Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Carrera:	Ingeniería de sistemas digitales y robótica	Materia:	Sistemas embebidos
Nombre del alumno:	Carlos Jesús Salguero Rosales Jorge Martínez Hernández Ricardo García Sedano Sherlyn Quetzal López Hernández	Semestre:	Décimo
Nombre del docente:	Héctor Eduardo de Cos Cholula		
Práctica No.	8	Nombre de la práctica:	Control Maestro-Esclavo

Tema:_____ Control Maestro-Esclavo _____

Introducción:

Un sistema embebido es un sistema de computación que realiza funciones específicas marcadas por el cliente-usuario. Los sistemas embebidos son controlados mediante microcontroladores o microprocesadores. Estos sistemas suelen contener interfaces de entrada y salida para la comunicación entre el usuario y el software. Además, suelen ser conocidos como empotrados, incrustados o integrado. En este contexto, para el desarrollo de esta práctica, se iba a utilizar la tarjeta myRio 1900. No obstante, por algún motivo, a todos los equipos de la clase no les funcionó la tarjeta en la implementación física de esta práctica. Por lo tanto, solo se hará de manera virtual.

Como se mencionó anteriormente, solo se hará uso del software LabVIEW 2015. Este entorno de programación se utiliza en el desarrollo de sistemas de prueba automatizadas. Con el software, se realizará un prototipo de 10 botones (push buttons) que enciendan o apaguen una bomba, un ventilador, una puerta, una cámara o el estado de un ser humano. Si se presiona el push button, se encenderá o apagará un LED. Todo esto se realizará probando el método de control de maestro - esclavo. No obstante, ¿qué es el control maestro - esclavo?

El control Maestro - esclavo es un método de comunicación para dispositivos de hardware en el que uno de los dos o varios dispositivos tiene el control unidireccional sobre los demás dispositivos electrónicos. Es decir, un solo dispositivo controla a los demás dispositivos. Este método es usado a menudo en lugares donde un solo dispositivo actúa como controlador (maestro), mientras que los otros dispositivos son los que lo controlan (esclavos). Este tipo de modelos son usados en la industria tecnológica.

Ilustración 1.- Ejemplo del modelo de control maestro - esclavo.

Objetivo de la práctica:

En la práctica, se simulará vía software el modelo de control maestro – esclavo. Pondremos 10 push buttons, lo cuales controlarán 5 LED. Para cada LED, se le conectarán dos push button. Un push apagará el led, y el otro push button lo encenderá. Así será cada LED, el cual simulará el estado de un dispositivo diferente (Cámara, ventilador, puerta, bomba o el estado de un humano). No obstante, el objetivo principal de está práctica será que el alumno pueda comprender el uso del labview 2015 implementando el modelo del control maestro – esclavo. En este caso, solo se usará la computadora con el software antes mencionado.

Equipo necesario y material de apoyo:

El equipo necesario para esta práctica será:

1.- Laptop con LabVIEW 2015:

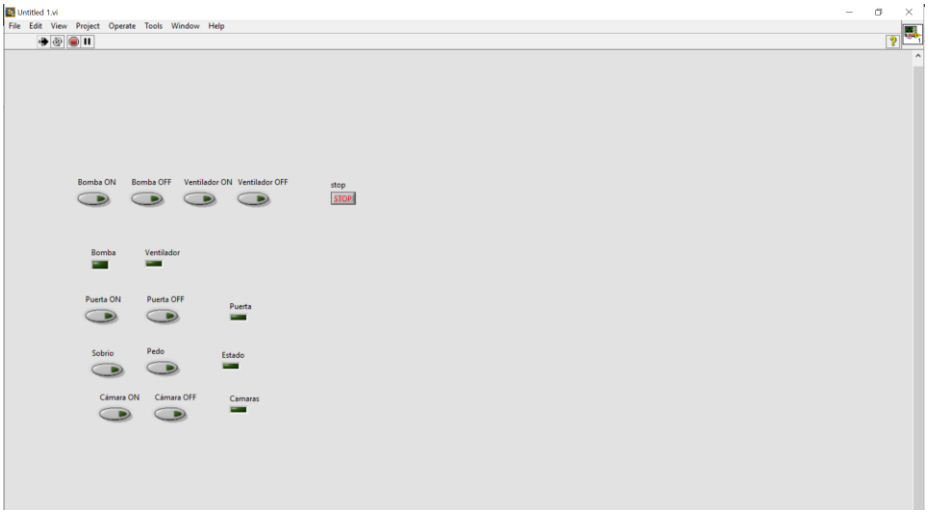

Ilustración 2.- Equipo de cómputo con LabVIEW 2015.

