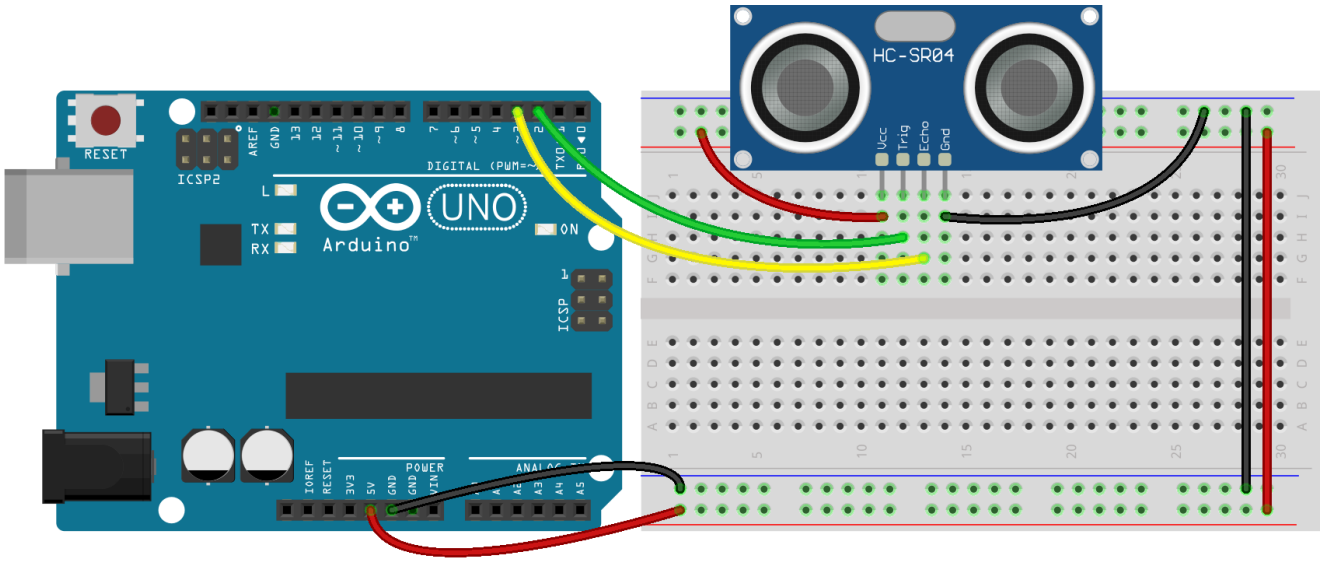
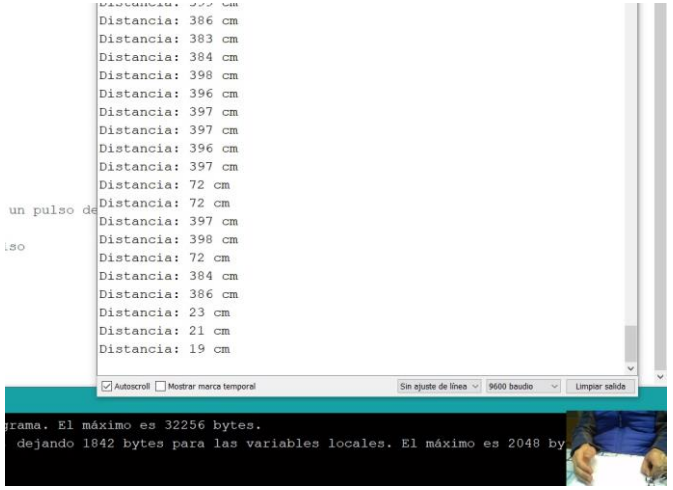
ARDUINO

Practica 8, MEDIDOR DE DISTANCIA



Se pretende medir al distancia desde la posición donde esta instalado el dispositivo HC-SR04

Podemos conectarlo en cualquiera de las patillas digitales. Es indistinto.



