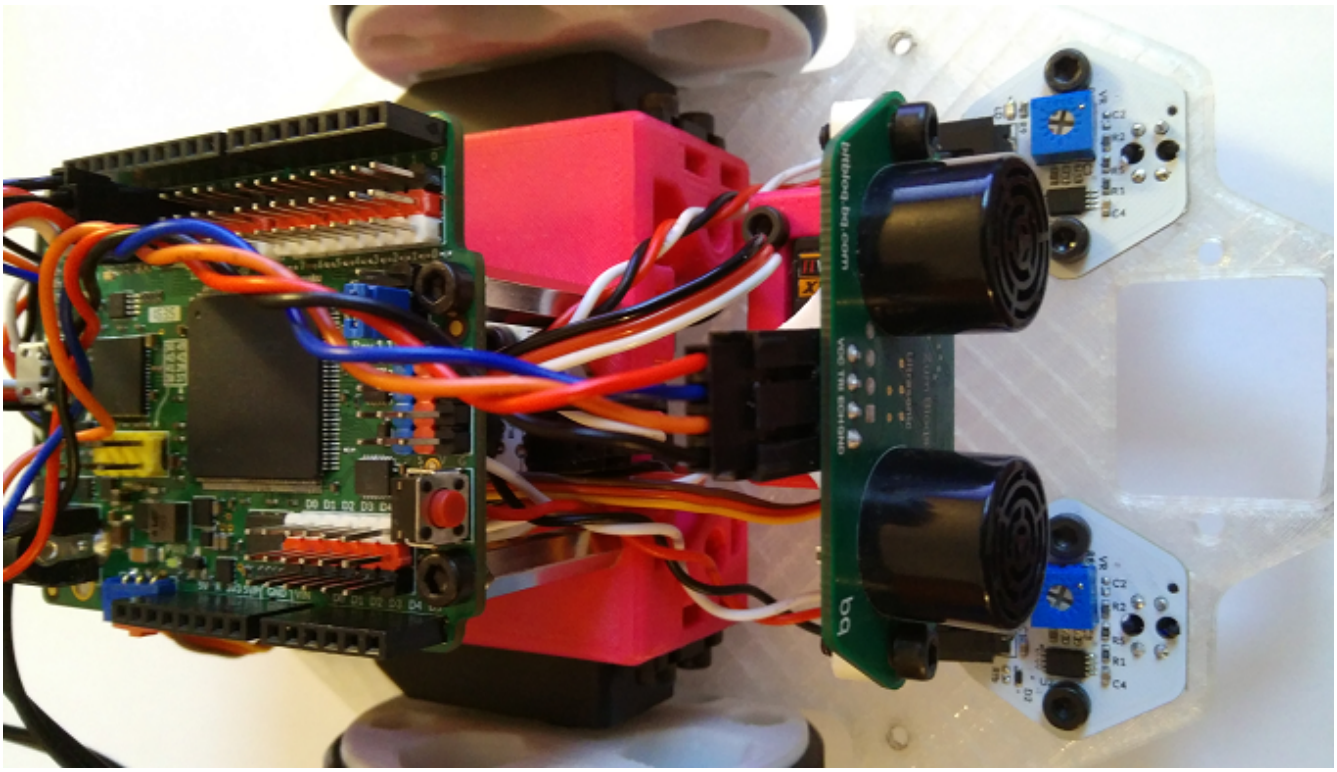


Proyecto : Control y Uso del Sensor de Ultrasonidos HCSR04 con las placas Icezum Alhambra y Alhambra II

Introducción.

Con esta revisión de mi proyecto sobre **Control y uso del sensor de ultrasonidos HCSR04 desde las placas FPGA Icezum Alhambra y Alhambra II** que ya publiqué hace unos meses , voy a intentar poner un poco de orden en la documentación que generé y os envié en su día , haciéndola más accesible a tod@s y quizás más didáctica para aquellos que se inician con nuestras placas FPGA Alhambra .

Para poder utilizar los bloques con los circuitos que he creado para controlar el sensor y visualizar los diferentes ejemplos prácticos que se incluyen en esta documentación , simplemente tenéis que descargar en vuestro ordenador esta carpeta como un archivo zip en el siguiente enlace :

https://github.com/makerventura/FPGA-Ultrasonidos-Sensor-HCSR04.git

Una vez realizado esto , abrid el programa **Icestudio** , os vais a **Herramientas > Colecciones > Añadir** y buscáis el archivo zip recién descargado para guardarlo en Icestudio .

Finalmente , para instalarla , hacéis lo siguiente : **Seleccionar > Colección** , seleccionándola entre las que os aparezcan guardadas .

Una vez instalada , como en cualquier otra colección de Icestudio , en la esquina superior derecha os aparecerán los bloques de código que he creado para es proyecto , mientras que en la parte izquierda , abriendo la pestaña **Archivo** , en las subcarpetas **bloques** y **Ejemplos** tendréis acceso respectivamente a cómo están construidos internamente los bloques de código y a los ejemplos documentados .

Pues ahora , sin más preámbulos , **allá vamos con los ultrasonidos !!**....

TEORÍA MOVIMIENTO DEL SONIDO A NIVEL DEL MAR , TEMPERATURA Y PRESION AMBIENTE

La **velocidad del sonido** es la dinámica de propagación de las ondas sonoras. En la atmósfera terrestre es de 343,2 m/s/1235,52 km/h(a 20 °C de temperatura, con 50 % de humedad y a nivel del mar) pero es variable en función del medio en el que se transmite.

Por simplicidad , para el grado de aproximación que requerimos en nuestro trabajo, podremos considerar en adelante lo siguiente :

V_{sonido} = 340 m/s

Tendremos en cuenta que la relación entre velocidad , espacio recorrido y tiempo empleado es V = e / t ,

y por tanto , denominando $\mathbf{e_r}$ al **espacio recorrido** , y suponiendo que queramos medir el espacio en mm recorrido por la onda del sonido en un cierto tiempo t , medido en microsegundos , nos quedará :

e_r (mm) = v_{sonido} * t (microsec)

e_r (mm) = 340 m/s * 10^3 mm/m * 1/ 10^6 sec/microsec * t (microsec) = 0.340 * t (microsec) mm

Como el espacio recorrido por la onda del sonido desde que sale del sensor US , rebota en el objeto y vuelve es el doble de la distancia al objeto :

D_{objeto} = 0.5 * e_r (mm) = 0.5 * 0.340 * t (microsec) mm = 0.17 * t (microsec) mm

Parámetros básicos generales de cómo funciona el sensor de ultrasonidos HCSR04.

En este pequeño cuaderno no pretendo hacer un análisis muy preciso de cómo funciona internamente este sensor , entre otras cosas porque yo no soy ningún entendido en la materia , así es que como para darle lecciones a nadie .

A mi me basta con entender este sensor electrónico como una especia de **caja negra a la cual le llegan unas señales y me devuelve una determinada información que a mi me toca interpretar y traducir** .

Y ese es justamente el punto de vista que yo pretendo que utilicéis para poder comprender lo que vendrá a continuación .

Antes de seguir , os dejo un enlace a la datasheet del sensor para que podáis revisarla en detalle :

https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf

Como podréis observar en la hoja técnica , nuestro sensor HCSR04 dispone de 4 pines denominados **Vcc , GND , Trig y Echo**.

Los dos primeros son para alimentar al sensor con Vcc a 5 V y GND a tierra , mientras que Trig y Echo son los que comandan el funcionamiento del mismo .

Veamos pues cómo funciona y cómo tenemos que actuar para comunicarnos con el sensor .

La verdad es que el procedimiento de trabajo con este sensor es muy sencillo :

1. Por el pin **Trig** del sensor le enviamos al mismo **un pulso de 10 µseg de duración** . Esta señal lo que hace es **activar o despertar al sensor**.
2. Al recibir esta señal , **envía al exterior un tren de 8 señales ultrasónicas o pulsos a 40 Khz de frecuencia y simultáneamente el pin Echo pasa de estado "0" a estado "1"**.
3. El sensor **se queda esperando a detectar la llegada del ECO de los pulsos sónicos enviados , con el pin Echo en estado "1" todo ese tiempo . En cuanto detecta la llegada del ECO , dicho pin Echo retorna una vez más a estado "0", y el sensor vuelve a estado reposo hasta que se le vuelve a activar por el pin Trig otra vez**.
4. Por tanto , la clave para ser capaz de medir la distancia a la que se encuentra el obstáculo detectado por el sensor debido al **eco** producido se encuentra en ser capaz de **contabilizar el tiempo que el pin Echo ha estado a "1"**, y eso es justo lo que nosotros vamos a utilizar para implementar todos nuestros circuitos y programas .

Aplicación de todos estos conceptos básicos generales al control y tratamiento de dicha información con nuestra tarjeta Icezum Alhambra o Alhambra II :

En base a todo lo dicho anteriormente , **¿ cómo tendremos que diseñar nosotros un circuito en la Alhambra para que sea capaz de controlar dicho sensor ? .**

Pues más o menos el proceso que debe seguir nuestro circuito en orden lógico deber ser el siguiente :

1. Por la patilla donde esté nuestra placa conectada al pin **Trig** enviamos un **pulso de 10 µseg de duración** para activar al sensor .
2. Por la patilla que está conectada al pin **Echo** esperaremos a que éste **pase de estado "0" a estado "1" .**
3. En el momento en que detectemos que **Echo** pasa a estado "1" **pondremos en marcha un conjunto de dos contadores de 8 bits en serie ,conectados a los tics que les envía un corazón de MICROSEGUNDOS**. Por tanto , estamos contabilizando los µseg que la señal Echo está en estado "1".
4. Cuando detectamos que el pin **Echo** retorna a estado "0" , detenemos los contadores y enviamos el resultado de ambos a sendos registros donde se guarda el resultado para el análisis posterior .

¿Porqué dos contadores de 8 bits y no 1 o 100 ? . Todo está fundamentado nuevamente en la hoja de datos del sensor , su capacidad de medir y nuestras necesidades de información .

Si volvemos a la hoja de características veremos que nos indica como rango máximo de trabajo **4 metros .** De manera que empleando la ecuación que relaciona velocidad , espacio y tiempo , tendremos un tiempo empleado por el eco emitido en retornar al sensor de :

t(µsec) = 4000 mm / 0.17 = 47059 µsec

Si usamos **1 único contador de 8 bits , solamente podremos contar 256 µsec** , lo que traducido a distancia significa :

256 µsec * 0.17 = 43.52 mm

, y con ello no cubrimos el rango de medida del sensor .

En cambio , si ponemos **2 contadores de 8 bits en serie , es decir , el segundo midiendo los desbordamientos o grupos de 256 µsec que cuenta el primero , seremos capaces de ampliar el rango de medida de nuestro contador de µsec a 256*256 = 65536 µsec** , que traducido a distancia significa :

65536 * 0.17 = 11141.12 mm = 11.14 metros

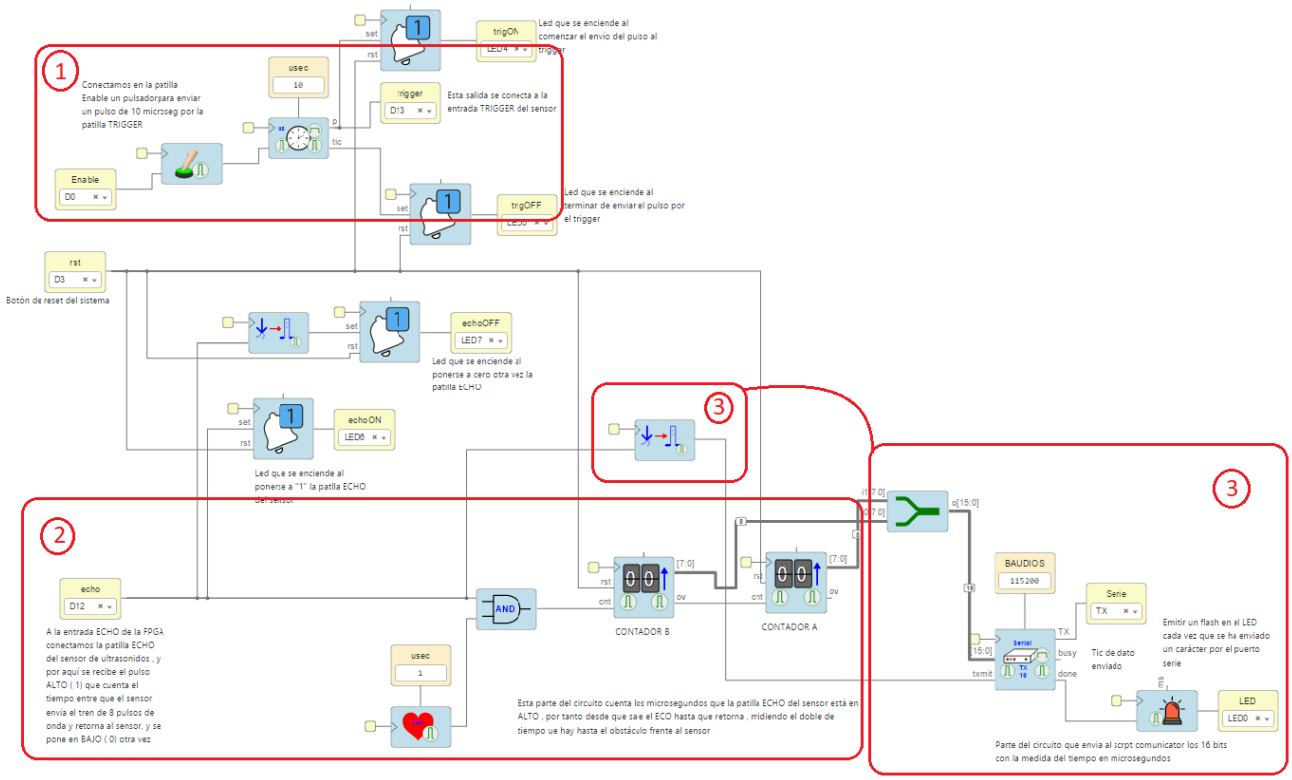
Más que suficiente para nuestras necesidades de cálculo , dado el rango de trabajo de nuestro sensor (4 metros) .

