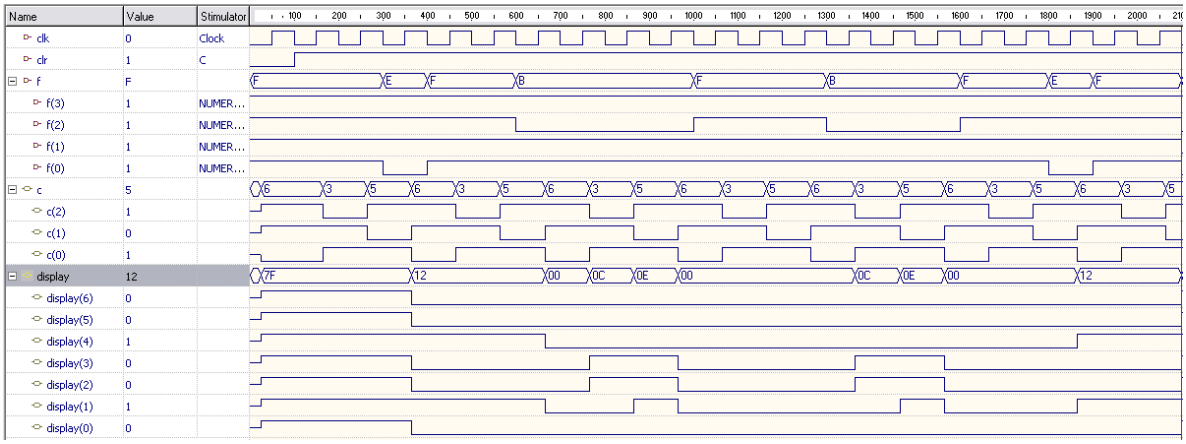
Córdova Pichardo Francisco Uziel

Practica 12

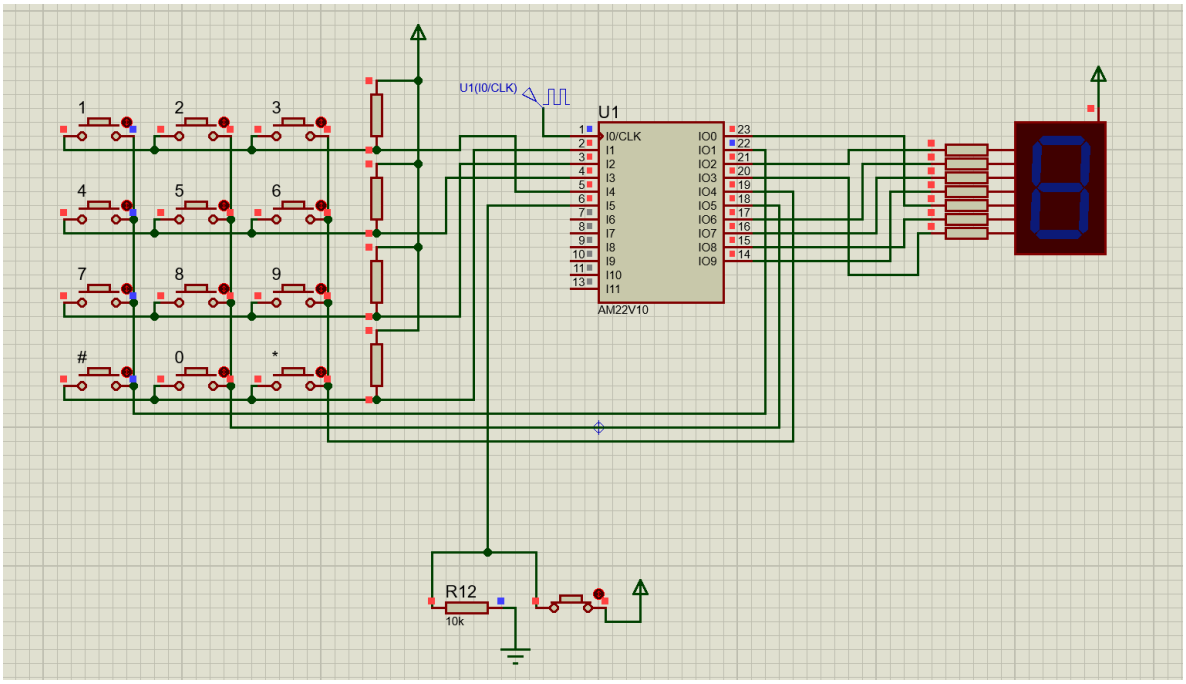
```
 1 library ieee;
 2 use ieee.std_logic_1164.all;
 3
 4 entity teclado is port(
 5     F: in std_logic_vector(3 downto 0);
 6     CLK, CLR: in std_logic;
 7     c: inout std_logic_vector(2 downto 0);
 8     display: inout std_logic_vector(6 downto 0));
 9 end teclado;
10
11 architecture a_teclado of teclado is
12 constant dig0: std_logic_vector(6 downto 0):= "0000001";
13 constant dig1: std_logic_vector(6 downto 0):= "1001111";
14 constant dig2: std_logic_vector(6 downto 0):= "0010010";
15 constant dig3: std_logic_vector(6 downto 0):= "0000110";
16 constant dig4: std_logic_vector(6 downto 0):= "1001100";
17 constant dig5: std_logic_vector(6 downto 0):= "0100100";
18 constant dig6: std_logic_vector(6 downto 0):= "0100000";
19 constant dig7: std_logic_vector(6 downto 0):= "0001110";
20 constant dig8: std_logic_vector(6 downto 0):= "0000000";
21 constant dig9: std_logic_vector(6 downto 0):= "0001100";
22 constant g: std_logic_vector(6 downto 0) := "0000100";
23 constant digA: std_logic_vector(6 downto 0):= "0001000";
24 constant NP: std_logic_vector(6 downto 0):= "1111111";
25
26
27 constant F0: std_logic_vector(3 downto 0):="1101";
28 constant F1: std_logic_vector(3 downto 0):="1011";
29 constant F2: std_logic_vector(3 downto 0):="0111";
30 constant F3: std_logic_vector(3 downto 0):="1111";
31 constant F4: std_logic_vector(3 downto 0):="1110";
32
33
34 constant C0: std_logic_vector(2 downto 0):="110";
35 constant C1: std_logic_vector(2 downto 0):="101";
36 constant C2: std_logic_vector(2 downto 0):="011";
37
```

