

# Uso de teclado

Tyrone Novillo

Facultad de ingeniería

Universidad de Cuenca

Cuenca, Ecuador

tyrone.novillo@ucuenca.edu.ec

**Resumen**—This project implements a hardware system that detects key presses on a matrix keypad and displays the corresponding data on a 7-segment display on the Basys 3 board. The design uses predefined VHDL components to improve modularity and code reuse, specifically incorporating an anti-bounce system for the keypad and a frequency divider component derived from the 100 MHz clock provided by the Basys 3.

**Keywords**— Matrix keypad, 7-segment display, Basys 3, VHDL components, anti-bounce system, frequency divider, hardware implementation, modular design, code reuse, digital circuit design.

## I. INTRODUCCIÓN

En esta práctica, se abordará el diseño e implementación de un sistema digital que permite la lectura de un teclado matricial y la visualización de las teclas presionadas en un display de 7 segmentos, utilizando la plataforma Basys 3 y el lenguaje de modelado de hardware VHDL. Se emplearán componentes predefinidos para facilitar la reutilización de código y mejorar la eficiencia del diseño, incluyendo un sistema antirebote para asegurar lecturas precisas del teclado y un divisor de frecuencia para generar señales de reloj adecuadas. Este enfoque modular no solo simplifica el proceso de desarrollo, sino que también permite una fácil integración y adaptación de componentes en futuros proyectos, destacando las ventajas de la reutilización de código en el diseño de sistemas digitales.

## II. DESARROLLO

Se requiere implementar un hardware que detecte el ingreso del teclado matricial y muestre los datos en los displays de 7 segmentos de la placa Basys 3. Para ello, se hará uso de los componentes del lenguaje de modelado de hardware vhdl. Los componentes que se requieren son dos: sistema antirebote para la lectura del teclado matricial y un componente para dividir frecuencia a partir del reloj de 100 MHz proporcionado por la Basys 3. Adicionalmente, se llamará al componente de lectura del teclado en el programa principal.

El teclado matricial compuesto de 4 filas y 4 columnas como el de la Figura 1 es un periférico que basa su funcionamiento en detección de circuitos abiertos y cerrados es decir, para la detección de una tecla presionada se procede de la siguiente forma:

- De manera periódica se envía un bit “1” al pin Fila(1), luego al pin Fila(2), y así sucesivamente hasta llegar al pin Fila(4); es decir, al vector Fila (1:4) se asignaría los vectores: 0001, luego 0010, luego 0100, finalmente 1000 para volver a repetir este proceso de manera indefinida.
- Al mismo tiempo que se realiza lo descrito en (1) se debe monitorear lo que se recibe en los pines Columna(1:4) con lo cual se puede detectar la tecla presionada

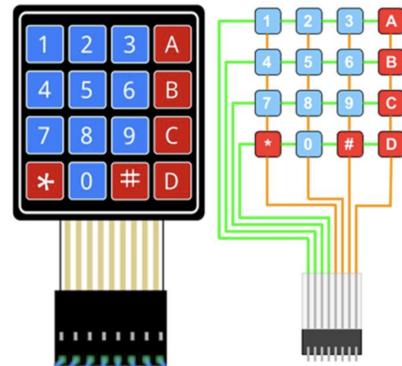


Figura 1: Teclado matricial y circuitería interna

### II-A. Componente para dividir frecuencias

La entidad divisor\_frecuencia se encarga de dividir la frecuencia de una señal de entrada (clk\_in) para generar una señal de salida (clk\_out) con una frecuencia más baja. Las constantes y puertos de entrada y salida se definen de la siguiente manera:

- clk\_in (entrada): Esta señal de tipo STD\_LOGIC representa el reloj de entrada cuya frecuencia se desea dividir.
- clk\_out (salida): Esta señal de tipo STD\_LOGIC es el reloj de salida, que tiene una frecuencia dividida según el factor especificado por la constante divisor.
- divisor (constante genérica): Es una constante de tipo integer que define el factor por el cual se va a dividir la frecuencia de la señal de entrada clk\_in. Este valor es configurable y determina cuántos ciclos de clk\_in se necesitan para cambiar el estado de clk\_out.