Pruebas de control iniciales del sensor de ultrasonidos HCSR04 . Desarrollo del bloque Sensor_Ultrasonidos_HCSR04 .

- **Pruebas-01 : Primeras pruebas de control del sensor HCSR04 . Activación del sensor y la medición mediante un pulsador . Envío de la información de distancia al PC por bus serie .**

Comenzamos con el circuito más básico que podamos diseñar y que nos permita confirmar que efectivamente tenemos el control sobre este sensor y que nos devuelve información coherente .

En la siguiente imagen tenemos el circuito que podéis encontrar en los ejemplos con el nombre de Pruebas-01 , y en dicha imagen he resaltado en cuadros rojos 3 áreas principales numeradas del 1 al 3 , despreciando el resto de elementos secundarios del circuito que o bien corresponden con conexiones a leds que nos sirven de testigos para confirmar que los pasos se están ejecutando o bien al circuito de reset para poner todo a cero y volver a realizar otra nueva prueba .



Nos centraremos únicamente en los 3 bloques de circuito recuadrados :

1. Cuando pulsamos el botón **Enable** ponemos en marcha un temporizador de 10 µsec que **por la salida p envía un pulso al pin Trig del sensor conectado a ella** . Esta señal es la que activa el sensor de ultrasonidos y hace que emita el tren de 8 pulsos al que nos referíamos anteriormente .
2. Antes de activar el sensor con el pulso de 10 µsec , la patilla **Echo** del sensor y por tanto el pin con la etiqueta **echo** de la Alhambra se encontraban con una señal de "0" lógico , de tal manera que la puerta AND del bloque de código 2 no enviaba señal al contador posterior .

   Pero al activarse el sensor , la patilla Echo del sensor **cambia a estado "1"** y por tanto **la puerta AND habilita las señales del corazón de microsegundos conectado a ella, que comienza a bombear tics hacia el contador B de 8 bits.**

   Cada vez que este primer contado B , menos significativo rebosa y llega a 256 , se envía un tic al segundo contador A , más significativo , conectado en serie con el anterior , que contabiliza "paquetes de 256 µsec".

   Cuando la entrada "echo" vuelve a estado "0" porque el sensor la recibido el **eco de los pulsos que envió y han rebotado en algún objeto**, la puerta AND retorna a "0" y la cuenta de microsegundos se para .
3. Simultáneamente , el cambio de estado en el pin "Echo" ha generado un flanco de bajada que es detectado por el bloque 3 del circuito y hace que los valores registrados en ese instante por los dos contadores de 8 bits sean enviados por el **transmisor serie Tx** hacia el PC , de forma que se puedan registrar en cualquier software de análisis del puerto serie , como por ejemplo el script_communicator . Para mayor información sobre este asunto , sería interesante revisar el Tutorial 30 de @Obijuan en el siguiente enlace : https://github.com/Obijuan/digital-electronics-with-open-FPGAs-tutorial/wiki/V%C3%ADdeo-30:-Puerto-serie .

Una vez hemos enviado la información almacenada en los registros de los 2 contadores A y B al script_communicator , el análisis de los datos y su conversión a distancia se hace de la siguiente forma :

Contamos el número total de microsegundos que han transcurrido con la entrada ECHO en alto , para lo cual tendremos que hacer la siguiente operación :

t (microsegundos) = (Valor almacenado en Contador A) * 256 + (Valor almacenado en Contador B)

Por ejemplo , supongamos que la tarjeta Alhambra nos devuelve la siguiente información :

Contador A = 003 Contador B = 146

t (microsegundos) = 3 * 256 + 146 = 914 microsec ,

Transformando dicha información a espacio recorrido , **la distancia al objeto en mm será** :

D_{objeto} = 0.17 * 914 = 155.38 mm <> 155 mm

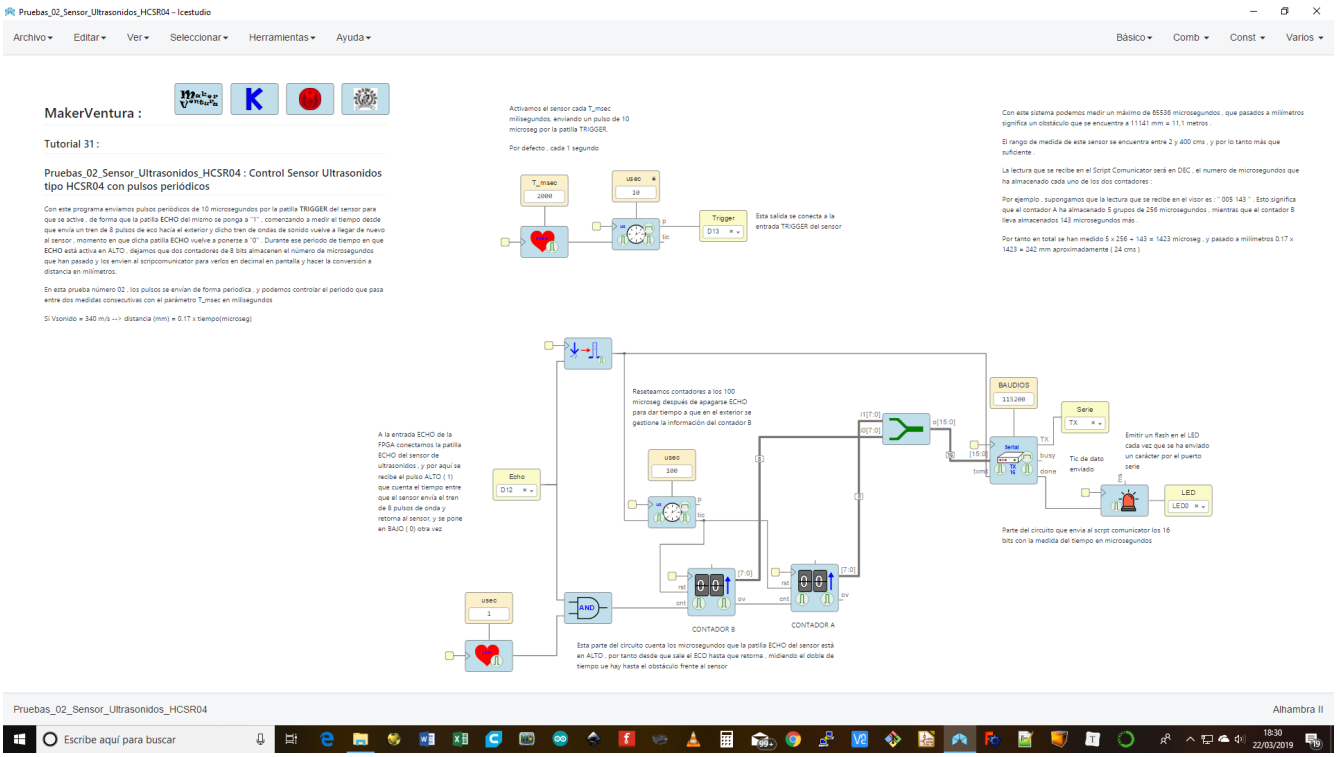
Aquí os dejo el enlace al vídeo de comprobación de funcionamiento :

Pruebas01 : https://youtu.be/JyrTvP4u1Wc

- **Pruebas-02 : Control del sensor y muestreo de distancias mediante un pulso periódico seleccionable . Enviamos la información de distancia recogida al PC por bus serie.**

En esta segunda prueba que realizamos , **el único concepto que vamos a cambiar en nuestro diseño es la forma en la que realizamos la toma de distancias con el sensor . En vez de activar el sensor de forma manual mediante un pulsador , lo haremos de manera periódica y automática a una frecuencia de muestreo que podremos alterar a conveniencia** .

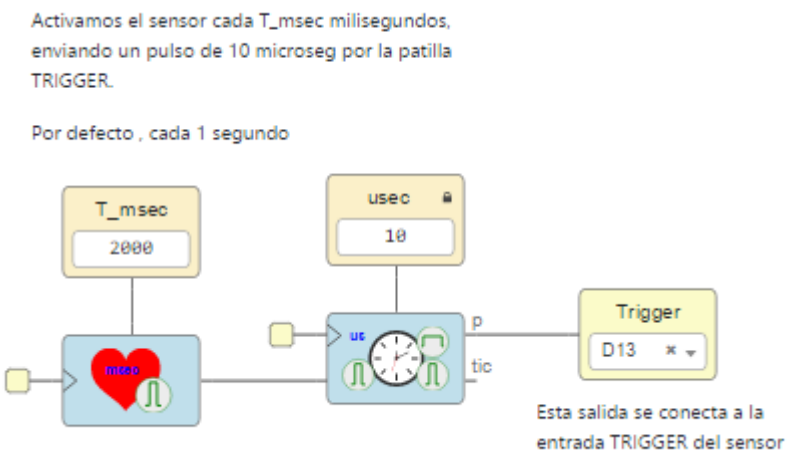
Aquí vemos primeramente una imagen global del circuito completo :



Y a continuación vamos a ver el detalle de las 4 áreas fundamentales en las que se divide el mismo :

1. Activación del sensor .

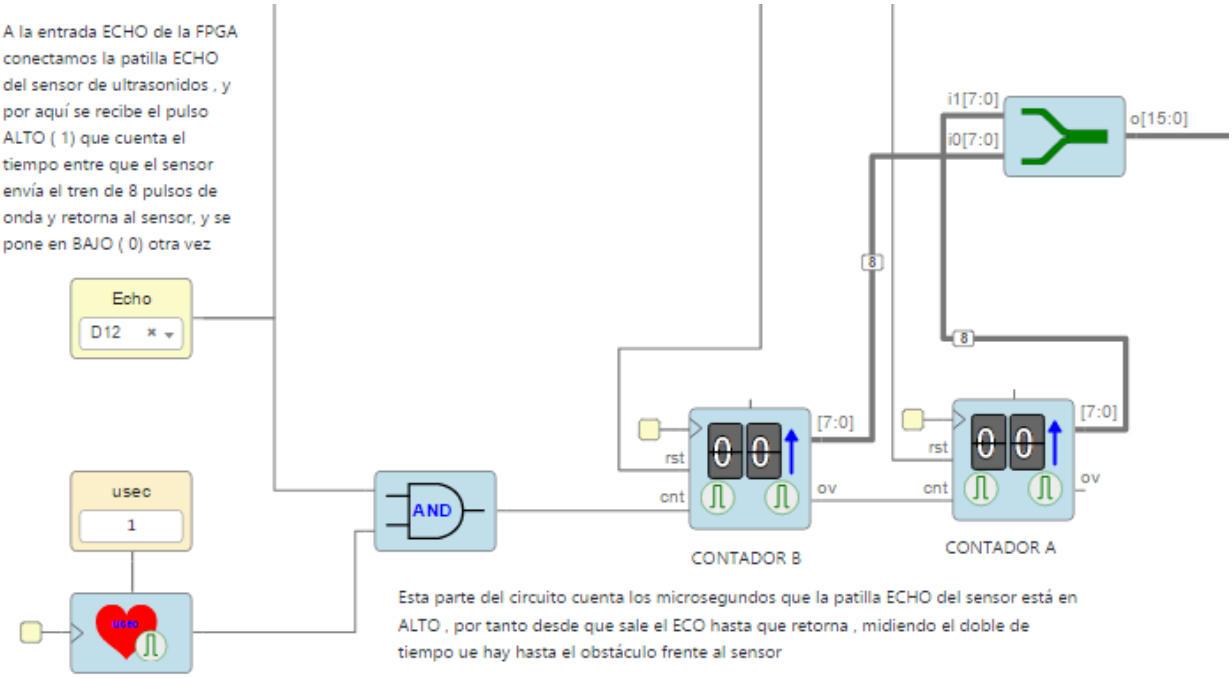
   Con un corazón de periodo T_msec (en este caso 2 segundos) **lanzamos pulsos de 10 µsec al pin Trig del sensor** para activarlo .



