

LECTURE 11

Ultrasonic Sensor and Application

Instructor: Osman Gul, Ph.D. Candidate

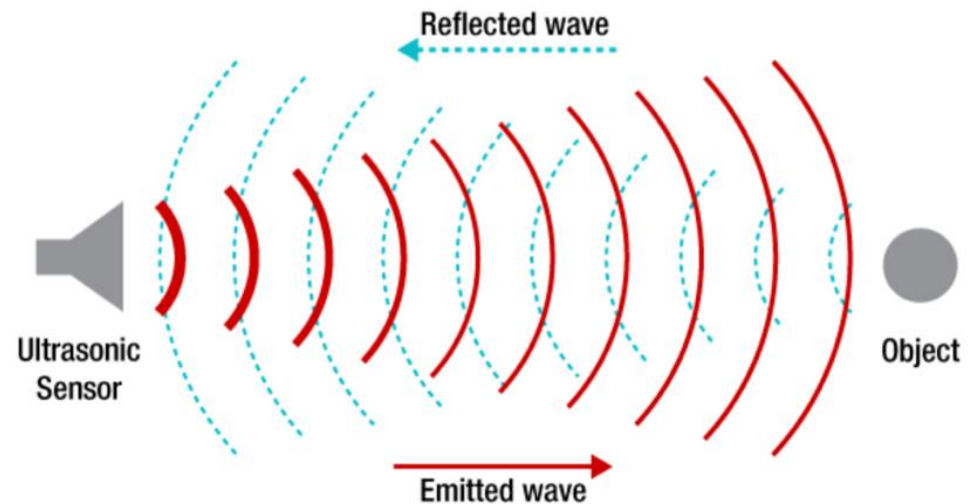
Department of Mechanical Engineering

Korea Advanced Institute of Science and Technology (KAIST)

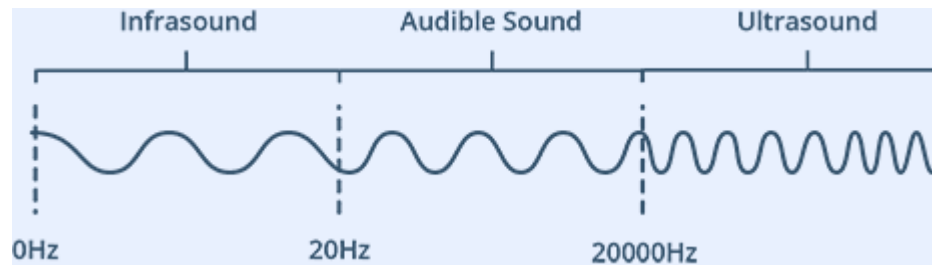
Syllabus:

Week	Date& Time*	Content
1		No Class (Course Add/Drop Period)
2	March 6 (Mon) 20-22	Orientation and Course Overview
3	March 13 (Mon) 20-22	Sensor Fundamentals
4	March 20 (Mon) 20-22	Arduino Fundamentals
5	March 27 (Mon) 20-22	Arduino Programming and Applications
6	April 3 (Mon) 20-22	Force Sensor and Application: Control Multiple LEDs
7	April 10 (Mon) 20-22	Light Sensor and Application: Solar Tracker
8	April 17	No Class (Midterm)
9	April 24 (Mon) 20-22	Humidity and Temperature Sensor and Application: Temperature Controlled Fan
10	May 1 (Mon) 20-22	Gas Sensor and Application: Smoke Detector
11	May 8 (Mon) 20-22	Sound Sensor and Application: Control LED by Clapping
12	May 15 (Mon) 20-22	Accelerometer Sensor and Application: Ping Pong Game
13	May 22 (Mon) 20-22	Ultrasonic Sensor and Application: Flappy Bird Game
14	May 29 (Mon) 20-22	Course Wrap-up
15	June 5	No Class (Final)
16	June 12	No Class (Final)

Ultrasonic sensor

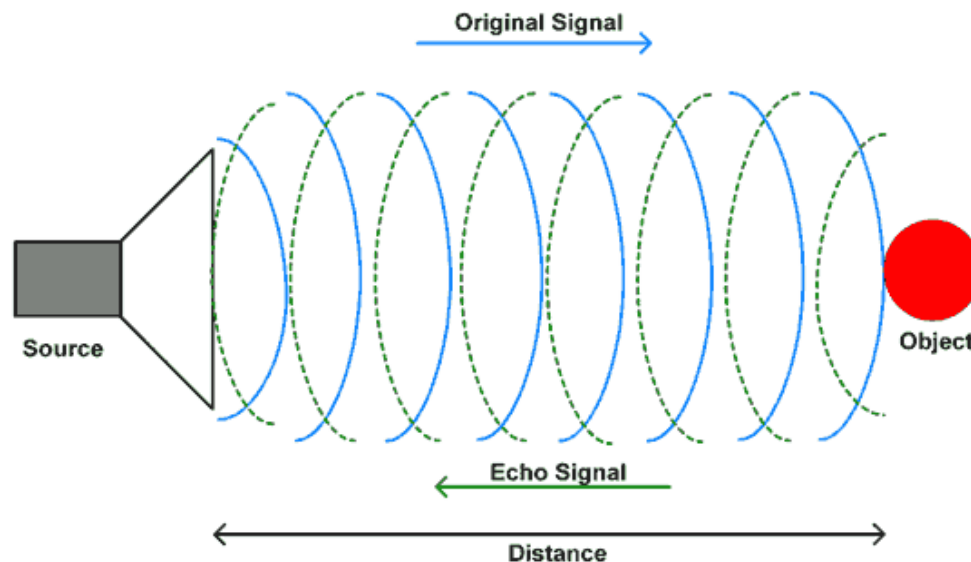


- Ultrasound is a high-pitched sound wave whose frequency exceeds the audible range of human hearing.

