

Diseño con Lógica Programable

# Crop Crusade: Tractor Tales

Ximena Lizeth Trejo Lavín A01198557  
Angelica Nahomi Velazquez Gaytán A00838198  
Valeria Meneses Parra A01735686  
Alondra Caspeta Juárez A01571416



# Introducción

Una de las grandes innovaciones dentro de la industria del entretenimiento han sido los videojuegos. Estos han evolucionado rápidamente y se han convertido en una forma de entretenimiento popular y lucrativa en todo el mundo (Saravia Numa, 2024). Además de comunicar historias de ciencia ficción, se utilizan en aplicaciones más serias, como el entrenamiento de soldados en el ejército de los Estados Unidos (Xataka, 2017).

# Justificación de la Problemática

El usuario interactuará principalmente a través de la operación de un tractor, y se destacarán los beneficios derivados de una operación eficaz, como una mayor cosecha recolectada en un tiempo reducido. El propósito es que los trabajadores de John Deere comprendan la relación directa entre sus acciones al operar el tractor y los resultados positivos que pueden lograr en el campo.



# JOHN DEERE

01.  
Definición del  
Proyecto

03.  
Programación  
de Procesador

02.  
Programación  
Placa - Unity

04.  
RTL del  
prototipo final



# Etapa 1. Definición del proyecto

## Videojuego

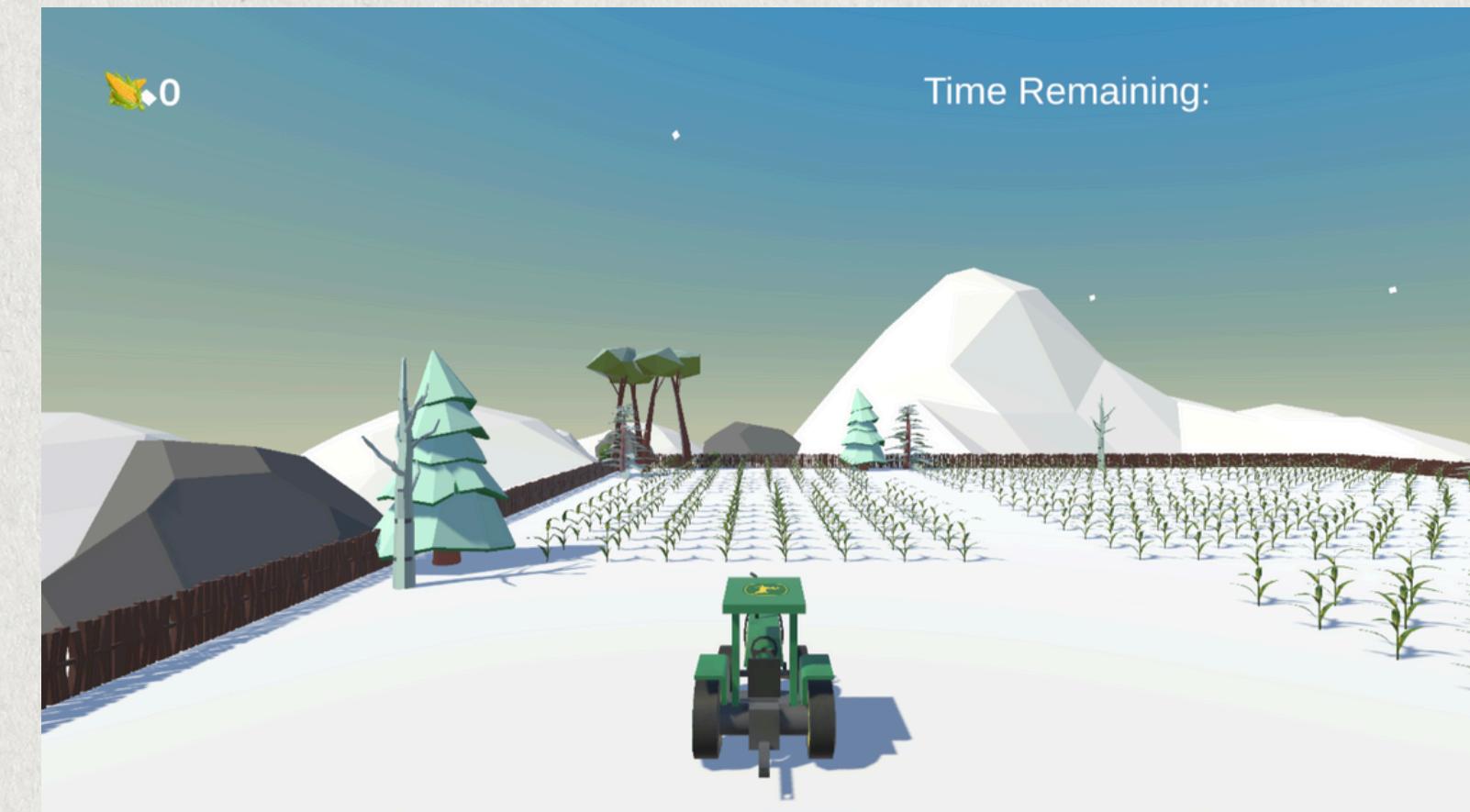
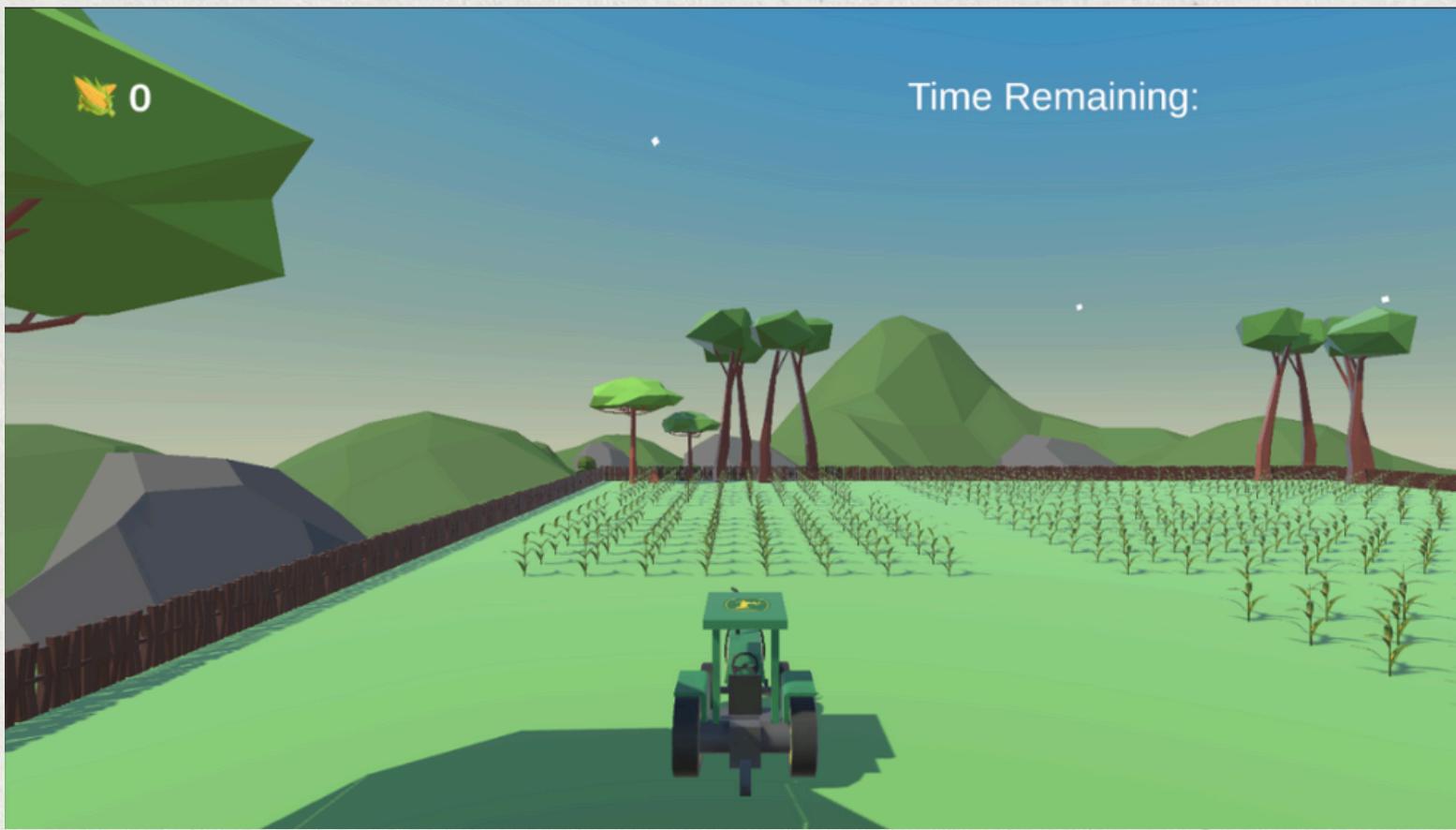
- Objetivo
- Mundos / Escenas
- Personajes
- Reglas del Juego
- Dinámica (Relación del juego con la placa)

