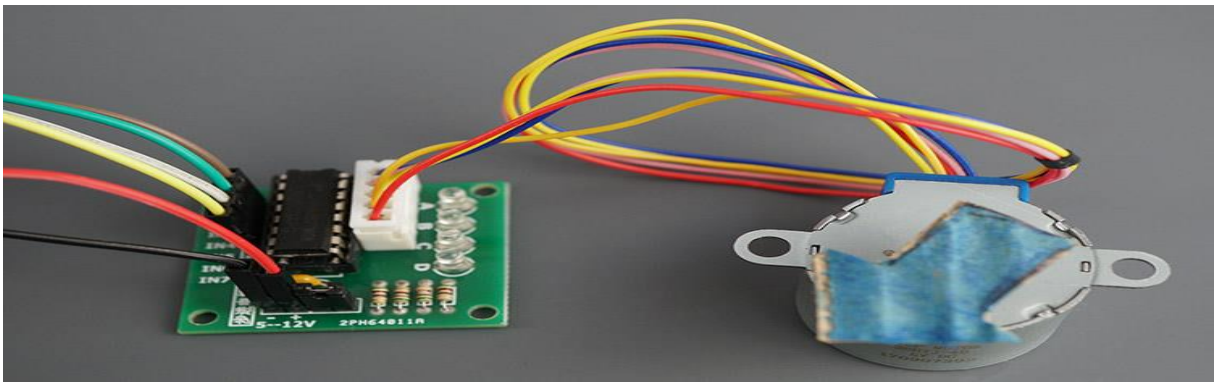
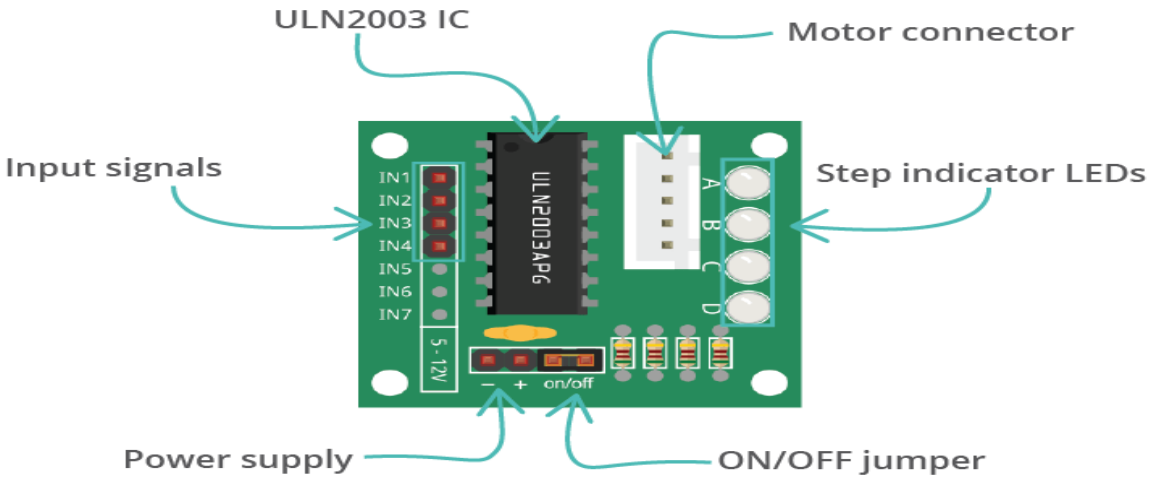
