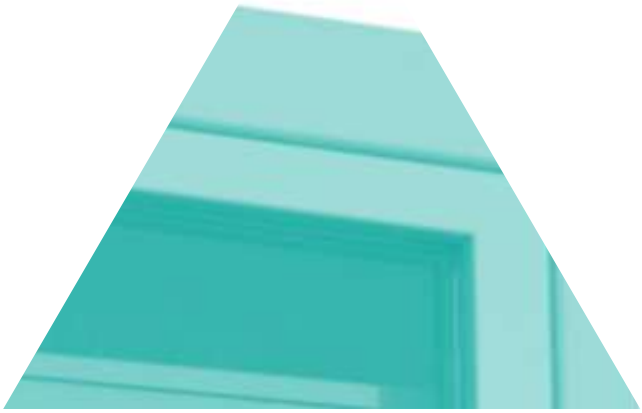


VHDL/Verilog

MAQUINA EXPENDEDORA DE BOTELLAS DE GEL ANTIBACTERIAL

Electrónica Digital: Circuitos Lógicos
Lenguaje de Descripción de Hardware (HDL): VHDL Y Verilog
Xilinx (64-Bit Project Navigator) & Adept

25/06/2020

Alva Ramírez Alan
Cervantes Rodríguez Diego
Martínez Durán Carlos Daniel

RESUMEN

En el presente proyecto se constituye el desarrollo de una máquina expendedora de botellas de gel antibacterial. La cual se desarrolla mediante el uso de una PFGA NEXYS 2, esta cuenta con las funciones descritas a continuación:

- Control de encendido y apagado (on/off) de toda la máquina.
- Selección del número de artículos.
- Sensores que indican si la máquina expendedora tiene artículos disponibles.
- Iluminación o anuncio auditivo de articulo disponible.
- Mensaje de bienvenida y despedida en display de 7 segmentos.
- Indicador de precio.
- Mecanismo de despacho de artículos.
- Alimentación de línea eléctrica.
- Mecanismo para la devolución de cambio.

ÍNDICE

INTRODUCCIÓN

El coronavirus SARS-Cov-2 es un virus que apareció en China en 2020. Después se extendió a todos los continentes del mundo provocando una pandemia. Actualmente Europa y América son los más afectados. Este nuevo virus, provoca la enfermedad conocida con el nombre de COVID-19 (Gobierno de México, 2020).

A 25 de junio de 2020 se han confirmado 197,000 casos y 24,324 defunciones en México, con una tasa de letalidad de 8.1%. La epidemia se ha propagado a lo largo del territorio nacional. El riesgo de contagio a nivel global y local es muy alto.

Hasta el momento no se cuenta con una vacuna para la prevención de la infección por COVID-2019, por lo que la mejor manera de prevenir la infección es evitar exponerse al virus.

Por lo anterior, algunas de las medidas que han demostrado que evitan el contagio es el lavado frecuente de manos con agua y jabón al menos por 60 segundos, en especial después de ir al baño, antes de comer, y después de sonarse la nariz, toser o estornudar, así como después de tener contacto directo con personas enfermas o su entorno, para ello se puede utilizar un producto para desinfección de manos que contenga 70% de alcohol (por 20-30 segundos).

Limpiar y desinfectar objetos y superficies que se tocan con frecuencia usando un producto común de limpieza en forma de rociador o toalla ayuda a prevenir contagios también.

La pandemia que estamos presenciando, representa un reto para la sociedad, desde puntos económicos, políticos, de salud, etc. Por lo que las soluciones tecnológicas pueden ser de gran ayuda para combatir este virus, es así como se presenta la propuesta de una maquina dispensadora de botellas de gel antibacterial (de 60 ml) (Ilustración 1).

El uso de gel antibacterial, es una medida de control para evitar la propagación del virus COVID-19, ya que, al estar elaborado a base de alcohol, este elimina los virus y bacterias en nuestras manos; por lo que es de gran importancia disponer de este producto en todo momento.

Ilustración 1 Dispensadora de botellas de gel antibacterial.

OBJETIVOS

OBJETIVO GENERAL

Ante la problemática de la pandemia mundial, y las medidas preventivas de "la nueva normalidad", se propone facilitar mediante una máquina expendedora de gel antibacterial, los medios necesarios para la desinfección oportuna de manos en los diferentes espacios públicos.

OBJETIVOS PARTICULARES

- Llevar a cabo el diseño mecánico del dispositivo mediante el programa SOLIDWORS.
- Realizar un pequeño estudio de costos, donde los materiales representen la mayor ventaja relación calidad/precio y un esquema el cual considere la facilidad de obtención hipotética de los mismos.
- Aplicación de conocimientos estructurales que permitan el uso de un control lógico mediante FPGA.
- Evaluación y diseño del sistema eléctrico de la maquinaria, así como el acondicionamiento de las diversas etapas de control.
- Implementación de un mecanismo de despacho y conteo de artículos.
- Implementación de sistemas de interacción con el usuario: Selector de número de artículos, indicadores de disponibilidad, mensaje de bienvenida, despedida y precio.
- Implementación de un sistema contador de monedas, el cual reciba monedas por gravedad y efectué la organización de estas para el conteo de estas y el vuelto de cambio correspondiente.

DESARROLLO

Recordando que el diseño mecatrónico es una forma sistemática para desarrollar un dispositivo tecnológico. De esta manera, se plantean los criterios más importantes que se deben tomar en cuenta en todo el proceso. Es por ello, que se plantea el diseño de la maquina dispensadora de botellas de gel antibacterial de la siguiente forma:

- Sistemas mecánicos y estructurales.
  - Diseño estructural.
  - Materiales.
  - Costos.
- Sistemas de control.
  - Programación en VHDL.
  - Módulos y subsistemas.
- Sistemas electrónicos y de potencia.
- Materiales y costos.

SISTEMAS MECÁNICOS Y ESTRUCTURALES

En ingeniería, el diseño mecánico es el proceso de dar forma, definir dimensiones, materiales, tecnología de fabricación y funcionamiento de una máquina para que cumpla una determinada función o funciones.

En este apartado procederemos a mostrar y describir el diseño estructural y mecánico de la máquina, este dispositivo tiene la particularidad de poseer dimensiones compactas (70x60x16cm) en comparación con otras máquinas. (Ilustración 2)

El dispositivo presentado tiene una capacidad de 30 botellas de 60 ml, dadas sus dimensiones reducidas es muy práctico para ser colocado en salones de clase, centros comerciales, cafeterías y centros de convivencia; para consultar los detalles del diseño dirigirse al anexo **A: Diseño Mecánico**, que se encuentra al final de este documento.

Es importante mencionar que toda la estructura está diseñada en lámina cold rolled (laminada en frío) y acero inoxidable, de diversos calibres, con acabados en pintura electroestática (blanca y azul), como se muestra en la hoja 1 del anexo A. El ensamblaje de la maquina consta de dos módulos:

- Estructura externa.
  - Cubierta.
  - Puerta.
  - Accesorios.
  - Botones.
  - Rejilla de cambio.
  - Acceso al producto.
- Estructura interna.
  - Canales.
  - Hélices.
  - Soportes.

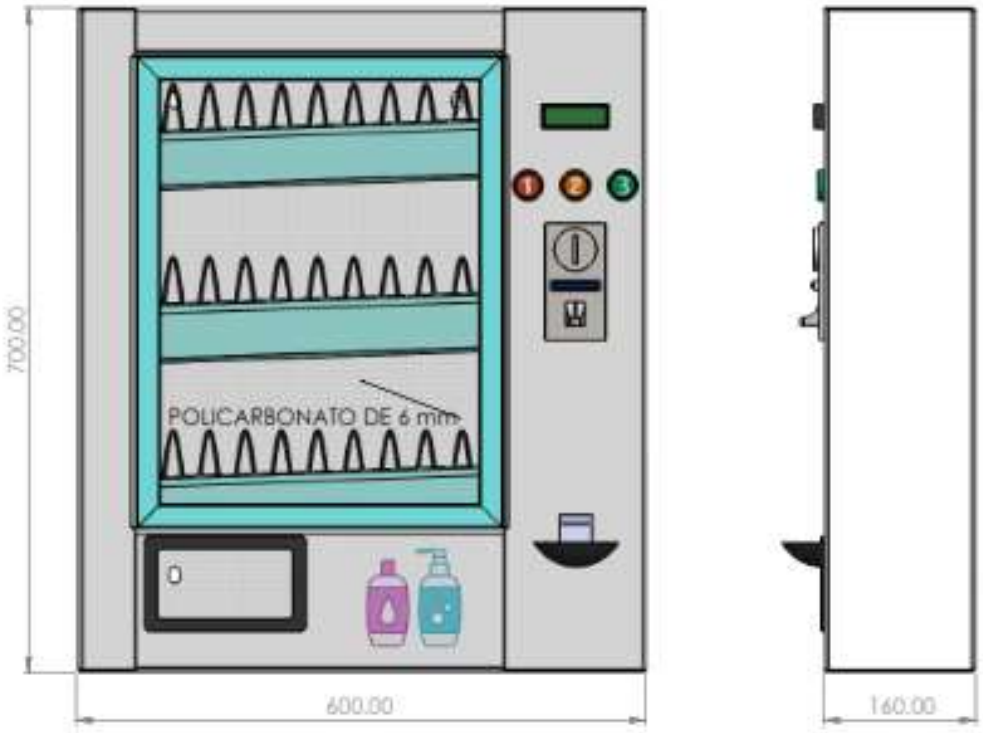


Ilustración 2 Dimensiones del dispensador.

ESTRUCTURA EXTERNA

Conformada por chapa metálica laminada en frío calibre 14 (1.9 mm de espesor), para dar rigidez a la estructura. En la Tabla 1 se muestra el despiece de la estructura y como se observa en la Ilustración 3, esta consta de todos los elementos con los que interactúa el usuario directamente como lo son: botones de selección de producto (3), pantalla (4), rendija de monedas (5 y 8) y puerta de acceso al producto (2).

N.° DE ELEMENTO	N.° DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	CUBIERTA1	LAMINA DE CR CAL 14	1
2	TAPA	VARIOS	1
3	boton	1"X1"	3
4	LCD	16X2	1
5	SMONEDA	VARIOS	1
6	SEPARADOR	ALUMINIO 6063 T6	1
7	SEPARADOR1	LAMINA DE ACERO INOX CAL. 18	1
8	SALIDA	VARIOS	1
9	PUERTA1	TUBO CUADRADO DE CR 1"	1
10	PUERTA2	POLICARBONATO DE 6 mm477x367 mm	1
11	ACTUADORES	LAMINA DE ACERO INOX CAL. 14	1
12	CHAROLA MONEDAS	LAMINA DE ACERO INOX. CAL.18	1

Tabla 1 De elementos de la estructura externa.

