Proyecto final de Curso – IS911 – IP 2020
Universidad Nacional Autónoma de Honduras
UNAH

Implementacion de Distanciómetro con Sensor Ultrasónico utilizando Microcontrolador PIC16F84A

Renata Mavelyn Dubón, 20171000808, Carlos Ramos, 20121008840, Mario Gómez, 20151001773

***Resumen*— Los microcontroladores son dispositivos electrónicos que a menudo ayudan en la programación o automatización de tareas que así lo requieran y que entre estos tenemos uno de los más populares en el mercado el PIC 16F84A Microcontrolador de 8 bits que es un dispositivo programable que se compone de una computadora digital, una unidad de memoria de datos, una unidad de memoria de programa y puertos de entrada/salida en un circuito integrado, funciona como un controlador de periféricos en un sistema mínimo. El µC depende de una alimentación de al menos 5V y 0V en sus entradas de Vdd y Vss respectivamente para su operación (el consumo del µC depende de las cargas de los puertos y de la frecuencia de trabajo), requiere de una señal de reloj que le indique la frecuencia de trabajo, está señal la introducimos a través de un oscilador de cristal de cuarzo XTal de 4Mhz en los pines OSC1 y OSC2, y una alimentación al pin MCLR, que es un pin de reset que activa al microcontrolador. El funcionamiento del µC está determinado por un programa almacenado en su memoria Flash ROM y puede programarse más de una vez para cambiar su estado y su comportamiento, lo que lo convierte al µC en una pieza esencial en el rápido desarrollo de aplicaciones electrónicas.**

I. Introducción

ESTE documento proporciona la explicación y desarrollo detallado del Proyecto de distanciometro usando el microcontrador PIC16F84A, aqui se aborda de manera detallada cual fue la manera en que se desarrollo el Proyecto desde la configuracion o disposicion de los componentes en el circuito, detallando asi cada conexion y su finalidad hasta el apartado del codigo en el cual se realizaron todos los calculos y configuraciones de los pines a los que se conecta cada dispositivo electronico, dando como resultado un Proyecto funcional y muy practico a la hora de replicarlo ya que como se explicara mas adelante con los conocimientos nesesarios en lenguaje ensamblador y la correcta disposicion de elementos este se puede llevar a cabo dejando resultados mas que satisfactorios.

II. Objetivos

General

Exponer de manera detallada la implementación y correcta utilización de elementos tan indispensables como lo son los microcontroladores en la electrónica moderna.

Específicos

- Detallar cada paso en la implementación, así como programación del microcontrolador utilizado en el siguiente proyecto.
- Programar una librería para el sensor ultrasónico con la cual se realizarán los cálculos de distancia.
- Desarrollar un circuito lo más compacto y económico para el siguiente proyecto, permitiendo este todo lo estipulado en la idea principal.

III. Planteamiento del Proyecto

Este proyecto fue implementado haciendo uso del microcontrolador pic16f84a, junto con otros componentes electrónicos: una pantalla LCD LM016L y un sensor ultrasónico HC-SR04 principalmente.

Se implementó un dispositivo capaz de medir distancias dentro de un rango de 6 a 246 cm con el uso del sensor y luego esta distancia la imprime en la LCD.

El desafío que se presentó en la elaboración del proyecto fue a la hora de programarlo. Para este fin se programó una librería con las funciones principales del sensor.

Se realizó también un segundo proyecto utilizando el sensor SRF04, el cual se programó con el uso de interrupción por desbordamiento del TMR0.

Cabe destacar que se descargó el paquete del módulo del sensor ultrasónico HC-SR04 de internet, ya que el software de simulación utilizado, Proteus 8 Professional, no lo incluía. Esto motivó al equipo a realizar el proyecto paralelo con el sensor ultrasónico que Proteus si incluye de manera predeterminada, el SRF04.

