Term Project: Gyroscope controlled pan tilt system

16.05.2024

Melike Nur SARI

EEE 102-04

Purpose of the project

The purpose of this project is to control a pan tilt system with a gyroscope. With this it will be possible to control a camera system with just hand movements.

Design Specification

This design controls a pan tilt system with the inputs it gets from the gyroscope. The system turns with servo motors. The servo motors are SG90 mini servos, and the gyroscope is MPU6050 gyroscope. MPU6050 is connected to Basys3 with I2C communication, and the system uses the angular velocity values from the x axis and y axis. There are 2 inputs, 1 inout and 3 outputs as follows:

clock : IN STD_LOGIC

rst: IN STD_LOGIC

servo1: OUT STD_LOGIC

servo2: OUT STD_LOGIC

sda : inout std_logic

scl : out std_logic

The clock is connected to the internal clock of Basys3. Sda and scl are essential for I2C communication, sda gives to and takes data from the gyroscope and scl is the clock for the gyroscope. servo1 and servo2 controls the two servos. The servo motors stay in middle, which is 90 degrees position, if the angular velocity is read as 0 from the gyroscope. If the gyroscope turns right in the x axis, then servo 2 turns 90 degrees to right and stays in the 180 degrees position. With this the pan tilt system turns right as well. If the gyroscope turns left, servo 2 turns 90 degrees left and stays in 0 degrees position. As a result, the system turns left. If the gyroscope turns upwards in the y axis, then servo 1 turns about 75 degrees upwards and stays in the 165-degree position, and the system turns upwards. And finally, if the gyroscope turns downwards then the servo 1 and the system turns 75 degrees downwards as well.

The pan-tilt is as following:

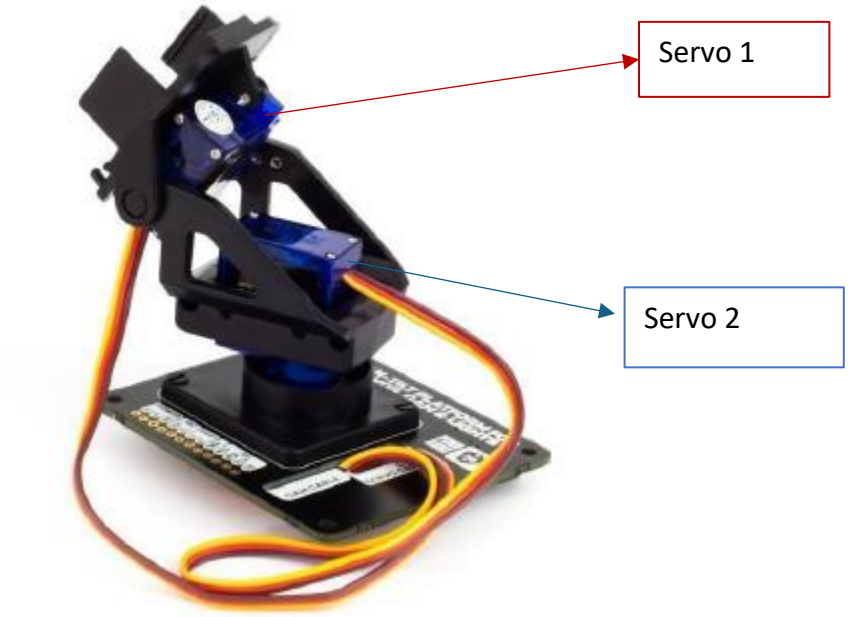

Figure 1: Pan- tilt system

Methodology

I2C protocol

The gyroscope is connected to Basys3 through I2C communication. The signals are given through sda and scl such that it works as the following:

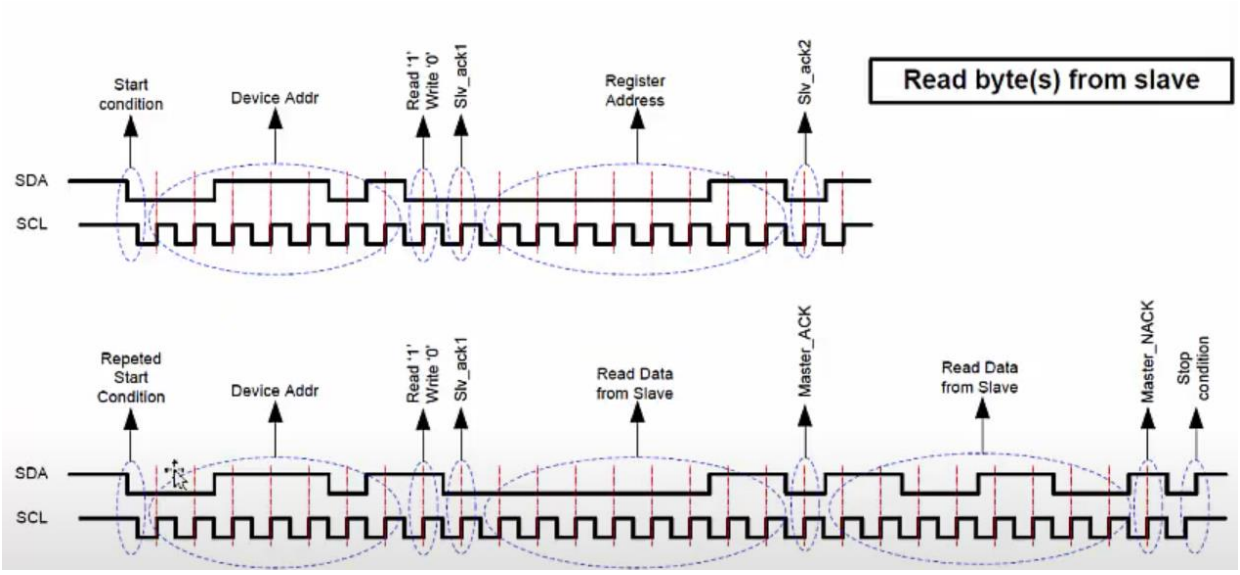

Figure 2: Example of how I2C communication protocol works

In this I2C communication protocol the master is Basys 3, and the slave is MPU 6050. The sda and scl are both 1 if specified otherwise.

First the code starts the sda and scl as both 1. Then sda becomes 0, and while it is 0 the clock signal to gyroscope, scl, also becomes 0. This gives the start command to MPU 6050. After that the code should not make sda 0 when scl is 1, if there is no start condition there. So sda should change value only on the falling edge of scl.

After the start condition the device address is given through sda. This is especially necessary for specifying the use of the MPU6050 if there were multiple slaves in the circuit. With the device address MPU 6050 understands that the master will use it and not any other slave. The device address of the MPU 6050 is "1101000".

Then, the master gives the write signal '0' to give the register address to MPU 6050. After the write signal and before the register address, the slave, MPU 6050, gives acknowledge signal, which makes sda '0'. With this we can understand that the slave acknowledged that we are trying to read data from it. The register address is used to read different datas from the gyroscope. For example, we first give the register address to sda as "01000100" to read the last 8 bits of the value of angular velocity of the x-axis. The register address used from the datasheet is as follows (the first column is the hexadecimal value and the second column is the decimal value of the 8 bit register address):

43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]

Table 1: Register addresses of MPU 6050

Later, we get the second acknowledge signal as '0' and then we repeat the start condition. We give the device address again, and this time we give the read signal '1' to read the values from the gyroscope. We receive our third acknowledgement signal as '0' and then we read the values. First, we read the last 8 bits of the gyroscope values of x-axis. After repeating the same process and changing the register address to "01000011" we read the first 8 bits of the gyroscope values of the x-axis. Then we get a 16 value of the angular velocity from the x-axis. We repeat the same process and get the values for the y axis this time.

Sda should read the value as "1111111111111111" if there are no angular movements on the gyroscope. However, the gyroscope will read a different value than

“1111111111111111” even if there is a slight movement on it, so to reduce this error, after checking which value the gyroscope reads when there is no movement, it is decided to keep the servo motors at 90 degrees when the gyroscope reads a value bigger than “1111100000000000”. If that is not the case in the values read from the x-axis, and if the 16th bit of the data is equal to ‘1’, then this means that the gyroscope is being turned to left. If that is not the case either, then it means that the gyroscope is being turned to the right. The system turns accordingly to these values. The same values apply to y-axis as well.

After reading the data, the master gives nack signal, the inverse acknowledge signal, which is ‘1’. Then scl becomes ‘1’, and while scl is ‘1’ sda becomes ‘1’ as well. This is the stop condition for the slave. After this we can repeat the process to read different values.