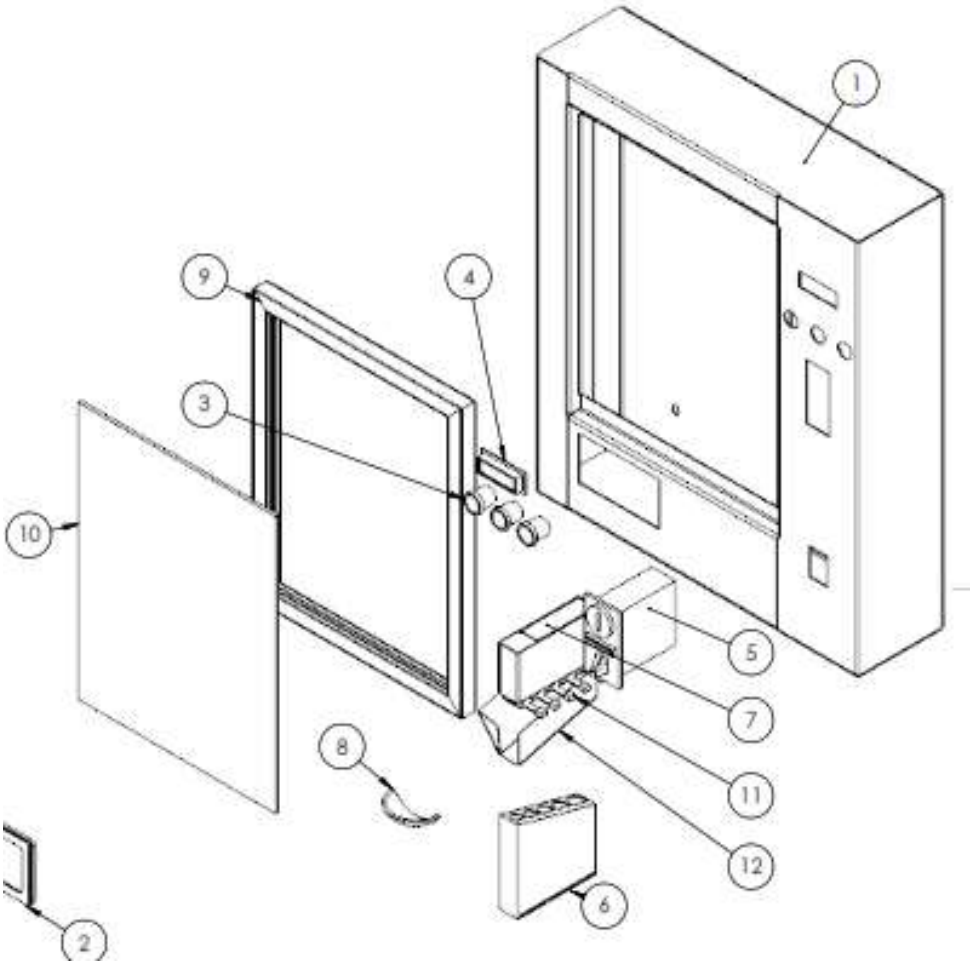


Ilustración 3 Despiece estructura externa.

La puerta (9) está diseñada en un perfil cuadrado de cold rolled calibre 16, con una ranura de 9.54 mm para un canal donde se aloja una placa de policarbonato de 6 mm de espesor (Ilustración 4).

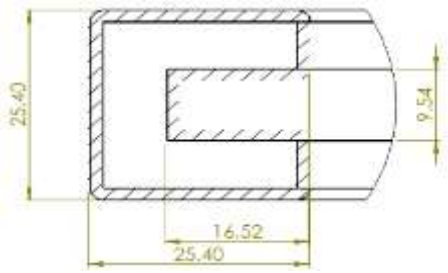


Ilustración 4 Sección transversal del marco de la puerta.

En la ilustración 3, se observan tres elementos que constituyen el sistema de acomodo de monedas, los cuales son el sensor receptor de monedas (5), el depósito de monedas (6) y una chapa en lámina de acero inoxidable (7), cuya configuración es la mostrada en la ilustración 5.

Las monedas se caen por gravedas y dependiendo del diametro de estas se depositan en el apartado correspondiente, como se observa en la Ilustración 5, en el extremo derecho, se aprecian las ranuras para cada una de las monedas, donde el oren y la capacidad para cada moneda es la que se muestra en la Tabla 2.

Valor	Cantidad
$1	84
$2	71
$5	59
$10	45

Tabla 2 Orden y cantidad de monedas por ranura.

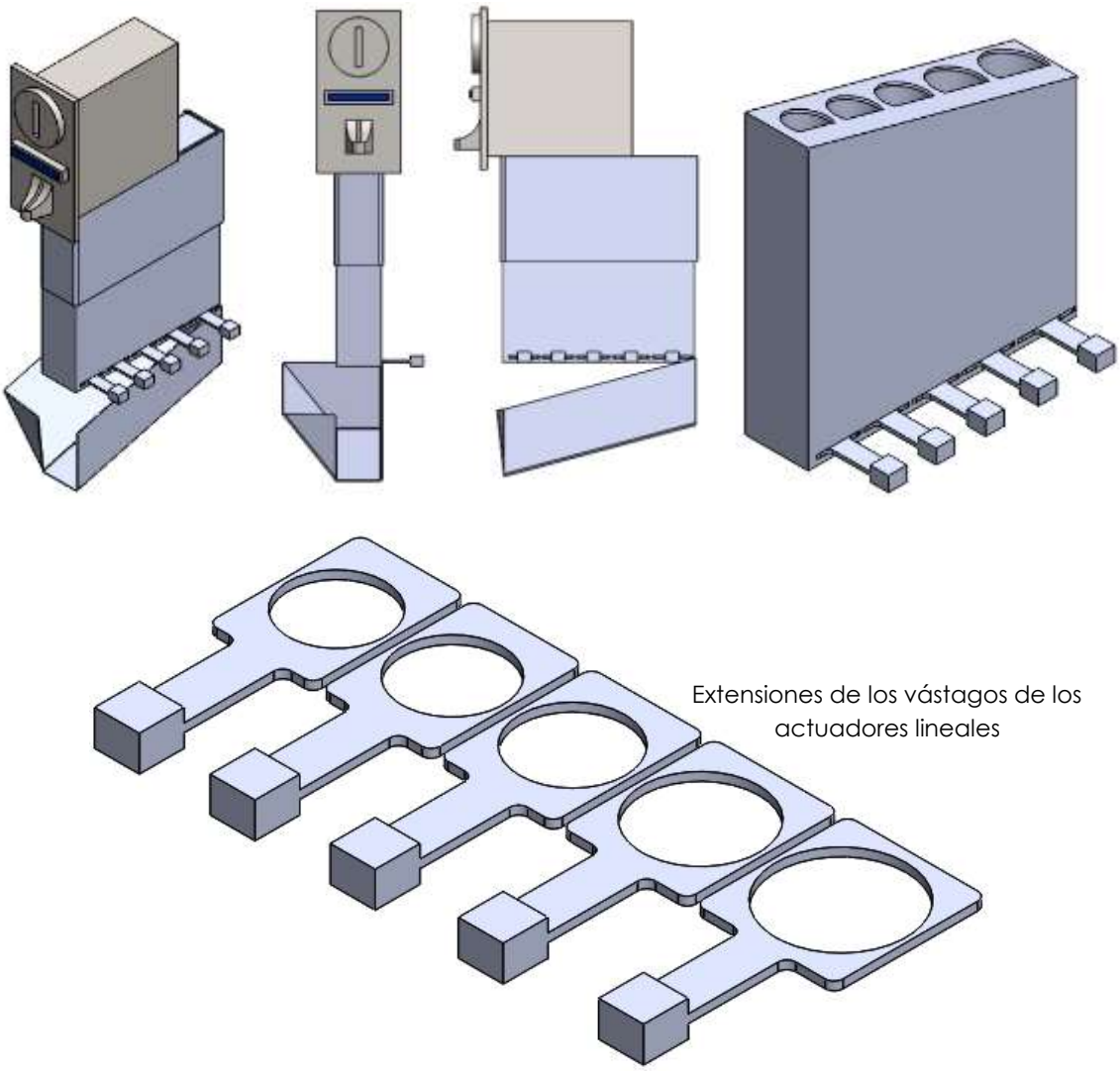


Ilustración 5 Sistema de almacenamiento de monedas.

En el depósito de monedas (6), en la parte inferior, se encuentran unas ranuras por donde actuadores lineales empujan las monedas para el cambio y estas llevan a la bandeja de salida por gravedad.

Como se observa en la parte inferior de la ilustración 5, la pieza (11), está diseñada en lámina de acero inoxidable calibre 14 (2 mm), donde cada barreno tiene un ajuste mínimo para que solo pueda ingresar una moneda, cuando el actuador esta desactivado, la extensión está dentro del depósito, si existe una moneda esta cae por su peso dentro de la extensión, cuando se activa el actuador este empuja la extensión que a su vez expulsa una moneda de la denominación deseada, esta cae por gravedad hasta una charola de metal (12) que tiene una inclinación que lleva las monedas hasta la salida de la máquina para ser retirada por el usuario.

ESTRUCTURA INTERNA

La estructura interna consta de varias piezas soldadas (Ilustración 6) y está diseñada en lámina de acero 1020 laminada en frío (cold rolled), el diseño final de sus partes se muestra en la hoja 1 del anexo A de forma individual y en la ilustración 7 se muestra la vista de explosión de su ensamble, en la tabla 3 se describe y nombra cada una de sus piezas.

La estructura cuenta con dos soportes principales (2 y 3), tres canales de acero (1) los cuales tienen una inclinación de 2° para facilitar el deslizamiento de las botellas, 3 hélices plásticas(5) las cuales empujan el producto al dar una vuelta y finalmente un charola de acero inoxidable por donde caen las botellas(4), evitando que se atoren dentro del dispensador, finalmente, tiene una ligera circunferencia al final de la pieza, lo que hace que al caer una botella esta continúe su trayectoria hasta la puerta de acceso al producto donde es retirado por el cliente.

La lista de materiales utilizados en el diseño estructural se muestra en la tabla 4.

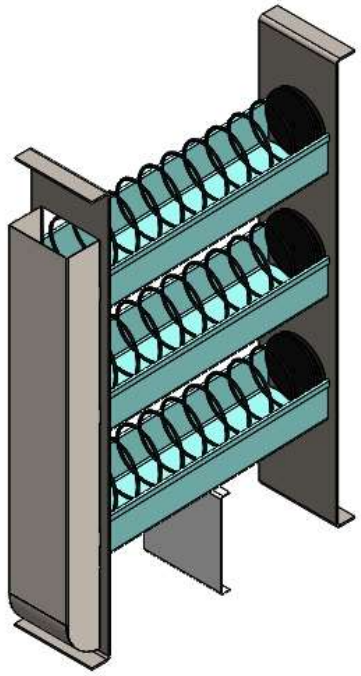


Ilustración 6 Estructura interna.

N.° DE ELEMENTO	N.° DE PIEZA	DESCRIPCIÓN	CANTIDAD
1	CANAL	LAMINA DE C.R. CAL 14	3
2	SOPORTED	LAMINA DE C.R. CAL 11	1
3	SOPORTEI	LAMINA DE C.R. CAL 11	1
4	CHAROLA	LAMINA DE ACERO INOX. CAL 20	1
5	HELICE	VARIOS	3
6	DIVISION	LAMINA DE C.R. CAL 16	1

Tabla 3 Despiece de la estructura interna.