## Máquina de Estados

- Representación de estados para la secuencia del videojuego
- Indicación de estado inicial
- Transiciones entre estados



# Videojuego

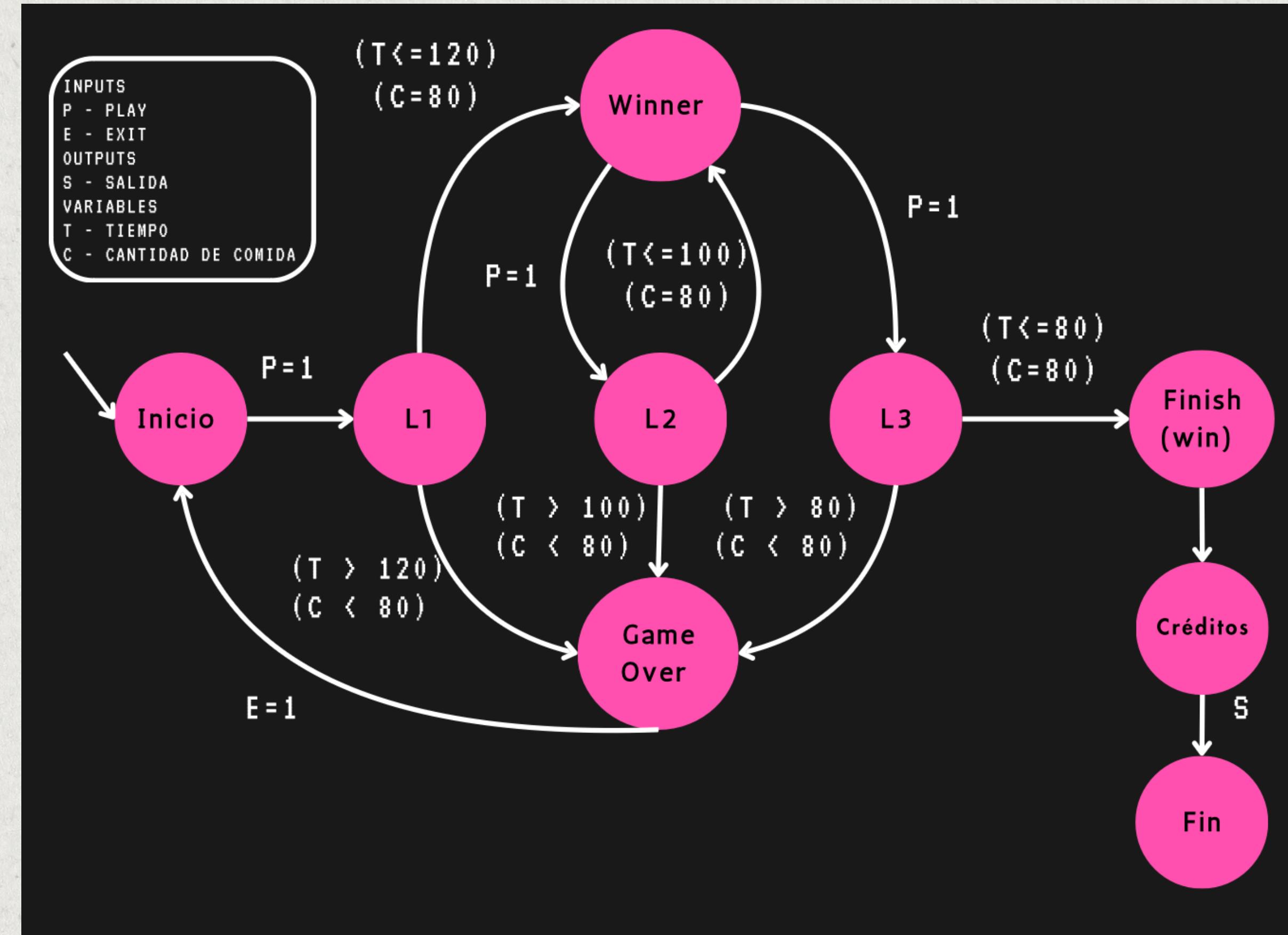


# Diseño en Unity

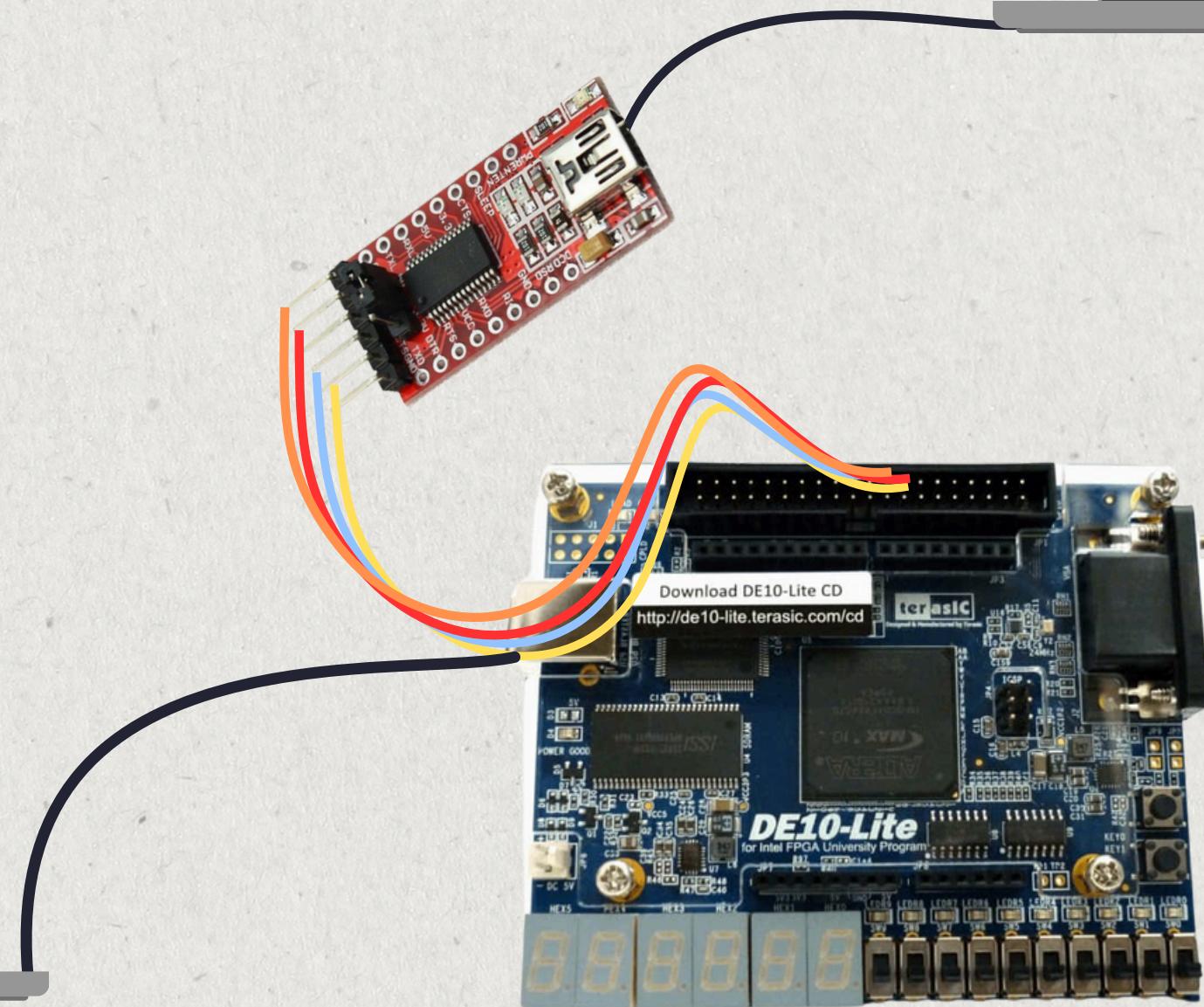
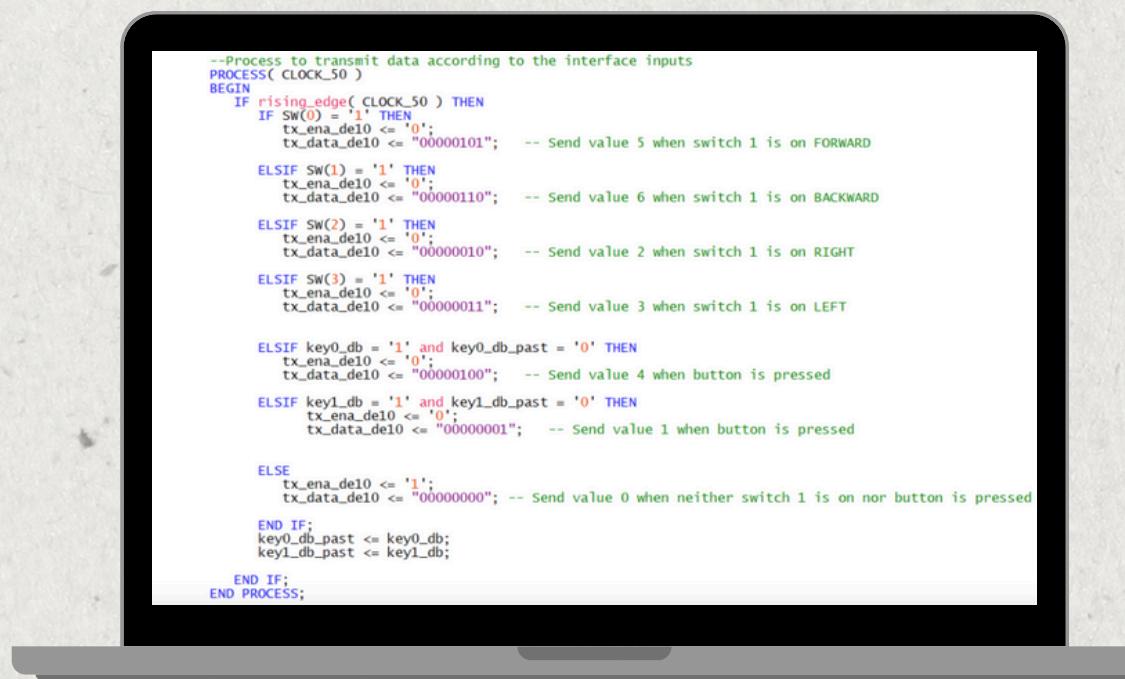


