SENSORE AD ULTRASUONI HC-SR04 E ARDUINO

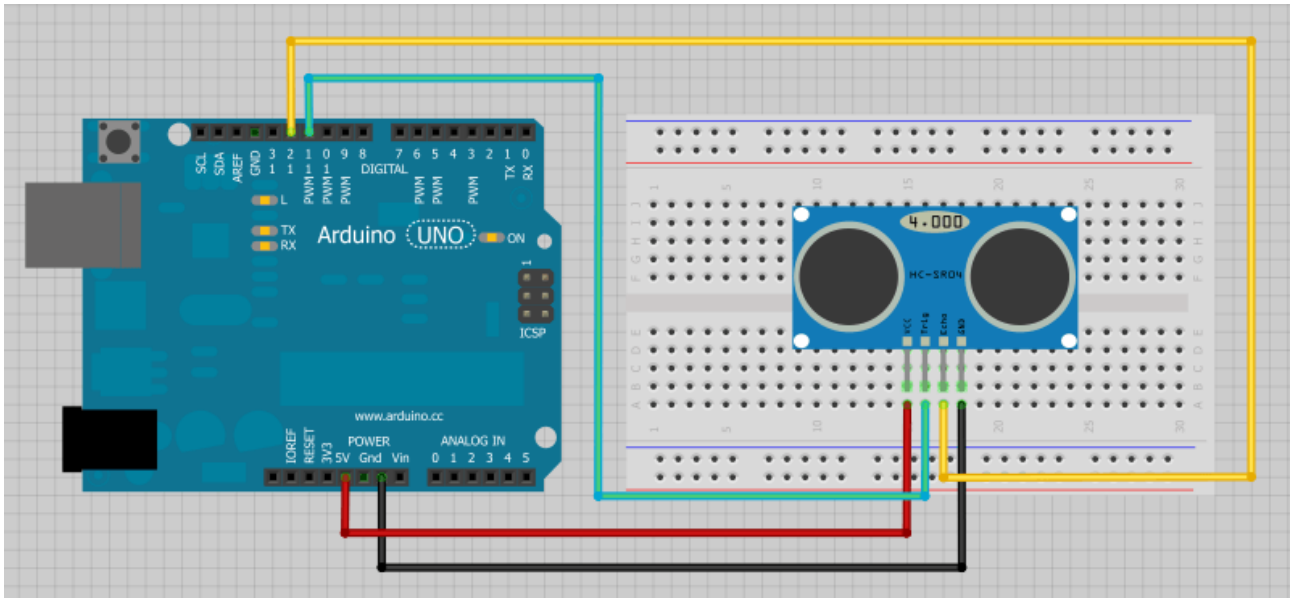
Giorgio De Nunzio – Giovanni Marsella

http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/ (con piccole modifiche)

http://www.giuseppecaccavale.it/arduino/sensore-ad-ultrasuoni-hc-sr04-arduino/ (con piccole modifiche)

http://www.maffucci.it/2014/11/15/edurobot-uno-come-costruire-il-vostro-primo-arduino-robot-lezione-2/