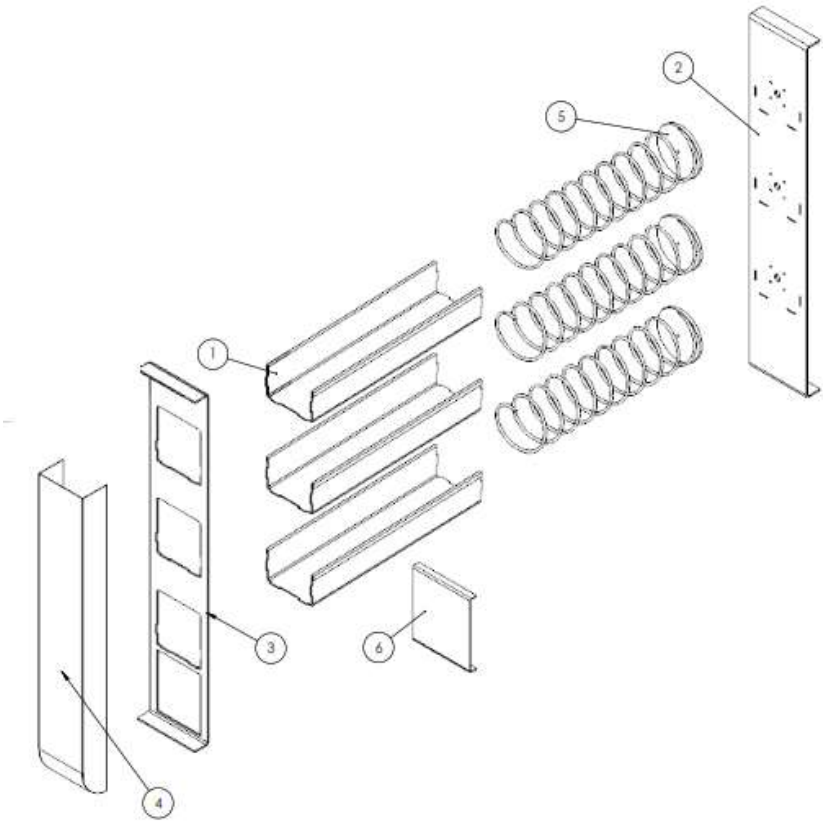
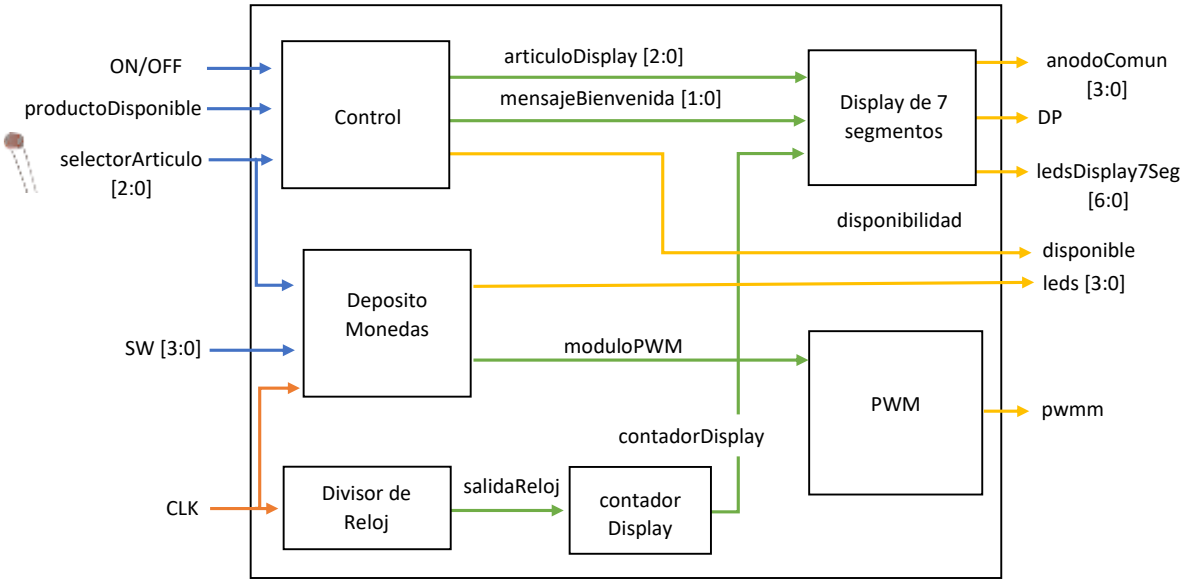


Ilustración 7 Despiece de la estructura interna.

Ilustración 8 Dispensador de botellas de gel antibacterial.

SISTEMAS DE CONTROL

Para el sistema de control de la maquina se utilizó una FPGA, ya que, debido a su rigidez en la sintaxis de las instrucciones de programación, además de ser el que mejor se adapta para el diseño jerárquico; cada uno de los módulos fueron diseñados por separado e implementados en un *Top Level Design* (TLD) como se muestra en la Ilustración 9.



El cable de la **señal PWM** será conectado directamente al puerto **JA1** de la Nexys 2 y la alimentación del servomotor se conectará a los puertos **+JA6 o +JA12** y **-JA5 o -JA11**.

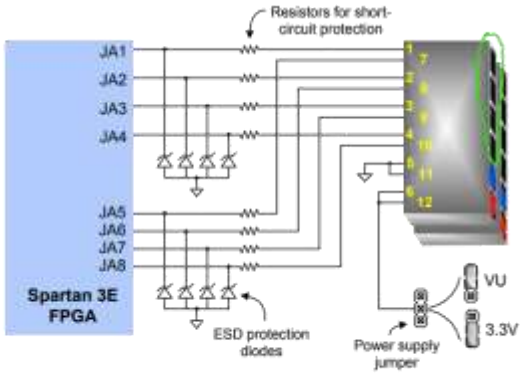


Figure 23: Nexys2 Pmod connector circuits

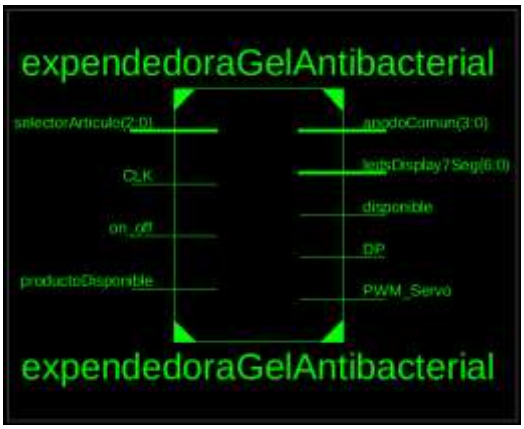
Table 3: Nexys2 Pmod Connector Pin Assignments

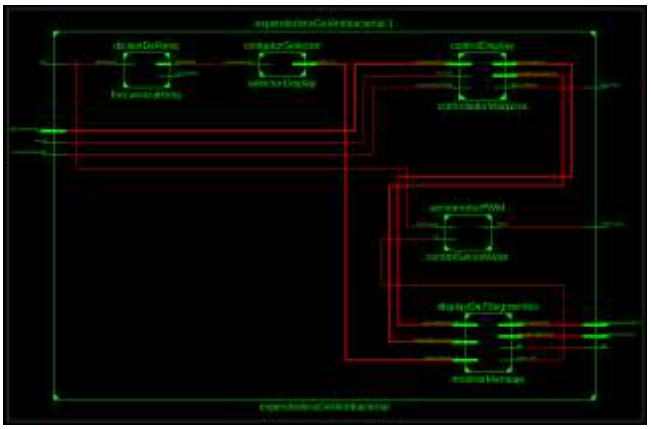
Pmod JA		Pmod JB		Pmod JC		Pmod JD	
JA1: L15	JA7: K13	JB1: M13	JB7: P17	JC1: G15	JC7: H15	JD1: J13	JD7: K14[1]
JA2: K12	JA8: L16	JB2: R18	JB8: R16	JC2: J16	JC8: F14	JD2: M18	JD8: K15[2]
JA3: L17	JA9: M14	JB3: R15	JB9: T18	JC3: G13	JC9: G16	JD3: N18	JD9: J15[3]
JA4: M15	JA10: M16	JB4: T17	JB10: U18	JC4: H16	JC10: J12	JD4:; P18	JD10: J14[4]

Notes: [1] shared with LD3 [2] shared with LD3 [3] shared with LD3 [4] shared with LD3

Ilustración 9 Top Level Design de la maquina expendedora de botellas de gel antibacterial y puertos de conexión.

DIAGRAMA TLD - EXPENDEDORAGELANTIBACTERIAL





TLD EXPENDEDORAGELANTIBACTERIAL - CÓDIGO VHDL

```
--6.-TLD (Top Level Design) máquina expendedora de gel antibacterial:
--Este código sirve para unir los módulos anteriores y poder realizar la operación de la máquina expendedora.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity expendedoraGelAntibacterial is
    Port ( on_off : in  STD_LOGIC;
           selectorArticulo : in  STD_LOGIC_VECTOR (2 downto 0);
           productoDisponible : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
                      ledsDisplay7Seg : out  STD_LOGIC_VECTOR (6 downto 0);
                      DP : out  STD_LOGIC;
           anodoComun : out  STD_LOGIC_VECTOR (3 downto 0);
           disponible : out  STD_LOGIC;
                      PWM_Servo : out  STD_LOGIC);

end expendedoraGelAntibacterial;

architecture Behavioral of expendedoraGelAntibacterial is
--SIGNAL: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo
--existe durante la ejecución del código y sirve para poder almacenar algún valor, se debe declarar dentro de la
--arquitectura y antes de su begin.
signal reloj : STD_LOGIC;--Reloj con frecuencia menor a 50MHz.
signal CLK50MHz : STD_LOGIC;--Reloj con frecuencia de 50MHz.
signal selector : STD_LOGIC_VECTOR (1 downto 0);--contador/selector para encender cada display de 7 segmentos
signal articulo : STD_LOGIC_VECTOR (2 downto 0);--selección de producto del usuario
signal mensaje : STD_LOGIC_VECTOR (1 downto 0);--mensaje mostrado en el display de 7 segmentos
signal onOffServo : STD_LOGIC;--Variable que enciende con 0 el servomotor y lo apaga con 1.

begin
          --INSTANCIAS:
          --Debo darle un nombre a cada instancia que cree, indicar el nombre de la entidad del módulo que quiero
          --instanciar, usar la palabra reservada port map(); y dentro de su paréntesis asignarle a todas las entradas y
          --salidas del módulo instanciado una entrada, salida o signal de ste mdulo separadas por comas una de la otra,
          --esto hará que lo que entre o salga del otro módulo, entre, salga o se guarde en este.
          --La sintaxis que debemos usar es la siguiente:

          --NombreInstancia            :         entity work.Entidad_Del_Modulo_Que_Queremos_Instanciar     port map(
          --                  Entrada_Del_Modulo_Instanciado => Entrada_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Salida_En_Este_Modulo,

          --                  Entrada_Del_Modulo_Instanciado => Salida_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Entrada_En_Este_Modulo,

          --                  Entrada_Del_Modulo_Instanciado => Signal_En_Este_Modulo,
          --                  Salida_Del_Modulo_Instanciado  => Signal_En_Este_Modulo
          --);

          --INSTANCIA DEL divisorDeReloj para obtener la frecuencia en la que quiero que se cree el contador/selector.
          frecuenciaReloj : entity work.divisorDeReloj port map(
                            relojNexys2 => CLK,
                            --La entrada relojNexys2 del divisorDeReloj se asigna a la entrada CLK de este módulo.
                            salidaReloj => reloj,
                            --La salida salidaReloj del divisorDeReloj se asigna a la signal reloj de este módulo.
                            salida50MHz => CLK50MHz
                            --La salida salidaReloj del divisorDeReloj se asigna a la signal CLK50MHz de este módulo.
                  );
          --INSTANCIA MODULO contadorSelector para obtener el contador/selector que prendera cada display individualmente.
          selectorDisplay : entity work.contadorSelector port map(
                            frecuenciaReloj => reloj,
                            --A la entrada frecuenciaReloj del contadorSelector se le asigna el valor de la signal
                            --reloj de este módulo.
```

```
                        contador => selector
                        --La salida contador del contadorSelector se asigna a la signal selector de este módulo.
                );
        --INSTANCIA MODULO control para que el usuario pueda prender la maquina o introducir el número de artículos.
        --que quiere obtener
        controladorMaquina : entity work.controlDisplay port map(
                        ON_OFF => on_off,
                        --La entrada ON_OFF del control se asigna a la entrada on_off de este módulo.
                        selectorArticulo => selectorArticulo,
                        --La entrada selectorArticulo del control se asigna a la entrada selectorArticulo de
                        --este módulo.
                        productoDisponible => productoDisponible,
                        --La entrada productoDisponible del módulo control se asigna a la entrada productoDisponible
                        --de este módulo.
                        articuloDisplay => articulo,
                        --La salida articuloDisplay del módulo control se asigna a la signal artículo de este módulo.
                        mensajeBienvenida => mensaje,
                        --La salida mensajeBienvenida del módulo control se asigna a la signal mensaje del módulo.
                        disponibilidad => disponible
                        --La salida disponibilidad del módulo control se asigna a la salida disponible de este modulo
                );
        --INSTANCIA DEL MODULO displayDe7Segmentos para recibir el selector del contador y prender cada uno de los 4
        --displays dependiendo de las entradas provenientes del módulo control y con la frecuencia del selector,
        --prendera y apagara los 4 displays tan rápido que para el ojo humano parezca que todos estén encendidos al
        --mismo tiempo con valores diferentes.
        mostrarMensaje : entity work.displayDe7Segmentos port map(
                        articuloDisplay => articulo,
                        --A la entrada articuloDisplay de displayDe7Segmentos se le asigna el valor de la signal
                        --artículo de este módulo.
                        mensajeBienvenida => mensaje,
                        --A la entrada mensajeBienvenida de displayDe7Segmentos se le asigna el valor de la signal
                        --mensaje de este módulo.
                        selectorMUX => selector,
                        --A la entrada selectorMUX de displayDe7Segmentos se le asigna el valor de la signal selector
                        --de este módulo.
                        prenderDisplay => anodoComun,
                        --La salida prenderDisplay de displayDe7Segmentos se asigna a la salida anodoComun.
                        ledsAhastaG => ledsDisplay7Seg,
                        --La salida ledsAhastaG de displayDe7Segmentos se asigna a la salida ledsDisplay7Seg.
                        DP => DP,
                        --La salida DP de displayDe7Segmentos se asigna a la salida DP de este módulo.
                        servo_ON => onOffServo
                        --La salida servo_ON de displayDe7Segmentos se asigna a la signal onOffServo de este módulo.
                );
        controlServoMotor : entity work.servomotorPWM port map(
                        CLKNexys2 => CLK,
                        --A la entrada CLKNexys2 de servomotorPWM se le asigna el valor de la signal CLK50MHz.
                        rst => onOffServo,
                        --A la entrada rst de servomotorPWM se le asigna el valor de la signal onOffServo del módulo.
                        PWM => PWM_Servo
                        --La salida PWM del servomotorPWM se asigna a la salida PWM_Servo de este módulo.
                );

end Behavioral;
```