Procedimiento:

1. Abrimos el software LabView, y se creará un nuevo proyecto. En este nuevo proyecto, se le dará clic a la opción “File”, y se seleccionará “New VI”. De esta manera, se abrirá un nuevo panel frontal y un diagrama de bloques (Dos ventanas).
2. En el panel frontal, se agregarán 10 push buttons y 5 LED. Cada LED representará un dispositivo electrónico diferente. Por lo tanto, a cada uno de los LED en nuestro proyecto se le puso Bomba, Ventilador, Puerta, Estado de un humano y Cámara. Después, a dos de los 10 push button (en cada caso) se le nombrará como un dispositivo. No obstante, la diferencia será que a un push se le pondrá “ON” y al otro “OFF”. De esta manera, se le indica al usuario que un botón será para encender el LED, y el otro para apagarlo. Finalmente, se agregará un botón de “Stop”.

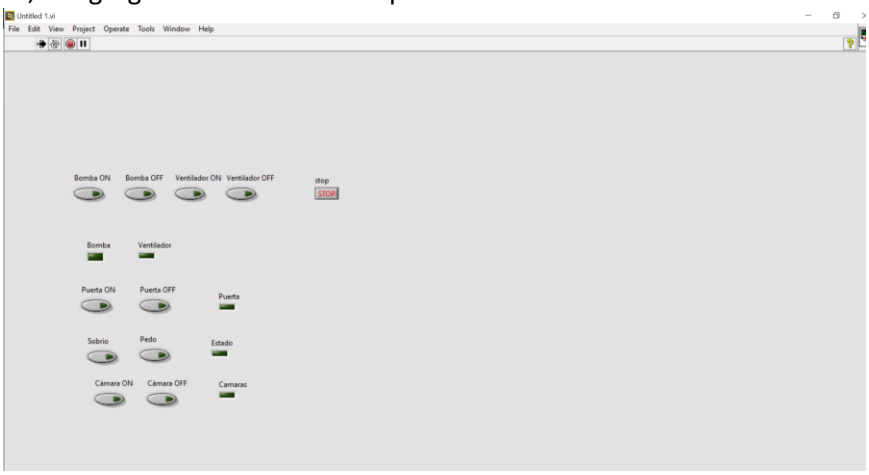

Ilustración 3.- Imagen que muestra cómo se verán los 10 push button y los 5 LED en el panel frontal.

3. Después, en la ventana del diagrama de bloques, se agregará dos “String constant”. A un se le pondrá “MAIN”, y al otro se le pondrá “Cadena de caracteres”. Los dos string constant serán conectados a una función “Obtain notifier”. Este obtain notifier se conectará a una estructura while loop. Dentro de está estructura, se agregará otra llamada “Event structure”. Ahora bien, dentro de la estructura while (No event) se agregará un OR.
4. Por el momento, es todo dentro del while. En la estructura evento, se agregarán más eventos dando clic derecho en el recuadro de en medio, y seleccionando la opción “Add Event”. Como se necesita que haya un apagador y un encendedor, se crearán 11 eventos. Serán 10 eventos (5 que enciendan un dispositivo y 5 que lo apaguen), y además se agregarán un evento llamado “STOP”, en el que indique si todos los dispositivos deben parar y reiniciar a un estado original.