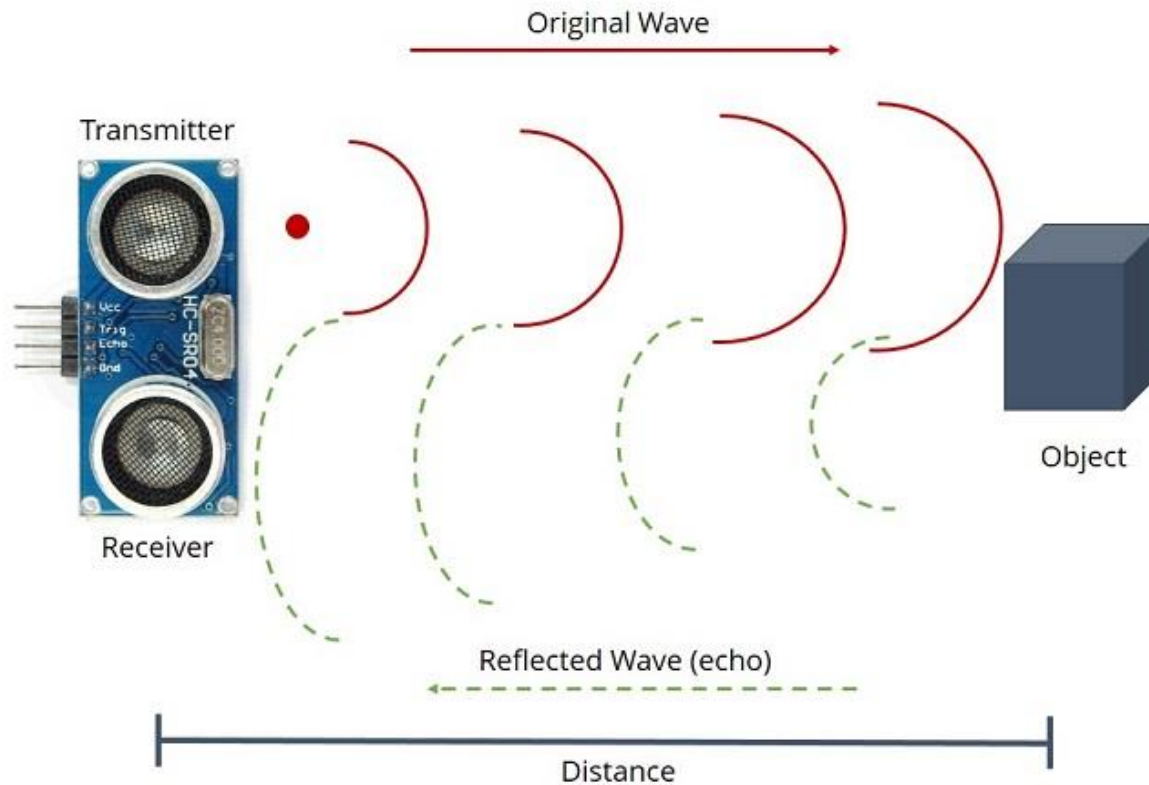


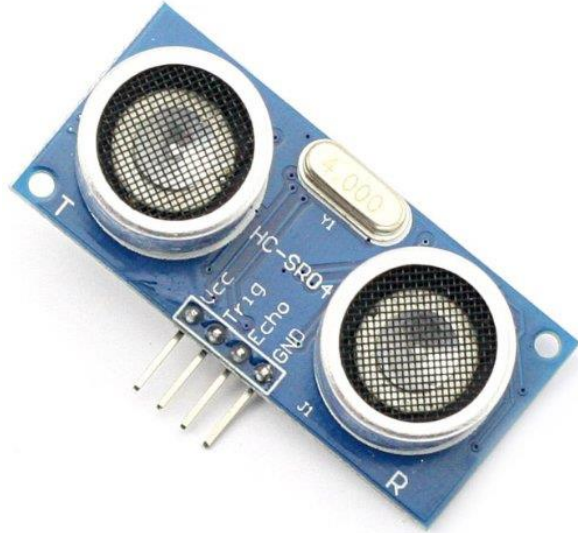
- Humans can hear sound waves that vibrate in the range of about 20 times a second (a deep rumbling noise) to 20,000 times a second (a high-pitched whistle). However, ultrasound has a frequency of more than 20,000 Hz and is therefore inaudible to humans.

- The ultrasonic sensor works on the principle of SONAR and RADAR system which is used to determine the distance to an object.
- An ultrasonic sensor generates high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver as shown in below figure.



- By measuring the time required for the echo to reach to the receiver, we can calculate the distance. This is the basic working principle of Ultrasonic module to measure distance.





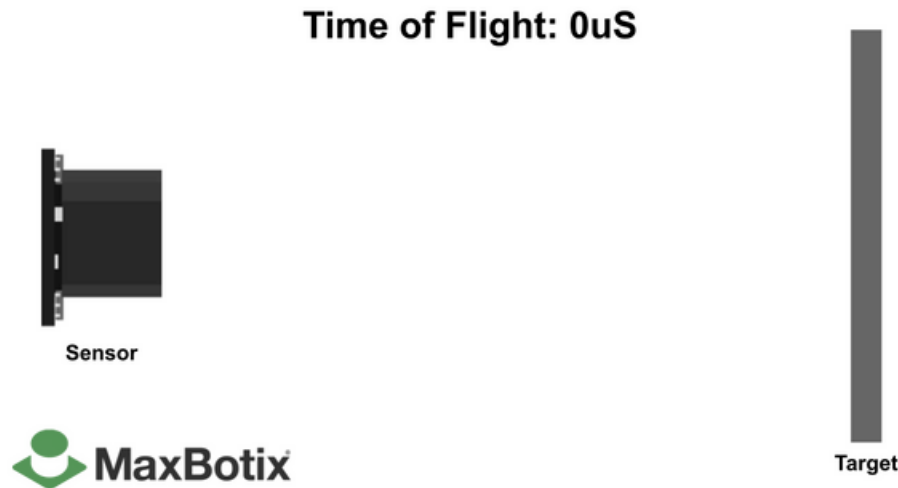
- HC-SR-04 has an ultrasonic transmitter, receiver and control circuit.
- In the ultrasonic module HCSR04, we have to give trigger pulse, so that it will generate ultrasound of frequency 40 kHz.
- After generating ultrasound i.e. 8 pulses of 40 kHz, it makes echo pin high. Echo pin remains high until it does not get the echo sound back. So the width of echo pin will be the time for sound to travel to the object and return back. Once we get the time we can calculate distance, as we know the speed of sound.

