de Bits (0F) para
;trabajar con los menos significativos.

;Se comparan estos 4 bits, cada comparación
;manda una letra en ASCII al puerto Serial, o
;manda un valor por defecto.

;Se tiene un retardo de 1/8 de segundo,
;aproximadamente.

```
1        CONSTANT PuertoBT, 00
2        CONSTANT PuertoLeeListoTX,    11
3        CONSTANT PuertoEscribeDatoTX, 12
4
5        NAMEREG s7, btn4
6        NAMEREG s5, DatoSerial
7        NAMEREG s6, EstadoTX
8
9        ADDRESS 000
10   loop:
11       INPUT btn4, PuertoBT
12       AND btn4, 0F
13
14       ;A   1 0 0 0
15       COMPARE btn4, 08
16       JUMP Z, salidaA
```

```
37   salidaA:
38       LOAD DatoSerial, "A"
39       CALL tx_uart
40       Jump loop
```

```
70   tx_uart:
71       INPUT EstadoTX, PuertoLeeListoTX
72       COMPARE EstadoTX, 01
73       JUMP Z, tx_uart
74       OUTPUT DatoSerial, PuertoEscribeDatoTX
75       CALL delay
76       RETURN
77
78   delay:
79       LOAD s2, 17
80       LOAD s1, D7
81       LOAD s0, 84
82
83   delay_loop:
84       SUB s0, 1'd
85       SUBCY s1, 0'd
86       SUBCY s2, 0'd
87       JUMP NZ, delay_loop
88       RETURN
```

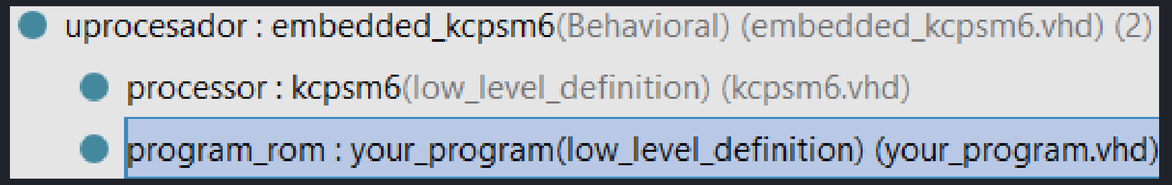
;Implementación ;en VHDL

;Para poder implementar el código de ;PicoBlaze en VDHL se uso el KCPSM6 ;Assembler.

;Con este archivo se remplazo el program_rom ;de la plantilla dada del uprocesador.

;Este componente es el encargado de todas las ;operaciones del codigo.

```
KCPSM6 Assembler v2.70
  Ken Chapman - Xilinx Ltd - 16th May 2014

Enter name of PSM file: your_program.psm

Reading top level PSM file...
  C:\Users\romer\Escritorio\mando_fpga\kcpsm6\your_program.psm

A total of 81 lines of PSM code have been read

Checking line labels
Checking CONSTANT directives
Checking STRING directives
Checking TABLE directives
Checking instructions

Writing formatted PSM file...
  C:\Users\romer\Escritorio\mando_fpga\kcpsm6\your_program.fmt

Expanding text strings
Expanding tables
Resolving addresses and Assembling Instructions
  Last occupied address: 036 hex
  Nominal program memory size: 1K (1024)    address(9:0)
  Occupied memory locations: 55
  Assembly completed successfully

Writing LOG file...
  C:\Users\romer\Escritorio\mando_fpga\kcpsm6\your_program.log
Writing HEX file...
  C:\Users\romer\Escritorio\mando_fpga\kcpsm6\your_program.hex
Writing VHDL file...
  C:\Users\romer\Escritorio\mando_fpga\kcpsm6\your_program.vhd
```