La arquitectura del componente se llama divisor y contiene dos señales internas: contador y clk\_temp. La señal contador es un contador que varía entre 0 y el valor de la constante divisor. La señal clk\_temp es una señal temporal que se utiliza para generar la salida clk\_out.

El proceso dentro de la arquitectura se ejecuta en cada flanco ascendente de la señal de entrada clk\_in. Si el valor del contador es menor que divisor - 1, el contador se incrementa en uno. Cuando el contador alcanza el valor divisor - 1, la señal clk\_temp cambia su estado (de ‘0’ a ‘1’ o de ‘1’ a ‘0’), y el contador se reinicia a 0. Este mecanismo asegura que la señal de salida clk\_out tenga una frecuencia dividida por el valor especificado en divisor.

### II-B. Componente de lectura de teclado y antirrebote

El componente teclado está diseñado para leer datos de un teclado matricial y cuenta con un sistema de anti-rebote.

El sistema de anti-rebote es esencial para evitar la detección incorrecta de teclas presionadas debido a los rebotes mecánicos que ocurren cuando una tecla es presionada. Si no se utilizara este sistema, se podrían generar múltiples detecciones de una misma tecla presionada, causando errores en la lectura del teclado.

A continuación, se describen los puertos de entrada y salida:

- **clk** (entrada): Esta señal de tipo `STD_LOGIC` representa el reloj principal del sistema.
- **columnas** (entrada): Es un vector de tipo `STD_LOGIC_VECTOR` de 4 bits que representa las columnas del teclado matricial.
- **filas** (salida): Es un vector de tipo `STD_LOGIC_VECTOR` de 4 bits que representa las filas del teclado matricial.
- **boton\_pres** (salida): Es un vector de tipo `STD_LOGIC_VECTOR` de 4 bits que indica qué botón del teclado ha sido presionado.
- **ind** (salida): Es una señal de tipo `STD_LOGIC` que sirve como indicador de que se ha detectado una tecla presionada.

En la arquitectura control del componente se definen dos constantes, `DELAY_1MS` y `DELAY_10MS`, que se utilizan para generar señales de reloj de 1 ms (`clk_1ms`) y 10 ms (`clk_10ms`), respectivamente, utilizando instancias del componente divisor\_frecuencia definido previamente.

Cada 10 ms, la señal `fila_reg` se desplaza para activar la siguiente fila del teclado, asegurando que todas las filas sean escaneadas periódicamente. Esto se traduce en enviar señales periódicamente a cada una de las filas del circuito de la Figura 2. Esta señal se asigna directamente a la salida `filas`.

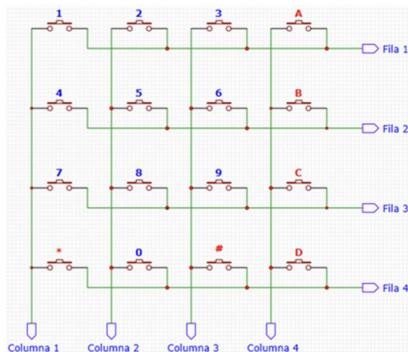


Figura 2: Esquema de circuito interno del teclado matricial

El sistema de anti-rebote funciona mediante el uso de registros de 8 bits para cada una de las teclas del teclado matricial. Los registros (`reg_0`, `reg_1`, `reg_2`, etc.) almacenan los valores de las columnas del teclado en función de la fila activa. Cada vez que se detecta un flanko ascendente en `clk_1ms`, el valor actual de las columnas se desplaza en cada uno de los registros correspondientes a la fila activa. Si el valor del registro se llena completamente con unos (0xFF), se considera que una tecla ha sido presionada, y se actualiza la señal `tecla_detectada` con el código correspondiente de la tecla.

