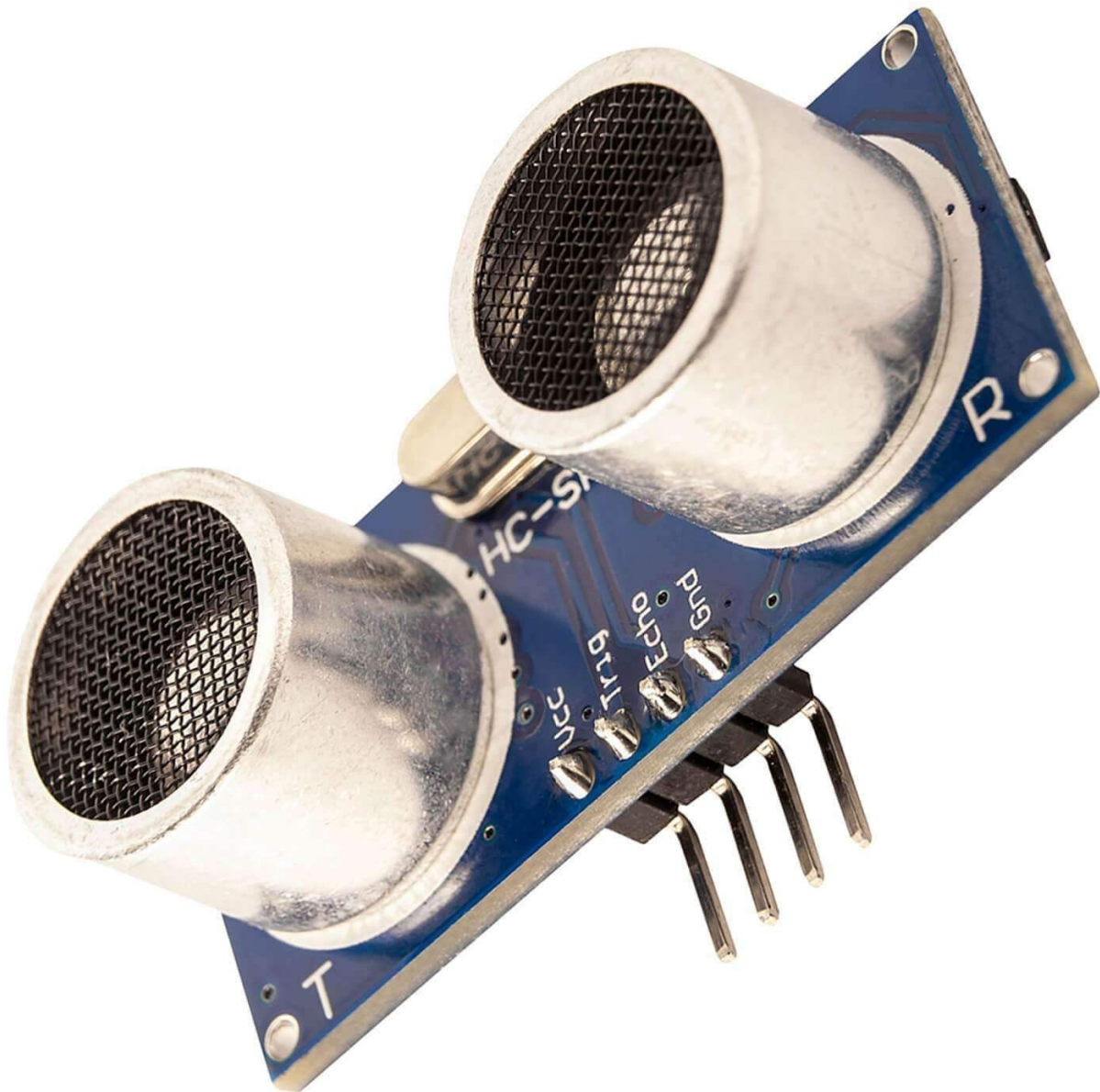


# AZ-Delivery

## Benvenuto!

Grazie per aver acquistato il nostro *Modulo Sensore a Ultrasuoni HC-SR04* AZ-Delivery. Nelle pagine seguenti, ti illustreremo come utilizzare e configurare questo pratico dispositivo.

**Buon divertimento!**





## Indice dei Contenuti

Introduzione.....	3
Calcolare la distanza.....	4
Specifiche.....	6
La piedinatura.....	7
Come configurare l'Arduino IDE.....	8
Come configurare il Raspberry Pi e il Python.....	12
Collegamento del modulo con Atmega328P.....	13
Esempio di sketch.....	14
Collegamento del modulo con Raspberry Pi.....	18
Librerie e strumenti per Python.....	20
Script Python.....	21



## **Introduzione**

Il modulo sensore ad ultrasuoni HC-SR04 è un dispositivo in grado di misurare la distanza utilizzando onde sonore ad ultrasuoni. Può determinare la distanza tra il modulo e gli altri oggetti nelle vicinanze con l'alta precisione.

Il sensore è comunemente usato in dispositivi di prevenzione degli ostacoli, vari dispositivi di misurazione della distanza, progetti di automazione, sistemi di parcheggio, allarmi di prossimità, ecc..

Il modulo è costituito da un trasmettitore ad ultrasuoni, un ricevitore, una parte elettronica di bordo come chip di amplificazione, un oscillatore a cristalli e altri componenti passivi come condensatori e resistenze.