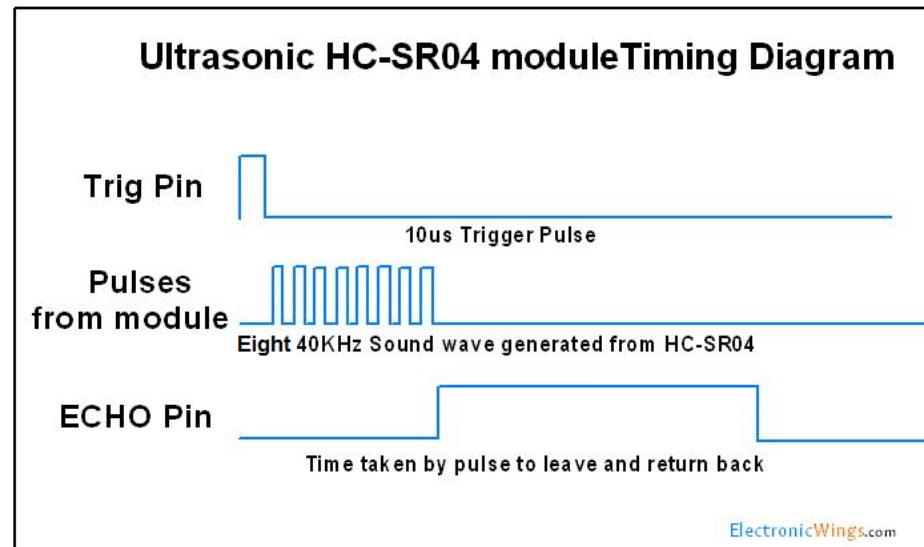
Ultrasonic Sensor HC-SR04 can measure up to range from 2 cm - 400 cm.

- The time between the transmission and reception of the signal allows us to calculate the distance to an object. This is possible because we know the sound's velocity in the air. Here's the formula:

distance to an object = ((speed of sound in the air)*time)/2

- speed of sound in the air at 20°C (68°F) = **343m/s**





1. We need to transmit trigger pulse of at least 10 us to the HC-SR04 Trig Pin.
2. Then the HC-SR04 automatically sends Eight 40 kHz sound wave and wait for rising edge output at Echo pin.
3. When the rising edge capture occurs at Echo pin, start the Timer and wait for falling edge on Echo pin.
4. As soon as the falling edge is captured at the Echo pin, read the count of the Timer. This time count is the time required by the sensor to detect an object and return back from an object.

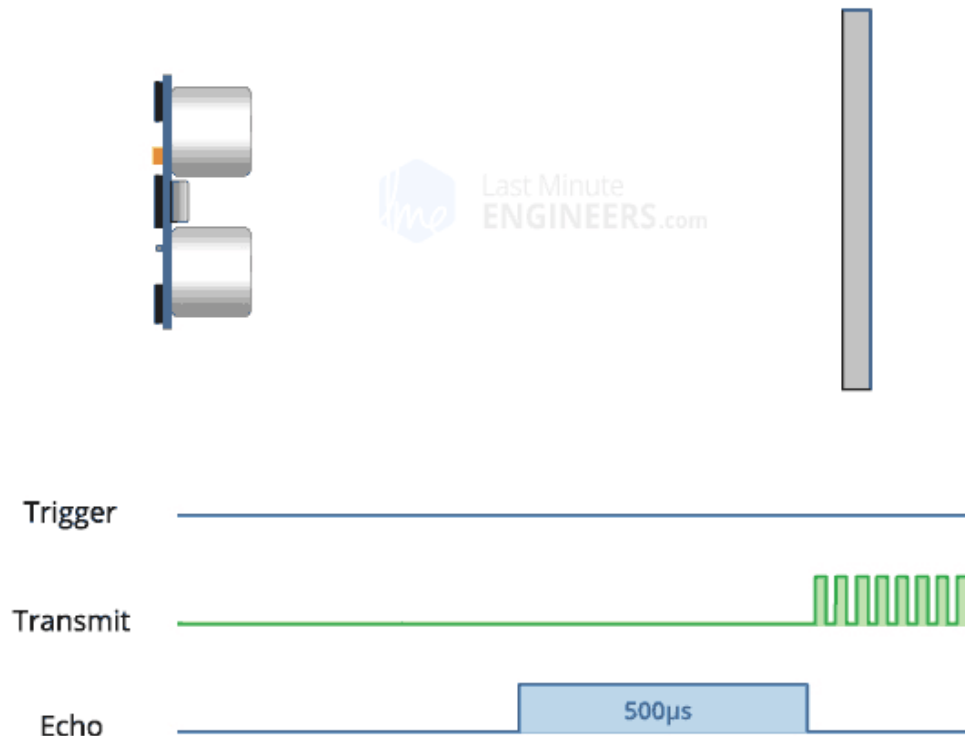
- It all starts when the trigger pin is set HIGH for $10\mu\text{s}$. In response, the sensor transmits an ultrasonic burst of eight pulses at 40 kHz. This 8-pulse pattern is specially designed so that the receiver can distinguish the transmitted pulses from ambient ultrasonic noise.
- These eight ultrasonic pulses travel through the air away from the transmitter. Meanwhile the echo pin goes HIGH to initiate the echo-back signal.
- If those pulses are not reflected back, the echo signal times out and goes low after 38ms (38 milliseconds). Thus a pulse of 38ms indicates no obstruction within the range of the sensor.