2. Medida de la duración de la señal ECHO .

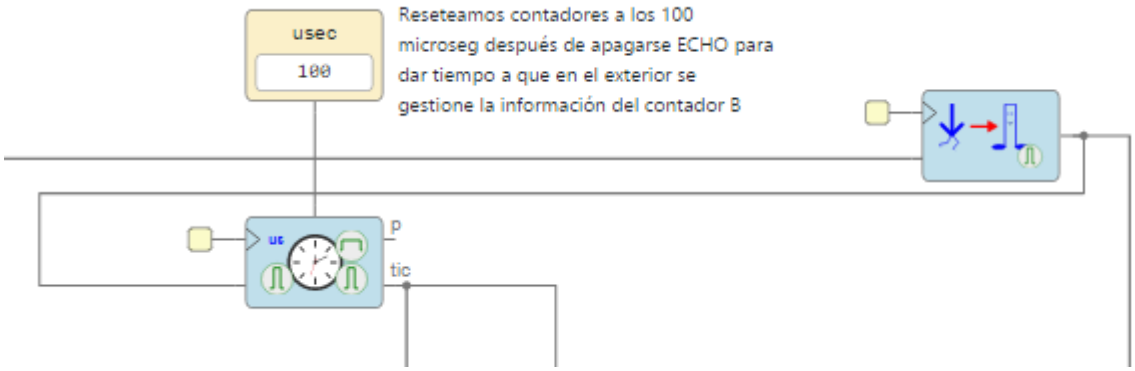
   Cuando por el pin **Echo** de la placa nuestro sensor empieza a enviar un "1" porque ya ha salido el tren de 8 pulsos de ultrasonidos , la puerta AND habilita el corazón de µsec , de manera que **se empiezan a registrar los tics en los contadores A y B**, hasta que dicho pin **Echo vuelve a estado "0" porque se ha recibido un eco** , en cuyo caso se detienen los contadores porque la puerta AND queda deshabilitada .

   Por tanto , el valor que tenemos en ese momento en los dos contadores A y B nos indica el número de µsec que ha estado el pin Echo del sensor en estado "1", y por tanto el tiempo desde que salido el tren de 8 ultrasonidos hasta que se recibe su eco .



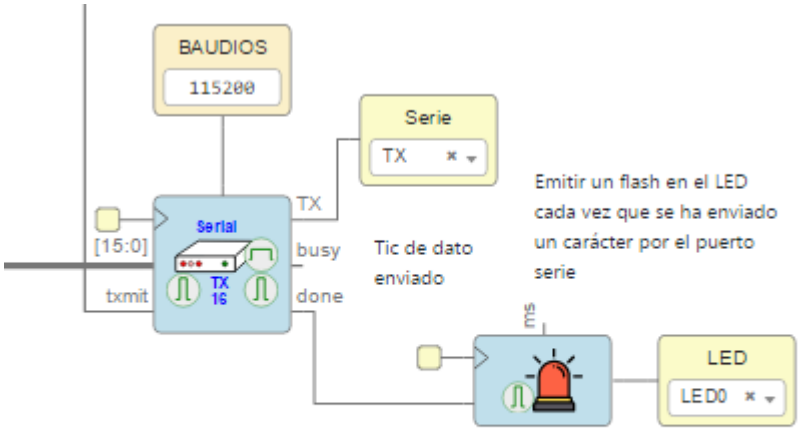
3. Circuito de reset y de lanzamiento de la comunicación.

   Cuando se detecta el cambio en el pin **Echo** de estado "1"a estado "0 ", hacemos dos cosas : Por una parte , **activamos el comunicador serie para enviar la información al PC** , y por otra , ** esperamos un cierto tiempo a que se envíe la información del contador B y **reseteamos contadores para que el circuito esté listo para una nueva medición**.



4. Circuito del comunicador serie al PC.

   Con esta última parte del circuito lo que hacemos es enviar en serie los 16 bits de información que guardaban los dos contadores A y B , y enviarlos en serie hacia el PC para mostrarlos por ejemplo en el programa scritp_communicator , cuando el circuito de reset le envía la activación de comunicación por la entrada **txmit** .



Parte del circuito que envia al scrpt comunicator los 16 bits
con la medida del tiempo en microsegundos

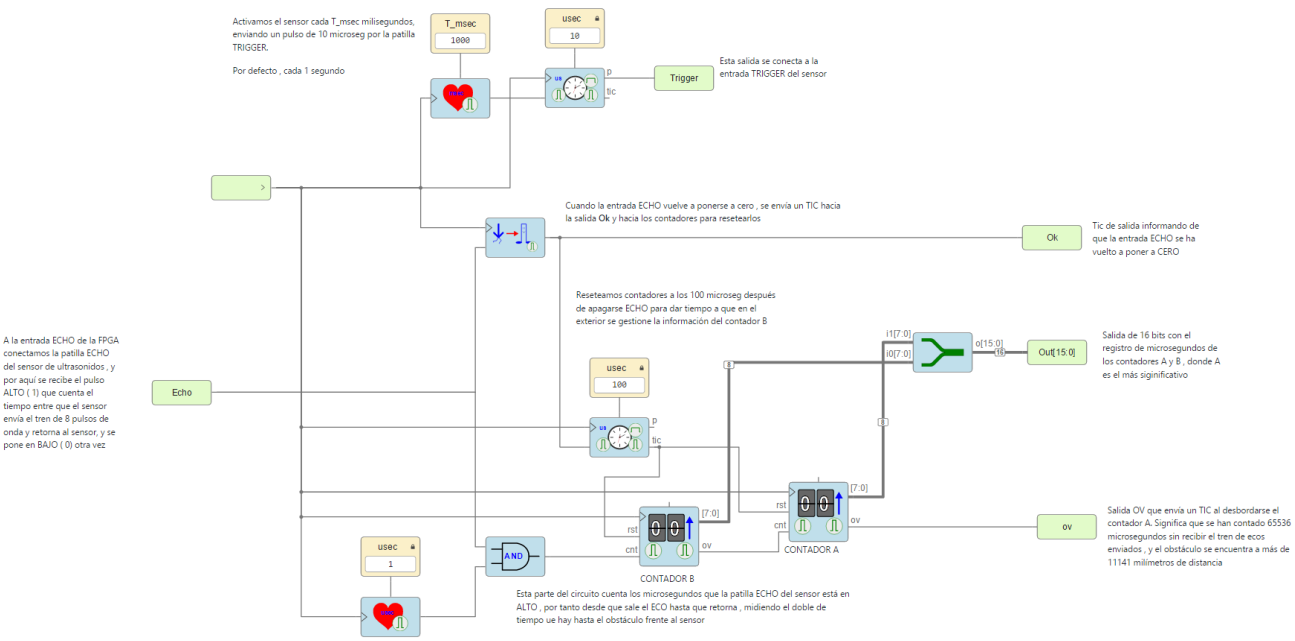
Aquí os dejo el enlace al vídeo de comprobación de funcionamiento de la misma :

Pruebas02 : https://youtu.be/AXRWEyyfnSM

- Pruebas-03 : Uso del bloque Sensor US_HC_SR-04 para realizar circuitos . Toma de muestras de distancias a frecuencia constante. Envío de las muestras recogidas al PC por bus serie .

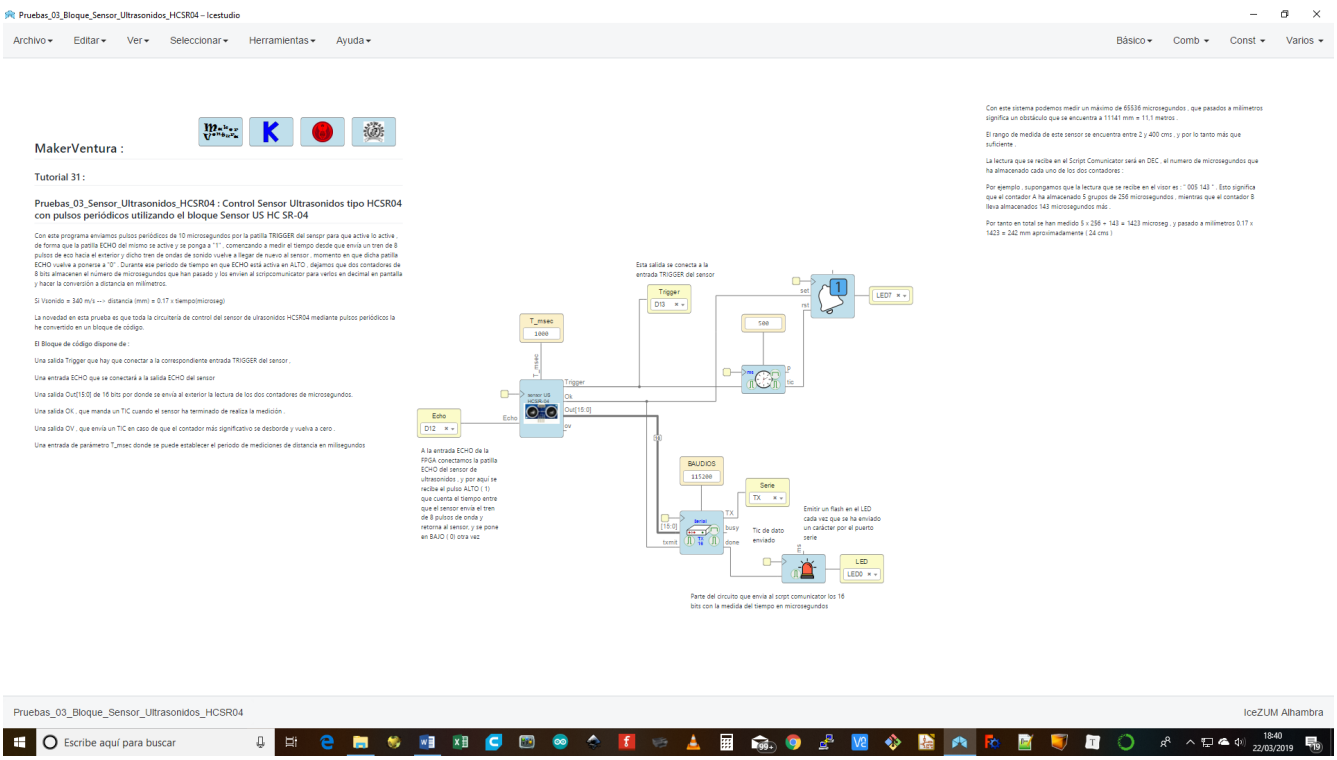
En esta prueba-03 , lo único que se pretende es convertir la rutina de comunicación con el sensor de ultrasonidos en un bloque de código , de forma que los circuitos que diseñemos posteriormente sean mucho más limpios y sencillos de entender .

Veamos una imagen de **cómo ha quedado el diseño del circuito correspondiente al bloque de código US_HC_SR-04.ice** :



Como se puede apreciar es un bloque de código con **1 entrada Echo** donde se conecta **el pin Echo del sensor** , **1 salida Trigger** , donde hemos de conectar **el pin Trig del sensor** , la **salida Out[15:0]** por donde obtendremos **los 16 bits con la información de los dos contadores de µsec**, una **salida OK** que nos da **un tic cuando se recibe la señal de eco y por tanto se termina la medición del tiempo** , y finalmente , **una salida OV** que nos da un tic **en caso de que se desborde el contador A más significativo** y lo queramos usar como señal de error para algo.