Il principio di funzionamento del modulo è il seguente. Le onde sonore ad ultrasuoni sono trasmesse dal trasmettitore di bordo. Quando c'è un ostacolo davanti al modulo, le onde sonore ultrasoniche vengono riflesse da un ostacolo che torna al modulo. L'onda sonora ultrasonica riflessa viene poi captata da un ricevitore di bordo. Questo viene poi emesso sul pin ECHO come segnale digitale con una frequenza direttamente correlata al tempo necessario ad un'onda sonora ad ultrasuoni per viaggiare dal modulo all'ostacolo e tornare al ricevitore.

## Calcolare la distanza

Il modulo emette onde sonore ad ultrasuoni ad una frequenza superiore al campo uditivo umano (più di 20kHz). La velocità delle onde sonore che viaggiano nell'aria è di circa 343 m/s a temperatura ambiente (20°C). Questo valore di velocità dipende dalle circostanze ambientali come la temperatura, l'umidità e le variazioni di pressione dell'aria. La velocità del suono nell'aria in realtà dipende molto di più dalla temperatura e molto poco dall'umidità e dalla pressione dell'aria. Aumenta il suo valore di circa 0,6 m/s per grado Celsius. Nella maggior parte dei casi ad una temperatura di 20°C si può utilizzare il valore di 343 m/s. Per calcolare con precisione la velocità dell'onda sonora ultrasonica nell'aria, utilizzare la seguente formula:

$$V \text{ (m/s)} = 331.3 + (0.606 \times T)$$

dove V (m/s) è la velocità dell'onda sonora ultrasonica nell'aria e T (°C) è la temperatura dell'aria.

Quando la velocità dell'onda sonora ultrasonica viene moltiplicata per il tempo in cui l'onda sonora ultrasonica ha viaggiato dal modulo all'ostacolo e ritorno, il risultato è la distanza dal modulo all'ostacolo e ritorno:

$$\text{Distanza} = \text{Velocità} \times \text{Tempo}$$

# Az-Delivery

Poiché le onde sonore viaggiano 343 metri al secondo, per misurare distanze da 20mm fino a 4000mm (il campo del modulo) le misure vengono effettuate in microsecondi. Quindi, la velocità delle onde sonore ultrasoniche dovrebbe essere convertita dalla  $m/s$  in  $cm/\mu s$ , che è:

$$343m/s = 0.0343 \text{ cm}/\mu s = 1/29 \text{ cm}/\mu s$$

$$13503.9in/s = 0.0135 \text{ in}/\mu s = 1/74 \text{ in}/\mu s \text{ (utilizzato nello sketch)}$$

Poi, per ottenere la distanza effettiva tra il modulo e l'ostacolo, il risultato precedente dovrebbe essere diviso per due, perché l'onda sonora ultrasonica viaggia dal modulo all'ostacolo e ritorno, quindi i calcoli vengono così fatti:

$$\text{Distanza (cm)} = \text{Velocità del suono (cm}/\mu s) \times \text{Tempo } (\mu s) / 2$$

## Specifiche

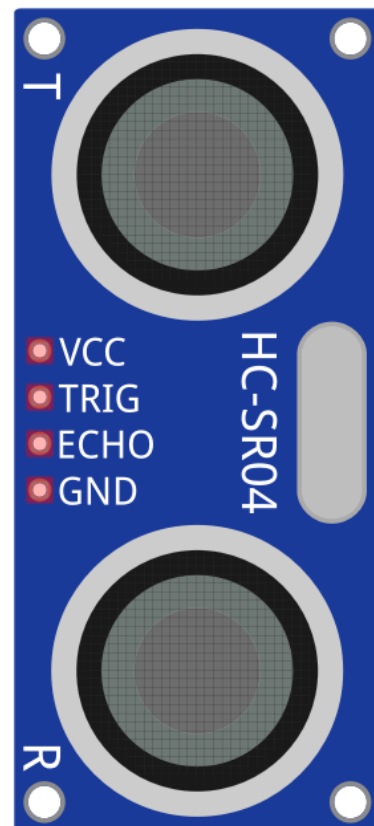
Tensione di alimentazione	fino a 5V
Tensione di esercizio:	da 3V a 5V
Tensione di uscita:	5V!
Consumo di corrente:	15mA
Corrente di quiescenza:	meno di 2mA
Frequenza ultrasonica:	40kHz
Segnale d'ingresso di attivazione:	10 $\mu$ S TTL pulsante
Campo di misura dell'angolo:	30 gradi
Campo di angolo effettivo:	meno di 15 gradi
Range di distanza operativa:	da 20mm a 4000mm (da 1in a 13 piedi)
Precisione dichiarata:	3mm, realisticamente 10mm
Dimensioni:	45 x 20 x 15mm (1.8 x 0.8 x 0.6in)

La distanza minima su cui sono valide le misure è di 20mm. Al di sotto di questa soglia, le letture possono diventare imprevedibili.