https://www.mpja.com/download/hc-sr04_ultrasonic_module_user_guidejohn.pdf



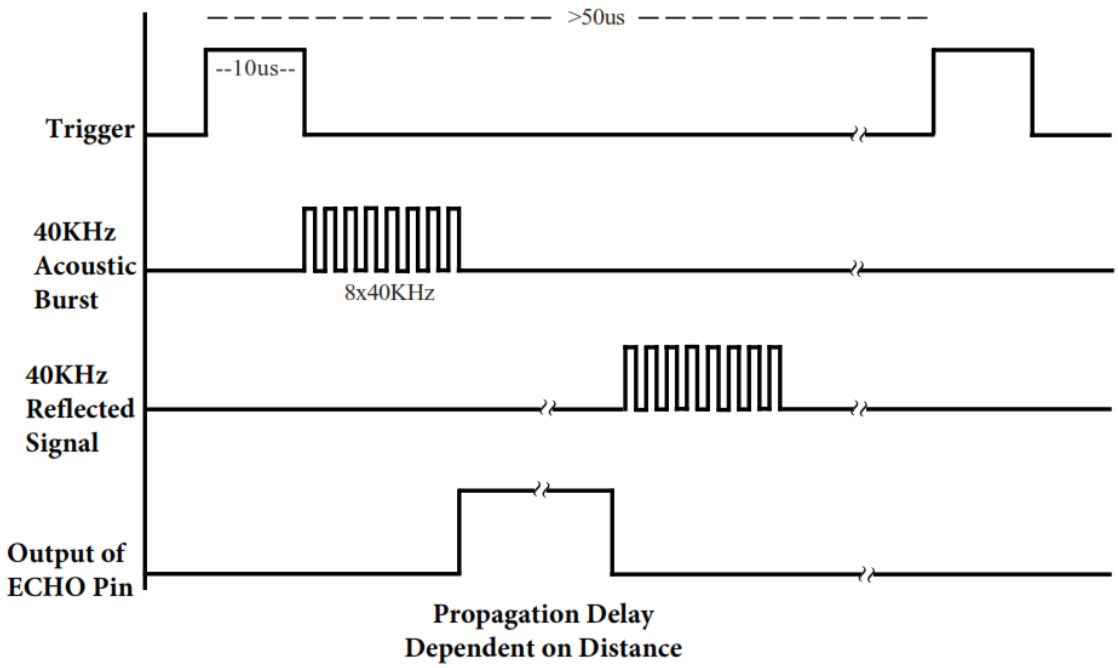
Misurare la distanza con un sensore ad ultrasuoni HC-SR04 può essere utile in molte occasioni, ad esempio in un sistema anti-intrusione che deve far suonare un allarme o un sistema robotizzato che deve evitare ostacoli. In questo tutorial viene utilizzato il sensore ad ultrasuoni HC-SR04, dispositivo economico e con un buon *range* operativo.

I sensori ad ultrasuoni non forniscono direttamente la misura della distanza dell'oggetto più vicino, ma misurano il tempo impiegato da un segnale sonoro a raggiungere l'oggetto e ritornare al sensore ("tempo di volo").

L'impulso ad ultrasuoni inviato dal sensore HC-SR04 ha la frequenza di circa 40KHz. Il tempo di volo è misurato in microsecondi. La tensione di funzionamento è 5V, quindi potremo alimentarlo direttamente utilizzando Arduino.

Il sensore HC-SR04 dispone di 4 pin: Vcc (+5V), Trigger, Echo, GND. Per attivare il sensore, si invia un impulso alto sul pin Trigger per almeno 10 microsecondi: a questo punto il trasmettitore T invierà il ping sonoro e aspetterà il ritorno delle onde riflesse; il ricevitore R risponderà sul pin Echo con un impulso alto della durata corrispondente a quella di viaggio delle onde sonore.

Se tale durata supera 38 ms (quindi 38000 μs), si considera che non sia stato incontrato alcun ostacolo (misure di tempo maggiori di 38ms non sono affidabili). Per sicurezza, prima di interrogare nuovamente il sensore, si aspettano in genere 50-60 ms per far sì che non vi siano interferenze con la misura successiva.



Il corretto pilotaggio del sensore prevede l'inserimento del comando `digitalWrite(triggerPort, LOW);` per una durata di 5 μs prima dell'impulso di trigger.

Per convertire l'intervallo di tempo misurato in una lunghezza, bisogna ricordare che la velocità del suono è di 331,5 m/s a 0 °C e di 343,4 m/s a 20 °C ed in generale varia secondo la relazione v = 331,4 + 0,62 T dove la temperatura T è misurata in °C.

Per la realizzazione del misuratore di distanza a ultrasuoni assumiamo di lavorare ad una temperatura ambiente di 20 °C e quindi la velocità del suono sarà di circa

v = 343 m/s = 343 * 10^2 cm / 10^6μs = 343 * 10^{-4} cm/μs = 0.0343 cm/μs.