The working principle of servo

Servo motors are working in the principle of pulse width modulation. To control the servos, we need to generate waveforms such as these:

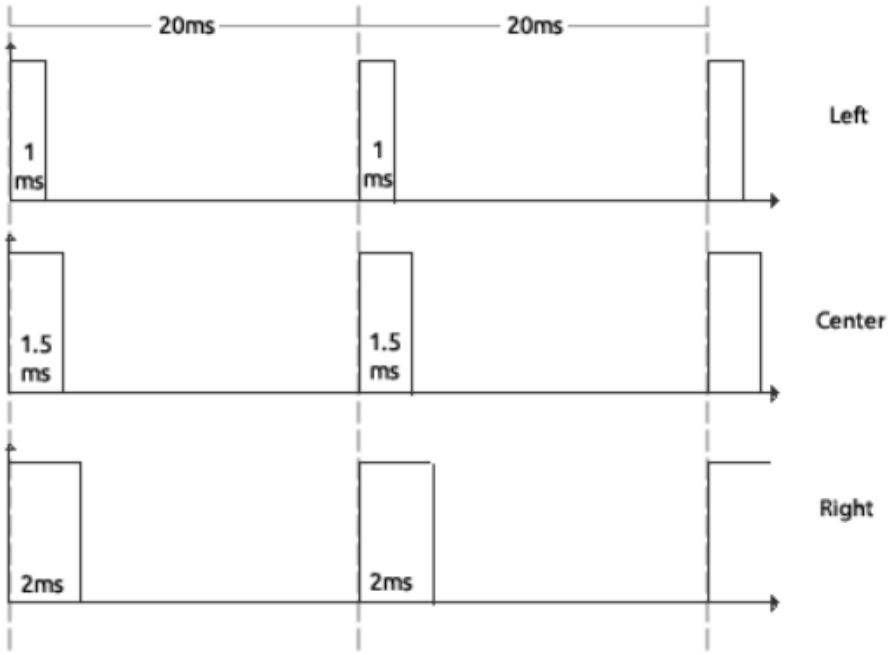

Figure 3: Pulse width modulation waveforms

The servos need 1 ms to 2 ms pulses in 20 ms periods to turn and servos can take up to 128 positions. That is why we need a clock frequency of f=((2 ms-1ms)/128)^(-1)= 128 kHz. I used a clock divider to make a 128 kHz clock from the internal clock of Basys 3 which is 100 MHz. From the gyroscope values, the servos turn accordingly. For example, if the

gyroscope turns right in x-axis, 2 ms pulses in 20 ms periods are sent to servo 2 and servo 2 turns right.

Results

The oscilloscope results show the sda and scl signals of MPU 6050. The green one is scl and yellow one is sda.

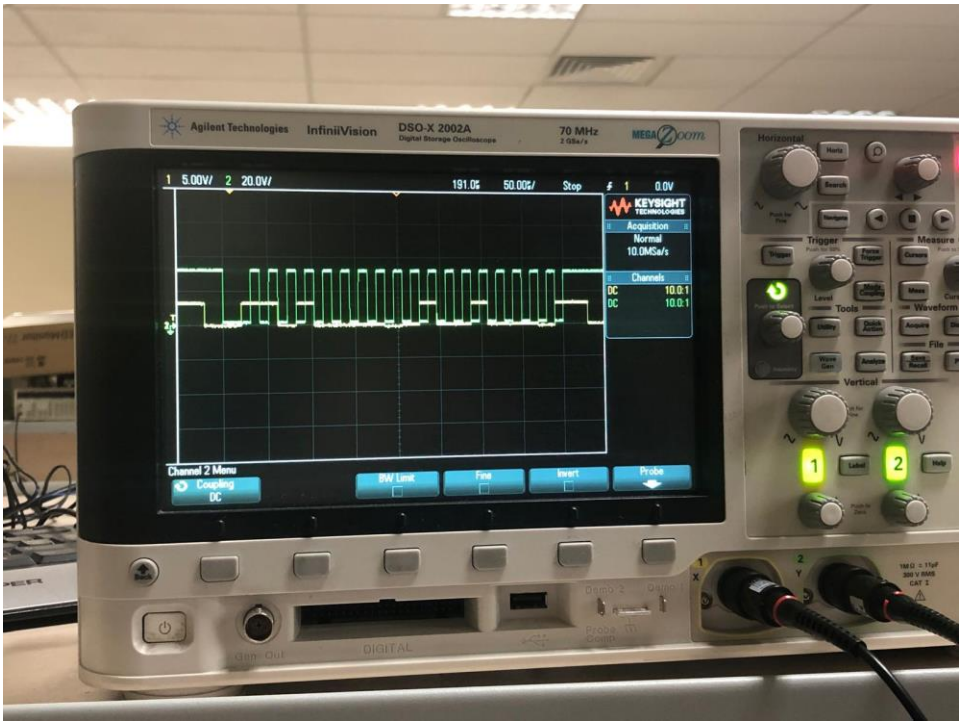

Figure 4: Oscilloscope results of I2C (1)

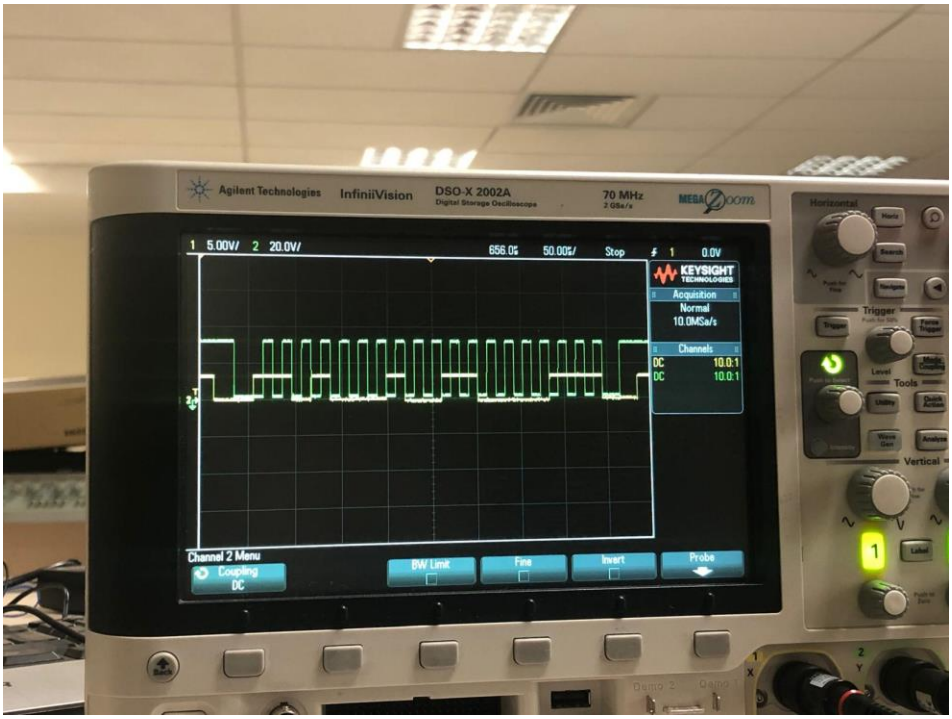

Figure 5: Oscilloscope results of I2C (2)

The results of the system are shown below:

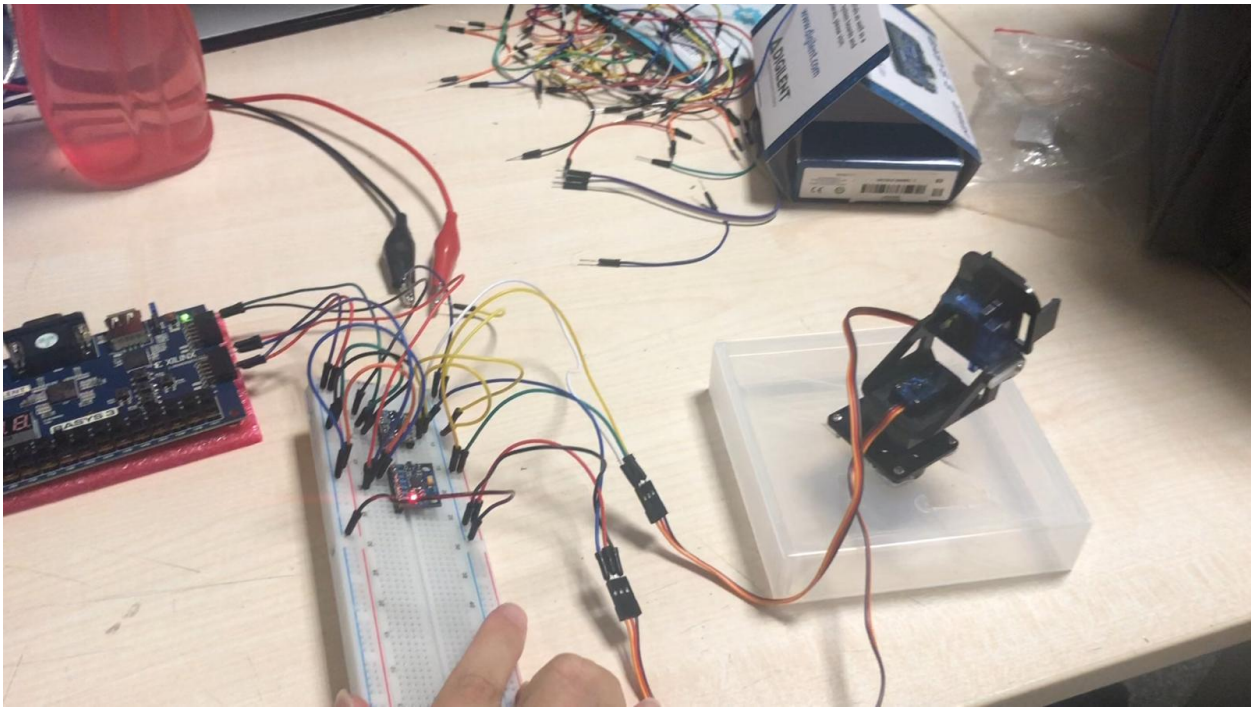

Figure 6: Gyroscope turns right

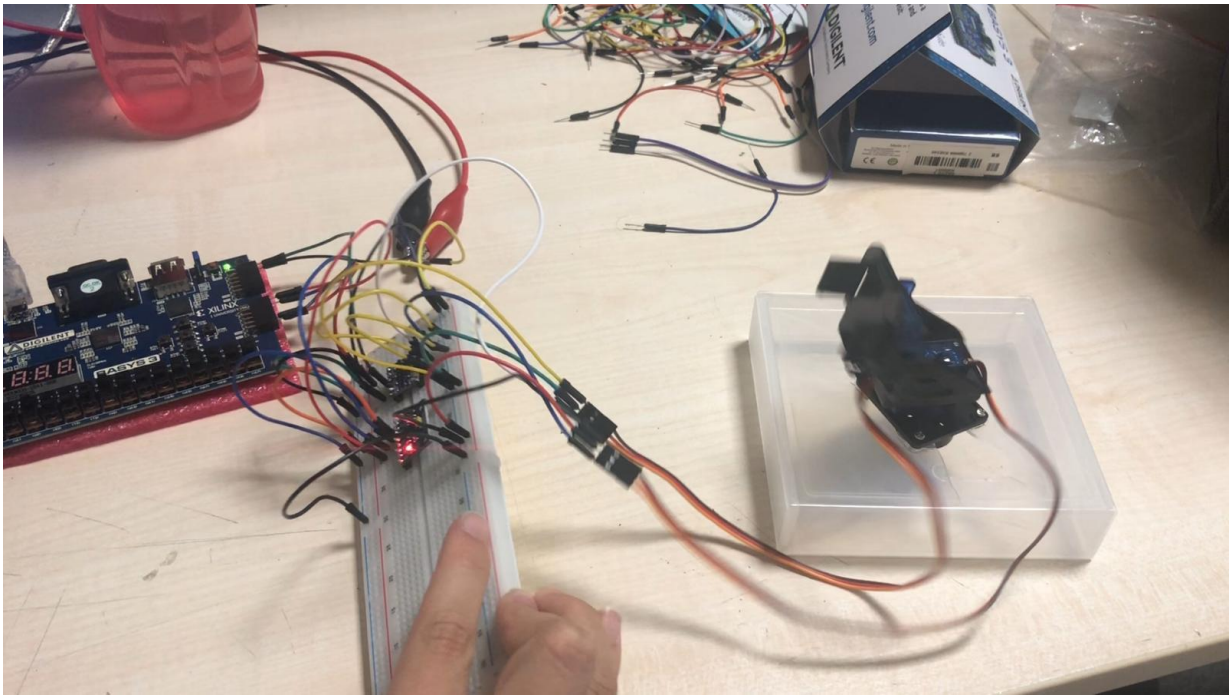

Figure 7: Gyroscope turns left

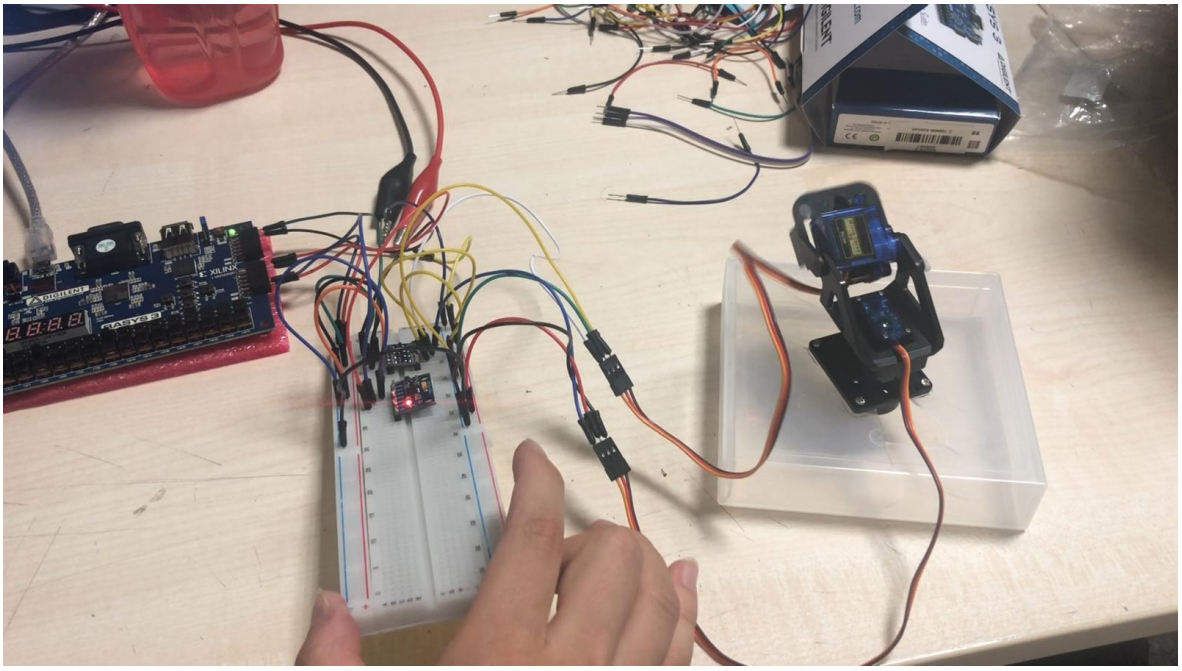

Figure 8: Gyroscope turns downward

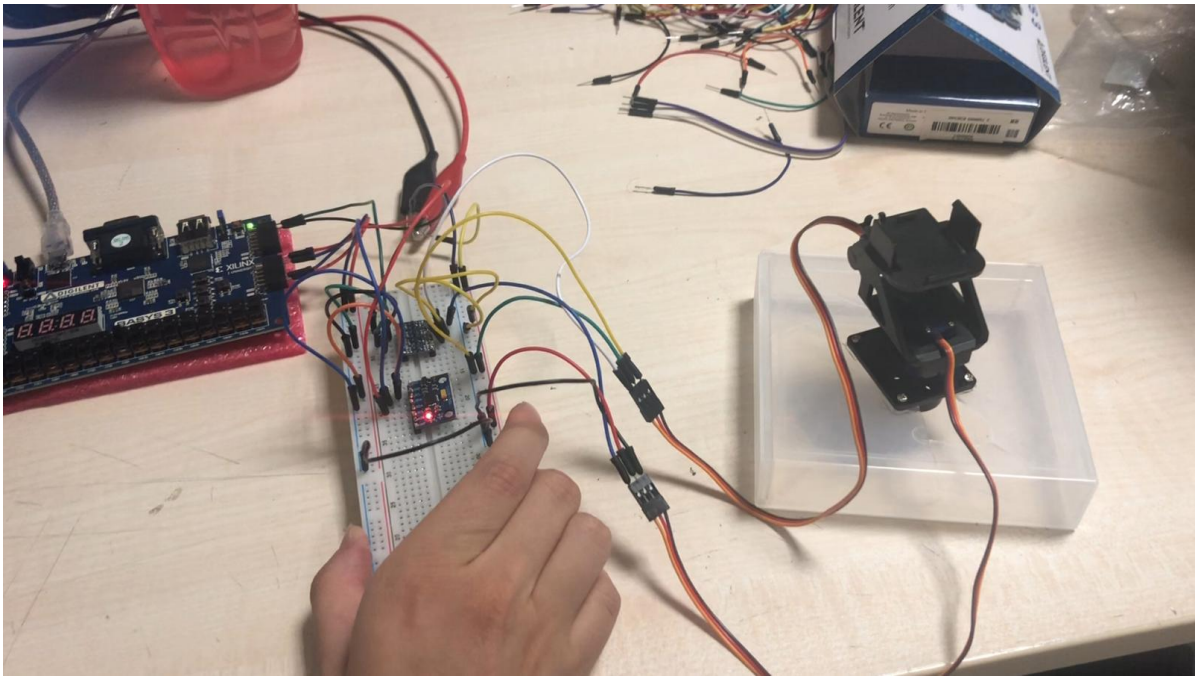

Figure 9: Gyroscope turns upwards

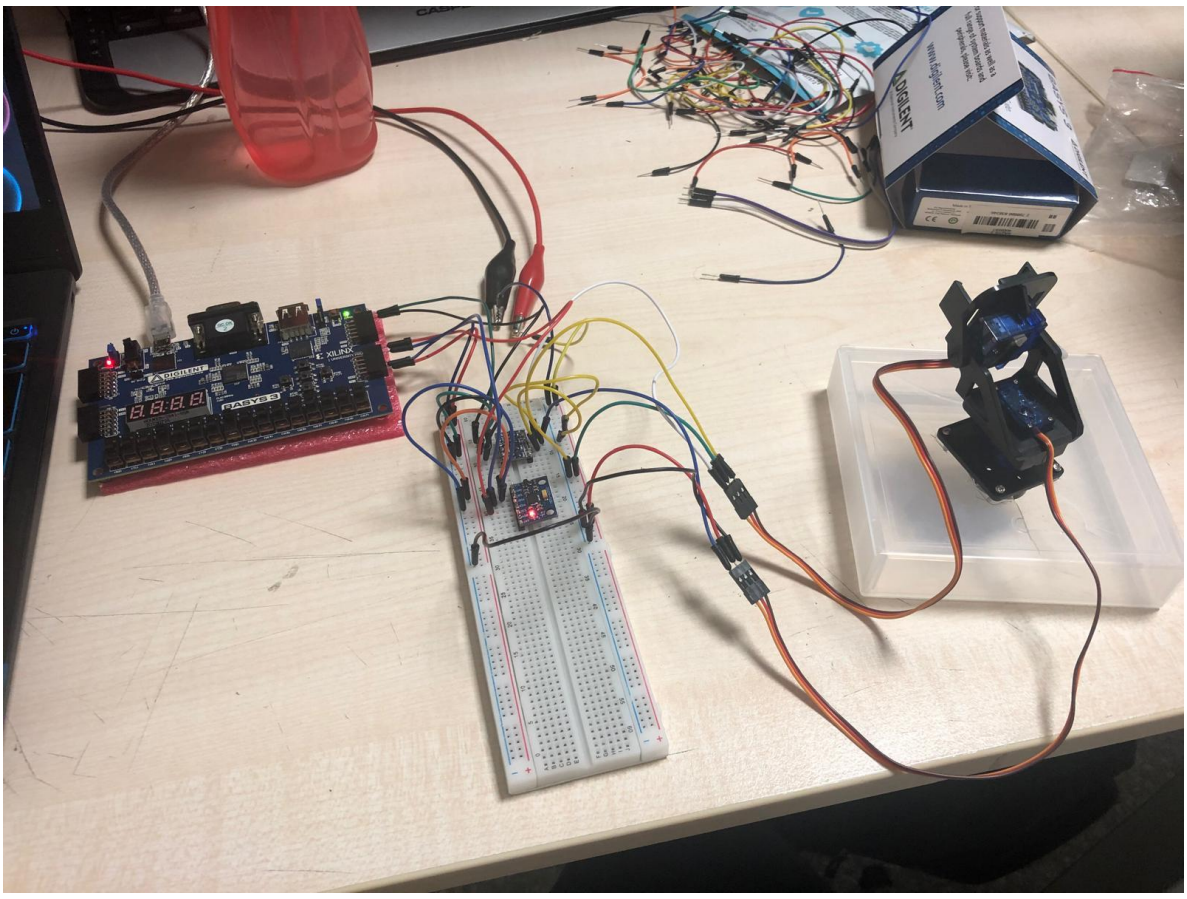

Figure 10: Circuit photo

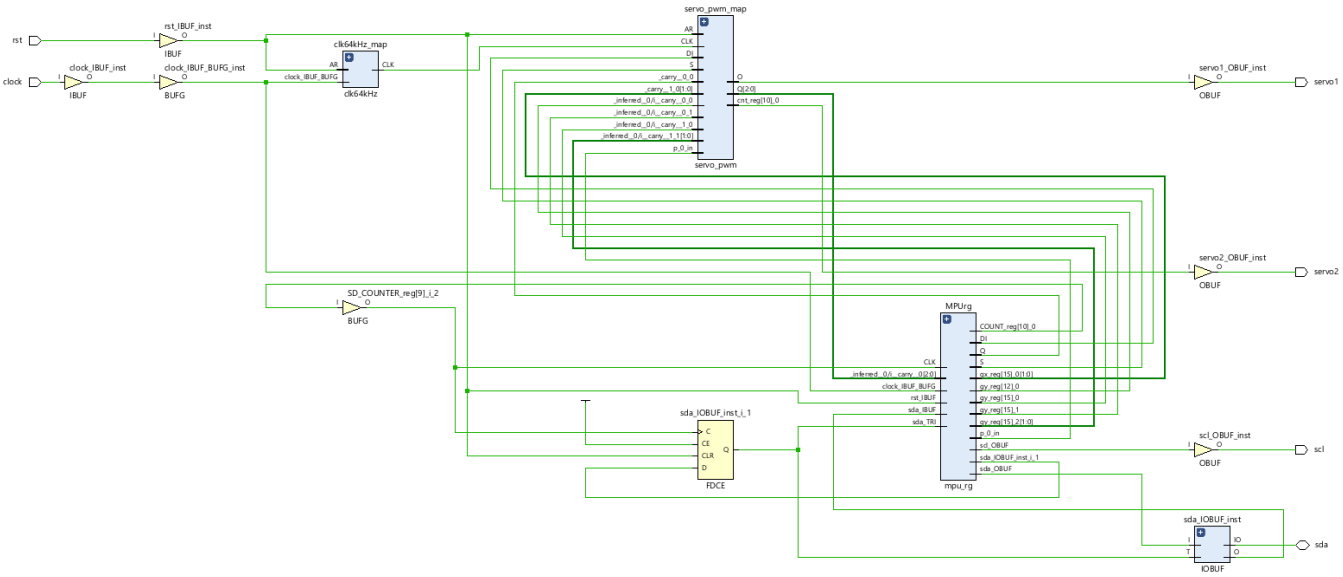

Figure 11: Schematic of the design

Conclusion

This experiment's purpose was to control a pan tilt system with a gyroscope. The project was successful. I was able to get values from MPU6050 with I2C communication protocol and the system turned according to these values with servo motors. I learnt how to use I2C communication protocol and how to control servo motors with pulse width modulation, I believe these, especially the I2C communication, will be beneficial for my other works in the future. I2C communication was complex to understand how it works and how the code works, so the code I chose was a bit too long. And I tried to shorten the code so that I only read values of the angular velocity from the x-axis and y-axis, however no matter what I tried the shortened code did not work. I read only 0 values or did not get the acknowledgement signal from the slave. Even though it worked for some time, when I closed and opened Basys 3 it stopped working, which I do not understand why happened. I can develop the code in the future so that it's shorter. And there are some other errors as well. When I turn the gyroscope upwards it turns a little bit right as well. This might be because of I turn the gyroscope right without realizing.

References

- Aykenar, M. Mehmet Burak Aykenar. 2021, 26 February.
- Moran, Daniel. “mpu6050-vhdl.” Github, 2020. https://github.com/danomora/mpu6050-vhdl. Retrieved on 19.05.2024.
- Ramos, Carlos A. “Servomotor Control with PWM and VHDL.” CodeProject. CodeProject, December 20, 2012. https://www.codeproject.com/Articles/513169/ServomotorControl-with-PWM-and-VHDL.