La señal `ind` se utiliza como un pulso indicador que señala que se ha detectado una tecla presionada. Esta señal se genera en cada flanko ascendente del reloj principal (`clk`) y se asigna a la salida `ind_pulse`.

### II-C. Programa principal

El componente `main` representa el programa principal que integra el uso del componente `teclado` para la lectura de un teclado matricial y la visualización del valor de la tecla presionada en un display de 7 segmentos. Los puertos de entrada y salida usados en esta entidad son los siguientes:

- **clk** (entrada): Señal de reloj principal del sistema.
- **columnas** (entrada): Vector de 4 bits que representa las columnas del teclado matricial.
- **filas** (salida): Vector de 4 bits que representa las filas del teclado matricial.

- **anodo** (salida): Vector de 4 bits que controla los anodos del display de 7 segmentos.
- **segmentos\_7** (salida): Vector de 7 bits que controla los segmentos del display de 7 segmentos.

En la arquitectura final del componente `main`, se definen señales internas para la comunicación con el componente `teclado`:

- **boton\_pres** (señal): Vector de 4 bits que almacena el valor del botón presionado.
- **ind** (señal): Señal que indica la detección de una tecla presionada.
- **numero** (señal): Vector de 4 bits que almacena el valor del botón presionado para ser mostrado en el display de 7 segmentos.

El componente `teclado` se instancia dentro del componente `main` de la siguiente manera:

```
teclado_inst : teclado
port map (clk => clk,
          columnas => columnas,
          filas => filas,
          boton_pres => boton_pres,
          ind => ind);
```

Esta instancia conecta las señales de entrada y salida del componente `teclado` con las correspondientes señales definidas en el componente `main`.

El proceso de mostrar el valor del botón presionado en el display de 7 segmentos se realiza mediante un proceso sensible al flanko ascendente del reloj (`clk`):

Luego, mediante el reloj de 100MHz, se ejecuta un proceso para asignar el valor de la señal `boton_pres` a la señal `numero` en cada flanko ascendente del reloj.

Finalmente, se realiza la asignación del valor de `numero` a la salida `segmentos_7` mediante una instrucción `with select`, que convierte el valor binario de `numero` en el correspondiente patrón de segmentos para el display de 7 segmentos.

La entidad `main` integra el componente `teclado` para leer las teclas presionadas en un teclado matricial y muestra el valor de la tecla presionada en un display de 7 segmentos. La señal `anodo` se fija en 0111 para habilitar el primer display de 7 segmentos. El componente `teclado` se encarga de detectar qué tecla ha sido presionada y envía esta información al componente principal, que luego la visualiza en el display de 7 segmentos mediante un proceso de asignación y conversión de valores.

### II-D. Constraints

El archivo de constraints proporciona la asignación de los puertos de la entidad `main` a los pines específicos de la placa Basys 3, asegurando que las señales de entrada y salida estén correctamente conectadas a los componentes físicos de la placa. Este archivo se utiliza para configurar las conexiones entre el diseño en VHDL y el hardware real.

Primero, se define el pin de la señal de reloj (`clk`) que proporciona la Basys de 100MHz. Luego, se crean las conexiones para cada segmento del display de 7 segmentos. Los segmentos (`segmentos_7[6]` a `segmentos_7[0]`) se asignan a los pines W7, W6, U8, V8, U5, V5, y U7 respectivamente. Los anodos del display (`anodo[0]` a `anodo[3]`) se asignan a los pines U2, U4, V4, y W4.

En cuanto al teclado matricial, las filas (`filas[3]` a `filas[0]`) se asignan a los pines J1, L2, J2, y G2, respectivamente. Las columnas (`columnas[0]` a `columnas[3]`) se asignan a los pines H1, K2, H2, y G3, respectivamente. Estos pines corresponden a las conexiones del header Pmod JA de la Basys 3, que es donde se conectarán las señales del teclado matricial. Esta conexión lo muestra la Figura 3.