DIVISOR DE RELOJ - **CÓDIGO VHDL**

```
--1.-DIVISOR DE RELOJ:
--Este proceso sirve para dictarle al reloj en que frecuencia quiero que opere.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerias para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería declarada para poder hacer operaciones matemáticas sin considerar el signo.
entity divisorDeReloj is
    Port ( relojNexys2 : in  STD_LOGIC;   --Reloj de 50MHz proporcionado por la NEXYS 2 en el puerto B8.
           salida50MHz : out  STD_LOGIC;  --Salida CLK de 50MHz proporcionado por la NEXYS 2.
           salidaReloj : out  STD_LOGIC); --Reloj que quiero con una frecuencia menor a 50MHz.
end divisorDeReloj;

--Arquitectura: Aquí se hará el divisor de reloj haciendo uso de una signal y condicionales if.
architecture frecuenciaNueva of divisorDeReloj is
--SIGNAL: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo
--existe durante la ejecución del código y sirve para poder almacenar algún valor, se debe declarar dentro de la
--arquitectura y antes de su begin, se le asignan valores con el símbolo :=
signal divisorDeReloj : STD_LOGIC_VECTOR (24 downto 0);
--Esta signal sirve para que podamos obtener una gran gama de frecuencias indicadas en la tabla del divisor de reloj
--dependiendo de la coordenada que elijamos tomar del vector para asignársela a la salida.
begin
        process(relojNexys2)
        begin
                if(rising_edge(relojNexys2)) then
                        --La instrucción rising_edge() hace que este condicional solo se ejecute cuando ocurra
                        --un flanco de subida en la señal de reloj clkNexys2 proveniente de la NEXYS 2.
                        divisorDeReloj <= divisorDeReloj + 1;  --Esto crea al divisor de reloj.
                end if;
        end process;
        --Debo asignar el contenido de una coordenada de la signal divisorDeReloj a salidaReloj para obtener
        --una frecuencia en específico, cada coordenada del vector corresponde a una frecuencia en la tabla del
        --divisor de reloj.
        salidaReloj <= divisorDeReloj(16);--En la tabla se ve que la coordenada 19 proporciona una frecuencia de 1.49Hz.
        salida50MHz <= relojNexys2;  --Salida del CLK sin el divisor de reloj para crear la señal PWM.

end frecuenciaNueva;
```

CONTADORSELECTOR - **CÓDIGO VHDL**

```
--2.-CONTADOR DE 2 BITS:
--En este código el contador es de 2 bits, esto implica que contara desde el cero hasta el 3 en forma binaria:
--00, 01, 10 y 11.
--Estos números binarios en el módulo siguiente representaran al selector y el selector lo que hará es guardar en la
--signal digito (del siguiente modulo también) 4 letras diferentes que muestren una palabra, este barrido se debe hacer
--en orden y con la frecuencia dictada por el divisorDeReloj para prender individualmente cada display de 7 segmentos.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerias que sirven solamente para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería para poder realizar operaciones matemáticas sin considerar el signo.

--Entidad: En la entidad se declara que el contador sea de 2 bits porque como en la NEXYS 2 hay 4 displays de 7
--segmentos, lo que se busca es que durante un ciclo de reloj todos los displays sean encendidos una vez, mostrando
--cada letra del letrero que corresponda, se puede hacer esto con solo 2 bits porque cuando el selector adopte los
--valores 00, 01, 10 y 11 prendera una vez cada uno de los 4 displays de 7 segmentos durante cada ciclo de reloj.
entity contadorSelector is
    Port ( frecuenciaReloj : in  STD_LOGIC;      --Este es el reloj proveniente del módulo divisorDeReloj
           contador : out  STD_LOGIC_VECTOR (1 downto 0));
end contadorSelector;

architecture contador2Bits of contadorSelector is
--SIGNAL: Existe solo en VHDL y sirve para almacenar datos que solo sobrevivirán durante la ejecución del programa,
--no está conectada a ningún puerto de la tarjeta de desarrollo y se le asignan valores con el símbolo :=
signal conteoAscendente : STD_LOGIC_VECTOR (1 downto 0) := "00"; --Se le da un valor inicial de 0 al vector.
begin
        process(frecuenciaReloj)
        begin
                --La instrucción rising_edge indica que cada que ocurra un flanco de subida en el reloj, se le sumara
                --un 1 binario al valor que tenía previamente el vector conteoAscendente, esto hará que se cree la
                --secuencia 00, 01, 10 y 11 en el selector, específicamente en ese orden.
                if(rising_edge(frecuenciaReloj)) then
                        conteoAscendente <= conteoAscendente + "01";
                        --El conteo solito volverá a ser 00 cuando se supere el valor 11 en el vector
                        --conteoAscendente de 2 bits.
                end if;
        end process;
        --Se usa una signal en vez de hacer el contador directo con la salida contador porque a las salidas solo se
        --les puede asignar un valor, no se les puede leer.
        contador <= conteoAscendente;

end contador2Bits;
```

CONTROLDISPLAY - **CÓDIGO VHDL**

```
--3.-CONTROL PARA LA EXPENDEDORA DE GEL ANTIBACTERIAL:
--Por medio de este módulo se enciende toda la máquina expendedora, se elige el tipo de producto que dará y
--se recibe la señal del sensor ldr (Light Dependant Resistor) para ver si hay productos disponibles o no.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerias que sirven solamente para poder usar el lenguaje VHDL.
entity controlDisplay is
    Port ( ON_OFF : in  STD_LOGIC;      --Switch que enciende o apaga toda la máquina expendedora.
           selectorArticulo : in  STD_LOGIC_VECTOR (2 downto 0);     --Elegir tipo de producto.
           productoDisponible : in  STD_LOGIC;   --Sensor LDR que indica si hay productos disponibles.
           articuloDisplay : out  STD_LOGIC_VECTOR (2 downto 0);     --Letrero que indica el articulo elegido.
           mensajeBienvenida : out  STD_LOGIC_VECTOR (1 downto 0);--Letrero que dice HOLA.
           disponibilidad : out  STD_LOGIC);      --Led que se enciende si hay productos disponibles.
end controlDisplay;

architecture Behavioral of controlDisplay is begin
--Los process se declaran en VHDL para poder utilizar condicionales y bucles, dentro de su paréntesis se
--declaran las entradas que se utilizarán.
process(ON_OFF, selectorArticulo, productoDisponible) begin
        if(productoDisponible = '1') then       --Si hay producto disponible, no se prendera el led disponibilidad.
                --El bit ON_OFF funciona como enable del led disponibilidad y el selector articuloDisplay.
                disponibilidad <= ON_OFF;     --Si hay productos disponibles se prendera un led.
                if (selectorArticulo = "001") then
                        articuloDisplay <= "00" & ON_OFF;       --selectorArticulo = articuloDisplay = 001 = Art.1
                        mensajeBienvenida <= "00";    --mensajeBienvenida 00 = Se muestra el letrero del artículo.
                elsif(selectorArticulo = "010") then
                        articuloDisplay <= "0" & ON_OFF & "0"; --selectorArticulo = articuloDisplay = 010 = Art.2
                        mensajeBienvenida <= "00";    --mensajeBienvenida 00 = Se muestra el letrero del artículo.
                elsif(selectorArticulo = "100") then
                        articuloDisplay <= ON_OFF & "00";       --selectorArticulo = articuloDisplay = 100 = Art.3
                        mensajeBienvenida <= "00";    --mensajeBienvenida 00 = Se muestra el letrero del artículo.
                else
                        articuloDisplay <= "000";          --articuloDisplay = 000 = Se muestra el letrero de saludo.
                        mensajeBienvenida <= "0" & ON_OFF; --mensajeBienvenida 01 = HOLA
                end if;
        else
                disponibilidad <= '0';                  --Si NO hay productos disponibles no se prenderá el led.
                articuloDisplay <= "000";               --No se mostrará ningún artículo seleccionado.
                mensajeBienvenida <= ON_OFF & "0";      --El mensajeBienvenida 10 = 00 = ---- (que significa vacío).
        end if;
end process;

end Behavioral;
```