EL medidor HC-SR04

Estos sensores emiten un sonido de alta frecuencia, tan alta que el ser humano no es capaz de percibirla. Esta señal viaja por el aire y rebota al estar en contacto con otro objeto. Al rebotar, la señal regresa al sensor. Cuando el sensor detecta que la señal regresó, este hace un cálculo del tiempo que le llevó a la señal de audio viajar y regresar al destino, con esto se puede medir la distancia del objeto que provocó el rebote de la señal de audio.

Uno de los sensores más utilizados en el mundo *maker* (mundo relacionado con el desarrollo de proyectos con Arduino y otras plataformas), es el sensor HC-SR04.

Este sensor contiene dos transductores, uno actúa como emisor de una señal de sonido a 40KHz y el otro actúa como receptor. Cuando el receptor captura la señal produce un pulso que sirve como referencia para determinar la distancia que la señal viajó.

Características:

- Voltaje de operación: 5 V
- Corriente de operación: 15 mA
- Frecuencia de trabajo: 40 KHz
- Rango máximo de detección: 4 m
- Rango mínimo de detección: 2 cm
- Ángulo de medición: 15 grados
- Señal de disparo: 10 µs

De la programación:
interpretación y cálculo

```
const int Trigger = 2;          //definimos las conexiones del módulo para Disparo y  Recepción
const int Echo = 3;

int tiempo;                     //definimos las variables, el tiempo que se tarda en recibir el pulso desde la emisión
int distancia;                  //el valor que nos resulta de la formula aplicada

void setup() {
  Serial.begin(9600);
  pinMode(Trigger, OUTPUT);             //ponemos el disparador como salida
  pinMode(Echo, INPUT);                 //ponemos el receptor como entrada
  digitalWrite(Trigger, LOW);           //fijamos el disparador a LOW para evitar falsas medidas
                  }

void loop() {
  digitalWrite(Trigger, HIGH);          //activamos el emisor
  delayMicroseconds(10);                //durante  10 microsegundos, fijarse en la orden
  digitalWrite(Trigger, LOW);           //apagamos el pulso
  tiempo = pulseIn(Echo, HIGH);         //pulseIn nos mide el tiempo trascurrido desde la emisión hasta que
                                        //Echo se activo , o sea recibió la señal de vuelta y lo pone en tiempo
  distancia = tiempo/59;                //aplicamos la formula

  Serial.print("Distancia: ");          //imprimimos la distancia medida y esperamos 300 milis para renovar
  Serial.print(distancia);
  Serial.println(" cm");
  delay(300);
                  }
```

De la programación: interpretación y cálculo

Declaramos dos variables de tipo entero para los pines Trigger y Echo. Dentro de la función void el Trigger lo declaramos como salida y el Echo como entrada. El Trigger es el pin donde le enviaremos una señal digital con duración de 10 µs para que el módulo ultrasónico sepa que queremos enviar una señal, por lo que para iniciar lo pondremos en estado bajo. El pin Echo, es donde recibiremos la señal de respuesta del módulo, esta se activa al recibir el sonido de regreso.

En la función loop, mandaremos una señal en alto, luego esperamos 10 µs y la ponemos en bajo, posteriormente a esto calcularemos el tiempo en alto de la señal de respuesta, esta estará relacionada con el tiempo que la señal tardó en ir y regresar. Esta respuesta la guardamos en la variable llamada tiempo, y utilizando la función pulseIN(Echo, HIGH) obtendremos el tiempo en alto de la señal recibida. Ahora tendremos guardado el tiempo en que la señal viajó, rebotó y regresó.

Posteriormente tendremos que interpretar este tiempo como una distancia, para lo cual acudiremos a las matemáticas:

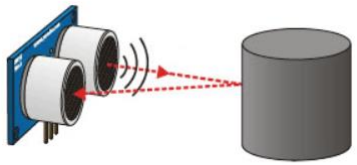
De la programación: interpretación y cálculo

$$velocidad = \frac{distancia}{tiempo}$$

La distancia es el tiempo que tarda en ir y volver hasta el sensor. La distancia del objetivo será la mitad de ese tiempo. La velocidad del sonido es la dinámica de propagación de las ondas sonoras. En la atmósfera terrestre es de 343 m/s (a 20 °C de temperatura, con 50 % de humedad y a nivel del mar).