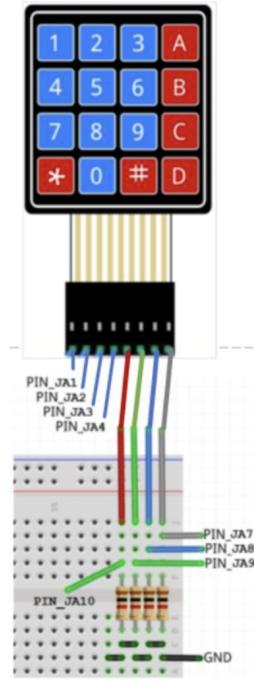


Figura 3: Conexión de pines de Basys 3 a teclado matricial.

Este archivo asegura que cada señal del diseño VHDL tenga una conexión física específica en la placa Basys 3, permitiendo que las señales del teclado matricial y del display de 7 segmentos se manejen adecuadamente. Las referencias a los pines físicos en la placa se basan en el esquema de la Basys 3, y la correspondencia de los pines puede verificarse visualmente utilizando la imagen de la placa que muestra la disposición de los pines para el teclado matricial y el display de 7 segmentos.

### III. CONCLUSIONES

El uso de componentes modulares, como el divisor\_frecuencia y teclado, permite reutilizar código de manera eficiente sin necesidad de reescribirlo, lo que ahorra tiempo y reduce la posibilidad de errores. La modularidad también facilita la comprensión y mantenimiento del diseño, ya que cada componente tiene una función específica y bien definida.

El sistema antirrebote del teclado es importante para asegurar la correcta detección de teclas presionadas. Este sistema funciona mediante el uso de registros de 8 bits para cada tecla del teclado matricial. Los registros almacenan los valores de las columnas del teclado en función de la fila activa. Cuando se detecta un flanko ascendente en la señal de reloj de 1 ms, el valor actual de las columnas se desplaza en los registros correspondientes. Si un registro se llena completamente con unos (0xFF), se interpreta que una tecla ha sido presionada y se actualiza la señal correspondiente.

Un teclado matricial funciona mediante la detección de circuitos abiertos y cerrados. Está compuesto de una matriz de filas y columnas. Para detectar una tecla presionada, se envían señales periódicas a cada fila y se monitorean las señales en las columnas. Cuando se presiona una tecla, se cierra un circuito entre la fila y la columna correspondiente, lo que permite identificar la tecla presionada.

### IV. ANEXOS

#### IV-A. Archivo main.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity main is
Port (
clk : in STD_LOGIC;
columnas : in STD_LOGIC_VECTOR (3 downto 0);
filas : out STD_LOGIC_VECTOR (3 downto 0);
anodo : out STD_LOGIC_VECTOR (3 downto 0);
segmentos_7 : out STD_LOGIC_VECTOR (6 downto 0)
);
end main;

architecture final of main is

signal boton_pres : STD_LOGIC_VECTOR (3 downto 0);
signal ind : STD_LOGIC;
signal numero : STD_LOGIC_VECTOR (3 downto 0);
component teclado
Port (
clk : in STD_LOGIC;
columnas : in STD_LOGIC_VECTOR (3 downto 0);
filas : out STD_LOGIC_VECTOR (3 downto 0);
boton_pres : out STD_LOGIC_VECTOR (3 downto 0);
ind : out STD_LOGIC
);
end component;

begin

anodo <= "0111";

-- Componente teclado
teclado_inst : teclado
port map (clk => clk, columnas => columnas, filas => filas);

-- Mostrar el valor del boton presionado en el display
process(clk)
begin
if rising_edge(clk) then
numero <= boton_pres;
end if;
end process;