En la imagen siguiente , tenemos el mismo circuito de la prueba-02 anterior , pero esta vez usando el bloque de código para controlar el sensor de ultrasonidos :

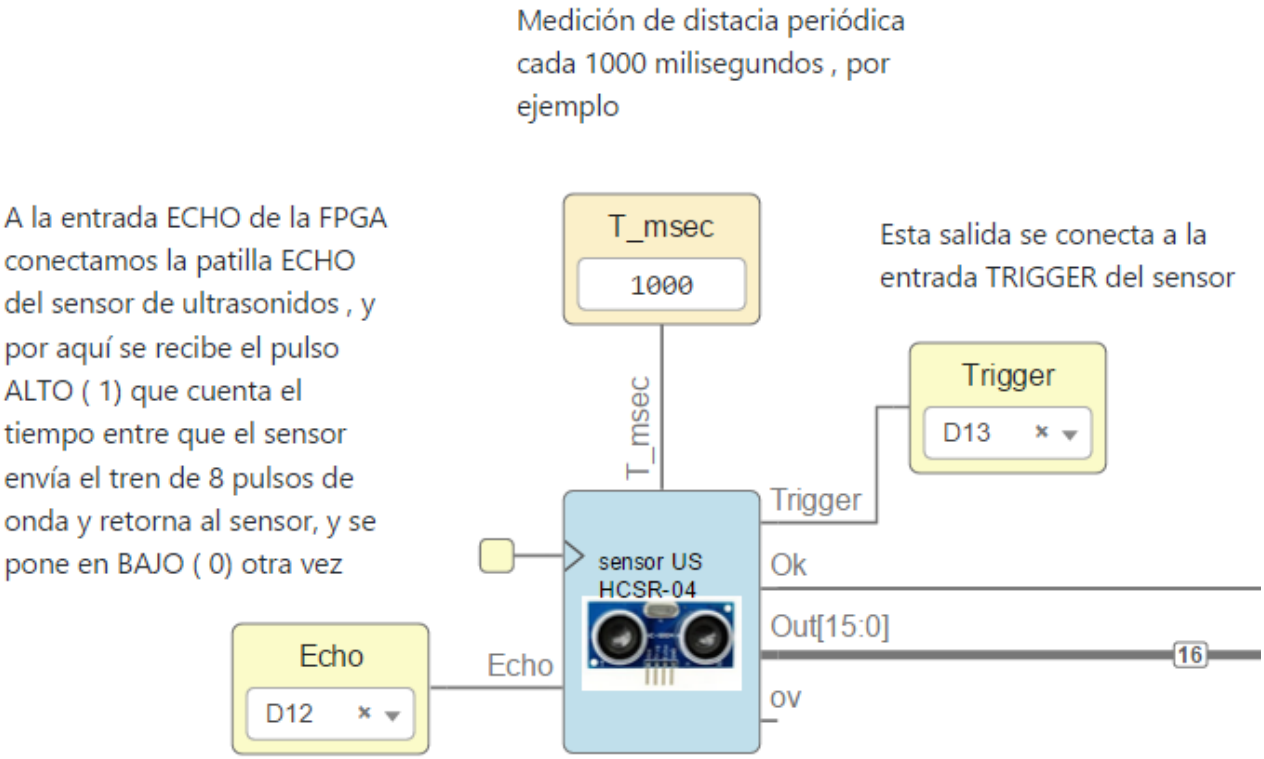


- Pruebas-04 : Toma de muestras periódicas con el sensor de ultrasonidos , mostrando el resultado del registro de cada uno de los dos contadores de 8 bits por los 8 leds internos de la placa Alhambra , seleccionados previamente mediante un multiplexor y un interruptor externo.

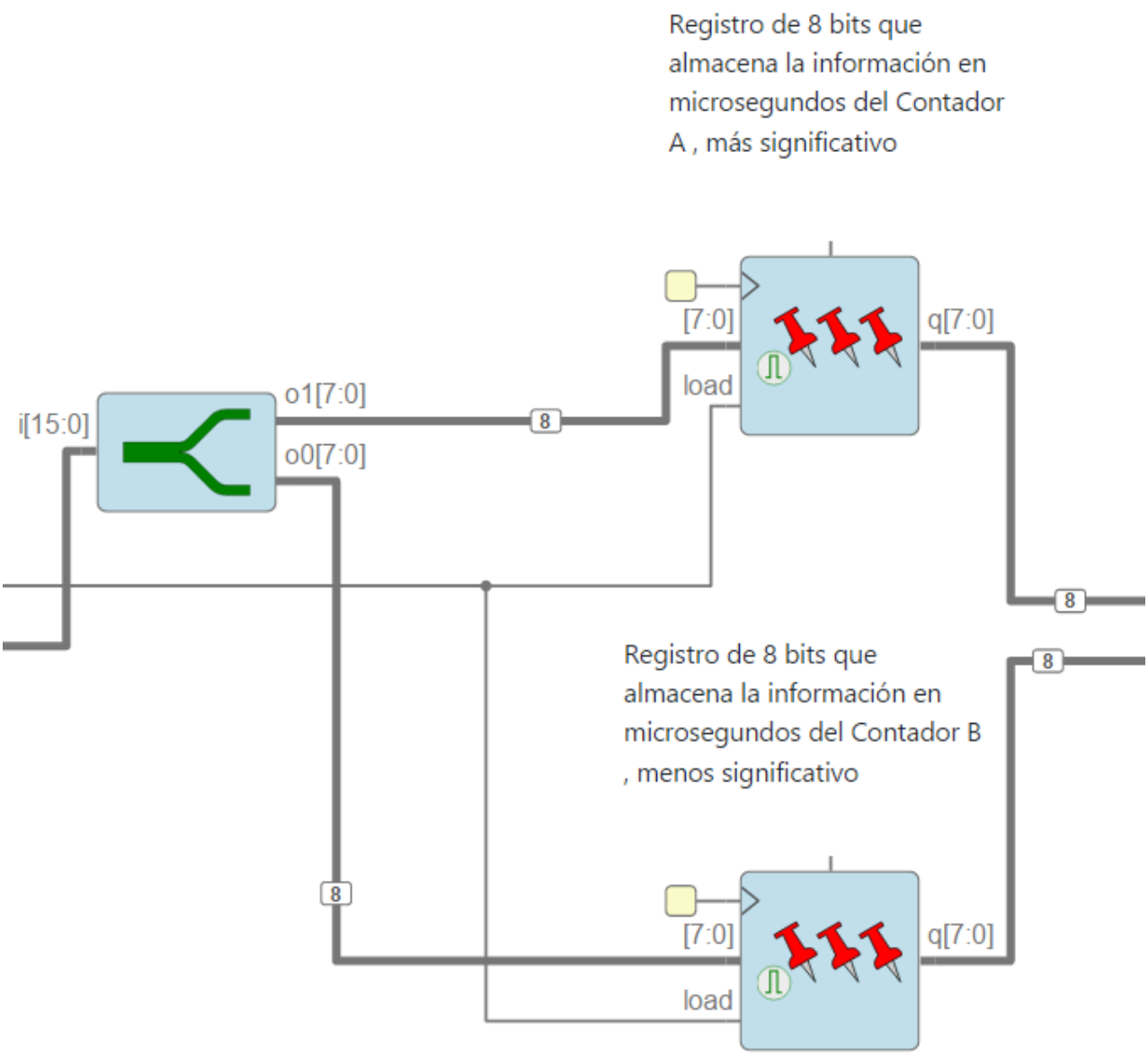
Con esta nueva prueba en vez de comunicar el resultado del valor del tiempo en µsec que hemos medido mediante los dos contadores de 8 bits , lo que vamos a hacer es enviar dicha información a los 8 leds internos de la placa de manera que nos mostrarán el valor de cada uno de ellos en binario , una vez lo seleccionemos mediante un multiplexor y un interruptor externo .

Aquí os muestro el circuito dividido en los tres bloques funcionales de código :

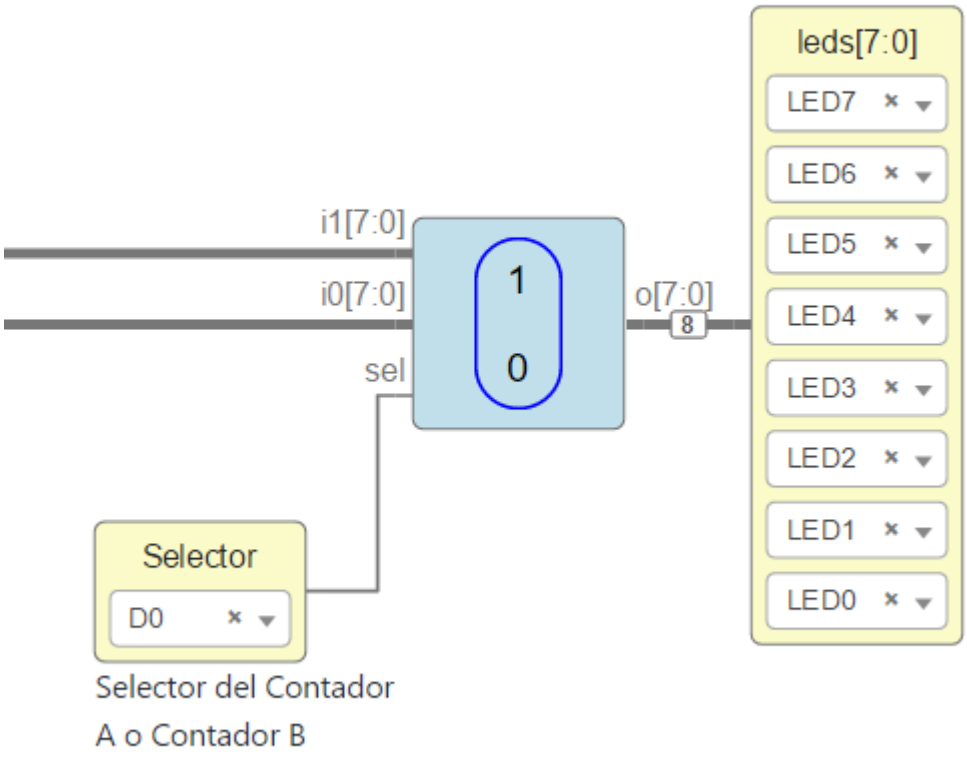
- Control y comunicación entre la placa y el sensor de ultrasonidos :



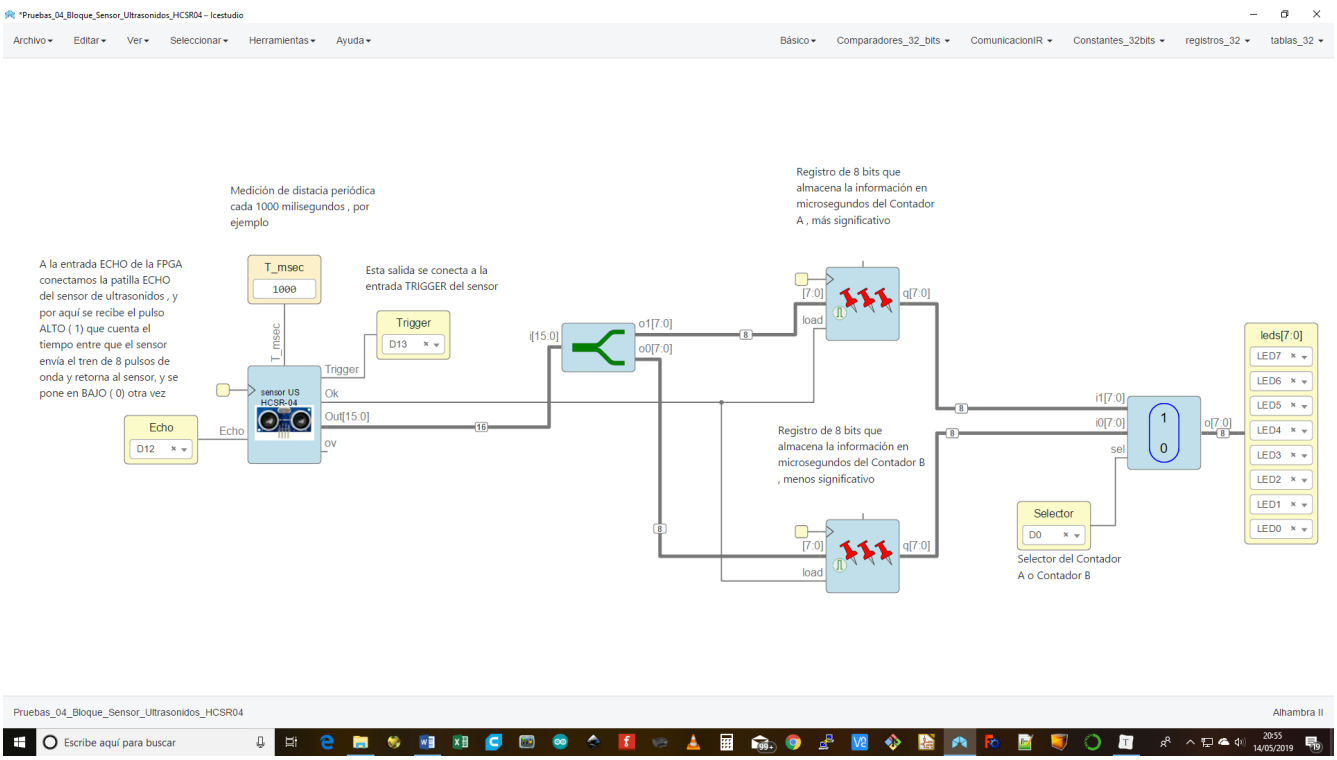
- Separación de la salida de 16 bits en dos registros de 8 bits , uno por cada contador A y B .



