Code:

```
int sensoroutput = A4; // the analog pin connected to the sensor
int ledoutput = 0; // pin connected to LED
int THRESHOLD = 100;
void setup()
{
pinMode(ledPin, OUTPUT);   // this function is used to declare led
connected pin as output
}
void loop()
{
int value = analogRead(sensoroutput); // function to read analog voltage
from sensor
if (value >= THRESHOLD)    // function to check voltage level from sensor
{
digitalWrite(ledoutput, HIGH);
delay(100); // to make the LED visible
}
else
digitalWrite(ledoutput, LOW);
}
```

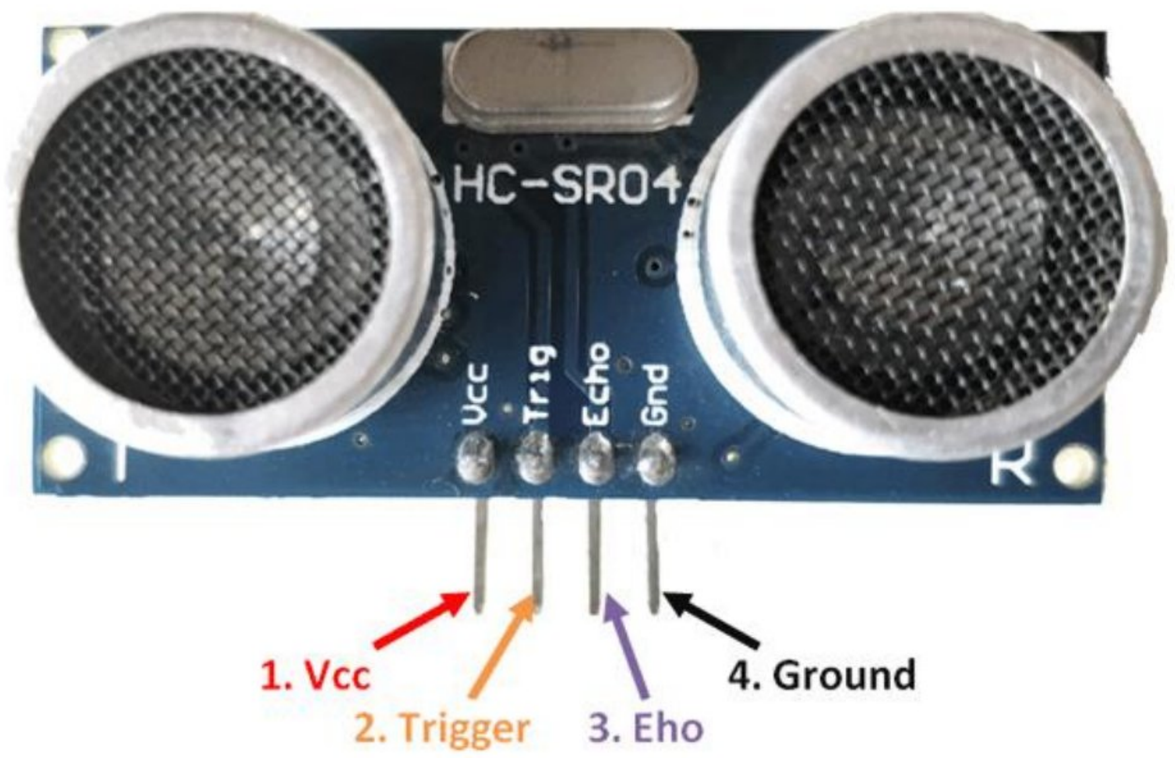
3. Ultrasonic Sensor:

Ultrasonic transducers or **ultrasonic sensors** are a type of acoustic sensor divided into three broad categories: transmitters, receivers and transceivers. Transmitters convert electrical signals into ultrasound, receivers convert ultrasound into electrical signals, and transceivers can both transmit and receive ultrasound.