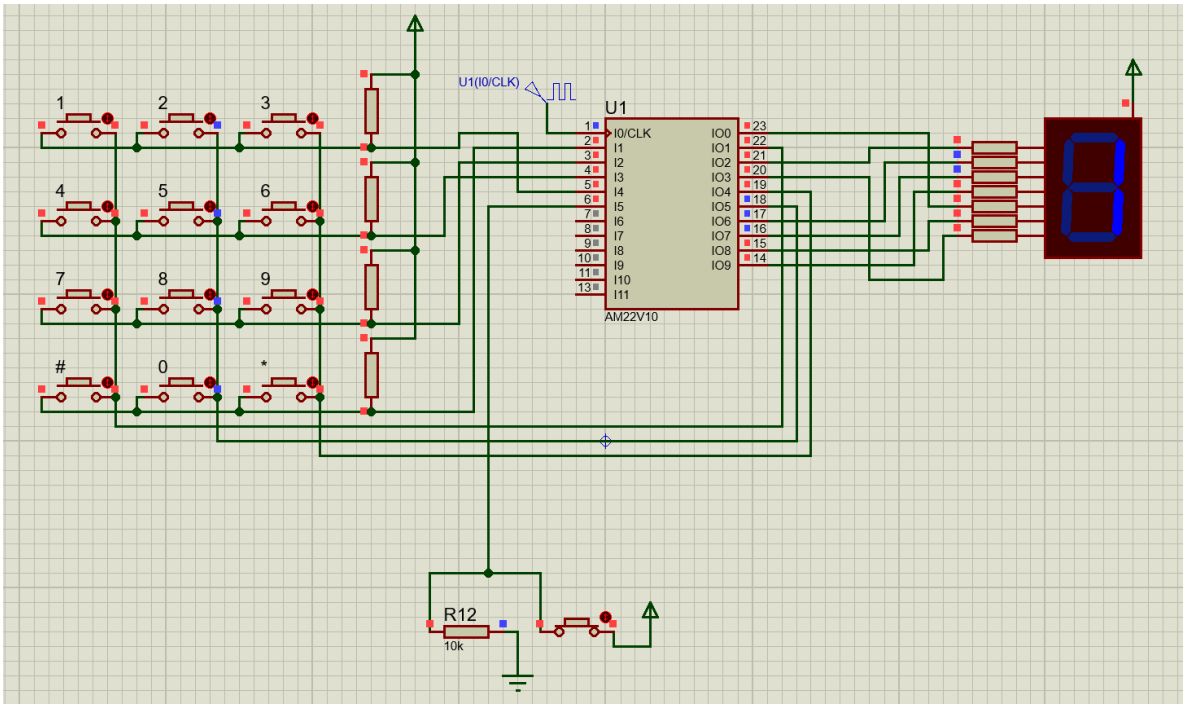
```
signal Aux: std_logic_vector(6 downto 0);
begin

    process(F, C) begin
        CASE F & C is
            WHEN (F0)&(C0) => Aux <= dig6;
            WHEN (F0)&(C1) => Aux <= dig5;
            WHEN (F0)&(C2) => Aux <= dig4;
            WHEN (F1)&(C0) => Aux <= dig9;
            WHEN (F1)&(C1) => Aux <= dig8;
            WHEN (F1)&(C2) => Aux <= dig7;
            WHEN (F2)&(C0) => Aux <= diga;
            WHEN (F2)&(C1) => Aux <= dig0;
            WHEN (F2)&(C2) => Aux <= g;
            WHEN (F4)&(C0) => Aux <= dig3;
            WHEN (F4)&(C1) => Aux <= dig2;
            WHEN (F4)&(C2) => Aux <= dig1;

            WHEN OTHERS => Aux <= NP;
        END CASE;
    END process DECO;

        process(CLK, CLR) begin
            if(CLR = '0') THEN
                C <= "110";
            elsif(CLK'EVENT AND CLK = '1') THEN
                C <= std_logic_vector(BIT_VECTOR(C) ROR 1);
            end if;
        END process ANILLO;

        process(CLK, CLR, F) begin
            if(CLR = '0') THEN
                display <= NP;
            elsif(CLK'EVENT AND CLK = '1') THEN
                if(F = F3) THEN
                    display <= display;
                else
                    display <= Aux;
                end if;
             end if;
        end process REG;
end a_teclado;
```