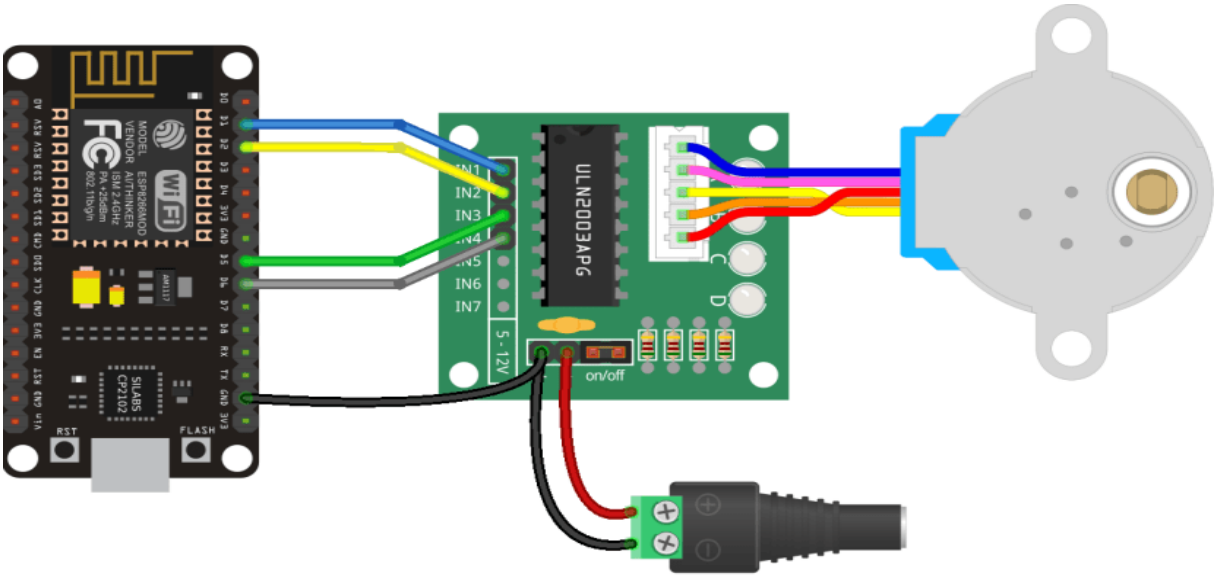
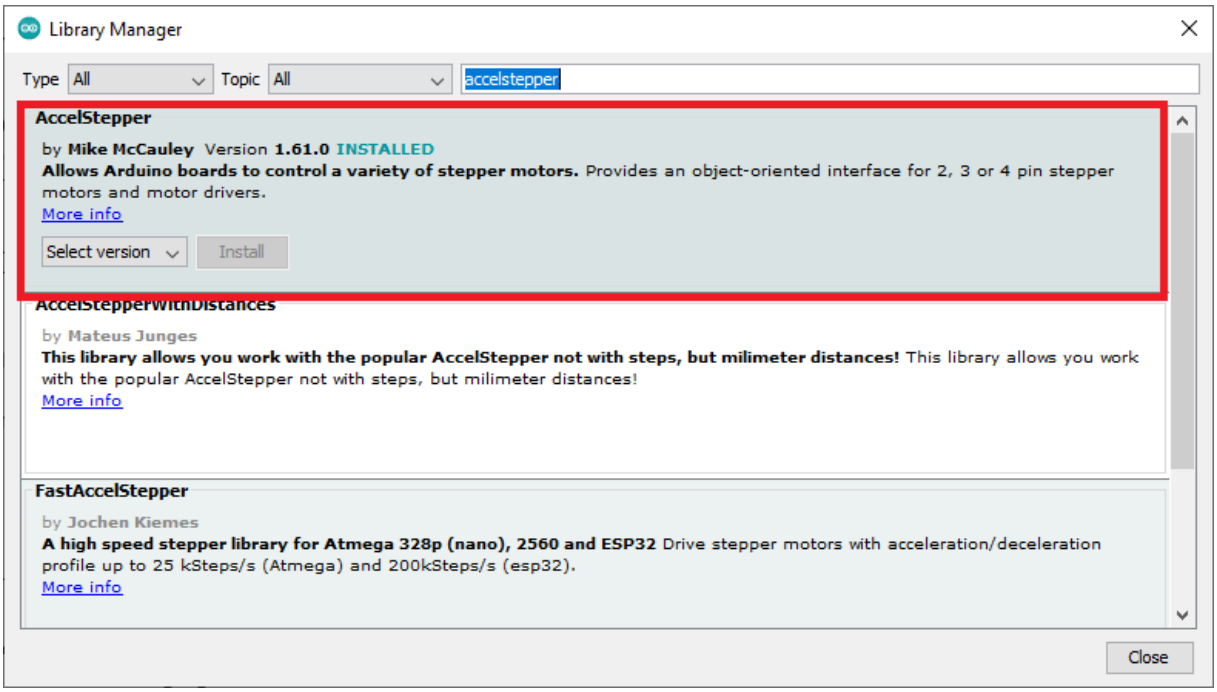
1. CONNEXION MOTEUR PAS A PAS