La distancia al objetivo en centímetros la obtenemos al multiplicar el tiempo que tarda entre dos (en milisegundos) por la velocidad del sonido (en milisegundos por centímetro).

$$\frac{343m}{s} x \frac{100cm}{m} x \frac{1s}{1.000.000\mu s} = \frac{1cm}{29{,}2\mu s}$$



Es decir, el sonido tarda 29,2 micro segundos en recorrer un centímetro.

$$\text{Distancia (cm)} = \left(\frac{duración}{2}\right) * 0{,}0344(cm/ms)$$

El motivo de dividir por dos el tiempo (además de la velocidad del sonido en las unidades apropiadas, que hemos calculado antes) es porque hemos medido el tiempo que tarda el pulso en ir y volver, por lo que la distancia recorrida por el pulso es el doble de la que queremos medir.

Práctica 9. Con librería NewPing

Está librería podemos descargarla desde:

https://playground.arduino.cc/Code/NewPing/#.UzGBePl_sXw

Download

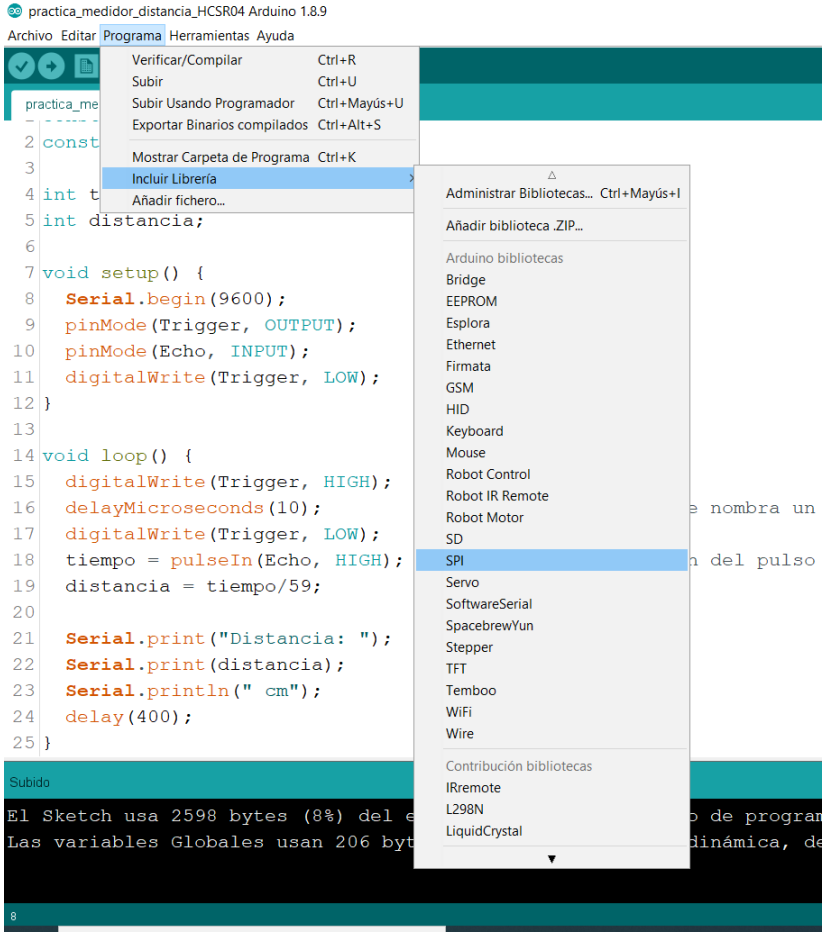
Download here: **Download NewPing Library**

Put the "NewPing" folder in "libraries\".

In the Arduino IDE, create a new sketch (or open one) and select from the menubar "Sktech->Import Library->NewPing".