Appendices

```
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 temp is
Port (  clock  : IN  STD_LOGIC;
      rst: IN  STD_LOGIC;

      servo1: OUT STD_LOGIC;
      servo2: OUT STD_LOGIC;
```

```
      sda    : inout std_logic;
scl    : out std_logic);
end temp;

architecture Behavioral of temp is
   COMPONENT servo_pwm
      PORT (
         clock   : IN  STD_LOGIC;
         rst : IN  STD_LOGIC;
         pos1   : IN  STD_LOGIC_VECTOR(6 downto 0);
         pos2: IN STD_LOGIC_VECTOR(6 downto 0);
         servo1 : OUT STD_LOGIC;
         servo2: OUT STD_LOGIC

      );
   END COMPONENT;

      COMPONENT clk64kHz
      PORT(
         clock    : in  STD_LOGIC;
         rst  : in  STD_LOGIC;
         clk_out: out STD_LOGIC
      );
   END COMPONENT;

   signal clk_out : STD_LOGIC := '0';
```

```
        COMPONENT mpu_rg is
PORT(
CLOCK_50:    IN       STD_LOGIC;
reset_n:    in std_logic;
en:       in std_logic;
I2C_SDAT: INOUT     STD_LOGIC;
I2C_SCLK: OUT STD_LOGIC;
gx: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
gy: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
gz: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
ax: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
ay: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
az: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
);
END COMPONENT;

component selectt is
Port(
gyro1: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
gyro2: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
sel1:OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
sel2:OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
);
END COMPONENT;
```

```
signal reg_output_1 : std_logic_vector(15 downto 0):=(others=>'0'); -------gx
signal reg_output_2 : std_logic_vector(15 downto 0):=(others=>'0');------gy
signal reg_output_3 : std_logic_vector(15 downto 0):=(others=>'0');------gz
signal reg_output_4 : std_logic_vector(15 downto 0):=(others=>'0');-----ax
signal reg_output_5 : std_logic_vector(15 downto 0):=(others=>'0');-----ay
signal reg_output_6 : std_logic_vector(15 downto 0):=(others=>'0');-----az
signal en: std_logic:='1';
signal pos1  :  STD_LOGIC_VECTOR(6 downto 0):=(others=>'0');
signal pos2: STD_LOGIC_VECTOR(6 DOWNTO 0):=(others=>'0');



begin

   servo_pwm_map: servo_pwm PORT MAP(
   clk_out, rst, pos1,pos2, servo1,servo2
   );
   clk64kHz_map: clk64kHz PORT MAP(
      clock, rst, clk_out
   );

   selectt_map:selectt PORT MAP(
   reg_output_1,reg_output_2, pos1,pos2);

  MPUrg : mpu_rg
port map(
```

```
CLOCK_50  => clock,
reset_n   => rst,
en                    => en,
I2C_SDAT  => sda,
I2C_SCLK  => scl,
gx      =>      reg_output_1(15 downto 0),
gy      =>      reg_output_2(15 downto 0),
gz      =>      reg_output_3(15 downto 0),
ax      =>      reg_output_4(15 downto 0),
ay      =>      reg_output_5(15 downto 0),
az      =>      reg_output_6(15 downto 0)
);



end Behavioral;
----------------------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity servo_pwm is
   PORT (
      clock   : IN  STD_LOGIC;
      rst : IN  STD_LOGIC;
      pos1   : IN  STD_LOGIC_VECTOR(6 downto 0);
      pos2: IN  STD_LOGIC_VECTOR(6 downto 0);
```

```
    servo1 : OUT STD_LOGIC;
    servo2 : OUT STD_LOGIC
  );
end servo_pwm;

architecture Behavioral of servo_pwm is
  signal pwmi1: unsigned(7 downto 0);
  signal pwmi2: unsigned(7 downto 0);
  signal cnt : unsigned(10 downto 0);



begin

  pwmi1 <= unsigned('0' & pos1) + 32;
  pwmi2 <= unsigned('0' & pos2) + 32;
  counter: process (rst, clock) begin
    if (rst = '1') then
      cnt <= (others => '0');
    elsif rising_edge(clock) then
      if (cnt = 1279) then
        cnt <= (others => '0');
      else
        cnt <= cnt + 1;
      end if;
    end if;
  end process;
```

```
    servo1 <= '1' when (cnt < pwmi1) else '0';
    servo2 <= '1' when (cnt < pwmi2) else '0';
end Behavioral;
----------------------------------------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clk128kHz is
    Port (
        clock   : in  STD_LOGIC;
        rst  : in  STD_LOGIC;
        clk_out: out STD_LOGIC
    );
end clk128kHz;

architecture Behavioral of clk128kHz is
    signal counter : integer range 0 to 780 := 0;
    signal temporal: STD_LOGIC;

begin
    freq_divider: process (rst, clock) begin
        if (rst = '1') then
            counter  <= 0;
            temporal <= '0';
```

```
    elsif rising_edge(clock) then
      if (counter = 780) then
        counter  <= 0;
        temporal <= NOT(temporal);

      else
        counter <= counter + 1;
      end if;
    end if;
  end process;

  clk_out <= temporal;
end Behavioral;
----------------------------------------------------------------------------------------------------------
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 selectt is
Port (   gyro1: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
      gyro2: IN STD_LOGIC_VECTOR(15 DOWNTO 0);
sel1:OUT STD_LOGIC_VECTOR(6 DOWNTO 0);
sel2:OUT STD_LOGIC_VECTOR(6 DOWNTO 0)
 );
end selectt;

architecture Behavioral of selectt is

begin

process(gyro2)
begin
if(gyro2>"1111100000000000") then
sel2<="1000000";
elsif (gyro2(15)='1') then
sel2<="1111111";
else
sel2<="0000000";
end if;
end process;

process(gyro1)
begin
if(gyro1>"1111100000000000") then
```

```
sel1<="1000000";
elsif (gyro1(15)='1') then
sel1<="1110101";
else
sel1<="0001011";
end if;
end process;

end Behavioral;
----------------------------------------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;



ENTITY mpu_rg is
PORT(
CLOCK_50:    IN       STD_LOGIC;
reset_n:    in std_logic;
en:       in std_logic;
I2C_SDAT: INOUT     STD_LOGIC;
I2C_SCLK: OUT STD_LOGIC;
gx: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
gy: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
gz: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
ax: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
```

```
ay: OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
az: OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
);
END mpu_rg;

ARCHITECTURE SOLUCION OF mpu_rg IS
SIGNAL resetn, GO,  SDI, SCLK: STD_LOGIC;
signal data: std_logic_vector(15 downto 0);
signal address: std_logic_vector(6 downto 0) := "1101000"; ---mpu6050  default address
SIGNAL SD_COUNTER: INTEGER  RANGE 0 TO 610;
SIGNAL COUNT: STD_LOGIC_VECTOR(10 downto 0);

BEGIN
RESETN <= reset_n;
Process(CLOCK_50)
BEGIN
IF(RISING_EDGE(CLOCK_50)) THEN COUNT<=COUNT+1;
END IF;
END PROCESS;

Process(COUNT(10), RESETN)
BEGIN
IF(RESETN='1') THEN
GO<='0';
ELSIF(en='1') THEN
GO<='1';
```

```
END IF;
END PROCESS;

PROCESS(COUNT(10), RESETN)
BEGIN
IF(RESETN='1') THEN
SD_COUNTER<=0;
ELSIF(RISING_EDGE(COUNT(10))) THEN
IF (GO/='1') THEN
SD_COUNTER<=0;
ELSIF(SD_COUNTER<603) THEN
SD_COUNTER<=SD_COUNTER+1;
ElSE
SD_COUNTER<=0;

END IF;
END IF;
END PROCESS;

--i2C OPERATION
PROCESS(COUNT(10), RESETN)
BEGIN
IF(RESETN='1') THEN
SCLK<='1';
SDI<='1';
ELSIF(RISING_EDGE(COUNT(10))) THEN
```

```
CASE(SD_COUNTER) IS

--
****************************************X0****************************************
***********
--START
WHEN 0      =>      SDI<='1'; SCLK<='1';
WHEN 1      =>      SDI<='0';
WHEN 2      =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 3      =>      SDI<=address(6);
WHEN 4      =>      SDI<=address(5);---*****************************
WHEN 5      =>      SDI<=address(4);
WHEN 6      =>      SDI<=address(3);
WHEN 7      =>      SDI<=address(2);
WHEN 8      =>      SDI<=address(1);
WHEN 9      =>      SDI<=address(0);
WHEN 10     =>      SDI<='0';
WHEN 11     =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 12     =>      SDI<='0';
WHEN 13     =>      SDI<='1';
WHEN 14     =>      SDI<='0';
WHEN 15     =>      SDI<='0';
```

```
WHEN 16     =>      SDI<='0';
WHEN 17     =>      SDI<='1'; --44
WHEN 18     =>      SDI<='0';
WHEN 19     =>      SDI<='0';
WHEN 20     =>      SDI<='Z';-- FROM Slave ACK
WHEN 21     =>

--START
WHEN 22     =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 23     =>      SDI<='0';
WHEN 24     =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 25     =>      SDI<=address(6);
WHEN 26     =>      SDI<=address(5);---**************************
WHEN 27     =>      SDI<=address(4);
WHEN 28     =>      SDI<=address(3);
WHEN 29     =>      SDI<=address(2);
WHEN 30     =>      SDI<=address(1);
WHEN 31     =>      SDI<=address(0);
WHEN 32     =>      SDI<='1';
WHEN 33     =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 34     =>
WHEN 35     =>      data(7)<=I2C_SDAT;
```

```
WHEN 36     =>      data(6)<=I2C_SDAT;
WHEN 37     =>      data(5)<=I2C_SDAT;
WHEN 38     =>      data(4)<=I2C_SDAT;
WHEN 39     =>      data(3)<=I2C_SDAT;
WHEN 40     =>      data(2)<=I2C_SDAT;