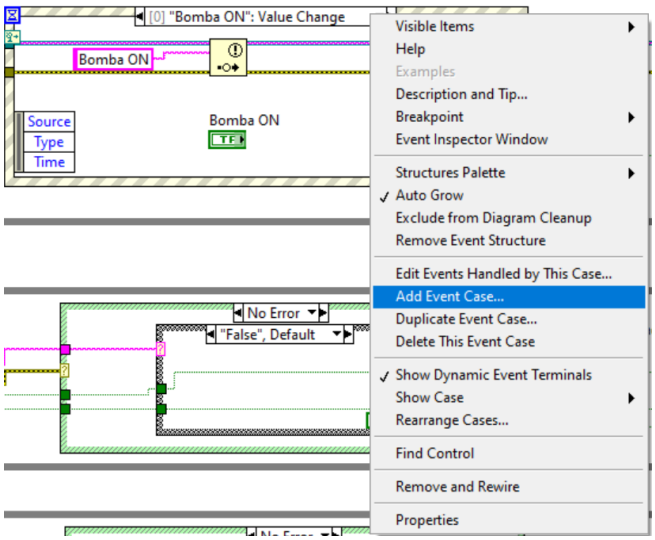

Ilustración 4.- Imagen que muestra que elección del clic derecho usar para agregar eventos en el "Event Structure".

5. Además, dentro de la estructura “Event structure” y dentro de cada evento (es decir, los 11) se agregará la función “Send notification”. Este se encargará de notificar si el dispositivo se encuentra en ON o en OFF (encendido o apagado) o STOP en caso del evento “Stop”. Se agregarán 10 false constant que se obtendrán dando clic derecho, y seleccionando la opción “Boolean”. Ahí se podrá elegir los false constant. Esta constante se agregará a los 10 eventos, excepto STOP. En este caso, se le agregará un True Constant. Los 10 false constant y el true constant irán conectados al OR afuera del event case que se agregó.
6. Dentro de cada evento, se agregarán los push buttons. De esta manera, se le indicará que push button se usará para encenderlo o apagarlo. Es posible hacerlos más chicos, si se selecciona la opción “View as icon”. De esta manera, dejará de verse como el icono push, y será más pequeño. Una vez que se acaba con esto, se agrega una función “Release Notifier”. Este se conectará con la función “Send notification”, que a su vez irá conectado con la función “Obtain Notifier”. Esto se hace, para que en caso de error, se avise que existe un error dentro de algunos de los dispositivos y pare. Recordar que esto debe hacer en cada uno de los 11 eventos. Con esto, se ha terminado con el control maestro. Ahora se seguirá con el esclavo.

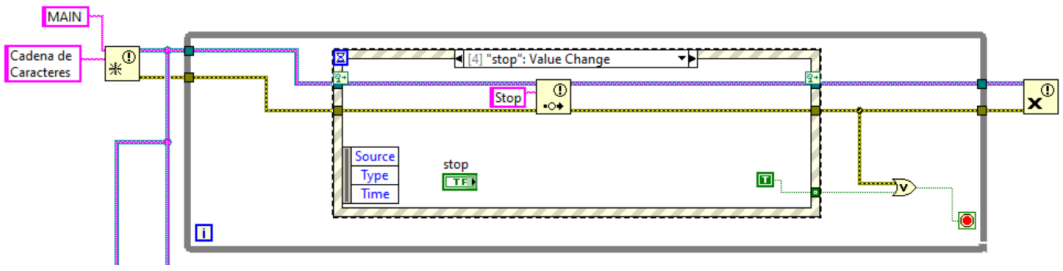

Ilustración 5.- Muestra al evento "STOP" con sus conexiones correspondientes.

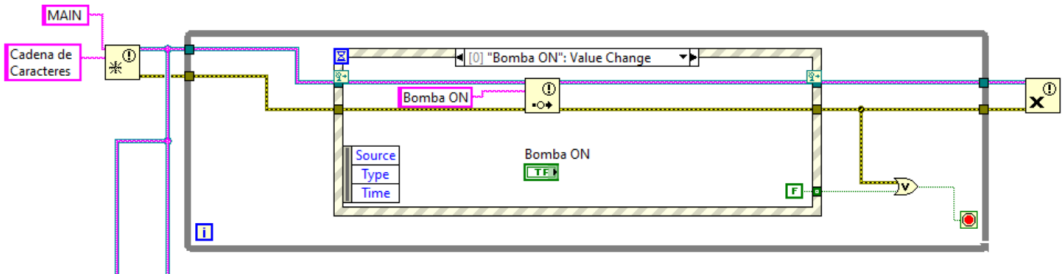

Ilustración 6.- La imagen muestra al evento "Bomba ON" con sus conexiones correspondientes. Tomar en cuenta que las conexiones de este evento son iguales a los de los demás 10 eventos (excepto STOP).