- ## HC-SR04:

  HC-SR04 Ultrasonic (US) sensor is a 4 pin module, whose pin names are Vcc, Trigger, Echo and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver.

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.



HOW IT WORKS:

The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below:

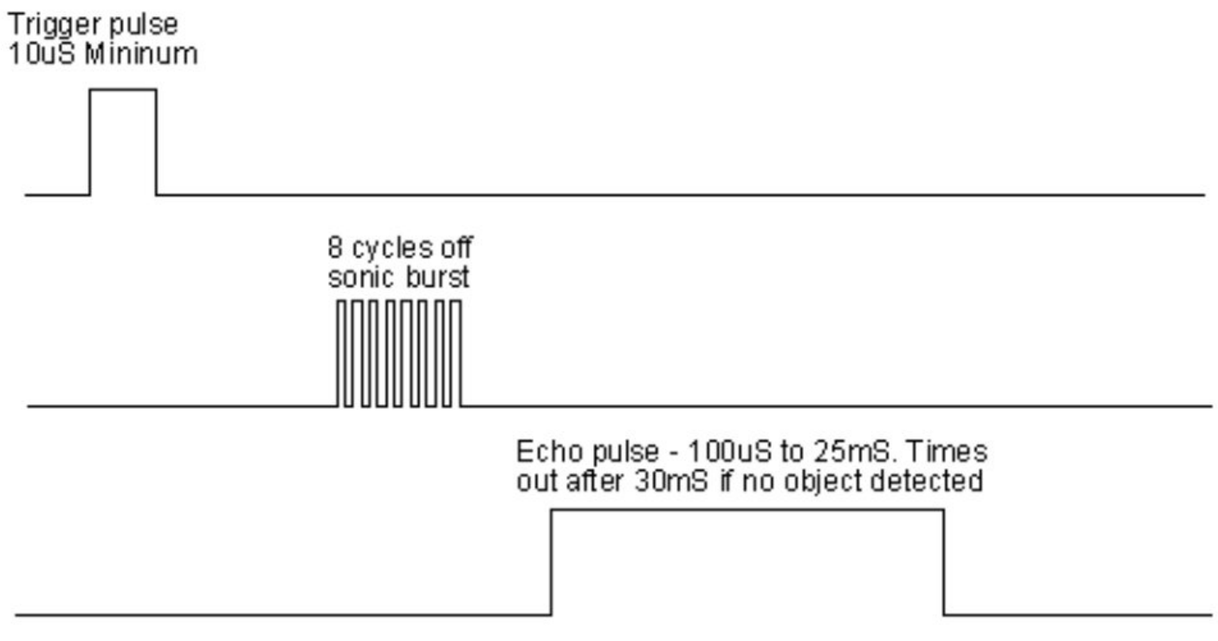
It emits an ultrasound at 40 000 Hz which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

HC-SR04 Sensor Features

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA

In order to generate the ultrasound, you need to set the Trig on a High State for 10 µs. That will send out an 8 cycle sonic burst which will travel at the speed sound and it will be received in the Echo pin. The Echo pin will output the time in microseconds the sound waves traveled.



A Trigger pulse burst of 10us is given to Trigger pin of module from the Arduino

The module Generates 40KHZ ultrasonic burst of 8 cycles which is Transmitted by the Transmit Transducer.