## La piedinatura

Il modulo HC-SR04 ha quattro pin. La piedinatura è mostrata nell'immagine seguente:

**POWER SUPPLY - VCC**  
**TRIGGER PULSE INPUT - TRIG**  
**ECHO PULSE OUTPUT - ECHO**  
**GROUND - GND**





**Nota:** la tensione di uscita del modulo è nel campo dei 5V. Per poter utilizzare il modulo con il Raspberry Pi, si deve utilizzare il dispositivo chiamato convertitore di livello logico. Altrimenti il collegamento dei segnali a 5V ai pin GPIO può causare danni al Raspberry Pi. A tal fine utilizzare il dispositivo denominato TXS0108E Convertitore di Livello Logico che AZ-Delivery offre.



## Come configurare l'Arduino IDE

Se l'Arduino IDE non è installato, seguire il [link](#) e scaricare il file di installazione del sistema operativo scelto.

### Download the Arduino IDE



**ARDUINO 1.8.9**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

**Windows** Installer, for Windows XP and up  
**Windows** ZIP file for non admin install

**Windows app** Requires Win 8.1 or 10  


**Mac OS X** 10.8 Mountain Lion or newer

**Linux** 32 bits  
**Linux** 64 bits  
**Linux** ARM 32 bits  
**Linux** ARM 64 bits

[Release Notes](#)  
[Source Code](#)  
[Checksums \(sha512\)](#)

Per gli utenti *Windows*, fare doppio clic sul file `.exe` scaricato e seguire le istruzioni nella finestra di installazione.

# Az-Delivery

Per gli utenti *Linux*, scaricare un file con estensione *.tar.xz*, che è necessario estrarre. Quando lo si estrae, andare nella directory estratta, e aprire il terminale in quella directory. È necessario eseguire due script *.sh*, il primo chiamato *arduino-linux-setup.sh* e il secondo chiamato *install.sh*.

Per eseguire il primo script nel terminale, eseguire il seguente comando:

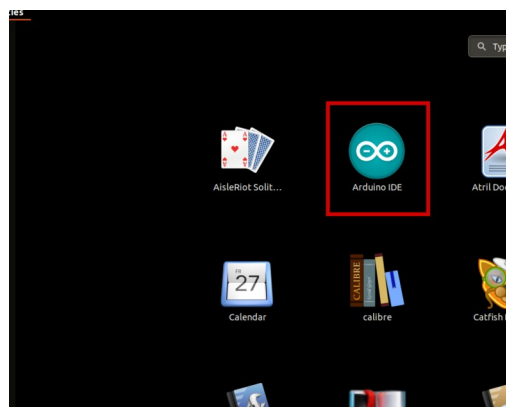
**sh arduino-linux-setup.sh user\_name**

**user\_name** - è il nome di un superutente nel sistema operativo Linux. Vi verrà richiesto di fornire la password per il superutente. Aspettate qualche minuto che lo script completi tutto.

Dopo l'installazione del primo script, si deve eseguire il secondo script chiamato *install.sh*. Nel terminale, eseguire il seguente comando:

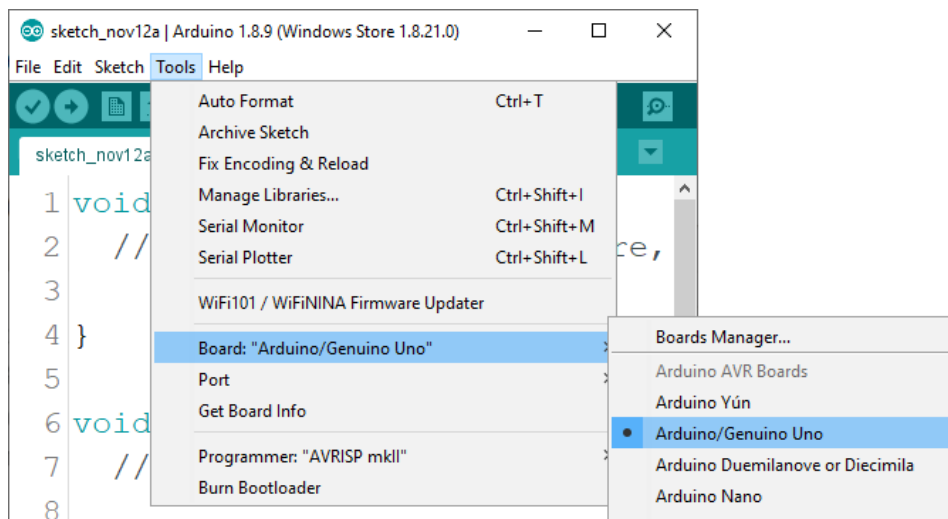
**sh install.sh**

Dopo l'installazione di questi script, andare su *Tutte le App*, dove troverai l'Arduino IDE installato.