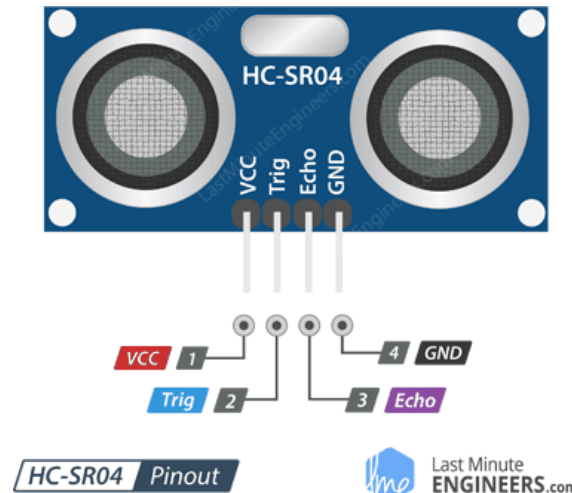


Last Minute
ENGINEERS.com



- If those pulses are reflected back, the echo pin goes low as soon as the signal is received. This generates a pulse on the echo pin whose width varies from $150\text{ }\mu\text{s}$ to 25 ms depending on the time taken to receive the signal.



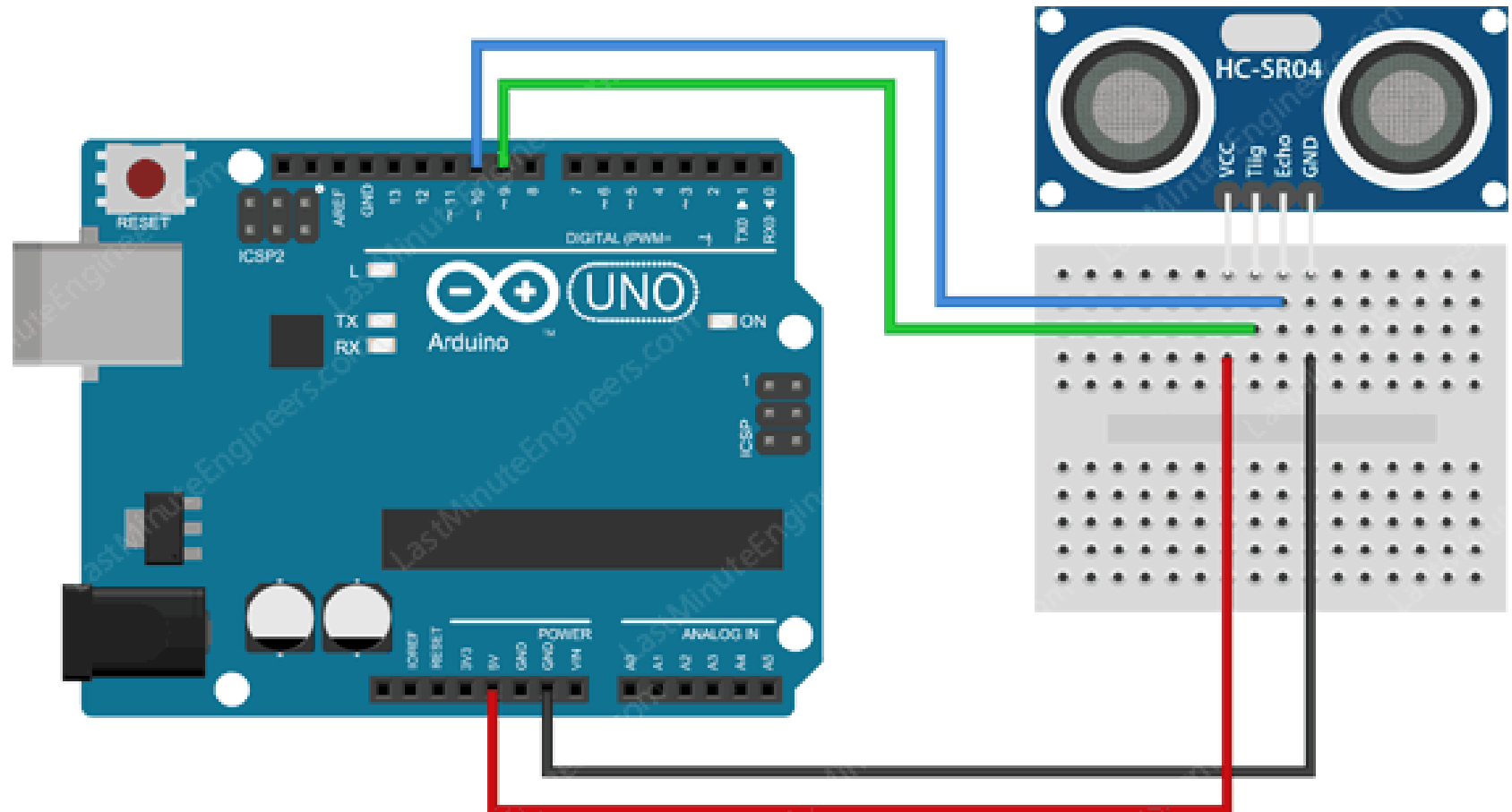


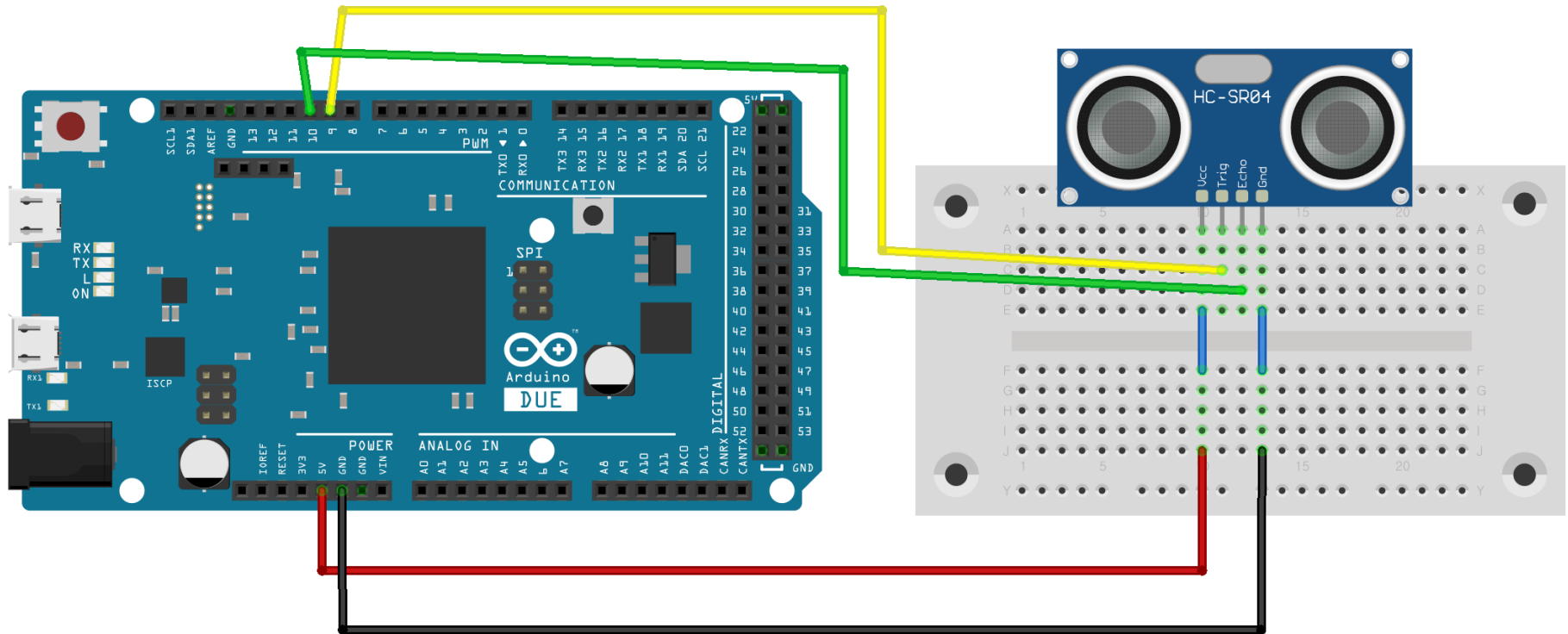
VCC supplies power to the HC-SR04 ultrasonic sensor. You can connect it to the 5V output from your Arduino.