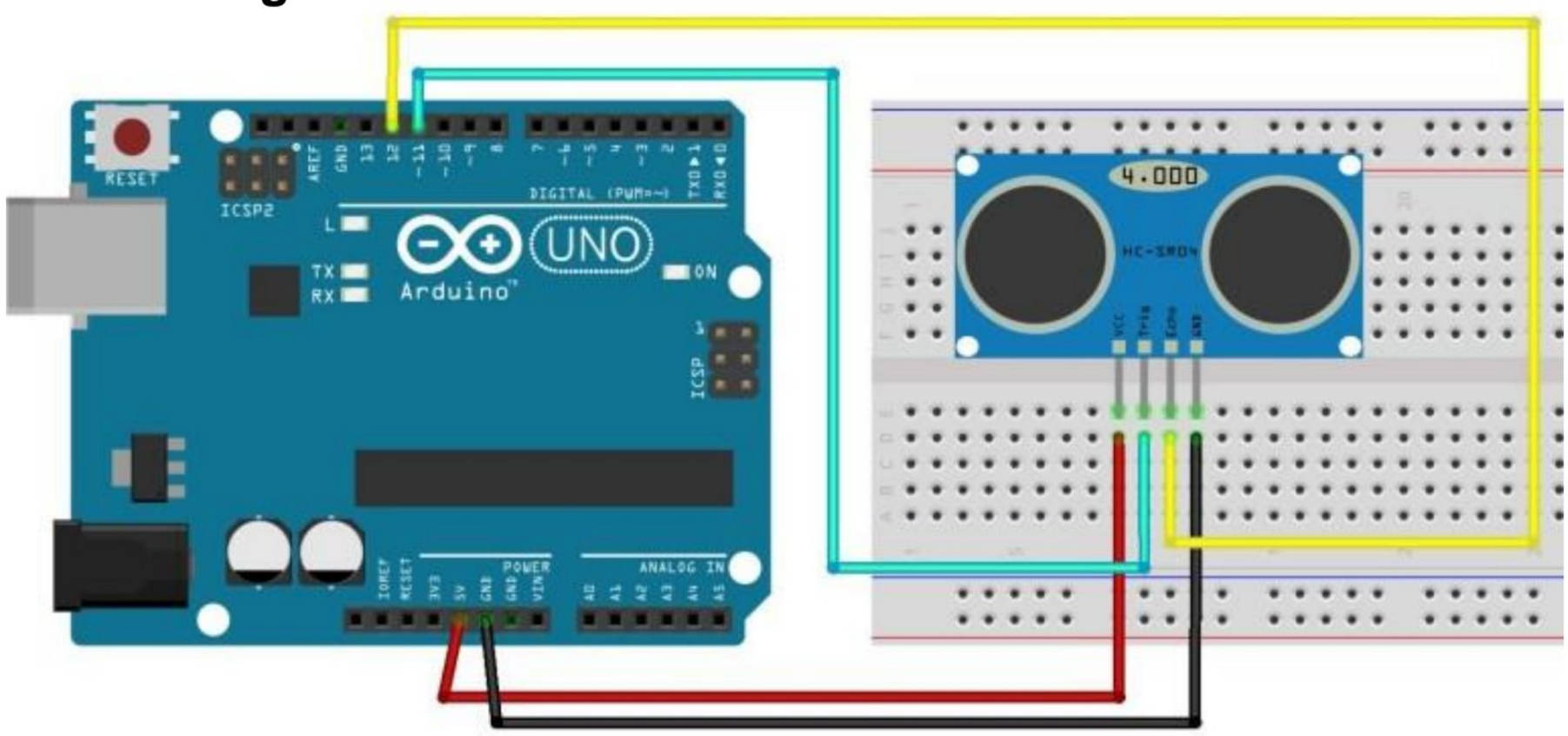
If any object detected,the burst returns back which is Received by Receive Transducer,sensed by Echo line & is used for distance calculation

3. Interfacing Ultrasonic Sensor with Arduino and measuring the distance of object placed in front of it

Components:

- Arduino Uno
- USB cable
- Breadboard
- Ultrasonic Sensor
- Jumper wires

Circuit Diagram:



Coding:

```
int trigPin = 11;    // Trigger
int echoPin = 12;    // Echo
long duration, cm, inches;

void setup() {
  //Serial Port begin
  Serial.begin (9600);
  //Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  digitalWrite(trigPin, LOW);
  delayMicroseconds(5);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the signal from the sensor: a HIGH pulse whose
  // duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(echoPin, INPUT);
  duration = pulseIn(echoPin, HIGH);

  // Convert the time into a distance
  cm = (duration/2) / 29.1;     // Divide by 29.1 or multiply by 0.0343
  inches = (duration/2) / 74;   // Divide by 74 or multiply by 0.0135

  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();

  delay(250);
}
```