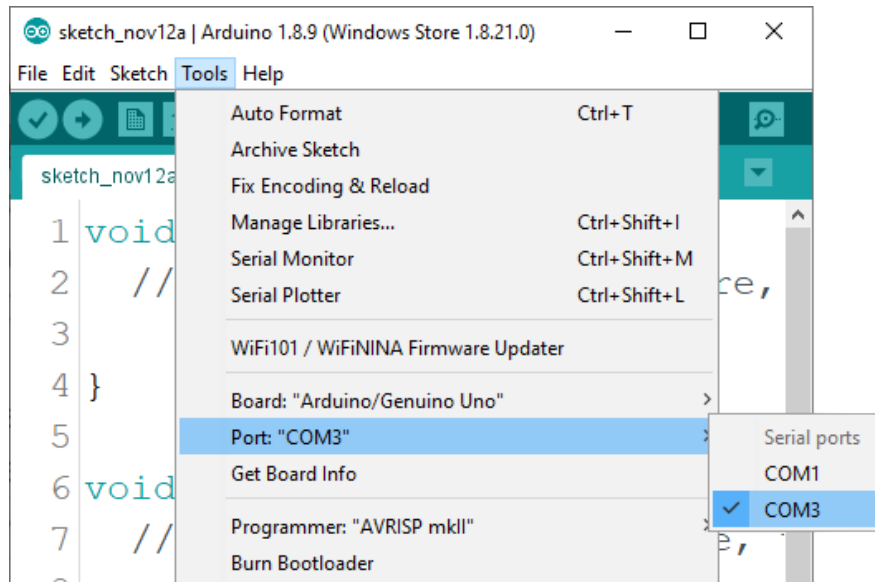
Quasi tutti i sistemi operativi sono dotati di un editor di testo preinstallato (ad esempio *Windows* viene fornito con *Notepad*, *Linux Ubuntu* viene fornito con *Gedit*, *Linux Raspbian* viene fornito con *Leafpad*, ecc..). Tutti questi editor di testo sono perfettamente adatti allo scopo dell'eBook.

La prossima cosa da fare è controllare se il PC è in grado di rilevare la scheda microcontrollore. Aprite l'Arduino IDE appena installato e andate su: *Strumenti > Scheda > {your board name here}*  
{your board name here} dovrebbe essere l'*Arduino/Genuino Uno*, come si può vedere nella seguente immagine:



È necessario selezionare la porta alla quale è collegata la scheda microcontrollore. Vai su: *Strumenti > Porta > {port name goes here}*  
e se avete collegato la scheda microcontrollore sulla porta usb dovrete vedere il nome della porta.

Se si utilizza l'Arduino IDE su Windows, i nomi delle porte sono i seguenti:



Per gli utenti *Linux*, il nome della porta è */dev/ttyUSBx* per esempio, dove x rappresenta un numero intero compreso tra 0 e 9, per esempio.



## **Come configurare il Raspberry Pi e il Python**

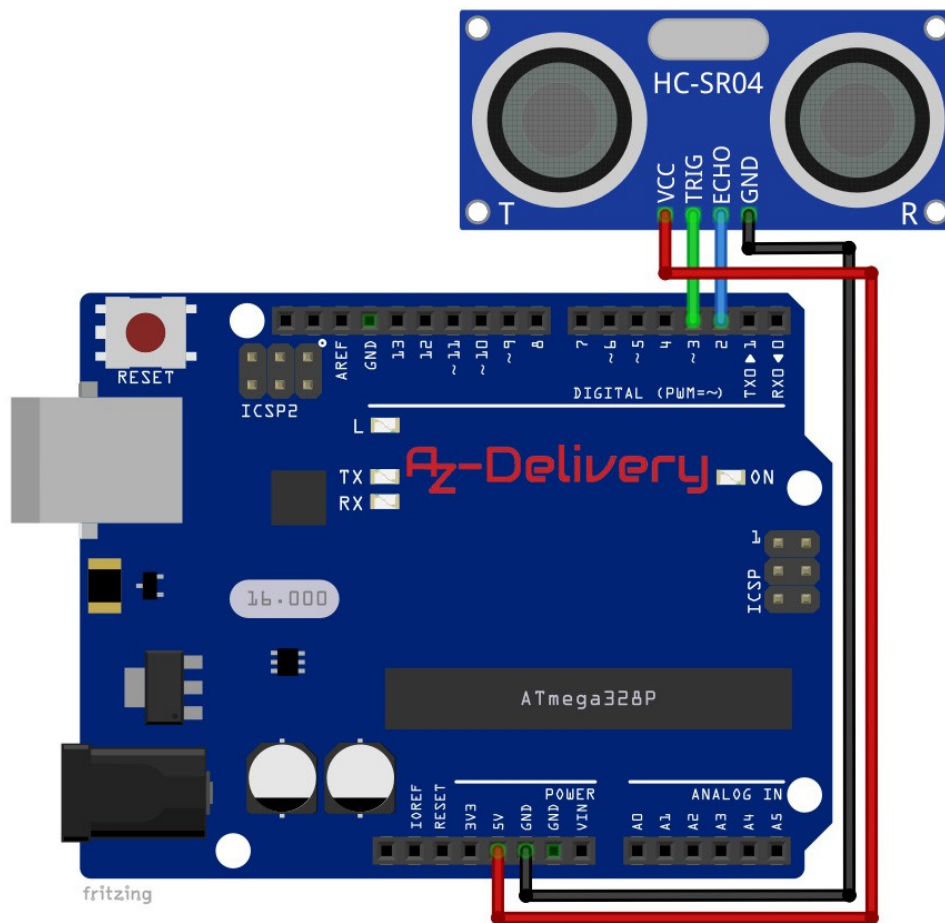
Per il Raspberry Pi, prima deve essere installato il sistema operativo, tutto deve essere impostato in modo da poter essere utilizzato in modalità Headless. La modalità Headless consente la connessione remota al Raspberry Pi, senza la necessità di uno schermo di un PC, mouse o tastiera. Le uniche cose di cui avete bisogno per questa modalità sono il Raspberry Pi, l'alimentazione e la connessione internet. Tutto questo è spiegato in dettaglio nell'eBook gratuito:

[Raspberry Pi Quick Startup Guide](#)

Il sistema operativo Raspbian viene fornito con il Python preinstallato.