Trig (Trigger) pin is used to trigger ultrasonic sound pulses. By setting this pin to HIGH for 10 μ s, the sensor initiates an ultrasonic burst.

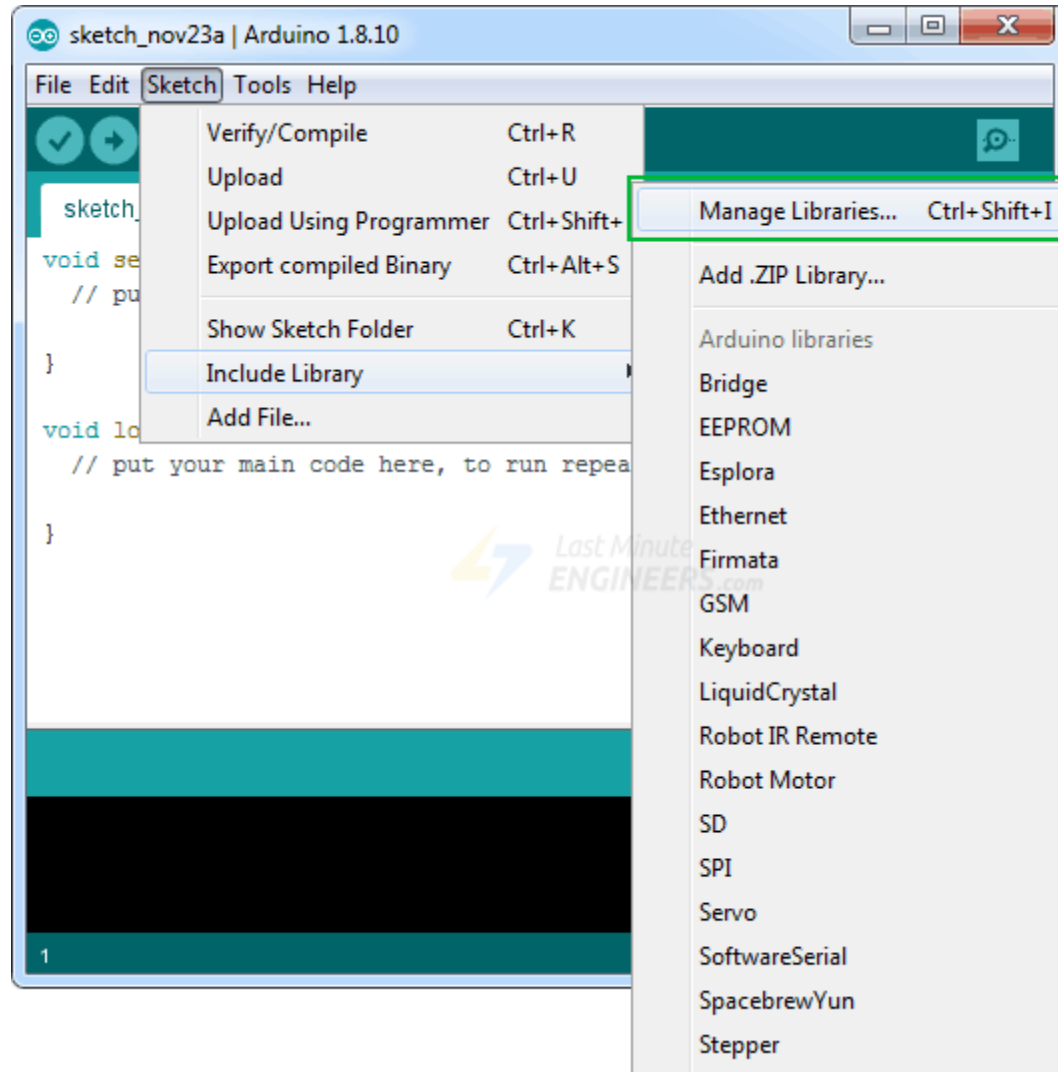
Echo pin goes high when the ultrasonic burst is transmitted and remains high until the sensor receives an echo, after which it goes low. By measuring the time the Echo pin stays high, the distance can be calculated.