-- Asignacion del valor de boton presionado al display
with numero select
segmentos_7 <=
"0000001" when x"0",
"1001111" when x"1",
"0010010" when x"2",
"0000110" when x"3",
"1001100" when x"4",
"0100100" when x"5",
"1100000" when x"6",
"0001111" when x"7",
"0000000" when x"8",
"0000100" when x"9",
"0001000" when x"A",
"1100000" when x"B",
"0110001" when x"C",
"1000010" when x"D",
"0110000" when x"E",
"0111000" when x"F",
"0000001" when others;
end final;
```

#### IV-B. Archivo divisor\_frecuencia.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity divisor_frecuencia is
generic (
divisor : integer
);
Port (
clk_in : in STD_LOGIC;
clk_out : out STD_LOGIC
);
end divisor_frecuencia;

architecture divisor of divisor_frecuencia is
signal contador : integer range 0 to divisor := 0;
signal clk_temp : STD_LOGIC := '0';
begin
process (clk_in)
begin
if rising_edge(clk_in) then
if contador < divisor - 1 then
contador <= contador + 1;
else
clk_temp <= not clk_temp;
contador <= 0;
end if;
end if;
end process;

clk_out <= clk_temp;
end divisor;

```

#### IV-C. Archivo teclado.vhd

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity teclado is
Port (
clk : in STD_LOGIC;
columnas : in STD_LOGIC_VECTOR (3 downto 0);
filas : out STD_LOGIC_VECTOR (3 downto 0);
boton_pres: out STD_LOGIC_VECTOR (3 downto 0);
ind : out STD_LOGIC
);
end teclado;

architecture control of teclado is
constant DELAY_1MS : integer := 50_000;
constant DELAY_10MS : integer := 500_000;

signal clk_1ms, clk_10ms : STD_LOGIC;
signal fila_reg : STD_LOGIC_VECTOR(3 downto 0) := "0000";
-- Inicializacion de registros de 8 bits con ceros
signal reg_1, reg_2, reg_3, reg_4, reg_5, reg_6, reg_7, reg_8, reg_9, reg_0, reg_A, reg_B, reg_C, reg_D;
signal tecla_detectada : STD_LOGIC_VECTOR(3 downto 0) := "0000";
signal tecla_presionada : STD_LOGIC_VECTOR(3 downto 0) := "0000";
signal ind_signal : STD_LOGIC := '0';      if rising_edge(clk) then
signal ind_pulse : STD_LOGIC := '0';      ind_signal <= '0';

```

```

component divisor_frecuencia
generic (
divisor : integer
);
Port (
clk_in : in STD_LOGIC;
clk_out : out STD_LOGIC
);
end component;

begin
-- Instancia del divisor de frecuencia para clk_1ms y clk_10ms
div1ms : divisor_frecuencia
generic map (divisor => DELAY_1MS)
port map (clk_in => clk, clk_out => clk_1ms);

div10ms : divisor_frecuencia
generic map (divisor => DELAY_10MS)
port map (clk_in => clk, clk_out => clk_10ms);

-- Enviar senales a filas
process(clk_10ms)
begin
if rising_edge(clk_10ms) then
fila_reg <= fila_reg(2 downto 0) & fila_reg(3);
end if;
end process;

filas <= fila_reg;

-- Proceso de anti-rebote
process(clk_1ms)
begin
if rising_edge(clk_1ms) then
case fila_reg is
when "0001" =>
reg_F <= reg_F(6 downto 0) & columnas(3);
reg_0 <= reg_0(6 downto 0) & columnas(2);
reg_E <= reg_E(6 downto 0) & columnas(1);
reg_D <= reg_D(6 downto 0) & columnas(0);
when "0010" =>
reg_7 <= reg_7(6 downto 0) & columnas(3);
reg_8 <= reg_8(6 downto 0) & columnas(2);
reg_9 <= reg_9(6 downto 0) & columnas(1);
reg_C <= reg_C(6 downto 0) & columnas(0);
when "0100" =>
reg_4 <= reg_4(6 downto 0) & columnas(3);
reg_5 <= reg_5(6 downto 0) & columnas(2);
reg_6 <= reg_6(6 downto 0) & columnas(1);
reg_B <= reg_B(6 downto 0) & columnas(0);
when "1000" =>
reg_1 <= reg_1(6 downto 0) & columnas(3);
reg_2 <= reg_2(6 downto 0) & columnas(2);
reg_3 <= reg_3(6 downto 0) & columnas(1);
reg_A <= reg_A(6 downto 0) & columnas(0);
when others =>
null;
end case;
end if;
end process;

```