DISPLAY DE 7 SEGMENTOS - **CÓDIGO VHDL**

```
--4.-DECODIFICADOR PARA MOSTRAR ALGO EN LOS 4 DISPLAYS DE 7 SEGMENTOS A LA VEZ:
--Este código sirve para mostrar 4 cosas diferentes en los 4 displays de 7 segmentos, esto por sí solo
--no se puede lograr porque los displays de 7 segmentos solo pueden ensenar un solo número a la vez en todos los
--displays de la NEXYS 2, para lograrlo debemos usar un Multiplexor, un Divisor de Reloj y un Contador.
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerias para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librerías declaradas para poder hacer operaciones matemáticas sin considerar el signo o para hacer conversiones de
--binario a integer (osea de binario a decimal).

entity displayDe7Segmentos is
    Port ( articuloDisplay : in  STD_LOGIC_VECTOR (2 downto 0);--Entrada que me indica el artículo elegido.
           mensajeBienvenida : in  STD_LOGIC_VECTOR (1 downto 0);--vector que me indica el mensaje a desplegar.
           selectorMUX : in  STD_LOGIC_VECTOR (1 downto 0);--Selector proveniente del módulo contadorDisplay.
           prenderDisplay : out  STD_LOGIC_VECTOR (3 downto 0);--Vector de nodos comunes para prender cada display.
           ledsAhastaG : out  STD_LOGIC_VECTOR (6 downto 0);--Vector con los leds A,B,C,D,E,F y G.
           DP : out  STD_LOGIC;--Puntito en los displays.
           servo_ON : out  STD_LOGIC);--Salida que enciende al servomotor cuando se haya seleccionado un artículo.
end displayDe7Segmentos;

--Arquitectura: Aquí se va a hacer un multiplexor que al recibir el selector del módulo contadorSelector, me permita
--mostrar un numero en solo uno de los displays de 7 segmentos.
architecture muxAdisplay of displayDe7Segmentos is
begin
        process(articuloDisplay, mensajeBienvenida, selectorMUX) begin
                if(mensajeBienvenida = "00") then -- mensajeBienvenida = 00 = Se muestra el letrero del artículo.
                        --if anidado para ver el estado del otro selector llamado articuloDisplay.
                        servo_ON <= '0'; --El servomotor está encendido.
                        if (articuloDisplay = "001") then
                                case selectorMUX is
                                        when "00"   =>
                                                prenderDisplay <= "0111";--Prende solo el 1er display.
                                                ledsAhastaG <= "0001000";--Letra A
                                                DP <= '1';--El puntito esta apagado.
                                        when "01"   =>
                                                prenderDisplay <= "1011";--Prende solo el 2do display.
                                                ledsAhastaG <= "1111010";--Letra r
                                                DP <= '1';--El puntito esta apagado.
                                        when "10"   =>
                                                prenderDisplay <= "1101";--Prende solo el 3er display.
                                                ledsAhastaG <= "1110000";--Letra t.
                                                DP <= '0';--El puntito esta encendido.
                                        when others =>
                                                prenderDisplay <= "1110";--Prende solo el 4to display.
                                                ledsAhastaG <= "1001111";--Numero 1
                                                DP <= '1';--El puntito esta apagado.
                                end case;
                        elsif(articuloDisplay = "010") then
                                case selectorMUX is
                                        when "00"   =>
                                                prenderDisplay <= "0111";--Prende solo el 1er display.
                                                ledsAhastaG <= "0001000";--Letra A
                                                DP <= '1';--El puntito esta apagado.
                                        when "01"   =>
                                                prenderDisplay <= "1011";--Prende solo el 2do display.
                                                ledsAhastaG <= "1111010";--Letra r
                                                DP <= '1';--El puntito esta apagado.
                                        when "10"   =>
                                                prenderDisplay <= "1101";--Prende solo el 3er display.
                                                ledsAhastaG <= "1110000";--Letra t.
                                                DP <= '0';--El puntito esta encendido.
                                        when others =>
                                                prenderDisplay <= "1110";--Prende solo el 4to display.
                                                ledsAhastaG <= "0010010";--Numero 2
                                                DP <= '1';--El puntito esta apagado.
                                end case;
                        elsif(articuloDisplay = "100") then
                                case selectorMUX is
                                        when "00"   =>
                                                prenderDisplay <= "0111";--Prende solo el 1er display.
                                                ledsAhastaG <= "0001000";--Letra A
                                                DP <= '1';--El puntito esta apagado.
                                        when "01"   =>
                                                prenderDisplay <= "1011";--Prende solo el 2do display.
                                                ledsAhastaG <= "1111010";--Letra r
                                                DP <= '1';--El puntito esta apagado.
                                        when "10"   =>
                                                prenderDisplay <= "1101";--Prende solo el 3er display.
                                                ledsAhastaG <= "1110000";--Letra t.
                                                DP <= '0';--El puntito esta encendido.
                                        when others =>
                                                prenderDisplay <= "1110";--Prende solo el 4to display.
                                                ledsAhastaG <= "0000110";--Numero 3
                                                DP <= '1';--El puntito esta apagado.
                                end case;
                        else
                                case selectorMUX is
                                when "00"   =>
                                        prenderDisplay <= "1111";--Los 4 displays están apagados.
                                        ledsAhastaG <= "1111110";--Todos los leds de los displays están apagados.
                                        DP <= '1';--El puntito esta apagado.
```

```
                                    when "01"   =>
                                              prenderDisplay <= "1111";--Los 4 displays están apagados.
                                              ledsAhastaG <= "1111110";--Todos los leds de los displays están apagados.
                                              DP <= '1';--El puntito esta apagado.
                                    when "10"   =>
                                              prenderDisplay <= "1111";--Los 4 displays están apagados.
                                              ledsAhastaG <= "1111110";--Todos los leds de los displays están apagados.
                                              DP <= '1';--El puntito esta apagado.
                                    when others =>
                                              prenderDisplay <= "1111";--Los 4 displays están apagados.
                                              ledsAhastaG <= "1111110";--Todos los leds de los displays están apagados.
                                              DP <= '1';--El puntito esta apagado.
                          end case;
                          end if;
                elsif(mensajeBienvenida = "01") then --Selector mensajeBienvenida = 01 = HOLA
                servo_ON <= '1'; --El servomotor está apagado.
                          case selectorMUX is
                                    when "00"   =>
                                                                  prenderDisplay <= "0111";--Prende solo el 1er display
                                                                  ledsAhastaG <= "1001000";--Letra H
                                                                  DP <= '1';--El puntito está apagado.
                                    when "01"   =>
                                                                  prenderDisplay <= "1011";--Prende solo el 2do display
                                                                  ledsAhastaG <= "0000001";--Letra O
                                                                  DP <= '1';--El puntito está apagado.
                                    when "10"   =>
                                                                  prenderDisplay <= "1101";--Prende solo el 3er display
                                                                  ledsAhastaG <= "1110001";--Letra L
                                                                  DP <= '1';--El puntito está apagado.
                                    when others =>
                                                                  prenderDisplay <= "1110";--Prende solo el 4to display
                                                                  ledsAhastaG <= "0001000";--Letra A
                                                                  DP <= '1';--El puntito está apagado.
                          end case;
                elsif(mensajeBienvenida = "10") then --Selector mensajeBienvenida = 10 = ---- (que significa vacio).
                servo_ON <= '1'; --El servomotor está apagado.
                          case selectorMUX is
                                    when "00"   =>
                                                                  prenderDisplay <= "0111";--Prende solo el 1er display
                                                                  ledsAhastaG <= "1111110";--símbolo -
                                                                  DP <= '1';--El puntito está apagado.
                                    when "01"   =>
                                                                  prenderDisplay <= "1011";--Prende solo el 2do display
                                                                  ledsAhastaG <= "1111110";--símbolo -
                                                                  DP <= '1';--El puntito está apagado.
                                    when "10"   =>
                                                                  prenderDisplay <= "1101";--Prende solo el 3er display
                                                                  ledsAhastaG <= "1111110";--símbolo -
                                                                  DP <= '1';--El puntito está apagado.
                                    when others =>
                                                                  prenderDisplay <= "1110";--Prende solo el 4to display
                                                                  ledsAhastaG <= "1111110";--símbolo -
                                                                  DP <= '1';--El puntito está apagado.
                          end case;
                else
                          prenderDisplay <= "1111";--Los 4 displays están apagados.
                          ledsAhastaG <= "1111111";--Todos los leds de los displays están apagados.
                          DP <= '1';--El puntito está apagado.
                end if;
        end process;

end muxAdisplay;
```

DEPÓSITO MONEDAS - **CÓDIGO VHDL**

```
--Este módulo calcula el precio total de compra, recibe el valor de las monedas ingresadas, realiza el
--cálculo del cambio, dependiendo de la cantidad ingresada, activa las salidas a los actuadores lineales
--y envia una señal de activación para el módulo PWM.
library IEEE;
use IEEE.std_logic_1164.all;
--Libreras para poder usar el lenguaje VHDL.
use IEEE.numeric_std.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
--Libreras declaradas para poder hacer operaciones matemáticas sin considerar el signo o para hacer conversiones de
--binario a integer (osea de binario a decimal).
entity depositoMonedas is
          Port( RELOJ50 : in STD_LOGIC;          --Reloj a 50MHz
                DATAINM : in STD_LOGIC_VECTOR (3 downto 0);  --Moneda ingresada [1 2 5 10]
                ARTICULOSM : in STD_LOGIC_VECTOR (2 downto 0);  --Cantidad de artículos a comprar
                MENSAJEM : out STD_LOGIC_VECTOR (4 downto 0); --Mensaje a LCD
                DATAOUTLED : out STD_LOGIC_VECTOR (3 downto 0); --A actuadores lineales [1 2 5 10]
                CAMM : out STD_LOGIC_VECTOR (3 downto 0);
                RESM : out STD_LOGIC_VECTOR (3 downto 0);
                DATAPWM : out std_logic
          );
end depositoMonedas;

architecture Behavioral of depositoMonedas is
--SIGNAL: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo
--existe durante la ejecución del código y sirve para poder almacenar algún valor, se debe declarar dentro de la
--arquitectura y antes de su begin, se le asignan valores con el símbolo :=
signal M1          : integer range 0 to 63 :=10; --Contador de Monedas de 1 peso.
signal M2          : integer range 0 to 63 :=10; --Contador de Monedas de 2 pesos.
```

```
signal M5          : integer range 0 to 63 :=10; --Contador de Monedas de 5 pesos.
signal M10         : integer range 0 to 63 :=10; --Contador de Monedas de 10 pesos.
--Cálculos resultantes de dinero.
signal DIN         : integer range 0 to 63 :=0; --Dinero ingresado.
signal PRET        : integer range 0 to 63 :=0; --Precio total de la compra.
signal CAM         : integer range 0 to 63 :=0; --Cambio.
signal CAM2        : integer range 0 to 63 :=0; --Iteraciones para el cálculo del cambio.
signal RES         : integer range 0 to 63 :=0; --Dinero restante.
signal contador : STD_LOGIC_VECTOR (24 DOWNTO 0):=(others => '0');
signal retardo : STD_LOGIC;
signal DATAOUTM    : STD_LOGIC_VECTOR (3 downto 0); --[1 2 5 10]

begin
        process(DATAINM, ARTICULOSM) begin
        --CALCULO DEL PRECIO A PAGAR
                case ARTICULOSM is
                        when "001" =>
                                        PRET <= 10; --1 BOTELLA ($10).
                                        MENSAJEM <= "00001"; --INGRESE $10.
                        when "010" =>
                                        PRET <= 20; --2 BOTELLAS ($20).
                                        MENSAJEM <= "00010"; --INGRESE $20.
                        when "100" =>
                                        PRET <= 30; --3 BOTELLAS ($30).
                                        MENSAJEM <= "00011"; --INGRESE $30.
                        when others =>
                                        PRET <= 0;--0 BOTELLAS.
                                        MENSAJEM <= "00100"; --SALUDO.
                end case;

        --INGRESO DE LAS MONEDAS
                case DATAINM is
                        when "1000" =>
                                if (M1 = 84) then                       --84 monedas de 1 peso
                                        MENSAJEM <= "01000";            --Deposito lleno FUERA DE SERVICIO
                                elsif(M1 = 0) then
                                        MENSAJEM <= "01001";            --Deposito vacío FUERA DE SERVICIO
                                else
                                        M1 <= M1 +1;
                                        DIN <= DIN + 1;
                                end if;
                        when "0100" =>
                                if (M2 = 71) then                       --71 monedas de 2 pesos
                                        MENSAJEM <= "01000";            --Deposito lleno FUERA DE SERVICIO
                                elsif(M2 = 0) then
                                        MENSAJEM <= "01001";            --Deposito vacío FUERA DE SERVICIO
                                else
                                        M2 <= M2 +1;
                                        DIN <= DIN + 2;
                                end if;
                        when "0010" =>
                                if (M5 = 59) then                       --59 monedas de 5 pesos
                                        MENSAJEM <= "01000";            --Deposito lleno FUERA DE SERVICIO
                                elsif(M5 = 0) then
                                        MENSAJEM <= "01001";            --Deposito vacío FUERA DE SERVICIO
                                else
                                        M5 <= M5 +1;
                                        DIN <= DIN + 5;
                                end if;
                        when "0001" =>
                                if (M10 = 45) then                      --45 monedas de 10 pesos
                                        MENSAJEM <= "01000";            --Deposito lleno FUERA DE SERVICIO
                                        elsif(M10 = 0) then
                                        MENSAJEM <= "01001";            --Deposito vacío FUERA DE SERVICIO
                                else
                                        M10 <= M10 +1;
                                        DIN <= DIN + 10;
                                end if;
                        when others => DIN <= 0;
                end case;

        --DINERO INGRESADO
        if (DIN = PRET) then            --EL DINERO INGRESADO ES IGUAL AL PRECIO DEL PRODUCTO
                MENSAJEM <= "00101"; --GRACIAS
        elsif (0<DIN and DIN < PRET) then --EL DINERO INGRESADO ES MENOR AL PRECIO DEL PRODUCTO
                MENSAJEM <= "00110"; --DEPOSITE (RES)
                RES <= PRET - DIN;
        elsif (DIN > PRET) then --EL DINERO INGRESADO EXCEDE EL PRECIO DEL PRODUCTO
                MENSAJEM <= "00111"; --SU CAMBIO ES (CAM)
                CAM <= DIN - PRET;
                CAM2 <= DIN - PRET;

        --CALCULO DEL CAMBIO
                if (CAM2 /= 0) then
                        for k in 0 to 8 loop
                                if (M5 /= 0 AND M2 /=0 AND M1 /= 0) then
                                        if (CAM2 >= 5) then
                                                CAM2 <= CAM2 - 5;
                                                DATAOUTM <= "0010";
                                                M5 <= M5 - 1;
                                        elsif (CAM2 >= 2) then
                                                CAM2 <= CAM2 - 2;
                                                DATAOUTM <="0100";
                                                M2 <= M2 - 1;
                                        elsif (CAM2 >= 1) then
```

```
                                        CAM2 <= CAM2 - 1;
                                        DATAOUTM <="1000";
                                        M1 <= M1 - 1;
                                end if;
                        elsif (M2 /=0 AND M1 /= 0) then
                                if (CAM2 >= 2) then
                                        CAM2 <= CAM2 - 2;
                                        DATAOUTM <="0100";
                                        M2 <= M2 - 1;
                                elsif (CAM2 >= 1) then
                                        CAM2 <= CAM2 - 1;
                                        DATAOUTM <="1000";
                                        M1 <= M1 - 1;
                                end if;
                        elsif (M1 /= 0) then
                                if (CAM2 >= 1) then
                                        CAM2 <= CAM2 - 1;
                                        DATAOUTM <="1000";
                                        M1 <= M1 - 1;
                                end if;
                        else
                                        MENSAJEM <= "01001";          --Deposito vacío FUERA DE SERVICIO
                                        DATAOUTM <= "0000";
                        end if;
                end loop;
        end if;
    else                                                              --HO HA INGRESADO DINERO
        MENSAJEM <= "00100"; --SALUDO
        DATAOUTM <= "0000";
    end if;
    end process;

    CAMM <= std_logic_vector(to_unsigned(CAM, CAMM'length));
    RESM <= std_logic_vector(to_unsigned(RES, RESM'length));
    --RETARDO A LEDS
    process(RELOJ50) begin --DIVISOR DE FRECUENCIA A 1HZ
        if RELOJ50'event and RELOJ50='1' then
            contador <= contador + '1';
        end if;
    end process;
    retardo <= contador(24); --FRECUENCIA A 1 HZ

    process(retardo) begin
        if retardo'event and retardo = '1' then
            DATAOUTLED <= DATAOUTM; -- SEÑAL RETARDADA
            DATAPWM <= '1';
        end if;
    end process;

end Behavioral;
```