VHDL

-- Top_Entity

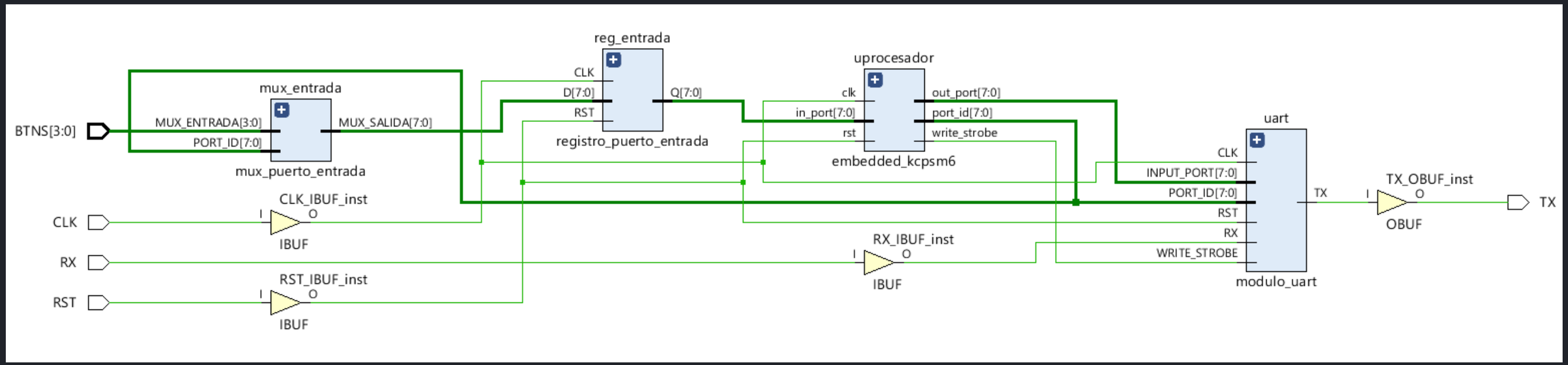
```
13  entity picoblaze_uart is
14      Port ( CLK : in STD_LOGIC;
15             BTNS : IN STD_LOGIC_VECTOR (3 downto 0);
16             RST : in STD_LOGIC;
17              TX : out STD_LOGIC;
18              RX : in STD_LOGIC);
19  end picoblaze_uart;
```

-- Componentes

```
82          mux_entrada : mux_puerto_entrada port map (
83                  MUX_ENTRADA => BTNS,
84                   MUX_SALIDA => mux_vector,
85                      PORT_ID => port_id_s
86              );
87
88          uart :  modulo_uart
89                  port map (
90                              CLK => CLK,
91                              RST => RST,
92                          PORT_ID => port_id_s,
93                       INPUT_PORT => out_port_s,
94                      OUTPUT_PORT => registro_entrada,
95                      WRITE_STROBE => write_strobe_s,
96                               TX => TX,
97                               RX => RX
```

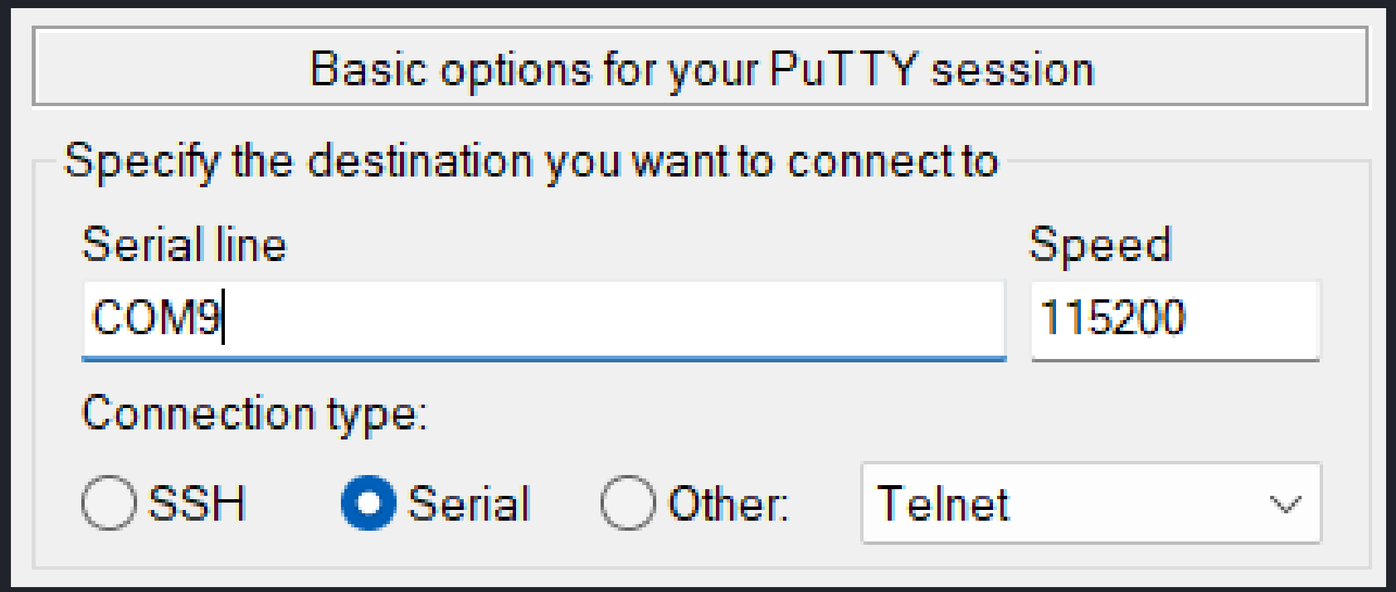
```
100         reg_entrada : registro_puerto_entrada
101                       port map (
102                                 CLK => CLK,
103                                 RST => RST,
104                                 D => mux_vector,
105                                 Q => in_port_s
106                                 );
107
108         uprocesador : embedded_kcpsm6
109                       port map(
110                                 in_port => in_port_s,
111                                out_port => out_port_s,
112                                 port_id => port_id_s,
113                            write_strobe => write_strobe_s,
114                          k_write_strobe => open,
115                             read_strobe => open,
116                               interrupt => interrupt_s,
117                           interrupt_ack => interrupt_ack_s,
118                                     clk => CLK,
119                                     rst => RST
120                       );
```