WHEN 41     =>      data(1)<=I2C_SDAT;
WHEN 42     =>      data(0)<=I2C_SDAT;
WHEN 43     => SDI<='0';--TO Slave ACK

--STOP
WHEN 44     =>      SDI<='0'; ---*********************************
WHEN 45     =>      SCLK<='1';
WHEN 46     =>      SDI <= '1';



--
****************************************X1****************************************
******************
--START
WHEN 48     =>      SDI<='1'; SCLK<='1';
WHEN 49     =>      SDI<='0';
WHEN 50     =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 51     =>      SDI<=address(6);
WHEN 52     =>      SDI<=address(5);---*****************************
```

```
WHEN 53     =>      SDI<=address(4);
WHEN 54     =>      SDI<=address(3);
WHEN 55     =>      SDI<=address(2);
WHEN 56     =>      SDI<=address(1);
WHEN 57     =>      SDI<=address(0);
WHEN 58     =>      SDI<='0';
WHEN 59     =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 60     =>      SDI<='0';
WHEN 61     =>      SDI<='1';
WHEN 62     =>      SDI<='0';
WHEN 63     =>      SDI<='0';
WHEN 64     =>      SDI<='0';
WHEN 65     =>      SDI<='0';
WHEN 66     =>      SDI<='1';
WHEN 67     =>      SDI<='1';
WHEN 68     =>      SDI<='Z';-- FROM Slave ACK
WHEN 69     =>

--START
WHEN 70     =>      SDI<='1'; SCLK<='1';----------**********
WHEN 71     =>      SDI<='0';
WHEN 72     =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
```

```
WHEN 73     =>      SDI<=address(6);
WHEN 74     =>      SDI<=address(5);---*****************************
WHEN 75     =>      SDI<=address(4);
WHEN 76     =>      SDI<=address(3);
WHEN 77     =>      SDI<=address(2);
WHEN 78     =>      SDI<=address(1);
WHEN 79     =>      SDI<=address(0);
WHEN 80     =>      SDI<='1';
WHEN 81     =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 82     =>
WHEN 83     =>      data(15)<=I2C_SDAT;
WHEN 84     =>      data(14)<=I2C_SDAT;
WHEN 85     =>      data(13)<=I2C_SDAT;
WHEN 86     =>      data(12)<=I2C_SDAT;
WHEN 87     =>      data(11)<=I2C_SDAT;
WHEN 88     =>      data(10)<=I2C_SDAT;
WHEN 89     =>      data(9)<=I2C_SDAT;
WHEN 90     =>      data(8)<=I2C_SDAT;
WHEN 91     => SDI<='1';--TO Slave ACK

--STOP
WHEN 92     =>      SDI<='0'; ---*********************************
WHEN 93     =>      SCLK<='1';gx<=data;
WHEN 94     =>      SDI <= '1';
```

```
--
****************************************Y0****************************************
****


--START
WHEN 96     =>      SDI<='1'; SCLK<='1';
WHEN 97     =>      SDI<='0';
WHEN 98     =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 99     =>      SDI<=address(6);
WHEN 100    =>      SDI<=address(5);---*****************************
WHEN 101    =>      SDI<=address(4);
WHEN 102    =>      SDI<=address(3);
WHEN 103    =>      SDI<=address(2);
WHEN 104    =>      SDI<=address(1);
WHEN 105    =>      SDI<=address(0);
WHEN 106    =>      SDI<='0';
WHEN 107    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 108    =>      SDI<='0';
WHEN 109    =>      SDI<='1';
WHEN 110    =>      SDI<='0';
WHEN 111    =>      SDI<='0';--46
```

```
WHEN 112    =>      SDI<='0';
WHEN 113    =>      SDI<='1';
WHEN 114    =>      SDI<='1';
WHEN 115    =>      SDI<='0';
WHEN 116    =>      SDI<='Z';-- FROM Slave ACK
WHEN 117    =>

--START
WHEN 118    =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 119    =>      SDI<='0';
WHEN 120    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 121    =>      SDI<=address(6);
WHEN 122    =>      SDI<=address(5);---**************************
WHEN 123    =>      SDI<=address(4);
WHEN 124    =>      SDI<=address(3);
WHEN 125    =>      SDI<=address(2);
WHEN 126    =>      SDI<=address(1);
WHEN 127    =>      SDI<=address(0);
WHEN 128    =>      SDI<='1';
WHEN 129    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 130    =>
WHEN 131    =>      data(7)<=I2C_SDAT;
```

```
WHEN 132    =>      data(6)<=I2C_SDAT;
WHEN 133    =>      data(5)<=I2C_SDAT;
WHEN 134    =>      data(4)<=I2C_SDAT;
WHEN 135    =>      data(3)<=I2C_SDAT;
WHEN 136    =>      data(2)<=I2C_SDAT;
WHEN 137    =>      data(1)<=I2C_SDAT;
WHEN 138    =>      data(0)<=I2C_SDAT;
WHEN 139    => SDI<='1';--TO Slave ACK

--STOP
WHEN 140    =>      SDI<='0'; ---*********************************
WHEN 141    =>      SCLK<='1';
WHEN 142    =>      SDI <= '1';



--
**************************************Y1*****************************************
**

--START
WHEN 144    =>      SDI<='1'; SCLK<='1';
WHEN 145    =>      SDI<='0';
WHEN 146    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 147    =>      SDI<=address(6);
```

```
WHEN 148    =>      SDI<=address(5);---*****************************
WHEN 149    =>      SDI<=address(4);
WHEN 150    =>      SDI<=address(3);
WHEN 151    =>      SDI<=address(2);
WHEN 152    =>      SDI<=address(1);
WHEN 153    =>      SDI<=address(0);
WHEN 154    =>      SDI<='0';
WHEN 155    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 156    =>      SDI<='0';
WHEN 157    =>      SDI<='1';
WHEN 158    =>      SDI<='0';
WHEN 159    =>      SDI<='0';
WHEN 160    =>      SDI<='0';--45
WHEN 161    =>      SDI<='1';
WHEN 162    =>      SDI<='0';
WHEN 163    =>      SDI<='1';
WHEN 164    =>      SDI<='Z';-- FROM Slave ACK
WHEN 165    =>

--START
WHEN 166    =>      SDI<='1'; SCLK<='1';--------------------*****************
WHEN 167    =>      SDI<='0';
WHEN 168    =>      SCLK<='0';
```

```
--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 169    =>      SDI<=address(6);
WHEN 170    =>      SDI<=address(5);---*****************************
WHEN 171    =>      SDI<=address(4);
WHEN 172    =>      SDI<=address(3);
WHEN 173    =>      SDI<=address(2);
WHEN 174    =>      SDI<=address(1);
WHEN 175    =>      SDI<=address(0);
WHEN 176    =>      SDI<='1';
WHEN 177    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 178    =>
WHEN 179    =>      data(15)<=I2C_SDAT;
WHEN 180    =>      data(14)<=I2C_SDAT;
WHEN 181    =>      data(13)<=I2C_SDAT;
WHEN 182    =>      data(12)<=I2C_SDAT;
WHEN 183    =>      data(11)<=I2C_SDAT;
WHEN 184    =>      data(10)<=I2C_SDAT;
WHEN 185    =>      data(9)<=I2C_SDAT;
WHEN 186    =>      data(8)<=I2C_SDAT;
WHEN 187    => SDI<='1';--TO Slave ACK

--STOP
WHEN 188    =>      SDI<='0'; ---*********************************
WHEN 189    =>      SCLK<='1'; gy<=data;
```

```
WHEN 190    =>      SDI <= '1';



--
****************************************Z0*****************************************
***

--START
WHEN 192    =>      SDI<='1'; SCLK<='1';
WHEN 193    =>      SDI<='0';
WHEN 194    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 195    =>      SDI<=address(6);
WHEN 196    =>      SDI<=address(5);---*****************************
WHEN 197    =>      SDI<=address(4);
WHEN 198    =>      SDI<=address(3);
WHEN 199    =>      SDI<=address(2);
WHEN 200    =>      SDI<=address(1);
WHEN 201    =>      SDI<=address(0);
WHEN 202    =>      SDI<='0';
WHEN 203    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 204    =>      SDI<='0';
WHEN 205    =>      SDI<='1';
```

```
WHEN 206    =>      SDI<='0';
WHEN 207    =>      SDI<='0';
WHEN 208    =>      SDI<='1';
WHEN 209    =>      SDI<='0';--48
WHEN 210    =>      SDI<='0';
WHEN 211    =>      SDI<='0';
WHEN 212    =>      SDI<='Z';-- FROM Slave ACK
WHEN 213    =>

--START
WHEN 214    =>      SDI<='1'; SCLK<='1';----------**********
WHEN 215    =>      SDI<='0';
WHEN 216    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 217    =>      SDI<=address(6);
WHEN 218    =>      SDI<=address(5);---**********
WHEN 219    =>      SDI<=address(4);
WHEN 220    =>      SDI<=address(3);
WHEN 221    =>      SDI<=address(2);
WHEN 222    =>      SDI<=address(1);
WHEN 223    =>      SDI<=address(0);
WHEN 224    =>      SDI<='1';
WHEN 225    =>      SDI<='Z';--FROM Slave ACK

--DATA
```

```
WHEN 226    =>
WHEN 227    =>      data(7)<=I2C_SDAT;
WHEN 228    =>      data(6)<=I2C_SDAT;
WHEN 229    =>      data(5)<=I2C_SDAT;
WHEN 230    =>      data(4)<=I2C_SDAT;
WHEN 231    =>      data(3)<=I2C_SDAT;
WHEN 232    =>      data(2)<=I2C_SDAT;
WHEN 233    =>      data(1)<=I2C_SDAT;
WHEN 234    =>      data(0)<=I2C_SDAT;
WHEN 235    => SDI<='1';--TO Slave ACK

--STOP
WHEN 236    =>      SDI<='0'; ---**********************************
WHEN 237    =>      SCLK<='1';
WHEN 238    =>      SDI <= '1';



--
****************************************Z1*****************************************
***

--START
WHEN 239    =>      SDI<='1'; SCLK<='1';
WHEN 240    =>      SDI<='0';
WHEN 241    =>      SCLK<='0';
```

```
--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 242    =>      SDI<=address(6);
WHEN 243    =>      SDI<=address(5);---*****************************