- Low Poly Fence Pack
- Cartoon Farm Crops
- Low Poly Woods
- Low Poly Mini Village
- Low Poly Christmas Add On
- Low Poly Trees
- Poly Style Vegetation
- Simple Nature Pack
- Simple Sky

# Máquina de Estados



# Etapa 2. Programación Placa - Unity



# Programación de Placa

```
--Process to transmit data according to the interface inputs
PROCESS( CLOCK_50 )
BEGIN
    IF rising_edge( CLOCK_50 ) THEN
        IF SW(0) = '1' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000101";    -- Send value 5 when switch 1 is on FORWARD
        ELSIF SW(1) = '1' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000110";    -- Send value 6 when switch 1 is on BACKWARD
        ELSIF SW(2) = '1' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000010";    -- Send value 2 when switch 1 is on RIGHT
        ELSIF SW(3) = '1' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000011";    -- Send value 3 when switch 1 is on LEFT
        ELSIF key0_db = '1' and key0_db_past = '0' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000100";    -- Send value 4 when button is pressed
        ELSIF key1_db = '1' and key1_db_past = '0' THEN
            tx_ena_de10 <= '0';
            tx_data_de10 <= "00000001";    -- Send value 1 when button is pressed
        ELSE
            tx_ena_de10 <= '1';
            tx_data_de10 <= "00000000"; -- Send value 0 when neither switch 1 is on nor button is pressed
        END IF;
        key0_db_past <= key0_db;
        key1_db_past <= key1_db;
    END IF;
END PROCESS;
```

```
uart_0      : uart      PORT MAP( CLOCK_50, SW(9), tx_ena_de10, tx_data_de10, GPIO_24, rx_busy_de10, rx_error_de10, rx_data_de10, tx_busy_de10, GPIO_25 );
button_0    : debounce   PORT MAP( CLOCK_50, KEY(0), key0_db );
button_1    : debounce   PORT MAP( CLOCK_50, KEY(1), key1_db );
display7seg : bcd7seg_sec PORT MAP( rx_data_de10(3 DOWNTO 0), HEX0 );
```

Fue indispensable el uso del FPGA (DE10-Lite Board de Altera). Este último se programa utilizando VHDL.

Debido a los requerimientos que expresó el socio formador, fue menester implementar los push buttons, los switches, y el display de 7 segmentos de la placa en relación con el juego.

# Programación en Unity (conexión serial)

Se necesitó de un espacio de nombres proporcionado por el framework .NET. Un espacio de nombres en programación se refiere a una forma de organizar y agrupar nombres de clases, interfaces, funciones y otros elementos dentro de un contexto específico.

Siendo en Unity System.IO.Ports el espacio de nombres empleado, esto brindó la comunicación con puertos seriales por medio de funciones integradas como write() y read().

| FTDI | FPGA    |
|------|---------|
| TX   | GPIO 27 |
| RX   | GPIO 28 |
| VCC  | 3.3 V   |
| GND  | GND     |

```
public SerialPort de10l = new SerialPort("COM4", 115200);
```

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using System;
6  using System.IO.Ports;
7  using TMPro;
8
9  public class de10 : MonoBehaviour
10 {
11     public int Recio = 0;
12     public int Break = 0;
13     public int Right = 0;
14     public int Left = 0;
15     public int Backward = 0;
16     public int Forward = 0;
17     public SerialPort de10l = new SerialPort("COM4", 115200);
18
19     private TextMeshProUGUI CornText;
20     public PlayerInventory playerInventory;
21
22
23     void Start()
24     {
25         CornText = GetComponent<TextMeshProUGUI>();
26         if (!de10l.IsOpen)
27         {
28             de10l.Open();
29             de10l.ReadTimeout = 1;
30         }
31
32         if (de10l.IsOpen)
33         {
34             var dataByte = new byte[] { 0x00 };
35             de10l.Write(dataByte, 0, 1);
36         }
37     }
}
```

```
39
40
41
42
43  void Update()
44  {
45      if (!de10l.IsOpen)
46      {
47          de10l.Open();
48          de10l.ReadTimeout = 1;
49      }
50
51      if (de10l.IsOpen)
52      {
53          try
54          {
55              int value;
56              if (de10l.BytesToRead > 0)
57              {
58                  value = de10l.ReadByte();
59                  de10l.DiscardInBuffer();
60              }
61              else
62              {
63                  value = 0;
64              }
65              if (value == 0x01)
66              {
67                  Recio = 1;
68              }
69              else if (value == 0x02)
70              {
71                  Right = 1;
72              }
73              else if (value == 0x03)
74              {
75                  Left = 1;
76              }
77          }
78      }
79  }
```

```
74     }
75     else if (value == 0x04)
76     {
77         Break = 1;
78     }
79     else if (value == 0x05)
80     {
81         Forward = 1;
82         Break = 0;
83     }
84     else if (value == 0x06)
85     {
86         Backward = 1;
87         Break = 0;
88     }
89     else
90     {
91         Recio = 0;
92         Right = 0;
93         Left = 0;
94         Forward = 0;
95         Backward = 0;
96     }
97     // Contador de maices
98     int MyInt = playerInventory.NumberOfCorns;
99     byte[] b = BitConverter.GetBytes(MyInt);
100    de10l.Write(b, 0, 1);
101
102    catch { }
103
104}
105
```