-- Diagrama RTL



Puerto Serial

Declaracion

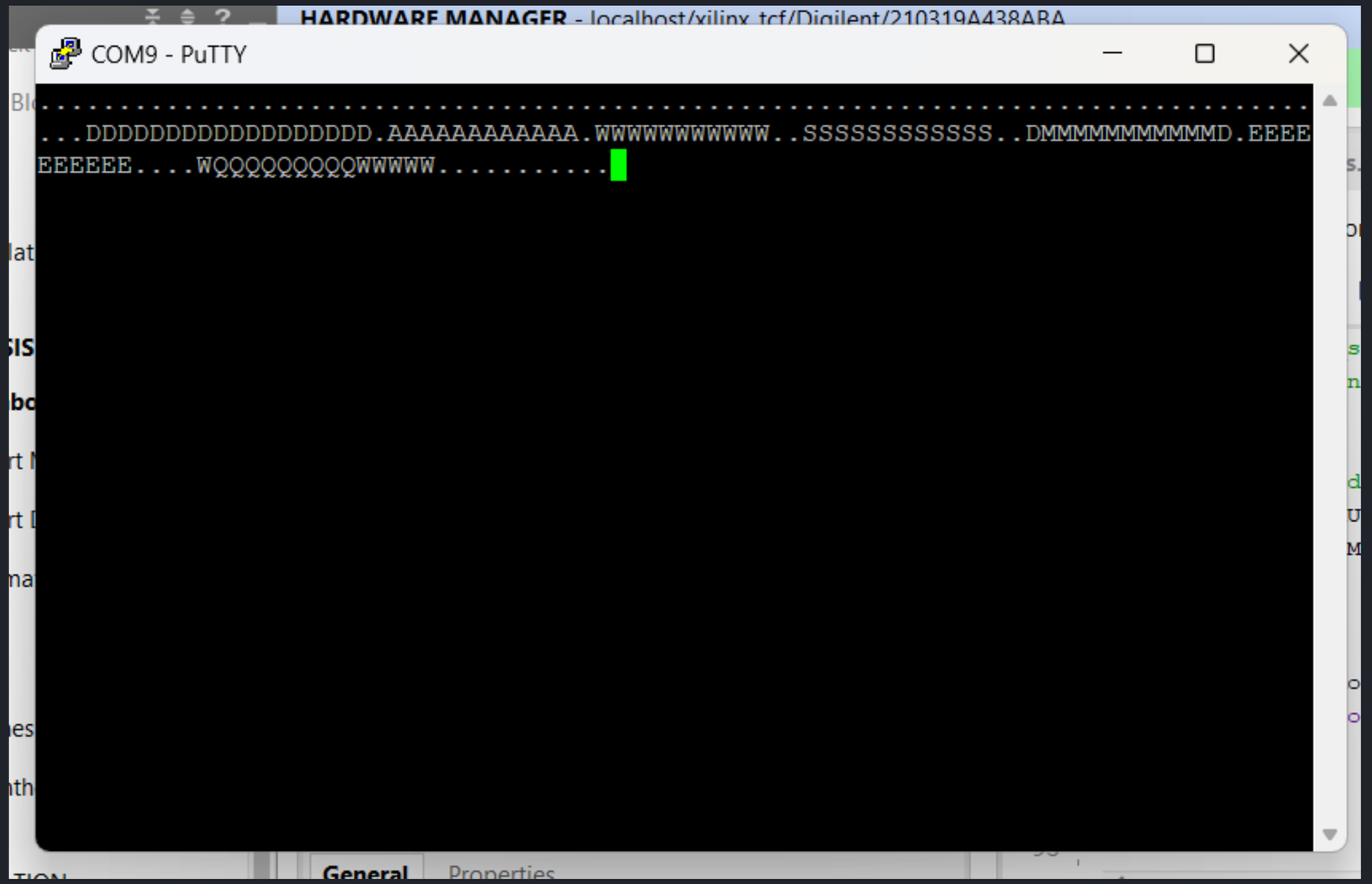


El puerto Serial que se utiliza va a una frecuencia de 115200.