Comprobamos en el programa de Arduino que no está instalada:

1. Programa
2. Incluir librería
   Listado completo



Con librería NewPing

Según el programador se usa así (esta definido en la pagina de descarga):

Constructor

NewPing sonar(trigger_pin, echo_pin [, max_cm_distance]);

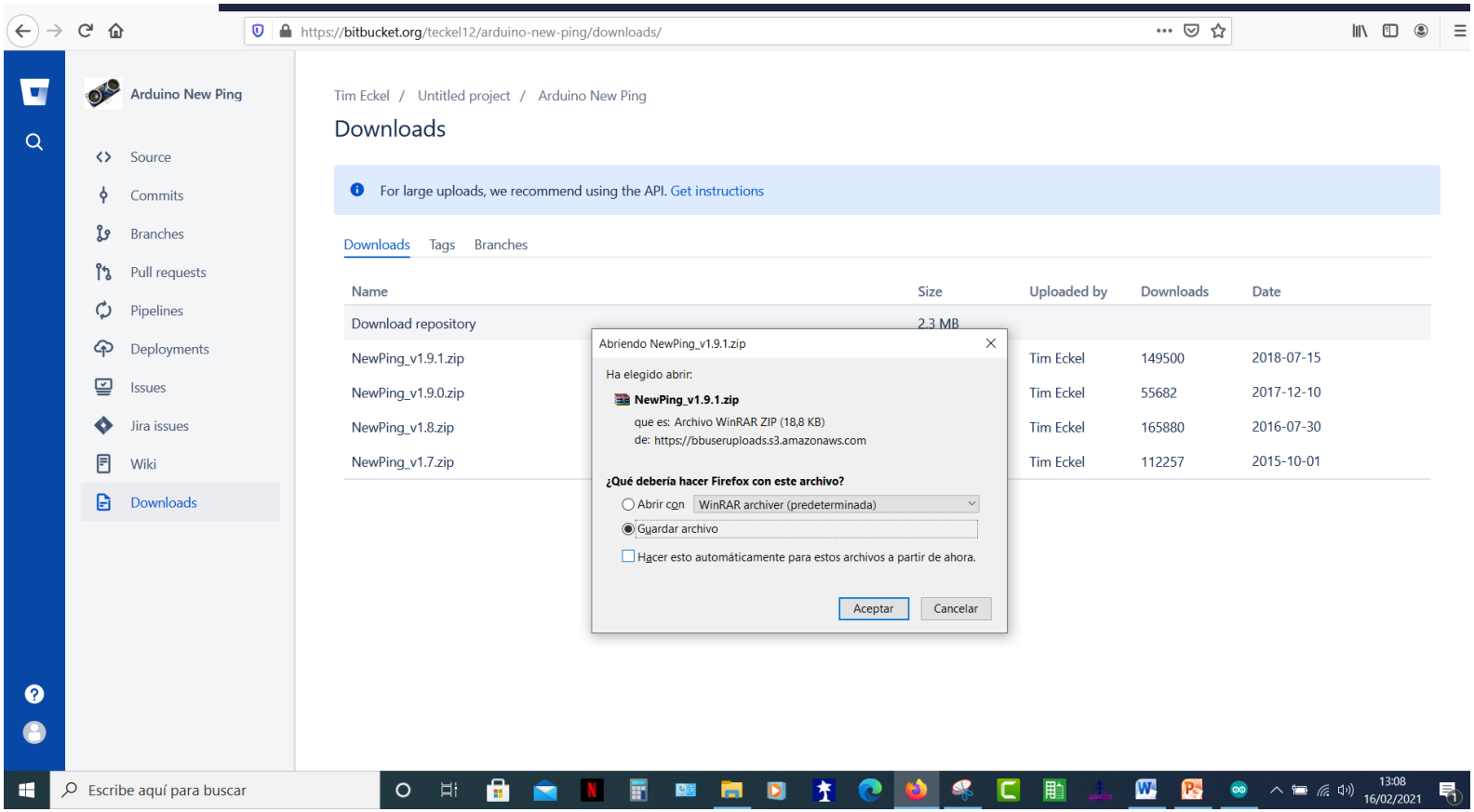
Example:

NewPing sonar(12, 11, 200);

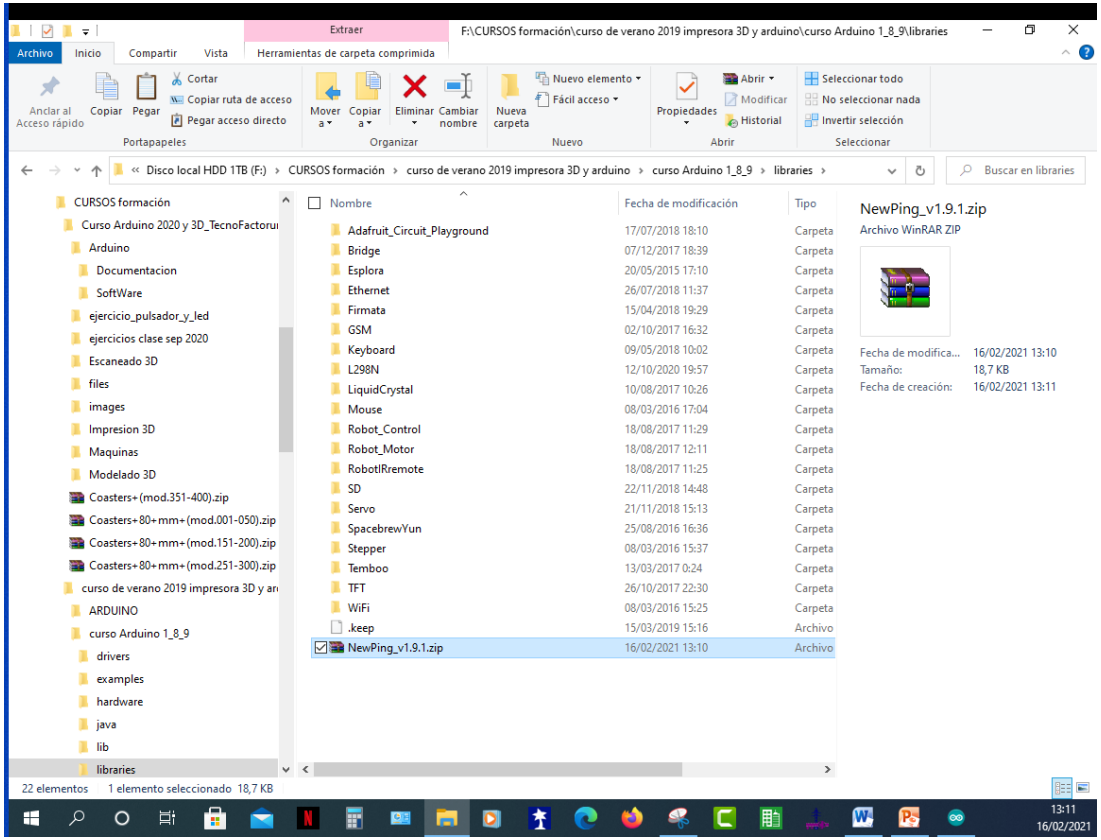
This initializes NewPing to use pin 12 for trigger output, pin 11 for echo input, with a maximum ping distance of 200cm. max_cm_distance is optional [default = 500cm]. If connecting using a single pin, specify the same pin for both trigger_pin and echo_pin as the same pin is doing both functions.

sonar.ping_cm(); - Send a ping, returns the distance in centimeters or 0 (zero) if no ping echo within set distance limit

Envía un ping, y nos muestra la distancia en centímetros, sino es cero, o se excede el límite fijado de medida.