- Selección del contador , A o B , a mostrar mediante un interruptor y multiplexor de 2 canales de 8 bits , y envío de la información a los 8 leds internos de la placa .



Y aquí finalmente una imagen completa del circuito :



Y enlace al vídeo de comprobación de funcionamiento :

Pruebas04 : https://youtu.be/hgIoTXtF1FE

Ejemplos de circuitos de desarrollo aplicando el sensor de ultrasonidos .

Vamos a ver a continuación y para terminar de documentar este sencillo cuaderno en el que tratamos de documentar la utilización del sensor de ultrasonidos HC SR-04 con una placa Icezum Alhambra o Alhambra II , una serie de 3 ejemplos básicos de aplicación del bloque de control de dicho sensor que acabamos de estrenar .

Por supuesto se pueden desarrollar ideas mucho más complejas , pero no me quiero extender demasiado en esta primera toma de contacto con el sensor .

- ### Avisador de distancia límite .

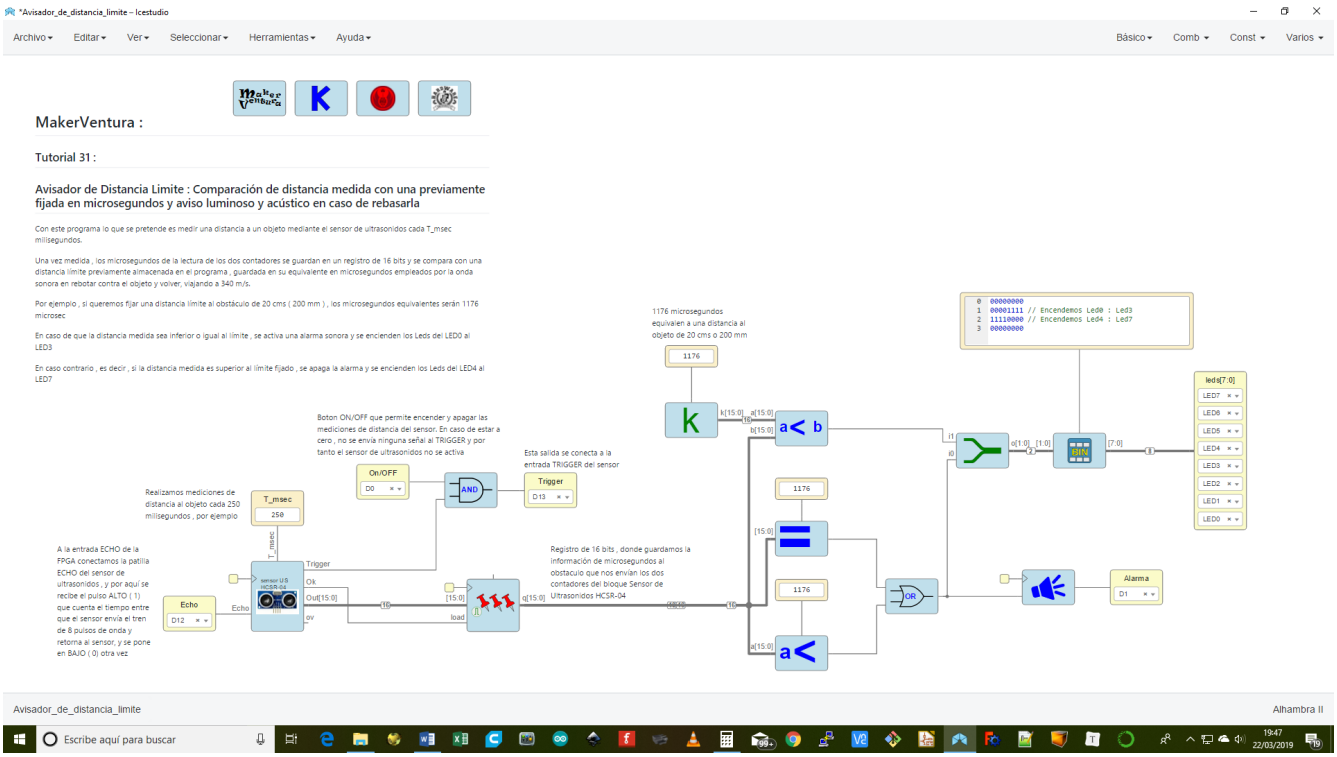
En esta primera y sencilla aplicación vamos a comparar la información de distancia que nos entrega el sensor con una distancia **LÍMITE** establecida con anterioridad .

Cuando nos acerquemos al sensor por debajo de dicho límite establecido , haremos que suene una alarma sonora , y al alejarnos otra vez , dicha alarma dejará de sonar .

Hago la aclaración aquí de que lo que nos envían los registros NO es información de distancia en sí , sino tiempo invertido en recorrer dicha distancia medido en µseg , de manera que de momento hay que convertir las distancias a programar en los límites a su equivalente en tiempo empleado en recorrerlas .

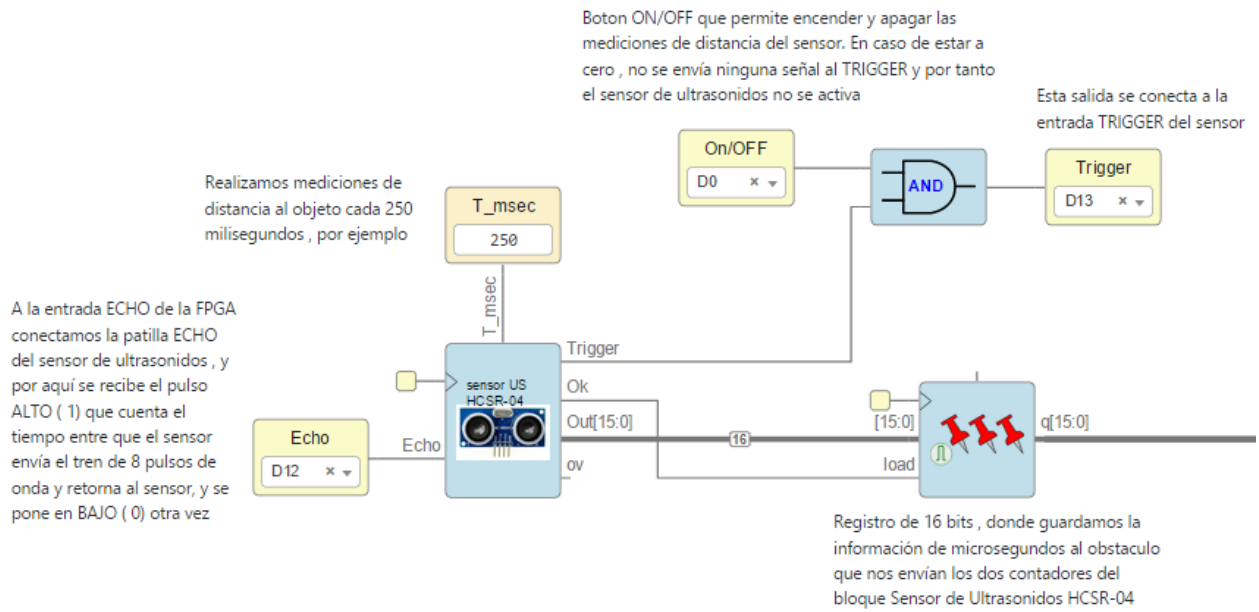
En una próxima revisión de mi bloque de código de control del sensor , cambiaré la salida de información de TIEMPO a DISTANCIA en mm , por ejemplo , para que el tratamiento posterior de la información sea más trasparente y sencillo .

Aquí tenemos primeramente una imagen general del circuito completo :



Y a continuación , vamos a ir viendo , paso a paso , los bloques funcionales que lo confirman , para poder entender a grandes rasgos la base de su funcionamiento :

1. **Conexión , control y registro de información del sensor de ultrasonidos :**



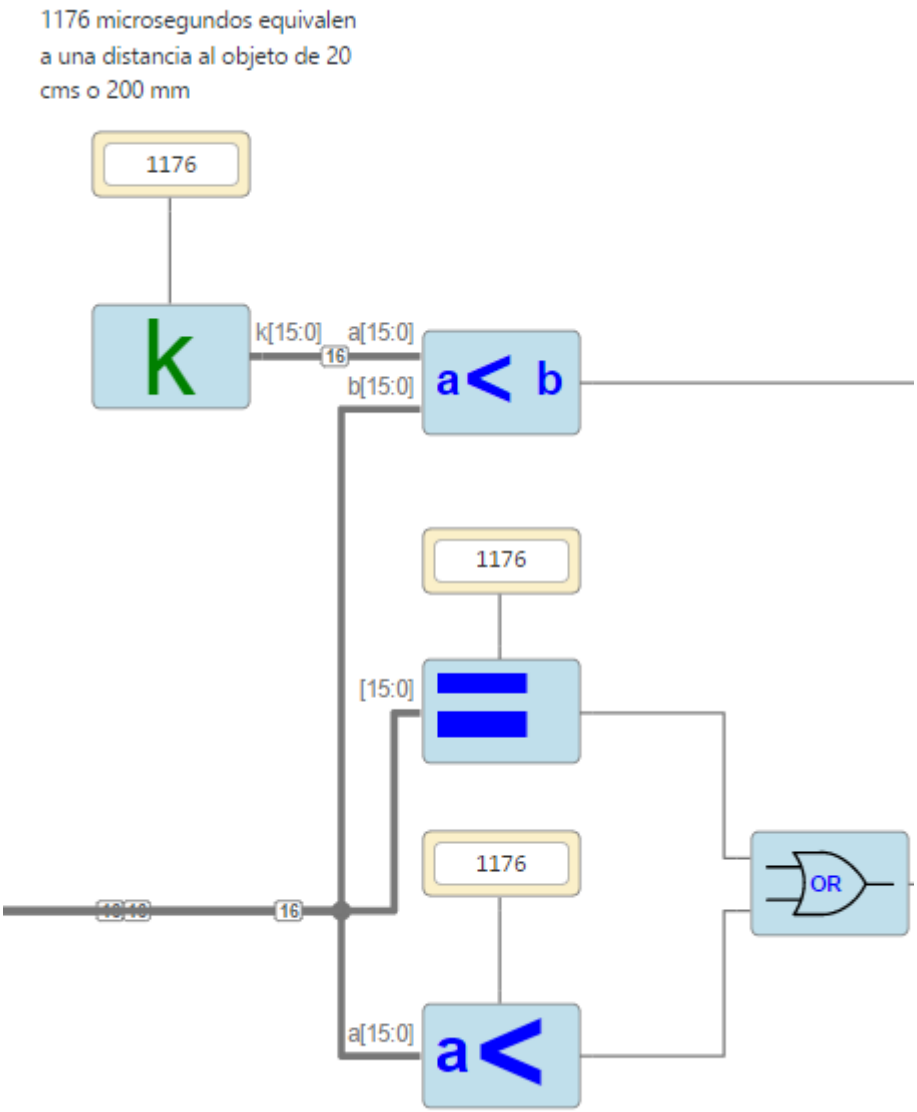
Esta primera parte del circuito es la encargada únicamente de :

- Conectar con los pines del sensor , **Echo y Trigger**.
- Establecer la frecuencia de muestreo de medidas a través del parámetro **T_msec** : En este caso cada 250 msec.
- Incluye un interruptor externo **ON/OFF** que a través de una puerta AND , nos permite controlar el encendido/apagado del sensor : Si está el interruptor en "0" , no se envía ningún pulso por el trigger

, así es que el sensor no funcionará .