IV. FUNDAMENTO TEORICO

Los sensores de ultrasonidos son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias que van desde pocos centímetros hasta varios metros. El sensor emite un sonido y mide el tiempo que la

señal tarda en regresar. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración. Estos sensores trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, colores, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo, han de ser deflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

V. LISTADO DE COMPONENTES Y PRESUPUESTO

Componentes

Componentes	Cantidad	Precio
Microcontrolador PIC16F84A	1	Lps. 415
LCD LMO16L	1	Lps. 102
Sensor Ultrasonico	1	Lps. 168
Resistencia variable	1	Lps. 25
Resistencia de 10k	1	Lps. 23
Capacitores de 22p	2	Lps. 38
Capacitor de 100n	1	Lps. 21
Crystal oscilador de cuarzo 4mhz	1	Lps. 35
Fuente de voltaje de 5v	1	Lps. 0
Total		Lps. 827

VI. FUNCIONAMIENTO

Se programó el microcontrolador para que esta lea los datos que captura sensor y los imprima en la LCD.

El algoritmo del programa realiza lo siguiente:

1. Verifica que la variable distancia sea menor a 254 que es la distancia máxima que a representar con el proyecto. Si es igual, se imprime mensaje "Distancia fuera de rango."
2. Imprime en pantalla "La distancia es:"
3. Llama la función Lanzar_Disparo de la librería Sensor_Ultrasonico, que es la que se encarga de activar el pin trigger y leer el pin echo del sensor, realizar el conteo de tiempo que se tarda la señal a llegar al pin echo y posteriormente realizar los cálculos correspondientes de la distancia.
4. Imprime en pantalla el dato de distancia haciendo uso de 3 dígitos bcd.
5. Repite el ciclo.

Descripción de la Librería Sensor_Ultrasonico
La librería cuenta con las siguientes constantes:

- multip = 17
- diviso = 10

Estos valores se explican abajo, al explicar las subrutinas de la librería.

La librería cuenta con las siguientes variables:

- waitForEcho: es un contrador de tiempo que se incrementa en uno cada 100microsegundos.
- distancia: variable en la que se almacena la distancia final.
- unidades: se convierte distancia a formato bcd y las unidades bcd se almacenan en esta variable
- decenas: se convierte distancia a formato bcd y las decenas bcd se almacenan en esta variable
- centenas: se convierte distancia a formato bcd y las centenas bcd se almacenan en esta variable
- dividendo: se utiliza en la subrutina divide
- divisor: divisor que se carga con la constante 10 en la subrutina divide
- div: almacena el cociente en subrutina divide
- aux: variable auxiliar en la que se carga el dividendo en la subrutina divide para no modificar su valor.
- DH: almacena los dígitos del nibble alto al final de subrutina multiplica
- DL: almacenas los dígitos del nibble bajo al final de subrutina multiplica

Las subrutinas que contiene son:

- Sensor_Inicializa: configura pin trigger como salida y echo como entrada
- Lanzar_Disparo: pone a 1 pin trigger y este envía la señal de disparo durante 10 microsegundos. Luego lo pone a cero y espera a que echo se ponga a 1, esta es la señal para empezar a contar el tiempo, se esperan 100 microsegundos cada vez antes de verificar el estado de echo, si este no se ha puesto a 0 entonces se incrementa variable waitForEcho. Para incrementar la variable waitForEcho la subrutina salta a una etiqueta que verifica el valor de dicha variable. Si este valor es igual a 127 entonces finaliza la lectura parque significa que se llegó a la distancia máxima representable. Se llegó al dato 127 con ayuda de despeje de la fórmula que se explica abajo.
  Cuando echo finalmente se pone a cero, al finalizar la lectura se pasa a realizar los cálculos de la distancia. Para esto se toma en cuenta la velocidad del sonido a través del aire que es de 34000 cm/s. Luego Si distancia es la distancia entre el objeto y el sensor entonces la distancia total recorrida por la señal es el doble de la distancia que necesitamos, ya que el recorrido fue de ida y regreso, por lo que:

  distancia = (waitForEcho*10^- 4s * 34000 cm/s) / 2

  Simplificando tenemos:

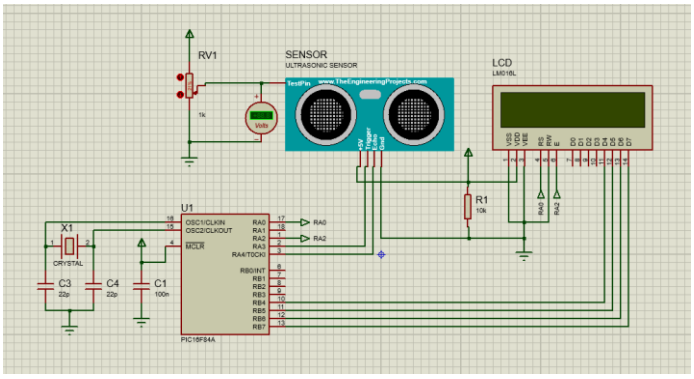
distancia = (waitForEcho * 17) / 10 cm

con esta información se procede a llamar la subrutina Multiplica, que multiplica como su nombre lo dice, el valor almacenado en variable waitForEcho con la constante 17 y lo guarda en la variable distancia. Luego de esta operación se llama la subrutina divide, que divide el contenido de la variable distancia entre 10. El resultado se guarda en la variable distancia.