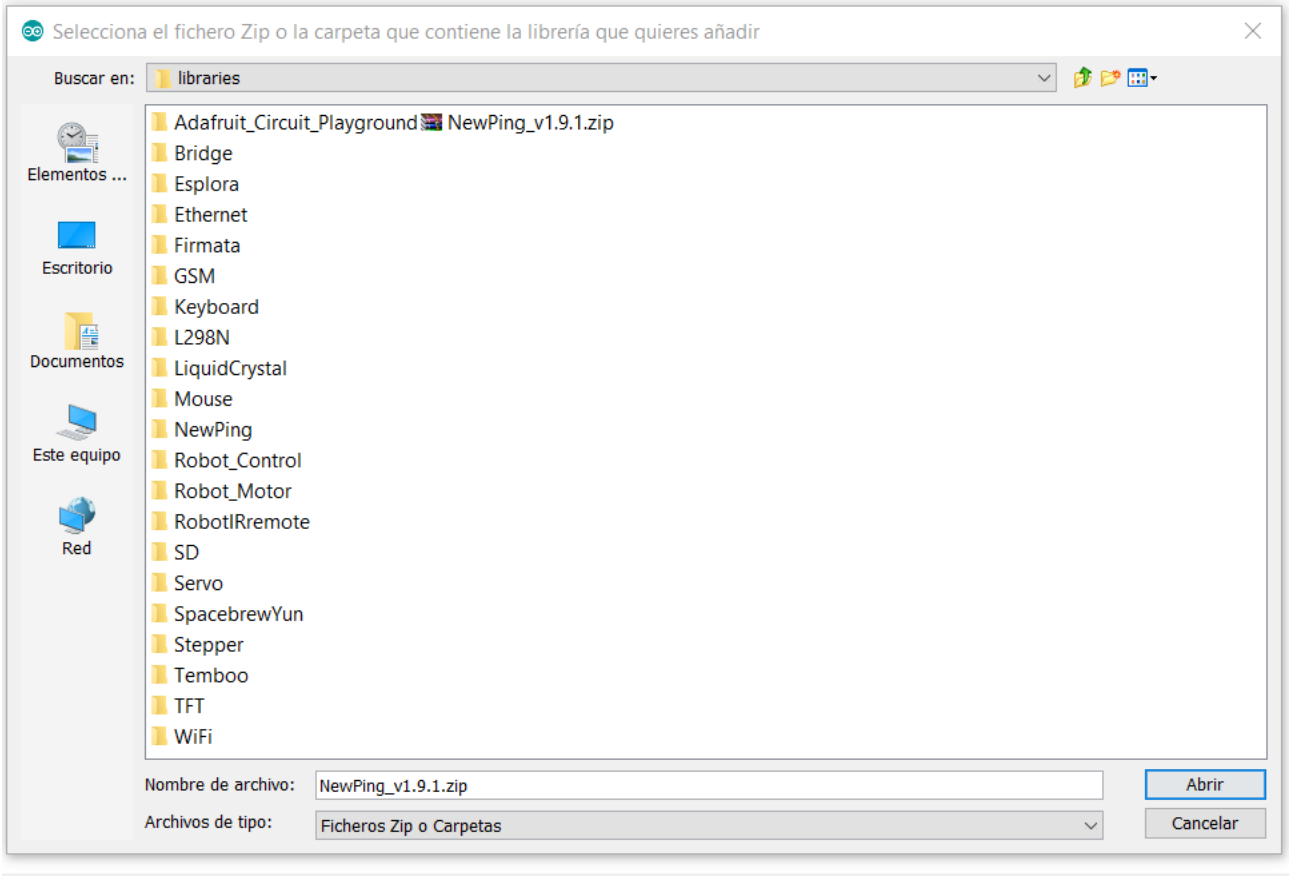


Descargamos la ultima versión , en la carpeta de Arduino - Libreries

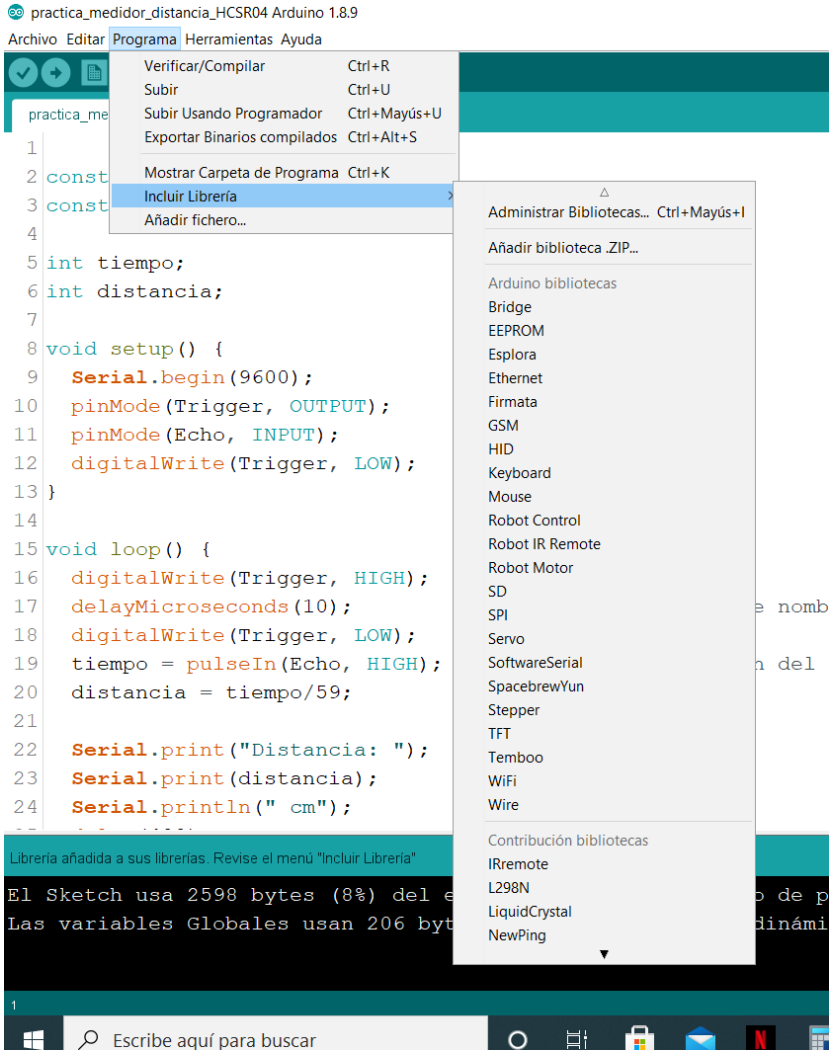


Copiada desde descargas a la carpeta de las librerías de Arduino.
Podemos descomprimir aquí, o dejar que la instale el programa , mediante el siguiente paso:

- Abrimos con la opción de seleccionar un fichero ZIP



- Y tiene que aparecer en el listado de librerías instalado:
- Ya la podemos incluir en nuestros programas, siempre y cuando sea compatible con la versión de Arduino y funcione, cosa que no ocurre siempre. Recordar que estamos trabajando con un programa de codigo abierto, donde cualquiera puede programar librería y ponerlas a disposición de los usuarios. Y lo que funciona perfectamente en un versión o en un equipo, no tiene porque comportarse igual en todos.
- Otra forma de instalar librerías es realizarlo online.



Con librería NewPing

```
#include <NewPing.h>                    //llamada a la librería

#define TRIGGER_PIN  12                 //define es una macro que sustituye el valor
#define ECHO_PIN     11                 //de la constante en todo el programa
#define MAX_DISTANCE 200                // sin = y ;


NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

void setup()
 {
  Serial.begin(9600);
 }

void loop()
 {
  delay(200);
  Serial.print(“Distancia medida = ");
  Serial.print(sonar.ping_cm());
  Serial.println("cm");
 }
```

Errores comunes 302

Uno de los errores más comunes que ocurren cuando copiamos código desde una web u otro sitio, es la inclusión de caracteres invisibles que nos provocan un fallo de compilación.

Para resolver esto podemos activar en:

1. Herramientas
2. Reparar codificación & recargar

Esto en ocasiones nos muestra los códigos invisibles que tenemos que borrar.

Sino toca escribir a mano el programa en un archivo nuevo.