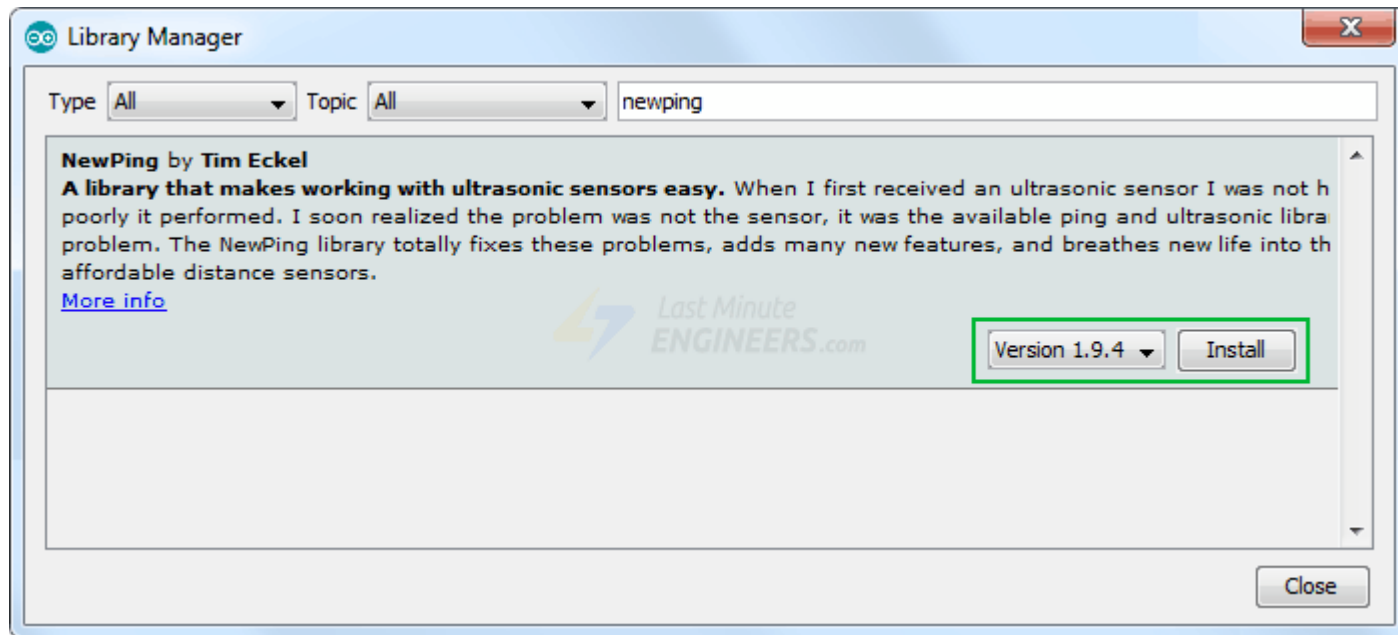
GND is the ground pin. Connect it to the ground of the Arduino.





fritzing






```
// Include NewPing Library
#include "NewPing.h"

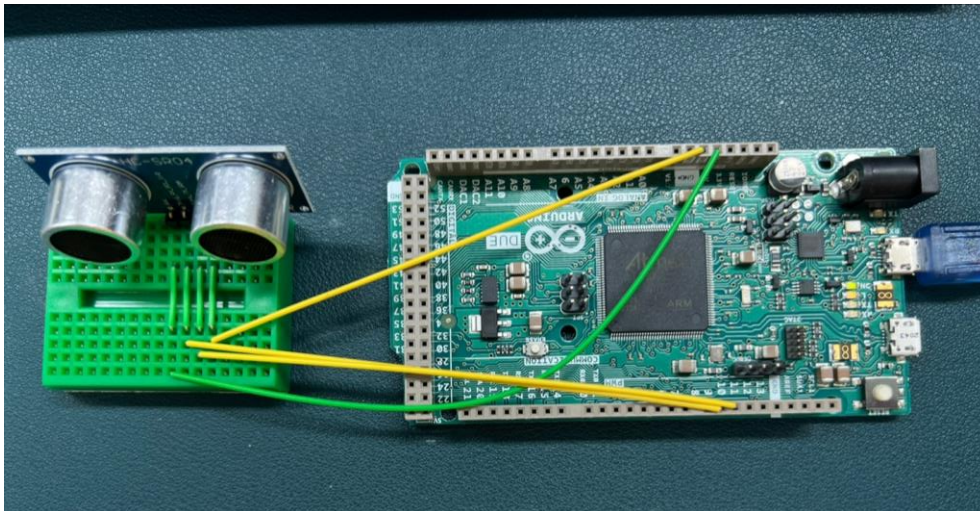
// Hook up HC-SR04 with Trig to Arduino Pin 9, Echo to Arduino pin 10
#define TRIGGER_PIN 9
#define ECHO_PIN 10

// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400

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

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

void loop() {
    Serial.print("Distance = ");
    Serial.print(sonar.ping_cm());
    Serial.println(" cm");
    delay(500);
}
```



COM13

```
Distance = 0 cm
Distance = 40 cm
Distance = 40 cm
Distance = 40 cm
Distance = 42 cm
Distance = 42 cm
Distance = 39 cm
Distance = 40 cm
Distance = 40 cm
Distance = 41 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 23 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 19 cm
Distance = 19 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 0 cm
Distance = 20 cm
Distance = 20 cm
Distance = 20 cm
```

First the Arduino pins are defined to which the Trig and Echo pins of the HC-SR04 are connected. We have also defined a constant called MAX_DISTANCE. It will set a maximum distance where pings beyond that distance are read as no ping "clear". MAX_DISTANCE is currently set to 400 [default = 500cm].

```
#define TRIGGER_PIN 9
#define ECHO_PIN 10
#define MAX_DISTANCE 400
```

After this, an instance of NewPing library named sonar is created.