- Y para finalizar , conecta **salida de datos Out[15:0] de los dos contadores internos con un registro de 16 bits** donde se almacena temporalmente la información recogida por el sensor .

2. **Tratamiento lógico de la información que nos llega desde el sensor :**

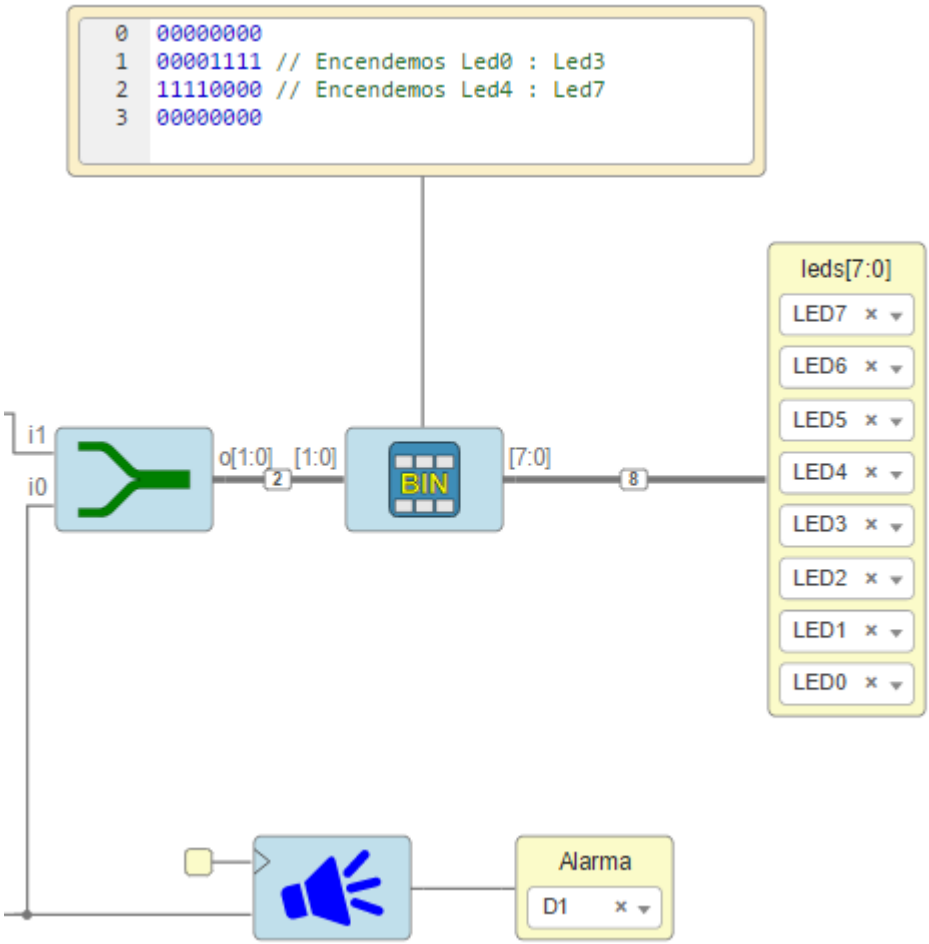


En segundo lugar tenemos la parte que realiza el **análisis de la información recibida desde el sensor** . Aquí es donde le establecemos la distancia límite que determina el que realicemos acciones posteriores o no . Dicha **distancia** hay que incluirla realmente como **tiempo empleado en recorrerla a la velocidad del sonido** , de tal manera que si por ejemplo en nuestro caso , quisiéramos que la lógica del circuito nos avise cuando el objeto se encuentre a 200 mm de distancia , **el parámetro que tendremos que introducirle al programa no es 200 mm , sino 200/0.17 = 1176 µsec aproximadamente**.

La línea superior del circuito pasa de "0" a "1" en el caso de que **el tiempo al objeto sea MAYOR QUE 1176 µsec**.

Mientras que la línea inferior del mismo enviará un "1" en caso opuesto , **cuando dicho tiempo recibido sea MENOR o IGUAL a 1176 µsec**.

3. **Acciones a tomar en función de la información que llega y la lógica del circuito :**



Bien , por la parte del circuito mencionada en el anterior apartado 2 , ya somos capaces de detectar cuando el objeto está a mayor , menor o igual distancia a una previamente establecida .

Ahora toca tomar decisiones : ¿ Qué queremos que nuestro programa realice con dicho conocimiento ? De esa toma de decisiones se encarga la tercera parte del circuito .

Ambas señales que nos envía la parte 2 las unimos en un bus de 2 bits de datos y se las entregamos para su análisis a **una tabla BIN, donde ya le hemos grabado qué tiene que hacer en cada uno de los casos posibles que se le van a presentar** . Los casos (i1,i0) = (0,0) o (1,1) son imposibles dado que el objeto no puede estar simultáneamente a mas y menos distancia de una prefijada , y por tanto el circuito no hace nada con ellos .

En el **caso (i1,i0) = (1,0) , el objeto se encuentra a más de 200 mm** , y el circuito **decide encender los leds (led4:led7) internos de la placa .**

Mientras que en el **caso (i1,i0) = (0,1) , el objeto se encuentra a una distancia igual o inferior de 200 mm** , y el circuito **decide encender los leds (led0:led3) internos de la placa .**

También , en circuito paralelo al anterior , en este último caso , **cuando i0 = 1 , se envía una señal a un buzzer para que suene una alarma acústica** .

Enlace al vídeo de comprobación de funcionamiento :

Avisador_de_distancia_limite : https://youtu.be/ic4v9K10ufo

- **Escala luminosa de 8 leds , función de la distancia del sensor a un determinado objeto .**

En este segundo ejemplo de aplicación lo que hacemos es generalizar la idea del ejemplo anterior y **dividir el campo espacial frente al sensor en una serie de ZONAS , según las cuales se irán dando instrucciones al programa para que encienda o apague en escala una serie de 8 leds en total**.

En nuestro caso la escala luminosa es capaz de distinguir desde 0 mm hasta 700 mm , en 8 intervalos de 100 mm cada uno :

000 - 100 mm : Led0 encendido

100 - 200 mm : Led0-Led1 encendidos

200 - 300 mm : Led0-Led1-Led2 encendidos

300 - 400 mm : Led0-Led1-Led2-Led3 encendidos

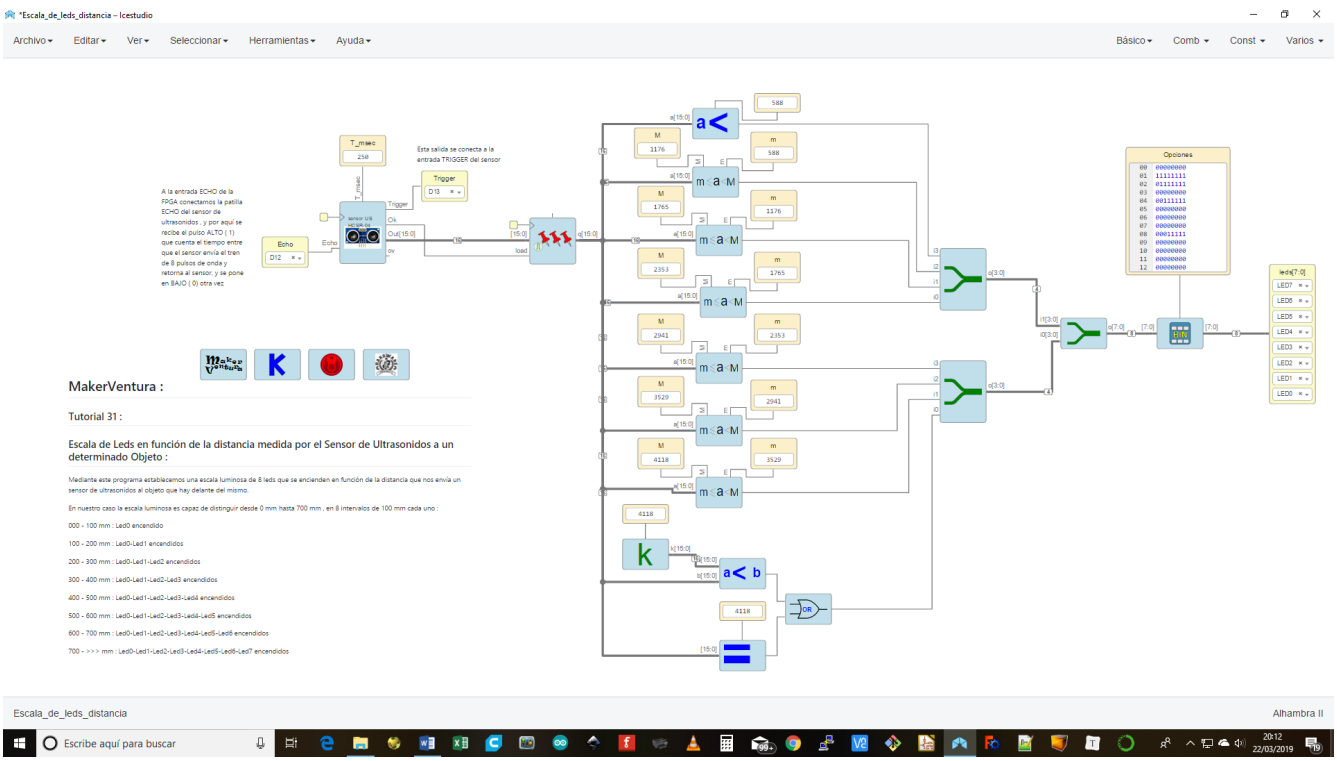
400 - 500 mm : Led0-Led1-Led2-Led3-Led4 encendidos

500 - 600 mm : Led0-Led1-Led2-Led3-Led4-Led5 encendidos

600 - 700 mm : Led0-Led1-Led2-Led3-Led4-Led5-Led6 encendidos

700 - >>> mm : Led0-Led1-Led2-Led3-Led4-Led5-Led6-Led7 encendidos

Aquí vemos primeramente una imagen global del circuito :

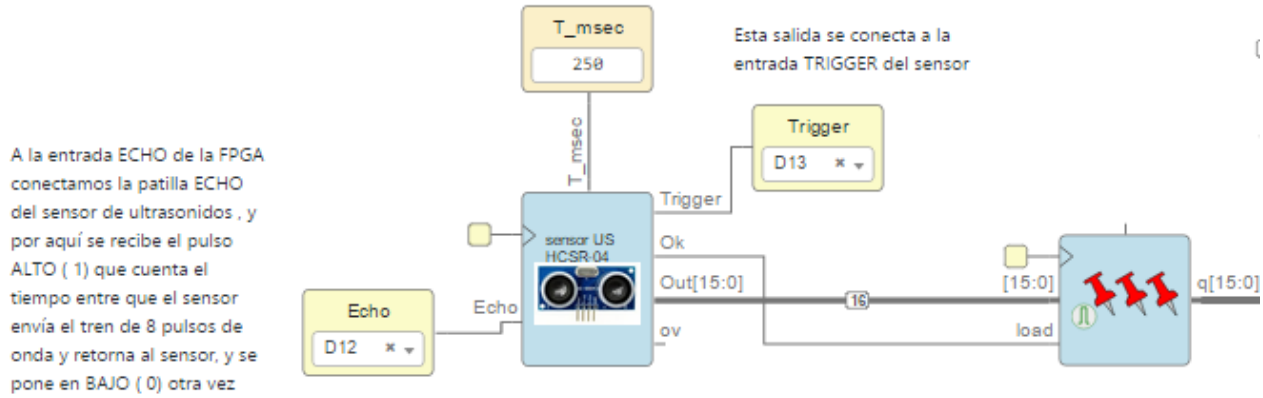


Y ahora veremos las tres parte principales en las que está dividido el mismo :