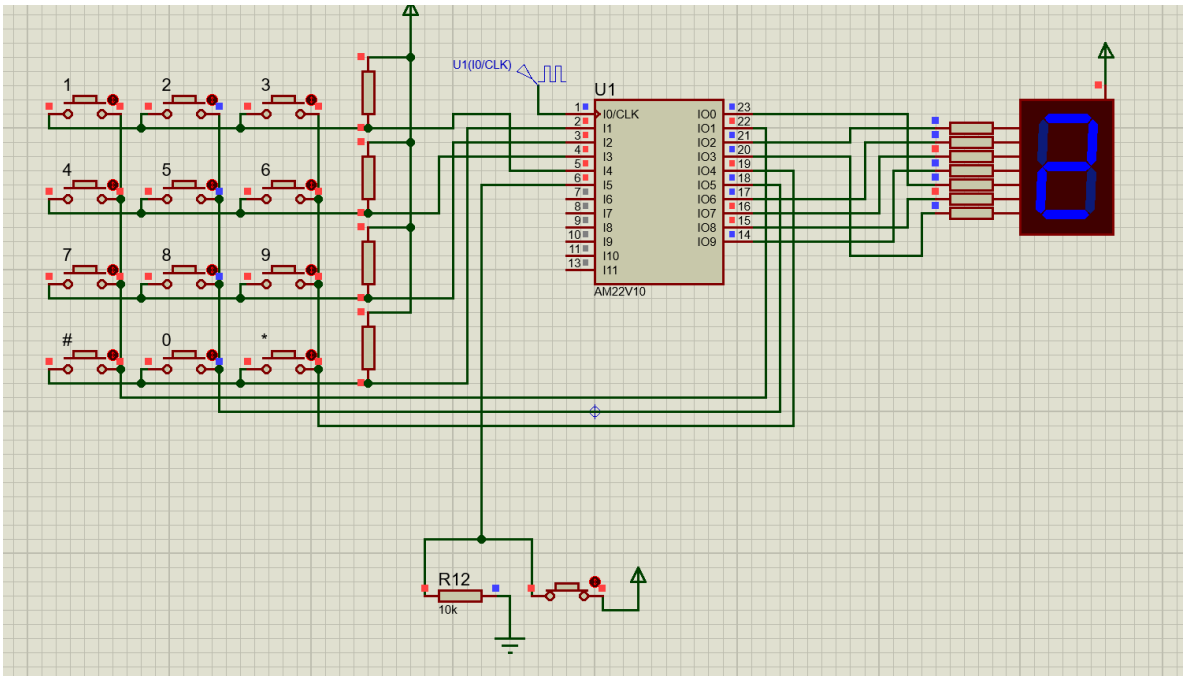


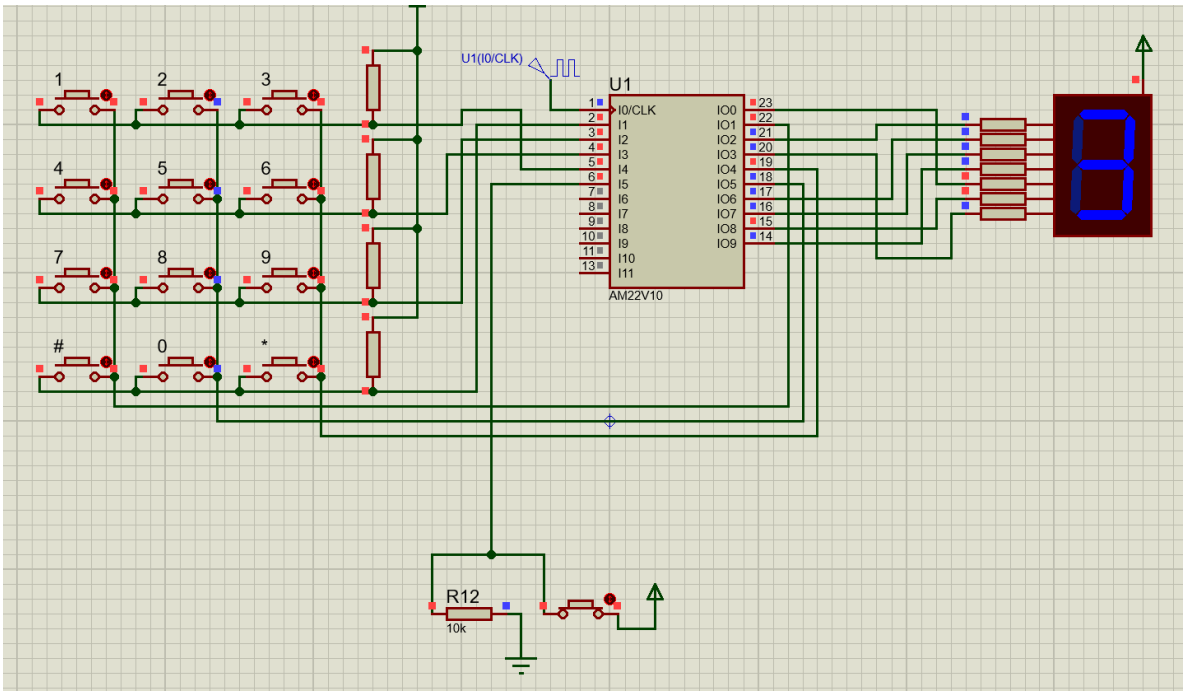
C22V10

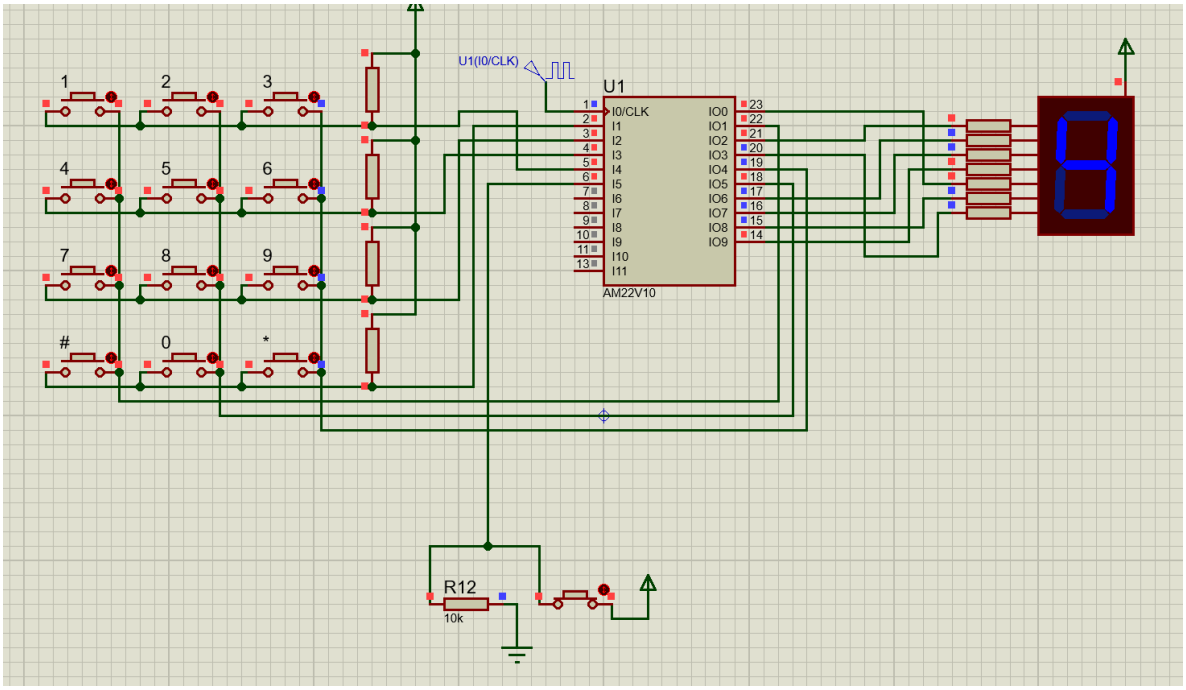
clk =	1	24	* not used
f(3) =	2	23	= display(2)
f(2) =	3	22	= c(2)
f(1) =	4	21	= display(6)
f(0) =	5	20	= display(0)
clr =	6	19	= c(0)
not used *	7	18	= c(1)
not used *	8	17	= display(5)
not used *	9	16	= display(4)
not used *	10	15	= display(1)
not used *	11	14	= display(3)
not used *	12	13	* not used