DRIVER



CODE SOURCE

```
/*
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp8266-
nodemcu-stepper-motor-28byj-48-uln2003/

  Permission is hereby granted, free of charge, to any person obtaining a
copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included
in all
  copies or substantial portions of the Software.

  Based on Stepper Motor Control - one revolution by Tom Igoe
*/

#include <AccelStepper.h>

const int stepsPerRevolution = 2048;  // change this to fit the number of
steps per revolution

// ULN2003 Motor Driver Pins
#define IN1 5
#define IN2 4
#define IN3 14
#define IN4 12

// initialize the stepper library
AccelStepper stepper(AccelStepper::HALF4WIRE, IN1, IN3, IN2, IN4);

void setup() {
```

```
  // initialize the serial port
  Serial.begin(115200);

  // set the speed and acceleration
  stepper.setMaxSpeed(500);
  stepper.setAcceleration(100);
  // set target position
  stepper.moveTo(stepsPerRevolution);
}

void loop() {
  // check current stepper motor position to invert direction
  if (stepper.distanceToGo() == 0){
    stepper.moveTo(-stepper.currentPosition());
    Serial.println("Changing direction");
  }
  // move the stepper motor (one step at a time)
  stepper.run();
}
```

EXPLICATION

Comment fonctionne le code

Tout d'abord, incluez le AccelStepper.h bibliothèque.

```
#include <AccelStepper.h>
```

Définissez les pas par tour de votre moteur pas à pas, dans notre cas, il s'agit de 2048 :

```
const int stepsPerRevolution = 2048;  // change this
```

```
to fit the number of steps per revolution
```

Définissez les broches d'entrée du moteur. Dans cet exemple, nous nous connectons aux GPIO 5, 4, 14 et 12, mais vous pouvez utiliser n'importe quel autre GPIO approprié.

```
#define IN1 5
```

```
#define IN2 4
```

```
#define IN3 14
```

```
#define IN4 12
```

Initialiser une instance de la bibliothèque AccelStepper appelée pas à pas. Passer en arguments : AccelStepper::HALF4WIRE pour indiquer que nous contrôlons le moteur pas à pas avec quatre fils et les broches d'entrée.

Dans le cas du moteur pas à pas 28BYJ-48, l'ordre des broches est EN 1, IN3, EN 2, IN4-cela peut être différent pour votre moteur.

```
AccelStepper stepper(AccelStepper::HALF4WIRE, IN1,
IN3, IN2, IN4);
```

Dans le mettre en place(), initialisez le moniteur série à un débit en bauds de 115 200.

```
Serial.begin(115200);
```

Réglez la vitesse maximale du moteur pas à pas à l'aide de la setMaxSpeed() méthode. Passez en argument la vitesse en pas par seconde.

```
stepper.setMaxSpeed(500);
```

Comme mentionné dans la documentation, le setMaxSpeed() La méthode définit la vitesse maximale autorisée. le Cours() fonction accélérera jusqu'à la vitesse définie par cette fonction.
Réglez l'accélération à l'aide de la setAcceleration() méthode. Passer en argument l'accélération en pas par seconde par seconde.

```
stepper.setAcceleration(100);
```

Ensuite, utilisez le déménager à() méthode pour définir une position cible. Puis le Cours() La fonction essaiera de déplacer le moteur (au plus un pas par appel) de la position actuelle à la position cible définie par l'appel le plus récent à cette fonction. Nous fixons la position cible à 2048 (ce qui est un tour complet dans le cas de ce moteur).

```
stepper.moveTo(stepsPerRevolution);
```

Dans le boucler(), nous allons faire tourner le moteur pas à pas dans le sens horaire et antihoraire.
Tout d'abord, nous vérifions si le moteur a déjà atteint sa position cible. Pour ce faire, nous pouvons utiliser le distanceToGo() fonction qui renvoie les pas de la position actuelle à la position cible.
Lorsque le moteur atteint sa position cible, cela signifie que le distanceToGo() la fonction reviendra 0 et l'instruction if suivante sera vraie.

```
if (stepper.distanceToGo() == 0){
```

Lorsque le moteur atteint sa position cible, nous définissons une nouvelle position cible, qui est la même que la position actuelle mais dans la direction opposée.