How the Code Works:

pulseIn():

Reads a pulse (either HIGH or LOW) on a pin. For example,
if value is HIGH, pulseIn() waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing. Returns the length of the pulse in microseconds or gives up and returns 0 if no complete pulse was received within the timeout. (for pulses of 10 micro seconds to 3 minutes)

Syntax: `pulseIn(pin, value)`

`pulseIn(pin, value, timeout)`

First, you create variables for the trigger and echo pin called `trigPin` and `echoPin`, respectively. The trigger pin is connected to digital Pin 11, and the echo pins is connected to digital Pin 12:

```
int trigPin = 11;
int echoPin = 12;
```

You also create three variables of type long: `duration`, `cm`and `inch`. The `duration`variable saves the time between the emission and reception of the signal. The `cm`variable will save the distance in centimeters, and the `inch` variable will save the distance in inches.

```
long duration, cm, inches;
```

In the `setup()`, initialize the serial port at a baud rate of 9600, and set the trigger pin as an output and the echo pin as an input.

```
//Serial Port begin
Serial.begin (9600);
//Define inputs and outputs
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
```

In the `loop()`, trigger the sensor by sending a HIGH pulse of 10 microseconds. But, before that, give a short LOW pulse to ensure you'll get a clean HIGH pulse:

```
digitalWrite(trigPin, LOW);
delayMicroseconds(5);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

Then, you can read the signal from the sensor – a HIGH pulse whose duration is the time in microseconds from the sending of the signal to the reception of its echo to an object.

```
duration = pulseIn(echoPin, HIGH);
```

Finally, you just need to convert the duration to a distance. We can calculate the distance by using the following formula:

distance = (traveltime/2) x speed of sound

```
The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS

Or in inches: 13503.9in/s = 0.0135in/uS = 1/74in/uS
```

We need to divide the `traveltime` by 2 because we have to take into account that the wave was sent, hit the object, and then returned back to the sensor.