7. Ahora nos seguimos con los esclavos. Se crearán 5 estructuras while. En el caso de nuestro equipo, realizamos solo 4 estructuras, ya que, dentro de una sola estructura, se realizaron 4 eventos (Bomba ON, Bomba OFF, Ventilador ON y Ventilador OFF), además de los eventos False y STOP. No obstante, el método de conexión será lo mismo en los demás eventos que controlen a un específico dispositivo. Si hubieran sido las 5 estructura, en cada una se hubieran tenido que crear 2 eventos “ON-OFF”, y las False y stop.
8. Se explicará de la manera en que el equipo lo realizó. Por esa razón, en la primera estructura while, se agregará una función “Wait on Notifications”. Además, se agregará un Case structure, y dentro de este “Case Structure” se agregará otro. La función “Wait on notifications” se conectará a las dos estructuras case. Además, la función “Wait on Notifications” se conectará a la función “Obtain Notifier”. Dentro de la primera estructura case, se agregarán dos eventos: “Error” y “No Error”. El primer case indica que, en caso de error, todos los dispositivos permanecerán con la constante “False” (Es decir, apagados), y además, no habrá un segundo case.
9. En el caso del segundo, indica que trabajan con normalidad. Una vez explicado esto, pasamos al siguiente “Case Structure”. En este habrá 4 eventos: “False” (Nos indica que no hay cambios, y los dispositivos permanecerá apagados), “DISPOSITIVO ON” (Indica que el dispositivo se encenderá), “DISPOSITIVO OFF” (Indica que el dispositivo se apagará) y “Stop” (Indica que el dispositivo o dispositivos pararán, y reiniciarán a sus estados actuales). Además, se agregará un false constant a cada uno de los eventos, y se conectará al “stop”, excepto en el evento “Stop”. En este evento se agregará una constante True.

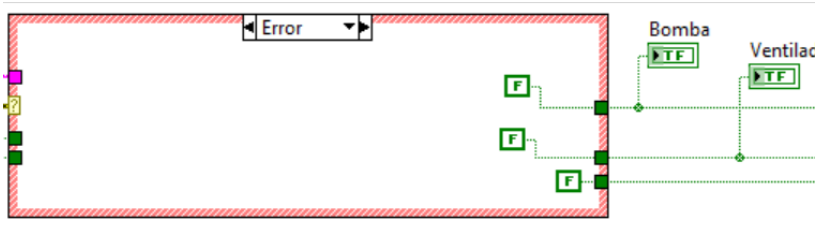

Ilustración 7.- En caso de error, los leds que indican que los dispositivos bomba y ventilador están trabajando, en este caso, estarán apagados por un "error".

10. En el while structure, se agregarán dos shift register (o uno, en el caso de que solo sea un solo dispositivo). Estos dos shift estarán conectados a un constante falsa. En el evento “False”, las conexiones pasarán normal, por lo tanto, los leds estarán apagados.

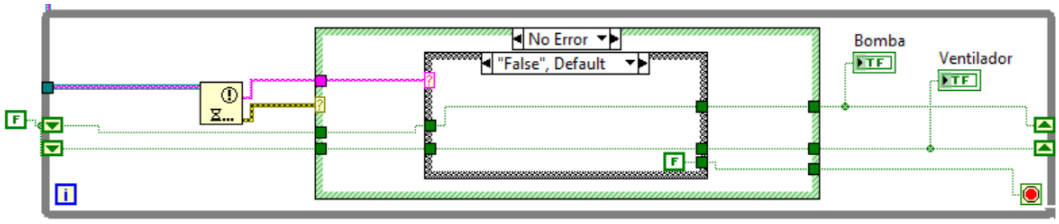

Ilustración 8.- Imagen que muestra el evento “False” con sus conexiones.

11. En el evento “Bomba On”, la conexión del primer shift se desconectará con el false constant. En lugar de eso, se agregará un “True constant”. El del ventilador permacerá en false. Esto se debe a que no se requiere encender el ventilador al mismo tiempo que la bomba, solo uno.