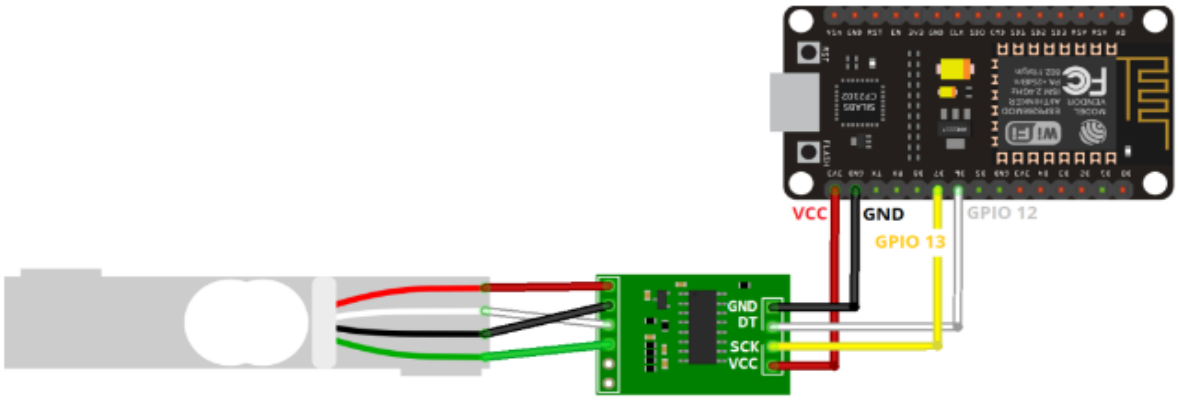
```
stepper.moveTo(-stepper.currentPosition());
```

Enfin, appelez stepper.run() pour déplacer le moteur pas à pas dans le boucler().