WHEN 244    =>      SDI<=address(4);
WHEN 245    =>      SDI<=address(3);
WHEN 246    =>      SDI<=address(2);
WHEN 247    =>      SDI<=address(1);
WHEN 248    =>      SDI<=address(0);
WHEN 249    =>      SDI<='0';
WHEN 250    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 251    =>      SDI<='0';
WHEN 252    =>      SDI<='1';
WHEN 253    =>      SDI<='0';
WHEN 254    =>      SDI<='0';
WHEN 255    =>      SDI<='0';
WHEN 256    =>      SDI<='1';--47
WHEN 257    =>      SDI<='1';
WHEN 258    =>      SDI<='1';
WHEN 259    =>      SDI<='Z';-- FROM Slave ACK
WHEN 260    =>

--START
WHEN 261    =>      SDI<='1'; SCLK<='1';----------*****************
WHEN 262    =>      SDI<='0';
```

```
WHEN 263    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 264    =>      SDI<=address(6);
WHEN 265    =>      SDI<=address(5);---*******************************
WHEN 266    =>      SDI<=address(4);
WHEN 267    =>      SDI<=address(3);
WHEN 268    =>      SDI<=address(2);
WHEN 269    =>      SDI<=address(1);
WHEN 270    =>      SDI<=address(0);
WHEN 271    =>      SDI<='1';
WHEN 272    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 273    =>
WHEN 274    =>      data(15)<=I2C_SDAT;
WHEN 275    =>      data(14)<=I2C_SDAT;
WHEN 276    =>      data(13)<=I2C_SDAT;
WHEN 277    =>      data(12)<=I2C_SDAT;
WHEN 278    =>      data(11)<=I2C_SDAT;
WHEN 279    =>      data(10)<=I2C_SDAT;
WHEN 280    =>      data(9)<=I2C_SDAT;
WHEN 281    =>      data(8)<=I2C_SDAT;
WHEN 282    => SDI<='1';--TO Slave ACK

--STOP
```

```
WHEN 283    =>      SDI<='0'; ---*********************************
WHEN 284    =>      SCLK<='1'; gz<=data;
WHEN 285    =>      SDI <= '1';
-------------------------------------------------------
--
**************************************ax1*****************************************
****


--START
WHEN 286    =>      SDI<='1'; SCLK<='1';
WHEN 287    =>      SDI<='0';
WHEN 288    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 289    =>      SDI<=address(6);
WHEN 290    =>      SDI<=address(5);---*******************************
WHEN 291    =>      SDI<=address(4);
WHEN 292    =>      SDI<=address(3);
WHEN 293    =>      SDI<=address(2);
WHEN 294    =>      SDI<=address(1);
WHEN 295    =>      SDI<=address(0);
WHEN 296    =>      SDI<='0';
WHEN 297    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 298    =>      SDI<='0';
```

```
WHEN 299    =>      SDI<='0';
WHEN 300    =>      SDI<='1';
WHEN 301    =>      SDI<='1';
WHEN 302    =>      SDI<='1';
WHEN 303    =>      SDI<='0';
WHEN 304    =>      SDI<='1';
WHEN 305    =>      SDI<='1';
WHEN 306    =>      SDI<='Z';-- FROM Slave ACK
WHEN 307    =>

--START
WHEN 308    =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 309    =>      SDI<='0';
WHEN 310    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 311    =>      SDI<=address(6);
WHEN 312    =>      SDI<=address(5);---******************************
WHEN 313    =>      SDI<=address(4);
WHEN 314    =>      SDI<=address(3);
WHEN 315    =>      SDI<=address(2);
WHEN 316    =>      SDI<=address(1);
WHEN 317    =>      SDI<=address(0);
WHEN 318    =>      SDI<='1';
WHEN 319    =>      SDI<='Z';--FROM Slave ACK
```

```
--DATA
WHEN 320    =>
WHEN 321    =>      data(15)<=I2C_SDAT;
WHEN 322    =>      data(14)<=I2C_SDAT;
WHEN 323    =>      data(13)<=I2C_SDAT;
WHEN 324    =>      data(12)<=I2C_SDAT;
WHEN 325    =>      data(11)<=I2C_SDAT;
WHEN 326    =>      data(10)<=I2C_SDAT;
WHEN 327    =>      data(9)<=I2C_SDAT;
WHEN 328    =>      data(8)<=I2C_SDAT;
WHEN 329    => SDI<='1';--TO Slave ACK

--STOP
WHEN 330    =>      SDI<='0'; ---**********************************
WHEN 331    =>      SCLK<='1';
WHEN 334    =>      SDI <= '1';
-------------------------------------------------------
-------------------------------------------------------
--
****************************************ax0*********************************************
****


--START
WHEN 335    =>      SDI<='1'; SCLK<='1';
WHEN 336    =>      SDI<='0';
WHEN 337    =>      SCLK<='0';
```

```
--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 338    =>      SDI<=address(6);
WHEN 339    =>      SDI<=address(5);---*******************************
WHEN 340    =>      SDI<=address(4);
WHEN 341    =>      SDI<=address(3);
WHEN 342    =>      SDI<=address(2);
WHEN 343    =>      SDI<=address(1);
WHEN 344    =>      SDI<=address(0);
WHEN 345    =>      SDI<='0';
WHEN 346    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 347    =>      SDI<='0';
WHEN 348    =>      SDI<='0';
WHEN 349    =>      SDI<='1';
WHEN 350    =>      SDI<='1';
WHEN 351    =>      SDI<='1';
WHEN 352    =>      SDI<='1';
WHEN 353    =>      SDI<='0';
WHEN 354    =>      SDI<='0';
WHEN 355    =>      SDI<='Z';-- FROM Slave ACK
WHEN 356    =>

--START
WHEN 357    =>      SDI<='1'; SCLK<='1';--------------------**********
```

```
WHEN 358    =>      SDI<='0';
WHEN 359    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 360    =>      SDI<=address(6);
WHEN 361    =>      SDI<=address(5);---*****************************
WHEN 362    =>      SDI<=address(4);
WHEN 363    =>      SDI<=address(3);
WHEN 364    =>      SDI<=address(2);
WHEN 365    =>      SDI<=address(1);
WHEN 366    =>      SDI<=address(0);
WHEN 367    =>      SDI<='1';
WHEN 368    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 369    =>
WHEN 370    =>      data(7)<=I2C_SDAT;
WHEN 371    =>      data(6)<=I2C_SDAT;
WHEN 372    =>      data(5)<=I2C_SDAT;
WHEN 373    =>      data(4)<=I2C_SDAT;
WHEN 374    =>      data(3)<=I2C_SDAT;
WHEN 375    =>      data(2)<=I2C_SDAT;
WHEN 376    =>      data(1)<=I2C_SDAT;
WHEN 377    =>      data(0)<=I2C_SDAT;
WHEN 378    => SDI<='1';--TO Slave ACK
```

```
--STOP
WHEN 379    =>      SDI<='0'; ---*********************************
WHEN 380    =>      SCLK<='1'; ax<=data;
WHEN 381    =>      SDI <= '1';
-------------------------------------------------------
-------------------------------------------------------
--
**************************************ay1*****************************************
****


--START
WHEN 382    =>      SDI<='1'; SCLK<='1';
WHEN 383    =>      SDI<='0';
WHEN 384    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 385    =>      SDI<=address(6);
WHEN 386    =>      SDI<=address(5);---*******************************
WHEN 387    =>      SDI<=address(4);
WHEN 388    =>      SDI<=address(3);
WHEN 389    =>      SDI<=address(2);
WHEN 390    =>      SDI<=address(1);
WHEN 391    =>      SDI<=address(0);
WHEN 392    =>      SDI<='0';
WHEN 393    =>      SDI<='Z';-- FROM Slave ACK
```

```
--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 394    =>      SDI<='0';
WHEN 395    =>      SDI<='0';
WHEN 396    =>      SDI<='1';
WHEN 397    =>      SDI<='1';
WHEN 398    =>      SDI<='1';
WHEN 399    =>      SDI<='1';
WHEN 400    =>      SDI<='0';
WHEN 401    =>      SDI<='1';
WHEN 402    =>      SDI<='Z';-- FROM Slave ACK
WHEN 403    =>

--START
WHEN 404    =>      SDI<='1'; SCLK<='1';----------**********
WHEN 405    =>      SDI<='0';
WHEN 406    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 407    =>      SDI<=address(6);
WHEN 408    =>      SDI<=address(5);---**********
WHEN 409    =>      SDI<=address(4);
WHEN 410    =>      SDI<=address(3);
WHEN 411    =>      SDI<=address(2);
WHEN 412    =>      SDI<=address(1);
WHEN 413    =>      SDI<=address(0);
WHEN 414    =>      SDI<='1';
```

```
WHEN 415    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 416    =>
WHEN 417    =>      data(15)<=I2C_SDAT;
WHEN 418    =>      data(14)<=I2C_SDAT;
WHEN 419    =>      data(13)<=I2C_SDAT;
WHEN 420    =>      data(12)<=I2C_SDAT;
WHEN 421    =>      data(11)<=I2C_SDAT;
WHEN 422    =>      data(10)<=I2C_SDAT;
WHEN 423    =>      data(9)<=I2C_SDAT;
WHEN 424    =>      data(8)<=I2C_SDAT;
WHEN 425    => SDI<='1';--TO Slave ACK

--STOP
WHEN 426    =>      SDI<='0'; ---**********************************
WHEN 427    =>      SCLK<='1';
WHEN 428    =>      SDI <= '1';
-------------------------------------------------------
-------------------------------------------------------
--
****************************************ay0*****************************************
****

--START
WHEN 429    =>      SDI<='1'; SCLK<='1';
```

```
WHEN 430    =>      SDI<='0';
WHEN 431    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 432    =>      SDI<=address(6);
WHEN 433    =>      SDI<=address(5);---*****************************
WHEN 434    =>      SDI<=address(4);
WHEN 435    =>      SDI<=address(3);
WHEN 436    =>      SDI<=address(2);
WHEN 437    =>      SDI<=address(1);
WHEN 438    =>      SDI<=address(0);
WHEN 439    =>      SDI<='0';