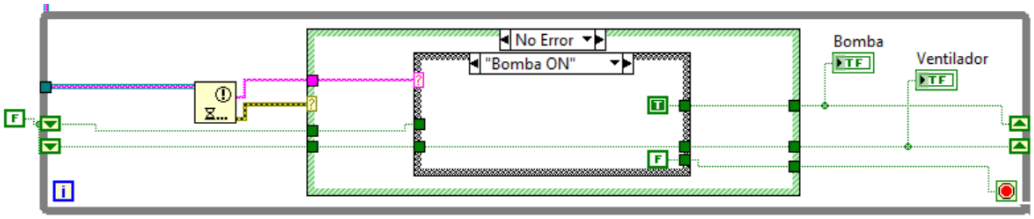

Ilustración 9.- Imagen que muestra el evento "Bomba ON" con sus conexiones.

12. En el evento “Bomba Off”, la conexión del primer shift se conectará a una constante false. Fuera de eso, no existirá otro cambio.

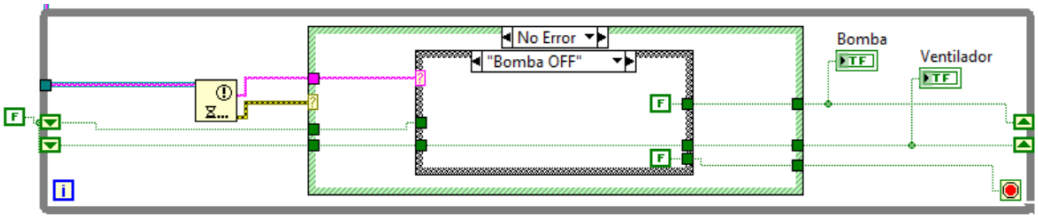

Ilustración 10.- Imagen que muestra el evento "Bomba OFF" con sus conexiones.

13. Finalmente, en el evento “Stop” permanecerá igual que el evento “false”. No obstante, la única diferencia aquí será que una constante True estará conectada al stop del while loop.

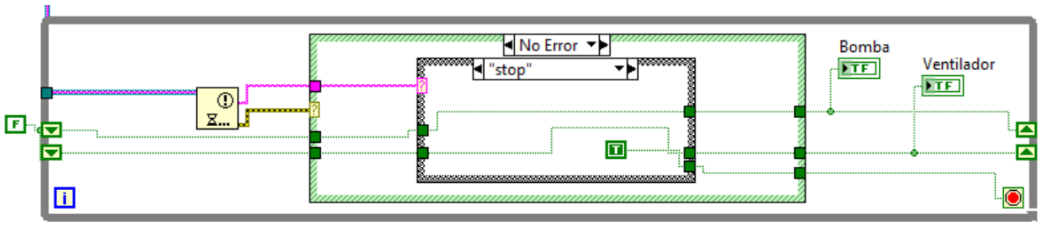

Ilustración 11.- Imagen que muestra el evento "Stop" con sus conexiones. Darse cuenta que es muy parecido al evento "false".

14. Una vez acabado esto, se replicará de la misma forma con cada uno de las otras cuatro (o tres en nuestro caso) estructuras while. En el caso de este equipo, se agregaron otros dos eventos de otro dispositivo en el primer while. Por lo tanto, en la primera estructura while hay 6 eventos (False, Bomba ON, Bomba OFF, Ventilador ON, Ventilador OFF, Stop). En las otras estructuras solo hay 4 (False, Stop, ON y OFF). De la misma forma en que se realizaron las conexiones para los eventos Bomba ON y Bomba OFF, se deberán replicar en los otros dispositivos. Con esto se habrá terminado con la práctica #8. Comprobar que las conexiones sean correctas.