```

if reg_0 = x"FF" then
tecla_detectada <= "0000";
ind_signal <= '1';
elsif reg_1 = x"FF" then
tecla_detectada <= "0001";
ind_signal <= '1';
elsif reg_2 = x"FF" then
tecla_detectada <= "0010";
ind_signal <= '1';
elsif reg_3 = x"FF" then
tecla_detectada <= "0011";
ind_signal <= '1';
elsif reg_4 = x"FF" then
tecla_detectada <= "0100";
ind_signal <= '1';
elsif reg_5 = x"FF" then
tecla_detectada <= "0101";
ind_signal <= '1';
elsif reg_6 = x"FF" then
tecla_detectada <= "0110";
ind_signal <= '1';
elsif reg_7 = x"FF" then
tecla_detectada <= "0111";
ind_signal <= '1';
elsif reg_8 = x"FF" then
tecla_detectada <= "1000";
ind_signal <= '1';
elsif reg_9 = x"FF" then
tecla_detectada <= "1001";
ind_signal <= '1';
elsif reg_A = x"FF" then
tecla_detectada <= "1010";
ind_signal <= '1';
elsif reg_B = x"FF" then
tecla_detectada <= "1011";
ind_signal <= '1';
elsif reg_C = x"FF" then
tecla_detectada <= "1100";
ind_signal <= '1';
elsif reg_D = x"FF" then
tecla_detectada <= "1101";
ind_signal <= '1';
elsif reg_E = x"FF" then
tecla_detectada <= "1110";
ind_signal <= '1';
elsif reg_F = x"FF" then
tecla_detectada <= "1111";
ind_signal <= '1';
end if;

if ind_signal = '1' then
tecla_presionada <= tecla_detectada;
end if;
end if;
end process;

-- Asignacion de la tecla presionada a la salida
boton_pres <= tecla_presionada;
-- Asignacion del pulso salida del componente
ind <= ind_pulse;

-- Proceso para generar un pulso de ind
process(clk)
begin
if rising_edge(clk) then
ind_pulse <= ind_signal;
end if;

```

**IV-D. Archivo de Constraints Basys-3-Master.xdc**

```

## Clock signal
set_property -dict { PACKAGE_PIN W5
    ↪ IOSTANDARD LVCMOS33 } [get_ports clk]
#create_clock -add -name sys_clk_pin -period
    ↪ 10.00 -waveform {0 5} [get_ports clk]

## Segment Display
set_property -dict { PACKAGE_PIN W7
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[6]}]
set_property -dict { PACKAGE_PIN W6
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[5]}]
set_property -dict { PACKAGE_PIN U8
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[4]}]
set_property -dict { PACKAGE_PIN V8
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[3]}]
set_property -dict { PACKAGE_PIN U5
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[2]}]
set_property -dict { PACKAGE_PIN V5
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[1]}]
set_property -dict { PACKAGE_PIN U7
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {segmentos_7[0]}]

#set_property -dict { PACKAGE_PIN V7
    ↪ IOSTANDARD LVCMOS33 } [get_ports dp]

set_property -dict { PACKAGE_PIN U2
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {anodo[0]}]
set_property -dict { PACKAGE_PIN U4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {anodo[1]}]
set_property -dict { PACKAGE_PIN V4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {anodo[2]}]
set_property -dict { PACKAGE_PIN W4
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {anodo[3]}]