```
30     // Mensaje de Unity | 0 referencias
31     public void FixedUpdate()
32     {
33         // boost
34         if (de10l.Recio == 1)
35         {
36             Recio n = new();
37             if (frente) n.Boost(gameObject, fuerzaImpulso);
38             else if (!frente) n.Boost(gameObject, -fuerzaImpulso);
39         }
40
41         // Going forward
42         if (de10l.Forward == 1)
43         {
44             aceleracionActual = aceleracion * 10;
45             frente = true;
46         }
47         // Going backward
48         else if (de10l.Backward == 1)
49         {
50             aceleracionActual = -aceleracion * 400;
51             frente = false;
52         }
53         else
54         {
55             aceleracionActual = 0;
56         }
57     }
```

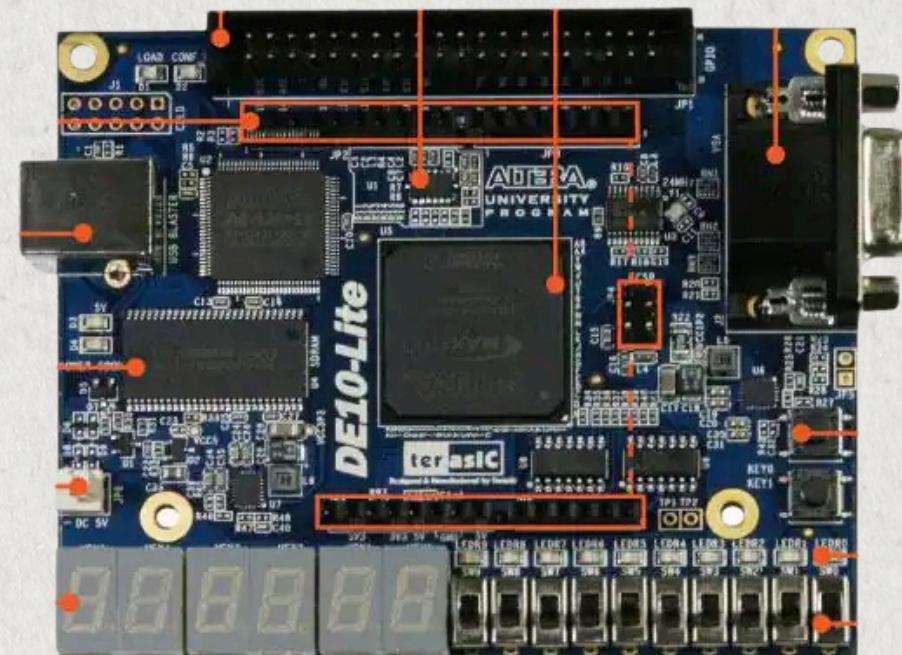
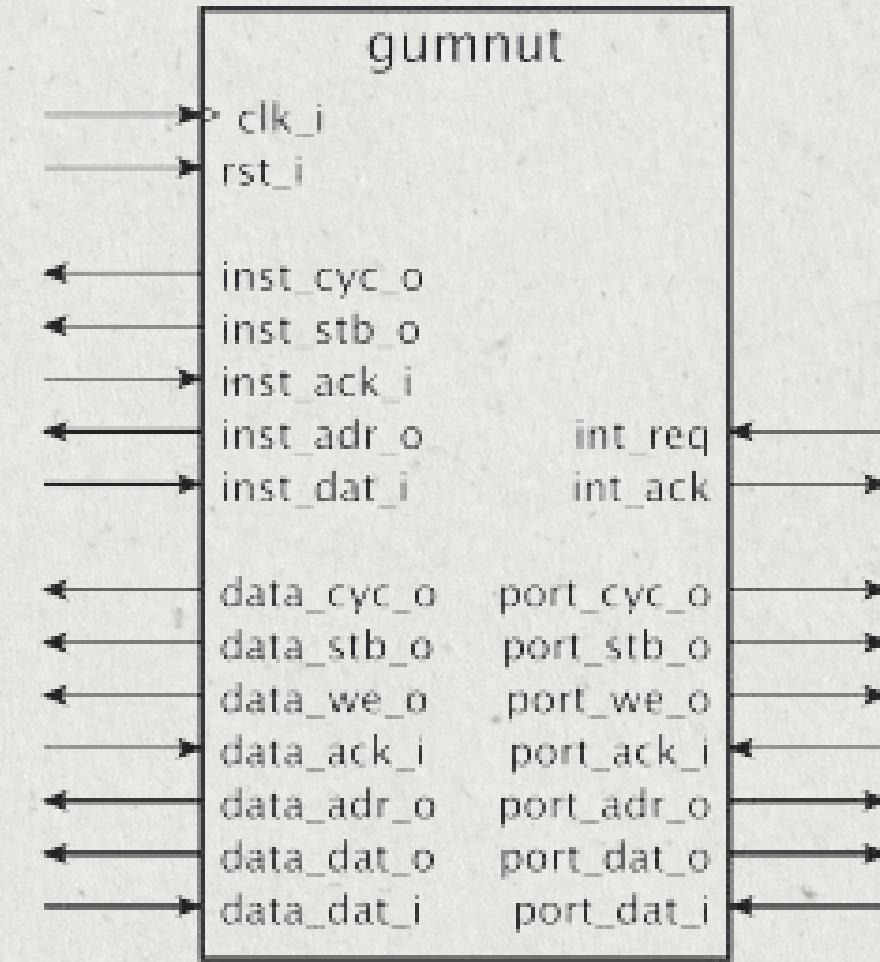
```
59     // Steering
60     if (de10l.Left == 1)
61     {
62         anguloActual = -anguloMaximo;
63     }
64     else if (de10l.Right == 1)
65     {
66         anguloActual = anguloMaximo;
67     }
68
69     // Break
70     if (de10l.Break == 1)
71         fuerzaFrenoActual = fuerzaFreno;
72     else
73         fuerzaFrenoActual = 0f;
74
75
76     frontRight.motorTorque = aceleracionActual;
77     frontLeft.motorTorque = aceleracionActual;
78
79     frontRight.brakeTorque = fuerzaFrenoActual;
80     frontLeft.brakeTorque = fuerzaFrenoActual;
81     backRight.brakeTorque = fuerzaFrenoActual;
82     backLeft.brakeTorque = fuerzaFrenoActual;
83
84     frontLeft.steerAngle = anguloActual;
85     frontRight.steerAngle = anguloActual;
86
87     UpdateWheel(frontLeft, frontLeftTransform);
88     UpdateWheel(frontRight, frontRightTransform); ;
```