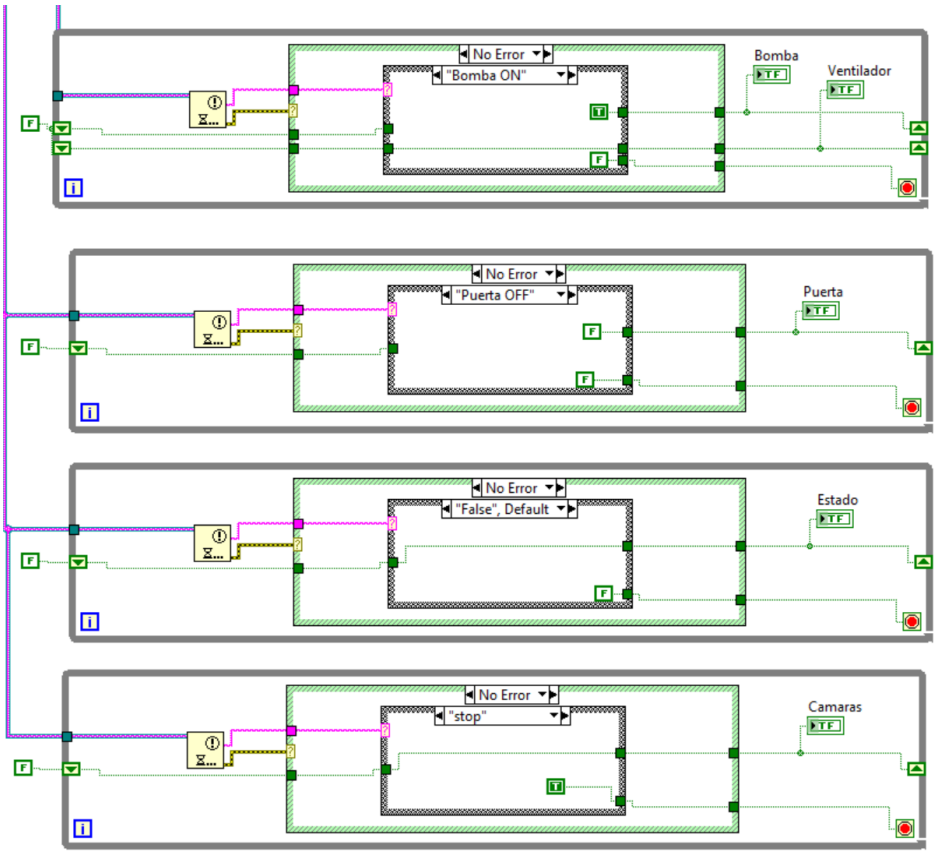

Ilustración 12.- Imagen que muestra un evento diferente en las cuatro estructuras while con sus conexiones.

Pruebas:

Simplemente, para comprobar que la práctica fue un éxito, se realizará un solo ejercicio con el led “Bomba”. Pulsaremos el botón “Bomba ON”, y se puede observar que, al hacerlo, el LED Bomba se encenderá. Esto indica que el dispositivo está encendido. Finalmente, al pulsar el botón “Bomba OFF”.

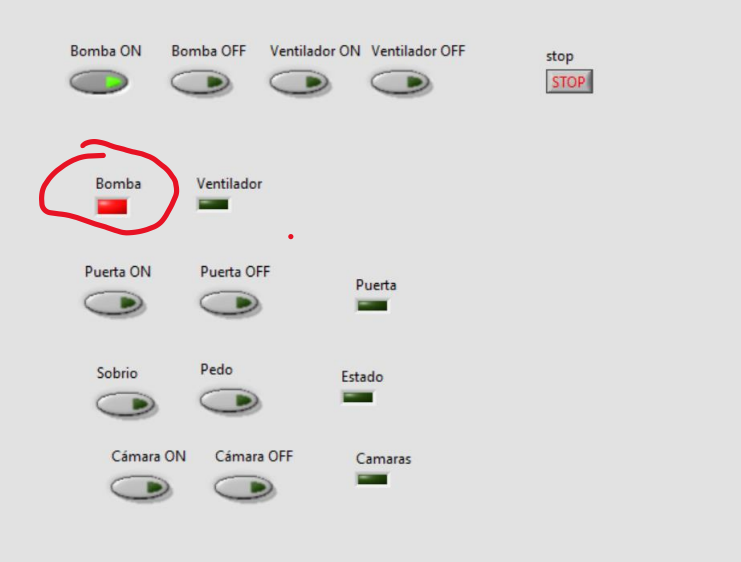

Ilustración 13.- La imagen muestra el resultado de pulsar el botón "Bomba ON".

Ilustración 14- La imagen muestra el resultado de pulsar el botón "Bomba OFF".