Ora, ricordando che v = s/t (v: velocità, s: spazio, t: tempo), lo spazio percorso sarà: s = v*t da cui

s = 0.0343 * t

se il tempo è espresso in μs. Per calcolare lo spazio percorso, bisogna tener conto che il suono percorre due volte la distanza da misurare (giunge sull'oggetto e ritorna indietro al sensore) quindi il valore di t misurato dev'essere diviso per 2. La formula corretta per la misura dello spazio percorso è quindi:

s = 0.0343 * t/2 (formula riportata nel codice in basso)

ossia:

s = 0.0171 * t

oppure:

s = t/58.31

e, arrotondando,

s = t/58

formula più semplice da ricordare (con t espresso in μs).

```
/*
 * created by Rui Santos, http://randomnerdtutorials.com
 *
 * Complete Guide for Ultrasonic Sensor HC-SR04
 *
    Ultrasonic sensor Pins:
        VCC: +5VDC
        Trig : Trigger (INPUT) - Pin11
        Echo: Echo (OUTPUT) - Pin 12
        GND: GND
 */

int trigPin = 11;    //Trig - green Jumper
int echoPin = 12;    //Echo - yellow Jumper
long duration;       // duration is in microsec
float distance;      // distance is in cm

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.
  duration = pulseIn(echoPin, HIGH);

  // convert the time into a distance
  distance = (duration/2.0 * 0.0343); // or, which is the same, /29.15

  Serial.print(distance);
  Serial.print("cm");
  Serial.println();

  delay(250);
}
```

Correzione delle incertezze di funzionamento dei sensori

I sensori non sono completamente affidabili e alle volte restituiscono valori erronei o vanno in timeout, in tal caso restituendo 0, anche se c'è un ostacolo vicino. Il sito:

https://therandomlab.blogspot.it/2015/05/repair-and-solve-faulty-hc-sr04.html?showComment=1476201584407#c443883765476979154

ora non più raggiungibile, riportava un modo solo software per ridurre il numero di false letture.
Ecco il codice, in forma di function; inserirlo nello sketch, richiamando la funzione dalla loop(); compilare e inviare ad Arduino. Verificare se il numero di errori compiuti dal sensore diminuisce.

```
float getDistance(int trigPin, int echoPin) // returns the distance (cm)
{
  long duration;
  float distance;

  // 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); // We send a 10us pulse
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH, 20000); // We wait for the echo to come back, with a
                                            // timeout of 20ms, which corresponds
                                            // approximately to 3m

  // pulseIn will only return 0 if it timed out.
  // (or if echoPin was already to 1, but it should not happen)

  if(duration == 0) // If we timed out
  {
    pinMode(echoPin, OUTPUT); // Then we set echo pin to output mode
    digitalWrite(echoPin, LOW); // We send a LOW pulse to the echo pin
    delayMicroseconds(200);
    pinMode(echoPin, INPUT); // And finally we come back to input mode
  }

  distance = (duration/2.0) / 29.1; //  /2 because the pulse is going and coming back

  return distance; //We return the result. Here you can find a 0 if we timed out
}
```

(un altro metodo è descritto qui: https://www.instructables.com/8-Tutorials-to-Solve-Problems-and-Improve-the-Perf/ al primo link e prevede l'inserimento di un resistore nel circuito)

Media di più letture

Scrivere due funzioni, chiamate `meanDistance()` e `pingSensor()`. La funzione `pingSensor()` contenga la sequenza di istruzioni, esaminata nel codice descritto precedentemente, necessaria per leggere la distanza di un ostacolo, e restituisca questo valore. La definizione potrebbe essere così:

```
float pingSensor() {
    float distance;
// inserire qui le istruzioni per attivare il sensore
// e leggere la distanza misurata, da porre in "distance"
    return distance;  // restituisce il valore misurato
}
```

La funzione `meanDistance()`, invece, chiami cinque volte la funzione `pingSensor()`, a distanza di 50ms una dall'altra, calcoli la media dei cinque valori, e la restituisca.

```
float meanDistance() {
    float mD, d1, d2, d3, d4, d5;  // conterranno la media delle letture, e le letture stesse
    d1 = pingSensor();
    delay(50);
    d2 = pingSensor();
    delay(50);
    d3 = pingSensor();
    delay(50);
    d4 = pingSensor();
    delay(50);
    d5 = pingSensor();
    delay(50);
    mD = (d1 + d2 + d3 + d4 + d5)/5;
    return mD;
}
```

Nella funzione `loop()` si chiami `meanDistance()` scrivendo, tramite il monitor seriale, il valore della distanza così letta. Il risultato dovrebbe essere più stabile rispetto a quello letto tramite il programma visto all'inizio di quest'esercizio.