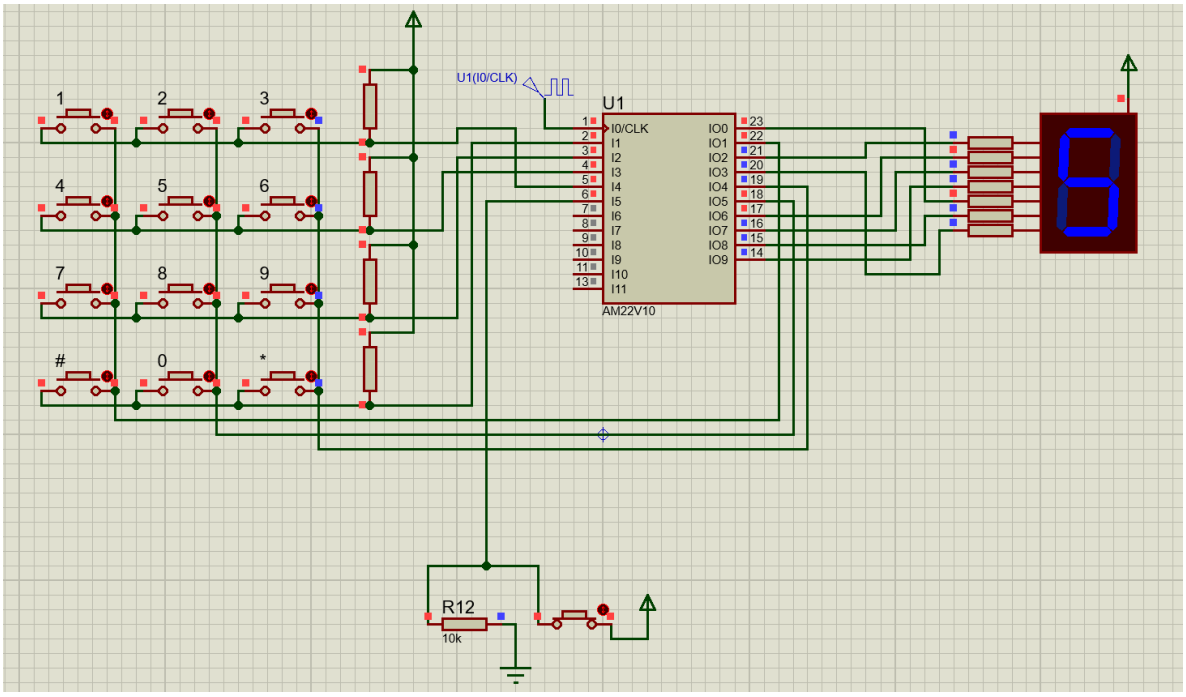


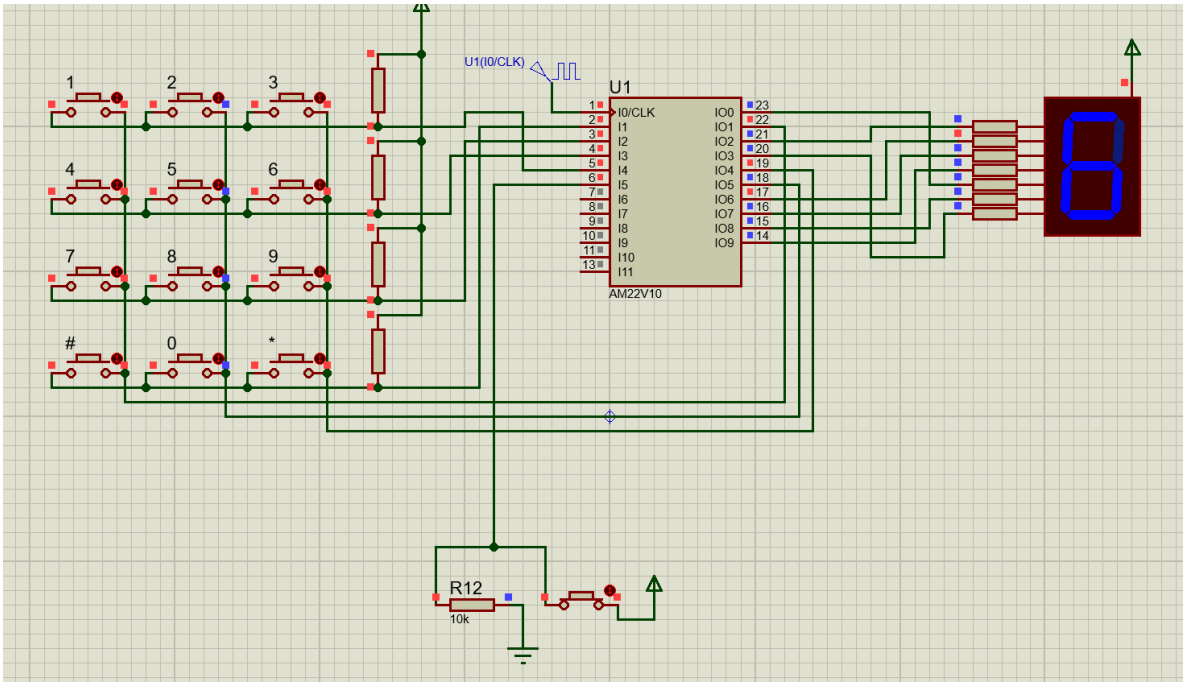

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

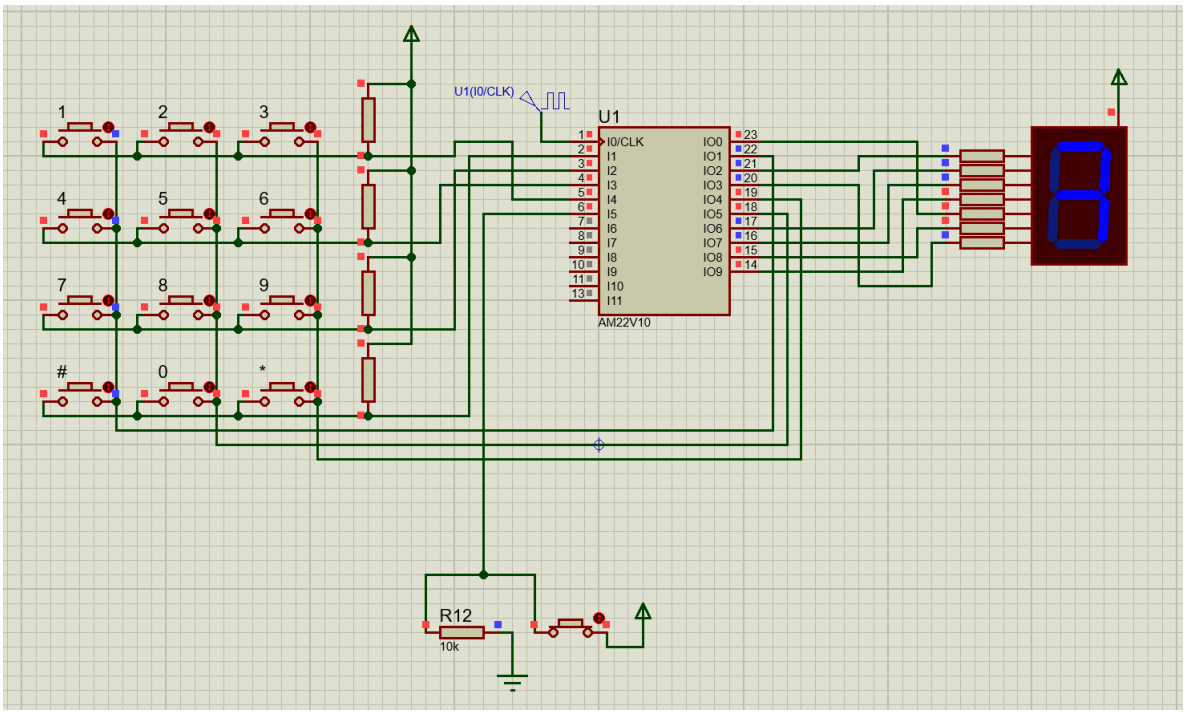

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

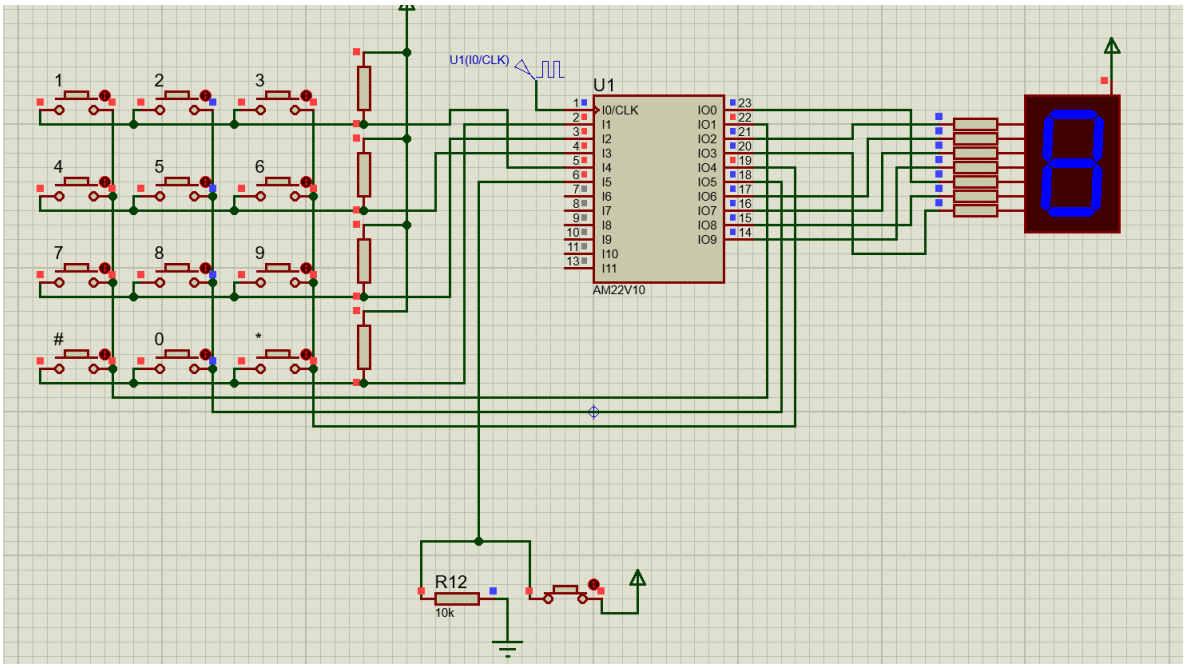