Resultados:

La siguiente fotografía muestra las condiciones iniciales de esta práctica. Se puede observar a ningún LED encendido. Esto es correcto, ya que, en caso de parar, los dispositivos regresan a sus condiciones iniciales; es decir, un “false”.

Ilustración 15.- Condiciones iniciales de la práctica 8

Además, dentro del diagrama de bloques, el programa comenzará en el evento cero (Bomba On). Es decir, dentro del maestro. No obstante, dependiendo del push button que se apriete, esté cambiará a un evento relacionado con su dispositivo.

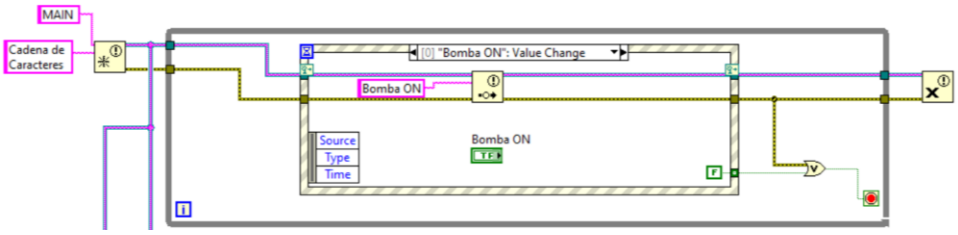

Ilustración 16.- Aunque en el maestro, está marcado el evento 0 como "Bomba On", en realidad este dependerá de qué push button se pulse.

En los esclavos, permanecerán en el evento “No error”, y dentro de ellos en la condición inicial “False” hasta que se pulse un botón “ON” o “OFF”.

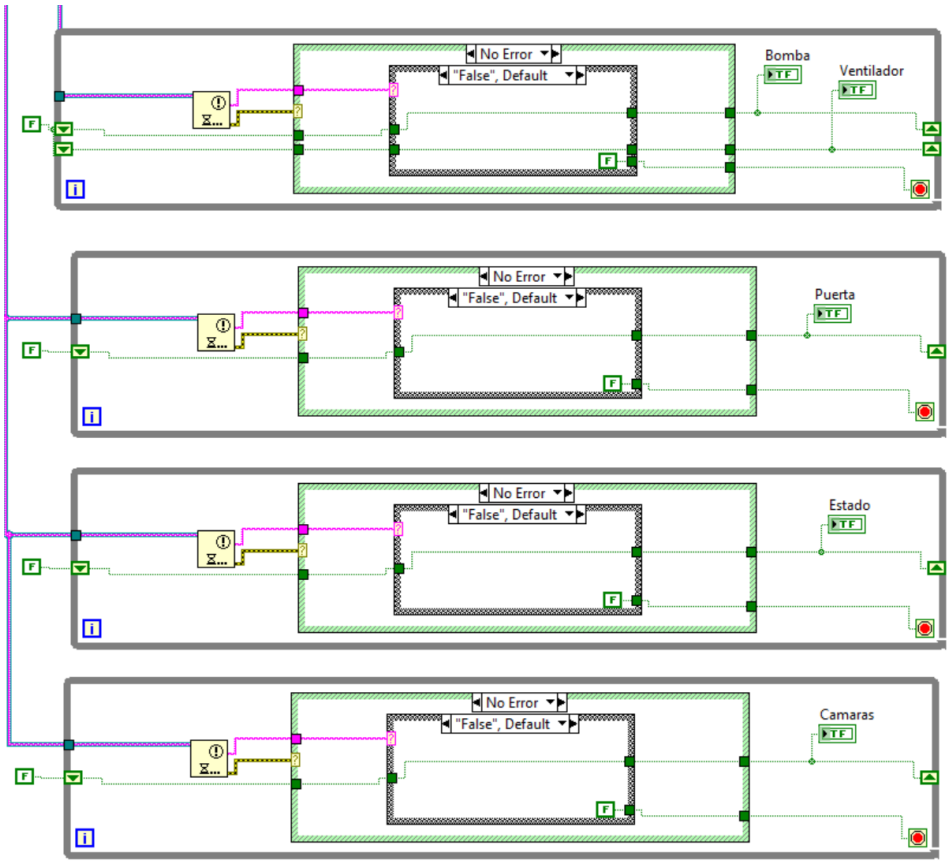

Ilustración 17.- Imagen que muestra a los esclavos con sus condiciones iniciales en el evento "No error" junto con "False".