Media di più letture tramite ciclo for

La funzione `meanDistance` può essere scritta più elegantemente con un ciclo for e con l'uso di un array per contenere le letture:

```
float meanDistance() {
    float mD, d[5];
    int j;
    for (j = 0; j < 5; j++) {
        d[j] = pingSensor();
        delay(50);
    }
    mD = (d[0] + d[1] + d[2] + d[3] + d[4])/5;
    return mD;
}
```

Altra variante: accumuliamo (sommiamo) i valori nel ciclo for:

```
float meanDistance() {
```

```
    float mD = 0;
    int j;
    for (j = 0; j < 5; j++) {
        mD = mD + pingSensor();
        delay(50);
    }
    mD = mD/5;
    return mD;
}
```

Si noti che in C è possibile scrivere l'istruzione

```
mD = mD + pingSensor();
```

così:

```
mD += pingSensor();
```

Valutare la linearità del sistema

Usare un foglio di carta millimetrata per mettere in grafico la distanza "vera" di un oggetto (misurata con una riga millimetrata) e la distanza misurata con HC-SR04.

Uso del sensore con un LED

Inserire un led rosso nel circuito, che si illumini quando vi è un ostacolo a meno di 10 cm. Per semplicità si può usare il led on board.

Uso del sensore con un LED, il ritorno

Inserire un led rosso nel circuito, che lampeggi quando vi è un ostacolo a meno di 30 cm, con frequenza di blink che aumenta via via che la distanza dell'ostacolo diminuisce. Per semplicità si può usare il led on board.

Uso del sensore con uno schermo LCD:

```
//HC RS04 Sensore ultrasuoni
//Giuseppe Caccavale
#include <LiquidCrystal.h>
const int triggerPort = 9;
const int echoPort = 10;
const int led = 13;

LiquidCrystal lcd(2, 3, 4, 5, 6, 7);

void setup() {

pinMode(triggerPort, OUTPUT);
pinMode(echoPort, INPUT);
pinMode(led, OUTPUT);
lcd.begin(16, 2);
lcd.setCursor(0, 0);
lcd.print( "Ultrasuoni ");
}

void loop() {

//porta bassa l'uscita del trigger
digitalWrite(triggerPort, LOW);
```

```
//invia un impulso di 10microsec su trigger
digitalWrite(triggerPort, HIGH);
delayMicroseconds( 10 );
digitalWrite(triggerPort, LOW);

long durata = pulseIn( echoPort, HIGH );

float distanza = 0.034 * durata / 2.0;

lcd.setCursor(0, 1);
lcd.print("dist.: ");

//dopo 38ms è fuori dalla portata del sensore
if( durata > 38000 ){
lcd.setCursor(0, 1);
lcd.println("Fuori portata   ");
}
else{
lcd.print(distanza);
lcd.println(" cm     ");
}

if(distanza < 10){
 digitalWrite(led, HIGH);
}
else{
 digitalWrite(led, LOW);
}

//Aspetta 1000 millisecondi
delay(1000);
}
```

Emulare i sensori di parcheggio 1

Costruire un circuito provvisto di un sensore HC-SR04, un led (con relativo resistore di limitazione della corrente), il cicalino. Il cicalino suoni quando un ostacolo è lontano meno di 30 cm dal sensore, e il suono sia tanto più acuto quanto più vicino è l'ostacolo.

Emulare i sensori di parcheggio 2

Variante: dopo aver scritto una funzione che faccia emettere un suono di frequenza e durata fissate, seguito da un periodo di silenzio variabile (passato come parametro alla funzione):

```
void beep(int silence) {
    int pin = ...;
    unsigned int frequency = 1000;
    unsigned long duration = 200;
    tone(pin, frequency, duration);  // emetti il suono e poi...
    delay(silence);                  // aspetta un po'
}
```

il codice dovrà fare suonare il cicalino se un oggetto è a meno di 30 cm dal sensore, e il ritmo del suono dev'essere via via più frequente quando l'oggetto è più vicino.

Emulare i sensori di parcheggio 3

Costruire un circuito provvisto di due sensori HC-SR04, due led di colore diverso (con relativi resistori di limitazione della corrente), il cicalino. Il cicalino suoni quando un ostacolo è lontano meno di 30 cm da uno dei due sensori; si accenda il led corrispondente al sensore al quale l'ostacolo è maggiormente vicino.