El puerto dependerá de la computadora. En este caso es el COM9.

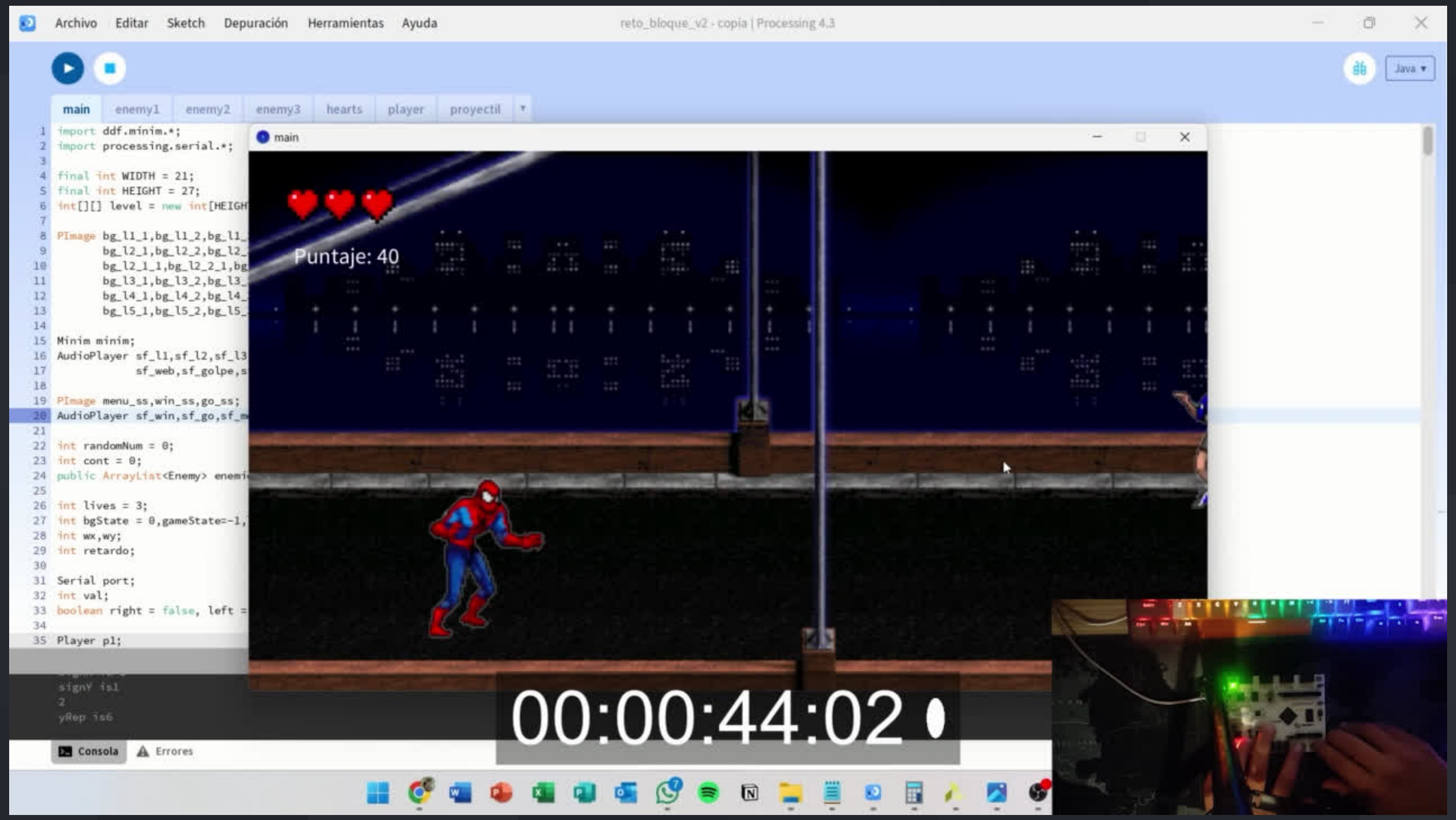
Se utilizo PuTTY para comprobar lo que mandaba el puerto serial.