Después, se procede a llamar la subrutina BIN_a_BCD de la librería bin_bcd proporcionada por el docente. Luego procede a mover las variables de esta librería a las del sensor, unidades, decenas y centenas respectivamente. Finalmente retorna al programa principal.

- Multiplica: esta subrutina suma el multiplicando, que es la variable waitForEcho, veces iguales a la magnitud del multiplicador, que es 17. Los bits del nibble alto se almacenan en DH y los del nibble bajo en DL.
- Divide: esta subrutina resta el divisor en este caso 10, del dividendo es que es el resultado de la multiplicación anterior y cuanta las veces que se hace, en variable div que hace las veces del cociente. Al final este resultado se almacena en variable distancia.

VII. MONTAJE



El montaje físico del proyecto por desgracia no pudo realizarse por problemas actuales de emergencia a nivel mundial por la presente pandemia que presenta el COVID-19, En este caso solo se mostrara una captura de pantalla del cirtcuito completo del proyecto el cual consta de un microcontrolador PIC16F84A, al cual se le han conectado un panel LCD LM016L en los puertos RB4-RB7 y RA0-RA2, un sensor ultrasonico en los puertos RA3-RA4, asi como un Crystal oscilador en los puertos OSC1-OSC2 y por ultimo un capacitor de 100n al Puerto MCLR.

VIII. RECOMENDACIONES

En primer lugar, hablaremos de la diferencia que se encuentra al analizar el Proyecto que realizamos con librería y el Proyecto convencional que algunos libros utilizados en clase proporcionan. Para ello desglosaremos nuestro proyecto en una explicación concisa de la siguiente manera:

NUESTRO PROYECTO:
En el proyecto que desarrollamos se logró realizar una librería donde se compactó toda la lógica necesaria para hacer funcionar el sensor ultrasónico, esto va desde la inicialización de los pines que utiliza el sensor hasta la realización de los cálculos para obtener una distancia aproximada de un objeto con respecto a dicho sensor. En la figura (fig.1) podemos observar que solo basta con colocar 2 líneas de código: *call Sensor_Inicializa* y *call Lanzar_Disparo*, para poner en funcionamiento nuestro sensor ultrasónico:

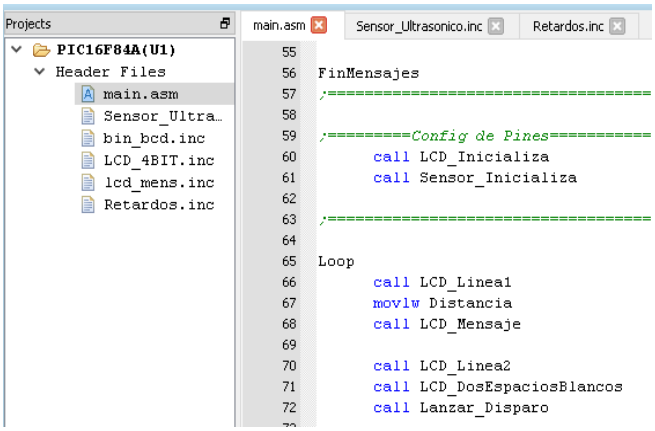


fig.1

Esto se logra gracias a la librería que se creó desde cero, en la cual inicializamos los pines para *Trigger* y *Echo* en los correspondientes pines *RA3* Y *RA4*. Luego se realizó una subrutina en la cual se comienza con el funcionamiento del sensor, a través del envío de una señal (5v) por el pin *Trigger*, con lo que el sensor procederá a activar el pin Echo y esperar que la señal simulada pueda retornar a dicho pin.

```
43
44      bsf trigger
45      call Retardo_10micros
46      bcf trigger
47  Esperar_Echo
48      ;revisamos el pin Echo
49      btfss echo
50      goto Esperar_Echo
51  Leer_Echo
52      ;una vez Echo = 1, esperamos a
53      ;va directamente relacionado c
54      call Retardo_100micros
55      goto Inc_Wait
56  Continue
57      btfsc echo
58      goto Leer_Echo
59  Calcular_Distancia
```

