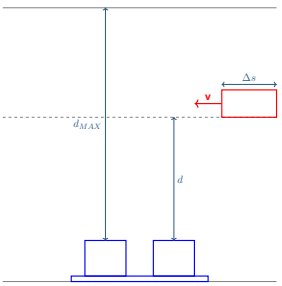
Chapter 1
A speedoMeter with Arduino

This document explains how to build a device to measure the speed of an object whose transversal size is known, using Arduino equipped with an ultrasonic sensor HCSR04.

1. Principle of operation



An HCSR04 emits ultrasonic signals and measures the time needed to sound waves to be reflected by an obstacle placed in front of it. Let d_{MAX} the distance between the sensor and a fixed screen in front of it: it could be a wall, a piece of cardboard, etc. The time measured by the sensor, in this case, is

$$T = 2\frac{d_{MAX}}{c}\,, \tag{1.1}$$

$c \simeq 340$ m/s being the speed of sound. If a second obstacle stands between the sensor and the screen, at distance $d < d_{MAX}$, the time required for the ultrasonic signal to be detected by the HCSR04 is

$$\tau = 2\frac{d}{c}\,. \tag{1.2}$$

If the obstacle moves with speed v in a direction perpendicular to the line joining the sensor with the screen, Arduino initially measures T, then measures τ for some time Δt, then measures T again. Here

$$\Delta t = \frac{\Delta s}{v}\,, \tag{1.3}$$

where Δs is the width of the obstacle. Measuring Δt and Δs allows the measurement of v.

2. Measuring the duration

Arduino can be programmed to continuously measure the time t needed to the ultrasonic signal to turn back once reflected by an obstacle. We repeat this measurement as long as $t > \alpha T$, where $\alpha < 1$ is chosen such that it is large enough to be easily estimated by eye, but small enough not to run into measurement fluctuations that might not verify the above condition. For example, for $d_{MAX} \simeq 1$ m, a value of $\alpha = 0.7$ is adequate.

As soon as $t > \alpha T$ is no longer true, we get the current time t_1 in microseconds, using the `micros()` function, and restart monitoring the distance, as long as $t \leqslant \alpha T$. While this condition is met, the obstacle, whose transversal width is Δs, is still in front of the sensor. Measuring the current time t_2 when the latter condition is no longer met, we get

$$\Delta t = t_2 - t_1 \,. \tag{1.4}$$

From this, we compute the speed, assumed to be constant, as

$$v = \frac{\Delta s}{\Delta t} \,. \tag{1.5}$$

This is an average velocity, where the average is taken within an interval Δt. Care must be taken to make Δt as small as possible, keeping Δs small, yet large enough to be spotted by Arduino.

The activity has been tested using a Pasco™ plastic cart ($\Delta s \simeq 16.5$ cm) running along an incline with $\theta \simeq 17.5^\circ$.

3. Possible experiments

All the experiments in which it is necessary to measure a speed can be done with this equipment. Free fall or sliding along an incline are two examples.

In experiments in which the motion of the part is due to gravity, $v = v(0) + \sqrt{2gh}$, thus, measuring the initial height h of the part, it is possible to use the data to obtain g.

Of course, it is even possible to invert the working principle: knowing the speed of an object, it is possible to measure its transversal size.

4. Arduino code

```
/*
  speedoMeter
  Copyright (C) 2023 Giovanni Organtini giovanni.organtini@uniroma1.it

  This program is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  This program is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with this program.  If not, see <https://www.gnu.org/licenses/>.
*/

#define TRIG 7    // the pin connected to the TRIG lead of the HCSR04
#define ECHO 8    // the pin connected to the ECHO lead of the HCSR04
#define DMAX 1.   // the maximum distance between the sensor and the moving part
#define S 0.165   // the length of the moving part (in m): adjust it to your needs

unsigned long threshold;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  digitalWrite(TRIG, LOW);
  // compute the time threshold for the HCSR04, as 70% of the
  // time needed to sound to cover the distance the
  // maximum possible distance
  threshold = floor(2*0.7*DMAX/340.*1e6);
  Serial.print("Threshold = ");
  Serial.print(threshold);
  Serial.print(" us, corresponding to ");
  Serial.print(threshold * 1.e-6 * 33000. / 2);
  Serial.println(" cm");
}

void loop() {
  unsigned long t;
  do {
    digitalWrite(TRIG, HIGH);  // trigger the sensor
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    t = pulseIn(ECHO, HIGH);   // measure the distance of the obstacle
```

```
  } while (t > threshold);      // wait until it goes below the threshold
  unsigned long t1 = micros(); // start time
  do {
    digitalWrite(TRIG, HIGH);  //repeat the distance measurement
    delayMicroseconds(10);
    digitalWrite(TRIG, LOW);
    t = pulseIn(ECHO, HIGH);
  } while (t <= threshold);
  unsigned long t2 = micros(); // stop time
  Serial.print("t = ");
  Serial.print(t2-t1);
  Serial.print(" us --> v = ");
  Serial.print(S/((t2 - t1)*1.e-6));
  Serial.println(" m/s");
  delay(1000);
}
// end of code
```