```
cm = (duration/2) / 29.1;
inches = (duration/2) / 74;
```

Finally, we print the results in the Serial Monitor:

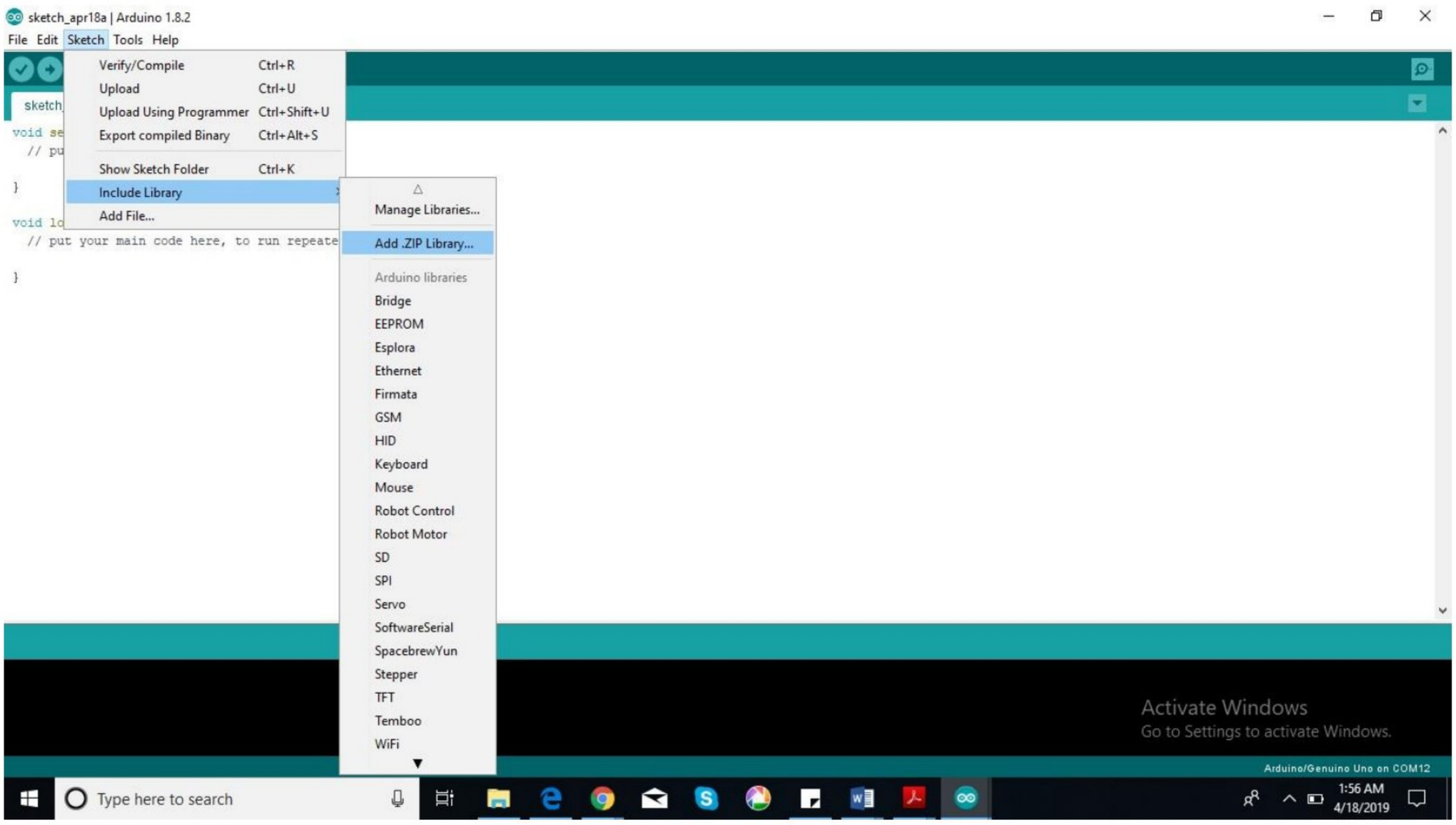
```
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
```

CODING FOR HC-SR04 USING NEW PING LIBRARY:

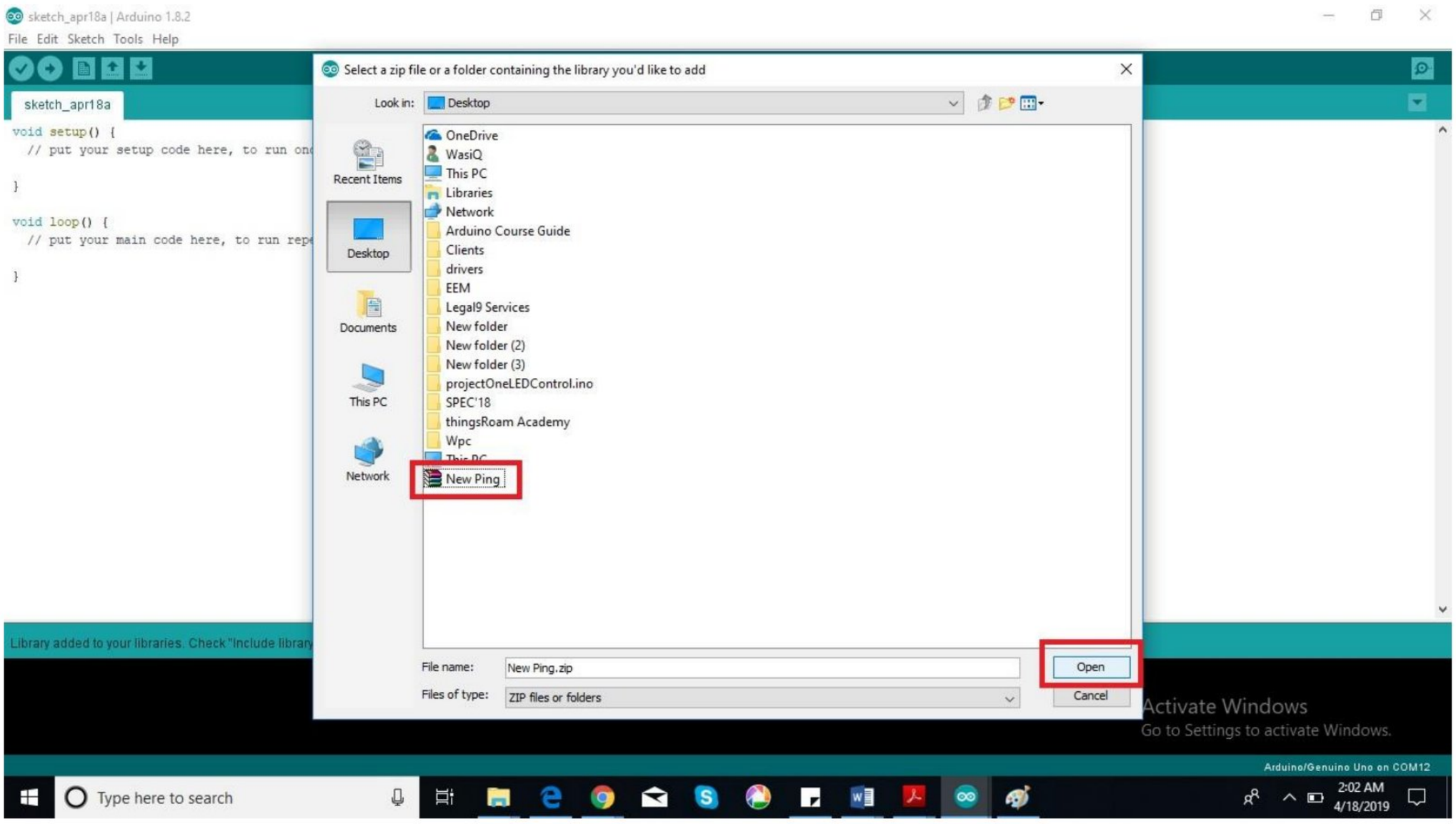
You can also use the NewPing library for HC-SR04. In order to use it, first you have to download the library and import it to the Arduino IDE.

To download the library click here.

Now open Arduino IDE. Goto Sketch> Include Library> Add .zip Library> Give path of NewPing library from your desktop

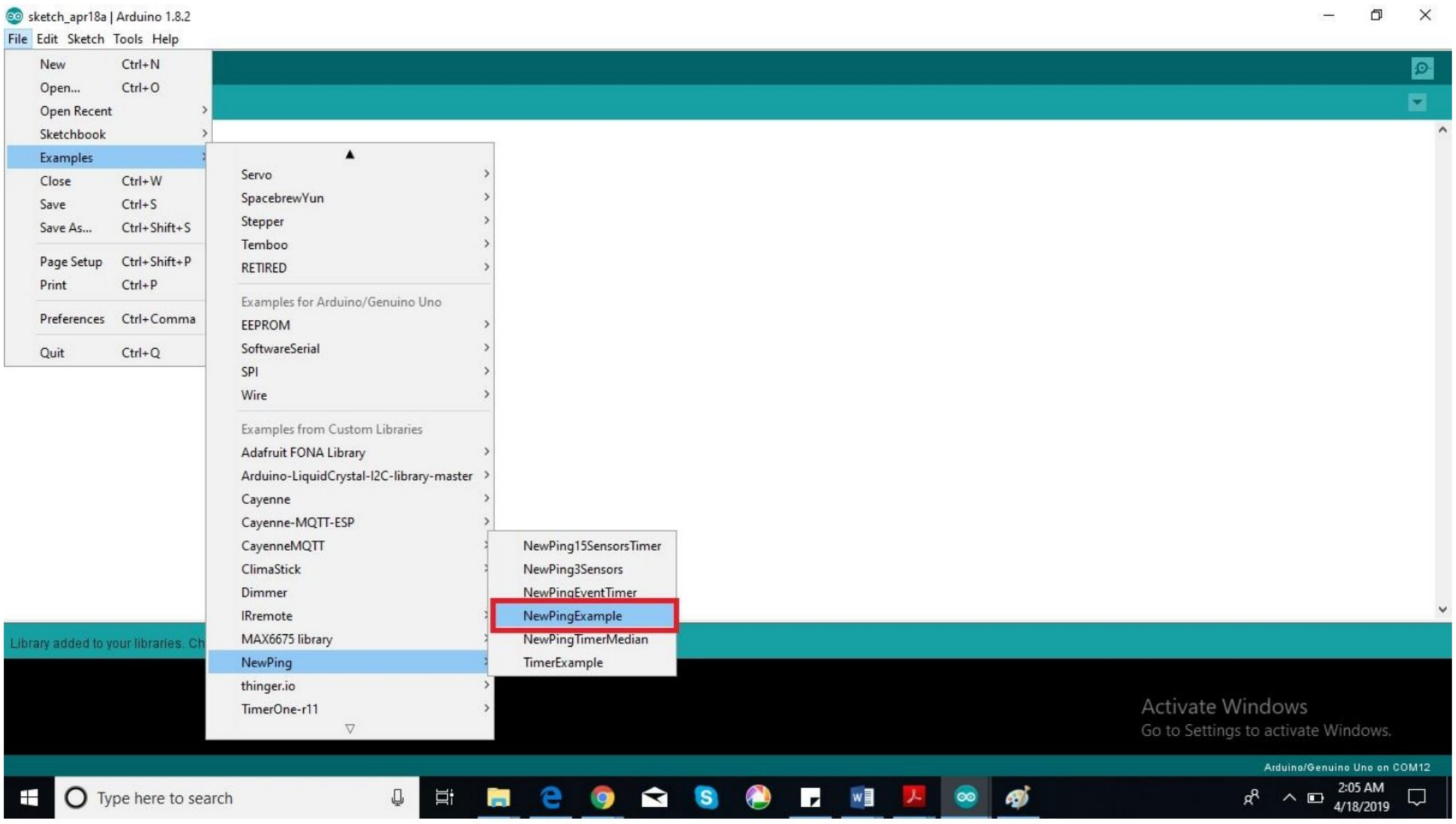


Prepared by:
M. Wasiq Pervez



Now, for the code, you can Goto **> Files> Examples> NewPing > NewPingExample >**