##Pmod Header JA
set_property -dict { PACKAGE_PIN J1
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {filas[3]}]; #Sch name = JA1
set_property -dict { PACKAGE_PIN L2
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {filas[2]}]; #Sch name = JA2
set_property -dict { PACKAGE_PIN J2
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {filas[1]}]; #Sch name = JA3
set_property -dict { PACKAGE_PIN G2
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {filas[0]}]; #Sch name = JA4
set_property -dict { PACKAGE_PIN H1
    ↪ IOSTANDARD LVCMOS33 } [get_ports
    ↪ {columnas[0]}]; #Sch name = JA7

```

```

set_property -dict { PACKAGE_PIN K2
    ↳ IOSTANDARD LVCMOS33 } [get_ports
    ↳ {columnas[1]}]; #Sch name = JA8
set_property -dict { PACKAGE_PIN H2
    ↳ IOSTANDARD LVCMOS33 } [get_ports
    ↳ {columnas[2]}]; #Sch name = JA9
set_property -dict { PACKAGE_PIN G3
    ↳ IOSTANDARD LVCMOS33 } [get_ports
    ↳ {columnas[3]}]; #Sch name = JA10

## Configuration options, can be used for all
## designs
set_property CONFIG_VOLTAGE 3.3
    ↳ [current_design]
set_property CFGBVS VCCO [current_design]

## SPI configuration mode options for QSPI
## boot, can be used for all designs
set_property BITSTREAM.GENERAL.COMPRESS TRUE
    ↳ [current_design]
set_property BITSTREAM.CONFIG.CONFIGRATE 33
    ↳ [current_design]
set_property CONFIG_MODE SPIx4
    ↳ [current_design]

```

*IV-E. Funcionamiento de la maquina de estados en la Basys 3*

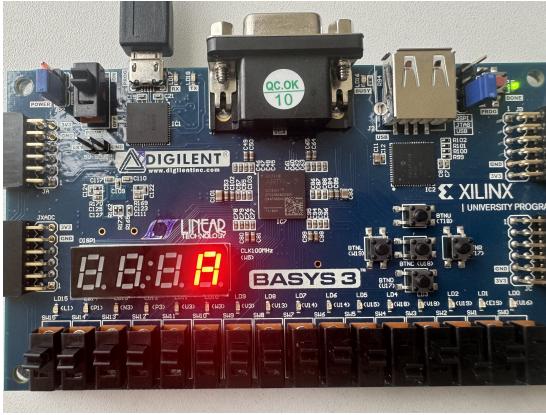


Figura 4: Display tras presionar tecla A

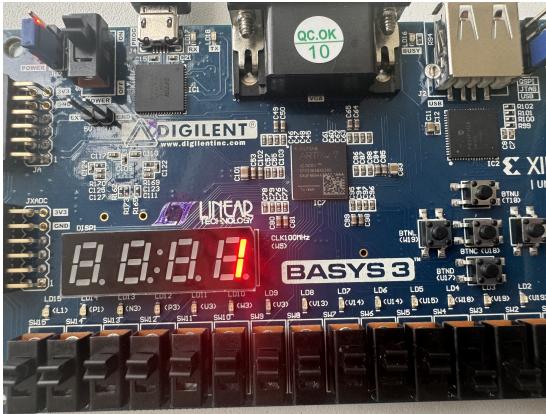


Figura 5: Display tras presionar tecla 1

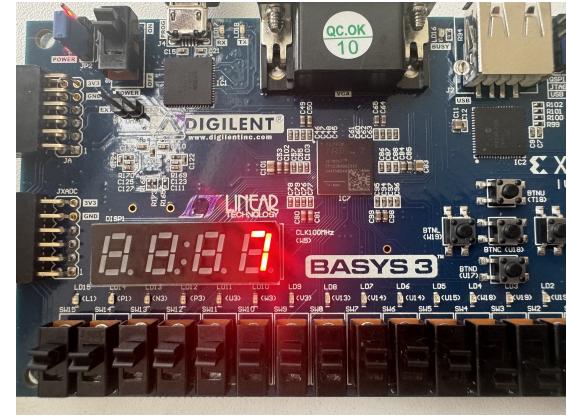


Figura 6: Display tras presionar tecla 7