CONTROL DE SERVOMOTOR PWM - **CÓDIGO VHDL**

```
--5.-SENAL PWM CON FRECUENCIA DE 50HZ PARA CONTROL DE SERVOMOTOR:
--Este proceso sirve para primero crear un divisor de reloj, donde se puede dictar al reloj en que frecuencia
--quiero que opere el servomotor, indicando así su velocidad de rotación, en este mismo proceso del divisor de reloj
--se crea un contador que creara la frecuencia de 50 Hz para la señal PWM, después lo que se hace es usar ese reloj
--modificado para crear otro contador que declare los pasos del estado en alto de la señal PWM, que durara de 1 a 2 ms
--para así mover el servomotor hacia varias posiciones donde debe parar durante una secuencia, que abarca posiciones de
--0 a 180 grados.

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--Librerías para poder usar el lenguaje VHDL.
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--Librería declarada para poder hacer operaciones matemáticas sin considerar el signo.

entity servomotorPWM is
    Port ( CLKNexys2 : in  STD_LOGIC;  --Entrada CLK de 50MHz del divisor de reloj.
           rst : in  STD_LOGIC;        --Botón de reset.
           PWM : out  STD_LOGIC);      --Señal PWM.
end servomotorPWM;

--Arquitectura: Aquí se hará el divisor de reloj y la señal PWM haciendo uso de una signal y condicionales if.
architecture frecuenciaNueva of servomotorPWM is
--SIGNAL: No es ni una entrada ni una salida porque no puede estar vinculada a ningún puerto de la NEXYS 2, solo
--existe durante la ejecución del código y sirve para poder almacenar algún valor, se debe declarar dentro de la
--arquitectura y antes de su begin, se le asignan valores con el símbolo :=
signal divisorDeReloj : STD_LOGIC_VECTOR (25 downto 0);    --Vector con el que se obtiene el divisor de reloj.
--Esta signal sirve para que podamos obtener una gran gama de frecuencias indicadas en la tabla del divisor de reloj
--dependiendo de la coordenada que elijamos tomar del vector, para asignársela al contador que dicta la velocidad del
--servomotor cuando va de una posición a otra.
signal conteoFrecuenciaPWM: integer range 1 to 1e6 := 1; --Signal para el contador de uno a 1,000,000.
--Variable que obtiene un conteo para obtener la senal de 50 Hz que manda pulsos al servomotor con el fin de que
--llegue a posiciones de 0 a 180 grados, esta tiene un periodo de T = 1/50 = 0.02s = 2ms, aunque después de mandar
--estos pulsos, puede tener un tiempo de espera antes de mandar el otro y eso es lo que dicta la velocidad de rotación
--que lleva al llegar de una posición a otra.
--El valor del conteo se obtiene al dividir el periodo de 20ms sobre el periodo de la señal de 50MHz proveniente del
--reloj de la Nexys 2: T50Mz = 1/50,000,000 = 20e-9s = 20ns. Por lo tanto, al realizar la operación se obtiene que:
--T50Hz = 0.02s = 2ms; T50Mz = 20e-9s = 20ns; ConteoFrecuenciaPWM = T50Hz/T50Mz = 2ms/20ns = 2e-3/2e-9 = 1,000,000.
signal selectorDutyCycle: integer range 0 to 2 := 0;       --Signal para el selector de posiciones de la señal PWM.
--La signal selectorDutyCycle cuenta de 0 a 2 porque crea una secuencia de 3 posiciones para el servomotor, que en este
--caso lo posiciona en 0, 90 y 180 grados consecutivamente, haciendo avanzar el estado en alto de la señal PWM de 1ms
--a 1.5ms y finalmente a 2ms.
```

```
signal DutyCycle_1a2ms: integer;                                                                    --
Crea el pulso en alto para la secuencia de 1 a 2ms.
--Guarda el valor del contador selectorDutyCycle que creara una secuencia de posiciones y lo multiplica por cierto
--numero de pasos, que en teoría son 50,000 para alcanzar un máximo de 1ms en el estado alto del duty cycle. El numero
--de pasos necesario para la multiplicacion se obtiene al dividir el tiempo de duración del pulso en alto entre el
--periodo de la señal de 50MHz obtenida del reloj de la Nexys 2, la operación es la siguiente:
--T1ms = 1ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleEnAlto = T1ms /T50Mz = 1e-3/2e-9 = 50,000.

begin
        --A los process se les puede dar un nombre diferente para diferenciarlos entre sí, se hace esto porque hay
        --programas de VHDL donde se usan varios process y de esta forma es más fácil no confundir uno con otro, la
        --sintaxis es la siguiente:
        --nombreProcess : process(Entradas o signals que se usan en el bucle, condicional u operación matemática) begin
        --                 Contenido del process.
        --end process;
        clkdiv: process(CLKNexys2, rst) begin
                --Cuando el botón Reset sea presionado valdrá 1 lógico y el divisor de reloj se reiniciará.
                if(rst = '1') then
                        divisorDeReloj <= "00000000000000000000000000";
                elsif(conteoFrecuenciaPWM > 1e6) then
                        --Se reinicia el conteo para obtener la frecuencia de 50 Hz de la señal PWM cuando se llegue
                        --al tope calculado anteriormente, donde: ConteoFrecuenciaPWM = T50Hz/T50Mz = 2ms/20ns =
                        --2e-3/2e-9 = 1,000,000 = 1e6.
                        conteoFrecuenciaPWM <= 1;
                elsif(rising_edge(CLKNexys2)) then
                        --La instrucción rising_edge() hace que este condicional solo se ejecute cuando ocurra un
                        --flanco de subida en la senal de reloj clkNexys2 proveniente de la NEXYS 2.
                        divisorDeReloj <= divisorDeReloj + 1;           --Esto crea al divisor de reloj.
                        --Esto crea la frecuencia de 50 Hz para la señal PWM.
                        conteoFrecuenciaPWM <= conteoFrecuenciaPWM + 1;
                end if;
        end process clkdiv;

        --Debo asignar el contenido de una coordenada de la signal divisorDeReloj a salidaReloj para obtener una
        --frecuencia en específico, cada coordenada del vector corresponde a una frecuencia en la tabla del divisor de
        --reloj y asignara cierta velocidad de rotación al servomotor cuando se dirige de una posición a otra.
        --El servomotor SG90 (chiquito azul) puede recibir una frecuencia mínima de 0.4 Hz y máxima de 2.8Hz, esto nos
        --da la posibilidad de elegir de la coordenada 24 (1.49 Hz) a la coordenada 26 (0.3725 Hz) que ya no esta
        --incluida en la tabla del divisor de reloj. La frecuencia mínima no esta tan definida, puede ser la que sea,
        --aquí está limitada por la tabla del divisor.
        highDutyCyclePWM: process(divisorDeReloj(24)) begin
                if(rising_edge(divisorDeReloj(24))) then
                        --La instrucción rising_edge() hace que este condicional solo se ejecute cuando ocurra un
                        --flanco de subida en la senal del divisorDeReloj en su coordenada 24, que proporciona una
                        --frecuencia de 1.49Hz.
                        if(selectorDutyCycle = 2) then
                                --Cuando el conteo del duty cycle en estado activo (1 lógico), que se da con una
                                --frecuencia de 1.49Hz, valga 2, se reiniciará el conteo, regresando el selector y
                                --servomotor a su posición inicial, ya que en este caso se eligen los tiempos de 0,
                                --0.5 y 1 ms, correspondientes a las posiciones 0, 90 y 180 grados.
                                selectorDutyCycle <= 0;
                        else
                                --Crea el conteo del estado en alto de la señal PWM con frecuencia de 50 Hz para
                                --que alcance las duraciones de 1 a 2ms, paulatinamente, creando una secuencia de
                                --posiciones que va de 0 a 90 grados, luego de 90 a 180 y regresa después a la de 0
                                --grados.
                                selectorDutyCycle <= selectorDutyCycle + 1;
                        end if;
                end if;
                --Con la variable DutyCycle_1a2ms se realiza el avance del pulso en la señal PWM de 1 a 2ms de forma
                --paulatina, creando la secuencia de posiciones, esto se hace multiplicando el número del selector por
                --un numero de pasos, para ello se toma en cuenta que el número de pasos necesario para la
                --multiplicación se obtiene al dividir el tiempo de duración del pulso en alto entre el periodo de la
                --señal de 50MHz obtenida del reloj de la Nexys 2, la operación es la siguiente:
                --T1ms = 1ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleEnAlto = T1ms /T50Mz = 1e-3/2e-9 = 50,000.
                --Esta es la teoría, pero los pasos del servomotor no siempre siguen esto al pie de la letra, por lo
                --que se debe calibrar el servomotor, viendo por cada número de pasos, la posición que alcanza.
                --AL REALIZAR LA CALIBRACION DEL SERVOMOTOR SE OBTIENE LO SIGUIENTE:
                --El servomotor SG90 (chiquito azul) realiza un giro de 0 a 180 grados con un paso de 92,109.
                --El servomotor SG90 (chiquito azul) realiza un giro de 0, 90 y 180 grados con un paso de 41,868.
                --Al realizar la operación de 92109/2 = 46054, podemos ver que las posiciones no son completamente
                --lineales, o al menos no tienen una pendiente de m = 1.
                --Este es un código de ejemplo, pero si en verdad se quisiera hacer una secuencia con un control super
                --preciso se tendría que ver el paso exacto que se debe dar para cada posición y luego crear un bucle
                --case que lleve a cada posición dicha secuencia.
                DutyCycle_1a2ms <= 46054 * selectorDutyCycle ; --25e3*0 = 0ms; 25e3*1 = 0.5ms; 25,000*2 = 50,000 = 1ms;
                --Se debe tomar en cuenta que, al realizar diferentes secuencias, se debe cambiar de igual manera el
                --numero de pasos, dividiendo el número de pasos que llega de 0 a 180 grados, osea 89,550, entre el
                --número de acciones de la secuencia, considerando además que, al terminar la secuencia, el motor
                --vuelve a su posición inicial de 0 grados.
        end process highDutyCyclePWM;

        --Para mantener el estado alto de la señal PWM de 1 a 2 ms con el fin de mover el servomotor de 0 a 180 grados,
        --se debe realizar un conteo como el que se realizó para crear la frecuencia de 50 Hz en la señal PWM, para ello
        --se realiza el siguiente calculo:
        --T1ms =1ms;  T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1ms/20ns   = 1e-3/20e-9   = 50,000;  0 grados servo.
        --T1ms =1.5ms;T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 1.5ms/20ns = 1.5e-3/20e-9 = 75,000;  90 grados servo.
        --T1ms =2ms; T50Mz = 20e-9s = 20ns; ConteoDutyCycleAlto = 2ms/20ns   = 2e-3/20e-9   = 100,000; 180 grados servo.
        --Con este cálculo podemos saber que para que el pulso en alto abarque un rango inicial de 1ms se debe usar 50e3
        --pasos, a este ancho inicial de pulso se le debe sumar el valor de la signal DutyCycle_1a2ms para que el pulso
        --paulatinamente vaya aumentando de 1 a 2ms, creando así la secuencia que va a las posiciones 0, 90 y 180°.
        --Pero recordemos que esta es la teoría, para alcanzar dichas posiciones se debe realizar una calibración, donde
        --veremos el número de pasos reales para alcanzar cada posición.
        --AL REALIZAR LA CALIBRACION DEL SERVOMOTOR SE OBTIENE LO SIGUIENTE:
        --El servomotor SG90 alcanza un rango de 1ms con un paso inicial de 26e3.
```

```
        pwmSignal: process(conteoFrecuenciaPWM, DutyCycle_1a2ms)begin
                --Este if crea el estado en alto y bajo de la señal PWM que crea la secuencia, yendo de 0 a 90 y 180°.
                if(conteoFrecuenciaPWM <= (26e3 + DutyCycle_1a2ms)) then
                        PWM <= '1';
                else
                        PWM <= '0';
                end if;
        end process pwmSignal;

end frecuenciaNueva;
```