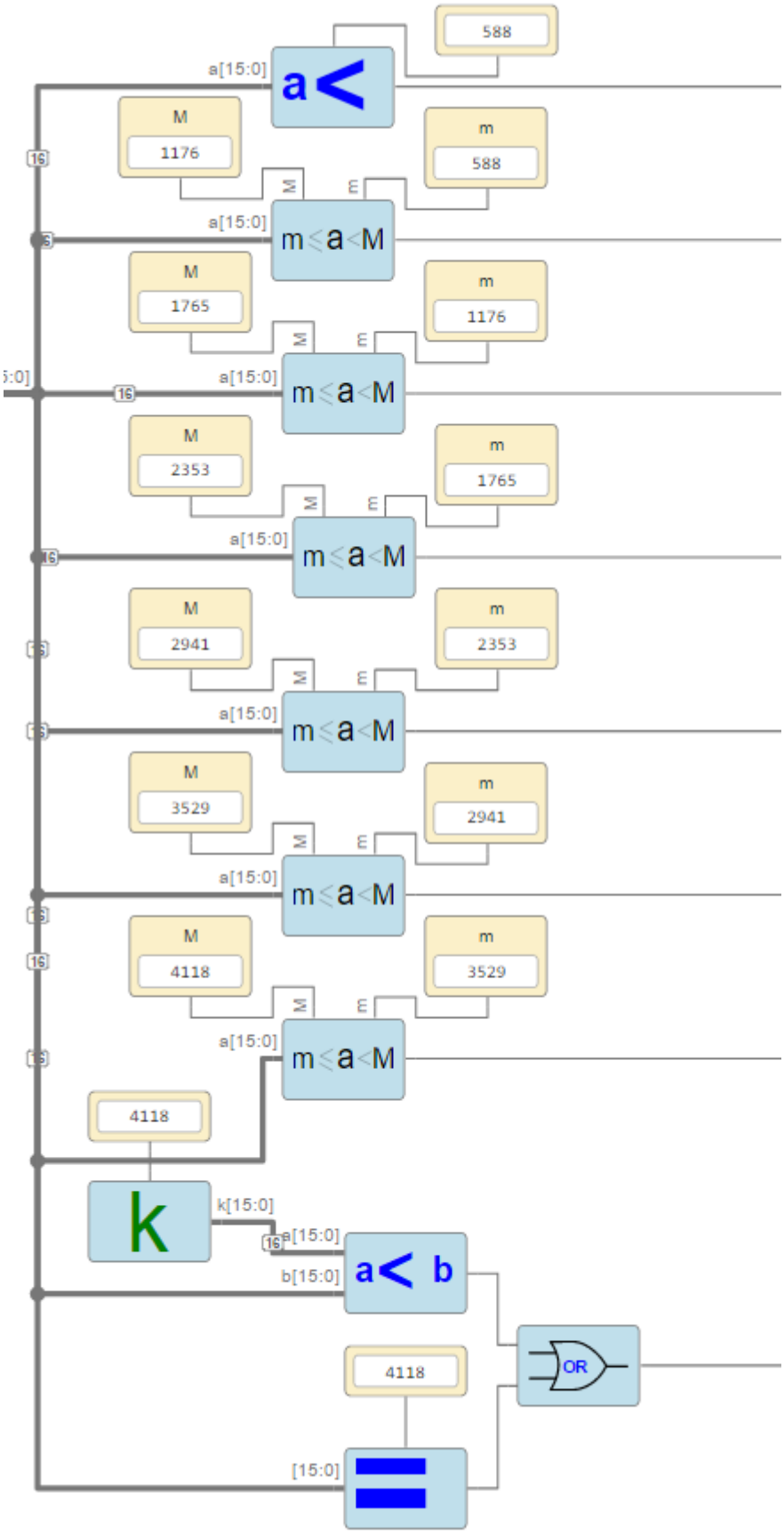
1. **Conexión , control y registro de información del sensor de ultrasonidos :**

   Nada nuevo respecto al ejemplo anterior , salvo que aquí no hay interruptor ON/OFF del sensor .

   Ver ejemplo anterior.



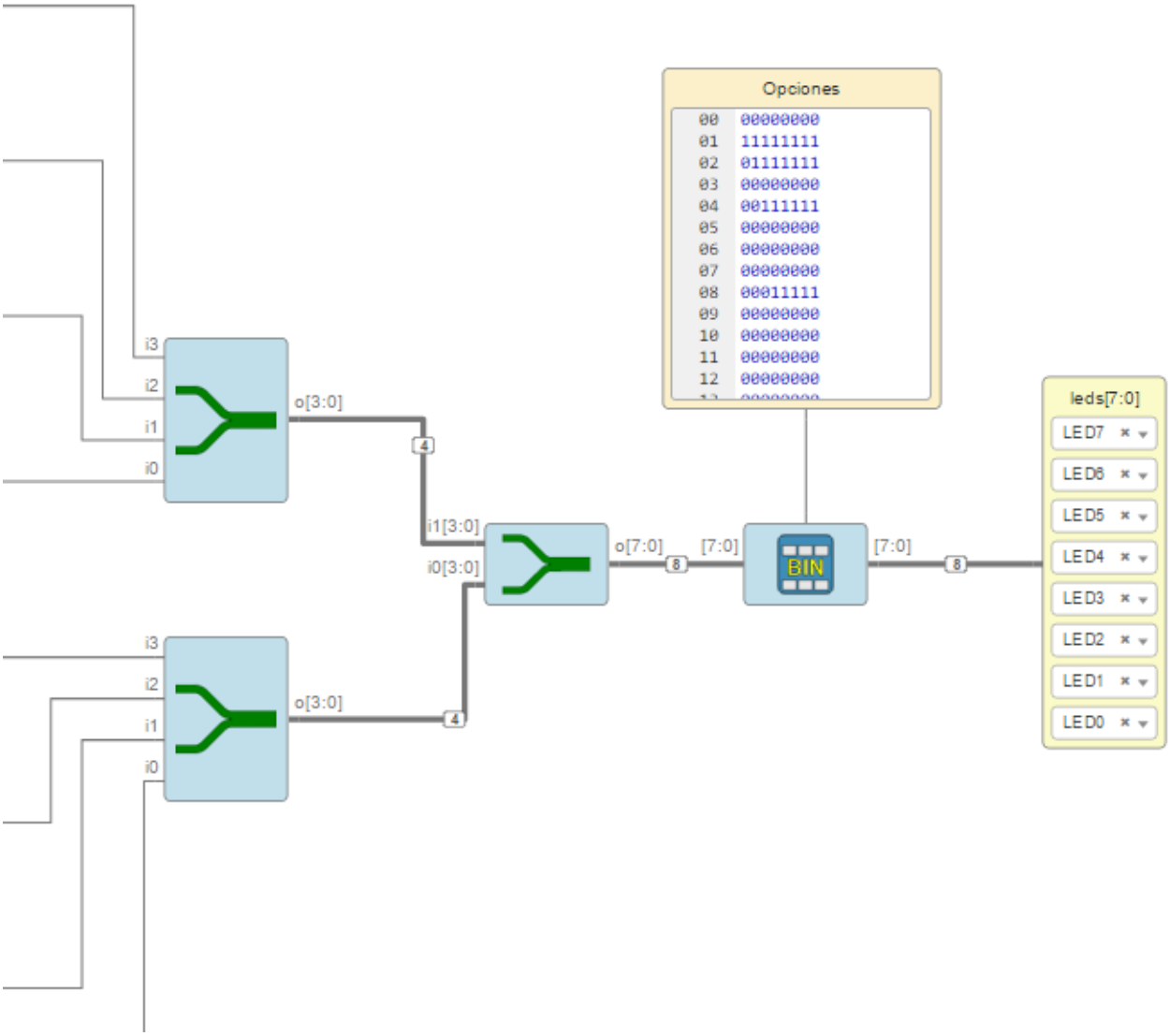
2. **Tratamiento lógico de la información que nos llega desde el sensor :**



La idea es extrapolar el ejemplo anterior . Dividimos el campo espacial delante del sensor en 8 zonas (tantas como leds tiene en nuestro caso la placa . Por supuesto se puede hacer cualquier otro rango) , y **mediante circuitos comparadores vamos permitiendo a la placa detectar y avisarnos, pasando de estado "0" a "1" , cuando compruebe que la distancia medida por el sensor coincida en valor con**

cualquiera de los sectores establecidos de antemano.

3. **Acciones a tomar en función de la información que llega y la lógica del circuito :**



Esta última fase también es muy parecida a la del ejemplo anterior , con la diferencia de que **la tabla BIN donde están establecidos los casos posibles y las acciones a tomar (encendido secuencial de los 8 leds) es de 8 entradas y 8 salidas para comandar los 8 leds de la placa .**

Solamente existen 8 casos posibles , en los que solamente uno de los 8 cables estará en estado "1" mientras que el resto permanecerá en estado "0", y para cada unos de esos casos , la tabla BIN le asigna una acción a tomar con los 8 leds. Para el resto de opciones , la tabla le asigna el valor de salida "00000000" .

Aquí os dejo también el enlace a un vídeo para que veáis su funcionamiento :

Escala_leds_Distancia : https://youtu.be/Km8XliO7zLU

- **Robot detector de obstáculos mediante sensor de ultrasonidos y cabezal giratorio 90º .**

Finalmente os propongo en este pequeño cuaderno una aplicación muy típica del sensor de ultrasonidos a un robot de escuela , para que veáis que es posible detectar obstáculos y generar decisiones adecuadas para no chocar con el entorno .

El sensor de ultrasonidos va montado en un cabezal gestionado por un pequeño servomotor que con una cierta frecuencia establecida , hace que el sensor rastree la distancia a los objetos que se encuentran alternativamente a derecha e izquierda del robot .

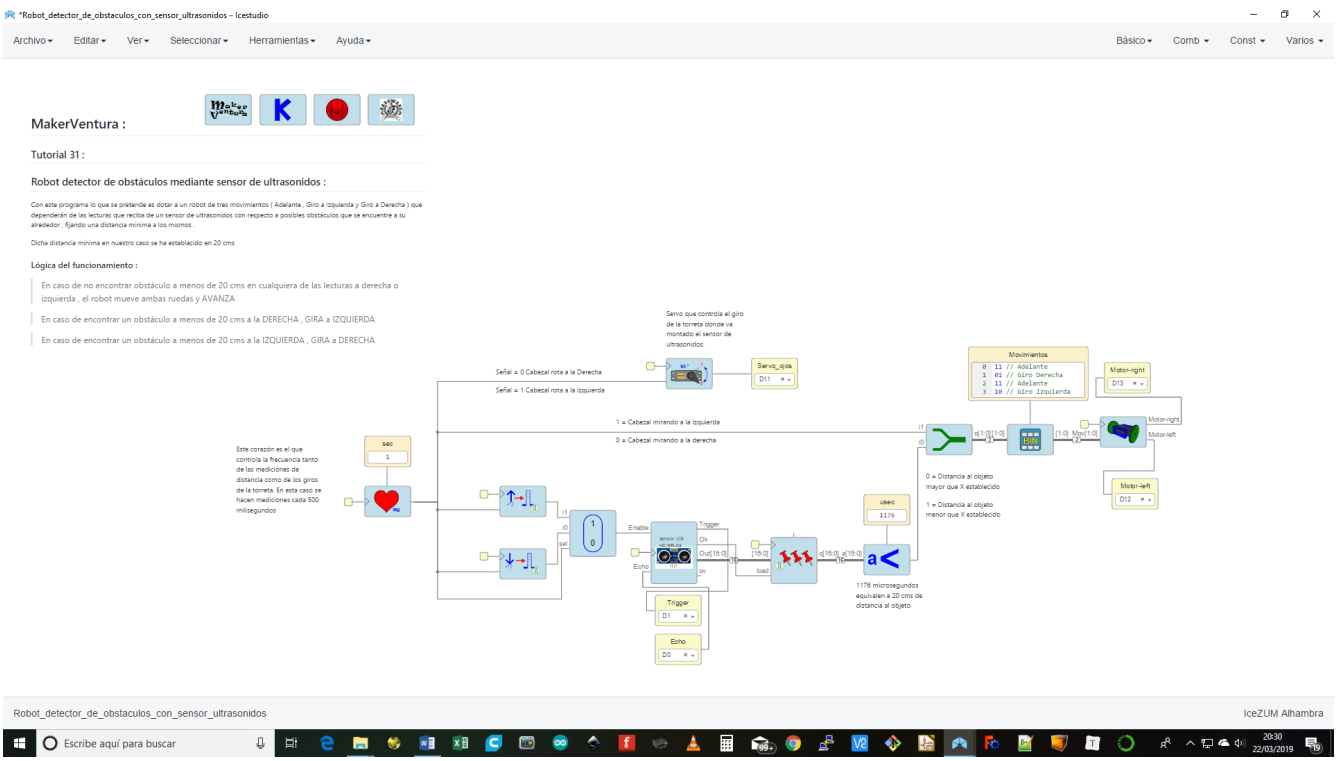
De manera general , el robot tiene como instrucción prioritaria **avanzar hacia delante con ambos motores encendidos**.