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

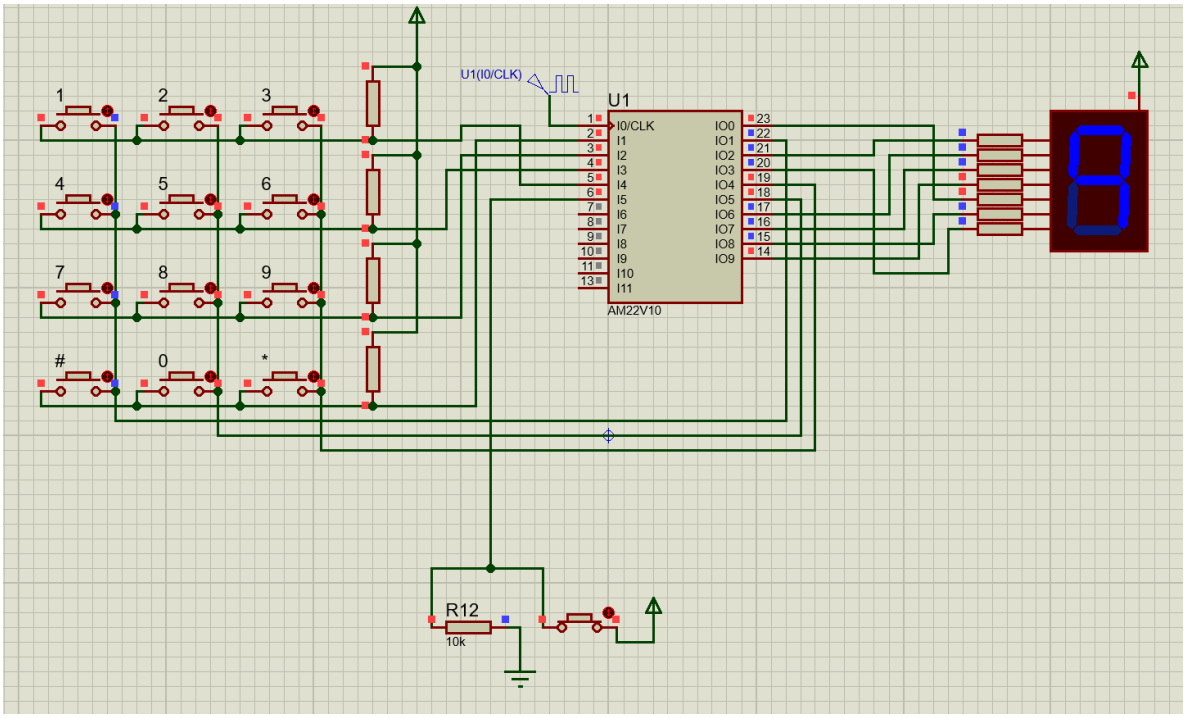

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

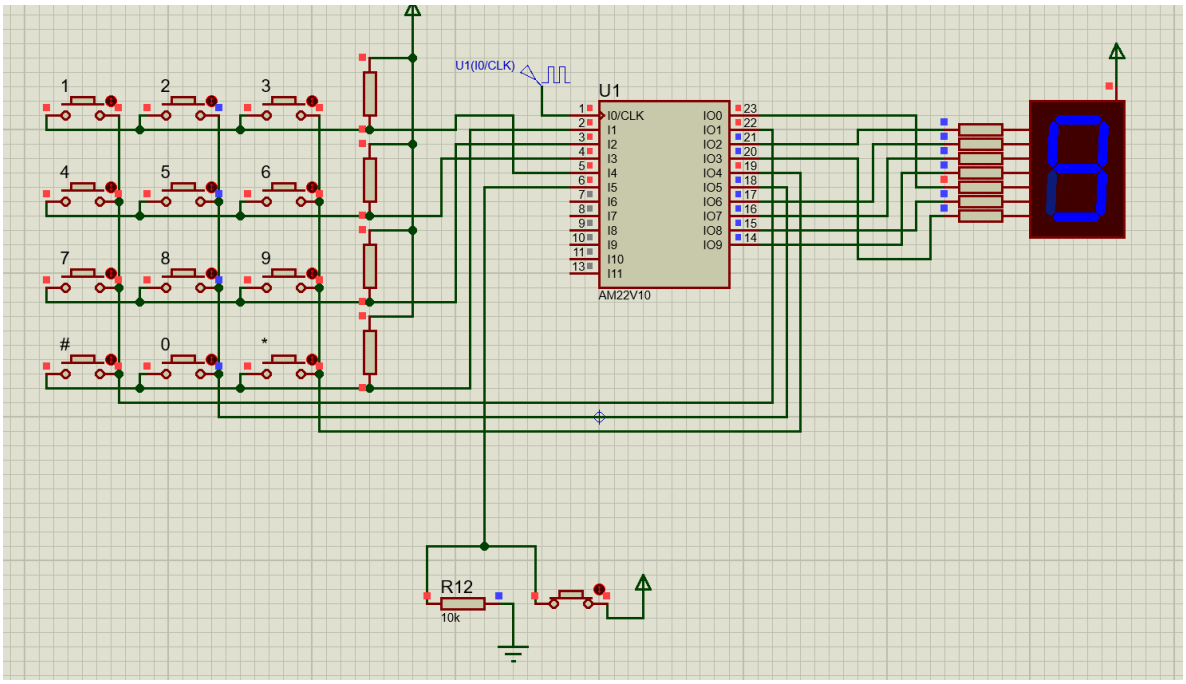

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

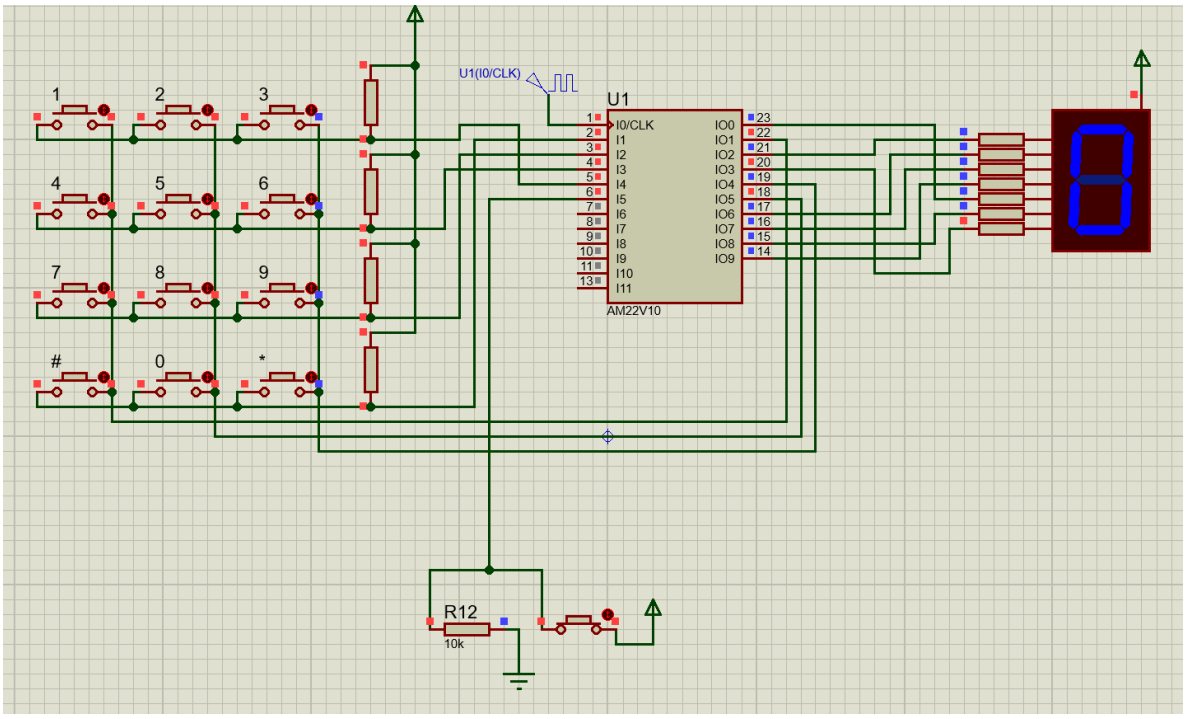