La siguiente imagen comprobará que el resultado de esta práctica es un éxito al pulsar los botones “ON” de los dispositivos ventilador, puerta y cámara. Después de está imagen, habrá otra que muestre que al pulsar los botones “OFF” de los dispositivos ventilador, puerta y cámara, el LED de cada uno se apagará. También se pudo pulsar el botón “Stop” para apagar todos los LED encendidos.

Ilustración 18.- Imagen que muestra que al pulsar los botones "ON" de ventilador, Puerta y Cámara, sus respectivos LEDS encenderán.

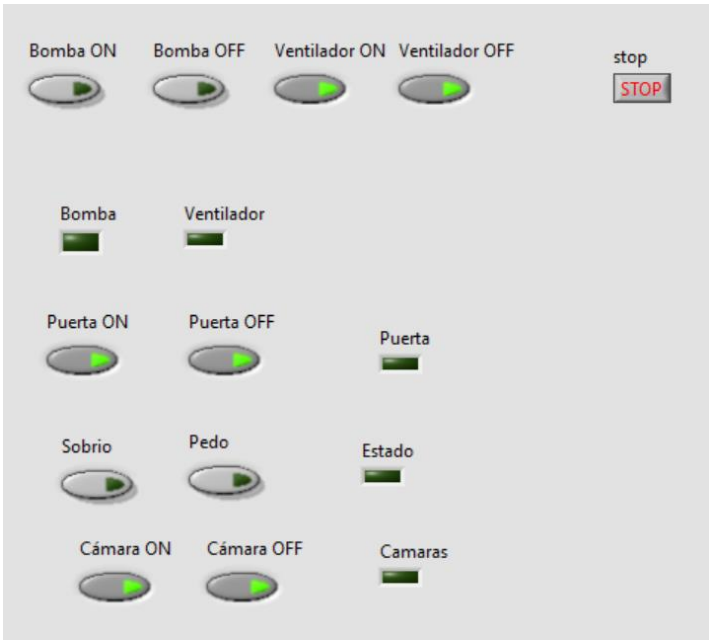

Ilustración 19.- Imagen que muestra que al pulsar los botones "OFF" de ventilador, Puerta y Cámara, sus respectivos LEDS se apagarán.

Link de la práctica: https://www.youtube.com/watch?v=IvuE9ieEwBM

Conclusiones y comentarios:

Como se mencionó anteriormente, el propósito de está práctica era que el estudiante pudiera implementar el modelo de control maestro – esclavo con unos simples leds y push buttons. Además, el estudiante podría aprender el funcionamiento de dos case anidados, así como de evento estructure y while loop. La obligación del equipo era constatar cómo se debía resolver. De esta manera, cada uno de los miembros podría obtener el aprendizaje deseado.

Por lo tanto, la evidencia mostrada anteriormente muestra que la práctica fue un éxito. Los estudiantes pudieron resolver otros ejemplos basados en el modelo de control de maestro – esclavo con lo que profesor mostró en la clase. Aunque las prácticas no consisten en repetir lo mismo que hace un profesor, cada uno de los miembros pudo resolver y aportar al equipo el entendimiento de la práctica para resolverlo. Actualmente, los miembros del equipo han mejorado su desarrollo en el software LabVIEW, por lo que sin lugar dudas, la práctica es un éxito.

Referencias:

- Lajara, J. & Pelegrí, J. (2007). Cap. 11 *Modelos de programación*. En LabVIEW. Entorno Gráfico de Programación. (pp. 268-270). Distrito Federal, México: ALFAOMEGA GRUPO EDITOR.
- Ollero, A. (2001). Cap. 13. *Sistemas Bilaterales Maestro – Esclavo*. En Robótica: Manipuladores y robots móviles. (pp. 403-406). Barcelona, España: MARCOMBO.
- Hurtado, R. (2016). Tutorial – *Comunicación Maestro/Esclavo: Parte 1 (Comunicación Serial).* BeetleCraft. https://beetlecraft.blogspot.com/2015/11/tutorial-comunicacion-maestro-esclavo.html