```
92     2 referencias
93     void UpdateWheel(WheelCollider col, Transform trans)
94     {
95         Vector3 position;
96         Quaternion rotation;
97         col.GetWorldPose(out position, out rotation);
98
99         trans.position = position;
100        trans.rotation = rotation;
101    }
102
103    0 referencias
104    void UpdateSteeringWheel(Transform steering)
105    {
106        Vector3 customRotationAxis = transform.position.normalized;
107
108        Quaternion rot = Quaternion.AngleAxis(anguloActual, customRotationAxis.normalized);
109
110    }
111
```

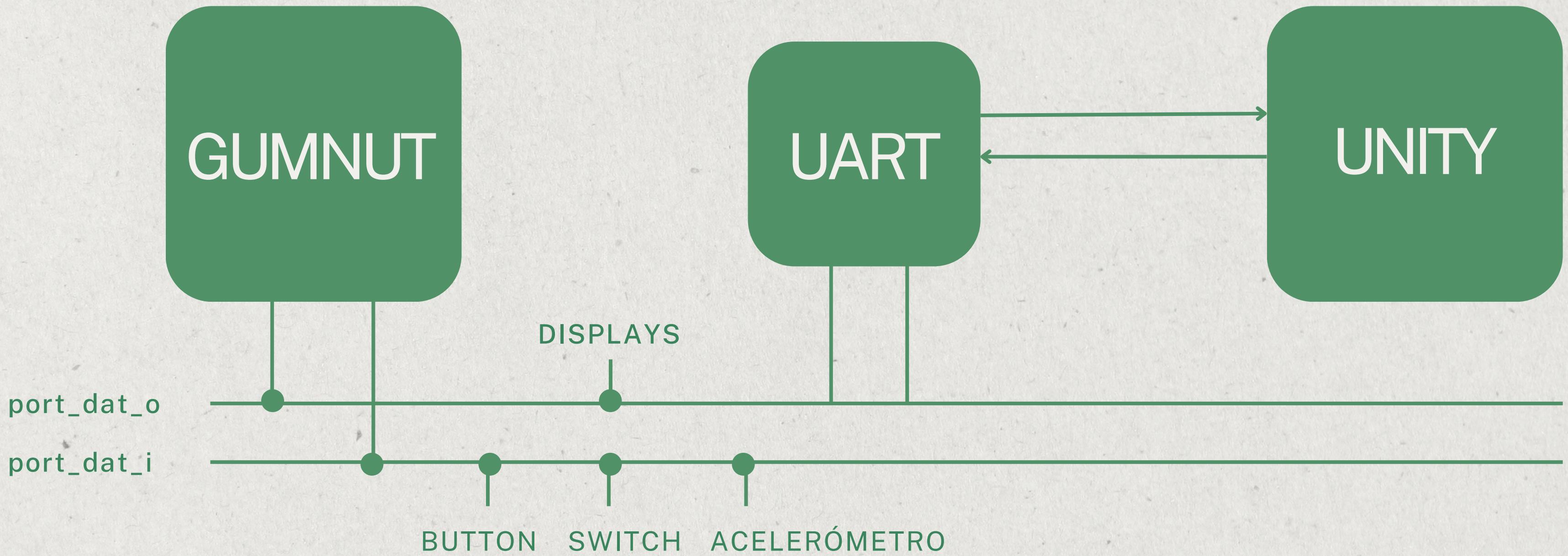
# Etapa 3.

## Programación de Procesador e integración de acelerómetro

- Se tiene un juego en Unity que utiliza botones, switches, displays de 7 segmentos y el acelerómetro de la placa para interactuar
- Se incluye la programación de un procesador, en donde todas las interfaces (botones, switches, puerto serial, displays, acelerómetro) interactúan con el procesador.
- Se incluye el software (código en ensamblador) para correr/ejecutar y darle funcionalidad al prototipo.



# Implementación Gumnut



```

BEGIN
    rst_i <= '0';

    gumnut      : gumnut_with_mem PORT MAP( CLOCK_50, rst_i, port_cyc_o, port_stb_o, port_we_o, '1', port_adr_o, port_dat_o, port_dat_i, int_req,
                                         int_ack);
    uart_0       : uart      PORT MAP( CLOCK_50, '1', tx_ena_de10, tx_data_de10, GPIO_24, rx_busy_de10, rx_error_de10, rx_data_de10, tx_busy_de10,
                                         GPIO_25 );
    button_0     : debounce   PORT MAP( CLOCK_50, KEY(0), key0_db );
    button_1     : debounce   PORT MAP( CLOCK_50, KEY(1), key1_db );
    acel_0 : acel      PORT MAP(CLOCK_50, SW(5 downto 4), GSENSOR_INT_I(1 downto 0), GSENSOR_SDI_I, GSENSOR_SDO_I, GSENSOR_CS_N_I, GSENSOR_SCLK_I,
                                 int_mov(9 downto 0));
    LEDR(9 downto 0) <= int_mov(9 downto 0);
    int_mov_modified <= int_mov(9 downto 6) & int_mov(3 downto 0);

-- Output => TX_DATA -> data memory address: 0x00
PROCESS( CLOCK_50, rst_i )
    BEGIN
        IF rst_i = '1' THEN
            tx_data_de10 <= ( OTHERS => '0' );
        ELSIF rising_edge( CLOCK_50 ) THEN
            IF port_adr_o = "00000000" and -- address port
               port_cyc_o = '1'           and -- control signals for I/O
               port_stb_o = '1'           and
               port_we_o = '1'           -- 'write' operation
            THEN
                tx_data_de10 <= port_dat_o;
            END IF;
        END IF;
    END PROCESS;

-- Output => TX_START -> data memory address: 0x01
PROCESS( CLOCK_50, rst_i )
    BEGIN
        IF rst_i = '1' THEN
            tx_ena_de10 <= '1';
        ELSIF rising_edge( CLOCK_50 ) THEN
            IF port_adr_o = "00000001" and -- address port
               port_cyc_o = '1'           and -- control signals for I/O
               port_stb_o = '1'           and
               port_we_o = '1'           -- 'write' operation
            THEN
                tx_ena_de10 <= port_dat_o(0);
            END IF;
        END IF;
    END PROCESS;

```

# VHDL

```

-- output => DISPLAY -> data memory address: 0x02
PROCESS( CLOCK_50, rst_i )
BEGIN
  IF rising_edge( CLOCK_50 ) THEN
    IF port_adr_o = "00000010" and -- address port
       port_cyc_o = '1' and -- control signals for I/O
       port_stb_o = '1' and
       port_we_o = '1' -- 'write' operation
    THEN
      num_int <= to_integer(unsigned(rx_data_de10));
      -- Convertir unidades, decenas y centenas
      decs <= std_logic_vector(to_unsigned(((num_int mod 100) / 10), 4));
      unids <= std_logic_vector(to_unsigned((num_int mod 10), 4));
      case unids is
        when "0000" =>
          HEX0 <= "11000000";
        when "0001" =>
          HEX0 <= "11111001";
        when "0010" =>
          HEX0 <= "10100100";
        when "0011" =>
          HEX0 <= "10110000";
        when "0100" =>
          HEX0 <= "10011001";
        when "0101" =>
          HEX0 <= "10010010";
        when "0110" =>
          HEX0 <= "10010010";
        when "0111" =>
          HEX0 <= "11111000";
        when "1000" =>
          HEX0 <= "10000000";
        when "1001" =>
          HEX0 <= "10011000";
        when "1010" =>
          HEX0 <= "10001000";
        when "1011" =>
          HEX0 <= "10000011";
        when "1100" =>
          HEX0 <= "10100111";
        when "1101" =>
          HEX0 <= "10100001";
        when "1110" =>
          HEX0 <= "10000110";
        when others =>
          HEX0 <= "10001110";
      end case;
    END IF;
  END PROCESS;

```

```

case decs is
  when "0000" =>
    HEX1 <= "11000000";
  when "0001" =>
    HEX1 <= "11111001";
  when "0010" =>
    HEX1 <= "10100100";
  when "0011" =>
    HEX1 <= "10110000";
  when "0100" =>
    HEX1 <= "10011001";
  when "0101" =>
    HEX1 <= "10010010";
  when "0110" =>
    HEX1 <= "10000010";
  when "0111" =>
    HEX1 <= "11111000";
  when "1000" =>
    HEX1 <= "10000000";
  when "1001" =>
    HEX1 <= "10011000";
  when "1010" =>
    HEX1 <= "10001000";
  when "1011" =>
    HEX1 <= "10000011";
  when "1100" =>
    HEX1 <= "10100111";
  when "1101" =>
    HEX1 <= "10100001";
  when "1110" =>
    HEX1 <= "10000110";
  when others =>
    HEX1 <= "10001110";
  end case;
END IF;
END IF;
END PROCESS;

```

```

port_dat_i <= "0000000" & key0_db WHEN (port_adr_o = "00000011") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
"0000000" & key1_db WHEN (port_adr_o = "00000100") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
"0000000" & SW(0) WHEN (port_adr_o = "00000101") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
"0000000" & SW(1) WHEN (port_adr_o = "00000110") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
"0000000" & SW_Int WHEN (port_adr_o = "00000111") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
int_mov_modified WHEN (port_adr_o = "00001000") and (port_cyc_o = '1') and (port_stb_o = '1') and (port_we_o = '0') ELSE
UNAFFECTED;

```

# Ensamblador

```

1  org 0x000 ; start here on reset
2          jmp main
3  ; Data memory layout
4          data
5 TX_DATA:    bss 1 ;0
6 TX_START:   bss 1 ;1
7 disp:       bss 1 ;2
8 KEY0:       bss 1 ;3
9 KEY1:       bss 1 ;4
10 SW0:        bss 1 ;5
11 SW1:        bss 1 ;6
12 SW_Int:     bss 1 ;7
13 int_mov:    bss 1 ;8
14 cont:       bss 1 ;9
15
16 ; Datos desde Unity
17 KEY0_data: byte 4
18 KEY1_data: byte 1
19 SW0_data:  byte 5
20 SW1_data:  byte 6
21 right_data: byte 2
22 left_data: byte 3
23 tx_disable: byte 1
24 SW_Aux:    byte 0
25