En este punto es cuando necesitamos obtener un valor, el cual es producto del incremento que se la da una variable contador llamada *waitForEcho*, dicha variable se incrementa cada 100microsegundos, esto hasta que el pin *Echo* se desactiva. El valor 127 utilizado en la subrutina indica el máximo valor hasta cual se puede incrementar la variable waitForEcho, esto debido a que el 127 representa un valor de 254cm de distancia, valor máximo a representar en este proyecto.

```
89  Inc_Wait
90      movlw .127
91
92      subwf waitForEcho,W
93      btfsc Zero
94      goto Finish_Read
95      incf waitForEcho,F
96
97      goto Continue
```

Después se procede a calcular la distancia, partiendo del valor obtenido y guardado en la variable *waitForEcho.* Para esto nos valemos de la siguiente fórmula:

D= (waitForEcho*34000cm/s) / (2*10000)

Dicha fórmula se obtiene de analizar ciertas observaciones, en primer lugar recordando que la distancia es igual al producto de la velocidad y el tiempo. Además la teoría nos recuerda que los pulsos enviados y retornados viajan a la velocidad del sonido, cuyo valor es 34000cm/s; de igual forma, como queremos la distancia total, entonces eso es 2 veces la distancia que necesitamos ya que es de ida y regreso. Cabe destacar que el valor de waitForEcho no se encuentra en cm por lo que necesitamos dividir dicha cantidad entre 10000 para obtener el valor correspondiente.

Es fácil darse cuenta que en dicha fórmula existen valores constantes, los cuales se pueden juntar y simplificar para obtener una fórmula más fácil de programar en el algoritmo:

D= (waitForEcho) ((34000cm/s) / (2*10000))

Lo cual simplificado nos da lo siguiente:

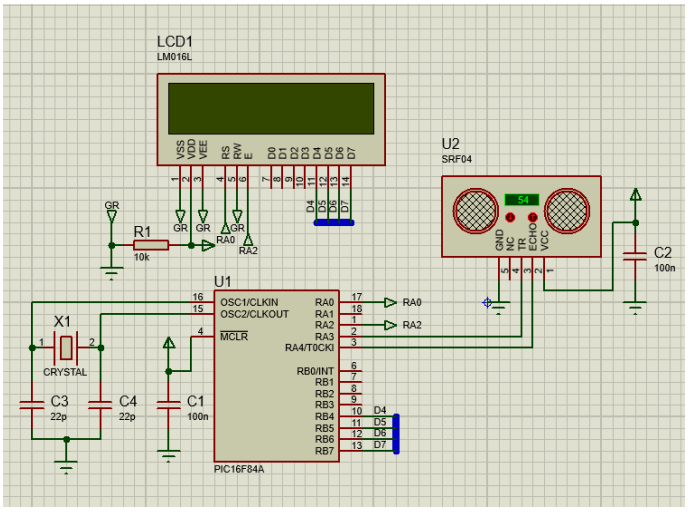
D= ((waitForEcho*17) / 10) cm

De esta forma ya solo resta realizar el código necesario para implementar dicho cálculo y lograr obtener el valor de la distancia en centímetros.

Para obtener dicho calculo, la librería implementa una subrutina que se encarga de hacer sumas continuas con el valor de *waitForEcho* un total de 17 veces. Al terminar este proceso pasa a la otra subrutina que se encargará de dividir el resultado obtenido por la subrutina anterior entre 10. Para ello esta subrutina hará restas continuas entre el valor y el divisor 10, contando cada resta y de esta forma obteniendo el resultado total que corresponderá a la distancia entre el objeto y el sensor. Para lograr estas operaciones de manera satisfactoria se va verificando el acarreo que se da por los cálculos de sumas y restas. Una vez finalizado esto, ya tenemos el valor correspondiente de la distancia.