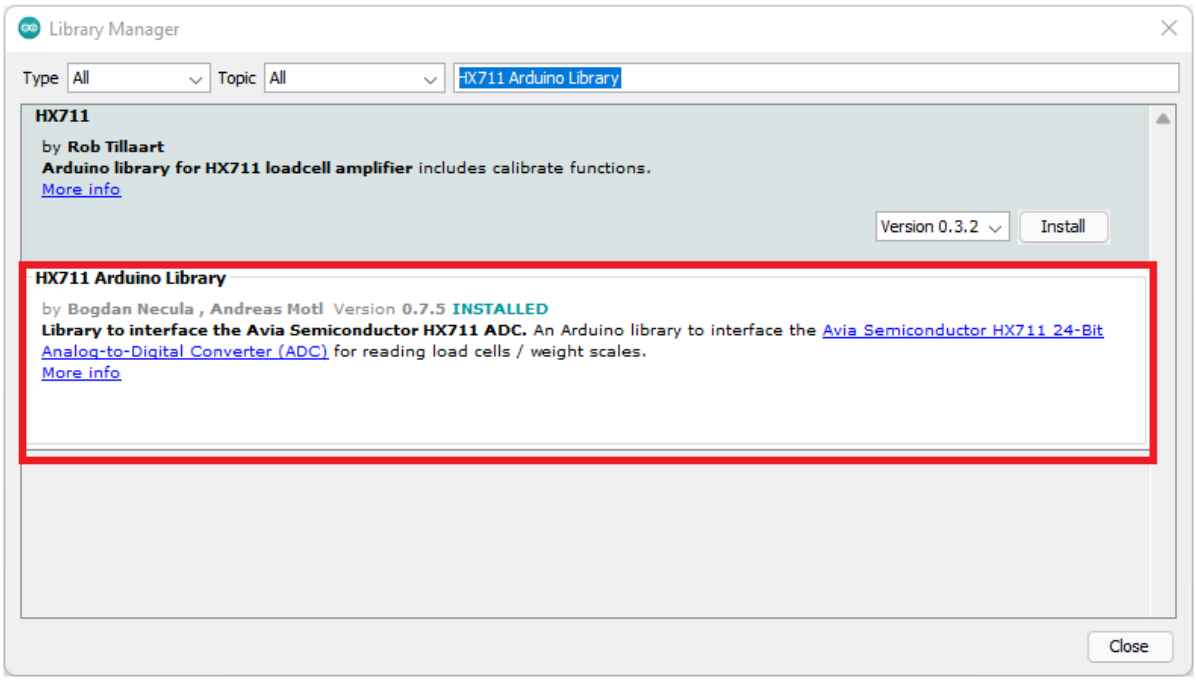
```
stepper.run();
```

2. CAPTEUR DE FORCE

Cellule de charge	HX711	HX711	ESP8266
Rouge (Mi+)	Mi+	Terre	Terre
Le noir (E-)	E-	DT	GPIO 12 (D6)
Blanc (UN-)	UN-	SCK	GPIO 13 (D7)
Vert (A+)	A+	VCC	3.3V