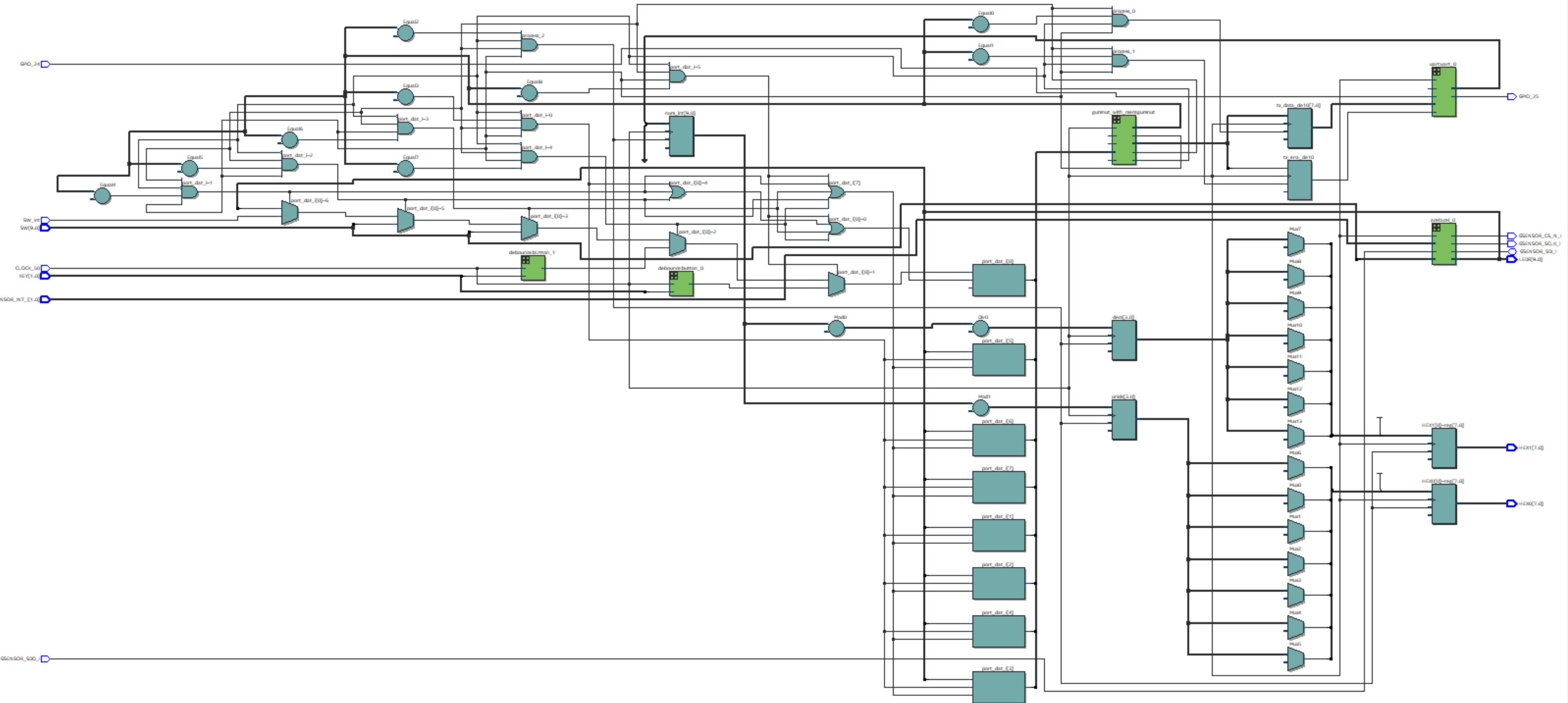
```

```

26 ; Main program
27         text
28         org 0x010
29
30 main:    jsb counter_func
31         inp r0, SW_Aux
32         and r4, r0, 0
33         bnz main
34         jsb next_SW0
35
36 counter_func: inp r6, cont
37             out r6, disp
38             ret
39
40 send_left: inp r1, SW_Int
41         inp r7, int_mov
42         and r1, r1, 1
43         bz main
44
45             and r2, r7, 0xC0
46             and r5, r7, 0x30
47             or r6, r2, r5
48             bz main
49             ldm r3, left_data
50             out r3, TX_DATA
51             out r0, TX_START
52             ldm r3, tx_disable
53             out r3, TX_START
54             jmp main
55
56 next_acel: inp r1, SW_Int
57         inp r7, int_mov
58         and r1, r1, 1
59         bz main
60         and r7, r7, 0xC0
61         and r3, r7, 0x03
62         or r4, r7, r3
63         bz send_left
64         ldm r2, right_data
65         out r2, TX_DATA
66         out r0, TX_START
67         ldm r2, tx_disable
68         out r2, TX_START
69
70 next_KEY0: inp r2, KEY0
71         and r2, r2, 1
72         bz next_KEY1
73         ldm r3, KEY0_data
74         out r3, TX_DATA
75         out r0, TX_START
76         ldm r3, tx_disable
77         out r3, TX_START
78
79 next_KEY1: inp r3, KEY1
80         and r3, r3, 1
81         bz next_acel
82         ldm r4, KEY1_data
83         out r4, TX_DATA
84         out r0, TX_START
85         ldm r4, tx_disable
86         out r4, TX_START
87
88 next_SW0: inp r4, SW0
89         and r4, r4, 1
90         bz next_SW1
91         ldm r5, SW0_data
92         out r5, TX_DATA
93         out r0, TX_START
94         ldm r5, tx_disable
95         out r5, TX_START
96         jmp main
97
98 next_SW1: inp r5, SW1
99         and r5, r5, 1
100        bz next_KEY0
101        ldm r6, SW1_data
102        out r6, TX_DATA
103        out r0, TX_START
104        ldm r6, tx_disable
105        out r6, TX_START