## Collegamento del modulo con Atmega328P

Collegare il modulo con Atmega328P come indicato nel seguente schema di collegamento:



Pin modulo	Pin Mc	Colore filo
VCC	5V	Filo rosso
TRIG	D2	Filo verde
ECHO	D3	Filo blu
GND	GND	Filo nero

# Az-Delivery

## Esempio di sketch

```
#define TRIG_PIN 2
#define ECHO_PIN 3
long duration, cm, inches;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}
void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(5);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);

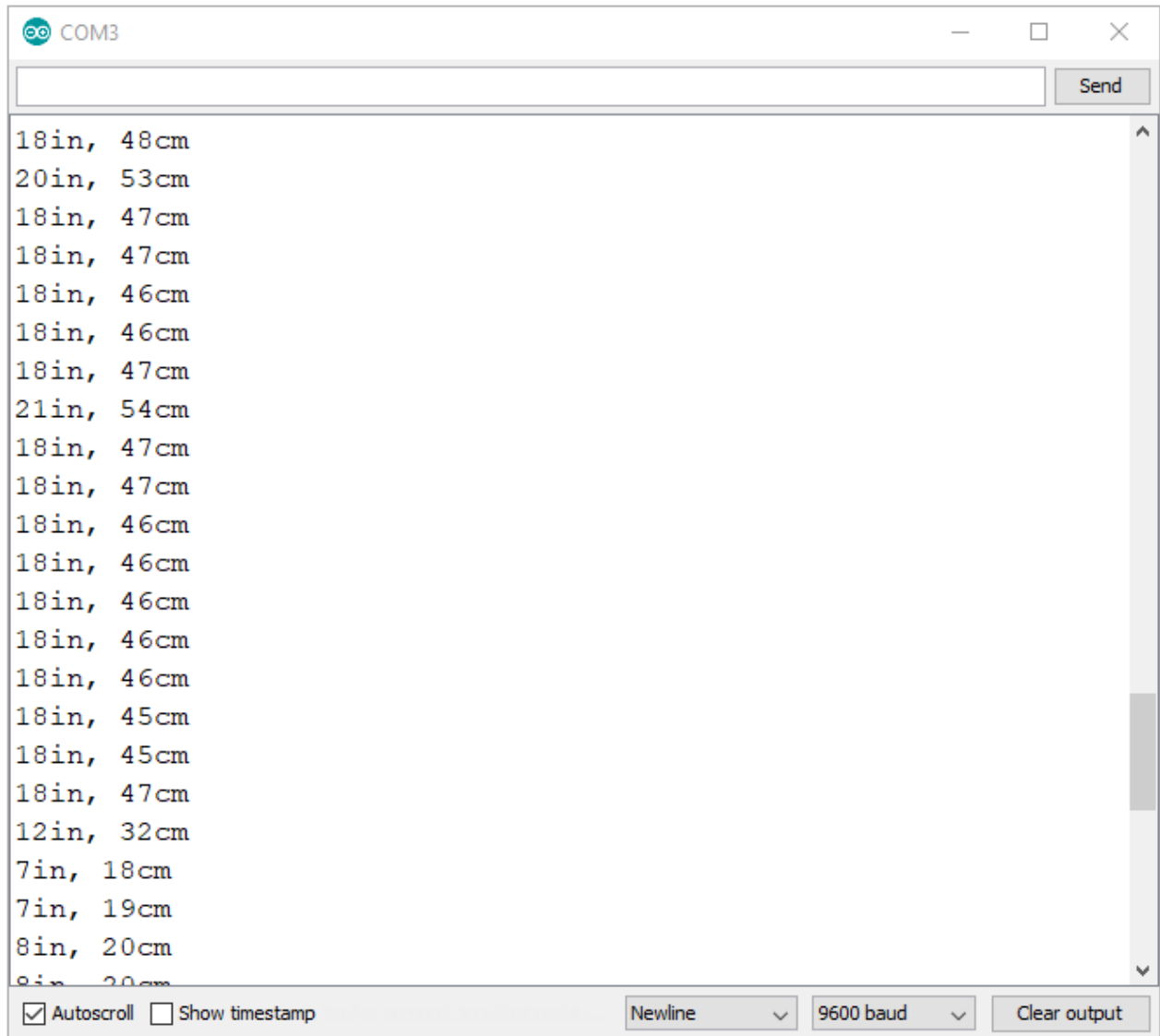
  cm = (duration / 2) / 29.1; // Divide by 29.1 or multiply by 0.0343
  inches = (duration / 2) / 74; // Divide by 74 or multiply by 0.0135

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

  delay(500);
}
```

# Az-Delivery

Caricare lo sketch e aprire il Monitor Seriale (*Strumenti* > *Monitor Seriale*). Il risultato dovrebbe assomigliare all'immagine seguente:





# Az-Delivery

Lo sketch inizia con la creazione di due macro *TRIG\_PIN* e *ECHO\_PIN*. Queste macro rappresentano i pin digitali di Atmega328P su cui sono collegati i pin del modulo.