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

In the loop, we simply call the ping_cm() function and print the result on the serial monitor. This function sends a ping and returns the distance in centimeters.

```
void loop() {
  Serial.print("Distance = ");
  Serial.print(sonar.ping_cm());
  Serial.println(" cm");
  delay(500);
}
```

```
// defines pins numbers
const int trigPin = 9;
const int echoPin = 10;
// defines variables
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
  Serial.begin(9600); // Starts the serial communication
}
void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.println(distance);
}
```

- First we have to define the Trig and Echo pins. In this case they are the pins number 9 and 10 on the Arduino Board and they are named trigPin and echoPin. Then we need a Long variable, named “duration” for the travel time that we will get from the sensor and an integer variable for the distance.

```
// defines pins numbers  
const int trigPin = 9;  
const int echoPin = 10;  
  
// defines variables  
long duration;  
int distance;
```

- In the setup we have to define the trigPin as an output and the echoPin as an Input and also start the serial communication for showing the results on the serial monitor.

```
void setup() {  
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output  
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input  
  Serial.begin(9600); // Starts the serial communication  
}
```

- In the loop first we have to make sure that the trigPin is clear so you have to set that pin on a LOW State for just 2 μ s. Now for generating the Ultra sound wave we have to set the trigPin on HIGH State for 10 μ s.

```
// Clears the trigPin  
digitalWrite(trigPin, LOW);  
delayMicroseconds(2);  
  
// Sets the trigPin on HIGH state for 10 micro seconds  
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);
```

- Using the ***pulseIn()*** function we read the travel time and put that value into the variable “duration”. This function has 2 parameters, the first one is the name of the Echo pin and for the second is the state of the pulse we are reading, either High or Low.

```
// Reads the echoPin, returns the sound wave travel time in microseconds  
duration = pulseIn(echoPin, HIGH);
```

- In this case, we need this set to it HIGH, as the HC-SR04 sensors sets the Echo pin to High after sending the 8 cycle ultrasonic burst from the transmitter. This actually starts the timing and once we receive the reflected sound wave the Echo pin will go to Low which stops the timing. At the end the function will return the length of the pulse in microseconds.
- For getting the distance we will multiply the duration by 0.034 and divide it by 2 as we explained this equation previously.

```
// Calculating the distance  
distance= duration*0.034/2;  
  
// Prints the distance on the Serial Monitor  
Serial.print("Distance: ");  
Serial.println(distance);
```

```
const int trigPin=11; //DECLARE TRIG PIN AT D11
int echoPin=10;      //DECLARE ECHO PIN AT D10
int safezone=10;
void setup()
{
  pinMode(trigPin,OUTPUT);
  pinMode(echoPin,INPUT);
  Serial.begin(9600);
}
void loop()
{
  long duration,cm;      //DECLARE VARIABLES TO STORE SENSOR O/P
  digitalWrite(trigPin,LOW); //MAKE THE TRIG PIN LOW
  delayMicroseconds(2);    //WAIT FOR FEW MICROSECONDS
  digitalWrite(trigPin,HIGH); //NOW SET THE TRIG PIN
  delayMicroseconds(5);    //WAIT FOR FEW MICROSECONDS UNTIL THE TRIG PULSE IS SENT
  digitalWrite(trigPin,LOW); //MAKE THE TRIG PIN LOW AGAIN
  duration=pulseIn(echoPin,HIGH); //MAKE ECHO PIN HIGH AND STORE THE BOUNCED PULSE IN VARIABLE DURATION
  cm=microsecondsToCentimeters(duration);
  long inch= cm/2.54;
  Serial.println(cm);
}
long microsecondsToCentimeters(long microseconds) //SPEED OF SOUND IS 29 uSEC PER CM
{
  return microseconds/29/2; //THE PULSE TRAVELS FROM THE SENSOR AND AGAIN COMES BACK SO WE DIVIDE IT BY 2 TO TAKE ONLY
  HALF OF THE TOTAL DISTANCE
}
```