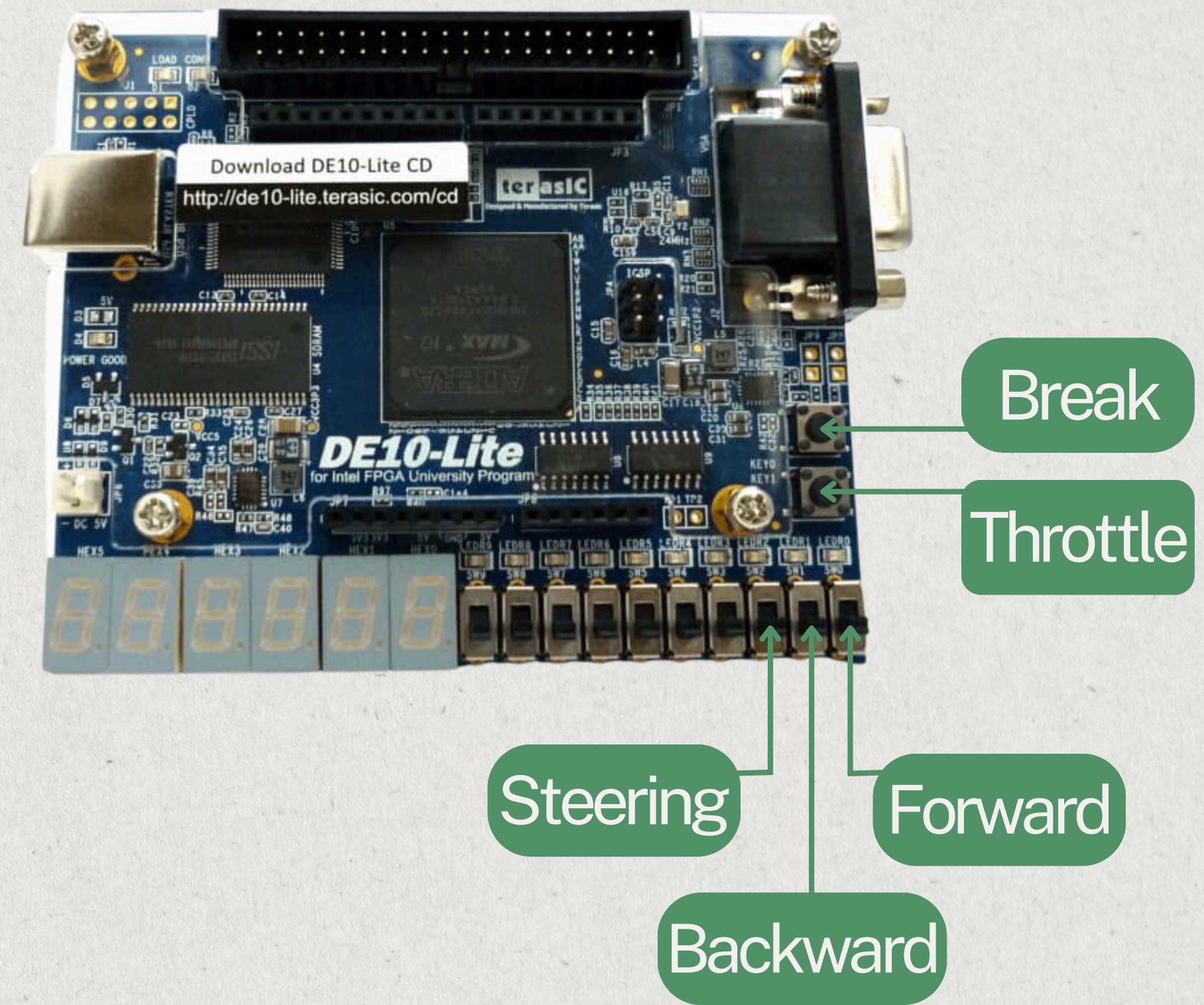
```

# RTL de Prototipo



# Manual de Instrucciones

- Para accionar cualquier funcionamiento del tractor, es necesario que ningún switch esté en alto al mismo tiempo.
- El switch para el steering del tractor únicamente es para direccionar las llantas delanteras del tractor a través del acelerómetro.
  - Ejemplo: Si al manejar usando el switch forward quiere girar a la derecha, debe bajar el switch para activar el steering. Posteriormente, inclinar la placa a la derecha para direccionar las llantas. Una vez direccionadas, puede bajar el switch y ahora subir el correspondiente a forward.



# Conclusiones

## Nahomi

Al finalizar este reto aprendimos sobre los sistemas ciber-físicos, donde el software y el hardware (en este caso el FPGA) integran lenguajes de alto y bajo nivel para lograr la programación del procesador. Esto fue enriquecedor, y una manera de ver todos los aprendizajes unirse. Además de reforzar el concepto de la comunicación serial para el recibimiento y la transmisión de datos.