UCF NEXYS 2 - **CÓDIGO VHDL**

```
//ENTRADAS DEL MODULO TLD:
//SWITCHES DE LA NEXYS 2: ON/OFF
net "on_off" loc = "R17";//Encendido y apagado de toda la maquina conectado al switch SW7

//SWITCHES PARA SELECCIONAR UN PRODUCTO
net "selectorArticulo[2]" loc = "D18";//Selecciono el articulo 3 con el push button BTN1.
net "selectorArticulo[1]" loc = "E18";//Selecciono el articulo 2 con el push button BTN2.
net "selectorArticulo[0]" loc = "H13";//Selecciono el articulo 1 con el push button BTN3.

//SWITCH PARA REPRESENTAR AL SENSOR LDR QUE INDICA LA DISPONIBILIDAD DE PRODUCTO
net "productoDisponible" loc = "N17";//Bit del sensor LDR representado por el switch SW6.

//ENTRADAS DEL MDULO divisorDeReloj
net "CLK" loc = "B8" ;//El reloj de 50MHz viene del puerto B8.


//SALIDAS DEL MODULO decodificadorBinHex
//Leds A,B,C,D,E,F y G del display de 7 segmentos
net "ledsDisplay7Seg[6]" loc = "L18";//1er bit del vector conectado al puerto CA (led A) del display de 7 segmentos.
net "ledsDisplay7Seg[5]" loc = "F18";//2do bit del vector conectado al puerto CB (led B) del display de 7 segmentos.
net "ledsDisplay7Seg[4]" loc = "D17";//3er bit del vector conectado al puerto CC (led C) del display de 7 segmentos.
net "ledsDisplay7Seg[3]" loc = "D16";//4to bit del vector conectado al puerto CD (led D) del display de 7 segmentos.
net "ledsDisplay7Seg[2]" loc = "G14";//5to bit del vector conectado al puerto CE (led E) del display de 7 segmentos.
net "ledsDisplay7Seg[1]" loc = "J17";//6to bit del vector conectado al puerto CF (led F) del display de 7 segmentos.
net "ledsDisplay7Seg[0]" loc = "H14";//7mo bit del vector conectado al puerto CG (led G) del display de 7 segmentos.

//Punto de los displays de 7 segmentos.
net "DP" loc = "C17";//Punto del display de 7 segmentos.

//SALIDAS DEL MODULO decodificadorBinHex
//Para que el orden que puse en el código se respete debo conectar el bit más significativo al nodo AN3 y el bit
//menos significativo al nodo AN0.
net "anodoComun[0]" loc = "F17";
//Bit menos singnificativo conectado al puerto AN0, 1er display de 7 segmentos de los 4 disponibles.
net "anodoComun[1]" loc = "H17";
//Conectado al puerto AN1, 2do display de 7 segmentos de los 4 disponibles.
net "anodoComun[2]" loc = "C18";
//Conectado al puerto AN2, 3er display de 7 segmentos de los 4 disponibles.
net "anodoComun[3]" loc = "F15";
//Bit mas singnificativo conectado al puerto AN3, 4to display de 7 segmentos de los 4 disponibles.

//Se indica si hay productos disponibles o no con el switch SW6.
net "disponible" loc = "J14";//el LED LD0 se enciende cuando hay productos disponibles y se apaga cuando no

//SALIDA DEL MODULO servomotorPWM

net "PWM_Servo" loc = "L15";//La señal PWM del servomotor está conectada al puerto JA1.
```

SISTEMAS ELECTRÓNICOS Y DE POTENCIA

La alimentación suministrada a la maquina se realiza mediante una tensión de entrada alterna de 127 VAC, después esta alimentación se rectifica y se entrega de forma individual a cada uno de sus componentes con una fuente de voltaje de 12 VDC/48 W (ilustración 10).

La lista de los componentes utilizados se muestra en la tabla 5.

El diagrama esquemático de alimentación proporcionado a los dispositivos eléctricos de potencia a 127 VAC se muestra en la ilustración 11, mostrando la fuente de alimentación de 127 VAC – 12 VDC (X2), una lampara de iluminación para los productos (X3) y una segunda lampara que se activa cuando el producto está listo para ser recogido por el cliente (X1), la cual se activa desde uno de los contactos normalmente abiertos de un relevador de 5V, activado por el sistema de control lógico programado en la tarjeta de desarrollo Nexys 2.

Ilustración 10 Fuente de alimentación de 12V a 48W.

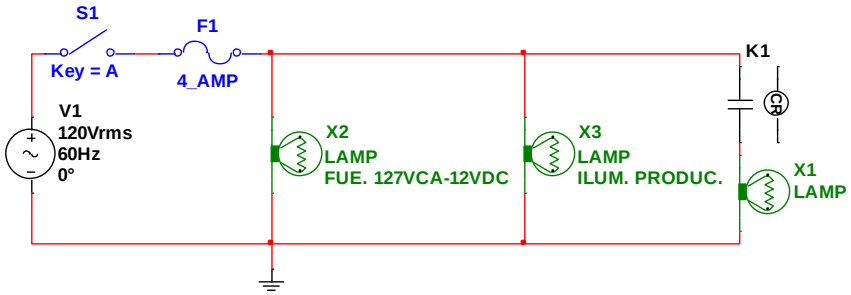


Ilustración 11 Dispositivos alimentados a 127 VAC.

Para el accionamiento de los motores a pasos se utilizó un controlador A4988, el cual recibe los pulsos de la FPGA, y esta alimentada a 12 VDC por la fuente de alimentación. En el controlador de motor, los pines STEP y DIR van conectados a las salidas digitales de la tarjeta Nexys, asignadas en el archivo UCF y la conexión de cada uno de los tres motores a pasos se muestra en la ilustración 12.

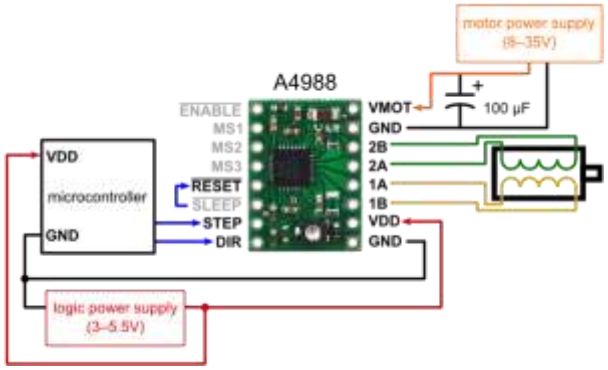


Ilustración 12 Conexión del controlador A4988 para motores a pasos bipolares.

El diseño implementado para el depósito de monedas se realiza de forma "automática" por decirlo de alguna forma, ya que las monedas caen por gravedad y se ordenan en un separador que las acomoda en función de sus dimensiones, pero para la devolución de monedas, es decir, "entregar el cambio", se hace uso de actuadores electromagnéticos (ilustración 13) que empujan la moneda deseada dependiendo de la salida asignada por la tarjeta, el eje del actuador está sujeto a la extensión del vástago (pieza 11) mostrado en la ilustración 5.

Ilustración 13 actuador lineal electromagnético.

Para la activación de estos dispositivos se hace uso de un módulo de relevadores, mediante una señal de emitida por la tarjeta se activa cada uno de los actuadores, separando así la etapa de potencia y etapa lógica de control, la conexión de los relevadores se muestra en el diagrama esquemático de la ilustración 14.

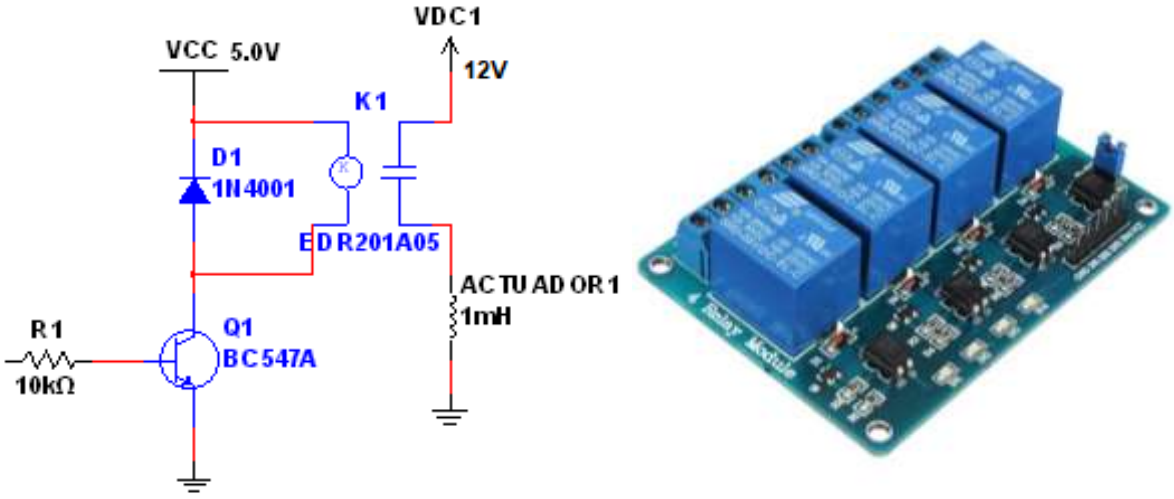


Ilustración 14 Conexión interna de los módulos de relevadores y módulo de relevadores.

El aceptador de monedas (Ilustración 15) nos permite añadir a la máquina expendedora la capacidad de cobro por medio de la detección de hasta 6 tipos de monedas. Los sensores detectan el grosor, diámetro y tiempo de caída de cada moneda para identificarlas y el dispositivo reporta el tipo de moneda mediante pulsos.

Este receptor de monedas puede aceptar hasta 6 tipos diferentes de monedas al mismo tiempo y es ampliamente utilizado en máquinas expendedoras, juegos de árcade, teléfonos públicos, etc.

Ilustración 15 Sensor aceptador de monedas.

El sensor de monedas genera una señal serial, la cual es recibida por la tarjeta, donde se decodifica el valor ingresado para asignarle un valor lógico que se utiliza en la programación de la máquina expendedora. La conexión a la tarjeta se realiza a través de un cable serial DB9, conectado al puerto serial de la tarjeta Nexys II (Ilustración 16).

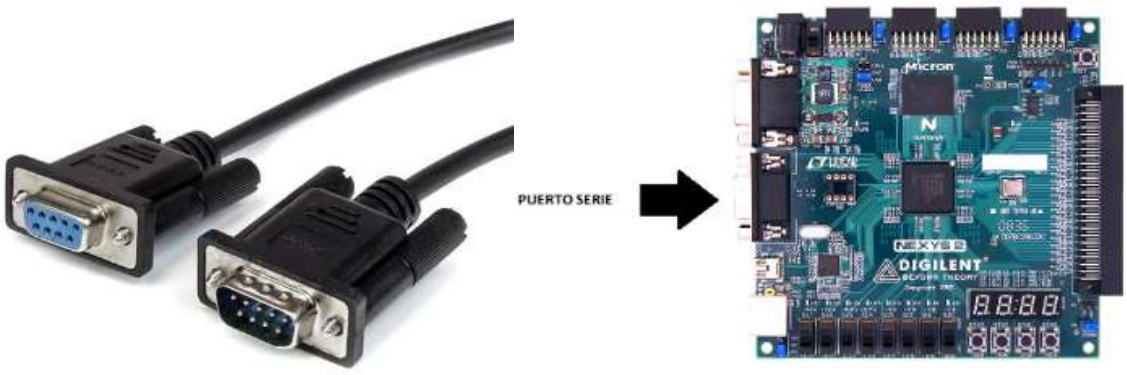


Ilustración 16 Conexión serial del sensor receptor de monedas.