Successivamente, vengono create tre variabili *long*: *durata*, *cm* e *pollici*. Nella variabile di *durata* viene salvato l'intervallo di tempo della misurazione. Nella variabile *cm* viene memorizzata la distanza calcolata in centimetri. Nella variabile *pollici* viene memorizzata la distanza calcolata in pollici.

Nella funzione *setup()* la comunicazione seriale viene avviata con un baud rate di 9600bps. Dopo di che vengono impostate le modalità per i pin digitali: *TRIG\_PIN* come *OUTPUT* e *ECHO\_PIN* come *INPUT*.

All'inizio della funzione *loop()*, *TRIG\_PIN* viene attivato nel seguente modo:

Prima di tutto, lo stato di *TRIG\_PIN* viene tenuto in stato *LOW* per 5 microsecondi, poi viene tenuto in stato *HIGH* per i 10 microsecondi e poi di nuovo nello stato *LOW*. In questo modo, l'onda sonora ultrasonica viene trasmessa nell'intervallo di tempo specifico, in modo che le misurazioni possano essere effettuate.

Successivamente, la funzione *pulseIn()* viene utilizzata per misurare la lunghezza dell'impulso su *ECHO\_PIN*. La funzione ha due argomenti e restituisce un valore lungo. Il primo argomento è il nome del pin su cui avviene la misurazione.

# Az-Delivery

Il secondo argomento della funzione *pulseIn()* può avere due valori: *HIGH* o *LOW*. Quando il valore del secondo argomento è *HIGH*, la funzione attende che il segnale su *ECHO\_PIN* cambi il suo stato da *LOW* a *HIGH* per avviare la misurazione. Se il valore del secondo argomento è *LOW*, allora la funzione attende che il segnale su *ECHO\_PIN* cambi il suo stato da *HIGH* a *LOW* per avviare la misurazione. Il valore di ritorno è il valore lungo senza segno, che rappresenta la lunghezza dell'impulso in microsecondi. la seguente riga di codice viene utilizzata per misurare la lunghezza dell'impulso su *ECHO\_PIN*:

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

Dove il valore di ritorno dalla funzione *pulseIn()* è memorizzato nella variabile di durata.

Successivamente viene effettuato il calcolo della distanza, con le seguenti righe di codice:

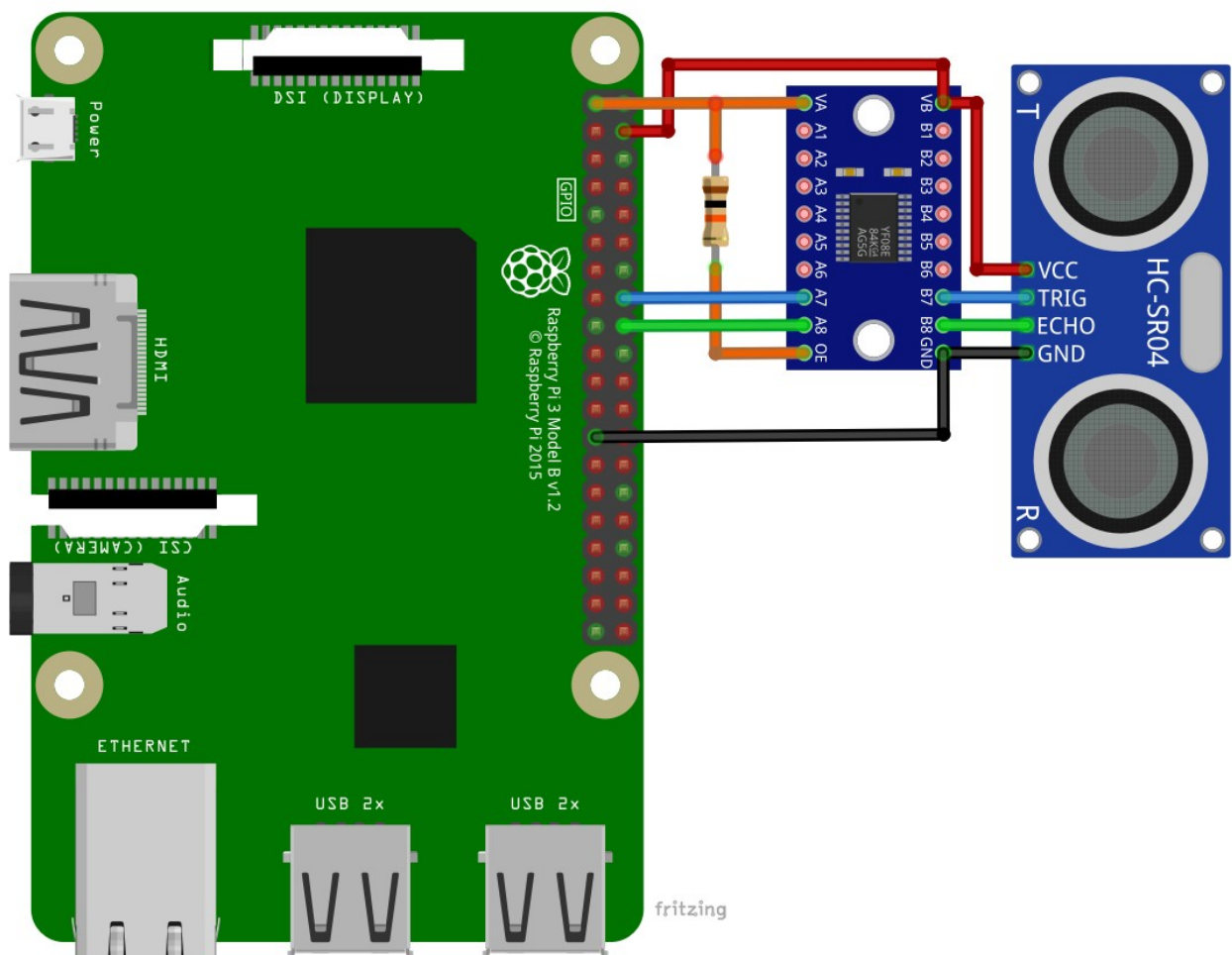
```
cm = (duration / 2) / 29.1;  
inches = (duration / 2) / 74;
```