WHEN 440    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 441    =>      SDI<='0';
WHEN 442    =>      SDI<='0';
WHEN 443    =>      SDI<='1';
WHEN 444    =>      SDI<='1';
WHEN 445    =>      SDI<='1';
WHEN 446    =>      SDI<='1';
WHEN 447    =>      SDI<='1';
WHEN 448    =>      SDI<='0';
WHEN 449    =>      SDI<='Z';-- FROM Slave ACK
WHEN 450    =>
```

```
--START
WHEN 451    =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 452    =>      SDI<='0';
WHEN 453    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 454    =>      SDI<=address(6);
WHEN 455    =>      SDI<=address(5);---******************************
WHEN 456    =>      SDI<=address(4);
WHEN 457    =>      SDI<=address(3);
WHEN 458    =>      SDI<=address(2);
WHEN 459    =>      SDI<=address(1);
WHEN 460    =>      SDI<=address(0);
WHEN 461    =>      SDI<='1';
WHEN 462    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 463    =>
WHEN 464    =>      data(7)<=I2C_SDAT;
WHEN 465    =>      data(6)<=I2C_SDAT;
WHEN 466    =>      data(5)<=I2C_SDAT;
WHEN 467    =>      data(4)<=I2C_SDAT;
WHEN 468    =>      data(3)<=I2C_SDAT;
WHEN 469    =>      data(2)<=I2C_SDAT;
WHEN 470    =>      data(1)<=I2C_SDAT;
WHEN 471    =>      data(0)<=I2C_SDAT;
```

```
WHEN 472    => SDI<='1';--TO Slave ACK

--STOP
WHEN 473    =>      SDI<='0'; ---**********************************
WHEN 474    =>      SCLK<='1'; ay<=data;
WHEN 475    =>      SDI <= '1';
-------------------------------------------------------


-------------------------------------------------------
--
****************************************az1*****************************************
****

--START
WHEN 476    =>      SDI<='1'; SCLK<='1';
WHEN 477    =>      SDI<='0';
WHEN 478    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 479    =>      SDI<=address(6);
WHEN 480    =>      SDI<=address(5);---******************************
WHEN 481    =>      SDI<=address(4);
WHEN 482    =>      SDI<=address(3);
WHEN 483    =>      SDI<=address(2);
WHEN 484    =>      SDI<=address(1);
WHEN 485    =>      SDI<=address(0);
```

```
WHEN 486    =>      SDI<='0';
WHEN 487    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 488    =>      SDI<='0';
WHEN 489    =>      SDI<='0';
WHEN 490    =>      SDI<='1';
WHEN 491    =>      SDI<='1';
WHEN 492    =>      SDI<='1';
WHEN 493    =>      SDI<='1';
WHEN 494    =>      SDI<='1';
WHEN 495    =>      SDI<='1';
WHEN 496    =>      SDI<='Z';-- FROM Slave ACK
WHEN 497    =>

--START
WHEN 498    =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 499    =>      SDI<='0';
WHEN 500    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 501    =>      SDI<=address(6);
WHEN 502    =>      SDI<=address(5);---******************************
WHEN 503    =>      SDI<=address(4);
WHEN 504    =>      SDI<=address(3);
WHEN 505    =>      SDI<=address(2);
```

```
WHEN 506    =>      SDI<=address(1);
WHEN 507    =>      SDI<=address(0);
WHEN 508    =>      SDI<='1';
WHEN 509    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 510    =>
WHEN 511    =>      data(15)<=I2C_SDAT;
WHEN 512    =>      data(14)<=I2C_SDAT;
WHEN 513    =>      data(13)<=I2C_SDAT;
WHEN 514    =>      data(12)<=I2C_SDAT;
WHEN 515    =>      data(11)<=I2C_SDAT;
WHEN 516    =>      data(10)<=I2C_SDAT;
WHEN 517    =>      data(9)<=I2C_SDAT;
WHEN 518    =>      data(8)<=I2C_SDAT;
WHEN 519    => SDI<='1';--TO Slave ACK

--STOP
WHEN 520    =>      SDI<='0'; ---**********************************
WHEN 521    =>      SCLK<='1';
WHEN 522    =>      SDI <= '1';
-------------------------------------------------------


-------------------------------------------------------
```

```
--
****************************************az0****************************************
****


--START
WHEN 523    =>      SDI<='1'; SCLK<='1';
WHEN 524    =>      SDI<='0';
WHEN 525    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - ESCRITURA
WHEN 526    =>      SDI<=address(6);
WHEN 527    =>      SDI<=address(5);---*****************************
WHEN 528    =>      SDI<=address(4);
WHEN 529    =>      SDI<=address(3);
WHEN 530    =>      SDI<=address(2);
WHEN 531    =>      SDI<=address(1);
WHEN 532    =>      SDI<=address(0);
WHEN 533    =>      SDI<='0';
WHEN 534    =>      SDI<='Z';-- FROM Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A LEER)
WHEN 535    =>      SDI<='0';
WHEN 536    =>      SDI<='1';
WHEN 537    =>      SDI<='0';
WHEN 538    =>      SDI<='0';
WHEN 539    =>      SDI<='0';
```

```
WHEN 540    =>      SDI<='0';
WHEN 541    =>      SDI<='0';
WHEN 542    =>      SDI<='0';
WHEN 543    =>      SDI<='Z';-- FROM Slave ACK
WHEN 544    =>

--START
WHEN 545    =>      SDI<='1'; SCLK<='1';--------------------**********
WHEN 546    =>      SDI<='0';
WHEN 547    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO - LECTURA
WHEN 548    =>      SDI<=address(6);
WHEN 549    =>      SDI<=address(5);---******************************
WHEN 550    =>      SDI<=address(4);
WHEN 551    =>      SDI<=address(3);
WHEN 552    =>      SDI<=address(2);
WHEN 553    =>      SDI<=address(1);
WHEN 554    =>      SDI<=address(0);
WHEN 555    =>      SDI<='1';
WHEN 556    =>      SDI<='Z';--FROM Slave ACK

--DATA
WHEN 557    =>
WHEN 558    =>      data(7)<=I2C_SDAT;
WHEN 559    =>      data(6)<=I2C_SDAT;
```

```
WHEN 560    =>      data(5)<=I2C_SDAT;
WHEN 561    =>      data(4)<=I2C_SDAT;
WHEN 562    =>      data(3)<=I2C_SDAT;
WHEN 563    =>      data(2)<=I2C_SDAT;
WHEN 564    =>      data(1)<=I2C_SDAT;
WHEN 565    =>      data(0)<=I2C_SDAT;
WHEN 566    => SDI<='1';--TO Slave ACK

--STOP
WHEN 567    =>      SDI<='0'; ---*********************************
WHEN 568    =>      SCLK<='1'; az<=data;
WHEN 569    =>      SDI <= '1';
-----------------------------------------------------
--*********************************REGISTRO
(0X6b)************************************************************
--START
WHEN 570    =>      SDI<='1'; SCLK<='1';
WHEN 571    =>      SDI<='0';
WHEN 572    =>      SCLK<='0';

--DIRECCIÓN DEL ESCLAVO  ESCRITURA
WHEN 573    =>      SDI<=address(6);
WHEN 574    =>      SDI<=address(5);---*****************************
WHEN 575    =>      SDI<=address(4);
WHEN 576    =>      SDI<=address(3);
WHEN 577    =>      SDI<=address(2);
```

```
WHEN 578    =>      SDI<=address(1);
WHEN 579    =>      SDI<=address(0);
WHEN 580    =>      SDI<='0';
WHEN 581    =>      SDI<='Z';--Slave ACK

--DIRECCIÓN DEL REGISTRO EN EL ESCLAVO (DIRECCIÓN DONDE VOY A ESCRIBIR)
WHEN 582    =>      SDI<='0';
WHEN 583    =>      SDI<='1';
WHEN 584    =>      SDI<='1';
WHEN 585    =>      SDI<='0';
WHEN 586    =>      SDI<='1';
WHEN 587    =>      SDI<='0';
WHEN 588    =>      SDI<='1';
WHEN 589    =>      SDI<='1';--(0X6b)
WHEN 590    =>      SDI<='Z';--Slave ACK

--DATA
WHEN 591    =>      SDI<='0';
WHEN 592    =>      SDI<='0';
WHEN 593    =>      SDI<='0';
WHEN 594    =>      SDI<='0';
WHEN 595    =>      SDI<='0';
WHEN 596    =>      SDI<='0';
WHEN 597    =>      SDI<='0';
WHEN 598    =>      SDI<='0';
WHEN 599    =>      SDI<='Z';--Slave ACK
```

```
--STOP
WHEN 600     =>                SDI<='0';--**************************************
WHEN 601     =>         SCLK<='1';
WHEN 602     =>        SDI <= '1';
-------------------------------------------------------------------


WHEN OTHERS =>    SDI<='1'; SCLK<='1';


END CASE;
END IF;
END PROCESS;
I2C_SCLK<= NOT(COUNT(10)) WHEN ( ((SD_COUNTER >= 4) AND (SD_COUNTER <= 22))
OR ((SD_COUNTER >= 26) AND (SD_COUNTER <= 44))
OR ((SD_COUNTER >= 52) AND (SD_COUNTER <= 70)) OR ((SD_COUNTER >= 74) AND
(SD_COUNTER <= 92))
OR ((SD_COUNTER >= 100) AND (SD_COUNTER <= 118)) OR ((SD_COUNTER >= 122) AND
(SD_COUNTER <= 140))
OR ((SD_COUNTER >= 148) AND (SD_COUNTER <= 166)) OR ((SD_COUNTER >= 170) AND
(SD_COUNTER <= 188))
OR ((SD_COUNTER >= 196) AND (SD_COUNTER <= 214)) OR ((SD_COUNTER >= 218) AND
(SD_COUNTER <= 236))
OR ((SD_COUNTER >= 243) AND (SD_COUNTER <= 261)) OR ((SD_COUNTER >= 265) AND
(SD_COUNTER <= 283))
OR ((SD_COUNTER >= 290) AND (SD_COUNTER <= 308)) OR ((SD_COUNTER >= 312) AND
(SD_COUNTER <= 330))
OR ((SD_COUNTER >= 339) AND (SD_COUNTER <= 357)) OR ((SD_COUNTER >= 361) AND
(SD_COUNTER <= 379))
```

```
OR ((SD_COUNTER >= 386) AND (SD_COUNTER <= 404)) OR ((SD_COUNTER >= 408) AND
(SD_COUNTER <= 426))

OR ((SD_COUNTER >= 433) AND (SD_COUNTER <= 451)) OR ((SD_COUNTER >= 455) AND
(SD_COUNTER <= 473))

OR ((SD_COUNTER >= 480) AND (SD_COUNTER <= 498)) OR ((SD_COUNTER >= 502) AND
(SD_COUNTER <= 520))

OR ((SD_COUNTER >= 527) AND (SD_COUNTER <= 545)) OR ((SD_COUNTER >= 549) AND
(SD_COUNTER <= 567))

OR ((SD_COUNTER >= 574) AND (SD_COUNTER <= 600))

) ELSE SCLK;

I2C_SDAT<= SDI;

END SOLUCION;
```