MATERIALES Y COSTOS

ESTRUCTURALES

Los materiales y costos, utilizados en este prototipo se muestran en el día 25 de junio de 2020, obtenidos del proveedor (METALES DÍAZ, 2020), para el diseño estructural y mecánico, como se indica en la tabla 4.

MATERIAL	DESCRIPCIÓN	#	C/U	C/T
Lamina laminada en frío	Calibre 11 1220x2440 mm	1	1053.78	1053.78
Lamina cold rolled	Calibre 14 1220x2440 mm	1	663.31	663.31
Lamina cold rolled	Calibre 16 1220x2440 mm	1	548.21	548.21
Lamina de acero inoxidable	304, calibre 14 1220x2440 mm	1	897.72	897.72
Lamina de acero inoxidable	304, calibre 18 1220x2440 mm	1	682.52	682.52
Lamina de acero inoxidable	304, calibre 20 1220x2440 mm	1	560.57	560.57
Aluminio 6063 T6	Solera de 2"x5"	1	492	492
Tubo cuadrado de CR	1", cal. 16, 6m	1	117.94	117.94
Policarbonato de 6 mm	477x367 mm	1	638	638
Hélice plástica	3 mm de espesor, 390 mm de longitud x10 vueltas	3	380.33	1140.99
Tapa	Acceso a producto 80x150 mm	1	89.71	89.71
			TOTAL	**6884.75**

Tabla 4 Materiales estructurales

ELÉCTRICOS Y ELECTRÓNICOS

Los materiales usados para el desarrollo de los sistemas de control e interconexión dispositivos se obtuvieron de diversas fuentes, como se indica al final de este documento, siendo los materiales utilizados, los que se muestran en la tabla 5.

Material	Descripción	#	Costo unitario	Costo total
Actuador electromagnético	9V, 10mm, 50g	4	68.73	274.92
Motor a pasos	Nema 17 17hs4023, 0.7A, 14kg/cm	3	178.31	534.93
DRIVER A4988	Imax = 2A, Vmax = 35V	3	19.16	57.48
Módulo de relevadores	4 relevadores 5VDC con optoacoplador	1	59	59
fuente de alimentación	12V, 48W	1	132.88	132.88
LCD 16X2	LCD 16X2	1	36.91	36.91
Botón pulsador	22mm, NA	3	34.8	104.4
Interruptor	2 polos, 1 tiro	1	5	5
sensor aceptador de monedas	6 monedas, 12V, 65mA	1	860	860
Lampara LED dicroico	E26, 24W, 127V	2	70	140
Nexys II	FPGA Xilinx Spartan 3E.	1	6032.2	6032.2
Varios	Cable, conectores, etc.	1	45	45
			TOTAL	**8282.72**

Tabla 5 Materiales eléctricos y electrónicos

CONCLUSIONES

Las circunstancias por las que estamos pasando nos han llevado a buscar nuevas técnicas de aprendizaje enfocadas al enfoque autodidacta, las cuales nos permitan obtener los conocimientos necesarios y medios para poder desarrollar este tipo de proyectos apoyados más que nada en simulaciones, para que el costo sea mínimo, sin la necesidad de salir de casa. El desarrollo de este proyecto nos planteó demasiados retos, no solo de diseño, programación o electrónica; sino también de comprensión de términos y funcionamiento de las operaciones implementadas en VHDL, ya que la lógica de programación convencional no aplica en lenguajes descriptivos de hardware y no se cuenta con una extensa comunidad, por lo que en internet no hay mucha información de varios temas.

Uno de los retos más grandes, fue poner a prueba el código de programación del sistema de control, ya que al no contar con las herramientas necesarias para poder realizar pruebas físicas y determinar si lo que hacíamos estaba siendo realizado de la forma correcta, por lo que se tuvo que depender de terceros para verificar su funcionamiento.

En el desarrollo del prototipo de la máquina expendedora de botellas de gel antibacterial, se hizo uso de algunos controladores externos para facilitar la conexión y el control de los dispositivos, además de proporcionar con mayor fuerza los actuadores de la máquina, porque los motores bipolares aguantan muchísima más carga que los unipolares, pero estos necesitan un controlador de motor externo, además, como la máquina es de dimensiones reducidas, se tienen que implementar mecanismos y dispositivos pequeños.

El diseño estructural y mecánico de la máquina, no represento una complicación tan grande, desde el diseño conceptual hasta el modelado de las piezas mecánicas y los planos nos llevó poco más de 2 días al hacerlo en equipo; tuvimos problemáticas al encontrar las cotizaciones de los materiales, ya que estos no son tan comunes como el de

los dispositivos electrónicos o eléctricos y para su consulta se requiere de contactar por correo electrónico a los proveedores, que tardan en responder hasta 7 días laborales.

GLOSARIO

SISTEMA:

> [RAE]
> Del lat. tardío systēma, y este del gr. σύστημα sýstēma.
> 1. m. Conjunto de reglas o principios sobre una materia racionalmente enlazados entre sí.
> 2. m. Conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto

De forma más general, podemos definir a un sistema como un arreglo, conjunto o combinación de cosas conectadas o relacionadas de manera que constituyen un todo.

De otra forma, podemos definirlo como un arreglo de componentes físicos conectados o relacionados de tal manera que formen una unidad completa o que puedan actuar como tal; en otras palabras: Un sistema es una combinación de componentes que actúan conjuntamente, con un determinado objetivo a cumplir.

Entrada de un sistema: Es una variable del sistema elegida de tal manera que se la utiliza como excitación del mismo.

Salida de un sistema: Es una variable del sistema elegida de tal modo que se la utiliza para analizar los efectos que produjo una excitación en la entrada de este.

CONTROL:

> [RAE]
> Del fr. contrôle.
> 1. m. Comprobación, inspección, fiscalización, intervención.
> 2. m. Dominio, mando, preponderancia.
> 5. m. Regulación, manual o automática, sobre un sistema.
> 7. m. Mando o dispositivo de regulación.

Esta palabra se usa para designar regulación, gobierno, dirección o comando.

SISTEMA DE CONTROL

Sistema de control: Es un arreglo de componentes físicos conectados de tal manera que el arreglo pueda comandar, dirigir o regular, asimismo o a otro sistema. Estos sistemas comandan dirigen o controlan dinámicamente.

Entrada de un sistema de control: Es una variable del sistema controlado que se elige de modo tal que mediante su manipulación se logra que el sistema cumpla un objetivo determinado.

Las variables de entrada son variables que ingresan al sistema y no dependen de ninguna otra variable interna del mismo.

No solo la señal de referencia (valor deseado de la salida del sistema) conforma una variable de entrada, también hay ciertas señales indeseadas, como son algunas perturbaciones externas, que se generan fuera del sistema y actúan sobre el sistema, afectando desfavorablemente la salida de este, comportándose también como una variable de entrada, cuyo valor no dependen de ninguna otra variable interna al sistema.

Salida de un sistema de control: Es una variable del sistema controlado que se elige de modo tal que mediante su estudio se analiza si el sistema cumple o no con los objetivos propuestos.

Realimentación: Propiedad de los sistemas que permiten que la salida del sistema o cualquier variable del mismo sea comparada con la entrada al sistema o con cualquier componente del sistema, de tal manera que pueda establecerse la acción de control apropiada entre la entrada y la salida.

SISTEMA COMBINACIONAL

Circuitos combinacionales: Son aquellos circuitos digitales con varias entradas y varias salidas, en los cuales la relación entre cada salida y las entradas puede ser expresada mediante una función lógica (expresiones algebraicas, tablas de verdad, circuito con puertas lógicas, etc.).

SISTEMA SECUENCIAL

Sistema Secuencial: Son aquellos en los cuales las salidas en un instante de tiempo determinado dependen de las entradas en ese instante y en instantes anteriores de tiempo.

Estado del Sistema: información almacenada

Número de estados posibles del sistema: Número máximo de informaciones almacenables.

Contador: Es un circuito secuencial construido a partir de biestables y puertas lógicas capaces de almacenar y contar los impulsos (a menudo relacionados con una señal de reloj), que recibe en la entrada destinada a tal efecto, así mismo también actúa como divisor de frecuencia

MODULACIÓN POR ANCHO DE PULSOS

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

Es una técnica de modulación escalar para generar una señal analógica utilizando una fuente digital. Una señal PWM consta de dos componentes principales que definen su comportamiento: un ciclo de trabajo y una frecuencia. El ciclo de trabajo describe la cantidad de tiempo que la señal está en un estado alto (encendido) como un porcentaje del tiempo total que se tarda en completar un ciclo. La frecuencia determina qué tan rápido el PWM completa un ciclo (es decir, 1000 Hz serían 1000 ciclos por segundo) y, por lo tanto, qué tan rápido cambia entre los estados alto y bajo. Al apagar y encender una señal digital a una velocidad suficientemente rápida, y con un cierto ciclo de trabajo, la salida parecerá comportarse como una señal analógica de voltaje constante cuando se suministra energía a los dispositivos.

REFERENCIAS

AliExpress. (25 de Junio de 2020). *KAYPW, fuente de alimentación conmutada*. Obtenido de https://es.aliexpress.com/item/33042313383.html?spm=a2g0o.productlist.0.0.5ffa6497V7dbeo&s=p&ad_pvid=202006251239415536308953309080006618444_1&algo_pvid=9032f157-76f2-42c1-95d3-580b8c33875d&algo_expid=9032f157-76f2-42c1-95d3-580b8c33875d-6&btsid=0ab6fb881

Ballardo, C. (06 de Marzo de 2020). Circuitos Lógicos. México, Ciudad de México, México: IPN.

Carrod electrónica. (25 de Junio de 2020). *Sensor Aceptador de Monedas Programable para 6 Tipos de Monedas*. Obtenido de https://www.carrod.mx/products/sensor-aceptador-de-monedas-programable-para-6-tipos-de-monedas

Cátedra de percepción y sistemas inteligentes. (s.f.). *Universidad del Valle*. Obtenido de Escuela de Ingeniería Eléctrica y Eléctronica: https://campusvirtual.univalle.edu.co/moodle/pluginfile.php/626094/mod_resource/content/1/Practica_compuertas.pdf

Digilent. (19 de Febrero de 2007). Digilent Nexys Board Reference Manual. Pullman, EUA.

Fonseca, E. (31 de Enero de 2011). *Diseño Digital Moderno*. Obtenido de https://bloganalisis1.files.wordpress.com/2011/01/secuenciador.pdf

Geek Factory. (25 de Junio de 2020). Obtenido de https://www.geekfactory.mx/

Gobierno de México. (22 de Junio de 2020). *Coronavirus*. Recuperado el 22 de Junio de 2020, de https://coronavirus.gob.mx/informacion-accesible/

LED CONTROLS. (25 de Junio de 2020). Obtenido de https://www.ledcontrols.com.mx/product/boton-pulsador-xb7-ea/

másluz. (25 de Junio de 2020). *DICROICOS*. Obtenido de LÁMPARA FOCO LED DICROICO E26 DE 4W A 127V: https://www.masluz.mx/lampara-foco-led-dicroico-e26-de-4w-a-127v/p

METALES DÍAZ. (22 de Junio de 2020). *METALES DÍAZ*. Obtenido de https://metalesdiaz.com/contacto/

DISEÑO MECÁNICO

Y

ESTRUCTURAL

EN ESTA SECCIÓN SE ANEXAN LOS PLANOS DE CADA UNA DE LAS PIEZAS, ELEMENTOS ESTRUCTURALES Y MECÁNICOS EMPLEADOS EN EL PROTOTIPO DE MAQUINA EXPENDEDORA DE BOTELLAS DE GEL ANTIBACTERIAL.