Ahora veremos una explicación breve del proyecto convencional que se encuentra en algunos libros de texto:

PROYECTO EJEMPLO:
El proyecto que se encuentra en el libro Microcontrolador PIC16F84, nos muestra un código que hace uso de una interrupción que utiliza el TMR0 cuando se desborda. Esto se hace para el cálculo de la distancia que se necesita conocer.



Podemos observar que el diagrama de circuito es prácticamente el mismo, con la única diferencia que en nuestro proyecto implementamos un sensor que contiene un pin de prueba para simular a que distancia está el objeto del sensor.

En cuanto al código que se realiza en este proyecto ejemplo, pudimos observar que se inicia con la configuración de pines

correspondientes a *Trigger* y *Echo*, así como la configuración del TMR0 que se utilizará en el evento de interrupción. De igual forma se genera el disparo correspondiente en el pin del Trigger, esto para lograr que posteriormente se active el pin de *Echo*, con el cual se empezará a realizar las operaciones correspondientes para encontrar la distancia del objeto con respecto al sensor.
En la siguiente imagen podemos ver de una forma sencilla lo que se explicó anteriormente:

```
67  ;=================Configuracion de pines=======
68          call LCD_Inicializa
69          bsf STATUS,RP0
70          bcf Disparo
71          bsf Eco
72          movlw b'00000000'
73          movwf OPTION_REG
74          bcf STATUS,RP0
75  ;==============================================
76
77          bcf Disparo
78
79  Loop
80          clrf Distancia
81          bsf Disparo
82          call Retardo_20micros
83          bcf Disparo
84
```

En este punto lo que falta es comenzar a generar la interrupción cuando el pin de Echo este activado. Para ello se hace una subrutina que verifica si el valor de *Echo* está a 1 es decir a (5v), ya que una vez entrando a ese valor, se procederá a configurar y habilitar la interrupción del TMR0. Esta interrupción se va ejecutar cada 60 microsegundos, por lo cual se carga al TMR0 con el valor de una etiqueta llamada TMR0_Carga60micros. Este valor es un valor experimental que produce el efecto de desbordamiento cada 60 microsegundos.

```
84
85  Espera_Eco_1
86          btfss Eco
87          goto Espera_Eco_1
88          movlw TMR0_Carga60micros
89          movwf TMR0
90          movlw b'10100000'
91          movwf INTCON
92
93  Espera_Eco_0
94          btfsc Eco
95          goto Espera_Eco_0
96          clrf INTCON
97          call Visualiza
98          call Retardo_2s
99          goto Loop
100
```

Además podemos observar en la imagen anterior que una vez habilitada la interrupción en la línea 91, el programa pasa a otra subrutina que se encarga de comprobar que el estado del pin *Echo* ha vuelto a 0. Como el pin Echo va a tardar un cierto tiempo (proporcional a la distancia que el objeto esta del sensor) en poner su estado a 0, la interrupción se va a estar ejecutando una cierta cantidad de veces, y una variable contador se estará incrementando cada vez que lo haga.

En la siguiente imagen se visualiza el código de la interrupción que hace posible el cálculo de la distancia para el objeto. Dicho código no ocupa muchas líneas de programación, lo cual hace que el programa pueda correr de manera más eficiente.

```
;================================SUBRUTINA
ServicioInterrupcion
        movlw TMR0_Carga60micros
        movwf TMR0
        movlw .1
        addwf Distancia,F
        movlw MaximaDistancia
        btfsc STATUS,C
        movwf Distancia
        bcf INTCON,TOIF
        retfie
```

La diferencia de este proyecto convencional con el que realizamos, es la forma en cómo se realizan los cálculos para obtener el valor aproximado de la distancia. Ya que en nuestro proyecto se implementa una librería realizada desde cero en donde se tuvo que analizar cómo obtener una fórmula precisa para el cálculo. En cambio en este proyecto ejemplo, se hace uso de una interrupción que permite obtener el valor de la distancia, esto al ejecutarse continuamente cada 60 microsegundos cuando el pin Echo aún está activado (5v).

POR TANTO:

Basándonos en todo el fundamento teórico y analítico que hemos realizado al comparar estos dos proyectos, recomendamos que un futuro cercano se pueda implementar una librería para el sensor ultrasónico, la cual haga uso de la interrupción de TMR0 para el cálculo de la distancia. De igual forma dicha librería deberá proporcionar la lógica correspondiente para imprimir la distancia en términos de centenas, decenas y unidades; así como un valor mínimo y máximo que serán usados en la librería para determinar el rango de distancia que se desea configurar para dicho sensor.