Prueba en PUTTY



Proccesing

// Juego


Archivo Editar Sketch Depuración Herramientas Ayuda
reto_bloque_v2 - copia | Processing 4.3
Java
main enemy1 enemy2 enemy3 hearts player proyectil
main
Puntaje: 40
00:00:44:02
Consola
Errores

// Main

// Enemy1

// Enemy2

// Enemy3

// Hearts

// Player

// Proyectil

```
import ddf.minim.*;
import processing.serial.*;

final int WIDTH = 21;
final int HEIGHT = 27;
int[][] level = new int[HEIGHT][WIDTH];

PImage bg_l1_1,bg_l1_2,bg_l1_3,
       bg_l2_1,bg_l2_2,bg_l2_3,
       bg_l2_1_1,bg_l2_2_1,bg_l2_3_1,
       bg_l3_1,bg_l3_2,bg_l3_3,
       bg_l4_1,bg_l4_2,bg_l4_3,
       bg_l5_1,bg_l5_2,bg_l5_3;

Minim minim;
AudioPlayer sf_l1,sf_l2,sf_l3,sf_l4,sf_l5,
            sf_web,sf_golpe,sf_web_1;

PImage menu_ss,win_ss,go_ss;
AudioPlayer sf_win,sf_go,sf_menu;

int randomNum = 0;
int cont = 0;
public ArrayList<Enemy> enemies = new ArrayList<Enemy>();

int lives = 3;
int bgState = 0,gameState=-1,levelNum=1,score=0,enemyDef=0;;
int wx,wy;
int retardo;

Serial port;
int val;
boolean right = false, left = false, up = false, space = false,start=false;
```

// Conexión serial

```
 3 import processing.serial.*;
40 Serial port;
```

```
47   port = new Serial(this, "COM9", 115200);
```

```
 93   if (0 < port.available()){
 94     val = port.read();
 95   }
 96   switch(val) {
 97     case 68: right = true; break; //"D"
 98     case 65: left = true; break; //"A"
 99     case 87: up = true; break; //"W"
100     case 83: space = true;break; //"S"
101     case 69: right = true;up = true;break;//"E"
102     case 81: left = true;up = true;break;//"Q"
103     case 77: start = true;break;//"M"
104     default: right=false;left=false;up=false;space=false;
105   }
```

// Enemy

```
interface Enemy{
int getX();
int getY();
int getAncho();
int getAlto();
int getLives();
void HitFlag();
void setX(int nx);
void setY(int ny);
void setlives(int nl);
}
```

```
class Enemy1 implements Enemy{
  int x,y;
  float xSpeed,ySpeed;
  float accel,deccel;
  float maxXspd,maxYspd;
  float xSave,ySave;
  int xRep,yRep;
  float gravity;
  float state;
  float stateRate;
  float stateShift;
  int charXShift;
  int charYShift;
  int pace;
  float randomNumber;
  int lives;
  PImage sprite;
  boolean hitflag;
  int slow;
```

// Gameplay

YouTube

¡GRACIAS!