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k

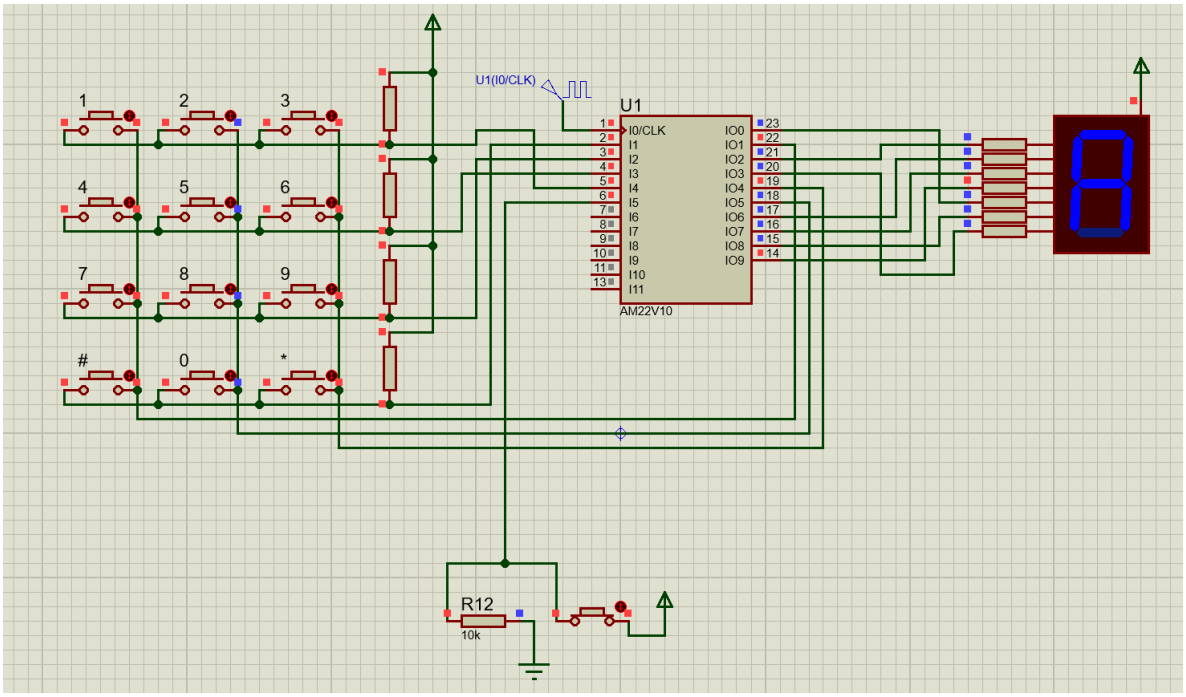

U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k


U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
1
2
3
4
5
6
7
8
9
#
0
*
R12
10k


1
2
3
4
5
6
7
8
9
#
0
*
U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
R12
10k


1
2
3
4
5
6
7
8
9
#
0
*
U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
R12
10k


1
2
3
4
5
6
7
8
9
#
0
*
U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
R12
10k


1
2
3
4
5
6
7
8
9
#
0
*
U1(I0/CLK)
U1
I0/CLK
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
AM22V10
R12
10k