Dopo di che, i dati vengono visualizzati nel Monitor Seriale.

Alla fine della funzione *loop()* viene impostata una pausa di ritardo di mezzo secondo. Questa pausa rappresenta la pausa tra due misure.

## Collegamento del modulo con Raspberry Pi

Collegare il modulo con il Raspberry Pi come indicato nel seguente schema di collegamento:



# Az-Delivery

Pin modulo	Pin Convertitore Logico (LLC)	Livello	Colore filo
VCC	VB		Filo rosso
TRIG	B7		Filo blu
ECHO	B8		Filo verde
GND	GND		Filo nero

Pin LLC	Pin Raspberry Pi	Pin fisico	Colore filo
VB	5V	4	Filo rosso
VA	3V3	1	Filo arancio
A7	GPIO23	16	Filo blu
A8	GPIO24	18	Filo verde
GND	GND	25	Filo nero
OE	3V3 via 10k $\Omega$ *	1	Filo arancio

\* Collegare il pin OE dell'LLC con 3V3 tramite una resistenza da 10k $\Omega$  (resistenza di pull-up) per abilitare il funzionamento dell'LLC.



## Librerie e strumenti per Python

Per utilizzare il modulo con il Raspberry Pi, è necessario installare la libreria *RPi.GPIO*. Se la libreria non è installata, aprire il terminale ed eseguire i seguenti comandi, uno per uno:

```
sudo apt-get update && sudo apt-get upgrade -y  
sudo apt-get install python3-rpi.gpio
```

L'esempio dello script originale di *Matt Hawkins* si trova al seguente [link](#).

## Script Python

```
import time
import RPi.GPIO as GPIO

def measure():
    speed_of_sound = 34300
    GPIO.output(TRIG_PIN, True)
    time.sleep(0.00001)
    GPIO.output(TRIG_PIN, False)
    start = time.time()
    while GPIO.input(ECHO_PIN) == 0:
        start = time.time()

    while GPIO.input(ECHO_PIN) == 1:
        stop = time.time()

    elapsed = stop - start
    distance = (elapsed * speed_of_sound) / 2
    return distance

def measure_average():
    distance1 = measure()
    time.sleep(0.1)
    distance2 = measure()
    time.sleep(0.1)
    distance3 = measure()
    distance = distance1 + distance2 + distance3
    distance = distance / 3
    return distance
```

# Az-Delivery

```
GPIO.setmode(GPIO.BCM)

TRIG_PIN = 23
ECHO_PIN = 24
GPIO.setup(TRIG_PIN, GPIO.OUT)
GPIO.setup(ECHO_PIN, GPIO.IN)

print('[Press Ctrl + C to end program!]\n')
try:
    GPIO.output(TRIG_PIN, False)
    time.sleep(0.5)

    while True:
        distance = measure_average()
        print('Distance: {:.1f}cm'.format(distance))
        time.sleep(1)

except KeyboardInterrupt:
    print('\nScript end!\n')

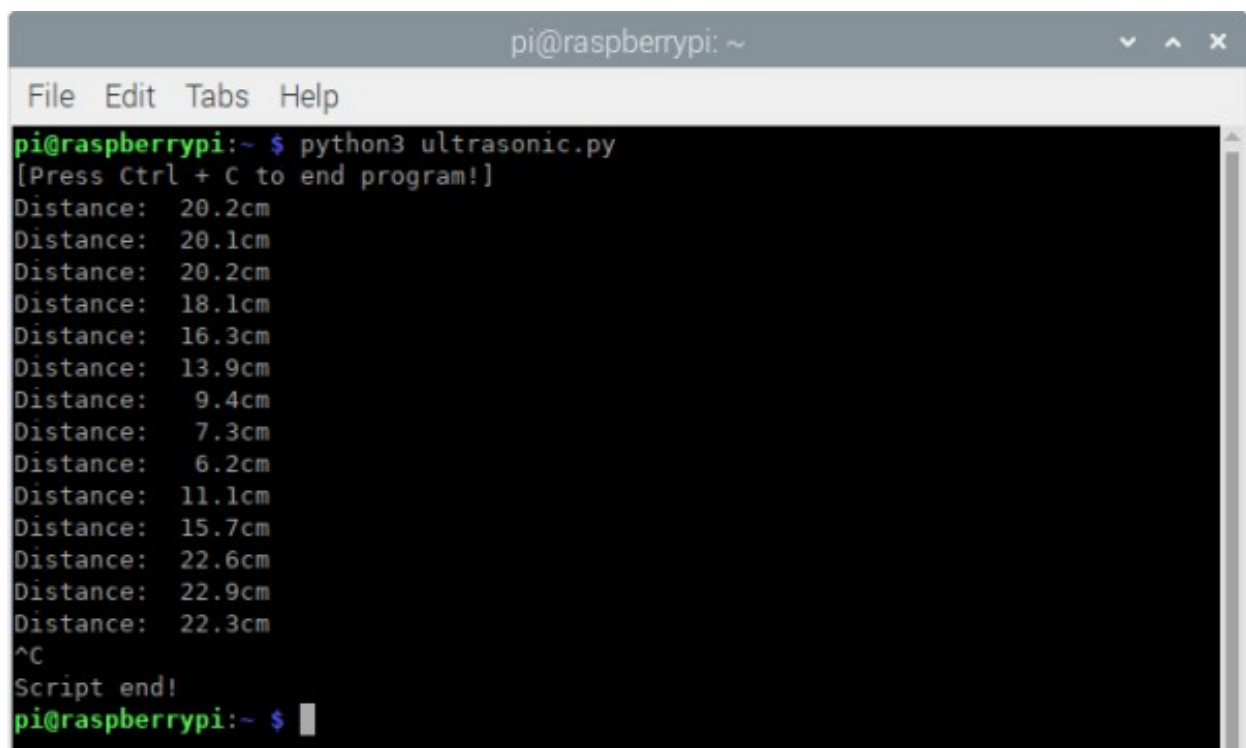
finally:
    GPIO.cleanup()
```