DRIVER

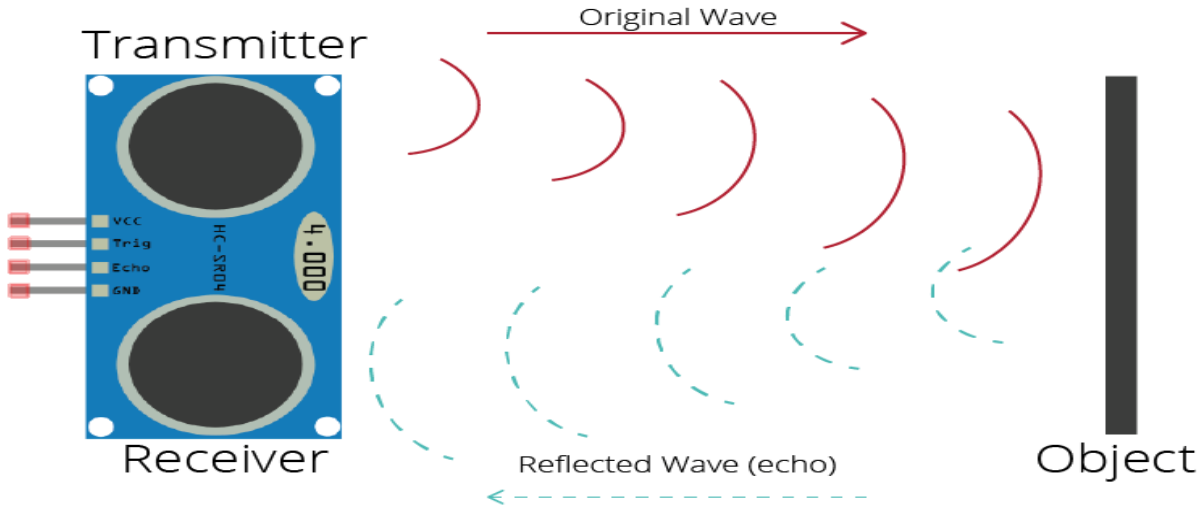


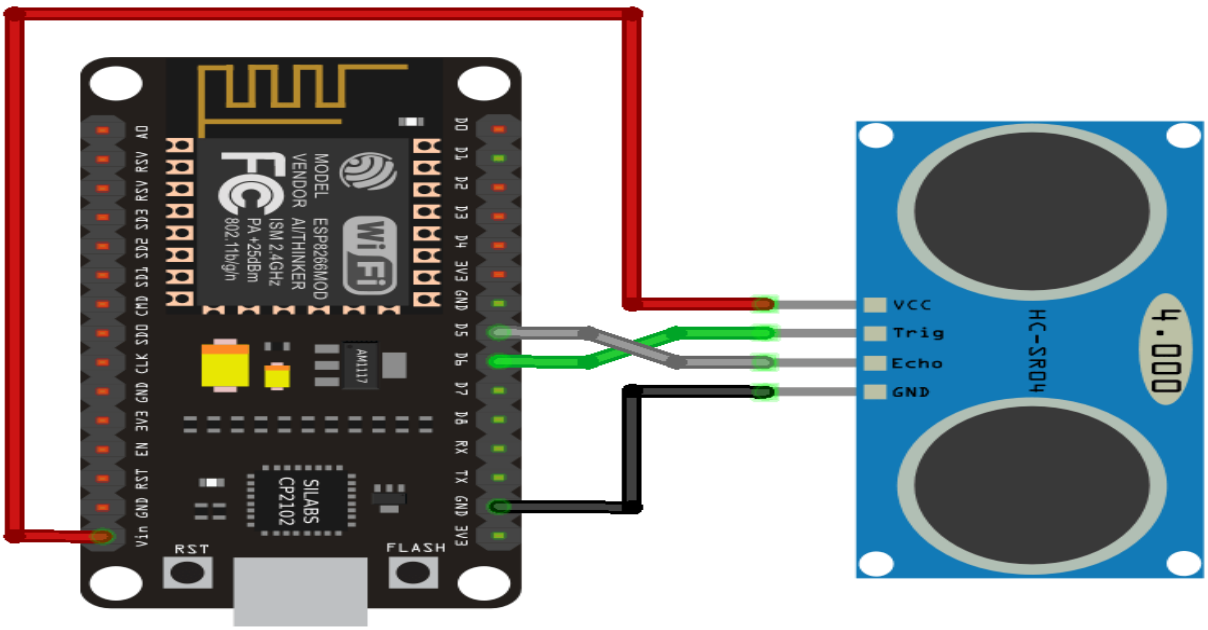
CODE SOURCE

```
/**
 * Complete project details at https://RandomNerdTutorials.com/esp8266-
load-cell-hx711/
 *
 * HX711 library for Arduino - example file
 * https://github.com/bogde/HX711
 *
 * MIT License
 * (c) 2018 Bogdan Necula
 *
**/

#include <Arduino.h>
#include "HX711.h"

// HX711 circuit wiring
const int LOADCELL_DOUT_PIN = 12;
const int LOADCELL_SCK_PIN = 13;

HX711 scale;

void setup() {
  Serial.begin(115200);
  Serial.println("HX711 Demo");
  Serial.println("Initializing the scale");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);

  Serial.println("Before setting up the scale:");
  Serial.print("read: \t\t");
  Serial.println(scale.read());      // print a raw reading from the ADC

  Serial.print("read average: \t\t");
  Serial.println(scale.read_average(20));   // print the average of 20
readings from the ADC

  Serial.print("get value: \t\t");
  Serial.println(scale.get_value(5));   // print the average of 5 readings
from the ADC minus the tare weight (not set yet)

  Serial.print("get units: \t\t");
  Serial.println(scale.get_units(5), 1);  // print the average of 5
readings from the ADC minus tare weight (not set) divided
            // by the SCALE parameter (not set yet)

  scale.set_scale(-478.507);
  //scale.set_scale(-471.497);                       // this value is
obtained by calibrating the scale with known weights; see the README for
details
  scale.tare();               // reset the scale to 0

  Serial.println("After setting up the scale:");

  Serial.print("read: \t\t");
  Serial.println(scale.read());                 // print a raw reading from
the ADC

  Serial.print("read average: \t\t");
```

```
  Serial.println(scale.read_average(20));       // print the average of 20
readings from the ADC

  Serial.print("get value: \t\t");
  Serial.println(scale.get_value(5));   // print the average of 5 readings
from the ADC minus the tare weight, set with tare()

  Serial.print("get units: \t\t");
  Serial.println(scale.get_units(5), 1);        // print the average of 5
readings from the ADC minus tare weight, divided
            // by the SCALE parameter set with set_scale

  Serial.println("Readings:");
}

void loop() {
  Serial.print("one reading:\t");
  Serial.print(scale.get_units(), 1);
  Serial.print("\t| average:\t");
  Serial.println(scale.get_units(10), 5);

  scale.power_down();             // put the ADC in sleep mode
  delay(5000);
  scale.power_up();
}
```

3. CAPTEUR ULTRA SON





CODE SOURCE

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.com/esp8266-
nodemcu-hc-sr04-ultrasonic-arduino/

  Permission is hereby granted, free of charge, to any person obtaining a
copy
  of this software and associated documentation files.

  The above copyright notice and this permission notice shall be included
in all
  copies or substantial portions of the Software.
*********/

const int trigPin = 12;
const int echoPin = 14;

//define sound velocity in cm/uS
#define SOUND_VELOCITY 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
float distanceInch;

void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}

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);

  // Calculate the distance
  distanceCm = duration * SOUND_VELOCITY/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;

  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);

  delay(1000);
}
```