Solamente altera dicha acción básica en dos casos :

- **Si detecta un objeto a menos de 20 cms de distancia cuando el sensor está mirando a la derecha , lo que hace es apagar el motor de la rueda izquierda , de forma que realizará un giro a izquierdas , huyendo del objeto .**
- En cambio , **si detecta un objeto a menos de 20 cms de distancia cuando el sensor está mirando a la izquierda , lo que hace es apagar el motor de la rueda derecha , de forma que realizará un giro a derechas , huyendo del objeto .**

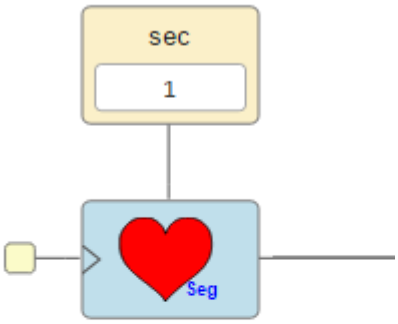
Por supuesto , este ejemplo se puede perfeccionar mucho más . Animo a tod@s los interesados en el tema a trabajar en el tema y mejorarlo todo lo que sea posible , aumentando la dificultad de la lógica del mismo hasta donde os lleve la imaginación .

Como en ejemplos anteriores , veamos primero una imagen general del circuito :



Y finalmente , nos vamos a detener un poco más en los grupos funcionales más relevantes , analizando cómo realizan su trabajo en el conjunto :

1. **Sincronización de todos los subcircuitos** .

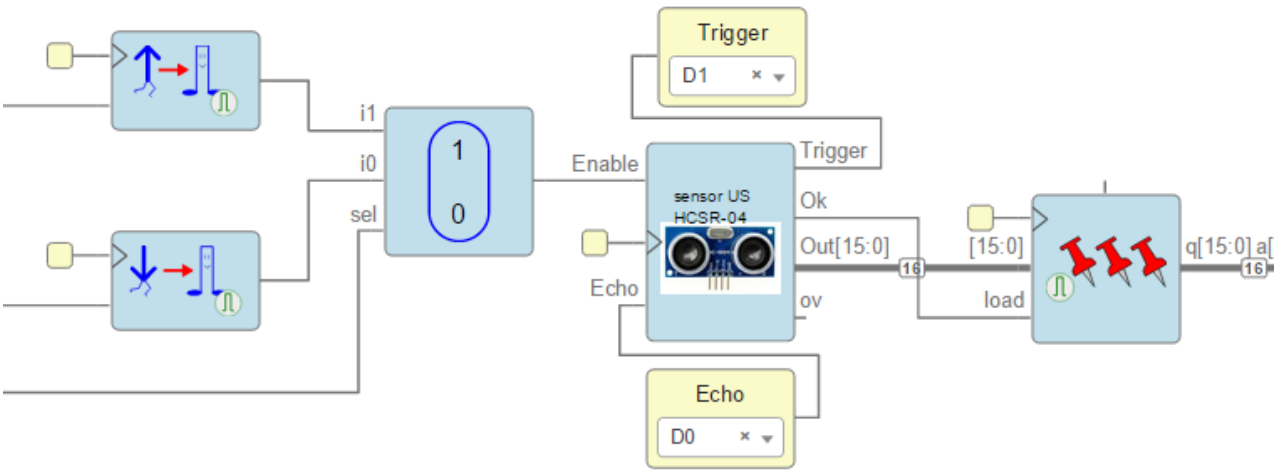


Este corazón es el que controla la frecuencia tanto de las mediciones de distancia como de los giros de la torreta. En esta caso se hacen mediciones cada 500 milisegundos

Este corazón es el que **sincroniza** absolutamente todas las acciones que se llevan a cabo en este circuito :

- Cada 500 milisec mueve el servomotor de la torreta hacia un lado del robot : derecha o izquierda.
- Genera los dos **pulsos** (cuando el corazón pasa de "0" a "1" , y cuando retorna de "1" a "0" . Los dos flancos de subida y bajada) **que activan el sensor por el trigger y hacen que el sensor mida distancias** a los objetos que hay frente a él en ese preciso momento .
- Informa al subcircuito que se encarga del análisis y toma de acciones de **la orientación del sensor en la torreta** .

2. **Conexión , control y registro de información del sensor de ultrasonidos :**



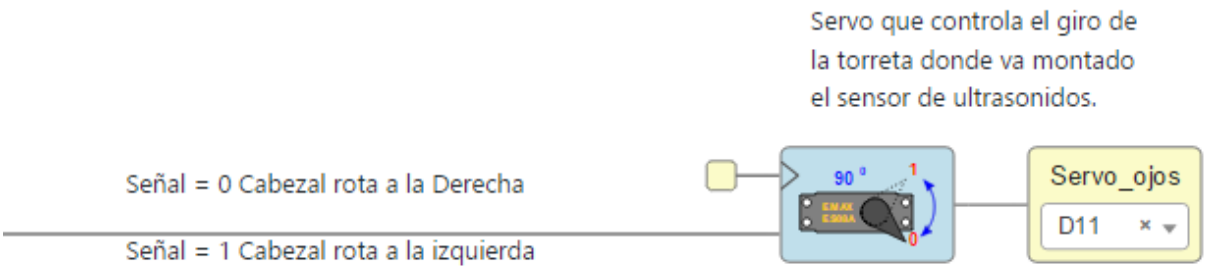
Aunque esta parte ya la conocemos de los ejercicios anteriores , veamos algún detalle especial :

En este ejemplo estoy usando otro bloque de control del sensor de ultrasonidos que dispone de una **entrada adicional Enable** . Esta entrada nos permite controlar desde el exterior **los instantes en los que deseamos activar el sensor para que mida distancia** . Todo el resto del bloque es idéntico al que hemos estado usando hasta ahora .

Los dos detectores de flanco conjuntamente con el multiplexor lo que hacen es **realizar una especie de corazón de tics virtual de periodo 500 milisecs** , es decir , hacen que por la entrada **Enable** del bloque controlador del sensor y sincronizado con el pulso global del robot , lleguen instrucciones de tomar medidas cada 500 milisecs .

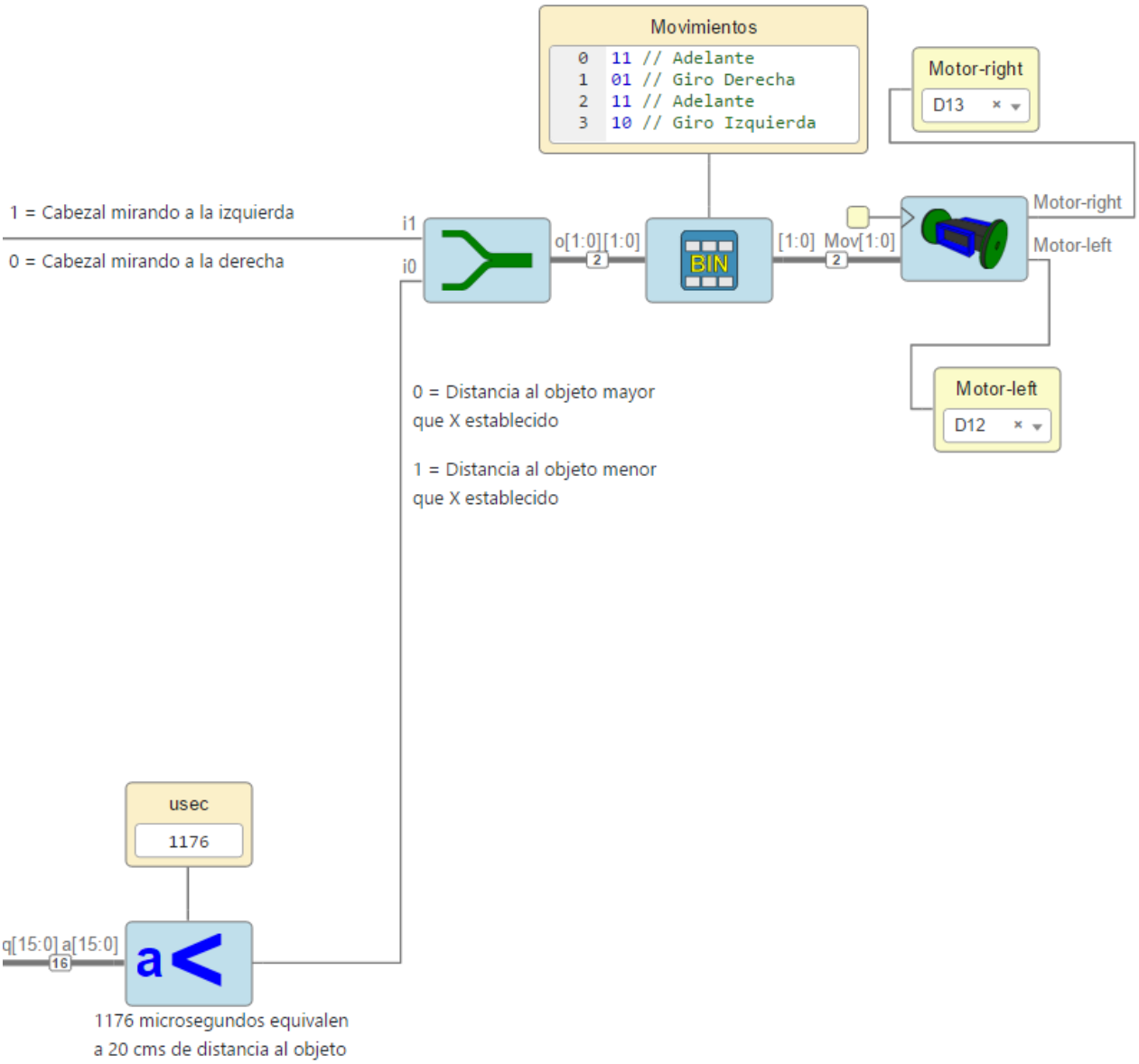
Finalmente , como en el resto de ejemplos anteriores , el dato de distancia que envía el bloque por la salida **Out[15:0]** en cada medición es guardado en un registro con la señal de **tic de finalización de medida** que envía la salida **ok** .

3. **Control de movimientos de la torreta donde va montado el sensor :**



Este subcircuito es el que controla la orientación de la torreta donde va acoplado el sensor . En la semionda cuadrada donde recibe un "1" desde el corazón , el servo hace que la torreta se oriente hacia la **izquierda** , mientras que cuando recibe un "0" la torreta se orientará hacia la **derecha**.

4. **Lógica de análisis de datos y acciones a tomar sobre las ruedas del robot :**



Y finalmente llegamos a la parte del circuito donde toca realizar el **tratamiento de los datos que llegan tanto de la posición de la torreta como de la proximidad crítica o no de algún obstáculo** .

Tanto la **orientación del sensor** como **la información sobre si existe obstáculo a menos de una distancia X prefijada - en nuestro caso 20 cms-** llegan a la entrada de la tabla BIN , y **en función de los valores ingresados por (i1,i0) se busca en la tabla la acción que tiene que realizar sobre los dos motores que accionan las ruedas del robot .**

Para terminar con este pequeño cuaderno sobre control del sensor de ultrasonidos HC-SR04 , aquí os dejo también el enlace al vídeo de comprobación de funcionamiento de este último ejemplo :

Robot_detector_de_Obstaculos_con_sensor_ultrasonidos : https://youtu.be/NPYH_u8R6IQ

Espero que os haya gustado , que os estimule la imaginación y os anime a realizar miles de otros circuitos con este sensor tan interesante .

Saludos,

MakerVentura , 2019