IX. CONCLUCIONES

- Como primera conclucion se puede decir que se han logrados los objetivos principales en la realizacion del proyecto ya que se logro simplificar la manera en la que se programo el sensor haciendo uso de una libreria la cual realiza todo los calculos dejando asi el programa principal mas limpio y ordenado en las instrucciones.
- Se logró comprender el funcionamiento que se le puede

dar al sensor con el uso de una librería externa que permita la realización de cálculos para el mismo y de esta forma garantice un código más limpio y preciso en el Loop principal del programa.

- Se visualizó de forma clara la diferencia entre realizar la lógica para el funcionamiento del sensor a través de una librería, así como por medio de una interrupción por desbordamiento del TMR0.
- Al momento de programar la librería se percibieron algunas limitaciones de precisión en los resultados de los cálculos, ya que sólo se trabaja con 8 bits y se pierde información y cifras significativas cuando hay desbordamiento en las sumas sucesivas de la multiplicación o no se toma en cuenta el residuo en las restas sucesivas de la división. Aun así se llegó a una aproximación lo suficientemente buena de la distancia, apreciable en la prueba realizada con la simulación.

X. REFERENCIAS

Páginas Web:

[1] https://www.theengineeringprojects.com/2015/02/ultrasonic-sensor-library-proteus.html
Página de donde se descargó el módulo del sensor HC-SR04 para Proteus.

[2] https://simple-circuit.com/pic16f84a-hc-sr04-ultrasonic-sensor-ccs/
Donde se obtuvo el esquema del circuito y en sí la idea general del proyecto.

[3] https://electrosome.com/hc-sr04-ultrasonic-sensor-pic/
Donde se obtuvo información acerca de los cálculos.

[4] https://electrocrea.com/blogs/tutoriales/33306499-sensor-ultrasonico#:~:text=Los%20sensores%20de%20ultrasonidos%20son,la%20se%C3%B1al%20tarda%20en%20regresar.
De aquí se extrajo información teórica del funcionamiento del sensor ultrasónico.

Libros:

[5] *Microcontrolador PIC16F84 Desarrollo de Proyectos*, Enrique Palacios Municio, Fernando Remiro Domínguez, Lucas J. López Perez, vol. I. Mexico: Editorial RA-MA, 2004, p. 515.

XI. ANEXOS

- https://mega.nz/file/Gg5XECqZ#MjFfls-t3ko_WylzdDRHUcqhvH824GWk2sSQ-oyhbLQ
  Video de evidencia de funcionamiento del Proyecto en simulación con software Proteus 8 Professional.
- https://mega.nz/file/GwgHFYDY#VJkplZqa6X6fKXitRHGuduyz031irRZgAA2ZT-sYbQk
  Video de funcionamiento, proyecto paralelo programado con interrupciones, haciendo uso de sensor SRF04 en simulación con software Proteus 8 Professional.
- https://mega.nz/file/3hRl1IZC#m-1yTjxJ4F_slI7yt5vwUcIfS_9ve-Eyx4Mvlw0d6qk
  Link de descarga de Proyecto paralelo programado con interrupciones.
- https://mega.nz/file/P5xEQCJa#35VZc97Rca2W-5nqhpYTZ9-StHtDDT-uJRg-ylkkrMk
  Link de descarga de los archivos del módulo del Sensor HC-SR04 para Proteus 8 Professional.
  Los archivos que aquí se incluyen deben ser descomprimidos y colocados en el directorio de Labcenter Electronics. Este se encuentra en la dirección discoLocal(C) > archivosDePrograma(x86)
  Ó bien discoLocal(C) > archivosDePrograma.
  Ya en el directorio Labcenter Electronics se accede al directorio Proteus 8 Professional > LIBRARY.
  En este directorio LIBRARY es donde deben colocarse los archivos contenidos en el archivo .rar.
  Luego de eso reiniciar Proteus, si ya estaba abierto, para que actualice su base de datos.
  Más información en los links siguientes.
- https://www.theengineeringprojects.com/2015/02/ultrasonic-sensor-library-proteus.html
- https://www.theengineeringprojects.com/2018/04/how-to-add-new-library-in-proteus-8.html