Prepared by:
M. Wasiq Pervez



Or copy and upload the following code:

```
#include <NewPing.h>

#define TRIGGER_PIN 11
#define ECHO_PIN 12
#define MAX_DISTANCE 200

// NewPing setup of pins and maximum distance
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);

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

void loop() {
   delay(50);
   unsigned int distance = sonar.ping_cm();
   Serial.print(distance);
   Serial.println("cm");
}
```

How the Code Works

Getting the distance to an object using the NewPing library is much simpler.

You start by including the NewPing library:

```
#include <NewPing.h>
```

Then, define the trigger and echo pin. The trigger pin is connected to the Arduino digital Pin 11, and the echo to Pin 12. You also need to define the `MAX_DISTANCE` variable to be able to use the library.

```
#define TRIGGER_PIN 11
#define ECHO_PIN 12
#define MAX_DISTANCE 200
```

Then, you create a `NewPing` instance called `sonar`:

```
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
```

In the `setup()`, you initialize the Serial communication at a baud rate of 9600.

```
Serial.begin(9600);
```

Finally, in the `loop()`, to get the distance you just need to use the `ping_cm()` method on the `sonar` object. This will give you the distance in centimeters.

```
unsigned int distance = sonar.ping_cm();
```

If you want to get the distance in inches you can use `sonar.ping_in()` instead.