# Az-Delivery

Salvare lo script con il nome *ultrasonic.py*. Per eseguire lo script, aprire il terminale nella directory in cui è stato salvato lo script ed eseguire il seguente comando:

**python3 ultrasonic.py**

Il risultato dovrebbe assomigliare all'immagine seguente:



```
pi@raspberrypi:~ $ python3 ultrasonic.py
[Press Ctrl + C to end program!]
Distance: 20.2cm
Distance: 20.1cm
Distance: 20.2cm
Distance: 18.1cm
Distance: 16.3cm
Distance: 13.9cm
Distance: 9.4cm
Distance: 7.3cm
Distance: 6.2cm
Distance: 11.1cm
Distance: 15.7cm
Distance: 22.6cm
Distance: 22.9cm
Distance: 22.3cm
^C
Script end!
pi@raspberrypi:~ $
```

Per fermare lo script premere *CTRL + C* sulla tastiera.



# Az-Delivery

Lo script inizia con l'importazione di due librerie: *time* e *RPi.GPIO*.

Successivamente, vengono create due funzioni: *measure()* e *measure\_average()*. Entrambe le funzioni non hanno argomenti e restituiscono il valore *double*. La funzione *measure()* viene utilizzata per misurare e calcolare la distanza. L'algoritmo per misurare e calcolare la distanza è in questa funzione. La funzione *measure\_average()* permette di effettuare tre misurazioni e di ricavare un valore medio da tre misurazioni. Il valore di ritorno di queste funzioni rappresenta il valore di distanza calcolato.

Quindi, la denominazione dei pin GPIO è impostata su BCM, e le modalità dei pin GPIO per *TRIG\_PIN* e *ECHO\_PIN* sono impostate rispettivamente su uscita e ingresso.

Dopo di che viene creato il blocco di codice *try-except-finally*. Nel blocco di codice try lo stato di *TRIG\_PIN* viene impostato su *LOW*, dopo di che viene creato il blocco di codice a loop indefinito (*while True:*).

Nel blocco di codice a ciclo indefinito, prima viene eseguita la funzione *measure\_average()* e il valore di ritorno viene memorizzato nella variabile *distanza*. Successivamente, il valore della variabile di distanza viene visualizzato nel terminale, con la seguente riga di codice:  
`print('Distance: {:5.1f}cm'.format(distanza))`

Dove "*5.1f*": "*5*" significa che l'uscita contiene 5 cifre decimali; "*.1*"

# Az-Delivery

significa che c'è una cifra dopo il punto decimale; e "f" significa che il valore è di tipo flottante.

Per fermare lo script premere *CTRL + C* sulla tastiera. Questo viene chiamato interruzione da tastiera. Quando l'interrupt della tastiera viene eseguito il blocco di codice *except* e un messaggio *Script end!* Viene visualizzato nel terminale.

Il blocco di codice *finally* viene eseguito al termine dello script. Quando viene eseguito il blocco di codice *finally*, viene eseguita la funzione chiamata *cleanup()*. La funzione *cleanup()* disabilita tutte le interfacce GPIO usate e le modalità dei pin GPIO.



E ora è tempo di imparare e di creare dei Progetti da solo. Lo puoi fare con l'aiuto di molti script di esempio e altri tutorial, che puoi trovare in internet.

**Se stai cercando dei microelettronica e accessori di alta qualità, AZ-Delivery Vertriebs GmbH è l'azienda giusta dove potrai trovarli. Ti forniremo numerosi esempi di applicazioni, guide di installazione complete, e-book, librerie e l'assistenza dei nostri esperti tecnici.**

<https://az-delivery.de>

Buon divertimento!

Impressum

<https://az-delivery.de/pages/about-us>