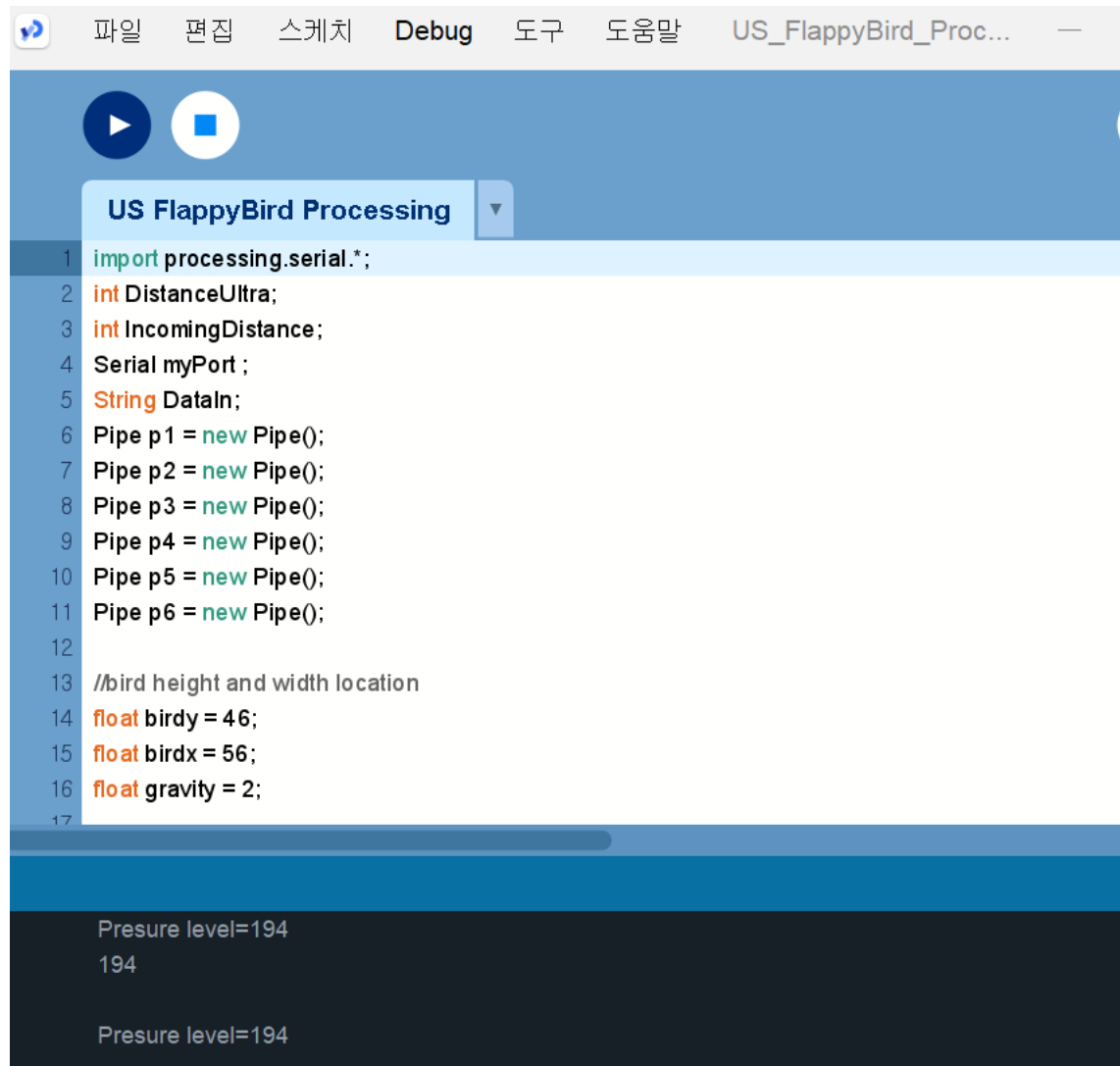

Create with code, everywhere

Processing is open source and is available for macOS, Windows, and Linux. Projects created with Processing are also cross-platform, and can be used on macOS, Windows, Android, Raspberry Pi, and many other Linux platforms.

Download Processing 4.2 for Windows

Windows • Intel 64-bit • 214 MB • ⓘ

<https://processing.org/download>



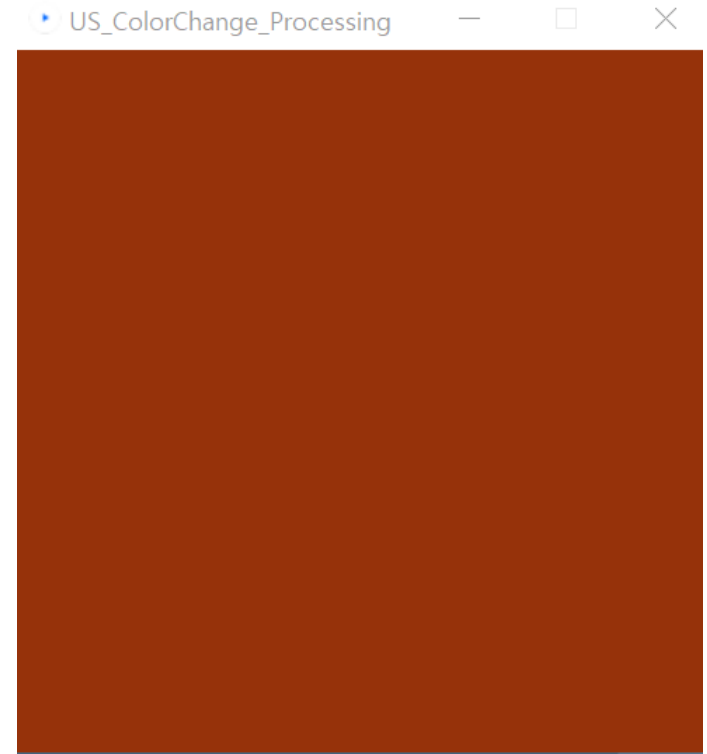
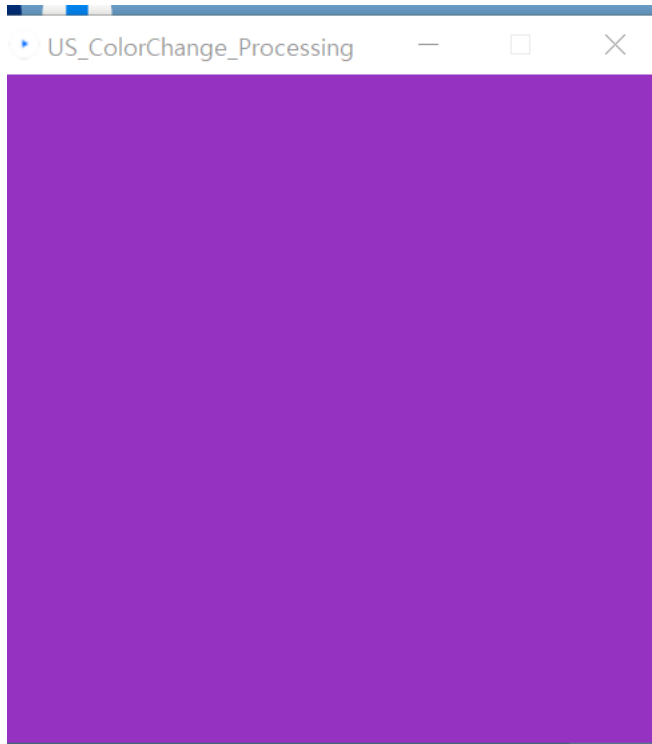
The screenshot shows the Processing IDE interface for a project named "US_FlappyBird_Proc...". The menu bar includes "파일", "편집", "스케치", "Debug", "도구", "도움말", and the project name. The toolbar has a play button and a stop button. The code editor displays the following code:

```
1 import processing.serial.*;
2 int DistanceUltra;
3 int IncomingDistance;
4 Serial myPort ;
5 String DataIn;
6 Pipe p1 = new Pipe();
7 Pipe p2 = new Pipe();
8 Pipe p3 = new Pipe();
9 Pipe p4 = new Pipe();
10 Pipe p5 = new Pipe();
11 Pipe p6 = new Pipe();
12
13 //bird height and width location
14 float birdy = 46;
15 float birdx = 56;
16 float gravity = 2;
17
```

Below the code editor, the console output shows:

```
Presure level=194
194
Presure level=194
```

Color Change with Ultrasonic Sensor



Flappy bird game with Ultrasonic Sensor