## Alondra

Funcionamiento de Unity y programar la transmisión de datos desde VHDL. Implementación de ensamblador para hacer uso de Gumnut como procesador, ayuda a conocer más sobre la relación entre la parte de software y hardware en el proyecto.

## Ximena

La aplicación de un FPGA con Gumnut ayuda a poder replicar el trabajo, que se encuentra en el bajo nivel, en algún dispositivo que tenga los mismos periféricos que se usan, pudiendo compartir este proyecto en un futuro.

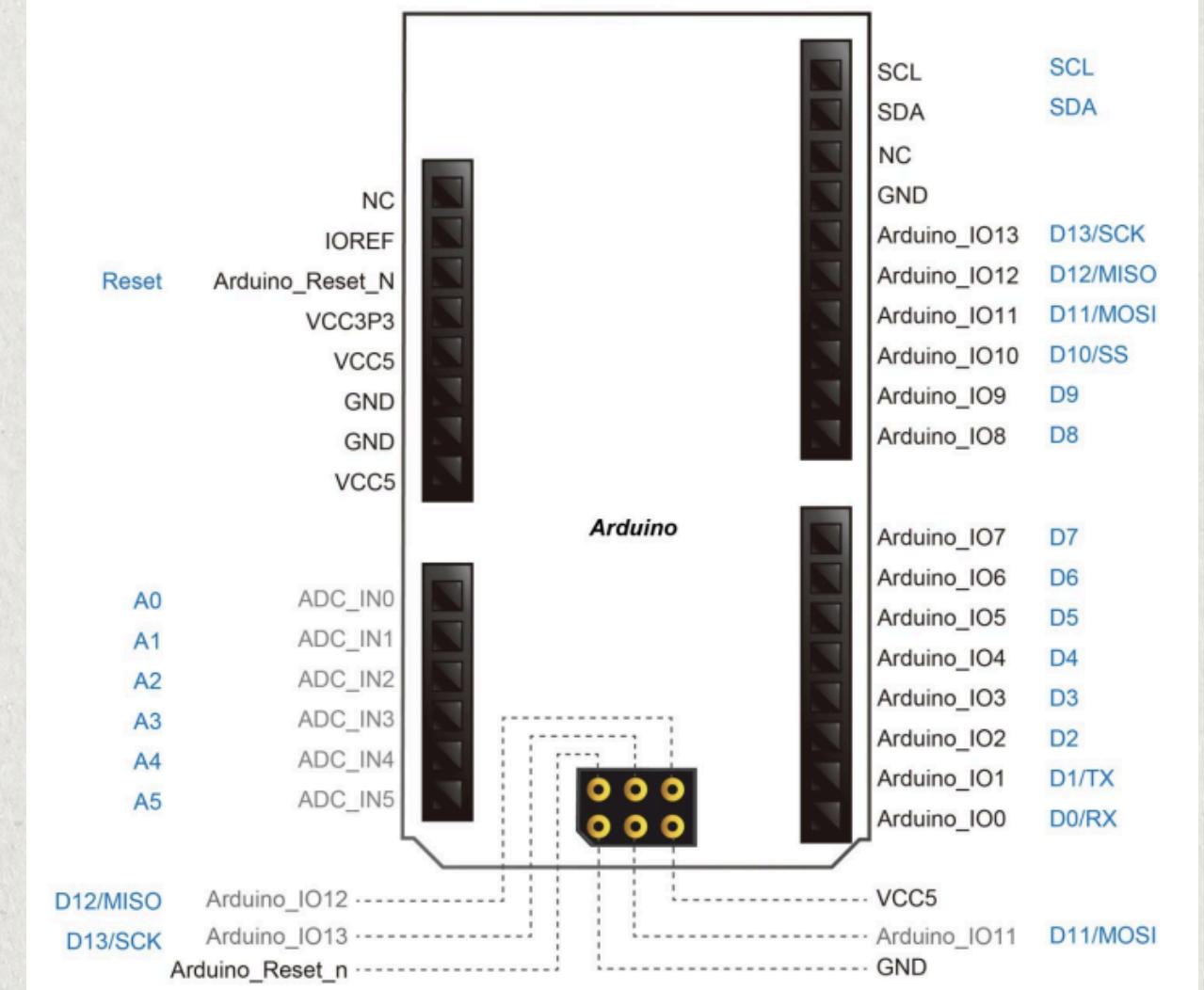
## Valeria

Durante este reto he aprendido nuevas herramientas y métodos para recibir y transmitir datos tanto con hardware y software, implementándolo en un videojuego y reflexionando en las ventajas que me da el uso de lenguaje en nivel alto y lenguaje en nivel bajo.

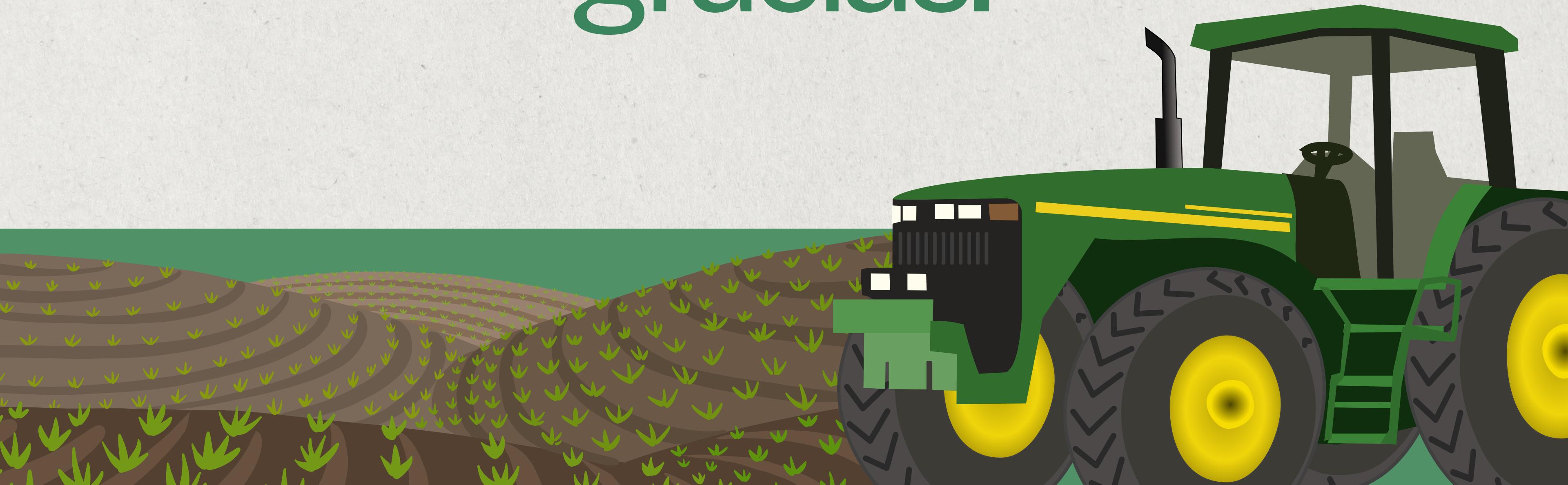
# Trabajo a futuro

## Mejoras:

- Diseño de un prototipo físico para implementar la placa.
- Sería interesante aplicar periféricos con Arduino.
- Utilizar una fuente externa de alimentación para la placa.

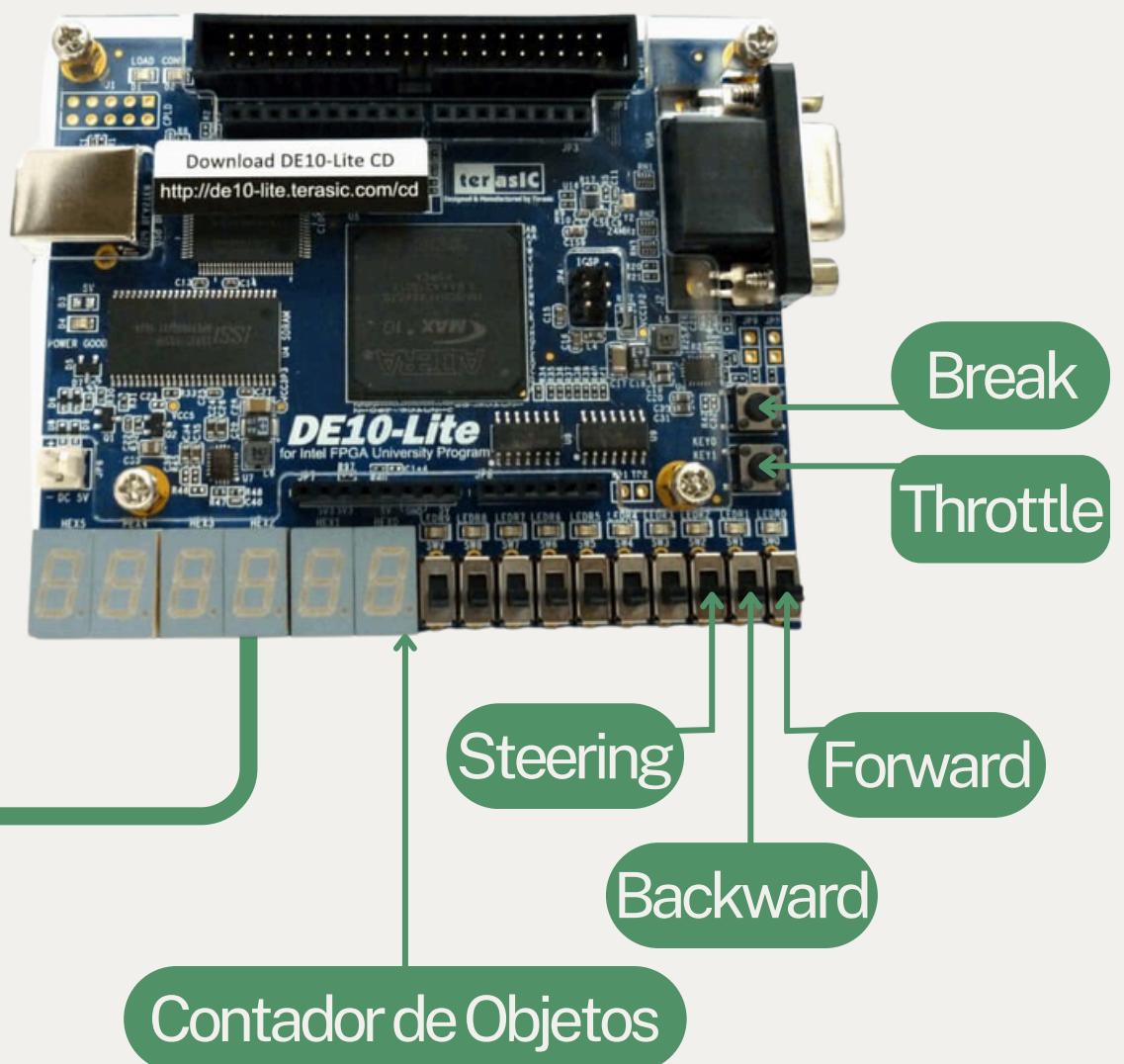
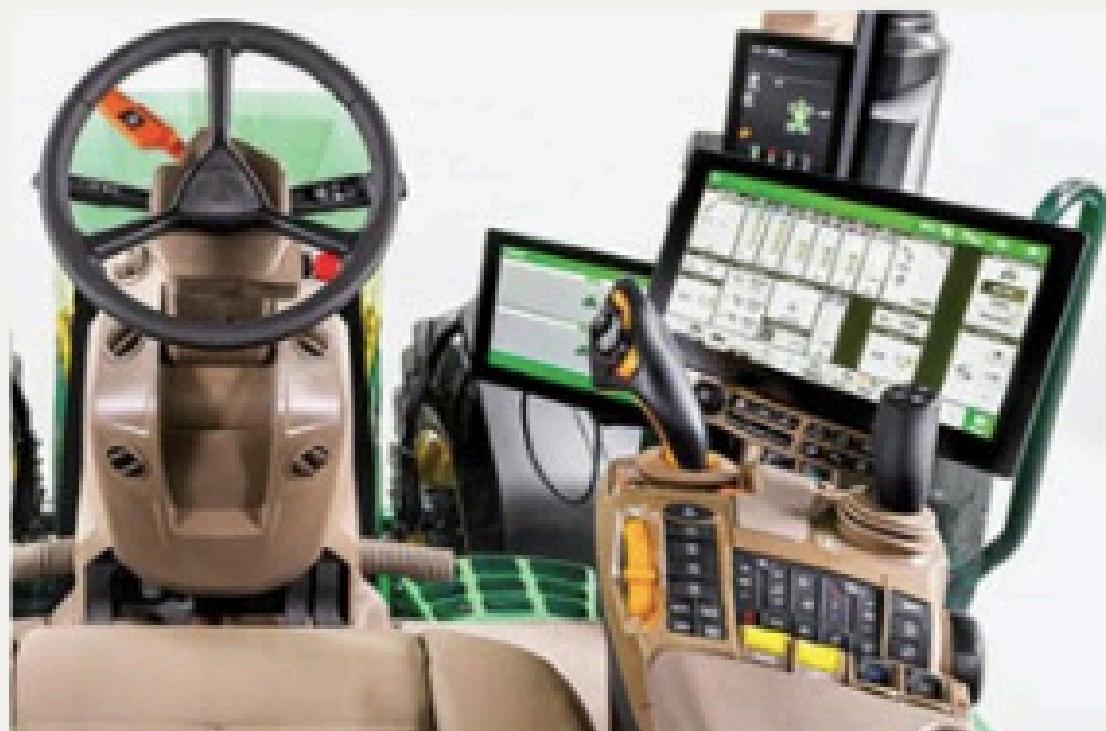


¡Muchas  
gracias!





## Conducción de un tractor



Comunicación serial